How-to Companion Guide

Lucidworks | Make Search a Superpower™

Table of Contents

How-to 11
- Access the Spark History Server 14
- Accessing HTTPS web resources with signed Enterprise/Self-signed CA Certificates .. 15
- Add a Detail Page 26
- Add a Query Pipeline Stage 29
- Add a Sidebar with Facets 31
- Add and remove users from applications 32
- Add Custom Headers to HTTP Requests 33
- Add Smart Answers Pipelines to an Existing App 35
- Add Tesseract Optical Character Recognition to Fusion Connectors .. 37
- Adding Twitter and YouTube Results 40
- Advanced Search 46
- AEM connector OAuth authorization 49
- App Studio Development & Deployment Guide 61
- App Studio Development Troubleshooting 64
- Assign Connectors Classic Jobs to Specific Nodes 67
- Automated Solr Backups in Fusion 5 71
- Automatically classify new documents at index time 75
- Automatically Classify New Documents at Index Time 79
- Automatically classify new queries 83
- Automatically Classify New Queries 88
- Avoid SharePoint Throttling 93
- Block a document in Experience Optimizer 96
- Block a Document in Experience Optimizer 97
- Block a product in Predictive Merchandiser 98
- Block a Product in Predictive Merchandiser 99
- Boost a document in Experience Optimizer 100
- Boost a product in Predictive Merchandiser 102
- Boost a Product in Predictive Merchandiser 103
- Boost or Block with the Query Workbench 104
- Build and Deploy A Query Stage Plugin 112
- Built-in SQL Aggregation Jobs Using Cloud Storage Buckets .. 115
- Bury a Product in Experience Optimizer 120
- Bury a Product in Predictive Merchandiser 122
- Change a Fusion User's Password 124
- Change a user password 125
- Changing the Color Scheme 126
- Changing the Logo 127
- Check the System State 128
- Clean Up Spark Driver Pods 132
- Cold Start Part 1 133

Compare Two Query Pipelines 139
Configure a Box.com Datasource 141
Configure a Data Model 154
Configure A Lucidworks AI Gateway Integration 164
Configure a new datasource 167
Configure a New Datasource 168
Configure a non-administrative crawl account in SharePoint Online .. 169
Configure a Non-Administrative Crawl Account in SharePoint Online .. 173
Configure a REST Call Job to Delete Old Signals 176
Configure a SharePoint Online V1 Optimized Datasource .. 180
Configure a SharePoint V1 Optimized Datasource 187
Configure a SharePoint V2 Datasource 197
Configure Active Directory Federation Services (AD FS) authentication for Fusion .. 208
Configure Activity Tracking 214
Configure AEM V2 Connector 215
Configure An Argo-Based Job to Access GCS 225
Configure An Argo-Based Job to Access S3 227
Configure Autocomplete 230
Configure Box.com tokens 234
Configure Box.com Tokens 237
Configure Dropbox Authentication 240
Configure field mapping for Couchbase 242
Configure Field Mapping for Couchbase 245
Configure Fusion 5.x for LDAP 248
Configure Fusion for Kerberos in Unix 257
Configure Fusion for Kerberos in Windows 265
Configure Fusion for LDAP 273
Configure Fusion for SAML 282
Configure Fusion for SSO 286
Configure Fusion Logging in Fusion 4.1 and 4.2 288
Configure Google Drive Authentication 289
Configure Grafana, Prometheus, Promtail, and Loki in Fusion .. 309
Configure Highlighting in the Query Workbench 315
Configure Kerberos Authentication for a Jive Datasource .. 321
Configure Managed Fusion for Kerberos in Unix 327
Configure Managed Fusion for Kerberos in Windows 335
Configure Managed Fusion for LDAP 343
Configure Managed Fusion for SAML 352
Configure Managed Fusion for SSO 356
Configure Multiple Nodepool Implementations 357
Configure Neural Hybrid Search 363
Configure OneDrive Authentication 379
Configure OpenID Connect authentication 385
Configure OpenID Connect Authentication 388
Configure Pod Affinity 391
Configure Ray/Seldon vector search 392
Configure Remote V2 Connectors 398

Configure Resource Limits 410
Configure Result Persistence 411
Configure Salesforce authentication 415
Configure Salesforce Authentication 416
Configure Security Trimming for SharePoint Optimized V2 .. 417
Configure Spark Job Resource Allocation 427
Configure Spark Jobs to Access Cloud Storage 429
Configure SSL for App Studio 436
Configure SSL for Solr embedded in Fusion 437
Configure the Fields in Results 442
Configure the Hadoop Client 443
Configure the Highlight Query Pipeline Stage 445
Configure the LWAI Neural Hybrid Search pipeline 449
Configure the LWAI Prediction index Stage 465
Configure the LWAI Prediction query stage 468
Configure the LWAI Vectorize pipeline 471
Configure the Rules Simulator Query Profile 479
Configure the Smart Answers pipelines 481
Configure the Smart Answers Pipelines (5.3 and later) .. 491
Configure the Spark History Server 501
Configure the Suggest Component 507
Configure Voice-Assisted Search 515
Configure Web Site Authentication 525
Configure website authentication 540
Connect to Fusion from App Studio 555
Connect to the Fusion SQL Service from JDBC 558
Crawl a Subset of OneDrive 559
Crawl an Authenticated Website with the Web Connector .. 561
Crawl JavaScript Web Sites 571
Crawl JavaScript websites 576
Crawl REST APIs using the Web connector 580
Crawl SharePoint Online with minimal throttling 588
Crawl SharePoint Online with Minimal Throttling 592
Crawl using the SharePoint Optimized V2 connector 596
Create a Custom Predictive Merchandiser Template Type .. 601
Create a Java Keystore for Use with Fusion 603
Create a new Business Rule 606
Create a New Job 609
Create a New Web App (4.2) 611
Create a query profile 617
Create a Query Profile 620
Create Aggregation Jobs 623
Create an App 626
Create an index profile 627
Create an Index Profile 628
Create and manage Fusion apps 629
Create and manage Managed Fusion apps 633

Create and Run a SQL Aggregation Job 637
Create Custom Reports 643
Create Custom Rule Actions with the API 645
Create Dashboards 655
Create the Initial Admin Password with the API 671
Create the Upload Model Parameters to Cloud job 672
Creating the Upload Model Parameters to Cloud Job 676
Debug Custom JavaScript Index Stages 680
Deduplicate web content using canonical tags 681
Deduplicate Web Content using Canonical Tags 682
Delete a Connector 683
Delete a Query Pipeline 689
Delete an index profile 691
Delete an Index Profile 692
Deploy a pre-trained scispaCy model 693
Deploy a Pre-trained scispaCy Model 698
Deploy a Search App in Fusion's Webapps Service (4.0) .. 703
Deploy a Search App in Fusion's Webapps Service (4.1) .. 706
Deploy a Search App in Fusion's Webapps Service (4.2) .. 713
Deploy a Standalone App 720
Deploy an App in a Servlet Container 723
Deploy App Studio 724
Deploy App Studio Enterprise to a Fusion 5 Cluster (GKE) .. 725
Deploy Fusion 5 on Amazon Elastic Kubernetes Service (EKS) .. 729
Deploy Fusion 5 on Azure Kubernetes Service (AKS) 742
Deploy Fusion 5 on Google Kubernetes Engine (GKE) 752
Deploy Fusion 5 on Other Kubernetes Platforms 766
Deploy Fusion at Scale 775
Deploy the sentiment-general Model 793
Deploy the sentiment-reviews Model 801
Deploying a WAR File 811
Develop a Custom Connector 812
Develop an App with App Studio 823
Develop and deploy a chunking machine learning model with Ray .. 824
Develop and Deploy a Machine Learning Model 844
Develop and deploy a machine learning model with Ray .. 868
Develop and Deploy with App Studio 881
Download an App Studio Project (4.1 and 4.2) 885
Dynamically Adding Icons to Fields 887
Edit parameters in the Query Workbench 888
Edit Parameters in the Query Workbench 889
Edit Rules in the Rules Simulator 890
Edit, Enable, or Disable a Rule 891
Enable and disable query pipeline stages 892
Enable and Disabling Query Pipeline Stages 893
Enable and Disabling Query Rewriting Strategies 894
Enable Asynchronous Query Pipeline Processing 895

Enable Jupyter 903
Enable or disable signals 904
Enable or Disable Signals 907
Enable Read-Only Root File System 910
Enable Recommendations 928
Enable Social Features 929
Enable Superset Fusion 5.4 to 5.5 934
Enable Time-based Partitioning 937
Enable Transport Layer Security (TLS) for Fusion Microservices .. 941
Enable User-for-item recommendations 946
Enable User-for-item Recommendations 947
Encrypt Sensitive Values 948
Evaluate a Smart Answers Query Pipeline 951
Export a Fusion App 959
Export a Managed Fusion App 966
Export Data from the DevOps Center 969
Extract content from Web pages 971
Extract Content from Web Pages 972
Extract short answers from longer documents 973
Extract Short Answers from Longer Documents 977
FAQ Solution Part 1 981
FAQ Solution Part 2 988
Fetch Azure Groups with the AD Connector for ACLs 995
Fetch Items-for-Item Recommendations (Collaborative/ALS Method) .. 1001
Fetch Items-for-Item Recommendations (Collaborative/BPR Method) .. 1008
Fetch Items-for-Item Recommendations (Content-Based Method) .. 1014
Fetch Items-for-User Recommendations (ALS Method) ... 1020
Fetch Items-for-User Recommendations (Collaborative/BPR Method) .. 1027
Fetch Results 1033
Format search results in the Query Workbench 1037
Format Search Results in the Query Workbench 1038
Fusion 4.x.x Zookeeper Ensemble Setup 1039
Fusion Impersonation via a Service Account 1043
Fusion Query Terms and Suggestions 1044
Get Logs for a Spark Job 1046
Get Started with App Studio 1048
Get the Status of App Studio 1049
Handle a Multi-Domain Active Directory 1050
Handle the No-Results Case 1052
Identify trending items or queries 1053
Identify Trending Items or Queries 1057
Import a Fusion App 1061
Import a Managed Fusion App 1068
Import Data with Hive 1071
Import Data with Pig 1078
Import Data with the REST API 1083
Import Signals 1092

Import Solr Collections with the Parallel Bulk Loader .. 1095
Import with the Bulk Loader 1101
Importing Security Certificates 1161
Include a Partial Page 1165
Incorporate Fusion Signals into Appkit Applications .. 1168
Increase SQL Resource Allocations 1170
Index Binary Data from JDBC 1173
Install a Connector - Fusion 4.x 1175
Install a Fusion 4.x Cluster 1179
Install a Fusion 4.x Cluster (Unix) 1182
Install a Fusion 4.x Cluster (Windows) 1189
Install a Fusion License 1195
Install a Springboard embed 1196
Install App Studio 1199
Install Fusion 4.x 1200
Install Fusion 4.x on a Single Node 1201
Install Grafana Dashboards 1207
Install Istio in self-hosted Fusion 1210
Install or update a connector 1213
Install or update a connector - 5.2 and later 1217
Install the Jive Add-On for Fusion 1221
Install the Jive Add-On for Managed Fusion 1222
Install the Random Content Connector 1223
Install the Spark History Server 1225
Integrate Fusion 4.x with an Existing Solr Deployment .. 1226
Integrate Fusion with Your Logging Infrastructure ... 1229
Join Signals with Item Metadata 1231
Keep Rules Consistent Between Clusters 1233
Launch a Remote V2 Plugin 1247
Lay Out a Page 1250
Lay Out Search Results in a Grid 1251
Limit the Crawl Scope for Web Sites 1252
List All Docker Images and Versions 1254
Localize an App 1257
Logging 1261
Lucidworks Documentation 1265
LWAI async summarization and keyword extraction pipeline .. 1350
LWAI NHS plus RAG pipeline 1353
Manage Collections in the Fusion UI 1357
Manage Collections in the Managed Fusion UI 1363
Manage Dashboards 1368
Manage user roles 1376
Merchandising Zero Results 1377
Migrate Fusion Objects 1383
Migrate Managed Fusion Objects 1397
Migrate Milvus to Solr vectors 1404
Migrate to Graph Security Trimming in Fusion 5.8 or later .. 1408

Migrate to Graph Security Trimming in Managed Fusion .. 1411
Optimizing Zero Results 1413
Package a Standalone App 1417
Persist Social Data 1419
Persist Social Data in a Database 1425
Persist Social Data in Fusion 1432
Pin a Product in Experience Optimizer 1433
Pin a product in Predictive Merchandiser 1437
Pin a Product in Predictive Merchandiser 1440
Platform Workflow Processors 1443
Process sitemaps on websites 1448
Protect Against CSRF Attacks 1450
Query rewrite jobs post-processing cleanup 1452
Query Rewrite Jobs Post-processing Cleanup 1460
Quickstart 1468
Read from the MongoDB Oplog 1479
Receive Technical Support 1481
Recreate a Milvus Collection 1489
Refine query results with data source labels 1492
Register for Twitter Credentials 1496
Register for X credentials 1497
Remotely Launch a Plugin 1498
Removing the Security Module 1501
Reorder Query Pipeline Stages 1507
Retrieve a List of Signal Types 1509
Run a Job 1511
Run App Studio in Demo Mode 1512
SAML Configuration 1514
Scale a Fusion 4.x Cluster 1520
Scale Spark Aggregations for Fusion 4.x 1525
Schedule a datasource job 1528
Schedule a Datasource Job 1529
Schedule a job 1530
Schedule a Job 1531
Secure Communication with a SolrCloud Cluster 1532
Security LDAP Realm that Connects to Active Directory .. 1537
See a Smart Answers (QnA) Job's Status and Output ... 1545
Select Fields to Display in the Query Workbench 1546
Self-Contained Deployment 1548
Self-Executing Deployment 1549
Send Your Initial Data Stream 1550
Set Up a Pre-Trained Cold Start Model for Smart Answers .. 1551
Set Up Active Directory Authentication for Appkit ... 1553
Set Up Active Directory Federation Services (ADFS) Authentication for App Studio or Appkit .. 1557
Set up App-Only Auth with Key for SharePoint Online .. 1563
Set Up Appkit Cookie Cracker Authentication 1570

Set up Dynamic Pricing 1572
Set Up Email Alerts 1576
Set Up Facebook Authentication 1579
Set Up Google Authentication 1585
Set Up HTTP Headers Authentication 1590
Set Up Kerberos Authentication 1593
Set Up LDAP Authentication 1597
Set Up Lucidworks Fusion Authentication 1600
Set Up NT Lan Manager Authentication for Confluence Connector .. 1604
Set Up Office 365 Authentication 1609
Set Up Persisting Alerts 1614
Set up semantic vector search 1617
Set Up Semantic Vector Search 1620
Set Up SiteMinder Authentication 1623
Set Up Spring Authentication 1624
Set Up Static Alerts 1630
Set Up Tomcat Container Authentication 1633
Set Up User-specified Alerts 1634
Set Up User-specified Email Alerts 1639
Set Up Windows Authentication 1643
Setting Up a Mock Platform Response 1663
Share Objects Between Fusion Apps 1668
Share Objects Between Managed Fusion Apps 1671
Shut Down App Studio 1674
Smart Answers Cold Start Part 2 1677
Smart Answers Instance Setup (Fusion 4.2 only) 1684
Solr Highlighting 1687
Spark Configuration for Fusion 4.x 1690
Spark Getting Started for Fusion 4.x 1698
Spark Operations 1707
Spark Troubleshooting 1710
Split Couchbase Documents 1715
SSL Security (Unix) 1717
SSL Security (Windows) 1729
Start App Studio 1740
Start or Stop Fusion 1744
Start or Stop the Fusion 5x SQL Service 1755
Start or Stop the Fusion SQL Service 1757
Start the Basic Connector Client 1760
Step-by-step Query Workbench workflow 1761
Step-by-step Query Workbench Workflow 1762
Subscribing to Events 1763
Support for distributed groups claims with the OpenID Connect realm in Azure .. 1765
Support for Distributed Groups Claims with the OpenID Connect Realm in Azure .. 1768
Tracking Fusion Signals 1770
Train a Smart Answers cold start model 1773
Train a Smart Answers Cold Start Model 1776

Train a Smart Answers Supervised Model 1779
Troubleshoot a JDBC Datasource 1787
Troubleshoot an Alfresco datasource 1789
Troubleshoot an Alfresco Datasource 1793
Troubleshoot App Studio Development 1797
Troubleshoot Chrome Binary for Web Connector 1801
Troubleshoot Custom Spark Jobs 1802
Troubleshoot failed datasource jobs 1803
Troubleshoot Fusion 4.x Upgrades 1806
Troubleshoot Performance Issues 1811
Troubleshoot Security Trimming Issues 1814
Troubleshoot SharePoint Permissions 1816
Troubleshoot SQL Queries 1817
Troubleshoot the DevOps Center 1818
Troubleshoot When Installing Fusion 4.x 1819
Troubleshooting Apache Pulsar Issues 1823
Troubleshooting Appkit App Builds and Deployments ... 1832
Tune a Couchbase datasource 1836
Tune a Couchbase Datasource 1837
Update Your Data 1838
Upgrade App Studio 1839
Upgrade Appkit in Existing Apps 1840
Upgrade Fusion Server 3.1.x to 4.2.y 1841
Upgrade Fusion Server 4.0.x to 4.2.y 1862
Upgrade Fusion Server 4.1.x to 4.2.y 1882
Upgrade Fusion Server 4.2.6 to Latest SP 1901
Upgrade Fusion Server 4.2.x to 4.2.y 1904
Upgrade the Appkit Social Module 1923
Upgrade to Fusion 4.x 1926
Upload a JDBC Driver to Fusion Server 1929
Upload a JDBC Driver to Managed Fusion 1933
Use a Configured Platform 1937
Use a Private Repository for Docker Images 1940
Use a Remote Connector with Pulsar Proxy 1942
Use Advanced Linguistics with Babel Street 1949
Use Business Rules in the Rules Editor 1962
Use Dashboards in Fusion 1978
Use Dashboards in Managed Fusion 1989
Use Experience Optimizer facets 1996
Use Experience Optimizer Facets 1999
Use Experience Optimizer query rewrites 2003
Use Experience Optimizer Query Rewrites 2013
Use Experience Optimizer Tasks 2023
Use Federated Search 2027
Use Git with Appkit 2031
Use Hierarchical Facets 2036
Use Jupyter with Fusion SQL 2046

Use Kerberos for JDBC Authentication 2059
Use metadata tags to refine search results 2062
Use Misspelling Detection 2065
Use page metadata 2078
Use Page Metadata 2081
Use phrase detection 2084
Use Phrase Detection 2088
Use Predictive Merchandiser Facets 2098
Use Predictive Merchandiser Query Rewrites 2108
Use Predictive Merchandiser Tasks 2128
Use Predictive Merchandiser templates 2132
Use Predictive Merchandiser Templates 2137
Use query rewrites in the Rules Editor 2142
Use Query Rewrites in the Rules Editor 2147
Use Spark Drivers 2152
Use Superset with Fusion SQL 2156
Use Superset with Managed Fusion SQL 2187
Use Synonym Detection 2217
Use System Metrics 2230
Use the Active Directory ACL Collection (V1) 2233
Use the Analytics screen in the Rules editor 2235
Use the Analytics Screen in the Rules Editor 2239
Use the Blob Editor 2247
Use the Code Editor 2253
Use the Object Explorer 2257
Use the Pipeline Stage JSON Editor 2269
Use the Query Elevation Component 2271
Use the UI to Create Custom Rules 2275
Use Tika asynchronous parsing 2276
Use Tika Asynchronous Parsing 2281
Use Virtual Tables with a Common Join Key 2286
User Access Request Parameters 2289
Using Java Processors 2292
Using the Appkit Tasks Module 2297
Validate Your SharePoint Online Crawl Account Permissions Using PowerShell .. 2302
View and Analyze Logs in Fusion 4.1 and 4.2 2310
View logs in Self-Hosted Fusion 5.6 and later 2312
View Product Details in Predictive Merchandiser 2316
View the SharePoint Export Database File 2318
Web V2 connector OAuth access token configuration ... 2321
Write SQL Aggregations 2324
Writing a Custom Authorization 2326

How-to guides for Fusion 5.9

Whether you're a search engineer looking to index and query your content or a DevOps guru who wants to manage your users and resources, we have a library of how-to articles to help you.

Facet Selections

Total Results: 258

Filter Results:

keyword

Fusion 5 Upgrade from 5.8.x

This article includes instructions for upgrading Fusion from one version to another. In some cases, the instructions do not vary. Other upgrades require special instructions. Start by checking upgrade details for your target version before continuing to the General upgrade process.

☑ Fusion 5.9

How-to

Train a Smart Answers Supervised Model

The Supervised solution for Smart Answers begins with training a model using your existing data and the Smart Answers Supervised Training job (QnA Supervised Training job in 5.1 and 5.2), as explained in this topic. The job includes an auto-tune feature that you can use instead of manually tuning the configuration.

☑ Fusion 5.9

How-to

Keep Rules Consistent Between Clusters

This article teaches you how to keep business rules consistent between clusters. Use a PUT API call to the Query Rewrite API's /query-rewrite/instances endpoint for bulk operations. The JSON payload sent with the PUT call must adhere to the following format:

☑ Fusion 5.9

How-to

Use the Object Explorer

Lucidworks offers free training to help you get started.

☑ Fusion 5.9

How-to

Use Predictive Merchandiser Tasks

This enables Interactive Mode. In Interactive Mode, you can interact directly with some of the widgets on the Predictive Merchandiser board, creating and managing business rules as you simulate your customers' searches. You can:

☑ Fusion 5.9

How-to

Manage Collections in the Fusion UI

Collections can be created or removed using the Fusion UI or the REST API.

☑ Fusion 5.9

How-to

Enable or Disable Signals

You can enable and disable signals using the Fusion UI or the REST API.

☑ Fusion 5.9

How-to

Configure the LWAI Prediction index Stage - *Lucidworks AI*

The LWAI Prediction index stage is a Fusion index pipeline stage that enriches your index with Generative AI predictions. It defaults to asynchronous processing, which does not block the pipeline while waiting for a response from Lucidworks AI.

☑ Fusion 5.9

How-to

Pin a Product in Experience Optimizer

You can pin a document so that it occupies a fixed position in the search results when your customers run a particular query. This can be helpful if you have bestsellers or popular items that you want customers to see first, for example.

☑ Fusion 5.9

How-to

Use Predictive Merchandiser Query Rewrites

You can create query rewrite rules in Predictive Merchandiser. This is similar to using the Rules Editor to create query rewrite rules, except there are additional options in Predictive Merchandiser.

Access the Spark History Server

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion*

Currently, the Spark History Server has only set up a ClusterIP. To expand access, port forward the server using `kubectl`:

```
kubectl get pods –n [NAMESPACE] # to find the Spark History Server pod
kubectl port–forward [POD_NAME] –n [NAMESPACE] 18080:18080
```

You can now access the Spark History Server at `http://localhost:18080`. Run a Spark job and confirm that you can see the logs appear in the UI.

For related topics, see *Spark Operations*.

Accessing HTTPS web resources with signed Enterprise/Self-signed CA Certificates

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion*

Sometimes when you call out to HTTPS resources in connectors, index pipelines, query pipelines, or spark jobs, the HTTPS web resource is signed using an Enterprise or Self-Signed CA. This results in `Caused by: sun.security.provider.certpath.SunCertPathBuilderException: unable to find valid certification path to requested target`.

You can ignore all SSL validation, but that's not secure.

Whether it's Spark, Connectors, Indexing, Query, or something else, adding trusted SSL certificates lets you can access these HTTPS web resources.

Obtain each public certificate in .pem format.

The first thing you need is the public certificates in X.509 PEM format. Refer to How to save a remote server SSL certificate locally as a file for information on how to do this.

One example is to run:

```
openssl s_client -connect self-signed.badssl.com:443 2>/dev/null
</dev/null | sed -ne '/-BEGIN CERTIFICATE-/,/-END CERTIFICATE-/p' >
/home/ndipiazza/Downloads/spark-ssl-test/badssl-com-chain.pem
```

This creates `/home/ndipiazza/Downloads/spark-ssl-test/badssl-com-chain.pem`, which is a public certificate from a public self-signed HTTPS site.

Trust store certificate option

1. *Embed at runtime* each enterprise/self-signed certificate to a local trust store used by the Apache HttpClient.

Add each public certificate to the trust store

Important

This option is only available in Fusion releases 5.9.2 and 5.11.0 and later.

Only use this option if you have installed Fusion using the `$tlsEnabled=true`, which follows this guide: *Enable Transport Layer Security (TLS) for Fusion Microservices*.

The steps are to:

1. Download the truststore from the `fusion-truststore` Kubernetes secret to a local folder.
2. Add any additional certificates to the truststore that you need using Java's `keytool`.
3. Upload the updated truststore back to the secrets store.

At this point, you will no longer have the SSL verification errors when communicating with your HTTPS resource.

Here is a bash script for this first option:

```
# add a cert to the truststore
cert_to_add=/home/ndipiazza/Downloads/spark-ssl-test/badssl-com-
chain.pem kubectl get secret fusion-truststore \ --
namespace=$NAMESPACE \ -o jsonpath="{['data']['truststore\\.jks']}" |
\ base64 -d > truststore.jks keytool -import -alias self-
signed.badssl.com -keystore truststore.jks -file "${cert_to_add}" -
storepass changeit echo "\"$(cat truststore.jks | base64 -w 0)\"" >
truststore.jks.base64 kubectl get secret fusion-truststore \ --
namespace=$NAMESPACE \ -o json | \ jq --slurpfile enc
truststore.jks.base64 '.data["truststore.jks"]=$enc[0]' | \ kubectl
apply -f -
```

VERIFYING IT WORKS

Test access to this HTTPS resource with a Scala example. Below is an example Scala script that uses an Apache HttpClient that fetches a URL, but that URL is self-signed. If you you do not have this public certificate in your trust store, SSL validation will fail.

In this example, we attempt to talk to `self-signed.badssl.com`, which is a public site with a self-signed SSL certificate. Prior to adding the SSL certificate into the trust store, it fails. When you add the cert, it works. Save this script as **Collections** → **Job** → **Add** → **Script**, then save this job and run it.

```
import org.apache.commons.io.IOUtils import
org.apache.http.client.methods.HttpGet import
org.apache.http.impl.client.HttpClients import java.io.{IOException,
InputStream} import java.security.cert.{CertificateFactory,
X509Certificate} import java.security.{GeneralSecurityException,
KeyStore} def url = "https://self-signed.badssl.com" def
createEmptyKeyStore(): KeyStore = { val keyStore =
KeyStore.getInstance("JKS") keyStore.load(null, null); keyStore; }
@throws[IOException] @throws[GeneralSecurityException] def
loadCertificate(publicCertIn: InputStream): X509Certificate = { val
factory = CertificateFactory.getInstance("X.509") val cert =
factory.generateCertificate(publicCertIn).asInstanceOf[X509Certificate
] cert } /** * Returns the text content from a REST URL. Returns a
blank String if there * is a problem. */ def
getRestContent(url:String): String = { val httpClient =
HttpClients.custom() .build(); val httpResponse =
httpClient.execute(new HttpGet(url)) val entity =
httpResponse.getEntity var content = "" if (entity != null) { val
inputStream = entity.getContent content =
IOUtils.toString(inputStream) inputStream.close() } httpClient.close()
content } val content = getRestContent(url) println(content);
```

Embed each enterprise/self-signed certificate to a local trust store used by the Apache HttpClient at runtime

This option embeds the public certificates for the trusted domains in the code, then inserts them into a local keystore used by Apache HttpClient.

Use this approach:

- If you do not have Kubernetes admin access to edit the keystore secret.
- If the HTTPS URL is isolated to a single job or pipeline stage, as it is simpler to do it this way versus adding it to the keystore.

HOW TO USE IT

1. Use the X.509 certificate file created earlier and copy it as text. In your program, create a map of `{alias, pem file string}` called `trustedPublicCertificates` that contains your public certificates.
2. Replace `trustedPublicCertificates` with an alias name as the key of the map, then set the PEM file string contents as the value.

3. Use the Apache HttpClient to access the resource.

EXAMPLES

Here are examples for Apache HttpClient using a custom set of public keys in the SSL trust store to communicate with HTTPS sites with enterprise/self-signed certificates .

Examples:

- *Scala*
- *Nashorn JavaScript*
- *Java*

Scala Example:

```
import org.apache.commons.io.IOUtils import
org.apache.http.client.methods.HttpGet import
org.apache.http.conn.ssl.SSLConnectionSocketFactory import
org.apache.http.impl.client.HttpClients import
org.apache.http.ssl.SSLContexts import java.io.{ByteArrayInputStream,
IOException, InputStream} import java.security.
{GeneralSecurityException, KeyStore} import java.security.cert.
{CertificateFactory, X509Certificate} import
javax.net.ssl.HostnameVerifier def url = "https://self-
signed.badssl.com" def trustedPublicCertificates = Map("sharepoint-
local" -> "-----BEGIN CERTIFICATE-----
\nMIIDeTCCAmGgAwIBAgIJAMnA8BB8xT6wMA0GCSqGSIb3DQEBCwUAMGIxCzAJBgNV\nBA
YTAlVTMRMwEQYDVQQIDApDYWxpZm9ybmlhMRYwFAYDVQQHDA1TYW4gRnJhbmNp\nc2NvMQ
8wDQYDVQQKDAZCYWRTU0wxFTATBgNVBAMMDCouYmFkc3NsLmNvbTAeFw0y\nMTEwMTEyMD
AzNTRaFw0yMzEwMTEyMDAzNTRaMGIxCzAJBgNVBAYTAlVTMRMwEQYD\nVQQIDApDYWxpZm
9ybmlhMRYwFAYDVQQHDA1TYW4gRnJhbmNpc2NvMQ8wDQYDVQQK\nDAZCYWRTU0wxFTATBg
NVBAMMDCouYmFkc3NsLmNvbTCCASIwDQYJKoZIhvcNAQEB\nBQADggEPADCCAQoCggEBAM
IE7PiM7gTCs9hQ1XBYzJMY61yoaEmwIrX5lZ6xKyx2\nPmzAS2BMTOqytMAPgLaw+XLJhg
L5XEFdEyt/ccRLvOmULlA3pmccYYz2QULFRtMW\nhyefdOsKnRFSJiFzbIRMeVXk0WvoBj
1IFVKtsyjbqv9u/2CVSndrOfEk0TG23U3A\nxPxTuW1CrbV8/q71FdIzSOciccfCFHpsKO
o3St/qbLVytH5aohbcabFXRNsKEqve\nww9HdFxBIuGa+RuT5q0iBikusbpJHAwnnqP7i/
dAcgCskgjZjFeEU4EFy+b+a1SY\nQCeFxxC7c3DvaRhBB0VVfPlkPz0sw6l865MaTIbRyo
UCAwEAAaMyMDAwCQYDVR0T\nBAIwADAjBgNVHREEHDAaggwqLmJhZHNzbC5jb22CCmJhZH
NzbC5jb20wDQYJKoZI\nhvcNAQELBQADggEBAC4DensZ5tCTeCNJbHABYPwwqLUFOMITKO
OgF3t8EqOan0CH\nST1NNi4jPslWrVhQ4Y3UbAhRBdqXl5N/NFfMzDosPpOjFgtifh8Z2s
3w8vdlEZzf\nA4mYTC8APgdpWyNgMsp8cdXQF7QOfdnqOfdnY+pfc8a8joObR7HEaeVxhJ
s+XL4E\nCLByw5FR+svkYgCbQGWIgrM1cRpmXemt6Gf/XgFNP2PdubxqDEcnWlTMk8FCBV
b1\nnnVDSiPjYShwnWsOOshshCRCAiIBPCKPX0QwKDComQlRrgMIvddaSzFFTKPoNZjC+\n
CUspSNnL7V9IIHvqKlRSmu+zIpm2VJCp1xLulk8=\n-----END CERTIFICATE-----")
def createEmptyKeyStore(): KeyStore = { val keyStore =
KeyStore.getInstance("JKS") keyStore.load(null, null); keyStore; }
@throws[IOException] @throws[GeneralSecurityException] def
```

```
loadCertificate(publicCertIn: InputStream): X509Certificate = { val
factory = CertificateFactory.getInstance("X.509") val cert =
factory.generateCertificate(publicCertIn).asInstanceOf[X509Certificate
] cert } /** * Returns the text content from a REST URL. Returns a
blank String if there * is a problem. */ def
getRestContent(url:String): String = { val sslBuilder =
SSLContexts.custom() val keyStore = createEmptyKeyStore() for
((alias,certificate) <- trustedPublicCertificates) {
keyStore.setCertificateEntry(alias, loadCertificate(new
ByteArrayInputStream(certificate.getBytes))) } val sslContextBuilder =
sslBuilder.loadTrustMaterial(keyStore, null) val sslContext =
sslContextBuilder.build() val hostnameVerifier = null :
HostnameVerifier val sslConSocFactory = new
SSLConnectionSocketFactory(sslContext, hostnameVerifier) val
httpClient = HttpClients.custom()
.setSSLSocketFactory(sslConSocFactory) .build(); val httpResponse =
httpClient.execute(new HttpGet(url)) val entity =
httpResponse.getEntity var content = "" if (entity != null) { val
inputStream = entity.getContent content =
IOUtils.toString(inputStream) inputStream.close() } httpClient.close()
content } val content = getRestContent(url) println(content);
```

Nashorn JavaScript Example

```
var IOUtils = org.apache.commons.io.IOUtils; var CloseableHttpResponse
= org.apache.http.client.methods.CloseableHttpResponse; var HttpGet =
org.apache.http.client.methods.HttpGet; var SSLConnectionSocketFactory
= org.apache.http.conn.ssl.SSLConnectionSocketFactory; var
CloseableHttpClient = org.apache.http.impl.client.CloseableHttpClient;
var HttpClients = org.apache.http.impl.client.HttpClients; var
SSLContextBuilder = org.apache.http.ssl.SSLContextBuilder; var
SSLContexts = org.apache.http.ssl.SSLContexts; var HostnameVerifier =
javax.net.ssl.HostnameVerifier; var SSLContext =
javax.net.ssl.SSLContext; var ByteArrayInputStream =
java.io.ByteArrayInputStream; var IOException = java.io.IOException;
var InputStream = java.io.InputStream; var StandardCharsets =
java.nio.charset.StandardCharsets; var KeyStore =
java.security.KeyStore; var KeyStoreException =
java.security.KeyStoreException; var NoSuchAlgorithmException =
java.security.NoSuchAlgorithmException; var CertificateException =
java.security.cert.CertificateException; var CertificateFactory =
java.security.cert.CertificateFactory; var X509Certificate =
java.security.cert.X509Certificate; var HashMap = java.util.HashMap;
var Map = java.util.Map; var TRUSTED_PUBLIC_CERTS = new HashMap();
TRUSTED_PUBLIC_CERTS.put("your-alias-here", "-----BEGIN CERTIFICATE---
--\nMIIDPTCCAiW...GOwbrrZXosMs=\n-----END CERTIFICATE-----"); var url
= "https://your-self-signed-site-or-whatever.local"; var sslBuilder =
SSLContexts.custom(); var keyStore = createEmptyKeyStore(); for each
(var e in TRUSTED_PUBLIC_CERTS.entrySet()) { try {
keyStore.setCertificateEntry(e.getKey(), loadCertificate(new
ByteArrayInputStream(e.getValue().getBytes()))); } catch (e) { throw
new RuntimeException(e); } } var sslContextBuilder =
sslBuilder.loadTrustMaterial(keyStore, null); var sslContext =
sslContextBuilder.build(); var sslConSocFactory = new
SSLConnectionSocketFactory(sslContext, null); var httpClient =
HttpClients.custom() .setSSLSocketFactory(sslConSocFactory) .build();
```

```
var response = httpClient.execute(new HttpGet(url)); try { if
(response.getStatusLine().getStatusCode() != 200) { throw new
Exception("Bad status code " + response.getStatusLine()); }
print(IOUtils.toString(response.getEntity().getContent(),
StandardCharsets.UTF_8)); } finally { response.close();
httpClient.close(); } function createEmptyKeyStore() { var keyStore =
KeyStore.getInstance("JKS"); keyStore.load(null, null); return
keyStore; } function loadCertificate(publicCertIn) { var factory =
CertificateFactory.getInstance("X.509"); return
factory.generateCertificate(publicCertIn); }
```

Java Example

```
import org.apache.commons.io.IOUtils; import
org.apache.http.client.methods.CloseableHttpResponse; import
org.apache.http.client.methods.HttpGet; import
org.apache.http.conn.ssl.SSLConnectionSocketFactory; import
org.apache.http.impl.client.CloseableHttpClient; import
org.apache.http.impl.client.HttpClients; import
org.apache.http.ssl.SSLContextBuilder; import
org.apache.http.ssl.SSLContexts; import
javax.net.ssl.HostnameVerifier; import javax.net.ssl.SSLContext;
import java.io.ByteArrayInputStream; import java.io.IOException;
import java.io.InputStream; import java.nio.charset.StandardCharsets;
import java.security.KeyStore; import java.security.KeyStoreException;
import java.security.NoSuchAlgorithmException; import
java.security.cert.CertificateException; import
java.security.cert.CertificateFactory; import
java.security.cert.X509Certificate; import java.util.HashMap; import
java.util.Map; public class HttpClientWithTruststoreLoadedAtRuntime {
static final String url = "https://self-signed.badssl.com"; static
final Map<String, String> TRUSTED_PUBLIC_CERTS = new HashMap<>();
static { // Put your certificate here TRUSTED_PUBLIC_CERTS.put("your-
alias-here" -> "-----BEGIN CERTIFICATE-----
\nMIIDeTCCAmGgAwIBAgIJAMnA8BB8xT6wMA0GCSqGSIb3DQEBCwUAMGIxCzAJBgNV\nBA
YTAlVTMRMwEQYDVQQIDApDYWxpZm9ybmlhMRYwFAYDVQQHDA1TYW4gRnJhbmNp\nc2NvMQ
8wDQYDVQQKDAZCYWRTU0wxFTATBgNVBAMMDCouYmFkc3NsLmNvbTAeFw0y\nMTEwMTEyMD
AzNTRaFw0yMzEwMTEyMDAzNTRaMGIxCzAJBgNVBAYTAlVTMRMwEQYD\nVQQIDApDYWxpZm
9ybmlhMRYwFAYDVQQHDA1TYW4gRnJhbmNpc2NvMQ8wDQYDVQQK\nDAZCYWRTU0wxFTATBg
NVBAMMDCouYmFkc3NsLmNvbTCCASIwDQYJKoZIhvcNAQEB\nBQADggEPADCCAQoCggEBAM
IE7PiM7gTCs9hQ1XBYzJMY61yoaEmwIrX5lZ6xKyx2\nPmzAS2BMTOqytMAPgLaw+XLJhg
L5XEFdEyt/ccRLvOmULlA3pmccYYz2QULFRtMW\nhyefdOsKnRFSJiFzbIRMeVXk0WvoBj
1IFVKtsyjbqv9u/2CVSndrOfEk0TG23U3A\nxPxTuW1CrbV8/q71FdIzSOciccfCFHpsKO
o3St/qbLVytH5aohbcabFXRNsKEqve\nww9HdFxBIuGa+RuT5q0iBikusbpJHAwnnqP7i/
dAcgCskgjZjFeEU4EFy+b+a1SY\nQCeFxxC7c3DvaRhBB0VVfPlkPz0sw6l865MaTIbRyo
```

```
UCAwEAAaMyMDAwCQYDVR0T\nBAIwADAjBgNVHREEHDAaggwqLmJhZHNzbC5jb22CCmJhZH
NzbC5jb20wDQYJKoZI\nhvcNAQELBQADggEBAC4DensZ5tCTeCNJbHABYPwwqLUFOMITKO
OgF3t8EqOan0CH\nST1NNi4jPslWrVhQ4Y3UbAhRBdqXl5N/NFfMzDosPpOjFgtifh8Z2s
3w8vdlEZzf\nA4mYTC8APgdpWyNgMsp8cdXQF7QOfdnqOfdnY+pfc8a8joObR7HEaeVxhJ
s+XL4E\nCLByw5FR+svkYgCbQGWIgrM1cRpmXemt6Gf/XgFNP2PdubxqDEcnWlTMk8FCBV
b1\nnnVDSiPjYShwnWsOOshshCRCAiIBPCKPX0QwKDComQlRrgMIvddaSzFFTKPoNZjC+\n
CUspSNnL7V9IIHvqKlRSmu+zIpm2VJCp1xLulk8=\n-----END CERTIFICATE-----")
} public static void main(String[] args) throws Exception {
SSLContextBuilder sslBuilder = SSLContexts.custom(); KeyStore keyStore
= createEmptyKeyStore(); TRUSTED_PUBLIC_CERTS.forEach((alias,
certificate) -> { try { keyStore.setCertificateEntry(alias,
loadCertificate(new ByteArrayInputStream(certificate.getBytes()))); }
catch (KeyStoreException | CertificateException e) { throw new
RuntimeException(e); } }); SSLContextBuilder sslContextBuilder =
sslBuilder.loadTrustMaterial(keyStore, null); SSLContext sslContext =
sslContextBuilder.build(); SSLConnectionSocketFactory sslConSocFactory
= new SSLConnectionSocketFactory(sslContext, (HostnameVerifier) null);
try (CloseableHttpClient httpClient = HttpClients.custom()
.setSSLSocketFactory(sslConSocFactory) .build()) { try
(CloseableHttpResponse response = httpClient.execute(new
HttpGet(url))) { if (response.getStatusLine().getStatusCode() != 200)
{ throw new Exception("Bad status code " + response.getStatusLine());
}
System.out.println(IOUtils.toString(response.getEntity().getContent(),
StandardCharsets.UTF_8)); } } } private static KeyStore
createEmptyKeyStore() throws KeyStoreException, CertificateException,
IOException, NoSuchAlgorithmException { KeyStore keyStore =
KeyStore.getInstance("JKS"); keyStore.load(null, null); return
keyStore; } private static X509Certificate loadCertificate(InputStream
publicCertIn) throws CertificateException { CertificateFactory factory
= CertificateFactory.getInstance("X.509"); return (X509Certificate)
factory.generateCertificate(publicCertIn); } }
```

Add a Detail Page

How-to

> Compatible versions: 4.2, 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *App Studio*

1. Add the following route to your `routes.js` or `search.routes.js`:

```
.state('details', { url: '/{slug}/{id}', templateUrl: function
(params) { if (params.slug === '') { params.slug = defaultPage;
} if (params.id === '') { return 'views/' + defaultPage +
'.html'; } return 'views/' + params.slug + '-detail.html'; },
controller: 'MainCtrl' })
```

2. Create a view called `search-detail.html`.
3. Ensure that your controller, `search.controller.js` or `main.js`, contains the line, `$scope.params = $stateParams;`. Also, ensure that `$stateParams` is injected into your controller.
4. Add a `search:query` tag in your `search-detail.html` page that contains a nested filter with the ID parameter passed in the URL as the filter:

```
<search:platform var="platform" conf="YOUR_PLATFORM_HERE">
</search:platform> <search:query var="query" parameters=""
results-per-page="1" max-results="1"> <search:filter field="id"
value="{{params.id}}"></search:filter> </search:query>
<search:response var="response" platform='platform'
query="query"> </search:response>
```

5. In your result, set your URL to link to the new detail page. Use that result's ID as a parameter:

```
<search:field name="title" url="search/{{result.id}}">
</search:field>
```

Invalid characters in the URL

If your ID includes non URL safe characters, you will need to create a new unique ID field with these characters removed.

You can refer to these in steps 4 and 5 by doing the following:

Step 4

```
<search:filter field="my_other_field_name" value="{{params.id}}">
</search:filter>
```

Step 5

```
<search:field name="title" url="search/{{result |
field:'my_other_field_name' | actual | encodeURIComponent}}">
</search:field>
```

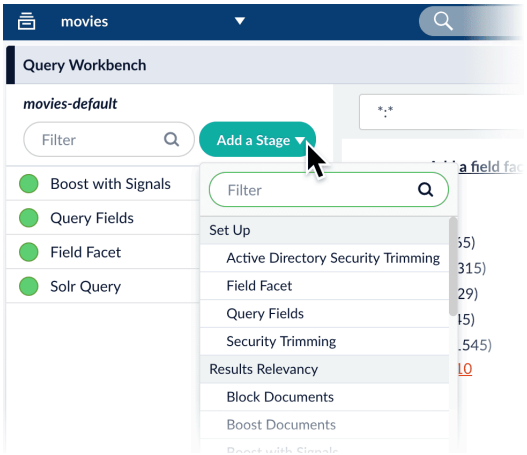
Now if the user goes to **`mysite.com/search/id-of-the-document`**, it will load that page with the ID passed into the URL.

Add a Query Pipeline Stage

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

In the Query Workbench, click **Add a Stage** to add query pipeline stages that can perform query setup, results relevancy, troubleshooting, and more.



The Solr Query stage is the only pipeline stage that is required for querying processes to complete, and therefore exists in every query pipeline. It is always the last stage in a series.

See these pages for more information:

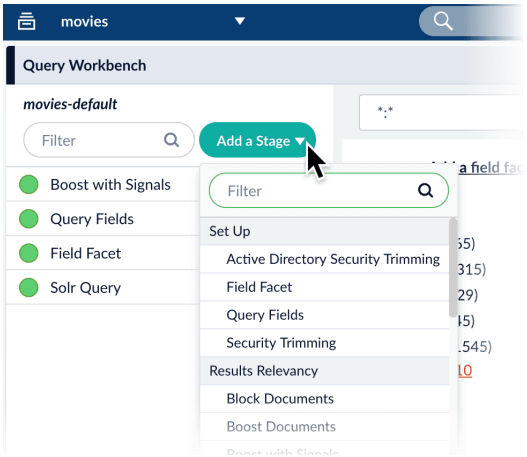
- *Query Workbench*
- *Query Pipeline Stages*
- *Solr Query stage*

Add a Query Pipeline Stage

How-to

> Compatible versions: 4.2, 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion Server*, *Fusion*

In the Query Workbench, click **Add a Stage** to add query pipeline stages that can perform query setup, results relevancy, troubleshooting, and more.



The Solr Query stage is the only pipeline stage that is required for querying processes to complete, and therefore exists in every query pipeline. It is always the last stage in a series.

See these pages for more information:

Fusion 5.x:

- *Query Workbench*
- *query pipeline stages*
- *Solr Query stage*

Fusion 4.x:

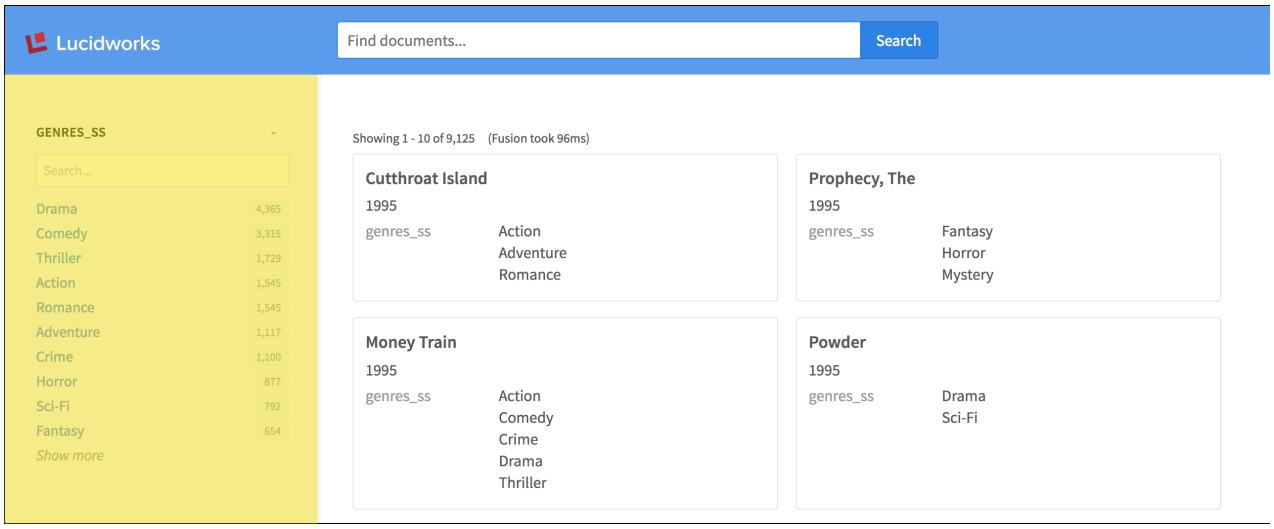
- *Query Workbench*
- *query pipeline stages*
- *Solr Query stage*

Add a Sidebar with Facets

How-to

> Compatible versions: 4.2
>
> Works with: *App Studio*

In App Studio, a sidebar is rendered using the simple `layout:sidebar` tag. Within the sidebar, the `search:facet-list` and `search:facet` tags contain the parameters that configure the facet fields and their display.




```
<layout:block md="1-3" lg="1-4" drawer="left" id="sidebar"
styling="blocksidebar-light"> <layout:sidebar> <layout:box> <!-- List
facets and give them intuitive labels --> <search:facet-list facet-
names="genres_ss" response="response" platform="platform"
query="query" styling="facet-list facet-list-wrappedheader">
<search:facet when-field="genres_ss" show="10" search-enabled="true"
platform="platform" query="query" max-characters="40" show="12" show-
more="24" collapsible="true"></search:facet> <search:facet show="10"
max-characters="40" show="12" show-more="24" collapsible="true">
</search:facet> </search:facet-list> </layout:box> </layout:sidebar>
</layout:block>
```

Add and remove users from applications

How-to

> Works with: Connected Search

This topic details how to add and remove *roles and permissions* to users from an application in Springboard.

Important

You must be a workspace owner or app admin to manage user access from an application.

Add a user to an application

1. *Sign in to Springboard*.
2. In the *Applications Manager screen*, click the three dots next to the application.
3. Click **Edit Permissions**.
4. Click **+ Add User**.
5. In the text box, type the user's name.
6. Select the user's name when it appears.
7. Select a role for this user. An **app admin** can view, edit, and delete the application. An **app viewer** can view the application and cannot make any changes.
8. Click **Save**. The user must sign out and sign in again to view the changes.

Remove a user from an application

1. *Sign in to Springboard*.
2. In the *Applications Manager screen*, click the three dots next to the application.
3. Click **Edit Permissions**.
4. Locate the user to remove and select **No Access** from the dropdown menu next to their name.
5. Click **Save**. The user must sign out and sign in again to view the changes.

Additional information

For conceptual information, see *Roles*.

To manage one user's roles for multiple applications, see *Manage user roles*.

Add Custom Headers to HTTP Requests

How-to

> Compatible versions: 4.2, 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion Connectors*, *Fusion Server*, *Fusion*

The Web V1 connector retrieves data from a Web site using HTTP and starting from a specified URL.

In Fusion 4.1 and later, you can optionally add custom headers to all http `get` requests from the Web connector. For example, you might want to add a header that includes `Connection: keep-alive` to prevent the connector from timing out while crawling your Web site.

To add a custom header, use the configuration parameter `f.addedHeaders`. To send multiple headers, use the following format:

```
Header1: Value1 Header2: Value2
```

To add a custom header in the Fusion UI on any node:

1. Click **Indexing > Datasources**.
2. Click **Add+**, then **Web**.
3. Enter a datasource ID and a start link.
4. Click **Link discovery**.
5. Fill in the **Headers to add to HTTP requests** field.
    - Add each header in the format `HeaderName: HeaderValue`.
    - To add multiple headers to all HTTP requests, put each header on a new line.
6. Fill in any remaining options to configure the datasource.
7. Click **Save**.

Add Custom Headers to HTTP Requests

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

The Web V1 connector retrieves data from a Web site using HTTP and starting from a specified URL.

You can optionally add custom headers to all http `get` requests from the Web connector. For example, you might want to add a header that includes `Connection: keep-alive` to prevent the connector from timing out while crawling your Web site.

To add a custom header, use the configuration parameter `f.addedHeaders`. To send multiple headers, use the following format:

```
Header1: Value1 Header2: Value2
```

To add a custom header in the Managed Fusion UI on any node:

1. Click **Indexing > Datasources**.
2. Click **Add+**, then **Web**.
3. Enter a datasource ID and a start link.
4. Click **Link discovery**.
5. Fill in the **Headers to add to HTTP requests** field.
    - Add each header in the format `HeaderName: HeaderValue`.
    - To add multiple headers to all HTTP requests, put each header on a new line.
6. Fill in any remaining options to configure the datasource.
7. Click **Save**.

Add Smart Answers Pipelines to an Existing App

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion*

When you create a new Fusion app, Fusion also creates the default *Smart Answers* index and query pipelines for that app.

If you have an existing app and you want to add the latest Smart Answers default pipelines, follow the instructions below.

Next, you can *configure the Smart Answers pipelines*.

Create the new Smart Answers Index Pipeline

To add the new Smart Answers index pipeline to an existing app, execute the following curl command, substituting for the `fusion_host` and the `app_name` in the url:

```
curl -u USERNAME:PASSWORD -X POST -H 'content-type: application/json'
-d '{"id":"smart-answers","stages":[{"type":"field-mapping"},
{"type":"solr-dynamic-field-name-mapping"},{"type":"model-to-
milvus","modelId":"smart-answers-
encoder","modelOutputVector":"vector","milvusCollection":"milvus_colle
ction","fieldToEncode":"change-me"},{"type":"solr-index"}]}'
http://{fusion_host}/api/apps/{app_name}/index-pipelines
```

Create the new Smart Answers Query Pipeline

To add the new Smart Answers query pipeline to an existing app, execute the following curl command, substituting for the `fusion_host` and the `app_name` in the url:

```
curl -u USERNAME:PASSWORD -X POST -H 'content-type: application/json'
-d '{"id":"smart-answers","stages":[{"type":"search-fields"},
{"type":"milvus-query-stage","modelId":"smart-answers-
encoder","modelOutputVector":"vector","milvusResultsContextKey":"milvu
s_results","milvusVectorsCollection":"milvus_collection","milvusNumRes
ults":10},{"type":"milvus-ensemble-query-
stage","ensembleExpression":"1.0*milvus_results","resultFieldName":"en
semble_score"},{"type":"solr-query"},{"type":"milvus-response-update-
stage"}]}' http://{fusion_host}/api/apps/{app_name}/query-pipelines
```

Add Tesseract Optical Character Recognition to Fusion Connectors

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion Connectors*, *Fusion*

Tesseract Optical Character Recognition (OCR) solution

The Tesseract OCR is an open source solution that can be added to interact with Fusion connectors in releases 5.2 and later. The example in this topic represents a classic REST service that interfaces with V1 connectors including functions such as file upload and web crawl.

Important

To set up OCR for V2 connectors, you must repeat this process for each individual Docker image related to the connector.

Prerequisites

The following must be established before adding the Tesseract OCR solution:

- A local environment for installing and managing Fusion 5 that includes Google Cloud Tools and other required components.
- The Docker daemon must be running on MacOS and a Docker account for hub.docker.com.
- Fusion release 5.2 or later installed and deployed.

Add Tesseract OCR solution

1. Execute the following to create a Docker file:

```
FROM lucidworks/classic-rest-service:5.2.1 USER root RUN apt-get
install -y tesseract-ocr USER 8764
```

The file:

- Directs Kubernetes Helm to use an existing image with the `<repo>/<image>:<tag>` format as the basis for the new image.
- Switches to the `root` user to perform the Tesseract install.
- Switches back to user `8764` because the classic REST service pod in Kubernetes is not permitted to run as `root`.

2. Build the new Docker image in the same directory as your `Dockerfile`. Enter values that reflect your image and directory. For example:
`docker build -t jdoe/lucidworks/classic-rest-service-ocr:1.0.1`
In Fusion 5.2 and later, the dependency check in Fusion must be included in any custom operation. You must add the dependency image where the custom connector image is stored (at the same level and in the same repository). The sample commands are:

```
docker pull lucidworks/check-fusion-dependency:v1.2.0 docker tag
lucidworks/check-fusion-dependency:v1.2.0 jdoe/check-fusion-
dependency:v1.2.0 docker push jdoe/check-fusion-
dependency:v1.2.0
```

Access the Docker hub to view the image-related information such as name, tag, digest, and operating system.

3. Open the `fusion_values.yaml` file and replace the existing connector image with the custom version. For example:

```
classic-rest-service: image: repository: jdoe name: classic-
rest-service-ocr tag: 1.0.0 nodeSelector: cloud.google.com/gke-
nodepool: default-pool
```

4. Execute the standard process to upgrade (rebuild) the Fusion cluster.
5. Access the Tesseract pod using ssh and run `tesseract -v` to verify Tesseract is installed and working correctly. The result is similar to the following:

```
<<K9s-Shell>> Pod: jdoe-poc/jdoe-classic-rest-service-0 |
Container: classic-rest-service fusion@jdoe-poc-classic-rest-
service-0:/$ tesseract -v tesseract 4.0.0 leptonica-1.76.0
libgif 5.1.4 : libjpeg 6b (libjpeg-turbo 1.5.2) : libpng 1.6.36
: libtiff 4.1.0 : zlib 1.2.11 : libwebp 0.6.1 : libopenjp2 2.3.0
Found AVX2 Found AVX Found SSE
```

6. Access each Fusion parser used for a datasource that performs OCR and select the following items:
   - **Apache Tika**
   - **Include images**

7. Scan one of the following files to test the OCR function:
   - A `.pdf` file, that may contain an underlying `.tiff` file
   - A `.jpeg` file
   - A `.gif` file

8. Verify the parser correctly extracts the information, which includes the `body_t` field.

Adding Twitter and YouTube Results

How-to

> Compatible versions: 4.2, 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *App Studio*, *Fusion*

To add Twitter and Youtube results to the page, you must follow five steps.

1. Get the platform authentication keys

To get Twitter oAuth keys visit the Twitter dev site for Single User OAuth tokens. To get a Youtube API key visit the Google Developers Console, register an application and generate keys.

2. Add the dependencies of the YouTube and Twitter platform.

Add the following dependencies to the `/pom.xml` file at the root of your project

```
<dependency> <groupId>twigkit</groupId>
<artifactId>twigkit.youtube</artifactId>
<version>${project.parent.version}</version> <exclusions> <exclusion>
<groupId>com.fasterxml.jackson.core</groupId> <artifactId>jackson-
core</artifactId> </exclusion> </exclusions> </dependency>
<dependency> <groupId>twigkit</groupId>
<artifactId>twigkit.twitter</artifactId>
<version>${project.parent.version}</version> </dependency>
```

If you are running a local server, then you must restart. Restarting the app-studio script pulls down any new library dependencies.

3. Add configuration files

Because both the Youtube and Twitter platforms take various oAuth and API keys, it is advised to keep these in centralized configuration files. Create these files:

SRC/MAIN/RESOURCES/CONF/PLATFORMS/TWITTER.CONF

```
name: twigkit.search.twitter.Twitter oAuthConsumerKey: XXXXXXXXXX
oAuthConsumerSecret: XXXXXXXXXX oAuthAccessToken: XXXXXXXXXX
oAuthAccessSecret: XXXXXXXXXX
```

SRC/MAIN/RESOURCES/CONF/PLATFORMS/YOUTUBE.CONF

```
name: twigkit.web.youtube.YouTube apiKey: XXXXXXXXXX channelId:
XXXXXXXXXX
```

4. Configure platform, query and response in your View

Next configure a platform, query, and response at the top of the page, as you would do with a standard search. You can copy/paste the below:

```
<!-- YouTube --> <search:platform var="youtubePlatform"
conf="platforms.fusion.youtube"></search:platform> <search:query
var="youtubeQuery" parameters="*" results-per-page="12">
</search:query> <search:response var="youtubeResponse"
platform="youtubePlatform" query="youtubeQuery"></search:response> <!-
- Twitter --> <search:platform var="twitterPlatform"
conf="platforms.fusion.twitter"></search:platform> <search:query
var="twitterQuery" parameters="*" results-per-page="12">
</search:query> <search:response var="twitterResponse"
platform="twitterPlatform" query="twitterQuery"></search:response>
```

You can adjust the `results-per-page` to taste based on the number of results you want. Currently both queries are set to search based on the current query (`parameters="*"`). Set this to an empty string to return just the latest videos/tweets accordingly.

5. Show results on the page

Add the following code to the same page where the platform, query and response were defined in the previous step:

```
<!-- YouTube results --> <search:result-list
response="youtubeResponse"> <search:result> <search:field
name="snippet.title" styling="title"></search:field> <iframe
width="400" height="300" id="player" ng-src="
{{'https://www.youtube.com/embed/' + result.id }}" frameborder="0" >
</iframe> </search:result> </search:result-list> <!-- Twitter results
--> <search:result-list response="twitterResponse"> <search:result>
<search:field name="text" styling="title"></search:field>
<search:field name="user.screen_name" styling="description">
</search:field> </search:result> </search:result-list>
```

If embedding a YouTube video player like in the above example, you must also add the YouTube domain as a trusted site. To do this, add the following code to the **`/src/main/webapp/search/app/app.js`** file after the controller definitions:

```
.config(function($sceDelegateProvider) {
$sceDelegateProvider.resourceUrlWhitelist([ 'self',
'https://www.youtube.com/embed/**']); });
```

After adding the above block, the entire **`/src/main/webapp/search/app/app.js`** file should look similar to the following:

```
'use strict'; const BUILD_PATH = '/dist/'; const basePath =
(document.getElementsByTagName('base')[0] || {href: ''}).href;
__webpack_public_path__ = window.__webpack_public_path__ =
basePath.replace(/\/$/, '') + BUILD_PATH; import
'../../styles/twigkit.less'; import {RoutesModule} from
'./routes/routes.module.js'; import {SearchController} from
'./controllers/search.controller'; import {LoginController} from
"./controllers/login.controller"; import {ServicesModule} from
"./services/services.module"; import {DirectivesModule} from
"./directives/directives.module.js"; let appModule = angular
.module('appStudioSearch', [ , 'ui.router' , 'ngAnimate' , 'lightning'
, RoutesModule.name , DirectivesModule.name , ServicesModule.name ])
.run(['$rootScope', '$window', '$twigkit', '$location', function
($rootScope, $window, $twigkit, $location ) {
$rootScope.$on('response_response_error', function (response) {
$rootScope.showErrorModal = true; }); $rootScope.closeErrorModal =
function () { $rootScope.showErrorModal = false; }
$rootScope.$on('httpInterceptorError', function (event, error) { /** *
The tests here must match the tests in *
twigkit.security.springsecurity.matchers.NonAjaxRequestMatcher::matche
s */ if ((error.status == '401' || error.status == '403') &&
(error.config.url.indexOf('twigkit/api') !== -1 ||
error.config.url.indexOf('twigkit/services') !== -1 ||
error.config.url.indexOf('twigkit/secure/services') !== -1 ||
error.config.url.indexOf('views/') !== -1 )) { // Error was a 403 and
URL was a non-user facing path - redirect to login
$rootScope.$broadcast('twigkitApi403', { error: error, url:
$location.url() }); var loginPath = $twigkit.getContextPath('/') +
$twigkit.getConstant('loginPage', 'login/'); // If we are already on
the login page, do not redirect if ($window.location.pathname !==
loginPath ) { $window.location.href = loginPath; } } }); }])
.controller('searchCtrl', SearchController) .controller('loginCtrl',
LoginController) .config(function($sceDelegateProvider) {
```

```
$sceDelegateProvider.resourceUrlWhitelist([ 'self', 'https://www.youtube.com/embed/**']); }); angular.bootstrap(document, ['appStudioSearch']);
```

Advanced Search

How-to

> Compatible versions: 4.2
>
> Works with: *App Studio*

App Studio provides an advanced search capability where a user can specify one or more filters to a query. This provides a more granular search experience, allowing for users to craft tailored queries to quickly find exactly what they are looking for.

How to Configure Advanced Search in App Studio Enterprise

1. Open `../src/main/webapp/search/app/controllers/search.controller.js`
2. Add the following code block inside the `SearchController` function:

```
$scope.advancedBox = { showAdvanced: false, toggleAdvanced:
function() { $scope.advancedBox.showAdvanced =
!$scope.advancedBox.showAdvanced; } };
```

3. Open `../src/main/webapp/views/partials/header.tpl.html`
4. Replace the `<search:box>` element with the code block below. The original `<search:box>` tag may span multiple lines and include nested elements such as `<instant:result-list>` and/or `<query:suggestions>`. You must replace the entire `<search:box>` tag with the following code block:
   Note: You must also update the `<advanced:date-range>` and `<advanced:text>` tags to point to fields in your data set. When using the `<advanced:text>` tag for seaching within field values you **must** use an untokenized field.

```
<search:advanced> <search:box placeholder="Search..."></search:box> <a ng-click="advancedBox.toggleAdvanced()" class="toggle" styling="text-white">Advanced Search</a> <div class="tk-popover" ng-show="advancedBox.showAdvanced"> <a ng-click="advancedBox.toggleAdvanced()" class="close">close</a> <layout:grid> <layout:block> <advanced:date-range fields="published_date_tdt=Published Date,last_modified_tdt=Last Modified Date" min-view="day"></advanced:date-range> <advanced:text fields="category_t=Category,theme_t=Theme"></advanced:text> </layout:block> </layout:grid> </div> </search:advanced>
```

5. Replace the `<layout:include>` tags in any HTML files that reference `header.tpl.html` file with an `<ng:include>` tag. Replace this line: `<layout:include file="views/partials/header.tpl.html"></layout:include>` With this line: `<ng:include src="'views/partials/header.tpl.html'"></ng:include>`
6. Add the following code block to `../src/main/webapp/styles/includes/custom.less` :
   Note: You may need to update the `color: rgb(255,255,255)` attribute in the `toggle` class below in order for the `Advanced Search…` text to be visible in the header.

```
.tk-advanced-search { top: -5px; .tk-advanced-search .tk-
advanced-search-input { -webkit-box-align: center; -ms-flex-
align: center; align-items: center; } .toggle { color:
rgb(255,255,255); font-size: 0.9em; text-decoration: underline;
float: right; margin-top: 2px; font-style: italic; } .tk-popover
{ background: #fff; &:before, &:after { display: none; } .close
{ float: right; margin: -12px -8px 0 0; } select { margin-
bottom: 0; margin-top: 0; } } } .tk-advanced-search .tk-
advanced-search-input .entry, .tk-advanced-search .tk-advanced-
search-input .field-select, .tk-advanced-search .tk-advanced-
search-input .tk-widget-date-range .tk-from-date { margin-right:
.5em; } .tk-widget-date-range .tk-from-date, .tk-widget-date-
range .tk-to-date { width: 48%; margin:0; } .tk-widget-date-
range .tk-to-date .tk-stl-datetimepicker { right:-40%; } .tk-
widget-date-range .tk-from-date .tk-stl-datetimepicker { left:
-40%; } .tk-advanced-search-input-text .entry { width:29.5%;
margin-right: .8em!important; } .tk-advanced-search .tk-popover
{ .tk-advanced-search-input { clear:both; width: 100%; max-
width: 100%; margin:10px 0 0; input { padding-left: 1em; }
.entry { // width: 40%; // max-width: 220px; input { width:
100%; margin:0; margin-top: 2px; display: inline-
block!important; } } } select { width: 28%; max-width: 150px;
height: 38px; } .tk-widget-date-range { label { display: none; }
input { margin-bottom: 0; } } }
```

7. Hard refresh the search UI in your browser. There will now be an **`Advanced Search…`** link under the search box in the header. Click this link to bring up the Advanced Search dialog box.

AEM connector OAuth authorization

How-to

> Compatible versions: 5.5, 5.9, 5.10
>
> Works with: *Fusion Connectors*, *Fusion*

The *AEM V2 Connector* supports OAuth 2.0 authorization with JWT token.

Supported authorization options

Requests are authorized by including an Access Token in the Authorization header.

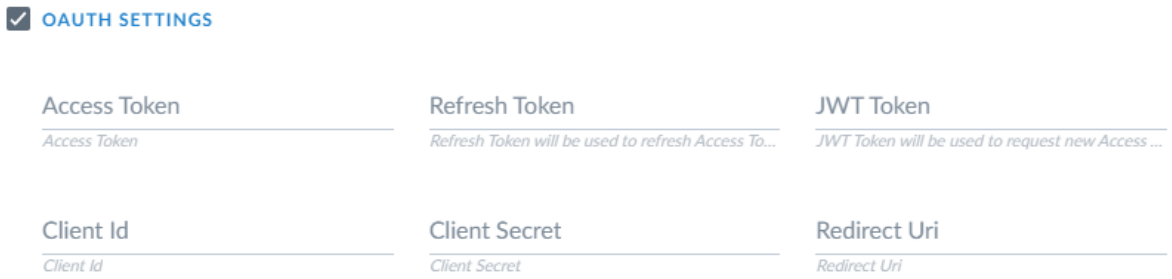
Example:

```
curl –H 'Authorization: Bearer {placeholderAccessToken}'
http://HOST:PORT/content/COMPANY/us/en/community/messaging.html
```

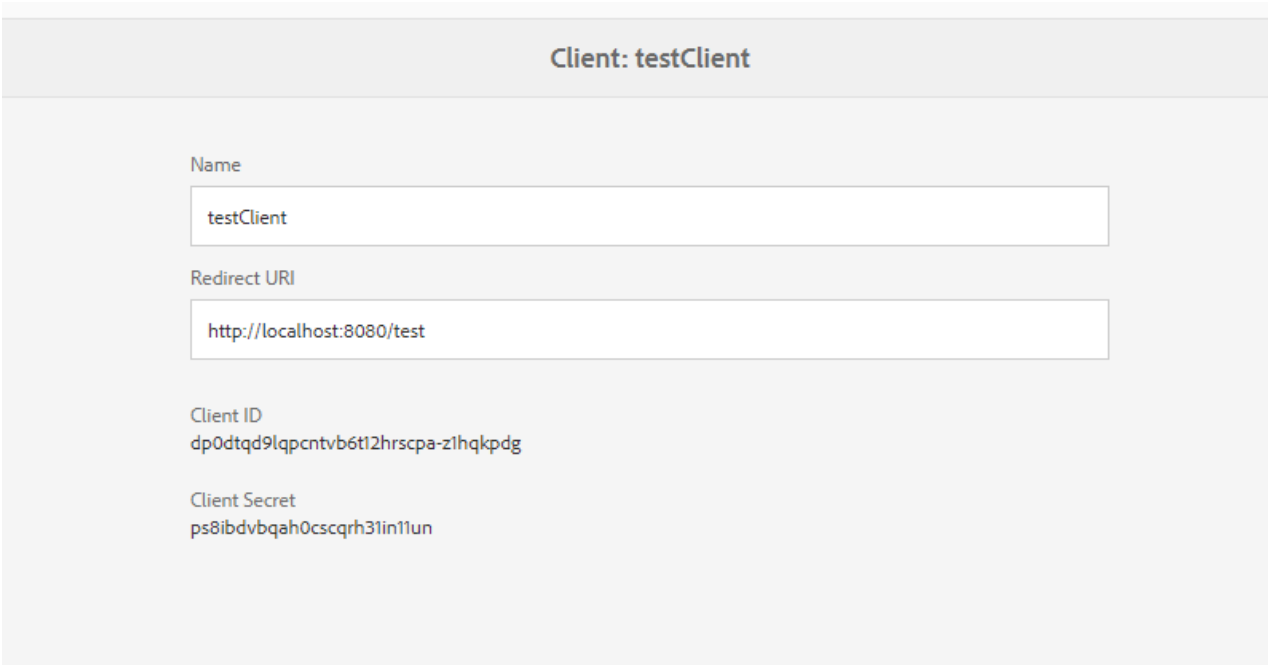
There are three ways the connector can get Access Token:

- From the datasource configuration
- From AEM server using Refresh Token
- From AEM server using JWT token

Request Access Token, Refresh Token, and JWT token manually and set them in the datasource configuration:



Other settings: `Client Id` , `Client Secret` , and `Redirect Uri` can be found in the AEM admin page under **Security→Oauth Clients**:



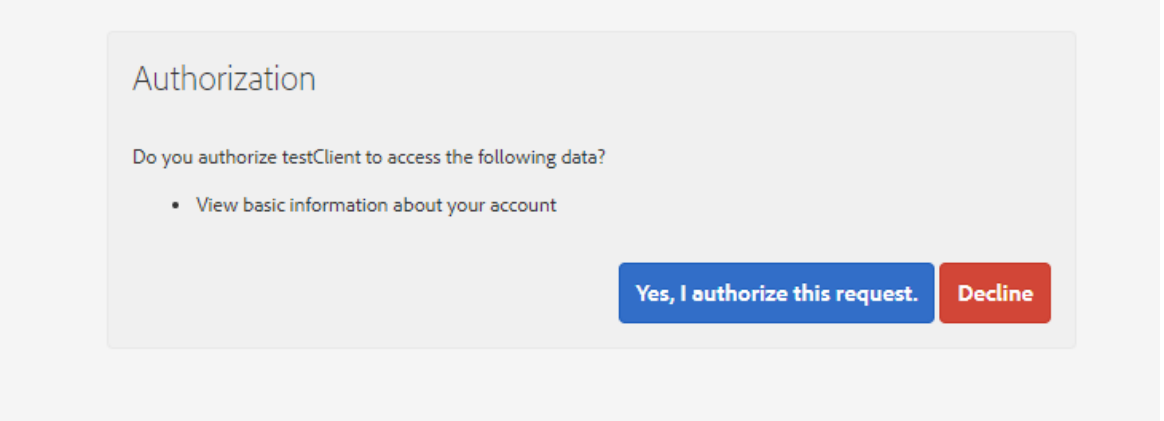
Getting Access Token

1. Open this URL in a browser:

```
http://HOST:PORT/oauth/authorize?
response_type=code&client_id=placeholderClientId&client_secret=p
laceholderClientSecret&username=admin&password=PASSWORD&scope=of
fline_access&redirect_uri=REDIRECT_URI
```

2. You are redirected to login page (if you are not logged in):

3. Logging in redirects you to confirm the authorization. Click **Yes, I authorize this request**.



4. You are redirected to the URL provided in `redirect_uri` with parameter `code`: `<REDIRECT_URI>?code=<AUTHORIZATION_CODE>`.
5. Copy the value of the `code` parameter. This is your Authorization Code.
6. Execute the request to get Access Token:

```
curl --location --request POST 'http://HOST:PORT/oauth/token?
code={placeholderAuthorizationCode}&client_id=
{placeholderClientId}&client_secret=
{placeholderClientSecret}&grant_type=authorization_code&redirect
_uri=<REDIRECT_URI>' --header 'Content-Type: application/x-www-
form-urlencoded' --header 'Accept: application/json'
```

Example:

```
curl --location --request POST
'http://34.71.168.50:4502/oauth/token?code=
{placeholderAuthorizationCode}&client_id=
{placeholderClientId}&client_secret=
{placeholderClientSecret}&grant_type=authorization_code&redirect
_uri=http://localhost:8080/test' --header 'Content-Type:
application/x-www-form-urlencoded' --header 'Accept:
application/json'
```

```
"access_token":"{placeholderAccessToken}","refresh_token":"
{placeholderRefreshToken}","expires_in":3600
```

Getting Refresh Token

To get Refresh Token, follow the same proccess for Access Token, but:

1. You must include `offline_access` in the scope list.
2. You must revoke all the previously obtained token. It can be done by clicking **Revoke All Tokens**.

Getting JWT Bearer Token

1. **Download Private Key** from the AEM Oauth client section.
   You should have downloaded file **store.p12**.
2. Run:

```
openssl pkcs12 -in store.p12 -out store.crt.pem -clcerts -
nokeys
```

When asked about password type `notasecret`.
You should have generated file named **store.crt.pem**.

3. Run

```
openssl pkcs12 -in store.p12 -passin pass:notasecret -nocerts -
nodes -out store.private.key.txt
```

You should have generated file named **store.private.key.txt**.

4. Create JWT token with the below payload and encrypt it with the private key using RS256:

```
{ "aud": "http://<HOST>:<PORT>/oauth/token", "iss": "<client
id>", "sub": "<user name>", "exp": <Current time in
milliseconds+expiry>, "iat": <Current time in milliseconds>,
"scope": "<scope>", "cty": "code" }
```

For example, install pyjwt to use this python script:

```
pip install pyjwt
```

```
import jwt payload_data = { "aud":
"http://34.71.168.50:4502/oauth/token", "iss":
"dp0dtqd9lqpcntvb6t12hrscpa-z1hqkpdg", "sub": "admin", "exp":
1697840880541, "iat": 1697740880541, "scope": "offline_access",
"cty": "code" } private_key = open('store.private.key.txt',
'r').read() token = jwt.encode( payload=payload_data,
key=private_key, algorithm='RS256' ) print(token)
```

AEM connector OAuth authorization

How-to

> Compatible versions: 5.5, 5.9, 5.10
>
> Works with: *Fusion Connectors*, *Managed Fusion*

The *AEM V2 Connector* supports OAuth 2.0 authorization with JWT token.

Supported authorization options

Requests are authorized by including an Access Token in the Authorization header.

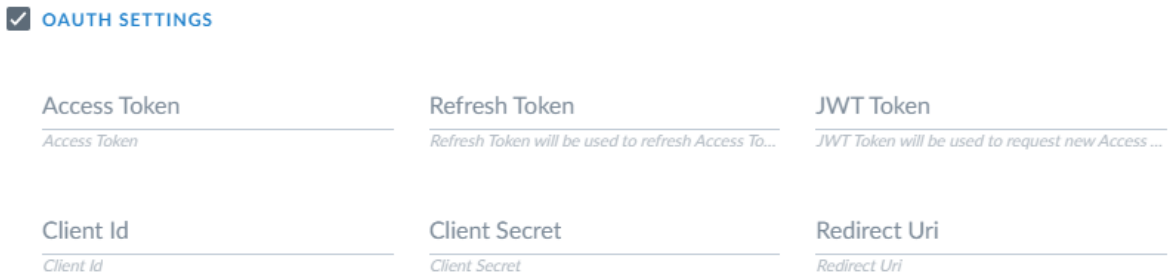
Example:

```
curl –H 'Authorization: Bearer {placeholderAccessToken}'
http://HOST:PORT/content/COMPANY/us/en/community/messaging.html
```

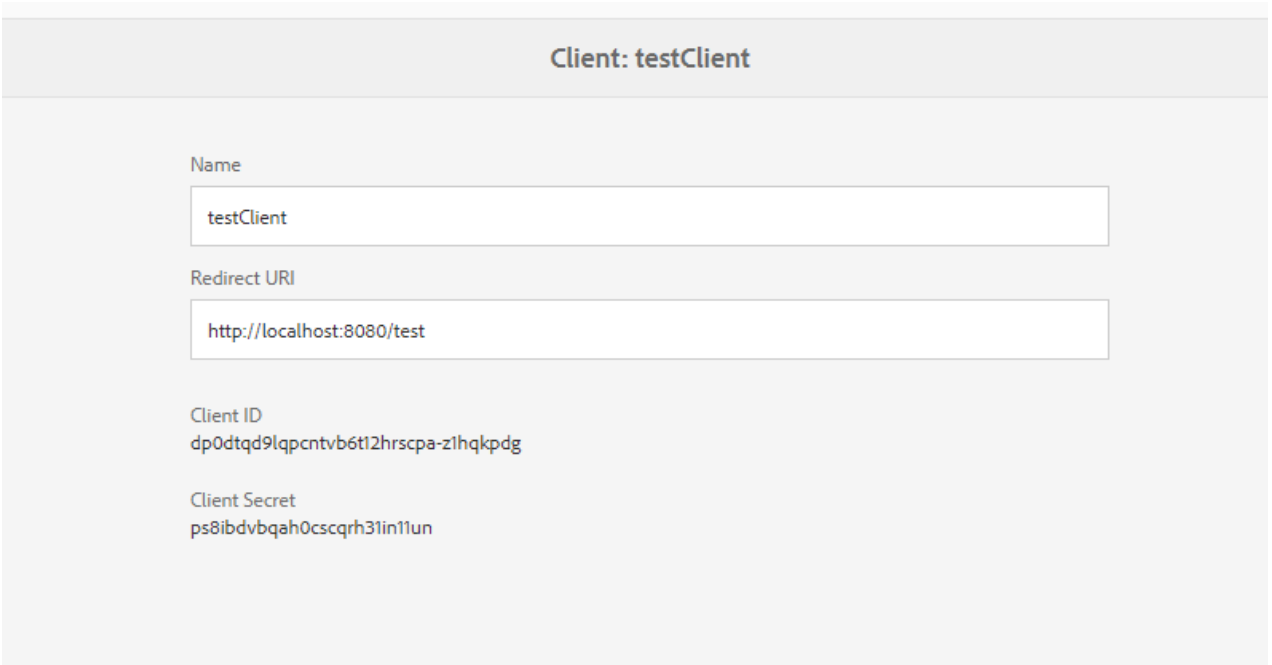
There are three ways the connector can get Access Token:

- From the datasource configuration
- From AEM server using Refresh Token
- From AEM server using JWT token

Request Access Token, Refresh Token, and JWT token manually and set them in the datasource configuration:



Other settings: `Client Id` , `Client Secret` , and `Redirect Uri` can be found in the AEM admin page under **Security→Oauth Clients**:



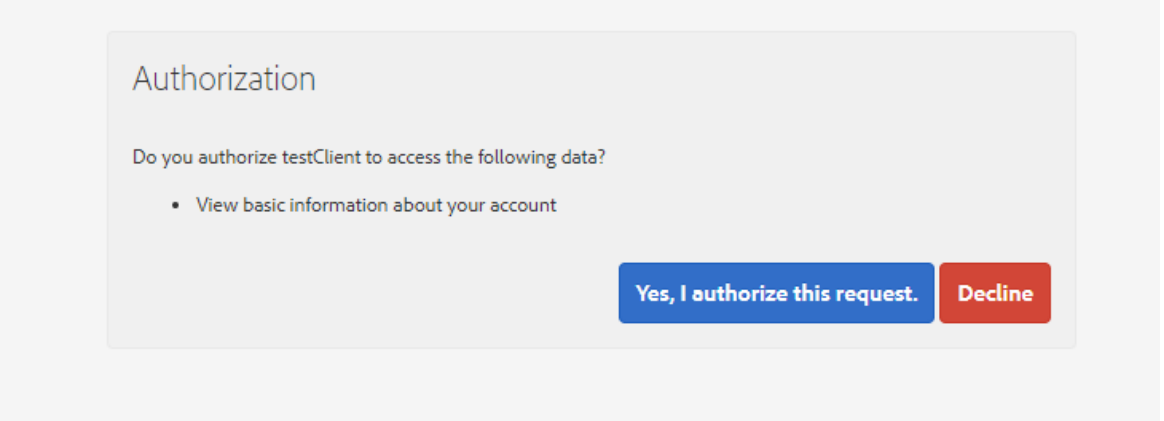
Getting Access Token

1. Open this URL in a browser:

```
http://HOST:PORT/oauth/authorize?
response_type=code&client_id=placeholderClientId&client_secret=p
laceholderClientSecret&username=admin&password=PASSWORD&scope=of
fline_access&redirect_uri=REDIRECT_URI
```

2. You are redirected to login page (if you are not logged in):

3. Logging in redirects you to confirm the authorization. Click **Yes, I authorize this request**.



4. You are redirected to the URL provided in `redirect_uri` with parameter `code`: `<REDIRECT_URI>?code=<AUTHORIZATION_CODE>`.
5. Copy the value of the `code` parameter. This is your Authorization Code.
6. Execute the request to get Access Token:

```
curl --location --request POST 'http://HOST:PORT/oauth/token?
code={placeholderAuthorizationCode}&client_id=
{placeholderClientId}&client_secret=
{placeholderClientSecret}&grant_type=authorization_code&redirect
_uri=<REDIRECT_URI>' --header 'Content-Type: application/x-www-
form-urlencoded' --header 'Accept: application/json'
```

Example:

```
curl --location --request POST
'http://34.71.168.50:4502/oauth/token?code=
{placeholderAuthorizationCode}&client_id=
{placeholderClientId}&client_secret=
{placeholderClientSecret}&grant_type=authorization_code&redirect
_uri=http://localhost:8080/test' --header 'Content-Type:
application/x-www-form-urlencoded' --header 'Accept:
application/json'
```

```
"access_token":"{placeholderAccessToken}","refresh_token":"
{placeholderRefreshToken}","expires_in":3600
```

Getting Refresh Token

To get Refresh Token, follow the same proccess for Access Token, but:

1. You must include `offline_access` in the scope list.
2. You must revoke all the previously obtained token. It can be done by clicking **Revoke All Tokens**.

Getting JWT Bearer Token

1. **Download Private Key** from the AEM Oauth client section.
   You should have downloaded file **store.p12**.
2. Run:

```
openssl pkcs12 -in store.p12 -out store.crt.pem -clcerts -
nokeys
```

When asked about password type `notasecret`.
You should have generated file named **store.crt.pem**.

3. Run

```
openssl pkcs12 -in store.p12 -passin pass:notasecret -nocerts -
nodes -out store.private.key.txt
```

You should have generated file named **store.private.key.txt**.

4. Create JWT token with the below payload and encrypt it with the private key using RS256:

```
{ "aud": "http://<HOST>:<PORT>/oauth/token", "iss": "<client
id>", "sub": "<user name>", "exp": <Current time in
milliseconds+expiry>, "iat": <Current time in milliseconds>,
"scope": "<scope>", "cty": "code" }
```

For example, install pyjwt to use this python script:

```
pip install pyjwt
```

```
import jwt payload_data = { "aud":
"http://34.71.168.50:4502/oauth/token", "iss":
"dp0dtqd9lqpcntvb6t12hrscpa-z1hqkpdg", "sub": "admin", "exp":
1697840880541, "iat": 1697740880541, "scope": "offline_access",
"cty": "code" } private_key = open('store.private.key.txt',
'r').read() token = jwt.encode( payload=payload_data,
key=private_key, algorithm='RS256' ) print(token)
```

App Studio Development & Deployment Guide

How-to

> Compatible versions: 4.2
>
> Works with: *App Studio*

Development

In building out a search application, App Studio developers use an IDE (Intellij, eclipse, VS Code), maven for server-side dependency management and its integrated jetty plugin, NPM 9.0.2 for client-side dependency management and Java 1.8 JDK.

We first will clone the repository of our project on our local system. From there we make additions and edits to the code which we then commit to the repo.

In order to test our changes in real time, we use the integrated maven jetty plugin to serve up our local application at *http://localhost:8080* where we can see most changes made to the code in real time.

Environment / Profile management

For each environment, we build out a profile to overlay specific parameters to this environment. See the section entitled 'Package the web application for different runtime environments - build profiles' here for more details: https://dev.twigkit.net/docs/Getting_started/Installation/Building_from_source/?q=profile

Example command: `mvn clean jetty:run –Dtwigkit.profile=profileName`

In App Studio apps, we can have different profiles for each environment (Dev, QA and Prod) as well as local development. These profiles contain configuration files that should overwrite the files with the corresponding names and directory locations in the main resources directory.

Setting up a profile

Following are the files you may want to specify for a given environment in each profile:

`message.service.fusion.conf` – This config file is responsible for the configuration of the way we send signals to the Fusion endpoint. The important property here is query-profile, which we should ensure points to the query profile that manages the main collection to which we want to send signals. I.e. – if we specify a query profile that points to the myCollection collection, App Studio will send signals to the myCollection_signals collection (if it does exist).

`platforms.fusion.fusion.conf` – This is our base Fusion platform configuration file. It manages all the configuration settings that are common amongst each Fusion platform. These settings are inherited by any config file in the platforms.fusion directory, as this file has the same name as the directory it resides in.

`platforms.fusion.data.conf` – This is our configuration file for the main platform, by default. The only required parameter is the query-profile to send these queries to. This is included in the profiles as during the course of development, there were some differences across environments in the name of this collection, so we set them individually for each environment.

The important property to note here is query-profile which tells App Studio which query profile to fire search requests against.

`security.kerberos.conf` – This is an example of a security configuration file. In this case, the file in which Kerberos security is configured. The parameters set here are referenced in our configuration of resources/spring-security.xml when the app is loaded. The main parameters to understand are service-principal and keytab-location. The service principal should be the one used when creating the keytab file. The keytab location is the location in which the keytab resides on the server that App Studio is running.

`security.security.conf` – The function of this file is to set general parameters related to security. The type parameter should be the type of security your app should use (kerberos, spring_security, saml etc..). The password parameter is the key that is used when hashing passwords using twigcrypt utility. It is to be used in combination with a 'secret seed' in order to generate the hash.

`services.api.fusion.conf` – This is the main file in which the connection to the Fusion API is configured. The host, port and protocol values are all given independently. The credentials given are that of a service account that has POST rights to enable the sending of signals to Fusion.

`logback.xml` – In this file, we can add additional <logger> tags to print out additional logging on specific classes in the webapps or local logs. The main logging comes from the root package `twigkit`.

Packaging and Deployment

Packaging

To deploy our code to the server, we build and package the source code and dependencies into a WAR file using maven.

Command: `mvn clean package`

In order to package the app with the proper connection details and other configuration settings for the particular environment (dev, prod, etc..) see the section above re the profile parameter.

Command: `mvn clean package -Dtwigkit.profile=profileName`

You can watch the logs for any compilation errors. If it's not already, rename the resulting WAR file to ROOT.war.

Deployment

CREATE A NEW APP STUDIO CONTEXT IN FUSION

Prerequisites:
1. Fusion is installed
2. A Fusion App is created

Execute the following command using a user who has Admin rights:

```
curl -v -u USERNAME:PASSWORD -H 'Content-type: application/json' -X
POST 'http://localhost:8764/api/apollo/apps/my-fusion-app-
name/webapps' -d '{"id": "myasapp","name": "My AS App","contextPath":
"/"}'
```

UPLOAD THE WAR TO THE FUSION WEBAPPS SERVICE

```
curl -v -u USERNAME:PASSWORD -H 'Content-type: application/zip' -X PUT
'http://localhost:8764/api/apollo/apps/my-fusion-app-
name/webapps/myasapp/war' --data-binary @ROOT.war
```

This will put the WAR file into the Fusion blob store, and tell the Webapps service that this is the file that should be started in the specified context.

App Studio Development Troubleshooting

How-to

> Compatible versions: 4.2
>
> Works with: *App Studio*

Troubleshooting

Data is not appearing on the page as expected

CHECK RESPONSE DATA

If you open your browser's dev tools, choose the 'Network' tab, and look for an XHR request similar to the one shown below (starting with platforms.fusion...) you can navigate through the values being returned from Fusion.

To determine which response is which (as you may have multiple) you will need to look at the platform that populates the response, that populates the element on your page.

You can then track down the value you are looking for in here. So if looking at results, a result list iterates through the results, so we should look inside a result. In each result, there's a set of fields and associated values.

Note that other processing may happen in between receiving the response and displaying it on the page (response processors, modifications in the HTML, security trimming, etc..)

+ image:/assets/images/app-studio/4.2/devguide1.png

+ image:/assets/images/app-studio/4.2/devguide2.png

GET QUERY FROM WEBAPPS/SERVLET LOGS AND FIRE IT AGAINST FUSION

If the data in your response object is not as you expect, you can get the query from the App Studio logs to fire directly against the Fusion endpoint, without App Studio as a middle-man. Note that you'll need your logger configured to print out debug logging for the twigkit class in logback.xml in your environment profile. Or if not using a profile, in the main resources directory.

```
<logger name="twigkit" level="DEBUG"/>
```

Look in the webapps logs are under **fusion/version/var/log/webapps/**. You can sort them by date `(ls -l)` and view the latest one with the suffix of stderrout. In here you will see some entries that should follow the format of the following:

```
08:57:41.058 DEBUG twigkit.search.solr.SolrPlatform - Searching Solr
with start=0&rows=20&hl=true&q=match&facet=true&facet.field=
{!ex%3Dgenre_ss}genre_ss&facet.field=year_i&fl=*,score&fq=
{!tag%3Dgenre_ss}genre_ss:(Animation)&username=admin&session=37045658-
9052-4ba4-b1d2-
0221be5e2d02&spellcheck=true&spellcheck.collate=true&q.op=OR took 200
ms
```

To determine which entry is the correct one, you'll need to check the query that populates the response you're interested in, and match the parameters you're setting in the tag as well as any Solr parameters being applied to the query via interaction on the page (keyword = q, facet filters = fq, etc..).

You can then go to the query workbench in Fusion and select the Published URI in the URI tab:

+ image:/assets/images/app-studio/4.2/devguide3.png

Remove every parameter given in this URL after 'select?', append the parameters from the webapps logs and put this URL into your browser. The result will be response of the query fired by App Studio, but unaltered by App Studio.

Page is totally blank/elements look broken or strange

Check browser console

If you open your browser's dev tools, choose the 'Console' tab, you can check for any issues related to client side components. If you see the following error, you should wipe your node modules directory and reinstall them before building. Otherwise typically an error here suggests there is a problem in your Javascript code.

```
Uncaught Error: [$injector:modulerr] Failed to instantiate module
myApp due to: Error: [$injector:modulerr] Failed to instantiate module
lightning due to: Error: [$injector:nomod] Module 'lightning' is not
available! You either misspelled the module name or forgot to load it.
If registering a module ensure that you specify the dependencies as
the second argument.
```

Perform the following steps to reinstall the node modules.

1. `rm -rf node_modules package-lock.json`
2. `pm cache clear -f`
3. `npm install`

Ensure there are no errors in any of your custom javascript code.

Assign Connectors Classic Jobs to Specific Nodes

How-to

> Compatible versions: 4.2
>
> Works with: *Fusion Server*

This page applies to Fusion version 4, starting with APIs added in Fusion 4.2.4

Get connector job's current node assignment

Find assigned node for a connectors data source job.

Method: GET

```
/api/apollo/connectors-classic/datasources/${yourDatasourceID}/node_assignment
```

Example:

```
proxyUrl=https://FUSION_HOST:8764 yourDatasourceID=sharepoint563263
curl -X GET -u USERNAME:PASSWORD "${proxyUrl}/api/apollo/connectors-
classic/datasources/${yourDatasourceID}/node_assignment"
```

Powershell:

```
$proxyUrl="https://FUSION_HOST:8764" $yourDatasourceID="yourdsid"
$username = "admin" $password = "password" $base64AuthInfo =
[Convert]::ToBase64String([Text.Encoding]::ASCII.GetBytes(("{0}:{1}" -
f $username,$password))) Invoke-RestMethod -Method Get -Headers
@{Authorization=("Basic {0}" -f $base64AuthInfo)}
"${proxyUrl}/api/apollo/connectors-
classic/datasources/${yourDatasourceID}/node_assignment"
```

Assign connector job to a specific node

You can stick a connectors job to a specific connectors-classic node.

Note

Use IP address and not the host name.

Method: PUT

```
/api/apollo/connectors-
classic/datasources/${yourDatasourceID}/node_assignment/http%3A%2F%2F${ipOfConnectorsCl
assicNode}%3A${portOfConnectorsClassic}%2Fconnectors%2Fv1%2Fconnectors-classic
```

Example:

Bash:

```
proxyUrl=https://FUSION_HOST:8764 yourDatasourceID=sharepoint563263
ipOfConnectorsClassicNode=192.168.1.78 portOfConnectorsClassic=8984
curl -X PUT -u USERNAME:PASSWORD "${proxyUrl}/api/apollo/connectors-
classic/datasources/${yourDatasourceID}/node_assignment/http%3A%2F%2F$
{ipOfConnectorsClassicNode}%3A${portOfConnectorsClassic}%2Fconnectors%
2Fv1%2Fconnectors-classic"
```

Powershell:

```
$proxyUrl="https://FUSION_HOST:8764" $yourDatasourceID="yourdsid"
$username = "admin" $password = "password" $base64AuthInfo =
[Convert]::ToBase64String([Text.Encoding]::ASCII.GetBytes(("{0}:{1}" -
f $username,$password))) $ipOfConnectorsClassicNode="192.168.1.78"
$portOfConnectorsClassic="8984" Invoke-RestMethod -Method Put -Headers
@{Authorization=("Basic {0}" -f $base64AuthInfo)}
"${proxyUrl}/api/apollo/connectors-
classic/datasources/${yourDatasourceID}/node_assignment/http%3A%2F%2F$
{ipOfConnectorsClassicNode}%3A${portOfConnectorsClassic}%2Fconnectors%
2Fv1%2Fconnectors-classic"
```

Clear connectors job assignment

Clear the job assignment from a connectors data source job.

Method: DELETE

`/api/apollo/connectors-classic/datasources/${yourDatasourceID}/node_assignment`

Bash:

```
proxyUrl=https://FUSION_HOST:8764 yourDatasourceID=sharepoint563263
curl -X DELETE -u USERNAME:PASSWORD
'${proxyUrl}/api/apollo/connectors-
classic/datasources/${yourDatasourceID}/node_assignment'
```

Powershell:

```
$proxyUrl="https://FUSION_HOST:8764" $yourDatasourceID="yourdsid"
$username = "admin" $password = "password" $base64AuthInfo =
[Convert]::ToBase64String([Text.Encoding]::ASCII.GetBytes(("{0}:{1}" -
f $username,$password))) Invoke-RestMethod -Method Delete -Headers
@{Authorization=("Basic {0}" -f $base64AuthInfo)}
"${proxyUrl}/api/apollo/connectors-
classic/datasources/${yourDatasourceID}/node_assignment"
```

Automated Solr Backups in Fusion 5

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion*

You can schedule backups of Solr collections, store the backups for a configurable period of time, and restore these backups into a specified Fusion cluster when needed.

This guide makes the following assumptions:

- You are using Google Kubernetes Engine (GKE).
- You used the setup scripts in the fusion-cloud-native repository to install Fusion.

Implementation

Backups are taken using the Solr collection BACKUP command. This requires that each Solr node has access to a shared volume or a `ReadWriteMany` volume in Kubernetes. Most cloud providers provide a simple way of creating a shared filestore and exposing it as a `PersistentVolumeClaim` within Kubernetes to mount into the Solr pods. An option is added into the **setup_f5_.sh** scripts in the **fusion-cloud-native** repository to provision these.

The backup action of the script is invoked by a Kubernetes CronJob to configure the backup schedule. The backups are saved to a configurable directory with an automatically generated name: `<collection_name>-<timestamp_in_some_format>`.

A separate CronJob is responsible for cleanup and retention of backups, letting cleanup be disabled. The retention periods allows to be less frequent backups as they become older. For example, a cluster that backs up a collection every 3 hours could specify a retention policy that:

- keeps all backups for a single day.
- keeps a single backup a day for a week.
- keeps a single backup a week for a month.
- keeps a single backup a month for 6 months.
- deletes all backups that are older than this time.

All times are configurable as part of the `configmap` for this service.

The process for restoring a collection is a manual step involving `kubectl run`. This action invokes the Solr `RESTORE` action pointing to the collection and name of the backup that should be restored.

Note

These instructions are for GKE only. For other platforms, backup and restoration involves copying the collection to the cloud and using *Parallel Bulk Loader*.

Installation

The solr-backup-runner requires that a `ReadWriteMany` volume is mounted onto all the Solr pods and the backup-runner pods so all pods back up to a consistent filesystem.

GKE example

The easiest way to install in GKE is by using a GCP Filestore volume as the ReadWriteMany volume.

1. Create the filestore.

```
gcloud --project "${GCLOUD_PROJECT}" filestore instances create
"${NFS_NAME}" --tier=STANDARD --file-
share=name="solrbackups,capacity=${SOLR_BACKUP_NFS_GB}GB" --
zone="${GCLOUD_ZONE}" --network=name="${network_name}"
```

2. Fetch the IP of the filestore when it has created.

```
NFS_IP="$(gcloud filestore instances describe "${NFS_NAME}" --
project="${GCLOUD_PROJECT}" --zone="${GCLOUD_ZONE}" --
format="value(networks.ipAddresses[0])")"
```

3. Create a Persistent Volume in kube that is backed by this volume.

```
cat <<EOF | kubectl -n "${NAMESPACE}" apply -f - apiVersion: v1
kind: PersistentVolume metadata: name: ${NAMESPACE}-solr-backups
annotations: pv.beta.Kubernetes.io/gid: "8983" spec: capacity:
storage: ${SOLR_BACKUP_NFS_GB}G accessModes: - ReadWriteMany
nfs: path: /solrbackups server: ${NFS_IP} EOF
```

4. Create a Persistent Volume Claim in the namespace that Solr is running in.

```
cat <<EOF | kubectl -n "${NAMESPACE}" apply -f - apiVersion: v1
kind: PersistentVolumeClaim metadata: name: fusion-solr-backup-
claim spec: volumeName: ${NAMESPACE}-solr-backups accessModes: -
ReadWriteMany storageClassName: "" resources: requests: storage:
${SOLR_BACKUP_NFS_GB}G EOF
```

5. Add the following values to your existing (or a new) helm values file.

```
solr-backup-runner: enabled: true sharedPersistentVolumeName:
fusion-solr-backup-claim solr: additionalInitContainers: - name:
chown-backup-directory securityContext: runAsUser: 0 image:
busybox:latest command: ['/bin/sh', '-c', "owner=$(stat -c '%u'
/mnt/solr-backups);  if [ ! \"${owner}\" = \"8983\" ]; then
chown -R 8983:8983 /mnt/solr-backups; fi "] volumeMounts: -
mountPath: /mnt/solr-backups name: solr-backups
additionalVolumes: - name: solr-backups persistentVolumeClaim:
claimName: fusion-solr-backup-claim additionalVolumeMounts: -
name: solr-backups mountPath: "/mnt/solr-backups"
```

6. Upgrade the release and Solr backups is enabled.

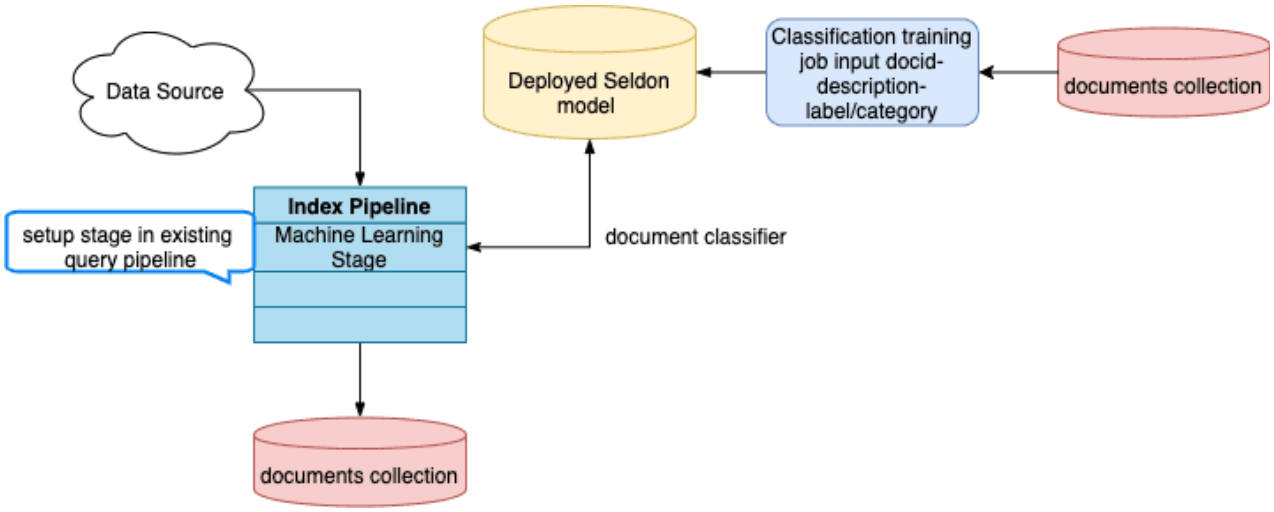
Automatically classify new documents at index time

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

You can predict the categories of new documents at index time by using *Classification* to analyze previously-classified documents in your index and produce a training model, then referencing the model in *Machine learning index stage*.

Classification job dataflow (documents)



How to configure new document classification

1. Sign in to Managed Fusion and click your application.
2. Click **Collections > Jobs > Add+ > Classification** to create a new Classification job.
3. In the **Model Deployment Name** field, enter an ID for the new classification model.
4. In the **Training Data Path** field, enter the collection name or cloud storage path where your main content is stored.
5. In the **Training Data Format** field, leave the default `solr` value if the **Training Data Path** is a collection. Otherwise, specify the format of your data in cloud storage.
6. In the **Training collection content field**, enter the name of the field that contains the content to analyze. The content field you choose depends on your use case and the typical user query types.

   For example, you could choose the description field if users tend to make descriptive queries like "4k TV" or "soft waterproof jacket". But if users are more likely to search for specific brands or products, such as "LG TV" or "North Face jacket", then the product name field might be more suitable.
7. In the **Training collection class field**, enter the name of the field that contains the category data.

   Tip

   For additional configuration details, see *Best practices* below.
8. Save the job.
9. Specify the model's name in the *Machine learning stage* of your index pipeline.
10. In the **Model input transformation script** field, enter the following:

```
/* Name of the document field to feed into the model. */ var
documentFeatureField = "body_t" /* Model input construction. */
var modelInput = new java.util.HashMap() modelInput.put("text",
doc.getFirstFieldValue(documentFeatureField)) modelInput
```

11. In the **Model output transformation script** field, enter the following:

```
// In case if top_k_predictions are needed var top1ClassField =
"top_1_class_s" var top1ScoreField = "top_1_score_d" var
topKClassesField = "top_k_classes_ss" var topKScoresField =
"top_k_scores_ds" var jsonOutput =
JSON.parse(modelOutput.get("_rawJsonResponse")) var parsedOutput
= {}; for (var i=0; i<jsonOutput["names"].length;i++){
parsedOutput[jsonOutput["names"][i]] = jsonOutput["ndarray"][i]
} doc.addField(top1ClassField, parsedOutput["top_1_class"][0])
doc.addField(top1ScoreField, parsedOutput["top_1_score"][0]) if
("top_k_classes" in parsedOutput) {
doc.addField(topKClassesField, new
java.util.ArrayList(parsedOutput["top_k_classes"][0]))
doc.addField(topKScoresField, new
java.util.ArrayList(parsedOutput["top_k_scores"][0])) }
```

12. Click **Apply**.
13. Save the query pipeline.

Custom output transformation script example

```
var top1ClassField = "top_1_class_s" var top1ScoreField =
"top_1_score_d" doc.addField(top1ClassField,
modelOutput.get("top_1_class")[0]) doc.addField(top1ScoreField,
modelOutput.get("top_1_score")[0])
```

Best practices for configuring the Classification job

This job analyzes how your existing documents are categorized and produces a classification model that can be used to predict the categories of new documents at index time.

In addition to the information in this topic, see *Classify new queries* for configuration and examples.

This job takes raw text and an associated single class as input. Although it trains on single classes, there is an option to predict the top several classes with their scores.

At a minimum, you must configure these:

- An ID for this job
- A **Method**; Logistic Regression is the default

- A **Model Deployment Name**
- The **Training Collection**
- The **Training collection content field**, the document field containing the raw text
- The **Training collection class field** containing the classes, labels, or other category data for the text

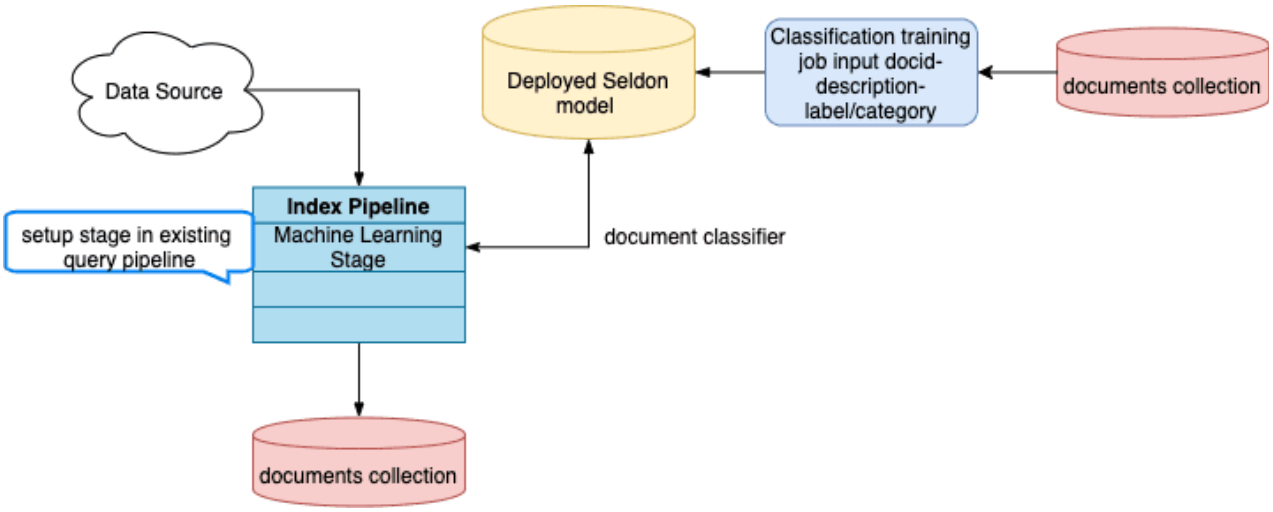
Automatically Classify New Documents at Index Time

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion*

You can predict the categories of new documents at index time by using the *Classification job* to analyze previously-classified documents in your index and produce a training model, then referencing the model in the *Machine Learning index stage*.

Classification job dataflow (documents)



How to configure new document classification

1. Navigate to **Collections** > **Jobs** > **Add+** > **Classification** to create a new Classification job.
2. Configure the job as follows:
   a. In the **Model Deployment Name** field, enter an ID for the new classification model.
   b. In the **Training Data Path** field, enter the collection name or cloud storage path where your main content is stored.
   c. In the **Training Data Format** field, leave the default `solr` value if the **Training Data Path** is a collection. Otherwise, specify the format of your data in cloud storage.
   d. In the **Training collection content field**, enter the name of the field that contains the content to analyze.
   The content field that you choose depends on your use case and the types of queries that your users commonly make.
   For example, you could choose the description field if users tend to make descriptive queries like "4k TV" or "soft waterproof jacket". But if users are more likely to search for specific brands or products, such as "LG TV" or "North Face jacket", then the product name field might be more suitable.
   e. In the **Training collection class field**, enter the name of the field that contains the category data.

> **Tip**
>
> For additional configuration details, see *Best practices* below.

3. Save the job.
4. Specify the model's name in the *Machine Learning stage* of your index pipeline.
5. In the **Model input transformation script** field, enter the following:

```
/* Name of the document field to feed into the model. */ var
documentFeatureField = "body_t" /* Model input construction. */
var modelInput = new java.util.HashMap() modelInput.put("text",
doc.getFirstFieldValue(documentFeatureField)) modelInput
```

6. In the **Model output transformation script** field, enter the following:

```
// In case if top_k_predictions are needed var top1ClassField =
"top_1_class_s" var top1ScoreField = "top_1_score_d" var
topKClassesField = "top_k_classes_ss" var topKScoresField =
"top_k_scores_ds" var jsonOutput =
JSON.parse(modelOutput.get("_rawJsonResponse")) var parsedOutput
= {}; for (var i=0; i<jsonOutput["names"].length;i++){
parsedOutput[jsonOutput["names"][i]] = jsonOutput["ndarray"][i]
} doc.addField(top1ClassField, parsedOutput["top_1_class"][0])
doc.addField(top1ScoreField, parsedOutput["top_1_score"][0]) if
("top_k_classes" in parsedOutput) {
doc.addField(topKClassesField, new
java.util.ArrayList(parsedOutput["top_k_classes"][0]))
doc.addField(topKScoresField, new
java.util.ArrayList(parsedOutput["top_k_scores"][0])) }
```

   a. Click **Apply**.

7. Save the query pipeline.

Custom output transformation script example

```
var top1ClassField = "top_1_class_s" var top1ScoreField =
"top_1_score_d" doc.addField(top1ClassField,
modelOutput.get("top_1_class")[0]) doc.addField(top1ScoreField,
modelOutput.get("top_1_score")[0])
```

Best practices for configuring the Classification job

This job analyzes how your existing documents are categorized and produces a classification model that can be used to predict the categories of new documents at index time.

In addition to the information in this topic, see *Automatically classify new queries* for configuration information and examples.

This job takes raw text and an associated single class as input. Although it trains on single classes, there is an option to predict the top several classes with their scores.

At a minimum, you must configure these:

- An ID for this job
- A **Method**; Logistic Regression is the default
- A **Model Deployment Name**
- The **Training Collection**
- The **Training collection content field**, the document field containing the raw text
- The **Training collection class field** containing the classes, labels, or other category data for the text

Automatically classify new queries

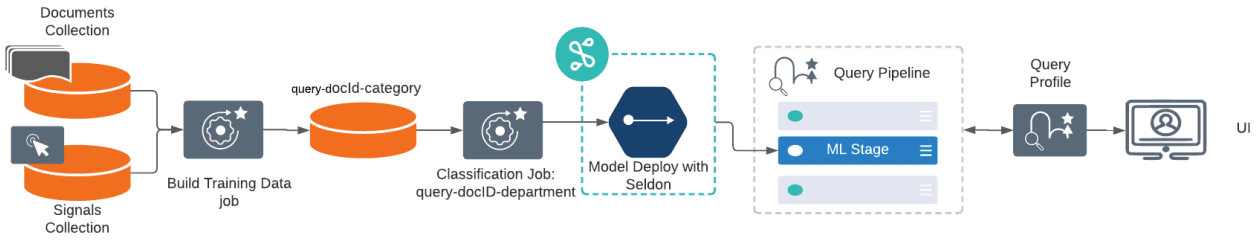
How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

You can predict the categories most likely to satisfy a new query using this workflow:

1. Use the *Build Training Data job* to join your signals data with your catalog data and produce training data in the form of query/class pairs.
2. Use the *Classification job* to train a classification model using the output collection of the Build Training Data job as the training collection.

Query-time classification workflow



See the detailed steps below.

To predict the categories of new queries

1. Navigate to **Collections** > **Jobs** > **Add+** > **Build Training Data** to create a new Build Training Data job.
2. Configure the job as follows:
    a. In the **Catalog Path** field, enter the collection name or cloud storage path where your main content is stored.
    b. In the **Catalog Format** field, enter `solr` if you are analyzing a Solr collection, or another format if your content is in the cloud.
    c. In the **Signals Path** field, enter the collection name or cloud storage path where your signals data is stored.
    d. In the **Output Path** field, enter the collection name or cloud storage path where you want to store the training data.
    e. In the **Category Field in Catalog** field, enter the field name for the category data in your main content.
    f. In the **Item ID Field in Catalog** field, enter the field name for the item IDs in your main content.
    g. Check that the values of **Item ID Field in Signals** and **Count Field in Signals** match the field names in your signals data.
3. Save the job.
4. Click **Run** > **Start** to run the job.
5. Navigate to **Collections** > **Jobs** > **Add+** > **Classification** to create a new Classification job.
6. Configure the job as follows:
    a. In the **Model Deployment Name** field, enter an ID for the new classification model.
    b. In the **Training Data Path** field, enter the collection name or cloud storage path from the Build Training Data job's **Output Path** field.
    c. In the **Training Data Format** field, leave the default `solr` value if the **Training Data Path** is a collection or if you used the default format in your Build Training Data job configuration.
    If you configured the Build Training Data job to output a different format, enter it here.
    d. In the **Training collection content field**, enter `query_s`, the default content field name in the Build Training Data job's output.
    e. In the **Training collection class field**, enter `category_s`, the default category field name in the Build Training Data job's output.

    Tip

    For additional configuration details, see *Best practices* below.

7. Save the job.
8. Verify that the Build Training Data job has finished successfully.
9. Click **Run** > **Start** to run the job.
10. Navigate to **Indexing** > **Query Workbench** > **Load** and select your query pipeline.
11. Configure the query pipeline as follows:

a. Add a new Machine Learning stage.
b. In the **Model ID** field, enter the name from the Classification job's **Model Deployment Name** field.
c. In the **Model input transformation script** field, enter the following:

```
var modelInput = new java.util.HashMap()
modelInput.put("text", request.getFirstParam("q"))
modelInput
```

d. In the **Model output transformation script** field, enter the following:

```
// In case if top_k_predictions are needed // To put into
response documents (can be done only after Solr Query
stage) var jsonOutput =
JSON.parse(modelOutput.get("_rawJsonResponse")) var
parsedOutput = {}; for (var i=0;
i<jsonOutput["names"].length;i++){
parsedOutput[jsonOutput["names"][i]] =
jsonOutput["ndarray"][i] } var docs =
response.get().getInnerResponse().getDocuments(); var
ndocs = new java.util.ArrayList(); for (var i=0;
i<docs.length;i++){ var doc = docs[i];
doc.putField("top_1_class", parsedOutput["top_1_class"]
[0]) doc.putField("top_1_score",
parsedOutput["top_1_score"][0]) if ("top_k_classes" in
parsedOutput) { doc.putField("top_k_classes", new
java.util.ArrayList(parsedOutput["top_k_classes"][0]))
doc.putField("top_k_scores", new
java.util.ArrayList(parsedOutput["top_k_scores"][0])) }
ndocs.add(doc); }
response.get().getInnerResponse().updateDocuments(ndocs);
```

e. Click **Apply**.

1. Save the query pipeline.

Custom output transformation script examples

```
// To put into request request.putSingleParam("class",
modelOutput.get("top_1_class")[0]) request.putSingleParam("score",
modelOutput.get("top_1_score")[0]) // Or for example to apply Filter
Query request.putSingleParam("fq", "class:" +
modelOutput.get("top_1_class")[0])
```

```
// To put into query context context.put("class",
modelOutput.get("top_1_class")[0]) context.put("score",
modelOutput.get("top_1_score")[0])
```

```
// To put into response documents (can be done only after Solr Query
stage) var docs = response.get().getInnerResponse().getDocuments();
var ndocs = new java.util.ArrayList(); for (var i=0;
i<docs.length;i++){ var doc = docs[i]; doc.putField("query_class",
modelOutput.get("top_1_class")[0]) doc.putField("query_score",
modelOutput.get("top_1_score")[0]) ndocs.add(doc); }
response.get().getInnerResponse().updateDocuments(ndocs);
```

Best practices for configuring the Classification job

This job analyzes how your existing documents are categorized and produces a classification model that can be used to predict the categories of new documents at index time.

In addition to the information in this topic, see *Classify new documents at index time* for configuration and examples.

This job takes raw text and an associated single class as input. Although it trains on single classes, there is an option to predict the top several classes with their scores.

At a minimum, you must configure these:

- An ID for this job
- A **Method**; Logistic Regression is the default
- A **Model Deployment Name**
- The **Training Collection**
- The **Training collection content field**, the document field containing the raw text

- The **Training collection class field** containing the classes, labels, or other category data for the text

Automatically Classify New Queries

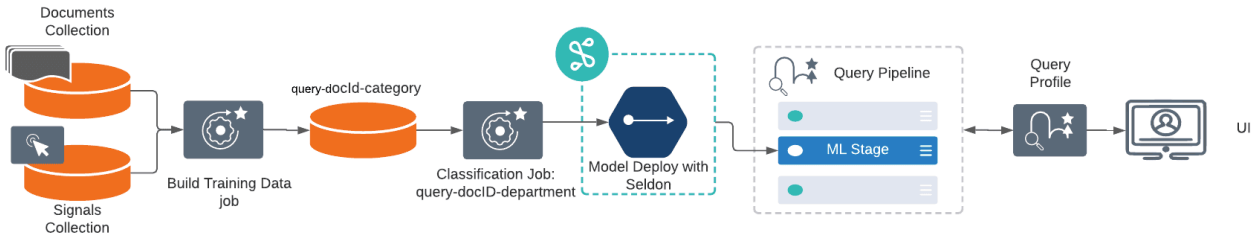
How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion*

You can predict the categories most likely to satisfy a new query using this workflow:

1. Use the *Build Training Data job* to join your signals data with your catalog data and produce training data in the form of query/class pairs.
2. Use the *Classification job* to train a classification model using the output collection of the Build Training Data job as the training collection.

Query-time classification workflow



See the detailed steps below.

To predict the categories of new queries

1. Navigate to **Collections** > **Jobs** > **Add+** > **Build Training Data** to create a new Build Training Data job.
2. Configure the job as follows:
    a. In the **Catalog Path** field, enter the collection name or cloud storage path where your main content is stored.
    b. In the **Catalog Format** field, enter `solr` if you are analyzing a Solr collection, or another format if your content is in the cloud.
    c. In the **Signals Path** field, enter the collection name or cloud storage path where your signals data is stored.
    d. In the **Output Path** field, enter the collection name or cloud storage path where you want to store the training data.
    e. In the **Category Field in Catalog** field, enter the field name for the category data in your main content.
    f. In the **Item ID Field in Catalog** field, enter the field name for the item IDs in your main content.
    g. Check that the values of **Item ID Field in Signals** and **Count Field in Signals** match the field names in your signals data.
3. Save the job.
4. Click **Run** > **Start** to run the job.
5. Navigate to **Collections** > **Jobs** > **Add+** > **Classification** to create a new Classification job.
6. Configure the job as follows:
    a. In the **Model Deployment Name** field, enter an ID for the new classification model.
    b. In the **Training Data Path** field, enter the collection name or cloud storage path from the Build Training Data job's **Output Path** field.
    c. In the **Training Data Format** field, leave the default `solr` value if the **Training Data Path** is a collection or if you used the default format in your Build Training Data job configuration.
    If you configured the Build Training Data job to output a different format, enter it here.
    d. In the **Training collection content field**, enter `query_s`, the default content field name in the Build Training Data job's output.
    e. In the **Training collection class field**, enter `category_s`, the default category field name in the Build Training Data job's output.

    Tip

    For additional configuration details, see *Best practices* below.

7. Save the job.
8. Verify that the Build Training Data job has finished successfully.
9. Click **Run** > **Start** to run the job.
10. Navigate to **Indexing** > **Query Workbench** > **Load** and select your query pipeline.
11. Configure the query pipeline as follows:

a. Add a new Machine Learning stage.
b. In the **Model ID** field, enter the name from the Classification job's **Model Deployment Name** field.
c. In the **Model input transformation script** field, enter the following:

```
var modelInput = new java.util.HashMap()
modelInput.put("text", request.getFirstParam("q"))
modelInput
```

d. In the **Model output transformation script** field, enter the following:

```
// In case if top_k_predictions are needed // To put into
response documents (can be done only after Solr Query
stage) var jsonOutput =
JSON.parse(modelOutput.get("_rawJsonResponse")) var
parsedOutput = {}; for (var i=0;
i<jsonOutput["names"].length;i++){
parsedOutput[jsonOutput["names"][i]] =
jsonOutput["ndarray"][i] } var docs =
response.get().getInnerResponse().getDocuments(); var
ndocs = new java.util.ArrayList(); for (var i=0;
i<docs.length;i++){ var doc = docs[i];
doc.putField("top_1_class", parsedOutput["top_1_class"]
[0]) doc.putField("top_1_score",
parsedOutput["top_1_score"][0]) if ("top_k_classes" in
parsedOutput) { doc.putField("top_k_classes", new
java.util.ArrayList(parsedOutput["top_k_classes"][0]))
doc.putField("top_k_scores", new
java.util.ArrayList(parsedOutput["top_k_scores"][0])) }
ndocs.add(doc); }
response.get().getInnerResponse().updateDocuments(ndocs);
```

e. Click **Apply**.

1. Save the query pipeline.

Custom output transformation script examples

```
// To put into request request.putSingleParam("class",
modelOutput.get("top_1_class")[0]) request.putSingleParam("score",
modelOutput.get("top_1_score")[0]) // Or for example to apply Filter
Query request.putSingleParam("fq", "class:" +
modelOutput.get("top_1_class")[0])
```

```
// To put into query context context.put("class",
modelOutput.get("top_1_class")[0]) context.put("score",
modelOutput.get("top_1_score")[0])
```

```
// To put into response documents (can be done only after Solr Query
stage) var docs = response.get().getInnerResponse().getDocuments();
var ndocs = new java.util.ArrayList(); for (var i=0;
i<docs.length;i++){ var doc = docs[i]; doc.putField("query_class",
modelOutput.get("top_1_class")[0]) doc.putField("query_score",
modelOutput.get("top_1_score")[0]) ndocs.add(doc); }
response.get().getInnerResponse().updateDocuments(ndocs);
```

Best practices for configuring the Classification job

This job analyzes how your existing documents are categorized and produces a classification model that can be used to predict the categories of new documents at index time.

In addition to the information in this topic, see *Automatically classify new documents at index time* for configuration information and examples.

This job takes raw text and an associated single class as input. Although it trains on single classes, there is an option to predict the top several classes with their scores.

At a minimum, you must configure these:

- An ID for this job
- A **Method**; Logistic Regression is the default
- A **Model Deployment Name**
- The **Training Collection**
- The **Training collection content field**, the document field containing the raw text

- The **Training collection class field** containing the classes, labels, or other category data for the text

Avoid SharePoint Throttling

How-to

> Compatible versions: 4.2, 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion Connectors*, *Fusion Server*, *Fusion*

When using a SharePoint connector to crawl SharePoint Online, rate limiting can be an issue. Learn more about throttling in SharePoint Online.

You have a few options to avoid throttling:

- *Decrease the number of threads.*
- *Stagger the datasource job schedules.*
- *Increase the number of retries.*

Decrease the number of threads

If you see many `429` / `503` errors, you are probably hitting SharePoint Online with too many concurrent fetchers.

How to decrease the number of threads

1. Set **Crawl Performance** > **Fetch Threads** to a lower value.
2. Set **Crawl Performance** > **Prefetch Threads** to a lower value.

Stagger the datasource jobs

If you have multiple SharePoint Online datasource jobs that run at the same time, use the job scheduler to stagger their schedules instead.

- *Fusion 4.x Scheduler*
- *Fusion 5.x Scheduler*

Increase the number of retries

By default, the connector is configured with retries. This provides a chance for the requests that were rate-limited to run again.

You can increase the number of retries and the interval between retries. The process is called exponential backoff, which gradually increases the delays between retries to increase the chances of a successful retry. This helps prevent missing documents due to rate limiting.

For SharePoint Online V1 Optimized, retry configuration parameters include:

- Retry attempts
- Retry maximum wait
- Retryer backoff delay (milliseconds)
- Retryer backoff max delay (milliseconds)
- Retryer backoff multiplier (decimal)

For SharePoint Optimized V2, retry configuration parameters include:

- Retry Delay
- Maximum Retries
- Delay Factor
- Maximum Delay Time
- Maximum Time Limit

Important

When you are receiving too many rate limiting errors, it is likely too many requests are being sent too frequently. Retrying may not help. One option is to decrease your traffic instead. If you want to continue sending the maximum number of requests, configure the **Retryer backoff multiplier** so it gets larger after every retry. The crawler will slow significantly and allow SharePoint to relax the throttling.

Avoid SharePoint Throttling

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

When using a SharePoint connector to crawl SharePoint Online, rate limiting can be an issue. Learn more about throttling in SharePoint Online.

You have a few options to avoid throttling:

- *Decrease the number of threads.*
- *Stagger the datasource job schedules.*
- *Increase the number of retries.*

Decrease the number of threads

If you see many `429` / `503` errors, you are probably hitting SharePoint Online with too many concurrent fetchers.

How to decrease the number of threads

1. Set **Crawl Performance** > **Fetch Threads** to a lower value.
2. Set **Crawl Performance** > **Prefetch Threads** to a lower value.

Stagger the datasource jobs

If you have multiple SharePoint Online datasource jobs that run at the same time, use the *Scheduler* to stagger their schedules instead.

Increase the number of retries

By default, the connector is configured with retries. This provides a chance for the requests that were rate-limited to run again.

You can increase the number of retries and the interval between retries. The process is called exponential backoff, which gradually increases the delays between retries to increase the chances of a successful retry. This helps prevent missing documents due to rate limiting.

For SharePoint Optimized V2, retry configuration parameters include:

- Retry Delay
- Maximum Retries
- Delay Factor
- Maximum Delay Time
- Maximum Time Limit

Important

When you are receiving too many rate limiting errors, it is likely too many requests are being sent too frequently. Retrying may not help. One option is to decrease your traffic instead. If you want to continue sending the maximum number of requests, configure the **Retryer backoff multiplier** so it gets larger after every retry. The crawler will slow significantly and allow SharePoint to relax the throttling.

Block a document in Experience Optimizer

How-to

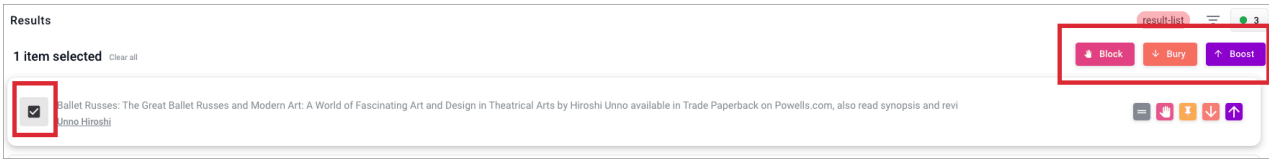
> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion*

You can block a document so that it does not appear in the search results when your customers run a particular query. This is helpful if a document has been removed from your catalog, or it is not yet available, for example.

When you block a document, a new rule is created automatically.

1. Hover your cursor over a document. A menu overlay will appear.
2. Hover your cursor over the **Actions** button. A dropdown menu will appear.
3. Click **Block**.

You can also block multiple products at once by clicking the checkbox in the top-left corner of the document tiles and selecting the **Block** button above the top row of products.



Block a Document in Experience Optimizer

How-to

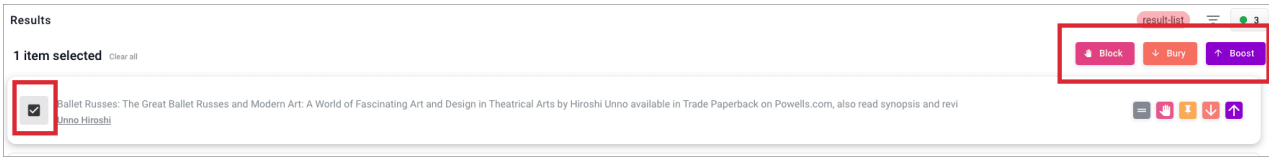
> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

You can block a document so that it does not appear in the search results when your customers run a particular query. This is helpful if a document has been removed from your catalog, or it is not yet available, for example.

When you block a document, a new rule is created automatically.

1. Hover your cursor over a document. A menu overlay will appear.
2. Hover your cursor over the **Actions** button. A dropdown menu will appear.
3. Click **Block**.

You can also block multiple products at once by clicking the checkbox in the top-left corner of the document tiles and selecting the **Block** button above the top row of products.



Block a product in Predictive Merchandiser

How-to

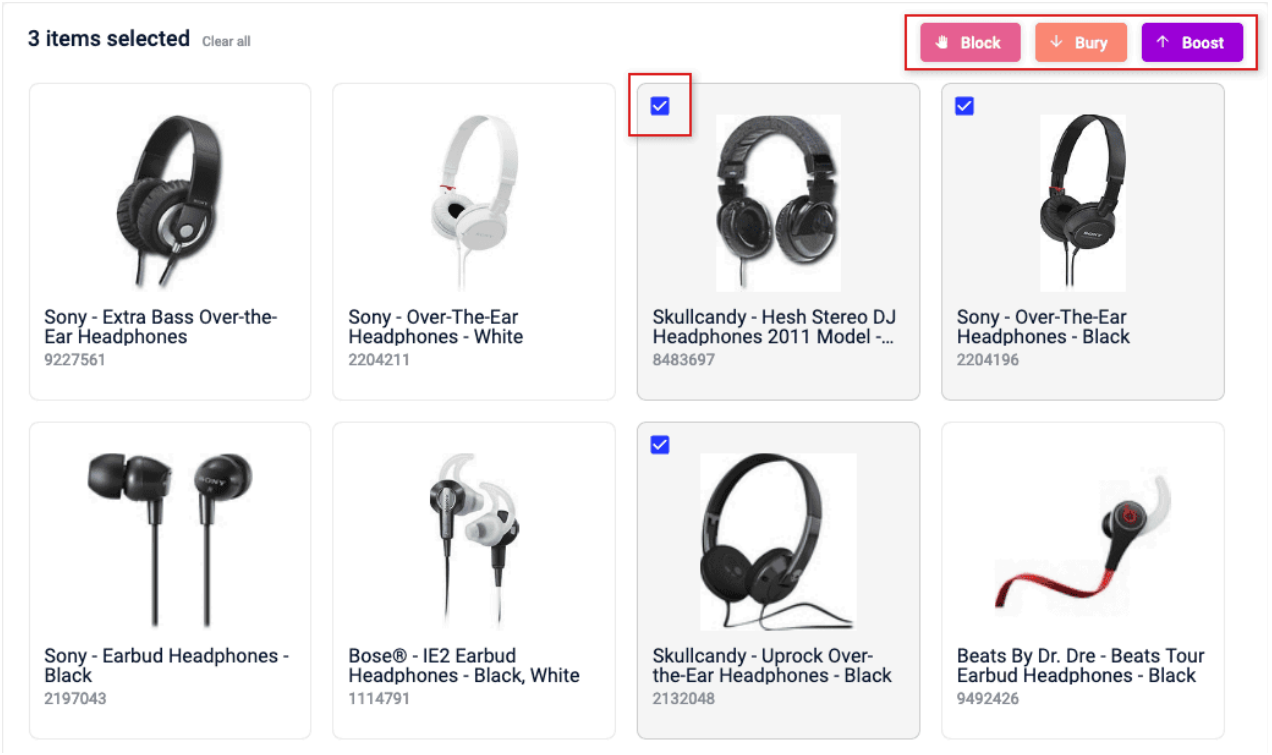
> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

You can block a product so that it does not appear in the search results when your customers run a particular query. This is helpful if a product has been removed from your catalog, or it is not yet available, for example.

When you block a product, a new rule is created automatically.

1. Hover your cursor over a product. A menu overlay appears.
2. Hover your cursor over the **Actions** button. A dropdown menu appears.
3. Click **Block Product**.

You can also block multiple products at once by clicking the checkbox in the top-left corner of the product tiles and selecting the **Block** button above the top row of products.



Block a Product in Predictive Merchandiser

How-to

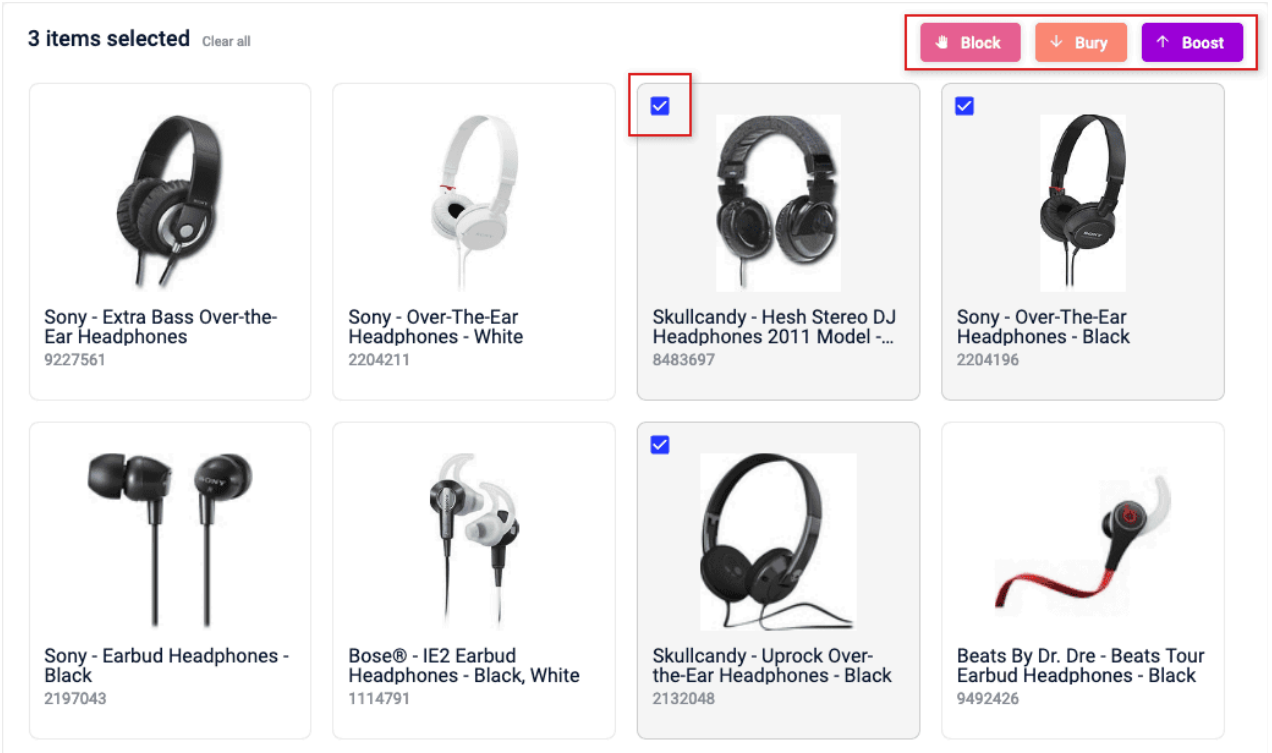
> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion*

You can block a product so that it does not appear in the search results when your customers run a particular query. This is helpful if a product has been removed from your catalog, or it is not yet available, for example.

When you block a product, a new rule is created automatically.

1. Hover your cursor over a product. A menu overlay will appear.
2. Hover your cursor over the **Actions** button. A dropdown menu will appear.
3. Select the **Block Product**.

You can also block multiple products at once by clicking the checkbox in the top-left corner of the product tiles and selecting the **Block** button above the top row of products.



Boost a document in Experience Optimizer

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion*

You can boost a document so that it occupies a higher position in the search results when your customers run a particular query. This can be helpful if you have new products in your catalog, or large stocks of some products, for example.

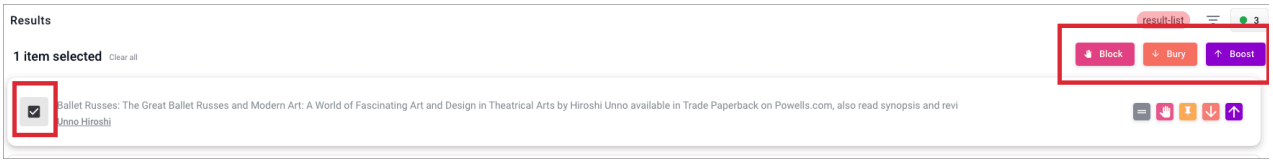
Important

Boosting a document increases the document's position in the results, but takes other factors into consideration while doing so. As a result, it does not guarantee the document will occupy a specific position in the list. Pinning a document, however, places the document at a specified position regardless of other factors.

When you boost a document, a new rule is created automatically.

1. Hover your cursor over a document. A menu overlay will appear.
2. Hover your cursor over the **Actions** button. A dropdown menu will appear.
3. Click **Boost**.

You can also boost multiple products at once by clicking the checkbox in the top-left corner of the product tiles and selecting the **Boost** button above the top row of products.



Boost a document in Experience Optimizer

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

You can boost a document to a higher position in the search results when your customers run a particular query. This can be helpful if, for example, you have new products in your catalog, or large stocks of some products.

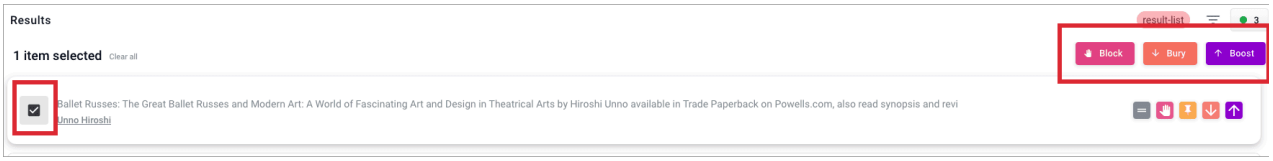
Important

Boosting a document increases the document's position in the results, but also takes other factors into consideration. As a result, boosting does not guarantee the document will occupy a specific position in the list. Pinning a document, however, places the document at a specified position regardless of other factors.

When you boost a document, a new rule is created automatically.

1. Hold the pointer over a document.
2. In the menu that displays, hold the pointer over the **Actions** button.
3. From the menu, click **Boost**.

You can also boost multiple products at once by clicking the checkbox of the product tiles and then clicking **Boost**.



Boost a product in Predictive Merchandiser

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

You can boost a product so that it occupies a higher position in the search results when your customers run a particular query. This can be helpful if you have new products in your catalog, or large stocks of some products, for example.

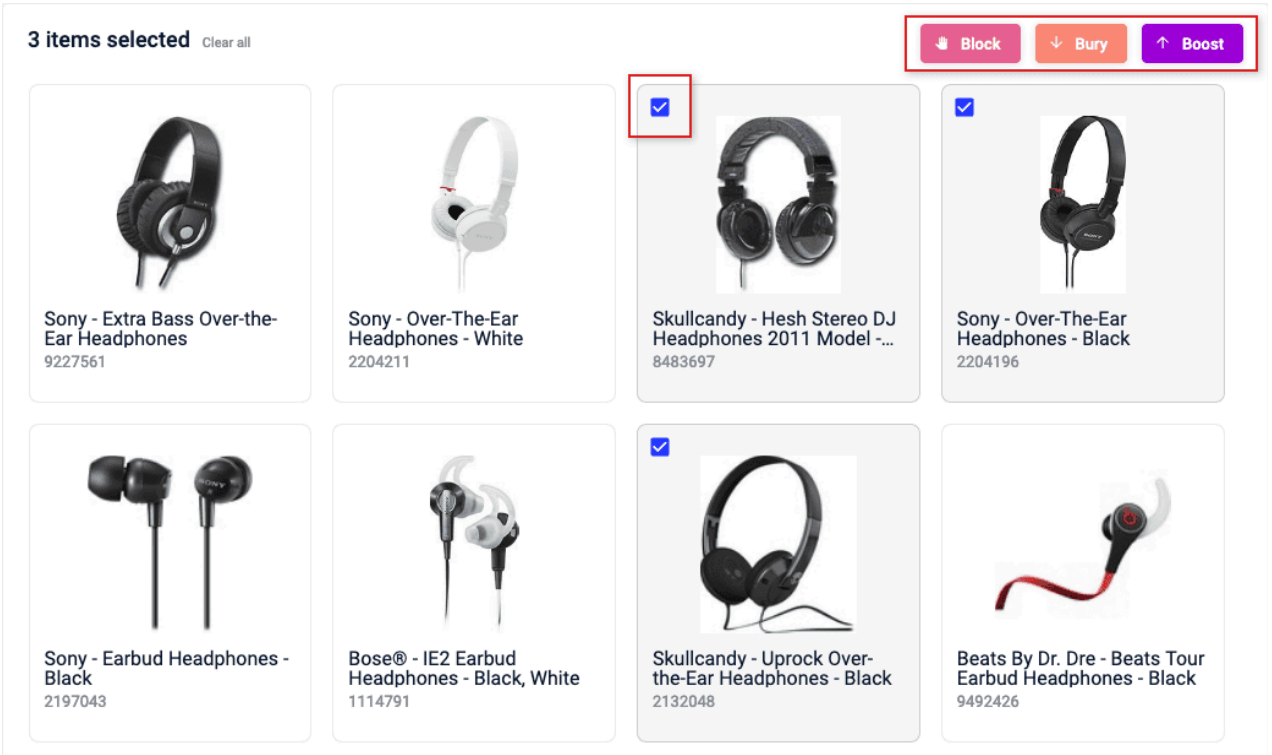
Important

Boosting a product increases the product's position in the results, but takes other factors into consideration while doing so. As a result, it does not guarantee the product will occupy a specific position in the list. Pinning a product, however, places the product at a specified position regardless of other factors.

When you boost a product, a new rule is created automatically.

1. Hover your cursor over a product. A menu overlay appears.
2. Hover your cursor over the **Actions** button. A dropdown menu appears.
3. Click **Boost**.

You can also boost multiple products at once by clicking the checkbox in the top-left corner of the product tiles and selecting the **Boost** button above the top row of products.



Boost a Product in Predictive Merchandiser

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion*

You can boost a product so that it occupies a higher position in the search results when your customers run a particular query. This can be helpful if you have new products in your catalog, or large stocks of some products, for example.

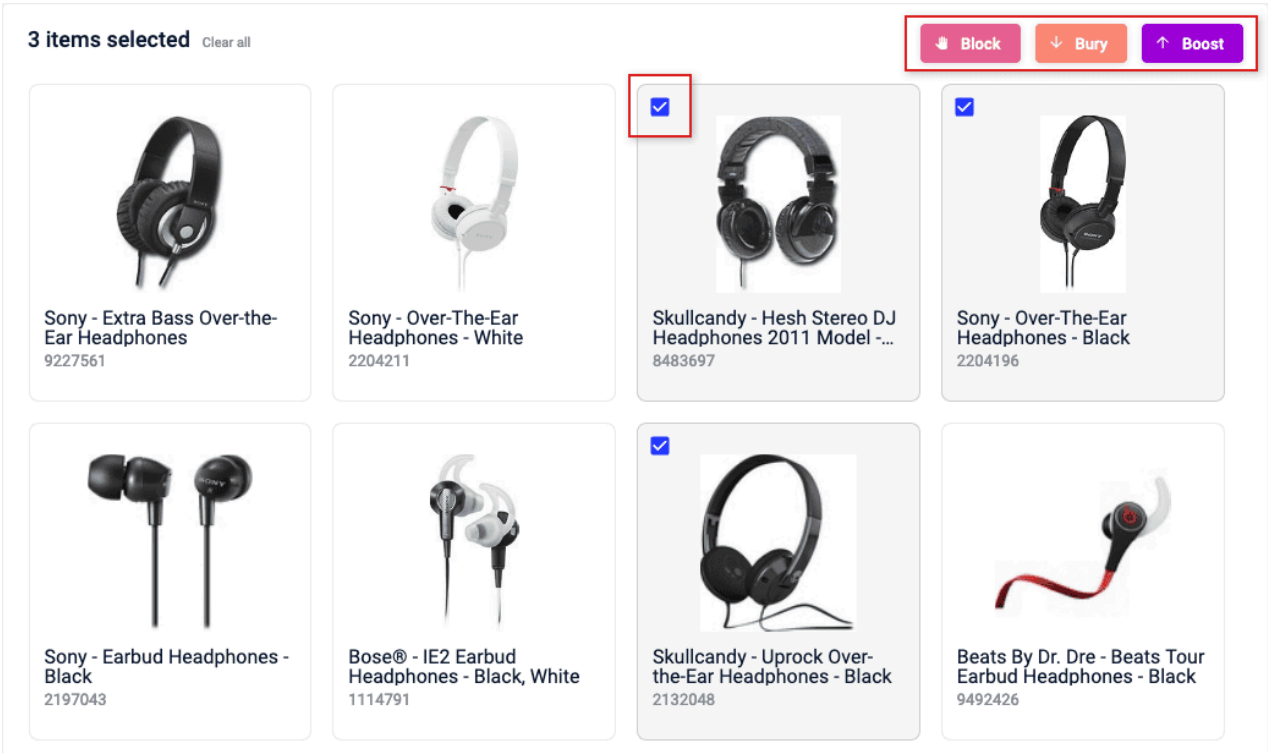
Important

Boosting a product increases the product's position in the results, but takes other factors into consideration while doing so. As a result, it does not guarantee the product will occupy a specific position in the list. Pinning a product, however, places the product at a specified position regardless of other factors.

When you boost a product, a new rule is created automatically.

1. Hover your cursor over a product. A menu overlay will appear.
2. Hover your cursor over the **Actions** button. A dropdown menu will appear.
3. Click **Boost**.

You can also boost multiple products at once by clicking the checkbox in the top-left corner of the product tiles and selecting the **Boost** button above the top row of products.



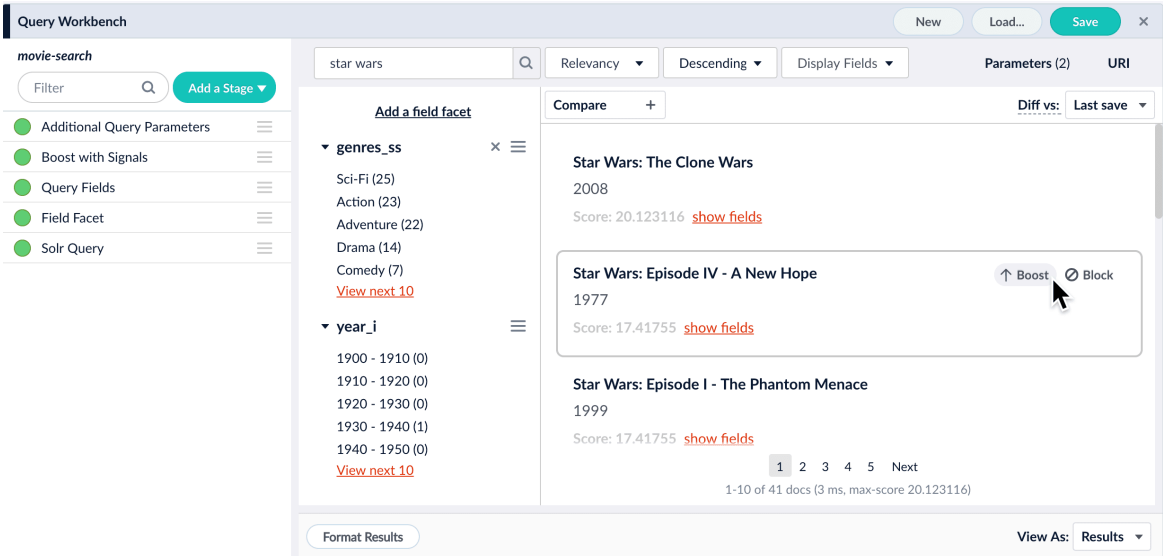
Boost or Block with the Query Workbench

How-to

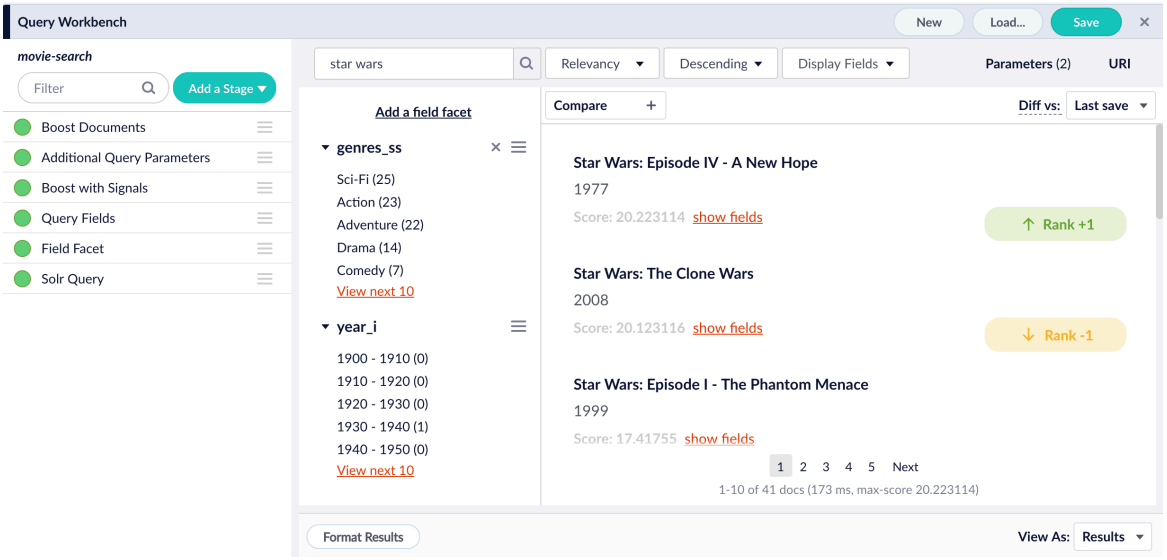
> Compatible versions: 4.2, 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion Server*, *Fusion*

How to boost a document in the Query Workbench

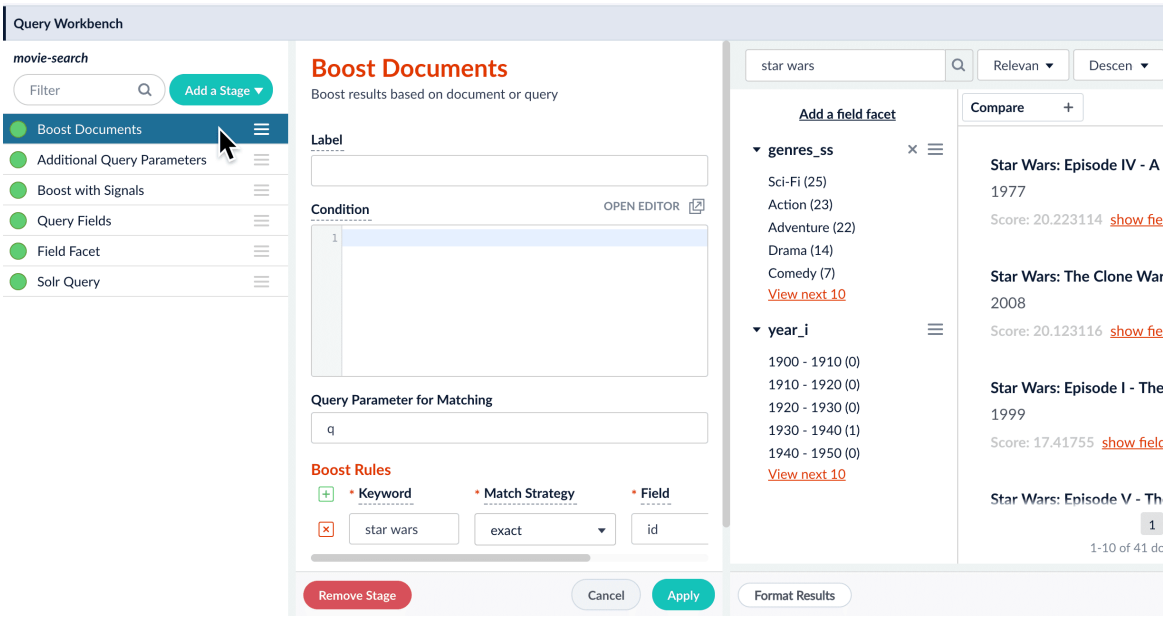
1. In the preview panel, hover over the document you want to boost.



2. Click **Boost**.
   The preview panel automatically updates the rankings of the search results, and tags the differences:



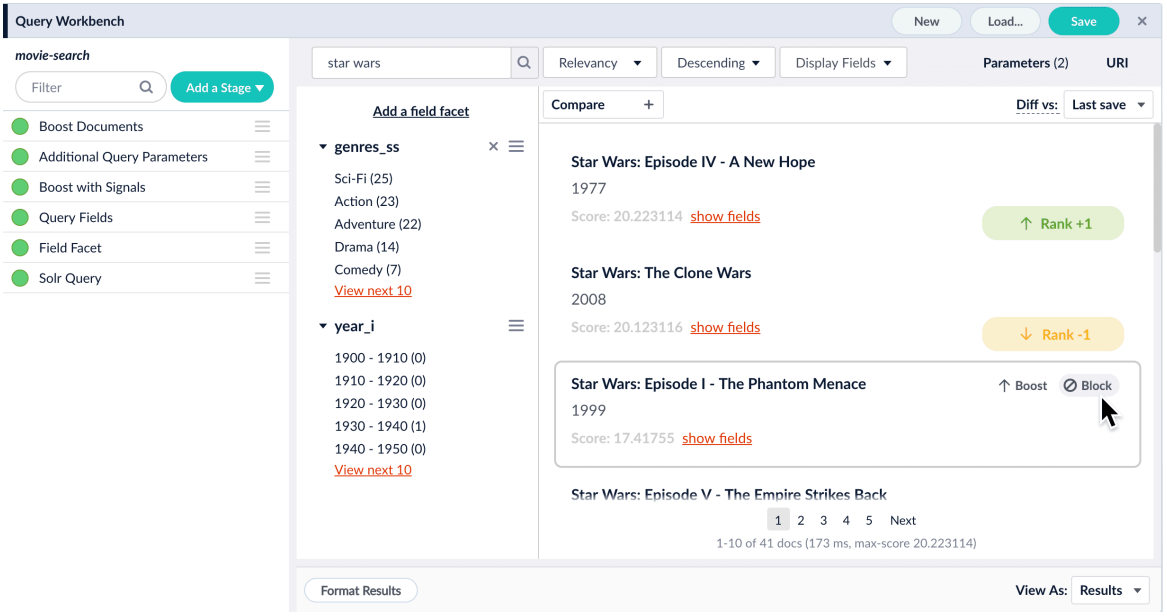
Notice that the Boost Documents stage is now in our pipeline. You can click this stage to view the boost rules you have added. This is also where you can remove boost rules:



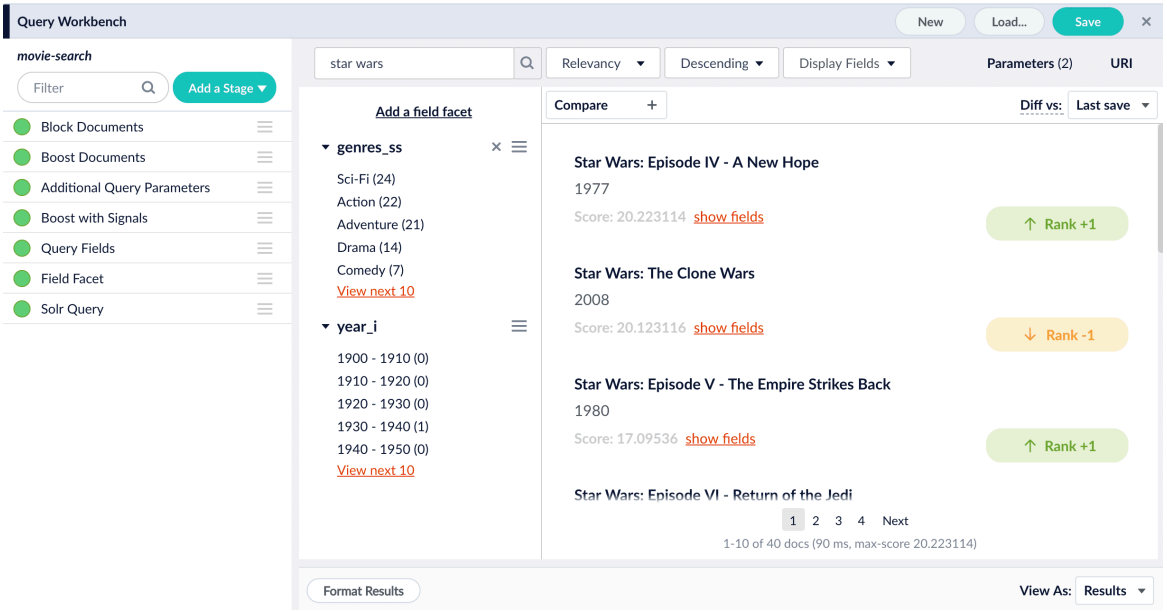
3. When you are satisfied with the results, save the query pipeline.

How to block a document in the Query Workbench

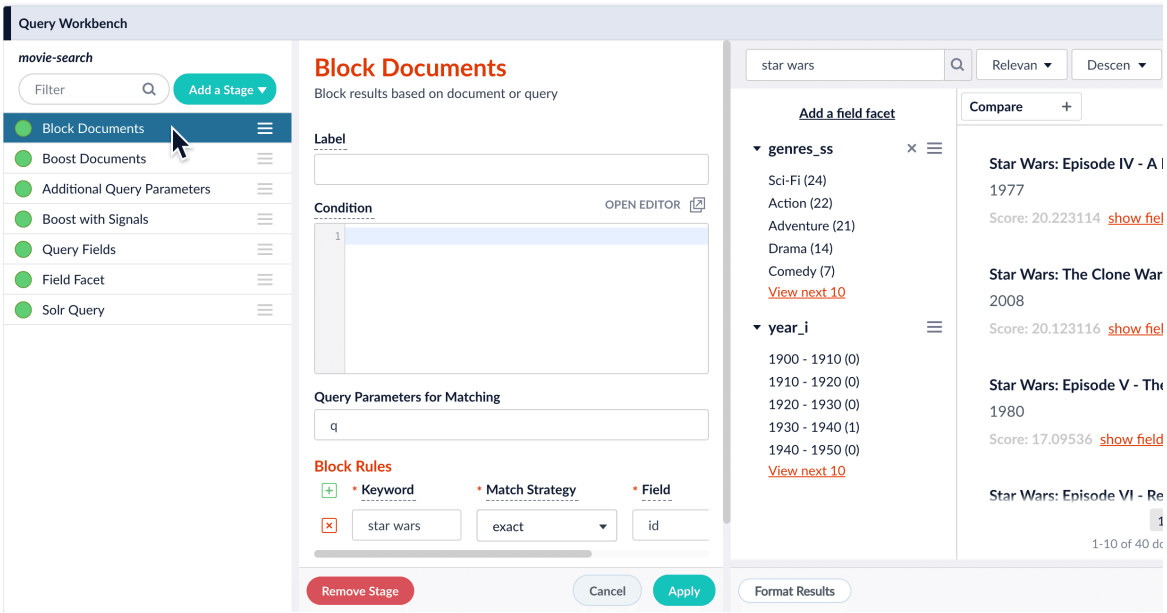
1. In the preview panel, hover over the document you want to block.



2. Click **Block**
   The preview panel automatically updates the rankings of the search results, and tags the differences.



Notice that the Block Documents stage is now in our pipeline. You can click this stage to view the block rules you have added. This is also where you can remove block rules:



3. When you are satisfied with the results, save the query pipeline.

See these pages for more information:

Fusion 5.x:

- *Boost Documents stage*
- *Block Documents stage*

Fusion 4.x:

- *Boost Documents stage*
- *Block Documents stage*

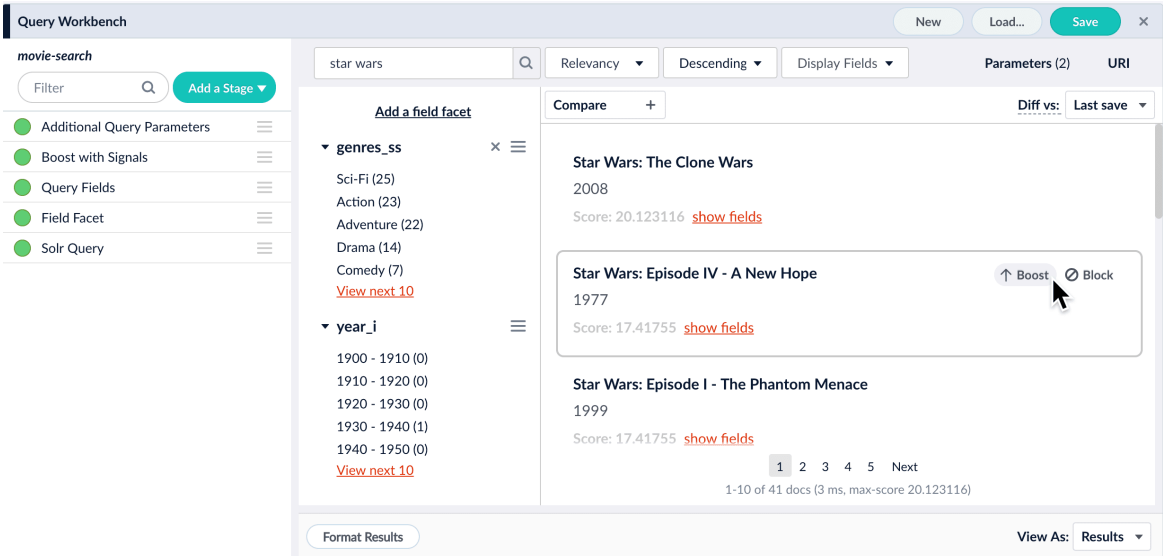
Boost or Block with the Query Workbench

How-to

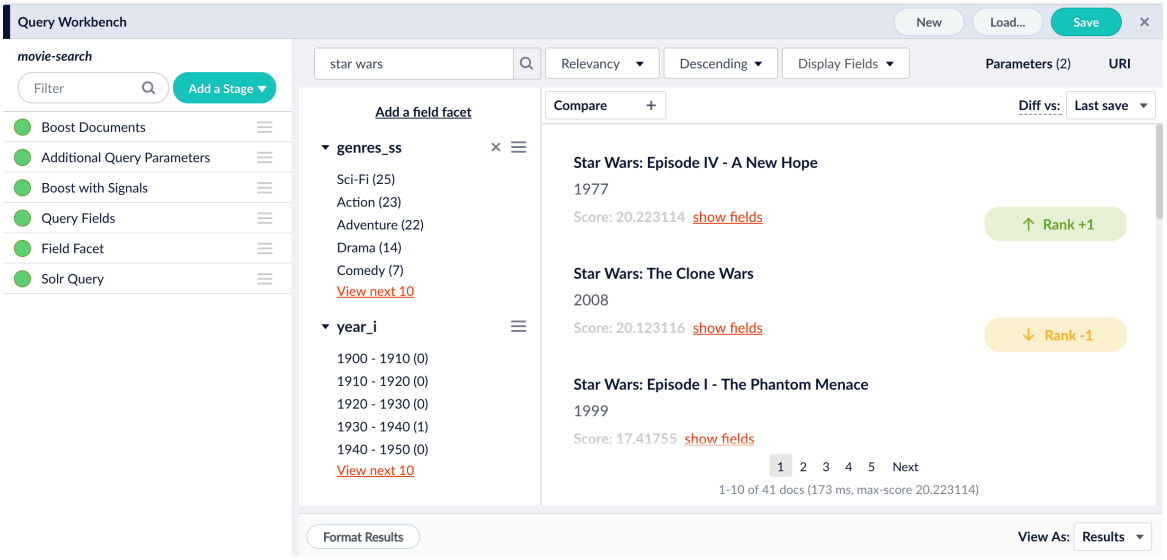
> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

How to boost a document in the Query Workbench

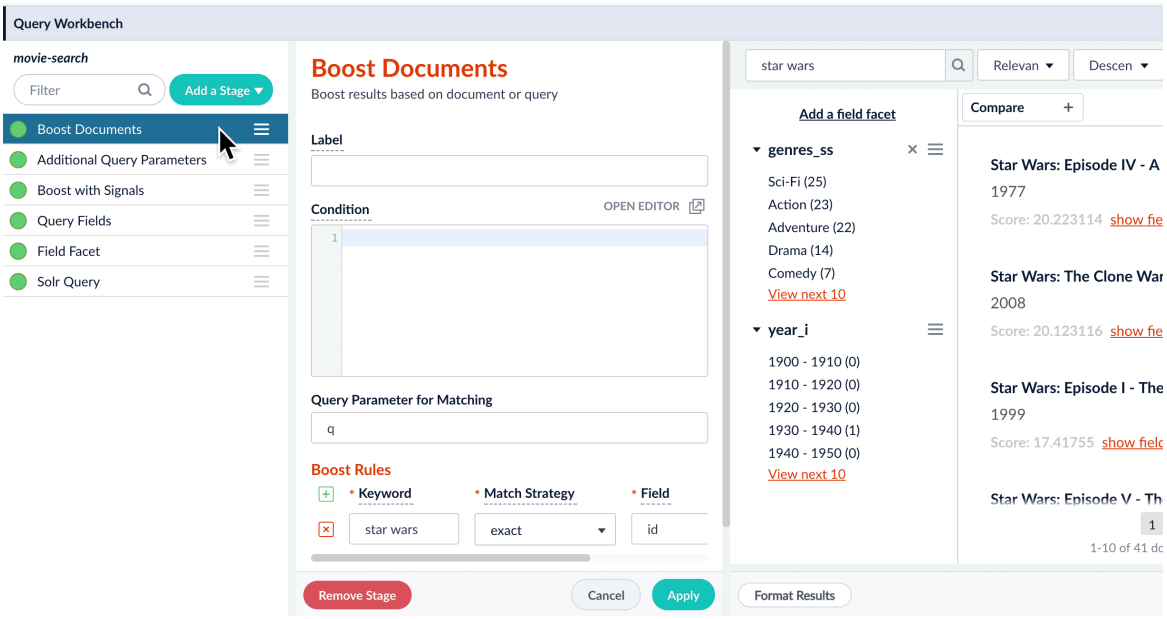
1. In the preview panel, hover over the document you want to boost.



2. Click **Boost**.
   The preview panel automatically updates the rankings of the search results, and tags the differences:



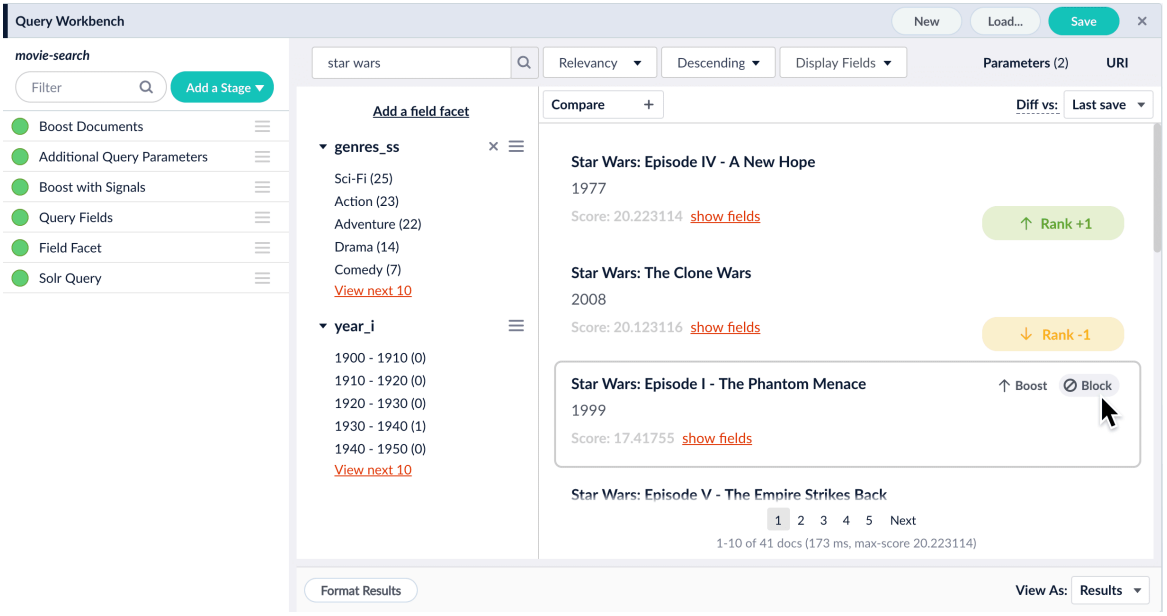
Notice that the Boost Documents stage is now in our pipeline. You can click this stage to view the boost rules you have added. This is also where you can remove boost rules:



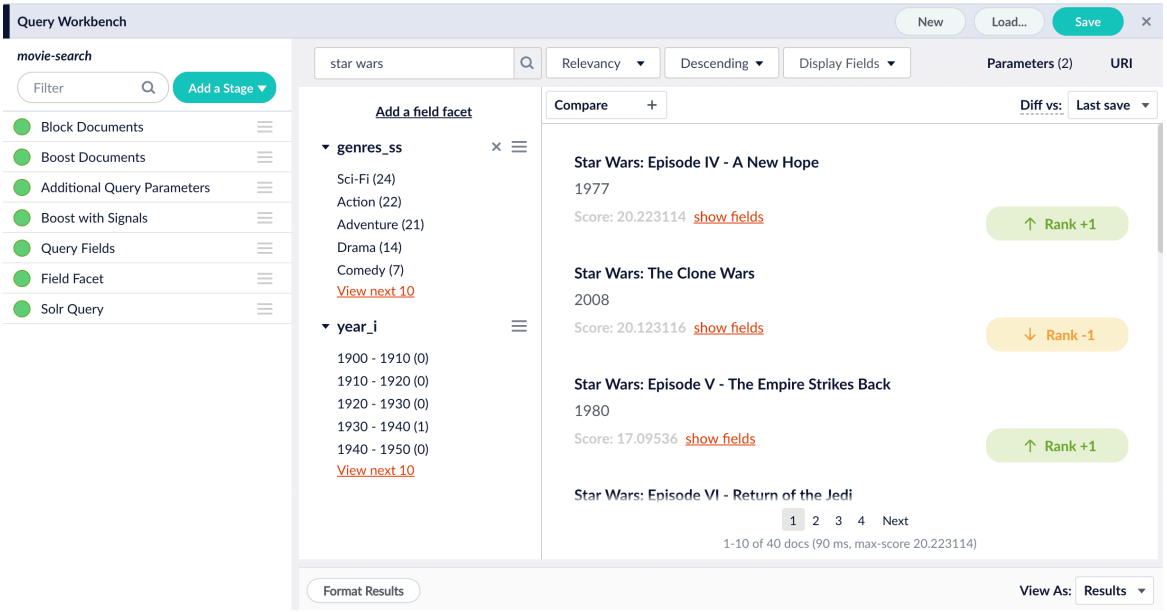
3. When you are satisfied with the results, save the query pipeline.

How to block a document in the Query Workbench

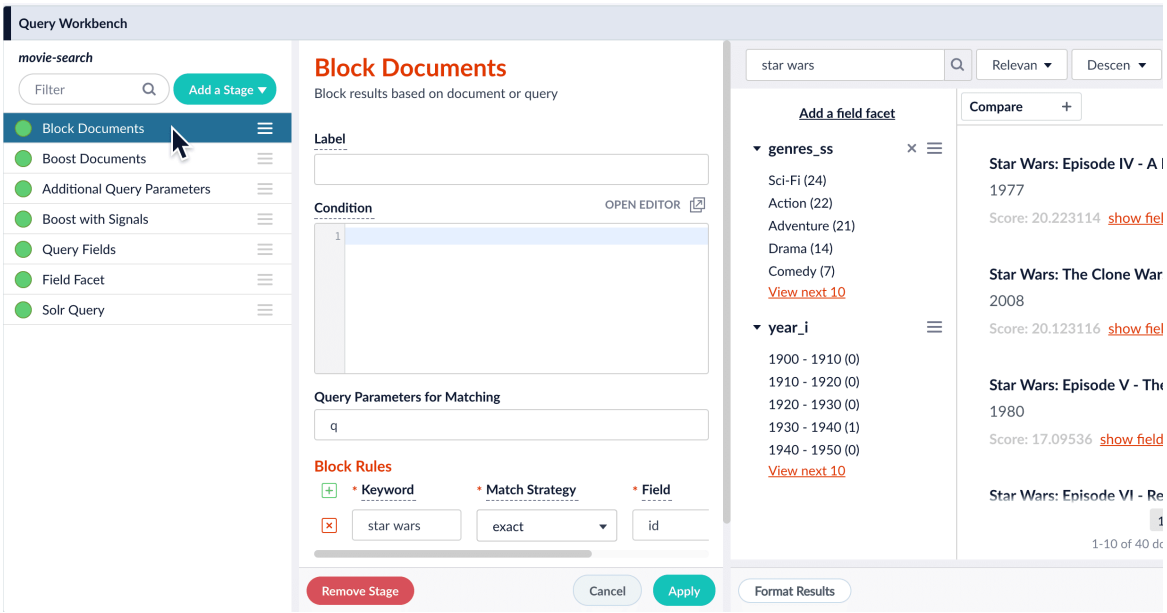
1. In the preview panel, hover over the document you want to block.



2. Click **Block**
   The preview panel automatically updates the rankings of the search results, and tags the differences.



Notice that the Block Documents stage is now in our pipeline. You can click this stage to view the block rules you have added. This is also where you can remove block rules:



3. When you are satisfied with the results, save the query pipeline.

See these pages for more information:

- *Boost Documents stage*
- *Block Documents stage*

Build and Deploy A Query Stage Plugin

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion*

This topic explains how to build and deploy a custom query stage plugin that you developed using the Query Stage SDK.

Fusion also comes with a number of *built-in query stages* that you can access using the *Query Workbench*.

Prerequisites

- Clone the repository .
- Develop your plugin using the *Query Stage SDK library*.

1. Build the plugin stage

From the main folder, call:

```
./gradlew -p examples/sample-plugin-stage clean assemblePlugin
```

Substitute the path to your project.

This creates a plugin `.zip` file (with required manifest file) inside the `build/libs` folder.

2. Deploy the stage

You can choose from several different ways to deploy the sample plugin stage:

- *Upload it to the blob store*
- *Use gradle*
- *Use the REST API*

After successful deployment, new stages should be visible in the `Stages` list in the Fusion Query Pipelines UI.

Upload it to the blob store

1. Navigate to **System** > **Blobs**.
2. Click **Add**.
3. Select **Query Stage Plugin**.
4. Click **Browse...** and select your plugin file.
5. Click **Upload**

Deploy it using gradle

```
./gradlew -p examples/sample-plugin-stage deploy -PfusionUser=[user] -
PfusionPassword=[password]
```

Deploy it with the REST API

```
curl -u [user]:[password] -X PUT -H "Content-Type:application/zip" --
data-binary @sample-plugin-stage-0.0.1.zip https://<fusion-
host>/api/query-stage-plugins
```

Built-in SQL Aggregation Jobs Using Cloud Storage Buckets

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion*

Built-in SQL aggregation jobs can be set up to use source files in Cloud storage buckets.

This process can be used with the following data types and Cloud storage systems:

- File formats such as `.parquet` and `.orc` files
- Cloud storage systems such as Google Cloud Storage (GCS), Amazon Web Services (AWS), and Azure Kubernetes Service (AKS).

Configure Parameters

Google Cloud Storage (GCS)

1. Create a Kubernetes secret with the necessary credentials. For more information about creating a secret containing the credentials JSON file, see *Configuring credentials for Spark jobs*.
2. When the secret is successfully created, set the following parameters:

GENERAL PARAMETERS		
Parameter Name	Example Value	Notes
SOURCE COLLECTION	`gs://<path_to_data>/*.parquet`	Value: URI path that contains the desired signal data files. Example value returns all parquet files in the directory, where `gs` is used to access GCS data.
DATA FORMAT	`parquet`	Value: File type of input file. Other value can be `orc`.
SPARK SETTINGS		
Parameter Name	Example Value	Notes
`spark.kubernetes.driver.secrets.{secret-name}`	`/mnt/gcp-secrets`	Value: The `{secret-name}` obtained during configuration. Example: `example-serviceaccount-key`.
`spark.kubernetes.executor.secrets.{secret-name}`	`/mnt/gcp-secrets`	Value: The `{secret-name}` obtained during configuration. Example: `example-serviceaccount-key`.
`spark.kubernetes.driverEnv.GOOGLE_APPLICATION_CREDENTIALS`	`/mnt/gcp-secrets/{secret-name}.json`	Value: The name of the `.json` file used to create the `{secret-name}` obtained during configuration. Example: `example-serviceaccount-key.json`.
`spark.executorEnv.GOOGLE_APPLICATION_CREDENTIALS`	`/mnt/gcp-secrets/{secret-name}.json`	Value: The name of the `.json` file used to create the `{secret-name}` obtained during configuration. Example: `example-serviceaccount-key.json`.
`spark.hadoop.google.cloud.auth.service.account.json.keyfile`	`/mnt/gcp-secrets/{secret-name}.json`	Value: The name of the `.json` file used to create the `{secret-name}` obtained during configuration. Example: `example-serviceaccount-key.json`.

Amazon Web Services (AWS)

1. Create a Kubernetes secret with the necessary credentials. For more information about creating a secret containing the credentials JSON file, see *Configuring credentials for Spark jobs*.
2. When the secret is successfully created, set the following parameters:

GENERAL PARAMETERS		
Parameter Name	Example Value	Notes
SOURCE COLLECTION	`s3a://<path_to_data>/*.parquet`	Value: URI path that contains the desired signal data files. Example value returns all parquet files in the directory, where `s3a` is used to access AWS data.
DATA FORMAT	`parquet`	Value: File type of input file. Other value can be `orc`.
SPARK SETTINGS		
Parameter Name	Example Value	Notes
`spark.kubernetes.driver.secretKeyRef.AWS_ACCESS_KEY_ID`	`{aws-secret-key}`	Value: The `aws-secret:key` obtained during configuration.
`spark.kubernetes.driver.secretKeyRef.AWS_SECRET_ACCESS_KEY`	`{aws-secret-secret}`	Value: The `aws-secret:secret` obtained during configuration.
`spark.kubernetes.executor.secretKeyRef.AWS_ACCESS_KEY_ID`	`{aws-secret-key}`	Value: The `aws-secret:key` obtained during configuration.
`spark.kubernetes.executor.secretKeyRef.AWS_SECRET_ACCESS_KEY`	`{aws-secret-secret}`	Value: The `aws-secret:secret` obtained during configuration.

Azure settings

GENERAL PARAMETERS		
Parameter Name	Example Value	Notes
SOURCE COLLECTION	`wasbs://<path_to_data>/*.parquet`	Value: URI path that contains the desired signal data files. Example value returns all parquet files in the directory, where `wasbs` is used to access Azure data.
DATA FORMAT	`parquet`	Value: File type of input file. Other value can be `orc`.
SPARK SETTINGS		
Parameter Name	Example Value	Notes
`spark.hadoop.fs.wasbs.impl`	`org.apache.hadoop.fs.azure.NativeAzureFileSystem`	Makes the system file available inside the Spark job.
`spark.hadoop.fs.azure.account.key.{storage-account-name}.blob.core.windows.net`	`{access-key-value}`	Obtain the values for {storage-account-name} and {access-key-value} from the Users Azure UI.

Bury a Product in Experience Optimizer

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

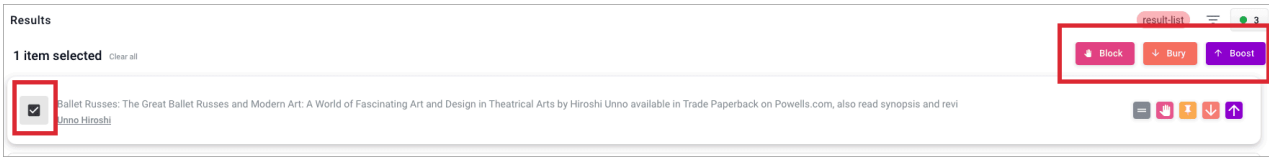
You can bury a document so that it occupies a lower position in the search results when your customers run a particular query. This can be helpful if some of the products in your catalog are temporarily out of stock, for example.

Burying a document typically moves it to the end of the list. However, its exact position will depend on multiple factors, including its relevance score and user signals in relation to other products.

When you bury a document, a new rule is created automatically.

1. Hover your cursor over a document. A menu overlay will appear.
2. Hover your cursor over the **Actions** button. A dropdown menu will appear.
3. Select the **Bury** button:

You can also bury multiple products at once by clicking the checkbox in the top-left corner of the document tiles and selecting the **Bury** button above the top row of products.



Bury a Product in Experience Optimizer

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion*

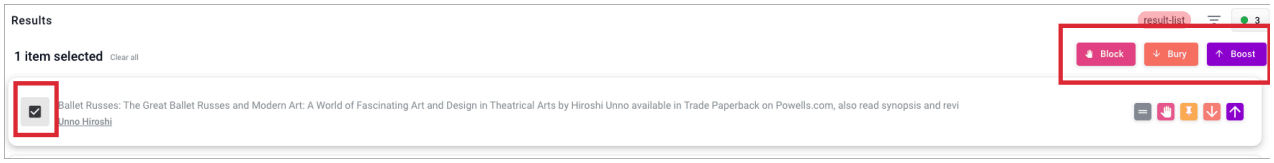
You can bury a document so that it occupies a lower position in the search results when your customers run a particular query. This can be helpful if some of the products in your catalog are temporarily out of stock, for example.

Burying a document typically moves it to the end of the list. However, its exact position will depend on multiple factors, including its relevance score and user signals in relation to other products.

When you bury a document, a new rule is created automatically.

1. Hover your cursor over a document. A menu overlay will appear.
2. Hover your cursor over the **Actions** button. A dropdown menu will appear.
3. Select the **Bury** button:

You can also bury multiple products at once by clicking the checkbox in the top-left corner of the document tiles and selecting the **Bury** button above the top row of products.



Bury a Product in Predictive Merchandiser

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion*

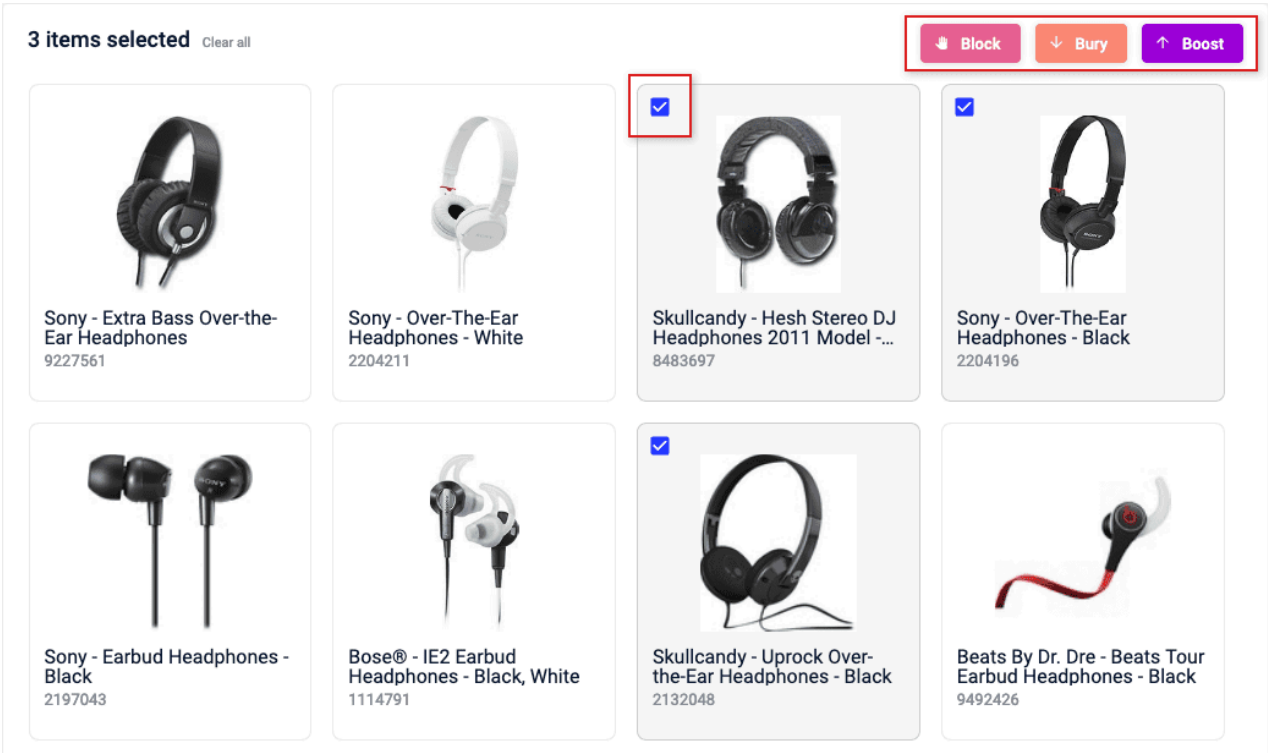
You can bury a product so that it occupies a lower position in the search results when your customers run a particular query. This can be helpful if some of the products in your catalog are temporarily out of stock, for example.

Burying a product typically moves it to the end of the list. However, its exact position will depend on multiple factors, including its relevance score and user signals in relation to other products.

When you bury a product, a new rule is created automatically.

1. Hover your cursor over a product. A menu overlay will appear.
2. Hover your cursor over the **Actions** button. A dropdown menu will appear.
3. Select the **Bury** button:

You can also bury multiple products at once by clicking the checkbox in the top-left corner of the product tiles and selecting the **Bury** button above the top row of products.



Bury a Product in Predictive Merchandiser

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

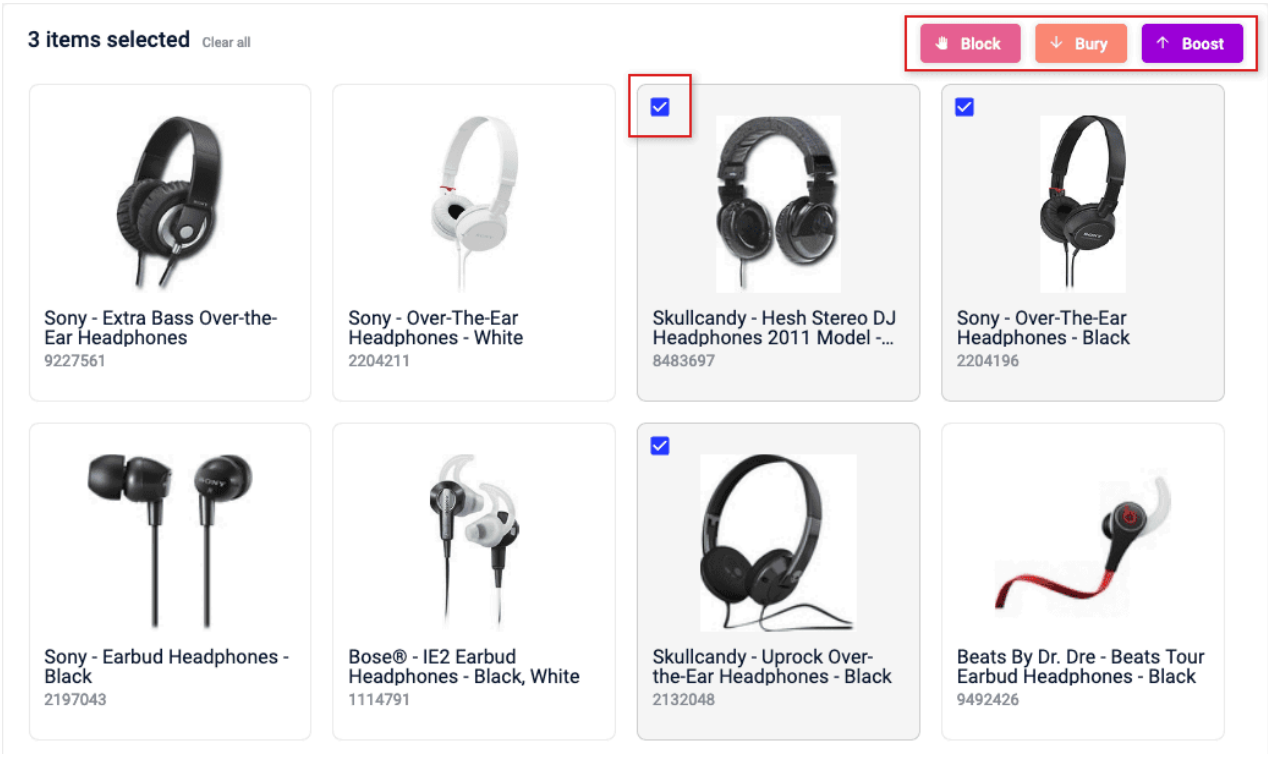
You can bury a product so that it occupies a lower position in the search results when your customers run a particular query. This can be helpful if some of the products in your catalog are temporarily out of stock, for example.

Burying a product typically moves it to the end of the list. However, its exact position will depend on multiple factors, including its relevance score and user signals in relation to other products.

When you bury a product, a new rule is created automatically.

1. Hover your cursor over a product. A menu overlay will appear.
2. Hover your cursor over the **Actions** button. A dropdown menu will appear.
3. Select the **Bury** button:

You can also bury multiple products at once by clicking the checkbox in the top-left corner of the product tiles and selecting the **Bury** button above the top row of products.



Change a Fusion User's Password

How-to

> Compatible versions: 4.2, 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion Server*, *Fusion*

Change your Fusion Server password

While you are logged in, from anywhere in the Fusion UI:

1. Click **Account Settings**.
2. Click **My Profile**.
3. Follow the UI instructions to update your password.

Change another user's Fusion Server password

If you are an admin, you can change the password for any of your users. While you are logged in, from any app in the Fusion UI:

1. Click **System > Access Control**.
2. Make sure the **Users** tab is selected.
3. Click the user's name.
4. Check the box under **Change password**.
5. Enter and confirm a new password, and click **Update**.

Reset a lost admin password

If you have lost the password for the admin account, contact Lucidworks Support .

Change a user password

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

Change your password

1. Sign in to Managed Fusion.
2. Click **Account Settings**.
3. Click **My Profile**.
4. Follow the UI instructions to update your password.

Change another user's password

If you are an admin, you can change the password for any of your users.

1. Sign in to Managed Fusion and click any application.
2. Click **System > Access Control**.
3. Click the **Users** tab if it is not selected.
4. Click the user's name.
5. Select the **Change password** box.
6. Enter and confirm a new password.
7. Click **Update**.

Reset a lost admin password

If you have lost the password for the admin account, contact Lucidworks Support .

Changing the Color Scheme

How-to

> Compatible versions: 4.2, 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *App Studio*

The principle colors used in the theme are define at the top of `styles/includes/theme.less` :

```
@color-primary: #273142; // Brand Primary @color-secondary: #202020;
// Utility Black (Non selectable) @color-tertiary: #4CAF50; // Utility
Green Links (Non selectable) @color-quaternary: #f7f7f7; // Utility
Grey page bg @color-quinary: #d9d9d9; // Utility Grey borders @color-
white: #fff; @color-black: #000; @link-color: #498fe2;
```

Change the `@color-primary` , `@color-secondary` , and `@color-tertiary` variables to the desired color.

For more precise control, the color of individual elements can be controlled in the styling partial stylesheets found in `styles/includes/` .

Changing the Logo

How-to

> Compatible versions: 4.2, 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *App Studio*

To change the logo

1. Prepare both a large and a small version of the logo. The large version should be at least 400 pixels wide; the small version should be 42 pixels tall, and no wider than 225 pixels. It is recommended to save the logos as transparent 24-bit PNGs. Place the two image files in the `assets` folder.
2. Update the image path in `webapp/login.jsp` (if present) for the large version of the logo. You can also adjust the width of the image tag at the same time.
3. Update the path in the `<layout:logo>` tag which is usually placed within `webapp/views/partials/header.tpl.html` for the small version of the logo.

Check the System State

How-to

> Compatible versions: 4.2
>
> Works with: *Fusion Server*, *Fusion*

As described in *Fusion 4.x Directories, Files, and Ports*, Fusion runs several components as separate JVMs running on different ports. Each of the components is capable of reporting its status. The proxy component reports status for all of the other components.

Status Check for Zookeeper and connectors-classic

To see whether each component has been started, a simple API call to the proxy (running on port 8764 by default) will return the status of each component of the system.

```
curl http://localhost:8764/api
```

The response should look similar to the following. If 'ping' is true for each service, all of the system components are running.

```
{"status":{"db":{"ping":true},"apolloZk":{"ping":true},"apollo":
{"ping":true}, "connectors":{"ping":true},"fusion":
{"ping":true}},"startTime":"2018-02-16T16:23:51Z", "initMeta":
{"initializedAt":"2018-02-15T21:50:43Z","version":"local",
"nativeRealmInstalled":"2018-02-15T21:53:44Z"},"enabledRealms":
[{"name":"native","realmType":"native"}, {"name":"webapps-jwt-
realm","realmType":"jwt"}],"version":"local"}
```

Solr Health Check

The Fusion UI and API services are not accessible if ZooKeeper and Solr are not in healthy state. A Solr health check can be performed with a ping request, for example:

```
curl http://localhost:8983/solr/collection-name/admin/ping
```

The response will be a JSON response, similar to the following:

```
{ "responseHeader": { "zkConnected": true, "status": 0, "QTime": 0,
"params": { "q": "{!lucene}*:*", "distrib": "false", "df": "_text_",
"rows": "10", "echoParams": "all" } }, "status": "OK" }
```

The `status` should be `OK` if Solr is able to return results for queries.

REST API Services Health Check

All of the Fusion API backend services (except Connectors and the UI) are started at port 8765 when the run.sh script is executed. The Fusion UI depends on all these services.

If all the services are started without any issues, then the below ping request should return the response `ok`.

```
curl http://<FUSION_HOST>/api
```

As an alternative check, you can also query the `system/status` endpoint, which returns the status.

```
curl http://localhost:8764/api/system/status
```

The response looks like something like this:

```
[ { "status" : "ok", "node" :
"http://127.0.0.1:8984/connectors/v1/system", "messages" : [ ] }, {
"status" : "ok", "node" : "http://127.0.0.1:8764/api/system",
"messages" : [ ] } ]
```

Connectors Classic Health Check

You can perform a health check of the Connectors Classic Service by sending a ping request to port 8984. Similar to the previous ping request, the returned response is `ok` if the service started successfully.

```
curl http://localhost:8984/connectors/v1
```

As an alternative check, you can also query the `system/status` endpoint, which returns the status.

```
curl http://localhost:8984/connectors/v1/system/status
```

The response looks like something like this:

```
[ { "status" : "ok", "node" :
"http://127.0.0.1:8984/connectors/v1/system", "messages" : [ ] }, {
"status" : "ok", "node" : "http://127.0.0.1:8764/api/system",
"messages" : [ ] } ]
```

Clean Up Spark Driver Pods

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion*

Spark driver pods are cleaned up using a Kubernetes cron job that runs every 15 minutes. This cron job is created automatically when the `job-launcher` microservice is installed in the Fusion cluster.

To clean up pods manually, run this command:

```
kubectl delete pods --namespace default --field-
selector=status.phase=Succeeded -l spark-role=driver
```

For related topics, see *Spark Operations*.

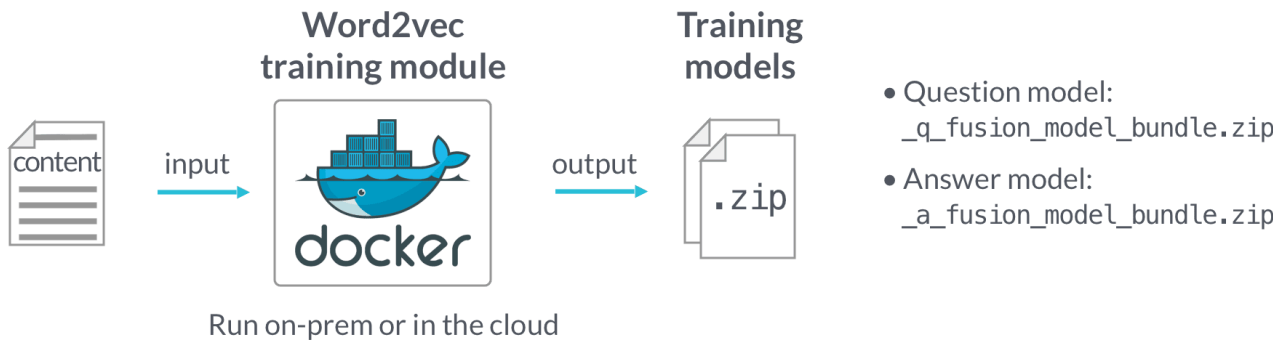
Cold Start Part 1

Model Training (Fusion 4.2 only)

How-to

> Compatible versions: 4.2
>
> Works with: *Fusion AI*

The *cold start solution for Smart Answers* in Fusion 4.2 begins with training a model using your existing content and the Word2vec training module in the Docker container. Once you have trained a model, you can *deploy it*.



See also *Advanced Model Training Configuration for Smart Answers*.

> **Tip**
>
> Begin collecting *signals* after your cold-start model is deployed. Once you have accumulated a sufficient number of signals, you can train a model more accurately, using the *FAQ solution*.

Prerequisites

You need a Docker Hub account with access to the `lucidworks/ai-lab` repository. Contact Lucidworks AI Labs with your Docker Hub account name to gain access to this repository.

Hardware requirements and instance setup

The Docker training module can run on either a CPU or a GPU.

CPU is sufficient for most use cases.

GPU is 2-8 times faster. GPU is recommended if want to try our model auto-tune feature on a large dataset (e.g. over 20k QA pairs) or if the training data size is over 1 million entries for any of the training scenarios.

You can run the Docker image either on-prem or in the cloud. We recommend a minimum of 32GB RAM and 32 cores for a CPU server or a cloud instance. If you choose a GPU machine, one GPU with 11GB memory is sufficient for most training use cases.

Below are examples of the AWS instance types that we recommend:

CPU:

- AMI: Ubuntu Server 18.04 LTS (HVM), SSD Volume Type
- Instance Type: c5.9xlarge (36 vCPUs, 72 GB RAM)
- Recommended storage volume: Suggested 150GB plus 2.5 times the total input data size
- Networking
    - Allow inbound port 4440
    - Allow inbound port 5550

GPU:

- AMI: Deep Learning AMI (Ubuntu) Version 22.0
- Instance Type: p2.xlarge (11.75 ECUs, 4 vCPUs, 61 GB RAM, EBS only)
- Recommended storage volume: Suggested 150GB plus 2.5 times of total input data size.
- Networking
    - Allow inbound port 4440
    - Allow inbound port 5550

Below are examples of the Azure GPU instance type that we recommend:

- Deep Learning Virtual Machine
- Linux OS
- VM Size: 1x Standard NC6 (6 vcpus, 56 GB memory)
- Recommended OS Disk Size: Suggested 150GB plus 2.5 times the total input data size
- Networking
    - Allow inbound port 4440
    - Allow inbound port 5550

Docker image setup

1. Determine the instance's external hostname.
   Refer to your cloud provider's documentation for instructions on how to do this.
2. Create a directory, such as `/opt/faq_training`, and ensure it meets *disk space requirements*. Also ensure that it is writable by the `docker` Linux user account.
   The training Docker container requires this directory to read input files, write temporary files, and write the output (deep learning models).
3. Run `docker login` and enter your account credentials.
4. Pull the `lucidworks/ai-lab:latest` image:

```
docker pull lucidworks/ai_lab:latest
```

5. Run the Docker image:
   For GPU instance:

```
docker run -d --rm \ -e "RD_HOSTNAME=<external hostname here>" \
-e "ENABLE_GPU=1" \ --name deep_faq \ -v <working directory
here>:/src/DL/working_dir \ --net=host \          --
runtime=nvidia \ lucidworks/ai_lab:latest
```

   For CPU-only instance:

```
docker run -d --rm \ -e "RD_HOSTNAME=<external hostname here>" \
--name deep_faq \ -v <working directory
here>:/src/DL/working_dir \ --net=host \
lucidworks/ai_lab:latest
```

   This starts the Docker container in the background.
6. Wait about one minute for the container to initialize.
   In your working directory, you should see these directories:

```
conf data input models output rundeck
```

- The `input` directory is where all training data and input files should be placed.
- The `output` directory contains the models that result from the training.
- All other directories are used by the model training container to save persistent files or temporary files.

> **Tip**
>
> You can clear the `data` folder periodically to save disk space.

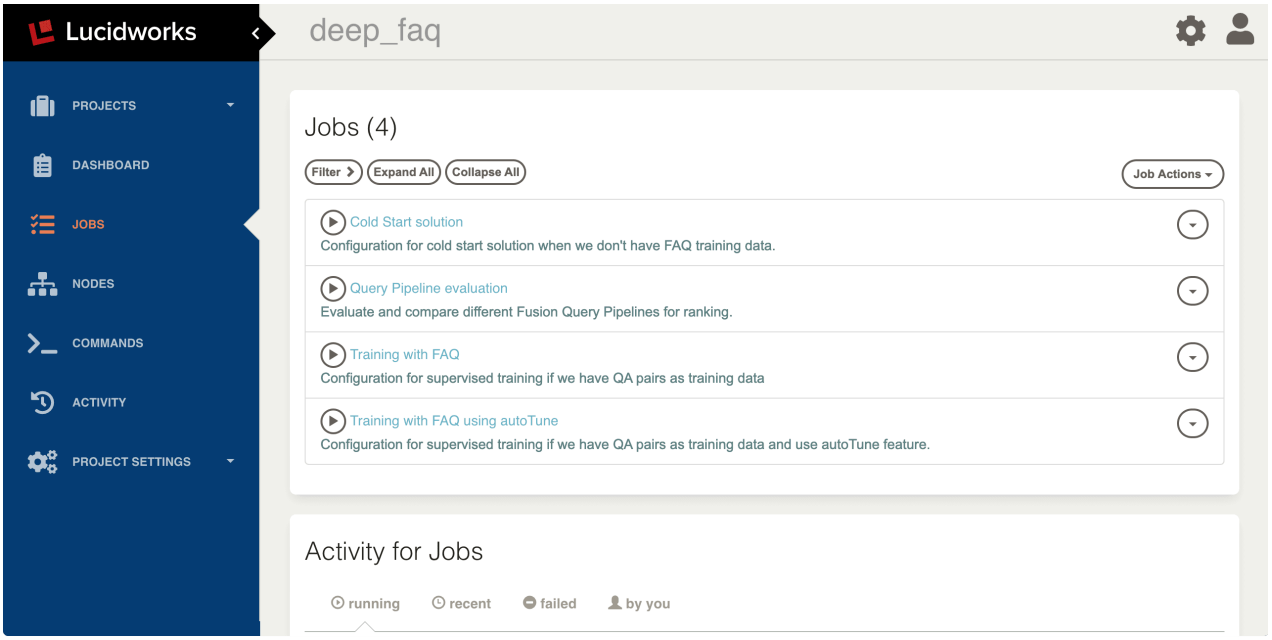
7. Point your browser to `http://<external hostname>:4440`.
   This redirects you to a Rundeck login screen.
8. Log in with your username and password.

Running Jobs

We use https://www.rundeck.com/open source[Rundeck^] to control job runs.

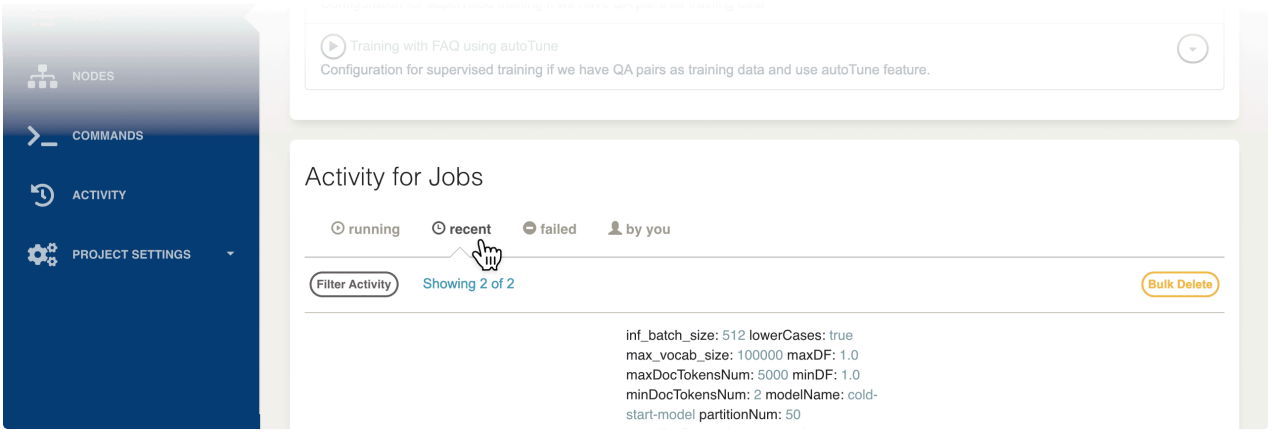
1. When Rundeck has started, click on the `deep_faq` project.
   This opens the **JOBS** panel, where you will see three jobs for different training scenarios and one job for query pipeline evaluation.
   - Click on a job to configure job parameters.
   - Click **Run Job Now** to start the job.
   - You can monitor the progress of a job by clicking on the **running** button under the **Activity for Jobs** section as shown in the screenshot below.

> **Note**
>
> If similar jobs have run before, then Rundeck estimates the total run time and displays a progress bar.



You can click on the running job to enter the activity control page, where you can kill the job or watch the log output.

If you want to reuse the previously-configured job, click the **recent** button under **Activity for Jobs**.



Select a recent job configuration that you want to modify, then click the button to enter the configuration page with previously-entered values.

For more details about Rundeck job control, refer to the Rundeck documentation.

Training scenarios and configuration

Click on the **Cold Start solution** job.

This job can learn about the vocabulary that you want to search against through Word2vec. And we can combine Solr with Word2vec at query time using our query pipeline.

Input data parameters

For cold start solution input, these parameters are required:

- Content documents file name
- Field which contains the content documents

There are two vocabulary learning options:

- You can use the pre-trained word embeddings included with our training module.
  In cold start mode, it gets the Word2vec vectors from the pre-trained file directly. The pre-trained word embeddings are built on a large corpus from the Internet, covering common words we use daily.

- If there are many domain-specific words or special tokens in your documents, we recommend training Word2vec using your own data.
  Set the **Generate customized embeddings** parameter to "true" to train from your own data. This uses the data and field specified in **Content documents file name** and **Field which contains the content documents** to train Word2vec vectors.

Tip

When you use the pre-trained embeddings, the log shows the percentage of covered vocabulary words (search for vocab coverage by embeddings in the log). If the number of covered vocabulary words is less than 80%, then you should train Word2vec from your own data.

Note

You must provide a **Model name** for model tracking and version control purposes. If you use the same model name across different runs, the new model will replace the old model with the same name.

See also the *Advanced Model Training Configuration for Smart Answers*.

Result models

Track the running steps in the log, which provides information on data pre-processing, training steps, evaluations, and model generation in the end.

After training is finished, the final models and their associated files are saved in zip file format, which can be downloaded from `http://<external hostname>:5550/{modelName}`.

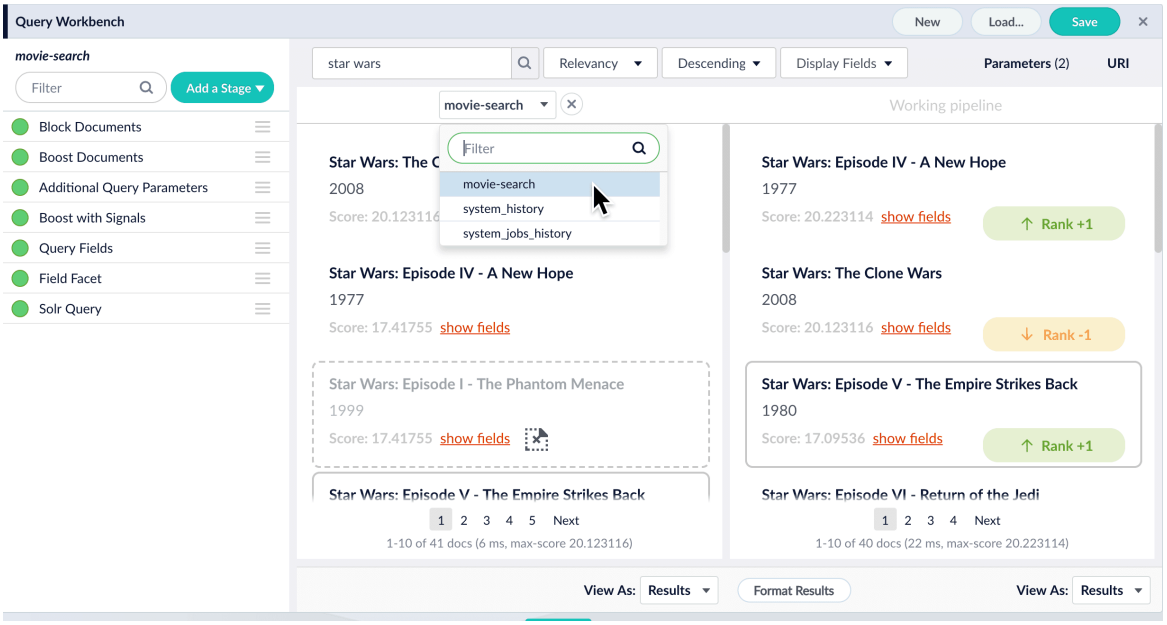
Download the two zipfiles ending with `_a_fusion_model_bundle.zip` and `_q_fusion_model_bundle.zip` to be used in the *model deployment* phase.

Compare Two Query Pipelines

How-to

> Compatible versions: 4.2, 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion Server*, *Fusion*

1. In the Query Workbench, click **Compare**.
   Another preview panel opens. In this view, you can compare results from one query pipeline side by side with another query pipeline.
2. In the left panel, select a pipeline to compare to your working pipeline.
   Now you can see how the search results differ between the two pipelines:



Tip

When you click a document in one panel, the other panel automatically scrolls to the same document.

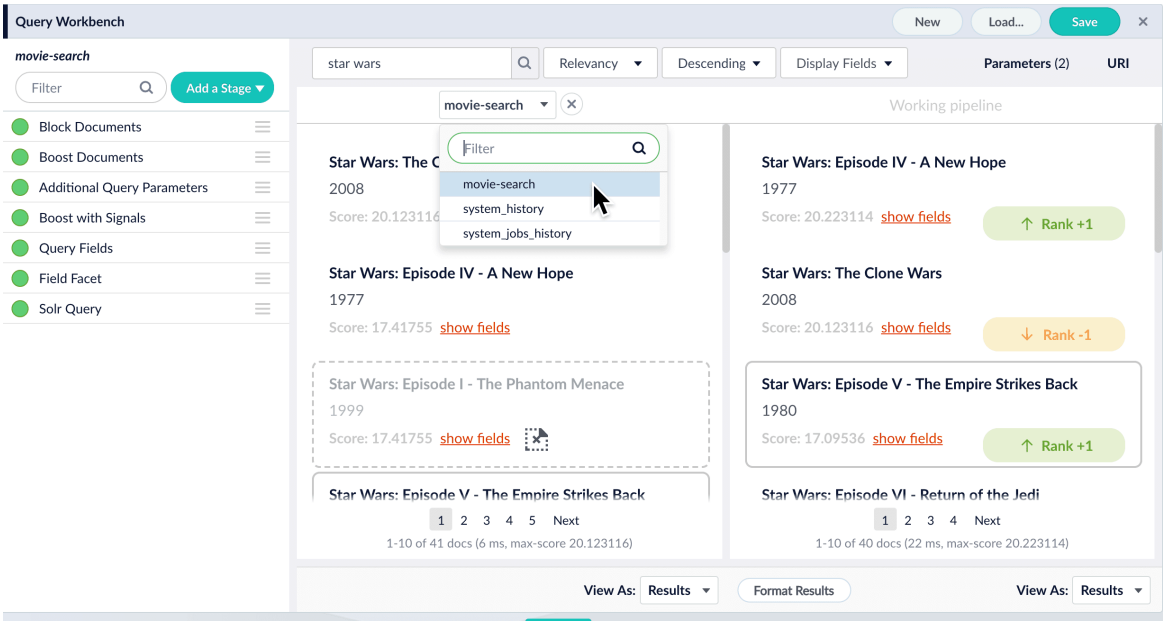
To exit compare mode, click the Close icon.

Compare Two Query Pipelines

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

1. In the Query Workbench, click **Compare**.
   Another preview panel opens. In this view, you can compare results from one query pipeline side by side with another query pipeline.
2. In the left panel, select a pipeline to compare to your working pipeline.
   Now you can see how the search results differ between the two pipelines:



> **Tip**
>
> When you click a document in one panel, the other panel automatically scrolls to the same document.

To exit compare mode, click the Close icon.

Configure a Box.com Datasource

How-to

> Compatible versions: 4.2, 5.4, 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion Connectors*, *Fusion Server*, *Fusion*

The Box connector retrieves data from a Box.com cloud-based data repository.

This topic applies to the Box.com V1 and Box.com V2 connectors. The Box.com V1 connector is available in Fusion 5.2 and earlier. The Box.com V2 connector is available in Fusion 5.3 and later.

Configuration overview

> **Note**
>
> These steps are for a multi-user Box.com data repository. For limited testing using a single user account, you can *create a Box app that uses Standard OAuth 2.0 authentication*.

Following is an overview of the steps required to set up Box and Fusion, and to crawl a Box data repository.

Set Up Box:

1. Sign up for a Box developer account.
2. Enable 2-step verification.
3. Create a Box app that Fusion can use to crawl the Box files.
4. Configure your app to use a Box service account.

Set Up Fusion:

5. Install Fusion's Box Connector.
6. Create datasources in Fusion that use the Box connector.

Crawl a Box Data Repository:

7. Crawl the Fusion datasources.

Set Up Box

Set up Box so that Fusion can crawl Box data repositories.

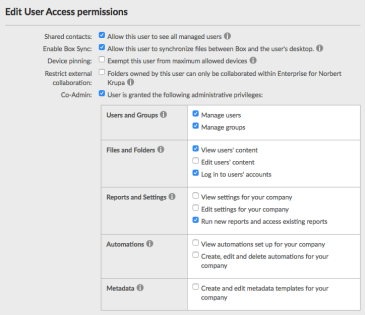
Step 1: Sign Up for a Box Developer Account

If you already have an account, proceed to *Step 2*.

1. Open the Box Developers Console .
2. In the top right corner, click **Sign Up**.
3. Select an appropriate Platform Developer plan.
4. Enter the requested information and click **Submit**.
5. Open the confirmation email and click **Verify Email**.
6. Log in to your Box account.

Step 2: Enable 2-Step Verification

1. Log in to your Box developer account.
    a. Open the Box Developers Console .
    b. Log in as the admin.
2. .Create the Box account that you want to use for crawling.
    a. Open the Users page in the Box admin console.
    b. Click **+Users** to create a new user account.
    c. Enter the Name and Email for the user, and then click **Add user**.
    d. Click the user you just created to enter its user settings.
    e. Make this user a Co-Admin by selecting Co-Admin checkbox. Once clicked, a pane titled "User is granted the following administrative privileges" appears. Select all of the following:
        - Manage users
        - Manage groups
        - View users' content
        - Log in to users' accounts
        - Run new reports and access existing reports



f. Click **Save**.
g. Close the Admin Console browser tab.

3. Enable 2-step verification for unrecognized logins:
   a. Open the Account Settings page. (You can reach this page from the drop-down menu under your initials.)
   b. On the **Account** tab, under **Authentication**, select **Require 2-step verification for unrecognized logins**.
   c. Choose your **Country** and enter a **Mobile Phone Number**, and then click **Continue**.
   d. Enter the verification code you receive, and then click **Continue**.
   e. If you are using a new mobile device, Box will send you a second code. Enter it, and then click **Submit**.
   f. Click **Save Changes**.

Step 3: Create a Box App that Fusion Can Use to Crawl the Box files

Create a Box app that uses OAuth 2.0 with JWT server authentication.

If you already have an app, *configure it*.

1. Open the Box Developers Console .
2. Click **Create New App**.
3. Select **Custom App**, and then click **Next**.
4. Click **OAuth 2.0 with JWT (Server Authentication)**, and then click **Next**.
5. Name your app, and then click **Create App**. The name must be globally unique across all apps created by all Box users.
6. Click **View Your App**.

Step 4: Configure Your App to Use a Box Service Account

1. .Use OpenSSL to create a private/public key pair:
   a. Install OpenSSL if you need to. Windows instructions are here .
   b. Open a Command Prompt window and run these commands to generate a private/public key pair:

```
openssl genrsa –aes128 –out private_key.pem 2048
```

Enter a password for the private key when prompted.

```
openssl rsa -pubout -in private_key.pem -out
public_key.pem
```

In the current directory of the Command Prompt, you now have private and public key files, `private_key.pem` and `public_key.pem` respectively.

2. Open the Box Developers Console, log in as Admin if you are asked to log in, and click your app.
3. .In the left navigation menu, click **Configuration**.
4. Configure scopes and advanced features:
    a. Under **Application Access**, select **Enterprise**.
    b. Under **Application Scopes**, deselect **Manage groups**.
    c. Under **Advanced Features**, enable **Generate User Access Tokens** and **Perform Actions as Users**.
    d. Click **Save Changes**.
5. .In the **Add and Manage Public Keys** area, click **Add a Public Key** and paste the contents of the `public_key.pem` file (generated from the *Box key creation*) into the text box.
    a. Make a note of the new **Public Key ID** that you just created.
6. .Under **OAuth 2.0 Credentials**, click **COPY** for the Client ID.
7. Authorize your app:
    a. Open the Box Admin Console.
    b. In the left navigation menu, click **Settings** > **Enterprise Settings** (or **Business Settings**) > **Apps**.
    c. Under **Custom Applications**, click **Authorize New App**.
    d. In the **API Key** box, paste the Client ID credential you copied in *step 6*, and then click **Next**.
    e. Read the App Authorization dialog and click **Authorize**.
    f. Close the Admin Console browser tab.

> **Note**
>
> If you change your app's configuration later, you must repeat this step to re-authorize your app.

8. Close the Dev Console browser tab.

Set Up Fusion

Set up Fusion to crawl Box data repositories.

Step 5: Install Fusion's Box Connector

1. Navigate to the *Fusion 4.x Connector Downloads*
2. Select the Box connector link for the release to download. The .zip file is downloaded.
3. Do not unzip the file.
4. Open the Fusion UI and click **System** > **Blobs**.
5. Click **Add**.
6. Select **Connector Plugin**.
7. Click **Choose File**, select the file, and then click **Open**.
8. Click **Upload**.

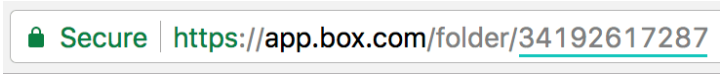
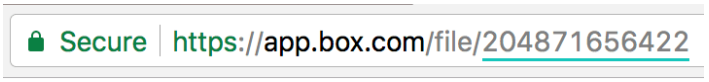
Step 6: Create Datasources

Create datasources that use the Box connector to access the Box data repository.

For each datasource:

1. In the Fusion UI, Navigate to **Indexing** > **Datasources**.

2. Click **Add**.
3. Select **Box.com**.
4. Fill in the form. Note the following regarding configuration settings to use:

Setting	Notes
Start Links	Each start link defined for the datasource must consist of a numeric Box file ID or directory ID. The root directory of any Box account has ID 0 (zero). To crawl your entire Box repository, enter '0'. These images indicate with underlines where you can get a folder ID or file ID. Select a folder or file at Box.com. Folder ID:  Enter the start link `34192617287`. File ID:  Enter the start link `204871656422`:
API Key	In the Box Developers Console, select the app. On the **Configuration** tab under **OAuth 2.0 Credentials**, use the **Client ID**.
API Secret	In the Box Developers Console, select the app. On the **Configuration** tab under **OAuth 2.0 Credentials**, use the **Client Secret**.
JWT App User ID	Email address that you use to sign in to your Box co-admin account. Use the *Co-admin account you created earlier* for this.
JWT Public Key Id	In the Box Developers Console, select the app. On the **Configuration** tab, under **Add and Manage Public Keys**, use the **ID** for a public key.
JWT Private Key	Base64-encoded contents of the *private-key file* that matches your JWT Public Key Id. Base64 encode the *entire* contents of the file, including the first and last lines. *(Fusion 5.0+ only.)*
JWT Private Key File	Full path to the *private-key file* you created that matches your JWT Public Key Id. *(Prior to Fusion 5.0 only.)*
JWT Private Key Password	Passphrase for the private key (from the private-key file you created during *Box key creation*).
Distributed crawl collection name	Collection that contains the *pre-fetch index*.
Box.com children responses per page	Use the default value of 1000.
Nested folder depth limit	Generally, you want a number that will crawl all documents, so keep the default value. For testing, you could reduce the number substantially to speed up the crawl.

Setting	Notes
Number of partition buckets	Divide the number of files by 5000. Use that number or 10000, whichever is smaller.
Number distributed crawl datasources	Use 1 to 27.
Number of pre-fetch index creator threads	A number between 2 and 5. Use 2 for small datasources and 5 for huge datasources (over 10 million files).

5. Click **Save**.

Crawl a Box Data Repository

Crawl a Box data repository.

Step 7: Crawl the Fusion Datasources

Crawl the datasources, which use Fusion's Box connector to access the Box data repository. Fusion's Box connector uses the pre-fetch index to fetch the contents of each file from Box.com, get metadata from both the distributed index and Box.com, and index the documents through Fusion's index pipeline.

You can:

- **Run the crawl now.**

    1. From the Fusion launcher, click **Search** > **Home** > **Datasources**.
    2. Click the datasource.
    3. Click **Start Crawl**.

- **Schedule the crawl** (For Fusion 4.2 through 5.5):

    1. From the Fusion launcher, click **Devops** > **Home** > **Scheduler**.
    2. Click the row for the job that corresponds to the datasource.
    3. Specify schedule information, and then click **Save**.

Configure a Box.com Datasource

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

The Box connector retrieves data from a Box.com cloud-based data repository.

Configuration overview

> **Note**
>
> These steps are for a multi-user Box.com data repository. For limited testing using a single user account, you can *create a Box app that uses Standard OAuth 2.0 authentication*.

Following is an overview of the steps required to set up Box and Managed Fusion, and to crawl a Box data repository.

Set Up Box:

1. Sign up for a Box developer account.
2. Enable 2-step verification.
3. Create a Box app that Managed Fusion can use to crawl the Box files.
4. Configure your app to use a Box service account.

Set Up Managed Fusion:

5. Install Managed Fusion's Box Connector.
6. Create datasources in Managed Fusion that use the Box connector.

Crawl a Box Data Repository:

7. Crawl the Managed Fusion datasources.

Set Up Box

Set up Box so that Managed Fusion can crawl Box data repositories.

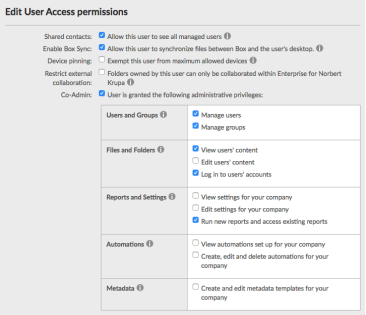
Step 1: Sign Up for a Box Developer Account

If you already have an account, proceed to *Step 2*.

1. Open the Box Developers Console .
2. In the top right corner, click **Sign Up**.
3. Select an appropriate Platform Developer plan.
4. Enter the requested information and click **Submit**.
5. Open the confirmation email and click **Verify Email**.
6. Log in to your Box account.

Step 2: Enable 2-Step Verification

1. Log in to your Box developer account.
    a. Open the Box Developers Console .
    b. Log in as the admin.
2. .Create the Box account that you want to use for crawling.
    a. Open the Users page in the Box admin console.
    b. Click **+Users** to create a new user account.
    c. Enter the Name and Email for the user, and then click **Add user**.
    d. Click the user you just created to enter its user settings.
    e. Make this user a Co-Admin by selecting Co-Admin checkbox. Once clicked, a pane titled "User is granted the following administrative privileges" appears. Select all of the following:
        - Manage users
        - Manage groups
        - View users' content
        - Log in to users' accounts
        - Run new reports and access existing reports



f. Click **Save**.
g. Close the Admin Console browser tab.

3. Enable 2-step verification for unrecognized logins:
   a. Open the Account Settings page. (You can reach this page from the drop-down menu under your initials.)
   b. On the **Account** tab, under **Authentication**, select **Require 2-step verification for unrecognized logins**.
   c. Choose your **Country** and enter a **Mobile Phone Number**, and then click **Continue**.
   d. Enter the verification code you receive, and then click **Continue**.
   e. If you are using a new mobile device, Box will send you a second code. Enter it, and then click **Submit**.
   f. Click **Save Changes**.

Step 3: Create a Box App that Managed Fusion Can Use to Crawl the Box files

Create a Box app that uses OAuth 2.0 with JWT server authentication.

If you already have an app, *configure it*.

1. Open the Box Developers Console .
2. Click **Create New App**.
3. Select **Custom App**, and then click **Next**.
4. Click **OAuth 2.0 with JWT (Server Authentication)**, and then click **Next**.
5. Name your app, and then click **Create App**. The name must be globally unique across all apps created by all Box users.
6. Click **View Your App**.

Step 4: Configure Your App to Use a Box Service Account

1. .Use OpenSSL to create a private/public key pair:
   a. Install OpenSSL if you need to. Windows instructions are here .
   b. Open a Command Prompt window and run these commands to generate a private/public key pair:

```
openssl genrsa -aes128 -out private_key.pem 2048
```

Enter a password for the private key when prompted.

```
openssl rsa -pubout -in private_key.pem -out
public_key.pem
```

In the current directory of the Command Prompt, you now have private and public key files, `private_key.pem` and `public_key.pem` respectively.

2. Open the Box Developers Console, log in as Admin if you are asked to log in, and click your app.
3. .In the left navigation menu, click **Configuration**.
4. Configure scopes and advanced features:
    a. Under **Application Access**, select **Enterprise**.
    b. Under **Application Scopes**, deselect **Manage groups**.
    c. Under **Advanced Features**, enable **Generate User Access Tokens** and **Perform Actions as Users**.
    d. Click **Save Changes**.
5. .In the **Add and Manage Public Keys** area, click **Add a Public Key** and paste the contents of the `public_key.pem` file (generated from the *Box key creation*) into the text box.
    a. Make a note of the new **Public Key ID** that you just created.
6. .Under **OAuth 2.0 Credentials**, click **COPY** for the Client ID.
7. Authorize your app:
    a. Open the Box Admin Console.
    b. In the left navigation menu, click **Settings** > **Enterprise Settings** (or **Business Settings**) > **Apps**.
    c. Under **Custom Applications**, click **Authorize New App**.
    d. In the **API Key** box, paste the Client ID credential you copied in *step 6*, and then click **Next**.
    e. Read the App Authorization dialog and click **Authorize**.
    f. Close the Admin Console browser tab.

> **Note**
>
> If you change your app's configuration later, you must repeat this step to re-authorize your app.

8. Close the Dev Console browser tab.

Set Up Managed Fusion

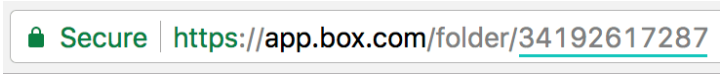
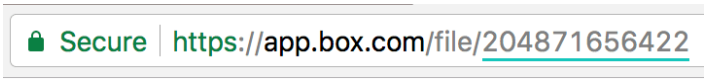
Set up Managed Fusion to crawl Box data repositories.

Step 5: Create Datasources

Create datasources that use the Box connector to access the Box data repository.

For each datasource:

1. In the Managed Fusion UI, Navigate to **Indexing** > **Datasources**.
2. Click **Add**.
3. Select **Box (V2)** under *Not Installed*.
4. Select **Box (V2)** again under *Installed*.
5. Fill in the form. Note the following regarding configuration settings to use:

Setting	Notes
Start Links	Each start link defined for the datasource must consist of a numeric Box file ID or directory ID. The root directory of any Box account has ID 0 (zero). To crawl your entire Box repository, enter '0'. These images indicate with underlines where you can get a folder ID or file ID. Select a folder or file at Box.com . Folder ID:  Enter the start link `34192617287` . File ID:  Enter the start link `204871656422` :
API Key	In the Box Developers Console , select the app. On the **Configuration** tab under **OAuth 2.0 Credentials**, use the **Client ID**.
API Secret	In the Box Developers Console , select the app. On the **Configuration** tab under **OAuth 2.0 Credentials**, use the **Client Secret**.
JWT App User ID	Email address that you use to sign in to your Box co-admin account. Use the *Co-admin account you created earlier* for this.
JWT Public Key Id	In the Box Developers Console , select the app. On the **Configuration** tab, under **Add and Manage Public Keys**, use the **ID** for a public key.
JWT Private Key	Base64-encoded contents of the *private-key file* that matches your JWT Public Key Id. Base64 encode the *entire* contents of the file, including the first and last lines.
JWT Private Key Password	Passphrase for the private key (from the private-key file you created during *Box key creation*).
Distributed crawl collection name	Collection that contains the *pre-fetch index*.
Box.com children responses per page	Use the default value of 1000.
Nested folder depth limit	Generally, you want a number to crawl all documents, so keep the default value. For testing, you could reduce the number substantially to speed up the crawl.
Number of partition buckets	Divide the number of files by 5000. Use that number or 10000, whichever is smaller.

Setting	Notes
Number distributed crawl datasources	Use 1 to 27.
Number of pre-fetch index creator threads	A number between 2 and 5. Use 2 for small datasources and 5 for huge datasources (over 10 million files).

6. Click **Save**.

Crawl a Box Data Repository

Crawl a Box data repository.

Step 6: Crawl the Managed Fusion Datasources

Crawl the datasources, which use Managed Fusion's Box connector to access the Box data repository. Managed Fusion's Box connector uses the pre-fetch index to fetch the contents of each file from Box.com, get metadata from both the distributed index and Box.com, and index the documents through Managed Fusion's index pipeline.

You can:

- **Run the crawl now.**

  1. From the Managed Fusion launcher, click **Search** > **Home** > **Datasources**.
  2. Click the datasource.
  3. Click **Start Crawl**.

Configure a Data Model

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion*

Data models simplify the process of getting started with Fusion by providing pre-configured objects to reduce the effort spent on basic starting tasks. This helps keep documents consistent between datasources and intuitive to the object's type.

See the *Data Models* topic for more information.

1. Configure the datasource

Important

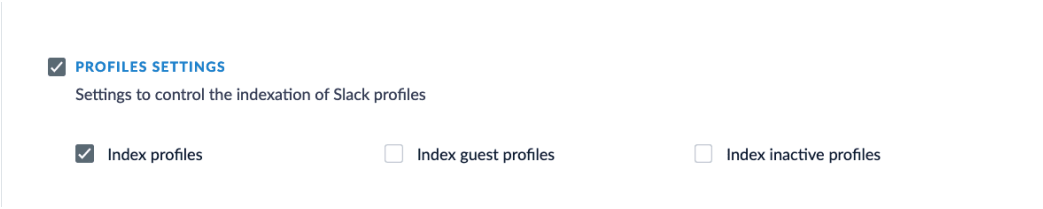
This section references the Slack V2 Connector, which was introduced in Fusion 5.4.0. However, Data Models were *introduced in Fusion 5.3.0*. The instructions on *configuring the data model* are valid for any connector in Fusion 5.3.0+ that indexes users.

1. Navigate to **Indexing > Datasources**.
2. Click the **Add** button, and choose the Slack V2 connector from the list.

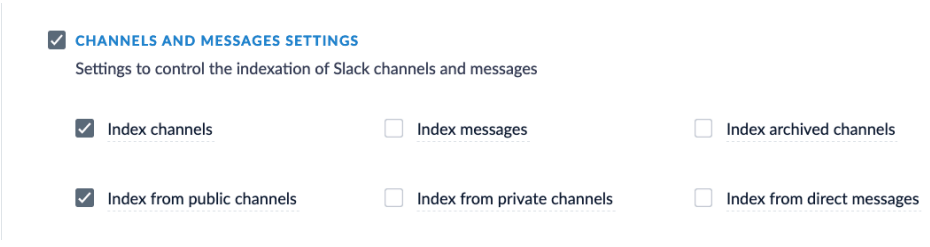
Note

Some connectors include built-in data models as a standard component. Others require you to manually create data models.

3. Complete the datasource configuration:
   - Ideally, use a pipeline specifically created for this data model. For now, you can create the pipeline by navigating to **Indexing > Index Pipelines** and clicking the **Add** button. Configure it later as *described below*.
   - Under **Profiles Settings**, select *Index profiles*:



   - Under **Channels and Messages Settings**, select *Index channels* and *Index from public channels*:



4. Click **Save**.
5. Click **Run** to run the indexing job.

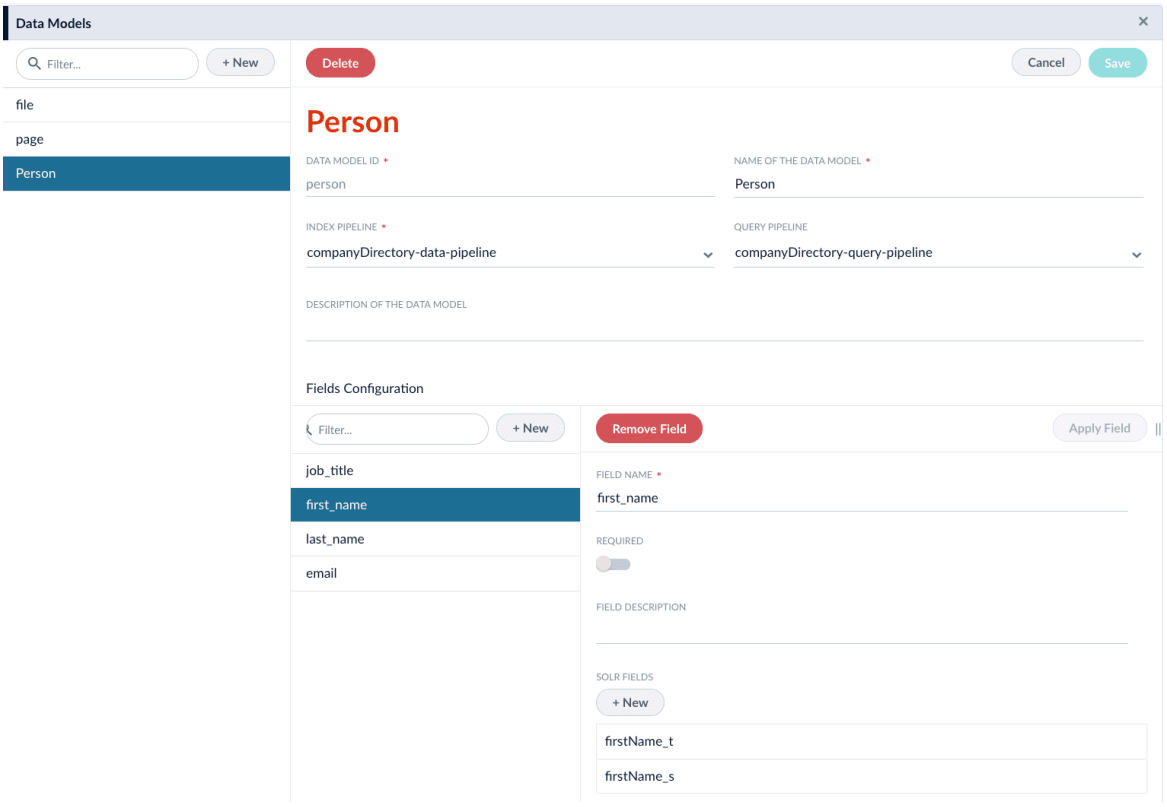
2. Configure the data model

Some connectors include built-in data models with pre-configured object types. However, you can add new data models or customize existing ones to fit your particular needs.

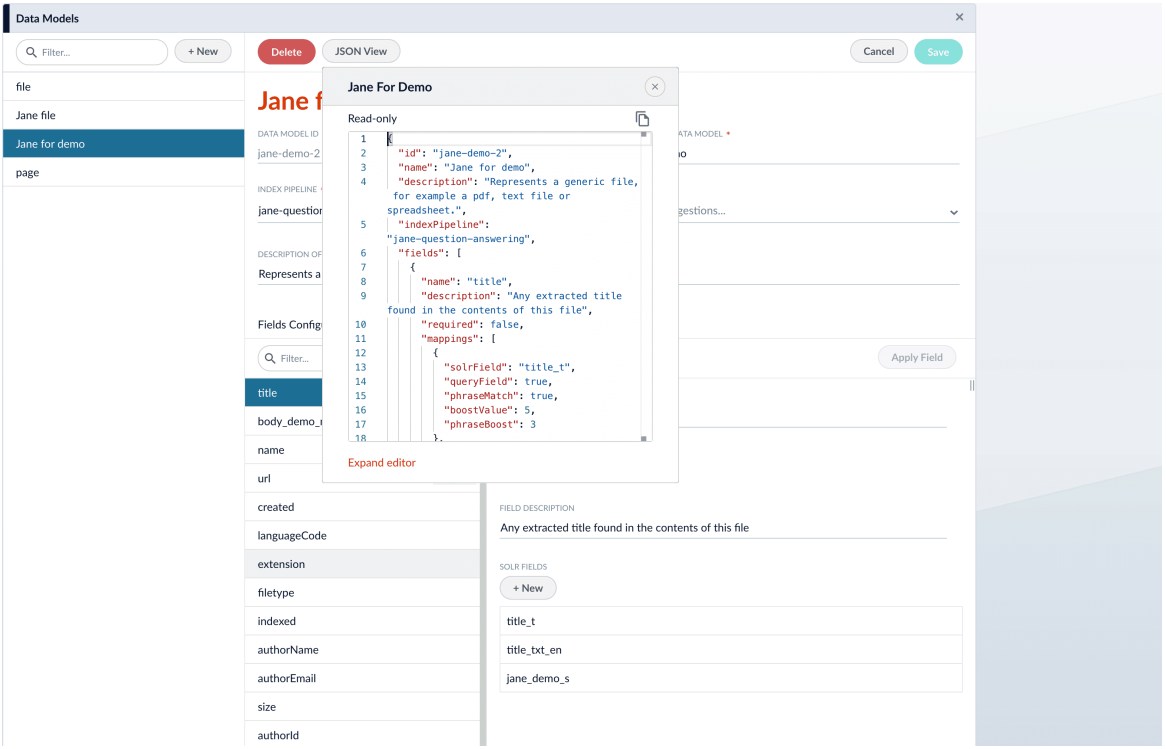
1. Navigate to **Indexing > Data Models**.
2. Click the **Add** button.

3. Create a new data model, `person`.
4. Assign the index pipeline for the data model. In this example, create a new index pipeline named `companyDirectory-data-pipeline`.
   This pipeline is ideally for operation with data models only. For the sake of this example, we will only be using the default stages. Applying additional logic/stages specific to the object would occur here.
5. Assign the query pipeline you will use to query the indexed documents. In this example, create a new query pipeline named `companyDirectory-query-pipeline`.
6. Click the **New** button under **Fields Configuration**.
7. Complete the required configurations, as detailed in *Data Models API Reference*. Create fields for the following:

Field Name	Solr Fields
`first_name`	`firstName_t`, `firstName_s`
`last_name`	`lastName_t`, `lastName_s`
`email`	`email_s`
`job_title`	`jobTitle_t`, `jobTitle_s`



8. If needed, edit the JSON for the data model.



9. Click the **Save** button to save the data model.

3. Configure the index pipeline

To begin, navigate to **Indexing > Index Workbench**. Alternatively, the `companyDirectory-index-pipeline` pipeline can be configured in **Indexing > Index Pipelines**, but you are not able to preview results.

Important

The raw data fields in your index coming from Slack may differ from the example data fields used in this article.

3.1. Data Model Mapping stage

1. Click the **Add a Stage** button, and choose **Data Model Mapping** from the list.
2. Use the **Data Model Type** dropdown to select the `person` data model.
3. *(optional)* Check the **Match Trigger** checkbox and assign the following values:

Field	Value
Field to match	`type_s`
Value to match	`user`

Tip

The **Value to match** field supports RegEx. You can assign multiple values. Alternatively, you can create additional **Data Model Mapping** stages.

4. Assign field mappings for the Slack datasource's raw datasources:

Source Field	Data Model Field
`first_name`	`first_name`
`last_name`	`last_name`
`email`	`email`

5. *(optional)* Check the **Keep unmapped fields** checkbox to preserve fields that are unmatched to the data model.
6. Click **Apply**.

3.2. Call Data Model Pipeline stage

Important

This stage must be placed *after* the Data Model Mapping stage.

1. Click the **Add a Stage** button, and choose **Call Data Model Pipeline** from the list.
2. Assign the value `_lw_data_model_type_s` to the **Data Model Type Field** field. This field is created when a document meets the criteria specified in the **Data Model Mapping**.
3. Click **Apply**.

Next steps

With the index pipeline configured, you are now ready to complete the indexing job by clicking the **Start Job** button.

Once complete, the documents are ready for viewing in the Query Workbench:

1. Navigate to **Querying > Query Workbench**.
2. Click the **Load** button.
3. Choose the query pipeline you specified when creating the data model, `companyDirectory-query-pipeline`.

More information

- *Data Models*
- *Data Models API reference*
- *Data Model Mapping Index Stage reference*
- *Call Data Model Pipeline Index Stage reference*

Configure a Data Model

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

Data models simplify the process of getting started with Fusion by providing pre-configured objects to reduce the effort spent on basic starting tasks. This helps keep documents consistent between datasources and intuitive to the object's type.

See the *Data Models* topic for more information.

1. Configure the datasource

Important

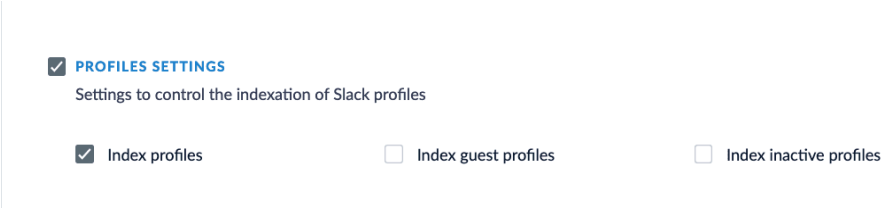
This section references the Slack V2 Connector as an example. The instructions on *configuring the data model* are valid for any connector that indexes users.

1. Navigate to **Indexing > Datasources**.
2. Click the **Add** button, and choose the Slack V2 connector from the list.

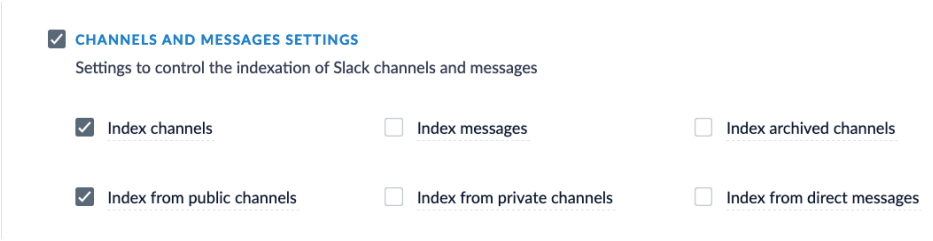
Note

Some connectors include built-in data models as a standard component. Others require you to manually create data models.

3. Complete the datasource configuration:
   - Ideally, use a pipeline specifically created for this data model. For now, you can create the pipeline by navigating to **Indexing > Index Pipelines** and clicking the **Add** button. Configure it later as *described below*.
   - Under **Profiles Settings**, select *Index profiles*:



   - Under **Channels and Messages Settings**, select *Index channels* and *Index from public channels*:



4. Click **Save**.
5. Click **Run** to run the indexing job.

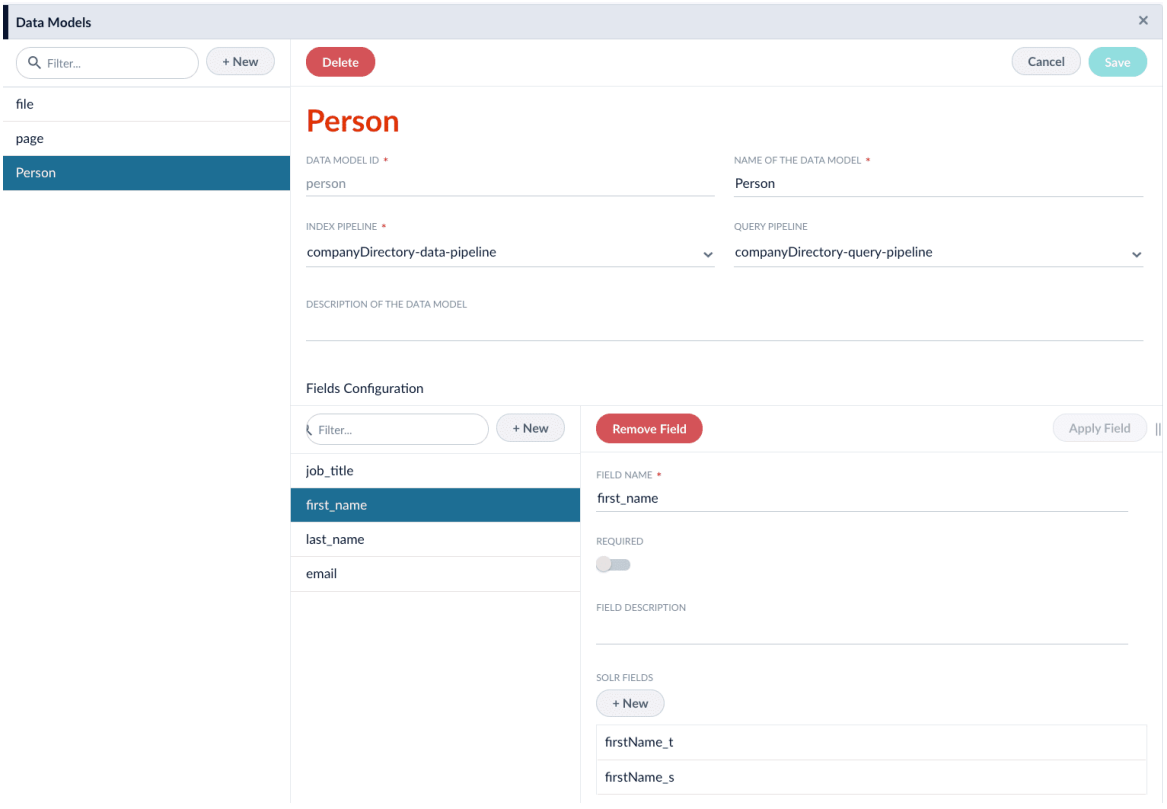
2. Configure the data model

Some connectors include built-in data models with pre-configured object types. However, you can add new data models or customize existing ones to fit your particular needs.

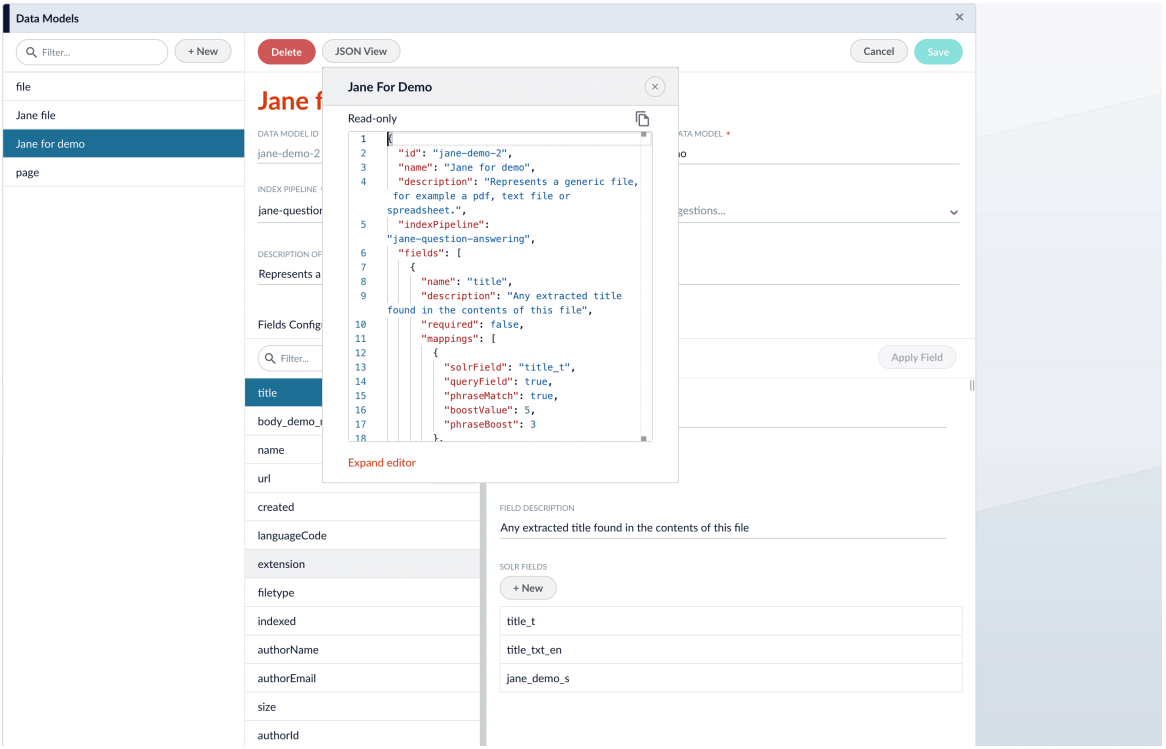
1. Navigate to **Indexing > Data Models**.
2. Click the **Add** button.
3. Create a new data model, `person`.

4. Assign the index pipeline for the data model. In this example, create a new index pipeline named `companyDirectory-data-pipeline`.
   This pipeline is ideally for operation with data models only. For the sake of this example, we will only be using the default stages. Applying additional logic/stages specific to the object would occur here.
5. Assign the query pipeline you will use to query the indexed documents. In this example, create a new query pipeline named `companyDirectory-query-pipeline`.
6. Click the **New** button under **Fields Configuration**.
7. Complete the required configurations, as detailed in *Data Models API Reference*. Create fields for the following:

Field Name	Solr Fields
`first_name`	`firstName_t`, `firstName_s`
`last_name`	`lastName_t`, `lastName_s`
`email`	`email_s`
`job_title`	`jobTitle_t`, `jobTitle_s`



8. If needed, edit the JSON for the data model.



9. Click the **Save** button to save the data model.

3. Configure the index pipeline

To begin, navigate to **Indexing > Index Workbench**. Alternatively, the `companyDirectory-index-pipeline` pipeline can be configured in **Indexing > Index Pipelines**, but you are not able to preview results.

Important

The raw data fields in your index coming from Slack may differ from the example data fields used in this article.

3.1. Data Model Mapping stage

1. Click the **Add a Stage** button, and choose **Data Model Mapping** from the list.
2. Use the **Data Model Type** dropdown to select the `person` data model.
3. *(optional)* Check the **Match Trigger** checkbox and assign the following values:

Field	Value
Field to match	`type_s`
Value to match	`user`

Tip

The **Value to match** field supports RegEx. You can assign multiple values. Alternatively, you can create additional **Data Model Mapping** stages.

4. Assign field mappings for the Slack datasource's raw datasources:

Source Field	Data Model Field
`first_name`	`first_name`
`last_name`	`last_name`
`email`	`email`

5. *(optional)* Check the **Keep unmapped fields** checkbox to preserve fields that are unmatched to the data model.
6. Click **Apply**.

3.2. Call Data Model Pipeline stage

Important

This stage must be placed *after* the Data Model Mapping stage.

1. Click the **Add a Stage** button, and choose **Call Data Model Pipeline** from the list.
2. Assign the value `_lw_data_model_type_s` to the **Data Model Type Field** field. This field is created when a document meets the criteria specified in the **Data Model Mapping**.
3. Click **Apply**.

Next steps

With the index pipeline configured, you are now ready to complete the indexing job by clicking the **Start Job** button.

Once complete, the documents are ready for viewing in the Query Workbench:

1. Navigate to **Querying > Query Workbench**.
2. Click the **Load** button.
3. Choose the query pipeline you specified when creating the data model, `companyDirectory-query-pipeline`.

More information

- *Data Models*
- *Data Models API reference*
- *Data Model Mapping Index Stage reference*
- *Call Data Model Pipeline Index Stage reference*

Configure A Lucidworks AI Gateway Integration

Lucidworks AI

How-to

> Compatible versions: 5.9
>
> Works with: Lucidworks AI

Before you can use Lucidworks AI with Lucidworks Platform, you must configure the *Lucidworks AI Gateway* to provide a secure, authenticated integration between self-hosted Fusion and your hosted models. This configuration is done through a secret properties file that you can find in the Lucidworks Platform UI.

Note

This feature is only available in Fusion 5.9.5 and later versions of Fusion 5.9.

Integrations are created for you by the Lucidworks team. But as a workspace owner, you can configure those integrations with Lucidworks AI Gateway. Each account can have its own set of credentials and associated scopes, which define the operations it can perform. If configuration properties are not provided at the account level, default settings are used instead.

To configure the Lucidworks AI Gateway, navigate to the megamenu and click **Models**.

1. On the Integrations tab, click your integration. If you don't see your integration, contact your Lucidworks representative.
2. Download or copy the YAML code and paste it into a file called `account.yaml`.
   The file for a single integration should look similar to this one:

```
lwai-gateway: lwai: credentials: |
     fusion.lwai.default.baseUrl:
https://APPLICATION_ID.applications.lucidworks.com
     fusion.lwai.default.authEndpoint:
https://identity.lucidworks.com/oauth2/XXXXXXXXXX/v1/token
     fusion.lwai.account[0].name: ACCOUNT_NAME
     fusion.lwai.account[0].scopes:
machinelearning.predict,machinelearning.model
     fusion.lwai.account[0].clientId: *****
     fusion.lwai.account[0].clientSecret: *****
```

   For a configuration with multiple integrations, it should look like this:

```
lwai-gateway: lwai: credentials: |
     fusion.lwai.default.authEndpoint:
https://identity.lucidworks.com/oauth2/XXXXXXXXXX/v1/token
     fusion.lwai.account[0].baseUrl:
https://APPLICATION_ID.applications.lucidworks.com
     fusion.lwai.account[0].name: ACCOUNT_NAME
     fusion.lwai.account[0].scopes:
machinelearning.predict,machinelearning.model
     fusion.lwai.account[0].clientId: *****
     fusion.lwai.account[0].clientSecret: *****
fusion.lwai.account[1].baseUrl:
https://APPLICATION_ID2.applications.lucidworks.com
fusion.lwai.account[1].name: ACCOUNT_NAME
fusion.lwai.account[1].scopes:
machinelearning.predict,machinelearning.model
fusion.lwai.account[1].clientId: *****
fusion.lwai.account[1].clientSecret: *****
```

Note

Non-admin users must have the following permissions to use Lucidworks AI integrations: `PUT,POST,GET:/LWAI-ACCOUNT-NAME/**` where `LWAI-ACCOUNT-NAME` must match the value of `fusion.lwai.account[n].name` in the integration YAML.

3. Apply the file to your Fusion configuration file. For example:

```
helm upgrade KUBERNETES_NAMESPACE lucidworks/fusion -f
FUSION_VALUES.yaml
```

Tip

You can confirm that the integration is working by *configuring Neural Hybrid Search*.

Configure a new datasource

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

Add the datasource and connector

1. Sign in to Managed Fusion and click any application.
2. Click **Indexing > Datasources > Add+**.
3. Select your connector.
   The connector configuration panel displays. The specific configuration options vary depending on the connector.

Configure the connector

Note

If you do not see your connector in the list, you may need to install it. For more information, see *Install a connector*.

To configure the connector:

1. Enter a useful name for your datasource in the **Datasource ID** field.
2. Select an option in the **Pipeline ID** field if different from the default.
3. Select an option in the **Parser** fields if different from the default.
4. Select your specific release and connector detail. For more information, see *Connectors Configuration Reference*.
5. Click **Save**.

Test the datasource configuration

1. Click **Indexing > Index Workbench > Load**.
2. Select the datasource ID you specified when you created the datasource.
3. Review the datasource configuration and a simulation of the results when you run this datasource job to index your data.
4. Adjust the configurations of your datasource, parsers, and index pipeline until the simulated results are satisfactory.
5. Click **Save**.

Index your data

1. In the Index Workbench, click **Start Job**.
2. When the job status is Finished, click **Querying > Query Workbench** to view the indexed documents and configure your query pipeline. For more information, see *Query Workbench*.

Configure a New Datasource

How-to

> Compatible versions: 4.2, 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion Connectors*, *Fusion Server*, *Fusion*

1. In your Fusion app, navigate to **Indexing** > **Datasources**.
2. Click **Add**.
3. Select your connector.
   The connector configuration panel appears. The specific configuration options vary depending on the connector.

If you do not see your connector in the list, you may need to install it.

1. Configure your connector's options:
   a. Enter a useful name for your datasource in the **Datasource ID** field.
   b. You can leave the **Pipeline ID** and **Parser** fields as-is to start with the defaults, or select different ones if you have them.
   c. Configure the connector-specific options.
      Select your specific release and connector detail from the *configuration reference topic*.
2. Click **Save**.
3. Test your datasource configuration:
   a. Navigate to the *Index Workbench* at **Indexing** > **Index Workbench**.
   b. Click **Load...**.
   c. Select the datasource ID you specified when you created the datasource.
      Now you can see your datasource configuration and a simulation of the results you can expect when you run this datasource job to index your data.
   d. Adjust the configurations of your datasource, parsers, and index pipeline until the simulated results are satisfactory.
   e. Click **Save**.
4. Index your data:
   a. In the Index Workbench, click **Start Job**.
   b. When the job status is **Finished**, navigate to the *Query Workbench* at **Querying** > **Query Workbench** to view the indexed documents and configure your query pipeline.

Configure a non-administrative crawl account in SharePoint Online

How-to

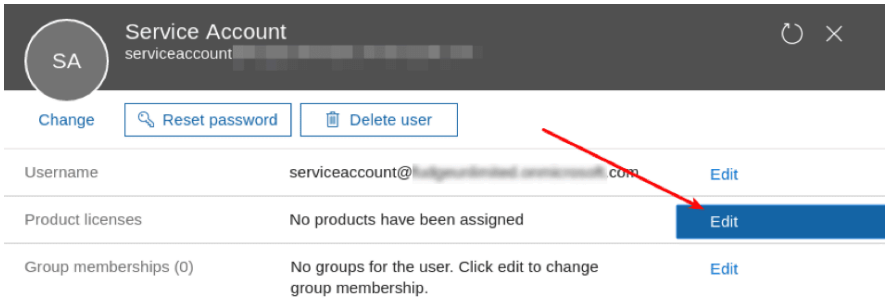
> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

The SharePoint connector retrieves content and metadata from an on-premises SharePoint repository.

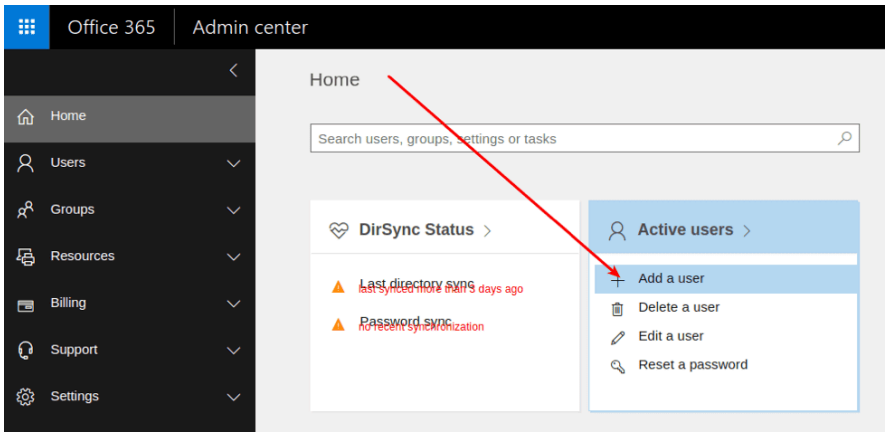
The steps below describe how to configure a crawl account in SharePoint Online without giving the account administrative access.

1. Create a Service Account

1. Log in as a SharePoint administrator, and go to your admin center.
2. If you are using an on-premise active directory synced to SharePoint Online, you need to create an active directory account and license the active directory account on SharePoint Online.



1. If you are using SharePoint Online user accounts, add a user named “Lucidworks Fusion Service Account”.



Create the account as **User (no administrator access)**.

2. Add a Crawl Permissions Level

To create a new permission level, click the gear symbol and go to **Site Settings > Site permissions**. Select **Permission Levels**, and click **Add a Permission Level**. Name the new permission level "Lucidworks Fusion Service Permission", and assign the following site permissions:

Name	Description
View Items	View items in lists and documents in document libraries.
Open Items	View the source of documents with server-side file handlers.
View Versions	View past versions of a list item or document.
View Application Pages	View forms, views, and application pages. Enumerate lists.
View Web Analytics Data	View reports on Web site usage.
Browse Directories	Enumerate files and folders in a Web site using SharePoint Designer and Web DAV interfaces.
View Pages	View pages in a Web site.
Enumerate Permissions	Enumerate permissions on the Web site, list, folder, document, or list item.
Browse User Information	View information about users of the Web site.
Use Remote Interfaces	Use SOAP, Web DAV, the Client Object Model or SharePoint Designer interfaces to access the Web site.
Open	Allows users to open a Web site, list, or folder in order to access items inside that container.
Edit Personal User Information	Allows a user to change his or her own user information, such as adding a picture.

3. Create a Fusion crawl group

For ***each*** top-level site you want to be able to crawl, you must create a site permissions group and assign the permissions level you created previously. Go to **Site Settings > Site permissions**. Click the **Create Group** symbol and name the new group "Lucidworks Fusion Crawl Accounts". Add the “Lucidworks Fusion Service Account” user, and any other user that you wish to have crawl permissions, to this group.

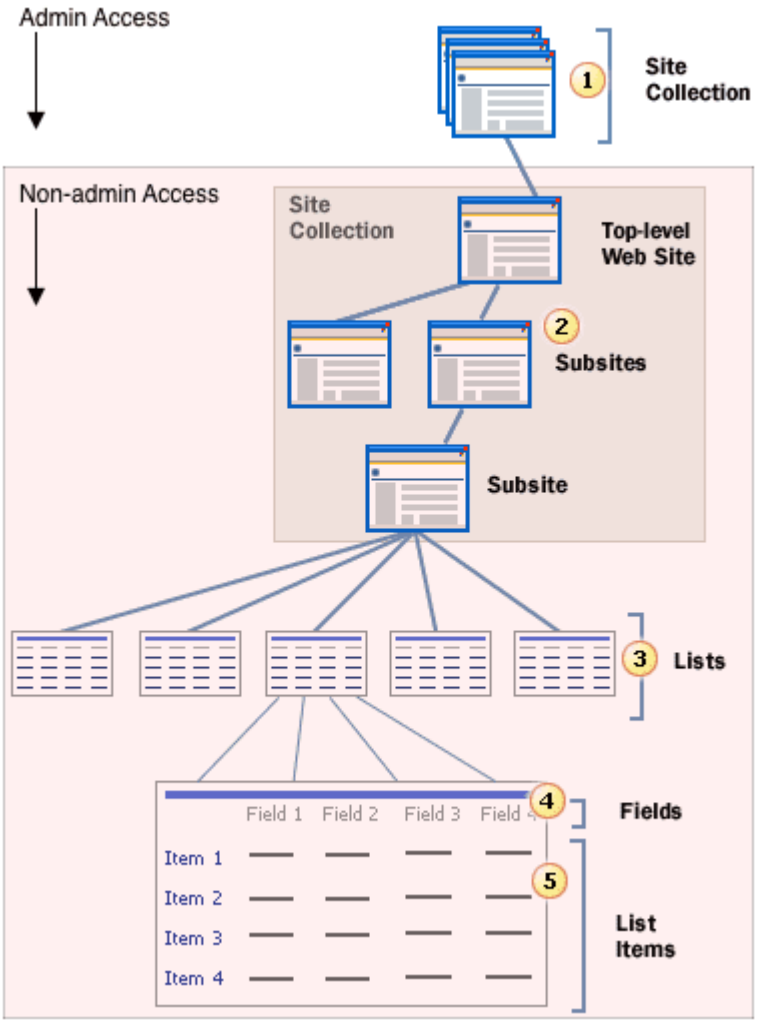
The “Lucidworks Fusion Service Account” user should now be able to crawl without administrator rights.

Limitations of a non-administrative crawl account in SharePoint Online

There are important limitations to crawling SharePoint Online with a non-administrative account. Only administrators are permitted to list site collections from SharePoint Online. To crawl multiple site collections from your SharePoint Online tenant, you must either:

1. List the site collections in the *Start Links* explicitly, **or;**
2. Provide a SharePoint administrator account when crawling SharePoint Online

The image below illustrates what information a non-administrator user can crawl:



Note

Although a non-administrator user can be allowed to list sub-sites in a site collection, the user cannot list the site collections of the tenant URL. For example, a non-administrator user may list the Sub-sites in `https://lucidworks.sharepoint.com/sites/sitecol`, such as `/sitecol/subsite1` and `/sitecol/subsite2`. However, only an administrator can list the site collections in https://lucidworks.sharepoint.com.

Configure a Non-Administrative Crawl Account in SharePoint Online

How-to

> Compatible versions: 4.2, 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion Connectors*, *Fusion Server*, *Fusion*

The SharePoint connector retrieves content and metadata from an on-premises SharePoint repository.

The steps below describe how to configure a crawl account in SharePoint Online without giving the account administrative access.

1. Create a site collection administrator account

To create a site collection administrator account, you must have administrator permissions for the site you want to crawl.

1. Log in to the SharePoint Online admin center as a SharePoint administrator.
2. Go to **Sites** > **Active sites** and select the site you want to crawl.
3. Go to **Membership** > **Site Admins** and click **Add site admin**.
4. Search for the user you want to designate as a site admin, then click **Add**.

The user is now listed as a site admin.

Note

The site collection administrator only has control over the selected site and its sub-sites.

2. Add a crawl permissions level

To create a new permission level, click the gear symbol and go to **Site Settings > Site permissions**. Select **Permission Levels**, and click **Add a Permission Level**. Name the new permission level "Lucidworks Fusion Service Permission", and assign the following site permissions:

Name	Description
View Items	View items in lists and documents in document libraries.
Open Items	View the source of documents with server-side file handlers.
View Versions	View past versions of a list item or document.
View Application Pages	View forms, views, and application pages. Enumerate lists.
View Web Analytics Data	View reports on Web site usage.
Browse Directories	Enumerate files and folders in a Web site using SharePoint Designer and Web DAV interfaces.
View Pages	View pages in a Web site.
Enumerate Permissions	Enumerate permissions on the Web site, list, folder, document, or list item.
Browse User Information	View information about users of the Web site.
Use Remote Interfaces	Use SOAP, Web DAV, the Client Object Model or SharePoint Designer interfaces to access the Web site.
Open	Allows users to open a Web site, list, or folder in order to access items inside that container.
Edit Personal User Information	Allows a user to change his or her own user information, such as adding a picture.

3. Create a Fusion crawl group

For ***each*** top-level site you want to be able to crawl, you must create a site permissions group and assign the permissions level you created previously. Go to **Site Settings > Site permissions**. Click the **Create Group** symbol and name the new group "Lucidworks Fusion Crawl Accounts". Add the site collection administrator, and any other user that you wish to have crawl permissions, to this group.

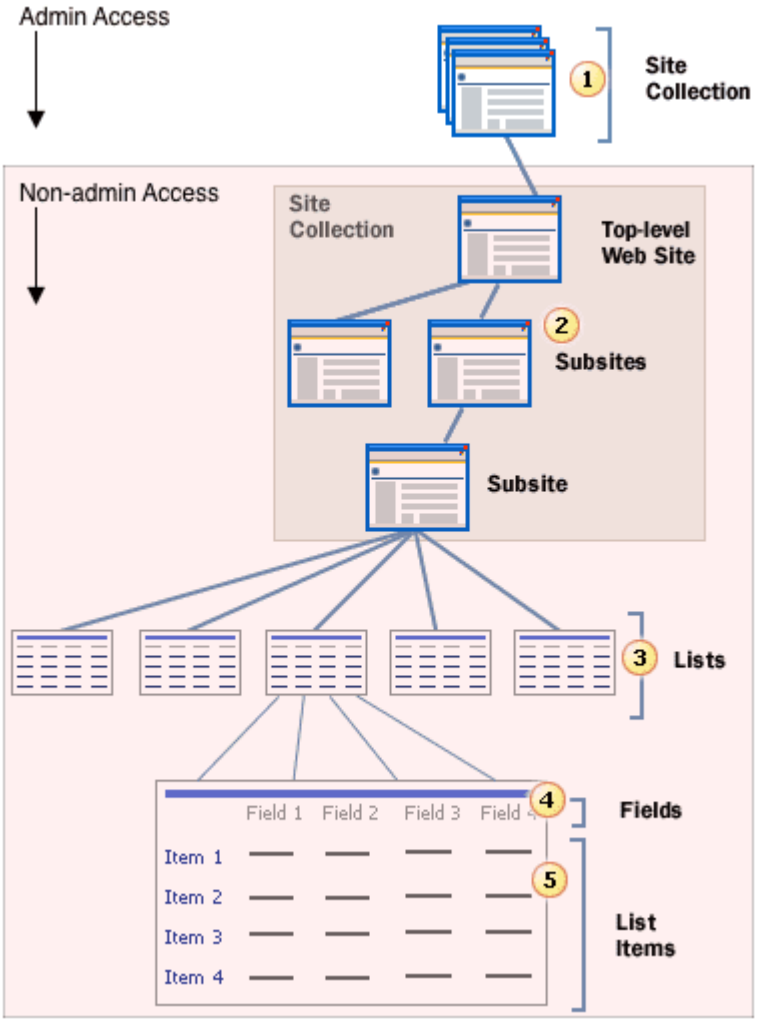
The site collection administrator should now be able to crawl without administrator rights.

Limitations of a non-administrative crawl account in SharePoint Online

There are important limitations to crawling SharePoint Online with a non-administrative account. Only administrators are permitted to list site collections from SharePoint Online. To crawl multiple site collections from your SharePoint Online tenant, you must either:

1. List the site collections in the *Start Links* explicitly, **or;**
2. Provide a SharePoint administrator account when crawling SharePoint Online

The image below illustrates what information a non-administrator user can crawl:



Note

Although a non-administrator user can be allowed to list sub-sites in a site collection, the user cannot list the site collections of the tenant URL. For example, a non-administrator user may list the Sub-sites in `https://lucidworks.sharepoint.com/sites/sitecol`, such as `/sitecol/subsite1` and `/sitecol/subsite2`. However, only an administrator can list the site collections in **https://lucidworks.sharepoint.com**.

Configure a REST Call Job to Delete Old Signals

How-to

> Compatible versions: 4.2, 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion AI*, *Fusion Server*, *Fusion*

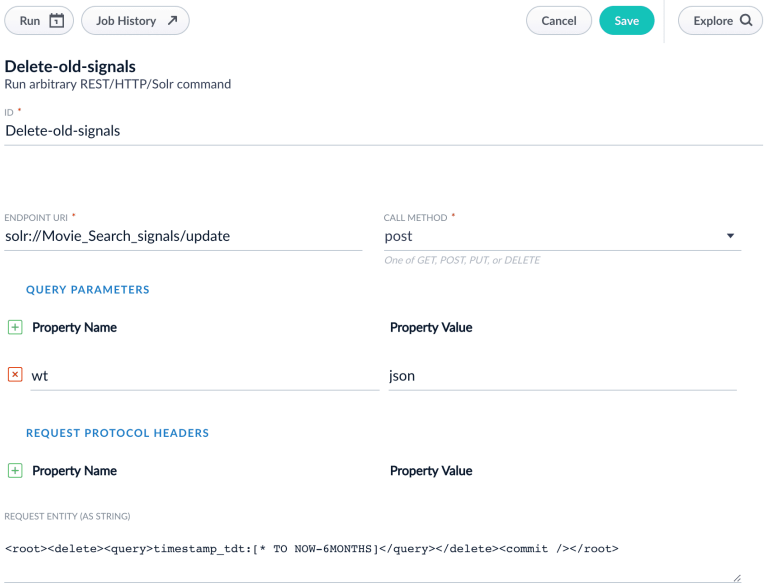
1. Navigate to **Collections** > **Jobs**.
2. Click **Add** and select **REST Call**.
   The REST Call job configuration panel appears.
3. Enter an arbitrary ID for this job, such as "Delete-old-signals".
4. Enter the following endpoint URI, substituting the name of your signals collection for `signalsCollectionName`:

```
solr://signalsCollectionName/update
```

5. In the **Call Method** field, select "post".
6. Under **Query Parameters**, enter the property name "wt" with the property value "json".
7. In the **Request entity (as string)** field, enter the following:

```
<root><delete><query>timestamp_tdt:[* TO NOW-2WEEKS]</query>
</delete><commit /></root>
```

See Working with Dates for details about date formatting.
Your job configuration should look similar to this:



Tip

You can configure a schedule for this job at **System** > **Scheduler**.

Configure a REST Call Job to Delete Old Signals

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

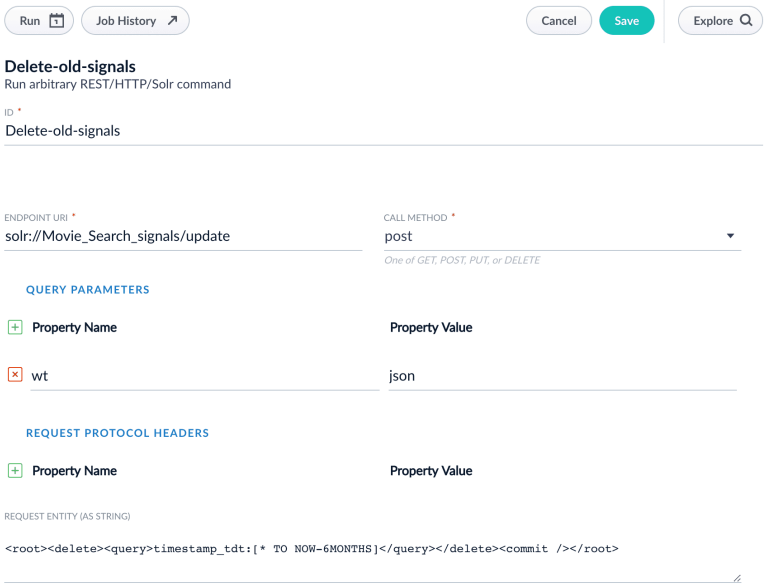
1. Navigate to **Collections** > **Jobs**.
2. Click **Add** and select **REST Call**.
   The REST Call job configuration panel appears.
3. Enter an arbitrary ID for this job, such as "Delete-old-signals".
4. Enter the following endpoint URI, substituting the name of your signals collection for `signalsCollectionName`:

```
solr://signalsCollectionName/update
```

5. In the **Call Method** field, select "post".
6. Under **Query Parameters**, enter the property name "wt" with the property value "json".
7. In the **Request entity (as string)** field, enter the following:

```
<root><delete><query>timestamp_tdt:[* TO NOW-2WEEKS]</query>
</delete><commit /></root>
```

See Working with Dates for details about date formatting.
Your job configuration should look similar to this:



Tip

You can configure a schedule for this job at **System** > **Scheduler**.

Configure a SharePoint Online V1 Optimized Datasource

How-to

> Works with: *Fusion Connectors*

The SharePoint Online V1 Optimized connector retrieves data from cloud-based SharePoint repositories. Authentication requires a Sharepoint user who has permissions to access Sharepoint via the SOAP API. This user must be registered with the Sharepoint Online authentication server; it is not necessarily the same as the user in Active Directory or LDAP.

Important

Before you configure the datasource, it is imperative you understand the information detailed in *SharePoint and SharePoint Online Connectors* for the release you are running.

1. Decide what you need to crawl

The first and most important thing to do is determine what you are trying to crawl, and to pick your “Start Links” accordingly.

Choose one of the following:

- *An entire SharePoint Web application* (all site collections in a specific SharePoint URL).
- *A subset of SharePoint site collections.*
- *A specific sub-site, list, or list item.*

How to crawl an entire SharePoint Web application

1. Leave the **Limit Documents > Fetch all site collections** option checked (as it is by default).
2. Specify the Web application URL as a site.
   For example: `https://lucidworks.sharepoint.local/`

Note

Crawling an entire SharePoint Web application requires administrative access to SharePoint.

How to crawl a subset of SharePoint site collections

1. Uncheck the **Limit Documents > Fetch all site collections** option.
2. Specify a "Start Link" for each site collection that you want to crawl.
   Examples: `https://lucidworks.sharepoint.local/sites/site1`, `https://lucidworks.sharepoint.local/sites/site2`, `https://lucidworks.sharepoint.local/sites/site3`

How to crawl a specific sub-site, list, or list item:

1. Uncheck the **Limit Documents > Fetch all site collections** option.
2. Specify a "Start Link" for each site collection that contains the item you want to fetch.
3. Specify a non-wildcard **Inclusive Regular Expression** for each parent.
   For example, if you want to crawl `https://lucidworks.sharepoint.local/sites/mysitecol/myparentsite/somesite` then you must include inclusive regexes for all parents along the way:

```
https\:\/\/lucidworks\.sharepoint\.local\/sites\/mysitecol
https\:\/\/lucidworks\.sharepoint\.local\/sites\/mysitecol\/mypa
rentsite
https\:\/\/lucidworks\.sharepoint\.local\/sites\/mysitecol\/some
site
https\:\/\/lucidworks\.sharepoint\.local\/sites\/mysitecol\/some
site\/.*
```

Important

If you exclude a parent item of the site, the connector will not crawl the site because it will never spider down to it during the crawl process.

2. Choose an authentication method

With the SharePoint Online connector, you have several options for authenticating.

1. **SharePoint service account** - This method is the equivalent of logging in under a user account. If the account is not an admin user, however, you need to grant the account access to each site collection you want to crawl.
   Required parameters:
   - Username
   - Password
   - Tenant

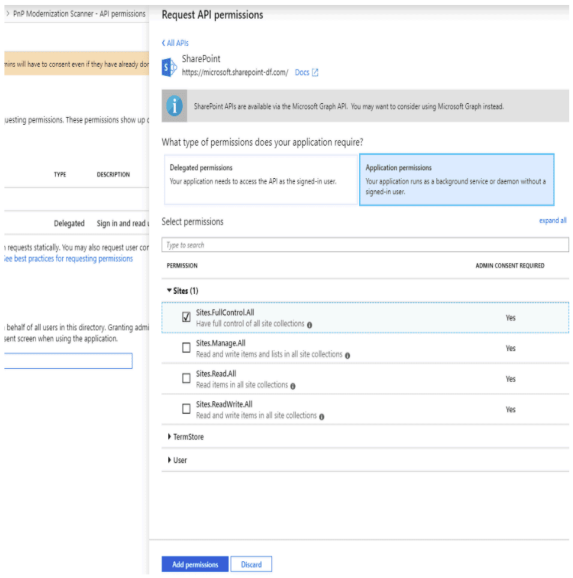
2. **App-only authentication using Azure AD with a private key** - Using this method requires the application key to have "Full Control" permissions. Otherwise, security authorization errors occur while crawling. *This is the most commonly used method.*
   Required parameters:
   - Client ID
   - PFX key in Base64 format
   - PFX key password
   - Tenant

   Set the following permission to crawl SharePoint:
   a. Click **API permissions** in the left panel.
   b. Navigate to **Application permissions > Sites**.
   c. Select the **Sites.FullControl.All** check box.
   d. Click **Add permissions** to save the changes.

Note

If the Sites.FullControl.All option is not selected, **403 errors** are generated while crawling.

For detailed information about configuration, see Granting access via Azure AD App-Only.



3. **App-only authentication using Azure AD with OAuth** - Using this method requires the application key to have "Full Control" permissions. Otherwise, security authorization errors occur while crawling. *This is the least commonly used method.*
   Required parameters:
   - Client ID
   - Client secret
   - Tenant

   You have two options here:
   - *Set up a crawl account* with only as much permission as it needs.
     This approach has the security advantage of providing minimal access to Fusion. However, the crawl account cannot retrieve the list of site collections behind a Web application URL. It cannot access the SharePoint Tenant Admin API to list all the site collections on your tenant. If you use this authentication method, you must enter each site collection to crawl in **Start Links**.
   - *Provide administrative access to crawl*

   Grant the App the following permission to crawl SharePoint:

```
<AppPermissionRequests AllowAppOnlyPolicy="true">
<AppPermissionRequest Scope="http://sharepoint/content/tenant"
Right="FullControl" /> </AppPermissionRequests>
```

Note

If the Right field is not set to FullControl, **403 errors** are generated while crawling.

For detailed information, see Granting access using SharePoint App-Only .

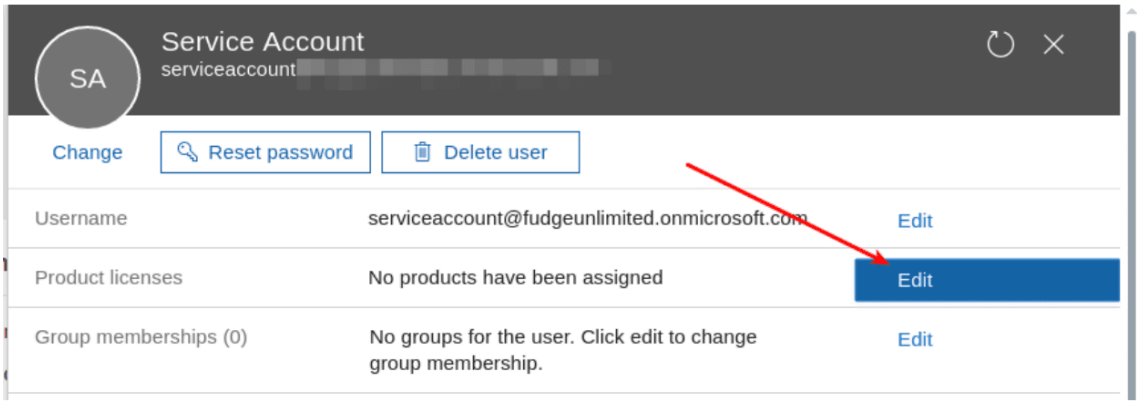
How to set up a crawl account

1. CREATE A SERVICE ACCOUNT AND LICENSE THE ACCOUNT (IF NEEDED)

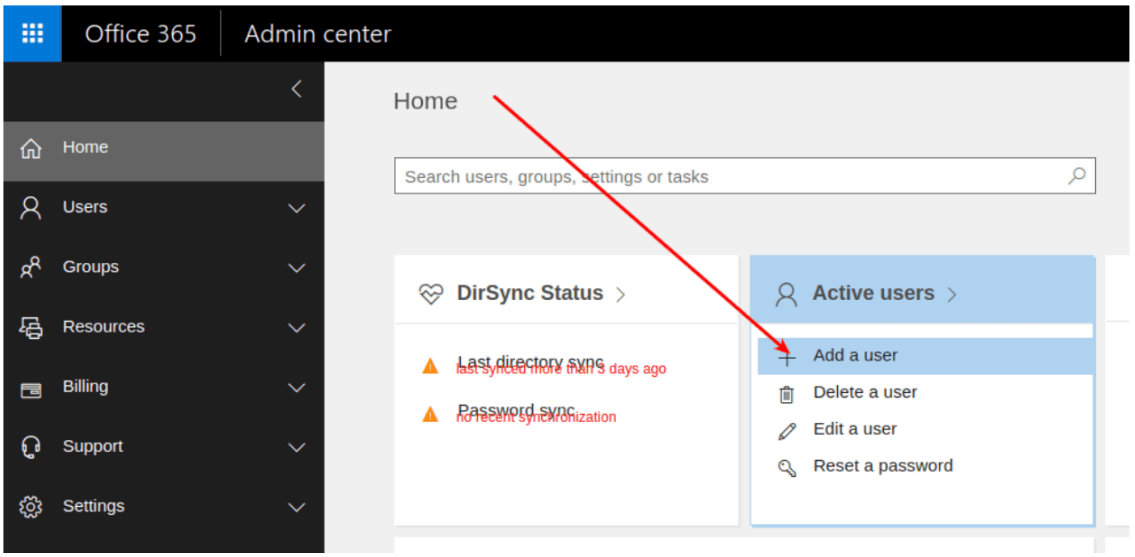
If you are crawling SharePoint Online, you may need to create a license for the crawl account.

Log in as a SharePoint administrator, and go to your admin center.

- *If you are using an on-premise active directory synced to SharePoint Online*, then you need to create an Active Directory account, and license the Active Directory account on SharePoint Online.



- *If you are using SharePoint Online user accounts*, add a user as the “Lucidworks Fusion service account”.



Add the user as “User (no administrator access)”.

2. CREATE A LUCIDWORKS FUSION CRAWL PERMISSION

1. Navigate to **Azure Active Directory** > **Conditional Access** > **Policies**.
2. Click **New Policy**.
3. Click **New permission level**. In this example, the permission level is named "LW Fusion".
4. Grant the following permissions:

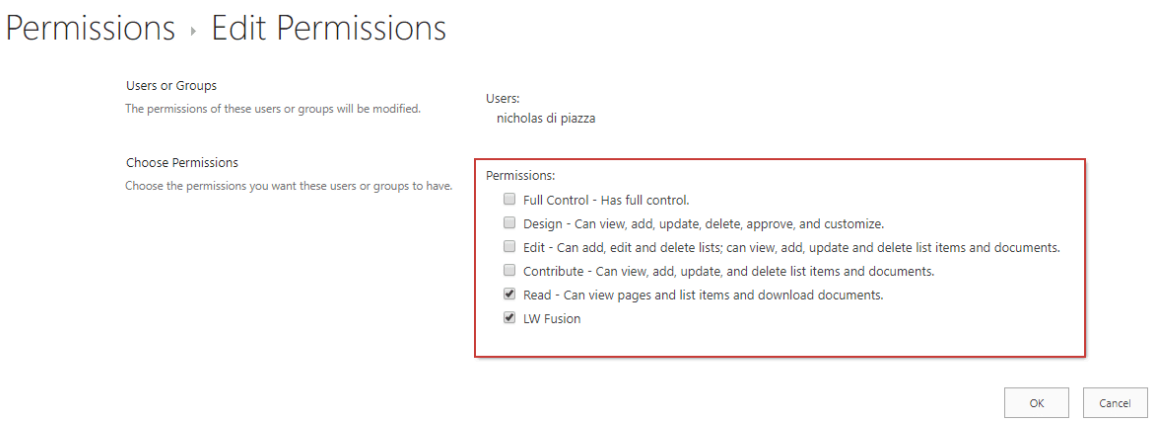
- **View Items** - View items in lists and documents in document libraries.
- **Open Items** - View the source of documents with server-side file handlers.
- **View Versions** - View past versions of a list item or document.
- **View Application Pages** - View forms, views, and application pages. Enumerate lists.

Site Permissions

- **Browse Directories** - Enumerate files and folders in a Web site using SharePoint Designer and Web DAV interfaces.
- **View Pages** - View pages in a Web site.
- **Enumerate Permissions** - Enumerate permissions on the Web site, list, folder, document, or list item.
- **Browse User Information** - View information about users of the Web site.
- **Use Remote Interfaces** - Use SOAP, Web DAV, the Client Object Model or SharePoint Designer interfaces to access the Web site.
- **Open** - Allows users to open a Web site, list, or folder in order to access items inside that container.

3. GRANT USER PERMISSION TO THE USER POLICY

1. Navigate to **Site settings > Site permissions > Advanced Permissions Settings > Grant permissions**.
2. Enter the desired user.
3. Grant **Read** and your custom Fusion permissions, "LW Fusion", for this user:



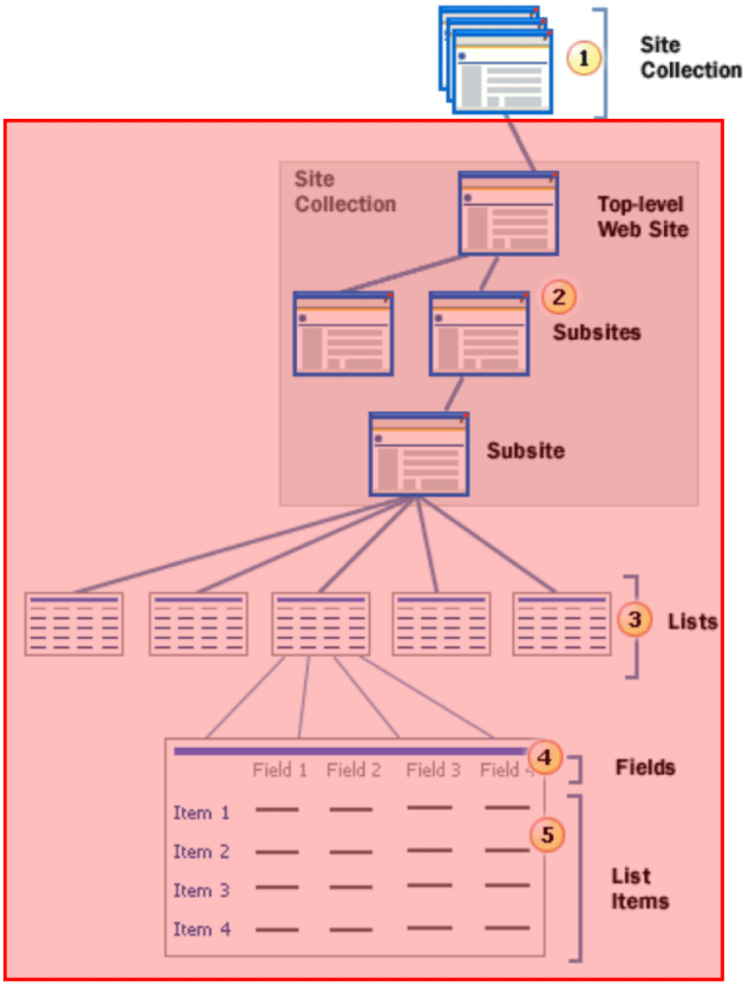
LIMITATIONS OF A CRAWLING SHAREPOINT ONLINE WITH A NON-ADMINISTRATIVE ACCOUNT

There is one important drawback of crawling SharePoint Online with a non-administrative account: Only SharePoint Online Administrators are allowed to list site collections from SharePoint Online.

So if you want to crawl multiple site collections from your SharePoint Online tenant, you must either

- list them in the Start Links explicitly, or
- provide a SharePoint administrator account when crawling SharePoint Online.

The diagram below illustrates in red what a non-administrator user can crawl:



A non-administrator can be configured to list sub-sites in a site collection. But a non-administrative user cannot list the site collections given the tenant URL.

For example: A non-admin user can list the sub-sites in `https://lucidworks.sharepoint.com/sites/sitecol`, such as `https://lucidworks.sharepoint.com/sites/sitecol/subsite1`, `https://lucidworks.sharepoint.com/sites/sitecol/subsite2`, and so on.

But only an admin can list the Site Collections in `https://lucidworks.sharepoint.com`.

How to provide admin access to crawl

You have several options for giving administrative access to Fusion to crawl your accounts, including:

- You can create a service account with admin access (not recommended).
- You can create an app-only authentication key.
  If you choose this approach, you can either use OAuth or the JWT private key option.

Configure a SharePoint V1 Optimized Datasource

How-to

> Works with: *Fusion Connectors*

The SharePoint connector retrieves content and metadata from an on-premises SharePoint repository.

1. Decide what you need to crawl

The first and most important thing to do is determine what you are trying to crawl, and to pick your “Start Links” accordingly.

Choose one of the following:

- *An entire SharePoint Web application* (all site collections in a specific SharePoint URL).
- *A subset of SharePoint site collections.*
- *A specific sub-site, list, or list item.*

How to crawl an entire SharePoint Web application

1. Leave the **Limit Documents > Fetch all site collections** option checked (as it is by default).
2. Specify the Web application URL as a site.
   For example: `https://lucidworks.sharepoint.local/`

> **Note**
>
> Crawling an entire SharePoint Web application requires administrative access to SharePoint.

How to crawl a subset of SharePoint site collections

1. Uncheck the **Limit Documents > Fetch all site collections** option.
2. Specify a "Start Link" for each site collection that you want to crawl.
   Examples: `https://lucidworks.sharepoint.local/sites/site1`, `https://lucidworks.sharepoint.local/sites/site2`, `https://lucidworks.sharepoint.local/sites/site3`

How to crawl a specific sub-site, list, or list item:

1. Uncheck the **Limit Documents > Fetch all site collections** option.
2. Specify a "Start Link" for each site collection that contains the item you want to fetch.
3. Specify a non-wildcard **Inclusive Regular Expression** for each parent.
   For example, if you want to crawl `https://lucidworks.sharepoint.local/sites/mysitecol/myparentsite/somesite` then you must include inclusive regexes for all parents along the way:

```
https\:\/\/lucidworks\.sharepoint\.local\/sites\/mysitecol
https\:\/\/lucidworks\.sharepoint\.local\/sites\/mysitecol\/mypa
rentsite
https\:\/\/lucidworks\.sharepoint\.local\/sites\/mysitecol\/some
site
https\:\/\/lucidworks\.sharepoint\.local\/sites\/mysitecol\/some
site\/.*
```

Important

If you exclude a parent item of the site, the connector will not crawl the site because it will never spider down to it during the crawl process.

2. Set up permissions for the crawl

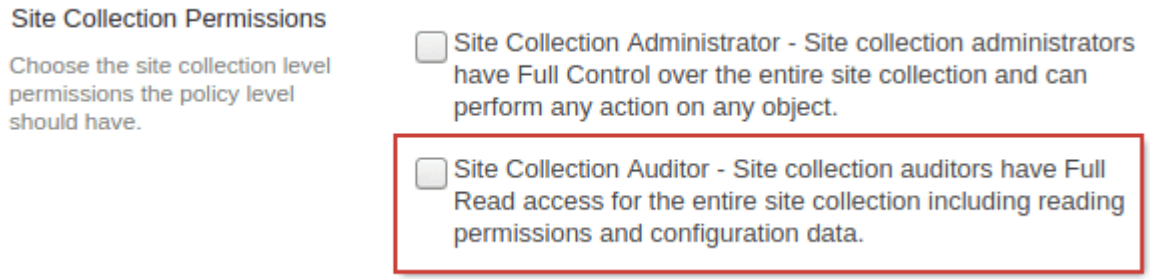
You have two options here:

- *Set up a crawl account* with only as much permission as it needs.
  This approach has the security advantage of providing minimal access to Fusion. However, the crawl account cannot retrieve the list of site collections behind a Web application URL. It cannot access the SharePoint Tenant Admin API to list all the site collections on your tenant. If you use this authentication method, you must enter each site collection to crawl in **Start Links**.
- *Provide administrative access to crawl*

How to set up a crawl account

1. CREATE A LUCIDWORKS FUSION CRAWL PERMISSION

1. Navigate to **Central Administration > Manage web application > Permission Policy**.
2. Click **Add permission policy level**. In this example, the permission level is named "fusion_crawl_policy".
3. If you need to list all site collections in a SharePoint web application, select the option **Site Collection Auditor**:



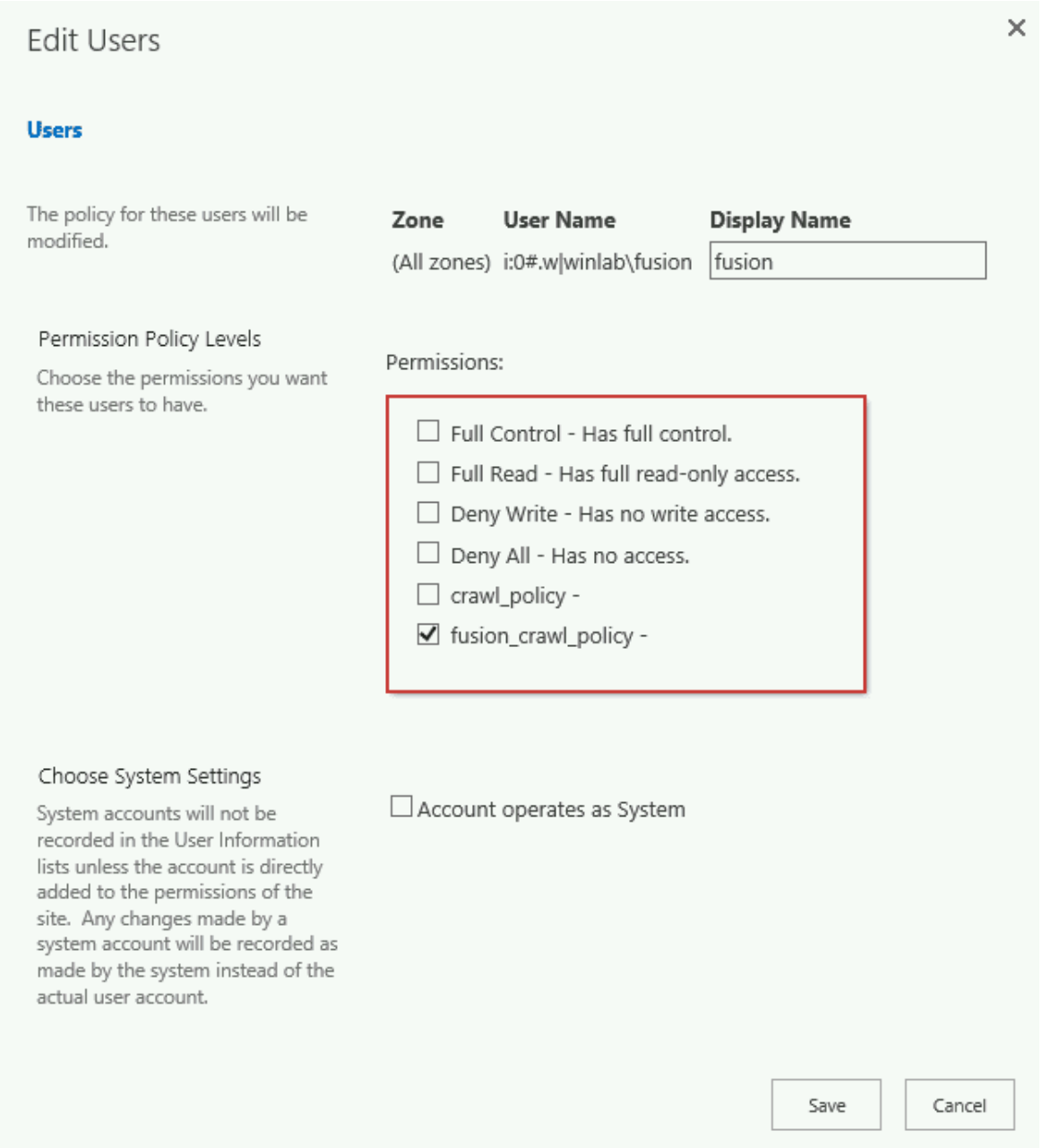
4. Grant the following permissions:
   - **View Items** - View items in lists and documents in document libraries.
   - **Open Items** - View the source of documents with server-side file handlers.
   - **View Versions** - View past versions of a list item or document.
   - **View Application Pages** - View forms, views, and application pages. Enumerate lists.

Site Permissions

- **Browse Directories** - Enumerate files and folders in a Web site using SharePoint Designer and Web DAV interfaces.
- **View Pages** - View pages in a Web site.
- **Enumerate Permissions** - Enumerate permissions on the Web site, list, folder, document, or list item.
- **Browse User Information** - View information about users of the Web site.
- **Use Remote Interfaces** - Use SOAP, Web DAV, the Client Object Model or SharePoint Designer interfaces to access the Web site.
- **Open** - Allows users to open a Web site, list, or folder in order to access items inside that container.

2. GRANT USER PERMISSION TO THE USER POLICY

1. Navigate to **Central Administration > Manage web application > User Policy > Add Users**.
2. Create a new user with the new policy permission level, "fusion_crawl_policy", selected:



How to provide admin access to crawl

See the SharePoint documentation for instructions.

3. Test user permissions

The following PowerShell script verifies permissions on the user account created to crawl SharePoint from Fusion.

Important

The script *must* be run by the user account on which the permissions were set. If rights were granted:

- On your account, *you* must run the script to verify the user rights are set correctly.
- On a different user account, the owner of that account must run the script.

1. Save the script with following file name: `test-sharepoint-permissions.ps1`.
2. Enter the first of the site collection URLs to crawl in the `$site_col_url` field of the script.
3. Save the changes.

Permission verification script

```
$site_col_url="https://your.sharepoint.local/sites/mysitecollection"
$cred = (Get-Credential) if (-not
([System.Management.Automation.PSTypeName]'ServerCertificateValidation
Callback').Type) { $certCallback = @"
    using System;
    using System.Net;
    using System.Net.Security;
    using System.Security.Cryptography.X509Certificates;
    public class ServerCertificateValidationCallback
    {
        public static void Ignore()
        {
            if(ServicePointManager.ServerCertificateValidationCallback
==null)
            {

ServicePointManager.ServerCertificateValidationCallback +=
                    delegate
                    (
                        Object obj,
                        X509Certificate certificate,
                        X509Chain chain,
                        SslPolicyErrors errors
                    )
                    {
                        return true;
                    };
            }
        }
    }
"@ Add-Type $certCallback }
[System.Net.ServicePointManager]::SecurityProtocol =
```

```
[System.Net.SecurityProtocolType]::Tls12;
[ServerCertificateValidationCallback]::Ignore() $headers = New-Object
"System.Collections.Generic.Dictionary[[String],[String]]"
$headers.Add("Content-Type", "text/xml") $headers.Add("SOAPAction",
"http://schemas.microsoft.com/sharepoint/soap/GetUpdatedFormDigestInfo
rmation") $headers.Add("X-RequestForceAuthentication", "true")
$headers.Add("X-FORMS_BASED_AUTH_ACCEPTED", "f") $body = "<?xml
version=`"1.0`" encoding=`"utf-8`"?>`n<soap:Envelope
xmlns:xsi=`"http://www.w3.org/2001/XMLSchema-instance`"
xmlns:xsd=`"http://www.w3.org/2001/XMLSchema`"
xmlns:soap=`"http://schemas.xmlsoap.org/soap/envelope/`">`n
<soap:Body>`n <GetUpdatedFormDigestInformation
xmlns=`"http://schemas.microsoft.com/sharepoint/soap/`" />`n
</soap:Body>`n</soap:Envelope>" $response = Invoke-RestMethod
"${site_col_url}/_vti_bin/sites.asmx" -Method 'POST' -Headers $headers
-Body $body -Credential $cred $digest_value =
$response.Envelope.Body.GetUpdatedFormDigestInformationResponse.FirstC
hild.DigestValue $headers = New-Object
"System.Collections.Generic.Dictionary[[String],[String]]"
$headers.Add("Content-Type", "text/xml") $headers.Add("X-
RequestForceAuthentication", "true") $headers.Add("X-RequestDigest",
$digest_value) $headers.Add("Accept", "application/json")
$headers.Add("X-FORMS_BASED_AUTH_ACCEPTED", "f") $body = @'
<Request AddExpandoFieldTypeSuffix="true" SchemaVersion="14.0.0.0"
LibraryVersion="16.0.0.0"
         ApplicationName=".NET Library"
xmlns="http://schemas.microsoft.com/sharepoint/clientquery/2009">
    <Actions>
        <ObjectPath Id="2" ObjectPathId="1"/>
        <ObjectPath Id="4" ObjectPathId="3"/>
        <Query Id="5" ObjectPathId="3">
            <Query SelectAllProperties="false">
                <Properties>
                    <Property Name="Webs" SelectAll="true">
```

```
                        <Query SelectAllProperties="false">
                            <Properties/>
                        </Query>
                    </Property>
                    <Property Name="Title" ScalarProperty="true"/>
                    <Property Name="ServerRelativeUrl"
ScalarProperty="true"/>
                    <Property Name="RoleDefinitions" SelectAll="true">
                        <Query SelectAllProperties="false">
                            <Properties/>
                        </Query>
                    </Property>
                    <Property Name="RoleAssignments" SelectAll="true">
                        <Query SelectAllProperties="false">
                            <Properties/>
                        </Query>
                    </Property>
                    <Property Name="HasUniqueRoleAssignments"
ScalarProperty="true"/>
                    <Property Name="Description"
ScalarProperty="true"/>
                    <Property Name="Id" ScalarProperty="true"/>
                    <Property Name="LastItemModifiedDate"
ScalarProperty="true"/>
                </Properties>
            </Query>
        </Query>
    </Actions>
    <ObjectPaths>
        <StaticProperty Id="1" TypeId="{3747adcd-a3c3-41b9-bfab-
4a64dd2f1e0a}" Name="Current"/>
        <Property Id="3" ParentId="1" Name="Web"/>
    </ObjectPaths>
</Request>
```

```
'@ $response = Invoke-RestMethod
"${site_col_url}/_vti_bin/client.svc/ProcessQuery" -Method 'POST' -
Headers $headers -Body $body -Credential $cred $response | ConvertTo-
Json -Depth 100
```

Successful query response

If the test script executes successfully, metadata is returned. The following is a sample of a successful response:

```
test-sharepoint-permissions.ps1 cmdlet Get-Credential at command
pipeline position 1 Supply values for the following parameters: [ {
"SchemaVersion": "14.0.0.0", "LibraryVersion": "16.0.10337.12109",
"ErrorInfo": null, "TraceCorrelationId": "c419a69f-1c06-b07f-b69b-
4d7720fd7756" }, 2, { "IsNull": false }, 4, { "IsNull": false }, 5, {
"_ObjectType_": "SP.Web", "_ObjectIdentity_": "c419a69f-1c06-b07f-
b69b-4d7720fd7756|740c6a0b-85e2-48a0-a494-e0f1759d4aa7:site:8992a373-
cdf0-4262-b240-9527c7174682:web:2080d74c-e181-43df-829f-ad5bee97b6f8",
"Webs": { "_ObjectType_": "SP.WebCollection", "_Child_Items_": [ {
"_ObjectType_": "SP.Web", ... truncated for brevity ...
"LastItemModifiedDate": "\/Date(1603731388000)\/" } ]
```

Failed query response

If the test script fails, either:

- An error code is generated. For example, an error code 401.
- An error message with explanatory information is returned. The following is a sample of a failed response:

```
Credential Invoke-RestMethod : The remote server returned an error:
(401) Unauthorized. At C:\Users\nicho\Documents\test-sharepoint-
permissions.ps1:47 char:13 + $response = Invoke-RestMethod
"${site_col_url}/_vti_bin/sites.asmx" - ... +
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ +
CategoryInfo : InvalidOperation:
(System.Net.HttpWebRequest:HttpWebRequest) [Invoke-RestMethod], WebExc
eption + FullyQualifiedErrorId :
WebCmdletWebResponseException,Microsoft.PowerShell.Commands.InvokeRest
MethodCommand Invoke-RestMethod : The remote server returned an error:
(401) Unauthorized. At C:\Users\nicho\Documents\test-sharepoint-
permissions.ps1:100 char:13 + $response = Invoke-RestMethod
"${site_col_url}/_vti_bin/client.svc/Pr ... +
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ +
CategoryInfo : InvalidOperation:
(System.Net.HttpWebRequest:HttpWebRequest) [Invoke-RestMethod], WebExc
eption + FullyQualifiedErrorId :
WebCmdletWebResponseException,Microsoft.PowerShell.Commands.InvokeRest
MethodCommand
```

Configure a SharePoint V2 Datasource

How-to

> Works with: *Fusion Connectors*

1. Decide what to crawl

Determine what to crawl and select one of the following:

- *An entire SharePoint Web application* (all site collections in a specific SharePoint URL).
- *A subset of SharePoint site collections.*
- *A specific sub-site, list, or list item.*

How to crawl an entire SharePoint Web application

1. Verify the **Limit Documents > Fetch all site collections** option is selected (default).
2. Specify the Web application URL as a site.
   For example: `https://lucidworks.sharepoint.local/`

> **Note**
>
> Administrative access to SharePoint is required to crawl an entire SharePoint Web application.

How to crawl a subset of SharePoint site collections

1. Uncheck the **Limit Documents > Fetch all site collections** option.
2. Specify a "Start Link" for each site collection to crawl.
   Examples include:
   - `https://lucidworks.sharepoint.local/sites/site1`
   - `https://lucidworks.sharepoint.local/sites/site2`
   - `https://lucidworks.sharepoint.local/sites/site3`

How to crawl a specific sub-site, list, or list item:

1. Uncheck the **Limit Documents > Fetch all site collections** option.
2. Specify a "Start Link" for each site collection that contains the item to fetch.
3. Specify a non-wildcard **Inclusive Regular Expression** for each parent.
   For example, if you want to crawl `https://lucidworks.sharepoint.local/sites/mysitecol/myparentsite/somesite`, then you must include inclusive regexes for all parents:

```
https\:\/\/lucidworks\.sharepoint\.local\/sites\/mysitecol
https\:\/\/lucidworks\.sharepoint\.local\/sites\/mysitecol\/mypa
rentsite
https\:\/\/lucidworks\.sharepoint\.local\/sites\/mysitecol\/some
site
https\:\/\/lucidworks\.sharepoint\.local\/sites\/mysitecol\/some
site\/.*
```

Important

If you exclude a parent item of the site, the connector does not crawl the site because it will not spider down to it during the crawl process.

2. Create permission and user policy for the crawl

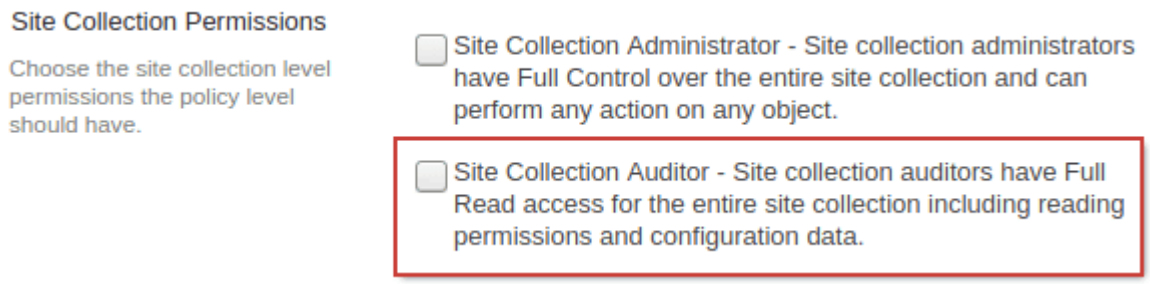
The options are:

- *Set up an on-prem crawl account* with only as much permission as it needs.
  This approach has the security advantage of providing minimal access to Fusion. However, the crawl account cannot retrieve the list of site collections behind a Web application URL.
- *Set up an online crawl account* with only as much permission as it needs.
  This approach has the security advantage of providing minimal access to Fusion. However, the crawl account cannot retrieve the list of site collections behind a Web application URL.
- *Provide administrative access to crawl*

How to set up an on-prem crawl account

CREATE A PERMISSION POLICY LEVEL

1. Navigate to **Central Administration > Manage web application > Permission Policy**.
2. Select **Add permission policy level**. In this example, the permission level is named **fusion_crawl_policy**.
3. If you need to list all site collections in a SharePoint web application, select the **Site Collection Auditor** option.



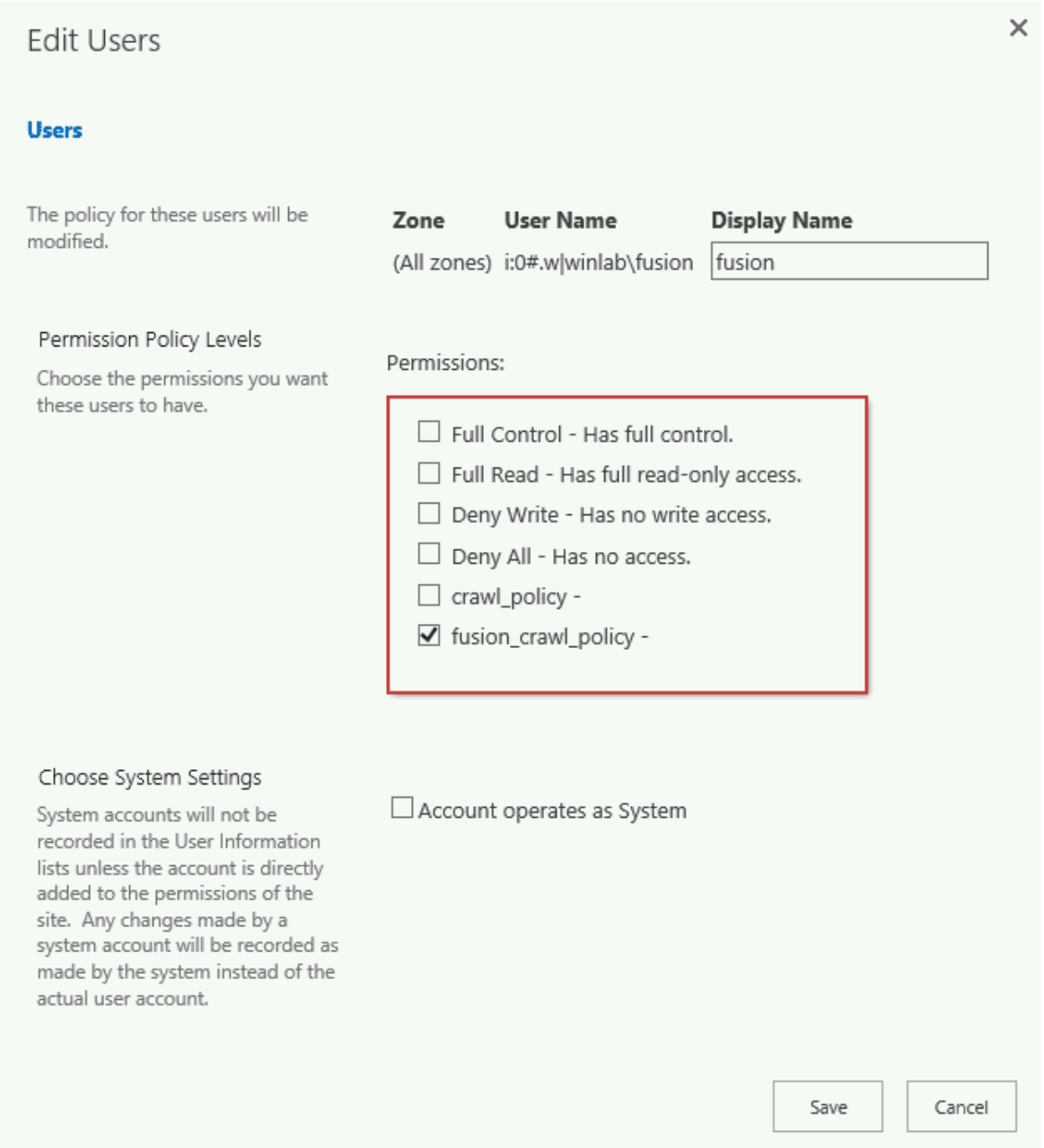
4. Grant the following permissions:
   - **View Items** - View items in lists and documents in document libraries.
   - **Open Items** - View the source of documents with server-side file handlers.
   - **View Versions** - View past versions of a list item or document.
   - **View Application Pages** - View forms, views, and application pages. Enumerate lists.

Site Permissions

- **Browse Directories** - Enumerate files and folders in a Web site using SharePoint Designer and Web DAV interfaces.
- **View Pages** - View pages in a Web site.
- **Enumerate Permissions** - Enumerate permissions on the Web site, list, folder, document, or list item.
- **Browse User Information** - View information about users of the Web site.
- **Use Remote Interfaces** - Use SOAP, Web DAV, the Client Object Model or SharePoint Designer interfaces to access the Web site.
- **Open** - Allows users to open a Web site, list, or folder in order to access items inside that container.

GRANT USER PERMISSION TO THE USER POLICY

1. Navigate to **Central Administration > Manage web application > User Policy > Add Users**.
2. Create a new user with the new **fusion_crawl_policy** permission level selected:



How to set up an online crawl account

CREATE A PERMISSION POLICY LEVEL

1. Navigate to **Site settings > Site permissions > Advanced Permission Settings**.
2. Select **New permission level**. In this example, the permission level is named **fusion_crawl_policy**.
3. Grant the following permissions:

- **View Items** - View items in lists and documents in document libraries.
- **Open Items** - View the source of documents with server-side file handlers.
- **View Versions** - View past versions of a list item or document.
- **View Application Pages** - View forms, views, and application pages. Enumerate lists.

Site Permissions

- **Browse Directories** - Enumerate files and folders in a Web site using SharePoint Designer and Web DAV interfaces.
- **View Pages** - View pages in a Web site.
- **Enumerate Permissions** - Enumerate permissions on the Web site, list, folder, document, or list item.
- **Browse User Information** - View information about users of the Web site.
- **Use Remote Interfaces** - Use SOAP, Web DAV, the Client Object Model or SharePoint Designer interfaces to access the Web site.
- **Open** - Allows users to open a Web site, list, or folder in order to access items inside that container.

GRANT USER PERMISSION

1. Navigate to **Site settings > Site permissions > Advanced Permission Settings**.
2. Select **Grant permissions**.
3. Enter the new user name and add the user.
4. Select a value in the **Select a permission level** field.
5. Select **Share**.
6. In the **Edit Permissions > Choose Permissions** section, select the following check boxes:
   - **Read.** Can view pages and list items and download documents.
   - **LW Fusion.**
7. Select **OK** to save the information.

Important

If you grant the service account the **Site Collection Auditor** permission, the Lucidworks Fusion SharePoint connector has write-level permission and can list:

- Sites in Site Collections
- SharePoint Site Collection site metadata

How to provide admin access to crawl

See the SharePoint documentation for instructions.

3. Test user permissions

The following PowerShell script verifies permissions on the user account created to crawl SharePoint from Fusion.

Important

The script *must* be run by the user account on which the permissions were set. If rights were granted:

- On your account, *you* must run the script to verify the user rights are set correctly.
- On a different user account, the owner of that account must run the script.

1. Save the script with following file name: `test-sharepoint-permissions.ps1`.
2. Enter the first of the site collection URLs to crawl in the `$site_col_url` field of the script.
3. Save the changes.

Permission verification script

```
$site_col_url="https://your.sharepoint.local/sites/mysitecollection"
$cred = (Get-Credential) if (-not
([System.Management.Automation.PSTypeName]'ServerCertificateValidation
Callback').Type) { $certCallback = @"
    using System;
    using System.Net;
    using System.Net.Security;
    using System.Security.Cryptography.X509Certificates;
    public class ServerCertificateValidationCallback
    {
        public static void Ignore()
        {
            if(ServicePointManager.ServerCertificateValidationCallback
==null)
            {

ServicePointManager.ServerCertificateValidationCallback +=
                    delegate
                    (
                        Object obj,
                        X509Certificate certificate,
                        X509Chain chain,
                        SslPolicyErrors errors
                    )
                    {
                        return true;
                    };
            }
        }
    }
"@ Add-Type $certCallback }
[System.Net.ServicePointManager]::SecurityProtocol =
```

```
[System.Net.SecurityProtocolType]::Tls12;
[ServerCertificateValidationCallback]::Ignore() $headers = New-Object
"System.Collections.Generic.Dictionary[[String],[String]]"
$headers.Add("Content-Type", "text/xml") $headers.Add("SOAPAction",
"http://schemas.microsoft.com/sharepoint/soap/GetUpdatedFormDigestInfo
rmation") $headers.Add("X-RequestForceAuthentication", "true")
$headers.Add("X-FORMS_BASED_AUTH_ACCEPTED", "f") $body = "<?xml
version=`"1.0`" encoding=`"utf-8`"?>`n<soap:Envelope
xmlns:xsi=`"http://www.w3.org/2001/XMLSchema-instance`"
xmlns:xsd=`"http://www.w3.org/2001/XMLSchema`"
xmlns:soap=`"http://schemas.xmlsoap.org/soap/envelope/`">`n
<soap:Body>`n <GetUpdatedFormDigestInformation
xmlns=`"http://schemas.microsoft.com/sharepoint/soap/`" />`n
</soap:Body>`n</soap:Envelope>" $response = Invoke-RestMethod
"${site_col_url}/_vti_bin/sites.asmx" -Method 'POST' -Headers $headers
-Body $body -Credential $cred $digest_value =
$response.Envelope.Body.GetUpdatedFormDigestInformationResponse.FirstC
hild.DigestValue $headers = New-Object
"System.Collections.Generic.Dictionary[[String],[String]]"
$headers.Add("Content-Type", "text/xml") $headers.Add("X-
RequestForceAuthentication", "true") $headers.Add("X-RequestDigest",
$digest_value) $headers.Add("Accept", "application/json")
$headers.Add("X-FORMS_BASED_AUTH_ACCEPTED", "f") $body = @'
<Request AddExpandoFieldTypeSuffix="true" SchemaVersion="14.0.0.0"
LibraryVersion="16.0.0.0"
         ApplicationName=".NET Library"
xmlns="http://schemas.microsoft.com/sharepoint/clientquery/2009">
    <Actions>
        <ObjectPath Id="2" ObjectPathId="1"/>
        <ObjectPath Id="4" ObjectPathId="3"/>
        <Query Id="5" ObjectPathId="3">
            <Query SelectAllProperties="false">
                <Properties>
                    <Property Name="Webs" SelectAll="true">
```

```
                        <Query SelectAllProperties="false">
                            <Properties/>
                        </Query>
                    </Property>
                    <Property Name="Title" ScalarProperty="true"/>
                    <Property Name="ServerRelativeUrl"
ScalarProperty="true"/>
                    <Property Name="RoleDefinitions" SelectAll="true">
                        <Query SelectAllProperties="false">
                            <Properties/>
                        </Query>
                    </Property>
                    <Property Name="RoleAssignments" SelectAll="true">
                        <Query SelectAllProperties="false">
                            <Properties/>
                        </Query>
                    </Property>
                    <Property Name="HasUniqueRoleAssignments"
ScalarProperty="true"/>
                    <Property Name="Description"
ScalarProperty="true"/>
                    <Property Name="Id" ScalarProperty="true"/>
                    <Property Name="LastItemModifiedDate"
ScalarProperty="true"/>
                </Properties>
            </Query>
        </Query>
    </Actions>
    <ObjectPaths>
        <StaticProperty Id="1" TypeId="{3747adcd-a3c3-41b9-bfab-
4a64dd2f1e0a}" Name="Current"/>
        <Property Id="3" ParentId="1" Name="Web"/>
    </ObjectPaths>
</Request>
```

```
'@ $response = Invoke-RestMethod
"${site_col_url}/_vti_bin/client.svc/ProcessQuery" -Method 'POST' -
Headers $headers -Body $body -Credential $cred $response | ConvertTo-
Json -Depth 100
```

Successful query response

If the test script executes successfully, metadata is returned. The following is a sample of a successful response:

```
test-sharepoint-permissions.ps1 cmdlet Get-Credential at command
pipeline position 1 Supply values for the following parameters: [ {
"SchemaVersion": "14.0.0.0", "LibraryVersion": "16.0.10337.12109",
"ErrorInfo": null, "TraceCorrelationId": "c419a69f-1c06-b07f-b69b-
4d7720fd7756" }, 2, { "IsNull": false }, 4, { "IsNull": false }, 5, {
"_ObjectType_": "SP.Web", "_ObjectIdentity_": "c419a69f-1c06-b07f-
b69b-4d7720fd7756|740c6a0b-85e2-48a0-a494-e0f1759d4aa7:site:8992a373-
cdf0-4262-b240-9527c7174682:web:2080d74c-e181-43df-829f-ad5bee97b6f8",
"Webs": { "_ObjectType_": "SP.WebCollection", "_Child_Items_": [ {
"_ObjectType_": "SP.Web", ... truncated for brevity ...
"LastItemModifiedDate": "\/Date(1603731388000)\/" } ]
```

Failed query response

If the test script fails, either:

- An error code is generated. For example, an error code 401.
- An error message with explanatory information is returned. The following is a sample of a failed response:

```
Credential Invoke-RestMethod : The remote server returned an error:
(401) Unauthorized. At C:\Users\nicho\Documents\test-sharepoint-
permissions.ps1:47 char:13 + $response = Invoke-RestMethod
"${site_col_url}/_vti_bin/sites.asmx" - ... +
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ +
CategoryInfo : InvalidOperation:
(System.Net.HttpWebRequest:HttpWebRequest) [Invoke-RestMethod], WebExc
eption + FullyQualifiedErrorId :
WebCmdletWebResponseException,Microsoft.PowerShell.Commands.InvokeRest
MethodCommand Invoke-RestMethod : The remote server returned an error:
(401) Unauthorized. At C:\Users\nicho\Documents\test-sharepoint-
permissions.ps1:100 char:13 + $response = Invoke-RestMethod
"${site_col_url}/_vti_bin/client.svc/Pr ... +
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ +
CategoryInfo : InvalidOperation:
(System.Net.HttpWebRequest:HttpWebRequest) [Invoke-RestMethod], WebExc
eption + FullyQualifiedErrorId :
WebCmdletWebResponseException,Microsoft.PowerShell.Commands.InvokeRest
MethodCommand
```

Configure Active Directory Federation Services (AD FS) authentication for Fusion

How-to

> Compatible versions: 4.2, 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion*

You can configure Active Directory Federation Services (AD FS) for SAML single sign-on (SSO) authentication in Fusion. This guide explains how to set up Spring Boot OAuth2 with AD FS.

Note

OAuth-based authentication against AD FS requires AD FS 3.0 or later which is available from Windows Server 2012 R2 onwards.

Configure a Security Realm in Fusion

In Fusion, create a *Security Realm* for AD FS.

1. In the Fusion workspace, navigate to **System** > **Access Control**.
2. Click **Security Realms**.
3. Click **Add Security Realm**.
4. Enter the following information for the new realm:
    - Enter a **Name**. The name must be unique and should be descriptive yet short.
    - Select **saml** from the **Type** pulldown menu.
      When you select a type, Fusion displays additional configuration options.
    - The default value for **Enabled** is true. This setting controls whether or not Fusion allows user logins for this security realm.
    - The default value for **Ephemeral Users** is false. When disabled, this setting prevents ephemeral users from being created in ZooKeeper during login. If enabled, this property negates **Auto Create Users**.
    - The default value for **Auto Create Users** is true. If enabled, a user account is created automatically upon initial authentication. If disabled, then a Fusion user with admin permissions must create Fusion users.

5. Under **SAML Realm**, configure the following information:
    - Enter the **Identity Provider URL**. This URL is used by the SAML authority for single sign-on. For example: `https://www.example.com/APP-PATH/adfs/ls`
    - Enter the URL of the **IdP Issuer**. For example: `http://www.example.com/adfs/services/trust`.
        - **IdP Issuer** must match `<saml:Issuer>` in the SAML payload.
    - Optional: Provide the **App Issuer**. This field is required if there is an `audienceRestriction` in the SAML assertion and must match `<saml:Audience>` in the SAML payload.
    - In the **Certificate Fingerprint**, paste the contents of the SAML authority certificate, *without the certificate header and footer*.
    - Optional: Enter the **User ID Attribute**. By default, the Fusion username is the same as the login name known to the Identity Provider. When another field or attribute in the user record stored by the IdP should be used as the Fusion username, that attribute name is the value of the User ID Attribute.
    - Optional: Provide a **Post Login Redirect URL**. If not set, the Fusion URL is used.
    - Optional: Provide a **Logout URL**.

6. Optional: Under **Groups Mapping**, specify the Group Name Attribute and add group mappings.
7. Click **Save**.

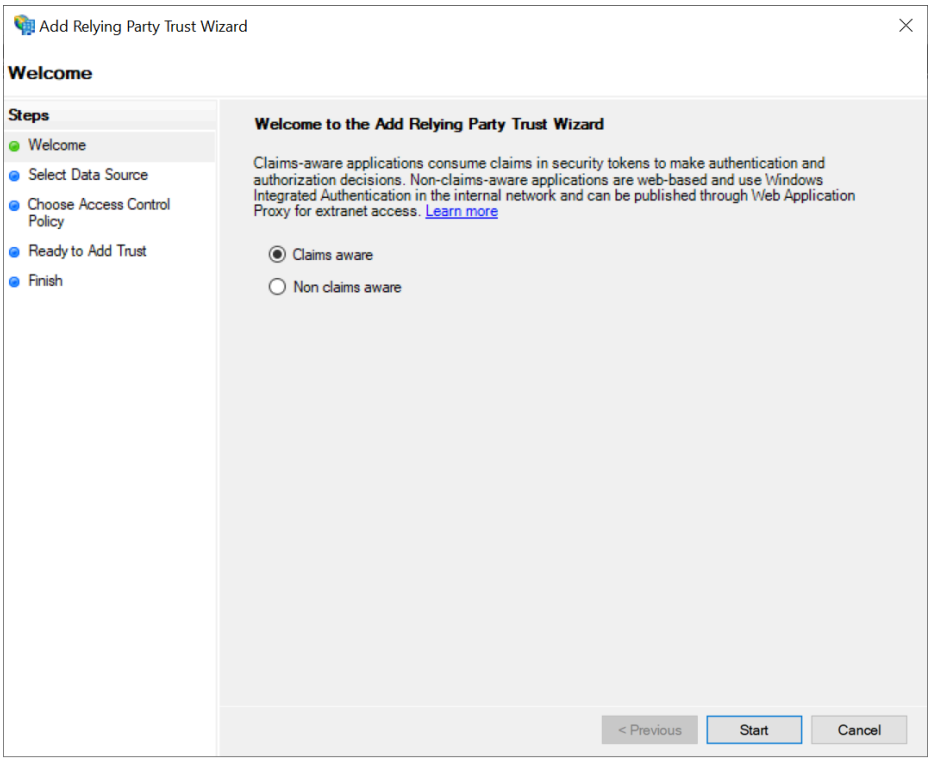
Set up the application (relying party trust) in AD FS

Create a relying party trust in the AD FS Management console. For more details, refer to the Microsoft AD FS documentation.

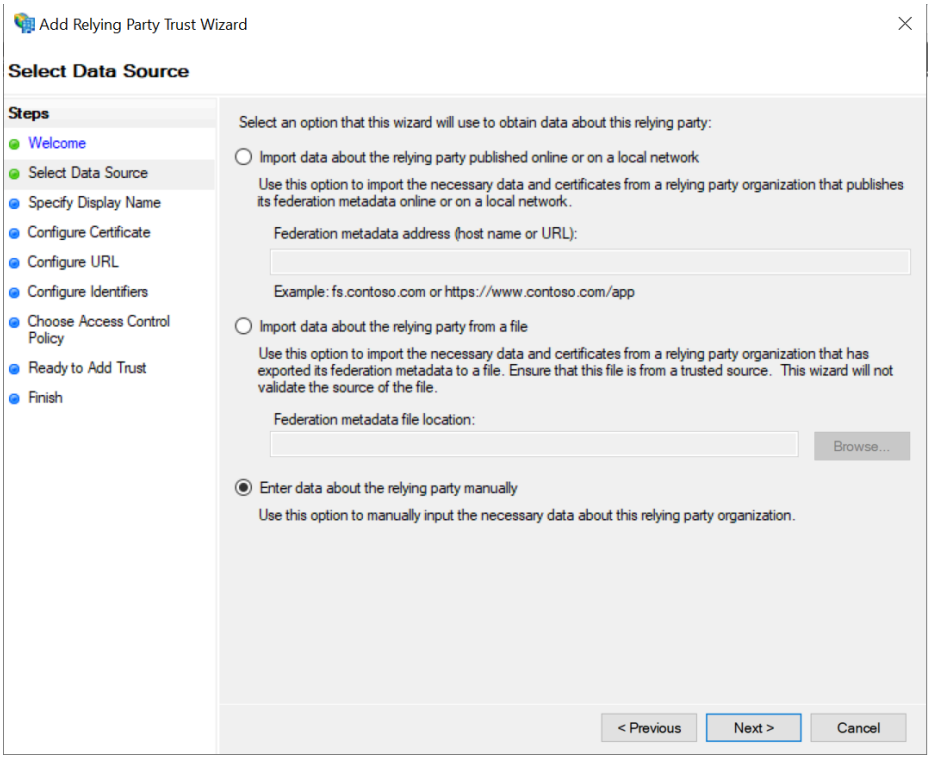
Note

The following screenshots may differ from your setup depending on the version of AD FS you're using.

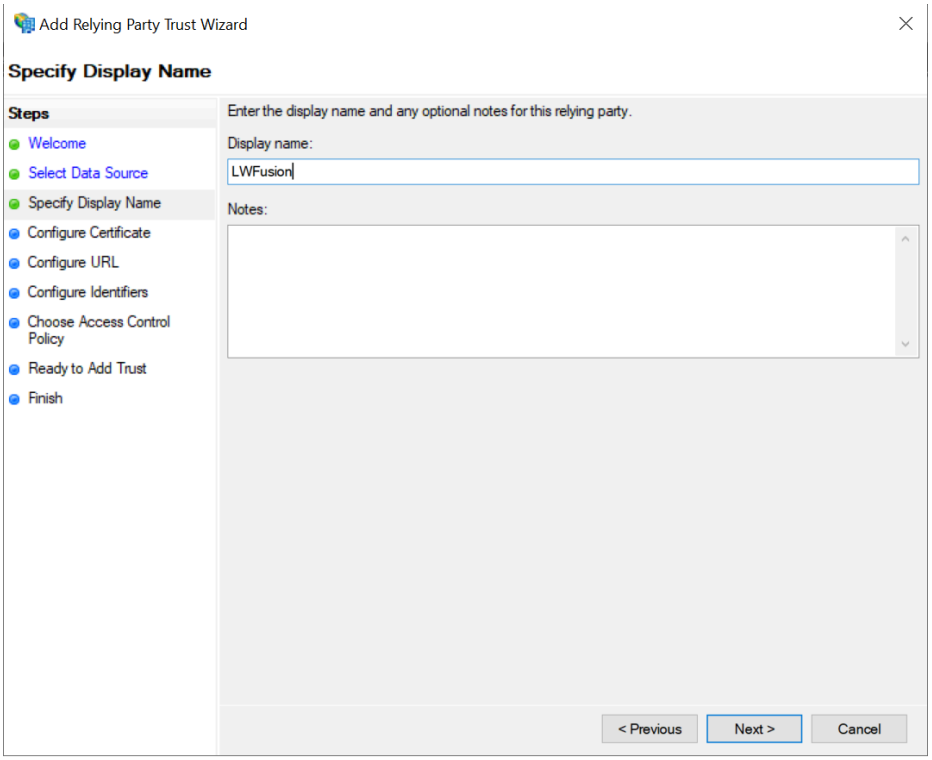
1. Launch the Add Relying Party Trust Wizard, then select **Claims aware**.



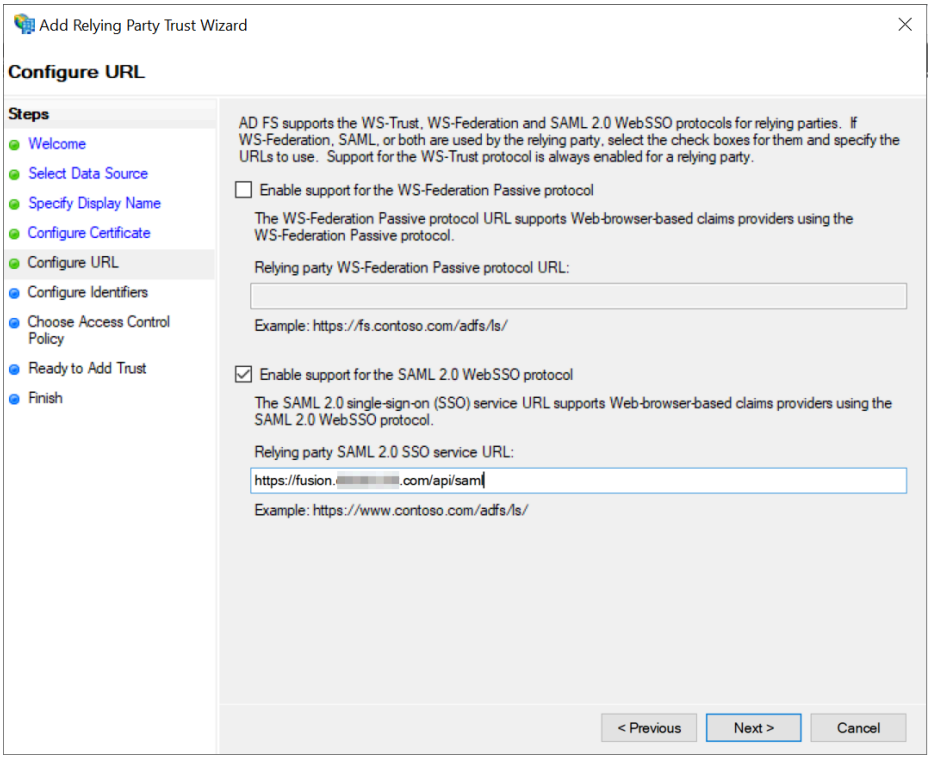
2. Choose the option to enter the data manually.



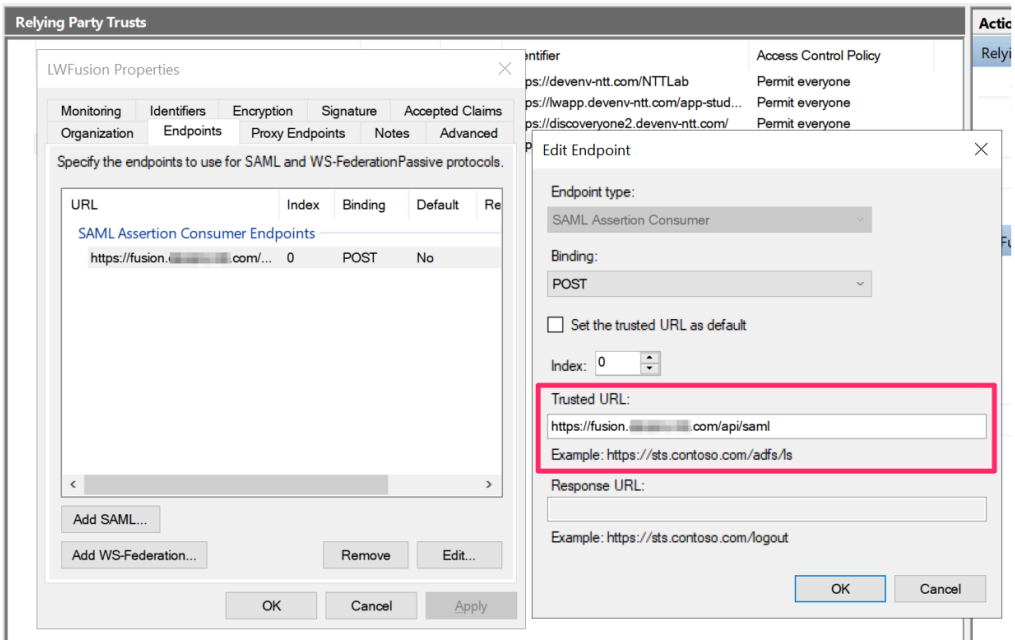
3. Set a Display name.



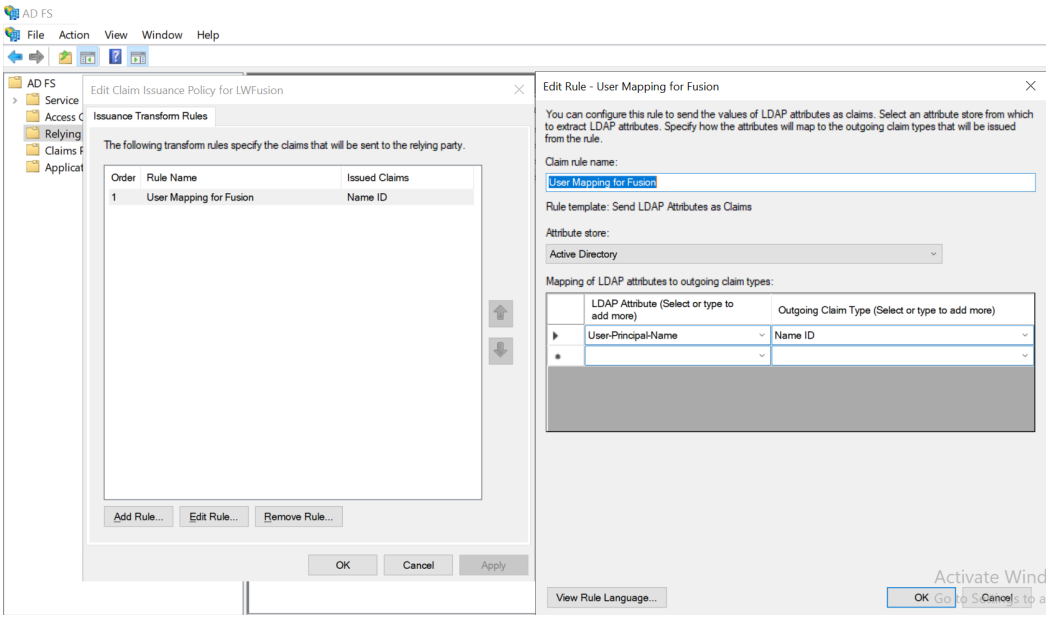
4. Configure URL. Enable the option for SAML 2.0 WebSSO protocol.



5. Set the **Relying party trust identifier** to the URL of the Fusion application.



6. Edit the Claim Rules to pass through the User Principal Name as the `NameID` claim.



7. In a terminal, create the OAuth AD FS `client` for the Fusion application:

```
Add-ADFSClient -Name "Fusion OAuth" -ClientId "1234567890-ABCDEF" -RedirectUri="http://localhost:8080/oauthLogin"
```

The `ClientId` should be a GUID and the `RedirectUri` must point to the Fusion application URL, with `/oauthLogin` appended. Here, `localhost` is used for testing a Fusion application running on the local development machine.

Configure the OAuth module for the application setup in AD FS

Create a new configuration file in `conf/security/oauth.conf`.

```
client-id: 168f3ee4-63fc-4723-a61a-6473f6cb515d adfs-url:
https://your-adfs-server/adfs resource: http://localhost:8080 high-
trust: false
```

Replace the `client-id`, `adfs-url` and `resource` parameters for your environment.

- The `client-id` is the ID that was set up against AD FS using the `Add-ADFSClient` PowerShell command.
- The `adfs-url` is the URL of the AD FS server with the `/adfs` context appended.
- The `resource` is the relying party trust identifier set up in AD FS management.
- The `high-trust` parameter is only required when integrating the AD FS and SharePoint modules. This article is only concerned with authentication against AD FS.

Test the authentication

When a user goes to the Fusion login page, they will be redirected to the AD FS OAuth login page. After the user logs in, the user will be returned to Fusion as an authenticated user. The Fusion user's details will also be populated with any basic information available from the decoded OAuth token such as roles and the user principal name.

Configure Activity Tracking

How-to

> Compatible versions: 4.2, 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *App Studio*

App Studio will send information on user clicks that trigger URL requests as well as when a user creates an annotation, for example, a bookmark or comment.

A signal event of type `click` will include, amongst other things, information about the URL that was clicked as well as the number of times that URL was clicked.

A signal event of type `annotation` will include, amongst other things, information about the annotation target, the collection, and the creator.

For a full list of properties that are stored, refer to the signals collection associated with your application.

Authorization

Activity tracking requires authorization. Both *service account impersonation* and the *Authenticating with the Sessions API* can be used for this purpose.

Service account impersonation requires credentials for the service account and these are stored in the Fusion platform configuration. These credentials can be accessed by setting the name of the Fusion platform configuration via the `platform-config` parameter in the Fusion activity tracking configuration.

If the credentials for a service account are not provided or cannot be found, a check will be made to see if a Fusion session cookie is available. Typically, this will be present when the user is known to the Fusion server and the Session API has been set up accordingly.

Configure AEM V2 Connector

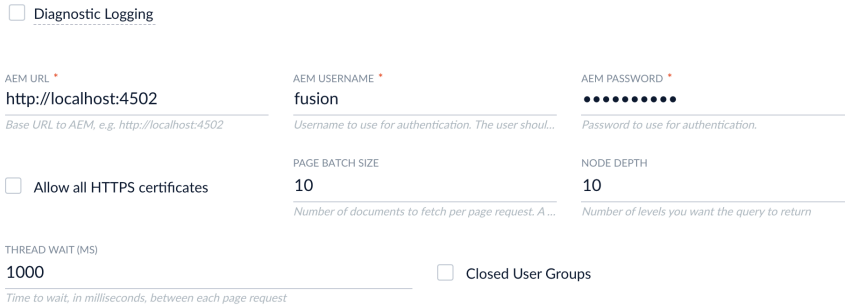
How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

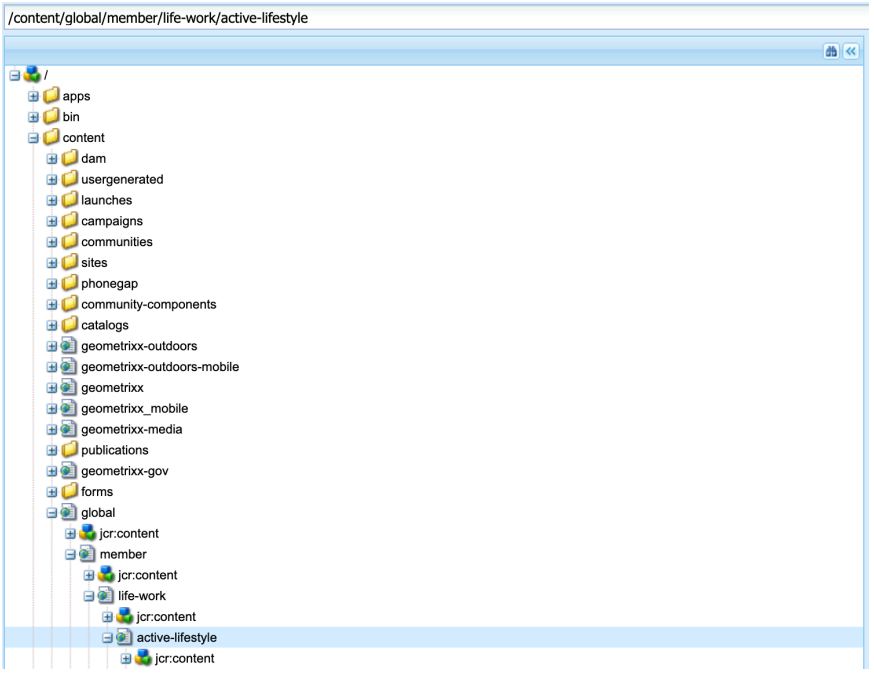
This document explains how to configure an AEM V2 connector to crawl data in Adobe Experience Manager. Refer to the *AEM reference* to learn more about how this connector works. This connector is compatible with Managed Fusion 5.5.1 and later.

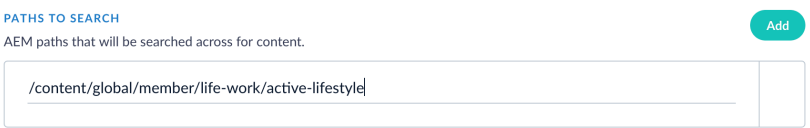
Configure AEM Datasource

1. In Managed Fusion, under Indexing > Datasources, click **Add**, then select **AEM**
2. Enter a Configuration ID
3. Enter the AEM URL (the URL used to access the AEM Admin UI) as well as the AEM username and password used to authenticate access to the QueryBuilder JSON Servlet.

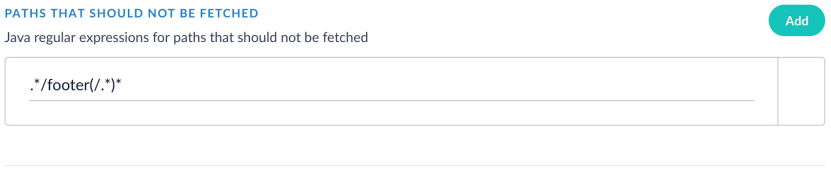


4. Go to AEM to log in and access CRXDE Lite. In the CRXDE Lite UI, select a path to crawl. Enter this path into Managed Fusion. Click **Add** to crawl multiple paths.

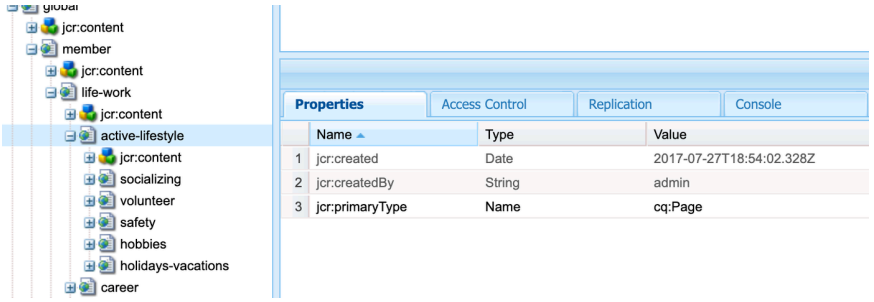


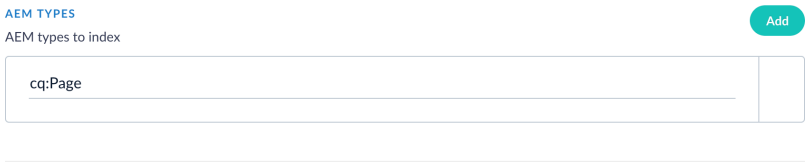


5. Optional: To exclude paths from the crawl, in Managed Fusion, enter a Java Regular Expression (regex) that represents paths to exclude in the indexed content.

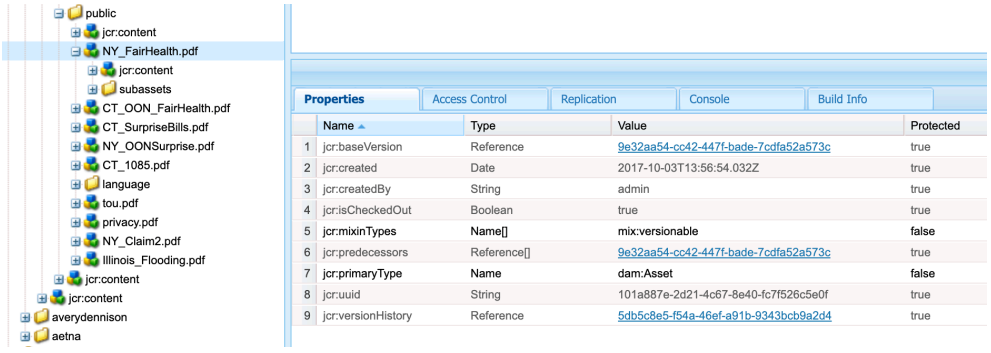


6. In Managed Fusion, enter the AEM type to crawl. In the CRXDE Lite UI, this is the `jcr:primaryType`. In this example, the AEM connector is configured to crawl the AEM Type `cq:Page`, which represents web content pages.





7. To index assets with a particular file extension, locate a file type in CRXDE Lite and enter the value of the `jcr:primaryType` into Fusion. In this example, the value of NY_FairHealth.pdf is `dam:Asset`.



8. You can choose which content properties to include and exclude from the index. These parameter values are represented by Java regex. For example, to only include properties that start with “jcr” enter `jcr:(.*)`.
9. In Managed Fusion, click **Save** when you’re done configuring the AEM datasource.

Configuration settings

Setting	Notes
AEM URL	Required. This is the URL used to access the AEM Admin UI.
AEM Username	Required. The user should have sufficient permissions to read content paths and access Users/Group APIs in case Security Trimming is needed.
AEM Password	Required.
Page Batch Size	Number of documents to fetch per page request. A higher value can increase crawling speed but also increases memory usage.
Thread wait (ms)	Number of milliseconds to wait between fetch requests. This property can be used to throttle a crawl if necessary.
Paths to search	Required.
Paths that should not be fetched	Java regex for paths that should not be fetched.
AEM Types	Required. AEM document type `jcr:primaryType` to include in the index. Examples: `cq:Page`, `dam:Asset`.
Attachment types	File extensions to index.
Content Property Include Regexes	A list of regex strings of content properties to include in indexed documents. Example: `jcr:.*`.
Content Property Exlude Regexes	A list of regex strings of content properties to exclude from indexed documents. Example: `sling:.*`
Enable Security Trimming	Enable this setting for content filtering of results based on the user's id passed in during query.
Group Mappings	AEM user groups mapped to indexed values in the security trimming field which are used to filter content based on user id passed in query.
Cache Expire Time (m)	Specifies how long a query is cached in minutes.

Field data population

There are multiple sources where AEM data is indexed. The `/bin/querybuilder.json` endpoint data is mandatory and must exist in order for a document to be indexed.

Note the list of fields that can appear in an indexed document:

Field	Source	Comments
`id`	`<AEM_URL>/bin/querybuilder.json`	Field: path
`content_txt`	`<AEM_URL>/bin/querybuilder.json`	Whole data in text format.
`<rest fields>`	`<AEM_URL>/bin/querybuilder.json`	All top level fields of JSON object.
`body_t`	`<AEM_URL>/crx/de/download.jsp`	Used if path ends with one of `Attachment types` OR path does not end with: `/jcr:content`.
`body_t`	`<AEM_URL><id>`	Used if there is no `jcr` data. If response status code is something other than 200, Fusion assumes there is no file to download under that path.
`body_t`	`[content_txt]`	Defaults to `[content_txt]` if `body_t` is empty.
`parentPage`	`Id` of document that contains attachment or link.	Populated in case of attachment/link.
`type`	File extension of the path.	Populated in case of attachment/link.
`file_size`	`<AEM_URL>/bin/querybuilder.json`	`:jcr:data;` used if `jcr` data is not empty.
`file_size`	`<AEM_URL>/bin/querybuilder.json`	`dam:size;` used if `jcr` data is empty.

Configure AEM V2 Connector

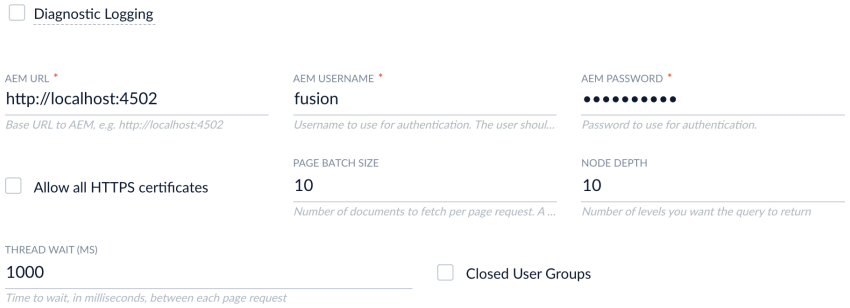
How-to

> Works with: *Fusion Connectors*

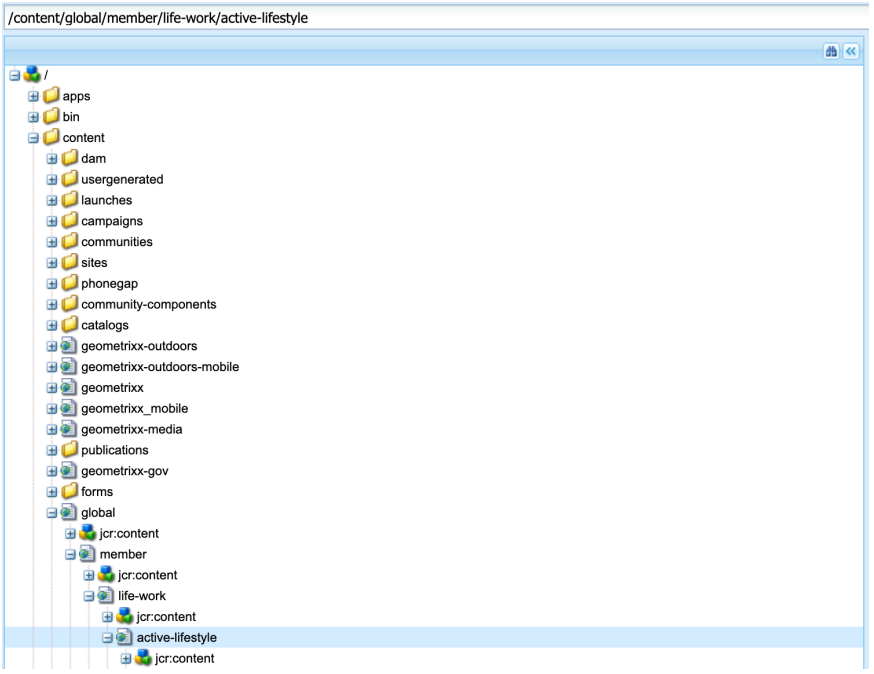
This document explains how to configure an AEM V2 connector to crawl data in Adobe Experience Manager. Refer to the *AEM reference* to learn more about how this connector works. This connector is compatible with Fusion 5.5.1 and later.

Configure AEM Datasource

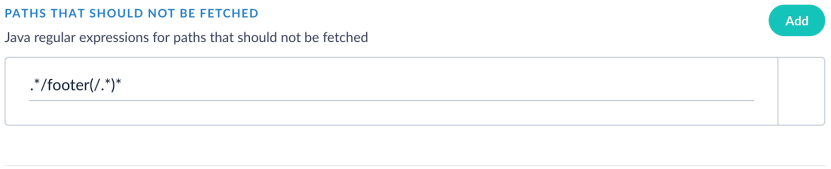
1. In Fusion, under Indexing > Datasources, click **Add**, then select **AEM**
2. Enter a Configuration ID
3. Enter the AEM URL (the URL used to access the AEM Admin UI) as well as the AEM username and password used to authenticate access to the QueryBuilder JSON Servlet.



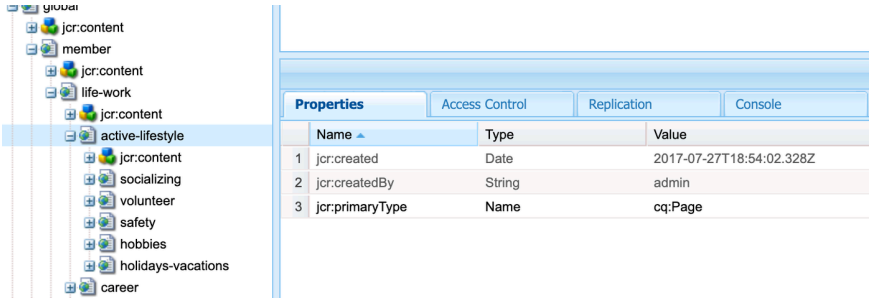
4. Go to AEM to log in and access CRXDE Lite. In the CRXDE Lite UI, select a path to crawl. Enter this path into Fusion. Click **Add** to crawl multiple paths.

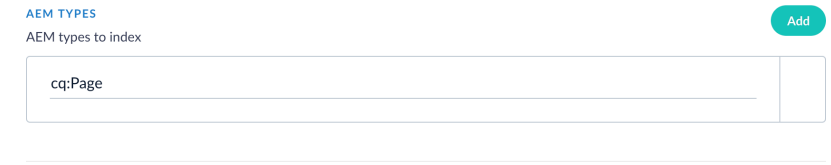


5. Optional: To exclude paths from the crawl, in Fusion, enter a Java Regular Expression (regex) that represents paths to exclude in the indexed content.

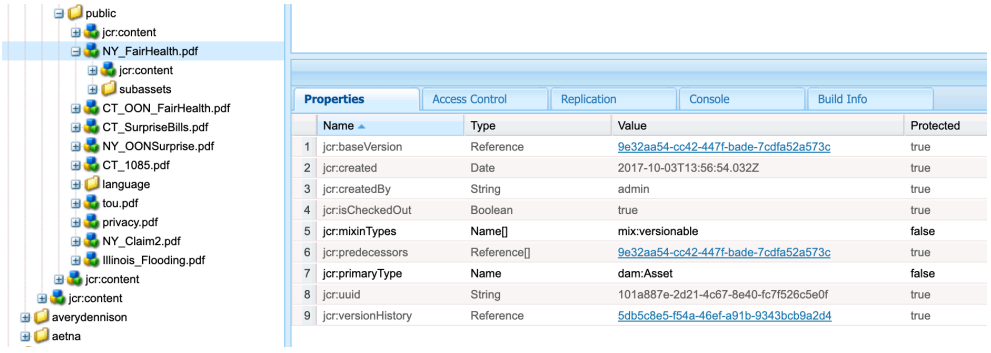


6. In Fusion, enter the AEM type to crawl. In the CRXDE Lite UI, this is the `jcr:primaryType`. In this example, the AEM connector is configured to crawl the AEM Type `cq:Page`, which represents web content pages.





7. To index assets with a particular file extension, locate a file type in CRXDE Lite and enter the value of the `jcr:primaryType` into Fusion. In this example, the value of NY_FairHealth.pdf is `dam:Asset`.



8. You can choose which content properties to include and exclude from the index. These parameter values are represented by Java regex. For example, to only include properties that start with “jcr” enter `jcr:(.*)`.
9. In Fusion, click **Save** when you’re done configuring the AEM datasource.

Configuration settings

Setting	Notes
AEM URL	Required. This is the URL used to access the AEM Admin UI.
AEM Username	Required. The user should have sufficient permissions to read content paths and access Users/Group APIs in case Security Trimming is needed.
AEM Password	Required.
Page Batch Size	Number of documents to fetch per page request. A higher value can increase crawling speed but also increases memory usage.
Thread wait (ms)	Number of milliseconds to wait between fetch requests. This property can be used to throttle a crawl if necessary.
Paths to search	Required.
Paths that should not be fetched	Java regex for paths that should not be fetched.
AEM Types	Required. AEM document type `jcr:primaryType` to include in the index. Examples: `cq:Page`, `dam:Asset`.
Attachment types	File extensions to index.
Content Property Include Regexes	A list of regex strings of content properties to include in indexed documents. Example: `jcr:.*`.
Content Property Exlude Regexes	A list of regex strings of content properties to exclude from indexed documents. Example: `sling:.*`
Enable Security Trimming	Enable this setting for content filtering of results based on the user's id passed in during query.
Group Mappings	AEM user groups mapped to indexed values in the security trimming field which are used to filter content based on user id passed in query.
Cache Expire Time (m)	Specifies how long a query is cached in minutes.

Field data population

There are multiple sources where AEM data is indexed. The `/bin/querybuilder.json` endpoint data is mandatory and must exist in order for a document to be indexed.

Note the list of fields that can appear in an indexed document:

Field	Source	Comments
`id`	`<AEM_URL>/bin/querybuilder.json`	Field: path
`content_txt`	`<AEM_URL>/bin/querybuilder.json`	Whole data in text format.
`<rest fields>`	`<AEM_URL>/bin/querybuilder.json`	All top level fields of JSON object.
`body_t`	`<AEM_URL>/crx/de/download.jsp`	Used if path ends with one of `Attachment types` OR path does not end with: `/jcr:content`.
`body_t`	`<AEM_URL><id>`	Used if there is no `jcr` data. If response status code is something other than 200, Fusion assumes there is no file to download under that path.
`body_t`	`[content_txt]`	Defaults to `[content_txt]` if `body_t` is empty.
`parentPage`	`Id` of document that contains attachment or link.	Populated in case of attachment/link.
`type`	File extension of the path.	Populated in case of attachment/link.
`file_size`	`<AEM_URL>/bin/querybuilder.json`	`:jcr:data;` used if `jcr` data is not empty.
`file_size`	`<AEM_URL>/bin/querybuilder.json`	`dam:size;` used if `jcr` data is empty.

Configure An Argo-Based Job to Access GCS

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion*

Some jobs can be configured to read from or write to Google Cloud Storage (GCS).

You can configure a combination of Solr and cloud-based input or output, that is, you can read from GCS and then write to Solr or vice versa. However, you cannot configure multiple storage sources for input or multiple storage targets for output; only one can be configured for each.

See also *Configure An Argo-Based Job to Access S3*.

Supported jobs

This procedure applies to these Argo jobs:

- Content based Recommender
- BPR Recommender
- Classification
- Evaluate QnA Pipeline
- QnA Coldstart Training
- QnA Supervised Training

For Spark jobs, see *Configure A Spark-Based Job to Access Cloud Storage*.

How to configure a job to access GCS

1. Gather the access key for your GCS account.
   See the GCS documentation .
2. Create a Kubernetes secret:

```
kubectl create secret generic my-gcs-serviceaccount-key --from-file=<my-gcs-serviceaccount-key>.json --namespace <fusion-namespace>
```

3. In the job's **Cloud storage secret name** field, enter the name of the secret for the GCS target as mounted in the Kubernetes namespace.
   This is the name you specified in the previous step. In the example above, the secret name is `my-gcs-serviceaccount-key`.

> **Tip**
>
> You can also find this name using `kubectl get secret -n <fusion-namespace>`.

4. In the job's **Additional Parameters**, add this parameter:
   - Parameter name: `google.cloud.auth.service.account.json.keyfile`
   - Parameter value: `<name of the keyfile that is available when the GCS secret is mounted to the pod>`

> **Note**
>
> The file name may be different than the secret name. You can check using `kubectl get secret -n <fusion-namespace> <secretname> -o yaml`.

Configure An Argo-Based Job to Access S3

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion*

Some jobs can be configured to read from or write to Amazon S3 (S3).

You can configure a combination of Solr and cloud-based input or output, that is, you can read from S3 and then write to Solr or vice versa. However, you cannot configure multiple storage sources for input or multiple storage targets for output; only one can be configured for each.

See also *Configure An Argo-Based Job to Access GCS*.

Supported jobs

This procedure applies to these Argo jobs:

- Content based Recommender
- BPR Recommender
- Classification
- Evaluate QnA Pipeline
- QnA Coldstart Training
- QnA Supervised Training

For Spark jobs, see *Configure A Spark-Based Job to Access Cloud Storage*.

How to configure a job to access S3

1. Gather the access key and secret key for your S3 account.
   See the AWS documentation .
2. Create a Kubernetes secret:

```
kubectl create secret generic aws-secret --from-literal=my-aws-
key-file='<access key>' --from-literal=my-aws-secret-
path='<secret key>' --namespace <fusion-namespace>
```

3. In the job's **Cloud storage secret name** field, enter the name of the secret for the S3 target as mounted in the Kubernetes namespace.
   This is the name you specified in the previous step. In the example above, the secret name is `aws-secret`.

   Tip

   You can also find this name using `kubectl get secret -n <fusion-namespace>`.

4. In the job's **Additional Parameters**, add these two parameters:
   - Param name: `fs.s3a.access.keyPath`
     Param value: `<name of the file containing the access key that is available when the S3 secret is mounted to the pod>`
   - Param name: `fs.s3a.secret.keyPath`
     Param value: `<name of the file containing the access secret that is available when the S3 secret is mounted to the pod>`

Note

The file name may be different than the secret name. You can check using `kubectl get secret -n <fusion-namespace> <secretname> -o yaml`.

Configure Autocomplete

How-to

> Compatible versions: 4.2
>
> Works with: *App Studio*

App Studio supports the ability to generate suggestions on the fly, also called autocomplete or type-ahead, using the Terms component or Suggest component embedded in Fusion's Solr core.

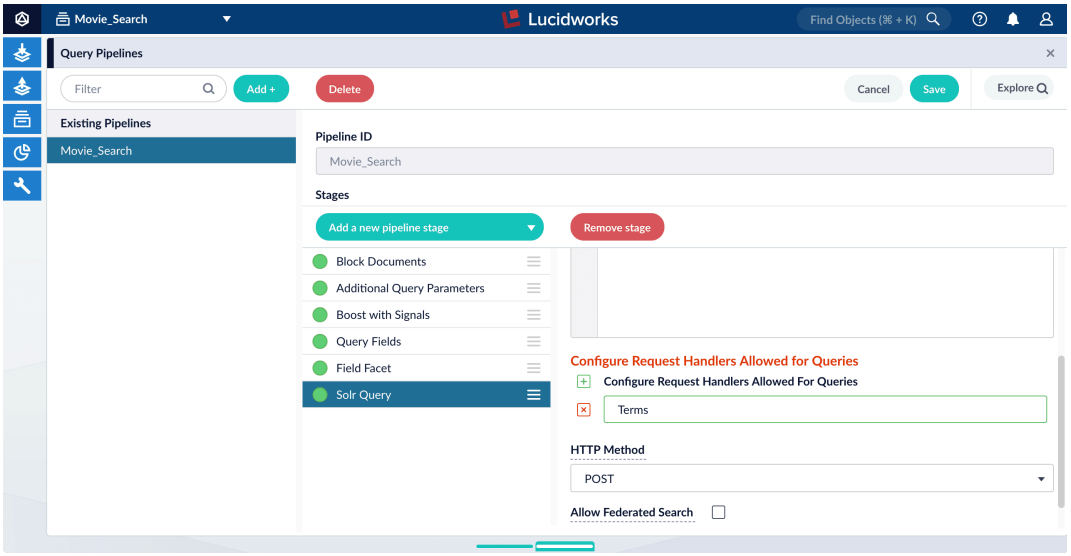
The Solr Terms component provides access to the indexed terms in a field and the number of documents that match each term. This can be used, for example, to find all terms in the Solr index that match an expansion of the given query term. For example, "car" might return CARRY, CARRIED, CARE, CARL, and so on.

Note

Since the Terms component only allows suggestions one term at a time, in cases when multiple terms are being typed, suggestions are given for the last term typed.

How to configure suggestions with the Terms component

1. Configure Fusion:
    a. In Fusion, navigate to **Querying** > **Query Pipelines**.
    b. Select the query pipeline you are using.
    c. Select the **Solr Query** pipeline stage.
    d. Under **configure Request Handlers Allowed for Queries**, click Add.
    e. Enter "Terms".



    f. Click **Save**.

Note

Unlike the Suggest component, you do not need to modify `solrconfig.xml` to use the Terms component; it is preconfigured.

2. Configure App Studio:
    a. Open `config/services/suggestions/terms.conf`.
    Its default content is as follows:

```
name:
twigkit.search.solr.suggestions.SolrQueryTermsService
source: platforms.fusion # List out all fields to base
your terms suggestions on term-fields: title title:
```

b. Change the default `source: platforms.fusion` to `source: platforms.fusion.data`.
c. Edit the value of `term-fields` as a comma-separated list of fields in your data from which to retrieve terms.

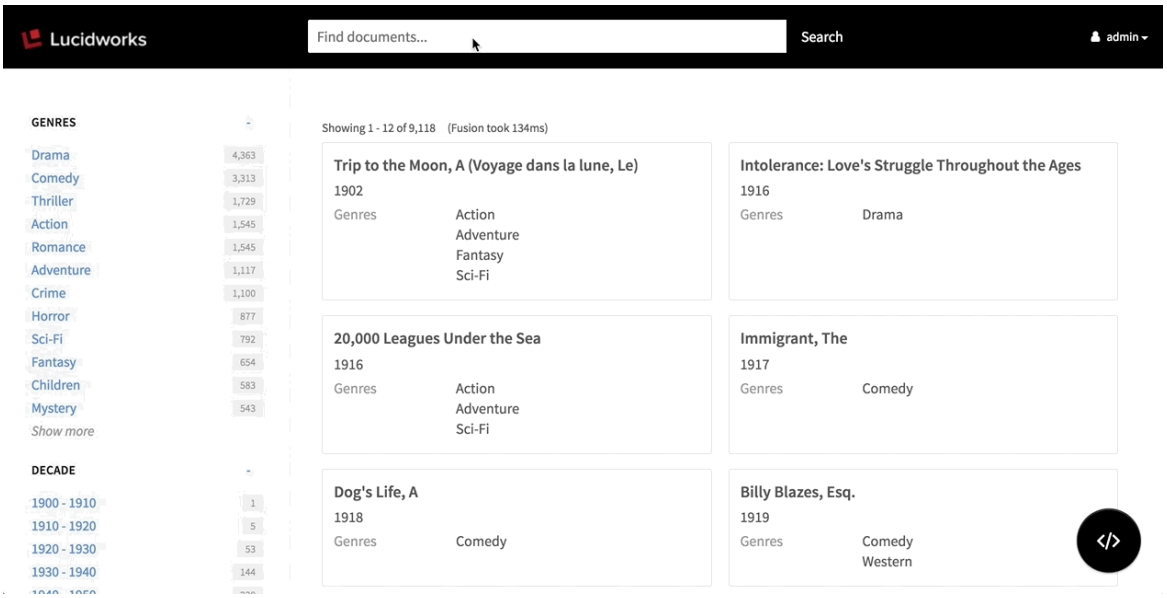
> **Tip**
>
> This corresponds to Solr's `terms.fl` parameter.

d. Edit the value of `title` to describe the group of suggestions returned by Fusion.
This title can be used to group suggestions in the UI. If you are implementing only one group of suggestions, you can leave this blank or set it to something generic, like "Suggested phrases".
e. Save and close the file.
f. Open `app/views/partials/header.tpl.html`.
g. Uncomment the following line:

```
<query-suggestions completion-
service="services.suggestions.terms" action="/search">
</query-suggestions>
```

h. Save the file.

3. Refresh the search UI in your browser.
4. Test the feature by typing a search phrase.
App Studio now displays suggested terms on the fly:



The remaining attributes shown above are optional and are the same as described above for when backwards compatibility is disabled.

Configure Box.com tokens

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

This topic explains how to configure Box.com authorization, access, and refresh tokens for the *Box.com V2 connector*.

Managed Fusion supports two methods of authentication with the Box API:

- JSON Web Token (JWT)
- OAuth2

Box app users using JWT

Box.com provides the Box Developer Edition , which lets a user access an application without having to create their own Box account.

App Auth uses the JSON Web Token (JWT) authentication architecture to establish a trusted connection with Box, allowing an application to provision and manage a Box account while minimizing the number of logins for a user and authentication services to manage.

Managed Fusion requires the following inputs to crawl your Box data.

Required options are highlighted.

UI Label, **API Name**	**Description**
JWT App User ID `f.fs.appUserId`	The Developer Edition API App User ID to use in the crawl.
JWT Public Key ID `f.fs.publicKeyId`	The public key prefix registered in Box Auth used for authentication.
JWT Private Key `f.fs.privateKeyBase64`	Base64-encoded JWT private key for the app user for authentication.
JWT Private Key File Path `f.fs.privateKeyFile`	Path to the JWT private key file for the app user for authentication.
JWT Private Key File Password `f.fs.privateKeyPassword`	The password that secures the public key.

Tip

The biggest advantage to using the JWT App Auth Users approach is that you do not have to generate new refresh tokens. The public/private key file combination remain valid indefinitely.

Authentication using OAuth 2.0

For limited testing using a single user account, you can create a Box app that uses Standard OAuth 2.0 authentication.

1. Access the Box Developers web portal and enter your Box developer account as the Admin.
2. Click **Log In**.
3. Open the page for creating a new app and click **Create New App**.
4. Click **Custom App**, and then click **Next**.
5. Click **Standard OAuth 2.0 (User Authentication)**, and then click **Next**.
6. Name your app, and then click **Create App**. The name must be globally unique across all apps created by all Box users.
7. Click **View Your App**.
8. On the Configuration page, complete the following:
    - Click the Authentication Method **Standard OAuth 2.0 (User Authentication)**.

- Set the **Redirect URI** to `http://localhost` or `http://0.0.0.0`. This address is not used by Managed Fusion, but cannot be left blank.
- Click **Save Changes**.

Configure Box.com Tokens

How-to

> Compatible versions: 4.2, 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion Connectors*, *Fusion Server*, *Fusion*

This topic explains how to configure Box.com authorization, access, and refresh tokens. The information applies to the Box.com V1 and Box.com V2 connectors. The Box.com V1 connector is available in Fusion 5.2 and earlier. The Box.com V2 connector is available in Fusion 5.3 and later.

Fusion supports two methods of authentication with the Box API:

- JSON Web Token (JWT)
- OAuth2

Box App Users Using JWT

Box.com has rather recently released a Box Developer Edition. The Box Developer Edition lets a user access an application without having to create their own Box account.

App Auth uses the JSON Web Token (JWT) authentication architecture to establish a trusted connection with Box, allowing an application to provision and manage a Box account while minimizing the number of logins for a user and authentication services to manage.

For this option, Fusion needs the inputs below to crawl your Box data.

Required options are highlighted.

UI Label, **API Name**	**Description**
JWT App User ID `f.fs.appUserId`	The Developer Edition API App User ID that you want to crawl as.
JWT Public Key ID `f.fs.publicKeyId`	The public key prefix registered in Box Auth that you want to use to authenticate with.
JWT Private Key `f.fs.privateKeyBase64`	Base64-encoded JWT private key for the app user you want to authenticate as. *(Fusion 5.0+ only.)*
JWT Private Key File Path `f.fs.privateKeyFile`	Path to the JWT private key file for the app user you want to authenticate as. *(Prior to Fusion 5.0 only.)*
JWT Private Key File Password `f.fs.privateKeyPassword`	The password that secures the public key.

Tip

The biggest advantage to using the JWT App Auth Users approach is that you do not have to generate new refresh tokens. The public/private key file combination remain valid indefinitely.

Authentication Using OAuth 2.0

For limited testing using a single user account, you can create a Box app that uses Standard OAuth 2.0 authentication.

1. Log in to your Box developer account as the Admin.
   a. Open the Box Developers web portal.
   b. In the top right corner, click **Log In**.
2. Open the page for creating a new app and click **Create New App**.
3. Click **Custom App**, and then click **Next**.

4. Click **Standard OAuth 2.0 (User Authentication)**, and then click **Next**.
5. Name your app, and then click **Create App**. The name must be globally unique across all apps created by all Box users.
6. Click **View Your App**.
7. On the Configuration page:
    a. Click the Authentication Method **Standard OAuth 2.0 (User Authentication)**.
    b. Set the **Redirect URI** to `http://localhost` or `http://0.0.0.0`. This address is not used by Fusion, but cannot be left blank.
    c. Click **Save Changes**.

Configure Dropbox Authentication

How-to

> Compatible versions: 4.2, 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion Connectors*, *Fusion Server*, *Fusion*

The Dropbox connector retrieves data from a Dropbox cloud-based data repository.

In order to authenticate to Dropbox, you must have an access token. This is obtained by first creating an app in the Dropbox platform as follows:

1. Go to https://www.dropbox.com/developers/apps and sign in with your Dropbox credentials.
   The **My Apps** page appears.
2. Click **Create app**.
   Dropbox prompts you to enter the following information:
   a. Choose an API: Select **Dropbox API**.
   b. Choose the type of access you need: Select **Full Dropbox**.
   c. Name your app: Enter an arbitrary string to identify your Dropbox app.
3. Click **Create app**.
   Dropbox displays the complete settings for your new app.
4. Under **OAuth 2** > **Generated access token**, click **Generate**.
5. Copy the new access token.
6. In Fusion UI, paste the access token into the **DROPBOX OAUTH ACCESS TOKEN** field of the Dropbox connector configuration panel.

> **Tip**
>
> If you are using the REST API, this is the value of the property `f.fs.oAuthAccessToken`.

Configure Dropbox Authentication

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

The Dropbox connector retrieves data from a Dropbox cloud-based data repository.

Create a Dropbox app

1. Access https://www.dropbox.com/developers/apps and sign in with your Dropbox credentials.
2. On the **My Apps** page, click **Create app**.
3. In the Choose an API field, select **Dropbox API**.
4. In the type of access field, select **Full Dropbox**.
5. In the Name your app field, enter a text string to identify your Dropbox app.
6. Click **Create app**.

Generate an access token

1. In your Dropbox app, click **OAuth 2 > Generated access token > Generate**.
2. Copy the new access token.

Configure the token in Managed Fusion

In the Managed Fusion UI, paste the access token into the Dropbox OAuth Access Token field of the Dropbox connector configuration panel.

> **Tip**
>
> If you are using the REST API, this is the value of the `f.fs.oAuthAccessToken` property.

Configure field mapping for Couchbase

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

The Couchbase V1 connector uses the Cross-Datacenter Replication (XDCR) feature of Couchbase to retrieve data stored in Couchbase continuously in real-time.

The Couchbase connector has built-in field mapping which allows mapping Couchbase fields to fields in your schema. The mapping configuration defines a field from your schema and an XPath-style path to the field in the Couchbase JSON document.

The field mapping can accept wildcards and double-wildcards to map fields automatically. Wildcards can be used, but only at the end of the path definition.

- `field_name=""` and `field_path=/docs/*` -maps all the fields under docs to the same name as given in JSON.
- `field_name=""` and `field_path=/docs/**` -maps all the fields under docs and their children fields to the same name as given in JSON.
- `field_name=searchField` and `field_path=/docs/*` -maps all the fields under /docs to a single field named 'searchField'.
- `field_name=searchField` and `field_path=/docs/**` -maps all the fields under /docs and their children fields to a single field named 'searchField'.

If mapping is not defined, a default mapping will be assigned, in the format of the second example above, for example, `field_name=""` and `field_path=/docs/**.`

This example shows some simple field mapping, using a single document such as this:

```
{ "first": "John", "last": "Doe", "grade": 8, "exams": [ { "subject":
"Maths", "test" : "term1", "marks": 90 }, { "subject": "Biology",
"test" : "term1", "marks": 86 } ] }
```

When you configure the datasource, you can define your field mapping as follows:

```
"field_mapping": [ { "field_name":"points_i",
"field_path":"/exams/marks" }, { "field_name":"", "field_path":"/**" }
]
```

Two mappings are defined. The first maps the `/exams/marks` field from Couchbase to the `points_i` field in Solr. The second maps all top-level and child fields from Couchbase to either the same field name in Solr or to a dynamic field rule.

After retrieving the document, it should look like this:

```
{ "first_s": "John", "last_s": "Doe", "grade_i": 8, "exams": [ {
"subject_s": "Maths", "test_s" : "term1", "points_i":90}, {
"subject_s": "Biology", "test_s" : "term1", "points_i":86} ] }
```

The `marks` field from the original document has been mapped to the `points_i` field; most of the other fields have been mapped to appropriate dynamic field rules.

Note that the representation of the document above is after it has been retrieved from Couchbase, but before it has been processed by the index pipelines. Since the index pipelines contain several stage types that can further transform the document, such as the Apache Tika Parser stage and the Field Mapping stage, the document that ends up indexed to Solr may be different from the document representation above. Some small iterations of crawling are recommended to be sure the documents are indexed as required.

Configure Field Mapping for Couchbase

How-to

> Compatible versions: 4.2, 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion Connectors*, *Fusion Server*, *Fusion*

The Couchbase V1 connector uses the Cross-Datacenter Replication (XDCR) feature of Couchbase to retrieve data stored in Couchbase continuously in real-time.

The Couchbase connector has built-in field mapping which allows mapping Couchbase fields to fields in your schema. The mapping configuration defines a field from your schema and an XPath-style path to the field in the Couchbase JSON document.

The field mapping can accept wildcards and double-wildcards to map fields automatically. Wildcards can be used, but only at the end of the path definition.

- `field_name=""` and `field_path=/docs/*` -maps all the fields under docs to the same name as given in JSON.
- `field_name=""` and `field_path=/docs/**` -maps all the fields under docs and their children fields to the same name as given in JSON.
- `field_name=searchField` and `field_path=/docs/*` -maps all the fields under /docs to a single field named 'searchField'.
- `field_name=searchField` and `field_path=/docs/**` -maps all the fields under /docs and their children fields to a single field named 'searchField'.

If mapping is not defined, a default mapping will be assigned, in the format of the second example above, i.e., `field_name=""` and `field_path=/docs/**.`

This example shows some simple field mapping, using a single document such as this:

```
{ "first": "John", "last": "Doe", "grade": 8, "exams": [ { "subject":
"Maths", "test" : "term1", "marks": 90 }, { "subject": "Biology",
"test" : "term1", "marks": 86 } ] }
```

When we configure the datasource, we can define our field mapping as follows:

```
"field_mapping": [ { "field_name":"points_i",
"field_path":"/exams/marks" }, { "field_name":"", "field_path":"/**" }
]
```

Two mappings are defined. The first will map the `/exams/marks` field from Couchbase to the `points_i` field in Solr. The second maps all top-level and child fields from Couchbase to either the same field name in Solr or to a dynamic field rule.

After retrieving the document, it should look like this:

```
{ "first_s": "John", "last_s": "Doe", "grade_i": 8, "exams": [ {
"subject_s": "Maths", "test_s" : "term1", "points_i":90}, {
"subject_s": "Biology", "test_s" : "term1", "points_i":86} ] }
```

The `marks` field from the original document has been mapped to the `points_i` field; most of the other fields have been mapped to appropriate dynamic field rules.

Note that the representation of the document above is after it has been retrieved from Couchbase, but before it has been processed by the index pipelines. Since the index pipelines contain several stage types that can further transform the document, such as the Apache Tika Parser stage and the Field Mapping stage, the document that ends up indexed to Solr may be different from the document representation above. Some small iterations of crawling are recommended to be sure the documents are indexed as required.

Configure Fusion 5.x for LDAP

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion*

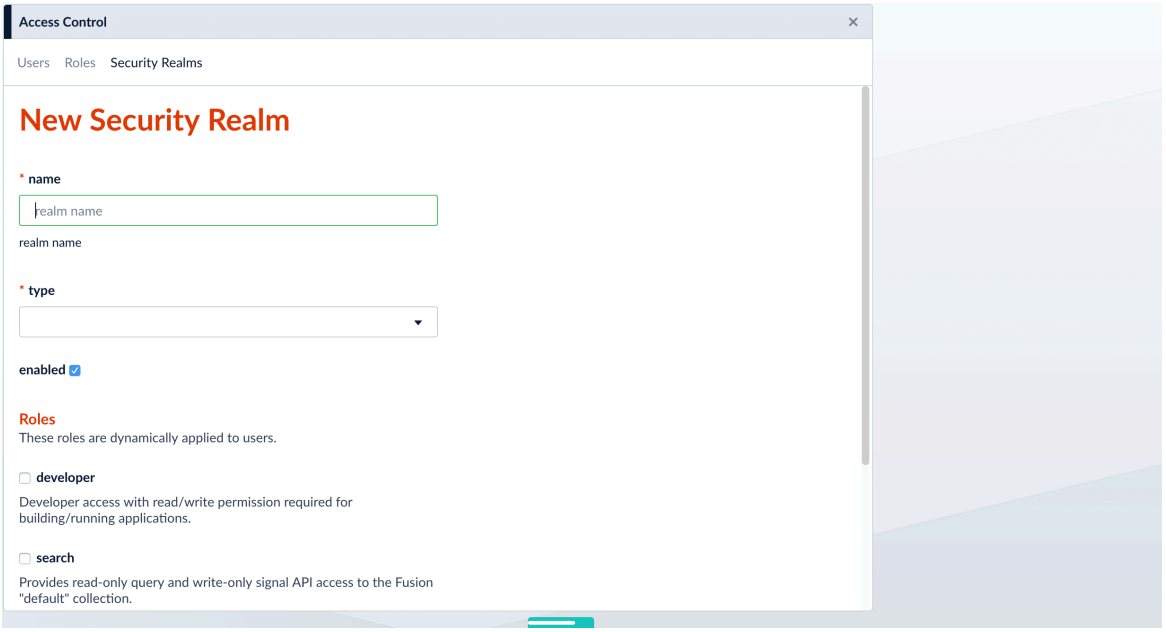
You can create security realms that use external LDAP servers for authentication. Optionally, Fusion can search in the LDAP for groups to which a user belongs, and then map those groups to Fusion roles. Fusion performs authorization using permissions stored in Fusion users and Fusion roles.

Note: Fusion *does not* use permissions from the LDAP for authorization of UI access or API requests. It only obtains group names (optionally), which are mapped to role names. If an *Active Directory Security Query Trimming Stage* is used, then directory-service permissions *are* used for trimming. If a connector supports security trimming, then connector permissions *are* used for trimming.

To configure Fusion to use an external LDAP as an authentication provider, you'll need to get information about the LDAP server(s) running on your system, either from your system or your sysadmin.

1. Add an LDAP Security Realm

1. Log in to the Fusion UI as the user admin, or as a different user with corresponding permissions.
2. Navigate to **System** > **Access Control**.
3. Click **Security Realms**.
4. Click **Add Security Realm**.



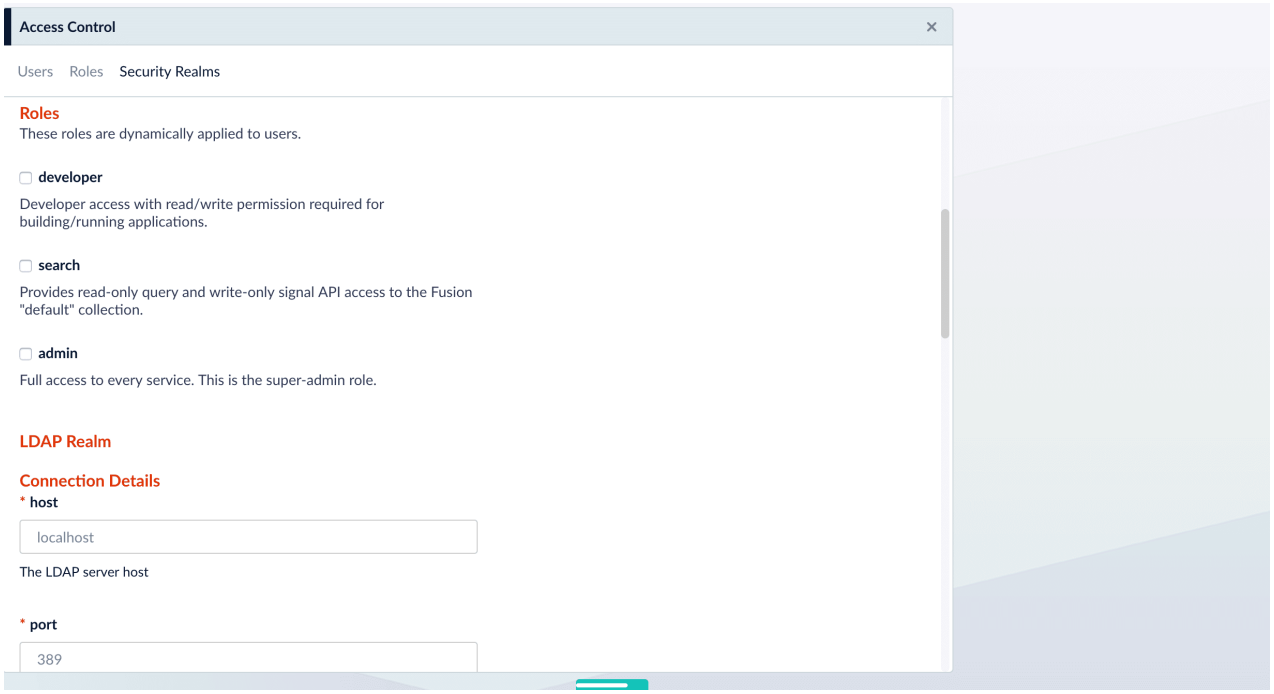
5. Specify info for the new realm:
   - `name` – Name of the security realm. It must be unique. It should be descriptive but short.
   - `type` – Select **ldap** from the pulldown menu.
     When you select **ldap**, Fusion displays additional, LDAP-specific configuration options.
   - `enabled` checkbox – Whether Fusion allows user logins for this security realm. The default is yes (checked).
   - `auto-create users` checkbox – Whether a user account is created automatically upon initial authentication. The default is yes (checked). If the checkbox is unchecked, then a Fusion user with admin permissions must create Fusion users.

6. Scroll down and specify additional options as explained in detail below.

2. Specify Static Roles (Optional)

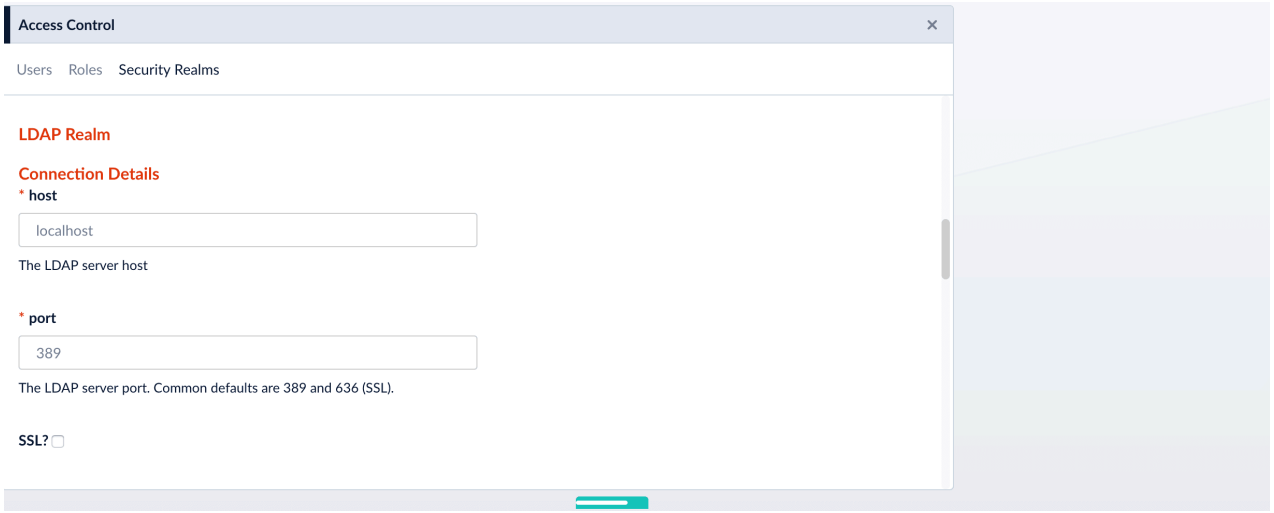
Specify one or more Fusion roles for the security realm. These roles are always considered. They do not depend on searching for LDAP groups and mapping group names to Fusion role names.

In a security realm, you can specify these static roles, add to the list of roles dynamically through an LDAP search, or both. If you do neither, Fusion uses only the role(s) and permissions defined for the user.



3. Specify LDAP Connection Details

Specify the hostname and port of the LDAP server. Check the checkbox if the server is running over SSL.



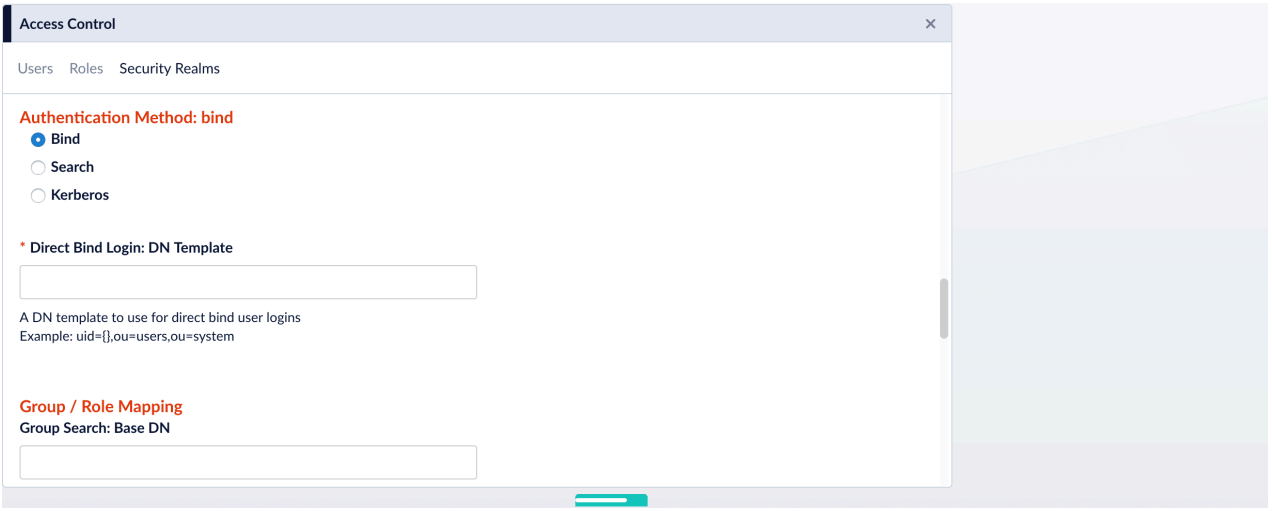
4. Specify the Authentication Method

Specify the authentication method:

- **Bind.** LDAP authentication is carried out via a single "Bind" operation. See *Bind* below.
- **Search.** LDAP authentication is carried out indirectly via a Search operation followed by a Bind operation. See *Search* below.
- **Kerberos.** Kerberos authenticates Fusion and an LDAP Search operation is carried out to find group-level authorizations. See *Kerberos* below.

Bind

Use the Bind authentication method when the Fusion login username matches a part of the LDAP distinguished name (DN). Specify the remainder of the LDAP DN in the "DN Template" configuration entry, which uses a single pair of curly braces (`{}`) as a placeholder for the value of the Fusion username.

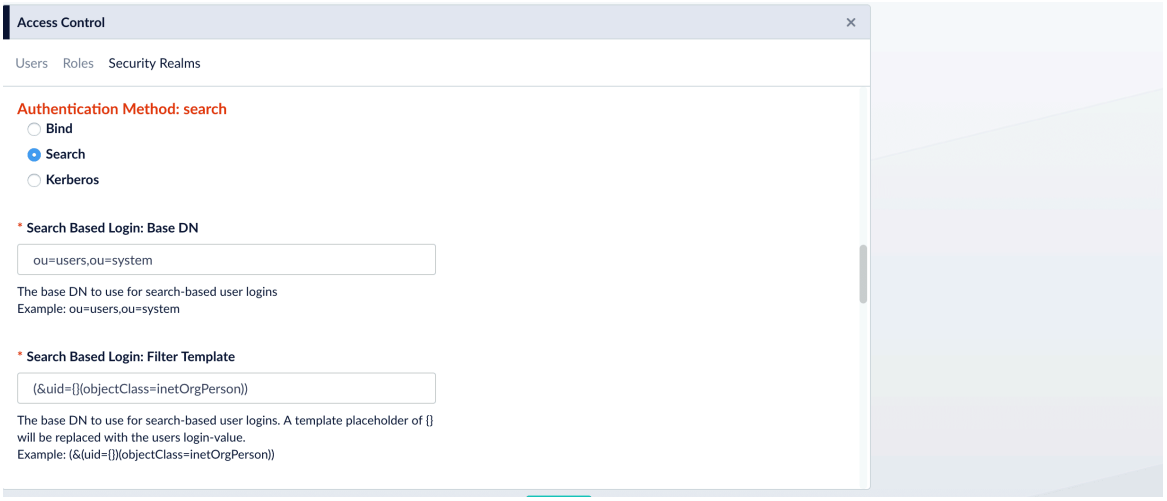


Search

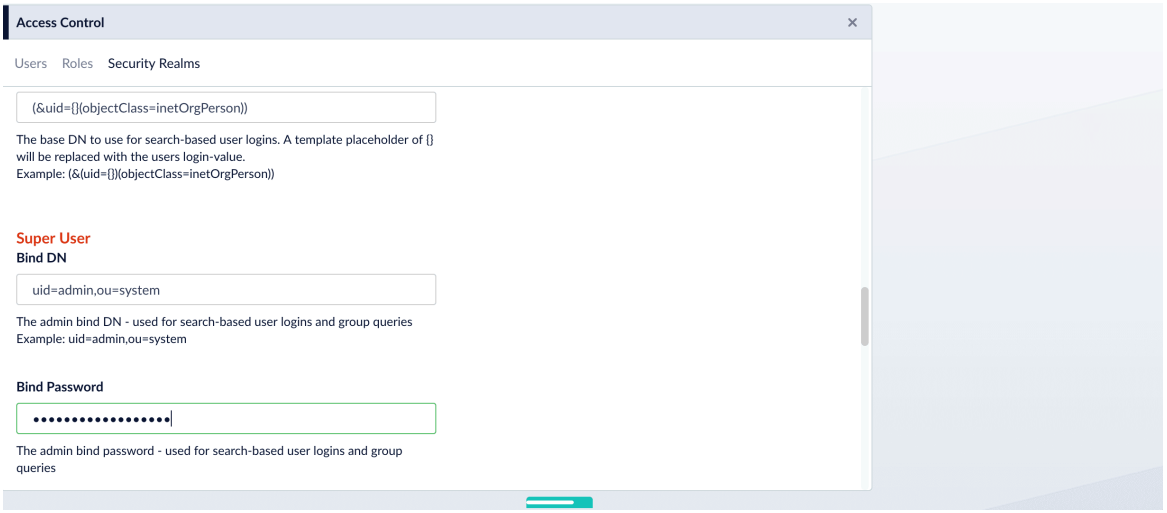
Use the Search authentication method when the username used for Fusion login *doesn't* match a part of the LDAP DN. The search request returns a valid user DN, which is then used together with the user password for authentication via a Bind request.

1. Construct a search request.
   The Search authentication method is generally required when working with Microsoft Active Directory servers. In this case, you need to know the username and password of *some* user who has sufficient privileges to query the LDAP server for user and group memberships; this user doesn't have to be the superuser.

   In addition to a privileged user DN and password, the Search authentication method requires constructing a search request. There are two parts to the request. The first part is the base DN of the LDAP directory tree that contains user account objects. The second part of the request is a Search Filter object that restricts the results to a matching subset of the information.

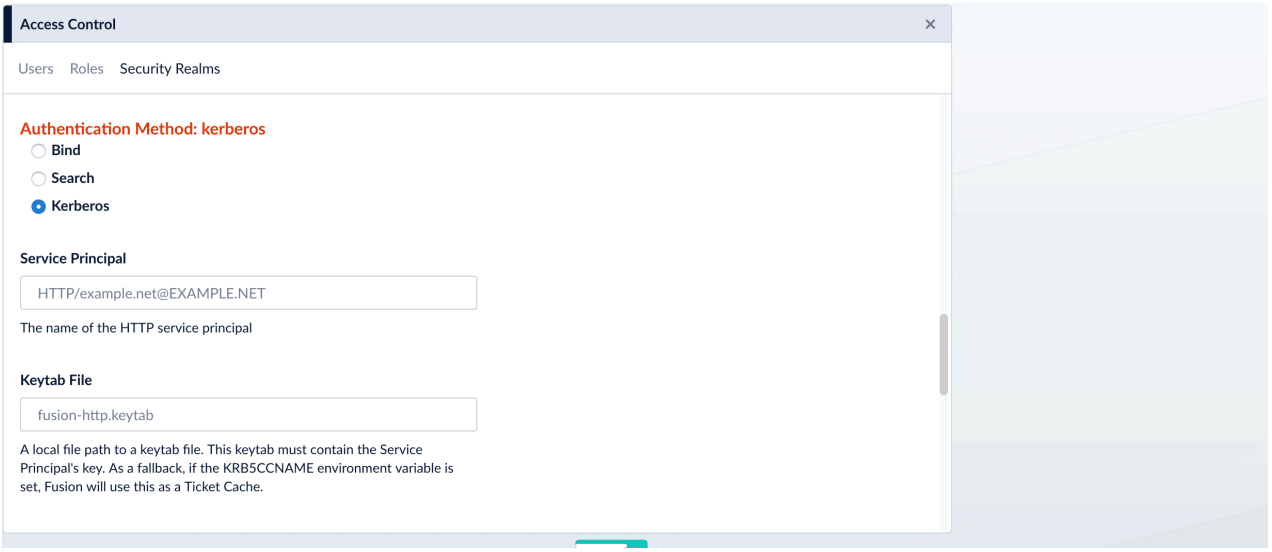


2. Provide the administrator bind DN:



Kerberos

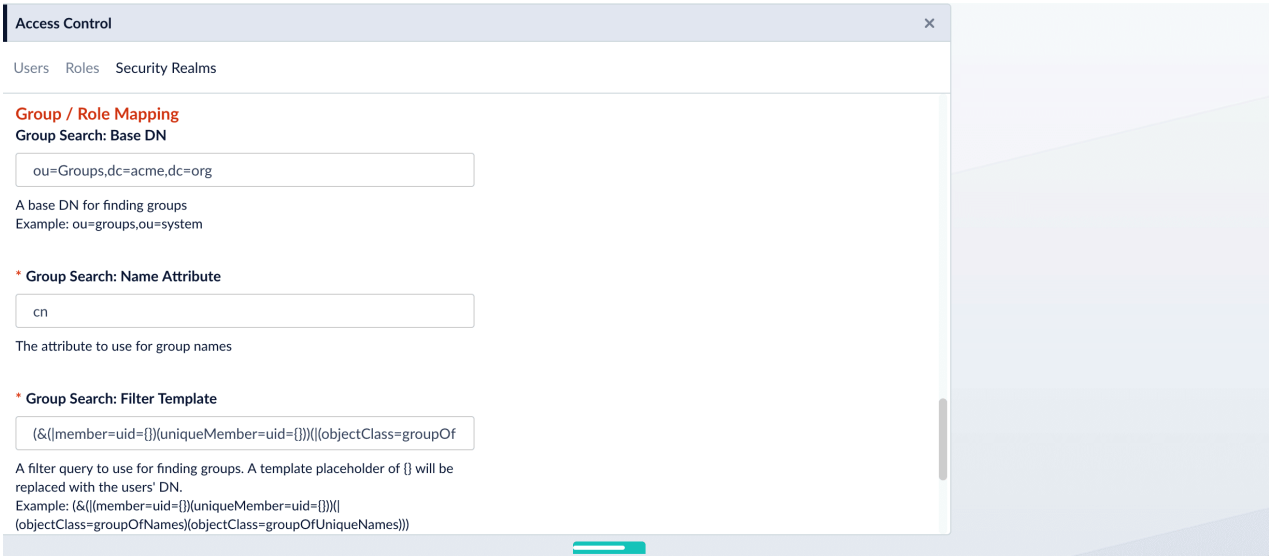
Use the Kerberos authentication method when Kerberos is the authentication provider.



5. Search for LDAP Groups (Optional)

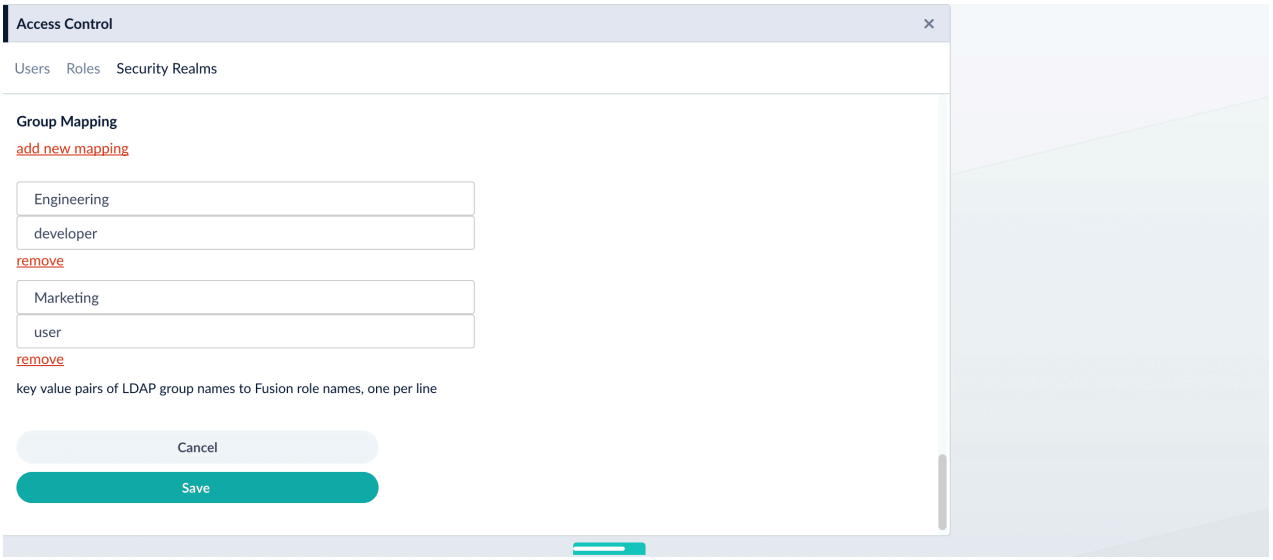
A Fusion role is a bundle of permissions tailored to the access needs of different kinds of users. Access to services and data for LDAP-managed users is controlled by mappings from LDAP users and groups to Fusion roles.

Roles can be assigned globally or restricted to specific LDAP groups. The security realm configuration panel contains a list of all Fusion roles with a checkbox for each, used to assign that role to all users in that realm. LDAP group names can be mapped directly to specific Fusion roles and LDAP group search and filter queries can also be used to map kinds of LDAP users to specific Fusion roles.



6. Map LDAP Groups to Fusion Roles (Optional)

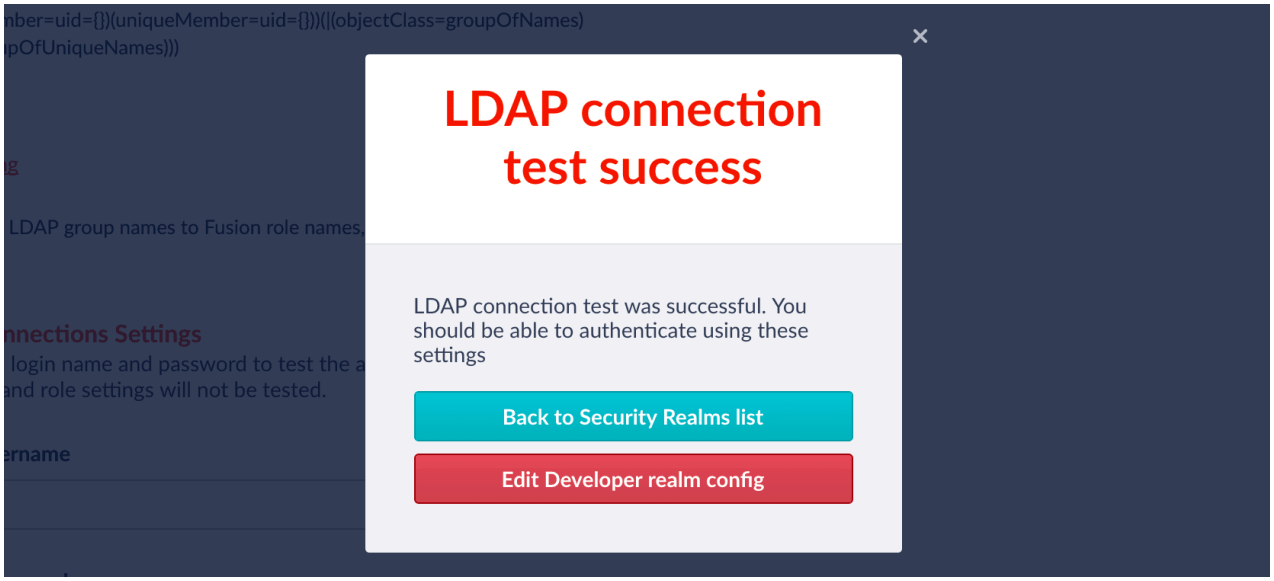
If LDAP group names returned by the search for groups match Fusion role names, you do not need to map the group names to role names. You must map any LDAP group names that do not match to Fusion role names (if you do not, they will not be used).



7. Save the Security Realm Configuration

Click **Save**.

Fusion reports whether or not authentication was successful:



8. Basic LDAP Concepts and Terminology

The LDAP protocol is used to share information about users, systems, networks, and services between servers on the internet. LDAP servers are used as a central store for usernames, passwords, and user and group permissions. Applications and services use the LDAP protocol to send user login and password information to the LDAP server. The server performs name lookup and password validation. LDAP servers also store Access Control Lists (ACLs) for file and directory objects which specify the users and groups and kinds of access allowed for those objects.

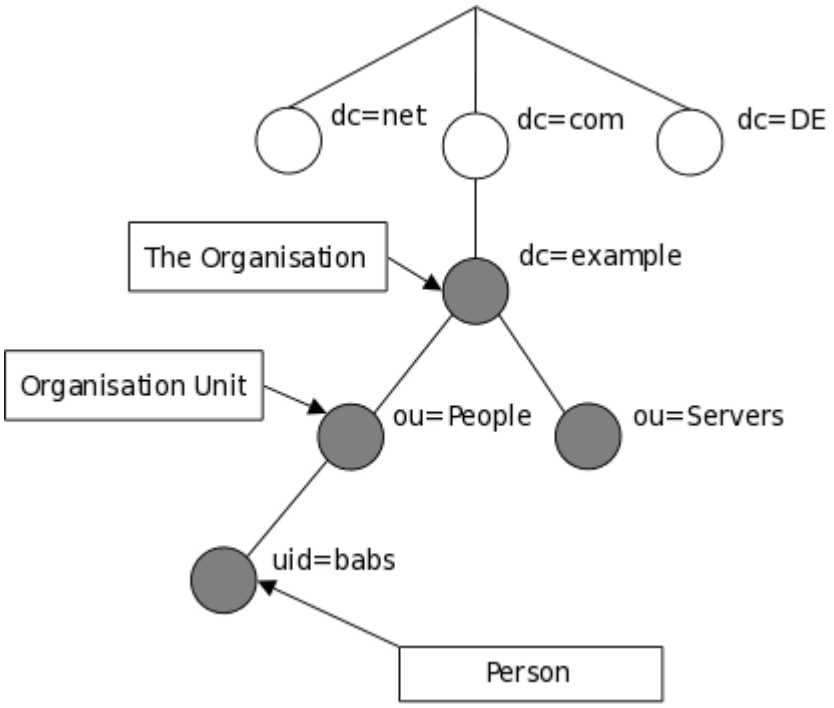
LDAP is an open standard protocol and there are many commercial and open source LDAP servers available. Microsoft environments generally use Active Directory. Unix servers use AD or other LDAP systems such as OpenLDAP, although many Unix systems do not use LDAP at all. To configure Fusion for LDAP, you'll need to get information about the LDAP server(s) running on your system either from your sysadmin or via system utilities.

Directories and Distinguished Names

An LDAP information store is a Directory Information Tree (DIT). The tree is composed of entry nodes; each node has a single parent and zero or more child nodes. Every node must have at least one attribute which uniquely distinguishes it from its siblings which is used as the node's Relative Distinguished Name (RDN). A node's Distinguished Name (DN) is a globally unique identifier.

The string representation of a DN is specified in RFC 4514 . It consists of the node's RDN followed by a comma, followed by the parent node's DN. The string representation of the RDN is the attribute-value pair name, connected by an equals ("=") sign. This recursive definition means that the DN of a node is composed by working from the node back through its parent and ancestor nodes up to the root node.

Here is a small example of a DIT:

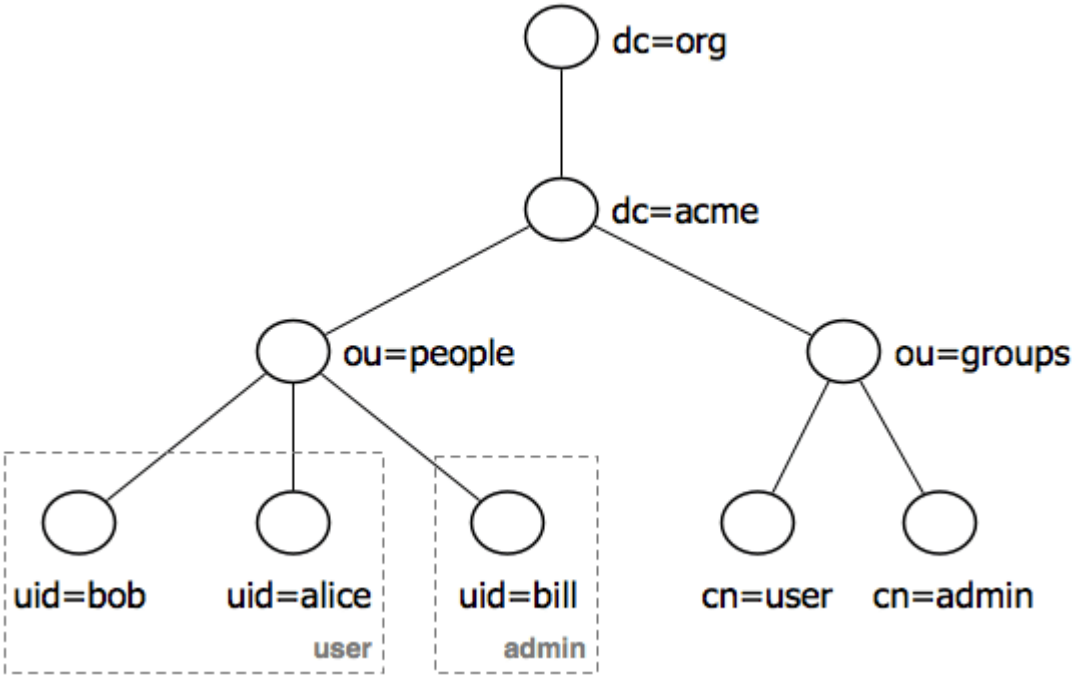


The person entry in this tree has the DN: "uid=babs, ou=people, dc=example, dc=com".

Attribute names include many short strings based on English words and abbreviations, e.g.:

Name	Description
cn	commonName
dc	domainComponent
mail	email address
ou	organizationalUnitName
sn	surname
uid	userId

LDAP entry attributes can refer to other LDAP entries by using the DN of the entry as value of that attribute. The following example of a directory which contains user and groups information shows how this works:



This tree contains two organizational units: "ou=people" and "ou=groups". The children of the "group" organizational unit are specific named groups, just as the child nodes of organization unit "people" are specific users. There are three user entries with RDNs "uid=bob", "uid=alice", "uid=bill" and two groups with RDNs "cn=user" and "cn=admin". The dotted lines and group labels around the person nodes indicate group membership. This relationship is declared on the groups nodes by adding an attribute named "member" whose value is a users DN. In the LDAP data interchange format (LDIF), this is written:

```
cn=user,ou=groups,dc=acme,dc=org member:
uid=bob,ou=people,dc=acme,dc=org member:
uid=alice,ou=people,dc=acme,dc=org cn=admin,ou=groups,dc=acme,dc=org
member: uid=bill,ou=people,dc=acme,dc=org
```

See the Wikipedia's LDAP entry for details.

LDAP Protocol Operations

For authentication purposes, Fusion sends Bind operation requests to the LDAP server. The Bind operation authenticates clients (and the users or applications behind them) to the directory server, establishes authorization identity used for subsequent operations on that connection, and specifies the LDAP protocol version that the client will use.

Depending on the way that the host system uses LDAP to store login information about users and groups, it may be necessary to send Search operation requests to the LDAP server as well. The Search operation retrieves partial or complete copies of entries matching a given set of criteria.

LDAP filters specify which entries should be returned. These are specified using prefix notation. Boolean operators are "&" for logical AND, "|" for logical OR, e.g., "A AND B" is written "(&(A)(B))". To tune and test search filters for a Unix-based LDAP system, see the ldapsearch command line utility documentation. For Active Directory systems, see AD Syntax Filters.

Configure Fusion for Kerberos in Unix

How-to

> Compatible versions: 5.9
>
> Works with: *Fusion Server*, *Fusion*

To configure the Fusion UI service to use Kerberos for user authentication, you must create a Kerberos *security realm*.

Note

Kerberos support requires Fusion 5.9.5.

Kerberos is a system that provides authenticated access for users and services on a network. Instead of sending passwords in plaintext over the network, encrypted passwords are used to generate time-sensitive tickets that are used for authentication. SPNEGO provides a mechanism for extending Kerberos to Web applications through the standard HTTP protocol.

Kerberos uses symmetric-key cryptography and a trusted third party called a Key Distribution Center (KDC) to authenticate users to a suite of network services. (By users we mean both end users and client programs). The computers managed by that KDC and any secondary KDCs constitute a realm. When a user authenticates to the KDC, the KDC sends a set of credentials (a ticket) specific to that session back to the user's machine. Kerberos-aware services use the ticket on the user's machine for authentication instead of requiring sign-on with a password. Because tickets are used rather than passwords, this provides the convenience of Single Sign-On (SSO) in addition to security.

A Kerberized process is one that has been configured so it can get tickets from a KDC and negotiate with Kerberos-aware services. When a user sends an HTTP request, Fusion tries to authenticate using the Kerberos/SPNEGO protocol. If the request was sent from a browser, Fusion does not display the initial sign-on panel; instead on login, the user sees the main Fusion collections panel.

Important

This article focuses on configuring Fusion for Kerberos *in Unix*. To learn how to configure Fusion for Windows, see *Configure Fusion for Kerberos in Windows*.

To Kerberize Fusion, you must:

- Configure the Kerberos client on the server that the Fusion UI service will be running on so that it can talk to the KDC (section *Configuring the Kerberos client* below).
- Configure the security realm of the Fusion UI (section *Configuring Fusion Authentication for Kerberos Realm* below).
- Depending on the encryption used by the KDC, you may also need to install additional Java security libraries into the Fusion distribution. These are freely available from Oracle, see download and installation instructions below.

To do this, you need following information, which you can get from your sysadmin:

- **Kerberos realm name.** In most cases, this is your domain name in upper case.
- **KDC name.** This is usually a combination of `kerberos.` and your domain name. For example, `kerberos.example.com`
- **Kerberos principal name and password.** A principal is a unique identity to which Kerberos can assign tickets. When the entity is a client program, this is called the Service Principal name.
- **A keytab file.** This holds the encrypted credentials.

The usual scenario in an enterprise organization is to have a Kerberos admin create a service principal with a random key password. Then, the admin generates a keytab, which is then used for Fusion service principal authentication.

The Kerberos commands needed for configuration and testing are:

- `kinit`. Obtain and cache a ticket from the KDC (i.e., domain login)
- `kdestroy`. Destroys credentials (i.e., domain logout)
- `klist`. Lists cached credentials
- `ktutil`. Create or add credentials to a keytab file

If your browser is not already configured to use the Kerberos/SPNEGO, you need to do so in order to test the Fusion configuration.

Configuring the Kerberos client

Step 1: Edit the Kerberos configuration file.

To configure your local Kerberos client so that it can talk to the Kerberized server, you must edit Kerberos configuration file named `krb5.conf`. On most Unix systems, this file is located at `/etc/krb5.conf`.

This file contains Kerberos configuration information, including the locations of KDCs and admin servers for the Kerberos realms of interest, defaults for the current realm and for Kerberos applications, and mappings of hostnames onto Kerberos realms.

If your organization realm name is "MYORG.ORG", and your KDC server is named "kerberos.myorg.org", then you edit two entries. The first entry is `libdefaults`. Set MYORG.ORG as the default realm:

```
[libdefaults] default_realm = MYORG.ORG
```

The second entry is `realms`. Add MYORG.ORG as a realm:

```
[realms] MYORG.ORG = { kdc = kerberos.myrealm.com }
```

For example, for realm LUCIDWORKS.IO, the krb5.conf file is just like the above example, except that instead of "myorg.org" we specify "lucidworks.io".

Step 2: Authenticate to Kerberos

The command `kinit` is the Kerberos authentication command. To get started, you authenticate to Kerberos using the Kerberos principal name and password (which you may need to obtain from your sysadmin). For this example, the principal name is "prince".

```
> kinit prince
```

The `kinit` command prompts for a password. Successful authentication is silent. Unsuccessful authentication results in an error message.

The command `klist` shows all cached Kerberos credentials. To check that you have successfully authenticated, run this command:

```
> klist
```

Output should be in this form, but with your data:

```
Credentials cache: API:C980F9F5-415C-4A3E-9C67-883C7D5FFFBE Principal:
prince@MYORG.ORG Issued Expires Principal May 6 15:14:55 2015 May 7
01:12:47 2015 krbtgt/MYORG.ORG@MYORG.ORG
```

The Service Principal Keytab file

The usual scenario in an enterprise organization is to have a Kerberos admin create a service principal with a random key password. Then, the admin generates a keytab, which is then used for Fusion service principal authentication. If you are your own Kerberos admin, then you will need to create this file for yourself.

Step 3. Create a Keytab file

The command `ktutil` creates the service principal keytab file which holds the encrypted credentials that the Fusion UI Proxy will use for Kerberos authentication. In order to generate the keytab file, you must have a set of cached credentials, therefore, first run the `kinit` command (step 2).

From the command line, run the command `ktutil`. You must enter your password twice.

```
> ktutil -k http-myrealm.org.keytab add -p HTTP/myrealm.org@MYORG.ORG
-e aes256-cts-hmac-sha1-96 -V 0
```

The -k argument specifies the name of the keytab file which will be created or updated. The command "add" takes the following argument flags:

- "-p" : service principal, format <service>/<fully.qualified.domain>@REALM
- "-e" : encryption type. Depending on the encryption type, you may need to download additional Java security libraries for strong encryption.

  Note

  The following encryption types are not supported by Fusion:
  - DES-CBC-CRC
  - DES_CBC_MD5
  - Microsoft Windows 2000 RC4-HMAC

- "-V" : key version number (kvno). Key version numbers are used in the Kerberos V5 protocol to distinguish between different keys in the same domain.

If successful, this command creates a keytab file called "http-myrealm.org.keytab". Note the directory you are in - you will need this full path when creating the Proxy realm-config later.

Step 4. Test the Keytab file

The location of this keytab file will be used to configure UI Proxy configuration. Before configuring the Fusion UI Proxy, you should check that the keytab file is valid. Testing the keytab requires the following sequence of steps:

- Clear any existing credentials via command `kdestroy`.

- Log in using the keytab as an argument to the command `kinit –kt <keytab file> <principal>`, where `<principal>` is the name of a principal within the keytab file.
- Examine your credentials via command `klist`.
- Clear credentials via command `kdestroy`, which removes any existing credentials, effectively logging you out of Kerberos.

To remove cached credentials, use the `kdestroy` command. This command succeeds silently. To check that credentials have been removed, re-run the `klist` command:

```
> kdestroy > klist klist: krb5_cc_get_principal: No credentials cache file found
```

Use the keytab to login as the service principal, without being prompted for a password:

```
> kinit –t http–myrealm.org.keytab HTTP/myrealm.org
```

Examine your credentials via the command `klist`. The output should be similar to this:

```
Credentials cache: API:51D488FF–5CD9–4E16–98FA–B47743F5B4ED Principal: HTTP/myrealm.org@MYORG.ORG Issued Expires Principal Apr 1 09:15:02 2015 Apr 1 19:13:42 2015 krbtgt/MYORG.ORG@MYORG.ORG
```

Logout again with `kdestroy`.

Configuring Fusion Authentication for Kerberos Realm

Once you have tested both the user and service principal logins, you must create the service principal realm-config in the Fusion Authentication Proxy. This allows the Proxy to authenticate to Kerberos as the service principal, without a password.

Step 5. Configure the Fusion Realm

Fusion security realms can be configured either via the Fusion UI Admin tool or the Fusion REST API. The advantage of using the Fusion UI Admin Tool is that a single panel's worth of configuration requires a series of calls to the REST API. It is important to understand the set of configuration properties collected by the Fusion UI and how they are used by the REST API.

A security realm for Kerberos has the following properties:

- name : unique string identifier
- realmType : "kerberos"
- enabled : whether or not the realm is available for users to use with system authentication
- config : this property is required for the realm type "kerberos". It takes two key-value pairs:
    - principal : the principal service name

- keytab : this must be the full path to the keytab file.

To configure Fusion via the Fusion UI:

1. Log in to Fusion as "admin" or as a user who has super-admin privileges.
2. In the left-hand side navigation bar, click **System > Access control**.
3. Click the **Security Realms** tab, then **Add Security Realm**.
4. On the New Security Realm form, choose type "kerberos" from the pulldown menu so that there are input boxes for the Kerberos realm properties "Service Principal" and "Keytab path".
5. Choose a realm name, and make sure the "enabled" box is checked.
6. If the **auto-created** box is unchecked, a user with admin privileges will need to create Fusion user accounts.
7. Select default roles. The default roles are assigned to a user the first time they access Fusion via this realm, using the Kerberos/SPNEGO protocol. For example, once you have defined a Kerberos realm "my-kerberos-realm" for domain "MYORG.ORG", when user "any.user" in domain "MYORG.ORG" authenticates to Fusion for the first time via this realm, they are added to the set of Fusion users as username "any.user@MYORG.ORG" and they have all default roles.
   It is prudent to allow the minimum set of default roles, as all users will have these permissions. Some users will require admin privileges and a few users will require super-admin privileges. There should always be a user with super-admin privileges that can authenticate to Fusion using the native security realm and can then grant permissions to individual users as needed.

   Note

   The Fusion proxy Kerberos realm works only with one Fusion host. For a multi-node Fusion cluster where each node resides on a different host, configure a separate realm for each host with a dedicated service/principal.

8. Enter the service principal and the full path to the keytab file.
9. Click **Save**. Fusion displays a confirmation message.

Kerberos/SPNEGO HTTP Authentication

SPNEGO provides a mechanism for extending Kerberos to applications that use the HTTP protocol including web browsers and the `curl` command-line utility.

Step 6. Configure the HTTP client

When a user sends a request to the Kerberized Fusion UI, a SPNEGO request (http[s]) is made. If the user is not already authenticated, the Fusion authentication proxy will yield a 401 status code and a Negotiate header. This status/header response triggers compatible clients to fetch a local ticket from their Kerberos "ticket tray". This ticket is then encoded and sent back to the Fusion. The Fusion authentication proxy will then decode the ticket, and perform a SPN.doAs(user) authentication request to the KDC/Authentication Service. Depending on the results, the proxy then successfully executes the original request (along with a session cookie) or a 401 (without the Negotiate). Clients can either choose to use the session cookie or continue authenticating on every request.

Configuring Web Browsers and `curl` for SPNEGO

The `--negotiate` option enables SPNEGO in `curl`.

IE and Safari require no additional configuration to use SPNEGO.

To configure Firefox, access the low level configuration page by loading the about:config page. Then go to the network.negotiate-auth.trusted-uris preference and add the hostname or the domain of the web server that is HTTP Kerberos SPNEGO protected (if using multiple domains and hostname use comma to separate them).

The Chrome browser must be launched from the command line with several added parameters.

To run Chrome on linux:

```
> google-chrome --enable-plugins --args\ --auth-server-
whitelist="*KERBEROS_DOMAIN"\ --auth-negotiate-delegate-
whitelist="*KERBEROS_DOMAIN"\ --auth-
schemes="basic,digest,ntlm,negotiate"
```

To run Chrome on a Mac:

```
> open 'Google Chrome.app' --args\ --auth-server-
whitelist="*ROGUECLOUD.COM"\ --auth-negotiate-delegate-
whitelist="*KERBEROS_DOMAIN"\ --auth-
schemes="basic,digest,ntlm,negotiate"
```

To run Chrome on Windows:

```
chrome.exe --auth-server-whitelist="*KERBEROS_DOMAIN"\ --auth-
negotiate-delegate-whitelist="*ROGUECLOUD.COM"\ --auth-
schemes="basic,digest,ntlm,negotiate"
```

For more information, see Using a Web Browser to Access an URL Protected by Kerberos HTTP SPNEGO and Authentication + Authorization.

Session cookies

A successful Kerberos/SPNEGO login will yield a session cookie, this cookie is identical to the cookie yielded by the Fusion authentication proxy's current POST-login-mechanism.

The expiration policy on the cookie is currently fixed at 8 hours. But has a 1 hour "idle" max, which means if you do not make a request for 1 hour, the cookie is invalidated. Otherwise the lifetime is pushed ahead until the 8 hour max is met.

The name of the cookie is "id" and the value is a UUID. This UUID is a key that maps to an in-memory value containing the real user ID.

Testing and Troubleshooting

Once you have configured the Kerberos security realm, you can test it as follows:

1. Log out of Fusion and shut down the browser.
2. Check that you have a valid Kerberos authentication ticket via the `klist` command.
3. Open a new browser session and access the Fusion installation.
   This should take you directly to the main Fusion panel, bypassing the Fusion "Welcome" login panel. To view your login profile, click on the profile icon at the top right. Your user name should be your login name

+ "@" + your domain name.

- If you see the Fusion login panel instead of the main Fusion panel, your browser is not configured for SPNEGO.
- If the Fusion display consists only of an empty top nav bar, this indicates an authentication failure. Check that the path to your keytab file is correct. Then check the Fusion logs.
- If your KDC uses a weak encryption method, Fusion rejects your login credentials. The following weak encryption types are disabled in Fusion:
    - DES-CBC-CRC
    - DES_CBC_MD5
    - Microsoft Windows 2000 RC4-HMAC

References and Tutorials

- https://en.wikipedia.org/wiki/Kerberos_%28protocol%29
- http://www.roguelynn.com/words/explain-like-im-5-kerberos/
- https://en.wikipedia.org/wiki/SPNEGO
  https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/6/html/Managing_Smart_Cards/Using_Kerberos.html#about-kerberos
- http://www.oracle.com/technetwork/articles/idm/weblogic-sso-kerberos-1619890.html -section "Install Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files"
- http://www.cisco.com/c/en/us/support/docs/security-vpn/kerberos/16087-1.html

Configure Fusion for Kerberos in Windows

How-to

> Compatible versions: 5.9
>
> Works with: *Fusion Server*, *Fusion*

To configure the Fusion UI service to use Kerberos for user authentication, you must create a Kerberos *security realm*.

Note

Kerberos support requires Fusion 5.9.5.

Kerberos is a system that provides authenticated access for users and services on a network. Instead of sending passwords in plaintext over the network, encrypted passwords are used to generate time-sensitive tickets that are used for authentication. SPNEGO provides a mechanism for extending Kerberos to Web applications through the standard HTTP protocol.

Kerberos uses symmetric-key cryptography and a trusted third party called a Key Distribution Center (KDC) to authenticate users to a suite of network services. (By users we mean both end users and client programs). The computers managed by that KDC and any secondary KDCs constitute a realm. When a user authenticates to the KDC, the KDC sends a set of credentials (a ticket) specific to that session back to the user's machine. Kerberos-aware services use the ticket on the user's machine for authentication instead of requiring sign-on with a password. Because tickets are used rather than passwords, this provides the convenience of Single Sign-On (SSO) in addition to security.

A Kerberized process is one that has been configured so it can get tickets from a KDC and negotiate with Kerberos-aware services. When a user sends an HTTP request, Fusion tries to authenticate using the Kerberos/SPNEGO protocol. If the request was sent from a browser, Fusion does not display the initial sign-on panel; instead on login, the user sees the main Fusion collections panel.

Important

This article focuses on configuring Fusion for Kerberos *in Windows*. To learn how to configure Fusion for Unix, see *Configure Fusion for Kerberos in Unix*.

Note

Kerberos support requires Fusion 5.9.5.

1. Set up and Configure the Active Directory Domain Service

1.1. Install the Active Directory Domain Service

To begin, follow Microsoft's instructions on how to install Active Directory Domain Services.

This topic uses the following domain throughout this example:

NETBIOS Domain	FUSIONIS
Fully qualified domain name	fusionis.life

1.2. Create a New User

Launch the Server Manager. Navigate to **Tools > Active Directory Users and Computers**. Within the `fusionis.life` server, locate the "Users" folder, right-click on the folder and select **New > User**.

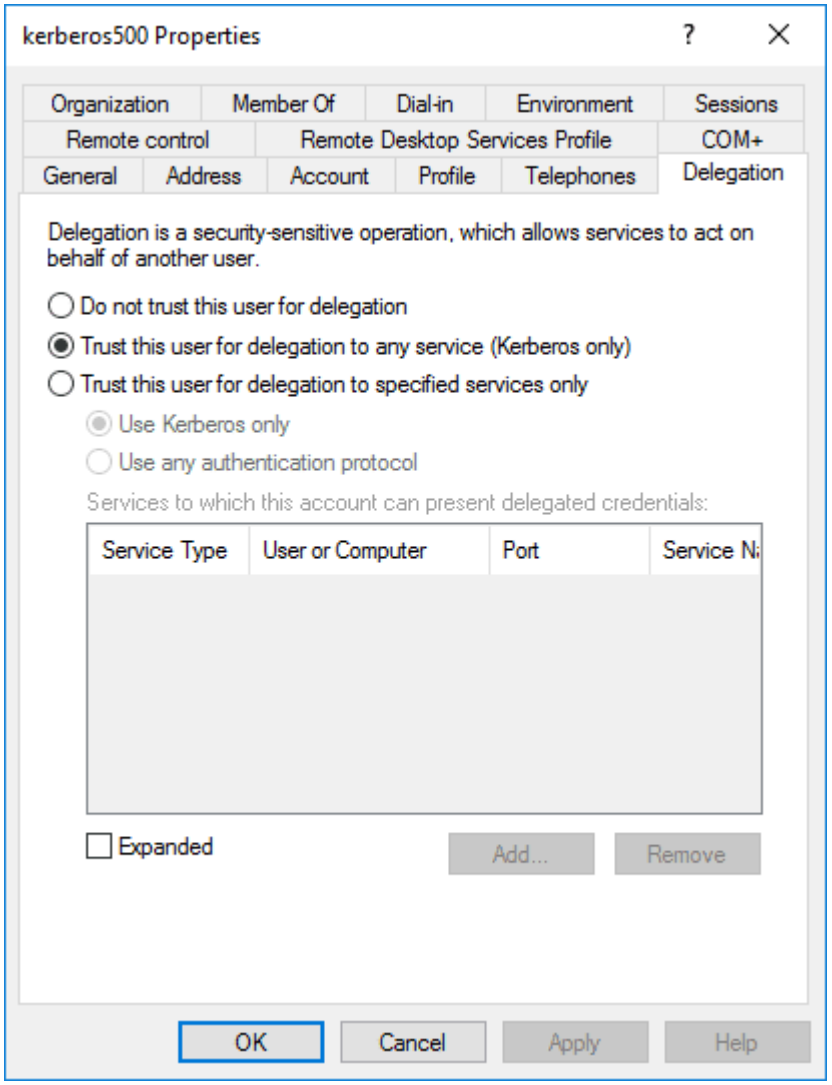
Important

When creating a user for this purpose, ensure the `First name` and `Display name` fields both hold the exact same value.

These instructions use the following user name throughout this example:

Distinguished name (DN)	CN = kerberos500 CN = Users DC = fusionis DC = life
User Principal Name (UPN)	kerberos500@fusionis.life
NETBIOS Domain\samAccountName	FUSIONIS\kerberos500

Establish the desired delegation rules for the new user in the "Delegation" tab.



1.3. Create a Keytab for the New User

Enter the following command into the command prompt to create a keytab for the user you just created:

```
ktpass -out c:\Users\Administrator\kerberos500.keytab -princ
HTTP/fusionis.life@FUSIONIS.LIFE -mapUser FUSIONIS\kerberos500 -mapOp
set -pass YOUR_PASSWORD -crypto AES256-SHA1 -pType KRB5_NT_PRINCIPAL -
kvno 0
```

+

Note

The following weak encryption types are not supported by Fusion:

- DES-CBC-CRC
- DES_CBC_MD5
- Microsoft Windows 2000 RC4-HMAC

The output will be similar to:

```
C:\Users\Administrator>ktpass -out
c:\Users\Administrator\kerberos500.keytab -princ
HTTP/fusionis.life@FUSIONIS.LIFE -mapUser kerberos500 -mapOp set -pass
FroFro123# -crypto AES256-SHA1 -pType KRB5_NT_PRINCIPAL Targeting
domain controller: WIN-OVV6VHBGIB8.fusionis.life Using legacy password
setting method Successfully mapped HTTP/fusionis.life to kerberos500.
Key created. Key created. Key created. Key created. Key created.
Output keytab to c:\Users\Administrator\kerberos500.keytab: Keytab
version: 0x502 keysize 71 HTTP/fusionis.life@FUSIONIS.LIFE ptype 1
(KRB5_NT_PRINCIPAL) vno 3 etype 0x1 (DES-CBC-CRC) keylength 8
(0xcd07200bea625d20) keysize 71 HTTP/fusionis.life@FUSIONIS.LIFE ptype
1 (KRB5_NT_PRINCIPAL) vno 3 etype 0x3 (DES-CBC-MD5) keylength 8
(0xcd07200bea625d20) keysize 79 HTTP/fusionis.life@FUSIONIS.LIFE ptype
1 (KRB5_NT_PRINCIPAL) vno 3 etype 0x17 (RC4-HMAC) keylength 16
(0xe39a141de38abd8750bf9c0bf49fd1c5) keysize 95
HTTP/fusionis.life@FUSIONIS.LIFE ptype 1 (KRB5_NT_PRINCIPAL) vno 3
etype 0x12 (AES256-SHA1) keylength 32
(0xdd6018ab718a4c312038189a20498f70c07d09a19d77ceea31cdd4cd1b08800a)
keysize 79 HTTP/fusionis.life@FUSIONIS.LIFE ptype 1
(KRB5_NT_PRINCIPAL) vno 3 etype 0x11 (AES128-SHA1) keylength 16
(0x9b610485eae2d97dcf0ecaf4ab38a32b)
```

This process will create the **`kerberos500.keytab`** file required to set up the Kerberos realm in Fusion. The command will also set the service principal to **`HTTP/fusionis.life@FUSIONIS.LIFE`**.

Important

If you see the following, the user was not created correctly:

```
DsCrackNames returned 0x2 in the name entry for
kerberos500. Ktpass:failed getting target domain for
specified user.
```

Delete the user, and repeat *the steps above*.

2. Set up Kerberos on Fusion Server

2.1. Install Fusion and set up the Kerberos Realm

If you have not done so already, *install Fusion now.*

Copy the keytab file to the server.

Note

The keytab file must be installed on each Fusion node.

Log into Fusion as a native admin and navigate to **DevOps > Access Control > Realms > New Realm**. Use the following parameters:

Type	Kerberos
Keytab file	The path to the keytab file you saved on the server
Service principal	The service principal you generated in the previous step
Kerberos name mapping	The field that specifies the mapping from a Kerberos Principal name to a short name

You can specify the default rule, which should work for most situations:

```
DEFAULT RULE:[1:$1@$0](.*@.*)s/@.*//
```

You can also add custom rules. See Mapping Kerberos Principals to Short Names for more information on creating custom rules.

Save your new realm before continuing. You should now be able test with Negotiate authentication.

3. Verify Kerberos is Working

You will test to verify that Kerberos is working in the following sections. You must log into Windows as a user in the protected realm, meaning that the service principal name you are using has access to generate a ticket for your user.

Verify with cURL

Windows users can log in as a user within the domain, launch the command prompt, and run the cURL command below:

```
curl --negotiate -u :
http://fusionis.life:8764/api/apollo/collections
```

Linux users may also need to run a `kinit` to login as a user prior to running.

Navigate your web browser to `http://fusionis.life:8764`. If you are automatically logged in, Kerberos is working.

Verify with Google Chrome

To verify Kerberos is working with Google Chrome, run Chrome with the following command parameters:

```
"c:\Program Files (x86)\Google\Chrome\Application\chrome.exe" --auth-
server-whitelist="*fusionis.life" --auth-negotiate-delegate-
whitelist="*fusionis.life" --auth-
schemes="basic,digest,ntlm,negotiate"
```

Navigate your web browser to `http://fusionis.life:8764`. If you are automatically logged in, Kerberos is working.

Verify with Firefox

To verify Kerberos is working with Firefox, navigate to `about:config` in your browser's URL bar. If a message appears stating "This might void your warranty!", click "I accept the risk!".

Search for the string `network.negotiate-auth.trusted-uris`, double-click the field, and enter `fusionis.life` into the field that appears. Click "Ok".

Navigate your web browser to `http://fusionis.life:8764`. If you are automatically logged in, Kerberos is working.

Troubleshooting

DEFECTIVE TOKEN DETECTED

If you encounter this issue, check the `var/log/proxy/proxy.log` after attempting to make a request.

See Stack Overflow for addtional steps to resolve this issue.

KERBEROS NAME MAPPING RULES

You may see the following error if you did not specify a mapping rule and the Proxy could not figure out how to map the Kerberos Principal name to a short name:

```
2021-07-30T15:15:03,584 - ERROR [qtp353842779-
52:clojure.tools.logging$eval11$fn__16@0] - {} - Uncaught application
error!
org.apache.hadoop.security.authentication.util.KerberosName$NoMatching
Rule: No rules applied to
YOURUSERNAME@YOURDOMAIN.SUBDOMAIN.EXAMPLE.COM
```

Adjust your Kerberos name mapping rules to resolve this issue.

Configure Fusion for LDAP

How-to

> Compatible versions: 4.2
>
> Works with: *Fusion Server*

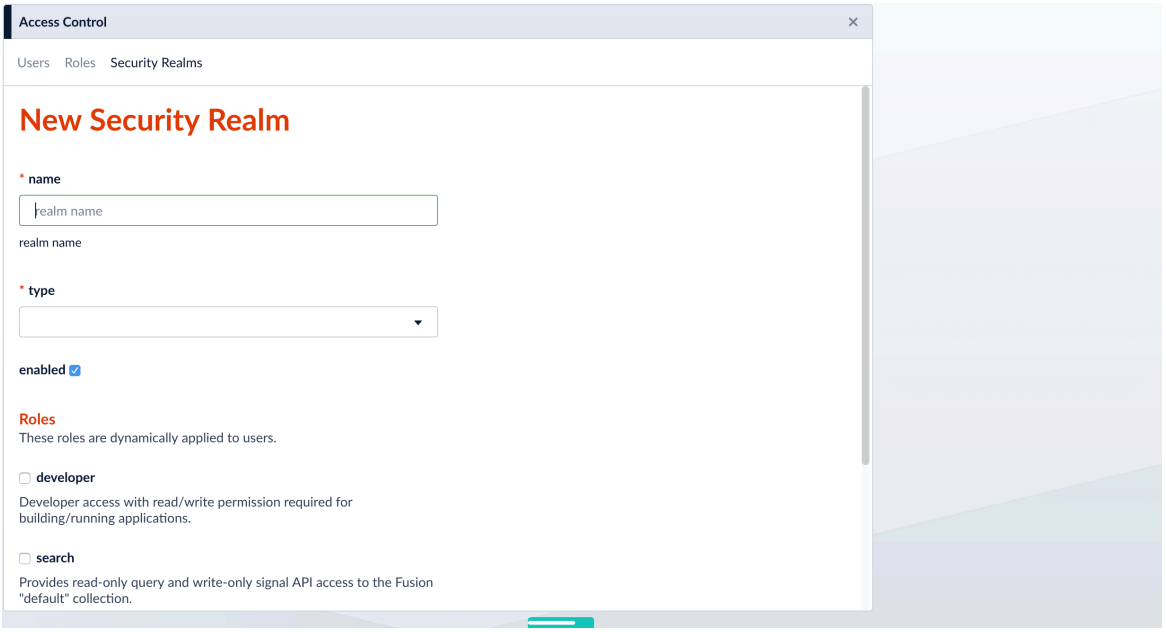
You can create security realms that use external LDAP servers for authentication. Optionally, Fusion can search in the LDAP for groups to which a user belongs, and then map those groups to Fusion roles. Fusion performs authorization using permissions stored in Fusion users and Fusion roles.

Note: Fusion *does not* use permissions from the LDAP for authorization of UI access or API requests. It only obtains group names (optionally), which are mapped to role names. If an *Active Directory Security Query Trimming Stage* is used, then directory-service permissions *are* used for trimming. If a connector supports security trimming, then connector permissions *are* used for trimming.

To configure Fusion to use an external LDAP as an authentication provider, you'll need to get information about the LDAP server(s) running on your system, either from your system or your sysadmin.

1. Add an LDAP Security Realm

1. Log in to the Fusion UI as the user admin, or as a different user with corresponding permissions.
2. Navigate to **System** > **Access Control**.
3. Click **Security Realms**.
4. Click **Add Security Realm**.



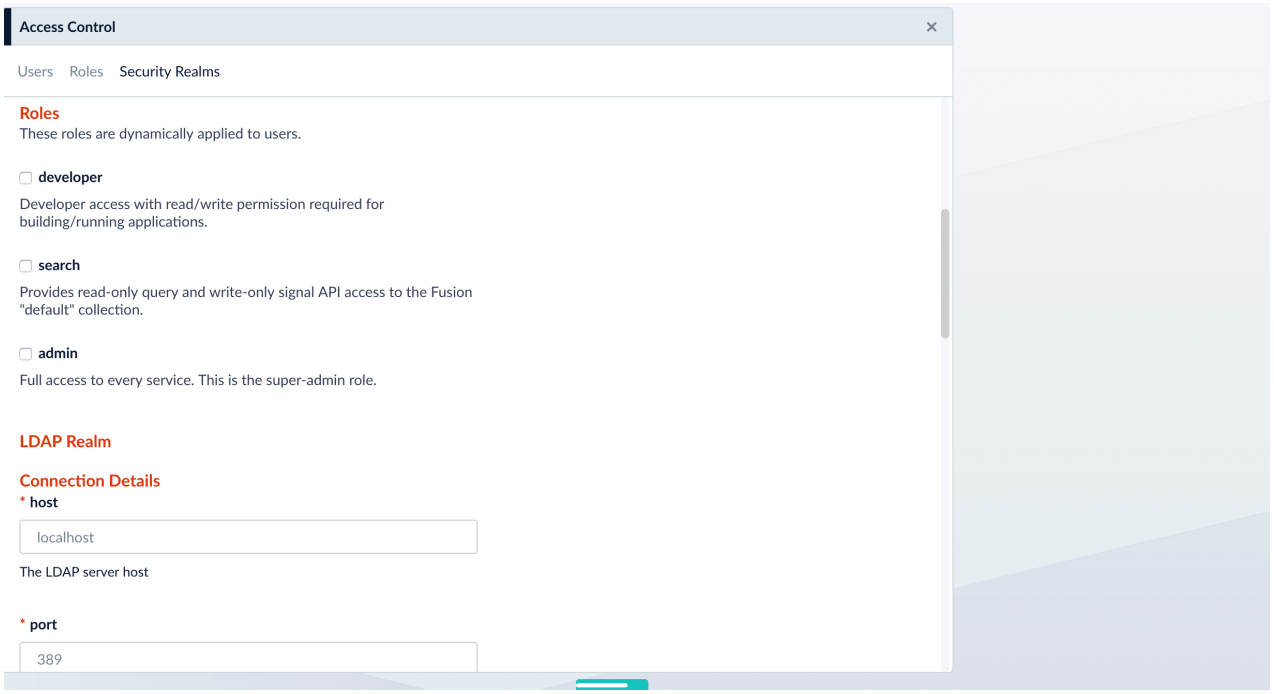
5. Specify info for the new realm:
   - `name` – Name of the security realm. It must be unique. It should be descriptive but short.
   - `type` – Select **ldap** from the pulldown menu.
     When you select **ldap**, Fusion displays additional, LDAP-specific configuration options.
   - `enabled` checkbox – Whether Fusion allows user logins for this security realm. The default is yes (checked).
   - `auto-create users` checkbox – Whether a user account is created automatically upon initial authentication. The default is yes (checked). If the checkbox is unchecked, then a Fusion user with admin permissions must create Fusion users.

6. Scroll down and specify additional options as explained in detail below.

2. Specify Static Roles (Optional)

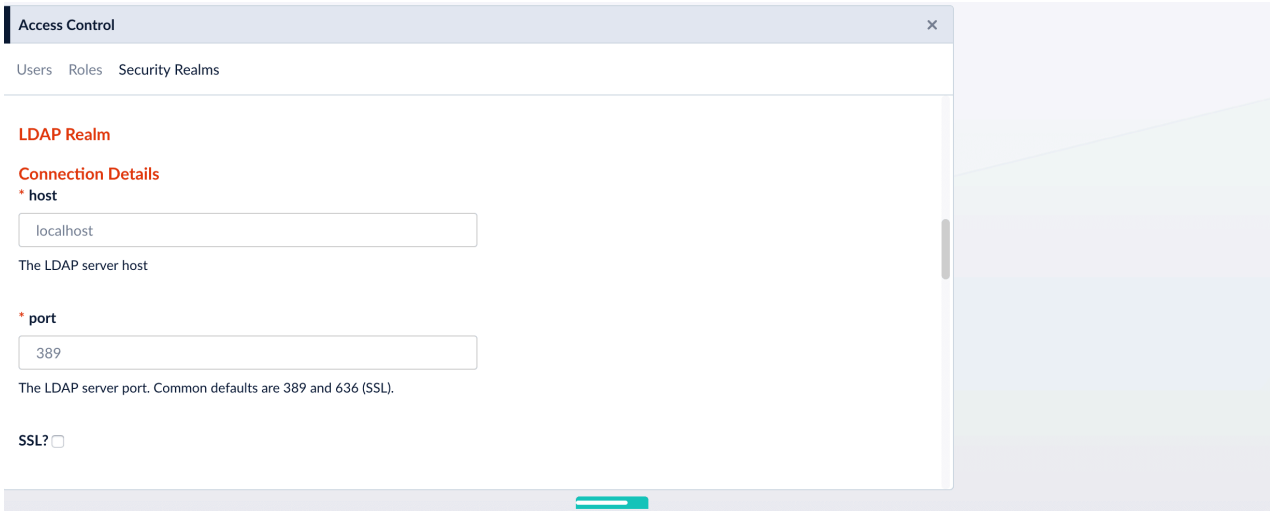
Specify one or more Fusion roles for the security realm. These roles are always considered. They do not depend on searching for LDAP groups and mapping group names to Fusion role names.

In a security realm, you can specify these static roles, add to the list of roles dynamically through an LDAP search, or both. If you do neither, Fusion uses only the role(s) and permissions defined for the user.



3. Specify LDAP Connection Details

Specify the hostname and port of the LDAP server. Check the checkbox if the server is running over SSL.



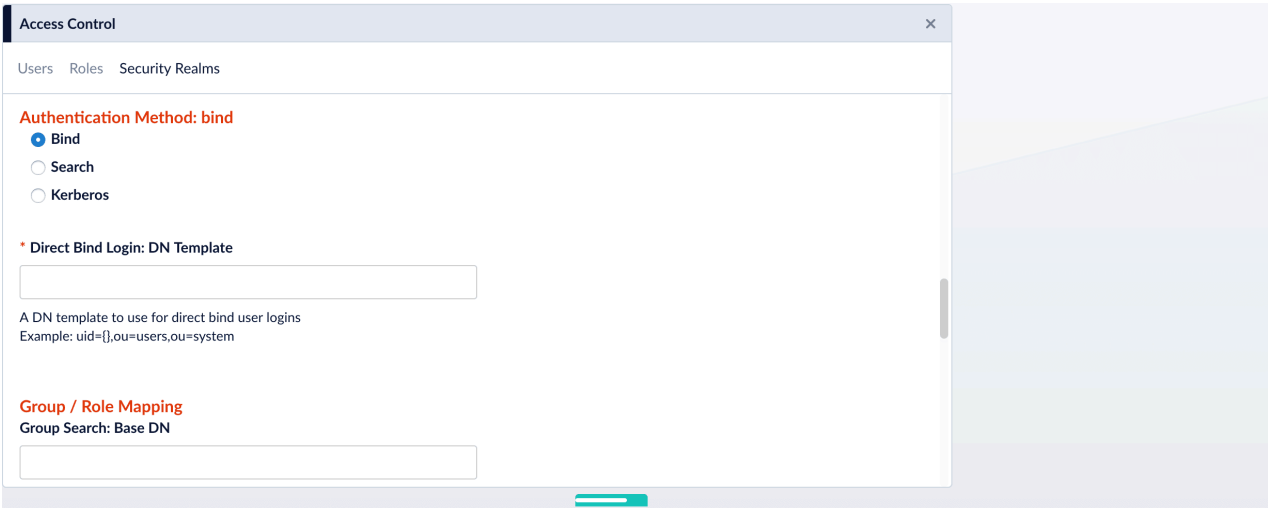
4. Specify the Authentication Method

Specify the authentication method:

- **Bind.** LDAP authentication is carried out via a single "Bind" operation. See *Bind* below.
- **Search.** LDAP authentication is carried out indirectly via a Search operation followed by a Bind operation. See *Search* below.
- **Kerberos.** Kerberos authenticates Fusion and an LDAP Search operation is carried out to find group-level authorizations. See *Kerberos* below.

Bind

Use the Bind authentication method when the Fusion login username matches a part of the LDAP distinguished name (DN). Specify the remainder of the LDAP DN in the "DN Template" configuration entry, which uses a single pair of curly braces (`{}`) as a placeholder for the value of the Fusion username.

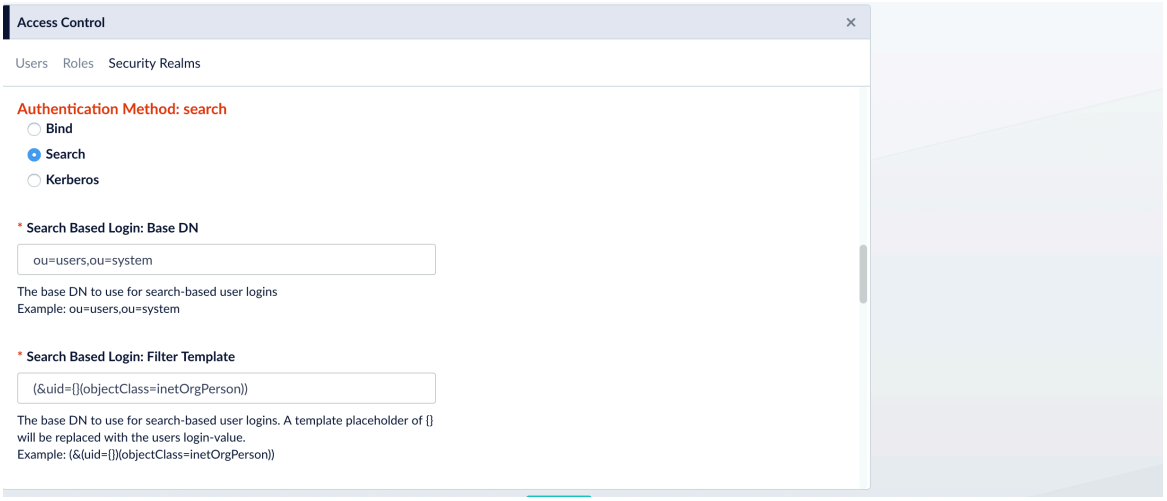


Search

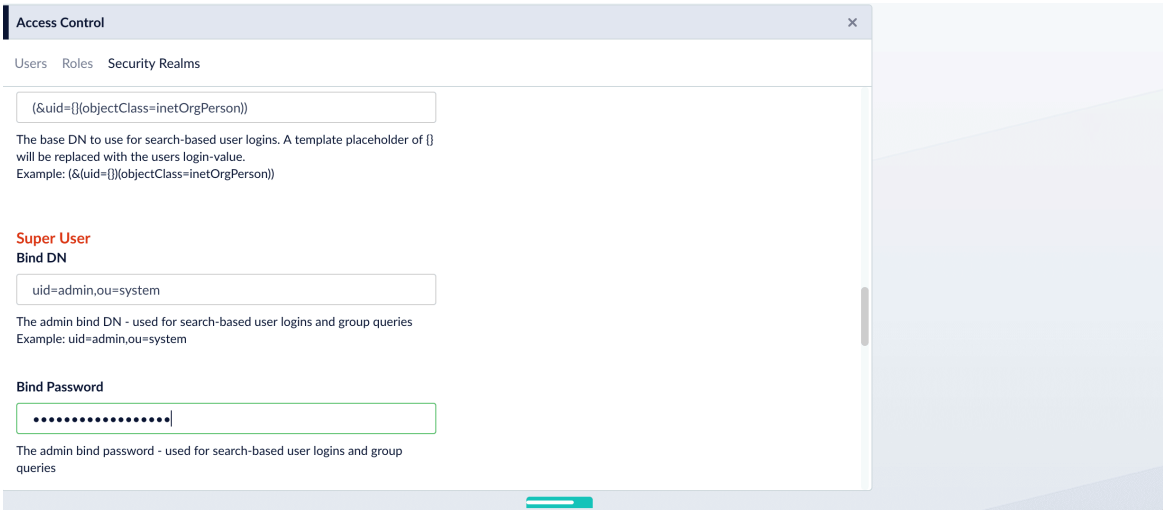
Use the Search authentication method when the username used for Fusion login *doesn't* match a part of the LDAP DN. The search request returns a valid user DN, which is then used together with the user password for authentication via a Bind request.

1. Construct a search request.
   The Search authentication method is generally required when working with Microsoft Active Directory servers. In this case, you need to know the username and password of *some* user who has sufficient privileges to query the LDAP server for user and group memberships; this user doesn't have to be the superuser.

   In addition to a privileged user DN and password, the Search authentication method requires constructing a search request. There are two parts to the request. The first part is the base DN of the LDAP directory tree that contains user account objects. The second part of the request is a Search Filter object that restricts the results to a matching subset of the information.

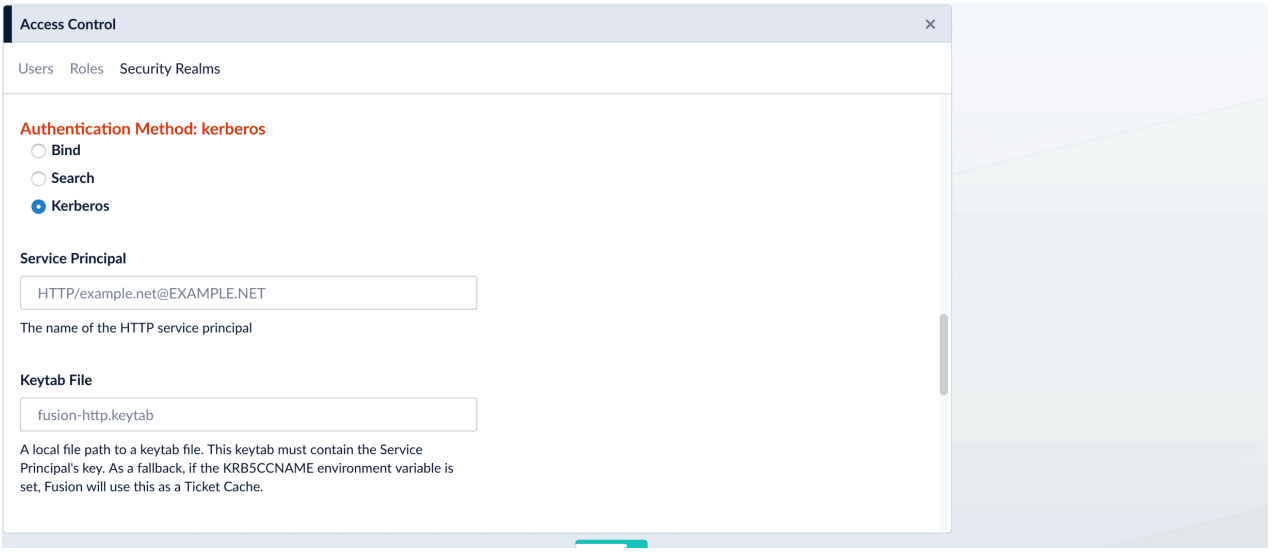


2. Provide the administrator bind DN:



Kerberos

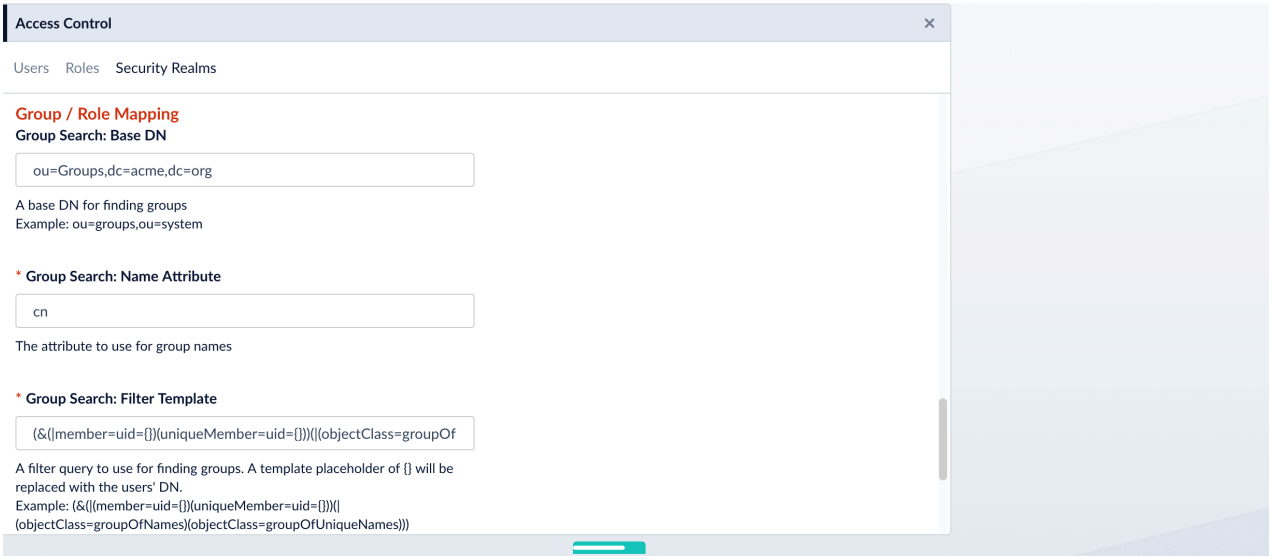
Use the Kerberos authentication method when Kerberos is the authentication provider.



5. Search for LDAP Groups (Optional)

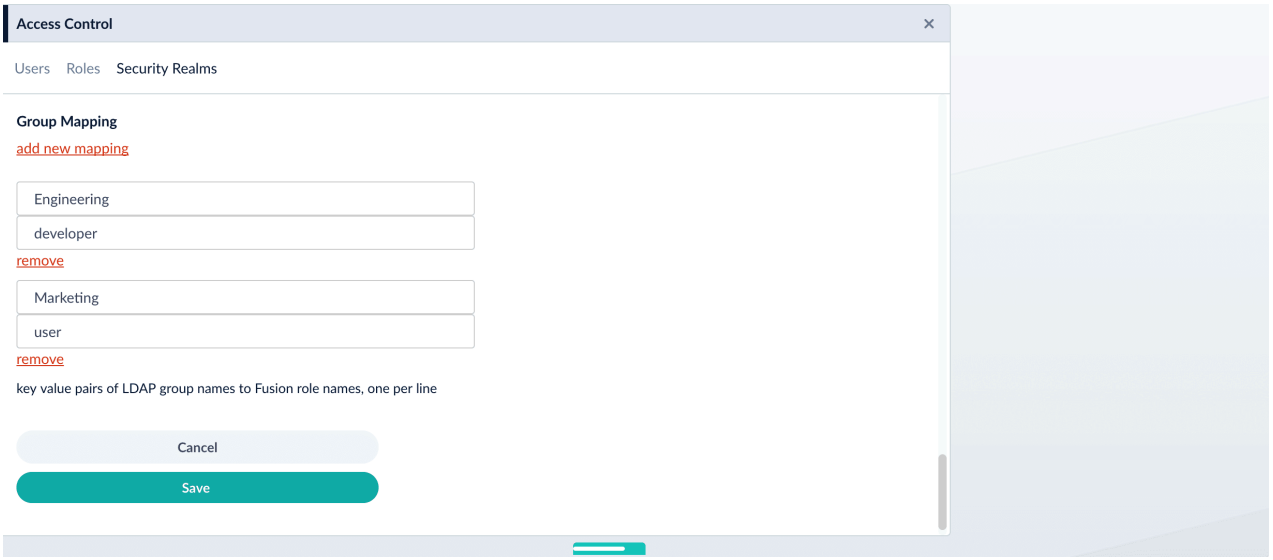
A Fusion role is a bundle of permissions tailored to the access needs of different kinds of users. Access to services and data for LDAP-managed users is controlled by mappings from LDAP users and groups to Fusion roles.

Roles can be assigned globally or restricted to specific LDAP groups. The security realm configuration panel contains a list of all Fusion roles with a checkbox for each, used to assign that role to all users in that realm. LDAP group names can be mapped directly to specific Fusion roles and LDAP group search and filter queries can also be used to map kinds of LDAP users to specific Fusion roles.



6. Map LDAP Groups to Fusion Roles (Optional)

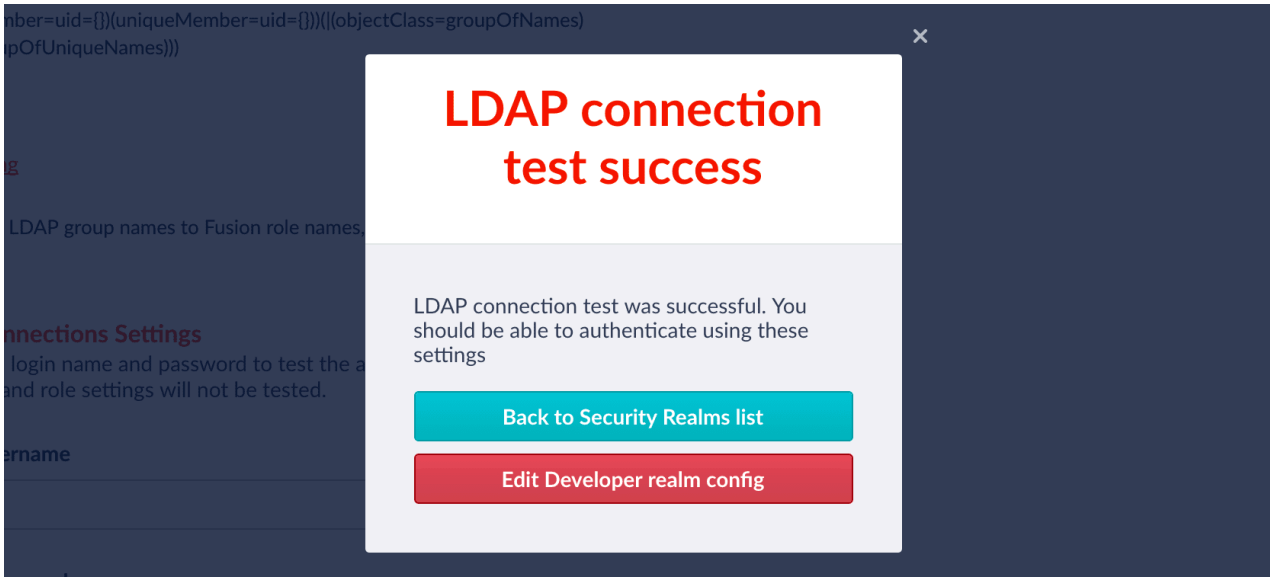
If LDAP group names returned by the search for groups match Fusion role names, you do not need to map the group names to role names. You must map any LDAP group names that do not match to Fusion role names (if you do not, they will not be used).



7. Save the Security Realm Configuration

Click **Save**.

Fusion reports whether or not authentication was successful:



8. Basic LDAP Concepts and Terminology

The LDAP protocol is used to share information about users, systems, networks, and services between servers on the internet. LDAP servers are used as a central store for usernames, passwords, and user and group permissions. Applications and services use the LDAP protocol to send user login and password information to the LDAP server. The server performs name lookup and password validation. LDAP servers also store Access Control Lists (ACLs) for file and directory objects which specify the users and groups and kinds of access allowed for those objects.

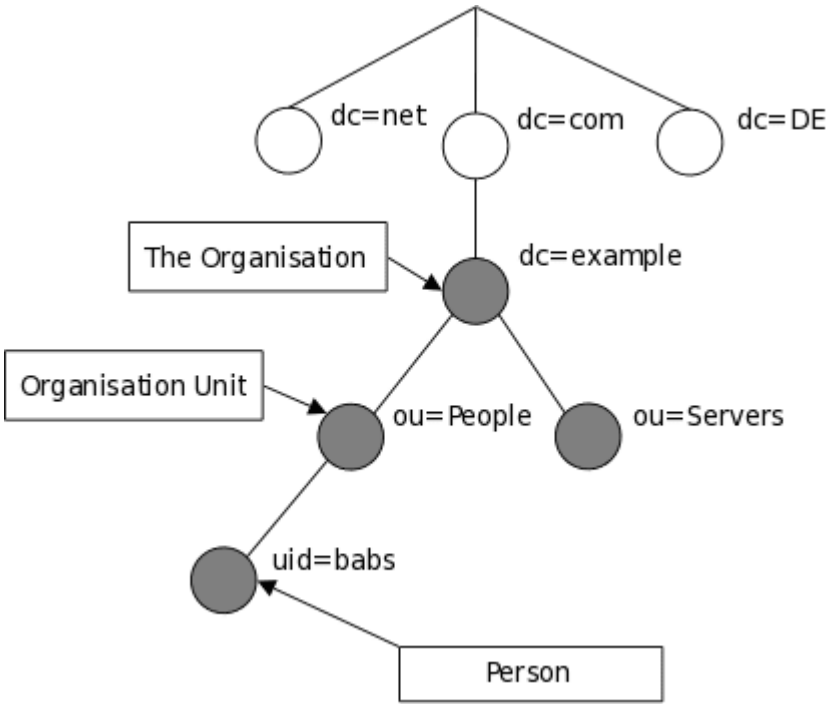
LDAP is an open standard protocol and there are many commercial and open source LDAP servers available. Microsoft environments generally use Active Directory. Unix servers use AD or other LDAP systems such as OpenLDAP, although many Unix systems do not use LDAP at all. To configure Fusion for LDAP, you'll need to get information about the LDAP server(s) running on your system either from your sysadmin or via system utilities.

Directories and Distinguished Names

An LDAP information store is a Directory Information Tree (DIT). The tree is composed of entry nodes; each node has a single parent and zero or more child nodes. Every node must have at least one attribute which uniquely distinguishes it from its siblings which is used as the node's Relative Distinguished Name (RDN). A node's Distinguished Name (DN) is a globally unique identifier.

The string representation of a DN is specified in RFC 4514 . It consists of the node's RDN followed by a comma, followed by the parent node's DN. The string representation of the RDN is the attribute-value pair name, connected by an equals ("=") sign. This recursive definition means that the DN of a node is composed by working from the node back through its parent and ancestor nodes up to the root node.

Here is a small example of a DIT:

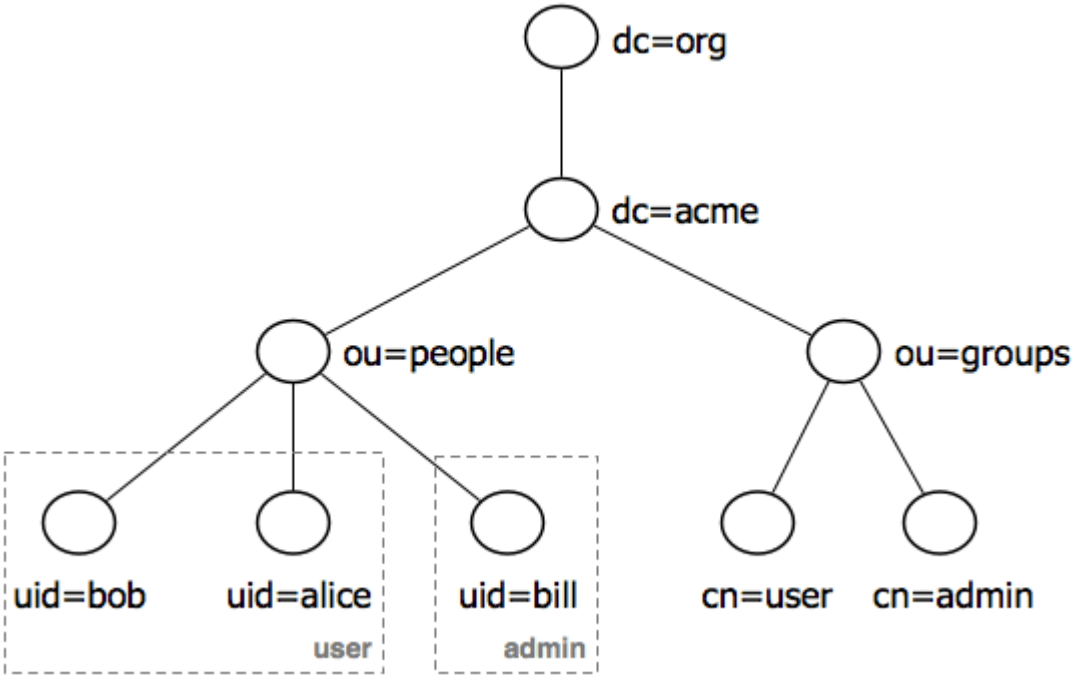


The person entry in this tree has the DN: "uid=babs, ou=people, dc=example, dc=com".

Attribute names include many short strings based on English words and abbreviations, e.g.:

Name	Description
cn	commonName
dc	domainComponent
mail	email address
ou	organizationalUnitName
sn	surname
uid	userId

LDAP entry attributes can refer to other LDAP entries by using the DN of the entry as value of that attribute. The following example of a directory which contains user and groups information shows how this works:



This tree contains two organizational units: "ou=people" and "ou=groups". The children of the "group" organizational unit are specific named groups, just as the child nodes of organization unit "people" are specific users. There are three user entries with RDNs "uid=bob", "uid=alice", "uid=bill" and two groups with RDNs "cn=user" and "cn=admin". The dotted lines and group labels around the person nodes indicate group membership. This relationship is declared on the groups nodes by adding an attribute named "member" whose value is a users DN. In the LDAP data interchange format (LDIF), this is written:

```
cn=user,ou=groups,dc=acme,dc=org member:
uid=bob,ou=people,dc=acme,dc=org member:
uid=alice,ou=people,dc=acme,dc=org cn=admin,ou=groups,dc=acme,dc=org
member: uid=bill,ou=people,dc=acme,dc=org
```

See the Wikipedia's LDAP entry for details.

LDAP Protocol Operations

For authentication purposes, Fusion sends Bind operation requests to the LDAP server. The Bind operation authenticates clients (and the users or applications behind them) to the directory server, establishes authorization identity used for subsequent operations on that connection, and specifies the LDAP protocol version that the client will use.

Depending on the way that the host system uses LDAP to store login information about users and groups, it may be necessary to send Search operation requests to the LDAP server as well. The Search operation retrieves partial or complete copies of entries matching a given set of criteria.

LDAP filters specify which entries should be returned. These are specified using prefix notation. Boolean operators are "&" for logical AND, "|" for logical OR, e.g., "A AND B" is written "(&(A)(B))". To tune and test search filters for a Unix-based LDAP system, see the ldapsearch command line utility documentation. For Active Directory systems, see AD Syntax Filters.

Configure Fusion for SAML

How-to

> Compatible versions: 4.2, 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion Server*, *Fusion*

SAML 2.0 is a standard for exchanging authentication and authorization data between security domains. The SAML protocol allows web-browser single sign-on (SSO) through a sequence of messages sent to and from the browser, which is the intermediary between Fusion and the SAML authority acting as the Identity Provider (IdP).

To configure Fusion to use SAML 2.0 for user authentication and authorization you must create a SAML *security realm*. In addition to configuring the Fusion security realm, you must configure the SAML identity provider to recognize the Fusion application.

Once Fusion is configured for a SAML realm, this realm is added to the list of available realms on the initial Fusion sign-on panel. When the SAML realm is chosen from the list of available realms, the browser then redirects to the IdP which handles user authentication. Upon successful authentication, the IdP sends a response back to the browser which contains authentication and authorization information as well as the URL of the Fusion application. The browser redirects back to the Fusion URL, passing along the SAML message with the user authentication and authorization information. Fusion then issues as session cookie which is used for subsequent user access.

Add a Security Realm

1. In the Fusion workspace, navigate to **System** > **Access Control**.
2. Click **Security Realms**.
3. Click **Add Security Realm**.
4. Enter the following information for the new realm:
    - Enter a **Name**. The name must be unique and should be descriptive yet short.
    - Select **saml** from the **Type** pulldown menu.
      When you select a type, Fusion displays additional configuration options.
    - The default value for **Enabled** is true. This setting controls whether or not Fusion allows user logins for this security realm.
    - The default value for **Ephemeral Users** is false. When disabled, this setting prevents ephemeral users from being created in ZooKeeper during login. If enabled, this property negates **Auto Create Users**.
    - The default value for **Auto Create Users** is true. If enabled, a user account is created automatically upon initial authentication. If disabled, then a Fusion user with admin permissions must create Fusion users.

5. Under **SAML Realm**, configure the following information:
    - Enter the **Identity Provider URL**. This URL is used by the SAML authority for single sign-on. For example: `https://www.my-idp.com/<my-app-path>/sso/saml`

      Note

      The URL format may differ depending on the SAML Identity Provider.

    - Enter the URL of the **IdP Issuer**. For example: `http://www.my-idp.com/exk686w2xi5KTuSXz0h7`.
        - **IdP Issuer** must match `<saml:Issuer>` in the SAML payload.
    - Optionally, provide the **App Issuer**. This field is required if there is an `audienceRestriction` in the SAML assertion and must match `<saml:Audience>` in the SAML payload.
    - In the **Certificate Fingerprint**, paste the contents of the SAML authority certificate, *without the certificate header and footer*.
    - Optional: Enter the **User ID Attribute**. By default, the Fusion username is the same as the login name known to the Identity Provider. When another field or attribute in the user record stored by the IdP should be used as the Fusion username, that attribute name is the value of the User ID Attribute.
    - Optional: Provide a **Post Login Redirect URL**. If not set, the Fusion URL is used.
    - Optional: Provide a **Logout URL**.

6. Optional: Under **Groups Mapping**, specify the Group Name Attribute and add group mappings.
7. Click **Save**.

Configure SAML identity provider

To finish setup, register Fusion with your SAML identity provider. The amount of information required to register varies depending on the SAML authority.

All systems require the Fusion URL to redirect to upon successful login. This includes the protocol, server, and port for the Fusion application, and path `api/saml`, such as `https://www.my-fusion-app.com:8764/api/saml`. If the Fusion application is running behind a load-balancer, then this URL is the load-balancer URL plus path `api/saml`. Note that the load-balancer should be session-sticky in order for the sequence of messages that comprise the SAML protocol to run to completion successfully.

Some authorities may require additional information. In particular, the SAML 2.0 "AudienceRestriction" tag may be part of the SAML message. This tag specifies the domain for which the SAML trust conditions are valid, which is usually the domain in which the Fusion app is running, such as `https://www.my-fusion-app`.

Example SAML Realm configuration

The Fusion endpoint `api/realm-configs` returns a JSON list of all the configuration objects for all realms. After configuring a SAML realm named `"saml-test"` using the okta.com developer preview tool, the configuration object for this realm is:

```
{ "name":"saml-test", "realmType":"saml", "enabled":true, "config":{
"autoCreateUsers":true, "idpUrl":"https://dev-
417804.oktapreview.com/app/dev417804_1/exk686w2xi5KTuSXz0h7/sso/saml",
"issuer":"http://www.okta.com/exk686w2xi5KTuSXz0h7",
"certificateFingerprint":"MIIDpDCCAoygAwIBAgIGAVQr4A4NMA0GCSqGSIb3DQEB
BQUAMIGSMQswCQYDVQQGEwJVUzETMBEG\nA1UECAwKQ2FsaWZvcm5pYTEWMBQGA1UEBwwN
U2FuIEZyYW5jaXNjbzENMAsGA1UECgwET2t0YTEU\nMBIGA1UECwwLU1NPUHJvdmlkZXIx
EzARBgNVBAMMCmRldi00MTc4MDQxHDAaBgkqhkiG9w0BCQEW\nDWluZm9Ab2t0YS5jb20w
HhcNMTYwNDE5MDAxNTI0WhcNMjYwNDE5MDAxNjI0WjCBkjELMAkGA1UE\nBhMCVVMxEzAR
BgNVBAgMCkNhbGlmb3JuaWExFjAUBgNVBAcMDVNhbiBGcmFuY2lzY28xDTALBgNV\nBAoM
BE9rdGExFDASBgNVBAsMC1NTT1Byb3ZpZGVyMRMwEQYDVQQDDApkZXYtNDE3ODA0MRwwGg
YJ\nKoZIhvcNAQkBFg1pbmZvQG9rdGEuY29tMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMI
IBCgKCAQEA\nuNz0trRFPw2d4xXoUvX2oWeZolTVeFtaTnB9SWUyjK4og0WdT7rNdBg10e
TvB2ezBwXCf24UGGui\nr1kjkZjiHDqDxKtzQYWpGuzLCjh/4PxKFGDaiUNKcE1Ig5myiE
BTvMvv99XtrcI75QdUGDhbMiBr\n2PR5FukWOYepzlBzqY0JSDzX9NYJBKPkz+syK4mj0I
6dqtYOU+bcTvjF9sR7jiHtQ+d0Zl8rz1Ca\nyuE3mNUtFJ1IJrY/RArhH1AB6mXbV/de1C
XmGhKQbqQAbx9SiKtki9n84gKEwuWdV0jIqcBLGxUQ\ngjbsaIVqed2oX+7F2fh6t0g/I8
NPnOWXOTvA+wIDAQABMA0GCSqGSIb3DQEBBQUAA4IBAQCPt+DR\nliIsHO/iVnmLFPGfqr
CO/gMv++xnq2wOB19YX7HhT1GIY2YZoVphrQpXH3OO0/8AZJ8ApCmq0E9x\nxUTQwQPBan
VqlyLtu1Hr0c6dbAqcd5PtaEe6Ci33nayWPydhOmitvIyb/WtWtbel9HcdUkoGvkAl\ng3
05jnxkhwGLJm4jkzYe+eaYhd6oG3/JcHqKDGYGf7i2Z+ny0D7vxTeBQ+8PbZfsUKg0PlKy
Tocb\nydSmDPISsA1xOH5zlw+hzFIdKgD5vW7QZLvpIclNc2hqki/nWl/CHut0TnUuP/V3
boREmhDu395n\n/u72pgNANfP7+2DTBb+CBTjGUsAxpKRF", "userIdAttribute":""
}, "roleNames":["search"], "id":"52e1c0d2-5e00-4c76-a3d4-
57f1381bdb4f", "createdAt":"2016-04-19T19:49:04Z", "updatedAt":"2016-
04-19T20:06:56Z" }
```


References

- http://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf
- http://docs.oasis-open.org/security/saml/v2.0/saml-glossary-2.0-os.pdf

Configure Fusion for SSO

How-to

> Compatible versions: 4.2, 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion Server*, *Fusion*

The "SSO Trusted HTTP" realm type (`trusted-http` in the REST API) is useful in single sign-on (SSO) environments.

If SSO is already set up in your environment, user identities and group information can be sent to Fusion through HTTP headers (REMOTE_USER, for example). The SSO Trusted HTTP realm type provides the configuration options for integrating this into Fusion's authentication systems. It also supports allowing access to only a set of known client IPs, and mapping groups to Fusion roles.

Use the *Realms API* to configure this realm type:

```
curl -u USERNAME:PASSWORD -H 'content-type:application/json' -X POST
:3000/api/realm-configs -d @./realm-config.json
```

Below is a sample configuration:

```
{"id":"test-id", "enabled":true, "name":"sso-test",
"realmType":"trusted-http", "config":{"identityKey":"REMOTE_USER",
"groups": {"key":"GROUPS", "delimiter":"|", "roleMapping":
[["a","admin"], ["b","foo"]]}, "allowedIps":["127.0.0.1",
"0:0:0:0:0:0:0:1", "localhost"]}}
```

`identityKey`	The name of an HTTP header. If this key is found in the request headers, its value is used as the identity of the client (username, for example).
`groups`	Configuration keys for auth groups: * `key` + The name of an HTTP header, used as the source of group names. * `delimiter` + The character used to split the value (defaults to comma). * `roleMapping` + A set of 2-tuples, used for mapping the external group values to Fusion Roles.
`allowedIps`	Allow access to only a set of known client IPs. When this property is defined and the client IP is not included in it, the realm logic return a 401. In Fusion 5.0.0+, leaving this field empty makes the realm nonoperational. To accept traffic from all destinations for development, you can use an IP such as 0.0.0.0/0. In production the concrete IP should be specified. Prior to Fusion 5.0.0, the `X-FORWARDED-FOR` header is inspected for this client IP first; the value is split on comma, and the first entry is taken. This would normally be used in cases where the client was forwarded to Fusion through one or more external proxy servers. If the `X-FORWARDED-FOR` header is not present in the request, the `REMOTE-ADDR` header value is used instead.

Configure Fusion Logging in Fusion 4.1 and 4.2

How-to

> Compatible versions: 4.2
>
> Works with: *Fusion Server*

Fusion uses the Apache Log4j 2 logging framework with Jetty to log each of the Fusion components.

Logging is configured with an XML configuration file named `log4j2.xml`. Log levels, frequencies, and log rotation policy can be configured by changing these configuration files:

API service	`https://FUSION_HOST:FUSION_PORT/conf/api-log4j2.xml`
Connectors	`https://FUSION_HOST:FUSION_PORT/conf/connectors-classic-log4j2.xml` `https://FUSION_HOST:FUSION_PORT/conf/connectors-rpc-log4j2.xml`
Solr	`https://FUSION_HOST:FUSION_PORT/conf/solr-log4j2.xml`

Spark	`https://FUSION_HOST:FUSION_PORT/conf/spark-driver-launcher-log4j2.xml`
	`https://FUSION_HOST:FUSION_PORT/conf/spark-driver-log4j2.xml`
	`https://FUSION_HOST:FUSION_PORT/conf/spark-driver-scripted-log4j2.xml`
	`https://FUSION_HOST:FUSION_PORT/conf/spark-executor-log4j2.xml`
	`https://FUSION_HOST:FUSION_PORT/conf/spark-master-log4j2.xml`
	`https://FUSION_HOST:FUSION_PORT/conf/spark-worker-log4j2.xml`

SQL	`https://FUSION_HOST:FUSION_PORT/conf/sql-log4j2.xml`
ZooKeeper	`https://FUSION_HOST:FUSION_PORT/conf/zookeeper-log4j2.xml`
Other Fusion services	`https://FUSION_HOST:FUSION_PORT/conf/proxy-log4j2.xml`

The Log4j2 Configuration guide provides documentation and examples of all logging configuration options.

The Fusion log shipper parses Fusion Java, HTTP, and garbage collector logs and sends them to a *system collection*. This system collection is used for *dashboards* in the Fusion UI.

Log shipping is enabled by default. You can disable log shipping, adjust which logs are parsed, or send log files to a custom collection name. For the configuration and details, see `https://FUSION_HOST:FUSION_PORT/conf/fusion.cors` (`fusion.properties` in Fusion 4.x) (on Unix or macOS) or `fusion\latest.x\conf\fusion.cors` (`fusion.properties` in Fusion 4.x) (on Windows).

Configure Google Drive Authentication

How-to

> Compatible versions: 4.2, 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion Connectors*, *Fusion Server*, *Fusion*

The Google Drive connector is used to index the documents in a Google Drive account.

There are two methods of Google Drive authentication for Fusion:

- *authentication for access to site-wide documents*
  Use a G-Suite account with admin-level access rights to configure access to all shared documents owned by the users in your organization.
- *authentication for access to per-user documents*
  This type of authentication gives you access to your own documents in Google Drive. Admin-level access rights are not required, but Fusion must be running in an authorized domain .

In both cases, you will get a client ID, client secret, and refresh token from Google. These become part of your datasource configuration in Fusion.

Authentication for access to site-wide documents

Important

When connecting to Google Drive through a service account, the **Viewed By Me timestamp is updated** for any fetched files.

In order to access all the shared documents by users in your organization, you must configure the Google Drive API and the Admin SDK. See the instructions in the knowledge base .

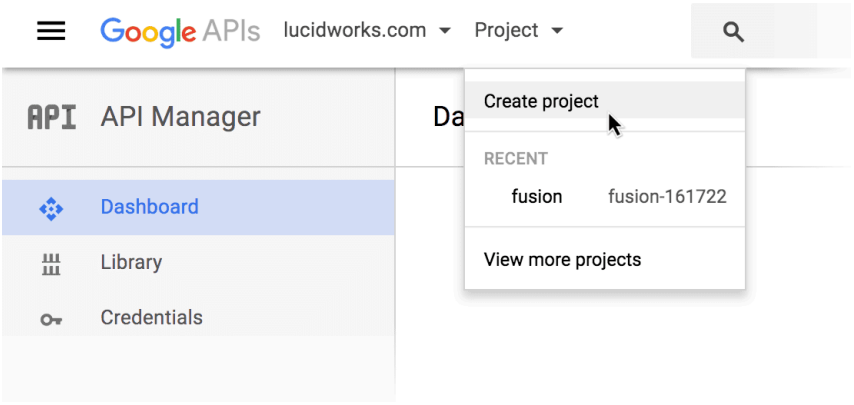
Note

You need a G-Suite account in order to perform these steps.

How to configure authentication for access to site-side documents

1. Log in to Google as a user with *admin-level access rights*.
2. Go to https://console.developers.google.com/ .
3. Create a Google project for Fusion:

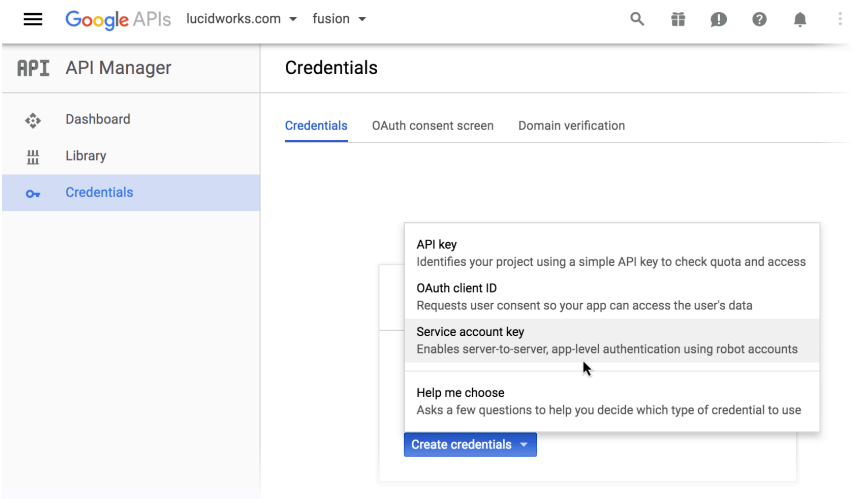
1. In the upper left, open the **Project** menu and select **Create Project**:



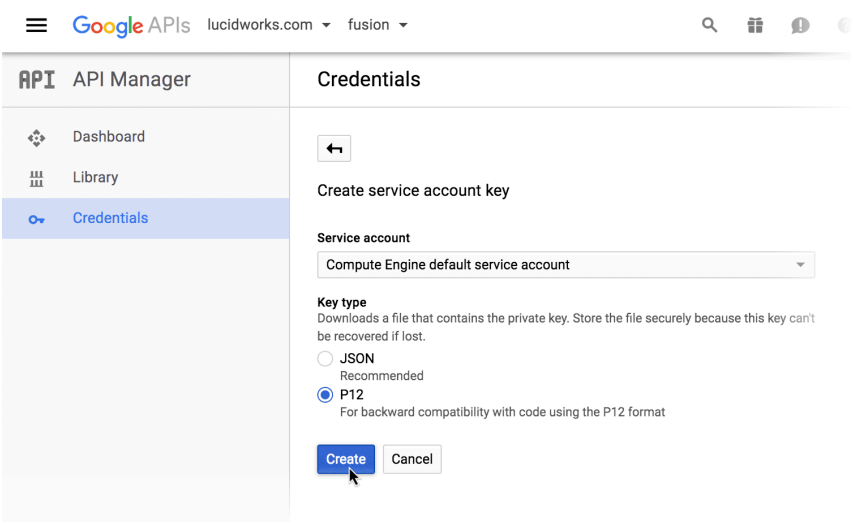
2. Enter a new project name, such as "fusion".
3. Click **Create**.
4. In the new project, click **Enable API**.
5. Under "Google Apps APIs", click **Drive API**.
6. Click **Enable**.
   Google may prompt you to create credentials. Do not create credentials here; we will do that a few steps later.
7. Click **Library**, then **Admin SDK**.
8. Click **Enable**.

4. Create a service account key:

1. Navigate to **Credentials** > **Create Credentials** > **Service account key**:



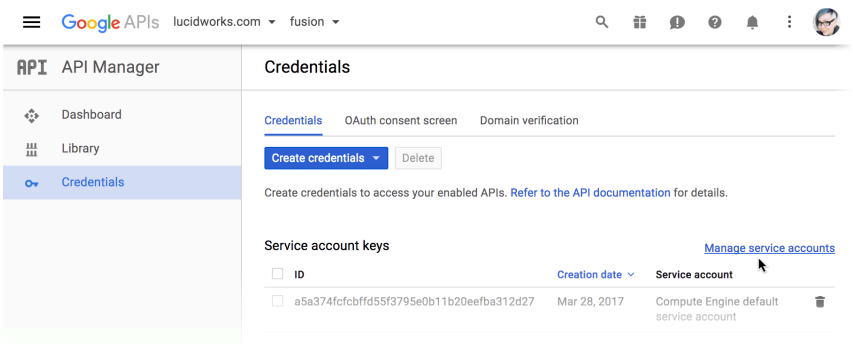
2. From the **Service account** list, select **Compute Engine default service account**.
3. Under "Key type", select **P12**.
4. Click **Create**.



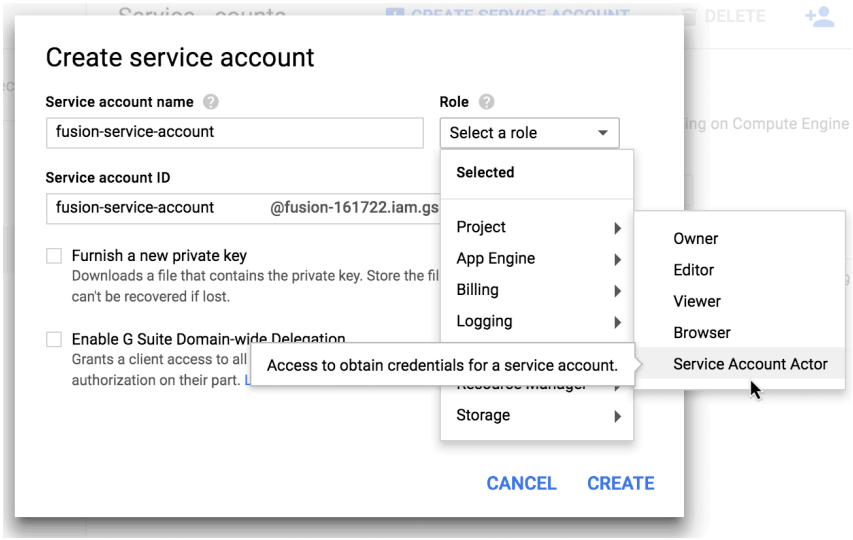
A new private key downloads automatically to your local drive. Google prompts you to store it securely and save the displayed password. The key and password will not be provided to you again.

5. Create a service account:

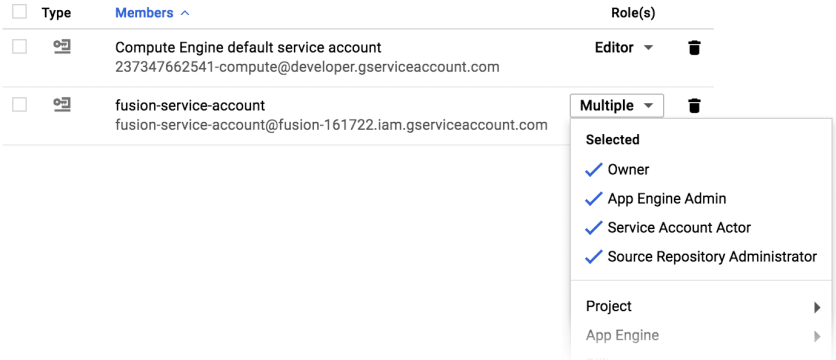
1. Click **Manage service accounts**:



2. Click **Create Service Account**.
3. Enter a service account name, such as "fusion-service-account".
4. From the **Role** list, select **Project** > **Service account actor**:



5. Select **Enable G Suite Domain-wide Delegation**.
6. Enter an arbitrary name under **Product name for the consent screen**.
7. Click **Create**.
   Google displays the list of service accounts.
8. Next to the "fusion-service-account", click **View Client ID**.
   You may need to scroll to the right in order to see this link.
9. Copy the client ID and service account name. Save them in a separate location.
10. Click the menu in the upper left and select **IAM & Admin**.
11. Next to the "fusion-service-account" project, select its permissions as shown below:

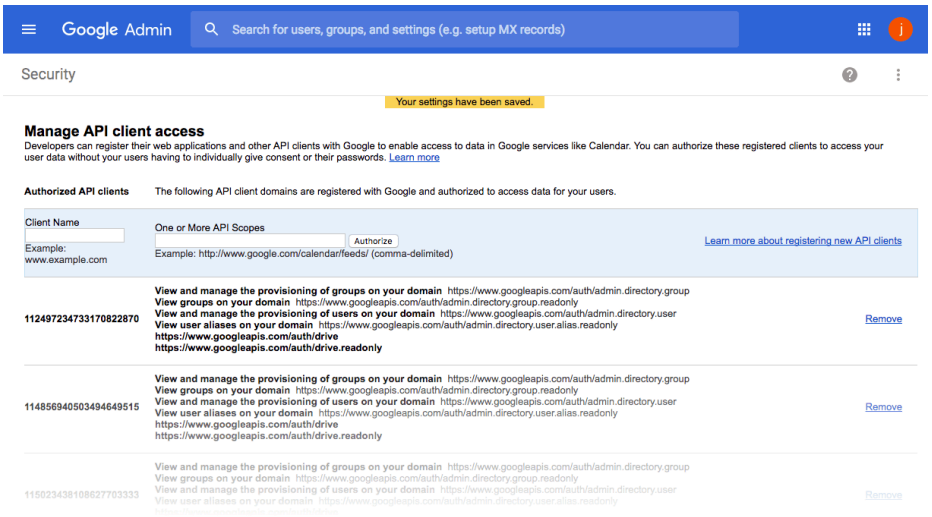


6. Go to https://admin.google.com and log in with your G-Suite account.
7. Navigate to **Admin Console** > **Security**.
   Remember, you must be logged in as a user with admin-level access rights.
8. Navigate to **Show more** > **Advanced settings** > **Manage API client access**.
9. Create a new API client:
   a. In the **Client Name** field, enter the client ID from your service account (above).
   b. In the **One or More API Scopes** field, enter the following:

```
https://www.googleapis.com/auth/admin.directory.group,http
s://www.googleapis.com/auth/admin.directory.group.readonly
,https://www.googleapis.com/auth/admin.directory.user,http
s://www.googleapis.com/auth/admin.directory.user.alias.rea
donly,https://www.googleapis.com/auth/drive,https://www.go
ogleapis.com/auth/drive.readonly
```

   c. Click **Authorize**.

The new API client authorization appears in the list:



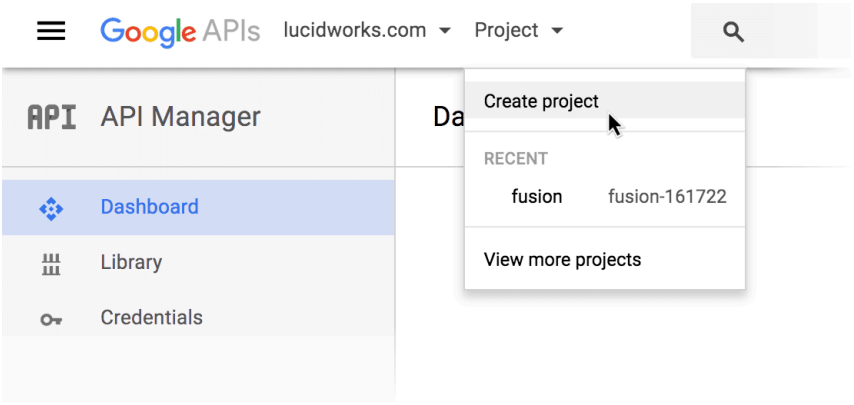
Authentication for access to per-user documents

These instructions show you how to configure Google to allow Fusion to crawl a specific user's Google Drive, including documents that other users have shared with them. Fusion must be running in an authorized domain.

How to configure authentication for access to per-user documents

1. Log in to Google as a user with *admin-level access rights*.
2. Go to https://console.developers.google.com/ .
3. Create a Google project for Fusion:

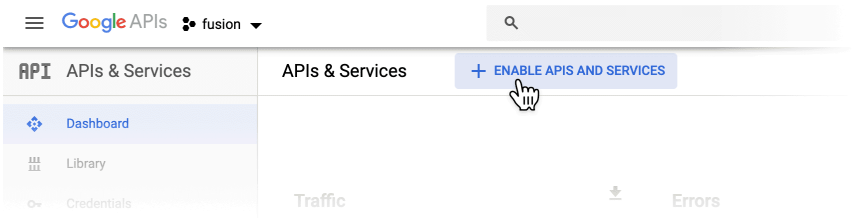
1. In the upper left, open the **Project** menu and select **Create Project**:



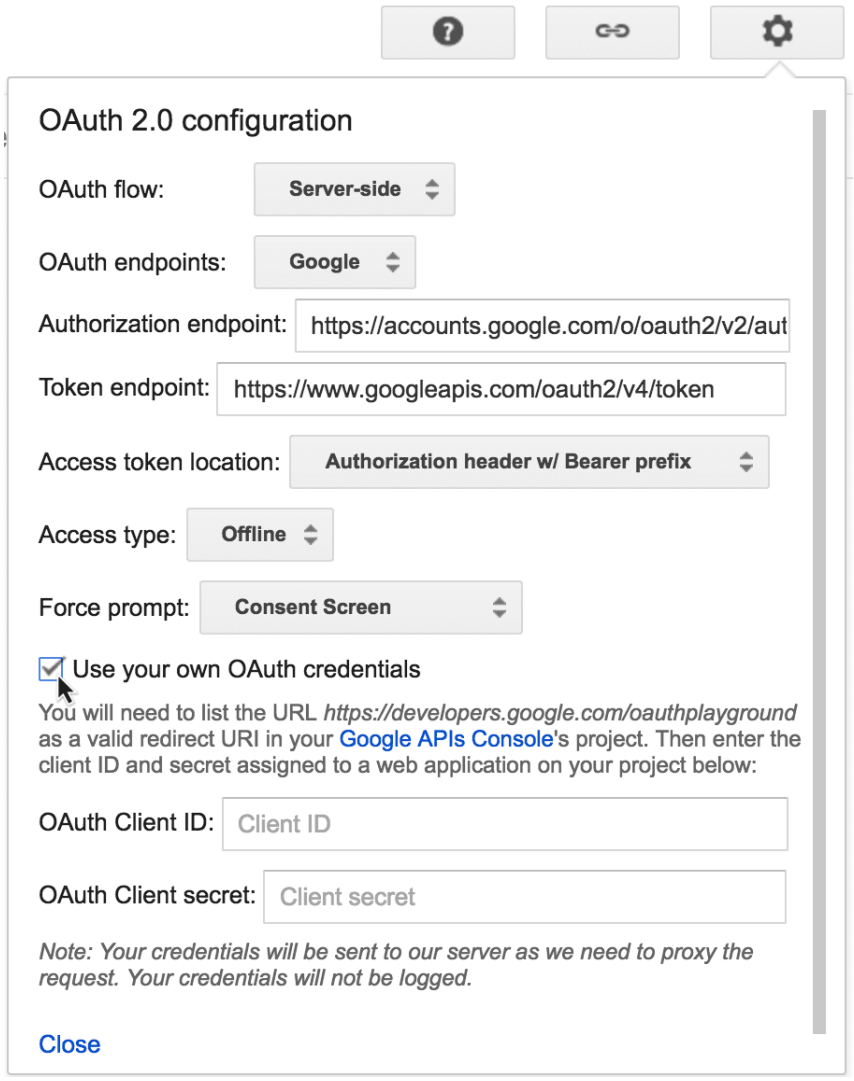
2. Enter a new project name, such as "fusion".
3. Click **Create**.

4. Create the client ID and client secret:

1. In the new project, click **Enable APIs and Services**.



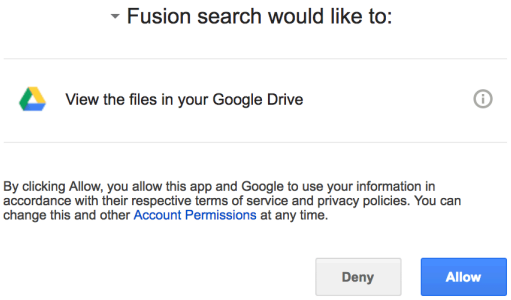
2. Under "Google Apps APIs", click **Drive API**.
3. Click **Enable**.
   Google may prompt you to create credentials, if this is the first time you have enabled this API.
4. Click **Credentials**, then **Create Credentials** > **Oauth client ID**.
5. Select **Web application**.
6. Enter a name for this Web application, such as "Fusion search".
7. In the **Authorized Javascript origins** field, enter "https://developers.google.com".
8. In the **Authorized redirect URIs** field, enter "https://developers.google.com/oauthplayground" and press Return on your keyboard.
9. In the **Authorized redirect URIs** field, enter "http://FUSION_HOST:8764/admin/oauth-redirect", specifying the hostname of your Fusion instance.
10. Click **Create**.
    Google displays the new client ID and client secret.
11. Copy the client ID and client secret. Save them in a separate location.
12. Click **OK**.
13. Go to https://developers.google.com/oauthplayground/ .
14. In the upper right, click the gear icon.
    The OAuth 2.0 configuration window opens.
15. Select **Use your own OAuth credentials**.



16. Enter your client ID and client secret.
17. Click **Close**.

5. Navigate to **Credentials** > **Oauth Consent Screen**.
6. Under **Authorized domains**, enter the Google-verified domain under which Fusion is running.
7. Click **Save**.
8. Add the credentials to the datasource configuration in the Fusion UI:

1. In the Google Drive datasource configuration panel, enter a string for the **Datasource ID**.
2. Enter the Google client ID and client secret.
3. Click **Get Refresh Token**.
   A new browser window opens, and Google prompts you for permission to access the documents:



4. Click **Allow**.
   Fusion automatically populates the **Google Drive Oauth Refresh Token** field.
5. In the **Startlinks** field, enter a starting URL to which this user has access.
   See below for details about the format for this value.
6. Click **Save**.

Configure Google Drive Authentication

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

The Google Drive connector is used to index the documents in a Google Drive account.

There are two methods of Google Drive authentication for Managed Fusion:

- *authentication for access to site-wide documents*
  Use a Google Workspace account with admin-level access rights to configure access to all shared documents owned by the users in your organization.
- *authentication for access to per-user documents*
  This type of authentication gives you access to your own documents in Google Drive. Admin-level access rights are not required, but Managed Fusion must be running in an authorized domain.

In both cases, you will get a client ID, client secret, and refresh token from Google. These become part of your datasource configuration in Managed Fusion.

Authentication for access to site-wide documents

Important

When connecting to Google Drive through a service account, the **Viewed By Me timestamp is updated** for any fetched files.

In order to access all the shared documents by users in your organization, you must configure the Google Drive API and the Admin SDK. See the instructions in the knowledge base.

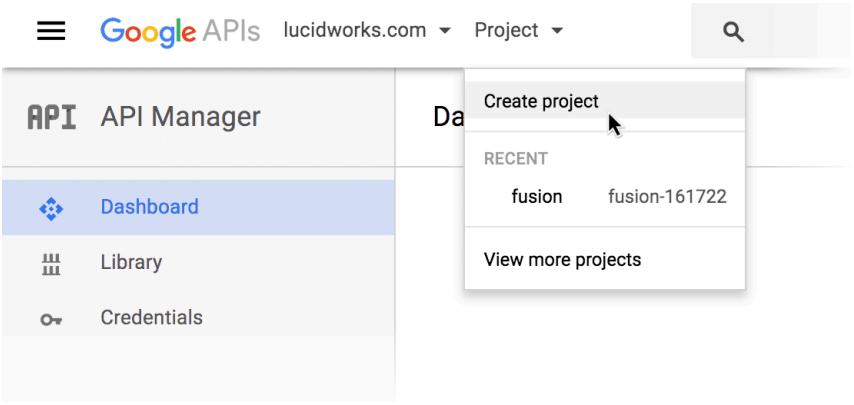
Note

You need a Google Workspace account in order to perform these steps.

How to configure authentication for access to site-side documents

1. Log in to Google as a user with *admin-level access rights*.
2. Go to https://console.developers.google.com/.
3. Create a Google project for Managed Fusion:

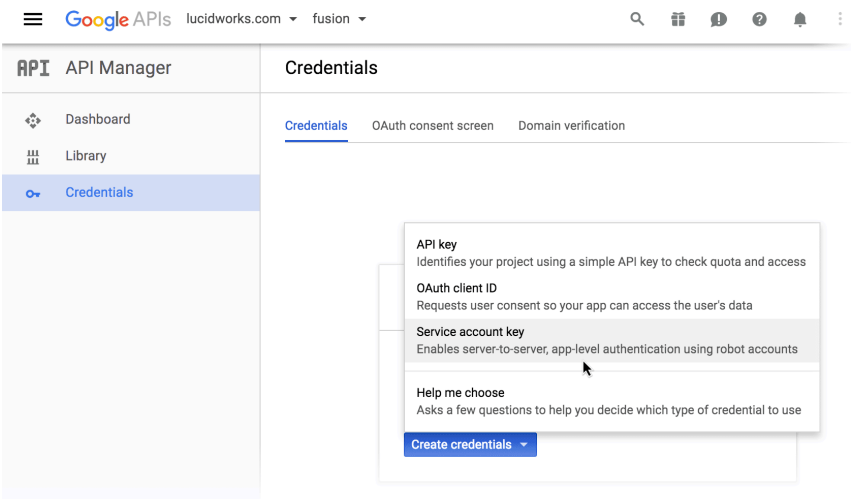
1. In the upper left, open the **Project** menu and select **Create Project**:



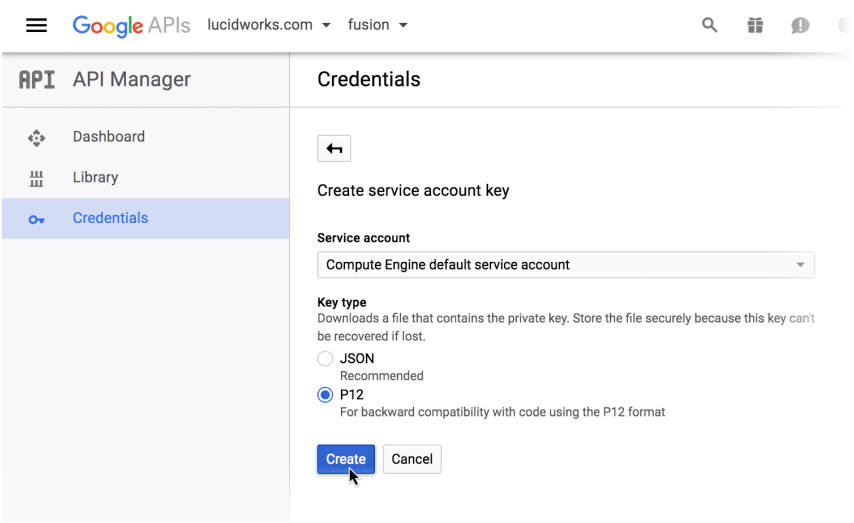
2. Enter a new project name, such as "fusion".
3. Click **Create**.
4. In the new project, click **Enable API**.
5. Under "Google Apps APIs", click **Drive API**.
6. Click **Enable**.
   Google may prompt you to create credentials. Do not create credentials here; we will do that a few steps later.
7. Click **Library**, then **Admin SDK**.
8. Click **Enable**.

4. Create a service account key:

1. Navigate to **Credentials** > **Create Credentials** > **Service account key**:



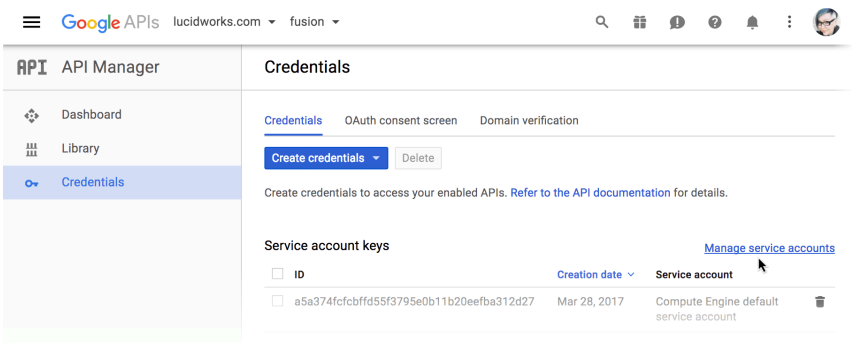
2. From the **Service account** list, select **Compute Engine default service account**.
3. Under "Key type", select **P12**.
4. Click **Create**.



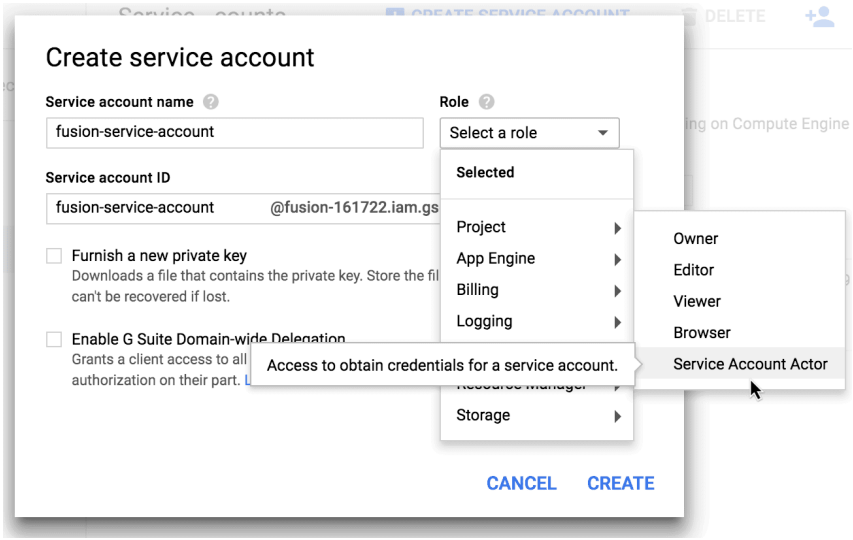
A new private key downloads automatically to your local drive. Google prompts you to store it securely and save the displayed password. The key and password will not be provided to you again.

5. Create a service account:

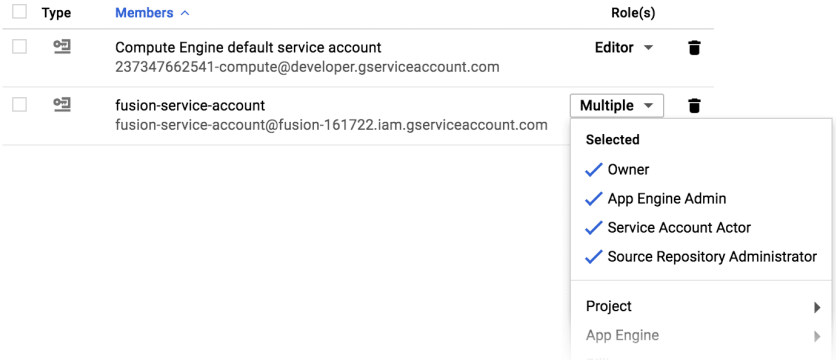
1. Click **Manage service accounts**:



2. Click **Create Service Account**.
3. Enter a service account name, such as "fusion-service-account".
4. From the **Role** list, select **Project** > **Service account actor**:



5. Select **Enable Domain-wide Delegation**.
6. Enter an arbitrary name under **Product name for the consent screen**.
7. Click **Create**.
   Google displays the list of service accounts.
8. Next to the "fusion-service-account", click **View Client ID**.
   You may need to scroll to the right in order to see this link.
9. Copy the client ID and service account name. Save them in a separate location.
10. Click the menu in the upper left and select **IAM & Admin**.
11. Next to the "fusion-service-account" project, select its permissions as shown below:

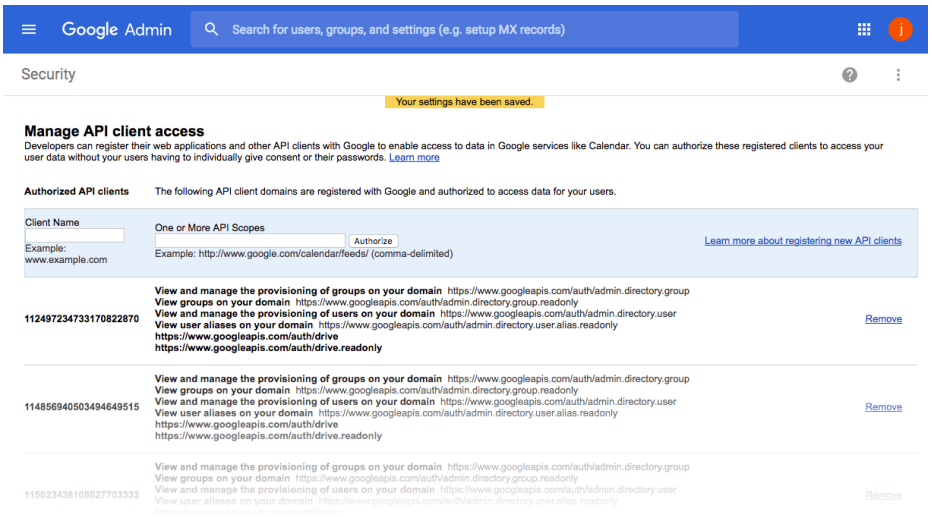


6. Go to https://admin.google.com and log in with your Google Workspace account.
7. Navigate to **Admin Console** > **Security**.
   Remember, you must be logged in as a user with admin-level access rights.
8. Navigate to **Show more** > **Advanced settings** > **Manage API client access**.
9. Create a new API client:
   a. In the **Client Name** field, enter the client ID from your service account (above).
   b. In the **One or More API Scopes** field, enter the following:

```
https://www.googleapis.com/auth/admin.directory.group,http
s://www.googleapis.com/auth/admin.directory.group.readonly
,https://www.googleapis.com/auth/admin.directory.user,http
s://www.googleapis.com/auth/admin.directory.user.alias.rea
donly,https://www.googleapis.com/auth/drive,https://www.go
ogleapis.com/auth/drive.readonly
```

   c. Click **Authorize**.

The new API client authorization appears in the list:



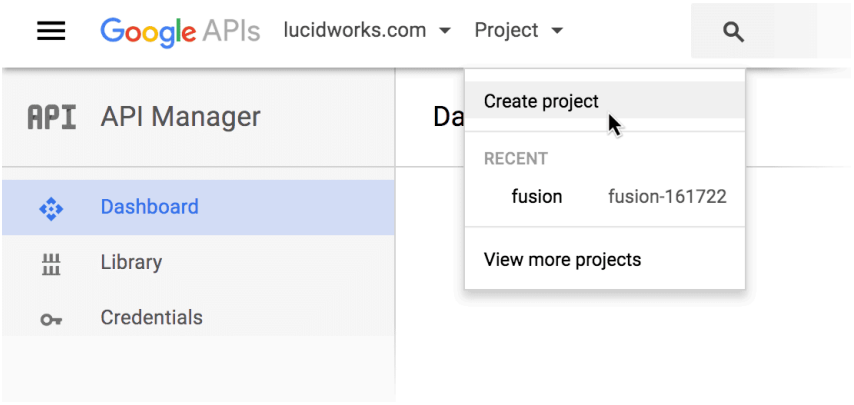
Authentication for access to per-user documents

These instructions show you how to configure Google to allow Managed Fusion to crawl a specific user's Google Drive, including documents that other users have shared with them. Managed Fusion must be running in an authorized domain.

How to configure authentication for access to per-user documents

1. Log in to Google as a user with *admin-level access rights*.
2. Go to https://console.developers.google.com/.
3. Create a Google project for Managed Fusion:

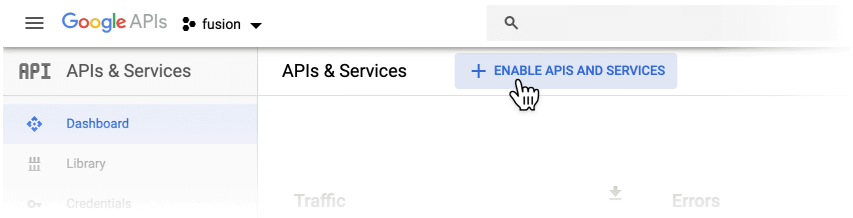
1. In the upper left, open the **Project** menu and select **Create Project**:



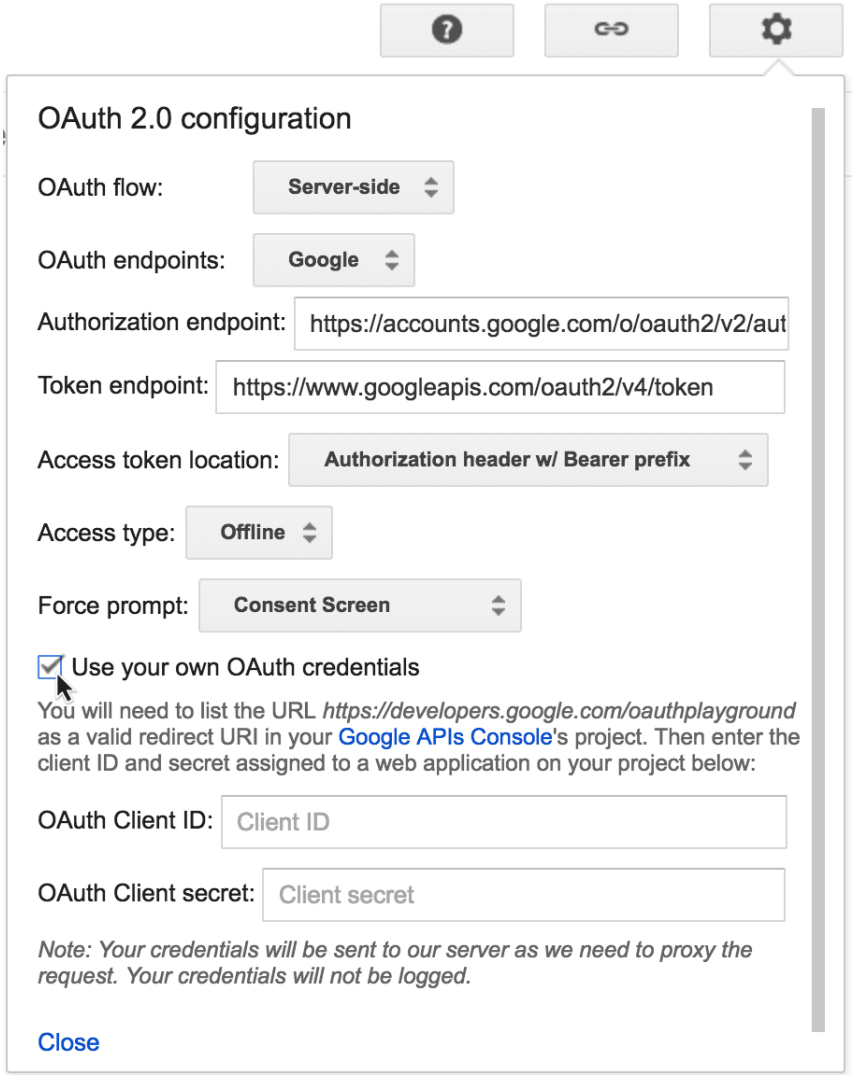
2. Enter a new project name, such as "fusion".
3. Click **Create**.

4. Create the client ID and client secret:

1. In the new project, click **Enable APIs and Services**.



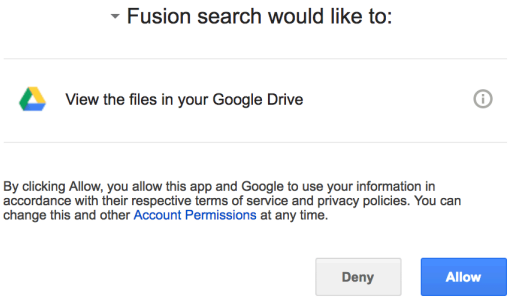
2. Under "Google Apps APIs", click **Drive API**.
3. Click **Enable**.
   Google may prompt you to create credentials, if this is the first time you have enabled this API.
4. Click **Credentials**, then **Create Credentials** > **Oauth client ID**.
5. Select **Web application**.
6. Enter a name for this Web application, such as "Managed Fusion search".
7. In the **Authorized Javascript origins** field, enter "https://developers.google.com".
8. In the **Authorized redirect URIs** field, enter "https://developers.google.com/oauthplayground" and press Return on your keyboard.
9. In the **Authorized redirect URIs** field, enter "https://EXAMPLE_COMPANY.b.lucidworks.cloud/admin/oauth-redirect", specifying the hostname of your Managed Fusion instance.
10. Click **Create**.
    Google displays the new client ID and client secret.
11. Copy the client ID and client secret. Save them in a separate location.
12. Click **OK**.
13. Go to https://developers.google.com/oauthplayground/ .
14. In the upper right, click the gear icon.
    The OAuth 2.0 configuration window opens.
15. Select **Use your own OAuth credentials**.



16. Enter your client ID and client secret.
17. Click **Close**.

5. Navigate to **Credentials** > **Oauth Consent Screen**.
6. Under **Authorized domains**, enter the Google-verified domain under which Managed Fusion is running.
7. Click **Save**.
8. Add the credentials to the datasource configuration in the Managed Fusion UI:

1. In the Google Drive datasource configuration panel, enter a string for the **Datasource ID**.
2. Enter the Google client ID and client secret.
3. Click **Get Refresh Token**.
   A new browser window opens, and Google prompts you for permission to access the documents:



4. Click **Allow**.
   Managed Fusion automatically populates the **Google Drive Oauth Refresh Token** field.
5. In the **Startlinks** field, enter a starting URL to which this user has access.
   See below for details about the format for this value.
6. Click **Save**.

Configure Grafana, Prometheus, Promtail, and Loki in Fusion

How-to

> Compatible versions: 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion*

Warning

Before you perform these installation instructions, you must delete any existing persistent volume claims (PVCs) related to Prometheus, Grafana, Promtail, and Loki on your namespace.

Clone the `fusion-cloud-native` *repository*

Open a terminal window and run following command:

```
git clone https://github.com/lucidworks/fusion-cloud-native.git
```

Install Grafana

1. In your local `fusion-cloud-native` repository, run the following command for your `<cluster>` and `<namespace>`:

   ```
   ./install-prom.sh -c <cluster> -n <namespace>
   ```

   The following is a sample output. The errors are related to resource limits on the sample cluster, and can be ignored. Similar errors may display for your cluster, and do not impact Grafana logging.

```
Adding the stable chart repo to helm repo list "prometheus-
community" already exists with the same configuration, skipping
"grafana" already exists with the same configuration, skipping
Installing Prometheus and Grafana for monitoring Fusion metrics
... this can take a few minutes. Hang tight while we grab the
latest from your chart repositories... ...Successfully got an
update from the "ckotzbauer" chart repository ...Successfully
got an update from the "lucidworks" chart repository
...Successfully got an update from the "grafana" chart
repository ...Successfully got an update from the "prometheus-
community" chart repository Update Complete. ⎈Happy Helming!⎈
Saving 2 charts Downloading prometheus from repo
https://prometheus-community.github.io/helm-charts Downloading
grafana from repo https://grafana.github.io/helm-charts Deleting
outdated charts Release "fe-foundry-monitoring" does not exist.
Installing it now. Error: context deadline exceeded Successfully
installed Prometheus and Grafana into the fe-foundry namespace.
NAME NAMESPACE REVISION UPDATED STATUS CHART APP VERSION fe-
foundry fe-foundry 11 2023-08-07 15:10:55.373825 -0700 PDT
deployed fusion-5.8.0 5.8.0 fe-foundry-jupyter fe-foundry 2
2023-07-20 11:29:38.481329 -0700 PDT deployed fusion-jupyter-
0.2.5 1.0 fe-foundry-monitoring fe-foundry 1 2023-08-10
11:41:06.113257 -0700 PDT failed fusion-monitoring-1.0.1 1.0.1
```

2. Using the Grafana service endpoint in the newly-installed Grafana helm release, run the following command:

```
kubectl get services
```

The following is a sample output.

```
NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE grafana
LoadBalancer <IP Address> <IP Address> 3000:32589/TCP 87m
```

If the output does not display, run the following command to expose Grafana, including an **EXTERNAL_IP** for your Grafana LoadBalancer service:

```
kubectl expose deployment <grafana-deployment-name> --
type=LoadBalancer --name=grafana --port=3000 --target-port=3000
```

Install Loki

To obtain Loki from the helm chart repository, run the following command for the unique **<loki-release-name>** for your cluster:

```
helm upgrade --install <loki-release-name> --namespace=<namespace>
grafana/loki-stack
```

If you do no enter the **<loki-release-name>** correctly, an error similar to the following displays:

```
Error: rendered manifests contain a resource that already exists.
Unable to continue with install: PodSecurityPolicy "loki" in namespace
"" exists and cannot be imported into the current release: invalid
ownership metadata; annotation validation error: key
"meta.helm.sh/release-namespace" must equal "fe-foundry": current
value is "ps-intl".
```

If the helm upgrade is successful, the following is a sample output.

```
Release "fe-foundry-loki" does not exist. Installing it now. W0810
11:47:07.890370 39624 warnings.go:70] policy/v1beta1 PodSecurityPolicy
is deprecated in v1.21+, unavailable in v1.25+ W0810 11:47:09.396246
39624 warnings.go:70] policy/v1beta1 PodSecurityPolicy is deprecated
in v1.21+, unavailable in v1.25+ NAME: fe-foundry-loki LAST DEPLOYED:
Thu Aug 10 11:47:07 2023 NAMESPACE: fe-foundry STATUS: deployed
REVISION: 1 NOTES: The Loki stack has been deployed to your cluster.
Loki can now be added as a datasource in Grafana. See
http://docs.grafana.org/features/datasources/loki/ for more detail.
```

Obtain Admin credentials for Grafana

1. After you validate Grafana is running by accessing `<EXTERNAL-IP>:3000`, run the following command to obtain an `<admin_password>` for your Grafana instance:

```
kubectl get secret --namespace <namespace> <release_name>-
monitoring-grafana -o jsonpath="{.data.admin-password}" | base64
--decode ; echo
```

2. Sign in to Grafana and change the password for security purposes.
3. Run the following command to display the promtail pods that are running:

```
kubectl get pods | grep -i promtail | nl
```

Promtail pods must match the number of Kubernetes nodes since an instance of promptail runs on each node.

Add the Loki datsource

1. Sign in to Grafana and in the toolbar, click the arrow below Home to display all of the options.
2. In the Configuration section, click **Data sources**.
3. Click **Add new data source**.
4. In the search bar for the data source, enter **Loki**.
5. In the URL field on the Settings screen, enter your unique `<loki-release-name:port>`. The default port for Loki is `3100`.

> **Note**
>
> If you encounter issues with the `<loki-release-name:port>` information, open a terminal and run `kubectl get services | grep loki` to display a list of every service with a name that contains `loki` along with its associated IP address and port.

6. Complete the other fields and click **Save & test**.

Additional information

For more information, see *View logs in Self-Hosted Fusion 5.6 and later*.

Configure Highlighting in the Query Workbench

How-to

> Compatible versions: 4.2, 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion Server*, *Fusion*

Search results can be displayed with the search terms highlighted, by adding these Solr query parameters to the Additional Query Parameters stage of the query pipeline:

- `hl=true`
- `hl.fl=*`

By default, the Query Workbench ignores these parameters when rendering search results. To view highlighted search results in the preview panel of the Query Workbench, you must configure the parameters above *and* enable the **Display highlighting** option.

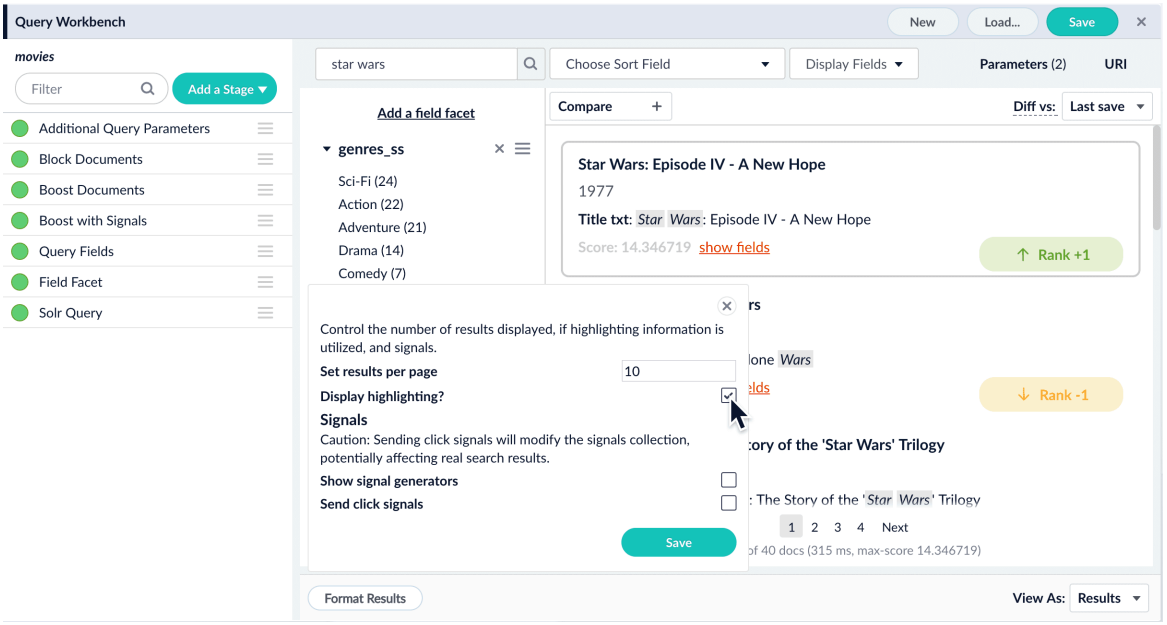
How does this differ from other highlight methods?

- **Highlight Query Pipeline Stage**
  Introduced in Fusion 5.2.0, the *Query Highlight pipeline stage* allows you to configure standard options for highlighting using Solr's Unified highlighter instead of adding Solr request parameters. For instructions, see *Configure the Highlight Query Pipeline Stage*.
- **Additional Query Parameters stage**
  - For Fusion 5.1.x, configure highlighting with the *Additional Query Parameters stage*. This method can still be used in Fusion 5.2.0 and later.
  - For Fusion 4.x.x, configure highlighting with the *Additional Query Parameters stage*.

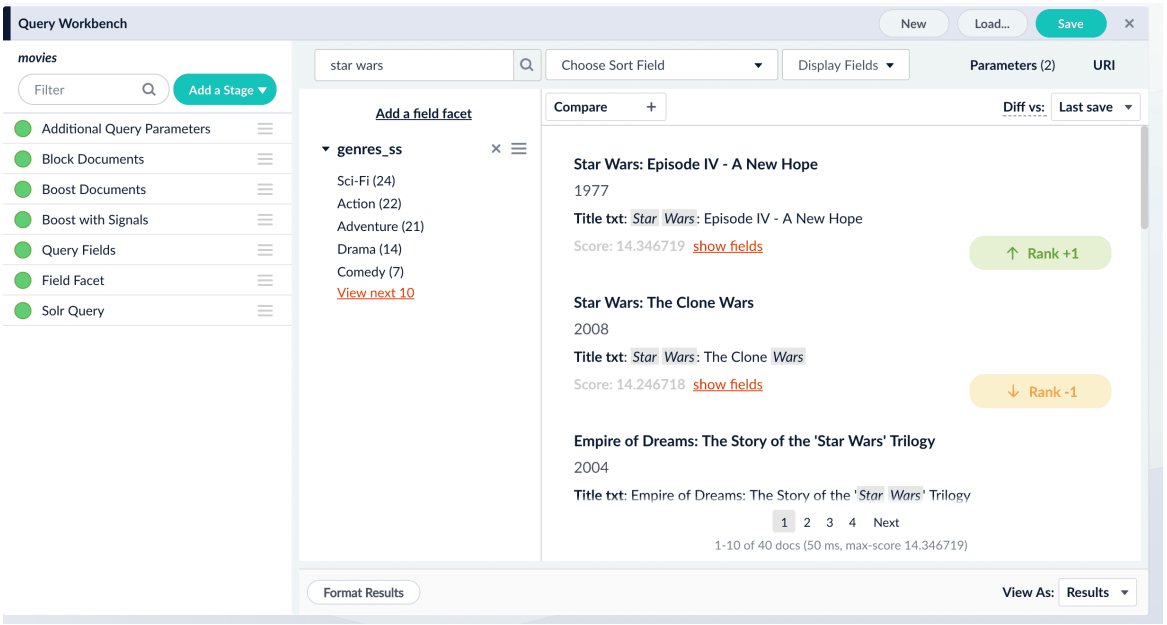
How to enable highlighting in the preview panel

1. Add the Additional Query Parameters stage to your query pipeline, or select it if it is already there.
2. Under **Parameters and Values**, add the following query parameters and values:

3. Click **Apply**.
   At this point, no highlighting appears in the preview panel.
4. At the bottom of the window, click **Format Results**.
5. Select **Display highlighting**.



6. Click **Save**.
   Now our search results include highlighted search terms:


Query Workbench
New
Load...
Save
movies
Filter
Add a Stage
Additional Query Parameters
Block Documents
Boost Documents
Boost with Signals
Query Fields
Field Facet
Solr Query
star wars
Choose Sort Field
Display Fields
Parameters (2)
URI
Add a field facet
genres_ss
Sci-Fi (24)
Action (22)
Adventure (21)
Drama (14)
Comedy (7)
View next 10
Compare
Diff vs:
Last save
Star Wars: Episode IV - A New Hope
1977
Title txt: Star Wars: Episode IV - A New Hope
Score: 14.346719 show fields
Rank +1
Star Wars: The Clone Wars
2008
Title txt: Star Wars: The Clone Wars
Score: 14.246718 show fields
Rank -1
Empire of Dreams: The Story of the 'Star Wars' Trilogy
2004
Title txt: Empire of Dreams: The Story of the 'Star Wars' Trilogy
1 2 3 4 Next
1-10 of 40 docs (50 ms, max-score 14.346719)
Format Results
View As:
Results

Configure Highlighting in the Query Workbench

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

Search results can be displayed with the search terms highlighted, by adding these Solr query parameters to the Additional Query Parameters stage of the query pipeline:

- `hl=true`
- `hl.fl=*`

By default, the Query Workbench ignores these parameters when rendering search results. To view highlighted search results in the preview panel of the Query Workbench, you must configure the parameters above *and* enable the **Display highlighting** option.

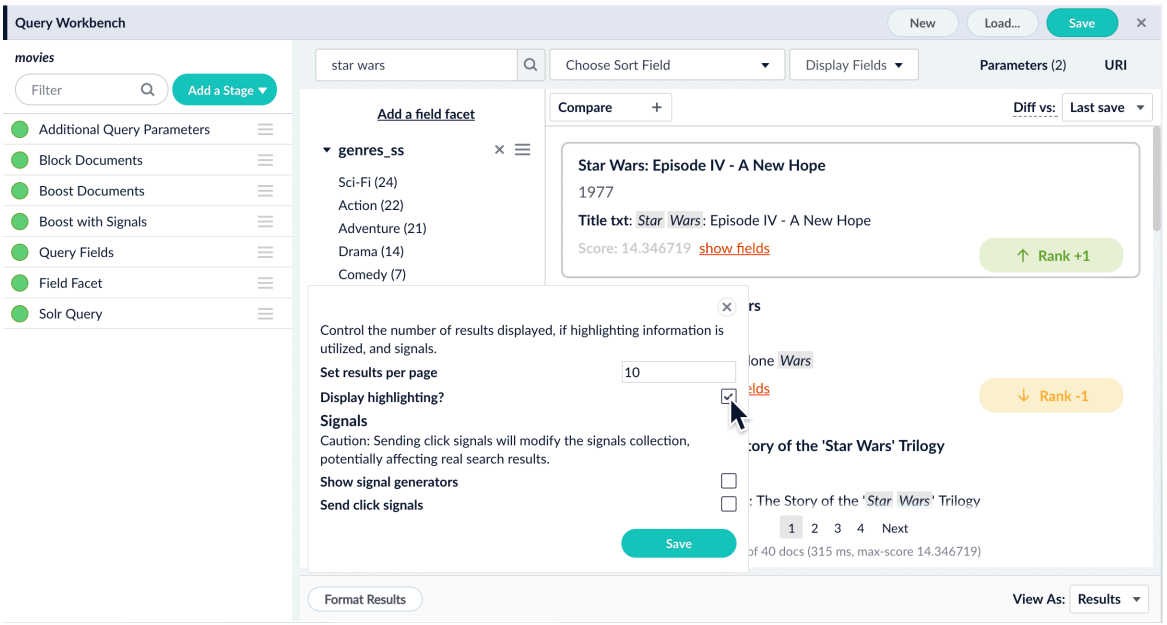
How does this differ from other highlight methods?

- **Highlight Query Pipeline Stage**
  The *Query Highlight pipeline stage* allows you to configure standard options for highlighting using Solr's Unified highlighter instead of adding Solr request parameters. For instructions, see *Configure the Highlight Query Pipeline Stage*.
- **Additional Query Parameters stage**
  - Configure highlighting with the *Additional Query Parameters stage*.

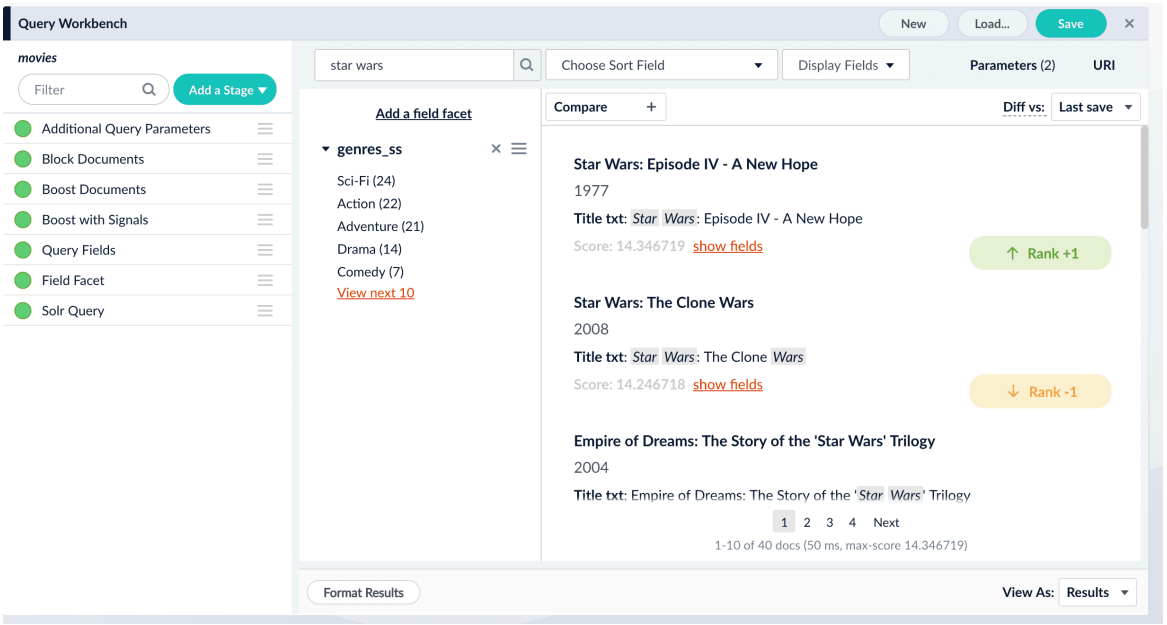
How to enable highlighting in the preview panel

1. Add the Additional Query Parameters stage to your query pipeline, or select it if it is already there.
2. Under **Parameters and Values**, add the following query parameters and values:

3. Click **Apply**.
   At this point, no highlighting appears in the preview panel.
4. At the bottom of the window, click **Format Results**.
5. Select **Display highlighting**.



6. Click **Save**.
   Now our search results include highlighted search terms:


Query Workbench
New
Load...
Save
movies
Filter
Add a Stage
Additional Query Parameters
Block Documents
Boost Documents
Boost with Signals
Query Fields
Field Facet
Solr Query
star wars
Choose Sort Field
Display Fields
Parameters (2)
URI
Add a field facet
genres_ss
Sci-Fi (24)
Action (22)
Adventure (21)
Drama (14)
Comedy (7)
View next 10
Compare
Diff vs:
Last save
Star Wars: Episode IV - A New Hope
1977
Title txt: Star Wars: Episode IV - A New Hope
Score: 14.346719 show fields
Rank +1
Star Wars: The Clone Wars
2008
Title txt: Star Wars: The Clone Wars
Score: 14.246718 show fields
Rank -1
Empire of Dreams: The Story of the 'Star Wars' Trilogy
2004
Title txt: Empire of Dreams: The Story of the 'Star Wars' Trilogy
1 2 3 4 Next
1-10 of 40 docs (50 ms, max-score 14.346719)
Format Results
View As:
Results

Configure Kerberos Authentication for a Jive Datasource

How-to

> Compatible versions: 5.9
>
> Works with: *Fusion Connectors*, *Fusion Server*, *Fusion*

Retrieve content from a Jive instance.

The Fusion Jive connector can crawl Jive repositories protected by Kerberos using SPNEGO. This is a way to access Jive without requiring a user's login credentials.

The Fusion Jive connector can optionally use Kerberos with SAML/Smart Form authentication.

To crawl a Kerberos-protected Jive repository, first create the necessary configuration files, then configure Fusion to use them.

Note

Kerberos support requires Fusion 5.9.5.

1. Create standard Java configuration files to connect to Kerberos

Fusion uses the JDK standard JAAS Kerberos implementation, which is based on three system properties that reference three separate files.

The files are as follows:

- On the Kerberos-protected server, a keytab file, named `kerberuser.keytab` in our examples.
- On the Fusion server, a configuration file named `login.conf`.
- On the Fusion server, an initialization file named `krb5.ini` (Windows) or `krb5.conf` (Linux).

1.1. Create a Kerberos keytab

Create and validate the keytab file for the Kerberos client principal you want to use to authenticate to the Jive repository.

If you do not specify the `kerberosPrincipalName` and `kerberosKeytabFilePath` or `kerberosKeytabBase64` when *creating the Fusion datasource*, Fusion uses the default login principal and ticket cache. You can see the default values by logging into the Fusion server as the user who runs Fusion and running `klist`.

If you do not want to use the default account and credentials, specify these configuration properties when creating a keytab as well as in the *Jive datasource setup*. Use the Kerberos user principal name (UPN), not the service principal name (SPN, which is used with the Kerberos *security realm*). In some cases the UPN can be a service.

In our examples, the Fusion Jive connector authenticates to the Jive repository using the user kerbuser@win.lab.lucidworks.com. We create a keytab file `kerbuser.keytab` for the user principal kerbuser@WIN.LAB.LUCIDWORKS.COM.

1.1.1. CREATE A KERBEROS KEYTAB ON WINDOWS

Example:

```
ktpass -out kerbuser.keytab -princ kerbuser@WIN.LAB.LUCIDWORKS.COM -
mapUser kerbuser -mapOp set -pass YOUR_PASSWORD -crypto AES256-SHA1 -
pType KRB5_NT_PRINCIPAL
```

+

Note

The following weak encryption types are not supported by Fusion:

- DES-CBC-CRC
- DES_CBC_MD5
- Microsoft Windows 2000 RC4-HMAC

1.1.2. CREATE A KERBEROS KEYTAB ON UBUNTU LINUX

Prerequisite: Install the `krb5-user` package: `sudo apt-get install krb5-user`

Example:

```
ktutil addent -password -p HTTP/kerbuser@WIN.LAB.LUCIDWORKS.COM -k 1 -
e aes128-cts-hmac-sha1-96 - it will ask you for password of kerbuser -
wkt kerbuser.keytab q
```

1.1.3. TEST THE KEYTAB

Once you create a keytab, verify that it works.

Prerequisite: You need a version of curl installed that allows SPNEGO. To test whether your version of curl does this, run `curl --version` and make sure SPNEGO is in the output.

Run the following curl command (replace the keytab path and site):

```
export KRB5CCNAME=FILE:/path/to/kerbuser.keytab curl -vvv --negotiate
-u : http://your-site.com
```

Note that the first request is a 401 status code for the negotiate request followed by a second request, which is a status of 200.

1.2. Create a login.conf and krb5.ini

On the Fusion server, create `login.conf` and `krb5.ini` files as follows.

1.2.1. CREATE A LOGIN.CONF ON WINDOWS

In this example, the keytab is stored at `C:\kerb\kerbuser.keytab`

```
KrbLogin { com.sun.security.auth.module.Krb5LoginModule required
useKeyTab=true storeKey=true keyTab="C:\kerb\kerbuser.keytab"
useTicketCache=true principal="kerbuser@WIN.LAB.LUCIDWORKS.COM"
debug=true; };
```

Create a login.conf on Linux

In this example, the keytab is stored at `/home/lucidworks/kb.keytab`

```
com.sun.security.jgss.initiate {
com.sun.security.auth.module.Krb5LoginModule required useKeyTab=true
storeKey=true keyTab="/home/lucidworks/kerbuser.keytab"
useTicketCache=true principal="kerbuser@WIN.LAB.LUCIDWORKS.COM"
debug=true; };
```

The format of the login.conf is described on the Oracle Web site .

1.2.2. CREATE A KRB5.INI OR KRB5.CONF

When you install krb5 on Linux, you can find a Kerberos configuration file in `/etc/krb5.conf` . You can optionally create a custom one instead.

Creating a `krb5.conf` is the same for Linux and Windows. On Windows the file is `krb5.ini` .

In this example the domain is WIN.LAB.LUCIDWORKS.COM, the Kerberos kdc host is `my.kdc-dns.com` , and the Kerberos admin server is `my-admin-server-dns.com` .

Example:

```
[libdefaults] default_realm = WIN.LAB.LUCIDWORKS.COM
default_tkt_enctypes = aes128-cts-hmac-sha1-96 default_tgs_enctypes =
aes128-cts-hmac-sha1-96 permitted_enctypes = aes128-cts-hmac-sha1-96
dns_lookup_realm = false dns_lookup_kdc = false ticket_lifetime = 24h
renew_lifetime = 7d forwardable = true udp_preference_limit = 1
[realms] WIN.LAB.LUCIDWORKS.COM = { kdc = my.kdc-dns.com admin_server
= my.admin-server-dns.com } [domain_realm] .WIN.LAB.LUCIDWORKS.COM =
WIN.LAB.LUCIDWORKS.COM WIN.LAB.LUCIDWORKS.COM =
WIN.LAB.LUCIDWORKS.COM
```

The format of the `krb.conf` or `krb5.ini` file is described in the MIT Kerberos documentation. You can change the encryption algorithms by changing the properties `default_tkt_enctypes`, `default_tgs_enctypes`, and `permitted_enctypes` as needed. For example:

```
default_tkt_enctypes = RC4-HMAC default_tgs_enctypes = RC4-HMAC
Permitted_enctypes = RC4-HMAC
```

2. Configure Fusion to use Kerberos

Once you have the keytab, `login.conf`, and `krb5.ini` files, configure Fusion to use Kerberos. You must set a property in a Fusion configuration file in addition to defining the datasource in the Fusion UI.

At the command line on any machine in your Fusion cluster:

1. In `$FUSION_HOME/conf/fusion.cors` (`fusion.properties` in Fusion 4.x), add the following property to the `connectors-classic jvmOptions` setting: `-Djavax.security.auth.useSubjectCredsOnly=false`
2. Restart the `connectors-classic` service using `./bin/connectors-classic restart` on Linux or `bin\connectors-classic.cmd restart` on Windows.

In the Fusion UI:

1. Click **Indexing > Datasources**. HEAD
2. Click **Add+**, then **Jive**.
3. Enter a datasource ID and a Jive instance URL.
4. Check **Enable SPNEGO/Kerberos Authentication**.
5. You can either use the default principal name or specify a principal name to use.
    - If you do not specify the principal name, then Fusion uses the default login principal and ticket cache. You can see those default values by logging into the Fusion server as the user who runs Fusion and running `klist`.
6. If you specify a principal name, you must provide a keytab, either in Base64 or as a file path.
    - If you specify a keytab file path, the file must be on the machine running the Fusion connector, for each connector's node in the cluster.
    - The Base64 option lets you supply the keytab in one place, in the UI.

7. Fill in any remaining options to configure the datasource.

Configure Managed Fusion for Kerberos in Unix

How-to

> Compatible versions: 5.9
>
> Works with: *Managed Fusion*

To configure the Managed Fusion UI service to use Kerberos for user authentication, you must create a Kerberos *security realm*.

Note

Kerberos support requires Managed Fusion 5.9.5.

Kerberos is a system that provides authenticated access for users and services on a network. Instead of sending passwords in plaintext over the network, encrypted passwords are used to generate time-sensitive tickets that are used for authentication. SPNEGO provides a mechanism for extending Kerberos to Web applications through the standard HTTP protocol.

Kerberos uses symmetric-key cryptography and a trusted third party called a Key Distribution Center (KDC) to authenticate users to a suite of network services. (By users we mean both end users and client programs). The computers managed by that KDC and any secondary KDCs constitute a realm. When a user authenticates to the KDC, the KDC sends a set of credentials (a ticket) specific to that session back to the user's machine. Kerberos-aware services use the ticket on the user's machine for authentication instead of requiring sign-on with a password. Because tickets are used rather than passwords, this provides the convenience of Single Sign-On (SSO) in addition to security.

A Kerberized process is one that has been configured so it can get tickets from a KDC and negotiate with Kerberos-aware services. When a user sends an HTTP request, Managed Fusion tries to authenticate using the Kerberos/SPNEGO protocol. If the request was sent from a browser, Managed Fusion does not display the initial sign-on panel; instead on login, the user sees the main Managed Fusion collections panel.

Important

This article focuses on configuring Managed Fusion for Kerberos *in Unix*. To learn how to configure Managed Fusion for Windows, see *Configure Managed Fusion for Kerberos in Windows*.

To Kerberize Managed Fusion, you must:

- Configure the Kerberos client on the server that the Managed Fusion UI service will be running on so that it can talk to the KDC (section *Configuring the Kerberos client* below).
- Configure the security realm of the Managed Fusion UI (section *Configuring Managed Fusion Authentication for Kerberos Realm* below).
- Depending on the encryption used by the KDC, you may also need to install additional Java security libraries into the Managed Fusion distribution. These are freely available from Oracle, see download and installation instructions below.

To do this, you need following information, which you can get from your sysadmin:

- **Kerberos realm name.** In most cases, this is your domain name in upper case.
- **KDC name.** This is usually a combination of `kerberos.` and your domain name. For example, `kerberos.example.com`
- **Kerberos principal name and password.** A principal is a unique identity to which Kerberos can assign tickets. When the entity is a client program, this is called the Service Principal name.
- **A keytab file.** This holds the encrypted credentials.

The usual scenario in an enterprise organization is to have a Kerberos admin create a service principal with a random key password. Then, the admin generates a keytab, which is then used for Managed Fusion service principal authentication.

The Kerberos commands needed for configuration and testing are:

- `kinit` . Obtain and cache a ticket from the KDC (domain login)
- `kdestroy` . Destroys credentials (domain logout)
- `klist` . Lists cached credentials
- `ktutil` . Create or add credentials to a keytab file

If your browser is not already configured to use the Kerberos/SPNEGO, you need to do so in order to test the Managed Fusion configuration.

Configuring the Kerberos client

Step 1: Edit the Kerberos configuration file.

To configure your local Kerberos client so that it can talk to the Kerberized server, you must edit Kerberos configuration file named `krb5.conf`. On most Unix systems, this file is located at `/etc/krb5.conf`.

This file contains Kerberos configuration information, including the locations of KDCs and admin servers for the Kerberos realms of interest, defaults for the current realm and for Kerberos applications, and mappings of hostnames onto Kerberos realms.

If your organization realm name is "MYORG.ORG", and your KDC server is named "kerberos.myorg.org", then you edit two entries. The first entry is `libdefaults`. Set MYORG.ORG as the default realm:

```
[libdefaults] default_realm = MYORG.ORG
```

The second entry is `realms`. Add MYORG.ORG as a realm:

```
[realms] MYORG.ORG = { kdc = kerberos.myrealm.com }
```

For example, for realm LUCIDWORKS.IO, the krb5.conf file is just like the above example, except that instead of "myorg.org" we specify "lucidworks.io".

Step 2: Authenticate to Kerberos

The command `kinit` is the Kerberos authentication command. To get started, you authenticate to Kerberos using the Kerberos principal name and password (which you may need to obtain from your sysadmin). For this example, the principal name is "prince."

```
> kinit prince
```

The `kinit` command prompts for a password. Successful authentication is silent. Unsuccessful authentication results in an error message.

The command `klist` shows all cached Kerberos credentials. To check that you have successfully authenticated, run this command:

```
> klist
```

Output should be in this form, but with your data:

```
Credentials cache: API:C980F9F5-415C-4A3E-9C67-883C7D5FFFBE Principal:
prince@MYORG.ORG Issued Expires Principal May 6 15:14:55 2015 May 7
01:12:47 2015 krbtgt/MYORG.ORG@MYORG.ORG
```

The Service Principal Keytab file

The usual scenario in an enterprise organization is to have a Kerberos admin create a service principal with a random key password. Then, the admin generates a keytab, which is then used for Managed Fusion service principal authentication. If you are your own Kerberos admin, then you will need to create this file for yourself.

Step 3. Create a Keytab file

The command `ktutil` creates the service principal keytab file which holds the encrypted credentials that the Managed Fusion UI Proxy will use for Kerberos authentication. In order to generate the keytab file, you must have a set of cached credentials, therefore, first run the `kinit` command (step 2).

From the command line, run the command `ktutil`. You must enter your password twice.

```
> ktutil -k http-myrealm.org.keytab add -p HTTP/myrealm.org@MYORG.ORG
-e aes256-cts-hmac-sha1-96 -V 0
```

The -k argument specifies the name of the keytab file which will be created or updated. The command "add" takes the following argument flags:

- "-p" : service principal, format <service>/<fully.qualified.domain>@REALM
- "-e" : encryption type . Depending on the encryption type, you may need to download additional Java security libraries for strong encryption.

  Note

  The following encryption types are not supported by Fusion:
  - DES-CBC-CRC
  - DES_CBC_MD5
  - Microsoft Windows 2000 RC4-HMAC

- "-V" : key version number (kvno). Key version numbers are used in the Kerberos V5 protocol to distinguish between different keys in the same domain.

If successful, this command creates a keytab file called "http-myrealm.org.keytab". Note the directory you are in - you will need this full path when creating the Proxy realm-config later.

Step 4. Test the Keytab file

The location of this keytab file will be used to configure UI Proxy configuration. Before configuring the Managed Fusion UI Proxy, you should check that the keytab file is valid. Testing the keytab requires the following sequence of steps:

- Clear any existing credentials via command `kdestroy`.

- Log in using the keytab as an argument to the command `kinit –kt <keytab file> <principal>`, where `<principal>` is the name of a principal within the keytab file.
- Examine your credentials via command `klist`.
- Clear credentials via command `kdestroy`, which removes any existing credentials, effectively logging you out of Kerberos.

To remove cached credentials, use the `kdestroy` command. This command succeeds silently. To check that credentials have been removed, re-run the klist command:

```
> kdestroy > klist klist: krb5_cc_get_principal: No credentials cache file found
```

Use the keytab to login as the service principal, without being prompted for a password:

```
> kinit –t http–myrealm.org.keytab HTTP/myrealm.org
```

Examine your credentials via the command `klist`. The output should be similar to this:

```
Credentials cache: API:51D488FF–5CD9–4E16–98FA–B47743F5B4ED Principal: HTTP/myrealm.org@MYORG.ORG Issued Expires Principal Apr 1 09:15:02 2015 Apr 1 19:13:42 2015 krbtgt/MYORG.ORG@MYORG.ORG
```

Logout again with `kdestroy`.

Configuring Managed Fusion Authentication for Kerberos Realm

Once you have tested both the user and service principal logins, you must create the service principal realm-config in the Managed Fusion Authentication Proxy. This allows the Proxy to authenticate to Kerberos as the service principal, without a password.

Step 5. Configure the Managed Fusion Realm

Managed Fusion security realms can be configured either via the Managed Fusion UI Admin tool or the Managed Fusion REST API. The advantage of using the Managed Fusion UI Admin Tool is that a single panel's worth of configuration requires a series of calls to the REST API. It is important to understand the set of configuration properties collected by the Managed Fusion UI and how they are used by the REST API.

A security realm for Kerberos has the following properties:

- name : unique string identifier
- realmType : "kerberos"
- enabled : whether or not the realm is available for users to use with system authentication
- config : this property is required for the realm type "kerberos". It takes two key-value pairs:
    - principal : the principal service name

- keytab : this must be the full path to the keytab file.

To configure Managed Fusion via the Managed Fusion UI:

1. Log in to Managed Fusion as "admin" or as a user who has super-admin privileges.
2. In the left-hand side navigation bar, click **System > Access control**.
3. Click the **Security Realms** tab, then **Add Security Realm**.
4. On the New Security Realm form, choose type "kerberos" from the pulldown menu so that there are input boxes for the Kerberos realm properties "Service Principal" and "Keytab path".
5. Choose a realm name, and make sure the "enabled" box is checked.
6. If the **auto-created** box is unchecked, a user with admin privileges needs to create Managed Fusion user accounts.
7. Select default roles. The default roles are assigned to a user the first time they access Managed Fusion via this realm, using the Kerberos/SPNEGO protocol. For example, once you have defined a Kerberos realm "my-kerberos-realm" for domain "MYORG.ORG", when user "any.user" in domain "MYORG.ORG" authenticates to Managed Fusion for the first time via this realm, they are added to the set of Managed Fusion users as username "any.user@MYORG.ORG" and they have all default roles.
   It is prudent to allow the minimum set of default roles, as all users will have these permissions. Some users require admin privileges and a few users require super-admin privileges. There should always be a user with super-admin privileges that can authenticate to Managed Fusion using the native security realm and can then grant permissions to individual users as needed.

   Note

   The Managed Fusion proxy Kerberos realm works only with one Managed Fusion host. For a multi-node Managed Fusion cluster where each node resides on a different host, configure a separate realm for each host with a dedicated service/principal.

8. Enter the service principal and the full path to the keytab file.
9. Click **Save**. Managed Fusion displays a confirmation message.

Kerberos/SPNEGO HTTP Authentication

SPNEGO provides a mechanism for extending Kerberos to applications that use the HTTP protocol including web browsers and the `curl` command-line utility.

Step 6. Configure the HTTP client

When a user sends a request to the Kerberized Managed Fusion UI, a SPNEGO request (http[s]) is made. If the user is not already authenticated, the Managed Fusion authentication proxy yields a 401 status code and a Negotiate header. This status/header response triggers compatible clients to fetch a local ticket from their Kerberos "ticket tray." This ticket is then encoded and sent back to the Managed Fusion. The Managed Fusion authentication proxy will then decode the ticket, and perform a SPN.doAs(user) authentication request to the KDC/Authentication Service. Depending on the results, the proxy then successfully executes the original request (along with a session cookie) or a 401 (without the Negotiate). Clients can either choose to use the session cookie or continue authenticating on every request.

Configuring Web Browsers and `curl` for SPNEGO

The `--negotiate` option enables SPNEGO in `curl`.

IE and Safari require no additional configuration to use SPNEGO.

To configure Firefox, access the low level configuration page by loading the about:config page. Then go to the network.negotiate-auth.trusted-uris preference and add the hostname or the domain of the web server that is HTTP Kerberos SPNEGO protected (if using multiple domains and hostname use comma to separate them).

The Chrome browser must be launched from the command line with several added parameters.

To run Chrome on linux:

```
> google-chrome --enable-plugins --args\ --auth-server-
whitelist="*KERBEROS_DOMAIN"\ --auth-negotiate-delegate-
whitelist="*KERBEROS_DOMAIN"\ --auth-
schemes="basic,digest,ntlm,negotiate"
```

To run Chrome on a Mac:

```
> open 'Google Chrome.app' --args\ --auth-server-
whitelist="*ROGUECLOUD.COM"\ --auth-negotiate-delegate-
whitelist="*KERBEROS_DOMAIN"\ --auth-
schemes="basic,digest,ntlm,negotiate"
```

To run Chrome on Windows:

```
chrome.exe --auth-server-whitelist="*KERBEROS_DOMAIN"\ --auth-
negotiate-delegate-whitelist="*ROGUECLOUD.COM"\ --auth-
schemes="basic,digest,ntlm,negotiate"
```

For more information, see Using a Web Browser to Access an URL Protected by Kerberos HTTP SPNEGO and Authentication + Authorization .

Session cookies

A successful Kerberos/SPNEGO login will yield a session cookie, this cookie is identical to the cookie yielded by the Managed Fusion authentication proxy's current POST-login-mechanism.

The expiration policy on the cookie is currently fixed at 8 hours. But has a 1 hour "idle" max, which means if you do not make a request for 1 hour, the cookie is invalidated. Otherwise the lifetime is pushed ahead until the 8 hour max is met.

The name of the cookie is "id" and the value is a UUID. This UUID is a key that maps to an in-memory value containing the real user ID.

Testing and Troubleshooting

Once you have configured the Kerberos security realm, you can test it by logging out of Managed Fusion and shutting down the browser.

Check that you have a valid Kerberos authentication ticket via the klist command.

Now open a new browser session and access the Managed Fusion installation, <domain>:<UI port>.

This should take you directly to the main Managed Fusion panel, bypassing the Managed Fusion "Welcome" login panel. To view your login profile, click on the profile icon at the top right. Your user name should be your login name + "@" + your domain name.

If instead, you see the Managed Fusion login panel, your browser is not configured for SPNEGO.

If the Managed Fusion display consists only of an empty top nav bar, this indicates an authentication failure. Check that the path to your keytab file is correct. Then check the Managed Fusion logs.

If your KDC uses "AES256 CTS mode with HMAC SHA1-96" for key encryption, the proxy will log this error when attempting to authenticate:

```
GSSException: Failure unspecified at GSS-API level (Mechanism level:
Encryption type AES256 CTS mode with HMAC SHA1-96 is not
supported/enabled)
```

To get around this, the Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy File will need to be downloaded and installed. It can be downloaded here :

Place the jars in your JAVA_HOME/jre/lib/security/ directory, then restart Managed Fusion.

Clicking on the "logout" icon on the top nav bar (rightmost icon or a padlock) takes you back to the main Managed Fusion panel. If you destroy your Kerberos credentials cache via the "kdestroy" command, the next time you logout of Managed Fusion, you will be logged out and the browser will display the Managed Fusion login panel.

References and Tutorials

- https://en.wikipedia.org/wiki/Kerberos_%28protocol%29
- http://www.roguelynn.com/words/explain-like-im-5-kerberos/
- https://en.wikipedia.org/wiki/SPNEGO
  https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/6/html/Managing_Smart_Cards/Using_Kerberos.html#about-kerberos
- http://www.oracle.com/technetwork/articles/idm/weblogic-sso-kerberos-1619890.html -section "Install Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files"
- http://www.cisco.com/c/en/us/support/docs/security-vpn/kerberos/16087-1.html

Configure Managed Fusion for Kerberos in Windows

How-to

> Compatible versions: 5.9
>
> Works with: *Managed Fusion*

To configure the Fusion UI service to use Kerberos for user authentication, you must create a Kerberos *security realm*.

Note

Kerberos support requires Fusion 5.9.5.

Kerberos is a system that provides authenticated access for users and services on a network. Instead of sending passwords in plaintext over the network, encrypted passwords are used to generate time-sensitive tickets that are used for authentication. SPNEGO provides a mechanism for extending Kerberos to Web applications through the standard HTTP protocol.

Kerberos uses symmetric-key cryptography and a trusted third party called a Key Distribution Center (KDC) to authenticate users to a suite of network services. (By users we mean both end users and client programs). The computers managed by that KDC and any secondary KDCs constitute a realm. When a user authenticates to the KDC, the KDC sends a set of credentials (a ticket) specific to that session back to the user's machine. Kerberos-aware services use the ticket on the user's machine for authentication instead of requiring sign-on with a password. Because tickets are used rather than passwords, this provides the convenience of Single Sign-On (SSO) in addition to security.

A Kerberized process is one that has been configured so it can get tickets from a KDC and negotiate with Kerberos-aware services. When a user sends an HTTP request, Fusion tries to authenticate using the Kerberos/SPNEGO protocol. If the request was sent from a browser, Fusion does not display the initial sign-on panel; instead on login, the user sees the main Fusion collections panel.

Note

Kerberos support requires Managed Fusion 5.9.5.

Important

This article focuses on configuring Managed Fusion for Kerberos *in Windows*. To learn how to configure Fusion for Unix, see *Configure Managed Fusion for Kerberos in Unix*.

1. Set up and Configure the Active Directory Domain Service

1.1. Install the Active Directory Domain Service

To begin, follow Microsoft's instructions on how to install Active Directory Domain Services.

This topic uses the following domain throughout this example:

NETBIOS Domain	FUSIONIS
Fully qualified domain name	fusionis.life

1.2. Create a New User

Launch the Server Manager. Navigate to **Tools > Active Directory Users and Computers**. Within the `fusionis.life` server, locate the "Users" folder, right-click on the folder and select **New > User**.

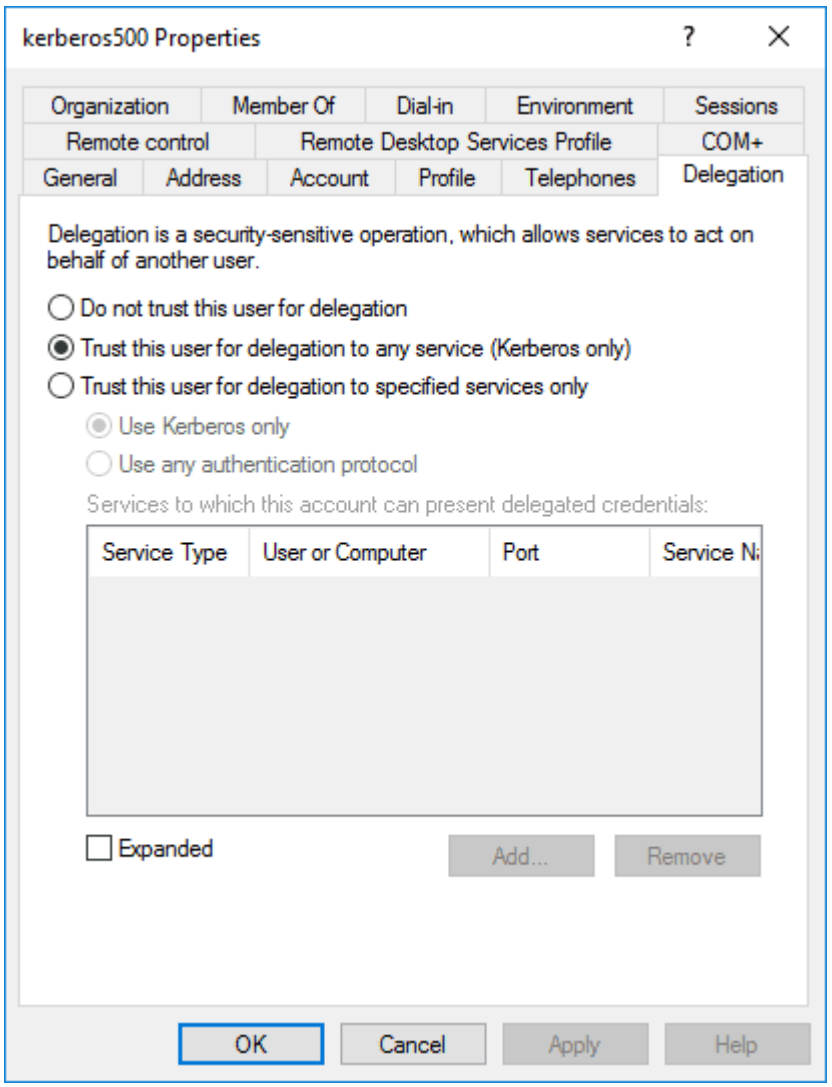
Important

When creating a user for this purpose, ensure the `First name` and `Display name` fields both hold the exact same value.

These instructions use the following user name throughout this example:

Distinguished name (DN)	CN = kerberos500 CN = Users DC = fusionis DC = life
User Principal Name (UPN)	kerberos500@fusionis.life
NETBIOS Domain\samAccountName	FUSIONIS\kerberos500

Establish the desired delegation rules for the new user in the "Delegation" tab.



1.3. Create a Keytab for the New User

Enter the following command into the command prompt to create a keytab for the user you just created:

```
ktpass -out c:\Users\Administrator\kerberos500.keytab -princ
HTTP/fusionis.life@FUSIONIS.LIFE -mapUser FUSIONIS\kerberos500 -mapOp
set -pass YOUR_PASSWORD -crypto AES256-SHA1 -pType KRB5_NT_PRINCIPAL -
kvno 0
```

+

Note

The following weak encryption types are not supported by Fusion:

- DES-CBC-CRC
- DES_CBC_MD5
- Microsoft Windows 2000 RC4-HMAC

The output will be similar to:

```
C:\Users\Administrator>ktpass -out
c:\Users\Administrator\kerberos500.keytab -princ
HTTP/fusionis.life@FUSIONIS.LIFE -mapUser kerberos500 -mapOp set -pass
FroFro123# -crypto AES256-SHA1 -pType KRB5_NT_PRINCIPAL Targeting
domain controller: WIN-OVV6VHBGIB8.fusionis.life Using legacy password
setting method Successfully mapped HTTP/fusionis.life to kerberos500.
Key created. Key created. Key created. Key created. Key created.
Output keytab to c:\Users\Administrator\kerberos500.keytab: Keytab
version: 0x502 keysize 71 HTTP/fusionis.life@FUSIONIS.LIFE ptype 1
(KRB5_NT_PRINCIPAL) vno 3 etype 0x1 (DES-CBC-CRC) keylength 8
(0xcd07200bea625d20) keysize 71 HTTP/fusionis.life@FUSIONIS.LIFE ptype
1 (KRB5_NT_PRINCIPAL) vno 3 etype 0x3 (DES-CBC-MD5) keylength 8
(0xcd07200bea625d20) keysize 79 HTTP/fusionis.life@FUSIONIS.LIFE ptype
1 (KRB5_NT_PRINCIPAL) vno 3 etype 0x17 (RC4-HMAC) keylength 16
(0xe39a141de38abd8750bf9c0bf49fd1c5) keysize 95
HTTP/fusionis.life@FUSIONIS.LIFE ptype 1 (KRB5_NT_PRINCIPAL) vno 3
etype 0x12 (AES256-SHA1) keylength 32
(0xdd6018ab718a4c312038189a20498f70c07d09a19d77ceea31cdd4cd1b08800a)
keysize 79 HTTP/fusionis.life@FUSIONIS.LIFE ptype 1
(KRB5_NT_PRINCIPAL) vno 3 etype 0x11 (AES128-SHA1) keylength 16
(0x9b610485eae2d97dcf0ecaf4ab38a32b)
```

This process will create the **`kerberos500.keytab`** file required to set up the Kerberos realm in Fusion. The command will also set the service principal to **`HTTP/fusionis.life@FUSIONIS.LIFE`**.

Important

If you see the following, the user was not created correctly:

```
DsCrackNames returned 0x2 in the name entry for
kerberos500. Ktpass:failed getting target domain for
specified user.
```

Delete the user, and repeat *the steps above*.

2. Set up Kerberos on Managed Fusion

2.1. Set up the Kerberos Realm

Get the keytab file copied to the server.

Note

The keytab file must be installed on each Fusion node.

Log into Fusion as a native admin and navigate to **System > Access Control > Security Realms > New Realm**. Use the following parameters:

Type	Kerberos
Keytab file	The path to the keytab file you saved on the server
Service principal	The service principal you generated in the previous step
Kerberos name mapping	The field that specifies the mapping from a Kerberos Principal name to a short name

You can specify the default rule, which should work for most situations:

```
DEFAULT RULE:[1:$1@$0](.*@.*)s/@.*//
```

You can also add custom rules. See Mapping Kerberos Principals to Short Names for more information on creating custom rules.

Save your new realm before continuing. You should now be able to test with Negotiate authentication.

3. Verify Kerberos is Working

You will test to verify that Kerberos is working in the following sections. You must log into Windows as a user in the protected realm, meaning that the service principal name you are using has access to generate a ticket for your user.

Verify with cURL

Windows users can log in as a user within the domain, launch the command prompt, and run the cURL command below:

```
curl --negotiate -u :
http://fusionis.life:8764/api/apollo/collections
```

Linux users may also need to run a `kinit` to login as a user prior to running.

Navigate your web browser to `http://fusionis.life:8764`. If you are automatically logged in, Kerberos is working.

Verify with Google Chrome

To verify Kerberos is working with Google Chrome, run Chrome with the following command parameters:

```
"c:\Program Files (x86)\Google\Chrome\Application\chrome.exe" --auth-
server-whitelist="*fusionis.life" --auth-negotiate-delegate-
whitelist="*fusionis.life" --auth-
schemes="basic,digest,ntlm,negotiate"
```

Navigate your web browser to `http://fusionis.life:8764`. If you are automatically logged in, Kerberos is working.

Verify with Firefox

To verify Kerberos is working with Firefox, navigate to `about:config` in your browser's URL bar. If a message appears stating "This might void your warranty!", click "I accept the risk!".

Search for the string `network.negotiate-auth.trusted-uris`, double-click the field, and enter `fusionis.life` into the field that appears. Click "Ok".

Navigate your web browser to `http://fusionis.life:8764`. If you are automatically logged in, Kerberos is working.

Troubleshooting

DEFECTIVE TOKEN DETECTED

If you encounter this issue, check the `var/log/proxy/proxy.log` after attempting to make a request.

See Stack Overflow for addtional steps to resolve this issue.

KERBEROS NAME MAPPING RULES

You may see the following error if you did not specify a mapping rule and the Proxy could not figure out how to map the Kerberos Principal name to a short name:

```
2021-07-30T15:15:03,584 - ERROR [qtp353842779-
52:clojure.tools.logging$eval11$fn__16@0] - {} - Uncaught application
error!
org.apache.hadoop.security.authentication.util.KerberosName$NoMatching
Rule: No rules applied to
YOURUSERNAME@YOURDOMAIN.SUBDOMAIN.EXAMPLE.COM
```

Adjust your Kerberos name mapping rules to resolve this issue.

Configure Managed Fusion for LDAP

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

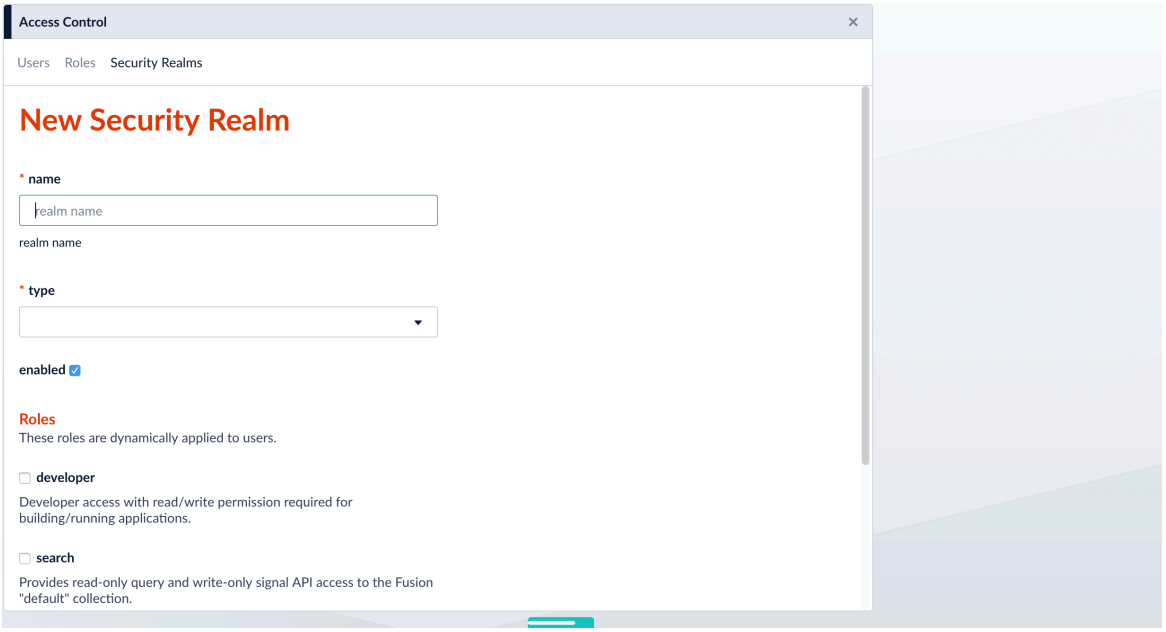
You can create security realms that use external LDAP servers for authentication. Optionally, Managed Fusion can search in the LDAP for groups to which a user belongs, and then map those groups to roles. Managed Fusion performs authorization using permissions stored in Managed Fusion users and roles.

Note: Managed Fusion *does not* use permissions from the LDAP for authorization of UI access or API requests. It only obtains group names (optionally), which are mapped to role names. If an *Active Directory Security Query Trimming Stage* is used, then directory-service permissions *are* used for trimming. If a connector supports security trimming, then connector permissions *are* used for trimming.

To configure Managed Fusion to use an external LDAP as an authentication provider, you'll need to get information about the LDAP server(s) running on your system, either from your system or your sysadmin.

1. Add an LDAP Security Realm

1. Log in to the Managed Fusion UI as the user admin, or as a different user with corresponding permissions.
2. Navigate to **System** > **Access Control**.
3. Click **Security Realms**.
4. Click **Add Security Realm**.



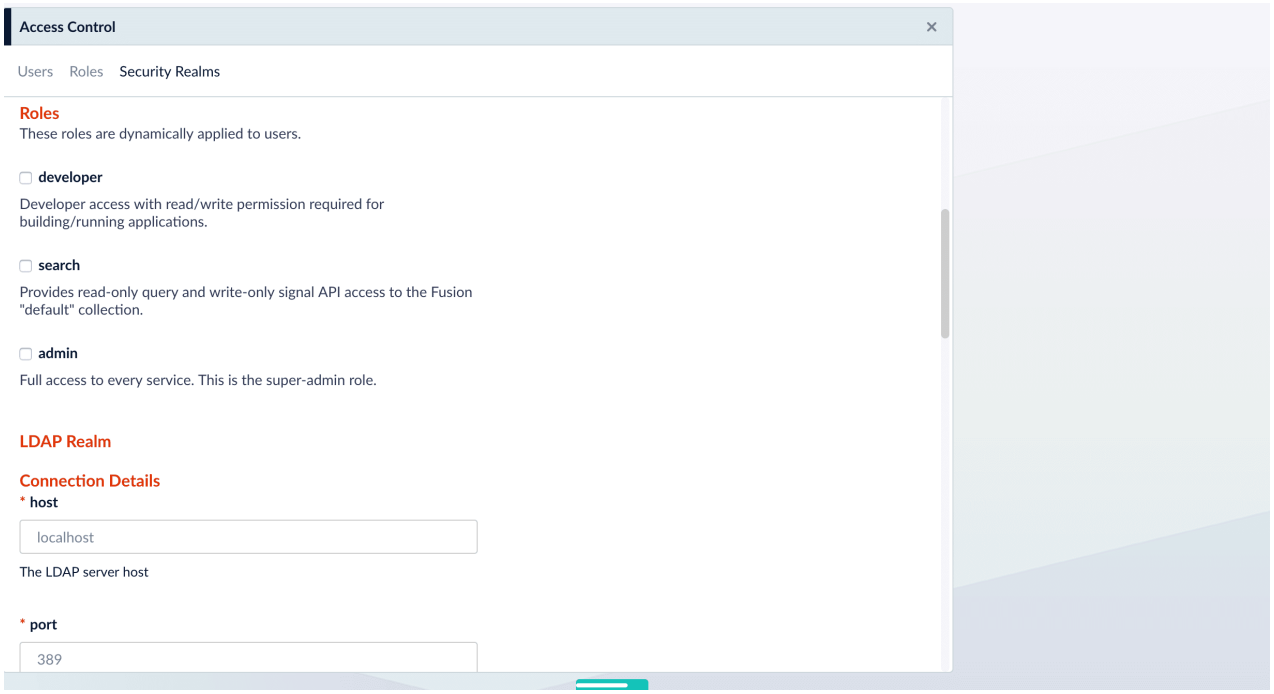
5. Specify info for the new realm:
   - `name` – Name of the security realm. It must be unique. It should be descriptive but short.
   - `type` – Select **ldap** from the pulldown menu.
     When you select **ldap**, Managed Fusion displays additional, LDAP-specific configuration options.
   - `enabled` checkbox – Whether Managed Fusion allows user logins for this security realm. The default is yes (checked).
   - `auto-create users` checkbox – Whether a user account is created automatically upon initial authentication. The default is yes (checked). If the checkbox is unchecked, then a Managed Fusion user with admin permissions must create Managed Fusion users.

6. Scroll down and specify additional options as explained in detail below.

2. Specify Static Roles (Optional)

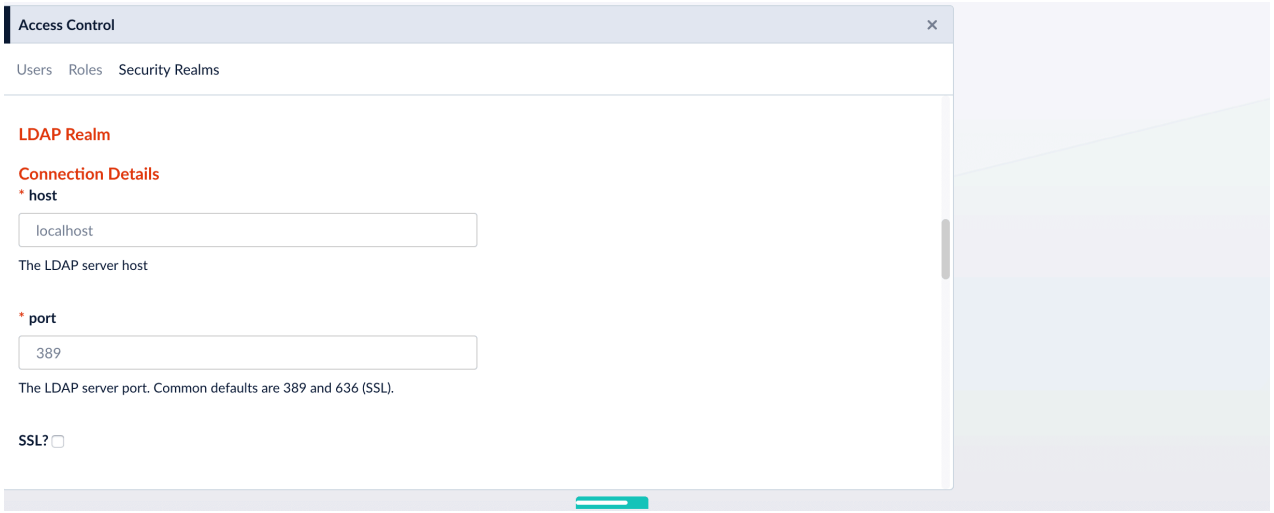
Specify one or more Managed Fusion roles for the security realm. These roles are always considered. They do not depend on searching for LDAP groups and mapping group names to Fusion role names.

In a security realm, you can specify these static roles, add to the list of roles dynamically through an LDAP search, or both. If you do neither, Managed Fusion uses only the role(s) and permissions defined for the user.



3. Specify LDAP Connection Details

Specify the hostname and port of the LDAP server. Check the checkbox if the server is running over SSL.



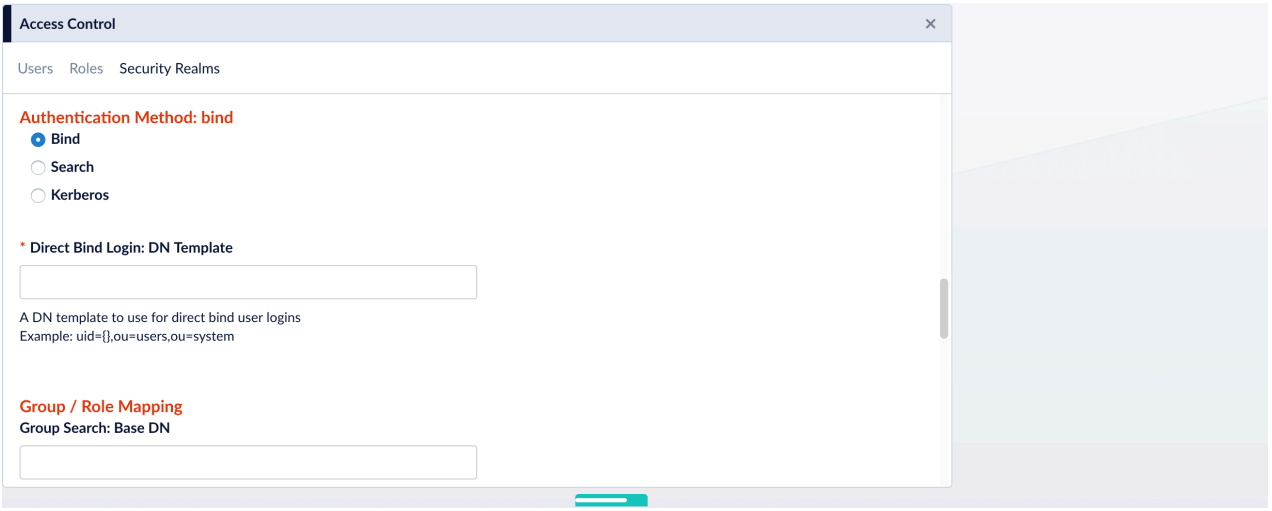
4. Specify the Authentication Method

Specify the authentication method:

- **Bind.** LDAP authentication is carried out via a single "Bind" operation. See *Bind* below.
- **Search.** LDAP authentication is carried out indirectly via a Search operation followed by a Bind operation. See *Search* below.
- **Kerberos.** Kerberos authenticates Managed Fusion and an LDAP Search operation is carried out to find group-level authorizations. See *Kerberos* below.

Bind

Use the Bind authentication method when the Managed Fusion login username matches a part of the LDAP distinguished name (DN). Specify the remainder of the LDAP DN in the "DN Template" configuration entry, which uses a single pair of curly braces (`{}`) as a placeholder for the value of the Managed Fusion username.

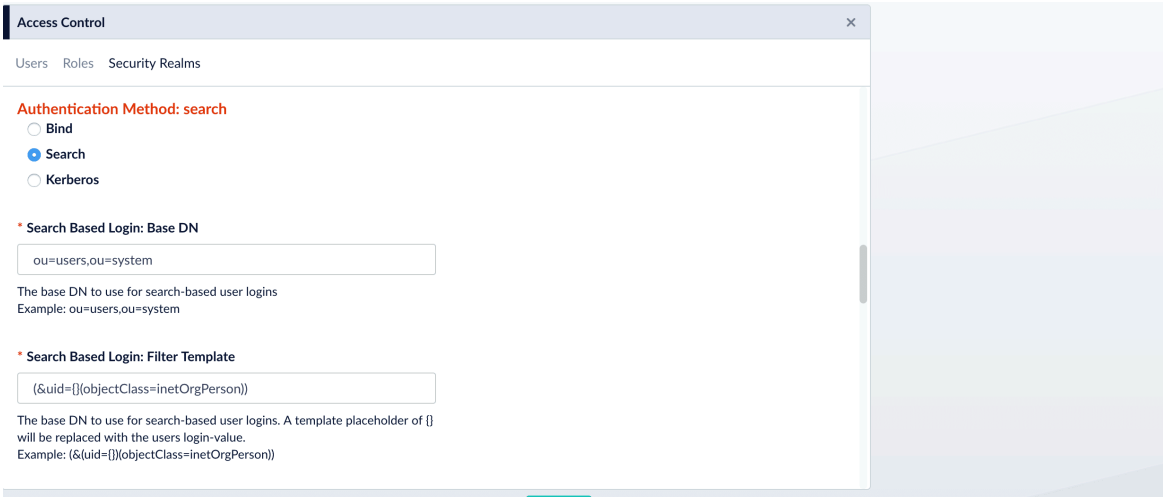


Search

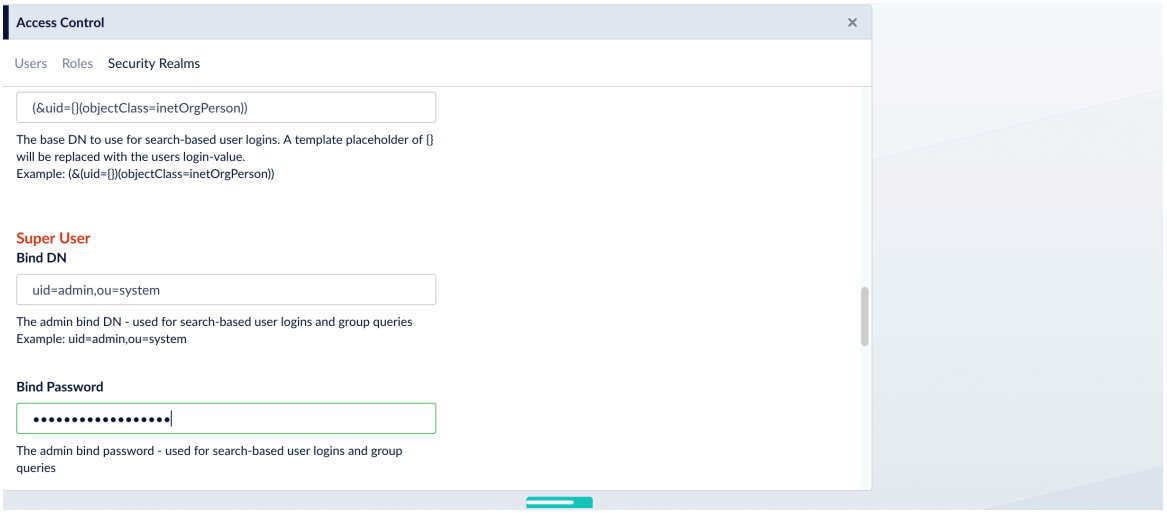
Use the Search authentication method when the username used for Managed Fusion login *doesn't* match a part of the LDAP DN. The search request returns a valid user DN, which is then used together with the user password for authentication via a Bind request.

1. Construct a search request.
   The Search authentication method is generally required when working with Microsoft Active Directory servers. In this case, you need to know the username and password of *some* user who has sufficient privileges to query the LDAP server for user and group memberships; this user doesn't have to be the superuser.

   In addition to a privileged user DN and password, the Search authentication method requires constructing a search request. There are two parts to the request. The first part is the base DN of the LDAP directory tree that contains user account objects. The second part of the request is a Search Filter object that restricts the results to a matching subset of the information.

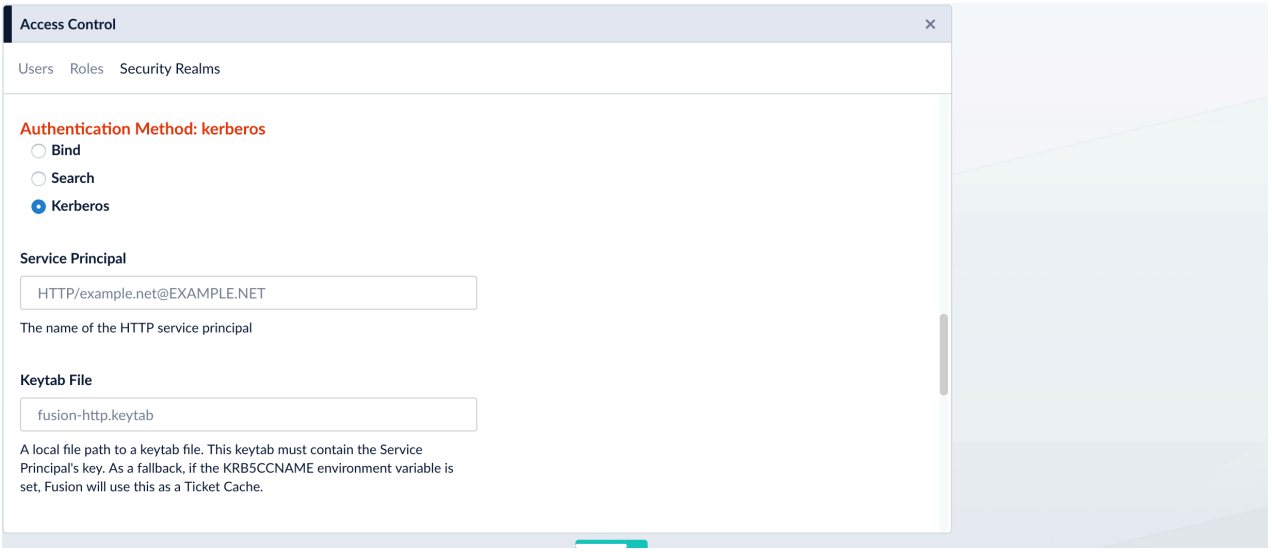


2. Provide the administrator bind DN:



Kerberos

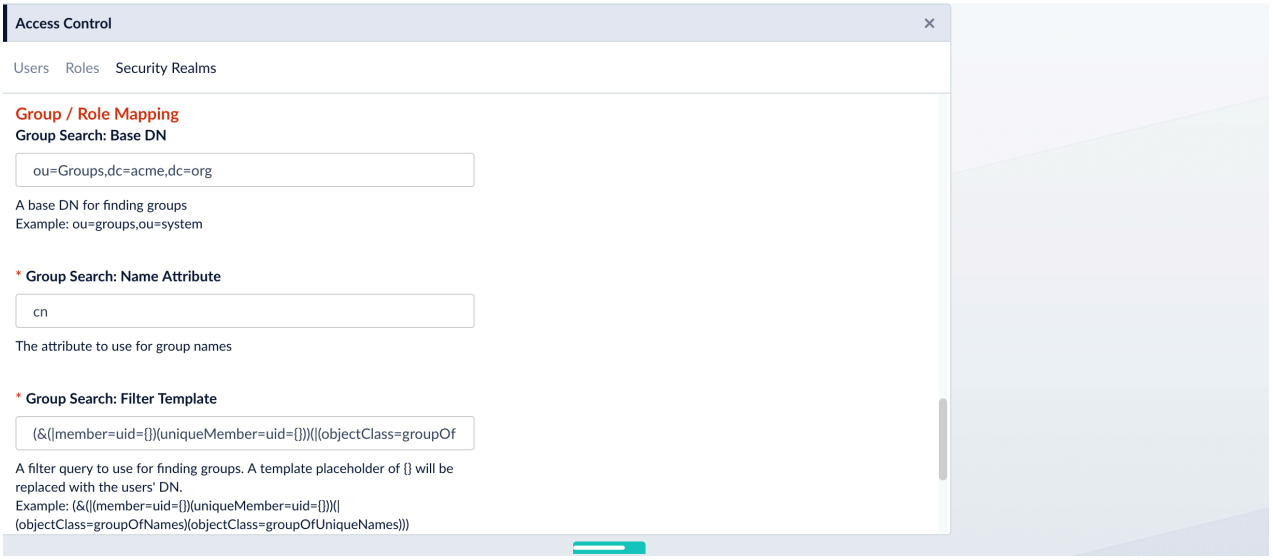
Use the Kerberos authentication method when Kerberos is the authentication provider.



5. Search for LDAP Groups (Optional)

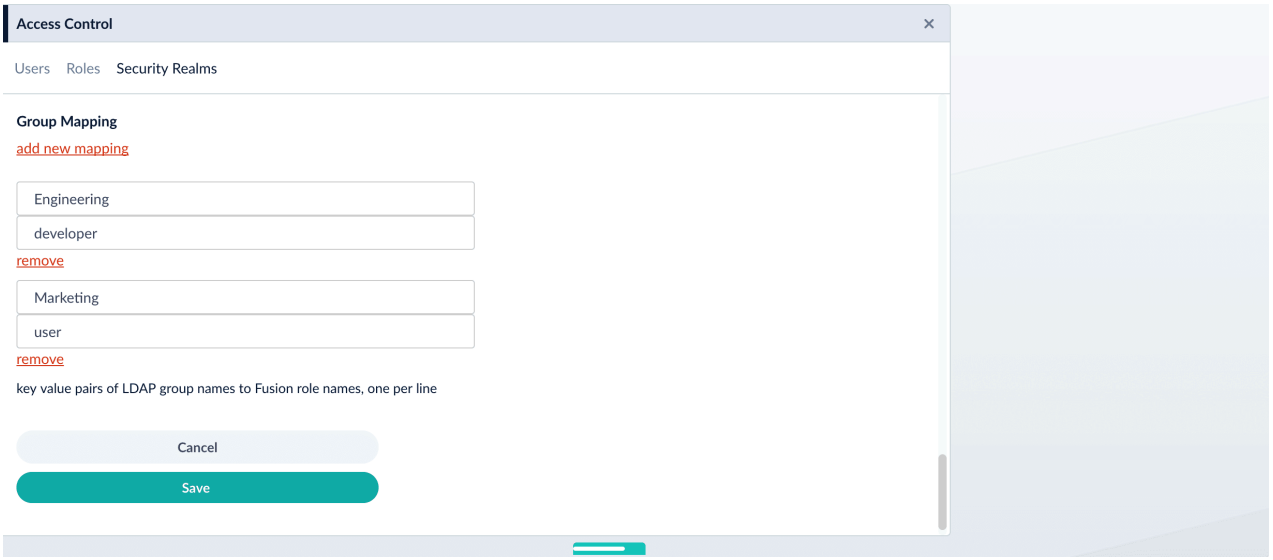
A Managed Fusion role is a bundle of permissions tailored to the access needs of different kinds of users. Access to services and data for LDAP-managed users is controlled by mappings from LDAP users and groups to Managed Fusion roles.

Roles can be assigned globally or restricted to specific LDAP groups. The security realm configuration panel contains a list of all Managed Fusion roles with a checkbox for each, used to assign that role to all users in that realm. LDAP group names can be mapped directly to specific Managed Fusion roles and LDAP group search and filter queries can also be used to map kinds of LDAP users to specific Managed Fusion roles.



6. Map LDAP Groups to Managed Fusion Roles (Optional)

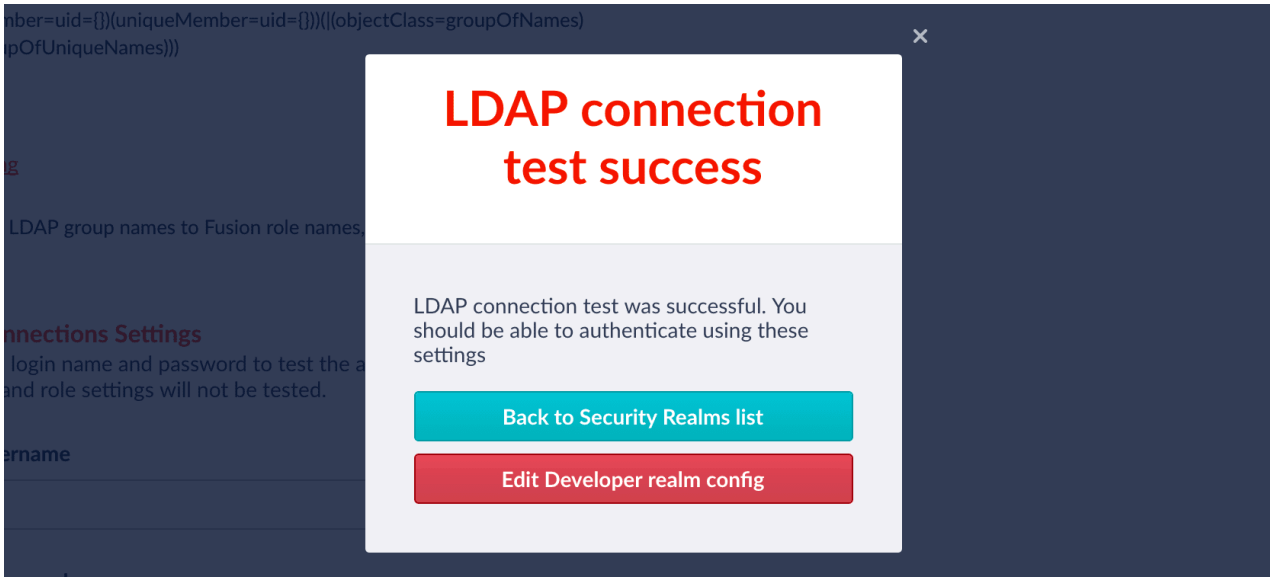
If LDAP group names returned by the search for groups match Managed Fusion role names, you do not need to map the group names to role names. You must map any LDAP group names that do not match to Managed Fusion role names (if you do not, they will not be used).



7. Save the Security Realm Configuration

Click **Save**.

Managed Fusion reports whether or not authentication was successful:



8. Basic LDAP Concepts and Terminology

The LDAP protocol is used to share information about users, systems, networks, and services between servers on the internet. LDAP servers are used as a central store for usernames, passwords, and user and group permissions. Applications and services use the LDAP protocol to send user login and password information to the LDAP server. The server performs name lookup and password validation. LDAP servers also store Access Control Lists (ACLs) for file and directory objects which specify the users and groups and kinds of access allowed for those objects.

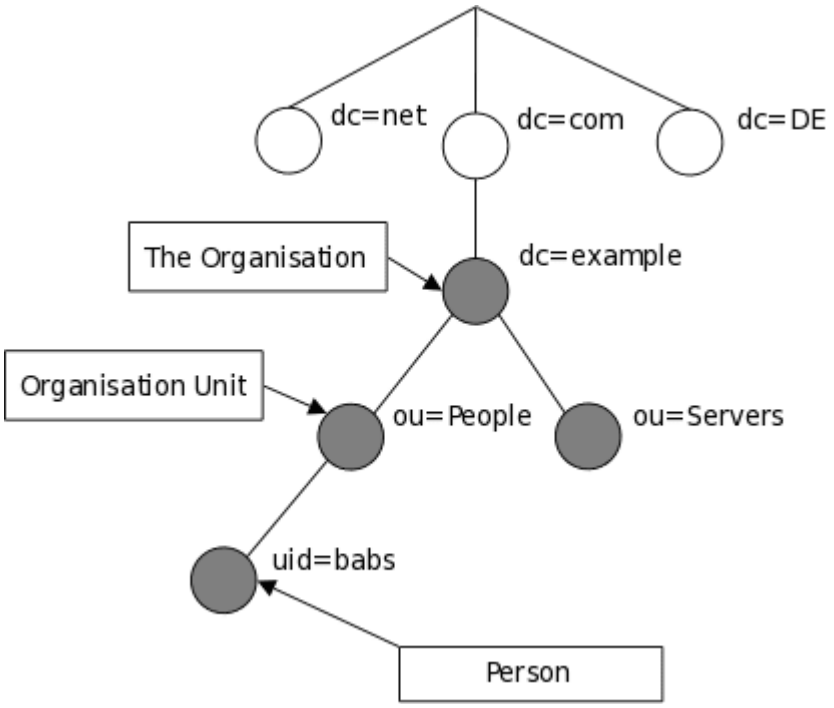
LDAP is an open standard protocol and there are many commercial and open source LDAP servers available. Microsoft environments generally use Active Directory. Unix servers use AD or other LDAP systems such as OpenLDAP, although many Unix systems do not use LDAP at all. To configure Managed Fusion for LDAP, you'll need to get information about the LDAP server(s) running on your system either from your sysadmin or via system utilities.

Directories and Distinguished Names

An LDAP information store is a Directory Information Tree (DIT). The tree is composed of entry nodes; each node has a single parent and zero or more child nodes. Every node must have at least one attribute which uniquely distinguishes it from its siblings which is used as the node's Relative Distinguished Name (RDN). A node's Distinguished Name (DN) is a globally unique identifier.

The string representation of a DN is specified in RFC 4514. It consists of the node's RDN followed by a comma, followed by the parent node's DN. The string representation of the RDN is the attribute-value pair name, connected by an equals ("=") sign. This recursive definition means that the DN of a node is composed by working from the node back through its parent and ancestor nodes up to the root node.

Here is a small example of a DIT:

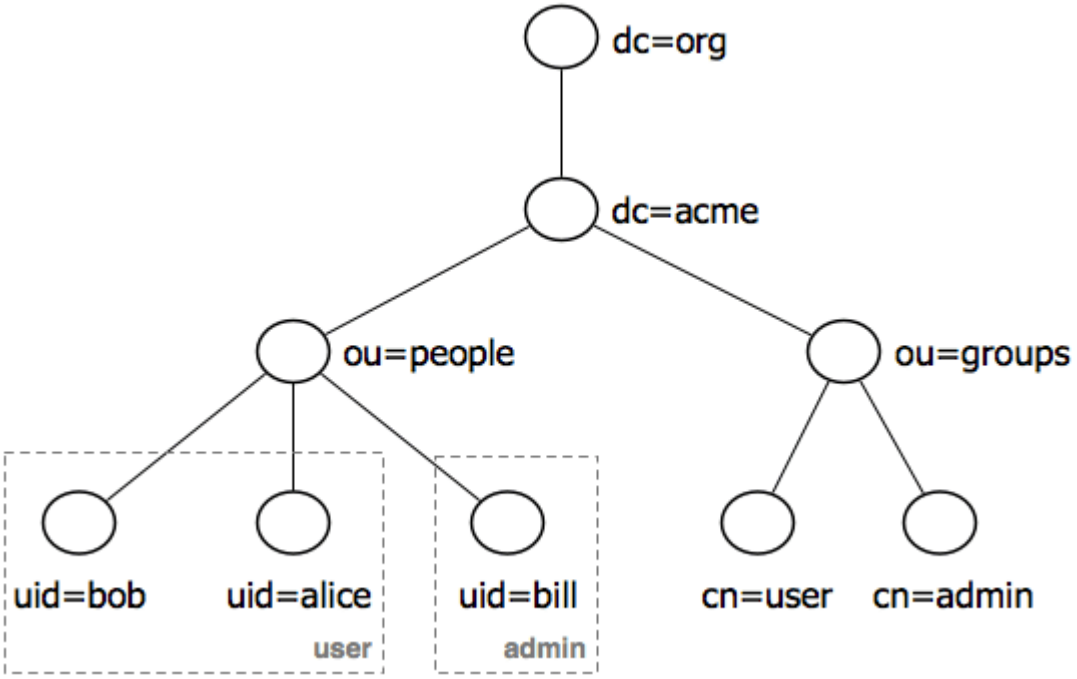


The person entry in this tree has the DN: "uid=babs, ou=people, dc=example, dc=com".

Attribute names include many short strings based on English words and abbreviations, e.g.:

Name	Description
cn	commonName
dc	domainComponent
mail	email address
ou	organizationalUnitName
sn	surname
uid	userId

LDAP entry attributes can refer to other LDAP entries by using the DN of the entry as value of that attribute. The following example of a directory which contains user and groups information shows how this works:



This tree contains two organizational units: "ou=people" and "ou=groups". The children of the "group" organizational unit are specific named groups, just as the child nodes of organization unit "people" are specific users. There are three user entries with RDNs "uid=bob", "uid=alice", "uid=bill" and two groups with RDNs "cn=user" and "cn=admin". The dotted lines and group labels around the person nodes indicate group membership. This relationship is declared on the groups nodes by adding an attribute named "member" whose value is a users DN. In the LDAP data interchange format (LDIF), this is written:

```
cn=user,ou=groups,dc=acme,dc=org member:
uid=bob,ou=people,dc=acme,dc=org member:
uid=alice,ou=people,dc=acme,dc=org cn=admin,ou=groups,dc=acme,dc=org
member: uid=bill,ou=people,dc=acme,dc=org
```

See the Wikipedia's LDAP entry for details.

LDAP Protocol Operations

For authentication purposes, Managed Fusion sends Bind operation requests to the LDAP server. The Bind operation authenticates clients (and the users or applications behind them) to the directory server, establishes authorization identity used for subsequent operations on that connection, and specifies the LDAP protocol version that the client will use.

Depending on the way that the host system uses LDAP to store login information about users and groups, it may be necessary to send Search operation requests to the LDAP server as well. The Search operation retrieves partial or complete copies of entries matching a given set of criteria.

LDAP filters specify which entries should be returned. These are specified using prefix notation. Boolean operators are "&" for logical AND, "|" for logical OR, e.g., "A AND B" is written "(&(A)(B))". To tune and test search filters for a Unix-based LDAP system, see the ldapsearch command line utility documentation. For Active Directory systems, see AD Syntax Filters.

Configure Managed Fusion for SAML

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

SAML 2.0 is a standard for exchanging authentication and authorization data between security domains.

The SAML protocol allows web-browser single sign-on (SSO) through a sequence of messages sent to and from the browser, which is the intermediary between Managed Fusion and the SAML authority acting as the Identity Provider (IdP).

To configure Managed Fusion to use SAML 2.0 for user authentication and authorization you must create a SAML *security realm*. In addition to configuring the Managed Fusion security realm, you must configure the SAML identity provider to recognize the Managed Fusion application.

Once Managed Fusion is configured for a SAML realm, this realm is added to the list of available realms on the initial Managed Fusion sign-on panel. When the SAML realm is chosen from the list of available realms, the browser then redirects to the IdP which handles user authentication. Upon successful authentication, the IdP sends a response back to the browser which contains authentication and authorization information as well as the URL of the Managed Fusion application. The browser redirects back to the Managed Fusion URL, passing along the SAML message with the user authentication and authorization information. Managed Fusion then issues as session cookie which is used for subsequent user access.

Add a Security Realm

1. Sign in to Managed Fusion and click your application.
2. Click **System > Access Control > Security Realms > Add Security Realm**.
3. In the Name field, enter a unique name for the security realm.
4. Click **Type** and select **saml**.
5. In the additional configuration options, review the default values and determine if you want to change them:
    - The default value for **Enabled** is true. This setting controls whether or not Managed Fusion allows user logins for this security realm.
    - The default value for **Ephemeral Users** is false. When disabled, this setting prevents ephemeral users from being created in ZooKeeper during login. If enabled, this property negates **Auto Create Users**.
    - The default value for **Auto Create Users** is true. If enabled, a user account is created automatically upon initial authentication. If disabled, a Managed Fusion user with admin permissions must create Managed Fusion users.

6. In the SAML Realm section, complete the following fields:
    - In the Identity Provider URL field, enter the URL used by the SAML authority for single sign-on. For example: `https://www.my-idp.com/<my-app-path>/sso/saml`.

      Note

      The URL format may differ depending on the SAML identity provider.

    - In the IdP Issuer field, enter the URL for the SAML identity provider. The IdP Issuer must match the `<saml:Issuer>` in the SAML payload. For example: `http://www.my-idp.com/exk686w2xi5KTuSXz0h7`.
    - In the App Issuer field, enter the URL for the application source. This field is only required if there is an `audienceRestriction` in the SAML assertion, in which case, the field must match `<saml:Audience>` in the SAML payload.
    - In the Certificate Fingerprint field, paste the contents of the SAML authority certificate without the certificate header and footer.
    - In the User ID Attribute field, enter the username or identifier. By default, the Managed Fusion username is the same as the login name known to the Identity Provider. When another field or attribute in the user record stored by the IdP should be used as the Managed Fusion username, that attribute name is the value of the User ID Attribute.
    - In the Post Login Redirect URL field, enter a URL to display after the user signs in. If not set, the Managed Fusion URL is used.
    - In the Logout URL field, enter a URL that displays when the user signs out. This field is optional.
    - In the Groups Mapping section, enter a value in Group Name Attribute and add group mappings.

7. Click **Save**.

Configure SAML identity provider

To finish setup, register Managed Fusion with your SAML identity provider. The amount of information required to register varies depending on the SAML authority.

Required information

Managed Fusion URL. This displays when the user successfully signs in. The information includes the:

- Protocol
- Server
- Managed Fusion application port
- Path format of `api/saml`.

For example, `https://EXAMPLE_COMPANY.b.lucidworks.cloud:6764/api/saml`.

If the Managed Fusion application is running behind a load-balancer, this URL is the load-balancer URL plus path `api/saml`. The load-balancer must be session-sticky so the sequence of messages that comprise the SAML protocol successfully run to completion.

Additional information

Some authorities may require additional information. In particular, the SAML 2.0 `AudienceRestriction` tag may be part of the SAML message. This tag specifies the domain where the SAML trust conditions are valid. This domain is usually where the Managed Fusion app is running. For example, `https://EXAMPLE_COMPANY.b.lucidworks.cloud`.

Example SAML Realm configuration

The Managed Fusion endpoint `api/realm_configs` returns a JSON list of all the configuration objects for all realms. After configuring a SAML realm named `"saml_test"` using the `okta.com` developer preview tool, the configuration object for this realm is:

```
{ "name":"saml-test", "realmType":"saml", "enabled":true, "config":{
"autoCreateUsers":true, "idpUrl":"https://dev-
417804.oktapreview.com/app/dev417804_1/exk686w2xi5KTuSXz0h7/sso/saml",
"issuer":"http://www.okta.com/exk686w2xi5KTuSXz0h7",
"certificateFingerprint":"MIIDpDCCAoygAwIBAgIGAVQr4A4NMA0GCSqGSIb3DQEB
BQUAMIGSMQswCQYDVQQGEwJVUzETMBEG\nA1UECAwKQ2FsaWZvcm5pYTEWMBQGA1UEBwwN
U2FuIEZyYW5jaXNjbzENMAsGA1UECgwET2t0YTEU\nMBIGA1UECwwLU1NPUHJvdmlkZXIx
EzARBgNVBAMMCmRldi00MTc4MDQxHDAaBgkqhkiG9w0BCQEW\nDWluZm9Ab2t0YS5jb20w
HhcNMTYwNDE5MDAxNTI0WhcNMjYwNDE5MDAxNjI0WjCBkjELMAkGA1UE\nBhMCVVMxEzAR
BgNVBAgMCkNhbGlmb3JuaWExFjAUBgNVBAcMDVNhbiBGcmFuY2lzY28xDTALBgNV\nBAoM
BE9rdGExFDASBgNVBAsMC1NTT1Byb3ZpZGVyMRMwEQYDVQQDDApkZXYtNDE3ODA0MRwwGg
YJ\nKoZIhvcNAQkBFg1pbmZvQG9rdGEuY29tMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMI
IBCgKCAQEA\nuNz0trRFPw2d4xXoUvX2oWeZolTVeFtaTnB9SWUyjK4og0WdT7rNdBg10e
TvB2ezBwXCf24UGGui\nr1kjkZjiHDqDxKtzQYWpGuzLCjh/4PxKFGDaiUNKcE1Ig5myiE
BTvMvv99XtrcI75QdUGDhbMiBr\n2PR5FukWOYepzlBzqY0JSDzX9NYJBKPkz+syK4mj0I
6dqtYOU+bcTvjF9sR7jiHtQ+d0Zl8rz1Ca\nyuE3mNUtFJ1IJrY/RArhH1AB6mXbV/de1C
XmGhKQbqQAbx9SiKtki9n84gKEwuWdV0jIqcBLGxUQ\ngjbsaIVqed2oX+7F2fh6t0g/I8
NPnOWXOTvA+wIDAQABMA0GCSqGSIb3DQEBBQUAA4IBAQCPt+DR\nliIsHO/iVnmLFPGfqr
CO/gMv++xnq2wOB19YX7HhT1GIY2YZoVphrQpXH3OO0/8AZJ8ApCmq0E9x\nxUTQwQPBan
VqlyLtu1Hr0c6dbAqcd5PtaEe6Ci33nayWPydhOmitvIyb/WtWtbel9HcdUkoGvkAl\ng3
05jnxkhwGLJm4jkzYe+eaYhd6oG3/JcHqKDGYGf7i2Z+ny0D7vxTeBQ+8PbZfsUKg0PlKy
Tocb\nydSmDPISsA1xOH5zlw+hzFIdKgD5vW7QZLvpIclNc2hqki/nWl/CHut0TnUuP/V3
boREmhDu395n\n/u72pgNANfP7+2DTBb+CBTjGUsAxpKRF", "userIdAttribute":""
}, "roleNames":["search"], "id":"52e1c0d2-5e00-4c76-a3d4-
57f1381bdb4f", "createdAt":"2016-04-19T19:49:04Z", "updatedAt":"2016-
04-19T20:06:56Z" }
```


References

- SAML 2.0 assertions and protocols
- SAML 2.0 glossary

Configure Managed Fusion for SSO

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

The "SSO Trusted HTTP" realm type (`trusted-http` in the REST API) is useful in single sign-on (SSO) environments.

If SSO is already set up in your environment, user identities and group information can be sent to Fusion through HTTP headers (REMOTE_USER, for example). The SSO Trusted HTTP realm type provides the configuration options for integrating this into Fusion's authentication systems. It also supports allowing access to only a set of known client IPs, and mapping groups to Fusion roles.

Use the *Realms API* to configure this realm type:

```
curl -u USERNAME:PASSWORD -H 'content-type:application/json' -X POST
:3000/api/realm-configs -d @./realm-config.json
```

Below is a sample configuration:

```
{"id":"test-id", "enabled":true, "name":"sso-test",
"realmType":"trusted-http", "config":{"identityKey":"REMOTE_USER",
"groups": {"key":"GROUPS", "delimiter":"|", "roleMapping":
[["a","admin"], ["b","foo"]]}, "allowedIps":["127.0.0.1",
"0:0:0:0:0:0:0:1", "localhost"]}}
```

`identityKey`	The name of an HTTP header. If this key is found in the request headers, its value is used as the identity of the client (username, for example).
`groups`	Configuration keys for auth groups: * `key` + The name of an HTTP header, used as the source of group names. * `delimiter` + The character used to split the value (defaults to comma). * `roleMapping` + A set of 2-tuples, used for mapping the external group values to Fusion Roles.
`allowedIps`	Allow access to only a set of known client IPs. When this property is defined and the client IP is not included in it, the realm logic return a 401. Leaving this field empty makes the realm nonoperational. To accept traffic from all destinations for development, you can use an IP such as 0.0.0.0/0. In production the concrete IP should be specified.

Configure Multiple Nodepool Implementations

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion*

Lucidworks recommends isolating search workloads from analytics workloads using multiple node pools. The included scripts do not do this for you; this is a manual process.

See *Workload Isolation with Multiple Node Pools* for more information.

1. Run the `customize_fusion_values.sh` script with the following arguments:

```
./customize_fusion_values.sh \ -c demo \ 1 -n fusion-lw \ 2 --
provider k8s \ --node-pool {} \ 3 --with-resource-limits \ --
with-affinity-rules \ --with-replicas
```

1	The name of your cluster. This example uses `demo`.
2	The name of your namespace.
3	By using the empty value `{}` as the value for the `--node-pool` argument, the script leaves the `nodePools` values unassigned. Assign these values in a later step. The example above outputs the following: `Created Fusion custom values yaml: k8s_ demo_fusion-lw_fusion_values.yaml Created Monitoring custom values yaml: k8s_ demo_fusion-lw_monitoring_values.yaml. Keep this file handy as you'll need it to customize your Monitoring installation. Create k8s_ demo_fusion-lw_upgrade_fusion.sh for upgrading you Fusion cluster. Please keep this script along with your custom values yaml file(s) in version control. Update nodeSelector values for k8s_ demo_fusion-lw_fusion_values.yaml and update the solr to use nodepool`

2. Update Solr in `k8s_demo_fusion-lw_fusion_values.yaml` to add `nodePools`. The `fusion_node_type` value connects the Solr `nodePools` values to the corresponding node pool.

```
solr: nodePools: - name: ""  1  - name: "analytics"  2  javaMem:
"-Xmx6g" replicaCount: 6 storageSize: "100Gi" nodeSelector:
fusion_node_type: analytics  3  resources: requests: cpu: 2
memory: 12Gi limits: cpu: 3 memory: 12Gi - name: "search"  4
javaMem: "-Xms11g -Xmx11g" replicaCount: 12 storageSize: "50Gi"
nodeSelector: fusion_node_type: search  5  resources: limits:
cpu: "7700m" memory: "26Gi" requests: cpu: "7000m" memory:
"25Gi" nodeSelector: cloud.google.com/gke-nodepool: default-pool
6  ...
```

1	The empty string `""` is the suffix for the default partition.
2	Overrides the settings for the **analytics** Solr pods.
3	Assigns the **analytics** Solr pods to the node pool and attaches the label `fusion_node_type=analytics`. You can use the `fusion_node_type` property in Solr auto-scaling policies to govern replica placement during collection creation.
4	Overrides the settings for the **search** Solr pods
5	Assigns the **search** Solr pods to the node pool and attaches the label `fusion_node_type=search`.
6	Sets the default settings for all Solr pods, if not specifically overridden in the `nodePools` section above. **Warning** Do not edit the **`nodePools`** value **`""`**. **Important** Only use a single **`fusion_node_type`** value per Solr **`nodeSelector`** value.

3. Update **`k8s_demo_fusion-lw_fusion_values.yaml`** to add appropriate **`nodeSelector`** values for all desired services.
4. Run the **`upgrade_fusion.sh`** script:
   The output resembles the following:

```
namespace/fusion-lw created Created namespace fusion-lw with
owner label Hang tight while we grab the latest from your chart
repositories... ...  1  NAME: fusion-lw LAST DEPLOYED: Wed Sep 15
18:25:55 2021 NAMESPACE: fusion-lw STATUS: deployed REVISION: 1
... NAME NAMESPACE REVISION UPDATED STATUS CHART APP VERSION
fusion-lw fusion-lw 1 2021-09-15 18:25:55.1470127 -0500 CDT
deployed fusion-5.4.2 5.4.2
```

1	The `...` indicates some parts of the output are removed to shorten the example response.

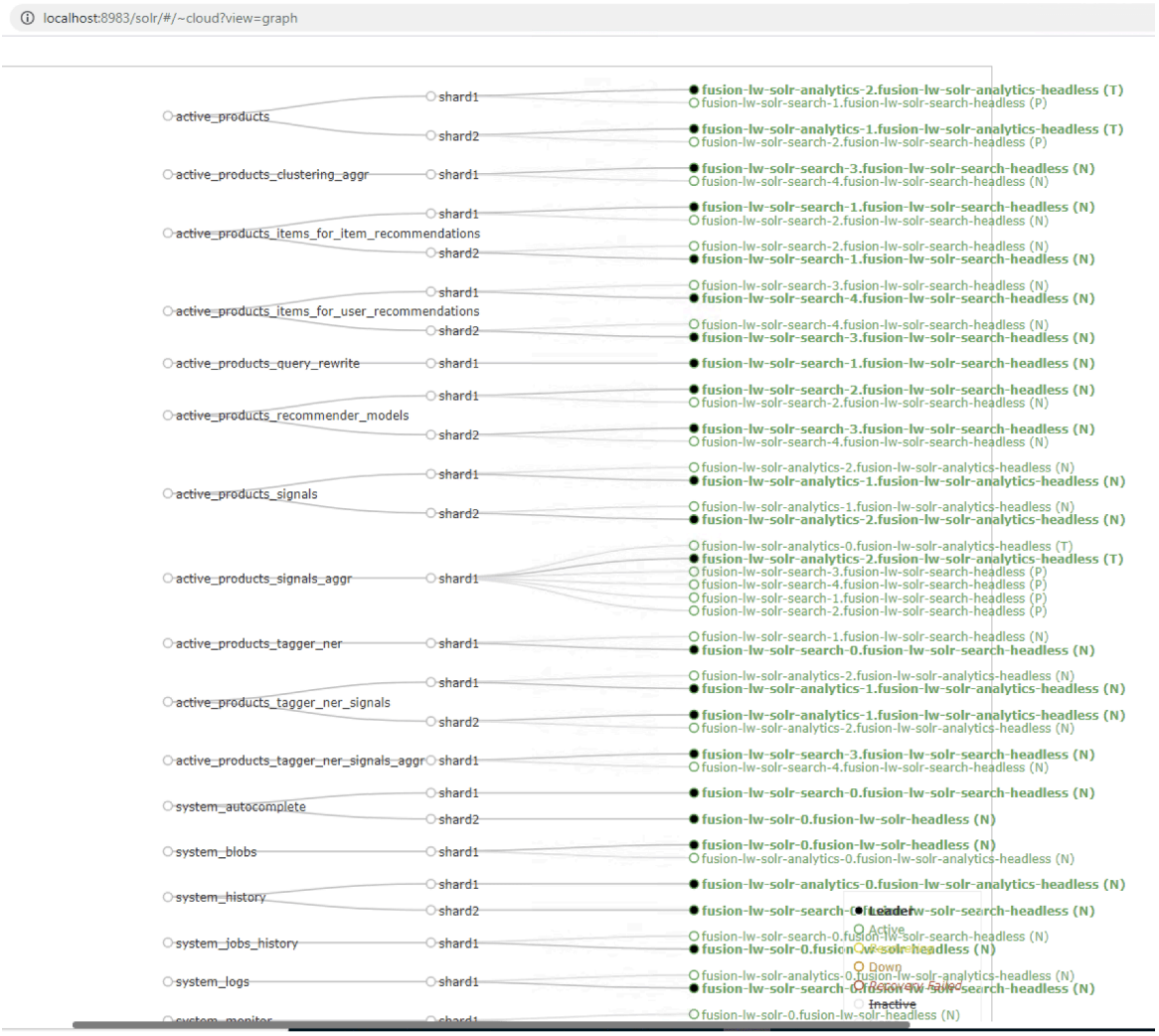
5. Install Prometheus and Grafana so you can monitor Fusion metrics by running the following:

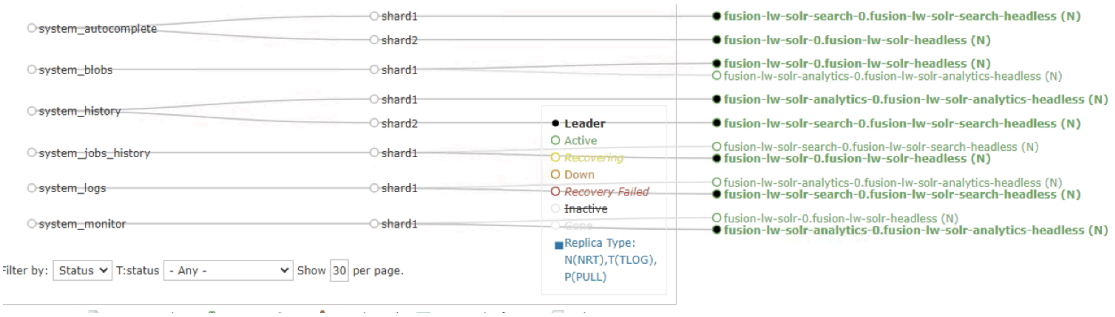
```
./install_prom.sh --provider k8s -c demo -n fusion-lw
```

The output resembles the following:

```
Adding the stable chart repo to helm repo list "stable" has been
added to your repositories Installing Prometheus and Grafana for
monitoring Fusion metrics ... this can take a few minutes. ...
NAME: fusion-lw-monitoring LAST DEPLOYED: Wed Sep 15 22:59:28
2021 NAMESPACE: fusion-lw STATUS: deployed REVISION: 1 NOTES:
The Prometheus server can be accessed via port 80 on the
following DNS name from within your cluster: fusion-lw-
monitoring-prometheus-server.fusion-lw.svc.cluster.local ...
Successfully installed Prometheus and Grafana into the fusion-lw
namespace. NAME NAMESPACE REVISION UPDATED STATUS CHART APP
VERSION fusion-lw fusion-lw 1 2021-09-15 18:25:55.1470127 -0500
CDT deployed fusion-5.4.2 5.4.2 fusion-lw-monitoring fusion-lw 1
2021-09-15 22:59:28.9054416 -0500 CDT deployed fusion-
monitoring-1.0.0 1.0.0
```

6. Update the Solr collections to use specific nodes. Apply the Solr policy and run **`update_app_coll_layout.sh`**. This script deletes the collections and recreates them with the new policy. Check the scripts for details on collection names.


localhost:8983/solr/#/~cloud?view=graph
active_products
shard1
fusion-lw-solr-analytics-2.fusion-lw-solr-analytics-headless (T)
fusion-lw-solr-search-1.fusion-lw-solr-search-headless (P)
shard2
fusion-lw-solr-analytics-1.fusion-lw-solr-analytics-headless (T)
fusion-lw-solr-search-2.fusion-lw-solr-search-headless (P)
active_products_clustering_aggr
shard1
fusion-lw-solr-search-3.fusion-lw-solr-search-headless (N)
fusion-lw-solr-search-4.fusion-lw-solr-search-headless (N)
active_products_items_for_item_recommendations
shard1
fusion-lw-solr-search-1.fusion-lw-solr-search-headless (N)
fusion-lw-solr-search-2.fusion-lw-solr-search-headless (N)
shard2
fusion-lw-solr-search-2.fusion-lw-solr-search-headless (N)
fusion-lw-solr-search-1.fusion-lw-solr-search-headless (N)
active_products_items_for_user_recommendations
shard1
fusion-lw-solr-search-3.fusion-lw-solr-search-headless (N)
fusion-lw-solr-search-4.fusion-lw-solr-search-headless (N)
shard2
fusion-lw-solr-search-4.fusion-lw-solr-search-headless (N)
fusion-lw-solr-search-3.fusion-lw-solr-search-headless (N)
active_products_query_rewrite
shard1
fusion-lw-solr-search-1.fusion-lw-solr-search-headless (N)
active_products_recommender_models
shard1
fusion-lw-solr-search-2.fusion-lw-solr-search-headless (N)
fusion-lw-solr-search-2.fusion-lw-solr-search-headless (N)
shard2
fusion-lw-solr-search-3.fusion-lw-solr-search-headless (N)
fusion-lw-solr-search-4.fusion-lw-solr-search-headless (N)
active_products_signals
shard1
fusion-lw-solr-analytics-2.fusion-lw-solr-analytics-headless (N)
fusion-lw-solr-analytics-1.fusion-lw-solr-analytics-headless (N)
shard2
fusion-lw-solr-analytics-1.fusion-lw-solr-analytics-headless (N)
fusion-lw-solr-analytics-2.fusion-lw-solr-analytics-headless (N)
active_products_signals_aggr
shard1
fusion-lw-solr-analytics-0.fusion-lw-solr-analytics-headless (T)
fusion-lw-solr-analytics-2.fusion-lw-solr-analytics-headless (T)
fusion-lw-solr-search-3.fusion-lw-solr-search-headless (P)
fusion-lw-solr-search-4.fusion-lw-solr-search-headless (P)
fusion-lw-solr-search-1.fusion-lw-solr-search-headless (P)
fusion-lw-solr-search-2.fusion-lw-solr-search-headless (P)
active_products_tagger_ner
shard1
fusion-lw-solr-search-1.fusion-lw-solr-search-headless (N)
fusion-lw-solr-search-0.fusion-lw-solr-search-headless (N)
active_products_tagger_ner_signals
shard1
fusion-lw-solr-analytics-2.fusion-lw-solr-analytics-headless (N)
fusion-lw-solr-analytics-1.fusion-lw-solr-analytics-headless (N)
shard2
fusion-lw-solr-analytics-1.fusion-lw-solr-analytics-headless (N)
fusion-lw-solr-analytics-2.fusion-lw-solr-analytics-headless (N)
active_products_tagger_ner_signals_aggr
shard1
fusion-lw-solr-search-3.fusion-lw-solr-search-headless (N)
fusion-lw-solr-search-4.fusion-lw-solr-search-headless (N)
system_autocomplete
shard1
fusion-lw-solr-search-0.fusion-lw-solr-search-headless (N)
shard2
fusion-lw-solr-0.fusion-lw-solr-headless (N)
system_blobs
shard1
fusion-lw-solr-0.fusion-lw-solr-headless (N)
fusion-lw-solr-analytics-0.fusion-lw-solr-analytics-headless (N)
system_history
shard1
fusion-lw-solr-analytics-0.fusion-lw-solr-analytics-headless (N)
shard2
fusion-lw-solr-search-0.fusion-lw-solr-search-headless (N)
system_jobs_history
shard1
fusion-lw-solr-search-0.fusion-lw-solr-search-headless (N)
fusion-lw-solr-0.fusion-lw-solr-headless (N)
system_logs
shard1
fusion-lw-solr-analytics-0.fusion-lw-solr-analytics-headless (N)
fusion-lw-solr-search-0.fusion-lw-solr-search-headless (N)
system_monitor
shard1
fusion-lw-solr-0.fusion-lw-solr-headless (N)
Leader
Active
Down
Inactive


system_autocomplete
shard1
fusion-lw-solr-search-0.fusion-lw-solr-search-headless (N)
shard2
fusion-lw-solr-0.fusion-lw-solr-headless (N)
system_blobs
shard1
fusion-lw-solr-0.fusion-lw-solr-headless (N)
fusion-lw-solr-analytics-0.fusion-lw-solr-analytics-headless (N)
system_history
shard1
fusion-lw-solr-analytics-0.fusion-lw-solr-analytics-headless (N)
shard2
fusion-lw-solr-search-0.fusion-lw-solr-search-headless (N)
system_jobs_history
shard1
fusion-lw-solr-search-0.fusion-lw-solr-search-headless (N)
fusion-lw-solr-0.fusion-lw-solr-headless (N)
system_logs
shard1
fusion-lw-solr-analytics-0.fusion-lw-solr-analytics-headless (N)
fusion-lw-solr-search-0.fusion-lw-solr-search-headless (N)
system_monitor
shard1
fusion-lw-solr-0.fusion-lw-solr-headless (N)
fusion-lw-solr-analytics-0.fusion-lw-solr-analytics-headless (N)
Leader
Active
Recovering
Down
Recovery Failed
Inactive
Replica Type: N(NRT),T(TLOG), P(PULL)
Filter by: Status T:status - Any - Show 30 per page.

Configure Neural Hybrid Search

How-to

> Compatible versions: 5.9
>
> Works with: *Managed Fusion*

Neural Hybrid Search combines lexical-semantic search with semantic vector search.

To use semantic vector search in Managed Fusion, you need to configure Neural Hybrid Search. Then you can choose the balance between lexical and semantic vector search that works best for your use case.

Before you begin, see *Neural Hybrid Search* for conceptual information that can help you understand how to configure this feature.

Important

This feature is currently only available to clients who have contracted with Lucidworks for features related to Neural Hybrid Search and Lucidworks AI.

Note

This feature is only available in Managed Fusion 5.9.x for versions 5.9.6+.

Configure vector search

This section explains how to configure vector search using Lucidworks AI, but you can also *configure it using Ray or Seldon*.

Before you set up the Lucidworks AI index and query stages, make sure you have set up your *Lucidworks AI Gateway integration*.

Configure the LWAI Vectorize Field index stage

To vectorize the index pipeline fields:

1. Sign in to Managed Fusion and click **Indexing > Index Pipelines**.
2. Click the pipeline you want to use.
3. Click **Add a new pipeline stage**.
4. In the AI section, click **LWAI Vectorize Field**.
5. In the **Label** field, enter a unique identifier for this stage.
6. In the **Condition** field, enter a script that results in true or false, which determines if the stage should process.
7. In the **Account Name** field, select the Lucidworks AI API account name defined in *Lucidworks AI Gateway*. If you do not see your account name or you are unsure which one to select, contact the Managed Fusion team at Lucidworks.
8. In the **Model** field, select the Lucidworks AI model to use for encoding.
   If you do not see your model name or you are unsure which one to select, contact the Managed Fusion team at Lucidworks.
   For more information about models, see:
   - *Pre-trained embedding models*
   - *Custom embedding model training*. To use a custom model, you must obtain the deployment ID from the deployments screen.
9. In the **Source** field, enter the name of the string field where the value should be submitted to the model for encoding. If the field is blank or does not exist, this stage is not processed. Template expressions are supported.
10. In the **Destination** field, enter the name of the field where the vector value from the model response is saved.
    If a value is entered in this field, the following information is added to the document:
    - `{Destination Field}` is the vector field.
    - `{Destination Field}_b` is the boolean value if the vector has been indexed.
11. In the **Use Case Configuration** section, click the **+** sign to enter the parameter name and value to send to Lucidworks AI. The `useCaseConfig` parameter that is common to embedding use cases is

`dataType`, but each use case may have other parameters. The value for the query stage is `query`.

12. Optionally, you can use the **Model Configuration** section for any additional parameters you want to send to Lucidworks AI. Several `modelConfig` parameters are common to generative AI use cases. For more information, see *Prediction API*.
13. Select the **Fail on Error** checkbox to generate an exception if an error occurs while generating a prediction for a document.
14. Click **Save**.
15. Index data using the new pipeline. Verify the vector field is indexed by confirming the field is present in documents. For reference information, see *LWAI Vectorize Field*.

Configure the LWAI Vectorize query stage

To vectorize the query in the query pipeline:

1. Sign in to Managed Fusion and click **Querying > Query Pipelines**.
2. Select the pipeline you want to use.
3. Click **Add a new pipeline stage**.
4. Click **LWAI Vectorize Query**.
5. In the **Label** field, enter a unique identifier for this stage.
6. In the **Condition** field, enter a script that results in true or false, which determines if the stage should process.
7. Select **Asynchronous Execution Config** if you want to run this stage asynchronously. If this field is enabled, complete the following fields:
    a. Select **Enable Async Execution**. Fusion automatically assigns an **Async ID** value to this stage. Change this to a more memorable string that describes the asynchronous stages you are merging, such as `signals` or `access_control`.
    b. Copy the **Async ID** value.

> **Note**
>
> For detailed information, see *Enable asynchronous query pipeline processing* and *Asynchronous query pipeline processing*.

8. In the **Account Name** field, select the name of the Lucidworks AI account.
   If you do not see your account name or you are unsure which one to select, contact the Managed Fusion team at Lucidworks.
9. In the **Model** field, select the Lucidworks AI model to use for encoding.
   If you do not see any model names and you are a non-admin Fusion user, verify with a Fusion administrator that your user account has these permissions: `PUT,POST,GET:/LWAI-ACCOUNT-NAME/**`
   For more information about models, see:
    - *Pre-trained embedding models*
    - *Custom embedding model training*
10. In the **Query Input** field, enter the location from which the query is retrieved.
11. In the **Output context variable** field, enter the name of the variable where the vector value from the response is saved.
12. In the **Use Case Configuration** section, click the **+** sign to enter the parameter name and value to send to Lucidworks AI. The `useCaseConfig` parameter that is common to embedding use cases is `dataType`, but each use case may have other parameters. The value for the query stage is `query`.
13. Optionally, you can use the **Model Configuration** section for any additional parameters you want to send to Lucidworks AI. Several `modelConfig` parameters are common to generative AI use cases. For more information, see *Prediction API*.
14. Select the **Fail on Error** checkbox to generate an exception if an error occurs during this stage.
15. Click **Save**.

Note

The **Top K** setting is set to 100 by default. We recommend leaving this as 100 or setting it to 200. This query stage must be placed *before* the ***Solr Query stage***. For more information, see *Reorder Query Pipeline Stages*.

Caution

Using additional pipeline stages

Vector Search does not support all available pipeline stages. At minimum, use the **Solr Query** and **LWAI Vectorize Query** stages. Do not use the **Query Fields** stage when setting up vector search.

Modify Solr managed-schema (5.9.4 and earlier)

This step is required if you're migrating a collection from a version of Managed Fusion that does not support Neural Hybrid Search. If creating a new collection in Managed Fusion 5.9.5, you can continue to *Configure Hybrid Query stage*.

1. Go to **System** > **Solr Config** and then click **managed-schema** to edit it.
2. Comment out `<copyField dest="_text_" source="*"/>` and add `<copyField dest="text" source="*_t"/>` below it. This will concatenate and index all `*_t fields`.
3. Add the following code block to the **managed-schema** file:

```
<fieldType class="solr.DenseVectorField" hnswBeamWidth="200"
hnswMaxConnections="45" name="knn_DIM_vector"
similarityFunction="cosine" vectorDimension="DIM"/>
<dynamicField docValues="false" indexed="true"
multiValued="false" name="*_512v" required="false" stored="true"
type="knn_DIM_vector"/>
```

Note

This example uses 512 vector dimension. If your model uses a different dimension, modify the code block to match your model. For example, `_1024v`. There is no limitation on supported vector dimensions.

Configure neural hybrid queries

In Managed Fusion 5.9.10 and later, you use the Neural Hybrid Query stage to configure neural hybrid queries. In Managed Fusion 5.9.9 and earlier, you use the Hybrid Query stage.

Configure the Neural Hybrid Query stage (5.9.10 and later)

Configure the Neural Hybrid Query stage in Managed Fusion 5.9.10 and later.

1. In the same query pipeline where you configured vector search, click **Add a new pipeline stage**, then select **Neural Hybrid Query**.
2. In the **Label** field, enter a unique identifier for this stage or leave blank to use the default value.
3. In the **Condition** field, enter a script that results in true or false, which determines if the stage should process, or leave blank.
4. In the **Lexical Query Input** field, enter the location from which the lexical query is retrieved. For example, **<request.params.q>**. Template expressions are supported.
5. In the **Lexical Query Weight** field, enter the relative weight of the lexical query. For example, **0.3**. If this value is **0**, no re-ranking will be applied using the lexical query scores.
6. In the **Lexical Query Squash Factor** field, enter a value that will be used to squash the lexical query score. The squash factor controls how much difference there is between the top-scoring documents and the rest. It helps ensure that documents with slightly lower scores still have a chance to show up near the top. For this value, Lucidworks recommends entering the inverse of the lexical maximum score across all queries for the given collection.
7. In the **Vector Query Field**, enter the name of the Solr field for k-nearest neighbor (KNN) vector search.
8. In the **Vector Input** field, enter the location from which the vector is retrieved. Template expressions are supported. For example, a value of `<ctx.vector>` evaluates the context variable resulting from a previous stage, such as the *LWAI Vectorize Query* stage.
9. In the **Vector Query Weight** field, enter the relative weight of the vector query. For example, **0.7**.
10. In the **Min Return Vector Similarity** field, enter the minimum vector similarity value to qualify as a match from the Vector portion of the hybrid query.
11. In the **Min Traversal Vector Similarity** field, enter the minimum vector similarity value to use when walking through the graph during the Vector portion of the hybrid query.
12. When enabled, the **Compute Vector Similarity for Lexical-Only Matches** setting computes vector similarity scores for documents in lexical search results but not in the initial vector search results. Select the checkbox to enable this setting.
13. If you want to use pre-filtering:
    a. Uncheck **Block pre-filtering**.
    In the Javascript context (`ctx`), the `preFilterKey` object becomes available.
    b. Add a *Javascript stage after* the Neural Hybrid Query stage and use it to configure your pre-filter. The `preFilter` object adds both the top-level `fq` and `preFilter` to the parameters for the vector query. You do not need to manually add the top level `fq` in the javascript stage. See the example below:

```
var QueryRequestAndResponse =
Java.type('com.lucidworks.apollo.pipeline.query.QueryReque
stAndResponse'); if(ctx.hasProperty("preFilterKey")) { var
preFilter = ctx.getProperty("preFilterKey"); var wrapper =
QueryRequestAndResponse.create(request,response,0)
preFilter.addFilter(wrapper, 'id:* OR foo_s:bar'); }
```

14. Click **Save**.

Make sure the **Hybrid Query** stage is ordered before the **Solr Query** stage. See *Reorder Query Pipeline Stages*.

Be aware that the Neural Hybrid Query stage uses new query parsers, so if you are *not* setting up a new collection, the following must be added to `solrconfig.xml` within the `<config>` tag:

```
<!-- FUSION NOTES: These query parsers are used with Solr-based vector
search --> <queryParser name="xvecSim"
class="org.apache.solr.lwbackported.XVecSimQParserPlugin"/>
<queryParser name="neuralHybrid"
class="org.apache.solr.lw.NeuralHybridQParserPlugin"/>
```

Configure the Hybrid Query stage (5.9.9 and earlier)

If you're setting up Neural Hybrid Search in Managed Fusion 5.9.9 and earlier, use the Hybrid Query stage. If you're using Managed Fusion 5.9.10 or later, use the *Neural Hybrid Query stage*.

1. In the same query pipeline where you configured vector search, click **Add a new pipeline stage**, then select **Hybrid Query**.
2. In the **Label** field, enter a unique identifier for this stage or leave blank to use the default value.
3. In the **Condition** field, enter a script that results in true or false, which determines if the stage should process, or leave blank.
4. In the **Lexical Query Input** field, enter the location from which the lexical query is retrieved. For example, **<request.params.q>**. Template expressions are supported.
5. In the **Lexical Query Weight** field, enter the relative weight of the lexical query. For example, **0.3**. If this value is **0**, no re-ranking will be applied using the lexical query scores.
6. In the **Number of Lexical Results** field, enter the number of lexical search results to include in re-ranking. For example, **1000**. A value is **0** is ignored.
7. In the **Vector Query Field**, enter the name of the Solr field for k-nearest neighbor (KNN) vector search.
8. In the **Vector Input** field, enter the location from which the vector is retrieved. Template expressions are supported. For example, a value of `<ctx.vector>` evaluates the context variable resulting from a previous stage, such as the *LWAI Vectorize Query* stage.
9. In the **Vector Query Weight** field, enter the relative weight of the vector query. For example, **0.7**.
10. Select the **Use KNN Query** checkbox to use the **knn** query parser and configure its options. This option cannot be selected if **Use VecSim Query** checkbox is selected. In addition, **Use KNN Query** is used if neither **Use KNN Query** or **Use VecSim Query** is selected.
    a. If the **Use KNN Query checkbox** is selected, enter a value in the **Number of Vector Results** field. For example, **1000**.
11. Select the **Use VecSim Query** checkbox to use the **vecSim** query parser and configure its options. This option cannot be selected if **Use KNN Query** checkbox is selected.
    If the **Use VecSim Query** checkbox is selected, enter values in the following fields:
    - **Min Return Vector Similarity**. Enter the minimum vector similarity value to qualify as a match from the Vector portion of the hybrid query.
    - **Min Traversal Vector Similarity**. Enter the minimum vector similarity value to use when walking through the graph during the Vector portion of the hybrid query. The value must be lower than, or equal to, the value in the Min Return Vector Similarity field.
12. In the **Minimum Vector Similarity Filter**, enter the value for a minimum similarity threshold for filtering documents. This option applies to all documents, regardless of other score boosting such as rules or signals.
13. Click **Save**.

Make sure the **Hybrid Query** stage is ordered before the **Solr Query** stage. See *Reorder Query Pipeline Stages*.

Perform hybrid searches

After setting up the stages, you can perform hybrid searches via the `knn` query parser as you would with Solr. Specify the search vector and include it in the query. For example, change the `q` parameter to a `knn` query parser string.

You can also preview the results in the *Query Workbench*. Try a few different queries, and adjust the weights and parameters in the Hybrid Query stage to find the best balance between lexical and semantic vector search for your use case. You can also disable and re-enable the Neural Hybrid Query stage to compare results with and without it.

Note

`XDenseVectorField` is not supported in Managed Fusion 5.9.5. Instead, use `DenseVectorField`.

Troubleshoot inconsistent results

Neural Hybrid Search leverages Solr semantic vector search, which has known behaviors which can be inconsistent at query time. These behaviors include score fluctuations with re-querying, documents showing and disappearing on re-querying, and (when SVS is configured without Hybrid stages) completely unfindable documents. This section outlines possible reasons for inconsistent behavior and resolutions steps.

NRT replicas and HNSW graph challenges

Lucidworks recommends using PULL and TLOG replicas. These replica types copy the index of the leader replica, which results in the same HNSW graph on every replica. When querying, the HNSW approximation query will be consistent given a static index.

In contrast, NRT replicas have their own index, so they will also have their own HNWS graph. HNSW is an Approximate Nearest Neighbor (ANN) algorithm, so it will not return exactly the same results for differently constructed graphs. This means that queries performed can and will return different results per HNWS graph (# of NRT replicas in a shard) which can lead to noticeable result shifts. When using NRT replicas, the shifts can be made less noticeable by increasing the `topK` parameter. Variation will still occur, but should be lower in the documents. Another way to mitigate shifts is to use Neural Hybrid Search with a vector similarity cutoff.

For more information, refer to Solr Types of Replicas.

In the case of Neural Hybrid Search, lexical BM25 and TF-IDF score differences that can occur with NRT replicas because of index differences for deleted documents can also affect combined *Hybrid score*. If you choose to use NRT replicas, then it is possible that any lexical and semantic vectors variations can and will be made worse.

Orphaning (Disconnected Nodes)

Solr's implementation of dense vector search depends on the Lucene implementation of HNSW ANN. The Lucene implementation has a known issue where, in some collections, nodes in the HNSW graph become unreachable via graph traversal, essentially becoming disconnected or "orphaned."

IDENTIFY ORPHANING

Run the following command to identify orphaning:

```
curl -sS -u 'USERNAME:PASSWORD'
'https://EXAMPLE_COMPANY.b.lucidworks.cloud/api/solrAdmin/default/COLL
ECTION_NAME/select'\ --form-string 'fl=id,vecSim:$vecSim' \ --form-
string 'rows=1' \ --form-string 'q=(*:* -{!knn f=VECTOR_FIELD
topK=999999 v=$vec})' \ --form-string
'vecSim=vectorSimilarity(VECTOR_FIELD,$vec)' \ --form-string
'vec=COMPATIBLE_VECTOR'
```

Note

If the collection doesn't have a vector for every document, include a filter so only the documents that have vectors are included. Filter on the boolean vector, as in this example: `--form-string 'fq=VECTOR_FIELD_b:true' \`

Construct a KNN exclusion query where topK is higher than the number of vectors in your collection If the number of vectors in your collection exceeds 999,999 then increase the value to be at least equal to that value.

If any are documents returned, there are orphans, and the `ids` you see are the orphans. Proceed to *Resolving orphans*. If no documents are returned, there are likely no orphans. You can try a few varying vectors to be certain.

RESOLVING ORPHANS

To resolve orphans, do the following:

1. Increase the HNSW Solr schema parameters `hnswBeamWidth` and `hnswMaxConnections` per the *Suggested values* below.
2. Save the schema.
3. Clear the index.
4. Re-index your collection.

Suggested values

Orphaning rate	`hnswBeamWidth`	`hnswMaxConnections`
5% or less	300	64
5%-25%	500	100
25% or more	3200	512

Configure Neural Hybrid Search

How-to

> Compatible versions: 5.9
>
> Works with: *Fusion*

Neural Hybrid Search combines lexical-semantic search with semantic vector search.

To use semantic vector search in Fusion, you need to configure Neural Hybrid Search. Then you can choose the balance between lexical and semantic vector search that works best for your use case.

Before you begin, see *Neural Hybrid Search* for conceptual information that can help you understand how to configure this feature.

Important

This feature is currently only available to clients who have contracted with Lucidworks for features related to Neural Hybrid Search and Lucidworks AI.

Note

This feature is only available in Fusion 5.9.5 and later versions of Fusion 5.9.

Lucidworks recommends setting up Neural Hybrid Search with Lucidworks AI, but you can instead use *Ray or Seldon vector search*. If using Lucidworks AI, you may use the default *LWAI Neural Hybrid Search pipeline*.

Configure vector search

This section explains how to configure vector search using Lucidworks AI, but you can also *configure it using Ray or Seldon*.

Before you set up the Lucidworks AI index and query stages, make sure you have set up your *Lucidworks AI Gateway integration*.

Configure the LWAI Vectorize Field index stage

To vectorize the index pipeline fields:

1. Sign in to Fusion and click **Indexing > Index Pipelines**.
2. Click the pipeline you want to use.
3. Click **Add a new pipeline stage**.
4. In the AI section, click **LWAI Vectorize Field**.
5. In the **Label** field, enter a unique identifier for this stage.
6. In the **Condition** field, enter a script that results in true or false, which determines if the stage should process.
7. In the **Account Name** field, select the Lucidworks AI API account name defined in the Lucidworks AI Gateway service.
   If you do not see your account name, check that your *Lucidworks AI Gateway integration* is correctly configured.
8. In the **Model** field, select the Lucidworks AI model to use for encoding.
   If you do not see any models names and you are a non-admin Fusion user, check that you have these permissions: `PUT,POST,GET:/LWAI-ACCOUNT-NAME/**`
   Your Fusion account name must match the name of the account that you selected in the **Account Name** dropdown.
   For more information about models, see:
   - *Pre-trained embedding models*
   - *Custom embedding model training*. To use a custom model, you must obtain the deployment ID from the deployments screen.
9. In the **Source** field, enter the name of the string field where the value should be submitted to the model for encoding. If the field is blank or does not exist, this stage is not processed. Template expressions are supported.
10. In the **Destination** field, enter the name of the field where the vector value from the model response is saved.

If a value is entered in this field, the following information is added to the document:

- `{Destination Field}` is the vector field.
- `{Destination Field}_b` is the boolean value if the vector has been indexed.

11. In the **Use Case Configuration** section, click the **+** sign to enter the parameter name and value to send to Lucidworks AI. The `useCaseConfig` parameter that is common to embedding use cases is `dataType`, but each use case may have other parameters. The value for the query stage is `query`.
12. Optionally, you can use the **Model Configuration** section for any additional parameters you want to send to Lucidworks AI. Several `modelConfig` parameters are common to generative AI use cases. For more information, see *Prediction API*.
13. Select the **Fail on Error** checkbox to generate an exception if an error occurs while generating a prediction for a document.
14. Click **Save**.
15. Index data using the new pipeline. Verify the vector field is indexed by confirming the field is present in documents.

For reference information, see *Lucidworks AI Vectorize Field*.

Configure the LWAI Vectorize Query stage

To vectorize the query in the query pipeline:

1. Sign in to Fusion and click **Querying > Query Pipelines**.
2. Select the pipeline you want to use.
3. Click **Add a new pipeline stage**.
4. Click **LWAI Vectorize Query**.
5. In the **Label** field, enter a unique identifier for this stage.
6. In the **Condition** field, enter a script that results in true or false, which determines if the stage should process.
7. Select **Asynchronous Execution Config** if you want to run this stage asynchronously. If this field is enabled, complete the following fields:
   a. Select **Enable Async Execution**. Fusion automatically assigns an **Async ID** value to this stage. Change this to a more memorable string that describes the asynchronous stages you are merging, such as `signals` or `access_control`.
   b. Copy the **Async ID** value.

> **Note**
>
> For detailed information, see *Enable asynchronous query pipeline processing* and *Asynchronous query pipeline processing*.

8. In the **Account Name** field, select the name of the Lucidworks AI account.
9. In the **Model** field, select the Lucidworks AI model to use for encoding.
   If you do not see any model names and you are a non-admin Fusion user, verify with a Fusion administrator that your user account has these permissions: `PUT,POST,GET:/LWAI-ACCOUNT-NAME/**`
   Your Fusion account name must match the name of the account that you selected in the **Account Name** dropdown.
   For more information about models, see:
   - *Pre-trained embedding models*
   - *Custom embedding model training*
10. In the **Query Input** field, enter the location from which the query is retrieved.
11. In the **Output context variable** field, enter the name of the variable where the vector value from the response is saved.
12. In the **Use Case Configuration** section, click the **+** sign to enter the parameter name and value to send to Lucidworks AI. The `useCaseConfig` parameter that is common to embedding use cases is `dataType`, but each use case may have other parameters. The value for the query stage is `query`.
13. Optionally, you can use the **Model Configuration** section for any additional parameters you want to send to Lucidworks AI. Several `modelConfig` parameters are common to generative AI use cases. For more information, see *Prediction API*.
14. Select the **Fail on Error** checkbox to generate an exception if an error occurs during this stage.

15. Click **Save**.

Note

The **Top K** setting is set to 100 by default. We recommend leaving this as 100 or setting it to 200.

This query stage must be placed *before* the ***Solr Query stage***. For more information, see *Reorder Query Pipeline Stages*.

Caution

Using additional pipeline stages

Vector Search does not support all available pipeline stages. At minimum, use the **Solr Query** and **LWAI Vectorize Query** stages. Do not use the **Query Fields** stage when setting up vector search.

Modify Solr managed-schema (5.9.4 and earlier)

This step is required if you're migrating a collection from a version of Fusion that does not support Neural Hybrid Search. If creating a new collection in Fusion 5.9.5 and later, you can continue to *Configure Hybrid Query stage*.

1. Go to **System** > **Solr Config** and then click **managed-schema** to edit it.
2. Comment out `<copyField dest="_text_" source="*"/>` and add `<copyField dest="text" source="*_t"/>` below it. This will concatenate and index all `*_t fields`.
3. Add the following code block to the **managed-schema** file:

```
<fieldType class="solr.DenseVectorField" hnswBeamWidth="200"
hnswMaxConnections="45" name="knn_DIM_vector"
similarityFunction="cosine" vectorDimension="DIM"/>
<dynamicField docValues="false" indexed="true"
multiValued="false" name="*_512v" required="false" stored="true"
type="knn_DIM_vector"/>
```

Note

This example uses 512 vector dimension. If your model uses a different dimension, modify the code block to match your model. For example, `_1024v`. There is no limitation on supported vector dimensions.

Configure neural hybrid queries

In Fusion 5.9.10 and later, you use the Neural Hybrid Query stage to configure neural hybrid queries. In Fusion 5.9.9 and earlier, you use the Hybrid Query stage.

Configure Neural Hybrid Query stage (5.9.10 and later)

1. In the same query pipeline where you configured vector search, click **Add a new pipeline stage**, then select **Neural Hybrid Query**.
2. In the **Label** field, enter a unique identifier for this stage or leave blank to use the default value.
3. In the **Condition** field, enter a script that results in true or false, which determines if the stage should process, or leave blank.
4. In the **Lexical Query Input** field, enter the location from which the lexical query is retrieved. For example, **<request.params.q>**. Template expressions are supported.
5. In the **Lexical Query Weight** field, enter the relative weight of the lexical query. For example, **0.3**. If this value is **0**, no re-ranking will be applied using the lexical query scores.
6. In the **Lexical Query Squash Factor** field, enter a value that will be used to squash the lexical query score. The squash factor controls how much difference there is between the top-scoring documents and the rest. It helps ensure that documents with slightly lower scores still have a chance to show up near the top. For this value, Lucidworks recommends entering the inverse of the lexical maximum score across all queries for the given collection.
7. In the **Vector Query Field**, enter the name of the Solr field for k-nearest neighbor (KNN) vector search.
8. In the **Vector Input** field, enter the location from which the vector is retrieved. Template expressions are supported. For example, a value of `<ctx.vector>` evaluates the context variable resulting from a previous stage, such as the *LWAI Vectorize Query* stage.
9. In the **Vector Query Weight** field, enter the relative weight of the vector query. For example, **0.7**.
10. In the **Min Return Vector Similarity** field, enter the minimum vector similarity value to qualify as a match from the Vector portion of the hybrid query.
11. In the **Min Traversal Vector Similarity** field, enter the minimum vector similarity value to use when walking through the graph during the Vector portion of the hybrid query.
12. When enabled, the **Compute Vector Similarity for Lexical-Only Matches** setting computes vector similarity scores for documents in lexical search results but not in the initial vector search results. Select the checkbox to enable this setting.
13. If you want to use pre-filtering:
    a. Uncheck **Block pre-filtering**.
    In the Javascript context (`ctx`), the `preFilterKey` object becomes available.
    b. Add a *Javascript stage after* the Neural Hybrid Query stage and use it to configure your pre-filter. The `preFilter` object adds both the top-level `fq` and `preFilter` to the parameters for the vector query. You do not need to manually add the top level `fq` in the javascript stage. See the example below:

```
var QueryRequestAndResponse =
Java.type('com.lucidworks.apollo.pipeline.query.QueryReque
stAndResponse'); if(ctx.hasProperty("preFilterKey")) { var
preFilter = ctx.getProperty("preFilterKey"); var wrapper =
QueryRequestAndResponse.create(request,response,0)
preFilter.addFilter(wrapper, 'id:* OR foo_s:bar'); }
```

14. Click **Save**.

Make sure the **Hybrid Query** stage is ordered before the **Solr Query** stage. See *Reorder Query Pipeline Stages*.

Be aware that the Neural Hybrid Query stage uses new query parsers, so if you are *not* setting up a new collection, the following must be added to `solrconfig.xml` within the `<config>` tag:

```
<!-- FUSION NOTES: These query parsers are used with Solr-based vector
search --> <queryParser name="xvecSim"
class="org.apache.solr.lwbackported.XVecSimQParserPlugin"/>
<queryParser name="neuralHybrid"
class="org.apache.solr.lw.NeuralHybridQParserPlugin"/>
```

Configure Hybrid Query stage (5.9.9 and earlier)

If you're setting up Neural Hybrid Search in Fusion 5.9.9 and earlier, use the Hybrid Query stage. If you're using Fusion 5.9.10 or later, use the *Neural Hybrid Query stage*.

1. In the same query pipeline where you configured vector search, click **Add a new pipeline stage**, then select **Hybrid Query**.
2. In the **Label** field, enter a unique identifier for this stage or leave blank to use the default value.
3. In the **Condition** field, enter a script that results in true or false, which determines if the stage should process, or leave blank.
4. In the **Lexical Query Input** field, enter the location from which the lexical query is retrieved. For example, **<request.params.q>**. Template expressions are supported.
5. In the **Lexical Query Weight** field, enter the relative weight of the lexical query. For example, **0.3**. If this value is **0**, no re-ranking will be applied using the lexical query scores.
6. In the **Number of Lexical Results** field, enter the number of lexical search results to include in re-ranking. For example, **1000**. A value is **0** is ignored.
7. In the **Vector Query Field**, enter the name of the Solr field for k-nearest neighbor (KNN) vector search.
8. In the **Vector Input** field, enter the location from which the vector is retrieved. Template expressions are supported. For example, a value of `<ctx.vector>` evaluates the context variable resulting from a previous stage, such as the *LWAI Vectorize Query* stage.
9. In the **Vector Query Weight** field, enter the relative weight of the vector query. For example, **0.7**.
10. Select the **Use KNN Query** checkbox to use the **knn** query parser and configure its options. This option cannot be selected if **Use VecSim Query** checkbox is selected. In addition, **Use KNN Query** is used if neither **Use KNN Query** or **Use VecSim Query** is selected.
    a. If the **Use KNN Query checkbox** is selected, enter a value in the **Number of Vector Results** field. For example, **1000**.
11. Select the **Use VecSim Query** checkbox to use the **vecSim** query parser and configure its options. This option cannot be selected if **Use KNN Query** checkbox is selected.
    If the **Use VecSim Query** checkbox is selected, enter values in the following fields:
    - **Min Return Vector Similarity**. Enter the minimum vector similarity value to qualify as a match from the Vector portion of the hybrid query.
    - **Min Traversal Vector Similarity**. Enter the minimum vector similarity value to use when walking through the graph during the Vector portion of the hybrid query. The value must be lower than, or equal to, the value in the Min Return Vector Similarity field.
12. In the **Minimum Vector Similarity Filter**, enter the value for a minimum similarity threshold for filtering documents. This option applies to all documents, regardless of other score boosting such as rules or signals.
13. Click **Save**.

Make sure the **Hybrid Query** stage is ordered before the **Solr Query** stage. See *Reorder Query Pipeline Stages*.

Perform hybrid searches

After setting up the stages, you can perform hybrid searches via the `knn` query parser as you would with Solr. Specify the search vector and include it in the query. For example, change the `q` parameter to a `knn` query parser string.

You can also preview the results in the *Query Workbench*. Try a few different queries, and adjust the weights and parameters in the Hybrid Query stage to find the best balance between lexical and semantic vector search for your use case. You can also disable and re-enable the Neural Hybrid Query stage to compare results with and without it.

Note

`XDenseVectorField` is not supported in Fusion 5.9.5 and above. Instead, use `DenseVectorField`.

Troubleshoot inconsistent results

Neural Hybrid Search leverages Solr semantic vector search, which has known behaviors which can be inconsistent at query time. These behaviors include score fluctuations with re-querying, documents showing and disappearing on re-querying, and (when SVS is configured without Hybrid stages) completely unfindable documents. This section outlines possible reasons for inconsistent behavior and resolutions steps.

NRT replicas and HNSW graph challenges

Lucidworks recommends using PULL and TLOG replicas. These replica types copy the index of the leader replica, which results in the same HNSW graph on every replica. When querying, the HNSW approximation query will be consistent given a static index.

In contrast, NRT replicas have their own index, so they will also have their own HNWS graph. HNSW is an Approximate Nearest Neighbor (ANN) algorithm, so it will not return exactly the same results for differently constructed graphs. This means that queries performed can and will return different results per HNWS graph (number of NRT replicas in a shard) which can lead to noticeable result shifts. When using NRT replicas, the shifts can be made less noticeable by increasing the `topK` parameter. Variation will still occur, but it should be lower in the documents. Another way to mitigate shifts is to use Neural Hybrid Search with a vector similarity cutoff.

For more information, refer to Solr Types of Replicas.

In the case of Neural Hybrid Search, lexical BM25 & TF-IDF score differences that can occur with NRT replicas because of index differences for deleted documents, can also affect combined *Hybrid score*. If you choose to use NRT replicas then it is possible that any lexical and/or semantic vectors variations can and will be exacerbated.

Orphaning (Disconnected Nodes)

Solr's implementation of dense vector search depends on the Lucene implementation of HNSW ANN. The Lucene implementation has a known issue where, in some collections, nodes in the HNSW graph become unreachable via graph traversal, essentially becoming disconnected or "orphaned."

IDENTIFY ORPHANING

Run the following command to identify orphaning:

```
curl -sS -u 'USERNAME:PASSWORD'
'https://FUSION_HOST:FUSION_PORT/api/solrAdmin/default/COLLECTION_NAME
/select'\ --form-string 'fl=id,vecSim:$vecSim' \ --form-string
'rows=1' \ --form-string 'q=(*:* -{!knn f=VECTOR_FIELD topK=999999
v=$vec})' \ --form-string 'vecSim=vectorSimilarity(VECTOR_FIELD,$vec)'
\ --form-string 'vec=COMPATIBLE_VECTOR'
```

Note

If the collection doesn't have a vector for every document, include a filter so only the documents that have vectors are included. Filter on the boolean vector, as in this example: `--form-string 'fq=VECTOR_FIELD_b:true' \`

Construct a KNN exclusion query where topK is higher than the number of vectors in your collection If the number of vectors in your collection exceeds 999,999 then increase the value to be at least equal to that value.

If any are documents returned, there are orphans, and the `ids` you see are the orphans. Proceed to *Resolving orphans*. If no documents are returned, there are likely no orphans. You can try a few varying vectors to be certain.

RESOLVING ORPHANS

To resolve orphans, do the following:

1. Increase the HNSW Solr schema parameters `hnswBeamWidth` and `hnswMaxConnections` per the *Suggested values* below.
2. Save the schema.
3. Clear the index.
4. Re-index your collection.

Suggested values

Orphaning rate	`hnswBeamWidth`	`hnswMaxConnections`
5% or less	300	64
5%-25%	500	100
25% or more	3200	512

Configure OneDrive Authentication

How-to

> Compatible versions: 4.2, 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion Connectors*, *Fusion Server*, *Fusion*

OneDrive is a file hosting service that is part of the Microsoft Office Online services. The Fusion OneDrive connector crawls a OneDrive for Business instance and retrieves data from it for indexing within Fusion.

To authenticate the Fusion OneDrive connector with a OneDrive application, first configure OneDrive with the correct permissions, then authenticate the connector.

1. Configure OneDrive for use with the Fusion connector

Create and register a Microsoft OneDrive App for use with the Fusion connector.

1. Navigate to the O365 login page .
2. Log in using an O365 admin account, or create a new one.
3. Use an existing application or create a new one. To create a new application:
    a. From the **My Applications** page, click **Add an app**.
    b. Give your app a name and click **Create**.
4. Take note of the Application ID. You will need this later.
5. Click **Generate new password** and take note of the password.
6. Under **Platforms**, click **Add platform** and then **Web**.
7. Fill in a redirect URL to your web site, ending in port 8090. For example: `http://localhost:8090`
8. Add the following permissions to the application:
    - Delegated permissions: `Files.Read.All`, `Sites.Read.All`, `User.Read`, `Directory.Read.All (Admin Only)`, `People.Read.All (Admin Only)`, `User.Read.All (Admin Only)`
    - Application permissions: `Directory.Read.All (Admin Only)`, `Files.Read.All (Admin Only)`, `People.Read.All (Admin Only)`, `Sites.Read.All (Admin Only)`, `User.Read.All (Admin Only)`
9. Click **Save**. The application is now ready to be authorized by an O365 account administrator for use with the connector.

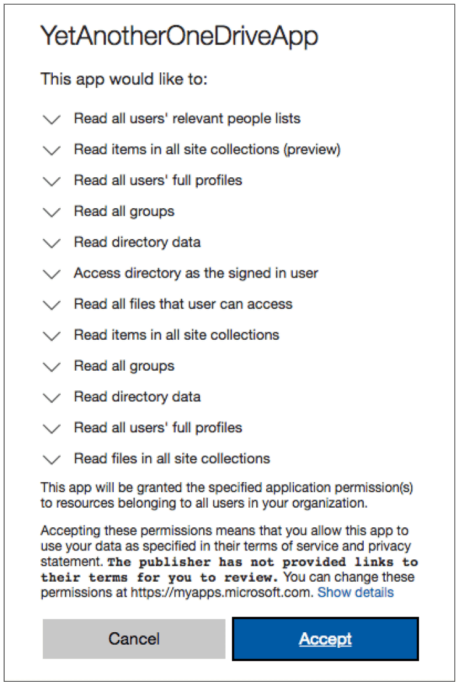
2. Authenticate the connector

To authenticate the Fusion connector:

1. Open a web browser and enter the following URL:

```
https://login.microsoftonline.com/<ACCOUNT_NAME>.onmicrosoft.com
/oauth2/authorize?client_id=
<APPLICATION_ID>&response_type=code&redirect_uri=http://<yoursit
e>&response_mode=query&resource=https://graph.microsoft.com&stat
e=12345
```

    - Replace `<ACCOUNT_NAME>` with the prefix name of your account.
    - Replace `<APPLICATION_ID>` in the `client_id` parameter with your Application ID from above.
    - Add your site URL for the uri parameter.
2. Optional: Enter ADFS issuer URI.
3. Access the URL. A list of permissions displays.



4. Click **Accept**. You can now use this application to crawl your OneDrive for Business accounts.

Configure OneDrive Authentication

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

OneDrive is a file hosting service that is part of the Microsoft Office Online services. The Fusion OneDrive connector crawls a OneDrive for Business instance and retrieves data from it for indexing within Fusion.

To authenticate the Managed Fusion OneDrive connector with a OneDrive application, first configure OneDrive with the correct permissions, then authenticate the connector.

1. Configure OneDrive for use with the Managed Fusion connector

Create and register a Microsoft OneDrive App for use with the Managed Fusion connector.

1. Navigate to the O365 login page .
2. Log in using an O365 admin account, or create a new one.
3. Use an existing application or create a new one. To create a new application:
    a. From the **My Applications** page, click **Add an app**.
    b. Give your app a name and click **Create**.
4. Take note of the Application ID. You will need this later.
5. Click **Generate new password** and take note of the password.
6. Under **Platforms**, click **Add platform** and then **Web**.
7. Fill in a redirect URL to your web site, ending in port 8090. For example: `http://localhost:8090`
8. Add the following permissions to the application:
    - Delegated permissions: `Files.Read.All`, `Sites.Read.All`, `User.Read`, `Directory.Read.All (Admin Only)`, `People.Read.All (Admin Only)`, `User.Read.All (Admin Only)`
    - Application permissions: `Directory.Read.All (Admin Only)`, `Files.Read.All (Admin Only)`, `People.Read.All (Admin Only)`, `Sites.Read.All (Admin Only)`, `User.Read.All (Admin Only)`
9. Click **Save**. The application is now ready to be authorized by an O365 account administrator for use with the connector.

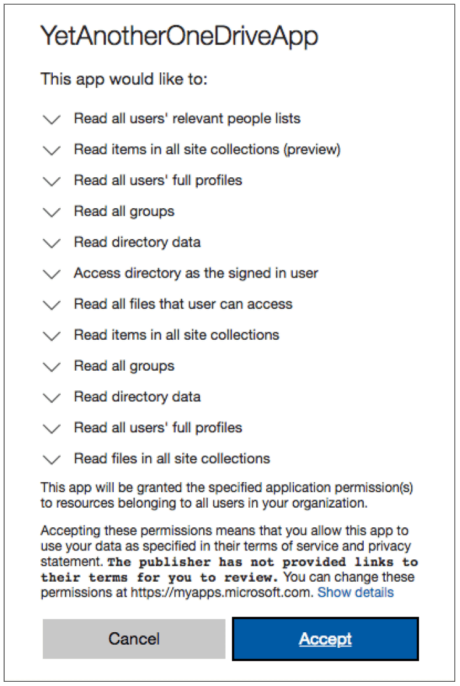
2. Authenticate the connector

To authenticate the Managed Fusion connector:

1. Open a web browser and enter the following URL:

```
https://login.microsoftonline.com/<ACCOUNT_NAME>.onmicrosoft.com
/oauth2/authorize?client_id=
<APPLICATION_ID>&response_type=code&redirect_uri=http://<yoursit
e>&response_mode=query&resource=https://graph.microsoft.com&stat
e=12345
```

    - Replace `<ACCOUNT_NAME>` with the prefix name of your account.
    - Replace `<APPLICATION_ID>` in the `client_id` parameter with your Application ID from above.
    - Add your site URL for the uri parameter.
2. Optional: Enter ADFS issuer URI.
3. Access the URL. A list of permissions displays.



4. Click **Accept**. You can now use this application to crawl your OneDrive for Business accounts.

Configure OpenID Connect authentication

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

OpenID Connect is an identity authorization layer which supplements the OAuth 2.0 protocol.

Important

Managed Fusion's OpenID Connect security realm has been tested with Google and Okta.

Configuration

Use the *Realms API* to configure this realm type:

```
curl -u USERNAME:PASSWORD -H 'content-type:application/json' -X POST
http://<managed-fusion-url>:6764/api/realm-configs -d @./realm-
config.json
```

Below is a sample configuration:

```
{ "realmType": "oidc", "name": "{your_oidcName}", "enabled": true,
"roleNames": [ "admin" ], "config": { "autoCreateUsers": true,
"groups": { "roleMapping": [ [ "role_user", "admin" ] ] }, "code": {
"clientSecret": "{your_clientSecret}", "redirectUri": "
{your_redirectUri}", "authorizationUri": "{your_authorizationUri}",
"tokenUri": "{your_tokenUri}" }, "clientId": "{your_clientId}",
"jwkSetUri": "{your_jwkSetUri}", "userIdAttribute": "email", "scope":
[ "openid", "email", "profile" ] } },
```

Required fields

Field	Description	Example
`name`	Name of the OIDC realm.	`oidc`
`clientSecret`	A secret value shared between the application and the authentication server.	N/A
`redirectUri`	The URI that displays after the user signs in.	`https://EXAMPLE_COMPANY.b.lucidworks.cloud:6764/admin`
`authorizationUri`	The authorization server URI.	`https://${yourOktaDomain}/oauth2/default/v1/authorize`
`tokenUri`	The URI where the access token is obtained.	`https://${yourOktaDomain}/oauth2/default/v1/token`
`clientId`	A unique value which identifies the client.	N/A
`jwkSetUri`	The URL of the authorization server's JSON Web Key Set (JWKS).	`https://${yourOktaDomain}/oauth2/default/v1/keys`

Google authentication

For authenticating with Google, use Google's OpenID Configuration to retrieve the required values for `authorizationUri`, `tokenUri`, `jwkSetUri`, and `issuer`.

Okta authentication

OpenID Connect authentication with Okta involves mapping Okta groups to Managed Fusion roles. The Okta group information can be retrieved from Okta's admin view:

1. Navigate to **API** > **Authorization Server**.
2. Select the server to configure for mapping.
3. In the Scope menu, add the authentication groups.
4. In the Claims menu, add new claim groups with ID token and set regexp to `.*`, which exposes all groups.

Configure OpenID Connect Authentication

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion Connectors*, *Fusion*

Configuration

Use the *Realms API* to configure this realm type:

```
curl -u USERNAME:PASSWORD -H 'content-type:application/json' -X POST
http://<fusion-url>:6764/api/realm-configs -d @./realm-config.json
```

Below is a sample configuration:

```
{ "realmType": "oidc", "name": "{your_oidcName}", "enabled": true,
"roleNames": [ "admin" ], "config": { "autoCreateUsers": true,
"groups": { "roleMapping": [ [ "role_user", "admin" ] ] }, "code": {
"clientSecret": "{your_clientSecret}", "redirectUri": "
{your_redirectUri}", "authorizationUri": "{your_authorizationUri}",
"tokenUri": "{your_tokenUri}" }, "clientId": "{your_clientId}",
"jwkSetUri": "{your_jwkSetUri}", "userIdAttribute": "email", "scope":
[ "openid", "email", "profile" ] } },
```

Required fields

Field	Description	Example
`name`	Name of the OIDC realm.	`oidc` .
`clientSecret`	A secret value shared between the application and the authentication server.	N/A
`redirectUri`	The URI to which the user will be redirected to after logging in.	`http://{fusion-url}:6764/admin`
`authorizationUri`	The authorization server URI.	`https://${yourOktaDomain}/oauth2/default/v1/authorize`
`tokenUri`	The URI to get access token from.	`https://${yourOktaDomain}/oauth2/default/v1/token`
`clientId`	A unique value which identifies the client.	N/A
`jwkSetUri`	The URL of the authorization server's JSON Web Key Set (JWKS).	`https://${yourOktaDomain}/oauth2/default/v1/keys`

Google authentication

For authenticating with Google, use Google's OpenID Configuration to retrieve the required values for `authorizationUri` , `tokenUri` , `jwkSetUri` , and `issuer` .

Okta authentication

OpenID Connect authentication with Okta involves mapping Okta groups to Fusion roles. The Okta group information can be retrieved from Okta's admin view:

1. Navigate to **API** > **Authorization Server**
2. Select the server you will configure for mapping
3. In the **Scope** menu, add the authentication groups
4. In the **Claims** menu, add new claim groups with ID token and set regexp to `.*`, which will expose all groups

Configure Pod Affinity

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion*

Affinity rules govern how Kubernetes schedules pods for Fusion components across the cluster. All components have the same affinity setup, which follows this logic:

- When scheduling, prefer to put a pod on a node in an availability zone that doesn't have a running instance of this component.
- Require that pods are deployed on a host that doesn't have a running instance of the component that is being scheduled.

With this logic, the loss of a host can only bring down one component at most. However, the cluster must be at least as large as the number of replicas in the largest deployment.

To run a large number of a component, consider relaxing the "required" policy by changing it to a "preferred" policy on the hostname for the `kubernetes.io/hostname` policies:

Before	`requiredDuringSchedulingIgnoredDuringExecution:`
After	`preferredDuringSchedulingIgnoredDuringExecution:`

If you used the `--with-affinity-rules` option when running the `./customize_fusion_values.sh` script, the pod affinity rules are configured for your cluster. Alternatively, copy `affinity.yaml`, and rename it using the following naming convention: `<provider>_<cluster>_<release>_fusion_affinity.yaml`.

To implement the file, append the following to your *upgrade script*:

```
MY_VALUES="${MY_VALUES} --values gke_search_f5_fusion_affinity.yaml"
```

Configure Ray/Seldon vector search

How-to

> Compatible versions: 5.9
>
> Works with: *Managed Fusion*

You can use Seldon or Ray models to vectorize text for *Neural Hybrid Search*.

> **Note**
>
> This feature is only available in Managed Fusion 5.9.x for versions 5.9.6+.

Prerequisites

Seldon

To vectorize text with Seldon, you'll first need to *develop and deploy a machine learning model*.

Ray

> **Note**
>
> This feature is only available in Managed Fusion 5.9.12 and later versions of Managed Fusion 5.9.

To vectorize text with Ray, you first need to *develop and deploy a machine learning model with Ray*.

Configure index pipeline

1. Sign into Managed Fusion, go to **Indexing** > **Index Pipelines**, then select an existing pipeline or create a new one.
2. Click **Add a new pipeline stage**, then select **Ray/Seldon Vectorize Field**. In Managed Fusion 5.9.11 and earlier, this stage is called **Seldon Vectorize Field**.
3. Fill in the required fields:
    a. Enter a **Model ID**. This is the name of the model you developed and deployed.
    b. Enter the **Model Input Field**. For example, `text`.
    c. Enter the **Model Output Vector Field**. For example, `vector`.
    d. Enter the **Source Field**. For example, `body_t`.
    e. Enter the **Destination Field**. For example, `body_512_v`.
4. Click **Save**.
5. Make sure the **Ray/Seldon Vectorize Field** stage is ordered before the **Solr Indexer** stage.

Index data using the new pipeline. Confirm that your vector field is indexed by checking to see that the field is present in documents.

Configure query pipeline

1. Go to **Querying** > **Query Pipelines**, then select an existing pipeline.
2. Click **Add a new pipeline stage**, then select **Ray/Seldon Vectorize Query**. In Managed Fusion 5.9.11 and earlier, this stage is called **Seldon Vectorize Query**.
3. Fill in the required fields, making sure to search against the field into which you indexed the vectors:
    a. Enter a **Model ID**. This is the name of the model you developed and deployed.
    b. Enter the **Query Input**.
    c. Enter the **Model Input Field**. For example, `text`.
    d. Enter the **Model Output Vector Field**. For example, `vector`.
    e. Enter the **Vector Context Key**. For example, `vector`.
4. Click **Save**.
5. Make sure the **Ray/Seldon Vectorize Query** stage is ordered before the **Solr Query** stage. See *Reorder Query Pipeline Stages*.

Perform vector searches

After setting up the stages, you can perform vector searches via the `knn` query parser as you would with Solr. Specify the search vector and include it in the query. For example, change the `q` parameter to a `knn` query parser string.

The **Ray/Seldon Vectorize Query** stage will encode user queries using the specified model and modify the `q` parameter to use the `knn` query parser, turning the query into a vector search.

Configure Ray/Seldon vector search

How-to

> Compatible versions: 5.9
>
> Works with: *Fusion*

You can use Seldon or Ray models to vectorize text for *Neural Hybrid Search*.

> **Note**
>
> This feature is only available in Fusion 5.9.5 and later versions of Fusion 5.9.

Prerequisites

Seldon

To vectorize text with Seldon, you first need to *develop and deploy a machine learning model with Seldon*.

Ray

> **Note**
>
> This feature is only available in Fusion 5.9.x for versions 5.9.12 and later.

To vectorize text with Ray, you first need to *develop and deploy a machine learning model with Ray*.

Configure index pipeline

1. Sign into Fusion, go to **Indexing** > **Index Pipelines**, then select an existing pipeline or create a new one.
2. Click **Add a new pipeline stage**, then select **Ray/Seldon Vectorize Field**. In Fusion 5.9.11 and earlier, this stage is called **Seldon Vectorize Field**.
3. Fill in the required fields:
    a. Enter a **Model ID**. This is the name of the model you developed and deployed.
    b. Enter the **Model Input Field**. For example, `text`.
    c. Enter the **Model Output Vector Field**. For example, `vector`.
    d. Enter the **Source Field**. For example, `body_t`.
    e. Enter the **Destination Field**. For example, `body_512_v`.
4. Click **Save**.
5. Make sure the **Seldon Vectorize Field** stage is ordered before the **Solr Indexer** stage.

Index data using the new pipeline. Confirm that your vector field is indexed by checking to see that the field is present in documents.

Configure query pipeline

1. Go to **Querying** > **Query Pipelines**, then select an existing pipeline.
2. Click **Add a new pipeline stage**, then select **Ray/Seldon Vectorize Query**. In Fusion 5.9.11 and earlier, this stage is called **Seldon Vectorize Field**.
3. Fill in the required fields, making sure to search against the field into which you indexed the vectors:
    a. Enter a **Model ID**. This is the name of the model you developed and deployed].
    b. Enter the **Query Input**.
    c. Enter the **Model Input Field**. For example, `text`.
    d. Enter the **Model Output Vector Field**. For example, `vector`.
    e. Enter the **Vector Context Key**. For example, `vector`.
4. Click **Save**.
5. Make sure the **Ray/Seldon Vectorize Query** stage is ordered before the **Solr Query** stage. See *Reorder Query Pipeline Stages*.

Perform vector searches

After setting up the stages, you can perform vector searches via the `knn` query parser as you would with Solr. Specify the search vector and include it in the query. For example, change the `q` parameter to a `knn`

query parser string.

The **Ray/Seldon Vectorize Query** stage will encode user queries using the specified model and modify the `q` parameter to use the `knn` query parser, turning the query into a vector search.

Configure Remote V2 Connectors

How-to

> Compatible versions: 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion Connectors*, *Managed Fusion*

If you need to index data from behind a firewall, you can configure a V2 connector to run remotely on-premises using TLS-enabled gRPC.

Note

Remote V2 Connectors are not available by default. Contact your Lucidworks representative for more information about enabling them in your Managed Fusion deployment.

Prerequisites

Before you can set up an on-prem V2 connector, you must configure the egress from your network to allow HTTP/2 communication into the Fusion cloud. You can use a *forward proxy server* to act as an intermediary between the connector and Fusion.

The following is required to run V2 connectors remotely:

- The plugin zip file and the connector-plugin-standalone JAR .
- A configured connector backend gRPC endpoint.
- Username and password of a user with a `remote-connectors` or `admin` role. This step is performed by Lucidworks.
- If the host where the remote connector is running is not configured to trust the server's TLS certificate, Lucidworks must help configure the file path of the trust certificate collection.

Note

If your version of Fusion doesn't have the `remote-connectors` role by default, Lucidworks can create one. No API or UI permissions are required for the role.

Connector compatibility

Only V2 connectors are able to run remotely on-premises.

The gRPC connector backend is not supported in Fusion environments deployed on AWS.

System requirements

The following is required for the on-prem host of the remote connector:

- (Managed Fusion 5.9.0-5.9.10) JVM version 11
- (Managed Fusion 5.9.11) JVM version 17
- Minimum of 2 CPUs
- 4GB Memory

Note that memory requirements depend on the number and size of ingested documents.

Enable backend ingress

Note

Contact Lucidworks support to complete this step.

In your `rpc-service/values.yaml` file, configure this section as needed:

```
ingress: enabled: false pathtype: "Prefix" path: "/" #host:
"ingress.example.com" ingressClassName: "nginx" # Fusion 5.9.6 only
tls: enabled: false certificateArn: "" # Enable the annotations field
to override the default annotations #annotations: ""
```

- Set `enabled` to `true` to enable the backend ingress.
- Set `pathtype` to `Prefix` or `Exact`.
- Set `path` to the path where the backend will be available.
- Set `host` to the host where the backend will be available.
- In Fusion 5.9.6 *only*, you can set `ingressClassName` to one of the following:
  - `nginx` for Nginx Ingress Controller
  - `alb` for AWS Application Load Balancer (ALB)

- Configure TLS and certificates according to your CA's procedures and policies.

Note

TLS must be enabled in order to use AWS ALB for ingress.

Connector configuration example

```
kafka-bridge: target: mynamespace-connectors-
backend.lucidworkstest.com:443 # mandatory plain-text: false #
optional, false by default. proxy-server: # optional - needed when a
forward proxy server is used to provide outbound access to the
standalone connector host: host port: some-port user: user # optional
password: password # optional trust: # optional - needed when the
client's system doesn't trust the server's certificate cert-
collection-filepath: path1 proxy: # mandatory fusion-proxy user: admin
password: password123 url: https://fusiontest.com/ # needed only when
the connector plugin requires blob store access plugin: # mandatory
path: ./fs.zip type: #optional - the suffix is added to the connector
id suffix: remote
```

Minimal example

```
kafka-bridge: target: mynamespace-connectors-
backend.lucidworkstest.com:443 proxy: user: admin password:
"password123" plugin: path: ./testplugin.zip
```

Logback XML configuration file example

```
<configuration> <appender name="CONSOLE"
class="ch.qos.logback.core.ConsoleAppender"> <encoder>
<pattern>%d{HH:mm:ss.SSS} [%thread] %-5level %logger{50} [%file:%line]
{%mdc} %msg%n</pattern> </encoder> </appender> <appender name="FILE"
class="ch.qos.logback.core.rolling.RollingFileAppender">
<file>./connector.log</file> <append>true</append> <rollingPolicy
class="ch.qos.logback.core.rolling.SizeAndTimeBasedRollingPolicy"> <!-
- rollover daily --> <fileNamePattern>./connector-%d{yyyy-MM-
dd}.%i.log.gz</fileNamePattern> <maxFileSize>50MB</maxFileSize>
<totalSizeCap>10GB</totalSizeCap> </rollingPolicy> <encoder>
<pattern>%d{HH:mm:ss.SSS} [%thread] %-5level %logger{50} [%file:%line]
{%mdc} %msg%n</pattern> </encoder> </appender> <root level="INFO">
<appender-ref ref="CONSOLE"/> <appender-ref ref="FILE"/> </root>
</configuration>
```

Run the remote connector

```
java [-Dlogging.config=[LOGBACK_XML_FILE]] \ -jar connector-plugin-
client-standalone.jar [YAML_CONFIG_FILE]
```

The `logging.config` property is optional. If not set, logging messages are sent to the console.

Test communication

You can run the connector in communication testing mode. This mode tests the communication with the backend without running the plugin, reports the result, and exits.

```
java –Dstandalone.connector.connectivity.test=true –jar connector–plugin–client–standalone.jar [YAML_CONFIG_FILE]
```

Encryption

In a deployment, communication to the connector's backend server is encrypted using TLS. You should only run this configuration without TLS in a testing scenario. To disable TLS, set `plain–text` to `true`.

Egress and proxy server configuration

One of the methods you can use to allow outbound communication from behind a firewall is a proxy server. You can configure a proxy server to allow certain communication traffic while blocking unauthorized communication. If you use a proxy server at the site where the connector is running, you must configure the following properties:

- **Host.** The hosts where the proxy server is running.
- **Port.** The port the proxy server is listening to for communication requests.
- **Credentials.** Optional proxy server user and password.

When you configure egress, it is important to disable any connection or activity timeouts because the connector uses long running gRPC calls.

Password encryption

If you use a login name and password in your configuration, run the following utility to encrypt the password:

1. Enter a user name and password in the *connector configuration YAML*.
2. Run the standalone JAR with this property:

```
–Dstandalone.connector.encrypt.password=true
```

3. Retrieve the encrypted passwords from the log that is created.
4. Replace the clear password in the configuration YAML with the encrypted password.

Connector restart (5.7 and earlier)

The connector will shut down automatically whenever the connection to the server is disrupted, to prevent it from getting into a bad state. Communication disruption can happen, for example, when the server running in the `connectors–backend` pod shuts down and is replaced by a new pod. Once the connector shuts down, connector configuration and job execution are disabled. To prevent that from happening, you should restart the connector as soon as possible.

You can use Linux scripts and utilities to restart the connector automatically, such as Monit.

Recoverable bridge (5.8 and later)

If communication to the remote connector is disrupted, the connector will try to recover communication and gRPC calls. By default, six attempts will be made to recover each gRPC call. The number of attempts can be configured with the `max–grpc–retries` bridge parameters.

Job expiration duration (5.9.5 only)

The timeout value for irresponsive backend jobs can be configured with the `job-expiration-duration-seconds` parameter. The default value is `120` seconds.

Use the remote connector

Once the connector is running, it is available in the *Datasources dropdown*. If the standalone connector terminates, it disappears from the list of available connectors. Once it is re-run, it is available again and configured connector instances will not get lost.

Enable asynchronous parsing (5.9 and later)

To separate document crawling from document parsing, *enable Tika Asynchronous Parsing* on remote V2 connectors.

Configure Remote V2 Connectors

How-to

> Compatible versions: 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion Connectors*, *Fusion*

If you need to index data from behind a firewall, you can configure a V2 connector to run remotely on-premises using TLS-enabled gRPC.

Prerequisites

Before you can set up an on-prem V2 connector, you must configure the egress from your network to allow HTTP/2 communication into the Fusion cloud. You can use a *forward proxy server* to act as an intermediary between the connector and Fusion.

The following is required to run V2 connectors remotely:

- The plugin zip file and the connector-plugin-standalone JAR .
- A configured connector backend gRPC endpoint.
- Username and password of a user with a `remote-connectors` or `admin` role.
- If the host where the remote connector is running is not configured to trust the server's TLS certificate, you must configure the file path of the trust certificate collection.

Note

If your version of Fusion doesn't have the `remote-connectors` role by default, you can create one. No API or UI permissions are required for the role.

Connector compatibility

Only V2 connectors are able to run remotely on-premises. You also need the remote connector client JAR file that matches your Fusion version. You can download the latest files at V2 Connectors Downloads .

Note

Whenever you upgrade Fusion, you must also update your remote connectors to match the new version of Fusion.

The gRPC connector backend is not supported in Fusion environments deployed on AWS.

System requirements

The following is required for the on-prem host of the remote connector:

- (Fusion 5.9.0-5.9.10) JVM version 11
- (Fusion 5.9.11) JVM version 17
- Minimum of 2 CPUs
- 4GB Memory

Note that memory requirements depend on the number and size of ingested documents.

Enable backend ingress

In your `values.yaml` file, configure this section as needed:

```
ingress: enabled: false pathtype: "Prefix" path: "/" #host:
"ingress.example.com" ingressClassName: "nginx" # Fusion 5.9.6 only
tls: enabled: false certificateArn: "" # Enable the annotations field
to override the default annotations #annotations: ""
```

- Set `enabled` to `true` to enable the backend ingress.
- Set `pathtype` to `Prefix` or `Exact`.
- Set `path` to the path where the backend will be available.
- Set `host` to the host where the backend will be available.
- In Fusion 5.9.6 *only*, you can set `ingressClassName` to one of the following:
  - `nginx` for Nginx Ingress Controller
  - `alb` for AWS Application Load Balancer (ALB)

- Configure TLS and certificates according to your CA's procedures and policies.

Note

TLS must be enabled in order to use AWS ALB for ingress.

Connector configuration example

```
kafka-bridge: target: mynamespace-connectors-
backend.lucidworkstest.com:443 # mandatory plain-text: false #
optional, false by default. proxy-server: # optional - needed when a
forward proxy server is used to provide outbound access to the
standalone connector host: host port: some-port user: user # optional
password: password # optional trust: # optional - needed when the
client's system doesn't trust the server's certificate cert-
collection-filepath: path1 proxy: # mandatory fusion-proxy user: admin
password: password123 url: https://fusiontest.com/ # needed only when
the connector plugin requires blob store access plugin: # mandatory
path: ./fs.zip type: #optional - the suffix is added to the connector
id suffix: remote
```

Minimal example

```
kafka-bridge: target: mynamespace-connectors-
backend.lucidworkstest.com:443 proxy: user: admin password:
"password123" plugin: path: ./testplugin.zip
```

Logback XML configuration file example

```
<configuration> <appender name="CONSOLE"
class="ch.qos.logback.core.ConsoleAppender"> <encoder
class="com.lucidworks.logging.logback.classic.LucidworksPatternLayoutE
ncoder"> <pattern>%d - %-5p [%t:%C{3.}@%L] - %m{nolookups}%n</pattern>
<charset>utf8</charset> </encoder> </appender> <appender name="FILE"
class="ch.qos.logback.core.rolling.RollingFileAppender">
<file>${LOGDIR:-.}/connector.log</file> <rollingPolicy
class="ch.qos.logback.core.rolling.SizeAndTimeBasedRollingPolicy"> <!-
- rollover daily --> <fileNamePattern>${LOGDIR:-.}/connector-%d{yyyy-
MM-dd}.%i.log.gz</fileNamePattern> <maxFileSize>50MB</maxFileSize>
<totalSizeCap>10GB</totalSizeCap> </rollingPolicy> <encoder
class="com.lucidworks.logging.logback.classic.LucidworksPatternLayoutE
ncoder"> <pattern>%d - %-5p [%t:%C{3.}@%L] - %m{nolookups}%n</pattern>
<charset>utf8</charset> </encoder> </appender> <root level="INFO">
<appender-ref ref="CONSOLE"/> <appender-ref ref="FILE"/> </root>
</configuration>
```

Run the remote connector

```
java [-Dlogging.config=[LOGBACK_XML_FILE]] \ -jar connector-plugin-
client-standalone.jar [YAML_CONFIG_FILE]
```

The `logging.config` property is optional. If not set, logging messages are sent to the console.

Test communication

You can run the connector in communication testing mode. This mode tests the communication with the backend without running the plugin, reports the result, and exits.

```
java -Dstandalone.connector.connectivity.test=true -jar connector-plugin-client-standalone.jar [YAML_CONFIG_FILE]
```

Encryption

In a deployment, communication to the connector's backend server is encrypted using TLS. You should only run this configuration without TLS in a testing scenario. To disable TLS, set `plain-text` to `true`.

Egress and proxy server configuration

One of the methods you can use to allow outbound communication from behind a firewall is a proxy server. You can configure a proxy server to allow certain communication traffic while blocking unauthorized communication. If you use a proxy server at the site where the connector is running, you must configure the following properties:

- **Host.** The hosts where the proxy server is running.
- **Port.** The port the proxy server is listening to for communication requests.
- **Credentials.** Optional proxy server user and password.

When you configure egress, it is important to disable any connection or activity timeouts because the connector uses long running gRPC calls.

Password encryption

If you use a login name and password in your configuration, run the following utility to encrypt the password:

1. Enter a user name and password in the *connector configuration YAML*.
2. Run the standalone JAR with this property:

```
-Dstandalone.connector.encrypt.password=true
```

3. Retrieve the encrypted passwords from the log that is created.
4. Replace the clear password in the configuration YAML with the encrypted password.

Connector restart (5.7 and earlier)

The connector will shut down automatically whenever the connection to the server is disrupted, to prevent it from getting into a bad state. Communication disruption can happen, for example, when the server running in the `connectors-backend` pod shuts down and is replaced by a new pod. Once the connector shuts down, connector configuration and job execution are disabled. To prevent that from happening, you should restart the connector as soon as possible.

You can use Linux scripts and utilities to restart the connector automatically, such as Monit.

Recoverable bridge (5.8 and later)

If communication to the remote connector is disrupted, the connector will try to recover communication and gRPC calls. By default, six attempts will be made to recover each gRPC call. The number of attempts can be configured with the `max-grpc-retries` bridge parameters.

Job expiration duration (5.9.5 only)

The timeout value for irresponsive backend jobs can be configured with the `job-expiration-duration-seconds` parameter. The default value is `120` seconds.

Use the remote connector

Once the connector is running, it is available in the *Datasources dropdown*. If the standalone connector terminates, it disappears from the list of available connectors. Once it is re-run, it is available again and configured connector instances will not get lost.

Enable asynchronous parsing (5.9 and later)

To separate document crawling from document parsing, *enable Tika Asynchronous Parsing* on remote V2 connectors.

Configure Resource Limits

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion*

Determining resource limits is a process that should take place after the initial setup of your cluster. This is especially true for proof-of-concept clusters. Once you have configured your cluster to accommodate a critical mass of data, tune the resource limits for your Fusion services.

In the case of production or production-like environments, define resource limits to help Kubernetes schedule pods correctly across the nodes in your cluster. This is important for Kubernetes clusters that host namespaces other than Fusion.

If you used the `--with-resource-limits` option when running the `./customize_fusion_values.sh` script, resource limits are already configured for your cluster. The script creates a YAML file for this purpose named `<provider>_<cluster>_<namespace>_fusion_resources.yaml`. Alternatively, you can copy `resources.yaml`, and rename it using the following naming convention: `<provider>_<cluster>_<release>_fusion_resources.yaml`.

You can refine the resource requests and limits as you test your cluster's behavior, while preparing for a production environment with Fusion.

Configure Result Persistence

How-to

> Compatible versions: 4.2
>
> Works with: *App Studio*

You can set up social comments, bookmarks, and saved searches so that they persist to a database.

When using the Appkit Social module, the Appkit application has been designed to save comments, bookmarks, and saved searches against logged in users. By default, the application uses an in memory database, the contents of which is lost on application restart.

To persist comments, bookmarks, and searches, we can set this up to work with any relational JDBC-compliant database (such as Oracle or MySQL).

Requirements

- Any relational JDBC-compliant database (such as Oracle or MySQL).
- Network access from the Appkit application to the relational database.

This database doesn't have to be empty because we prefix all of our tables with a `twigkit` identifier, but for simplicity we would prefer having our own schema since you probably won't need to join these tables with other schemas.

Database specific setup instructions

Which database is used for storing the metadata collected using the Social module is controlled via POM dependencies. By default a lightweight, in-memory Derby database is used, as per the following `pom.xml` dependency:

For version 4.2 and later:

```
<dependency> <groupId>twigkit</groupId>
<artifactId>twigkit.social.provider.db.derby.memory</artifactId>
<version>${project.parent.version}</version> </dependency>
```

- For version 4.1 and earlier:*

```
<dependency> <groupId>twigkit</groupId>
<artifactId>twigkit.social.db.derby.memory</artifactId>
<version>${project.parent.version}</version> </dependency>
```

To use a disk-based version of the same (persisting social metadata between restarts) change the above to:

For version 4.2 and later:

```
<dependency> <groupId>twigkit</groupId>
<artifactId>twigkit.social.db.provider.derby</artifactId>
<version>${project.parent.version}</version> </dependency>
```

For version 4.1 and earlier:

```
<dependency> <groupId>twigkit</groupId>
<artifactId>twigkit.social.db.derby</artifactId>
<version>${project.parent.version}</version> </dependency>
```

You can specify the Derby system directory, which is the directory that will contain the social database, by adding the following flag to your Java options:

```
-Dderby.system.home=/my/derby/directory
```

If the system directory that you specify with **`derby.system.home`** does not exist at startup, Derby creates the directory automatically.

You will need to configure a **`persistence.xml`** file with the appropriate connection strings etc. Click here for *Derby-memory* and *Derby* examples.

Other pre-configured versions are also available, including:

```
db.mysql db.oracle db.sqlserver
```

Click here for *MySQL*, *Oracle*, and *SQL Server* examples of persistence.xml files.

This file should be created at:
`src/main/resources/META-INF/persistence.xml`

ID GENERATOR MAPPINGS IN THE PERSISTENCE.XML

Regardless of which persistence.xml you use, please ensure the following property is added to the persistence.xml file:

```
<property name="hibernate.id.new_generator_mappings" value="false" />
```

This will ensure both pre-and post-Hibernate 5 IDs continue to be generated using the same mapping.

Troubleshooting database configuration

CONNECTION ISSUES

Many databases require the application to manage the connections, refreshing them and maintaining enough open connections to service the load on the database. In some cases the default connection management provided is inadequate for production setups.

If you notice bad performance, connections going stale, and associated errors (usually intermittently) the default connection pooling is probably inadequate for your environment.

To remedy this situation you can use a third-party connection pooling technology. We recommend 'C3P0', which can be configured with the following simple steps:

1. Add the dependency for Hibernate c3p0 to the **pom.xml**:

```
<!-- Hibernate c3p0 connection pooling --> <dependency>
<groupId>org.hibernate</groupId> <artifactId>hibernate-
c3p0</artifactId> <version>X.X.X.Final</version> </dependency>
```

   The version of Hibernate c3p0 you should use depends on the version of Appkit you are using. To begin with try version 4.1.7.Final (legacy) or 5.2.2.Final.

2. Add the configuration for C3P0 to the **persistence.xml**:

```
<!-- c3p0 connection pool settings --> <property
name="hibernate.connection.provider_class"
value="org.hibernate.connection.C3P0ConnectionProvider" />
<property name="hibernate.c3p0.max_size" value="100" />
<property name="hibernate.c3p0.min_size" value="0" /> <property
name="hibernate.c3p0.acquire_increment" value="1" /> <property
name="hibernate.c3p0.idle_test_period" value="300" /> <property
name="hibernate.c3p0.max_statements" value="0" /> <property
name="hibernate.c3p0.timeout" value="100" />
```

The settings above should be adequate for a standard Appkit application using the Social module, but they can be adjusted as desired.

Configure Salesforce authentication

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

Use the Salesforce REST API to extract data from a Salesforce repository via a Salesforce Connected App.

In addition to a user password, Salesforce authentication requires a security token automatically generated by Salesforce.

1. Obtain the security token by following Reset Your Security Token in Salesforce's online help.
2. Append the token to the end of the value found in the `Salesforce password` property in your Salesforce datasource configuration.
   For example, if your password is `PasswordABC`, and your security token is `0123456789`, the value of the Salesforce password property should be `PasswordABC0123456789`.

Note

You must request a new Salesforce security token if the user password is changed. If this happens, alter the value of the `Salesforce password` property to match the new password and security token.

Configure Salesforce Authentication

How-to

> Compatible versions: 4.2, 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion Connectors*, *Fusion Server*, *Fusion*

Use the Salesforce REST API to extract data from a Salesforce repository via a Salesforce Connected App.

In addition to a user password, Salesforce authentication requires a security token automatically generated by Salesforce.

1. Obtain the security token by following Reset Your Security Token in Salesforce's online help.
2. Append the token to the end of the value found in the `Salesforce password` property in your Salesforce datasource configuration.
   For example, if your password is `PasswordABC`, and your security token is `0123456789`, the value of the Salesforce password property should be `PasswordABC0123456789`.

Note

You must request a new Salesforce security token if the user password is changed. If this happens, alter the value of the `Salesforce password` property to match the new password and security token.

Configure Security Trimming for SharePoint Optimized V2

How-to

> Compatible versions: 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

You can configure the SharePoint Optimized V2 connector to use security trimming so that query results are filtered based on the roles and permissions assigned to the user.

To configure security trimming, you'll need to set up and run a SharePoint Optimized V2 datasource, an LDAP ACLs V2 datasource, and a Graph Security Trimming query stage in the same app and collection.

When a crawl is run, the SharePoint Optimized V2 and LDAP ACLs V2 datasources must index the content documents and ACL documents to the same collection.

- **ACL documents**: Users, Groups, and their Role Assignments.
- **Content documents**: The SharePoint objects with metadata and content (Sites, Lists, Items). These documents have `_lw_acl_ss` fields which determines who can see the docs when searching.

Set up the SharePoint datasource

1. Navigate to **Indexing > Datasources**.
2. *Install the datasource connector* if not already installed.
3. Click **Add** and select **SharePoint Optimized V2**.
4. Fill in all required fields.
5. Configure only one authentication method. Enable **NLTM Authentication Settings** or **SharePoint Online Authentication** and configure the fields as explained below.

NTLM Authentication

This method connects to SharePoint on-premises server instances, such as SharePoint Server 2013, 2016, and 2019. When using this authentication method, the connector will index `contentDocuments` and the following `aclDocuments` : `sharepointGroups` , `siteAdmins` , `roleDefinition` , and `roleAssignment` .

To use this authentication method, in your **SharePoint Optimized V2** datasource, select the **NTLM Authentication Settings** checkbox and configure the following fields:

- **User**
- **Password**
- **Domain**
- **Workstation**

SharePoint Online Authentication

These methods connect to SharePoint Online server instances. When using one of these methods, the connector will index `contentDocuments` and the following `aclDocuments` : `sharepointGroups` , `siteAdmins` , `roleDefinition` , `roleAssignment` and `sharepointUsers` in which `loginName` ends with `onmicrosoft.com` .

BASIC

To use this authentication method, in your **SharePoint Optimized V2** datasource, select the **SharePoint Online Authentication** checkbox and configure the following fields:

- **Sharepoint online account**
- **Password**

APP ONLY (OAUTH PROTOCOL)

To use this authentication method, in your **SharePoint Optimized V2** datasource, select the **SharePoint Online Authentication** checkbox and configure the following fields:

- **Azure AD client ID**
- **Azure AD tenant**
- **Azure AD Client Secret**
- **Azure AD login endpoint (advanced)**
- **Azure AD Refresh Token (advanced)**

APP ONLY WITH PRIVATE KEY

To use this authentication method, in your **SharePoint Optimized V2** datasource, select the **SharePoint Online Authentication** checkbox and configure the following fields:

- **Azure AD client ID**
- **Azure AD tenant**
- **Azure AD login endpoint**
- **Azure AD PKCS12 Base64 Keystore**
- **Azure AD PKCS12 Keystore Password**

Set up the LDAP datasource

1. Navigate to **Indexing > Datasources**.
2. *Install the datasource connector* if not already installed.
3. Click **Add** and select **LDAP and Azure ACLs Connector (V2)**.
4. Fill in all required fields.
5. Configure authentication methods. Enter LDAP login credentials and/or enable **Azure AD Properties** and configure the fields as explained below.

LDAP Authentication

This method connects to an LDAP AD server. When using this method, the connector will index the following `aclDocuments` : `ldapUsers` , and `ldapGroups` .

To use this authentication method, in your **LDAP and Azure ACLs Connector (V2)** datasource, configure the following fields:

- **Login User Principal**
- **Login Password**

Azure AD Authentication

This method connects to an Azure AD server. When using this method, the connector will index the following `aclDocuments` : `azureUsers` , and `azureGroups`

To use this authentication method, in your **LDAP and Azure ACLs Connector (V2)** datasource, select the **Azure AD Properties** checkbox and configure the following fields:

- **Azure AD Tenant ID**
- **Azure AD Client ID**
- **Azure AD Client Secret**

Supported authentication methods for security trimming

	LDAP AD	Azure AD
SharePoint On-Premises	NTLM Authentication and LDAP Authentication	NTLM Authentication and Azure AD Authentication
SharePoint Online	N/A	Any SharePoint Online authentication method and Azure AD Authentication

Configure ACL collection

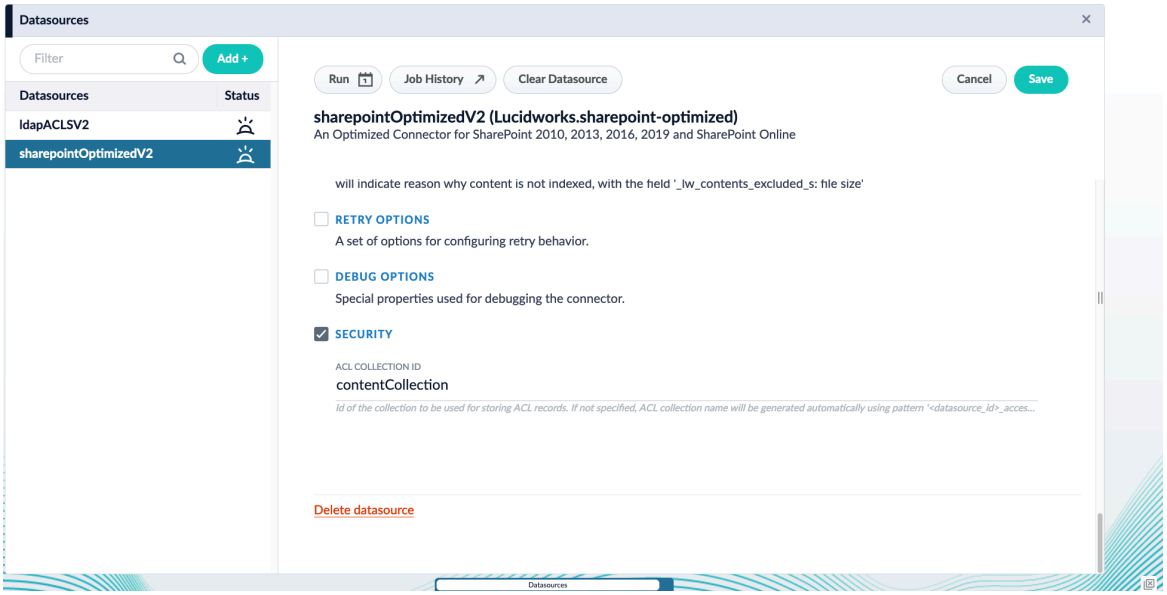
The SharePoint Optimized V2 and LDAP ACLs V2 datasources must index the content documents and ACL documents to the same collection. Ensure both datasources use the same value, `contentCollection` , for the field **ACL Collection ID**.

IF USING SHAREPOINT-OPTIMIZED AND LDAP-ACLS < V2.0.0

Update the **ACL Collection Id** in the datasource configuration.

The SharePoint-Optimized and LDAP-ACLs datasources must index their `content_documents` and `acl_documents` to the same collection. Make sure the property **Security → ACL Collection** in both datasources have the same value. In both datasources, SharePoint-Optimized and LDAP-ACLs, check the property **Security → ACL Collection Id** and make sure it points to the same content-collection.

1. Navigate to **Indexing > Datasources**.
2. Open your SharePoint Optimized V2 or LDAP ACLs V2 datasource.
3. Under **Security**, update the configuration to use `contentCollection` as the **ACL Collection ID**.



> **Tip**
>
> The **Security** checkbox must be checked for this field to appear.

4. Save the configuration.

Repeat this process for all required datasources.

IF USING SHAREPOINT-OPTIMIZED AND LDAP-ACLS >= V2.0.0

Recreate or update the datasources. If only updated, it is not possible to go back to the configuration of a previous plugin version.

By default, the LDAP-ACLs and SharePoint-Optimized V2 datasources will index the `content_documents` and `acl_documents` to the same collection.

1. Navigate to **Indexing > Datasources**.
2. Open your SharePoint Optimized V2 or LDAP ACLs V2 datasource.
3. Under **Graph Security Filtering Configuration**, select **Enable security trimming**.

Repeat this process for all required datasources.

Set up Graph Security Trimming

A *Graph Security Trimming stage* is used to pull all nested groups for a user. Then the Solr join query takes those ACL IDs found in the graph query and filters out everything that does not match one of the ACLs.

1. Navigate to **Querying** > **Query Pipelines**.
2. Open the query pipeline associated with your SharePoint Optimized V2 or LDAP ACLs V2 data.
3. Click **Add a new pipeline stage** and select **Graph Security Trimming**.
4. Configure the stage with the following settings:

Field	Value
User ID source	`query_param` or `header`
User ID key	The key that contains the User ID

Test the configuration

To confirm that security trimming works as configured, run the following test:

1. First, run the SharePoint Optimized V2 and LDAP ACLs V2 datasources.
2. Run a series of queries to test user permissions are working as intended:
    a. Run a query using a User ID key with no permissions. You should see no search results.
    b. Run a query using a User ID key that has access to some documents. You should see some search results.
    c. Run a query using a User ID key that has access to all documents. You should see all documents.

> **Note**
>
> Facet by `_lw_document_type_s: contentDocument` to see only the SharePoint docs, otherwise aclDocuments will be also shown.

Configure Security Trimming for SharePoint Optimized V2

How-to

> Compatible versions: 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion Connectors*, *Fusion Server*, *Fusion*

You can configure the SharePoint Optimized V2 connector to use security trimming so that query results are filtered based on the roles and permissions assigned to the user.

To configure security trimming, you'll need to set up and run a SharePoint Optimized V2 datasource, an LDAP ACLs V2 datasource, and a Graph Security Trimming query stage in the same app and collection.

When a crawl is run, the SharePoint Optimized V2 and LDAP ACLs V2 datasources must index the content documents and ACL documents to the same collection.

- **ACL documents**: Users, Groups, and their Role Assignments.
- **Content documents**: The SharePoint objects with metadata and content (Sites, Lists, Items). These documents have `_lw_acl_ss` fields which determines who can see the docs when searching.

Set up the SharePoint datasource

1. Navigate to **Indexing > Datasources**.
2. *Install the datasource connector* if not already installed.
3. Click **Add** and select **SharePoint Optimized V2**.
4. Fill in all required fields.
5. Configure only one authentication method. Enable **NLTM Authentication Settings** or **SharePoint Online Authentication** and configure the fields as explained below.

NTLM Authentication

This method connects to SharePoint on-premises server instances, such as SharePoint Server 2013, 2016, and 2019. When using this authentication method, the connector will index `contentDocuments` and the following `aclDocuments` : `sharepointGroups` , `siteAdmins` , `roleDefinition` , and `roleAssignment` .

To use this authentication method, in your **SharePoint Optimized V2** datasource, select the **NTLM Authentication Settings** checkbox and configure the following fields:

- **User**
- **Password**
- **Domain**
- **Workstation**

SharePoint Online Authentication

These methods connect to SharePoint Online server instances. When using one of these methods, the connector will index `contentDocuments` and the following `aclDocuments` : `sharepointGroups` , `siteAdmins` , `roleDefinition` , `roleAssignment` and `sharepointUsers` in which `loginName` ends with `onmicrosoft.com` .

BASIC

To use this authentication method, in your **SharePoint Optimized V2** datasource, select the **SharePoint Online Authentication** checkbox and configure the following fields:

- **Sharepoint online account**
- **Password**

APP ONLY (OAUTH PROTOCOL)

To use this authentication method, in your **SharePoint Optimized V2** datasource, select the **SharePoint Online Authentication** checkbox and configure the following fields:

- **Azure AD client ID**
- **Azure AD tenant**
- **Azure AD Client Secret**
- **Azure AD login endpoint (advanced)**
- **Azure AD Refresh Token (advanced)**

APP ONLY WITH PRIVATE KEY

To use this authentication method, in your **SharePoint Optimized V2** datasource, select the **SharePoint Online Authentication** checkbox and configure the following fields:

- **Azure AD client ID**
- **Azure AD tenant**
- **Azure AD login endpoint**
- **Azure AD PKCS12 Base64 Keystore**
- **Azure AD PKCS12 Keystore Password**

Set up the LDAP datasource

1. Navigate to **Indexing > Datasources**.
2. *Install the datasource connector* if not already installed.
3. Click **Add** and select **LDAP and Azure ACLs Connector (V2)**.
4. Fill in all required fields.
5. Configure authentication methods. Enter LDAP login credentials and/or enable **Azure AD Properties** and configure the fields as explained below.

LDAP Authentication

This method connects to an LDAP AD server. When using this method, the connector will index the following `aclDocuments` : `ldapUsers` , and `ldapGroups` .

To use this authentication method, in your **LDAP and Azure ACLs Connector (V2)** datasource, configure the following fields:

- **Login User Principal**
- **Login Password**

Azure AD Authentication

This method connects to an Azure AD server. When using this method, the connector will index the following `aclDocuments` : `azureUsers` , and `azureGroups`

To use this authentication method, in your **LDAP and Azure ACLs Connector (V2)** datasource, select the **Azure AD Properties** checkbox and configure the following fields:

- **Azure AD Tenant ID**
- **Azure AD Client ID**
- **Azure AD Client Secret**

Supported authentication methods for security trimming

	LDAP AD	Azure AD
SharePoint On-Premises	NTLM Authentication and LDAP Authentication	NTLM Authentication and Azure AD Authentication
SharePoint Online	N/A	Any SharePoint Online authentication method and Azure AD Authentication

Configure ACL collection

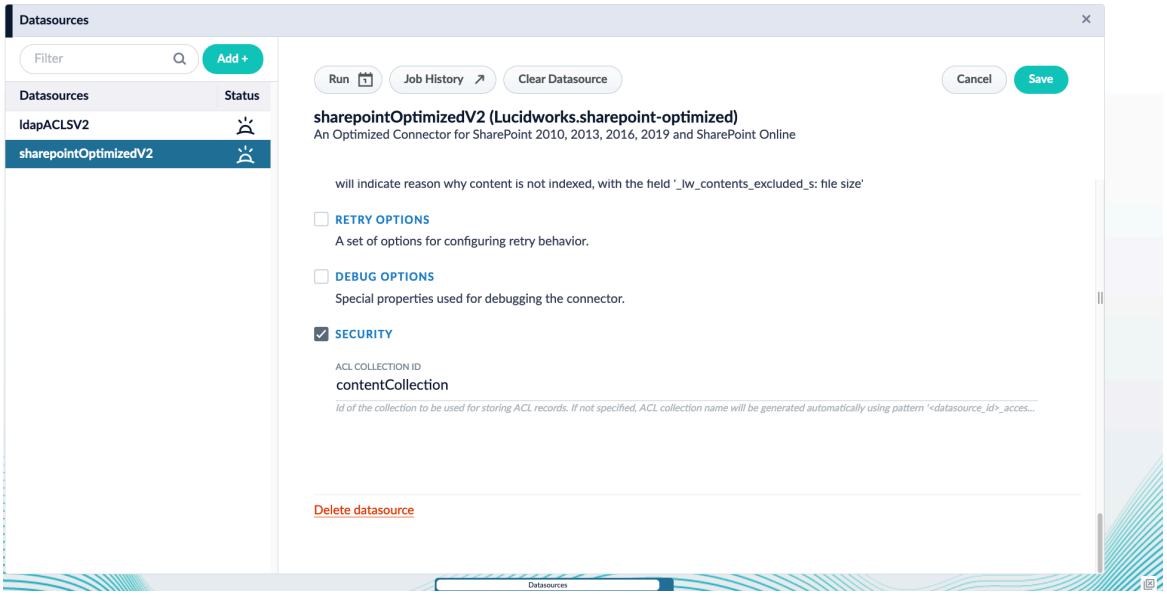
The SharePoint Optimized V2 and LDAP ACLs V2 datasources must index the content documents and ACL documents to the same collection. Ensure both datasources use the same value, `contentCollection` , for the field **ACL Collection ID**.

IF USING SHAREPOINT-OPTIMIZED AND LDAP-ACLS < V2.0.0

Update the **ACL Collection Id** in the datasource configuration.

The SharePoint-Optimized and LDAP-ACLs datasources must index their `content_documents` and `acl_documents` to the same collection. Make sure the property **Security → ACL Collection** in both datasources have the same value. In both datasources, SharePoint-Optimized and LDAP-ACLs, check the property **Security → ACL Collection Id** and make sure it points to the same content-collection.

1. Navigate to **Indexing > Datasources**.
2. Open your SharePoint Optimized V2 or LDAP ACLs V2 datasource.
3. Under **Security**, update the configuration to use `contentCollection` as the **ACL Collection ID**.



Tip

The **Security** checkbox must be checked for this field to appear.

4. Save the configuration.

Repeat this process for all required datasources.

IF USING SHAREPOINT-OPTIMIZED AND LDAP-ACLS >= V2.0.0

Recreate or update the datasources. If only updated, it is not possible to go back to the configuration of a previous plugin version.

By default, the LDAP-ACLs and SharePoint-Optimized V2 datasources will index the `content_documents` and `acl_documents` to the same collection.

1. Navigate to **Indexing > Datasources**.
2. Open your SharePoint Optimized V2 or LDAP ACLs V2 datasource.
3. Under **Graph Security Filtering Configuration**, select **Enable security trimming**.

Repeat this process for all required datasources.

Set up Graph Security Trimming

A *Graph Security Trimming stage* is used to pull all nested groups for a user. Then the Solr join query takes those ACL IDs found in the graph query and filters out everything that does not match one of the ACLs.

1. Navigate to **Querying** > **Query Pipelines**.
2. Open the query pipeline associated with your SharePoint Optimized V2 or LDAP ACLs V2 data.
3. Click **Add a new pipeline stage** and select **Graph Security Trimming**.
4. Configure the stage with the following settings:

Field	Value
User ID source	`query_param` or `header`
User ID key	The key that contains the User ID

Test the configuration

To confirm that security trimming works as configured, run the following test:

1. First, run the SharePoint Optimized V2 and LDAP ACLs V2 datasources.
2. Run a series of queries to test user permissions are working as intended:
    a. Run a query using a User ID key with no permissions. You should see no search results.
    b. Run a query using a User ID key that has access to some documents. You should see some search results.
    c. Run a query using a User ID key that has access to all documents. You should see all documents.

> **Note**
>
> Facet by `_lw_document_type_s: contentDocument` to see only the SharePoint docs, otherwise aclDocuments will be also shown.

Configure Spark Job Resource Allocation

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion*

For related topics, see *Spark Operations*.

Number of instances and cores allocated

To set the number of cores allocated for a job, add the following parameter keys and values in the Spark Settings field. This is done within the "advanced" job properties in the Fusion UI or the `sparkConfig` object, if defining a job via the Fusion API.

Parameter Key	Example Value
`spark.executor.instances`	3
`spark.kubernetes.executor.request.cores`	3
`spark.executor.cores`	6
`spark.driver.cores`	1

If `spark.kubernetes.executor.request.cores` is unset, the default configuration, Spark sets the number of CPUs for the executor pod to be the same number as `spark.executor.cores`. For exmaple, if `spark.executor.cores` is `3`, Spark allocates 3 CPUs for the executor pod and runs 3 tasks in parallel. To under-allocate the CPU for the executor pod and still run multiple tasks in parallel, set `spark.kubernetes.executor.request.cores` to a lower value than `spark.executor.cores`.

The ratio for `spark.kubernetes.executor.request.cores` to `spark.executor.cores` depends on the type of job: either CPU-bound or I/O-bound. Allocate more memory to the executor if more tasks are running in parallel on a single executor pod.

If these settings not specified, the job launches with a driver using one core and 3GB of memory plus two executors, each using one core with 1GB of memory.

Memory allocation

The amount of memory allocated to the driver and executors is controlled on a per-job basis using the `spark.executor.memory` and `spark.driver.memory` parameters in the Spark Settings section of the job definition. This is found in the Fusion UI or within the `sparkConfig` object in the JSON definition of the job.

Parameter Key	Example Value
`spark.executor.memory`	6g
`spark.driver.memory`	2g

Configure Spark Job Resource Allocation

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

For related topics, see *Spark Operations*.

NUMBER OF INSTANCES AND CORES ALLOCATED

To set the number of cores allocated for a job, add the following parameter keys and values in the Spark Settings field. This is done within the "advanced" job properties in the Managed Fusion UI or the `sparkConfig` object, if defining a job via the Managed Fusion API.

Parameter Key	Example Value
`spark.executor.instances`	3
`spark.kubernetes.executor.request.cores`	3
`spark.executor.cores`	6
`spark.driver.cores`	1

If `spark.kubernetes.executor.request.cores` is unset, the default configuration, Spark sets the number of CPUs for the executor pod to be the same number as `spark.executor.cores`. For exmaple, if `spark.executor.cores` is `3`, Spark allocates 3 CPUs for the executor pod and runs 3 tasks in parallel. To under-allocate the CPU for the executor pod and still run multiple tasks in parallel, set `spark.kubernetes.executor.request.cores` to a lower value than `spark.executor.cores`.

The ratio for `spark.kubernetes.executor.request.cores` to `spark.executor.cores` depends on the type of job: either CPU-bound or I/O-bound. Allocate more memory to the executor if more tasks are running in parallel on a single executor pod.

If these settings not specified, the job launches with a driver using one core and 3GB of memory plus two executors, each using one core with 1GB of memory.

MEMORY ALLOCATION

The amount of memory allocated to the driver and executors is controlled on a per-job basis using the `spark.executor.memory` and `spark.driver.memory` parameters in the Spark Settings section of the job definition. This is found in the Managed Fusion UI or within the `sparkConfig` object in the JSON definition of the job.

Parameter Key	Example Value
`spark.executor.memory`	6g
`spark.driver.memory`	2g

Configure Spark Jobs to Access Cloud Storage

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion*

For related topics, see *Spark Operations*.

Supported jobs

This procedure applies to Spark-based jobs:

- ALS Recommender
- Cluster Labeling
- Co-occurrence Similarity
- Collection Analysis
- Create Seldon Core Model Deployment Job
- Delete Seldon Core Model Deployment Job
- Document Clustering
- Ground Truth
- Head/Tail Analysis
- Item Similarity Recommender
- Legacy Item Recommender
- Legacy Item Similarity
- Levenshtein Spell Checking
- Logistic Regression Classifier Training
- Matrix Decomposition-Based Query-Query Similarity
- Outlier Detection
- Parallel Bulk Loader
- Parameterized SQL Aggregation
- Phrase Extraction
- Query-to-Query Session-Based Similarity
- Query-to-Query Similarity
- Random Forest Classifier Training
- Ranking Metrics
- SQL Aggregation
- SQL-Based Experiment Metric (deprecated)
- Statistically Interesting Phrases
- Synonym Detection Jobs
- Synonym and Similar Queries Detection Jobs
- Token and Phrase Spell Correction
- Word2Vec Model Training

For Argo-based jobs, see *Configure An Argo-Based Job to Access GCS* and *Configure An Argo-Based Job to Access S3*.

Amazon Web Services (AWS) and Google Cloud Storage (GCS) credentials can be configured per job or per cluster.

Configuring credentials for Spark jobs

GCS

The examples in this subsection use placeholder values. See the table below for descriptions of the placeholders:

Placeholder	Description
<key name>	Name of the Solr GCS service account key.
<key file path>	Path to the Solr GCS service account key.

1. Create a secret containing the credentials JSON file:

```
kubectl create secret generic <key name> --from-file=/<key file path>/<key name>.json
```

For more information, see Creating and managing service account keys . The topic is used to generate your organization's GOOGLE_APPLICATION_CREDENTIALS, which are needed to create an extra config map.

2. Create an extra config map in Kubernetes setting the required properties for GCP.
   a. Create a properties file with GCP properties:

```
cat gcp-launcher.properties
spark.kubernetes.driverEnv.GOOGLE_APPLICATION_CREDENTIALS = /mnt/gcp-secrets/<key name>.json
spark.kubernetes.driver.secrets.<key name> = /mnt/gcp-secrets spark.kubernetes.executor.secrets.<key name> = /mnt/gcp-secrets
spark.executorEnv.GOOGLE_APPLICATION_CREDENTIALS = /mnt/gcp-secrets/<key name>.json
spark.hadoop.google.cloud.auth.service.account.json.keyfile = /mnt/gcp-secrets/<key name>.json
```

   b. Create a config map based on the properties file:

```
kubectl create configmap gcp-launcher --from-file=gcp-launcher.properties
```

3. Add the **gcp-launcher** config map to **values.yaml** under **job-launcher**:

```
configSources: - gcp-launcher
```

AWS S3

AWS credentials can't be set with a single file. Instead, set two environment variables referring to the key and secret using the instructions below:

1. Create a secret pointing to the credentials:

```
kubectl create secret generic aws-secret --from-
literal=key='<access key>' --from-literal=secret='<secret key>'
```

2. Create an extra config map in Kubernetes setting the required properties for AWS:
   a. Create a properties file with AWS properties:

```
cat aws-launcher.properties
spark.kubernetes.driver.secretKeyRef.AWS_ACCESS_KEY_ID=aws
-secret:key
spark.kubernetes.driver.secretKeyRef.AWS_SECRET_ACCESS_KEY
=aws-secret:secret
spark.kubernetes.executor.secretKeyRef.AWS_ACCESS_KEY_ID=a
ws-secret:key
spark.kubernetes.executor.secretKeyRef.AWS_SECRET_ACCESS_K
EY=aws-secret:secret
```

   b. Create a config map based on the properties file:

```
kubectl create configmap aws-launcher --from-file=aws-
launcher.properties
```

3. Add the **`aws-launcher`** config map to **`values.yaml`** under **`job-launcher`**:

```
configSources: - aws-launcher
```

Azure Data Lake

Configuring Azure through environment variables or `configMaps` isn't possible yet. Instead, manually upload the `core-site.xml` file into the `job-launcher` pod at `/app/spark-dist/conf`. See below for an example `core-site.xml` file:

```
<property> <name>dfs.adls.oauth2.access.token.provider.type</name>
<value>ClientCredential</value> </property> <property>
<name>dfs.adls.oauth2.refresh.url</name> <value> Insert Your OAuth 2.0
Endpoint URL Value Here </value> </property> <property>
<name>dfs.adls.oauth2.client.id</name> <value> Insert Your Application
ID Here </value> </property> <property>
<name>dfs.adls.oauth2.credential</name> <value>Insert the Secret Key
Value Here </value> </property> <property> <name>fs.adl.impl</name>
<value>org.apache.hadoop.fs.adl.AdlFileSystem</value> </property>
<property> <name>fs.AbstractFileSystem.adl.impl</name>
<value>org.apache.hadoop.fs.adl.Adl</value> </property>
```

Note

At this time, only Data Lake Gen 1 is supported.

Configuring credentials per job

1. Create a Kubernetes secret with the GCP/AWS credentials.
2. Add the Spark configuration to configure the secrets for the Spark driver/executor.

GCS

The examples in this subsection use placeholder values. See the table below for descriptions of the placeholders:

Placeholder	Description
<key name>	Name of the Solr GCS service account key.
<key file path>	Path to the Solr GCS service account key.

1. Create a secret containing the credentials JSON file.

```
kubectl create secret generic <key name> --from-file=/<key file path>/<key name>.json
```

See Creating and managing service account keys for more details.

2. Toggle the **Advanced** configuration in the job UI, and add the following to the Spark configuration:

```
spark.kubernetes.driver.secrets.<key name> = /mnt/gcp-secrets
spark.kubernetes.executor.secrets.<key name> = /mnt/gcp-secrets
spark.kubernetes.driverEnv.GOOGLE_APPLICATION_CREDENTIALS = /mnt/gcp-secrets/<key name>.json
spark.executorEnv.GOOGLE_APPLICATION_CREDENTIALS = /mnt/gcp-secrets/<key name>.json
spark.hadoop.google.cloud.auth.service.account.json.keyfile = /mnt/gcp-secrets/<key name>.json
```

AWS S3

AWS credentials can't be set with a single file. Instead, set two environment variables referring to the key and secret using the instructions below:

1. Create a secret pointing to the credentials:

```
kubectl create secret generic aws-secret --from-literal=key='<access key>' --from-literal=secret='<secret key>'
```

2. Toggle the **Advanced** configuration in the job UI, and add the following to the Spark configuration:

```
spark.kubernetes.driver.secretKeyRef.AWS_ACCESS_KEY_ID=aws-
secret:key
spark.kubernetes.driver.secretKeyRef.AWS_SECRET_ACCESS_KEY=aws-
secret:secret
spark.kubernetes.executor.secretKeyRef.AWS_ACCESS_KEY_ID=aws-
secret:key
spark.kubernetes.executor.secretKeyRef.AWS_SECRET_ACCESS_KEY=aws
-secret:secret
```

Configure SSL for App Studio

How-to

> Compatible versions: 4.2
>
> Works with: *App Studio*

App Studio can be served over HTTPS using SSL encryption. You can use the default keystore for development and testing, or use your own keystore for production. Then, invoke the App Studio startup script using the SSL parameters.

SSL keystore

We include a keystore file with a default self-signed key for development and testing.

For proper security in a production environment, import your own keystore into the `keystore.jks` file, or copy it to a new file. If you copy it to a new file, use the `-Dtwigkit.keystore.file` (described below) to specify its location.

SSL parameters

To enable SSL, you specify the following parameters on the command line when invoking the startup script:

Parameter	Description	Default
`-Dtwigkit.ssl=true`	Enable SSL.	`false`
`-Dtwigkit.https.port`	Set the port.	`8765`
`-Dtwigkit.keystore.file`	The keystore path/filename, relative to the `app-studio` directory.	`keystore.jks`
`-Dtwigkit.keystore.password`	The keystore password.	`p4ssw0rd`
`-Dtwigkit.keystore.alias`	The name of the key in the keystore to be used.	`default-key`

Configure SSL for Solr embedded in Fusion

How-to

> Compatible versions: 4.2
>
> Works with: *Fusion Server*

Prerequisite

Create or determine the fully qualified path for your Fusion directory, referred to as `{FUSION_HOME}` in this document. For example, `c:/fusion/4.2/`.

Configure SSL for Solr embedded in Fusion

1. Navigate to the following directory:
   `{FUSION_HOME}/apps/jetty/solr/etc`
2. Create a self-signed certificate from an existing certificate in a production environment.
   Use the following example and enter information for your system:
   ```
   keytool -genkeypair -alias localhost -keyalg RSA -keysize 2048 -keypass secret -storepass secret -validity 365 -keystore keystore -ext SAN=DNS:localhost,IP:127.0.0.1 -dname "CN=localhost, OU=org unit, O=org, L=loc, ST=st, C=country"
   ```

   Important

   Only use a trusted Certificate Authority (CA) to generate certificates.

3. Navigate to the following directory:
   `{FUSION_HOME}/apps/jetty/solr`
4. Enter the following command to enable SSL on the Solr jetty instance:
   `java -jar {FUSION_HOME}/apps/jetty/home/start.jar --add-to-start=https`
5. Navigate to the following directory:
   `cd {FUSION_HOME}`
6. Enter the following to create the obfuscated (OBF) password:
   `java -cp ./apps/libs/jetty-util-9.3.8.v20160314.jar org.eclipse.jetty.util.security.Password secret`
7. Open the `{FUSION_HOME}/apps/jetty/solr/start.ini` file and set the following properties (using the designated OBF password):
   `jetty.sslContext.keyStorePassword jetty.sslContext.keyManagerPassword jetty.sslContext.trustStorePassword`
8. Enter the following to set the Solr SSL port:
   `jetty.ssl.port=<PORT_FOR_HTTPS>`

Disable HTTP for Solr

Warning

HTTP *must* be disabled for services such as connectors to work correctly with Solr.

Open the `fusion/4.2/apps/jetty/solr/start.d/http.ini` file and enter a `#` to comment out the `--module=http` entry:

```
#--module=http
```

Edit the fusion.properties file

1. Open the `fusion/4.2/conf/fusion.properties` file.
2. Enter the Solr keystore information in the Java Virtual Machine (JVM) for all services:
   ```
   <service>.jvmOptions=-Djavax.net.ssl.trustStore={FUSION_HOME}/apps/jetty/solr/etc/keystore -Djavax.net.ssl.trustStorePassword=secret -Djavax.net.ssl.keyStore={FUSION_HOME}/apps/jetty/solr/etc/ keystore -Djavax.net.ssl.keyStorePassword=secret
   ```
3. Uncomment the `default.address` field and enter the hostname of the server validated in the SSL certificate:
   `default.address=localhost`

4. Enter the value designated in `jetty.ssl.port=<PORT_FOR_HTTPS>` in the `solr.port` field.
5. Append the `solr.ssl=true` entry to the file.
6. Save and close the file.

Start Zookeeper and Solr

1. Enter the following command to start Zookeeper:
   `${FUSION_HOME}/bin/zookeeper start`
2. Enter the following command to start Solr:
   `${FUSION_HOME}/bin/solr start`

Configure Solr cluster for https

Enter the following command to use Zookeeper to configure the Solr cluster for `https` instead of `http`:

```
${FUSION_HOME}/apps/solr-dist/server/scripts/cloud-scripts/zkcli.sh -zkhost localhost:9983/lwfusion/${FUSION_VERSION}/solr -cmd clusterprop -name urlScheme -val https
```

Solr SSL shell script

The `solr_ssl.sh` script contains all of the steps to configure SSL for Solr embedded in Fusion. To use the script, enter your system's values and information in the file, and then execute the script:

```
# Update these variables as needed. FUSION_HOME="<your FUSION_HOME
location>" VALIDATED_ADDRESS="localhost" SOLR_PORT="8493"
FUSION_VERSION="4.2.6" keytool -genkeypair -keystore
"${FUSION_HOME}/apps/jetty/solr/etc/keystore" -dname "CN=CommonName,
OU=Unknown, O=Unknown, L=Unknown, ST=Unknown, C=Unknown" -keypass
secret -storepass secret -keyalg RSA -alias selfsigned -deststoretype
pkcs12 -ext SAN=dns:${VALIDATED_ADDRESS},ip:127.0.0.1 # enable ssl on
solr cd "${FUSION_HOME}/apps/jetty/solr" java -jar
"${FUSION_HOME}/apps/jetty/home/start.jar" --add-to-start=https
KEYSTORE_ARG="-
Djavax.net.ssl.trustStore=${FUSION_HOME}/apps/jetty/solr/etc/keystore
-Djavax.net.ssl.trustStorePassword=secret -
Djavax.net.ssl.keyStore=${FUSION_HOME}/apps/jetty/solr/etc/keystore -
Djavax.net.ssl.keyStorePassword=secret" # add the solr properties echo
"" >> ${FUSION_HOME}/apps/jetty/solr/start.ini echo "" >>
${FUSION_HOME}/apps/jetty/solr/start.ini echo
"jetty.ssl.port=${SOLR_PORT}" >>
${FUSION_HOME}/apps/jetty/solr/start.ini echo
"jetty.sslContext.keyStorePassword=OBF:1yta1t331v8w1v9q1t331ytc" >>
${FUSION_HOME}/apps/jetty/solr/start.ini echo
"jetty.sslContext.keyManagerPassword=OBF:1yta1t331v8w1v9q1t331ytc" >>
${FUSION_HOME}/apps/jetty/solr/start.ini echo
"jetty.sslContext.trustStorePassword=OBF:1yta1t331v8w1v9q1t331ytc" >>
${FUSION_HOME}/apps/jetty/solr/start.ini # set the keystore and
truststore on solr jvm sed -i -e
"s%solr.jvmOptions.*%solr.jvmOptions=-Xmx2g -Xss256k
${KEYSTORE_ARG}%g" ${FUSION_HOME}/conf/fusion.properties # set the
truststore on other jvms sed -i -e
"s%api.jvmOptions.*%api.jvmOptions=-Xmx1g -Xss256k -
Dhttp.maxConnections=1000 ${KEYSTORE_ARG}%g"
${FUSION_HOME}/conf/fusion.properties sed -i -e "s%connectors-
classic.jvmOptions.*%connectors-classic.jvmOptions=-Xmx1g -Xss256k
${KEYSTORE_ARG}%g" ${FUSION_HOME}/conf/fusion.properties sed -i -e
```

```
"s%connectors-rpc.jvmOptions.*%connectors-rpc.jvmOptions=-Xmx1g -
Xss256k -Xms512m ${KEYSTORE_ARG}%g"
${FUSION_HOME}/conf/fusion.properties sed -i -e
"s%proxy.jvmOptions.*%proxy.jvmOptions=-Xmx512m ${KEYSTORE_ARG}%g"
${FUSION_HOME}/conf/fusion.properties # Change the solr port to the
SSL port and enable SSL on the validated address sed -i -e
"s%solr.port.*%solr.port=${SOLR_PORT}%g"
${FUSION_HOME}/conf/fusion.properties echo "" >>
${FUSION_HOME}/conf/fusion.properties echo "solr.ssl=true" >>
${FUSION_HOME}/conf/fusion.properties echo "default.address =
${VALIDATED_ADDRESS}" >> ${FUSION_HOME}/conf/fusion.properties #
Disable HTTP echo "#--module=http" >
${FUSION_HOME}/apps/jetty/solr/start.d/http.ini # Start zookeeper
${FUSION_HOME}/bin/zookeeper start ${FUSION_HOME}/bin/solr start # Use
zookeeper to configure the solr cluster to use HTTPS instead of HTTP
${FUSION_HOME}/apps/solr-dist/server/scripts/cloud-scripts/zkcli.sh -
zkhost localhost:9983/lwfusion/${FUSION_VERSION}/solr -cmd clusterprop
-name urlScheme -val https # Ready to go at this point. Restart
Fusion! ${FUSION_HOME}/bin/fusion restart
```

Configure the Fields in Results

How-to

> Compatible versions: 4.2
>
> Works with: *App Studio*

The `search:result` and `search:field` tags configure which fields are displayed with each individual search result. A field can be styled as the title or description, and any additional fields can have a variety of styles.

```
<search:result> <search:field name="title_txt" styling="title"
urlfield></search:field> <search:field name="year_i"
styling="description" max-characters="150"></search:field>
<search:field name="genres_ss" styling="label-left"></search:field>
</search:result>
```

Configure the Hadoop Client

How-to

> Works with: *Fusion Connectors*

The Apache Hadoop 2 Connector is a MapReduce-enabled crawler that is compatible with Apache Hadoop v2.x.

The connector services must be able to access the Hadoop client in file `$HADOOP_HOME/bin/hadoop`, so it must either be installed on one of the nodes of the Hadoop cluster (such as the `nameNode`), or a client supported by your specific distribution must be installed on the same server as the connectors. The Hadoop client must be configured to access the Hadoop cluster so the crawler is able to access the Hadoop cluster for content processing.

Note

Instructions for setting up any of the supported Hadoop distributions is beyond the scope of this document. We recommend reading one of the many tutorials found online or one of the books on Hadoop.

This connector writes to the `hadoop.tmp.dir and the` `/tmp` directory in HDFS, so Fusion should be started by a user who has read/write permissions for both.

Permission issues

Using any flavor of Hadoop, you will need to be aware of the way Hadoop and systems based on Hadoop (such as CDH, MapR, etc.) handle permissions for services that communicate with other nodes.

Hadoop services execute under specific user credentials: a quadruplet consisting of user name, group name, numeric user id, numeric group id. Installations that follow the manual usually use user 'mapr' and group 'mapr' (or similar), with numeric ids assigned by the operating system (e.g., uid=1000, gid=20). When the system is configured to enforce user permissions (which is the default in some systems), any client that connects to Hadoop services has to use a quadruplet that exists on the server. This means that ALL values in this quadruplet must be equal between the client and the server, i.e., an account with the same user, group, uid, and gid must exist on both client and server machines.

When a client attempts to access a resource on Hadoop filesystems (or the `JobTracker`, which also uses this authentication method) it sends its credentials, which are looked up on the server, and if an exactly matching record is found then those local permissions will be used to determine read/write access. If no such account is found then the user is treated as "other" in the sense of the permission model.

This means that the crawlers for the HDFS data source should be able to crawl Hadoop or MapR filesystems without any authentication, as long as there is a read (and execute for directories) access for "other" users granted on the target resources. Authenticated users will be able to access resources owned by their equivalent account.

However, the Hadoop crawling described on this page require write access to a `/tmp` directory to use as a working directory. In many cases, this directory does not exist, or if it does, it does not have write access to "other" (not authenticated) users. Therefore users of these data sources should make sure that there is a `/tmp` directory on the target filesystem that is writable using their local user credentials, be it a recognized user, group, or "other". If a local user is recognized by the server then it is enough to create a `/tmp` directory that is owned by that user. If there is no such user, then the `/tmp` directory must be modified to have write permissions for "other" users. The working directory can be modified to be another directory that can be used for temporary working storage that has the correct permissions.

Configuration for a Kerberos Hadoop cluster

Kerberos is a system that provides authenticated access for users and services on a network. Instead of sending passwords in plaintext over the network, encrypted passwords are used to generate time-sensitive tickets which are used for authentication. Kerberos uses symmetric-key cryptography and a trusted third party called a Key Distribution Center (KDC) to authenticate users to a suite of network services. When a user authenticates to the KDC, the KDC sends a set of credentials (a ticket) specific to that session back to the user's machine.

To work with a Kerberized Hadoop cluster you must have a set of credentials. These are generated by running the "kinit" program. The datasource can be configured to run this program, in which case, the following information must be specified: the full path to the program, the Kerberos principal name, the location of a keytab file and the name of the file in which to store the ticket.

Configure the Highlight Query Pipeline Stage

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion*

The Highlight stage allows you to highlight key parts of fields by generating snippets and appending match tags to the highlighted terms. The highlight match tags can then be styled in your web search application.

How does this differ from other highlight methods?

Highlight Query Pipeline Stage

Introduced in Fusion 5.2.0, the *Query Highlight pipeline stage* allows you to configure standard options for highlighting using Solr's Unified highlighter instead of adding Solr request parameters. For instructions, see *Configure the Highlight Query Pipeline Stage*.

Additional Query Parameters stage

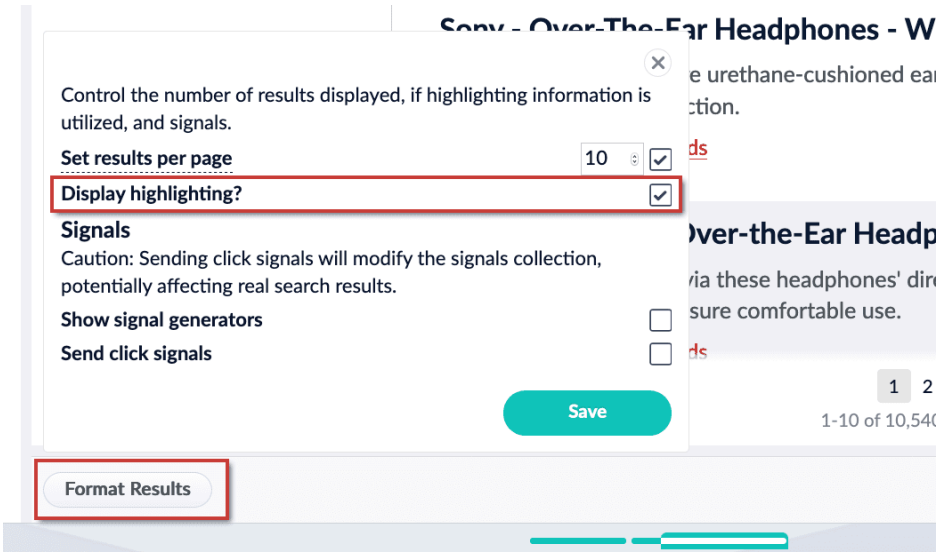
For Fusion 5.2.0 and later, the same method used in Fusion 5.1.x can be used. Configure highlighting with the *Additional Query Parameters stage*. For instructions, see *Configure Highlighting in the Query Workbench*.

Instructions

1. Navigate to **Querying** > **Query Workbench**.
2. Click **Add a Stage** and choose **Highlight** from the list.
3. Click the **+ Highlight Fields** button, and input the field name(s) to configure highlighting for. Repeat to enter multiple fields, if desired.
4. Assign a ***Snippet count*** value. With a value of `3`, Fusion will attempt to generate 3 snippets containing highlighted terms.
5. Assign a ***Snippet length*** value. With a value of `100`, Fusion will attempt to create snippets that are around 100 characters in length.
6. Assign ***Match tag*** values. These values are used to wrap the highlighted terms. By default, `<em>` and `</em>` are used. You can use other values for the match tags, such as `<span style="background-color:powderblue;">` and `</span>`, but the Fusion UI will not reflect your custom styles.
7. (optional) Set **Default highlight** to `true` if you want to generate snippets even when no highlights are found. The **Snippet count** and **Snippet length** values are still in effect.
8. Click the **Apply** button to preview highlighting.

Important

To see highlighting in the Query Workbench, turn it on by setting **Format Results** > **Display highlighting?** to `true`:



9. If you are satisfied with your changes, click the **Save** button to save the pipeline.

Configure the Highlight Query Pipeline Stage

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

The Highlight stage allows you to highlight key parts of fields by generating snippets and appending match tags to the highlighted terms. The highlight match tags can then be styled in your web search application.

How does this differ from other highlight methods?

Highlight Query Pipeline Stage

The *Query Highlight pipeline stage* allows you to configure standard options for highlighting using Solr's Unified highlighter instead of adding Solr request parameters. For instructions, see *Configure the Highlight Query Pipeline Stage*.

Additional Query Parameters stage

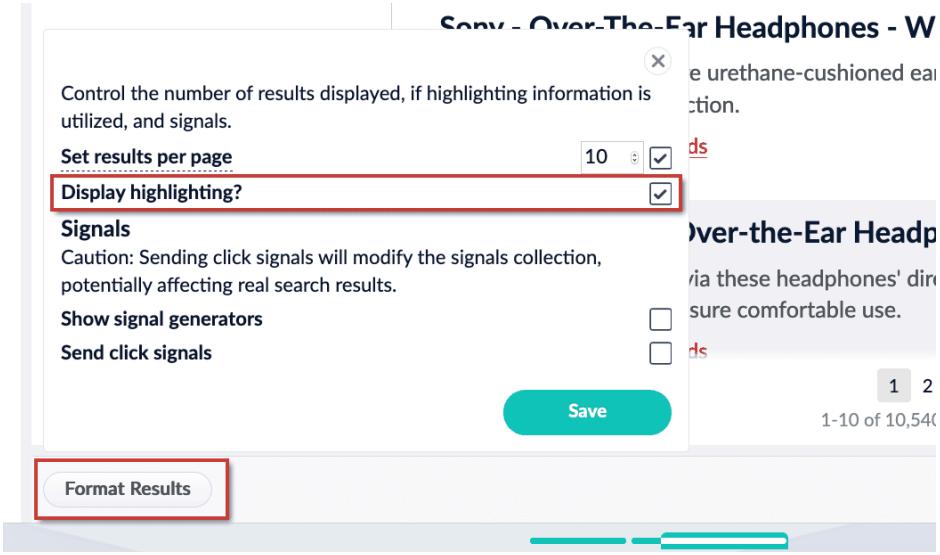
Configure highlighting with the *Additional Query Parameters stage*. For instructions, see *Configure Highlighting in the Query Workbench*.

Instructions

1. Navigate to **Querying** > **Query Workbench**.
2. Click **Add a Stage** and choose **Highlight** from the list.
3. Click the **+ Highlight Fields** button, and input the field name(s) to configure highlighting for. Repeat to enter multiple fields, if desired.
4. Assign a ***Snippet count*** value. With a value of `3`, Fusion will attempt to generate 3 snippets containing highlighted terms.
5. Assign a ***Snippet length*** value. With a value of `100`, Fusion will attempt to create snippets that are around 100 characters in length.
6. Assign ***Match tag*** values. These values are used to wrap the highlighted terms. By default, `<em>` and `</em>` are used. You can use other values for the match tags, such as `<span style="background-color:powderblue;">` and `</span>`, but the Managed Fusion UI will not reflect your custom styles.
7. (optional) Set **Default highlight** to `true` if you want to generate snippets even when no highlights are found. The **Snippet count** and **Snippet length** values are still in effect.
8. Click the **Apply** button to preview highlighting.

Important

To see highlighting in the Query Workbench, turn it on by setting **Format Results** > **Display highlighting?** to `true`:



9. If you are satisfied with your changes, click the **Save** button to save the pipeline.

Configure the LWAI Neural Hybrid Search pipeline

Lucidworks AI

How-to

> Compatible versions: 5.9
>
> Works with: *Managed Fusion*

The LWAI Neural Hybrid Search pipeline is a default pipeline that contains all the required query stages to set up *Neural Hybrid Search* using *Lucidworks AI*.

Important

This feature is currently only available to clients who have contracted with Lucidworks for features related to Neural Hybrid Search and Lucidworks AI.

Note

This feature is only available in Managed Fusion 5.9.x for versions 5.9.6+.

This pipeline uses the following stages:

- *Text Tagger*
- *Boost with Signals*
- *Query Fields*
- *LWAI Vectorize Query*
- *Hybrid Query (5.9.9 and earlier)*
- *Neural Hybrid Query (5.9.10 and later)*
- *Apply Rules*
- *Solr Query*
- *Modify Response with Rules*

Note

Milvus is deprecated. To migrate an existing Milvus collection to Solr vector, see the *Milvus to Solr migration guide*. Migrating prior to Milvus removal is important to prevent disruptions to pipeline performance.

Configure the pipeline

To add the Neural Hybrid Search (NHS) query pipeline:

1. Sign in to Managed Fusion and click **Querying > Query Pipelines**.
2. Select the default **LWAI-neural-hybrid-search-NHS** pipeline.
3. Configure the following stages included in the default pipeline.

Text Tagger

The *Text Tagger* stage queries a Solr text tagger request handler to perform spell correction, phrase boosting, and synonym expansion.

1. In the **Label** field, enter a unique identifier for this stage or leave blank to use the default value.
2. In the **Condition** field, enter a script that results in true or false, which determines if the stage should process, or leave blank.
3. In the **Tagger Collection** field, enter where the tagger request is sent. The default is the **query_rewrite** collection for the application selected. You must enter a collection with only one shard because Text Tagger does not support multi-shard collections. Template expressions are supported.
4. In the **Param to Tag** field, enter a value of **q**, which is the name of the parameter in the request containing text to tag. This field is ignored on DSL requests.
5. In the **Save Tags in Context** field, enter the tags to save in context instead of applying directly to the incoming query in this stage. This enables downstream stages to apply the tags after completing other processing. This field is ignored on DSL requests.
6. Select the following checkboxes:

- Spell Correction
- Phrase Boosting
- Synonym Expansion
- Remove Words
- Tail Rewrites

7. In the **Filter Override** field, enter your filter to override filtering for built-in tagger doc types.
8. In the **Original Term Boost for Synonyms** field, enter the boost to use for the original term during synonym expansion. For example, **2**. To disable this function, enter **-1**.
9. In the **Default Phrase Boost** field, enter the boost to use as a default for phrases that do not have a boost value set. For example, **2**. To disable this function, enter **-1**.
10. In the **Default Phrase Slop** field, enter the distance between the terms of the query while still considering it a phrase match. For example, **10**.
11. In the **Overlapping Tag Policy** field, select the default value of **longest_dominant_right** to ensure the retained tags have no overlaps. The value is the algorithm that determines which tags in an overlapping set should be retained, versus being pruned away. On DSL requests, this field is ignored so the default value of **longest_dominant_right** is always used. The available options correspond to Solr Tagger Handler overlaps: **all**, **no_sub**, or **longest_dominant_right**. Setting the value to **all** or **no_sub** allows more rewrites to potentially be applied to a query, but can increase the chance of producing undesirable rewrites.
12. In the **Additional Params to be included in the Text Tagger Request** section, enter optional values you want to include.
13. In the **Max Wait for Lookup (ms)** field, enter the number of milliseconds to wait for the call to the remote tagger collection to return. For example, **500**. To disable this function, enter **-1**.
14. In the **Skip Query Regex** field, enter the pattern that identifies queries that are skipped because they contain that pattern. For example, you may want to skip single term queries with wildcards.
15. Click **Save**.

Boost with Signals

The *Boost with Signals* stage uses aggregated signals to selectively boost items in the set of search results.

1. In the **Label** field, enter a unique identifier for this stage or leave blank to use the default value.
2. In the **Condition** field, enter a script that results in true or false, which determines if the stage should process, or leave blank.
3. Select the **Asynchronous Execution Config** check box to process this stage asynchronously.
4. In the **Number of Recommendations field**, enter the number of documents to return in the query. For example, **10**.
5. In the **Number of Signals** field, enter the number of signals to process when getting recommended items. For example, **100**.
6. In the **Aggregation Type** field, enter an applicable value. For example, **click@doc_id,filters,query**.
7. In the **Solr Field to Boost On** field, enter which Solr field to use when applying recommendation boosts. For example, **id**.
8. In the **Boost Method** field, select the boost method to use. If the **defType!=edismax** for the main query, select **query-parser**. Another example is **query-param**.
9. In the **Boost Param** field, select one of the following values:
    - **boost** to multiply scores by the boost values
    - **bq** to add optional clauses to main query
10. In the **Solr Query parameters** section, enter the following **Parameter Name:Parameter Value** entries:
    - **qf**: **query_t**
    - **pf**: **query_t^50**
    - **pf**: **query_t-3^20**
    - **pf2**: **query_t^20**
    - **pf2**: **query_t~3^10**
    - **pf3**: **query_t^10**
    - **pf3**: **query_t~3^5**
    - **boost**: **map(query({!field f=query_s v=$q}),0,0,1,20)**
    - **mm**: **50%**
    - **defType**: **edismax**
    - **sort**: **score desc, weight_d desc**
    - **fq**: **weight_d:[** TO **]**

11. In the **Rollup Field**, enter the field name to use when rolling up documents that have the same doc id. For example, **doc_id_s**.
12. In the **Rollup weight field**, enter the field name to use for signal weights. For example, **weight_d**.
13. In the **Rollup weight strategy** field, select one of the following methods to use when rolling up the weight:
    - **max**
    - **sum**
14. In the **Final Boost Weight Expression** field, enter the optional expression to compute the final boost weight using a combination of fields returned by Solr. For example, **score** and **weight_d**. Set to **weight_d** for similar behavior as older versions. Another example is **math:log(weight_d + 1) + 10 * math:log(score+1)**.
15. In the **Document Weights Context Key** field, enter the context key in which to save boosts for **docId:weight_d**.
16. In the **Query Param** field, enter the default value of **q**, which is the name of the parameter in the request containing query to boost.
17. Select the **Include Enriched Query** checkbox to:
    - Enable the stage to combine the user's original query with the output of any stages that enrich the query, such as the Text Tagger stage.
    - Expand the recall of the boost lookup. However, precision may be impacted.
    - Enable the stage to change the configured **mm** parameter to accommodate additional terms added to the boost lookup query.
18. In the **Update Policy** field, select to **replace** or **append** the boost parameter in the final query.
19. Click **Save**.

Query Fields

The *Query Fields* stage configures query parameters for a Solr search.

1. In the **Label** field, enter a unique identifier for this stage or leave blank to use the default value.
2. In the **Condition** field, enter a script that results in true or false, which determines if the stage should process, or leave blank.
3. In the **Number of Results** field, enter the number of query fields to return. For example, **10**.
4. In the **Result Offset** field, enter the value that designates the starting position of the first result from the beginning of the data being queried. Offset is zero-based, so a value of **9** counts from 0 through 9 and the first record returned is the 10th record discovered in the query.
5. In the **Results Sort Order** section, enter values for the following fields:
    - Sort Type. For example, sort on expression, field, query, or relevancy.
    - Field Name or Expression on which to sort.
    - Sort Order. Select either **asc** or **desc**.
6. In the **Search Fields** section, enter the field name and boost values to include in this stage.
7. In the **Return Fields** section, enter the field names to include in this stage.
8. Select the **Return Score** checkbox to set as **true** and determine if a score is determined for this stage.
9. In the **Minimum Should Match** field, enter the minimum string to match for this stage.
10. Select the **Grouping Options** checkbox to enter values in the following fields:
    - Grouping Field. The field name on which to group results.
    - Group Size. The number of results per group.
    - Group Sort. Enter values in the following fields:
        - Sort Type. For example, sort on expression, field, query, or relevancy.
        - Field Name or Expression on which to sort.
        - Sort Order. Select either **asc** or **desc**.
    - Group Leader Strategy. Select this checkbox to define selection criteria for the representative document from each group. Only one method may be used at a time. Defaults to **relevancy** if not specified. Values are:
        - By Field Value. If selected, include documents with the minimum or maximum value for the indicated field to be the representative document for each group.
        - By Sort. If selected, include the representative document for each group based on the order in which they are returned with the given sort criteria.

11. Click **Save**.

LWAI Vectorize Query

The *LWAI Vectorize Query* stage configures parameters to generate a vector by using a Lucidworks AI (LWAI) embedding model to encode the input to a vector representation. This stage is ignored if the input is blank or a wildcard of either an asterisk (*****) or a colon (**:**) is used.

1. In the **Label** field, enter a unique identifier for this stage.
2. In the **Condition** field, enter a script that results in true or false, which determines if the stage should process.
3. In the **Account Name** field, select the name of the Lucidworks AI account.
   If you do not see your account name or you are unsure which one to select, contact the Managed Fusion team at Lucidworks.
4. In the **Model** field, select the Lucidworks AI model to use for encoding.
   If you do not see any model names and you are a non-admin Fusion user, verify with a Fusion administrator that your user account has these permissions: `PUT,POST,GET:/LWAI-ACCOUNT-NAME/**`
   For more information, see:
   - *Pre-trained embedding models*
   - *Custom embedding model training*
5. In the **Query Input** field, enter the location from which the query is retrieved.
6. In the **Output context variable** field, enter the name of the variable where the vector value from the response is saved.
7. In the **Use Case Configuration** section, click the **+** sign to enter the parameter name and value to send to Lucidworks AI. The `useCaseConfig` parameter that is common to generative AI and embedding use cases is `dataType`, but each use case may have other parameters. The value for the query stage is `query`.
8. In the **Model Configuration** section, click the **+** sign to enter the parameter name and value to send to Lucidworks AI. Several `modelConfig` parameters are common to generative AI use cases. For more information, see *Prediction API*.
9. Select the **Fail on Error** checkbox to generate an exception if an error occurs during this stage.
10. Click **Save**.

> **Note**
>
> The **Top K** setting is set to 100 by default. We recommend leaving this as 100 or setting it to 200.

This query stage must be placed *before* the ***Solr Query stage***. For more information, see *Reorder Query Pipeline Stages*.

Hybrid Query (5.9.9 and earlier)

The *Hybrid Query* stage is a combination of semantic vector search and lexical search. This stage is ignored if the input is blank or a wildcard of either an asterisk (*****) or a colon (**:**).

In addition, this stage does not function correctly if the incoming **q** parameter is a Solr query parser string. For example, **field_t:foo** rather than a raw user query string.

The resulting query is *always* written to **<request.params.q>**.

1. In the **Label** field, enter a unique identifier for this stage or leave blank to use the default value.
2. In the **Condition** field, enter a script that results in true or false, which determines if the stage should process, or leave blank.
3. In the **Lexical Query Input** field, enter the location from which the lexical query is retrieved. For example, **<request.params.q>**. Template expressions are supported.
4. In the **Lexical Query Weight** field, enter the relative weight of the lexical query. For example, **0.3**. If this value is **0**, no re-ranking will be applied using the lexical query scores.
5. In the **Number of Lexical Results** field, enter the number of lexical search results to include in re-ranking. For example, **1000**. A value is **0** is ignored.
6. In the **Vector Query Field**, enter the name of the Solr field for k-nearest neighbor (KNN) vector search.

7. In the **Vector Input** field, enter the location from which the vector is retrieved. Template expressions are supported. For example, a value of `<ctx.vector>` evaluates the context variable resulting from a previous stage, such as the *LWAI Vectorize Query* stage.
8. In the **Vector Query Weight** field, enter the relative weight of the vector query. For example, **0.7**.
9. Select the **Use KNN Query** checkbox to use the **knn** query parser and configure its options. This option cannot be selected if **Use VecSim Query** checkbox is selected. In addition, **Use KNN Query** is used if neither **Use KNN Query** or **Use VecSim Query** is selected.
    a. If the **Use KNN Query checkbox** is selected, enter a value in the **Number of Vector Results** field. For example, **1000**.
10. Select the **Use VecSim Query** checkbox to use the **vecSim** query parser and configure its options. This option cannot be selected if **Use KNN Query** checkbox is selected.
    If the **Use VecSim Query** checkbox is selected, enter values in the following fields:
    - **Min Return Vector Similarity**. Enter the minimum vector similarity value to qualify as a match from the Vector portion of the hybrid query.
    - **Min Traversal Vector Similarity**. Enter the minimum vector similarity value to use when walking through the graph during the Vector portion of the hybrid query. The value must be lower than, or equal to, the value in the Min Return Vector Similarity field.
11. In the **Minimum Vector Similarity Filter**, enter the value for a minimum similarity threshold for filtering documents. This option applies to all documents, regardless of other score boosting such as rules or signals.
12. Click **Save**.

Neural Hybrid Query (5.9.10 and later)

The *Neural Hybrid Query* stage is a combination of semantic vector search and lexical search. This stage is ignored if the input is blank or a wildcard of either an asterisk (*****) or a colon (**:**).

In addition, this stage does not function correctly if the incoming **q** parameter is a Solr query parser string. For example, **field_t:foo** rather than a raw user query string.

The resulting query is *always* written to **<request.params.q>**.

1. In the **Label** field, enter a unique identifier for this stage or leave blank to use the default value.
2. In the **Condition** field, enter a script that results in true or false, which determines if the stage should process, or leave blank.
3. In the **Lexical Query Input** field, enter the location from which the lexical query is retrieved. For example, **<request.params.q>**. Template expressions are supported.
4. In the **Lexical Query Weight** field, enter the relative weight of the lexical query. For example, **0.3**. If this value is **0**, no re-ranking will be applied using the lexical query scores.
5. In the **Lexical Query Squash Factor** field, enter a value that will be used to squash the lexical query score. The squash factor controls how much difference there is between the top-scoring documents and the rest. It helps ensure that documents with slightly lower scores still have a chance to show up near the top. For this value, Lucidworks recommends entering the inverse of the lexical maximum score across all queries for the given collection.
6. In the **Vector Query Field**, enter the name of the Solr field for k-nearest neighbor (KNN) vector search.
7. In the **Vector Input** field, enter the location from which the vector is retrieved. Template expressions are supported. For example, a value of `<ctx.vector>` evaluates the context variable resulting from a previous stage, such as the *LWAI Vectorize Query* stage.
8. In the **Vector Query Weight** field, enter the relative weight of the vector query. For example, **0.7**.
9. In the **Min Return Vector Similarity** field, enter the minimum vector similarity value to qualify as a match from the Vector portion of the hybrid query.
10. In the **Min Traversal Vector Similarity** field, enter the minimum vector similarity value to use when walking through the graph during the Vector portion of the hybrid query.
11. When enabled, the **Compute Vector Similarity for Lexical-Only Matches** setting computes vector similarity scores for documents in lexical search results but not in the initial vector search results. Select the checkbox to enable this setting.
12. If you want to use pre-filtering:
    a. Uncheck **Block pre-filtering**.
    In the Javascript context (`ctx`), the `preFilterKey` object becomes available.
    b. Add a *Javascript stage after* the Neural Hybrid Query stage and use it to configure your pre-filter. The `preFilter` object adds both the top-level `fq` and `preFilter` to the parameters for the vector query. You do not need to manually add the top level `fq` in the javascript stage. See

the example below:

```
var QueryRequestAndResponse =
Java.type('com.lucidworks.apollo.pipeline.query.QueryReque
stAndResponse'); if(ctx.hasProperty("preFilterKey")) { var
preFilter = ctx.getProperty("preFilterKey"); var wrapper =
QueryRequestAndResponse.create(request,response,0)
preFilter.addFilter(wrapper, 'id:* OR foo_s:bar'); }
```

13. Click **Save**.

Apply Rules

The *Apply Rules* stage applies the rules configured in the collection to the query.

1. In the **Label** field, enter a unique identifier for this stage or leave blank to use the default value.
2. In the **Condition** field, enter a script that results in true or false, which determines if the stage should process, or leave blank.
3. Select the **Asynchronous Execution Config** check box to process this stage asynchronously.
4. In the **Collection field**, enter the name of the collection that contains the rules. If this field does not contain a value, the default rules collection for the selected application is used. Template expressions are supported.
5. In the **Request Handler** field, enter a value of **select**.
6. In the **HTTP Method** field, select **POST**.
7. In the **Rule Triggering Limit** field, enter the maximum number of business rules to be triggered by the query. The default rules matching limit is **100**. This configuration overwrites the **rows** parameter set in Query Parameters section.
8. In the **Query Parameters** section, enter the names and values to use when querying the rules collection. If you set the **rows** parameter here, it will be overwritten by the configuration in the Rule Triggering Limit field.
9. In the **Subquery Rewrite Pipeline id** field, enter the value to call a Managed Fusion query pipeline to modify the rule-retrieving subquery. Template expressions are supported.
10. In the **Headers** section, enter the names and values to use in this stage.
11. Select the **Use Original Query If No Rules Match** checkbox so the stage will try to match rules using the original query (un-tagged) sent into the Text Tagger stage, if available.
12. Select the **Partially Matched Filter Queries Will Trigger the Rule** checkbox so the stage will trigger filter rules as long as there is one filter query in the query parameter that matches the filter specified in the rule.
13. In the **Max Wait for Lookup (ms)**, enter the number of milliseconds to wait for the call to the remote tagger collection to return. For example, **500**. To disable this function, enter **-1**.
14. Click **Save**.

Solr Query

The *Solr Query* stage sends the search request to Solr.

1. In the **Label** field, enter a unique identifier for this stage or leave blank to use the default value.
2. In the **Condition** field, enter a script that results in true or false, which determines if the stage should process, or leave blank.
3. In the **Configure Request Handlers Allowed for Queries** section, enter a value for the request handlers used in this stage.
4. In the **HTTP Method** field, select **POST**.

5. Select the **Allow Federated Search** checkbox to enable the use of Solr **collection** and **shards** parameters for this stage.
6. Select the **Generate Response Signal** checkbox to generate a response signal containing metadata about the response from Solr. Response signals are used by App Insights and experiments. This setting only applies if the searchLogs and signals features are enabled for the collection. To avoid generating response signals as Users type, do not select this option.
7. In the **Exclude Response Signal Criteria** section, enter query parameters and Regex patterns to prevent generating a response signal based on specific parameters in the query. For example, use these fields to enable response signals in general, but to disable for auto-complete queries.
8. In the **Preferred Replica Type** field, select **pull** to specify a replicate type that will be given a higher order of precedence when querying Solr. This preference will only be applied for queries that target multiple shards.
9. Click **Save**.

Modify Response with Rules

The *Modify Response with Rules* stage modifies the response from Solr using matching rules from the Apply Rules stage.

1. In the **Label** field, enter a unique identifier for this stage or leave blank to use the default value.
2. In the **Condition** field, enter a script that results in true or false, which determines if the stage should process, or leave blank.
3. In the **Facet Field Label Blob ID**, enter the ID for a blob containing labels for facet fields to add to the response.
4. In the **Facet Label Parse Delimiter** field, enter **||** as the delimiter to parse each facet label mapping in the blob. A Java regular expression is also a valid value. Regex must start with **^** and end with **$**.
5. Click **Save**.

Order the stages

For the pipeline to operate correctly, the stages must be in the following order:

- *Text Tagger*
- *Boost with Signals*
- *Query Fields*
- *LWAI Vectorize Query*
- *Hybrid Query*
- *Apply Rules*
- *Solr Query*
- *Modify Response with Rules*

When you have ordered the stages, click **Save**.

Configure the LWAI Neural Hybrid Search pipeline

Lucidworks AI

How-to

> Compatible versions: 5.9
>
> Works with: *Fusion*

The LWAI Neural Hybrid Search pipeline is a default pipeline that contains all the required query stages to set up *Neural Hybrid Search* using *Lucidworks AI*.

Important

This feature is currently only available to clients who have contracted with Lucidworks for features related to Neural Hybrid Search and Lucidworks AI.

Note

This feature is only available in Fusion 5.9.5 and later versions of Fusion 5.9.

This pipeline uses the following stages:

- *Text Tagger*
- *Boost with Signals*
- *Query Fields*
- *LWAI Vectorize Query*
- *Hybrid Query (5.9.9 and earlier)*
- *Neural Hybrid Query (5.9.10 and later)*
- *Apply Rules*
- *Solr Query*
- *Modify Response with Rules*

Note

Milvus is deprecated. To migrate an existing Milvus collection to Solr vector, see the *Milvus to Solr migration guide*. Migrating prior to Milvus removal is important to prevent disruptions to pipeline performance.

Configure the pipeline

To add the Neural Hybrid Search (NHS) query pipeline:

1. Sign in to Fusion and click **Querying > Query Pipelines**.
2. Select the default **LWAI-neural-hybrid-search-NHS** pipeline.
3. Configure the following stages included in the default pipeline.

Text Tagger

The *Text Tagger* stage queries a Solr text tagger request handler to perform spell correction, phrase boosting, and synonym expansion.

1. In the **Label** field, enter a unique identifier for this stage or leave blank to use the default value.
2. In the **Condition** field, enter a script that results in true or false, which determines if the stage should process, or leave blank.
3. In the **Tagger Collection** field, enter where the tagger request is sent. The default is the **query_rewrite** collection for the application selected. You must enter a collection with only one shard because Text Tagger does not support multi-shard collections. Template expressions are supported.
4. In the **Param to Tag** field, enter a value of **q**, which is the name of the parameter in the request containing text to tag. This field is ignored on DSL requests.
5. In the **Save Tags in Context** field, enter the tags to save in context instead of applying directly to the incoming query in this stage. This enables downstream stages to apply the tags after completing other processing. This field is ignored on DSL requests.
6. Select the following checkboxes:

- Spell Correction
- Phrase Boosting
- Synonym Expansion
- Remove Words
- Tail Rewrites

7. In the **Filter Override** field, enter your filter to override filtering for built-in tagger doc types.
8. In the **Original Term Boost for Synonyms** field, enter the boost to use for the original term during synonym expansion. For example, **2**. To disable this function, enter **-1**.
9. In the **Default Phrase Boost** field, enter the boost to use as a default for phrases that do not have a boost value set. For example, **2**. To disable this function, enter **-1**.
10. In the **Default Phrase Slop** field, enter the distance between the terms of the query while still considering it a phrase match. For example, **10**.
11. In the **Overlapping Tag Policy** field, select the default value of **longest_dominant_right** to ensure the retained tags have no overlaps. The value is the algorithm that determines which tags in an overlapping set should be retained, versus being pruned away. On DSL requests, this field is ignored so the default value of **longest_dominant_right** is always used. The available options correspond to Solr Tagger Handler overlaps: **all**, **no_sub**, or **longest_dominant_right**. Setting the value to **all** or **no_sub** allows more rewrites to potentially be applied to a query, but can increase the chance of producing undesirable rewrites.
12. In the **Additional Params to be included in the Text Tagger Request** section, enter optional values you want to include.
13. In the **Max Wait for Lookup (ms)** field, enter the number of milliseconds to wait for the call to the remote tagger collection to return. For example, **500**. To disable this function, enter **-1**.
14. In the **Skip Query Regex** field, enter the pattern that identifies queries that are skipped because they contain that pattern. For example, you may want to skip single term queries with wildcards.
15. Click **Save**.

Boost with Signals

The *Boost with Signals* stage uses aggregated signals to selectively boost items in the set of search results.

1. In the **Label** field, enter a unique identifier for this stage or leave blank to use the default value.
2. In the **Condition** field, enter a script that results in true or false, which determines if the stage should process, or leave blank.
3. Select the **Asynchronous Execution Config** check box to process this stage asynchronously.
4. In the **Number of Recommendations field**, enter the number of documents to return in the query. For example, **10**.
5. In the **Number of Signals** field, enter the number of signals to process when getting recommended items. For example, **100**.
6. In the **Aggregation Type** field, enter an applicable value. For example, **click@doc_id,filters,query**.
7. In the **Solr Field to Boost On** field, enter which Solr field to use when applying recommendation boosts. For example, **id**.
8. In the **Boost Method** field, select the boost method to use. If the **defType!=edismax** for the main query, select **query-parser**. Another example is **query-param**.
9. In the **Boost Param** field, select one of the following values:
    - **boost** to multiply scores by the boost values
    - **bq** to add optional clauses to main query

10. In the **Solr Query parameters** section, enter the following **Parameter Name:Parameter Value** entries:
    - **qf**: **query_t**
    - **pf**: **query_t^50**
    - **pf**: **query_t-3^20**
    - **pf2**: **query_t^20**
    - **pf2**: **query_t~3^10**
    - **pf3**: **query_t^10**
    - **pf3**: **query_t~3^5**
    - **boost**: **map(query({!field f=query_s v=$q}),0,0,1,20)**
    - **mm**: **50%**
    - **defType**: **edismax**
    - **sort**: **score desc, weight_d desc**
    - **fq**: **weight_d:[** TO **]**

11. In the **Rollup Field**, enter the field name to use when rolling up documents that have the same doc id. For example, **doc_id_s**.
12. In the **Rollup weight field**, enter the field name to use for signal weights. For example, **weight_d**.
13. In the **Rollup weight strategy** field, select one of the following methods to use when rolling up the weight:
    - **max**
    - **sum**

14. In the **Final Boost Weight Expression** field, enter the optional expression to compute the final boost weight using a combination of fields returned by Solr. For example, **score** and **weight_d**. Set to **weight_d** for similar behavior as older versions. Another example is **math:log(weight_d + 1) + 10 * math:log(score+1)**.
15. In the **Document Weights Context Key** field, enter the context key in which to save boosts for **docId:weight_d**.
16. In the **Query Param** field, enter the default value of **q**, which is the name of the parameter in the request containing query to boost.
17. Select the **Include Enriched Query** checkbox to:
    - Enable the stage to combine the user's original query with the output of any stages that enrich the query, such as the Text Tagger stage.
    - Expand the recall of the boost lookup. However, precision may be impacted.
    - Enable the stage to change the configured **mm** parameter to accommodate additional terms added to the boost lookup query.

18. In the **Update Policy** field, select to **replace** or **append** the boost parameter in the final query.
19. Click **Save**.

Query Fields

The *Query Fields* stage configures query parameters for a Solr search.

1. In the **Label** field, enter a unique identifier for this stage or leave blank to use the default value.
2. In the **Condition** field, enter a script that results in true or false, which determines if the stage should process, or leave blank.
3. In the **Number of Results** field, enter the number of query fields to return. For example, **10**.
4. In the **Result Offset** field, enter the value that designates the starting position of the first result from the beginning of the data being queried. Offset is zero-based, so a value of **9** counts from 0 through 9 and the first record returned is the 10th record discovered in the query.
5. In the **Results Sort Order** section, enter values for the following fields:
    - Sort Type. For example, sort on expression, field, query, or relevancy.
    - Field Name or Expression on which to sort.
    - Sort Order. Select either **asc** or **desc**.

6. In the **Search Fields** section, enter the field name and boost values to include in this stage.
7. In the **Return Fields** section, enter the field names to include in this stage.
8. Select the **Return Score** checkbox to set as **true** and determine if a score is determined for this stage.
9. In the **Minimum Should Match** field, enter the minimum string to match for this stage.
10. Select the **Grouping Options** checkbox to enter values in the following fields:
    - Grouping Field. The field name on which to group results.
    - Group Size. The number of results per group.
    - Group Sort. Enter values in the following fields:
        - Sort Type. For example, sort on expression, field, query, or relevancy.
        - Field Name or Expression on which to sort.
        - Sort Order. Select either **asc** or **desc**.

    - Group Leader Strategy. Select this checkbox to define selection criteria for the representative document from each group. Only one method may be used at a time. Defaults to **relevancy** if not specified. Values are:
        - By Field Value. If selected, include documents with the minimum or maximum value for the indicated field to be the representative document for each group.
        - By Sort. If selected, include the representative document for each group based on the order in which they are returned with the given sort criteria.

11. Click **Save**.

LWAI Vectorize Query

The *LWAI Vectorize Query* stage configures parameters to generate a vector by using a Lucidworks AI (LWAI) embedding model to encode the input to a vector representation. This stage is ignored if the input is blank or a wildcard of either an asterisk (*****) or a colon (**:**) is used.

1. In the **Label** field, enter a unique identifier for this stage.
2. In the **Condition** field, enter a script that results in true or false, which determines if the stage should process.
3. In the **Account Name** field, select the name of the Lucidworks AI account.
   If your account name does not appear in the list or you are unsure which one to select, check your *Lucidworks AI Gateway configuration*.
4. In the **Model** field, select the Lucidworks AI model to use for encoding.
   If you do not see any model names and you are a non-admin Fusion user, verify with a Fusion administrator that your user account has these permissions: `PUT,POST,GET:/LWAI-ACCOUNT-NAME/**`
   Your Fusion account name must match the name of the account that you selected in the **Account Name** dropdown.
   For more information about models, see:
   - *Pre-trained embedding models*
   - *Custom embedding model training*
5. In the **Query Input** field, enter the location from which the query is retrieved.
6. In the **Output context variable** field, enter the name of the variable where the vector value from the response is saved.
7. In the **Use Case Configuration** section, click the **+** sign to enter the parameter name and value to send to Lucidworks AI. The `useCaseConfig` parameter that is common to generative AI and embedding use cases is `dataType`, but each use case may have other parameters. The value for the query stage is `query`.
8. In the **Model Configuration** section, click the **+** sign to enter the parameter name and value to send to Lucidworks AI. Several `modelConfig` parameters are common to generative AI use cases. For more information, see *Prediction API*.
9. Select the **Fail on Error** checkbox to generate an exception if an error occurs during this stage.
10. Click **Save**.

> **Note**
>
> The **Top K** setting is set to 100 by default. We recommend leaving this as 100 or setting it to 200.

This query stage must be placed *before* the ***Solr Query stage***. For more information, see *Reorder Query Pipeline Stages*.

Hybrid Query (5.9.9 and earlier) - deprecated

The *Hybrid Query* stage is a combination of semantic vector search and lexical search. This stage is ignored if the input is blank or a wildcard of either an asterisk (*****) or a colon (**:**).

In addition, this stage does not function correctly if the incoming **q** parameter is a Solr query parser string. For example, **field_t:foo** rather than a raw user query string.

The resulting query is *always* written to **<request.params.q>**.

1. In the **Label** field, enter a unique identifier for this stage or leave blank to use the default value.
2. In the **Condition** field, enter a script that results in true or false, which determines if the stage should process, or leave blank.
3. In the **Lexical Query Input** field, enter the location from which the lexical query is retrieved. For example, **<request.params.q>**. Template expressions are supported.
4. In the **Lexical Query Weight** field, enter the relative weight of the lexical query. For example, **0.3**. If this value is **0**, no re-ranking will be applied using the lexical query scores.

5. In the **Number of Lexical Results** field, enter the number of lexical search results to include in re-ranking. For example, **1000**. A value is **0** is ignored.
6. In the **Vector Query Field**, enter the name of the Solr field for k-nearest neighbor (KNN) vector search.
7. In the **Vector Input** field, enter the location from which the vector is retrieved. Template expressions are supported. For example, a value of `<ctx.vector>` evaluates the context variable resulting from a previous stage, such as the *LWAI Vectorize Query* stage.
8. In the **Vector Query Weight** field, enter the relative weight of the vector query. For example, **0.7**.
9. Select the **Use KNN Query** checkbox to use the **knn** query parser and configure its options. This option cannot be selected if **Use VecSim Query** checkbox is selected. In addition, **Use KNN Query** is used if neither **Use KNN Query** or **Use VecSim Query** is selected.
    a. If the **Use KNN Query checkbox** is selected, enter a value in the **Number of Vector Results** field. For example, **1000**.
10. Select the **Use VecSim Query** checkbox to use the **vecSim** query parser and configure its options. This option cannot be selected if **Use KNN Query** checkbox is selected.
    If the **Use VecSim Query** checkbox is selected, enter values in the following fields:
    - **Min Return Vector Similarity**. Enter the minimum vector similarity value to qualify as a match from the Vector portion of the hybrid query.
    - **Min Traversal Vector Similarity**. Enter the minimum vector similarity value to use when walking through the graph during the Vector portion of the hybrid query. The value must be lower than, or equal to, the value in the Min Return Vector Similarity field.
11. In the **Minimum Vector Similarity Filter**, enter the value for a minimum similarity threshold for filtering documents. This option applies to all documents, regardless of other score boosting such as rules or signals.
12. Click **Save**.

Neural Hybrid Query (5.9.10 and later)

The *Neural Hybrid Query* stage is a combination of semantic vector search and lexical search. This stage is ignored if the input is blank or a wildcard of either an asterisk (*****) or a colon (**:**).

In addition, this stage does not function correctly if the incoming **q** parameter is a Solr query parser string. For example, **field_t:foo** rather than a raw user query string.

The resulting query is *always* written to **<request.params.q>**.

1. In the **Label** field, enter a unique identifier for this stage or leave blank to use the default value.
2. In the **Condition** field, enter a script that results in true or false, which determines if the stage should process, or leave blank.
3. In the **Lexical Query Input** field, enter the location from which the lexical query is retrieved. For example, **<request.params.q>**. Template expressions are supported.
4. In the **Lexical Query Weight** field, enter the relative weight of the lexical query. For example, **0.3**. If this value is **0**, no re-ranking will be applied using the lexical query scores.
5. In the **Lexical Query Squash Factor** field, enter a value that will be used to squash the lexical query score. The squash factor controls how much difference there is between the top-scoring documents and the rest. It helps ensure that documents with slightly lower scores still have a chance to show up near the top. For this value, Lucidworks recommends entering the inverse of the lexical maximum score across all queries for the given collection.
6. In the **Vector Query Field**, enter the name of the Solr field for k-nearest neighbor (KNN) vector search.
7. In the **Vector Input** field, enter the location from which the vector is retrieved. Template expressions are supported. For example, a value of `<ctx.vector>` evaluates the context variable resulting from a previous stage, such as the *LWAI Vectorize Query* stage.
8. In the **Vector Query Weight** field, enter the relative weight of the vector query. For example, **0.7**.
9. In the **Min Return Vector Similarity** field, enter the minimum vector similarity value to qualify as a match from the Vector portion of the hybrid query.
10. In the **Min Traversal Vector Similarity** field, enter the minimum vector similarity value to use when walking through the graph during the Vector portion of the hybrid query.
11. When enabled, the **Compute Vector Similarity for Lexical-Only Matches** setting computes vector similarity scores for documents in lexical search results but not in the initial vector search results. Select the checkbox to enable this setting.
12. If you want to use pre-filtering:
    a. Uncheck **Block pre-filtering**.
    In the Javascript context (`ctx`), the `preFilterKey` object becomes available.

b. Add a *Javascript stage after* the Neural Hybrid Query stage and use it to configure your pre-filter. The `preFilter` object adds both the top-level `fq` and `preFilter` to the parameters for the vector query. You do not need to manually add the top level `fq` in the javascript stage. See the example below:

```
var QueryRequestAndResponse =
Java.type('com.lucidworks.apollo.pipeline.query.QueryReque
stAndResponse'); if(ctx.hasProperty("preFilterKey")) { var
preFilter = ctx.getProperty("preFilterKey"); var wrapper =
QueryRequestAndResponse.create(request,response,0)
preFilter.addFilter(wrapper, 'id:* OR foo_s:bar'); }
```

13. Click **Save**.

Apply Rules

The *Apply Rules* stage applies the rules configured in the collection to the query.

1. In the **Label** field, enter a unique identifier for this stage or leave blank to use the default value.
2. In the **Condition** field, enter a script that results in true or false, which determines if the stage should process, or leave blank.
3. Select the **Asynchronous Execution Config** check box to process this stage asynchronously.
4. In the **Collection field**, enter the name of the collection that contains the rules. If this field does not contain a value, the default rules collection for the selected application is used. Template expressions are supported.
5. In the **Request Handler** field, enter a value of **select**.
6. In the **HTTP Method** field, select **POST**.
7. In the **Rule Triggering Limit** field, enter the maximum number of business rules to be triggered by the query. The default rules matching limit is **100**. This configuration overwrites the **rows** parameter set in Query Parameters section.
8. In the **Query Parameters** section, enter the names and values to use when querying the rules collection. If you set the **rows** parameter here, it will be overwritten by the configuration in the Rule Triggering Limit field.
9. In the **Subquery Rewrite Pipeline id** field, enter the value to call a Fusion query pipeline to modify the rule-retrieving subquery. Template expressions are supported.
10. In the **Headers** section, enter the names and values to use in this stage.
11. Select the **Use Original Query If No Rules Match** checkbox so the stage will try to match rules using the original query (un-tagged) sent into the Text Tagger stage, if available.
12. Select the **Partially Matched Filter Queries Will Trigger the Rule** checkbox so the stage will trigger filter rules as long as there is one filter query in the query parameter that matches the filter specified in the rule.
13. In the **Max Wait for Lookup (ms)**, enter the number of milliseconds to wait for the call to the remote tagger collection to return. For example, **500**. To disable this function, enter **-1**.
14. Click **Save**.

Solr Query

The *Solr Query* stage sends the search request to Solr.

1. In the **Label** field, enter a unique identifier for this stage or leave blank to use the default value.
2. In the **Condition** field, enter a script that results in true or false, which determines if the stage should process, or leave blank.

3. In the **Configure Request Handlers Allowed for Queries** section, enter a value for the request handlers used in this stage.
4. In the **HTTP Method** field, select **POST**.
5. Select the **Allow Federated Search** checkbox to enable the use of Solr **collection** and **shards** parameters for this stage.
6. Select the **Generate Response Signal** checkbox to generate a response signal containing metadata about the response from Solr. Response signals are used by App Insights and experiments. This setting only applies if the searchLogs and signals features are enabled for the collection. To avoid generating response signals as Users type, do not select this option.
7. In the **Exclude Response Signal Criteria** section, enter query parameters and Regex patterns to prevent generating a response signal based on specific parameters in the query. For example, use these fields to enable response signals in general, but to disable for auto-complete queries.
8. In the **Preferred Replica Type** field, select **pull** to specify a replicate type that will be given a higher order of precedence when querying Solr. This preference will only be applied for queries that target multiple shards.
9. Click **Save**.

Modify Response with Rules

The *Modify Response with Rules* stage modifies the response from Solr using matching rules from the Apply Rules stage.

1. In the **Label** field, enter a unique identifier for this stage or leave blank to use the default value.
2. In the **Condition** field, enter a script that results in true or false, which determines if the stage should process, or leave blank.
3. In the **Facet Field Label Blob ID**, enter the ID for a blob containing labels for facet fields to add to the response.
4. In the **Facet Label Parse Delimiter** field, enter **||** as the delimiter to parse each facet label mapping in the blob. A Java regular expression is also a valid value. Regex must start with **^** and end with **$**.
5. Click **Save**.

Order the stages

For the pipeline to operate correctly, the stages must be in the following order:

- *Text Tagger*
- *Boost with Signals*
- *Query Fields*
- *LWAI Vectorize Query*
- *Hybrid Query*
- *Apply Rules*
- *Solr Query*
- *Modify Response with Rules*

When you have ordered the stages, click **Save**.

Configure the LWAI Prediction index Stage

Lucidworks AI

How-to

> Compatible versions: 5.9
>
> Works with: *Fusion*, *Managed Fusion*

The LWAI Prediction index stage is a Fusion index pipeline stage that enriches your index with *Generative AI* predictions. It defaults to asynchronous processing, which does not block the pipeline while waiting for a response from Lucidworks AI.

For reference information, see *LWAI Prediction index stage*. For the LWAI Prediction query stage, see *Configure the LWAI Prediction query stage*.

To use this stage, non-admin Fusion users must be granted the `PUT,POST,GET:/LWAI-ACCOUNT-NAME/**` permission in Fusion, which is the Lucidworks AI API Account Name defined in Lucidworks AI Gateway when this stage is configured.

To configure this stage:

1. Sign in to Fusion and click **Indexing > Index Pipelines**.
2. Click **Add+** to add a new pipeline.
3. Enter the name in **Pipeline ID**.
4. Click **Add a new pipeline stage**.
5. In the AI section, click **LWAI Prediction**.
6. In the **Label** field, enter a unique identifier for this stage.
7. In the **Condition** field, enter a script that results in true or false, which determines if the stage should process.
8. In the **Account Name** field, select the Lucidworks AI API account name defined in *Lucidworks AI Gateway*.
9. In the **Use Case** field, select the Lucidworks AI use case to associate with this stage.
    - To generate a list of the use cases for your organization, see *Use Case API*.
    - If the **Call Asynchronously?** check box is selected, see available use cases described in *Async Prediction API*.
    - If the **Call Asynchronously?** check box is *not* selected, see available use cases described in *Prediction API*.
10. In the **Model** field, select the Lucidworks AI model to associate with this stage.
    If you do not see any model names and you are a non-admin Fusion user, verify with a Fusion administrator that your user account has these permissions: `PUT,POST,GET:/LWAI-ACCOUNT-NAME/**`
    Your Fusion account name must match the name of the account that you selected in the **Account Name** dropdown.
    For more information about models, see:
    - *Pre-trained embedding models*
    - *Custom embedding model training*
    - *Generative AI models*
11. In the **Input context variable** variable field, enter the name of the variable in context to be used as input. Template expressions are supported.
12. In the **Destination field name and context output** field, enter the name that will be used as both the field name in the document where the prediction is written and the context variable that contains the prediction.
    - If the **Call Asynchronously?** check box is selected and a value is entered in this field:
        - `{destination name}_t` is the full response.
        - In the document:
            - `_lw_ai_properties_ss` contains the Lucidworks account, boolean setting for async, use case, input for the call, and the collection.
            - `_lw_ai_request_count` is the number of GET requests by `predictionId` and `_lw_ai_success_count` is the number of responses without errors. These two fields are used for debugging only. Based on the deployment, the most useful measure is the ratio of `_lw_ai_success_count` divided by `_lw_ai_request_count and then adjusting as much as possible to achieve 1.0.
            - `enriched_ss` contains the use case. This can be used as a boolean value to verify if the use case indexed successfully.
    - If the **Call Asynchronously?** check box is *not* selected and a value is entered in this field:
        - `{destination name}_t` is the full response.

- If *no* value is entered in this field (regardless of the **Call Asynchronously?** check box setting):
    - `lw_ai{use case}_t` is the `response.response` object, which is the raw model output.
    - `lw_ai{use case}_response_s` is the full response.

13. In the **Use Case Configuration** section, click the **+** sign to enter the parameter name and value to send to Lucidworks AI. The `useCaseConfig` parameter is only applicable to certain use cases.
    - If the **Call Asynchronously?** check box is selected, `useCaseConfig` information for each applicable use case is described in *Async Prediction API*.
    - If the **Call Asynchronously?** check box is *not* selected, `useCaseConfig` information for each applicable use case is described in *Prediction API*.

14. In the **Model Configuration** section, click the **+** sign to enter the parameter name and value to send to Lucidworks AI. Several `modelConfig` parameters are common to generative AI use cases.
    - If the **Call Asynchronously?** check box is selected, `modelConfig` information is described in *Async Prediction API*.
    - If the **Call Asynchronously?** check box is *not* selected, `modelConfig` information is described in *Prediction API*.

15. In the **API Key** field, enter the secret value specified in the external model. For:
    - OpenAI models, `"apiKey"` is the value in the model's `"[OPENAI_API_KEY]"` field. For more information, see Authentication API keys.
    - Azure OpenAI models, `"apiKey"` is the value generated by Azure in either the model's `"[KEY1 or KEY2]"` field. For requirements to use Azure models, see *Generative AI models*.
    - Google VertexAI models, `"apiKey"` is the value in the model's `"[BASE64_ENCODED_GOOGLE_SERVICE_ACCOUNT_KEY]"` field. For more information, see Create and delete Google service account keys.

16. To run the API call asynchronously, select the **Call Asynchronously?** check box to specify the stage is to use the *Lucidworks AI Async Prediction API* endpoints. If this is selected, the API call does not block the pipeline while waiting for a response from Lucidworks AI.
    If the check box is *not* selected, the API call uses the *Prediction API*, which uses the pipeline until a response is received from Lucidworks AI. Performance of other API calls can be impacted.
17. In the **Maximum Asynchronous Call Tries** field, enter the maximum number of times to send an asynchronous API call before the system generates a failure error.
18. Select the **Fail on Error** checkbox to generate an exception if an error occurs while generating a prediction for a document.
19. Click **Save**.

Additional requirements

Additional requirements to use async calls include:

- Use a V2 connector. Only V2 connectors work for this task and not other options, such as PBL or V1 connectors.
- Remove the `Apache Tika` stage from your parser because it can cause datasource failures with the following error: "The following components failed: [class com.lucidworks.connectors.service.components.job.processor.DefaultDataProcessor : Only Tika Container parser can support Async Parsing.]"
- Replace the `Solr Indexer` stage with the `Solr Partial Update Indexer` stage with the following settings:
    - `Enable Concurrency Control` set to off
    - `Reject Update if Solr Document is not Present` set to off
    - `Process All Pipeline Doc Fields` set to on
    - `Allow reserved fields` set to on
    - A parameter with `Update Type`, `Field Name` & `Value` in `Updates`

Configure the LWAI Prediction query stage

Lucidworks AI

How-to

> Compatible versions: 5.9
>
> Works with: *Fusion*, *Managed Fusion*

The LWAI Prediction AI query stage is a Fusion pipeline query stage that enriches your search results with *Generative AI* predictions.

For reference information, see *LWAI Prediction query stage*. For the LWAI Prediction index stage, see *Configure the LWAI Prediction index stage*.

To use this stage, non-admin Fusion users must be granted the `PUT,POST,GET:/LWAI-ACCOUNT-NAME/**` permission in Fusion, which is the Lucidworks AI API Account Name defined in Lucidworks AI Gateway when this stage is configured.

To configure this stage:

1. Sign in to Fusion and click **Querying > Query Pipelines**.
2. Click **Add+** to add a new pipeline.
3. Enter the name in **Pipeline ID**.
4. Click **Add a new pipeline stage**.
5. In the AI section, click **LWAI Prediction**.
6. In the **Label** field, enter a unique identifier for this stage.
7. In the **Condition** field, enter a script that results in true or false, which determines if the stage should process.
8. Select **Asynchronous Execution Config** if you want to run this stage asynchronously. If this field is enabled, complete the following fields:
    a. Select **Enable Async Execution**. Fusion automatically assigns an **Async ID** value to this stage. Change this to a more memorable string that describes the asynchronous stages you are merging, such as `signals` or `access_control`.
    b. Copy the **Async ID** value.

    Note

    For detailed information about Fusion, see *Enable asynchronous query pipeline processing* and *Asynchronous query pipeline processing*.
    For detailed information about Managed Fusion, see *Enable asynchronous query pipeline processing* and *Asynchronous query pipeline processing*.

9. In the **Account Name** field, select the name of the Lucidworks AI integration defined when the integration was created.
10. In the **Use Case** field, select the Lucidworks AI use case to associate with this stage.
    - To generate a list of the use cases for your organization, see *Use Case API*.
    - The available use cases are described in *Prediction API*.
11. In the **Model** field, select the Lucidworks AI model to associate with this stage.
    If you do not see any model names and you are a non-admin Fusion user, verify with a Fusion administrator that your user account has these permissions: `PUT,POST,GET:/LWAI-ACCOUNT-NAME/**`
    Your Fusion account name must match the name of the account that you selected in the **Account Name** dropdown.
    For more information about models, see:
    - *Pre-trained embedding models*
    - *Custom embedding model training*
    - *Generative AI models*
12. In the **Input context variable** variable field, enter the name of the variable in context to be used as input. Template expressions are supported.
13. In the **Destination variable name and context output** field, enter the name that will be used as both the query response header in the prediction results and the context variable that contains the prediction.
    - If a value is entered in this field:
        - `{destination name}_t` is the full response.
        - In the context:

        - `_lw_ai_properties_ss` contains the Lucidworks account, boolean setting for async, use case, input for the call, and the collection.

- If *no* value is entered in this field:
    - `lw_ai{use case}_t` is the `response.response` object, which is the raw model output.
    - `lw_ai{use case}_response_s` is the full response.

14. Select the **Include Response Documents?** check box to include the response documents in the Lucidworks AI request. This option is only available for certain use cases. If this is selected, run the Solr Query stage to ensure documents exist before running the LWAI Prediction query stage.
    Response documents must be included in the RAG use case, which supports attaching a maximum of 3 response documents. To prevent errors, enter all of the entries described in the Document Field Mappings section.
15. In the **Document Field Mappings** section, enter the **LW AI Document field name** and its corresponding **Response document field name** to map from input documents to the fields accepted by the Prediction API RAG use case. The fields are described in the *Prediction API*.
    If information is not entered in this section, the default mappings are used.
    - The `body` and `source` fields are required.
        - `body` - `description_t` is the contents of the document.
        - `source` - `link_t` is the URL/ID of the document.
    - The `title` and `date` fields are optional.
        - `title` - `title_t` is the title of the document.
        - `date` - `_lw_file_modified_tdt` is the creation date of the document in epoch time format.

16. In the **Use Case Configuration** section, click the **+** sign to enter the parameter name and value to send to Lucidworks AI.
    - The `useCaseConfig` parameter is only applicable to certain use cases. For more information, see the *Async Prediction API* and the *Prediction API*.
    - The `memoryUuid` parameter is required in the Standalone Query Rewriter use case, and is optional in the RAG use case.
      For more information, see *Prediction API*.

17. In the **Model Configuration** section, click the **+** sign to enter the parameter name and value to send to Lucidworks AI. Several `modelConfig` parameters are common to generative AI use cases. For more information, see *Prediction API*.
18. In the **API Key** field, enter the secret value specified in the external model. For:
    - OpenAI models, `"apiKey"` is the value in the model's `"[OPENAI_API_KEY]"` field. For more information, see Authentication API keys.
    - Azure OpenAI models, `"apiKey"` is the value generated by Azure in either the model's `"[KEY1 or KEY2]"` field. For requirements to use Azure models, see *Generative AI models*.
    - Google VertexAI models, `"apiKey"` is the value in the model's `"[BASE64_ENCODED_GOOGLE_SERVICE_ACCOUNT_KEY]"` field. For more information, see Create and delete Google service account keys.

19. Select the **Fail on Error** checkbox to generate an exception if an error occurs during this stage.
20. Click **Save**.

Configure the LWAI Vectorize pipeline

Lucidworks AI

How-to

> Compatible versions: 5.9
>
> Works with: *Managed Fusion*

The LWAI Vectorize pipeline is a default pipeline that contains the required index stages to set up vector search using *Lucidworks AI*.

Important

This feature is currently only available to clients who have contracted with Lucidworks for features related to Neural Hybrid Search and Lucidworks AI.

Note

This feature is only available in Managed Fusion 5.9.x for versions 5.9.6+.

This pipeline uses the following stages:

- *Field Mapping*
- *Solr Dynamic Field Name Mapping*
- *LWAI Vectorize Field*
- *Solr Indexer*

Configure the pipeline

To add the Lucidworks AI (LWAI) Vectorize index pipeline:

1. Sign in to Managed Fusion and click **Indexing > Index Pipelines**.
2. Select the default **LWAI-vectorize** pipeline.
3. Configure the following stages included in the default pipeline.

Field Mapping

The *Field Mapping* stage customizes mapping of the fields in an index pipeline document to fields in the Solr scheme.

To configure this stage for the index pipeline:

1. In the **Label** field, enter a unique identifier for this stage or leave blank to use the default value.
2. In the **Condition** field, enter a script that results in true or false, which determines if the stage should process, or leave blank.
3. Select the **Allow System Fields Mapping?** checkbox to map system fields in this stage.
4. In the **Field Retention** section, enter specific fields to either keep or delete.
5. In the **Field Value Updates** section, enter specific fields and then designate the value to either add to the field, or set on the field. When a value is added, any values previously on the field are retained. When a value is set, any values previously on the field are overwritten by the new value entered.
6. In the **Field Translations** section, enter specific fields to either move or copy to a different field. When a field is moved, the values from the source field are moved over to the target field and the source field is removed. When a field is copied, the values from the source field are copied over to the target field and the source field is retained.
7. Select the **Unmapped Fields** checkbox to specify the operation on the fields *not* mapped in the previous sections. Select the **Keep** checkbox to keep all unmapped fields. This is the only option you need to select for the LWAI-vectorize stage.
8. Click **Save**.

Solr Dynamic Field Name Mapping

The *Solr Dynamic Field Name Mapping* stage maps pipeline document fields to Solr dynamic fields.

1. In the **Label** field, enter a unique identifier for this stage or leave blank to use the default value.

2. In the **Condition** field, enter a script that results in true or false, which determines if the stage should process, or leave blank.
3. Select the **Duplicate Single-Valued Fields as Multi-Valued Fields** checkbox to enable indexing of field data into both single-valued and multi-valued Solr fields. For example, if this option is selected, the `phone` field is indexed into both the `phone_s` single-valued field and the `phone_ss` multi-valued field. If this option is not selected, the `phone` field is indexed into only the `phone_s` single-valued field.
4. In the **Field Not To Map** section, enter the names of the fields that should not be mapped by this stage.
5. Select the **Text Fields Advanced Indexing** checkbox to enable indexing of text data that doesn't exceed a specific maximum length, into both tokenized and non-tokenized fields. For example, if this option is selected, the `name` text field with a value of **John Smith** is indexed into both the `name_t` and `name_s` fields allowing relevant search using `name_t` field (by matching to a Smith query) and also proper faceting and sorting using `name_s` field (using John Smith for sorting or faceting). If this option is not selected, the `name` text field is indexed into only the `name_t` text field by default.
6. In the **Max Length for Advanced Indexing of Text Fields** field, enter a value used to determine how many characters of the incoming text is indexed. For example, **100**.
7. Click **Save**.

LWAI Vectorize Field

The *LWAI Vectorize* stage invokes a Lucidworks AI model to encode a string field to a vector representation. This stage is skipped if the field to encode doesn't exist or is null on the pipeline document.

1. In the **Label** field, enter a unique identifier for this stage.
2. In the **Condition** field, enter a script that results in true or false, which determines if the stage should process.
3. In the **Account Name** field, select the Lucidworks AI API account name defined in *Lucidworks AI Gateway*. If you do not see your account name or you are unsure which one to select, contact the Managed Fusion team at Lucidworks.
4. In the **Model** field, select the Lucidworks AI model to use for encoding.
   If you do not see your model name or you are unsure which one to select, contact the Managed Fusion team at Lucidworks.

   For more information, see:
   - *Pre-trained embedding models*
   - *Custom embedding model training*. To use a custom model, you must obtain the deployment ID from the deployments screen.

5. In the **Source** field, enter the name of the string field where the value should be submitted to the model for encoding. If the field is blank or does not exist, this stage is not processed. Template expressions are supported.
6. In the **Destination** field, enter the name of the field where the vector value from the model response is saved.
   - If a value is entered in this field, the following information added to the document:
     - `{Destination Field}_b` is the boolean value if the vector has been indexed.
     - `{Destination Field}` is the vector field.

7. In the **Use Case Configuration** section, click the **+** sign to enter the parameter name and value to send to Lucidworks AI. The `useCaseConfig` parameter that is common to embedding use cases is `dataType`, but each use case may have other parameters. The value for the query stage is `query`.
8. The **Model Configuration** section is not currently available.
9. The **Call Asynchronously?** check box is not currently available.
10. Select the **Fail on Error** checkbox to generate an exception if an error occurs while generating a prediction for a document.
11. Click **Save**.
12. Index data using the new pipeline. Verify the vector field is indexed by confirming the field is present in documents.

Solr Indexer

The *Solr Indexer* stage transforms a Managed Fusion pipeline document into a Solr document, and sends it to Solr for indexing into a collection.

To configure this stage for the index pipeline:

1. In the **Label** field, enter a unique identifier for this stage or leave blank to use the default value.
2. In the **Condition** field, enter a script that results in true or false, which determines if the stage should process, or leave blank.
3. Select the **Map to Solr Schema** checkbox to select and add static and dynamic fields to map in this stage.
4. Select the **Add a field listing all document fields** checkbox to add the `_lw_fields_ss` multi-valued field to the document, which lists all fields that are being sent to Solr.
5. In the **Additional Date Formats** section, enter date formats to include in this stage.
6. In the **Additional Update Request Parameters** section, enter the parameter names and values to update the request parameters.
7. Select the **Buffer Documents and Send Them To Solr in Batches** checkbox to process the documents in batches for this stage.
8. In the **Buffer Size** field, enter the number of documents in a batch before sending the batch to Solr. If no value is specified, the default value for this search cluster is used.
9. In the **Buffer Flush Interval (milliseconds)** field, enter the maximum number of milliseconds to hold the batch before sending the batch to Solr. If no value is specified, the default value for this search cluster is used.
10. Select the **Allow expensive request parameters** checkbox to allow `commit=true` and `optimize=true` to be passed to Solr when specified as request parameters coming into this pipeline. Document commands that specify commit or optimize are still respected even if this checkbox is not selected.
11. Select the **Unmapped Fields Mapping** checkbox to specify the information for all of the fields *not* mapped in the previous sections.
    - In the **Source** Field, enter the name of the unmapped field to be mapped.
    - In the **Target** Field, enter the name of the Solr field to which the unmapped field is mapped.
    - In the **Operation** field, select how the field is mapped. The options are:
        - **Add** the unmapped field to the Solr field.
        - **Copy** the unmapped field to the Solr field and retain the value in the Source field.
        - **Delete** the unmapped field.
        - **Keep** the unmapped field and do not map it to a Solr field.
        - **Move** (replace) the Solr field value with the unmapped field Source value and remove the value from the Source field.
        - **Set** the value of the unmapped field to the value in the Solr field.
12. Click **Save**.

Order the stages

For the pipeline to operate correctly, the stages must be in the following order:

- *Field Mapping*
- *Solr Dynamic Field Name Mapping*
- *LWAI Vectorize Field*
- *Solr Indexer*

When you have ordered the stages, click **Save**.

Configure the LWAI Vectorize pipeline

Lucidworks AI

How-to

> Compatible versions: 5.9
>
> Works with: *Fusion*

The LWAI Vectorize pipeline is a default pipeline that contains the required index stages to set up vector search using *Lucidworks AI*.

For more information,refer to *Configure Neural Hybrid Search*.

Important

This feature is currently only available to clients who have contracted with Lucidworks for features related to Neural Hybrid Search and Lucidworks AI.

Note

This feature is only available in Fusion 5.9.5 and later versions of Fusion 5.9.

This pipeline uses the following stages:

- *Field Mapping*
- *Solr Dynamic Field Name Mapping*
- *LWAI Vectorize Field*
- *Solr Indexer*

Configure the pipeline

To add the Lucidworks AI (LWAI) Vectorize index pipeline:

1. Sign in to Fusion and click **Indexing > Index Pipelines**.
2. Select the default **LWAI-vectorize** pipeline.
3. Configure the following stages included in the default pipeline.

Field Mapping

The *Field Mapping* stage customizes mapping of the fields in an index pipeline document to fields in the Solr scheme.

To configure this stage for the index pipeline:

1. In the **Label** field, enter a unique identifier for this stage or leave blank to use the default value.
2. In the **Condition** field, enter a script that results in true or false, which determines if the stage should process, or leave blank.
3. Select the **Allow System Fields Mapping?** checkbox to map system fields in this stage.
4. In the **Field Retention** section, enter specific fields to either keep or delete.
5. In the **Field Value Updates** section, enter specific fields and then designate the value to either add to the field, or set on the field. When a value is added, any values previously on the field are retained. When a value is set, any values previously on the field are overwritten by the new value entered.
6. In the **Field Translations** section, enter specific fields to either move or copy to a different field. When a field is moved, the values from the source field are moved over to the target field and the source field is removed. When a field is copied, the values from the source field are copied over to the target field and the source field is retained.
7. Select the **Unmapped Fields** checkbox to specify the operation on the fields *not* mapped in the previous sections. Select the **Keep** checkbox to keep all unmapped fields. This is the only option you need to select for the LWAI-vectorize stage.
8. Click **Save**.

Solr Dynamic Field Name Mapping

The *Solr Dynamic Field Name Mapping* stage maps pipeline document fields to Solr dynamic fields.

1. In the **Label** field, enter a unique identifier for this stage or leave blank to use the default value.
2. In the **Condition** field, enter a script that results in true or false, which determines if the stage should process, or leave blank.
3. Select the **Duplicate Single-Valued Fields as Multi-Valued Fields** checkbox to enable indexing of field data into both single-valued and multi-valued Solr fields. For example, if this option is selected, the `phone` field is indexed into both the `phone_s` single-valued field and the `phone_ss` multi-valued field. If this option is not selected, the `phone` field is indexed into only the `phone_s` single-valued field.
4. In the **Field Not To Map** section, enter the names of the fields that should not be mapped by this stage.
5. Select the **Text Fields Advanced Indexing** checkbox to enable indexing of text data that doesn't exceed a specific maximum length, into both tokenized and non-tokenized fields. For example, if this option is selected, the `name` text field with a value of **John Smith** is indexed into both the `name_t` and `name_s` fields allowing relevant search using `name_t` field (by matching to a Smith query) and also proper faceting and sorting using `name_s` field (using John Smith for sorting or faceting). If this option is not selected, the `name` text field is indexed into only the `name_t` text field by default.
6. In the **Max Length for Advanced Indexing of Text Fields** field, enter a value used to determine how many characters of the incoming text is indexed. For example, **100**.
7. Click **Save**.

LWAI Vectorize Field

The *LWAI Vectorize* stage invokes a Lucidworks AI model to encode a string field to a vector representation. This stage is skipped if the field to encode doesn't exist or is null on the pipeline document.

1. In the **Label** field, enter a unique identifier for this stage.
2. In the **Condition** field, enter a script that results in true or false, which determines if the stage should process.
3. In the **Account Name** field, select the Lucidworks AI API account name defined in *Lucidworks AI Gateway*. If your account name does not appear in the list or you are unsure which one to select, check your *Lucidworks AI Gateway configuration*.
4. In the **Model** field, select the Lucidworks AI model to use for encoding.
   If your model does not appear in the list or you are unsure which one to select, check your *Lucidworks AI Gateway configuration*.

   For more information, see:
   - *Pre-trained embedding models*
   - *Custom embedding model training*. To use a custom model, you must obtain the deployment ID from the deployments screen.

5. In the **Source** field, enter the name of the string field where the value should be submitted to the model for encoding. If the field is blank or does not exist, this stage is not processed. Template expressions are supported.
6. In the **Destination** field, enter the name of the field where the vector value from the model response is saved.
   - If a value is entered in this field, the following information is added to the document:
     - `{Destination Field}_b` is the boolean value if the vector has been indexed.
     - `{Destination Field}` is the vector field.

7. In the **Use Case Configuration** section, click the **+** sign to enter the parameter name and value to send to Lucidworks AI. The `useCaseConfig` parameter that is common to embedding use cases is `dataType`, but each use case may have other parameters. The value for the query stage is `query`.
8. The **Model Configuration** section is not currently available.
9. The **Call Asynchronously?** check box is not currently available.
10. Select the **Fail on Error** checkbox to generate an exception if an error occurs while generating a prediction for a document.
11. Click **Save**.
12. Index data using the new pipeline. Verify the vector field is indexed by confirming the field is present in documents.

Solr Indexer

The *Solr Indexer* stage transforms a Fusion pipeline document into a Solr document, and sends it to Solr for indexing into a collection.

To configure this stage for the index pipeline:

1. In the **Label** field, enter a unique identifier for this stage or leave blank to use the default value.
2. In the **Condition** field, enter a script that results in true or false, which determines if the stage should process, or leave blank.
3. Select the **Map to Solr Schema** checkbox to select and add static and dynamic fields to map in this stage.
4. Select the **Add a field listing all document fields** checkbox to add the `_lw_fields_ss` multi-valued field to the document, which lists all fields that are being sent to Solr.
5. In the **Additional Date Formats** section, enter date formats to include in this stage.
6. In the **Additional Update Request Parameters** section, enter the parameter names and values to update the request parameters.
7. Select the **Buffer Documents and Send Them To Solr in Batches** checkbox to process the documents in batches for this stage.
8. In the **Buffer Size** field, enter the number of documents in a batch before sending the batch to Solr. If no value is specified, the default value for this search cluster is used.
9. In the **Buffer Flush Interval (milliseconds)** field, enter the maximum number of milliseconds to hold the batch before sending the batch to Solr. If no value is specified, the default value for this search cluster is used.
10. Select the **Allow expensive request parameters** checkbox to allow `commit=true` and `optimize=true` to be passed to Solr when specified as request parameters coming into this pipeline. Document commands that specify commit or optimize are still respected even if this checkbox is not selected.
11. Select the **Unmapped Fields Mapping** checkbox to specify the information for all of the fields *not* mapped in the previous sections.
    - In the **Source** Field, enter the name of the unmapped field to be mapped.
    - In the **Target** Field, enter the name of the Solr field to which the unmapped field is mapped.
    - In the **Operation** field, select how the field is mapped. The options are:
        - **Add** the unmapped field to the Solr field.
        - **Copy** the unmapped field to the Solr field and retain the value in the Source field.
        - **Delete** the unmapped field.
        - **Keep** the unmapped field and do not map it to a Solr field.
        - **Move** (replace) the Solr field value with the unmapped field Source value and remove the value from the Source field.
        - **Set** the value of the unmapped field to the value in the Solr field.
12. Click **Save**.

Order the stages

For the pipeline to operate correctly, the stages must be in the following order:

- *Field Mapping*
- *Solr Dynamic Field Name Mapping*
- *LWAI Vectorize Field*
- *Solr Indexer*

When you have ordered the stages, click **Save**.

Configure the Rules Simulator Query Profile

How-to

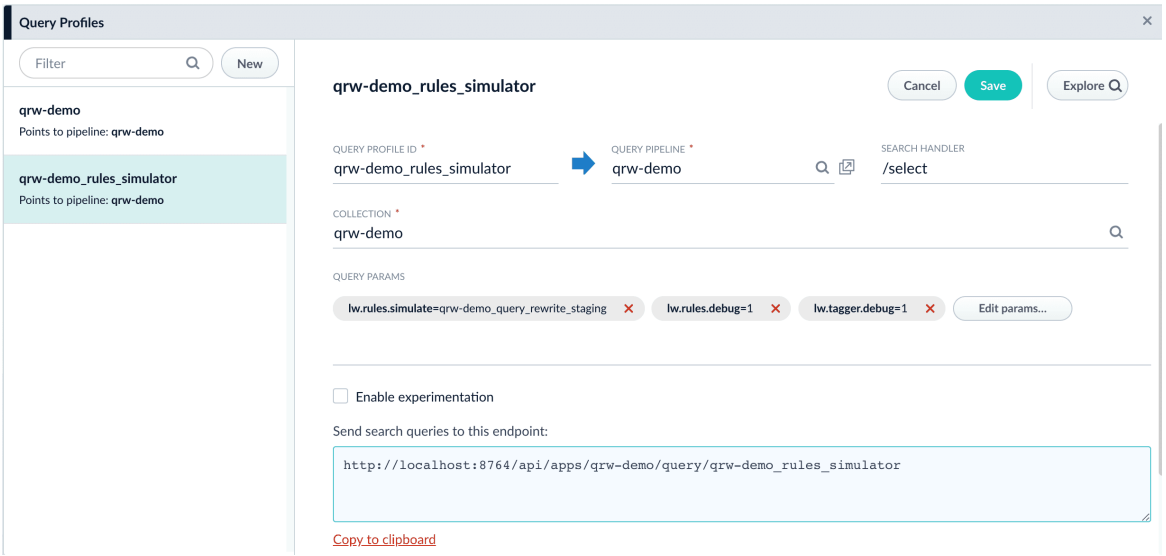
> Compatible versions: 4.2, 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion AI*, *Fusion Server*, *Fusion*

Each app has a `_rules_simulator` query profile, configured to use the `_query_rewrite_staging` collection for *query rewrites* instead of the `_query_rewrite` collection. This profile is created automatically whenever a new app is created.

By default, this query profile points to your default query pipeline and collection. You can configure it to point to any pipeline or collection, for example when testing a new pipeline before it has been deployed.

How to change the query pipeline, collection, and query parameters used by the `_rules_simulator` query profile

1. Open the Fusion UI.
2. Navigate to **Querying** > **Query Profiles**.
3. Select the `_rules_simulator` query profile for your app.
   For example, if your app is called "Demo" then the name of the query profile is `Demo_rules_simulator`.



4. Modify the configuration as desired.
5. Click **Save**.

Configure the Rules Simulator Query Profile

How-to

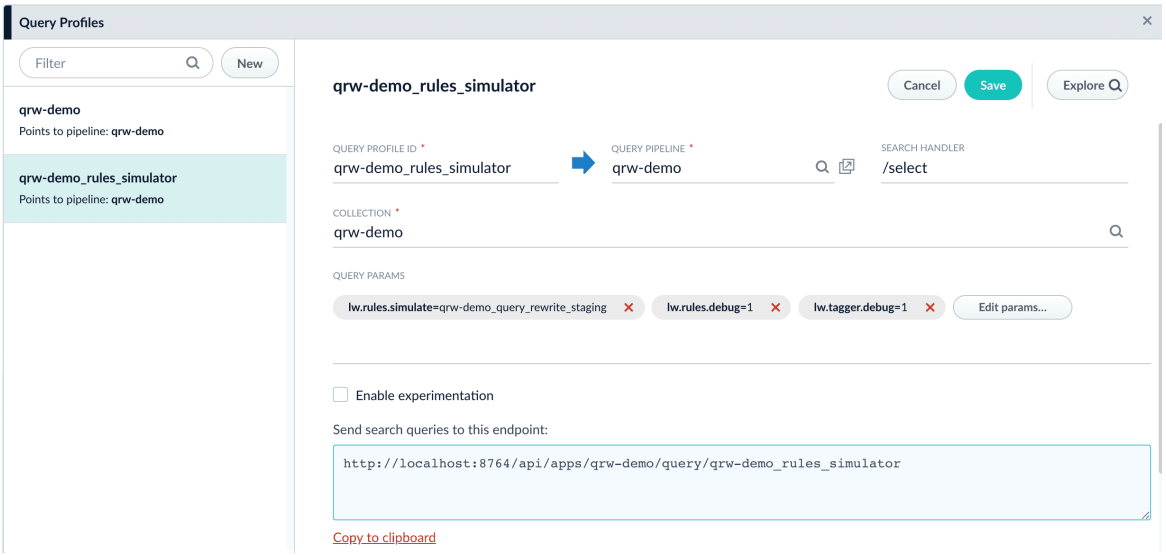
> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

Each app has a `_rules_simulator` query profile, configured to use the `_query_rewrite_staging` collection for *query rewrites* instead of the `_query_rewrite` collection. This profile is created automatically whenever a new app is created.

By default, this query profile points to your default query pipeline and collection. You can configure it to point to any pipeline or collection, for example when testing a new pipeline before it has been deployed.

How to change the query pipeline, collection, and query parameters used by the `_rules_simulator` query profile

1. Open the Fusion UI.
2. Navigate to **Querying** > **Query Profiles**.
3. Select the `_rules_simulator` query profile for your app.
   For example, if your app is called "Demo" then the name of the query profile is `Demo_rules_simulator`.



4. Modify the configuration as desired.
5. Click **Save**.

Configure the Smart Answers pipelines

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

Before beginning this procedure, train a machine learning model using either the *FAQ method* or the *cold start method*.

Regardless of how you set up your model, the deployment procedure is the same:

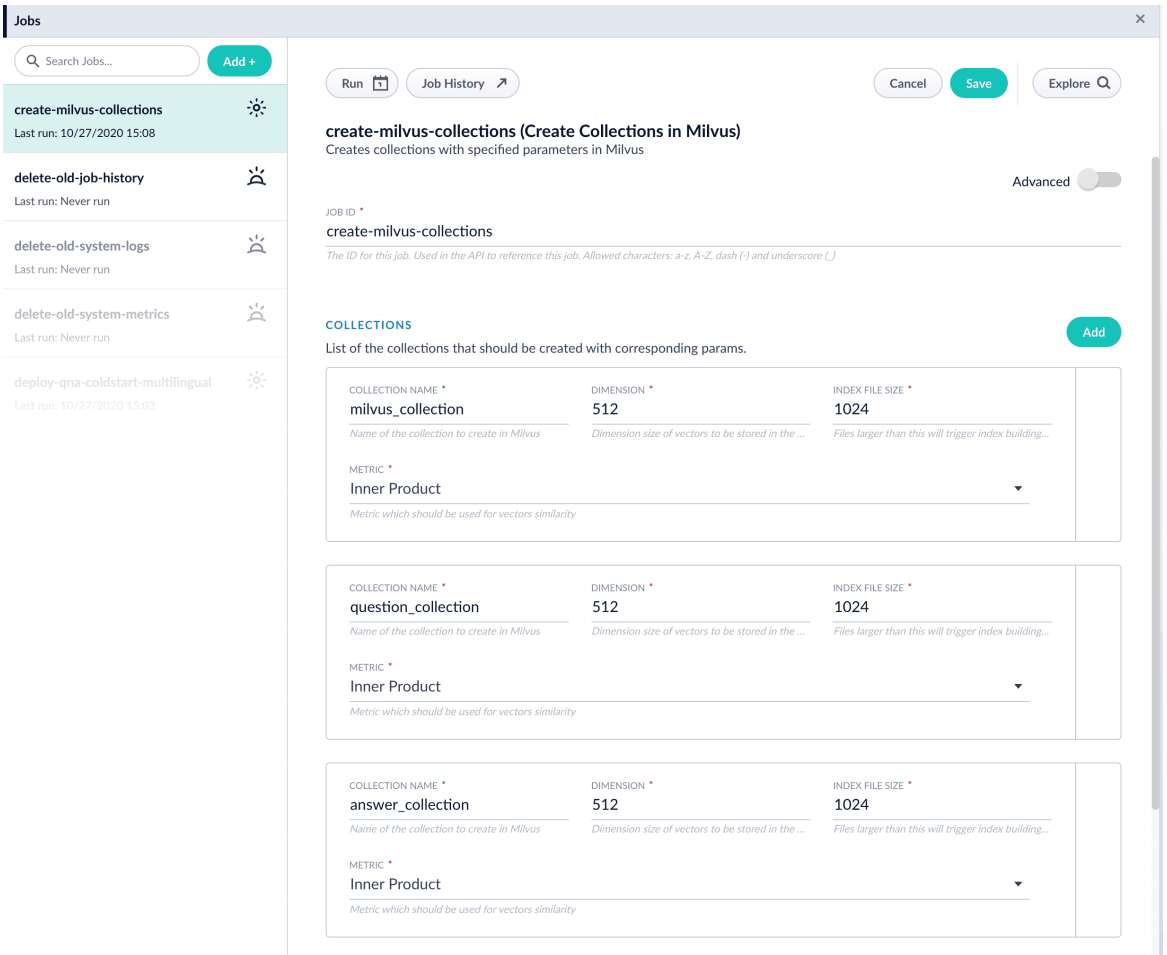
1. *Create the* `Milvus` *collection.*
2. *Configure the* `smart-answers` *index pipeline.*
3. *Configure the* `smart-answers` *query pipeline.*

See also *Best Practices* and *Advanced Model Training Configuration for Smart Answers*.

1. Create the Milvus collection

For complete details about job configuration options, see the *Create Collections in Milvus job*.

1. Navigate to **Collections** > **Jobs** > **Add +** and select **Create Collections in Milvus**.
2. Configure the job:
   a. Enter an ID for this job.
   b. Under **Collections**, click **Add**.
   c. Enter a collection name.
   d. In the **Dimension** field, enter the dimension size of vectors to store in this Milvus collection. The Dimension should match the size of the vectors returned by the encoding model. For example, the `Smart Answers Pre-trained Coldstart` models outputs vectors of 512 dimension size. Dimensionality of encoders trained by `Smart Answers Supervised Training` job depends on the provided parameters and printed in the training job logs. See *Smart Answers Supervised model training* for more details.
3. Click **Save**.
   The `Create Collections in Milvus` job can be used to create multiple collections at once. In this image, the first collection is used in the *indexing* and *query* steps. The other two collections are used in the *example*.



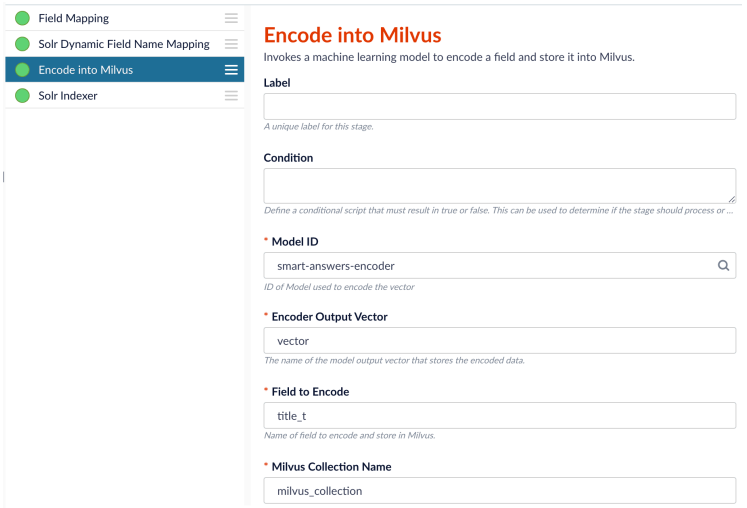
4. Click **Run** > **Start** to run the job.

2. Configure the index pipeline

1. Open the Index Workbench.
2. Load or create your datasource using the default smart-answers index pipeline.

3. Configure the *Encode into Milvus stage*:
   a. change the value of **Model ID** to match the model deployment name you chose when you *configured the model training job*.
   b. Change `Field to Encode` to the document field name to be processed and encoded into dense vectors.
   c. Ensure the `Encoder Output Vector` matches the output vector from the chosen model.
   d. Ensure the `Milvus Collection Name` matches the collection name created via the `Create Milvus Collection` job.

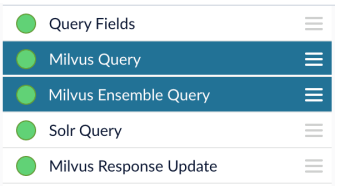
> **Tip**
>
> To test out your settings, turn on `Fail on Error` in the `Encode into Milvus` stage and Apply the changes. This will cause an error message to display if any settings need to be changed.



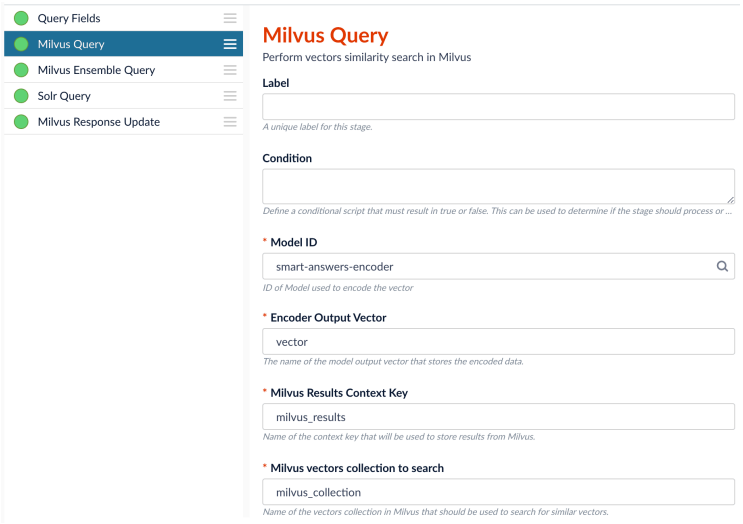
4. Save the datasource.
5. Index your data.

3. Configure the query pipeline

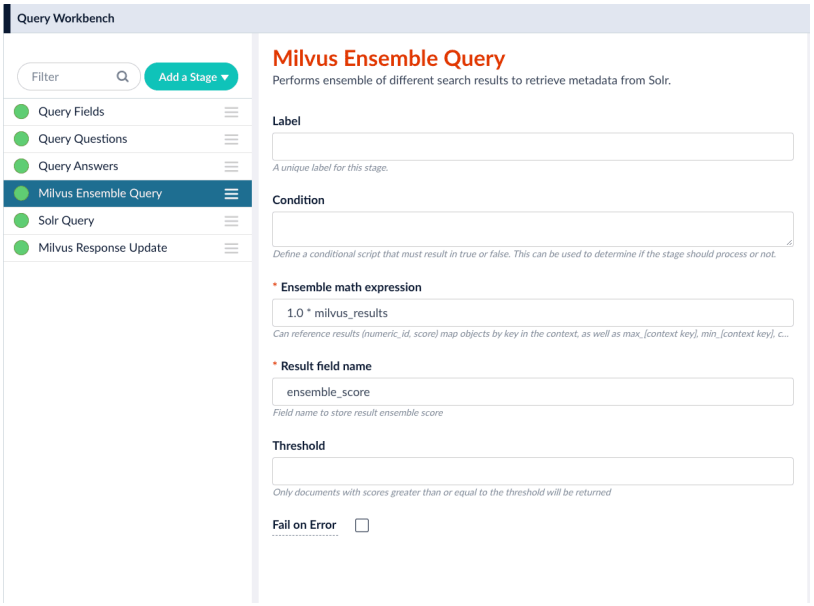
1. Open the Query Workbench.
2. Load the default smart-answers query pipeline.



3. Configure the *Milvus Query stage*:
   a. Change the **Model ID** value to match the model deployment name you chose when you *configured the model training job*.
   b. Ensure the `Encoder Output Vector` matches the output vector from the chosen model.
   c. Ensure the `Milvus Collection Name` matches the collection name created via the `Create Milvus Collection` job.
   d. `Milvus Results Context Key` can be changed as needed. It will be used in the Milvus Ensemble Query Stage to calculate the query score.



4. In the *Milvus Ensemble Query stage*, update the `Ensemble math expression` as needed based on your model and the name used in the prior stage for the storing the Milvus results.
   You can also set the `Threshold` so that the Milvus Ensemble Query Stage will only return items with a score **greater than** or **equal to** the configured value.



5. Save the query pipeline.

Pipeline Setup Example

Index and retrieve the question and answer together

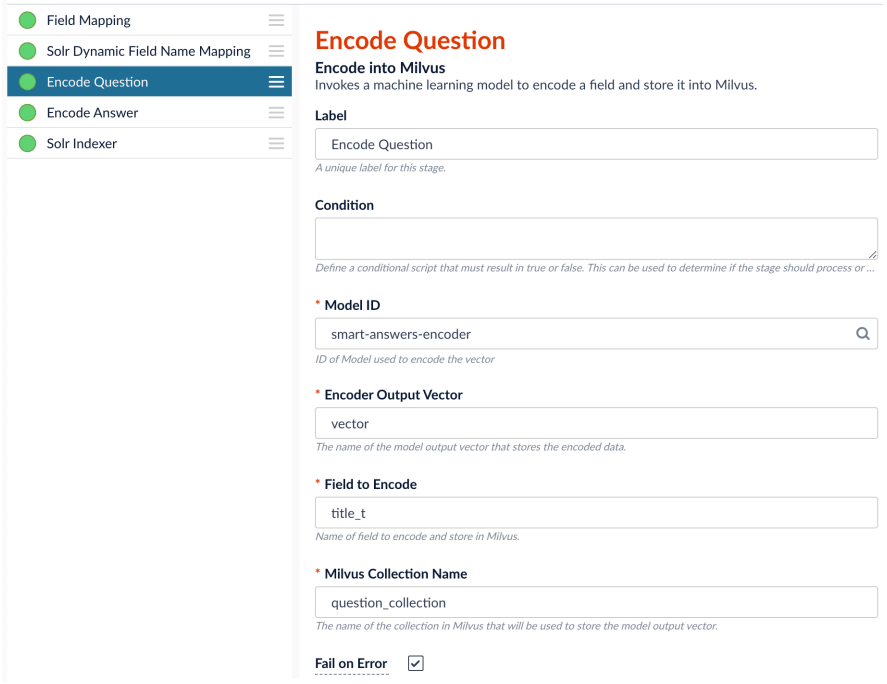
To show question and answer together in one document (that is, treat the question as the title and the answer as the description), you can index them together in the same document. You can still use the default `smart-answers` index and query pipelines with a few additional changes.

Prior to configuring the Smart Answers pipelines, use the `Create Milvus Collection` job to create two collections, `question_collection` and `answer_collection`, to store the encoded "questions" and the encoded "answers", respectively.

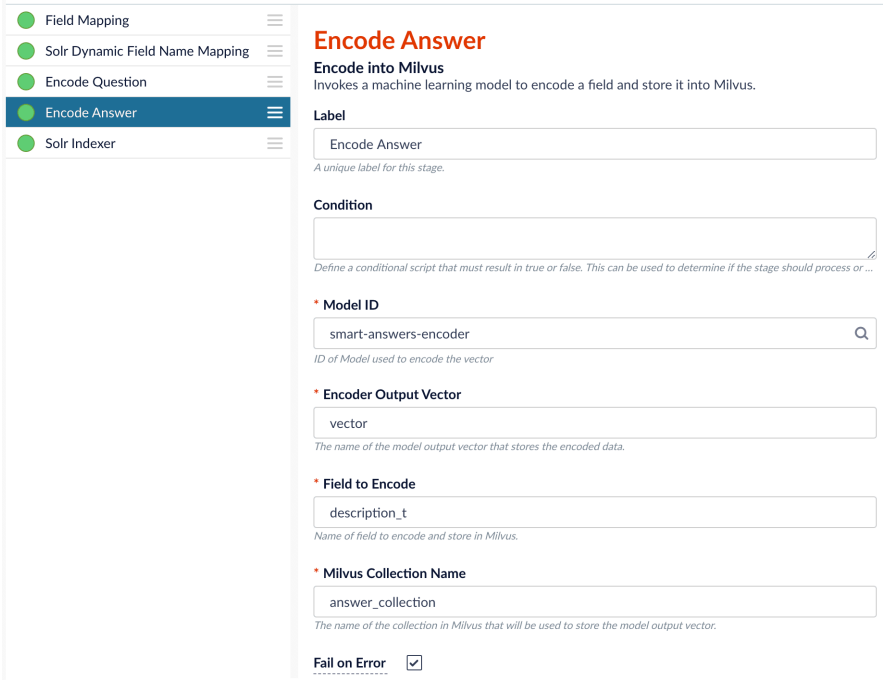
INDEX PIPELINE

As shown in the pictures below, you will need two Encode into Milvus stages, named Encode Question and Encode Answer respectively.

Encode Question (Encode Into Milvus) stage



Encode Answer (Encode Into Milvus) stage



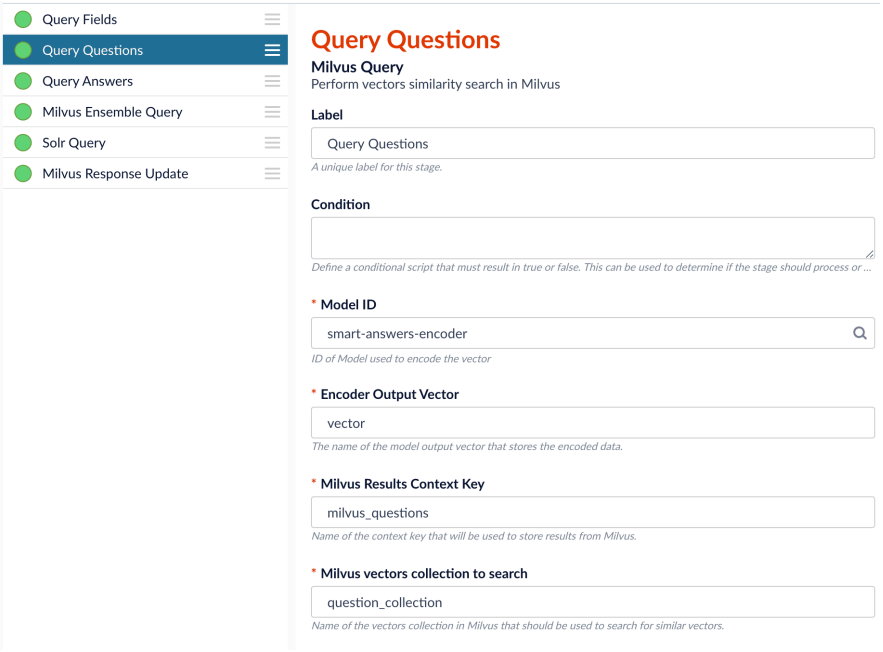
In the Encode Question stage, specify `Field to Encode` to be `title_t` and change the `Milvus Collection Name` to match the new Milvus collection, `question_collection`.

In the Encode Answer stage, specify `Field to Encode` to be `description_t` and change the `Milvus Collection Name` to match the new Milvus collection, `answer_collection`.

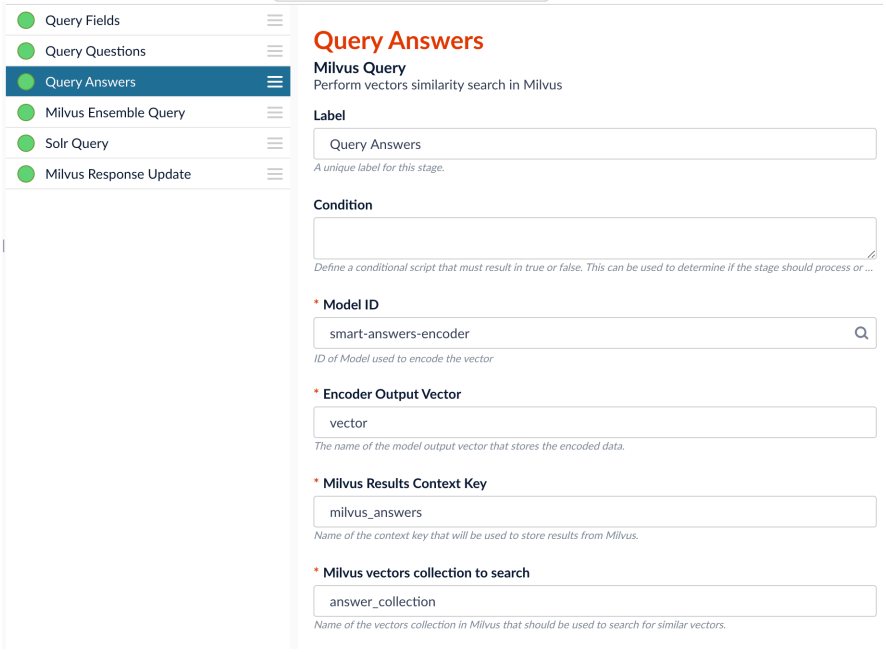
QUERY PIPELINE

Since we have two dense vectors generated during indexing, at query time we need to compute both query to question distance and query to answer distance. This can be set up as the pictures shown below with two Milvus Query Stages, one for each of the two Milvus collections. To store those two distances separately, the `Milvus Results Context Key` needs to be different in each of these two stages.

In the Query Questions stage, we set the `Milvus Results Context Key` to `milvus_questions` and the Milvus collection name to `question_collection`..Query Questions (Milvus Query) stage

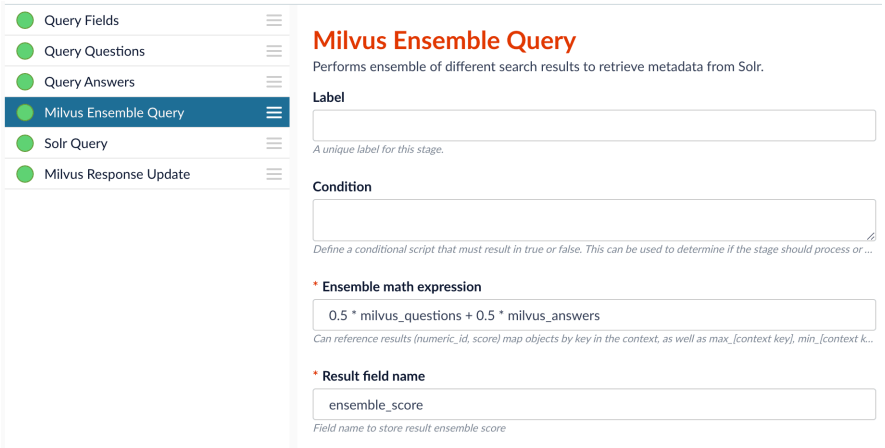


In the Query Answers stage, we set the `Milvus Results Context Key` to `milvus_answers` and the Milvus collection name to `answer_collection` .Query Answers (Milvus Query) stage



Now we can ensemble them together with the Milvus Ensemble Query Stage with the `Ensemble math expression` combining the results from the two query stages. If we want the question scores and answer scores weighted equally, we would use: `0.5 * milvus_questions + 0.5 * milvus_answers` . This is recommended especially when you have limited FAQ dataset and want to utilize both question and answer information.

Milvus Ensemble Query stage

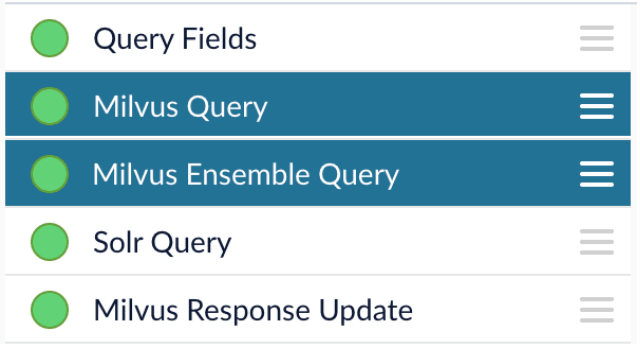


Evaluate the query pipeline

The *Evaluate QnA Pipeline job* evaluates the rankings of results from any Smart Answers pipeline and finds the best set of weights in the ensemble score. See *Evaluate a Smart Answers Pipeline* for setup instructions.

Detailed pipeline setup

Typically, you can use the *default pipelines* included with Managed Fusion. These pipelines now utilize Milvus to store encoded vectors and to calculate vector similarity. This topic provides information you can use to customize the Smart Answers pipelines. See also *Configure The Smart Answers Pipelines*.

"smart-answers" index pipeline		• *Encode into Milvus stage*
"smart-answers" query pipeline		• *Query Fields* • *Milvus Query* • *Milvus Ensemble Query* • *Milvus Response Update*

Create the Milvus collection

Prior to indexing data, the *Create Collections in Milvus job* can be used to create the Milvus collection(s) used by the Smart Answers pipelines.

- `Job ID`. A unique identifier for the job.
- `Collection Name`. A name for the Milvus collection you are creating. This name is used in both the Smart Answer Index and the Smart Answer Query pipelines.
- `Dimension`. The dimension size of the vectors to store in this Milvus collection. The Dimension should match the size of the vectors returned by the encryption model. For example, if the model was created with either the `Smart Answers Coldstart Training` job or the `Smart Answers`

`Supervised Training` job with the Model Base `word_en_300d_2M`, then the dimension would be 300.

- `Index file size`. Files with more documents than this will cause Milvus to build an index on this collection.
- `Metric`. The type of metric used to calculate vector similarity scores. `Inner Product` is recommended. It produces values between 0 and 1, where a higher value means higher similarity.

Index pipeline setup

Stages in the default "smart-answers" index pipeline

Only one custom index stage needs to be configured in your index pipeline, the *Encode into Milvus index stage*.

THE ENCODE INTO MILVUS INDEX STAGE

> **Tip**
>
> If you are using a dynamic schema, make sure this stage is added *after* the *Solr Dynamic Field Name Mapping stage*.

The *Encode into Milvus* index stage uses the specified model to encode the `Field to Encode` and store it in Milvus in the given Milvus collection. There are several required parameters:

- `Model ID`. The ID of the model.
- `Encoder Output Vector`. The name of the field that stores the compressed dense vectors output from the model. Default value: `vector`.
- `Field to Encode`. The text field to encode into a dense vector, such as `answer_t` or `body_t`.
- `Milvus Collection Name`. The name of the collection you created via the *Create Milvus Collection* job, which will store the dense vectors. When creating the collection you specify the type of **Metric** to use to calculate vector similarity. This stage can be used multiple times to encode additional fields, each into a different Milvus collection. See how to *index and retrieve the question and answer together*.

Query pipeline setup

THE QUERY FIELDS STAGE

The first stage is **Query Fields**. For more information see the *Query Fields stage*.

THE MILVUS QUERY STAGE

The *Milvus Query* stage encodes the query into a vector using the specified model. It then performs a vector similarity search against the specified Milvus collection and returns a list of the best document matches.

- `Model ID`. The ID of the model used when configuring the *model training job*.
- `Encoder Output Vector`. The name of the output vector from the specified model, which will contain the query encoded as a vector. Defaults to vector.
- `Milvus Collection Name`. The name of the collection that you used in the `Encode into Milvus` index stage to store the encoded vectors.
- `Milvus Results Context Key`. The name of the variable used to store the vector distances. It can be changed as needed. It will be used in the Milvus Ensemble Query Stage to calculate the query score for the document.

- **Number of Results**. The number of highest scoring results returned from Milvus. This stage would typically be used the same number of times that the **Encode into Milvus** index stage is used, each with a different Milvus collection and a different **Milvus Results Context Key**.

THE MILVUS ENSEMBLE QUERY STAGE

The *Milvus Ensemble Query* takes the results of the Milvus Query stage(s) and calculates the **ensemble score**, which is used to return the best matches.

- **Ensemble math expression**. The mathematical expression used to calculate the **ensemble score**. It should reference the value(s) variable name specified in the **Milvus Results Context Key** parameter in the Milvus Query stage.
- **Result field name**. The name of the field used to store the **ensemble score**. It defaults to **ensemble_score**.
- **Threshold** - A parameter that filters the stage results to remove items that fall below the configured score. Items with a score at, or above, the threshold will be returned.

THE MILVUS RESPONSE UPDATE QUERY STAGE

The *Milvus Response Update Query* stage does not need to be configured and can be skipped if desired. It inserts the Milvus values, including the **ensemble_score**, into each of the returned documents, which is particularly useful when there is more than one **Milvus Query Stage**. This stage needs to come after the **Solr Query** stage.

Short answer extraction

By default, the question-answering query pipelines return complete documents that answer questions. Optionally, you can extract just a paragraph, a sentence, or a few words that answer the question. See *Extract Short Answers from Longer Documents*.

Configure the Smart Answers Pipelines (5.3 and later)

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion*

Before beginning this procedure, train a machine learning model using either the *FAQ method* or the *cold start method*.

Regardless of how you set up your model, the deployment procedure is the same:

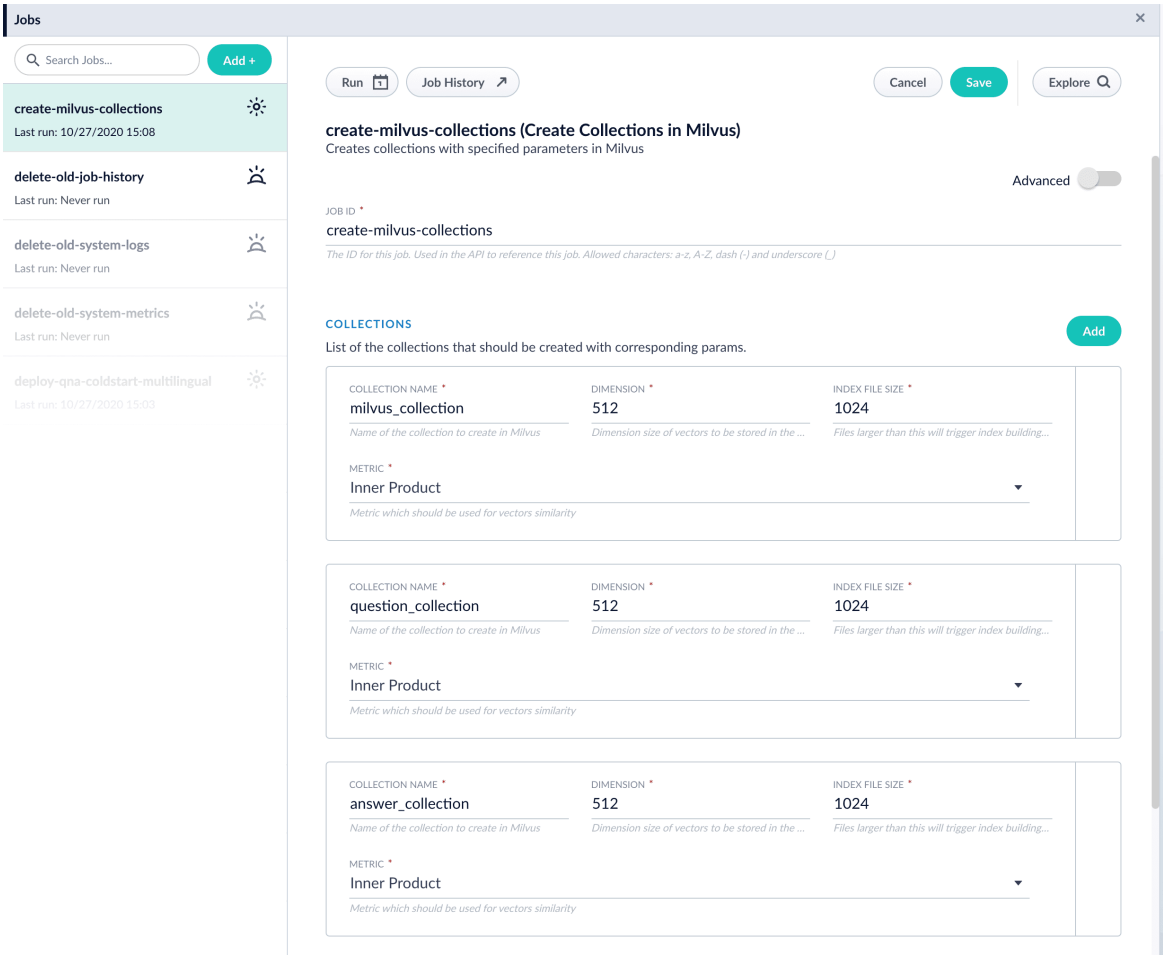
1. *Create the* `Milvus` *collection.*
2. *Configure the* `smart-answers` *index pipeline.*
3. *Configure the* `smart-answers` *query pipeline.*

See also *Best Practices*, *Advanced Model Training Configuration for Smart Answers*, and *Smart Answers Detailed Pipeline Setup*.

1. Create the Milvus collection

For complete details about job configuration options, see the *Create Collections in Milvus job*.

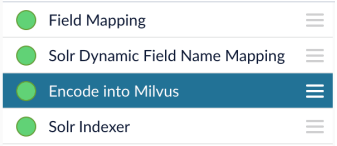
1. Navigate to **Collections** > **Jobs** > **Add +** and select **Create Collections in Milvus**.
2. Configure the job:
   a. Enter an ID for this job.
   b. Under **Collections**, click **Add**.
   c. Enter a collection name.
   d. In the **Dimension** field, enter the dimension size of vectors to store in this Milvus collection. The Dimension should match the size of the vectors returned by the encoding model. For example, the `Smart Answers Pre-trained Coldstart` models outputs vectors of 512 dimension size. Dimensionality of encoders trained by `Smart Answers Supervised Training` job depends on the provided parameters and printed in the training job logs. See *Smart Answers Supervised model training* for more details.
3. Click **Save**.
   The `Create Collections in Milvus` job can be used to create multiple collections at once. In this image, the first collection is used in the *indexing* and *query* steps. The other two collections are used in the *example*.



4. Click **Run** > **Start** to run the job.

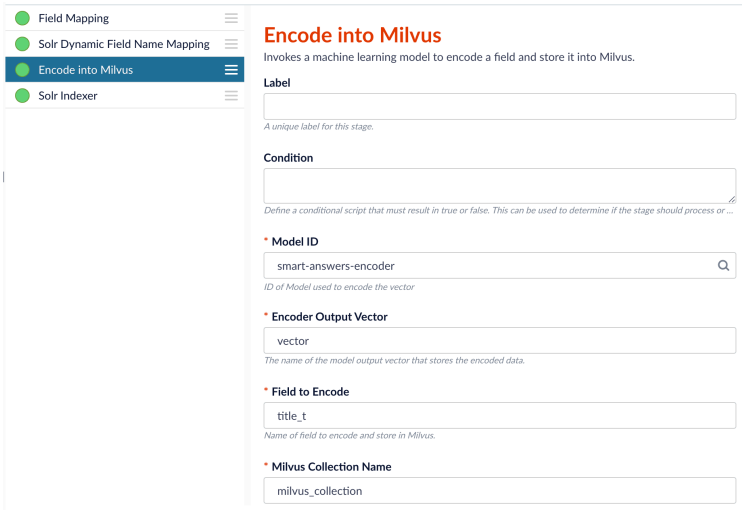
2. Configure the index pipeline

1. Open the Index Workbench.
2. Load or create your datasource using the default smart-answers index pipeline.



3. Configure the *Encode into Milvus stage*:
   a. change the value of **Model ID** to match the model deployment name you chose when you *configured the model training job*.
   b. Change `Field to Encode` to the document field name to be processed and encoded into dense vectors.
   c. Ensure the `Encoder Output Vector` matches the output vector from the chosen model.
   d. Ensure the `Milvus Collection Name` matches the collection name created via the `Create Milvus Collection` job.

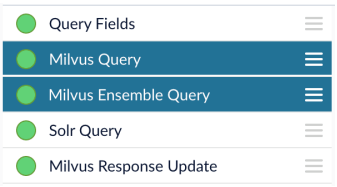
> **Tip**
>
> To test out your settings, turn on `Fail on Error` in the `Encode into Milvus` stage and Apply the changes. This will cause an error message to display if any settings need to be changed.



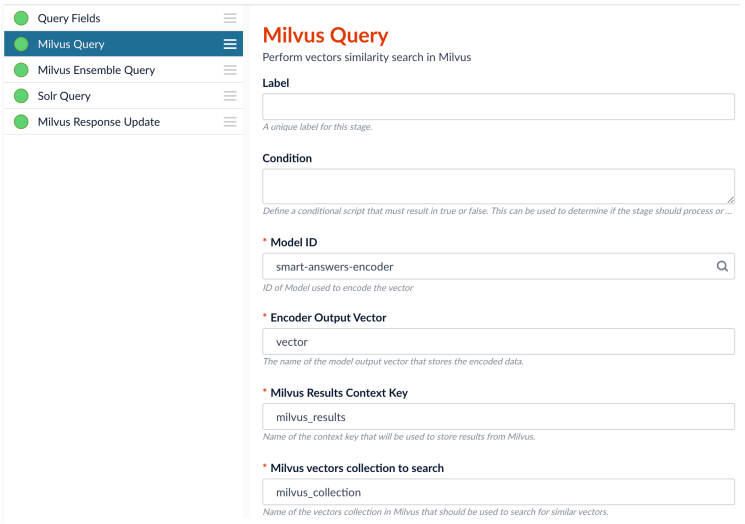
4. Save the datasource.
5. Index your data.

3. Configure the query pipeline

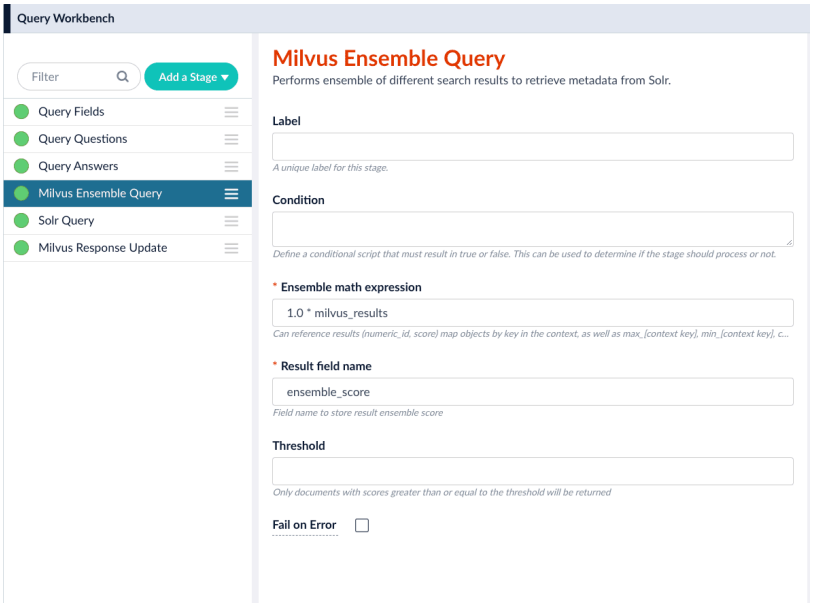
1. Open the Query Workbench.
2. Load the default smart-answers query pipeline.



3. Configure the *Milvus Query stage*:
   a. Change the **Model ID** value to match the model deployment name you chose when you *configured the model training job*.
   b. Ensure the `Encoder Output Vector` matches the output vector from the chosen model.
   c. Ensure the `Milvus Collection Name` matches the collection name created via the `Create Milvus Collection` job.
   d. `Milvus Results Context Key` can be changed as needed. It will be used in the Milvus Ensemble Query Stage to calculate the query score.



4. In the *Milvus Ensemble Query stage*, update the `Ensemble math expression` as needed based on your model and the name used in the prior stage for the storing the Milvus results.
   In versions 5.4 and later, you can also set the `Threshold` so that the Milvus Ensemble Query Stage will only return items with a score **greater than** or **equal to** the configured value.



5. Save the query pipeline.

Pipeline Setup Example

Index and retrieve the question and answer together

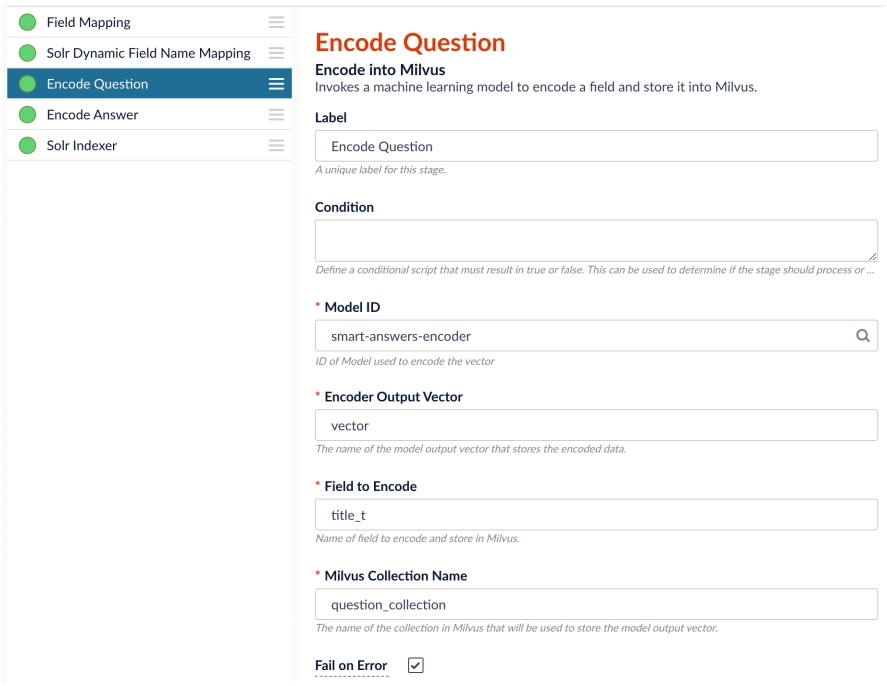
To show question and answer together in one document (that is, treat the question as the title and the answer as the description), you can index them together in the same document. You can still use the default `smart-answers` index and query pipelines with a few additional changes.

Prior to configuring the Smart Answers pipelines, use the `Create Milvus Collection` job to create two collections, `question_collection` and `answer_collection`, to store the encoded "questions" and the encoded "answers", respectively.

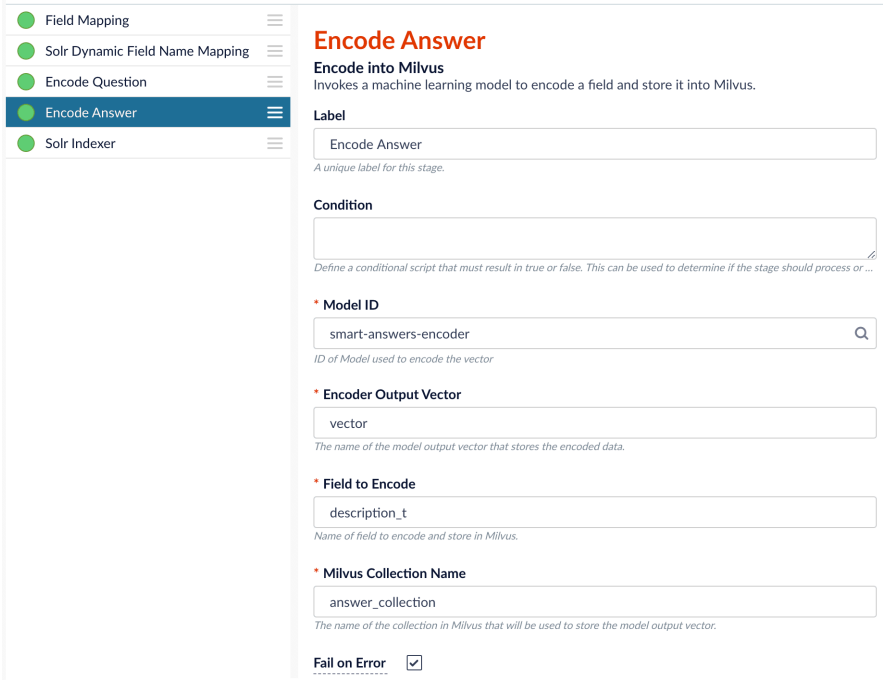
INDEX PIPELINE

As shown in the pictures below, you will need two Encode into Milvus stages, named Encode Question and Encode Answer respectively.

Encode Question (Encode Into Milvus) stage



Encode Answer (Encode Into Milvus) stage



In the Encode Question stage, specify `Field to Encode` to be `title_t` and change the `Milvus Collection Name` to match the new Milvus collection, `question_collection`.

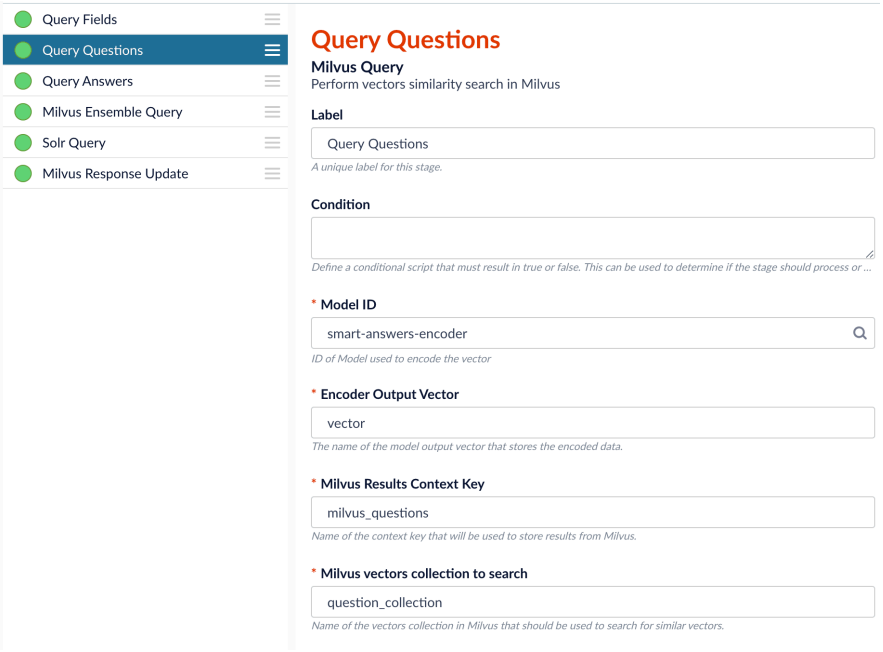
In the Encode Answer stage, specify `Field to Encode` to be `description_t` and change the `Milvus Collection Name` to match the new Milvus collection, `answer_collection`.

(For more detail, see *Smart Answers Detailed Pipeline Setup*.)

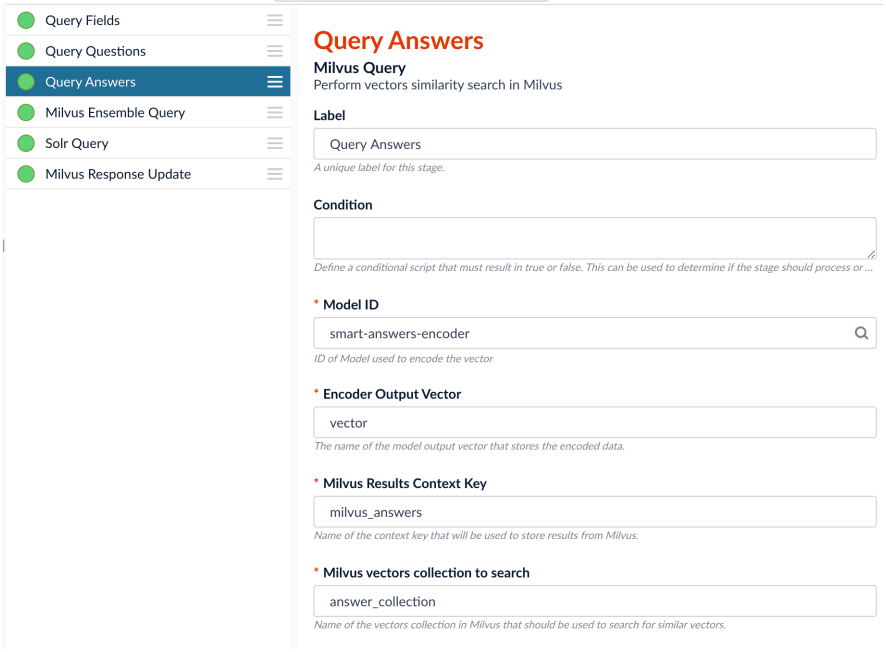
QUERY PIPELINE

Since we have two dense vectors generated during indexing, at query time we need to compute both query to question distance and query to answer distance. This can be set up as the pictures shown below with two Milvus Query Stages, one for each of the two Milvus collections. To store those two distances separately, the `Milvus Results Context Key` needs to be different in each of these two stages.

In the Query Questions stage, we set the `Milvus Results Context Key` to `milvus_questions` and the Milvus collection name to `question_collection`. .Query Questions (Milvus Query) stage

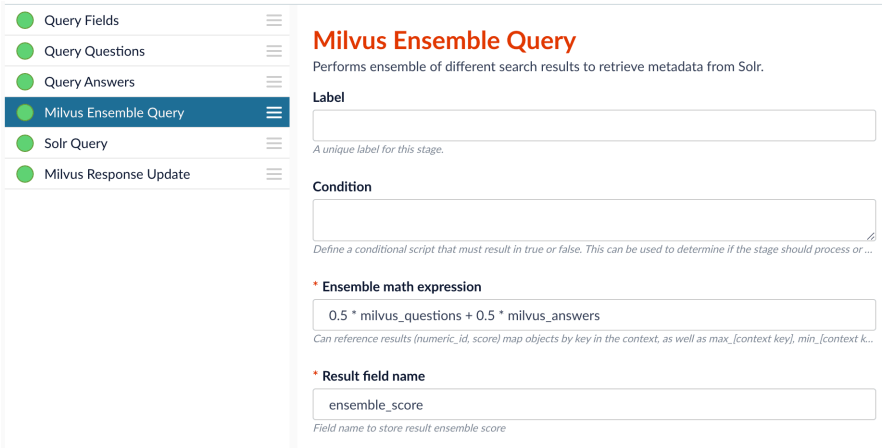


In the Query Answers stage, we set the `Milvus Results Context Key` to `milvus_answers` and the Milvus collection name to `answer_collection` .Query Answers (Milvus Query) stage



Now we can ensemble them together with the Milvus Ensemble Query Stage with the `Ensemble math expression` combining the results from the two query stages. If we want the question scores and answer scores weighted equally, we would use: `0.5 * milvus_questions + 0.5 * milvus_answers` . This is recommended especially when you have limited FAQ dataset and want to utilize both question and answer information.

Milvus Ensemble Query stage

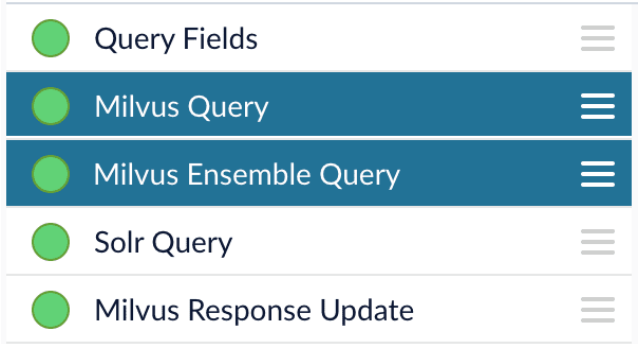


Evaluate the query pipeline

The *Evaluate QnA Pipeline job* evaluates the rankings of results from any Smart Answers pipeline and finds the best set of weights in the ensemble score. See *Evaluate a Smart Answers Pipeline* for setup instructions.

Detailed pipeline setup

Typically, you can use the *default pipelines* included with Fusion AI. These pipelines now utilize Milvus to store encoded vectors and to calculate vector similarity. This topic provides information you can use to customize the Smart Answers pipelines. See also *Configure The Smart Answers Pipelines*.

"smart-answers" index pipeline		• *Encode into Milvus stage*
"smart-answers" query pipeline		• *Query Fields* • *Milvus Query* • *Milvus Ensemble Query* • *Milvus Response Update*

Create the Milvus collection

Prior to indexing data, the *Create Collections in Milvus job* can be used to create the Milvus collection(s) used by the Smart Answers pipelines (see *Milvus overview*).

- `Job ID`. A unique identifier for the job.
- `Collection Name`. A name for the Milvus collection you are creating. This name is used in both the Smart Answer Index and the Smart Answer Query pipelines.
- `Dimension`. The dimension size of the vectors to store in this Milvus collection. The Dimension should match the size of the vectors returned by the encryption model. For example, if the model was created with either the `Smart Answers Coldstart Training` job or the `Smart Answers`

`Supervised Training` job with the Model Base `word_en_300d_2M`, then the dimension would be 300.

- `Index file size`. Files with more documents than this will cause Milvus to build an index on this collection.
- `Metric`. The type of metric used to calculate vector similarity scores. `Inner Product` is recommended. It produces values between 0 and 1, where a higher value means higher similarity.

Index pipeline setup

Stages in the default "smart-answers" index pipeline

Only one custom index stage needs to be configured in your index pipeline, the *Encode into Milvus index stage*.

THE ENCODE INTO MILVUS INDEX STAGE

> **Tip**
>
> If you are using a dynamic schema, make sure this stage is added *after* the *Solr Dynamic Field Name Mapping stage*.

The *Encode into Milvus* index stage uses the specified model to encode the `Field to Encode` and store it in Milvus in the given Milvus collection. There are several required parameters:

- `Model ID`. The ID of the model.
- `Encoder Output Vector`. The name of the field that stores the compressed dense vectors output from the model. Default value: `vector`.
- `Field to Encode`. The text field to encode into a dense vector, such as `answer_t` or `body_t`.
- `Milvus Collection Name`. The name of the collection you created via the *Create Milvus Collection* job, which will store the dense vectors. When creating the collection you specify the type of **Metric** to use to calculate vector similarity. This stage can be used multiple times to encode additional fields, each into a different Milvus collection. See how to *index and retrieve the question and answer together*.

Query pipeline setup

THE QUERY FIELDS STAGE

The first stage is **Query Fields**. For more information see the *Query Fields stage*.

THE MILVUS QUERY STAGE

The *Milvus Query* stage encodes the query into a vector using the specified model. It then performs a vector similarity search against the specified Milvus collection and returns a list of the best document matches.

- `Model ID`. The ID of the model used when configuring the *model training job*.
- `Encoder Output Vector`. The name of the output vector from the specified model, which will contain the query encoded as a vector. Defaults to vector.
- `Milvus Collection Name`. The name of the collection that you used in the `Encode into Milvus` index stage to store the encoded vectors.
- `Milvus Results Context Key`. The name of the variable used to store the vector distances. It can be changed as needed. It will be used in the Milvus Ensemble Query Stage to calculate the query score for the document.

- **Number of Results**. The number of highest scoring results returned from Milvus. This stage would typically be used the same number of times that the **Encode into Milvus** index stage is used, each with a different Milvus collection and a different **Milvus Results Context Key**.

THE MILVUS ENSEMBLE QUERY STAGE

The *Milvus Ensemble Query* takes the results of the Milvus Query stage(s) and calculates the **ensemble score**, which is used to return the best matches.

- **Ensemble math expression**. The mathematical expression used to calculate the **ensemble score**. It should reference the value(s) variable name specified in the **Milvus Results Context Key** parameter in the Milvus Query stage.
- **Result field name**. The name of the field used to store the **ensemble score**. It defaults to **ensemble_score**.
- **Threshold** - A parameter that filters the stage results to remove items that fall below the configured score. Items with a score at, or above, the threshold will be returned.

> **Note**
>
> The **Threshold** feature is only available in Fusion 5.4 and later.

THE MILVUS RESPONSE UPDATE QUERY STAGE

The *Milvus Response Update Query* stage does not need to be configured and can be skipped if desired. It inserts the Milvus values, including the **ensemble_score**, into each of the returned documents, which is particularly useful when there is more than one **Milvus Query Stage**. This stage needs to come after the **Solr Query** stage.

Short answer extraction

By default, the question-answering query pipelines return complete documents that answer questions. Optionally, you can extract just a paragraph, a sentence, or a few words that answer the question. See *Extract Short Answers from Longer Documents*.

Configure the Spark History Server

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion*

For related topics, see *Spark Operations*.

Recommended configuration

For Fusion, configure the Spark History Server to store and read Spark logs in cloud storage. For installations on Google Kubernetes Engine, set these keys in the `values.yaml` file:

```
gcs: enableGCS: true secret: history-secrets  1  key: sparkhistory.json
 1  logDirectory: gs://[BUCKET_NAME] service:  2  type: ClusterIP port:
number: 18080 pvc: enablePVC: false nfs: enableExampleNFS: false  3
```

1	The `key` and `secret` fields provide the Spark History Server with the details of where it can find an account with access to the Google Cloud Storage bucket given in `logDirectory`. Later examples show how to set up a new service account that's shared between the Spark History Server and the Spark driver/executors for both viewing and writing logs.
2	By default, the Spark History Server Helm chart creates an external LoadBalancer, exposing it to outside access. In this example, the `service` key overrides the default. The Spark History Server is set up on an internal IP within your cluster only and is not exposed externally. Later examples show how to access the Spark History Server.
3	The `nfs.enableExampleNFS` option turns off the unneeded default NFS server set up by the Spark History Server.

To give the Spark History Server access to the Google Cloud Storage bucket where the logs are kept:

1. Use `gcloud` to create a new service account:

```
export ACCOUNT_NAME=sparkhistory export GCP_PROJECT_ID=
[PROJECT_ID] gcloud iam service-accounts create ${ACCOUNT_NAME}
--display-name "${ACCOUNT_NAME}"
```

> **Note**
>
> If you have an existing service account you wish to use instead, you can skip the `create` command, though you will still need to create the JSON key pair and ensure that the existing account can read and write to the log bucket.

2. Use `keys create` to create a JSON key pair, and upload it to your cluster as a Kubernetes secret:

```
gcloud iam service-accounts keys create "${ACCOUNT_NAME}.json" --iam-account "${ACCOUNT_NAME}@${GCP_PROJECT_ID}.iam.gserviceaccount.com"
```

3. Give the service account the `storage/admin` role, allowing it to perform "create" and "view" operations:

```
gcloud projects add-iam-policy-binding ${GCP_PROJECT_ID} --member "serviceAccount:${ACCOUNT_NAME}@${GCP_PROJECT_ID}.iam.gserviceaccount.com" --role roles/storage.admin
```

4. Run the `gsutil` command to apply the service account to your chosen bucket.

```
gsutil iam ch serviceAccount:${ACCOUNT_NAME}@${GCP_PROJECT_ID}.iam.gserviceaccount.com:objectAdmin gs://[BUCKET_NAME]
```

5. Upload the JSON key pair into the cluster as a secret:

```
kubectl -n [NAMESPACE] create secret generic history-secrets --from-file=sparkhistory.json
```

The Spark History Server can now be installed with `helm install [namespace]-spark-history-server stable/spark-history-server --values values.yaml`.

Other configurations

Azure

The Azure configuration process is similar Google Kubernetes Engine. However, logs are stored in Azure Blob Storage, and you can use SAS token or key access.

```
echo "your-storage-account-name" >> azure-storage-account-name echo
"your-container-name" >> azure-blob-container-name echo "your-azure-
blob-sas-key" >> azure-blob-sas-key  1  kubectl create secret generic
azure-secrets --from-file=azure-storage-account-name --from-
file=azure-blob-container-name [--from-file=azure-blob-sas-key | --
from-file=azure-storage-account-key]
```

1	This line is used to authenticate with a SAS token. Replace the line with `echo "your-azure-storage-account-key" >> azure-storage-account-key` to use a storage account key instead.

To use SAS token access, the **`values.yaml`** file resembles the following:

```
wasbs: enableWASBS: true secret: azure-secrets sasKeyName: azure-blob-
sas-key storageAccountNameKeyName: azure-storage-account-name
containerKeyName: azure-blob-container-name logDirectory:
[BUCKET_NAME]
```

For non-SAS access, the **`values.yaml`** file resembles the following:

```
wasbs: enableWASBS: true secret: azure-secrets sasKeyMode: false
storageAccountKeyName: azure-storage-account-key
storageAccountNameKeyName: azure-storage-account-name
containerKeyName: azure-blob-container-name logDirectory:
[BUCKET_NAME]
```

Amazon Web Services

In AWS, you can use IAM roles or an access/secret key pair. The use of AWS IAM roles is preferred over using an access/secret key pair, but both options are described.

```
aws iam list-access-keys --user-name your-user-name --output text |
awk '{print $2}' >> aws-access-key echo "your-aws-secret-key" >> aws-
secret-key kubectl create secret generic aws-secrets --from-file=aws-
access-key --from-file=aws-secret-key
```

For IAM, the `values.yaml` file resembles the following:

```
s3: enableS3: true logDirectory: s3a://[BUCKET_NAME]
```

Note

The `values.yaml` file uses the Hadoop `s3a://` link instead of `s3://`.

For an access/secret pair, add the secret:

```
s3: enableS3: true enableIAM: false accessKeyName: aws-access-key
secretKeyName: aws-secret-key logDirectory: s3a://[BUCKET_NAME]
```

Configuring Spark

After starting the Spark History Server, update the config map for Fusion's job-launcher service so it can write logs to the same location that Spark History Server is reading from.

In this example, Fusion is installed into a namespace called `sparkhistory`.

1. Before editing the config map, make a copy of the existing settings in case you need to revert the changes.

```
kubectl get cm -n [NAMESPACE] sparkhistory-job-launcher -o yaml
> sparkhistory-job-launcher.yaml
```

2. Edit the config map to write the logs to the same Google Cloud Storage bucket we configured the Spark History Server to read from.

```
kubectl edit cm -n [NAMESPACE] sparkhistory-job-launcher
```

3. Update the **spark** key with the new YAML settings below:

```
spark: hadoop: fs: AbstractFileSystem: gs: impl:
com.google.cloud.hadoop.fs.gcs.GoogleHadoopFS gs: impl:
com.google.cloud.hadoop.fs.gcs.GoogleHadoopFileSystem google:
cloud: auth: service: account: json: keyfile: /etc/history-
secrets/[ACCOUNT_NAME].json eventLog: enabled: true compress:
true dir: gs://[BUCKET_NAME] … kubernetes: driver: secrets:
history-secrets: /etc/history-secrets container: … executor:
secrets: history-secrets: /etc/history-secrets container: … …
```

The YAML settings inform Spark of the location of the secret and the settings that specify the location of the Spark **eventLog**. The settings also inform Spark how to access GCS with the **spark.hadoop.fs.AbstractFileSystem.gs.impl** and **spark.hadoop.fs.gs.impl** keys.

4. Delete the **job-launcher** pod. The new **job-launcher** pod will apply the new configuration to later jobs.

Configure the Suggest Component

How-to

> Compatible versions: 4.2, 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion Server*, *Fusion*

1. Choose a suggester implementation from the Solr Suggester documentation, and define it in the **solrconfig.xml** file.

Tip

You can edit the **solrconfig.xml** file inside of the Fusion Admin UI. Navigate to **System > Solr Config**, and select the **solrconfig.xml** file to begin.

In this example, create a suggester called **mySuggester** in **solrconfig.xml** using the `AnalyzingInfixLookupFactory`. Define `description` as the suggester's target field:

```
<searchComponent name="suggest" class="solr.SuggestComponent">
<lst name="suggester"> <str name="name">mySuggester</str> <str
name="lookupImpl">AnalyzingInfixLookupFactory</str> <str
name="dictionaryImpl">DocumentDictionaryFactory</str> <str
name="field">description</str> <str
name="suggestAnalyzerFieldType">string</str> <str
name="buildOnStartup">false</str> </lst> </searchComponent>
```

2. Ensure that the field used for autosuggest is correctly defined in the schema:

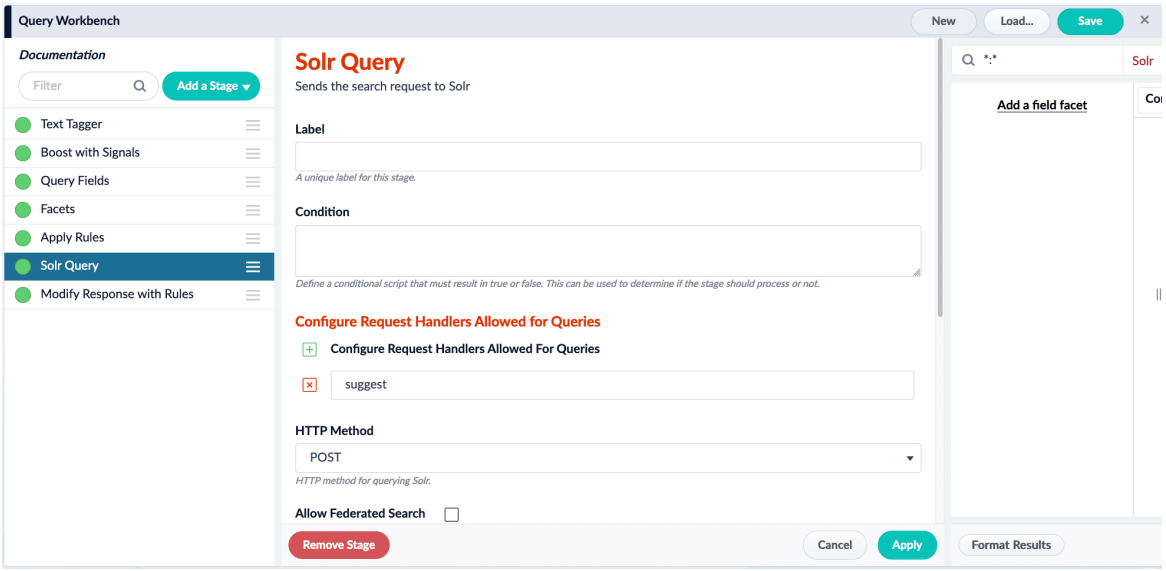
```
<field name="description" type="text_general" stored="true"
indexed="true" multivalued="false" />
```

3. Once modified, upload the custom configuration files to ZooKeeper using ZooKeeper's command line interface.

Use the zkcli command `putfile` to replace an existing ZooKeeper configuration file. Typically, `CONFIG_NAME` will likely represent the name of the collection. Alternatively, the name represents a custom configuration.

```
https://FUSION_HOST:FUSION_PORT/apps/solr-
dist/server/scripts/cloud-scripts/zkcli.sh -z localhost:2181 -
cmd putfile /configs/CONFIG_NAME/solrconfig.xml
/Desktop/solrconfig.xml
```

4. To enable autosuggest in Fusion, add the `suggest` request handler to the allowed request handlers in the Solr Query stage of the query pipeline.



Build the suggester should using `…/suggest?suggest.build=true`. For example:

```
http://localhost:8764/api/query-
pipelines/PIPELINE/collections/COLLECTION_NAME/suggest?
suggest.build=true
```

5. Start searching!

```
http://localhost:8764/api/query-pipelines/PIPELINE/collections/COLLECTION_NAME/suggest?suggest.q=ca...
```

Configure the Suggest Component

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

1. Choose a suggester implementation from the Solr Suggester documentation, and define it in the **solrconfig.xml** file.

Tip

You can edit the **solrconfig.xml** file inside of the Managed Fusion Admin UI. Navigate to **System > Solr Config**, and select the **solrconfig.xml** file to begin.

In this example, create a suggester called **mySuggester** in **solrconfig.xml** using the `AnalyzingInfixLookupFactory`. Define `description` as the suggester's target field:

```
<searchComponent name="suggest" class="solr.SuggestComponent">
<lst name="suggester"> <str name="name">mySuggester</str> <str
name="lookupImpl">AnalyzingInfixLookupFactory</str> <str
name="dictionaryImpl">DocumentDictionaryFactory</str> <str
name="field">description</str> <str
name="suggestAnalyzerFieldType">string</str> <str
name="buildOnStartup">false</str> </lst> </searchComponent>
```

2. Ensure that the field used for autosuggest is correctly defined in the schema:

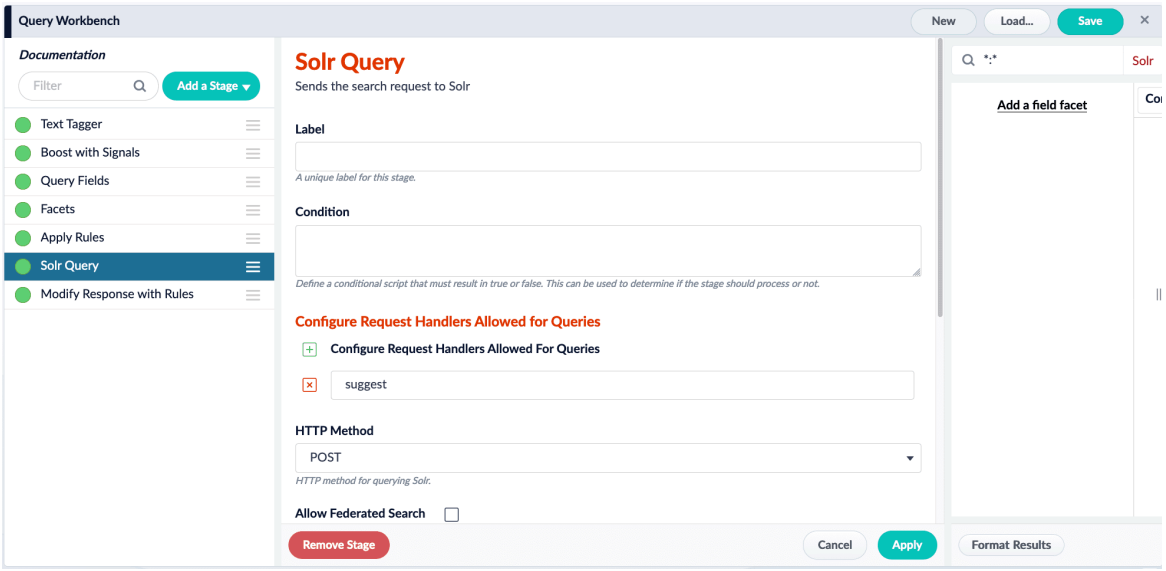
```
<field name="description" type="text_general" stored="true"
indexed="true" multivalued="false" />
```

3. Once modified, submit a request at Lucidworks support and ask them to upload the custom configuration files to ZooKeeper using ZooKeeper's command line interface.

Use the zkcli command `putfile` to replace an existing ZooKeeper configuration file. Typically, `CONFIG_NAME` will likely represent the name of the collection. Alternatively, the name represents a custom configuration.

```
https://EXAMPLE_COMPANY.b.lucidworks.cloud:6764/apps/solr-
dist/server/scripts/cloud-scripts/zkcli.sh -z localhost:2181 -
cmd putfile /configs/CONFIG_NAME/solrconfig.xml
/Desktop/solrconfig.xml
```

4. To enable autosuggest in Managed Fusion, add the `suggest` request handler to the allowed request handlers in the Solr Query stage of the query pipeline.



Build the suggester using `…/suggest?suggest.build=true`. For example:

```
https://EXAMPLE_COMPANY.b.lucidworks.cloud:6764/api/query-
pipelines/PIPELINE/collections/COLLECTION_NAME/suggest?
suggest.build=true
```

5. Start searching!

```
https://EXAMPLE_COMPANY.b.lucidworks.cloud:6764/api/query-pipelines/PIPELINE/collections/COLLECTION_NAME/suggest?suggest.q=ca...
```

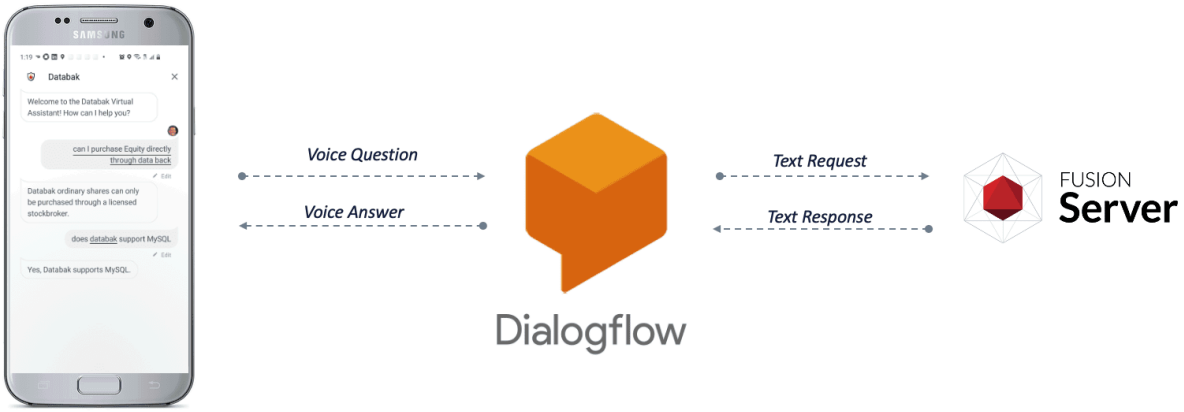
Configure Voice-Assisted Search

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

You can implement a voice chatbot using *Smart Answers* to integrate your search platform with a voice interface application. **This procedure uses Google Dialogflow as an example**, but you can use other voice interface platforms that support the following:

- Speech-to-text translation
  Translate the customer's speech into a text-based query for Fusion.
- Workflow routing
  Route queries to your Smart Answers query profile.
- Text-to speech translation
  Translate Fusion's text response into speech for the customer.



The high-level steps will be similar for any voice interface application:

1. Enable Smart Answers on your Fusion platform.
2. Configure your voice interface application to perform speech-to-text on voice queries.
3. Configure your voice interface application to route queries to the Fusion query profile associated with your Smart Answers query pipeline.
4. Configure your voice interface application to perform text-to-speech on Fusion's responses.

In this example procedure, we provide Javascript code that you can copy and customize, which performs the routing to your Fusion cluster.

1. Enable Smart Answers

To enable Smart Answers, you need a trained machine learning model, a question-and-answer body of content, and a query pipeline configured to apply the machine learning model to the content.

See *Smart Answers: Getting started* for detailed steps to enable Smart Answers.

2. Perform speech-to-text on voice queries

In the case of Google Dialogflow, this functionality is enabled by default whenever you create a Dialogflow agent. See your voice interface platform's documentation to find out how to enable speech-to-text for your application.

3. Route voice queries to Fusion

The procedure below shows you how to configure a Google Dialogflow agent as an example to demonstrate one possible configuration for routing voice queries to Fusion. For other voice interface platforms, check your documentation.

This example configures the agent so that all intents use a webhook. A code sample is provided.

1. Make sure you have created a billing account for Google Dialogflow, in order to enable all the required features.
2. *Download the* `voice.js` *file* and update the variables below to match your Fusion environment:

```
const FUSION_APP_NAME = 'YourAppName'; const
FUSION_QUERY_PROFILE_NAME = 'QueryProfileName'; const
FUSION_ADMIN_URL = 'https://EXAMPLE_COMPANY.b.lucidworks.cloud';
const FUSION_ADMIN_USERNAME = 'username'; const
FUSION_ADMIN_PASSWORD = 'password'; const FUSION_CREDENTIALS =
'Basic xxxxxxxsssssssssssss';
```

If your Fusion response field is not called `answer_t`, then you also need to update the field name in this line of the `voice.js` file:

```
agent.add(result.data.response.docs[0].answer_t);
```

3. In your Google Dialogflow agent interface, navigate to **Intents**.
   A newly-created Google Dialogflow agent contains two default intents:
   - Default Fallback Intent
   - Default Welcome Intent

> **Note**
>
> The `voice.js` code includes a handler for each of the default intents, plus "Try again" and "Yes" intents (which you will create below). If your application includes additional intents, you may need to add those to the code.

4. Navigate to **Intents** > **Default Fallback Intent** and configure it as follows:
   a. Delete all of the items in the **Context**, **Events**, **Training Phrases**, **Actions**, and **Response** sections.
   b. Under **Fulfillment**, select **Enable webhook call for this intent**.

   c. Click **Save**.

5. Click **Default Welcome Intent** and configure it as follows:
   a. Delete all of the items in the **Training Phrases** and **Responses** sections.
   b. Under **Fulfillment**, select **Enable webhook call for this intent**.
   c. Click **Save**.

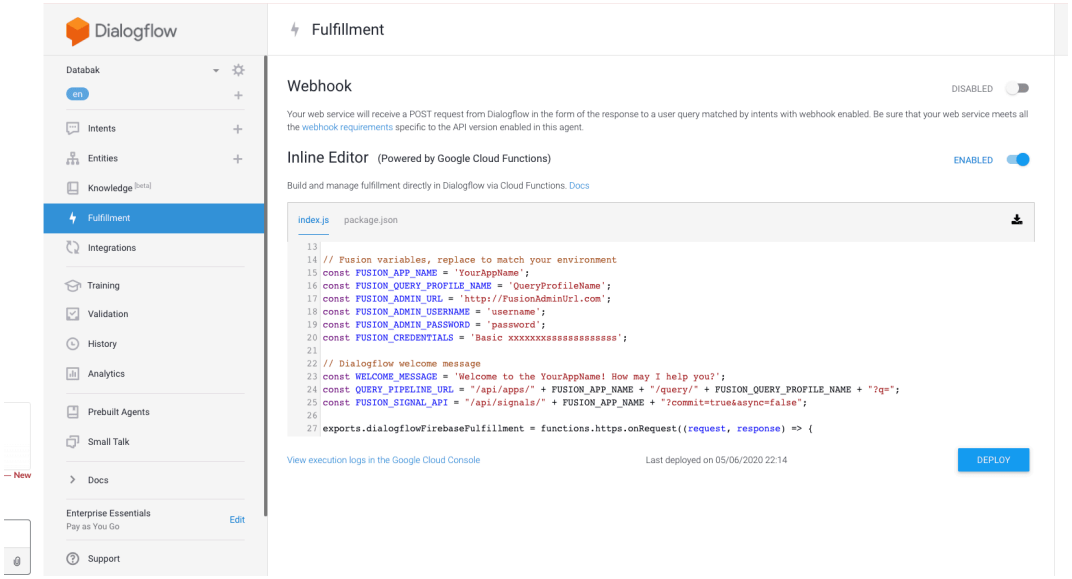
6. Navigate to **Intents** > **Create Intent**, call the new intent "Try again" (case-sensitive), and configure it as follows:
   a. Delete all of the items in the **Context**, **Events**, **Actions**, and **Response** sections.
   b. Under **Training Phrases**, add "Try again" (case-sensitive).

c. Under **Fulfillment**, select **Enable webhook call for this intent**.
d. Click **Save**.

7. Navigate to **Intents** > **Create Intent**, call the new intent "Yes" (case-sensitive), and configure it as follows:
   a. Delete all of the items in the **Context**, **Events**, **Actions**, and **Response** sections.
   b. Under **Training Phrases**, add the following phrases (case-sensitive):
      - "Try again"
      - "Yep"
      - "Perfect"

   c. Under **Fulfillment**, select **Enable webhook call for this intent**.
   d. Click **Save**.

8. Navigate to **fulfillment** and configure it as follows:
   a. Select **Enable Inline Editor**.
   b. Overwrite the default code with the `voice.js` code that you customized in step 2 above.



c. Click **Deploy**.

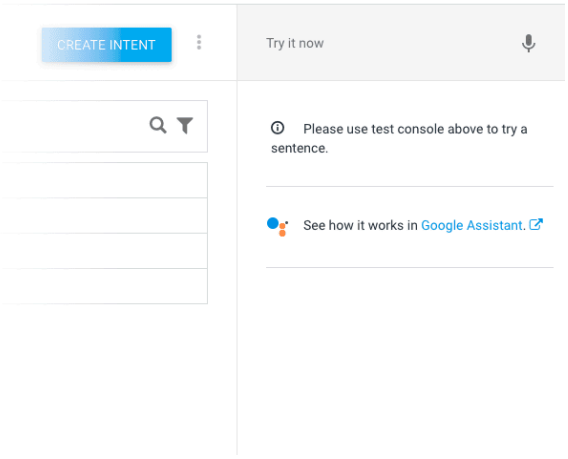
> **Note**
>
> In order to enable external API calls to Fusion, you must upgrade your Google Firebase billing account to the "Pay as you go" plan. Google Firebase is also where you can view the console logs for these API calls.

4. Perform text-to-speech on Fusion's responses

1. In your Google Dialogflow agent interface, click the gear icon next to your agent's name.
2. Under **Automatic Spell Correction**, enable **Allow ML to correct spelling of query during request processing**.
3. Under **Text to speech**, select **Enable Automatic Text to Speech**.
4. Click **Save**.

5. Test your configuration

1. In your Google Dialogflow agent interface, click the **Google Assistant** link in the right-hand sidebar.



2. Click **Talk to <YOUR AGENT NAME>**.
3. Click the microphone and ask a question.
   If you do not receive a response that makes sense, check the logs in Google Firebase for information you can use to troubleshoot your configuration.

Tip

Once your voice chatbot is working, you can click **Integrations** to access a pre-built widget that you can paste into the code for your front-end application.

See the Google Dialogflow documentation for more details.

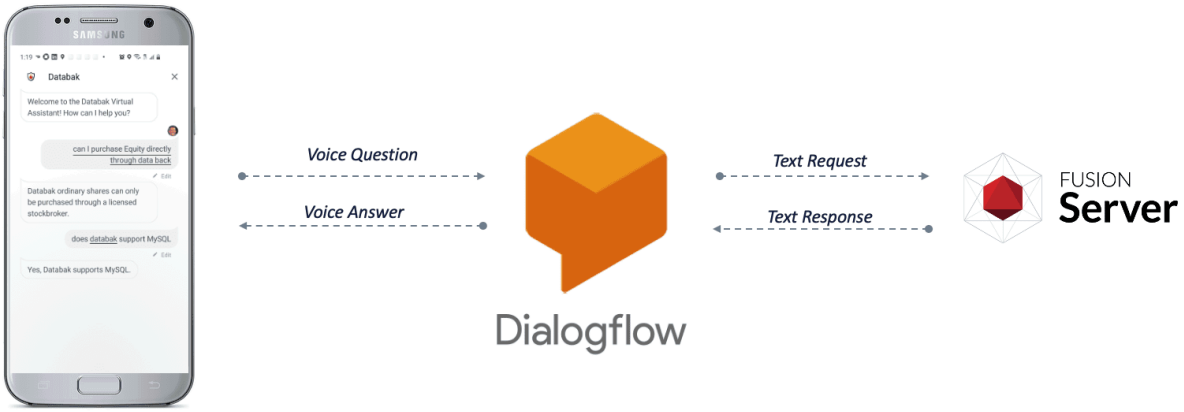
Configure Voice-Assisted Search

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion*

You can implement a voice chatbot using *Smart Answers* to integrate your search platform with a voice interface application. **This procedure uses Google Dialogflow as an example**, but you can use other voice interface platforms that support the following:

- Speech-to-text translation
  Translate the customer's speech into a text-based query for Fusion.
- Workflow routing
  Route queries to your Smart Answers query profile.
- Text-to speech translation
  Translate Fusion's text response into speech for the customer.



The high-level steps will be similar for any voice interface application:

1. Enable Smart Answers on your Fusion platform.
2. Configure your voice interface application to perform speech-to-text on voice queries.
3. Configure your voice interface application to route queries to the Fusion query profile associated with your Smart Answers query pipeline.
4. Configure your voice interface application to perform text-to-speech on Fusion's responses.

In this example procedure, we provide Javascript code that you can copy and customize, which performs the routing to your Fusion cluster.

1. Enable Smart Answers

To enable Smart Answers, you need a trained machine learning model, a question-and-answer body of content, and a query pipeline configured to apply the machine learning model to the content.

See *Smart Answers: Getting started* for detailed steps to enable Smart Answers.

2. Perform speech-to-text on voice queries

In the case of Google Dialogflow, this functionality is enabled by default whenever you create a Dialogflow agent. See your voice interface platform's documentation to find out how to enable speech-to-text for your application.

3. Route voice queries to Fusion

The procedure below shows you how to configure a Google Dialogflow agent as an example to demonstrate one possible configuration for routing voice queries to Fusion. For other voice interface platforms, check your documentation.

In this example, we will configure the agent so that all intents use a webhook, for which we provide a code sample.

1. Make sure you have created a billing account for Google Dialogflow, in order to enable all the required features.
2. *Download the* `voice.js` *file* and update the variables below to match your Fusion environment:

```
const FUSION_APP_NAME = 'YourAppName'; const
FUSION_QUERY_PROFILE_NAME = 'QueryProfileName'; const
FUSION_ADMIN_URL = 'http://FusionAdminUrl.com'; const
FUSION_ADMIN_USERNAME = 'username'; const FUSION_ADMIN_PASSWORD
= 'password'; const FUSION_CREDENTIALS = 'Basic
xxxxxxxsssssssssssss';
```

Note

The `FUSION_ADMIN_URL = 'http://FusionAdminUrl.com'` variable is also referred to as `FUSION_ADMIN_URL = 'http://FUSION_HOST:FUSION_PORT'`.

If your Fusion response field is not called `answer_t`, then you also need to update the field name in this line of the `voice.js` file:

```
agent.add(result.data.response.docs[0].answer_t);
```

3. In your Google Dialogflow agent interface, navigate to **Intents**.
   A newly-created Google Dialogflow agent contains two default intents:
   - Default Fallback Intent
   - Default Welcome Intent

Note

The `voice.js` code includes a handler for each of the default intents, plus "Try again" and "Yes" intents (which you will create below). If your application includes additional intents, you may need to add those to the code.

4. Navigate to **Intents** > **Default Fallback Intent** and configure it as follows:
   a. Delete all of the items in the **Context**, **Events**, **Training Phrases**, **Actions**, and **Response** sections.
   b. Under **Fulfillment**, select **Enable webhook call for this intent**.

   c. Click **Save**.

5. Click **Default Welcome Intent** and configure it as follows:
   a. Delete all of the items in the **Training Phrases** and **Responses** sections.

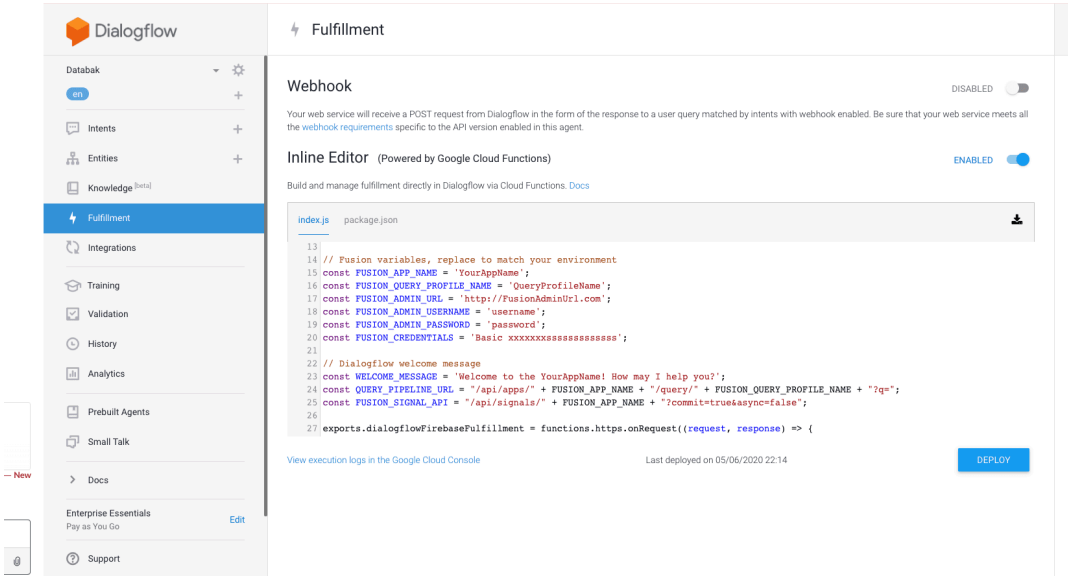
b. Under **Fulfillment**, select **Enable webhook call for this intent**.
c. Click **Save**.

6. Navigate to **Intents** > **Create Intent**, call the new intent "Try again" (case-sensitive), and configure it as follows:
   a. Delete all of the items in the **Context**, **Events**, **Actions**, and **Response** sections.
   b. Under **Training Phrases**, add "Try again" (case-sensitive).
   c. Under **Fulfillment**, select **Enable webhook call for this intent**.
   d. Click **Save**.

7. Navigate to **Intents** > **Create Intent**, call the new intent "Yes" (case-sensitive), and configure it as follows:
   a. Delete all of the items in the **Context**, **Events**, **Actions**, and **Response** sections.
   b. Under **Training Phrases**, add the following phrases (case-sensitive):
      - "Try again"
      - "Yep"
      - "Perfect"

   c. Under **Fulfillment**, select **Enable webhook call for this intent**.
   d. Click **Save**.

8. Navigate to **fulfillment** and configure it as follows:
   a. Select **Enable Inline Editor**.
   b. Overwrite the default code with the `voice.js` code that you customized in step 2 above.



c. Click **Deploy**.

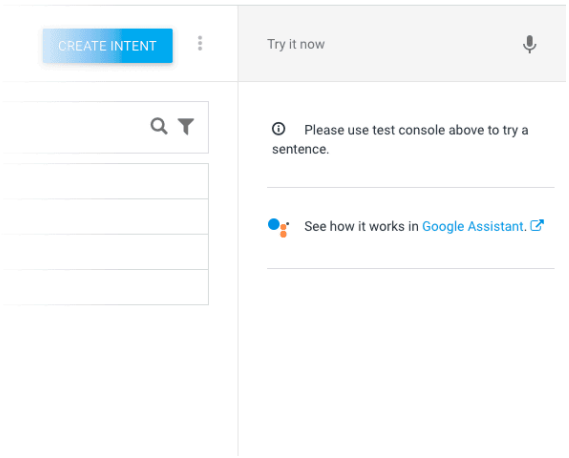
> **Note**
>
> In order to enable external API calls to Fusion, you must upgrade your Google Firebase billing account to the "Pay as you go" plan. Google Firebase is also where you can view the console logs for these API calls.

4. Perform text-to-speech on Fusion's responses

1. In your Google Dialogflow agent interface, click the gear icon next to your agent's name.
2. Under **Automatic Spell Correction**, enable **Allow ML to correct spelling of query during request processing**.
3. Under **Text to speech**, select **Enable Automatic Text to Speech**.
4. Click **Save**.

5. Test your configuration

1. In your Google Dialogflow agent interface, click the **Google Assistant** link in the right-hand sidebar.



2. Click **Talk to <YOUR AGENT NAME>**.
3. Click the microphone and ask a question.
   If you do not receive a response that makes sense, check the logs in Google Firebase for information you can use to troubleshoot your configuration.

> **Tip**
>
> Once your voice chatbot is working, you can click **Integrations** to access a pre-built widget that you can paste into the code for your front-end application.

See the Google Dialogflow documentation for more details.

Configure Web Site Authentication

How-to

> Compatible versions: 4.2, 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion Connectors*, *Fusion Server*, *Fusion*

The Web V1 connector retrieves data from a Web site using HTTP and starting from a specified URL.

The Web connector supports Basic, Digest, Form, and NTLM authentication to websites.

The credentials for a crawl are stored in a credentials file in `https://FUSION_HOST:FUSION_PORT/data/connectors/container/lucid.web/datasourceName`, where `datasourceName` is the name of the datasource. After you create a datasource, Fusion creates this directory for you. The file should be a JSON formatted file, ending with the `.json` file extension. When defining the datasource, use the name of the file in the **Authentication credentials filename** field in the UI (or for the `f.credentialsFile` property if using the REST API).

All authentication types require the credentials file to include a property called `type` that defines the type of authentication to use. The other required properties vary depending on the type of authentication chosen.

Basic form-based Authentication

To use basic form-based authentication, use `form` for the type. The other properties are:

- ttl. The "time to live" for the session created after authentication. After the specified time, the crawler logs in again to keep the crawl activity from failing due to an expired session. This value is defined in seconds.
- action. The action to take to log in. That is, the URL for the login form.
- params. The parameters for the form, likely the username and password, but could be other required properties. In the example below, we pass two parameters, `os_username` and `os_password`, which are expected by the system we crawl.

Here is an example using form-based authentication:

```
[ { "credential" : { "type" : "form", "ttl" : 300000, "action" :
"http://some.server.com/login.action?
os_destination=%2Fpages%2Fviewpage.action%3Ftitle%3DAcme%2B5%2BDocumen
tation%26spaceKey%3DAcme5", "params" : { "os_username" : "username",
"os_password" : "password" } } } ]
```

Complex Form-based Authentication

Some websites do not manage their own authentication, but rather trust a third-party authority to authenticate the user. An example of this is websites that use SAML to log in a user via a central single-signon authority. To configure fusion to log in to a website like this, use `smartForm` for the type. The other properties are:

- ttl. The "time to live" for the session created after authentication. After the specified time, the crawler logs in again to keep the crawl activity from failing due to an expired session. This value is defined in seconds.
- loginUrl. The URL on which the first page that initializes the login chain is located
- params. A list of parameters to use for the form logins, likely the username and password, but could be other required properties. In the example below, we pass two parameters, `os_username` and `os_password`, which are expected by the system we crawl. Additionally we expect that once that login has happened, a new form is presented to the user which then posts back to where we came from. No data need to be entered in this form, which is why we include an empty `{ }` in the params list.

Here is an example using form-based authentication:

```
[ { "credential" : { "type" : "smartForm", "ttl" : 300000, "loginUrl"
: "http://some.example.com/login", "params" : [{ "os_username" :
"username", "os_password" : "password" }, { } ] } } ]
```

To figure out what parameters you need to specify, turn off JavaScript in your browser and go through the login work flow. Though you normally see only a single login form on your screen, you might find many more forms you need to submit before you get logged in when JavaScript is not available to perform those form submissions automatically. Each form in that login chain needs to be represented in the list of `params`. If no user input is required, simply include an empty `{ }`.

Basic and Digest Authentication

Basic and Digest authentication are simple HTTP authentication methods still in use in some places. To use either of these types, in the credentials file, for the `type` property use "basic" or "digest". The other properties are:

- host. The host of the site.
- port. The port, if any.
- userName. The username to use for authentication.
- password. The password for the userName.
- realm. The *security realm* for the site, if any.

Example basic auth configuration:

```
[ { "credential" : { "type" : "basic", "ttl" : 300000, "userName" :
"usr", "password" : "pswd", "host":"hostname.exampledomain.com”
"port": 443 } } ]
```

NTLM Authentication

To use NTLM authentication, in the credentials file, for the `type` property, use `ntlm`. The other properties available are:

- host. The host of the site.
- port. The port, if any.
- userName. The username to use for authentication.
- password. The password for the userName.
- realm. The *security realm* for the site, if any.
- domain. The domain.
- workstation. The workstation, as needed.

Example NTLM credential configuration:

```
[ {"credential" : { "type" : "ntlm", "ttl" : 300000, "port" : 80,
"host" : "someHost", "domain" : "someDomain", "userName" : "someUser",
"password" : "XXXXXXXX" } } ]
```

Crawl a Web site protected by Kerberos

In Fusion 4.1 and later, the Web connector can crawl Web sites protected by Kerberos using SPNEGO. This is a way to access Web sites without requiring a user's login credentials.

Note

Kerberos support requires Fusion 5.9.5.

The Fusion Web connector can optionally use Kerberos with SAML/Smart Form authentication.

To crawl Kerberos-protected Web sites, first create the necessary configuration files, then configure Fusion to use them.

Create standard Java configuration files to connect to Kerberos

Fusion uses the JDK standard JAAS Kerberos implementation, which is based on three system properties that reference three separate files.

The files are as follows:

- On the Kerberos-protected server, a keytab file, named `kerberuser.keytab` in our examples.
- On the Fusion system, a configuration file named `login.conf`.
- On the Fusion system, an initialization file named `krb5.ini`.

CREATE A KERBEROS KEYTAB

Create and validate the keytab file for the Kerberos client principal you want to use to authenticate to the website.

If you do not specify the `kerberosPrincipalName` and `kerberosKeytabFilePath` or `kerberosKeytabBase64` when *creating the Fusion datasource*, Fusion uses the default login principal and ticket cache. You can see the default values by logging into the Fusion server as the user who runs Fusion and running `klist`.

If you do not want to use the default account and credentials, specify these configuration properties when creating a keytab as well as in the *Web datasource setup*. Use the Kerberos user principal name (UPN), not the service principal name (SPN, which is used with the Kerberos *security realm*). In some cases the UPN can be a service.

In our examples, the Fusion Web crawler authenticates to the Web sites using the user kerbuser@win.lab.lucidworks.com. We create a keytab file `kerbuser.keytab` for the user principal kerbuser@WIN.LAB.LUCIDWORKS.COM.

Create a Kerberos keytab on Windows

Example:

```
ktpass -out kerbuser.keytab -princ kerbuser@WIN.LAB.LUCIDWORKS.COM -
mapUser kerbuser -mapOp set -pass YOUR_PASSWORD -crypto AES256-SHA1 -
pType KRB5_NT_PRINCIPAL
```

+

Note

The following weak encryption types are not supported by Fusion:

- DES-CBC-CRC
- DES_CBC_MD5
- Microsoft Windows 2000 RC4-HMAC

Create a Kerberos keytab on Ubuntu Linux

Prerequisite: Install the `krb5-user` package: `sudo apt-get install krb5-user`

Example:

```
ktutil addent -password -p HTTP/kerbuser@WIN.LAB.LUCIDWORKS.COM -k 1 -
e aes128-cts-hmac-sha1-96 - it will ask you for password of kerbuser -
wkt kerbuser.keytab q
```

Test the keytab

Once you create a keytab, verify that it works.

Prerequisite: You need a version of curl installed that allows SPNEGO. To test whether your version of curl does this, run `curl --version` and make sure SPNEGO is in the output.

Run the following curl command (replace the keytab path and site):

```
export KRB5CCNAME=FILE:/path/to/kerbuser.keytab curl -vvv --negotiate
-u : http://your-site.com
```

Note that the first request is a 401 status code for the negotiate request followed by a second request, which is a status of 200.

CREATE A LOGIN.CONF AND KRB5.INI

On the Fusion server, create `login.conf` and `krb5.ini` files as follows.

Create a login.conf on Windows

In this example, the keytab is stored at `C:\\kerb\\kerbuser.keytab`

```
KrbLogin { com.sun.security.auth.module.Krb5LoginModule required
useKeyTab=true storeKey=true keyTab="C:\\kerb\\kerbuser.keytab"
useTicketCache=true principal="kerbuser@WIN.LAB.LUCIDWORKS.COM"
debug=true; };
```

Create a login.conf on Linux

In this example, the keytab is stored at `/home/lucidworks/kb.keytab`

```
com.sun.security.jgss.initiate {
com.sun.security.auth.module.Krb5LoginModule required useKeyTab=true
storeKey=true keyTab="/home/lucidworks/kerbuser.keytab"
useTicketCache=true principal="kerbuser@WIN.LAB.LUCIDWORKS.COM"
debug=true; };
```

The format of the login.conf is described on the Oracle Web site.

Create a krb5.ini

When you install krb5 on Linux, you can find a Kerberos configuration file in `/etc/krb5.conf`. You can optionally create a custom one instead.

Creating a `krb5.conf` is the same for Linux and Windows. On Windows the file is `krb5.ini`.

In this example the domain is WIN.LAB.LUCIDWORKS.COM, the Kerberos kdc host is `my.kdc-dns.com`, and the Kerberos admin server is `my-admin-server-dns.com`.

Example:

```
[libdefaults] default_realm = WIN.LAB.LUCIDWORKS.COM
default_tkt_enctypes = aes128-cts-hmac-sha1-96 default_tgs_enctypes =
aes128-cts-hmac-sha1-96 permitted_enctypes = aes128-cts-hmac-sha1-96
dns_lookup_realm = false dns_lookup_kdc = false ticket_lifetime = 24h
renew_lifetime = 7d forwardable = true udp_preference_limit = 1
[realms] WIN.LAB.LUCIDWORKS.COM = { kdc = my.kdc-dns.com admin_server
= my.admin-server-dns.com } [domain_realm] .WIN.LAB.LUCIDWORKS.COM =
WIN.LAB.LUCIDWORKS.COM WIN.LAB.LUCIDWORKS.COM =
WIN.LAB.LUCIDWORKS.COM
```

The format of the `krb5.ini` file is described in the MIT Kerberos documentation. You can change the encryption algorithms by changing the properties `default_tkt_enctypes`, `default_tgs_enctypes`, and `permitted_enctypes` as needed. For example:

```
default_tkt_enctypes = RC4-HMAC default_tgs_enctypes = RC4-HMAC
Permitted_enctypes = RC4-HMAC
```

Configure Fusion to use Kerberos

Once you have the keytab, `login.conf`, and `krb5.ini` files, configure Fusion to use Kerberos. You must set a property in a Fusion configuration file in addition to defining the datasource in the Fusion UI.

At the command line on every machine in your Fusion cluster:

1. In `$FUSION_HOME/conf/fusion.cors` (`fusion.properties` in Fusion 4.x), add the following property to the `connectors-classic.jvmOptions` setting: `-Djavax.security.auth.useSubjectCredsOnly=false`
2. Restart the `connectors-classic` service using `./bin/connectors-classic restart` on Linux or `bin\connectors-classic.cmd restart` on Windows.

In the Fusion UI:

1. Click **Indexing > Datasources**.
2. Click **Add+**, then **Web**.
3. Enter a datasource ID and a start link.
4. Click **Crawl authorization**.
5. At the bottom of the section, check **Enable SPNEGO/Kerberos Authentication**.
6. You can either use the default principal name or specify a principal name to use.
    - If you do not specify the principal name, then Fusion uses the default login principal and ticket cache. You can see those default values by logging into the Fusion server as the user who runs Fusion and running `klist`.
7. If you specify a principal name, you must provide a keytab, either in Base64 or as a file path.
    - If you specify a keytab file path, the file must be on the machine running the Fusion connector, for each connector's node in the cluster.
    - The Base64 option lets you supply the keytab in one place, in the UI.

8. Fill in any remaining options to configure the datasource.
9. Click **Save**.

Troubleshoot Kerberos authentication

JAVAX.SECURITY.AUTH.LOGIN.LOGINEXCEPTION: NO KEY TO STORE

Problem: When trying to crawl a Kerberos-authenticated Web site, you get an error like this:

```
crawler.common.ComponentInitException: Could not initialize
Spnego/Kerberos. at
crawler.fetch.impl.http.WebFetcher.init(WebFetcher.java:282) ~
[lucid.web-4.0.2.jar:?] at
crawler.common.ComponentFactory.initComponent(ComponentFactory.java:37
) ~[lucid.anda-4.0.2.jar:?] at
crawler.Crawler.initComponents(Crawler.java:125) ~[lucid.anda-
4.0.2.jar:?] at crawler.Crawler.init(Crawler.java:108) ~[lucid.anda-
4.0.2.jar:?] at
crawler.common.ComponentFactory.initComponent(ComponentFactory.java:37
) ~[lucid.anda-4.0.2.jar:?] at
crawler.common.config.CrawlConfig.buildCrawler(CrawlConfig.java:212) ~
[lucid.anda-4.0.2.jar:?] at
com.lucidworks.connectors.anda.AndaFetcher.start(AndaFetcher.java:139)
[lucid.anda-4.0.2.jar:?] at
com.lucidworks.connectors.ConnectorJob.start(ConnectorJob.java:200)
[lucid.shared-4.0.2.jar:?] at
com.lucidworks.connectors.Connector$RunnableJob.run(Connector.java:319
) [lucid.shared-4.0.2.jar:?] Caused by: java.lang.Exception: Could not
perform spnego/kerberos login. java.security.krb5.conf =
/etc/krb5.conf,, Keytab file =
/home/ndipiazza/Downloads/kerbuser.keytab, login config =
{principal=HTTP/kerbuser@WIN.LAB.LUCIDWORKS.COM, debug=false,
storeKey=true, keyTab=/home/ndipiazza/Downloads/kerbuser.keytab,
useKeyTab=true, useTicketCache=true, refreshKrb5Config=true} at
com.lucidworks.connectors.spnego.SpnegoAuth.<init>(SpnegoAuth.java:83)
~[?:?] at crawler.fetch.impl.http.WebFetcher.init(WebFetcher.java:279)
~[?:?] ... 8 more Caused by: javax.security.auth.login.LoginException:
No key to store at
com.sun.security.auth.module.Krb5LoginModule.commit(Krb5LoginModule.ja
va:1119) ~[?:1.8.0_161] at
sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method) ~
[?:1.8.0_161] at
```

```
sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.j
ava:62) ~[?:1.8.0_161] at
sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccess
orImpl.java:43) ~[?:1.8.0_161] at
java.lang.reflect.Method.invoke(Method.java:498) ~[?:1.8.0_161] at
javax.security.auth.login.LoginContext.invoke(LoginContext.java:755) ~
[?:1.8.0_161] at
javax.security.auth.login.LoginContext.access$000(LoginContext.java:19
5) ~[?:1.8.0_161] at
javax.security.auth.login.LoginContext$4.run(LoginContext.java:682) ~
[?:1.8.0_161] at
javax.security.auth.login.LoginContext$4.run(LoginContext.java:680) ~
[?:1.8.0_161] at java.security.AccessController.doPrivileged(Native
Method) ~[?:1.8.0_161] at
javax.security.auth.login.LoginContext.invokePriv(LoginContext.java:68
0) ~[?:1.8.0_161] at
javax.security.auth.login.LoginContext.login(LoginContext.java:588) ~
[?:1.8.0_161] at com.lucidworks.connectors.spnego.SpnegoAuth.<init>
(SpnegoAuth.java:76) ~[?:?] at
crawler.fetch.impl.http.WebFetcher.init(WebFetcher.java:279) ~[?:?]
... 8 more
```

Resolution:

First test your keytab as described in *test the keytab* above.

If your keytab passes validation, another possibility is that the `/tmp/krb*` cache file got corrupted or is not compatible after you went through other troubleshooting steps. To rule that out, remove the `/tmp/krb*` cache file on all hosts, restart your connectors-classic, and try the crawl again. That is, on each host:

```
rm -f /tmp/krb* $FUSION_HOME/bin/connectors-classic restart
```

401 ERROR

Problem: Crawling using the Web connector with Kerberos results in a 401 error, but curl with Kerberos works fine.

Resolution:

Make sure you have this system property set in `connectors-classic jvmOptions` **on all nodes**:

```
-Djavax.security.auth.useSubjectCredsOnly=false
```

You must restart `connectors-classic` after making that change.

If that doesn't work, make sure the user you are authenticating with from Curl matches the user you are trying to authenticate with from the Web connector. To see your Kerberos principal user name, run `klist`.

ERROR: "PRE-AUTHENTICATION INFORMATION WAS INVALID - IDENTIFIER DOESN'T MATCH EXPECTED VALUE"

Problem: When crawling using the Web connector with Kerberos enabled, you get an error like this:

```
crawler.common.ComponentInitException: Could not initialize
Spnego/Kerberos. at
crawler.fetch.impl.http.WebFetcher.init(WebFetcher.java:282) ~
[lucid.web-4.0.2.jar:?] at
crawler.common.ComponentFactory.initComponent(ComponentFactory.java:37
) ~[lucid.anda-4.0.2.jar:?] at
crawler.Crawler.initComponents(Crawler.java:125) ~[lucid.anda-
4.0.2.jar:?] at crawler.Crawler.init(Crawler.java:108) ~[lucid.anda-
4.0.2.jar:?] at
crawler.common.ComponentFactory.initComponent(ComponentFactory.java:37
) ~[lucid.anda-4.0.2.jar:?] at
crawler.common.config.CrawlConfig.buildCrawler(CrawlConfig.java:212) ~
[lucid.anda-4.0.2.jar:?] at
com.lucidworks.connectors.anda.AndaFetcher.start(AndaFetcher.java:139)
[lucid.anda-4.0.2.jar:?] at
com.lucidworks.connectors.ConnectorJob.start(ConnectorJob.java:200)
[lucid.shared-4.0.2.jar:?] at
com.lucidworks.connectors.Connector$RunnableJob.run(Connector.java:319
) [lucid.shared-4.0.2.jar:?] Caused by: java.lang.Exception: Could not
perform spnego/kerberos login. java.security.krb5.conf =
/etc/krb5.conf,, Keytab file =
/home/ndipiazza/Downloads/kerbuser.keytab, login config =
{principal=kerbuser@WIN.LAB.LUCIDWORKS.COM, debug=false,
storeKey=true, keyTab=/home/ndipiazza/Downloads/kerbuser.keytab,
useKeyTab=true, useTicketCache=true, refreshKrb5Config=true} at
com.lucidworks.connectors.spnego.SpnegoAuth.<init>(SpnegoAuth.java:83)
~[?:?] at crawler.fetch.impl.http.WebFetcher.init(WebFetcher.java:279)
~[?:?] ... 8 more Caused by: javax.security.auth.login.LoginException:
Pre-authentication information was invalid (24) at
com.sun.security.auth.module.Krb5LoginModule.attemptAuthentication(Krb
5LoginModule.java:804) ~[?:1.8.0_161] at
com.sun.security.auth.module.Krb5LoginModule.login(Krb5LoginModule.jav
a:617) ~[?:1.8.0_161] at
```

```
sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method) ~
[?:1.8.0_161] at
sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.j
ava:62) ~[?:1.8.0_161] at
sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccess
orImpl.java:43) ~[?:1.8.0_161] at
java.lang.reflect.Method.invoke(Method.java:498) ~[?:1.8.0_161] at
javax.security.auth.login.LoginContext.invoke(LoginContext.java:755) ~
[?:1.8.0_161] at
javax.security.auth.login.LoginContext.access$000(LoginContext.java:19
5) ~[?:1.8.0_161] at
javax.security.auth.login.LoginContext$4.run(LoginContext.java:682) ~
[?:1.8.0_161] at
javax.security.auth.login.LoginContext$4.run(LoginContext.java:680) ~
[?:1.8.0_161] at java.security.AccessController.doPrivileged(Native
Method) ~[?:1.8.0_161] at
javax.security.auth.login.LoginContext.invokePriv(LoginContext.java:68
0) ~[?:1.8.0_161] at
javax.security.auth.login.LoginContext.login(LoginContext.java:587) ~
[?:1.8.0_161] at com.lucidworks.connectors.spnego.SpnegoAuth.<init>
(SpnegoAuth.java:76) ~[?:?] at
crawler.fetch.impl.http.WebFetcher.init(WebFetcher.java:279) ~[?:?]
... 8 more Caused by: sun.security.krb5.KrbException: Pre-
authentication information was invalid (24) at
sun.security.krb5.KrbAsRep.<init>(KrbAsRep.java:76) ~[?:1.8.0_161] at
sun.security.krb5.KrbAsReqBuilder.send(KrbAsReqBuilder.java:316) ~
[?:1.8.0_161] at
sun.security.krb5.KrbAsReqBuilder.action(KrbAsReqBuilder.java:361) ~
[?:1.8.0_161] at
com.sun.security.auth.module.Krb5LoginModule.attemptAuthentication(Krb
5LoginModule.java:776) ~[?:1.8.0_161] at
com.sun.security.auth.module.Krb5LoginModule.login(Krb5LoginModule.jav
a:617) ~[?:1.8.0_161] at
sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method) ~
```

```
[?:1.8.0_161] at
sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.j
ava:62) ~[?:1.8.0_161] at
sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccess
orImpl.java:43) ~[?:1.8.0_161] at
java.lang.reflect.Method.invoke(Method.java:498) ~[?:1.8.0_161] at
javax.security.auth.login.LoginContext.invoke(LoginContext.java:755) ~
[?:1.8.0_161] at
javax.security.auth.login.LoginContext.access$000(LoginContext.java:19
5) ~[?:1.8.0_161] at
javax.security.auth.login.LoginContext$4.run(LoginContext.java:682) ~
[?:1.8.0_161] at
javax.security.auth.login.LoginContext$4.run(LoginContext.java:680) ~
[?:1.8.0_161] at java.security.AccessController.doPrivileged(Native
Method) ~[?:1.8.0_161] at
javax.security.auth.login.LoginContext.invokePriv(LoginContext.java:68
0) ~[?:1.8.0_161] at
javax.security.auth.login.LoginContext.login(LoginContext.java:587) ~
[?:1.8.0_161] at com.lucidworks.connectors.spnego.SpnegoAuth.<init>
(SpnegoAuth.java:76) ~[?:?] at
crawler.fetch.impl.http.WebFetcher.init(WebFetcher.java:279) ~[?:?]
... 8 more Caused by: sun.security.krb5.Asn1Exception: Identifier does
not match expected value (906) at
sun.security.krb5.internal.KDCRep.init(KDCRep.java:140) ~[?:1.8.0_161]
at sun.security.krb5.internal.ASRep.init(ASRep.java:64) ~[?:1.8.0_161]
at sun.security.krb5.internal.ASRep.<init>(ASRep.java:59) ~
[?:1.8.0_161] at sun.security.krb5.KrbAsRep.<init>(KrbAsRep.java:60) ~
[?:1.8.0_161] at
sun.security.krb5.KrbAsReqBuilder.send(KrbAsReqBuilder.java:316) ~
[?:1.8.0_161] at
sun.security.krb5.KrbAsReqBuilder.action(KrbAsReqBuilder.java:361) ~
[?:1.8.0_161] at
com.sun.security.auth.module.Krb5LoginModule.attemptAuthentication(Krb
5LoginModule.java:776) ~[?:1.8.0_161] at
```

```
com.sun.security.auth.module.Krb5LoginModule.login(Krb5LoginModule.jav
a:617) ~[?:1.8.0_161] at
sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method) ~
[?:1.8.0_161] at
sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.j
ava:62) ~[?:1.8.0_161] at
sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccess
orImpl.java:43) ~[?:1.8.0_161] at
java.lang.reflect.Method.invoke(Method.java:498) ~[?:1.8.0_161] at
javax.security.auth.login.LoginContext.invoke(LoginContext.java:755) ~
[?:1.8.0_161] at
javax.security.auth.login.LoginContext.access$000(LoginContext.java:19
5) ~[?:1.8.0_161] at
javax.security.auth.login.LoginContext$4.run(LoginContext.java:682) ~
[?:1.8.0_161] at
javax.security.auth.login.LoginContext$4.run(LoginContext.java:680) ~
[?:1.8.0_161] at java.security.AccessController.doPrivileged(Native
Method) ~[?:1.8.0_161] at
javax.security.auth.login.LoginContext.invokePriv(LoginContext.java:68
0) ~[?:1.8.0_161] at
javax.security.auth.login.LoginContext.login(LoginContext.java:587) ~
[?:1.8.0_161] at com.lucidworks.connectors.spnego.SpnegoAuth.<init>
(SpnegoAuth.java:76) ~[?:?] at
crawler.fetch.impl.http.WebFetcher.init(WebFetcher.java:279) ~[?:?]
... 8 more
```

Resolution:

Your keytab's principal name doesn't match the value on the ticket server. Check the principal name for your user.

Configure website authentication

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

The Web connector retrieves data from a Web site using HTTP and starting from a specified URL.

Managed Fusion releases 5.5 - 5.6.0 use the *Web V1 connector*. Managed Fusion releases 5.6.1 and later use the *Web V2 connector*.

The Web connector supports Basic, Digest, Form, and NTLM authentication to websites.

The credentials for a crawl are stored in a credentials file in `https://EXAMPLE_COMPANY.b.lucidworks.cloud:6764/data/connectors/container/lucid.web/datasourceName`, where `datasourceName` is the name of the datasource.

After you create a datasource, Managed Fusion creates this directory for you. The file should be a JSON formatted file, ending with the `.json` file extension.

When defining the datasource, use the name of the file in the Authentication credentials filename field in the UI (or for the `f.credentialsFile` property if using the REST API).

All authentication types require the credentials file to include a property called `type` that defines the type of authentication to use. The other required properties vary depending on the type of authentication chosen.

Basic form-based authentication

To use basic form-based authentication, use the value of form for the `type`. The other properties are:

- `ttl` is the "time to live" for the session created after authentication. After the specified time, the crawler logs in again to keep the crawl activity from failing due to an expired session. This value is defined in seconds.
- `action` is the action to take to log in. That is, the URL for the login form.
- `params` are the parameters for the form, such as the username and password. But this must also include required properties. In the following example, two parameters are specified: `os_username` and `os_password`, which are expected by the system we crawl.

Here is an example using form-based authentication:

```
[ { "credential" : { "type" : "form", "ttl" : 300000, "action" :
"http://some.server.com/login.action?
os_destination=%2Fpages%2Fviewpage.action%3Ftitle%3DAcme%2B5%2BDocumen
tation%26spaceKey%3DAcme5", "params" : { "os_username" : "username",
"os_password" : "password" } } } ]
```

Complex Form-based authentication

Some websites do not manage their own authentication, but rather trust a third-party authority to authenticate the user. An example of this is websites that use SAML to sign in a user via a central single-signon authority.

To configure Managed Fusion to sign in to a website like this, use the value of smartForm for the `type` property. The other properties are:

- `ttl` is the "time to live" for the session created after authentication. After the specified time, the crawler logs in again to keep the crawl activity from failing due to an expired session. This value is defined in seconds.
- `loginUrl` is the URL on which the first page that initializes the login chain is located.
- `params` are the parameters for form sign in credentials, such as the username and password. But this must also include required properties. In the following example, two parameters are specified: `os_username` and `os_password`, which are expected by the system we crawl.

After the sign in occurs, a new form displays to the user which then posts back to where we came from. No data needs to be entered in this form, which is the example includes an empty `{ }` in the params list.

Here is an example using form-based authentication:

```
[ { "credential" : { "type" : "smartForm", "ttl" : 300000, "loginUrl"
: "http://some.example.com/login", "params" : [{ "os_username" :
"username", "os_password" : "password" }, { } ] } } ]
```

To figure out what parameters you need to specify, turn off JavaScript in your browser and go through the sign in work flow.

Though you normally see only a single login form on your screen, you might find many more forms you need to submit before you get signed in when JavaScript is not available to perform those form submissions automatically.

Each form in that sign-in chain needs to be represented in the list of `params`. If no user input is required, use an empty `{ }`.

Basic and Digest Authentication

Basic and Digest authentication are simple `HTTP` authentication methods still in use in some places. To use either of these types, in the credentials file, for the `type` property use a value of basic or digest. The other properties are:

- `host` is the host of the site.
- `port` is the port, if any.
- `userName` is the user identifier to use for authentication.
- `password` is the password associated with the `userName`.
- `realm` is the security realm for the site. For more information, see *Security realms*.

Example basic auth configuration:

```
[ { "credential" : { "type" : "basic", "ttl" : 300000, "userName" :
"usr", "password" : "pswd", "host":"hostname.exampledomain.com”
"port": 443 } } ]
```

NTLM authentication

To use NTLM authentication, in the credentials file, use a value of ntlm in the `type` property. The other properties available are:

- `host` is the host of the site.
- `port` is the port, if any.
- `userName` is the user identifier to use for authentication.
- `password` is the password associated with the `userName`.
- `realm` is the security realm for the site. For more information, see *Security realms*.
- `domain` is the domain of the site.
- `workstation` is the identifier for the user computer, if needed.

Example NTLM credential configuration:

```
[ {"credential" : { "type" : "ntlm", "ttl" : 300000, "port" : 80,
"host" : "someHost", "domain" : "someDomain", "userName" : "someUser",
"password" : "XXXXXXXX" } } ]
```

Crawl a website protected by Kerberos

The Web connector can crawl Web sites protected by Kerberos using SPNEGO without requiring a user's sign-in credentials.

Note

Kerberos support requires Managed Fusion 5.9.5.

The Managed Fusion Web connector can also use Kerberos with SAML/Smart Form authentication.

To crawl Kerberos-protected Web sites, first create the necessary configuration files, then configure Managed Fusion to use them.

Create standard Java configuration files to connect to Kerberos

Managed Fusion uses the JDK standard JAAS Kerberos implementation, which is based on three system properties that reference three separate files.

The files are as follows:

- On the Kerberos-protected server, a keytab file, named `kerberuser.keytab` in our examples.
- On the Managed Fusion system, a configuration file named `login.conf`.
- On the Managed Fusion system, an initialization file named `krb5.ini`.

CREATE A KERBEROS KEYTAB

Create and validate the keytab file for the Kerberos client principal you want to use to authenticate to the website.

If you do not specify the `kerberosPrincipalName` and `kerberosKeytabFilePath` or `kerberosKeytabBase64` when *creating the Managed Fusion datasource*, Managed Fusion uses the default sign-in principal and ticket cache. You can see the default values by logging into the Managed Fusion server as the user who runs Managed Fusion and running `klist`.

If you do not want to use the default account and credentials, specify these configuration properties when creating a keytab as well as in the *Web datasource setup*. Use the Kerberos user principal name (UPN), not the service principal name (SPN, which is used with the Kerberos *security realm*). In some cases the UPN can be a service.

In our examples, the Managed Fusion Web crawler authenticates to the Web sites using the user kerbuser@win.lab.lucidworks.com. We create a keytab file `kerbuser.keytab` for the user principal kerbuser@WIN.LAB.LUCIDWORKS.COM.

Create a Kerberos keytab on Windows

Example:

```
ktpass -out kerbuser.keytab -princ kerbuser@WIN.LAB.LUCIDWORKS.COM -
mapUser kerbuser -mapOp set -pass YOUR_PASSWORD -crypto AES256-SHA1 -
pType KRB5_NT_PRINCIPAL
```

+

Note

The following weak encryption types are not supported by Fusion:

- DES-CBC-CRC
- DES_CBC_MD5
- Microsoft Windows 2000 RC4-HMAC

Create a Kerberos keytab on Ubuntu Linux

Prerequisite: Install the `krb5-user` package: `sudo apt-get install krb5-user`.

Example:

```
ktutil addent -password -p HTTP/kerbuser@WIN.LAB.LUCIDWORKS.COM -k 1 -
e aes128-cts-hmac-sha1-96 - it will ask you for password of kerbuser -
wkt kerbuser.keytab q
```

Test the keytab

Once you create a keytab, verify that it works.

Prerequisite: You need a version of curl installed that allows SPNEGO. To test whether your version of curl does this, run `curl --version` and make sure SPNEGO is in the output.

Run the following curl command (replace the keytab path and site):

```
export KRB5CCNAME=FILE:/path/to/kerbuser.keytab curl -vvv --negotiate
-u : http://your-site.com
```

Note that the first request is a 401 status code for the negotiate request followed by a second request, which is a status of 200.

CREATE A LOGIN.CONF AND KRB5.INI

On the Managed Fusion server, create `login.conf` and `krb5.ini` files as follows.

Create a login.conf on Windows

In this example, the keytab is stored at `C:\\kerb\\kerbuser.keytab`.

```
KrbLogin { com.sun.security.auth.module.Krb5LoginModule required
useKeyTab=true storeKey=true keyTab="C:\\kerb\\kerbuser.keytab"
useTicketCache=true principal="kerbuser@WIN.LAB.LUCIDWORKS.COM"
debug=true; };
```

Create a login.conf on Linux

In this example, the keytab is stored at `/home/lucidworks/kb.keytab`.

```
com.sun.security.jgss.initiate {
com.sun.security.auth.module.Krb5LoginModule required useKeyTab=true
storeKey=true keyTab="/home/lucidworks/kerbuser.keytab"
useTicketCache=true principal="kerbuser@WIN.LAB.LUCIDWORKS.COM"
debug=true; };
```

The format of the login.conf is described on the Oracle Web site.

Create a krb5.ini

When you install krb5 on Linux, you can find a Kerberos configuration file in `/etc/krb5.conf`. You can optionally create a custom one instead.

Creating a `krb5.conf` is the same for Linux and Windows. On Windows the file is `krb5.ini`.

In this example the domain is WIN.LAB.LUCIDWORKS.COM, the Kerberos kdc host is `my.kdc-dns.com`, and the Kerberos admin server is `my-admin-server-dns.com`.

Example:

```
[libdefaults] default_realm = WIN.LAB.LUCIDWORKS.COM
default_tkt_enctypes = aes128-cts-hmac-sha1-96 default_tgs_enctypes =
aes128-cts-hmac-sha1-96 permitted_enctypes = aes128-cts-hmac-sha1-96
dns_lookup_realm = false dns_lookup_kdc = false ticket_lifetime = 24h
renew_lifetime = 7d forwardable = true udp_preference_limit = 1
[realms] WIN.LAB.LUCIDWORKS.COM = { kdc = my.kdc-dns.com admin_server
= my.admin-server-dns.com } [domain_realm] .WIN.LAB.LUCIDWORKS.COM =
WIN.LAB.LUCIDWORKS.COM WIN.LAB.LUCIDWORKS.COM =
WIN.LAB.LUCIDWORKS.COM
```

The format of the `krb5.ini` file is described in the MIT Kerberos documentation. You can change the encryption algorithms by changing the properties `default_tkt_enctypes`, `default_tgs_enctypes`, and `permitted_enctypes` as needed. For example:

```
default_tkt_enctypes = RC4-HMAC default_tgs_enctypes = RC4-HMAC
Permitted_enctypes = RC4-HMAC
```

Configure Managed Fusion to use Kerberos

Once you have the keytab, `login.conf`, and `krb5.ini` files, configure Managed Fusion to use Kerberos. You must set a property in a Managed Fusion configuration file in addition to defining the datasource in the Managed Fusion UI.

At the command line on every machine in your Managed Fusion cluster:

1. In `$FUSION_HOME/conf/fusion.cors`, add the following property to the `connectors-classic.jvmOptions` setting: `-Djavax.security.auth.useSubjectCredsOnly=false`
2. Restart the `connectors-classic` service using `./bin/connectors-classic restart` on Linux or `bin\connectors-classic.cmd restart` on Windows.

In the Managed Fusion UI:

1. Click **Indexing > Datasources > Add+ > Web**.
2. Enter a datasource ID and a start link.
3. Click **Crawl authorization**.
4. At the bottom of the section, check **Enable SPNEGO/Kerberos Authentication**.
5. You can either use the default principal name or specify a principal name to use.
    - If you do not specify the principal name, Managed Fusion uses the default login principal and ticket cache. You can see those default values by logging into the Managed Fusion server as the user who runs Managed Fusion and running `klist`.
6. If you specify a principal name, you must provide a keytab, either in Base64 or as a file path.
    - If you specify a keytab file path, the file must be on the machine running the Managed Fusion connector for each connector's node in the cluster.
    - The Base64 option lets you supply the keytab in one place in the UI.

7. Fill in any remaining options to configure the datasource.
8. Click **Save**.

Troubleshoot Kerberos authentication

JAVAX.SECURITY.AUTH.LOGIN.LOGINEXCEPTION: NO KEY TO STORE

Problem: When trying to crawl a Kerberos-authenticated Web site, you get an error like this:

```
crawler.common.ComponentInitException: Could not initialize
Spnego/Kerberos. at
crawler.fetch.impl.http.WebFetcher.init(WebFetcher.java:282) ~
[lucid.web-4.0.2.jar:?] at
crawler.common.ComponentFactory.initComponent(ComponentFactory.java:37
) ~[lucid.anda-4.0.2.jar:?] at
crawler.Crawler.initComponents(Crawler.java:125) ~[lucid.anda-
4.0.2.jar:?] at crawler.Crawler.init(Crawler.java:108) ~[lucid.anda-
4.0.2.jar:?] at
crawler.common.ComponentFactory.initComponent(ComponentFactory.java:37
) ~[lucid.anda-4.0.2.jar:?] at
crawler.common.config.CrawlConfig.buildCrawler(CrawlConfig.java:212) ~
[lucid.anda-4.0.2.jar:?] at
com.lucidworks.connectors.anda.AndaFetcher.start(AndaFetcher.java:139)
[lucid.anda-4.0.2.jar:?] at
com.lucidworks.connectors.ConnectorJob.start(ConnectorJob.java:200)
[lucid.shared-4.0.2.jar:?] at
com.lucidworks.connectors.Connector$RunnableJob.run(Connector.java:319
) [lucid.shared-4.0.2.jar:?] Caused by: java.lang.Exception: Could not
perform spnego/kerberos login. java.security.krb5.conf =
/etc/krb5.conf,, Keytab file =
/home/ndipiazza/Downloads/kerbuser.keytab, login config =
{principal=HTTP/kerbuser@WIN.LAB.LUCIDWORKS.COM, debug=false,
storeKey=true, keyTab=/home/ndipiazza/Downloads/kerbuser.keytab,
useKeyTab=true, useTicketCache=true, refreshKrb5Config=true} at
com.lucidworks.connectors.spnego.SpnegoAuth.<init>(SpnegoAuth.java:83)
~[?:?] at crawler.fetch.impl.http.WebFetcher.init(WebFetcher.java:279)
~[?:?] ... 8 more Caused by: javax.security.auth.login.LoginException:
No key to store at
com.sun.security.auth.module.Krb5LoginModule.commit(Krb5LoginModule.ja
va:1119) ~[?:1.8.0_161] at
sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method) ~
[?:1.8.0_161] at
```

```
sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.j
ava:62) ~[?:1.8.0_161] at
sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccess
orImpl.java:43) ~[?:1.8.0_161] at
java.lang.reflect.Method.invoke(Method.java:498) ~[?:1.8.0_161] at
javax.security.auth.login.LoginContext.invoke(LoginContext.java:755) ~
[?:1.8.0_161] at
javax.security.auth.login.LoginContext.access$000(LoginContext.java:19
5) ~[?:1.8.0_161] at
javax.security.auth.login.LoginContext$4.run(LoginContext.java:682) ~
[?:1.8.0_161] at
javax.security.auth.login.LoginContext$4.run(LoginContext.java:680) ~
[?:1.8.0_161] at java.security.AccessController.doPrivileged(Native
Method) ~[?:1.8.0_161] at
javax.security.auth.login.LoginContext.invokePriv(LoginContext.java:68
0) ~[?:1.8.0_161] at
javax.security.auth.login.LoginContext.login(LoginContext.java:588) ~
[?:1.8.0_161] at com.lucidworks.connectors.spnego.SpnegoAuth.<init>
(SpnegoAuth.java:76) ~[?:?] at
crawler.fetch.impl.http.WebFetcher.init(WebFetcher.java:279) ~[?:?]
... 8 more
```

Resolution:

First test your keytab as described in *test the keytab*.

If your keytab passes validation, another possibility is that the `/tmp/krb*` cache file got corrupted or is not compatible after you went through other troubleshooting steps. To rule that out, remove the `/tmp/krb*` cache file on all hosts, restart your connectors-classic, and try the crawl again. That is, on each host:

```
rm -f /tmp/krb* $FUSION_HOME/bin/connectors-classic restart
```

401 ERROR

Problem: Crawling using the Web connector with Kerberos results in a 401 error, but curl with Kerberos works fine.

Resolution:

Make sure you have this system property set in **`connectors-classic jvmOptions`** **on all nodes**:

```
-Djavax.security.auth.useSubjectCredsOnly=false
```

You must restart `connectors-classic` after making that change.

If that doesn't work, make sure the user you are authenticating with from Curl matches the user you are trying to authenticate with from the Web connector. To see your Kerberos principal user name, run `klist`.

ERROR: "PRE-AUTHENTICATION INFORMATION WAS INVALID - IDENTIFIER DOESN'T MATCH EXPECTED VALUE"

Problem: When crawling using the Web connector with Kerberos enabled, you get an error like this:

```
crawler.common.ComponentInitException: Could not initialize
Spnego/Kerberos. at
crawler.fetch.impl.http.WebFetcher.init(WebFetcher.java:282) ~
[lucid.web-4.0.2.jar:?] at
crawler.common.ComponentFactory.initComponent(ComponentFactory.java:37
) ~[lucid.anda-4.0.2.jar:?] at
crawler.Crawler.initComponents(Crawler.java:125) ~[lucid.anda-
4.0.2.jar:?] at crawler.Crawler.init(Crawler.java:108) ~[lucid.anda-
4.0.2.jar:?] at
crawler.common.ComponentFactory.initComponent(ComponentFactory.java:37
) ~[lucid.anda-4.0.2.jar:?] at
crawler.common.config.CrawlConfig.buildCrawler(CrawlConfig.java:212) ~
[lucid.anda-4.0.2.jar:?] at
com.lucidworks.connectors.anda.AndaFetcher.start(AndaFetcher.java:139)
[lucid.anda-4.0.2.jar:?] at
com.lucidworks.connectors.ConnectorJob.start(ConnectorJob.java:200)
[lucid.shared-4.0.2.jar:?] at
com.lucidworks.connectors.Connector$RunnableJob.run(Connector.java:319
) [lucid.shared-4.0.2.jar:?] Caused by: java.lang.Exception: Could not
perform spnego/kerberos login. java.security.krb5.conf =
/etc/krb5.conf,, Keytab file =
/home/ndipiazza/Downloads/kerbuser.keytab, login config =
{principal=kerbuser@WIN.LAB.LUCIDWORKS.COM, debug=false,
storeKey=true, keyTab=/home/ndipiazza/Downloads/kerbuser.keytab,
useKeyTab=true, useTicketCache=true, refreshKrb5Config=true} at
com.lucidworks.connectors.spnego.SpnegoAuth.<init>(SpnegoAuth.java:83)
~[?:?] at crawler.fetch.impl.http.WebFetcher.init(WebFetcher.java:279)
~[?:?] ... 8 more Caused by: javax.security.auth.login.LoginException:
Pre-authentication information was invalid (24) at
com.sun.security.auth.module.Krb5LoginModule.attemptAuthentication(Krb
5LoginModule.java:804) ~[?:1.8.0_161] at
com.sun.security.auth.module.Krb5LoginModule.login(Krb5LoginModule.jav
a:617) ~[?:1.8.0_161] at
```

```
sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method) ~
[?:1.8.0_161] at
sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.j
ava:62) ~[?:1.8.0_161] at
sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccess
orImpl.java:43) ~[?:1.8.0_161] at
java.lang.reflect.Method.invoke(Method.java:498) ~[?:1.8.0_161] at
javax.security.auth.login.LoginContext.invoke(LoginContext.java:755) ~
[?:1.8.0_161] at
javax.security.auth.login.LoginContext.access$000(LoginContext.java:19
5) ~[?:1.8.0_161] at
javax.security.auth.login.LoginContext$4.run(LoginContext.java:682) ~
[?:1.8.0_161] at
javax.security.auth.login.LoginContext$4.run(LoginContext.java:680) ~
[?:1.8.0_161] at java.security.AccessController.doPrivileged(Native
Method) ~[?:1.8.0_161] at
javax.security.auth.login.LoginContext.invokePriv(LoginContext.java:68
0) ~[?:1.8.0_161] at
javax.security.auth.login.LoginContext.login(LoginContext.java:587) ~
[?:1.8.0_161] at com.lucidworks.connectors.spnego.SpnegoAuth.<init>
(SpnegoAuth.java:76) ~[?:?] at
crawler.fetch.impl.http.WebFetcher.init(WebFetcher.java:279) ~[?:?]
... 8 more Caused by: sun.security.krb5.KrbException: Pre-
authentication information was invalid (24) at
sun.security.krb5.KrbAsRep.<init>(KrbAsRep.java:76) ~[?:1.8.0_161] at
sun.security.krb5.KrbAsReqBuilder.send(KrbAsReqBuilder.java:316) ~
[?:1.8.0_161] at
sun.security.krb5.KrbAsReqBuilder.action(KrbAsReqBuilder.java:361) ~
[?:1.8.0_161] at
com.sun.security.auth.module.Krb5LoginModule.attemptAuthentication(Krb
5LoginModule.java:776) ~[?:1.8.0_161] at
com.sun.security.auth.module.Krb5LoginModule.login(Krb5LoginModule.jav
a:617) ~[?:1.8.0_161] at
sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method) ~
```

```
[?:1.8.0_161] at
sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.j
ava:62) ~[?:1.8.0_161] at
sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccess
orImpl.java:43) ~[?:1.8.0_161] at
java.lang.reflect.Method.invoke(Method.java:498) ~[?:1.8.0_161] at
javax.security.auth.login.LoginContext.invoke(LoginContext.java:755) ~
[?:1.8.0_161] at
javax.security.auth.login.LoginContext.access$000(LoginContext.java:19
5) ~[?:1.8.0_161] at
javax.security.auth.login.LoginContext$4.run(LoginContext.java:682) ~
[?:1.8.0_161] at
javax.security.auth.login.LoginContext$4.run(LoginContext.java:680) ~
[?:1.8.0_161] at java.security.AccessController.doPrivileged(Native
Method) ~[?:1.8.0_161] at
javax.security.auth.login.LoginContext.invokePriv(LoginContext.java:68
0) ~[?:1.8.0_161] at
javax.security.auth.login.LoginContext.login(LoginContext.java:587) ~
[?:1.8.0_161] at com.lucidworks.connectors.spnego.SpnegoAuth.<init>
(SpnegoAuth.java:76) ~[?:?] at
crawler.fetch.impl.http.WebFetcher.init(WebFetcher.java:279) ~[?:?]
... 8 more Caused by: sun.security.krb5.Asn1Exception: Identifier does
not match expected value (906) at
sun.security.krb5.internal.KDCRep.init(KDCRep.java:140) ~[?:1.8.0_161]
at sun.security.krb5.internal.ASRep.init(ASRep.java:64) ~[?:1.8.0_161]
at sun.security.krb5.internal.ASRep.<init>(ASRep.java:59) ~
[?:1.8.0_161] at sun.security.krb5.KrbAsRep.<init>(KrbAsRep.java:60) ~
[?:1.8.0_161] at
sun.security.krb5.KrbAsReqBuilder.send(KrbAsReqBuilder.java:316) ~
[?:1.8.0_161] at
sun.security.krb5.KrbAsReqBuilder.action(KrbAsReqBuilder.java:361) ~
[?:1.8.0_161] at
com.sun.security.auth.module.Krb5LoginModule.attemptAuthentication(Krb
5LoginModule.java:776) ~[?:1.8.0_161] at
```

```
com.sun.security.auth.module.Krb5LoginModule.login(Krb5LoginModule.jav
a:617) ~[?:1.8.0_161] at
sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method) ~
[?:1.8.0_161] at
sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.j
ava:62) ~[?:1.8.0_161] at
sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccess
orImpl.java:43) ~[?:1.8.0_161] at
java.lang.reflect.Method.invoke(Method.java:498) ~[?:1.8.0_161] at
javax.security.auth.login.LoginContext.invoke(LoginContext.java:755) ~
[?:1.8.0_161] at
javax.security.auth.login.LoginContext.access$000(LoginContext.java:19
5) ~[?:1.8.0_161] at
javax.security.auth.login.LoginContext$4.run(LoginContext.java:682) ~
[?:1.8.0_161] at
javax.security.auth.login.LoginContext$4.run(LoginContext.java:680) ~
[?:1.8.0_161] at java.security.AccessController.doPrivileged(Native
Method) ~[?:1.8.0_161] at
javax.security.auth.login.LoginContext.invokePriv(LoginContext.java:68
0) ~[?:1.8.0_161] at
javax.security.auth.login.LoginContext.login(LoginContext.java:587) ~
[?:1.8.0_161] at com.lucidworks.connectors.spnego.SpnegoAuth.<init>
(SpnegoAuth.java:76) ~[?:?] at
crawler.fetch.impl.http.WebFetcher.init(WebFetcher.java:279) ~[?:?]
... 8 more
```

Resolution:

Your keytab's principal name doesn't match the value on the ticket server. Check the principal name for your user.

Connect to Fusion from App Studio

How-to

> Compatible versions: 4.2
>
> Works with: *App Studio*

The most important tags in every page template are the first three, which configure how App Studio connects to Fusion:

- Define which provider to use (the *"search:platform" tag*).
- Create a query from the URL, or any parameters you explicitly declare (the *"search:query" tag*).
- Send the query to the platform and obtain a response (the *"search:response" tag*).

Below is an example from the default template (located at `app/views/search.html` for version 4.0):

```
<search:platform var="platform" conf="platforms.fusion.data">
</search:platform> <search:query var="query" parameters="*" results-
per-page="12"></search:query> <search:response var="response"
platform="platform" query="query"> <widget:spinner></widget:spinner>
</search:response>
```

The `search:platform` *tag*

The `search:platform` tag specifies the endpoint where App Studio obtains its data, pointing to a specific platform configuration endpoint.

See `search:query` for additional available parameters.

In this case we declare a data provider (platform) for a particular Fusion collection and set of parameters to search through an index.

```
<search:platform var="platform" conf="platforms.fusion.data">
</search:platform>
```

See `search:platform` for additional available parameters.

The `search:query` *tag*

The `search:query` tag constructs an App Studio query object. You can choose to build a query from scratch by passing in the parameters you want or you can have it automatically created from the page URL.

Here is how you build a query using all query string parameters, and default to 12 results per page:

```
<search:query var="query" parameters="*" results-per-page="12">
</search:query>
```

If you did not want the user to be able to specify results per page you would simply omit that parameter when building a query from the URL:

```
<search:query var="query" parameters='*,-rpp' results-per-page="12">
</search:query>
```

See `search:query` for additional available parameters.

The `search:response` *tag*

The `search:response` tag brings the platform and the query together.

In the example below we refer to the previously-defined platform and query by variable name:

```
<search:response var="response" platform="platform" query="query">
<widget:spinner></widget:spinner> </search:response>
```

See `search:response` for additional available parameters.

Connect to the Fusion SQL Service from JDBC

How-to

> Compatible versions: 4.2
>
> Works with: *Fusion Connectors*, *Fusion Server*, *Fusion*

The default JDBC properties for connecting to the Fusion SQL service are:

Driver:

```
org.apache.hive.jdbc.HiveDriver
```

Driver JAR:

Unix:

```
/opt/fusion/latest.x/apps/libs/hive-jdbc-shaded-2.1.1.jar
```

Windows:

```
C:\lucidworks\fusion\latest.x\apps\libs\hive-jdbc-shaded-2.1.1.jar
```

JDBC URL:

```
jdbc:hive2://localhost:8768/default;transportMode=http;httpPath=fusion
```

The username and password are the same as the ones you use to authenticate to Fusion.

Crawl a Subset of OneDrive

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

OneDrive is a file hosting service that is part of the Microsoft Office Online services. The Fusion OneDrive connector crawls a OneDrive for Business instance and retrieves data from it for indexing within Fusion.

You can optionally configure the OneDrive connector to crawl only the drives of certain users that you specify.

To limit the crawl to user-specific drives:

1. *Install the OneDrive connector*
2. In the Managed Fusion UI, navigate to the OneDrive connector configuration at **Indexing > Datasources > Add > OneDrive**.
3. In the `User principal name (UPN) filter` field, specify a list of users (user principal names, or UPNs) to retrieve documents from.
    - A user's UPN is the one they use for logging into OneDrive.
    - The `User principal name (UPN) filter` field is an array, so you can set multiple UPNs.

Troubleshooting tips:

- All UPNs are validated, by retrieving the user drives. If the request fails, it is logged in `$fusion_home/var/log/connectors/connectors-rpc/connectors-rpc.log`.
- If validation fails for all UPNs set, then the crawl job does not start.
- If at least one UPN is valid, then the validation succeeds and the job starts. While the job is running, requests to invalid UPNs fail and are logged.
- If more than 10 UPNs are set, then the validation is skipped for performance reasons.

Crawl a Subset of OneDrive

How-to

> Compatible versions: 4.2, 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion Connectors*, *Fusion Server*, *Fusion*

OneDrive is a file hosting service that is part of the Microsoft Office Online services. The Fusion OneDrive connector crawls a OneDrive for Business instance and retrieves data from it for indexing within Fusion.

You can optionally configure the OneDrive connector to crawl only the drives of certain users that you specify.

To limit the crawl to user-specific drives:

1. Install the OneDrive connector:
    - *Install A Connector - Fusion 5.x*
    - *Install A Connector - Fusion 4.x*
2. In the Fusion UI, navigate to the OneDrive connector configuration at **Indexing > Datasources > Add > OneDrive**.
3. In the `User principal name (UPN) filter` field, specify a list of users (user principal names, or UPNs) to retrieve documents from.
    - A user's UPN is the one they use for logging into OneDrive.
    - The `User principal name (UPN) filter` field is an array, so you can set multiple UPNs.

Troubleshooting tips:

- All UPNs are validated, by retrieving the user drives. If the request fails, it is logged in `$fusion_home/var/log/connectors/connectors-rpc/connectors-rpc.log`.
- If validation fails for all UPNs set, then the crawl job does not start.
- If at least one UPN is valid, then the validation succeeds and the job starts. While the job is running, requests to invalid UPNs fail and are logged.
- If more than 10 UPNs are set, then the validation is skipped for performance reasons.

Crawl an Authenticated Website with the Web Connector

How-to

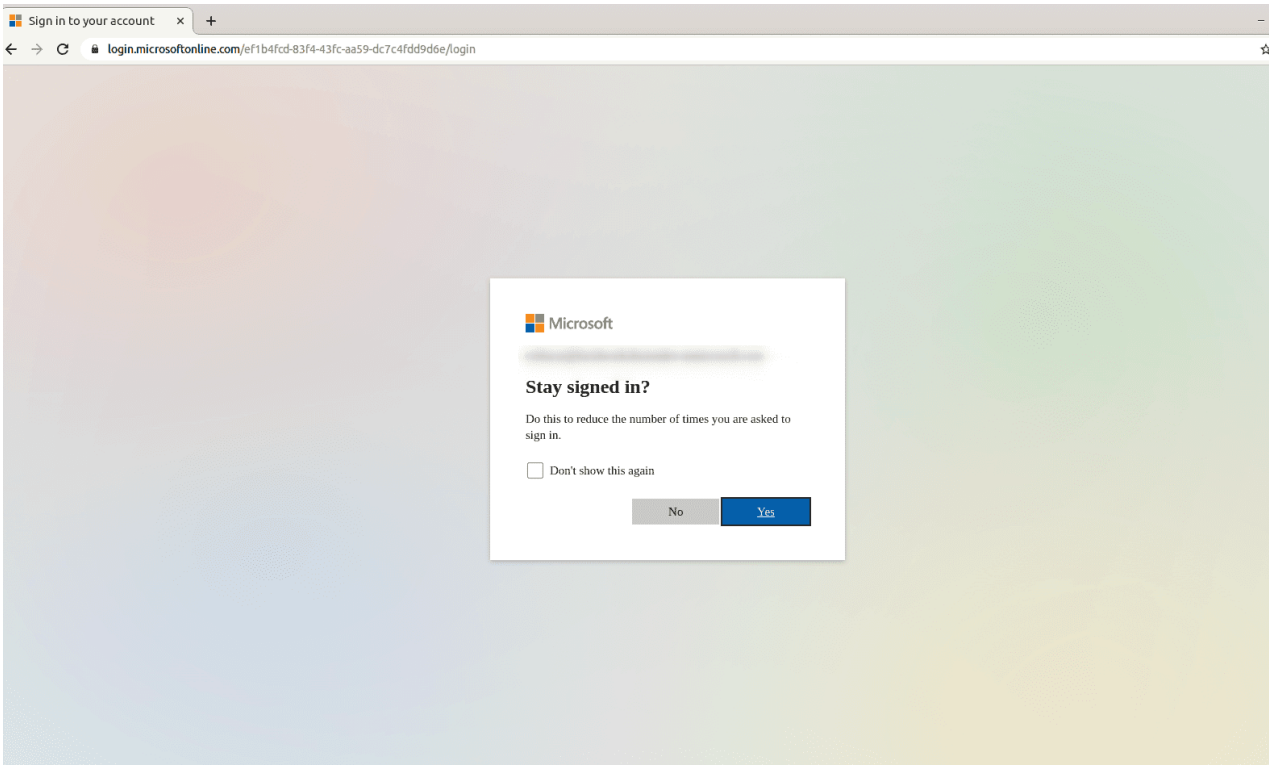
> Works with: *Fusion Connectors*

This article explains how to use the Web V1 connector to crawl a website that is protected by SSO, SAML, or OAuth.

Overview

This example uses SharePoint Online, a website that uses the Microsoft identity platform.

To crawl a website behind a login, configure the Web connector to authenticate with valid login credentials. In this example, there is also a final prompt to stay signed in. Configure the connector to use the **SAML/Smart Form authentication** method.



The Microsoft identity platform also requires JavaScript to authenticate, which means you must also configure the connector to use **JavaScript Evaluation**.

Prerequisites

Before you can configure the Web connector, you must locate and copy the login parameters from the website you're trying to crawl.

1. Open a private or incognito window in your internet browser. This example uses Google Chrome.
2. Navigate to the site that you want to crawl with authentication.
3. Identify the URL for the login page.
   For example, from `http://some-website-with-auth.com`, navigate to the page that displays the login form, then copy the page URL, such as `http://some-website-with-auth.com/sso/login`.
   Use this URL as the value of the `loginUrl` parameter (**URL** in the Fusion UI) explained in *Complex form-based authentication*.
4. On the login page, identify the fields used for inputting the username and password.
   You can do this by right-clicking on the form fields and selecting **Inspect element** to open the developer tools, where the corresponding HTML element is highlighted.
   In most cases it is an `<input>` element that has a `name` attribute and you can specify the field as this name value. For example:

```
<input id="resolving_input" name="login" class="signin-
textfield" autocorrect="off" autocapitalize="off" type="text">
```

5. Add the username field as a parameter to the authentication section of the Web connector. The **Property Name** is `login` and the **Property Value** is the username the connector will use to log in.
6. Add the password field name as the `passwordParamName` (**Password Parameter** in the Fusion UI).
7. On the site login page, right-click **Submit** (or equivalent) and select **Inspect element**.
    - If the button is an `<input type="submit"/>`, then the SmartForm login picks it up automatically.
    - If the button is another element (such as `<button>`, `<a>`, `<div>`, and so on) then you must add a parameter with the special prefix `::submitButtonXPath::`, then add an XPath expression that points to the submit button. For example:
      `::submitButtonXPath::://button[@name='loginButton']`
8. If there is no `name` attribute on the `<input>` elements, then you must specify a parameter to tell the Web connector how to find the input element. You can use any of these special selector formats for the parameter name:

```
;;BY_XPATH;;//input[@id='someId'] ;;BY_ID;;someid
;;BY_NAME;;somename ;;BY_CLASS_NAME;;someCssClassName
;;BY_CSS_SELECTOR;;.div#selector
```

Sometimes your web page asks you a random question, such as `What is the name of your first dog?`

In this case add another special parameter:

```
::WhenXPath::XPath of element to check against::Either
@attributeToCheckAgainst or text to check against the text of the
element::Value To Match::Field selector to set the value of only if
the conditional check matched
```

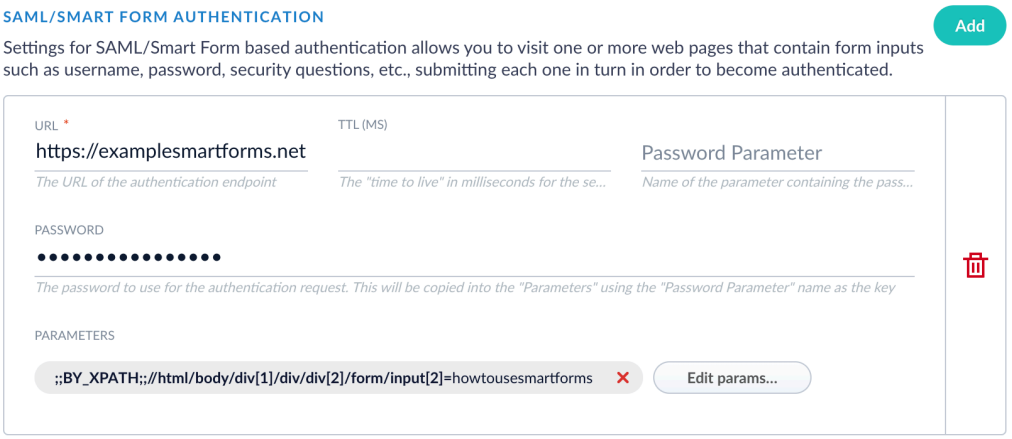
Here is an example of three different parameters where your site might ask one of three questions randomly:

```
::WhenXPath::://div[@tag='Your question']::text::What is the name of
your first dog?::;;BY_ID;;answer ::WhenXPath::://div[@tag='Your
question']::text::In what city were you born?::;;BY_ID;;answer
::WhenXPath::://input[@id='Your question']::@value::In what city were
you born?::;;BY_ID;;answer
```

Configure Web connector

Configure the connector to use login credentials and JavaScript to successfully authenticate through your website.

1. In Fusion, navigate to **Indexing** > **Datasources**.
2. Click **Add**, then select **Web (v1)** (in Fusion 5.12, select **Web (Deprecated)**) from the dropdown.
3. *Configure the datasource* and fill in *required fields*.
4. Expand **Crawl Authorization**, scroll down to **SAML/Smart Form Authentication**, then click **Add**.
    a. Enter the **URL** of the website that requires authentication.
    b. Enter the password to use for the authentication request.
    c. Set the **TTL (ms)** so there is enough time to load the form pieces on the page before Fusion attempts to run the Smart Form step.
    d. Add **Parameters** as needed. For example, if the login page requires you to enter your name in addition to a password, add a parameter, specifying the **Property Name** and **Property Value**. Add a parameter for every required field on the login page.



    e. If applicable, create a **SAML/Smart Form Authentication** for every subsequent login page. These steps trigger from top to bottom.

5. Expand the **JavaScript Evaluation** section and enable **Evaluate JavaScript during SmartForm Authentication**. Enable **Evaluate Javascript** if you would like to use a JavaScript evaluation engine for *all* requests, not just the authentication.

Tip

You can find the **Property Name** and **Property Value** for any given step on the login page. Use your browser's Developer Tools to inspect the input field, then copy the XPath.

Install Chromium dependencies (Fusion 4.x only)

Important

This section only applies to 4.x.x releases. For 5.x.x releases, Chrome dependencies are already installed in the Docker container.

To enable high-performance JavaScript evaluation, you must install some dependencies, then set `f.crawlJS` /"Evaluate JavaScript" *and* `f.useHighPerfJsEval` /"High Performance Mode" to "true".

Enable high-performance mode with Chromium

1. Install the dependencies, using one of the scripts packaged with Fusion:
   - Windows: `https://FUSION_HOST:FUSION_PORT/bin/install-high-perf-web-deps.ps1`
   - Linux and OSX: `https://FUSION_HOST:FUSION_PORT/bin/install-high-perf-web-deps.sh`

   When the script finishes a confirmation message displays:

```
Successfully installed high-performance JS eval mode
dependencies for Lucidworks Fusion web connector.
```

2. In the Fusion UI, configure your Web data source:
    a. Set **Evaluate Javascript** to "true".
    b. Set **High Performance Mode** to "true".

    An additional `f.headlessBrowser` parameter can be set to "false" to display the browser windows during processing. It is "true" by default. Non-headless mode is available only using the "High-performance" mode.
3. Save the datasource configuration.

Note

If Fusion is running on Docker, you must either mount an `shm` directory using the argument `-v /dev/shm:/dev/shm` or use the flag `--shm-size=2g` to use the host's shared memory. The default shm size `64m` will result in failing crawls with logs showing error messages like `org.openqa.selenium.WebDriverException: Failed to decode response from marionette`. See Geckodriver issue 1193 for more details.

Test the configuration

1. In Fusion, navigate to **Indexing** > **Datasources**.
2. Click on the web connector you configured.
3. Set your **Fetch Threads** and **Emit Threads**. It simplifies testing when only one thread is trying to authenticate at a time.
    a. Expand **Crawl Performance**.
    b. Set **Fetch Threads** and **Emit Threads** to `1`.
4. Enable **Diagnostic logging**. Diagnostic mode will allow the HTML contents to print out in the logs for each step of the log on process, so you can use that to see what the browser doing. This can be useful because you can see where the browser was when/if you got into a bad state.
    a. Toggle **Advanced** to view advanced settings.
    b. Enable **Diagnostic mode**.
5. Headless mode (Fusion 4.x only). If you are on a Windows Server or have the ability to install Fusion somewhere an active desktop environment is available, such as Mac or Linux with Desktop, you can use `headless mode = false` to test with an actual web browser.
    a. Expand **JavaScript Evaluation**.
    b. Disable **Headless Browser**.
    c. When you run the crawl, a browser window opens and performs the log in process.
6. Click **Run** to start the datasource job. Review results in the Query Workbench.

Crawl an Authenticated Website with the Web Connector

How-to

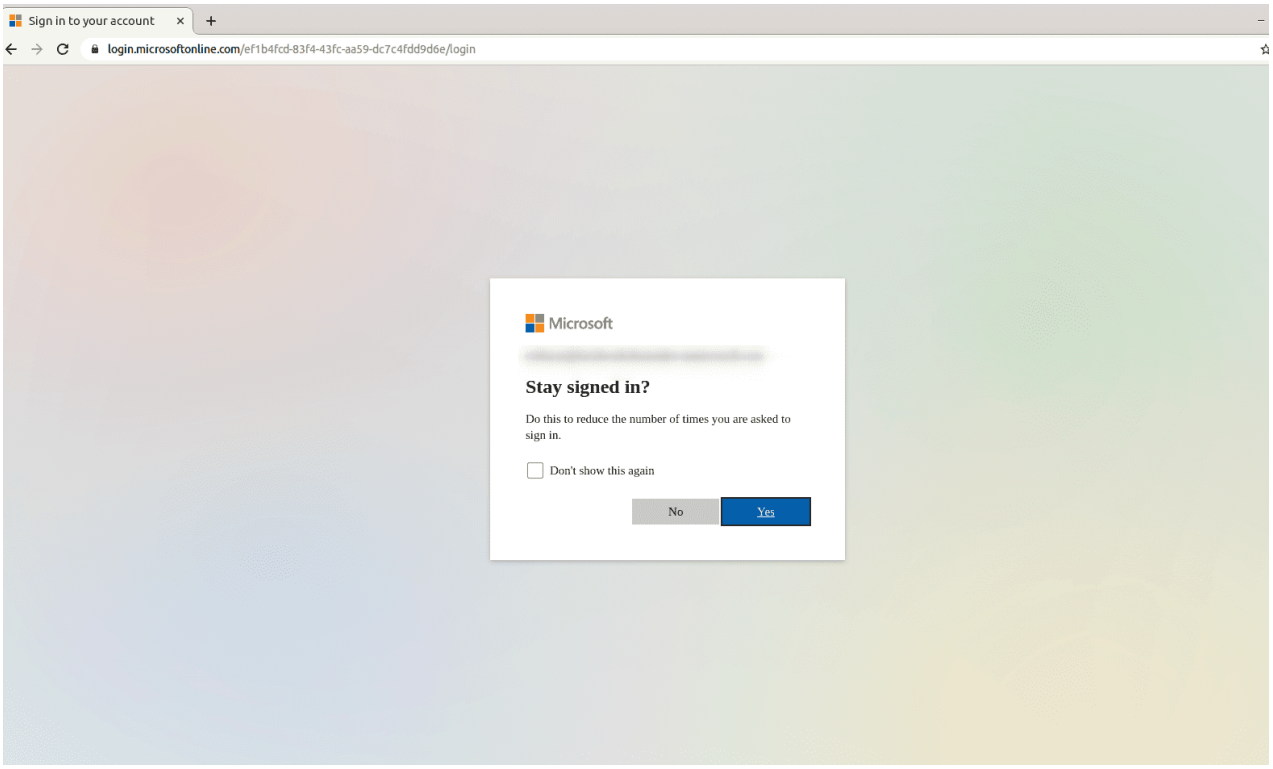
> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

This article explains how to use the Web V1 connector to crawl a website that is protected by SSO, SAML, or OAuth.

Overview

This example uses SharePoint Online, a website that uses the Microsoft identity platform.

To crawl a website behind a login, configure the Web connector to authenticate with valid login credentials. In this example, there is also a final prompt to stay signed in. Configure the connector to use the **SAML/Smart Form authentication** method.



The Microsoft identity platform also requires JavaScript to authenticate, which means you must also configure the connector to use **JavaScript Evaluation**.

Prerequisites

Before you can configure the Web connector, you must locate and copy the login parameters from the website you're trying to crawl.

Authentication with JavaScript Evaluation

To use authentication when JavaScript evaluation is enabled, use the SmartForm (SAML) option because it can sign in to a web site like a typical browser user.

SmartForm sign-in functionality is more powerful when JavaScript evaluation is enabled:

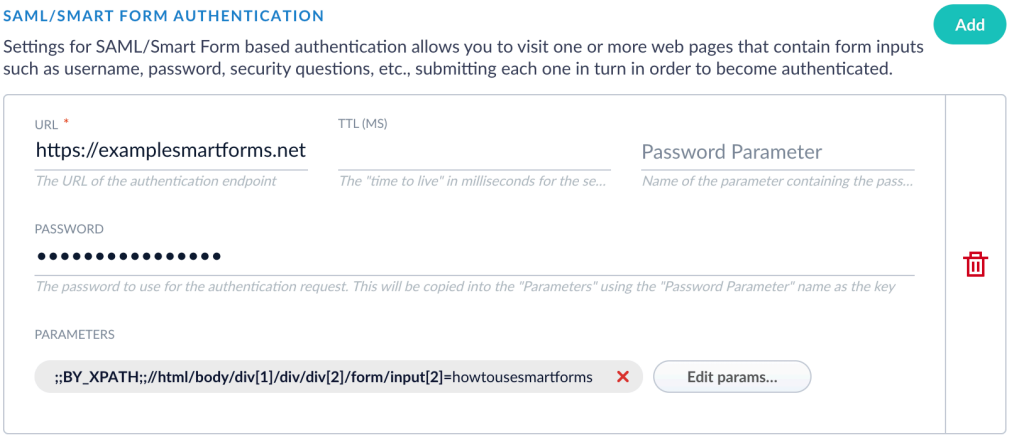
- You can perform sign-in on forms that might be JavaScript rendered.
- You can use a variety of HTML selectors to find the elements to enter as sign-in information. By contrast, when JavaScript evaluation is disabled, you can only provide inputs using the `name` attribute of `<input>` elements.

Configure Web connector

Configure the connector to use login credentials and JavaScript to successfully authenticate through your website.

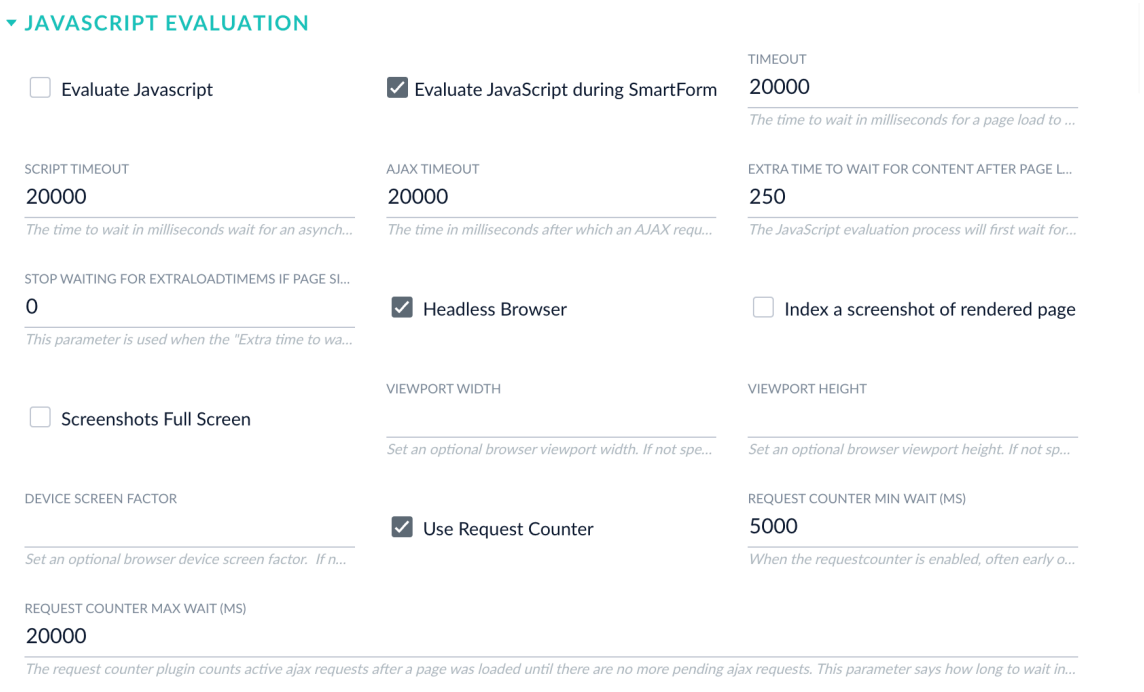
1. In Managed Fusion, navigate to **Indexing** > **Datasources**.
2. Click **Add**, then select **Web (v1)** (in Managed Fusion 5.12, select **Web (Deprecated)**) from the dropdown.
3. *Configure the datasource* and fill in *required fields*.
4. Expand **Crawl Authorization**, scroll down to **SAML/Smart Form Authentication**, then click **Add**.
    a. Enter the **URL** of the website that requires authentication.

b. Enter the password to use for the authentication request.
c. Set the **TTL (ms)** so there is enough time to load the form pieces on the page before Fusion attempts to run the Smart Form step.
d. Add **Parameters** as needed. For example, if the login page requires you to enter your name in addition to a password, add a parameter, specifying the **Property Name** and **Property Value**. Add a parameter for every required field on the login page.



e. If applicable, create a **SAML/Smart Form Authentication** for every subsequent login page. These steps trigger from top to bottom.

5. Expand the **JavaScript Evaluation** section and enable **Evaluate JavaScript during SmartForm Authentication**. Enable **Evaluate Javascript** if you would like to use a JavaScript evaluation engine for *all* requests, not just the authentication.



Tip

You can find the **Property Name** and **Property Value** for any given step on the login page. Use your browser's Developer Tools to inspect the input field, then copy the XPath.

Test the configuration

1. In Fusion, navigate to **Indexing** > **Datasources**.
2. Click on the web connector you configured.
3. Set your **Fetch Threads** and **Emit Threads**. It simplifies testing when only one thread is trying to authenticate at a time.
    a. Expand **Crawl Performance**.
    b. Set **Fetch Threads** and **Emit Threads** to `1`.
4. Enable **Diagnostic logging**. Diagnostic mode will allow the HTML contents to print out in the logs for each step of the log on process, so you can use that to see what the browser doing. This can be useful because you can see where the browser was when/if you got into a bad state.
    a. Toggle **Advanced** to view advanced settings.
    b. Enable **Diagnostic mode**.
5. Click **Run** to start the datasource job. Review results in the Query Workbench.

Crawl JavaScript Web Sites

How-to

> Compatible versions: 4.2, 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion Connectors*, *Fusion Server*, *Fusion*

JavaScript-enabled web sites require a different crawling configuration than plain HTML web sites do.

Enable JavaScript evaluation

To enable JavaScript evaluation, set the `f.crawlJS` /"Evaluate Javascript" parameter to "true". When this option is enabled, the connector crawls links rendered from JavaScript evaluation, using a headless browser by default (see below).

JavaScript evaluation is fastest when using High Performance JavaScript Evaluation mode, which uses the Chromium browser with the Web connector. There is also a High Performance Mode parameter, which uses the Chromium web browser to crawl web sites.

How is "High performance JavaScript evaluation" different than the non-high performance mode? In high performance mode, the following are true:

- JavaScript evaluation is faster.
- JavaScript evaluated content is more accurate.
- You can run in "non-headless" mode to watch the browser crawl your content on a desktop, for debugging purposes.
- You can save screenshots of pages and index them as base64 along with your other web page content.
- The process uses more RAM.
- The process uses more CPU.

Dependencies for High Performance JavaScript Evaluation

Important

This section only applies to 4.x.x releases. For 5.x.x releases, Chrome dependencies are already installed in the Docker container.

To enable high-performance JavaScript evaluation, you must install some dependencies, then set `f.crawlJS` /"Evaluate JavaScript" *and* `f.useHighPerfJsEval` /"High Performance Mode" to "true".

Enable high-performance mode with Chromium

1. Install the dependencies, using one of the scripts packaged with Fusion:
   - Windows: `https://FUSION_HOST:FUSION_PORT/bin/install-high-perf-web-deps.ps1`
   - Linux and OSX: `https://FUSION_HOST:FUSION_PORT/bin/install-high-perf-web-deps.sh`

   When the script finishes a confirmation message displays:

   ```
   Successfully installed high-performance JS eval mode
   dependencies for Lucidworks Fusion web connector.
   ```

2. In the Fusion UI, configure your Web data source:
   a. Set **Evaluate Javascript** to "true".
   b. Set **High Performance Mode** to "true".

   An additional `f.headlessBrowser` parameter can be set to "false" to display the browser windows during processing. It is "true" by default. Non-headless mode is available only using the "High-performance" mode.
3. Save the datasource configuration.

Note

If Fusion is running on Docker, you must either mount an `shm` directory using the argument `-v /dev/shm:/dev/shm` or use the flag `--shm-size=2g` to use the host's shared memory. The default shm size `64m` will result in failing crawls with logs showing error messages like `org.openqa.selenium.WebDriverException: Failed to decode response from marionette`. See Geckodriver issue 1193 for more details.

Screenshots with High Performance Mode

To enable screenshots, configure these parameters:

- `f.takeScreenshot` - Enable screenshots. Default: false.
- `f.screenshotFullscreen` - Take full-screen screenshots. Default: false.

These additional viewport parameters exist mainly to support screenshots, but the viewport settings can also affect the overall output:

- `f.viewportWidth` - View port width in pixels. Default: 800.
- `f.viewportHeight` - View port height in pixels. Default: 600.
- `f.deviceScreenFactor` - Device screen factor. Default: 1 (no scaling). See the Android Screen compatibility overview for a description about device displays.
- `f.simulateMobile` - (advanced property) Tell the browser to "emulate" a mobile browser. Default: false.
- `f.mobileScreenWidth` - (advanced property) Only applicable with `f.simulateMobile`. Sets the screen width in pixels.
- `f.mobileScreenHeight` - (advanced property) Only applicable with `f.simulateMobile`. Sets the screen height in pixels.

Authentication with JavaScript Evaluation

To use authentication when JavaScript evaluation is enabled, use the SmartForm (SAML) option because it can log in to a web site like a typical browser user.

SmartForm login functionality is more powerful when JavaScript evaluation is enabled:

- You can perform login on forms that might be JavaScript rendered.
- You can use a variety of HTML selectors to find the elements to enter as login information. By contrast, when JavaScript evaluation is disabled, you can only provide inputs using the `name` attribute of `<input>` elements.

Configure authentication for JavaScript-enabled crawling

1. Open a private or incognito window in your internet browser. This example uses Google Chrome.
2. Navigate to the site that you want to crawl with authentication.
3. Identify the URL for the login page.
   For example, from `http://some-website-with-auth.com`, navigate to the page that displays the login form, then copy the page URL, such as `http://some-website-with-auth.com/sso/login`.
   Use this URL as the value of the `loginUrl` parameter (**URL** in the Fusion UI) explained in *Complex form-based authentication*.
4. On the login page, identify the fields used for inputting the username and password.
   You can do this by right-clicking on the form fields and selecting **Inspect element** to open the developer tools, where the corresponding HTML element is highlighted.
   In most cases it is an `<input>` element that has a `name` attribute and you can specify the field as this name value. For example:

```
<input id="resolving_input" name="login" class="signin-
textfield" autocorrect="off" autocapitalize="off" type="text">
```

5. Add the username field as a parameter to the authentication section of the Web connector. The **Property Name** is `login` and the **Property Value** is the username the connector will use to log in.
6. Add the password field name as the `passwordParamName` (**Password Parameter** in the Fusion UI).
7. On the site login page, right-click **Submit** (or equivalent) and select **Inspect element**.
    - If the button is an `<input type="submit"/>`, then the SmartForm login picks it up automatically.
    - If the button is another element (such as `<button>`, `<a>`, `<div>`, and so on) then you must add a parameter with the special prefix `::submitButtonXPath::`, then add an XPath expression that points to the submit button. For example:
      `::submitButtonXPath::://button[@name='loginButton']`
8. If there is no `name` attribute on the `<input>` elements, then you must specify a parameter to tell the Web connector how to find the input element. You can use any of these special selector formats for the parameter name:

```
;;BY_XPATH;;//input[@id='someId'] ;;BY_ID;;someid
;;BY_NAME;;somename ;;BY_CLASS_NAME;;someCssClassName
;;BY_CSS_SELECTOR;;.div#selector
```

Sometimes your web page asks you a random question, such as `What is the name of your first dog?`

In this case add another special parameter:

```
::WhenXPath::XPath of element to check against::Either
@attributeToCheckAgainst or text to check against the text of the
element::Value To Match::Field selector to set the value of only if
the conditional check matched
```

Here is an example of three different parameters where your site might ask one of three questions randomly:

```
::WhenXPath::://div[@tag='Your question']::text::What is the name of your first dog?::;;BY_ID;;answer ::WhenXPath::://div[@tag='Your question']::text::In what city were you born?::;;BY_ID;;answer ::WhenXPath::://input[@id='Your question']::@value::In what city were you born?::;;BY_ID;;answer
```

Debug the JavaScript Evaluation Stage using Non-headless Chromium

When testing the Web connector with Chromium, it helps to access Fusion through a GUI-enabled browser. Configure a Web data source with your website, enable advanced mode, set the **Crawl Performance** > **Fetch Threads** to `1`, and uncheck **Javascript Evaluation** > **Headless Browser**.

This results in the Web fetcher using a single instance of Chromium in a window where you can see the fetch documents. This is helpful if you are getting an unexpected result from the Chromium evaluation stage.

Crawl JavaScript websites

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

The Web connector retrieves data from a web site using HTTP and starting from a specified URL.

Managed Fusion releases 5.5 - 5.6.0 use the *Web V1 connector*. Managed Fusion releases 5.6.1 and later use the *Web V2 connector*. JavaScript Evaluation is not supported in the Web V2 connector.

JavaScript-enabled web sites require a different crawling configuration than plain HTML web sites do.

Enable JavaScript evaluation

To enable JavaScript evaluation, set the `f.crawlJS` /"Evaluate Javascript" parameter to "true". When this option is enabled, the connector crawls links rendered from JavaScript evaluation, using the `f.headlessBrowser` parameter set to "true".

JavaScript evaluation is optimal when using High Performance JavaScript Evaluation mode, which uses the Chromium browser with the Web connector. There is also a High Performance Mode parameter, which uses the Chromium web browser to crawl web sites.

How is "High performance JavaScript evaluation" different than the non-high performance mode? In high performance mode, the following are true:

- JavaScript evaluation is faster.
- JavaScript evaluated content is more accurate.
- You can run in "non-headless" mode to watch the browser crawl your content on a desktop for debugging purposes.
- You can save screenshots of pages and index them as base64 along with your other web page content.
- The process uses more RAM.
- The process uses more CPU.

Dependencies for high performance JavaScript evaluation

Chrome dependencies are installed in the Docker container.

Screenshots with high performance mode

To enable screenshots, configure these parameters:

- `f.takeScreenshot` to enable screenshots. Default: false.
- `f.screenshotFullscreen` to take full-screen screenshots. Default: false.

These additional viewport parameters exist mainly to support screenshots, but the viewport settings can also affect the overall output:

- `f.viewportWidth` to view port width in pixels. Default: 800.
- `f.viewportHeight` to view port height in pixels. Default: 600.
- `f.deviceScreenFactor` is the device screen factor. Default: 1 (no scaling). See the Android Screen compatibility overview for a description about device displays.
- `f.simulateMobile` is the advanced property that tells the browser to "emulate" a mobile browser. Default: false.
- `f.mobileScreenWidth` is the advanced property that is only applicable with `f.simulateMobile` and sets the screen width in pixels.
- `f.mobileScreenHeight` is the advanced property that is only applicable with `f.simulateMobile` and sets the screen height in pixels.

Authentication with JavaScript Evaluation

To use authentication when JavaScript evaluation is enabled, use the SmartForm (SAML) option because it can sign in to a web site like a typical browser user.

SmartForm sign-in functionality is more powerful when JavaScript evaluation is enabled:

- You can perform sign-in on forms that might be JavaScript rendered.
- You can use a variety of HTML selectors to find the elements to enter as sign-in information. By contrast, when JavaScript evaluation is disabled, you can only provide inputs using the `name` attribute of `<input>` elements.

Configure authentication for JavaScript-enabled crawling

1. Launch Chrome.
2. Click **File** > **New Private Window**.
3. Navigate to the site that you want to crawl with authentication.
   For example, navigate to `http://some-website-with-auth.com`.
4. Identify the URL for the sign-in page.
   For example, from `http://some-website-with-auth.com`, navigate to the page that displays the sign-in form, then copy the page URL, such as `http://some-website-with-auth.com/sso/login`.
   Use this URL as the value of the `loginUrl` parameter which is the **URL** field in the Managed Fusion UI.
   For more information, see *Complex form-based authentication*.
5. On the sign-in page, identify the fields used for inputting the username and password.
   You can do this by right-clicking on the form fields and selecting **Inspect element** to open the developer tools, where the corresponding HTML element is highlighted.
   In most cases it is an `<input>` element that has a `name` attribute and you can specify the field as this name value. For example:

```
<input id="resolving_input" name="login" class="signin-
textfield" autocorrect="off" autocapitalize="off" type="text">
```

6. Add the username field as a Property to the SmartForm sign-in. That is, add a Property, where the Property Name is `login` and the Property Value is the username.
7. Add the password field name as the `passwordParamName`, which is the **Password Parameter** in the Managed Fusion UI.
8. On the site sign-in page, right-click **Submit** and click **Inspect element**.
   - If the button is an `<input type="submit"/>`, then the SmartForm sign-in picks it up automatically.
   - If the button is another element (such as `<button>`, `<a>`, `<div>`, and so on), then you must add a parameter whose name has the special prefix `::submitButtonXPath::` and then add an XPath expression that points to the submit button. For example:
     `::submitButtonXPath::://button[@name='loginButton']`
9. If there is no `name` attribute on the `<input>` elements, then you must specify a parameter to tell the Web connector how to find the input element. You can use any of these special selector formats for the parameter name:

```
;;BY_XPATH;;//input[@id='someId'] ;;BY_ID;;someid
;;BY_NAME;;somename ;;BY_CLASS_NAME;;someCssClassName
;;BY_CSS_SELECTOR;;.div#selector
```

Sometimes your Web page asks you a random question, such as `What is the name of your first dog?`

In this case we add another special parameter:

```
::WhenXPath::XPath of element to check against::Either
@attributeToCheckAgainst or text to check against the text of the
element::Value To Match::Field selector to set the value of only if
the conditional check matched
```

Here is an example of three different parameters where our site might ask one of three questions randomly:

```
::WhenXPath::///div[@tag='Your question']::text::What is the name of
your first dog?::;;BY_ID;;answer ::WhenXPath::///div[@tag='Your
question']::text::In what city were you born?::;;BY_ID;;answer
::WhenXPath::///input[@id='Your question']::@value::In what city were
you born?::;;BY_ID;;answer
```

Debug the JavaScript Evaluation Stage using Non-headless Chromium

When testing the Web connector with Chromium, it helps to install Managed Fusion on a workstation with desktop abilities, such as Windows, Mac, or Linux with a desktop. Then configure a Web datasource with your website, enable advanced mode, set the **Crawl Performance** > **Fetch Threads** to `1`, and uncheck **Javascript Evaluation** > **Headless Browser**.

This results in the Web fetcher using a single instance of Chromium in a window where you can see the fetch documents. This is helpful if you are getting an unexpected result from the Chromium evaluation stage.

Crawl REST APIs using the Web connector

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

This article describes how to crawl REST API endpoints associated with Web connectors using Managed Fusion. For more information about available Web connectors, see *Connectors Configuration Reference*.

Before crawling a REST API endpoint, the following prerequisites must be met:

- All endpoints are available using *bulk start links* or a *sitemap*
- The response data is in a parseable format (JSON, XML, etc.)

Options

Using bulk start links

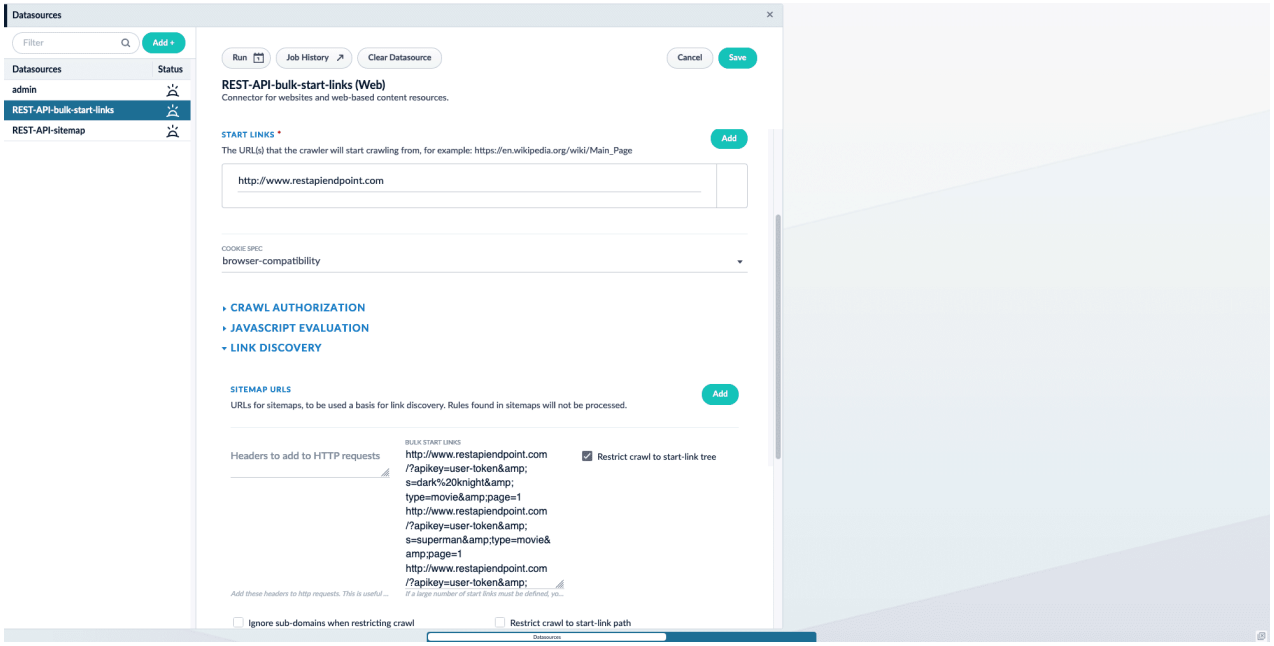
If you have a small number of endpoints you want to crawl, enter each endpoint as a bulk start link.

To crawl the API endpoints using bulk start links:

1. Add a new Web connector datasource. To learn how to configure a new datasource, see *Configure a New Datasource*.
2. Under Start links, enter the main domain that contains the sitemap. For example, `http://www.restapiendpoint.com`.
3. In the Link discovery section under Bulk Start Links, enter the URLs you want to crawl. Separate links with a new line. For example:

```
http://www.restapiendpoint.com/?apikey=user-
token&amp;s=dark%20knight&amp;type=movie&amp;page=1
http://www.restapiendpoint.com/?apikey=user-
token&amp;s=superman&amp;type=movie&amp;page=1
http://www.restapiendpoint.com/?apikey=user-
token&amp;s=superman&amp;type=movie&amp;page=2
```

4. Save and run the job.
5. Once complete, check the *results* in the Index Workbench.



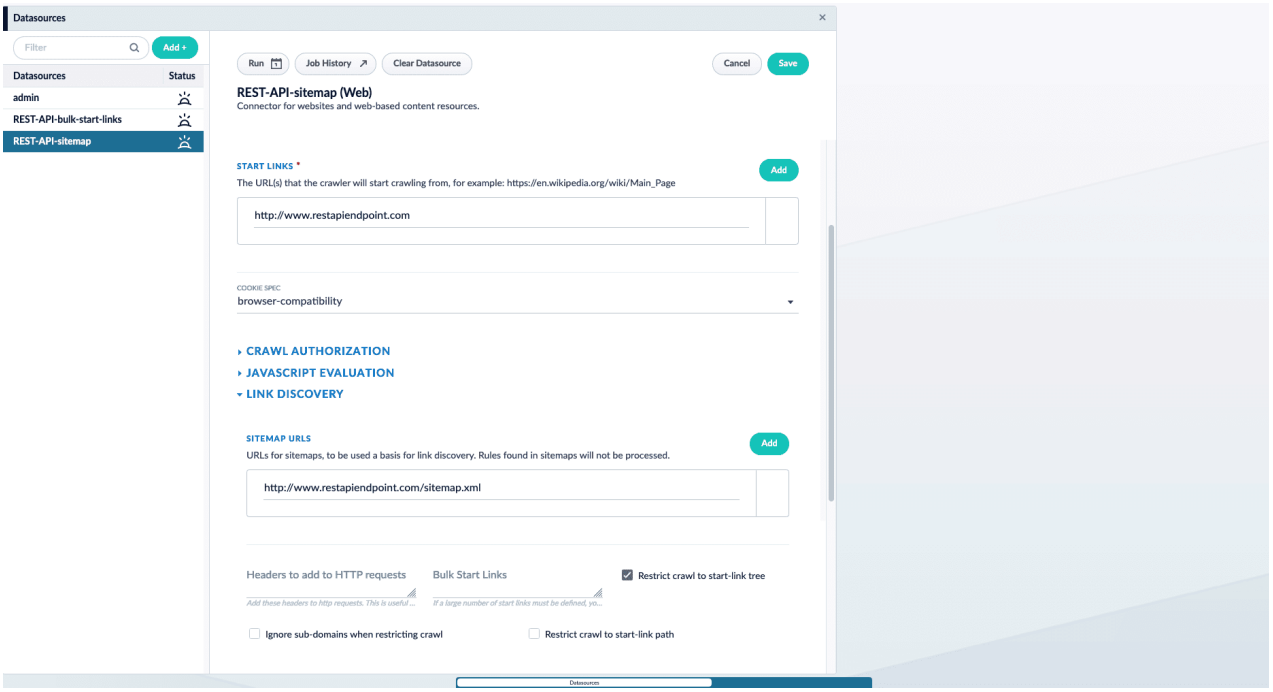
Using a sitemap

If you have a large number of endpoints you want to crawl, use a sitemap containing the API endpoint locations. This is also helpful if someone without access to Managed Fusion maintains the list of endpoint URLs. An example sitemap:

```
<?xml version="1.0" encoding="UTF-8"?> <urlset
xmlns="http://www.sitemaps.org/schemas/sitemap/0.9"> <url>
<loc>http://www.restapiendpoint.com/?apikey=user-
token&amp;s=dark%20knight&amp;type=movie&amp;page=1</loc>
<lastmod>2021-07-21</lastmod> <changefreq>daily</changefreq>
<priority>0.8</priority> </url> <url>
<loc>http://www.restapiendpoint.com/?apikey=user-
token&amp;s=superman&amp;type=movie&amp;page=1</loc> <lastmod>2021-07-
21</lastmod> <changefreq>daily</changefreq> <priority>0.8</priority>
</url> <url> <loc>http://www.restapiendpoint.com/?apikey=user-
token&amp;s=superman&amp;type=movie&amp;page=2</loc> <lastmod>2021-07-
21</lastmod> <changefreq>daily</changefreq> <priority>0.8</priority>
</url> <url> <loc>http://www.restapiendpoint.com/?apikey=user-
token&amp;s=superman&amp;type=movie&amp;page=3</loc> <lastmod>2021-07-
21</lastmod> <changefreq>daily</changefreq> <priority>0.8</priority>
</url> <url> <loc>http://www.restapiendpoint.com/?apikey=user-
token&amp;s=batman&amp;type=movie&amp;page=1</loc> <lastmod>2021-07-
21</lastmod> <changefreq>daily</changefreq> <priority>0.8</priority>
</url> </urlset>
```

To crawl the API endpoints using the sitemap:

1. Add a new Web connector datasource. To learn how to configure a new datasource, see *Configure a New Datasource*.
2. Under Start links, enter the main domain that contains the sitemap. For example, `http://www.restapiendpoint.com`.
3. In the Link discovery section under Sitemap URLs, click **Add**.
4. Enter the URL of the sitemap. For example, `http://www.restapiendpoint.com/sitemap.xml`.
5. Save and run the job.
6. Once complete, check the *results* in the Index Workbench.



Results

Both the *bulk start links* and *sitemap* options achieve the same result. Managed Fusion indexes the JSON response provided at the endpoints. If an array of JSON objects is available, Managed Fusion indexes each object and an individual document.

For example, Managed Fusion creates three documents from the JSON response below:

```
{ "Search": [{ "Title": "Batman v Superman: Dawn of Justice", "Year":
"2016", "imdbID": "tt2975590", "Type": "movie", "Poster":
"https://m.media-
amazon.com/images/M/MV5BYThjYzcyYzItNTVjNy00NDk0LTgwMWQtYjMwNmNlNWJhMz
MyXkEyXkFqcGdeQXVyMTQxNzMzNDI@._V1_SX300.jpg" }, { "Title": "Superman
Returns", "Year": "2006", "imdbID": "tt0348150", "Type": "movie",
"Poster": "https://m.media-
amazon.com/images/M/MV5BNzY2ZDQ2MTctYzlhOC00MWJhLTgxMmItMDgzNDQwMDdhOW
I2XkEyXkFqcGdeQXVyNjc1NTYyMjg@._V1_SX300.jpg" }, { "Title":
"Superman", "Year": "1978", "imdbID": "tt0078346", "Type": "movie",
"Poster": "https://m.media-
amazon.com/images/M/MV5BMzA0YWMwMTUtMTVhNC00NjRkLWE2ZTgtOWEzNjJhYzNiMT
lkXkEyXkFqcGdeQXVyNjc1NTYyMjg@._V1_SX300.jpg" }], "totalResults": "3",
"Response": "True" }
```

Crawl REST APIs using the Web connector

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion*, *Fusion Connectors*

This article describes how to crawl REST API endpoints associated with Web connectors using Fusion. For more information about available Web connectors, see *Fusion connectors*.

Before crawling a REST API endpoint, the following prerequisites must be met:

- All endpoints are available using *bulk start links* or a *sitemap*
- The response data is in a parseable format (JSON, XML, etc.)

Options

Using bulk start links

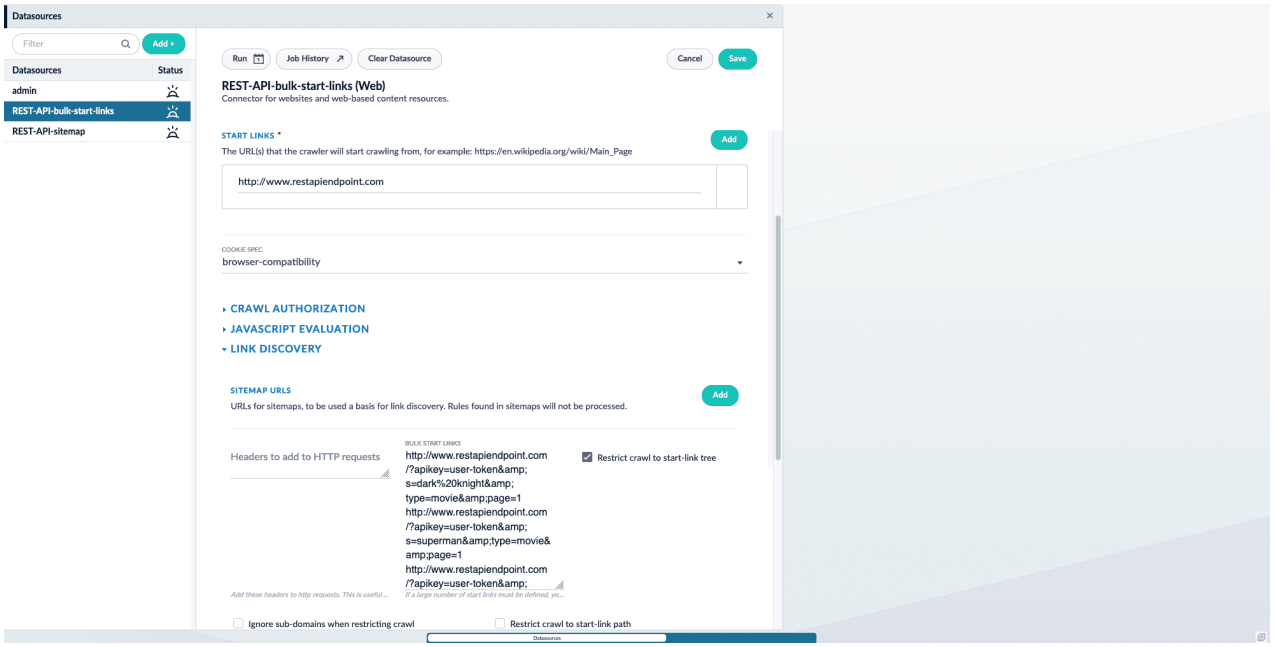
If you have a small number of endpoints you want to crawl, enter each endpoint as a bulk start link.

To crawl the API endpoints using bulk start links:

1. Add a new Web connector datasource. To learn how to configure a new datasource, see *Configure a New Datasource*.
2. Under **Start links**, enter the main domain that contains the sitemap. For example, `http://www.restapiendpoint.com`.
3. In the **Link discovery** section under **Bulk Start Links**, enter the URLs you want to crawl. Separate links with a new line. For example:

```
http://www.restapiendpoint.com/?apikey=user-
token&amp;s=dark%20knight&amp;type=movie&amp;page=1
http://www.restapiendpoint.com/?apikey=user-
token&amp;s=superman&amp;type=movie&amp;page=1
http://www.restapiendpoint.com/?apikey=user-
token&amp;s=superman&amp;type=movie&amp;page=2
```

4. Save and run the job.
5. Once complete, check the *results* in the Index Workbench.



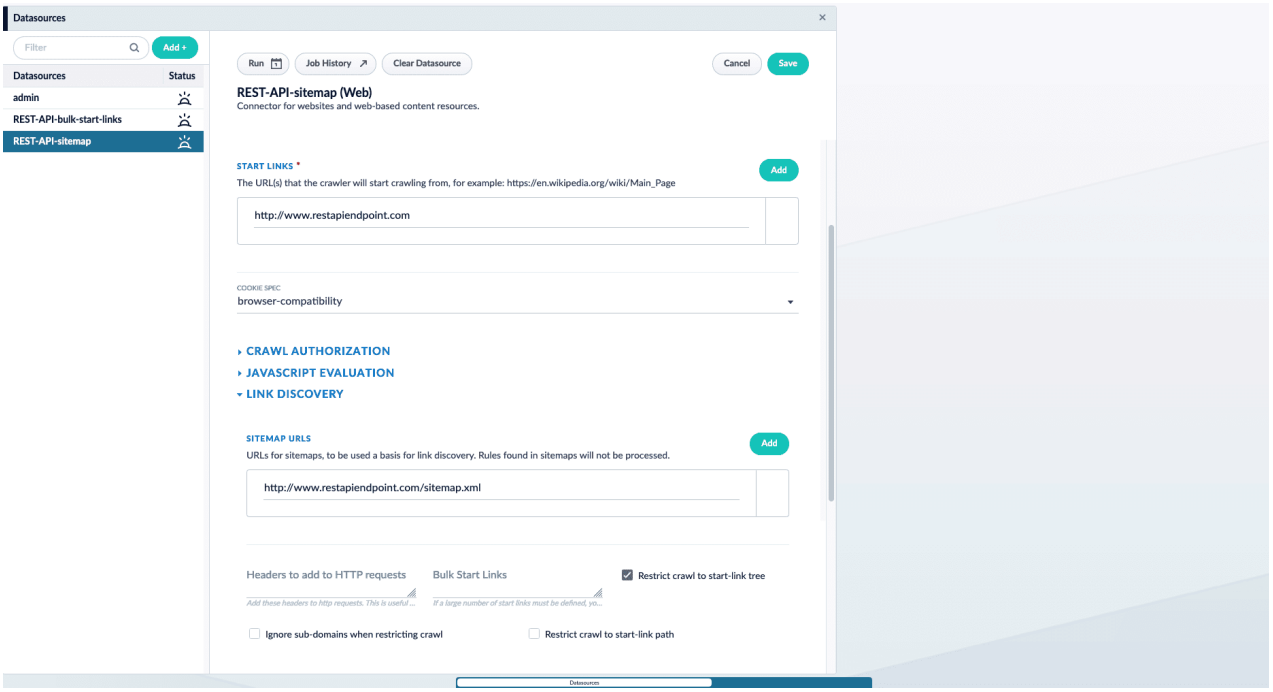
Using a sitemap

If you have a large number of endpoints you want to crawl, use a sitemap containing the API endpoint locations. This is also helpful if someone without access to Fusion maintains the list of endpoint URLs. An example sitemap:

```
<?xml version="1.0" encoding="UTF-8"?> <urlset
xmlns="http://www.sitemaps.org/schemas/sitemap/0.9"> <url>
<loc>http://www.restapiendpoint.com/?apikey=user-
token&amp;s=dark%20knight&amp;type=movie&amp;page=1</loc>
<lastmod>2021-07-21</lastmod> <changefreq>daily</changefreq>
<priority>0.8</priority> </url> <url>
<loc>http://www.restapiendpoint.com/?apikey=user-
token&amp;s=superman&amp;type=movie&amp;page=1</loc> <lastmod>2021-07-
21</lastmod> <changefreq>daily</changefreq> <priority>0.8</priority>
</url> <url> <loc>http://www.restapiendpoint.com/?apikey=user-
token&amp;s=superman&amp;type=movie&amp;page=2</loc> <lastmod>2021-07-
21</lastmod> <changefreq>daily</changefreq> <priority>0.8</priority>
</url> <url> <loc>http://www.restapiendpoint.com/?apikey=user-
token&amp;s=superman&amp;type=movie&amp;page=3</loc> <lastmod>2021-07-
21</lastmod> <changefreq>daily</changefreq> <priority>0.8</priority>
</url> <url> <loc>http://www.restapiendpoint.com/?apikey=user-
token&amp;s=batman&amp;type=movie&amp;page=1</loc> <lastmod>2021-07-
21</lastmod> <changefreq>daily</changefreq> <priority>0.8</priority>
</url> </urlset>
```

To crawl the API endpoints using the sitemap:

1. Add a new Web connector datasource. To learn how to configure a new datasource, see *Configure a New Datasource*.
2. Under **Start links**, enter the main domain that contains the sitemap. For example, `http://www.restapiendpoint.com`.
3. In the **Link discovery** section under **Sitemap URLs**, click the **Add** button.
4. Enter the URL of the sitemap. For example, `http://www.restapiendpoint.com/sitemap.xml`.
5. Save and run the job.
6. Once complete, check the *results* in the Index Workbench.



Results

Both options above achieve the same result. Fusion indexes the JSON response provided at the endpoints. If an array of JSON objects is available, Fusion indexes each object and an individual document.

For example, Fusion creates three documents from the JSON response below:


```
{ "Search": [{ "Title": "Batman v Superman: Dawn of Justice", "Year":
"2016", "imdbID": "tt2975590", "Type": "movie", "Poster":
"https://m.media-
amazon.com/images/M/MV5BYThjYzcyYzItNTVjNy00NDk0LTgwMWQtYjMwNmNlNWJhMz
MyXkEyXkFqcGdeQXVyMTQxNzMzNDI@._V1_SX300.jpg" }, { "Title": "Superman
Returns", "Year": "2006", "imdbID": "tt0348150", "Type": "movie",
"Poster": "https://m.media-
amazon.com/images/M/MV5BNzY2ZDQ2MTctYzlhOC00MWJhLTgxMmItMDgzNDQwMDdhOW
I2XkEyXkFqcGdeQXVyNjc1NTYyMjg@._V1_SX300.jpg" }, { "Title":
"Superman", "Year": "1978", "imdbID": "tt0078346", "Type": "movie",
"Poster": "https://m.media-
amazon.com/images/M/MV5BMzA0YWMwMTUtMTVhNC00NjRkLWE2ZTgtOWEzNjJhYzNiMT
lkXkEyXkFqcGdeQXVyNjc1NTYyMjg@._V1_SX300.jpg" }], "totalResults": "3",
"Response": "True" }
```

Crawl SharePoint Online with minimal throttling

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

SharePoint Online is a shared cloud service. Indexing content with SharePoint Online is much slower than crawling SharePoint On-Premises servers. SharePoint Online throttles the traffic heavily, keeping the SharePoint services responsive and healthy.

Microsoft has published a purposely vague article, which does not include a throttling algorithm, for further understanding. See How to Avoid Throttling .

Based on the information provided, this article discusses the techniques currently used to eliminate throttling. These recommendations are subject to change, as this is an *evolving topic*.

Update the User Agent String to Link to an Azure App

Each SharePoint datasource requires your company name in the user agent string.

Example:

```
ISV|AcmeInc|Fusion/5.3
```

This “decorates” traffic uniquely so Microsoft knows exactly who API calls are coming from.

Warning

Do not use the default user agent string. It does not *uniquely identify* your company’s traffic.

Divide and Conquer with Multiple Service Accounts

SharePoint Online rate limits are due to multiple threads accessing SharePoint APIs concurrently. SharePoint sees that sort of traffic and throttles it heavily.

To reduce the traffic and throttling:

- Split SharePoint site collections between datasources.
- Create multiple SharePoint Online service accounts and spread them across the datasources.

Example -configured SharePoint online optimized connector datasource **SPO_DS_1**:

```
Start links: https://tenant.sharepoint.com/sites/test1
https://tenant.sharepoint.com/sites/test2
https://tenant.sharepoint.com/sites/test3
https://tenant.sharepoint.com/sites/test4
https://tenant.sharepoint.com/sites/test5 Service account:
service.account@tenant.onmicrosoft.com Fetch Threads: 5 Number of
prefetch threads: 10 User agent: ISV|YourCompanyName|Fusion/5.x.x
```

The crawling will begin with great speed. However, eventually errors are generated in the logs:

```
Error message: [429 TOO MANY REQUESTS]
```

Divide and conquer:

1. Create five service accounts, exactly like the first.
2. Create five datasources. Give each datasource its very own service account and a subset of the SharePoint site collections that are to be crawled.
3. Set the fetch threads to 1.

CONFIGURATION RESULTS:

- SPO_DS_1:
  Start links: **https://tenant.sharepoint.com/sites/test1**
  Service account: **service.account@tenant1.onmicrosoft.com**
  Fetch Threads: **1**
  Number of prefetch threads: **1**
- SPO_DS_2:
  Start links: **https://tenant.sharepoint.com/sites/test2**
  Service account: **service.account@tenant2.onmicrosoft.com**
  Fetch Threads: **1**
  Number of prefetch threads: **1**
- SPO_DS_3:
  Start links: **https://tenant.sharepoint.com/sites/test3**
  Service account: **service.account@tenant3.onmicrosoft.com**
  Fetch Threads: **1**
  Number of prefetch threads: **1**
- SPO_DS_4:
  Start links: **https://tenant.sharepoint.com/sites/test4**
  Service account: **service.account@tenant4.onmicrosoft.com**
  Fetch Threads: **1**
  Number of prefetch threads: **1**
- SPO_DS_5:
  Start links: **https://tenant.sharepoint.com/sites/test5**
  Service account: **service.account@tenant5.onmicrosoft.com**
  Fetch Threads: **1**
  Number of prefetch threads: **1**

The idea is to reduce rate limiting. Microsoft will see requests coming from five accounts rather than a single account. To prevent multiple SharePoint jobs from running concurrently, chain jobs.

Note

The key is to prevent Managed Fusion from making too many SharePoint API requests to the SharePoint Online servers concurrently. If too aggressive, Microsoft will throttle.

Lucidworks Job Scheduler to Limit Concurrent Datasources

In addition to limiting a single datasource from making to many connections as once, limit the number of SharePoint Online datasources running concurrently. To avoid too many concurrent datasources running, use the Lucidworks Job Scheduler to chain SharePoint datasources to run one at a time.

See *Schedule a Job*

Use the **“Trigger job after another data source completes”** feature.

Example single datasource at a time:
SPO_DS_1 Schedule: Every day at 06:00:00
SPO_DS_2 Schedule: Trigger job upon completion of SPO_DS_1
SPO_DS_3 Schedule: Trigger job upon completion of SPO_DS_2
SPO_DS_4 Schedule: Trigger job upon completion of SPO_DS_3
SPO_DS_5 Schedule: Trigger job upon completion of SPO_DS_4

In the above case Job 2 (SPO_DS_2) is triggered when Job 1 (SPO_DS_1) completes and so on.

For practical purposes one datasource crawling may be too slow. To increase the number of concurrent jobs use '**MaxConcurrentSharePointJobs**'.

Example 'MaxConcurrentSharePointJobs=3':
SPO_DS_1 Schedule: Every day at 06:00:00
SPO_DS_2 Schedule: Trigger job upon completion of SPO_DS_1
SPO_DS_3 Schedule: Every day at 06:00:00
SPO_DS_4 Schedule: Trigger job upon completion of SPO_DS_3
SPO_DS_5 Schedule: Every day at 06:00:00
SPO_DS_6 Schedule: Trigger job upon completion of SPO_DS_5

Crawl SharePoint Online with Minimal Throttling

How-to

> Compatible versions: 4.2, 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion Connectors*, *Fusion Server*, *Fusion*

SharePoint Online is a shared cloud service. Indexing content with SharePoint Online is much slower than crawling SharePoint On-Premises servers. SharePoint Online throttles the traffic heavily, keeping the SharePoint services responsive and healthy.

Microsoft has published a purposely vague article, which does not include a throttling algorithm, for further understanding How to Avoid Throttling .

Based on the information provided, we will discuss the techniques currently used to eliminate throttling. These recommendations are subject to change, as this is an *evolving topic*.

Update the User Agent String to Link to an Azure App

Each SharePoint datasource requires your company name in the user agent string.

Example:

```
ISV|AcmeInc|Fusion/5.3
```

This “decorates” traffic uniquely so Microsoft will know exactly who API calls are coming from.

Warning

Do not use the default user agent string. It will not *uniquely identify* your company’s traffic.

Divide and Conquer with Multiple Service Accounts

SharePoint Online rate limits are due to multiple threads accessing SharePoint APIs concurrently. SharePoint sees that sort of traffic and throttles it heavily.

To reduce the traffic and throttling:

- Split SharePoint site collections between datasources.
- Create multiple SharePoint Online service accounts and spread them across the datasources.

Example -configured SharePoint online optimized connector datasource **SPO_DS_1**:

```
Start links: https://tenant.sharepoint.com/sites/test1
https://tenant.sharepoint.com/sites/test2
https://tenant.sharepoint.com/sites/test3
https://tenant.sharepoint.com/sites/test4
https://tenant.sharepoint.com/sites/test5 Service account:
service.account@tenant.onmicrosoft.com Fetch Threads: 5 Number of
prefetch threads: 10 User agent: ISV|YourCompanyName|Fusion/5.x.x
```

The crawling will begin with great speed. However, eventually errors are generated in the logs:

```
Error message: [429 TOO MANY REQUESTS]
```

DIVIDE AND CONQUER:

1. Create five service accounts, exactly like the first.
2. Create five datasources. Give each datasource its very own service account and a subset of the SharePoint site collections that are to be crawled.
3. Set the fetch threads to 1.

Configuration Results:

- SPO_DS_1:
  Start links: **https://tenant.sharepoint.com/sites/test1**
  Service account: **service.account@tenant1.onmicrosoft.com**
  Fetch Threads: **1**
  Number of prefetch threads: **1**
- SPO_DS_2:
  Start links: **https://tenant.sharepoint.com/sites/test2**
  Service account: **service.account@tenant2.onmicrosoft.com**
  Fetch Threads: **1**
  Number of prefetch threads: **1**
- SPO_DS_3:
  Start links: **https://tenant.sharepoint.com/sites/test3**
  Service account: **service.account@tenant3.onmicrosoft.com**
  Fetch Threads: **1**
  Number of prefetch threads: **1**
- SPO_DS_4:
  Start links: **https://tenant.sharepoint.com/sites/test4**
  Service account: **service.account@tenant4.onmicrosoft.com**
  Fetch Threads: **1**
  Number of prefetch threads: **1**
- SPO_DS_5:
  Start links: **https://tenant.sharepoint.com/sites/test5**
  Service account: **service.account@tenant5.onmicrosoft.com**
  Fetch Threads: **1**
  Number of prefetch threads: **1**

The idea is to reduce rate limiting. Microsoft will see requests coming from five accounts rather than a single account. To prevent multiple SharePoint jobs from running concurrently, chain jobs.

Note

The key is to prevent Fusion from making too many SharePoint API requests to the SharePoint Online servers concurrently. If too aggressive, Microsoft will throttle.

Lucidworks Job Scheduler to Limit Concurrent Datasources

In addition to limiting a single datasource from making to many connections as once, limit the number of SharePoint Online datasources running concurrently. To avoid too many concurrent datasources running, use the Lucidworks Job Scheduler to chain SharePoint datasources to run one at a time.

See *Schedule a Job*

Use the **"Trigger job after another data source completes"** feature.

Example single datasource at a time:
SPO_DS_1 Schedule: Every day at 06:00:00
SPO_DS_2 Schedule: Trigger job upon completion of SPO_DS_1
SPO_DS_3 Schedule: Trigger job upon completion of SPO_DS_2
SPO_DS_4 Schedule: Trigger job upon completion of SPO_DS_3
SPO_DS_5 Schedule: Trigger job upon completion of SPO_DS_4

In the above case Job 2 (SPO_DS_2) is triggered when Job 1 (SPO_DS_1) completes and so on.

For practical purposes one datasource crawling may be too slow. To increase the number of concurrent jobs use '**MaxConcurrentSharePointJobs**'.

Example 'MaxConcurrentSharePointJobs=3':
SPO_DS_1 Schedule: Every day at 06:00:00
SPO_DS_2 Schedule: Trigger job upon completion of SPO_DS_1
SPO_DS_3 Schedule: Every day at 06:00:00
SPO_DS_4 Schedule: Trigger job upon completion of SPO_DS_3
SPO_DS_5 Schedule: Every day at 06:00:00
SPO_DS_6 Schedule: Trigger job upon completion of SPO_DS_5

Crawl using the SharePoint Optimized V2 connector

How-to

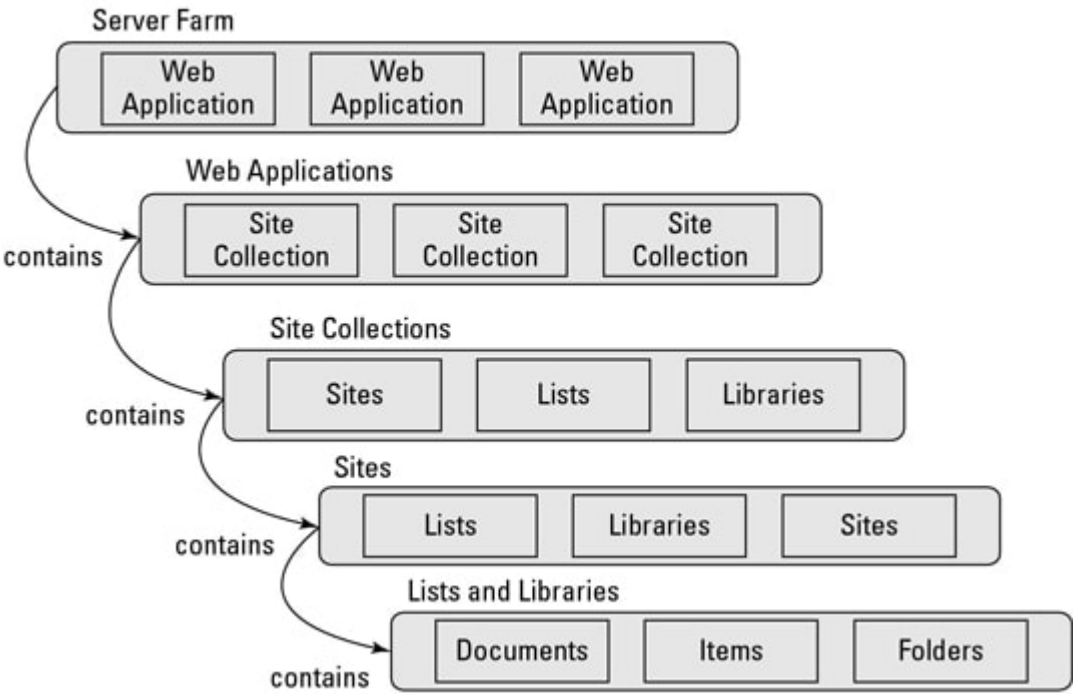
> Compatible versions: 4.2, 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion Connectors*, *Fusion Server*, *Fusion*

Decide what you need to crawl

First understand the different types of objects in SharePoint.

link:Get a quick overview of the SharePoint object model

You have to choose what you want to crawl.



Crawl all site collections

> **Note**
>
> This option is only available if you crawl with administrative permissions as listing all site collections in the web application requires administrative access.

In the Fusion UI, specify the **Web Application URL** to your SharePoint web application and check **Fetch all site collections**.

This will now list all site collections and crawl them one at a time.

Crawl a subset of site collections

Do not check the **Fetch all site collections** checkbox and list each site collection you wish to crawl.

Specify each site collection relative to the **Web Application URL**.

Crawl a subset of sites, lists, or list items (URLs)

You can also limit your crawl to only crawl specific sites, lists, folders, files, or list items.

You will specify a single site collection in the Site Collection List as in the previous example.

Then you can **Restrict to specific SharePoint items**. Copy and paste the URL of the resource from the browser into this list of items.

When crawling, the connector will convert the URLs specified into inclusive regular expressions.

Decide how you would like to authenticate

With SharePoint, there are multiple options when it comes to authenticating.

On-premise - NTLM

Specify using the **NTLM Authentication Settings** in the Fusion UI.

SharePoint Online authentication options

When using the SharePoint Online connector, several options exist for authenticating.

SHAREPOINT SITE COLLECTION ADMINISTRATOR

Caution

Using a SharePoint service account is not recommended as the account type confers several configurable permissions which are too powerful and broad for the scope of a crawl.

You can create and promote a user to Site Collection Administrator for a specific site. Promoting a user to Site Collection Administrator grants all necessary permissions for a crawl, without granting broader administrative control over the entirety of the SharePoint environment. Administrators can be monitored and have their access revoked through the SharePoint Admin Center, mitigating potential security risks.

For more information on SharePoint site administration, see the Sharepoint documentation .

APP-ONLY AUTHENTICATION - AZURE AD WITH PRIVATE KEY

This method of authentication is the most commonly used. It requires giving the application key **Full Control** permission, otherwise you get security authorization errors while crawling.

Note

The `username` field is mandatory. If using app-only authentication, you can use a fake value in this field, for example `xxx@lucidworks.com`.

Enter in the UI:

- Client ID
- PFX key (in base 64 format)
- PFX key password
- Tenant

APP-ONLY AUTHENTICATION - AZURE AD WITH OAUTH

This method of authentication is also commonly used. It requires giving the application key **Full Control** permission, otherwise you get security authorization errors while crawling.

Note

The `username` field is mandatory. If using app-only authentication, you can use a fake value in this field, for example `xxx@lucidworks.com`.

Enter in the UI:

- Client ID
- Client Secret
- Tenant

Understanding incremental crawls

After you have performed your first successful full crawl (no blocking errors), all subsequent crawls are incremental crawls.

Incremental crawls use SharePoint's Changes API to do the following:

- For each site collection, use the change token (timestamp) to get all changes since the full crawl was started:
    - Additions
    - Updates
    - Deletions

If you are crawling with **Fetch all site collections** checked, the incremental crawl also detects when a site collection itself was deleted and removes it from your index.

> **Note**
>
> During incremental crawl, all fields with the *lw* prefix must remain in the content documents. Failure to do so results in incremental crawls not working as expected. If you are going to filter on fields, leave the *lw* fields as-is.

Force full crawl

To perform incremental crawls, the **Force Full Crawl** property must be disabled, which it is by default

If a crawl starts, and **Force Full Crawl** is enabled, incremental crawl behavior is skipped and instead a full crawl is performed:

- MapDb tables are recreated and previously persisted data is deleted.
- The plugin performs the requests of all SharePoint objects just like the first crawl.
- All sharepoint-items are reindexed.
- Given incremental crawl is skipped, changes with `delete` are not detected and documents deleted from the SharePoint server are not removed from the index, resulting in orphaned documents.

Be aware for SharePoint Optimized V2 connector <v2.0.0:

- After a first crawl, MapDb information is persisted in PVC if configured, otherwise it is saved in the `/tmp` path.
- If PVC is configured in the environment and you need to perform full-crawls from scratch:
    - Clear the datasource as it also removes MapDb.

- If PVC is not configured and it is needed to perform full crawls from scratch:
    - Clearing the datasource does not remove the MapDb saved in the `/tmp` path from a previous crawl.
    - Enable the **Force Full Crawl** property, as this forces MapDb files to be removed.

Since SharePoint Optimized V2 connector v2.0.0, to perform full-crawls from scratch: * Clear the datasource and enable the full-crawl property. * It does not matter if PVC is configured or not, as MapDb information is not persisted by the plugin.

Crawl or incremental crawl and MapDb

At crawl time, MapDb is used to persist:

- SharePoint objects metadata (sites, lists, items, and attachments). The plugin reads the metadata saved and builds the documents to send to the connector server.
- siteCollections checkpoints.
- Permissions inheritance (list of all SharePoint objects and permissions associated).
- Site-collection taxonomy terms (if property is enabled).

- List of siteCollections.
- Folders fetched.

Note

Since SharePoint Optimized V2 connector v1.5.0, SharePoint objects metadata is not persisted as the metadata is removed when the crawl finishes.

When the incremental crawl is performed, SharePoint Optimized V2 connector <v2.0.0 uses the persisted information saved in previous crawls:

- siteCollections checkpoints to know where to start a subsequent crawl (incremental crawl).
- Information related to permissions inheritance to know which permissions and documents to update.
- Information related to the list of siteCollections to perform siteCollection deletion.
- Information related to the list of folders to update folders/nested folders.
- Taxonomy terms to perform incremental crawl of this feature.

When mapDb is persisted between crawls, this information needs to be deleted to perform a full recrawl.

Note

Since SharePoint Optimized V2 connector v2.0.0, the plugin does not need to persist the information mentioned above: site collection checkpoints, list of site collections, folders, or permissions. Only taxonomy terms requires mapDb usage and is needed only at crawl time.

Create a Custom Predictive Merchandiser Template Type

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion*

This article teaches you how to create custom *Templates* for Predictive Merchandiser. Once created, see *Use Predictive Merchandiser Templates* to learn how to use your custom template.

To begin, access Templates in the *Rules Editor* by navigating to **Relevance** > **Rules** > **Templates**.

Create a new template type

1. From the Template Manager screen, click **New Template**.
2. Add a user-friendly **Display Name** for your template.
3. In the **type** field, create a new template type by entering the name and clicking **Add**.
4. Click **Save**. The custom template type and template display in the list of templates.
5. Configure the template as desired. See *Use Predictive Merchandiser Templates* for more information.
6. Click **Publish**. Review the changes, and click **Confirm and Publish** to publish your custom template type and template. The template is now available in the **Merchandiser** screen. The template activates based on your configurations.

Create a Custom Predictive Merchandiser Template Type

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

This article teaches you how to create custom *Templates* for Predictive Merchandiser. Once created, see *Use Predictive Merchandiser Templates* to learn how to use your custom template.

To begin, access Templates in the *Rules Editor* by navigating to **Relevance** > **Rules** > **Templates**.

Create a new template type

1. From the Template Manager screen, click **New Template**.
2. Add a user-friendly **Display Name** for your template.
3. In the **type** field, create a new template type by entering the name and clicking **Add**.
4. Click **Save**. The custom template type and template display in the list of templates.
5. Configure the template as desired. See *Use Predictive Merchandiser Templates* for more information.
6. Click **Publish**. Review the changes, and click **Confirm and Publish** to publish your custom template type and template. The template is now available in the **Merchandiser** screen. The template activates based on your configurations.

Create a Java Keystore for Use with Fusion

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion*

Enterprise customers may want to use a signed certificate from an Enterprise Certificate Authority (CA), such as GoDaddy or Let's Encrypt for their Fusion 5 deployment.

The Fusion 5 default setup will create a self-signed Java Keystore (JKS) file that is used as the JSON Web Token (JWT) signer.

The following instructions detail how to create the `api-gateway.jks` file using a CA signed certificate.

Prerequisites

- Oracle Java JDK 1.8

New installations

1. Create a JKS

Run the following command, replacing the `dname` (Distinguished Name) field with the name that matches your customer environment, and the `keypass` and `storepass` fields with a secure password.

This will create the `api-gateway.jks` :

```
keytool -genkeypair -keyalg rsa -keysize 2048 -validity 3650 -dname "CN=sample, OU=sample, O=sample, L=sample, S=sample, C=sample" -storetype jks -keystore api-gateway.jks -keypass password -storepass password -alias jwt-signer
```

Caution

If you change the `alias` or keystore/truststore passwords, you will need to update the api-gateway deployment YAML with the new values.

2. Create a Certificate Signing Request (CSR)

Run the following command to create a new `fusion.csr` , which will be sent to the CA:

```
keytool -keystore api-gateway.jks -certreq -alias jwt-signer -file fusion.csr
```

3. Send the CSR to your CA to get a signed certificate

The CA will return one of two possible responses:

- A **PKCS12** certificate file, such as `CAreply.pkcs` .
  This is a full keystore containing all the keys you need, including the root certificate and the signed chain certificate.
- Several **PEM** files, such as `0000_cert.pem` , `0000_chain.pem` , and `0001_chain.pem` .
  These files contain the root certificate, the chain certificate(s), and the signed chain certificate.

4. Import the signed certificate into the `api-gateway.jks`

- If you received a **PKCS** file, import it using the following keytool command:

```
keytool -importcert -alias jwt-signer -file <CAreply.pkcs> -keystore api-gateway.jks -trustcacerts
```

- If you receive **PEM** files, run the following commands to import the keys:
  - `keytool -importcert -alias root -file 0000_cert.pem -keystore api-gateway.jks -trustcacerts`
  - `keytool -importcert -alias intermediate -file 0000_chain.pem -keystore api-gateway.jks -trustcacerts`
  - `keytool -importcert -alias jwt-signer -file 0001_chain.pem -keystore api-gateway.jks -trustcacerts`

5. Upload the `api-gateway.jks` as a Kubernetes generic secret

The api-gateway service will pull down the **`api-gateway.jks`** file during initialization as a Kubernetes secret:

1. Delete the **`<namespace>-api-gateway.jks`** secret if it already exists.
2. Upload the new secret with the following command:

```
k create secret generic <namespace>-api-gateway-jks --from-file api-gateway.jks
```

6. Install Fusion 5 using Helm

See *Deploy Fusion 5 on Other Kubernetes Platforms* for more information.

Existing installation

1. Create a JKS with your CA certificate or the java keytool

If you are using the Java keytool, you can run the same command used in the *Create a JKS* section:

```
keytool -genkeypair -keyalg rsa -keysize 2048 -validity 3650 -dname "CN=sample,
OU=sample, O=sample, L=sample, S=sample, C=sample" -storetype jks -keystore api-
gateway.jks -keypass myPassword -storepass myPassword -alias jwt-signer
```

2. Scale down Fusion deployment to 0

This step will allow Fusion services to find the latest JKS once it is scaled up at the final step.

See *Deploy Fusion at Scale* for more information on the process.

3. Delete the existing Kubernetes JKS secret for Fusion

As in the *Upload the* `api-gateway.jks` *as a Kubernetes generic secret* section, delete the existing **`<namespace>-api-gateway.jks`** secret.

4. Create a Kubernetes secret with the JKS using the name of the deployment

Upload a new secret using the api-gateway.jks file with the following command:

```
k create secret generic <deployment>-api-gateway-jks --from-file api-gateway.jks
```

5. Scale up Fusion deployment

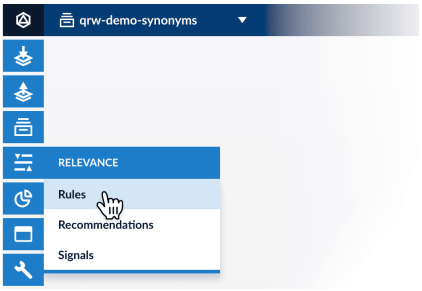
Fusion should accept the JKS once this step is complete.

Create a new Business Rule

How-to

> Compatible versions: 4.2
>
> Works with: *Fusion AI*

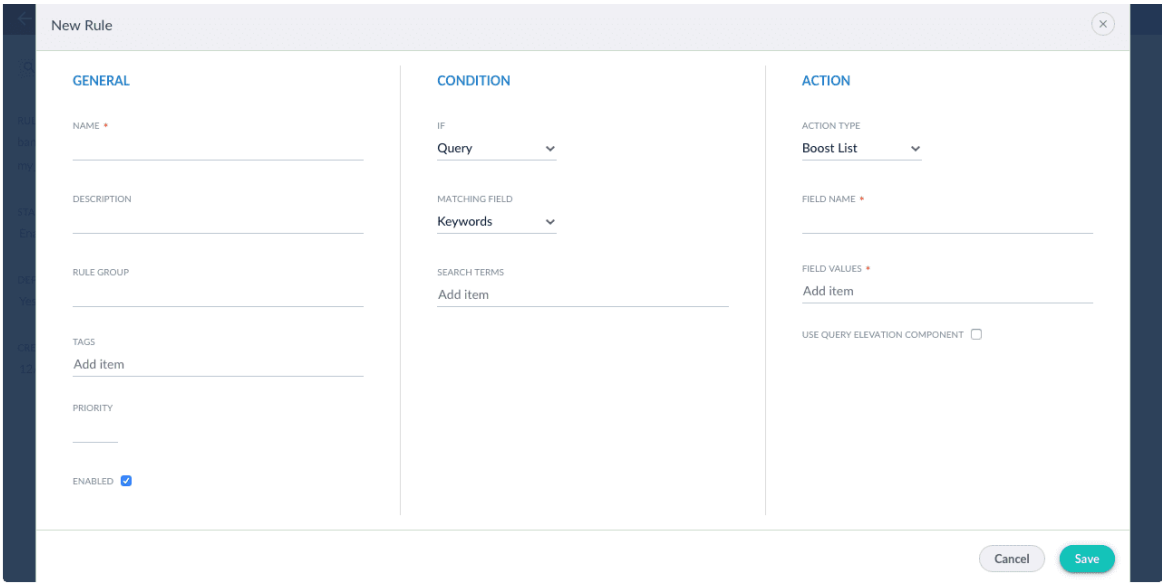
1. Navigate to **Relevance** > **Query Rewriting**.



2. Under **Business Rules**, click **View**.

3. Click the ⊕ icon.

   The **New Rule** window appears:



General	Condition	Action
In the **General** column, only the **Name** field is required. Other fields are optional: • **Description** is an arbitrary string you can use to describe this rule. • **Rule Group** can be a user-defined group that you use to organize your rules. • **Tags** are another way to organize your rules. Tags appear as facets in the Business Rules interface, so you can filter the set of visible rules by tag. • **Priority** can be used to determine which rule should apply first if multiple rules are activated for the same request. Priority is an integer value, ranging from 1 to infinity, with the value 1 given highest priority. If multiple rules have the same value, those rules are applied in a random order. • **Enabled** means that this rule is applied (but not necessarily published). Disabling a rule helps ensure that it is not accidentally published.	Conditions are triggers that activate the rule when they match the current date and time, query, or field values. See *Rule conditions* for more information.	A rule can take different types of actions when the specified conditions are met. See *Action types* for more information. You can also create *custom actions*.

4. Click **Save**.

Create a New Job

How-to

> Compatible versions: 4.2, 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion Server*, *Fusion*

1. In the Fusion workspace, navigate to **Collections > Jobs**.
2. Click **Add** and select the job type. The New Job Configuration panel appears. This panel is different for each job type.
3. Configure the new job as needed.
4. Click **Save**.

Create a New Job

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

1. In the Fusion workspace, navigate to **Collections > Jobs**.
2. Click **Add** and select the job type. The New Job Configuration panel appears. This panel is different for each job type.
3. Configure the new job as needed.
4. Click **Save**.

Create a New Web App (4.2)

How-to

> Compatible versions: 4.2
>
> Works with: *App Studio*

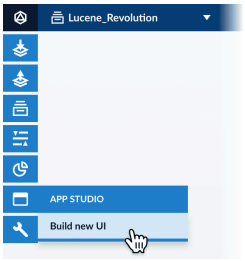
The current version of App Studio creates a single-page search application that you can further customize through the built-in *code editor*.

Prerequisites

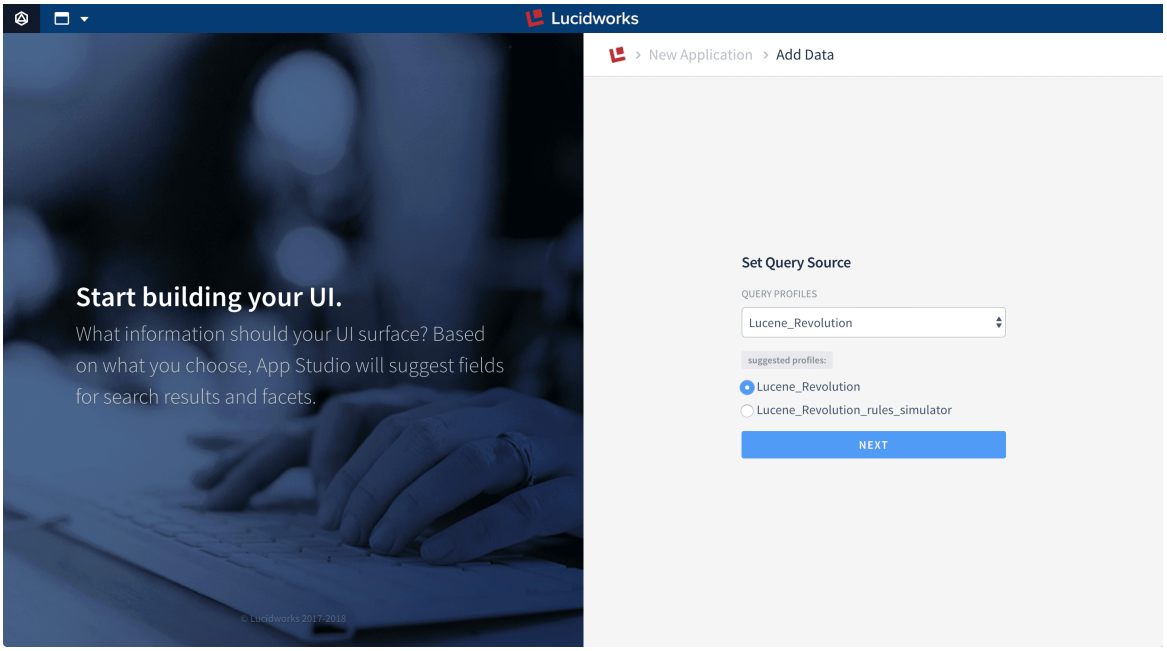
The instructions in this topic assume that you have a running instance of *Fusion Server*, with one or more *collections* of indexed data and at least one *query profile*.

How to create a new Web app

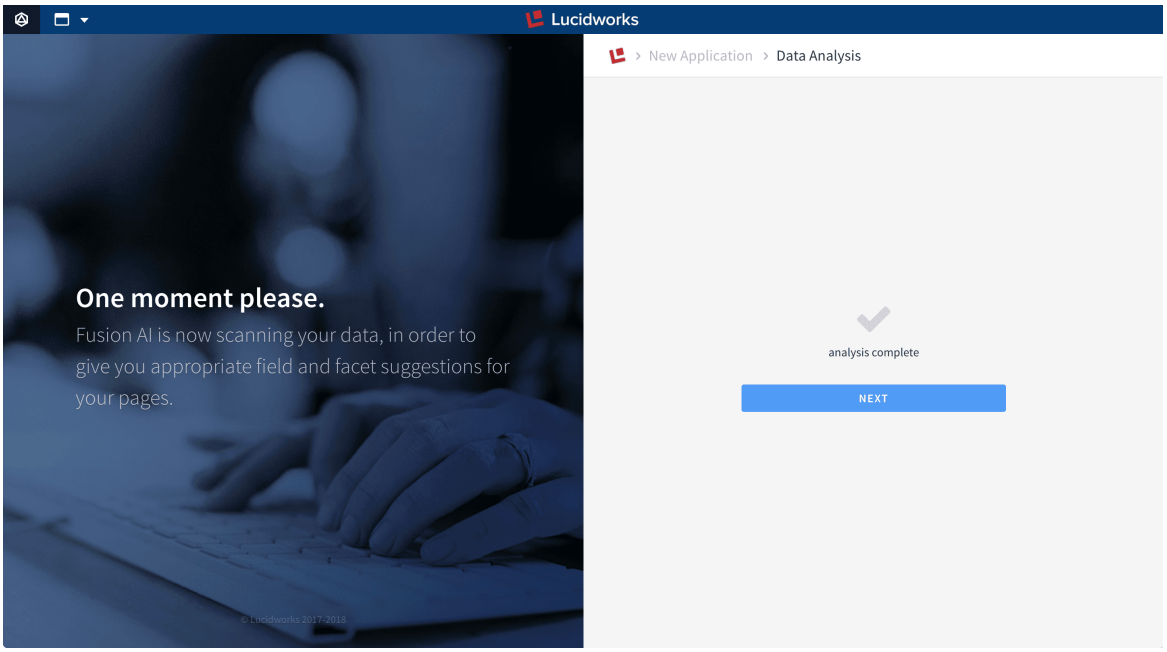
1. In the Fusion workspace, navigate to **App Studio** > **Build New UI**.



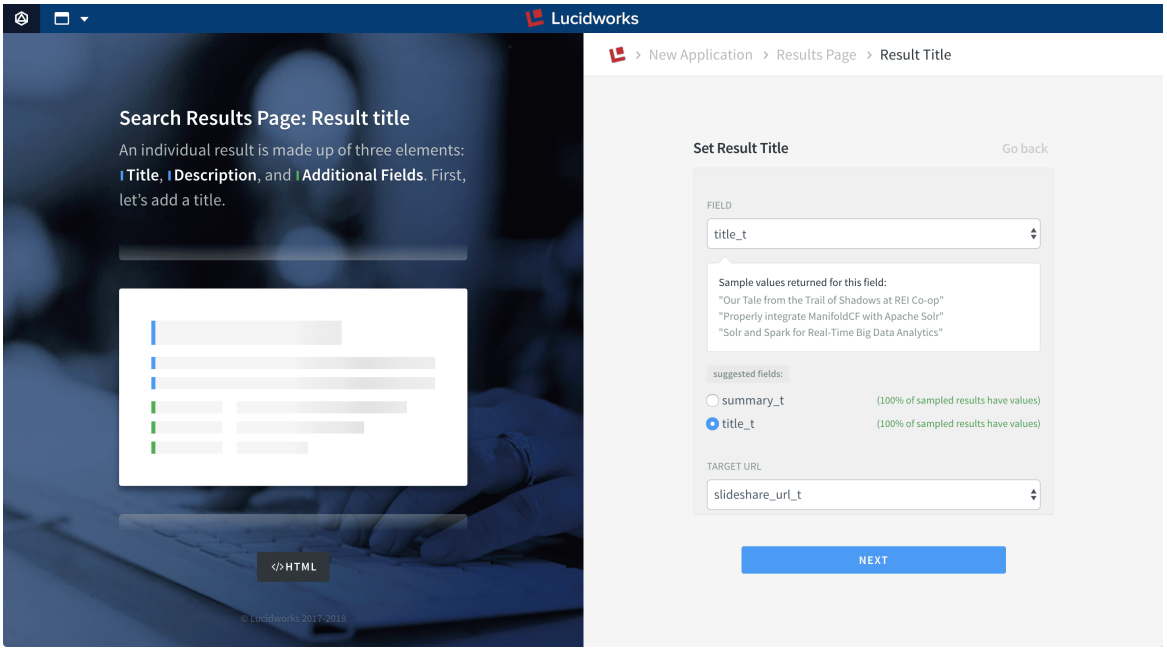
2. On the **Set Query Source** page, select the query profile to use for your search UI:



3. Click **Next**.
   App Studio connects to the specified collection and query profile to analyze your searchable data:

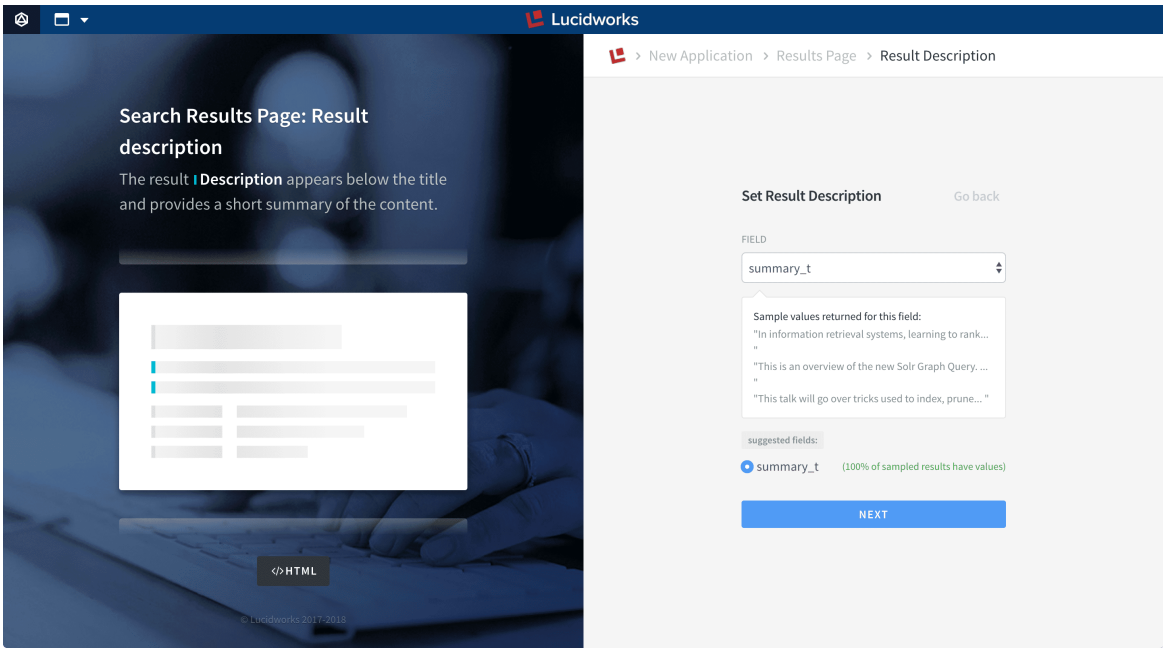


4. Click **Next**.
5. On the **Set Result Title** page, select the field to use as the title of each search result:

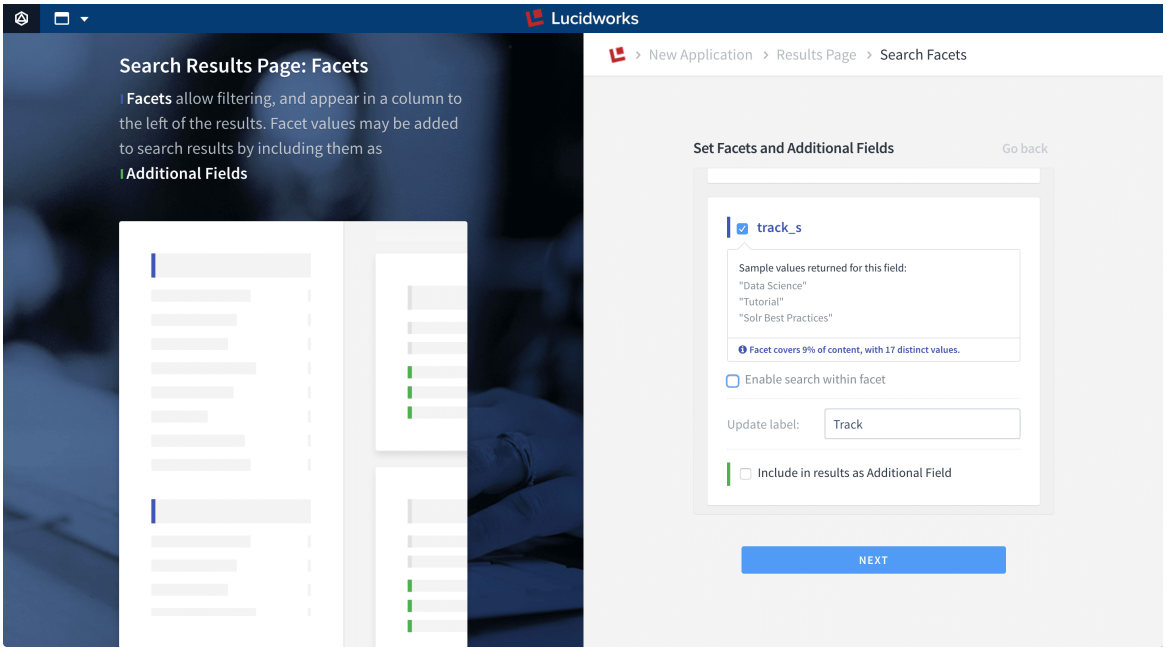


Optionally, you can select a **Target URL** field, which contains the URL to use as the link in each search result.

6. Click **Next**.
7. On the **Set Result Description** page, select the field to use as the description of each search result:



8. Click **Next**.
9. On the **Set Facets and Additional Fields** page, select one or more multi-valued fields to use as facets:

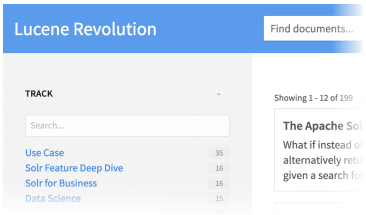


Note

In some cases, facet fields are not selectable in the wizard. In this case, you can define facets later using markup.

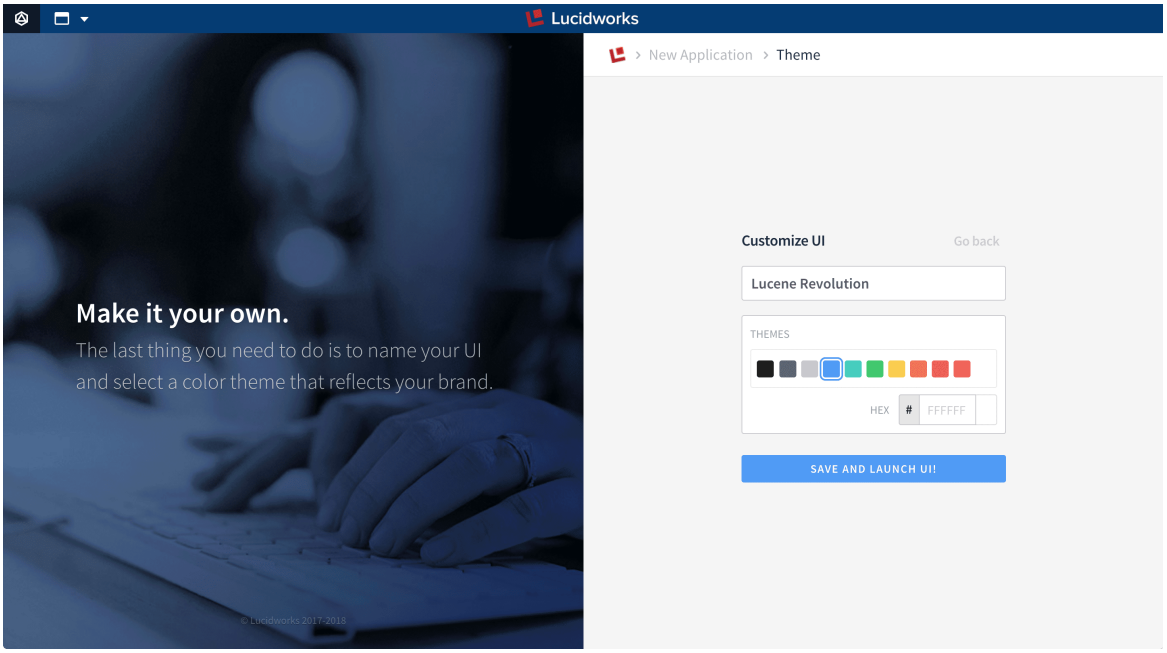
Optionally:

- Select **Enable search within facet** to add a search box for searching the facet field names to the Facets pane.

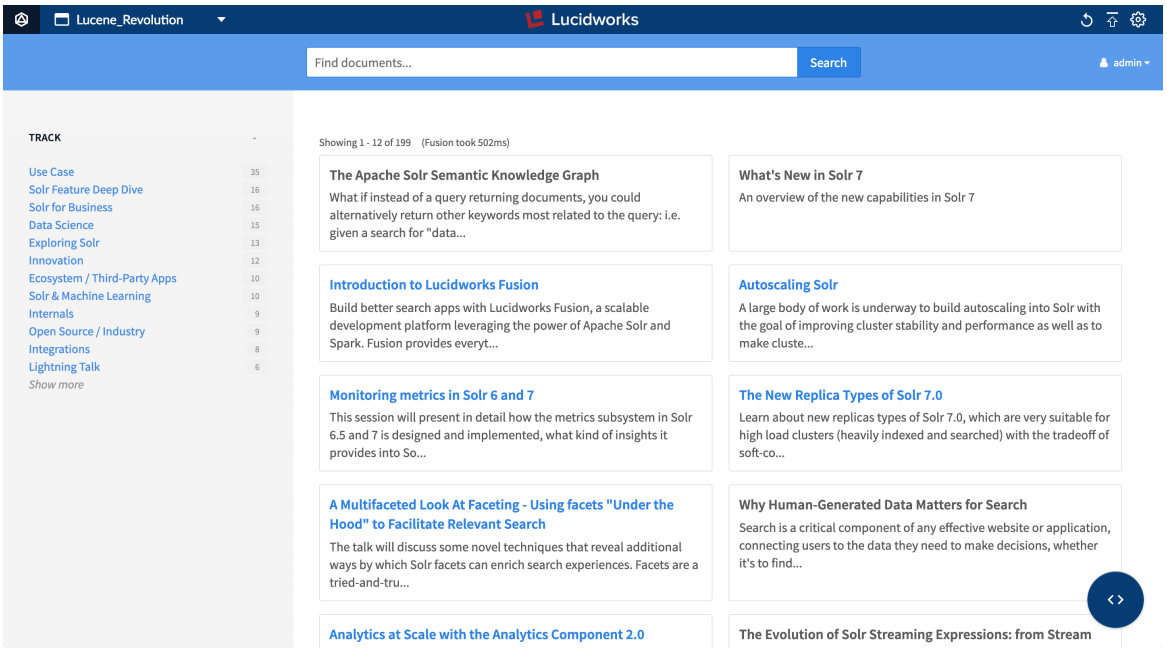


- Enter a user-friendly label for each facet field, such as "Genre" instead of "genres_ss".
- Select **Include in results as Additional Field** to display the values of the facet fields within each search result.

10. Click **Next**.
11. On the **Customize UI** page, enter a title and select a color theme:



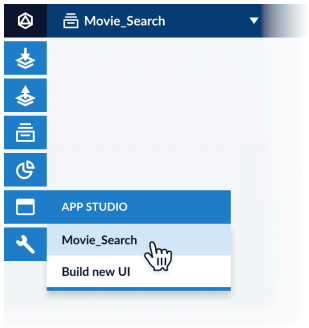
12. Click **Save and Launch UI**.
    A preview of your search interface and data appear:



Note

It can take about a minute for the preview to appear. If the preview does not appear, refresh the browser tab.

After you have created a search UI, you can open it from the Fusion workspace by navigating to the **App Studio** menu and selecting the search UI:



What is next

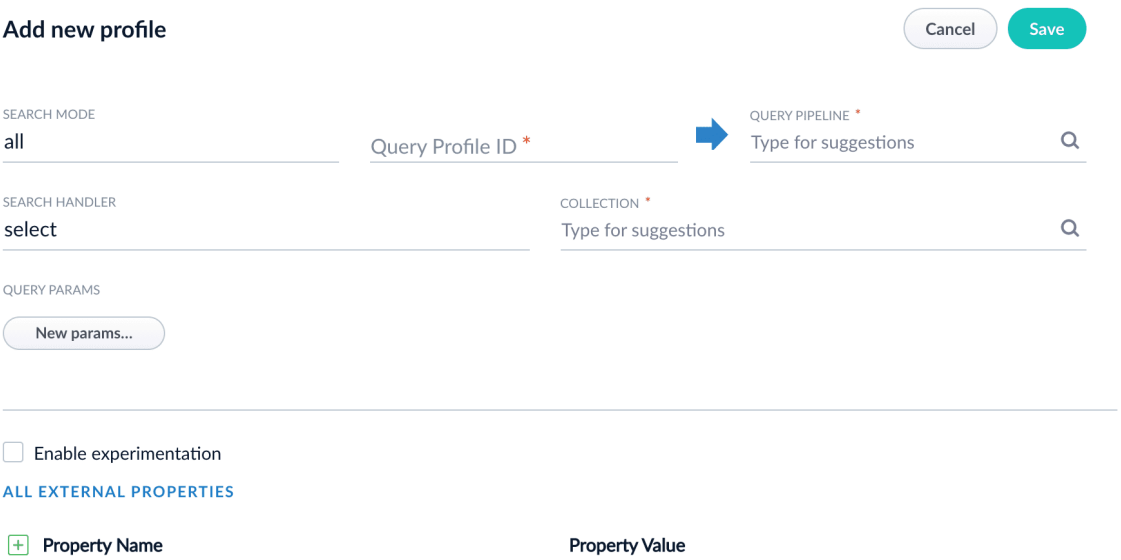
- See the *Develop an App* to learn about App Studio's markup templating language and how app pages are constructed.
- *Deployment* explains how to deploy your customized app in a production environment.

Create a query profile

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

1. In the Fusion workspace, navigate to **Querying** > **Query Profiles**.
2. Click **New**.
   The **Add Query Profile** panel appears:



3. Enter a search mode to use with this profile.
   - `dsl` (*Domain Specific Language*) uses expressive search queries and responses via a structured, modern JSON format.
   - `legacy` primarily uses Solr parameters. See the *Solr Query language cheat sheet*.
   - `all` uses both DSL and legacy search modes. This is the default value.

   The default value of `all` works well for most use cases.

4. Enter a **Query Profile ID**.
5. Select a query pipeline to associate with this profile.
6. Enter a Solr search handler to use with this profile.
   The default of `select` works well for most use cases.
7. Select a collection to associate with this profile.
8. Optionally, click **New params...** to enter Solr request parameters to add to the request URL when submitting queries using this profile.
9. Optionally, select **Enable experimentation** to configure this profile for use with *experiments*.
   a. Select the percentage of experiment traffic to direct to this profile.
   b. Select the experiment to associate with this profile, or select **Add Experiment** to configure a new one.
      See *Set up an experiment* for details.
10. Click **Save**.
    The query profile window displays the request URL for sending queries to this profile.

LucidAcademy

Lucidworks offers free training to help you get started.

The **Course** for **Index and Query Profiles** focuses on how index and query profiles operate, and how to configure and manage them:

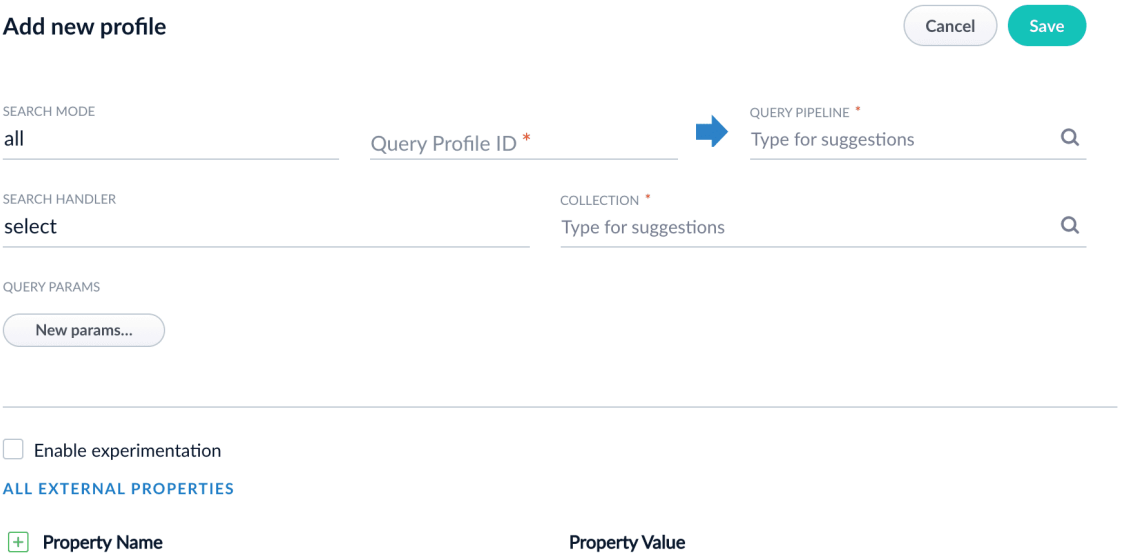
Visit the LucidAcademy to see the full training catalog.

Create a Query Profile

How-to

> Compatible versions: 4.2, 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion Server*, *Fusion*

1. In the Fusion workspace, navigate to **Querying** > **Query Profiles**.
2. Click **New**.
   The **Add Query Profile** panel appears:



3. Enter a search mode to use with this profile.
   - `dsl` (*Domain Specific Language*) uses expressive search queries and responses via a structured, modern JSON format.
   - `legacy` primarily uses Solr parameters. See the *Solr Query Language cheat sheet*.
   - `all` uses both DSL and legacy search modes. This is the default value.

   The default value of `all` works well for most use cases.

   Note

   The **Search Mode** field is available for Fusion 5.4.x and later.

4. Enter a **Query Profile ID**.
5. Select a query pipeline to associate with this profile.
6. Enter a Solr search handler to use with this profile.
   The default of `select` works well for most use cases.
7. Select a collection to associate with this profile.
8. Optionally, click **New params...** to enter Solr request parameters to add to the request URL when submitting queries using this profile.
9. Optionally, select **Enable experimentation** to configure this profile for use with *experiments*.
   a. Select the percentage of experiment traffic to direct to this profile.
   b. Select the experiment to associate with this profile, or select **Add Experiment** to configure a new one.
      See *Set up an experiment* for details.
10. Click **Save**.
    The query profile window displays the request URL for sending queries to this profile.

LucidAcademy

Lucidworks offers free training to help you get started.

The **Course** for **Index and Query Profiles** focuses on how index and query profiles operate, and how to configure and manage them:

Visit the LucidAcademy to see the full training catalog.

Create Aggregation Jobs

How-to

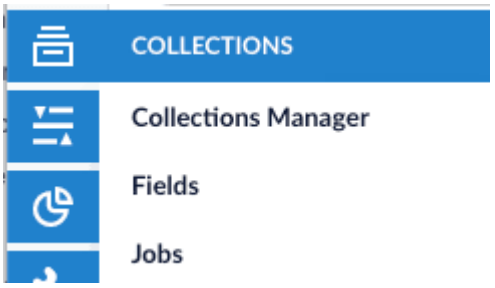
> Compatible versions: 4.2
>
> Works with: *Fusion Server*

Aggregations are created automatically whenever you enable signals or recommendations. This topic explains how to create or modify aggregations individually. You can do this using the Fusion UI or the Jobs API.

Creating an aggregation job

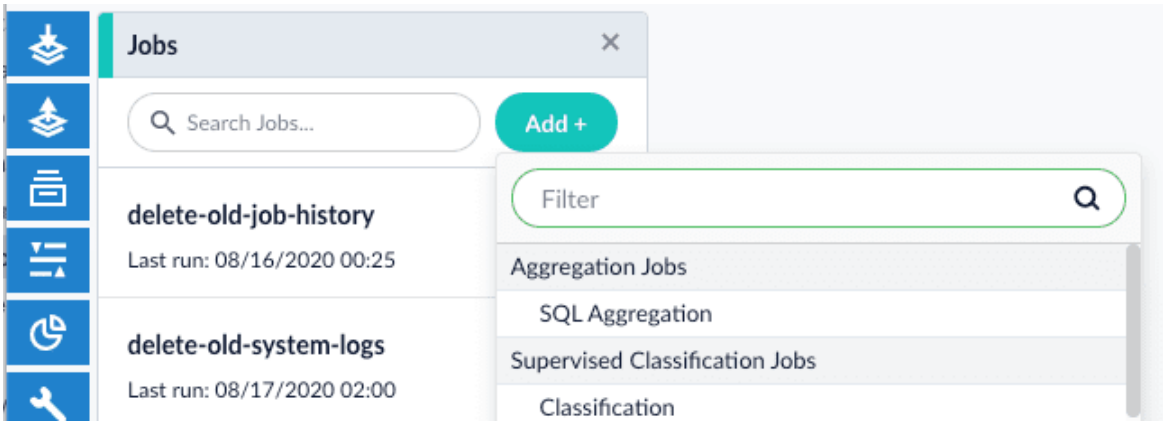
An aggregation is a type of job. Aggregation jobs can be created or modified under **Collections** or using **Search** > **Jobs** in the Fusion UI.

1. Navigate to **Jobs**.

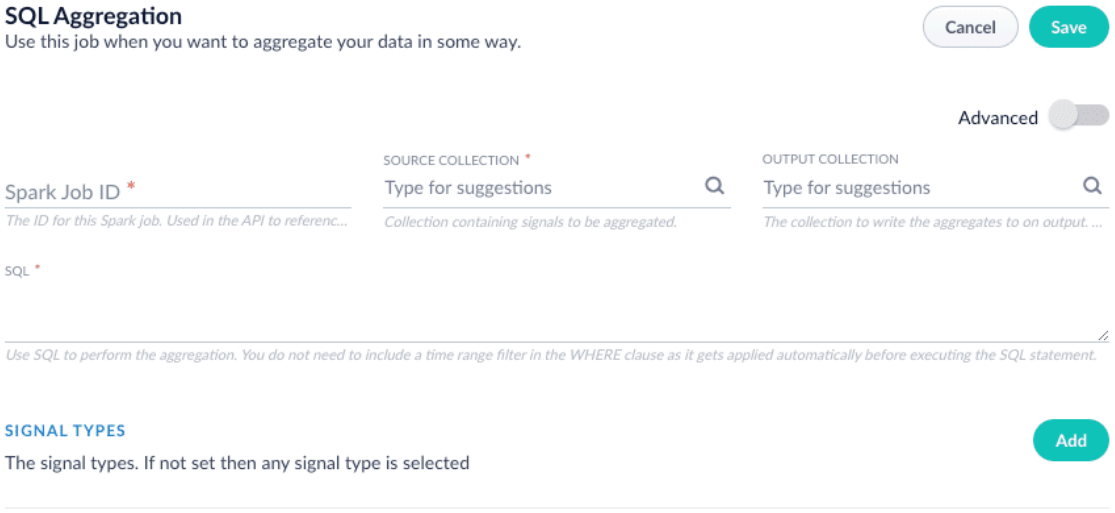


2. Click **Add**.

> **Note**
>
> Fusion 5.x.x supports only SQL aggregations. To avoid upgrade issues, Lucidworks recommends you use SQL aggregations in Fusion 4.x.x even though that release supports both regular (functional) aggregations and SQL aggregations.



3. Select **SQL Aggregation**.
   The New Job Configuration panel appears.



4. Enter an arbitrary Spark job ID.
5. Enter the name of the signals collection to be aggregated.

> **Note**
>
> Be sure to specify the signals collection (usually `<primarycollectionname>_signals`), not the primary (`<primarycollectionname>`) collection.

6. Configure the aggregation parameters as needed.
   a. For SQL aggregation jobs, see the *parameter list* here for more information.
7. Click **Save**.
   The new aggregation job appears in the jobs list. Now you can *run it* or *schedule it*.

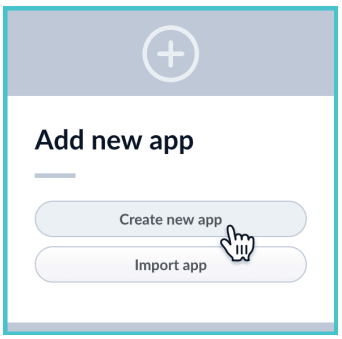
Create an App

How-to

> Compatible versions: 4.2
>
> Works with: *App Studio*

To create an app in App Studio:

1. Navigate to the launcher and click **Create new app**.



2. Enter an app name.
   Optionally, you can also enter a description and select the app tile color.
   If you want to choose the name of the first collection in the app, then deselect **Create new app default collection**. When you do this and click **Create app** (in the next step), Fusion Server prompts you to enter a name for the first collection. Otherwise, Fusion Server creates a default collection whose name is the same as the app name.
3. Click **Create App**.
   The Fusion workspace appears, and you are ready to work with your app's first collection.

Create an index profile

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

1. Sign in to Managed Fusion and click your application.
2. Click **Indexing > Index Profiles > New**.
3. Click **New**.
4. In the Add Index Profile section, enter a value in Index Profile ID.
5. Select an index pipeline to associate with this profile.
6. Select a parser to associate with this profile.
7. Select a collection to associate with this profile.
8. Click **New params...** to enter Solr request parameters to add to the request URL when indexing documents using this profile. This section is optional.
9. Click **Save**.

The index profile window displays the request URL for sending documents to this profile.

LucidAcademy

Lucidworks offers free training to help you get started.

The **Course** for **Index and Query Profiles** focuses on how index and query profiles operate, and how to configure and manage them:

Visit the LucidAcademy to see the full training catalog.

Create an Index Profile

How-to

> Compatible versions: 4.2, 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion Server*, *Fusion*

1. In the Fusion workspace, navigate to **Indexing** > **Index Profiles**.
2. Click **New**.
   The **Add Index Profile** panel appears.
3. Enter an **Index Profile ID**.
4. Select an index pipeline to associate with this profile.
5. Select a parser to associate with this profile.
6. Select a collection to associate with this profile.
7. Optionally, click **New params...** to enter Solr request parameters to add to the request URL when indexing documents using this profile.
8. Click **Save**.
   The index profile window displays the request URL for sending documents to this profile.

LucidAcademy

Lucidworks offers free training to help you get started.

The **Course** for **Index and Query Profiles** focuses on how index and query profiles operate, and how to configure and manage them:

Visit the LucidAcademy to see the full training catalog.

Create and manage Fusion apps

How-to

> Compatible versions: 4.2, 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion Server*, *Fusion*

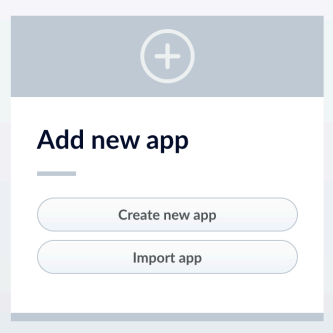
You can create, modify, and delete apps in the Fusion UI or with the *Apps API*.

Manage apps in the Fusion UI

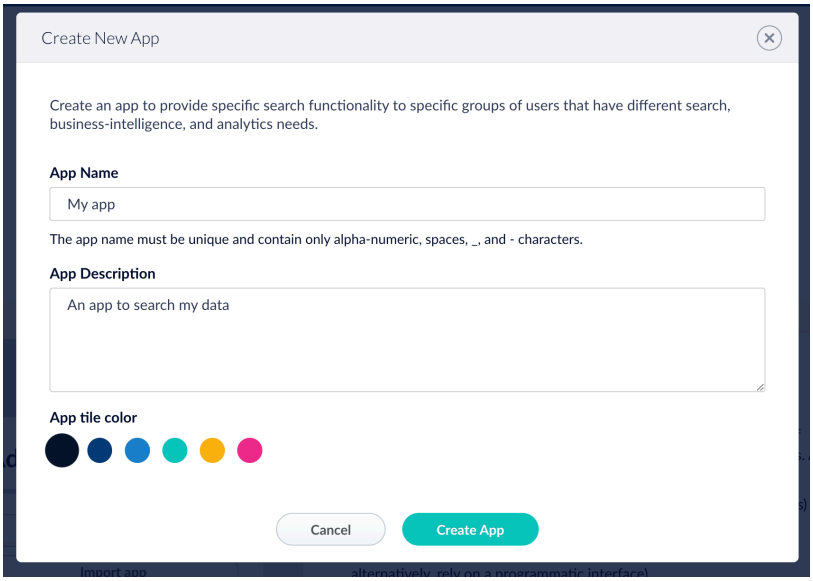
Create an app

Fusion prompts you to create an app the first time you log in. You can create additional apps at anytime from Fusion's home page. To import an app instead, see *Import a Fusion app*.

1. Log into Fusion and navigate to the home screen.
2. Click **Create new app**.



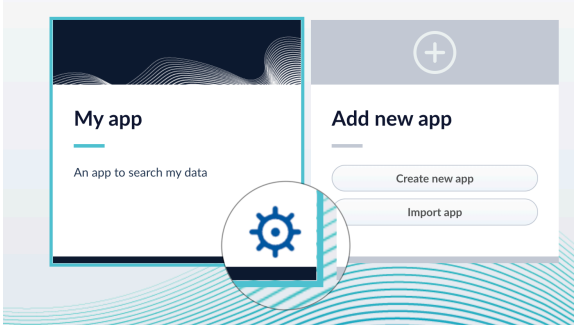
3. Enter a **Name**, **Description**, and select an **App tile color**.



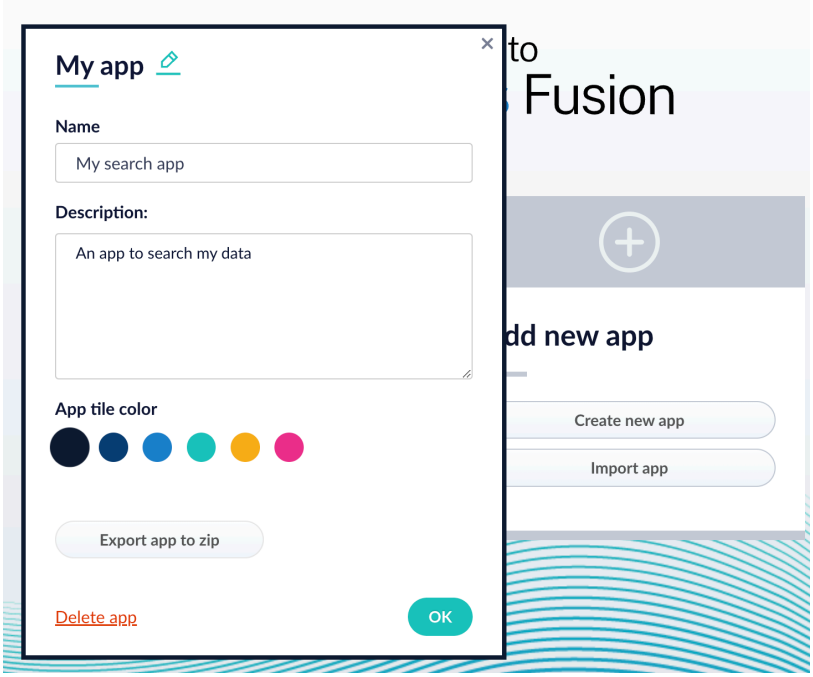
4. Click **Create App**.

Modify or delete an app

1. Log into Fusion and navigate to the home screen.
2. Click the **Configure** icon on the app you want to modify or delete.

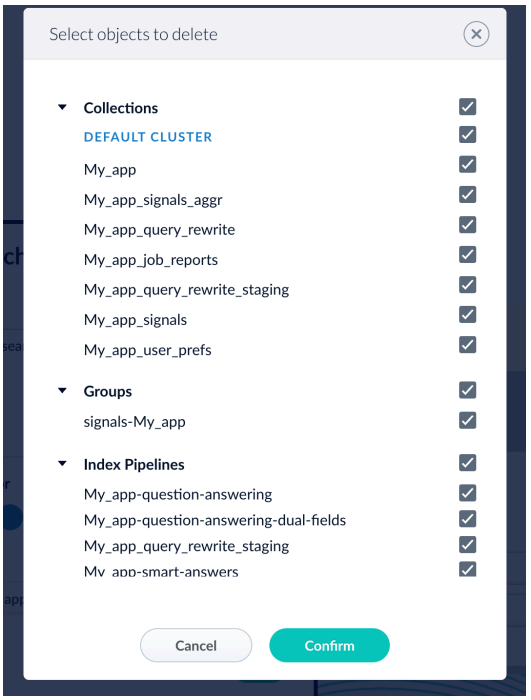


3. To modify the app, click the edit icon .



   a. Edit the **Name** or **Description**.
   b. Click **OK**.

4. To delete the app, click **Delete app**.
   a. A dialog box appears where you can choose which associated objects to delete or retain. Only objects not associated with other objects are displayed. All objects are selected by default.
   b. Click **Confirm**.



Manage apps with the Apps API

Create an app

```
curl -u USERNAME:PASSWORD -X POST -H 'Content-type: application/json'
-d '{"id": "MyNewApp", "name":"My new app", "description":"A really
really great new app"}' https://FUSION_HOST/api/apps
```

Delete an app

Run the following command, replacing `{id}` with the name of your app. This operation take about a minute to complete.

```
curl -u USERNAME:PASSWORD -X DELETE https://FUSION_HOST/api/apps/{id}
```

Create and manage Managed Fusion apps

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

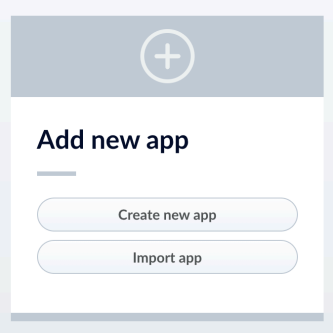
You can delete an app in the Managed Fusion UI or with the *Apps API*.

Manage apps in the Managed Fusion UI

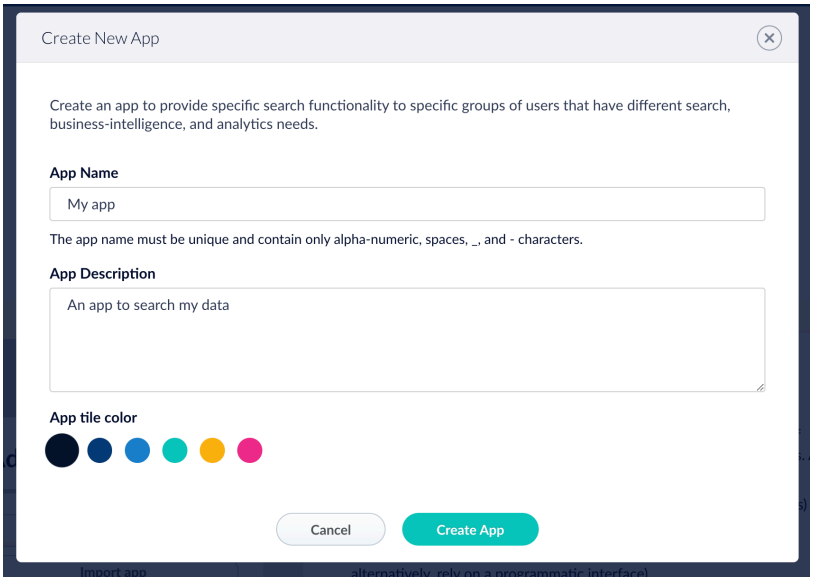
Create an app

Managed Fusion prompts you to create an app the first time you log in. You can create additional apps at anytime from Fusion's home page. To import an app instead, see *Import a Fusion app*.

1. Log into Fusion and navigate to the home screen.
2. Click **Create new app**.



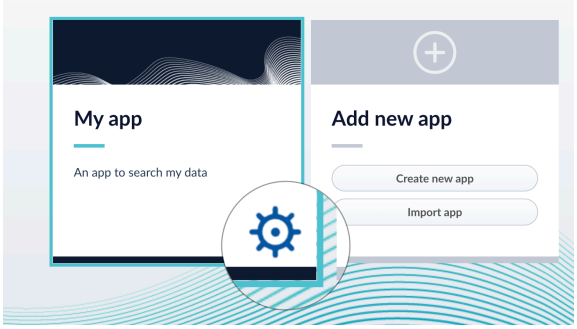
3. Enter a **Name**, **Description**, and select an **App tile color**.



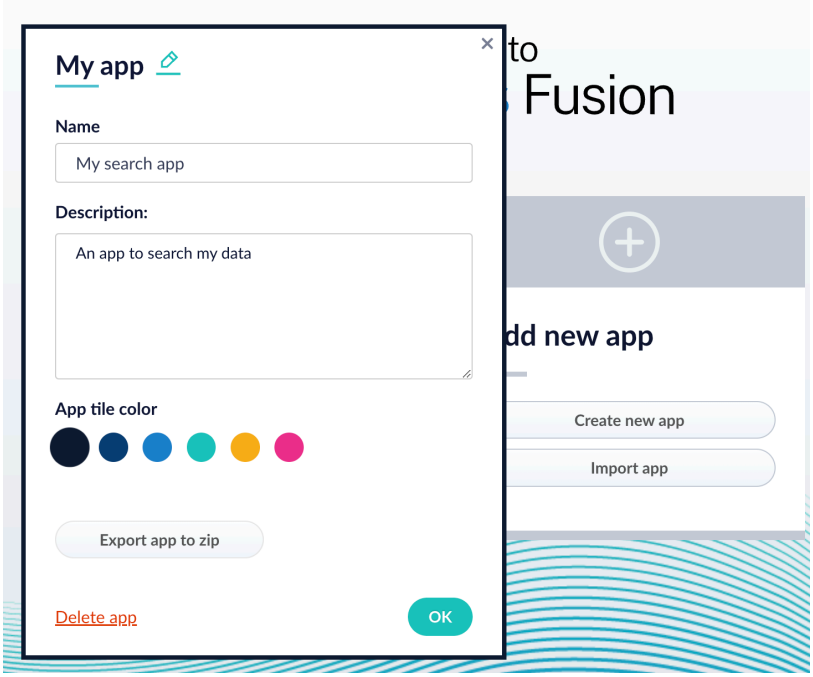
4. Click **Create App**.

Modify or delete an app

1. Log into Fusion and navigate to the home screen.
2. Click the **Configure** icon on the app you want to modify or delete.

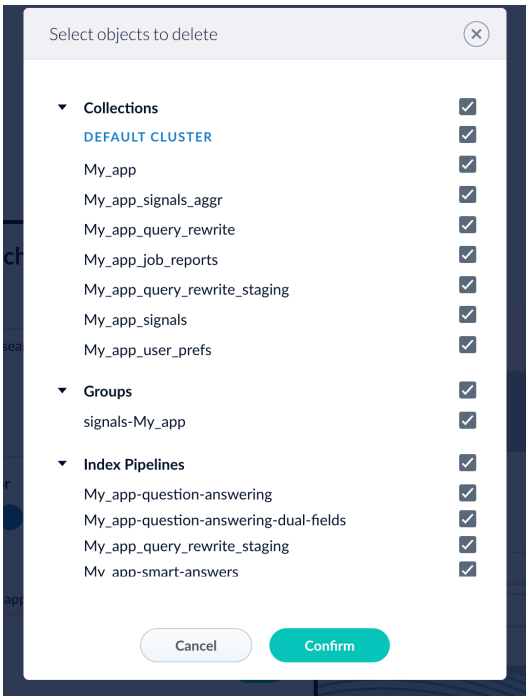


3. To modify the app, click the edit icon .



   a. Edit the **Name** or **Description**.
   b. Click **OK**.

4. To delete the app, click **Delete app**.
   a. A dialog box appears where you can choose which associated objects to delete or retain. Only objects not associated with other objects are displayed. All objects are selected by default.
   b. Click **Confirm**.



Manage apps with the Apps API

Create an app

```
curl -u USERNAME:PASSWORD -X POST -H 'Content-type: application/json'
-d '{"id": "MyNewApp", "name":"My new app", "description":"A really
really great new app"}'
https://EXAMPLE_COMPANY.b.lucidworks.cloud/api/apps
```

Delete an app

Run the following command, replacing `{id}` with the name of your app. This operation take about a minute to complete.

```
curl -u USERNAME:PASSWORD -X DELETE
https://https://EXAMPLE_COMPANY.b.lucidworks.cloud/api/apps/{id}
```

Create and Run a SQL Aggregation Job

How-to

> Compatible versions: 4.2, 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion Server*, *Fusion*

You can perform a SQL aggregation on a signals collection for a datasource (or on some other collection), through the Fusion UI or using the Fusion API.

Preliminaries

Before you can create and run a SQL aggregation job, you must create an app, create a collection (or use the default collection for the app), and add a datasource. Before you run a SQL aggregation job, you need signal data. Otherwise, there is nothing to aggregate.

Use the Fusion UI

You can use the Fusion UI to perform a SQL aggregation.

Set up a SQL aggregation job

1. In the Fusion application, navigate to **Collections > Jobs**.
2. Click **Add** and select **SQL Aggregation** from the dropdown list.
3. Specify an arbitrary **Spark Job ID**.
4. For the **Source Collection**, select the collection that contains the data to aggregate.

   Tip

   This is *not* the base collection. For example, to aggregate the signals in `experiment_signals`, you would select `experiment_signals`, not `experiment`.

5. Click the **SQL** field to enter aggregation statements. Click **Close** to close the dialog box.

   Note

   Select the **Advanced** option in the top right corner to enter detailed parameters. See *SQL Aggregation Jobs* for more information.

6. Enter applicable **Signal Types**.
7. Enter the **Data Format**.
8. Save the job.

Run a SQL aggregation job

1. With the app open, navigate to **Collections > Jobs**.
2. In the list of jobs, click the job you want to run, and then click **Run**.

   Note

   If you do not want to run the job manually, you can also *Schedule a Job*.

Use the Fusion API

You can use the Fusion API to perform a SQL aggregation. For example:

Configure a SQL aggregation job:

```
curl ":{api-
port}/api/spark/configurations/experiment_signals_aggregation" {
"type" : "spark", "id" : "experiment_click_signals_aggregation",
"definition" : { "id" : "experiment_click_signals_aggregation", "sql"
: "SELECT SUM(count_i) AS aggr_count_i, query AS query_s, doc_id AS
doc_id_s, time_decay(count_i, timestamp_tdt) AS weight_d FROM
default_signals WHERE type_s='click' GROUP BY query, doc_id",
"selectQuery" : "*:*", "outputPipeline" : "_system",
"rollupAggregator" : "SQL", "sourceRemove" : false, "sourceCatchup" :
true, "outputRollup" : true, "parameters" : [ { "key" :
"optimizeOutput", "value" : "4" } ] }, "inputCollection" :
"experiment_click_signals", "rows" : 10000 }
```

Run a SQL aggregation job:

```
curl -u USERNAME:PASSWORD -X POST -H "Content-Type: application/json"
http://localhost:{api-
port}/api/jobs/spark:experiment_click_signals_aggregation/actions -d
'{"action": "start"}'
```

Create and Run a SQL Aggregation Job

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

You can perform a SQL aggregation on a signals collection for a datasource (or on some other collection), through the Managed Fusion UI or using the Managed Fusion API.

Preliminaries

Before you can create and run a SQL aggregation job, you must create an app, create a collection (or use the default collection for the app), and add a datasource. Before you run a SQL aggregation job, you need signal data. Otherwise, there is nothing to aggregate.

Use the Managed Fusion UI

You can use the Managed Fusion UI to perform a SQL aggregation.

Set up a SQL aggregation job

1. In the Managed Fusion application, navigate to **Collections > Jobs**.
2. Click **Add** and select **SQL Aggregation** from the dropdown list.
3. Specify an arbitrary **Spark Job ID**.
4. For the **Source Collection**, select the collection that contains the data to aggregate.

   Tip

   This is *not* the base collection. For example, to aggregate the signals in `experiment_signals`, you would select `experiment_signals`, not `experiment`.

5. Click the **SQL** field to enter aggregation statements. Click **Close** to close the dialog box.

   Note

   Select the **Advanced** option in the top right corner to enter detailed parameters. See *SQL Aggregation Jobs* for more information.

6. Enter applicable **Signal Types**.
7. Enter the **Data Format**.
8. Save the job.

Run a SQL aggregation job

1. With the app open, navigate to **Collections > Jobs**.
2. In the list of jobs, click the job you want to run, and then click **Run**.

   Note

   If you do not want to run the job manually, you can also *Schedule a Job*.

Use the Managed Fusion API

You can use the Managed Fusion API to perform a SQL aggregation. For example:

Configure a SQL aggregation job:

```
curl
":API_PORT/api/spark/configurations/experiment_signals_aggregation" {
"type" : "spark", "id" : "experiment_click_signals_aggregation",
"definition" : { "id" : "experiment_click_signals_aggregation", "sql"
: "SELECT SUM(count_i) AS aggr_count_i, query AS query_s, doc_id AS
doc_id_s, time_decay(count_i, timestamp_tdt) AS weight_d FROM
default_signals WHERE type_s='click' GROUP BY query, doc_id",
"selectQuery" : "*:*", "outputPipeline" : "_system",
"rollupAggregator" : "SQL", "sourceRemove" : false, "sourceCatchup" :
true, "outputRollup" : true, "parameters" : [ { "key" :
"optimizeOutput", "value" : "4" } ] }, "inputCollection" :
"experiment_click_signals", "rows" : 10000 }
```

Run a SQL aggregation job:

```
curl -u USERNAME:PASSWORD -X POST -H "Content-Type: application/json"
https://EXAMPLE_COMPANY.b.lucidworks.cloud:API_PORT/api/jobs/spark:exp
eriment_click_signals_aggregation/actions -d '{"action": "start"}'
```

Create Custom Reports

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

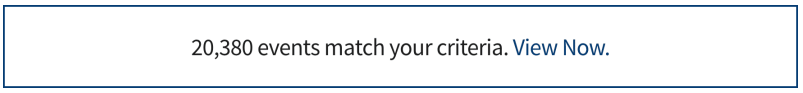
There are several ways to create a custom report:

On the Analytics page for App Insights

1. Apply one or more filters.
2. Click **+ New custom report**.
3. Define the report's parameters.
4. Click **Create and launch report**.

On the Dashboard page

1. Filter any dashboard.
2. Scroll to the bottom.
3. Click **<n> events match your criteria. View Now.**



This takes you to the *Events* page, where you can navigate to **Analytics** > **+New Custom Report**. Define additional parameters for the report, then click **Create and launch report**.

Create Custom Reports

How-to

> Compatible versions: 4.2, 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion Server*, *Fusion*

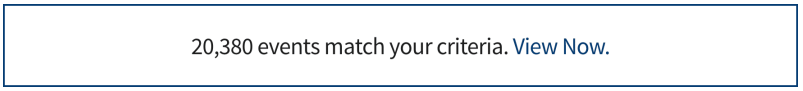
There are several ways to create a custom report:

On the Analytics page for App Insights

1. Apply one or more filters.
2. Click **+ New custom report**.
3. Define the report's parameters.
4. Click **Create and launch report**.

On the Dashboard page

1. Filter any dashboard.
2. Scroll to the bottom.
3. Click **<n> events match your criteria. View Now.**



This takes you to the *Events* page, where you can navigate to **Analytics** > **+New Custom Report**. Define additional parameters for the report, then click **Create and launch report**.

Create Custom Rule Actions with the API

How-to

> Compatible versions: 4.2, 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion Server*, *Fusion AI*, *Fusion*

Custom rule actions allow you to define a unique action to take when certain conditions are met. They involve three primary components: a query pipeline, a custom rule type, and a rule that is triggered by conditions defined by the user.

This article describes the general workflow for creating a custom rule action. The example assumes the following components are used:

Component	Name
Query pipeline	`custom-rule-pipeline`
Custom rule type	`custom-rule-type`
Rule	`custom-rule-action`

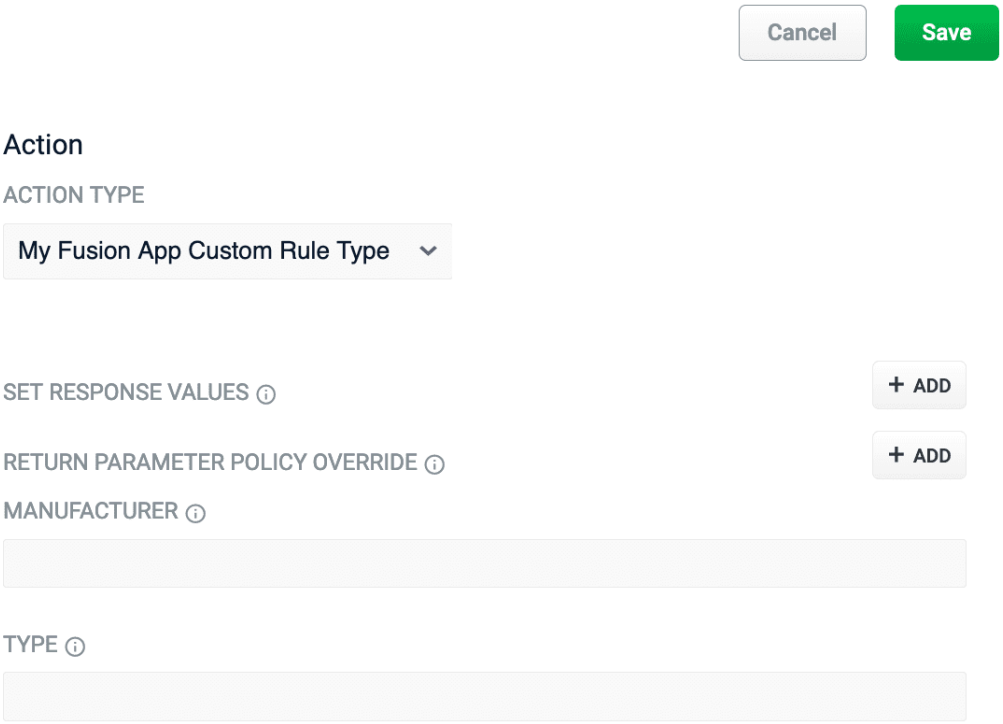
1. Create a query pipeline, `custom-rule-pipeline`, that is used to complete the rule action.
2. Create a custom rule type, `custom-rule-type`, that uses your query pipeline. Here are some important parameters to include in your `POST` request:

Required	Parameter	Description	Example
✅	`id`	A unique ID for the custom rule type. The `id` value must be unique across all apps in Fusion (global namespace).	`custom-rule-type`
✅	`pipeline_id`	The ID of the pipeline that is invoked during rule processing.	`custom-rule-pipeline`
✅	`display_type`	A user-friendly name for the custom rule type.	`My Custom Rule`
Rule properties, included as part of the `schema`, define the values required from the user to create a rule that uses the custom rule type.			
✅	`custom_param1`	The ID of the rule's parameter. This field is used in the API but is not visible in the Rules Editor.	`1st-rule-parameter`
✅	`type`	The type of user-input.	`string`
✅	`title`	A user-friendly name for the parameter.	`Manufacturer`
✅	`description`	A user-friendly description of what the field is used for. The `description` value is shown in the field's tooltip in the Rules Editor.	`The manufacturer of the product this rule action applies to.`

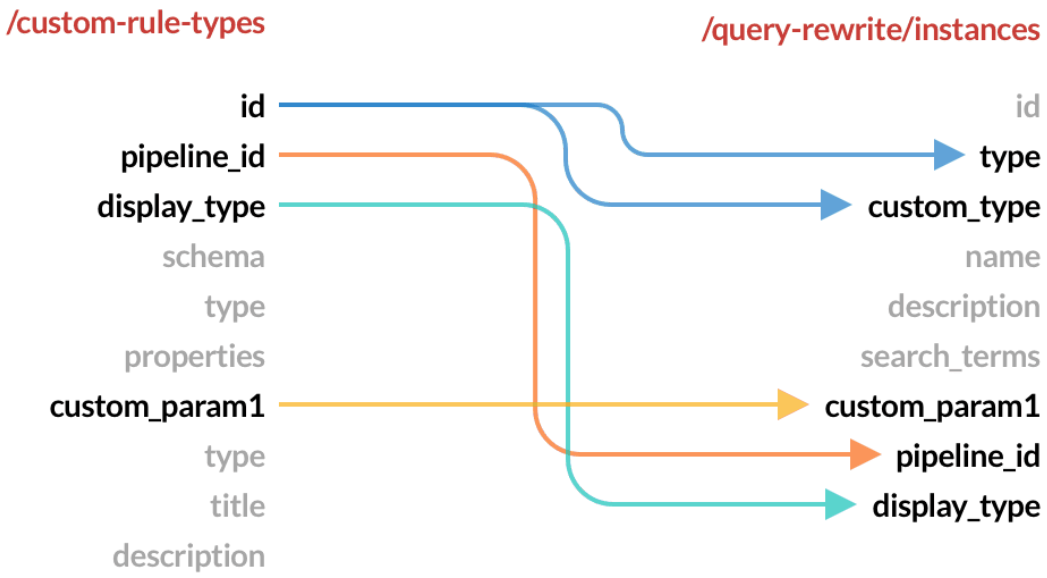
For example:

```
curl -u USERNAME:PASSWORD -XPOST -H "Content-type:application/json" \
https://FUSION_HOST:6764/api/apps/{app-name}/custom-rule-types -d '{
  "id":"{custom-rule-type}",
  "pipeline_id":"{query-pipeline}",
  "display_type":"My Custom Rule",
  "schema": {
    "type" : "object",
    "properties" : {
      "custom_param1" : {
        "type" : "string",
        "title" : "Manufacturer",
        "description" : "The manufacturer of the product this rule
action applies to."
      },
      "custom_param2" : {
        "type": "string",
        "title": "Type",
        "description": "The type of product this rule action applies
to."
      }
    }
  }
}'
```

1. Verify the custom rule type was created by visiting `https://FUSION_HOST:6764/api/apps/myFusionApp/custom-rule-types`. Custom rule types are also available in the Rules Editor:



2. Create a rule, `custom-rule-action`, that uses your custom rule, `custom-rule-type`. If you are creating the rule using the Query Rewrite API, the following image depicts which parameters are shared between the custom rule type and the rule:



Note that the custom rule type `id` value is used for the rule's `type` *and* `custom_type` fields. Remember to assign values to the custom rule properties, `custom_param1` and `custom_param2`.

```
curl -u USERNAME:PASSWORD -XPOST -H "Content-type:application/json" \
http://localhost:6764/api/apps/myFusionApp/query-rewrite/instances -d '{ "id":"123456", "type":"custom-rule-type", "custom_type":"custom-rule-type", "name":"My Rule", "description":"A rule that uses my custom rule, custom-rule-type.", "search_terms":["upcoming sale"], "custom_param1":"ABC", "custom_param2":"XYZ", "pipeline_id":"custom-rule-pipeline", "display_type":"My Custom Rule" }'
```

3. Verify your rule was created by checking the rule ID, `123456`, with the Query Rewrite API: `http://localhost:6764/api/apps/myFusionApp/query-rewrite/instances/123456`
4. Test your custom rule using your normal rule development process.

Create Custom Rule Actions with the API

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

Custom rule actions allow you to define a unique action to take when certain conditions are met. They involve three primary components: a query pipeline, a custom rule type, and a rule that is triggered by conditions defined by the user.

This article describes the general workflow for creating a custom rule action. The example assumes the following components are used:

Component	Name
Query pipeline	`custom-rule-pipeline`
Custom rule type	`custom-rule-type`
Rule	`custom-rule-action`

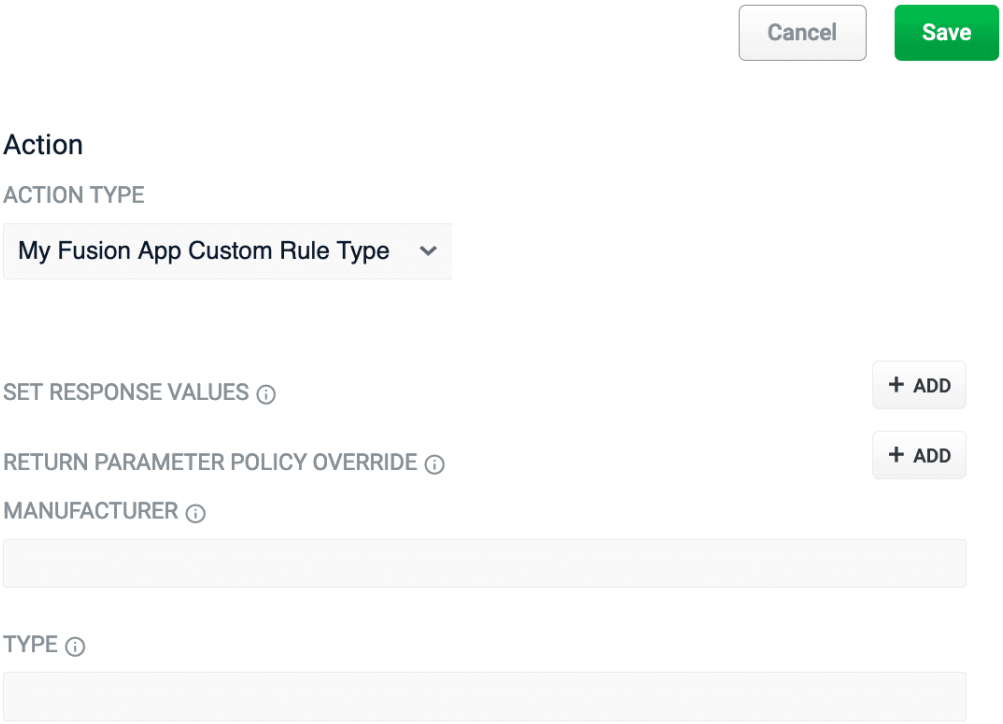
1. Create a query pipeline, `custom-rule-pipeline`, that is used to complete the rule action.
2. Create a custom rule type, `custom-rule-type`, that uses your query pipeline. Here are some important parameters to include in your `POST` request:

Required	Parameter	Description	Example
✅	`id`	A unique ID for the custom rule type. The `id` value must be unique across all apps in Managed Fusion (global namespace).	`custom-rule-type`
✅	`pipeline_id`	The ID of the pipeline that is invoked during rule processing.	`custom-rule-pipeline`
✅	`display_type`	A user-friendly name for the custom rule type.	`My Custom Rule`
Rule properties, included as part of the `schema`, define the values required from the user to create a rule that uses the custom rule type.			
✅	`custom_param1`	The ID of the rule's parameter. This field is used in the API but is not visible in the Rules Editor.	`1st-rule-parameter`
✅	`type`	The type of user-input.	`string`
✅	`title`	A user-friendly name for the parameter.	`Manufacturer`
✅	`description`	A user-friendly description of what the field is used for. The `description` value is shown in the field's tooltip in the Rules Editor.	`The manufacturer of the product this rule action applies to.`

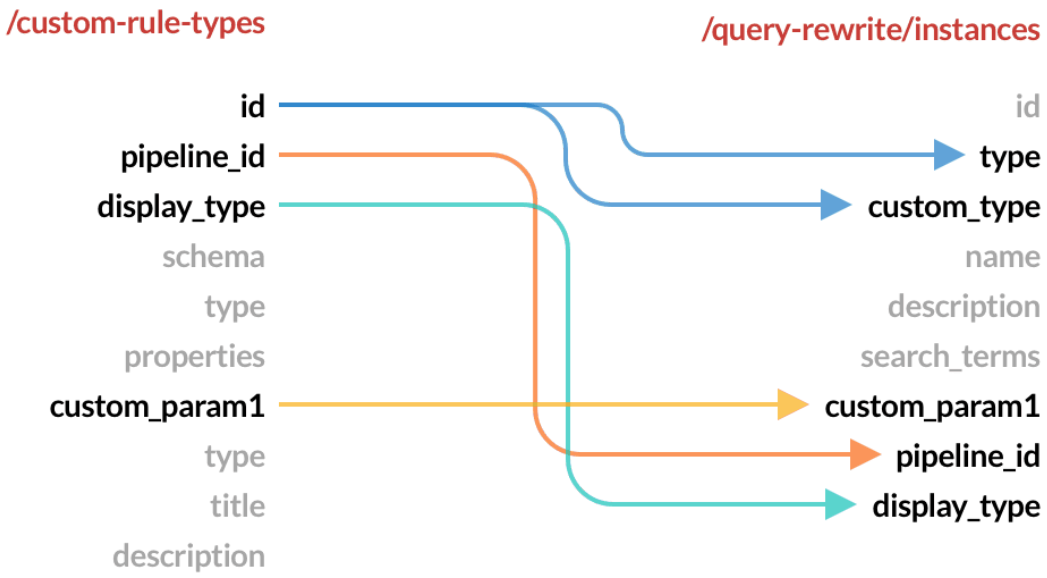
For example:

```
curl -u USERNAME:PASSWORD -XPOST -H "Content-type:application/json" \
https://EXAMPLE_COMPANY.b.lucidworks.cloud/api/apps/{app-name}/custom-
rule-types -d '{
  "id":"{custom-rule-type}",
  "pipeline_id":"{query-pipeline}",
  "display_type":"My Custom Rule",
  "schema": {
    "type" : "object",
    "properties" : {
      "custom_param1" : {
        "type" : "string",
        "title" : "Manufacturer",
        "description" : "The manufacturer of the product this rule
action applies to."
      },
      "custom_param2" : {
        "type": "string",
        "title": "Type",
        "description": "The type of product this rule action applies
to."
      }
    }
  }
}'
```

1. Verify the custom rule type was created by visiting `https://EXAMPLE_COMPANY.b.lucidworks.cloud/api/apps/myFusionApp/custom-rule-types`. Custom rule types are also available in the Rules Editor:



2. Create a rule, `custom-rule-action`, that uses your custom rule, `custom-rule-type`. If you are creating the rule using the Query Rewrite API, the following image depicts which parameters are shared between the custom rule type and the rule:



Note that the custom rule type `id` value is used for the rule's `type` *and* `custom_type` fields. Remember to assign values to the custom rule properties, `custom_param1` and `custom_param2`.

```
curl -u USERNAME:PASSWORD -XPOST -H "Content-type:application/json" \
https://EXAMPLE_COMPANY.b.lucidworks.cloud/api/apps/myFusionApp/query-rewrite/instances -d '{ "id":"123456",
"type":"custom-rule-type", "custom_type":"custom-rule-type",
"name":"My Rule", "description":"A rule that uses my custom rule, custom-rule-type.", "search_terms":["upcoming sale"],
"custom_param1":"ABC", "custom_param2":"XYZ",
"pipeline_id":"custom-rule-pipeline", "display_type":"My Custom Rule" }'
```

3. Verify your rule was created by checking the rule ID, **`123456`**, with the Query Rewrite API: **`https://EXAMPLE_COMPANY.b.lucidworks.cloud/api/apps/myFusionApp/query-rewrite/instances/123456`**
4. Test your custom rule using your normal rule development process.

Create Dashboards

How-to

> Compatible versions: 4.2, 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion Server*, *Fusion*

Create dashboards that meet the needs of users.

Tip

If you configure the dashboard to use existing data and add a Filtering panel, the panels in the dashboard work *as you add panels and configure them*. You do not need to wait until you have finished configuring the entire dashboard before viewing the panels recently added or edited.

Requirements for a dashboard

Requirements for a dashboard are:

- **Input panel(s).** You must add one or more input panels that let you specify which data the dashboard will display.
- **Filtering panel.** You must add a filtering panel to ensure that the dashboard functions correctly.
- **Output panel(s).** You must add one or more output panels to display the data.

Steps to create a dashboard

To create a dashboard, perform these actions in the order specified:

- *Choose the best starting point*
- *Configure dashboard settings*
- *Lay out the dashboard*
- *Configure the panels*
- *Save the dashboard*
- *Test the dashboard*

Choose a starting point

You can create a new dashboard from scratch or modify an existing dashboard.

To open the dashboard, complete the following:

1. From the Fusion workspace, click **Analytics** > **Dashboards**, or click **System** > **Log Viewer**.
2. In the upper right corner of the default dashboard, click the link for the dashboard you want to use as a starting point.

Configure dashboard settings

The dashboard settings affect all the panels on a dashboard. To configure settings, click **Configure dashboard**.

Option	Notes
Define where Fusion send queries from the dashboard	Click the **Solr** tab. • To use a Fusion query pipeline, select **Use Fusion**. • To send queries to Solr, deselect **Use Fusion**.
Set a default collection that Fusion queries when the dashboard opens	Click the **Solr** tab and specify the name of the collection to query. To use the same dashboard for multiple collections, let the dashboard user choose a collection. On the **Controls** tab, select **Show Collections Picker**.
Specify optional global query parameters to append to all dashboard queries	Click the **Solr** tab and enter **Global Query Parameters**. For example, to set the default search field to the field `message_t`, specify the global query parameter `&df=message_t`.

Lay out the dashboard

A dashboard consists of rows and panels, and possibly columns (if you use Column panels). The table provides the available options.

Option	Notes
Add a row	Add a new row to the dashboard.
Configure a row	Configure the selected row.
Add panel to an empty row	1. Click **Add panel to an empty row**. 2. Click **Add panel**. 3. Select **Panel Type**. 4. Configure the panel. 5. Click **Add Panel**.
Add a panel to a row that already contains panels	1. Click **Add a panel to this row**. 2. Click **Add panel**. 3. Select **Panel Type**. 4. Configure the panel. 5. Click **Add Panel**.
Row Full	An indicator on the left side of a row that is full.
Resize a panel	1. Click **Configure** . 2. Click the **General** tab. 3. In the **Span** field, specify a width between 1 (one twelfth) and 12 (twelve twelfths).
Drag a panel to a different location	1. In the upper right corner of the panel, click-and-hold the panel type. 2. Drag the panel to the new location and release the panel. If you drag the panel on top of a different panel, the existing panel is shifted to another location.
Remove Panel	Remove a specific panel from the dashboard.
Place panels in a column	Configure panels in a column format.

Option	Notes
Add text	Add text to the dashboard, such as instructions to use the dashboard or descriptions of the content in the dashboard.

Configure panels

This section describes general tasks that apply to all panels and tasks that apply to specific types of panels.

Important

As a best practice, we recommend you configure each panel as you add it. Then save and test to verify it is set correctly before you add another panel.

Option	Notes
 Configure	The tabs that are available based on panel types are described in this section. Not all tabs are present for all panel types. • General.* The name and width of the panel, whether it is editable, and whether the panel will let users inspect the queries. • Panel.* The primary tab for configuring the panel. This configuration affects *how* data is displayed, but also in some cases, *which* data is displayed. • Info.* Optionally, add a help message that Fusion displays when a user inspects the queries for a panel. • Fields.* Add and remove fields in a Table panel. • Paging.* Control pagination of a Table panel. • Queries.* View the panel query. Optionally, add one or more custom Solr query parameters. Fusion adds these to the panel query.
View and configure Histogram and Range Facet panels	Click the **View** up arrow to view and modify the configuration. Click the down arrow to hide the configuration.
 Help	Display help information for a setting.
 Inspect	View the query that a panel sends to Solr. Fusion displays both the panel query and the custom query if it exists.
Make a panel editable	You must first remove, then add the panel back. 1. Click **Remove Panel**. 2. Click either **Add panel to an empty row** or **Add Panel**. 3. Enter the appropriate values and click the **Editable** check box. 4. Click **Add Panel**.
Make a panel non-editable	1. Click **Configure**. 2. On the **General** tab, deselect the **Editable** check box. 3. Save the changes.

CONFIGURE INPUT PANEL SETTINGS

The elements on input panels determine the information displayed and is based on:

- Panel configuration
- Panel contributions to the query
- User interactions with the panels

Option	Notes
Add a custom query for a panel	Certain panel types let you specify a custom query to add to the panel query. The combined query is sent to the Fusion query pipeline or to Solr.
Filter	Filters select subsets of the data in a collection. For example, records in a time range or in which the field Gender has the value `Female`. A filtering panel is required for data interactivity to work. The filtering panel displays the filters that are in effect. You can also use a Filtering panel to add, remove, edit, and toggle filters.
Facet	Use a Facet panel to choose data in one or more facets. For example, if the facets Gender and Graduation Year are defined, a user could choose the Gender `Male` and the Graduation Year `2010`.

CONFIGURE SPECIFIC TYPES OF PANELS

Option	Notes
 Add Search Box	Click to add a query-entry field to a Query panel.
 Delete Search Box	Click to remove a search box from a Query panel.
Move a column	Click the right or left arrow, respectively, to shift the column one space in that direction.

Save the dashboard

You can save a dashboard to Solr, a file, or a GitHub gist. We recommend that you always save dashboards to Solr (even if you also save them as files or gists).

Option	Notes
Save in Solr	1. Click **Save**. 2. Under **Solr**, enter a name for the dashboard. 3. Check or deselect the **Make Public** checkbox. If this checkbox is selected, the dashboard is shared with everyone. 4. Click **Save**.
 Export	Click to enter the number of rows to export and select the destination file type. For example, CSV, XML, or JSON. Fusion downloads the dashboard as a file in the browser's download directory. The filename is named `dashboard\_name-large\_number`. For example, `Storefront-1502124761166`. The file is a JSON file. It lacks the file type `.json`, which is not really needed (you could add it). If you want to place the file in a different location, move it.
Save as a GitHub gist	When you save a dashboard as a GitHub gist, you do not need to sign in to Github first. Fusion stores the gist as an anonymous gist. You must configure the dashboard correctly to save as a GitHub gist. To configure the dashboard: 1. Click **Configure dashboard**. 2. Click the **Controls** tab. 3. In the **Allow saving to** field, select **Gist**. 4. At the top right of the dashboard, click **Save**. 5. Under **Gist**, enter a name for the dashboard. The name becomes the title of the dashboard, which is stored in the JSON file as the value of `"title"`. 6. Click **Save Gist** . Fusion displays the URL at which the gist resides. If you save the dashboard again, you get a different anonymous gist and a different URL. **Important** Navigate to the URL *immediately*. Save the URL. Bookmark it. Write it down. You cannot find the gist by searching for it, and Gist identifiers do not persist for long in the Fusion UI. (The last gist saved is available in the Load dialog for the duration of the browser session.)

Test the dashboard

View and interact with the dashboard, and verify each panel displays the expected data. If necessary, adjust the placement and settings of the panels based on user feedback.

Create Dashboards

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

Create dashboards that meet the needs of users.

Tip

If you configure the dashboard to use existing data and add a Filtering panel, the panels in the dashboard work *as you add panels and configure them*. You do not need to wait until you have finished configuring the entire dashboard before viewing the panels recently added or edited.

Requirements for a dashboard

Requirements for a dashboard are:

- **Input panel(s).** You must add one or more input panels that let you specify which data the dashboard will display.
- **Filtering panel.** You must add a filtering panel to ensure that the dashboard functions correctly.
- **Output panel(s).** You must add one or more output panels to display the data.

Steps to create a dashboard

To create a dashboard, perform these actions in the order specified:

- *Choose the best starting point*
- *Configure dashboard settings*
- *Lay out the dashboard*
- *Configure the panels*
- *Save the dashboard*
- *Test the dashboard*

Choose a starting point

You can create a new dashboard from scratch or modify an existing dashboard.

To open the dashboard, complete the following:

1. From the Managed Fusion workspace, click **Analytics** > **Dashboards**, or click **System** > **Log Viewer**.
2. In the upper right corner of the default dashboard, click the link for the dashboard you want to use as a starting point.

Configure dashboard settings

The dashboard settings affect all the panels on a dashboard. To configure settings, click **Configure dashboard**.

Option	Notes
Define where Managed Fusion send queries from the dashboard	Click the **Solr** tab. • To use a Managed Fusion query pipeline, select **Use Fusion**. • To send queries to Solr, deselect **Use Fusion**.
Set a default collection that Managed Fusion queries when the dashboard opens	Click the **Solr** tab and specify the name of the collection to query. To use the same dashboard for multiple collections, let the dashboard user choose a collection. On the **Controls** tab, select **Show Collections Picker**.
Specify optional global query parameters to append to all dashboard queries	Click the **Solr** tab and enter **Global Query Parameters**. For example, to set the default search field to the field `message_t`, specify the global query parameter `&df=message_t`.

Lay out the dashboard

A dashboard consists of rows and panels, and possibly columns (if you use Column panels). The table provides the available options.

Option	Notes
Add a row	Add a new row to the dashboard.
Configure a row	Configure the selected row.
Add panel to an empty row	1. Click **Add panel to an empty row**. 2. Click **Add panel**. 3. Select **Panel Type**. 4. Configure the panel. 5. Click **Add Panel**.
Add a panel to a row that already contains panels	1. Click **Add a panel to this row**. 2. Click **Add panel**. 3. Select **Panel Type**. 4. Configure the panel. 5. Click **Add Panel**.
Row Full	An indicator on the left side of a row that is full.
Resize a panel	1. Click **Configure** . 2. Click the **General** tab. 3. In the **Span** field, specify a width between 1 (one twelfth) and 12 (twelve twelfths).
Drag a panel to a different location	1. In the upper right corner of the panel, click-and-hold the panel type. 2. Drag the panel to the new location and release the panel. If you drag the panel on top of a different panel, the existing panel is shifted to another location.
Remove Panel	Remove a specific panel from the dashboard.
Place panels in a column	Configure panels in a column format.

Option	Notes
Add text	Add text to the dashboard, such as instructions to use the dashboard or descriptions of the content in the dashboard.

Configure panels

This section describes general tasks that apply to all panels and tasks that apply to specific types of panels.

Important

As a best practice, we recommend you configure each panel as you add it. Then save and test to verify it is set correctly before you add another panel.

Option	Notes
 Configure	The tabs that are available based on panel types are described in this section. Not all tabs are present for all panel types. • **General.** The name and width of the panel, whether it is editable, and whether the panel will let users inspect the queries. • **Panel.** The primary tab for configuring the panel. This configuration affects *how* data is displayed, but also in some cases, *which* data is displayed. • **Info.** Optionally, add a help message that Managed Fusion displays when a user inspects the queries for a panel. • **Fields.** Add and remove fields in a Table panel. • **Paging.** Control pagination of a Table panel. • **Queries.** View the panel query. Optionally, add one or more custom Solr query parameters. Managed Fusion adds these to the panel query.
View and configure Histogram and Range Facet panels	Click the **View** up arrow to view and modify the configuration. Click the down arrow to hide the configuration.
 Help	Display help information for a setting.
 Inspect	View the query that a panel sends to Solr. Managed Fusion displays both the panel query and the custom query if it exists.
Make a panel editable	You must first remove, then add the panel back. 1. Click **Remove Panel**. 2. Click either **Add panel to an empty row** or **Add Panel**. 3. Enter the appropriate values and click the **Editable** check box. 4. Click **Add Panel**.
Make a panel non-editable	1. Click **Configure**. 2. On the **General** tab, deselect the **Editable** check box. 3. Save the changes.

CONFIGURE INPUT PANEL SETTINGS

The elements on input panels determine the information displayed and is based on:

- Panel configuration
- Panel contributions to the query
- User interactions with the panels

Option	Notes
Add a custom query for a panel	Certain panel types let you specify a custom query to add to the panel query. The combined query is sent to the Managed Fusion query pipeline or to Solr.
Filter	Filters select subsets of the data in a collection. For example, records in a time range or in which the field Gender has the value `Female`. A filtering panel is required for data interactivity to work. The filtering panel displays the filters that are in effect. You can also use a Filtering panel to add, remove, edit, and toggle filters.
Facet	Use a Facet panel to choose data in one or more facets. For example, if the facets Gender and Graduation Year are defined, a user could choose the Gender `Male` and the Graduation Year `2010`.

CONFIGURE SPECIFIC TYPES OF PANELS

Option	Notes
 Add Search Box	Click to add a query-entry field to a Query panel.
 Delete Search Box	Click to remove a search box from a Query panel.
Move a column	Click the right or left arrow, respectively, to shift the column one space in that direction.

Save the dashboard

You can save a dashboard to Solr, a file, or a GitHub gist. We recommend that you always save dashboards to Solr (even if you also save them as files or gists).

Option	Notes
Save in Solr	1. Click **Save**. 2. Under **Solr**, enter a name for the dashboard. 3. Check or deselect the **Make Public** checkbox. If this checkbox is selected, the dashboard is shared with everyone. 4. Click **Save**.
 Export	Click to enter the number of rows to export and select the destination file type. For example, CSV, XML, or JSON. Managed Fusion downloads the dashboard as a file in the browser's download directory. The filename is named `dashboard\_name-large\_number`. For example, `Storefront-1502124761166`. The file is a JSON file. It lacks the file type `.json`, which is not really needed (you could add it). If you want to place the file in a different location, move it.
Save as a GitHub gist	When you save a dashboard as a GitHub gist, you do not need to sign in to Github first. Managed Fusion stores the gist as an anonymous gist. You must configure the dashboard correctly to save as a GitHub gist. To configure the dashboard: 1. Click **Configure dashboard**. 2. Click the **Controls** tab. 3. In the **Allow saving to** field, select **Gist**. 4. At the top right of the dashboard, click **Save**. 5. Under **Gist**, enter a name for the dashboard. The name becomes the title of the dashboard, which is stored in the JSON file as the value of `"title"`. 6. Click **Save Gist** . Managed Fusion displays the URL at which the gist resides. If you save the dashboard again, you get a different anonymous gist and a different URL. **Important** Navigate to the URL *immediately*. Save the URL. Bookmark it. Write it down. You cannot find the gist by searching for it, and Gist identifiers do not persist for long in the Managed Fusion UI. (The last gist saved is available in the Load dialog for the duration of the browser session.)

Test the dashboard

View and interact with the dashboard, and verify each panel displays the expected data. If necessary, adjust the placement and settings of the panels based on user feedback.

Create the Initial Admin Password with the API

How-to

> Compatible versions: 4.2, 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion Server*, *Fusion*

After creating a new instance of Fusion, you will need to create a password for the admin user. This is typically done in the Fusion UI. In some cases, such as during the development stage when the Admin UI microservice is not running, you may want to create this password using the API.

Use the following example commands to create the initial admin password with the API, in this case, `password123`:

In Fusion 5.x

```
curl -X POST -H 'content-type: application/json'
'http://localhost:6764/api' --data-binary '{"password":
"password123"}'
```

In Fusion 4.x

```
curl -X POST -H 'content-type: application/json'
'http://localhost:8764/api' --data-binary '{"password":
"password123"}'
```

Create the Upload Model Parameters to Cloud job

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

The Upload Model Parameters to Cloud job lets you store a trained model's training to a blob in cloud storage.

This document describes the steps to create the Upload Model Parameters to Cloud job and deploy a model with it.

For more information, see *Upload model parameters to Cloud*.

Deploy a model

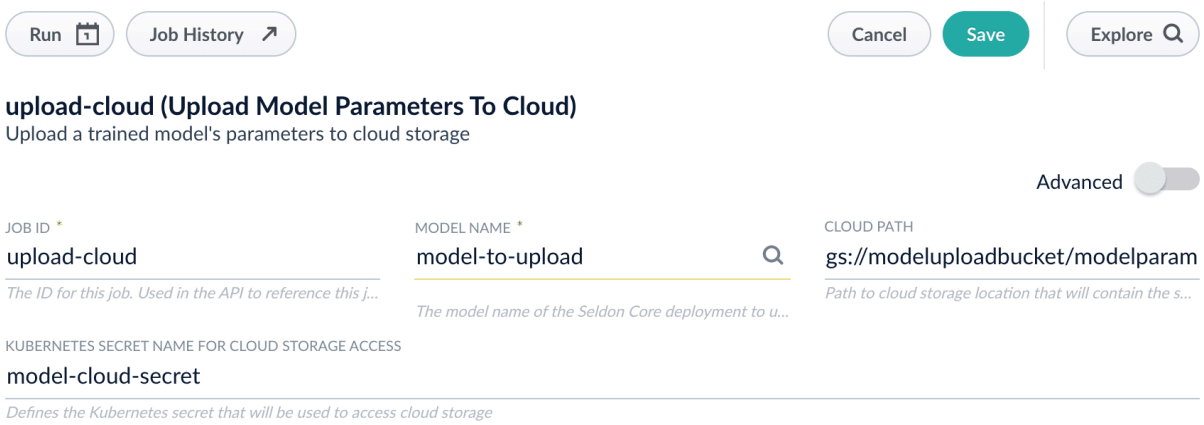
First, deploy a *Smart Answers supervised model*.

Create the job

1. Sign in to Managed Fusion and click your application.
2. Click **Collections > Jobs > Add+ > Upload Model Parameters to Cloud** to create a new instance of that job.
3. In the job ID field, enter a value of upload-cloud or a value appropriate for your model.
4. In the Model Name field, locate the name of the deployed model.
5. In the Cloud Path field, enter the location to store the parameters blob.
   You will also need to provide the Kubernetes secret that provides the access details for the cloud storage path.
6. Run the job, then check the cloud storage location to verify that the parameter file has been saved.
   The exported file will also have the version number (based on epoch time) of the model appended to it.

Caution

Take note of the full cloud path; you will need it to deploy the job later.



GCS

1. Obtain the Google Cloud Storage credentials JSON file for accessing the storage bucket you intend to use for transferring models.
2. Add the credentials JSON file as a secret with the following command:

```
kubectl create secret generic model-cloud-secret --from-
file=credentials.json -n [NAMESPACE]
```

S3

1. Create a JSON file with the following keys:

Note

`"region"` is optional and will be set to `us-east-1` if a value is not entered

```
{ "key": [KEY], "secret": [SECRET], "region": [REGION] }
```

2. Upload the JSON file as a secret in the cluster:

```
kubectl create secret generic model-cloud-secret --from-
file=model.json -n [NAMESPACE]
```

The secret can now be referenced as `model-cloud-secret` in the job configuration.

Azure Blob Storage

1. Create a JSON file with the layout below:

```
{ "key": [KEY], "secret": [SECRET] }
```

2. Create the `model-cloud-secret` secret in the Kubernetes cluster:

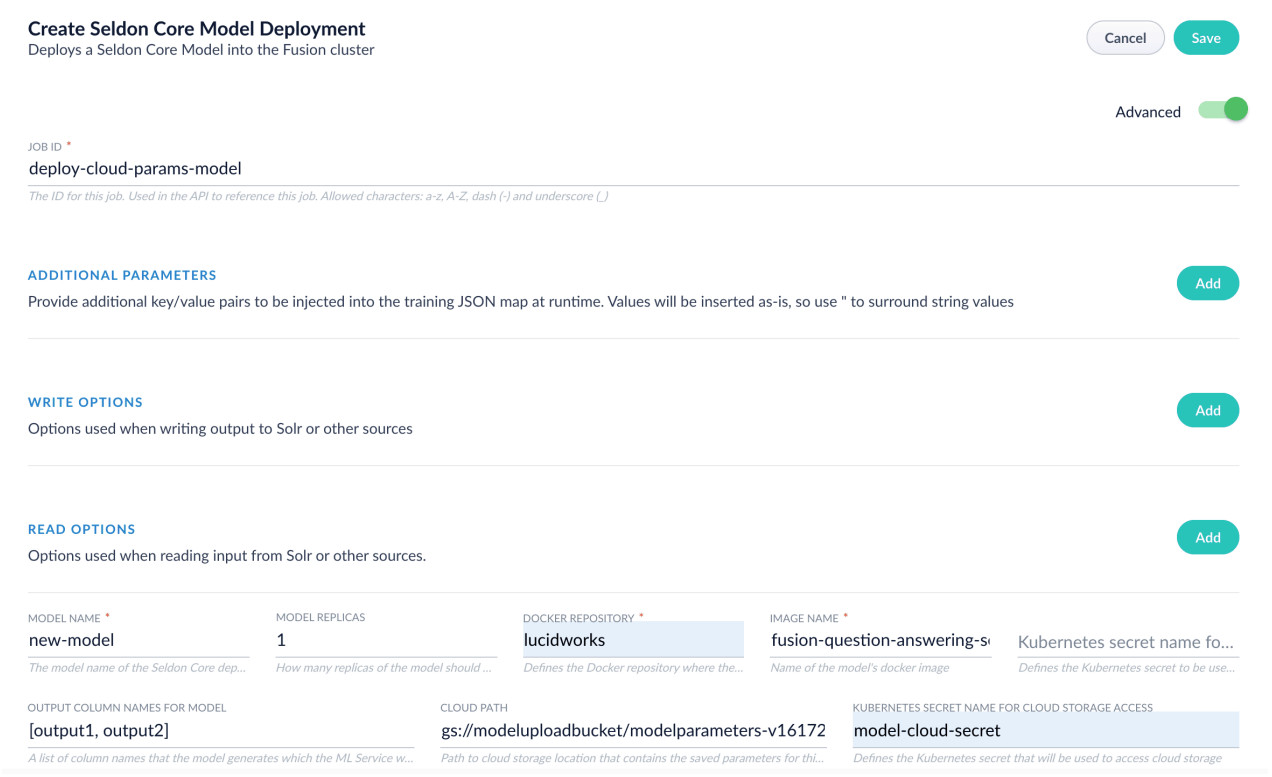
```
kubectl create secret generic model-cloud-secret --from-
file=model.json -n [NAMESPACE]
```

Deploy a new model with the Upload Model Parameters job

The new model can run in either the same Managed Fusion cluster or a different cluster.

If you are planning to transfer a model to a different cluster, note that you will need to set up the Kubernetes secrets in that cluster as well, though the access for deploying can be limited to read-only.

1. Click to create new **Create Seldon Core Model Deployment** job and toggle **Advanced**.
2. Enter the full cloud path, including the version number, in the *Cloud Path* box.
3. Enter the Kubernetes secret that will be used to access the path in the *Kubernetes Secret Name for Cloud Storage Access* box.
4. For this example, you will be using the job to deploy a trained Smart Answers model, so use the `fusion-question-answering-seldon:5.4.0` model image from the `lucidworks` repository in the *Image Name* box.



4. Run the job. Confirm that the model deploys into the cluster and is running.

Creating the Upload Model Parameters to Cloud Job

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion*

The Upload Model Parameters to Cloud job lets you store a trained model's training to a blob in cloud storage.

This document describes the steps to create the Upload Model Parameters to Cloud job and deploy a model with it.

See *Upload Model Parameters to Cloud* to learn more about the Upload Model Parameters to Cloud job.

Deploy a model

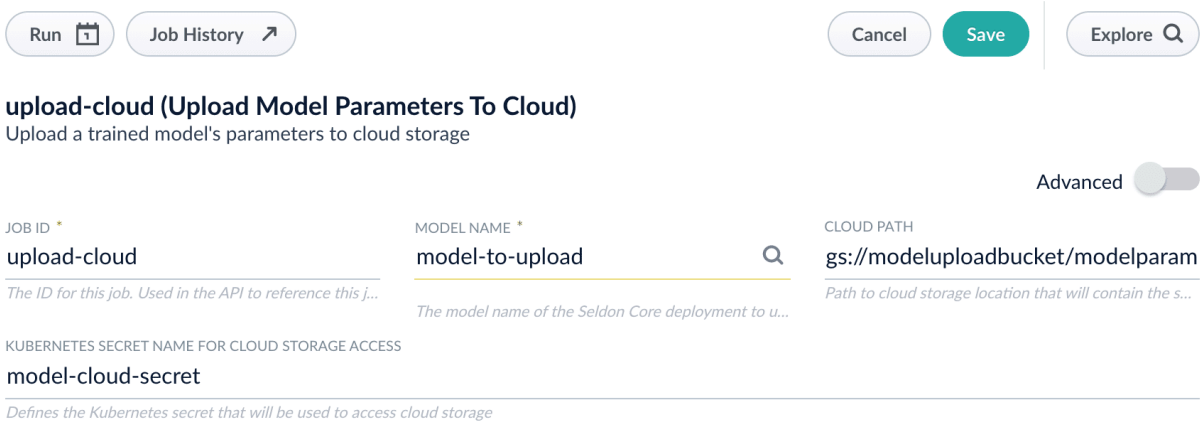
First, deploy a ***Smart Answers Supervised Model***.

Create the job

1. In the **Collections > Jobs** menu, click **Add +** and select **Upload Model Parameters to Cloud** to create a new instance of that job.
2. Set the job ID to `upload-cloud` (or something appropriate for your model) and find the name of the deployed model in the **Model Name** text box.
3. Enter the **Cloud Path** to direct to the location you would like to store the parameters blob in.
   You will also need to provide the Kubernetes secret that provides the access details for the cloud storage path.
4. Run the job, then check the cloud storage location to verify that the parameter file has been saved.
   The exported file will also have the version number (based on epoch time) of the model appended to it.

Caution

Take note of the full cloud path; you will need it to deploy the job later.



GCS

1. Obtain the Google Cloud Storage credentials JSON file for accessing the storage bucket you intend to use for transferring models.
2. Add the credentials JSON file as a secret with the following command:

```
kubectl create secret generic model-cloud-secret --from-
file=credentials.json -n [NAMESPACE]
```

S3

1. Create a JSON file with the following keys:

Note

`"region"` is optional and will be set to `us-east-1` if a value is not entered

```
{ "key": [KEY], "secret": [SECRET], "region": [REGION] }
```

2. Upload the JSON file as a secret in the cluster:

```
kubectl create secret generic model-cloud-secret --from-
file=model.json -n [NAMESPACE]
```

The secret can now be referenced as `model-cloud-secret` in the job configuration.

Azure Blob Storage

1. Create a JSON file with the layout below:

```
{ "key": [KEY], "secret": [SECRET] }
```

2. Create the `model-cloud-secret` secret in the Kubernetes cluster:

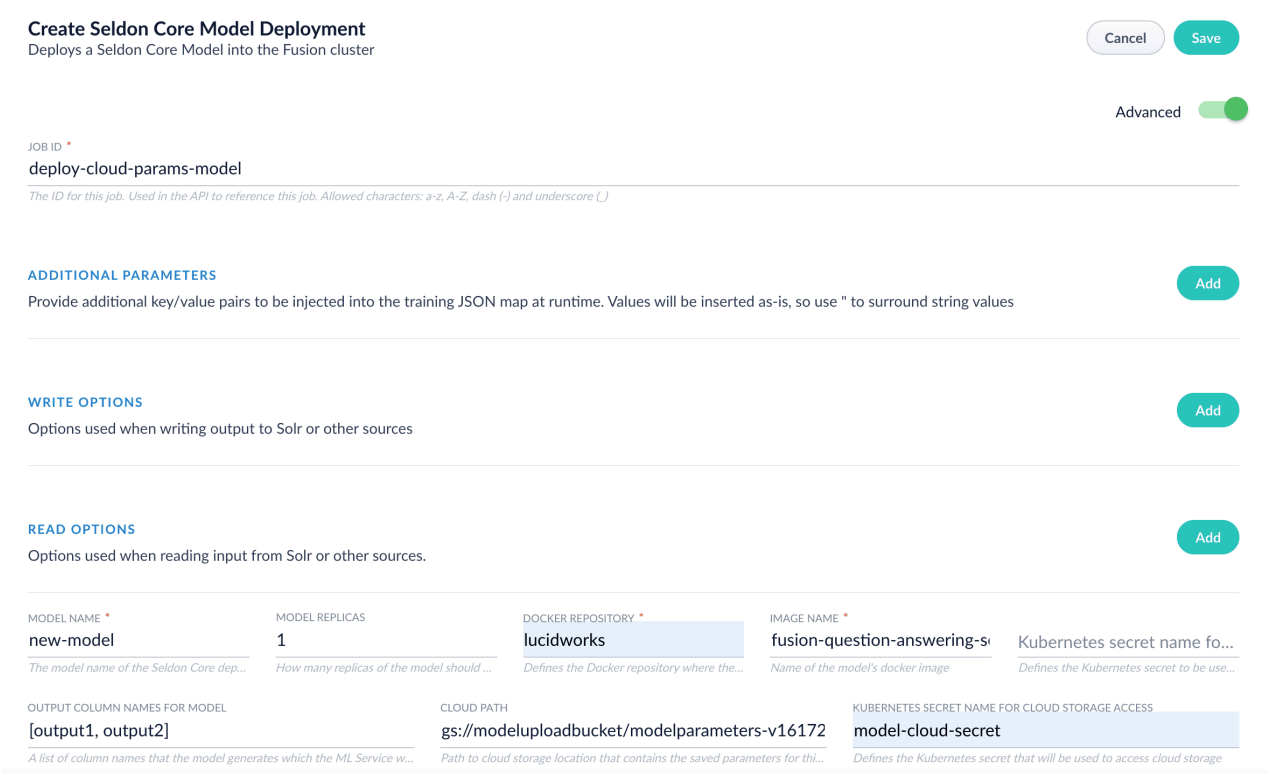
```
kubectl create secret generic model-cloud-secret --from-
file=model.json -n [NAMESPACE]
```

Deploy a new model with the Upload Model Parameters job

The new model can run in either the same Fusion cluster or a different cluster.

If you are planning on transferring a model to a different cluster, note that you will need to set up the Kubernetes secrets in that cluster as well, though the access for deploying can be limited to read-only.

1. Create a new **Create Seldon Core Model Deployment** job and toggle **Advanced**.
2. Enter the full cloud path, including the version number, in the *Cloud Path* box.
3. Enter the Kubernetes secret that will be used to access the path in the *Kubernetes Secret Name for Cloud Storage Access* box.
4. For this example, you will be using the job to deploy a trained Smart Answers model, so use the `fusion-question-answering-seldon:5.4.0` model image from the `lucidworks` repository in the *Image Name* box.



4. Run the job. Confirm that the model deploys into the cluster and is running.

Debug Custom JavaScript Index Stages

How-to

> Compatible versions: 4.2, 5.5
>
> Works with: *Fusion Server*, *Fusion*

To debug a JavaScript Index stage you can:

- Check the Fusion API server logs for errors.
- Use the `logger` object for print debugging (in the Fusion log file `api.log`).

The JavaScript Engine Used by Fusion

The JavaScript engine used by Fusion is the Nashorn engine from Oracle. See The Nashorn Java API for details.

Creating and accessing Java types

The following information is taken from Oracle's JavaScript programming guide section 3, Using Java From Scripts.

To create script objects that access and reference Java types from Javascript use the `Java.type()` function:

```
var ArrayList = Java.type("java.util.ArrayList"); var a = new ArrayList;
```

Deduplicate web content using canonical tags

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

The Web V1 connector retrieves data from a Web site using HTTP and starting from a specified URL.

In content management and online shopping systems, it is common for the same content to be accessed through multiple URLs. Content syndication helps you distribute content to different URLs and domains, consolidate link signals for the duplicate or similar content, and track metrics for a single product or topic. But it creates some challenges when people use search engines to reach your page.

The Fusion Web connector can leverage canonical meta tags in your website's HTML to deduplicate web pages.

To deduplicate web pages using canonical tags in the Fusion UI:

1. When configuring your Web datasource, toggle **Advanced** at the top of the page.
2. Under **Dedupe**, click **Dedupe documents**.
3. Make sure **Deduplication via canonical tag** is checked.

Deduplicate Web Content using Canonical Tags

How-to

> Compatible versions: 4.2, 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion Connectors*, *Fusion Server*, *Fusion*

The Web V1 connector retrieves data from a Web site using HTTP and starting from a specified URL.

In content management and online shopping systems, it is common for the same content to be accessed through multiple URLs. Content syndication helps you distribute content to different URLs and domains, consolidate link signals for the duplicate or similar content, and track metrics for a single product or topic. But it creates some challenges when people use search engines to reach your page.

The Fusion Web connector can leverage canonical meta tags in your website's HTML to deduplicate web pages.

To deduplicate web pages using canonical tags in the Fusion UI:

1. When configuring your Web datasource, toggle **Advanced** at the top of the page.
2. Under **Dedupe**, click **Dedupe documents**.
3. Make sure **Deduplication via canonical tag** is checked.

Delete a Connector

How-to

> Compatible versions: 4.2, 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion Connectors*, *Fusion Server*, *Fusion*

You can delete a connector using the Fusion UI or the *Blob Store API*.

Deleting a connector using the Fusion UI

1. In the Fusion UI, navigate to **System** > **Blobs**.
2. Under **Connector Plugin**, select the connector to delete.
3. Click **Delete Blob**.
   Fusion prompts you to confirm that you want to delete the blob.
4. Click **Yes, Delete**.
   The connector disappears from the blob list.

Deleting a connector using the REST API

1. Get the list of blobs of the connector plugin type:

```
curl –u USERNAME:PASSWORD http://localhost:8764/api/blobs?
resourceType=plugin:connector
```

2. Locate the connector you want to delete, and copy its ID.
   For example, the Jive connector ID is `lucid.jive` :

```
{ "name" : "lucid.jive", "contentType" : "application/zip",
"size" : 125302, "modifiedTime" : "2017–06–13T17:49:20.171Z",
"version" : 1570112704530612224, "md5" :
"7032bf2c038bb2d1e27aee82c056c0fb", "metadata" : {
"connectorBootstrapPluginName" : "lucid.jive", "resourceType" :
"plugin:connector" } }
```

3. Delete the connector as follows:

```
curl –u USERNAME:PASSWORD –X DELETE
http://localhost:8764/api/blobs/<id>
```

   For example

```
curl –u USERNAME:PASSWORD –X DELETE
http://localhost:8764/api/blobs/lucid.jive
```

A null response indicates success. You can verify that the connector is deleted like this:

```
curl –u USERNAME:PASSWORD http://localhost:8764/api/blobs | grep
lucid.jive
```

Delete a Connector

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

You can delete a connector using the Managed Fusion UI or the *Blob Store API*.

Deleting a connector using the Managed Fusion UI

1. In the Managed Fusion UI, navigate to **System** > **Blobs**.
2. Under **Connector Plugin**, select the connector to delete.
3. Click **Delete Blob**.
   Managed Fusion prompts you to confirm that you want to delete the blob.
4. Click **Yes, Delete**.
   The connector disappears from the blob list.

Deleting a connector using the REST API

1. Get the list of blobs of the connector plugin type:

```
curl -u USERNAME:PASSWORD
https://EXAMPLE_COMPANY.b.lucidworks.cloud/api/blobs?
resourceType=plugin:connector
```

2. Locate the connector you want to delete, and copy its ID.
   For example, the Jive connector ID is `lucid.jive` :

```
{ "name" : "lucid.jive", "contentType" : "application/zip",
"size" : 125302, "modifiedTime" : "2017-06-13T17:49:20.171Z",
"version" : 1570112704530612224, "md5" :
"7032bf2c038bb2d1e27aee82c056c0fb", "metadata" : {
"connectorBootstrapPluginName" : "lucid.jive", "resourceType" :
"plugin:connector" } }
```

3. Delete the connector as follows:

```
curl -u USERNAME:PASSWORD -X DELETE
https://EXAMPLE_COMPANY.b.lucidworks.cloud:8764/api/blobs/<id>
```

For example

```
curl -u USERNAME:PASSWORD -X DELETE
https://EXAMPLE_COMPANY.b.lucidworks.cloud:8764/api/blobs/lucid.
jive
```

A null response indicates success. You can verify that the connector is deleted like this:

```
curl -u USERNAME:PASSWORD
https://EXAMPLE_COMPANY.b.lucidworks.cloud:8764/api/blobs | grep
lucid.jive
```

Delete a Query Pipeline

How-to

> Compatible versions: 4.2, 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion Server*, *Fusion*

1. In the Fusion workspace, navigate to **Querying** > **Query Profiles**.
2. In the profiles list, select the query profile to delete.
3. In the query profile configuration panel, click **Delete profile**.

Delete a Query Pipeline

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

1. In the Fusion workspace, navigate to **Querying** > **Query Profiles**.
2. In the profiles list, select the query profile to delete.
3. In the query profile configuration panel, click **Delete profile**.

Delete an index profile

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

1. Sign in to Managed Fusion and click your application.
2. Click **Indexing > Index Profiles**.
3. In the profiles list, select the index profile to delete.
4. In the index profile configuration panel, click **Delete profile**.

Delete an Index Profile

How-to

> Compatible versions: 4.2, 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion Server*, *Fusion*

1. In the Fusion workspace, navigate to **Indexing** > **Index Profiles**.
2. In the profiles list, select the index profile to delete.
3. In the index profile configuration panel, click **Delete profile**.

Deploy a pre-trained scispaCy model

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

Important

This article uses a pre-packaged model, which you *do not* need to download to deploy. To use the pre-packaged model, skip to *Deploy model to Fusion*. The section *Create the model* describes how to complete this process on your own.

Create the model (OPTIONAL)

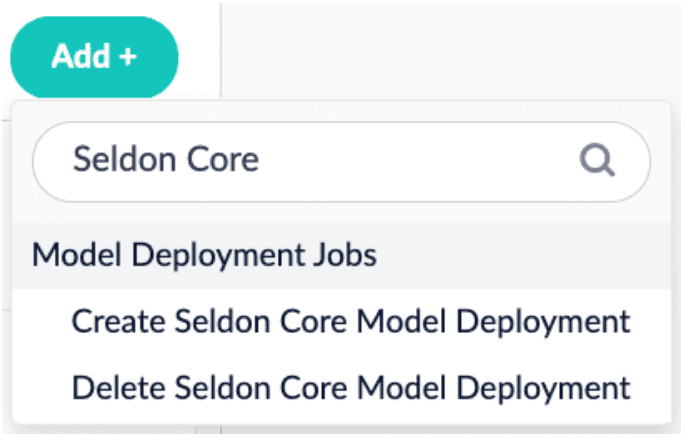
Note

Skip this section to use the pre-packaged model.

1. *Download the* `scispacy.ipynb` *file* and open it in Jupyter Notebook (or a similar alternative).
2. Follow the steps in the notebook, substituting your custom values as needed.

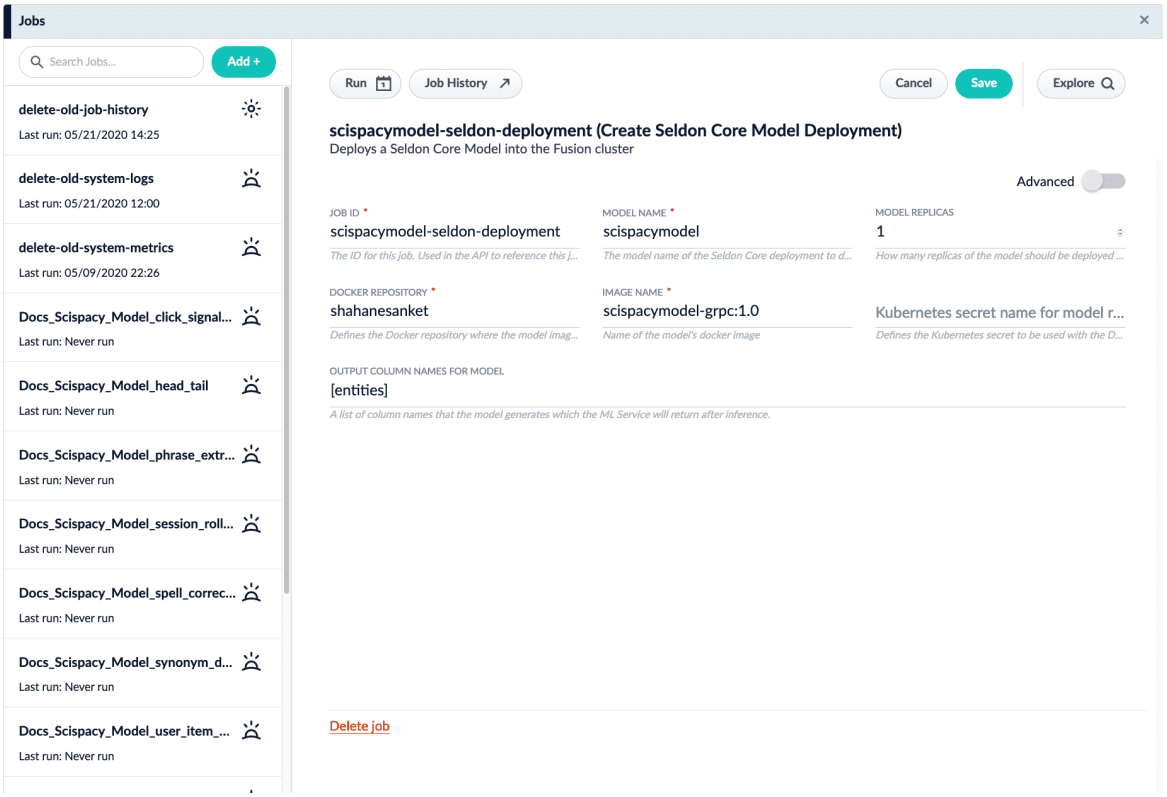
Deploy model to Fusion

1. Navigate to **Collections > Jobs**.
2. Click the **Add** button.
3. Select the **Create Seldon Core Model Deployment** under **Model Deployment Jobs**.



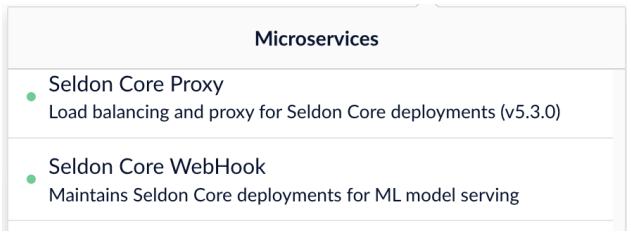
4. Enter the values for your model deployment. If you are using the pre-packaged model, use the following values:

Parameter	Value
Jobs ID	`scispacymodel-seldon-deployment`
Model Name	`scispacymodel`
Docker Repository	`shahanesanket`
Image Name	`scispacymodel-grpc:1.0`
Output Column Names for Model	`[entities]`



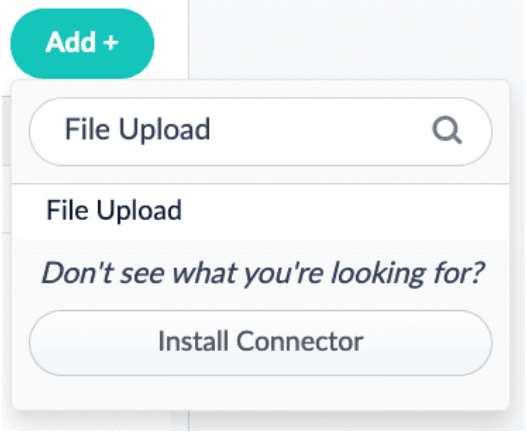
5. Run the job by clicking **Run** and selecting **Start**.

When the job completes successfully, the model is deployed. Check the list of microservices to verify:

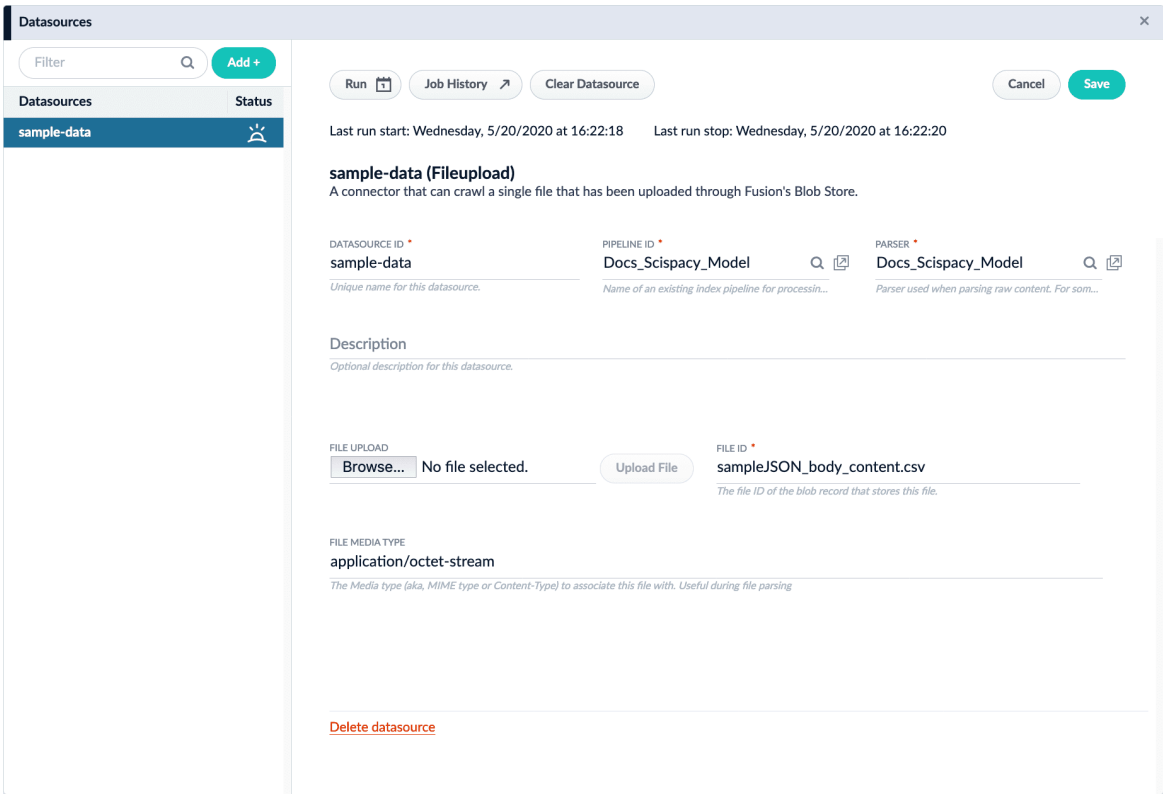


Import sample data

1. *Download and save* the sample data file `sampleJSON_body_content.csv`.
2. Navigate to **Indexing > Datasources**.
3. Click the **Add** button and select **File Upload V2**.



4. Click **Browse**, select the `sampleJSON_body_content.csv` file, and click **Open**. Click the **Upload File** button to complete the upload process.
5. Assign a value to the **Datasource ID** parameter. This article uses the ID `sample-data`.



6. Click the **Save** button.

Create a Machine Learning stage in the Index Workbench

1. Navigate to **Indexing > Index Workbench**.
2. Click the **Load** button.
3. Choose the datasource you created.
4. Click the **Add a Stage** button.
5. Choose the **Machine Learning** stage.
6. In the **Model ID** field, enter `scispacymodel`.
7. In the **Model input transformation script** field, enter the following script:

```
var modelInput = new java.util.HashMap() var list = new
java.util.ArrayList() list.add(doc.getFirstFieldValue("body_t"))
modelInput.put("text", list) modelInput
```

8. In the **Model output transformation script** field, enter the following script:

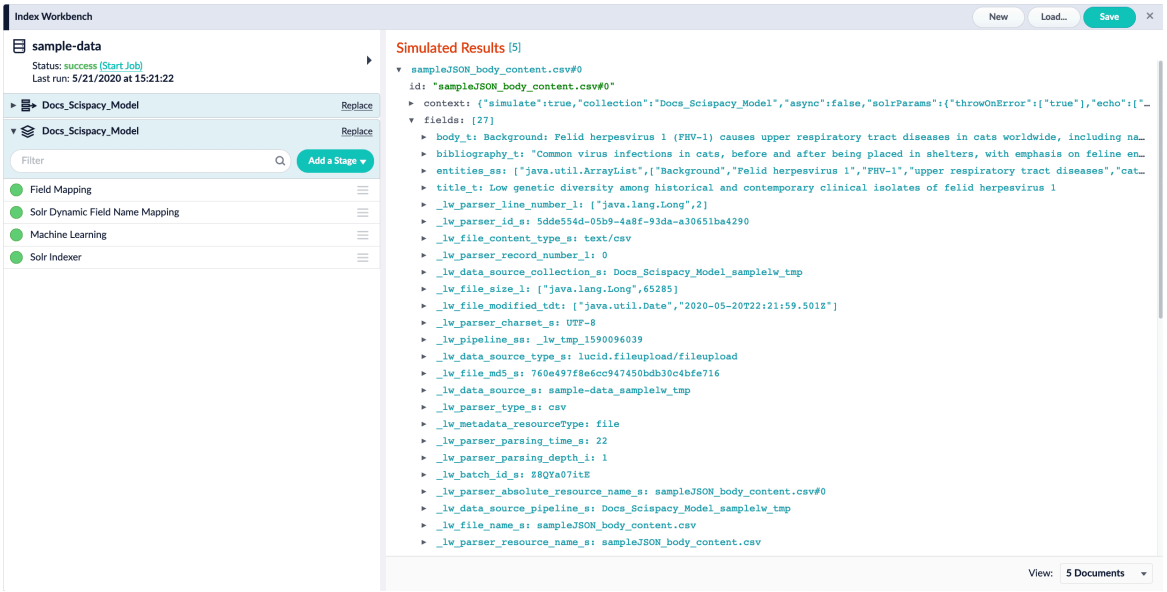
```
doc.addField("entities_ss", modelOutput.get("entities"))
```

9. Click the **Apply** button.

Verify results

1. Click the **Start Job** button, and allow the job to finish.

2. Check the simulated results. If everything was successful, the results will resemble this:



Deploy a Pre-trained scispaCy Model

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion*

Important

This article uses a pre-packaged model, which you *do not* need to download to deploy. To use the pre-packaged model, skip to *Deploy model to Fusion*. The section *Create the model* describes how to complete this process on your own.

Create the model (OPTIONAL)

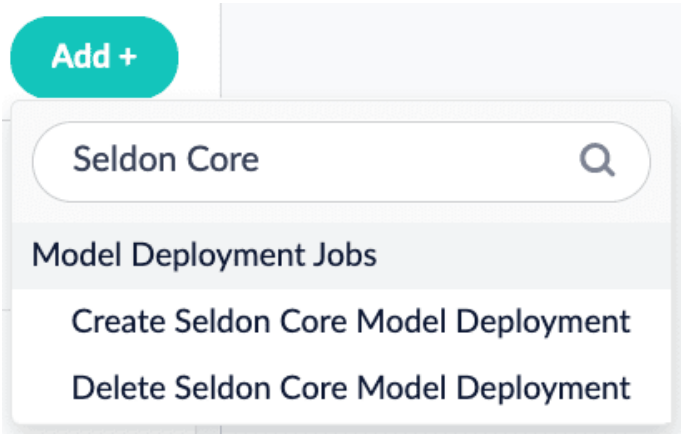
Note

Skip this section to use the pre-packaged model.

1. *Download the* `scispacy.ipynb` *file* and open it in Jupyter Notebook (or a similar alternative).
2. Follow the steps in the notebook, substituting your custom values as needed.

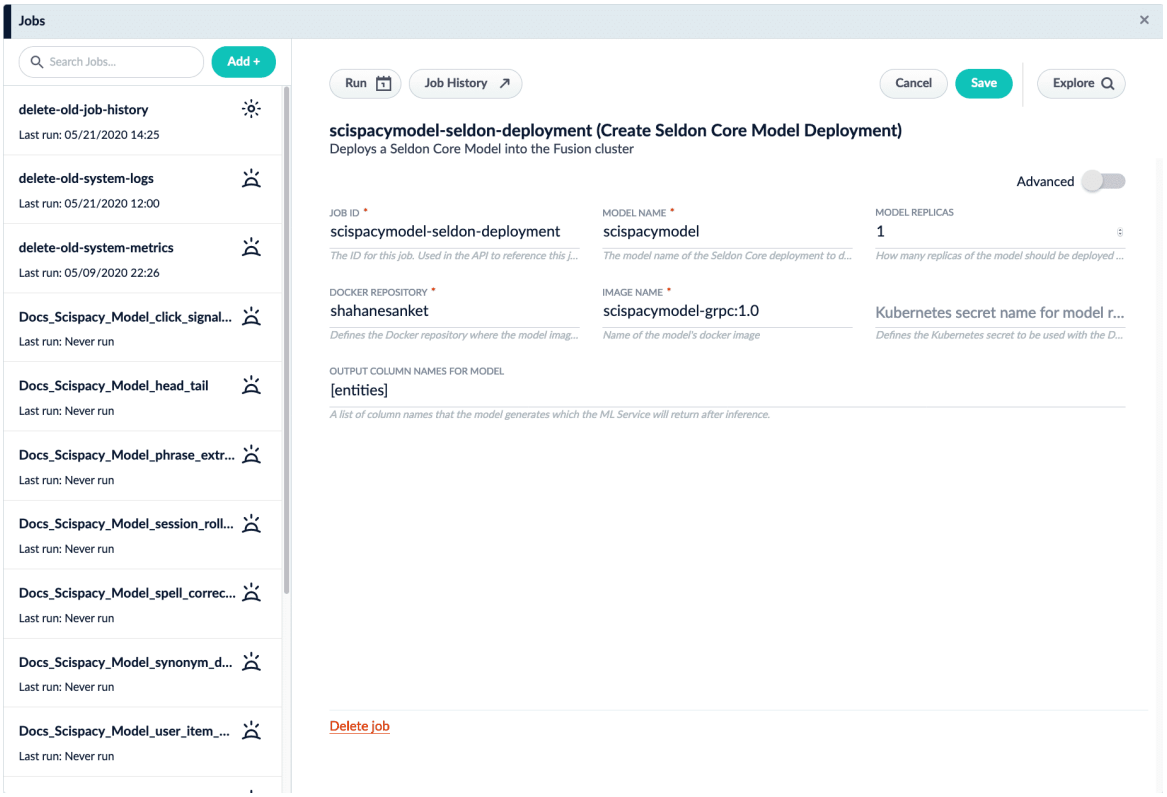
Deploy model to Fusion

1. Navigate to **Collections > Jobs**.
2. Click the **Add** button.
3. Select the **Create Seldon Core Model Deployment** under **Model Deployment Jobs**.



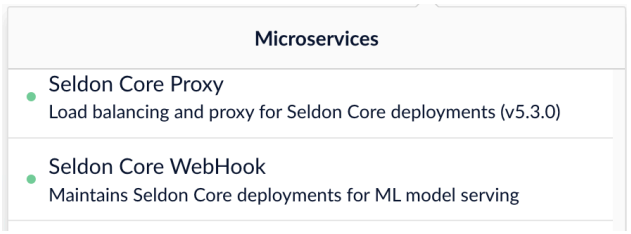
4. Enter the values for your model deployment. If you are using the pre-packaged model, use the following values:

Parameter	Value
Jobs ID	`scispacymodel-seldon-deployment`
Model Name	`scispacymodel`
Docker Repository	`shahanesanket`
Image Name	`scispacymodel-grpc:1.0`
Output Column Names for Model	`[entities]`



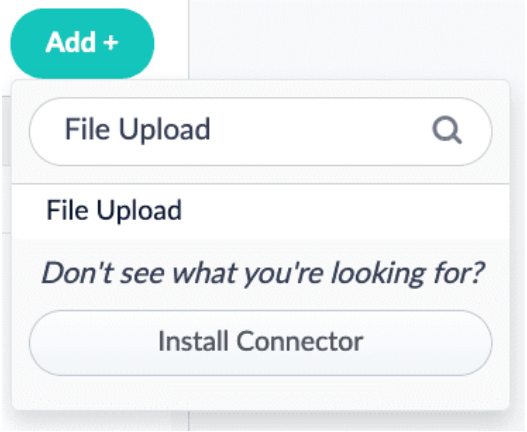
5. Run the job by clicking **Run** and selecting **Start**.

When the job completes successfully, the model is deployed. Check the list of microservices to verify:

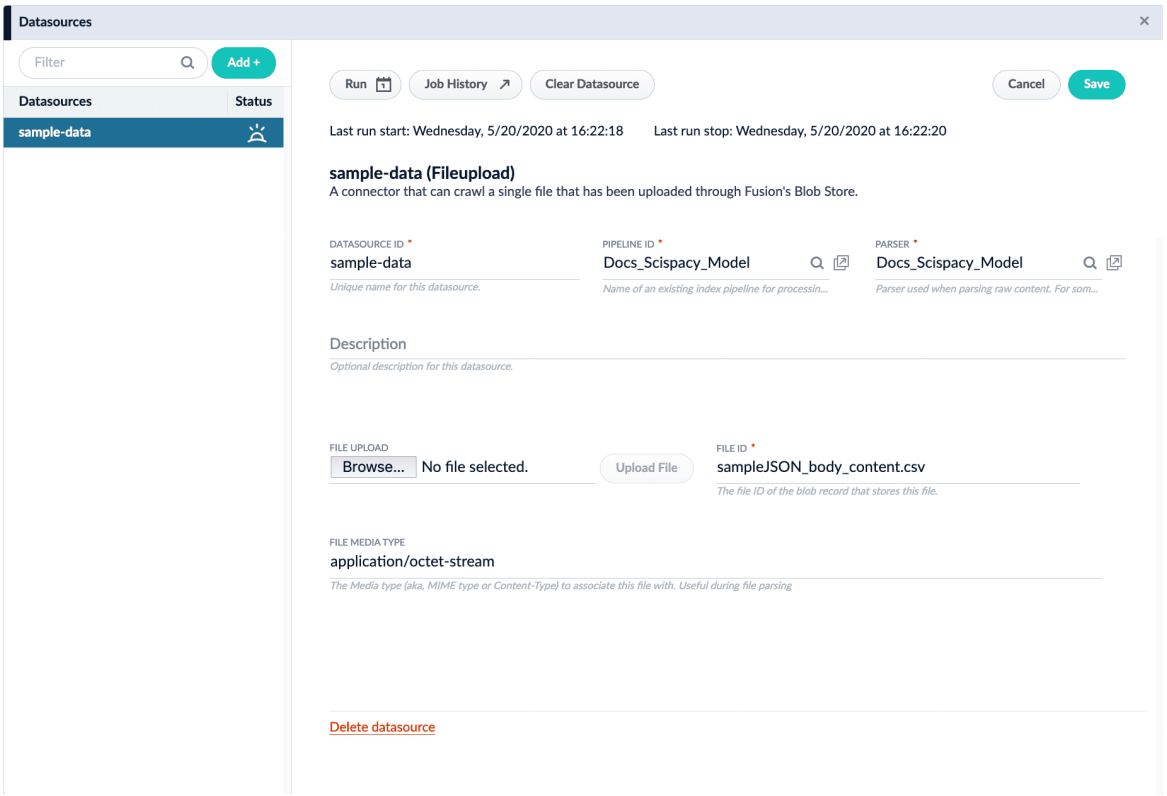


Import sample data

1. *Download and save* the sample data file `sampleJSON_body_content.csv`.
2. Navigate to **Indexing > Datasources**.
3. Click the **Add** button and select **File Upload V2**.



4. Click **Browse**, select the `sampleJSON_body_content.csv` file, and click **Open**. Click the **Upload File** button to complete the upload process.
5. Assign a value to the **Datasource ID** parameter. This article uses the ID `sample-data`.



6. Click the **Save** button.

Create a Machine Learning stage in the Index Workbench

1. Navigate to **Indexing > Index Workbench**.
2. Click the **Load** button.
3. Choose the datasource you created.
4. Click the **Add a Stage** button.
5. Choose the **Machine Learning** stage.
6. In the **Model ID** field, enter `scispacymodel`.
7. In the **Model input transformation script** field, enter the following script:

```
var modelInput = new java.util.HashMap() var list = new
java.util.ArrayList() list.add(doc.getFirstFieldValue("body_t"))
modelInput.put("text", list) modelInput
```

8. In the **Model output transformation script** field, enter the following script:

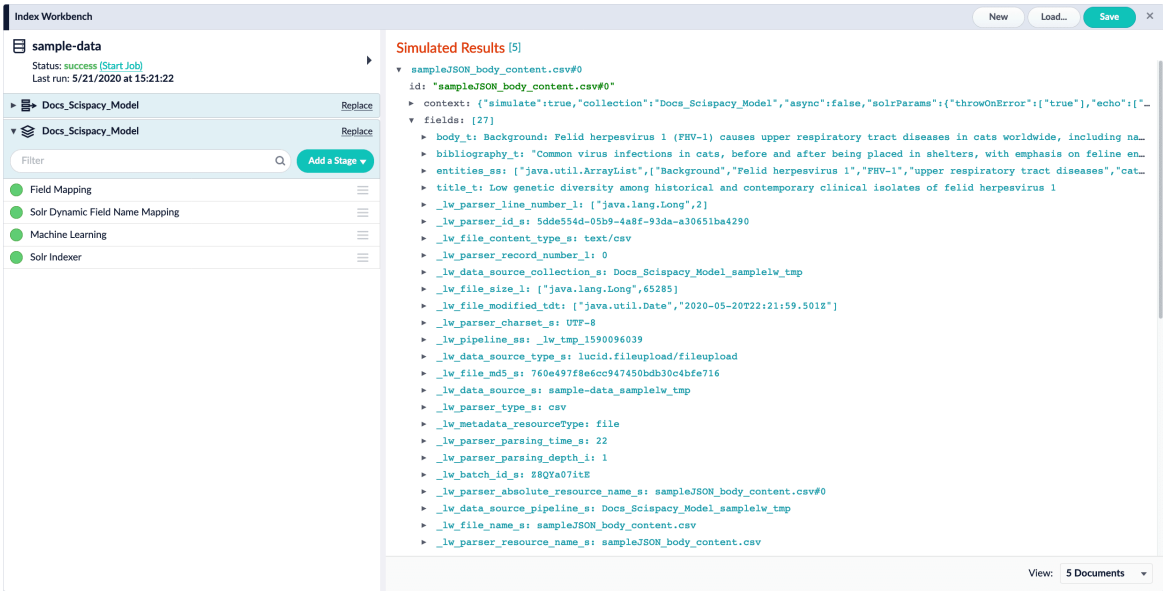
```
doc.addField("entities_ss", modelOutput.get("entities"))
```

9. Click the **Apply** button.

Verify results

1. Click the **Start Job** button, and allow the job to finish.

2. Check the simulated results. If everything was successful, the results will resemble this:



Deploy a Search App in Fusion's Webapps Service (4.0)

How-to

> Compatible versions: 4.2
>
> Works with: *App Studio*

You can use App Studio to deploy a search app in Fusion Server. You develop your search interface, and then publish the search app to Fusion's *Webapps service*.

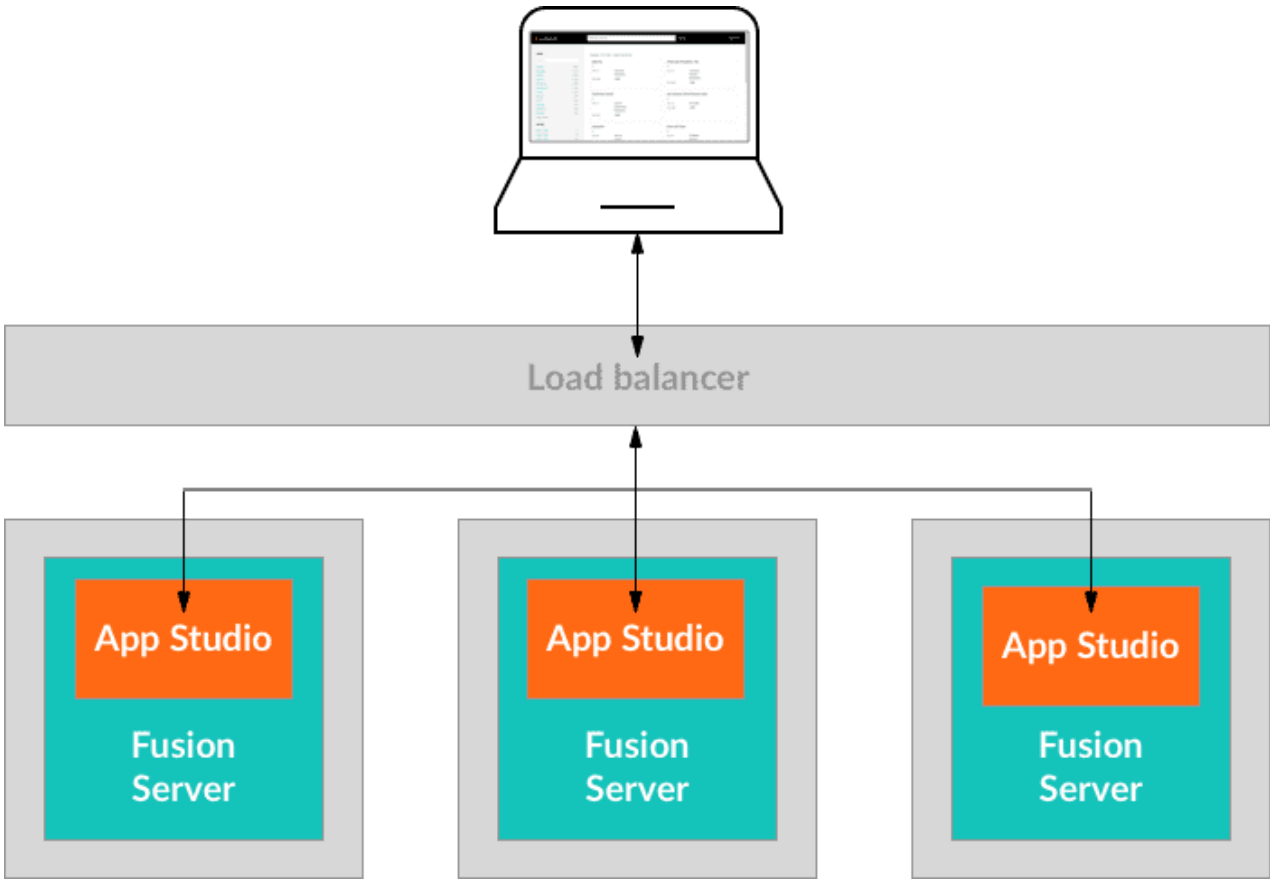
Important

Simplified instructions can be found in *Deploy App Studio*.

Note

This deployment type leverages your existing production environment, requiring no dedicated nodes for App Studio. However, you might need to deploy additional Fusion nodes if you anticipate significant traffic to your search interface.

UI deployment overview



Here is an overview of the steps:

1. *Install App Studio*.
2. *Develop your search interface*.
3. Publish your UI to Fusion Server using the instructions below.

Your webapp becomes accessible on port 8780.

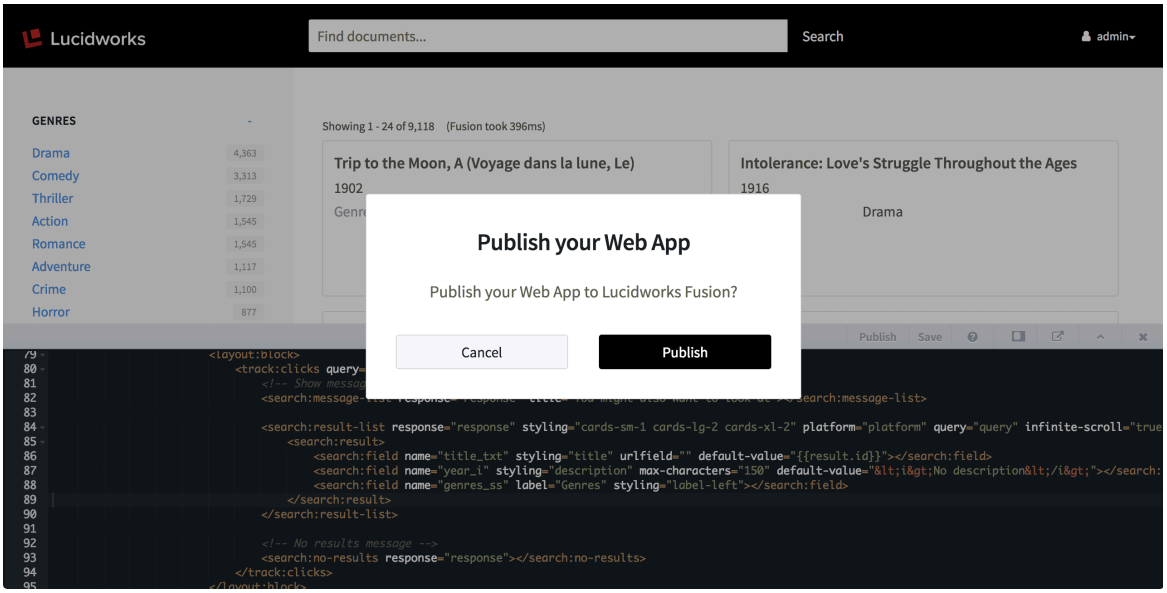
Publishing a search interface to Fusion Server

1. Open your search interface at `http://<app-studio-host>:8080` in your browser.
2. Open the code editor:
   - Press the ESC key

   or

- Click the code editor button:

3. Click **Publish**.
   App Studio prompts you to confirm that you want to publish your webapp to Fusion:



4. Click **Publish** to complete the deployment.

Running a dedicated node for a search interface

When deploying Fusion nodes dedicated to serving one or more search interfaces, only the following Fusion services need to run:

- `agent`
- `api`
- `log-shipper`
- `webapps`

When starting a dedicated node, you can minimize the number of running services by starting it like this.

Unix and MacOS:

```
cd fusion/latest.x/bin ./agent start ./api start ./webapps start
```

Windows:

```
cd fusion\latest.x\bin agent.cmd start api.cms start log-shipper.cmd
start webapps.cmd start
```

Deploy a Search App in Fusion's Webapps Service (4.1)

How-to

> Compatible versions: 4.2
>
> Works with: *App Studio*

You can use App Studio to deploy a search app in Fusion Server. You develop your search interface, and then publish the search app to Fusion's *Webapps service*.

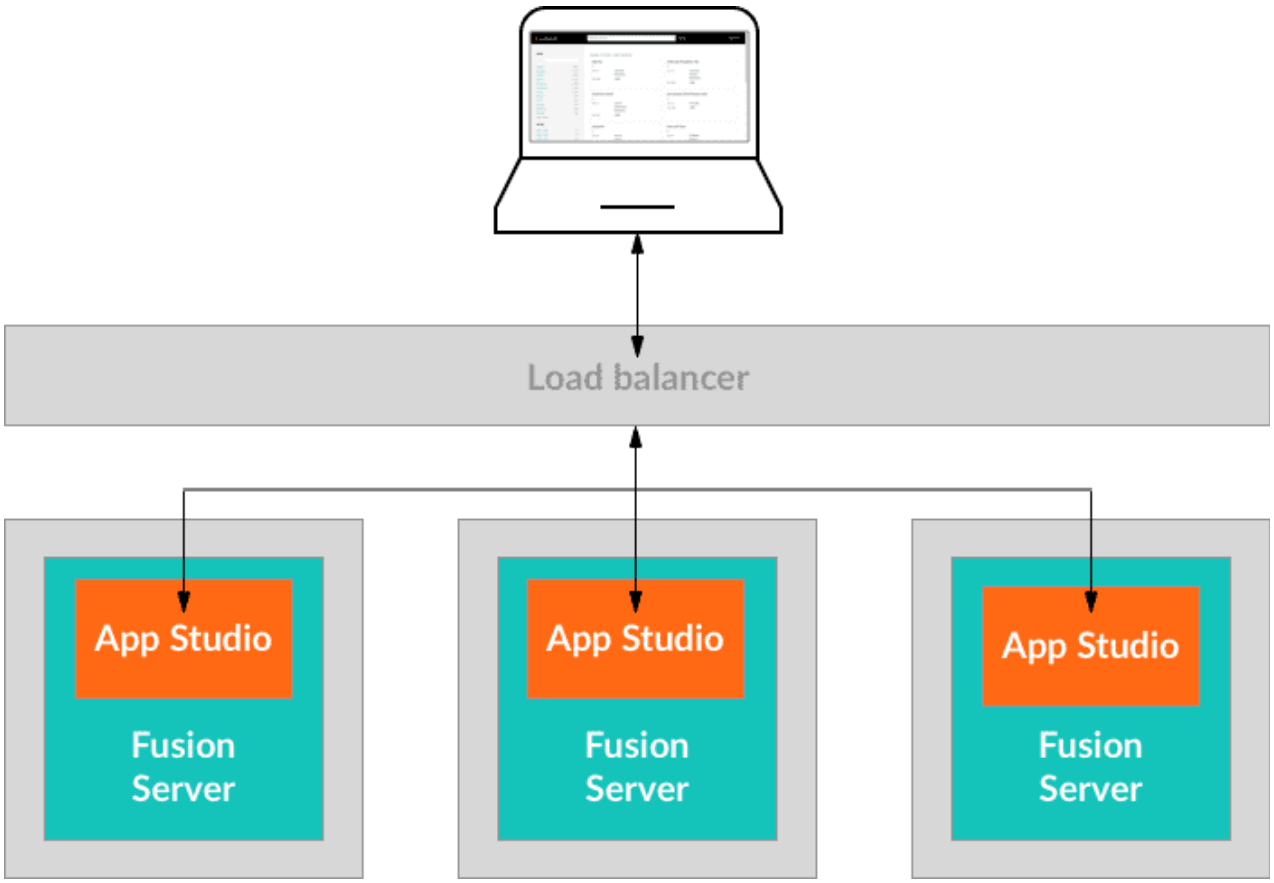
Important

Simplified instructions can be found in *Deploy App Studio*.

Note

This deployment type leverages your existing production environment, requiring no dedicated nodes for App Studio. However, you might need to deploy additional Fusion nodes if you anticipate significant traffic to your search interface.

UI deployment overview



Here is an overview of the steps:

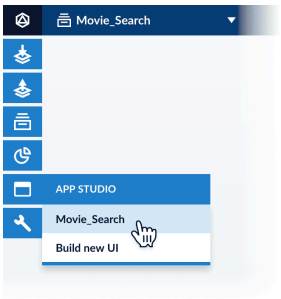
1. *Install App Studio*.
2. *Develop your search interface*.
3. Publish your UI to Fusion Server using the instructions below.

Your webapp then becomes accessible at `http://localhost:8764/webapps/<project-name>`.

Publishing a search interface to Fusion Server

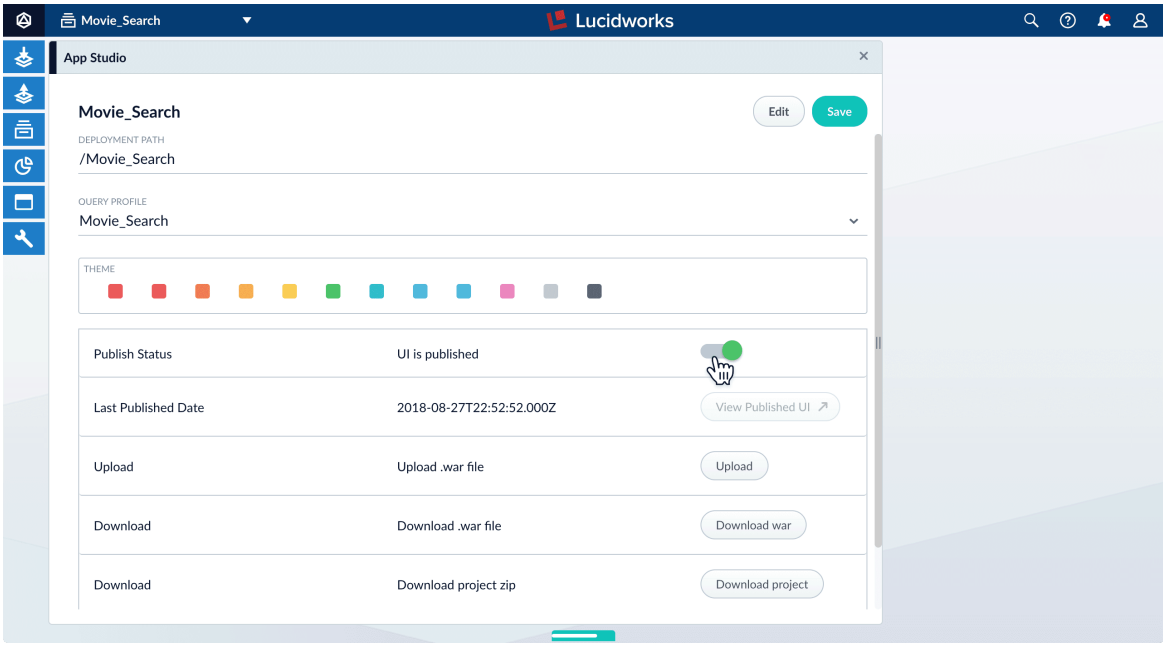
These steps publish your search interface to a Fusion instance (and its cluster) where the App Studio project already exists.

1. In the Fusion workspace, navigate to **App Studio** and select your search interface.



The configuration panel appears.

2. Next to **Publish Status**, click the toggle:



3. Click **Save**.
   The interface is only published after you click **Save**, which also enables the **View Published UI** button.
4. Click **View Published UI** to launch a new window where you can view the interface as an end user.
   Your published interface is available at `http://localhost:8764/webapps/<project-name>`.

Migrating a search interface to other Fusion hosts

If you have published your search interface on one Fusion Host in a cluster, then it is published throughout the cluster.

There are two ways to migrate a search interface to a different Fusion cluster:

Export and import the Fusion app	**Download and upload the WAR (`.war`) file**
This migrates all objects in the Fusion app, such as datasources, query profiles, schedules, and so on, in addition to the search interface. The search interface will be editable on the target hosts.	This migrates only the search interface. The search interface will not be editable on the target hosts. If you are uploading the file into a Fusion app other than the one in which it was created, the new Fusion app must include a query profile and data fields whose names are identical to the ones already configured in the search interface. For instructions on downloading or building a WAR file and uploading it to Fusion, see below.

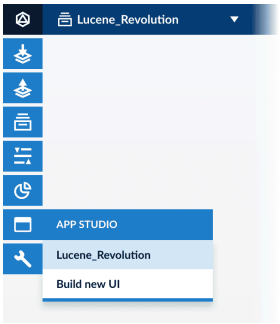
Download or build a WAR file

You can distribute a search interface as a WAR (`.war`) file.

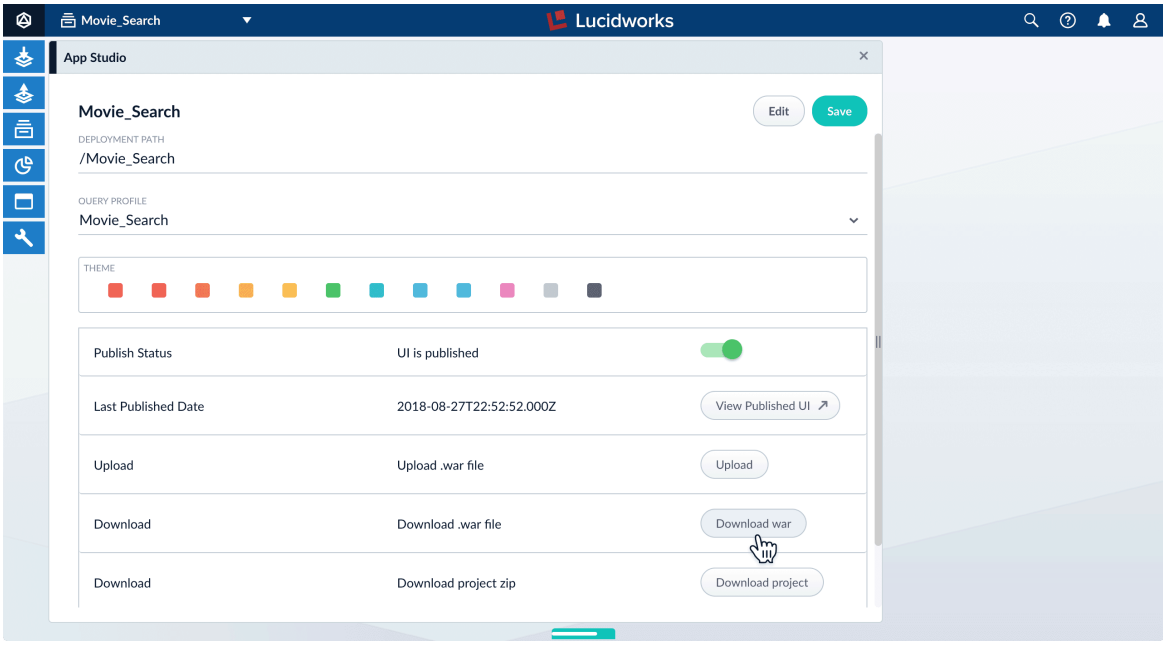
You can download a WAR file from the Fusion UI or create one from a *downloaded project*.

DOWNLOADING A WAR FILE

1. In the Fusion workspace, navigate to **App Studio** and select your search interface.



   The configuration panel appears.
2. Click **Download war**:



BUILDING A WAR FILE

A downloaded project comes with the scripts `app-studio` (for Unix) and `app-studio.bat` (for Windows) that create WAR files.

How to build a WAR file

Unix and MacOS:

1. In a shell window, switch to your project directory:

```
cd /path/to/project-directory
```

2. Create application files:

```
./app-studio package
```

Windows:

1. In a terminal window, switch to your project directory:

```
cd \path\to\project-directory
```

2. Create application files:

```
app-studio.bat package
```

Application files created:

The script creates the following files in the `dist` directory of your project:

- `search-app-latest.x.jar`
  This is a *executable Java JAR file* file.
- `search-app-project.zip`
  Use this file to share your project with other search interface developers.
- `search-app.war`
  You can upload this file to Fusion to deploy it. See the *Deployment*.

Upload the WAR file

You can upload the WAR file through the Fusion UI or the REST API.

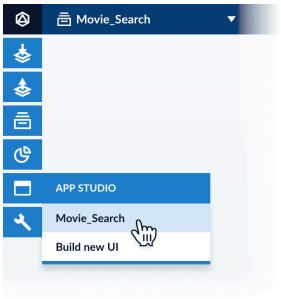
UPLOADING WITH THE FUSION UI

You can upload your project (as a WAR file) in the App Studio configuration panel.

Note

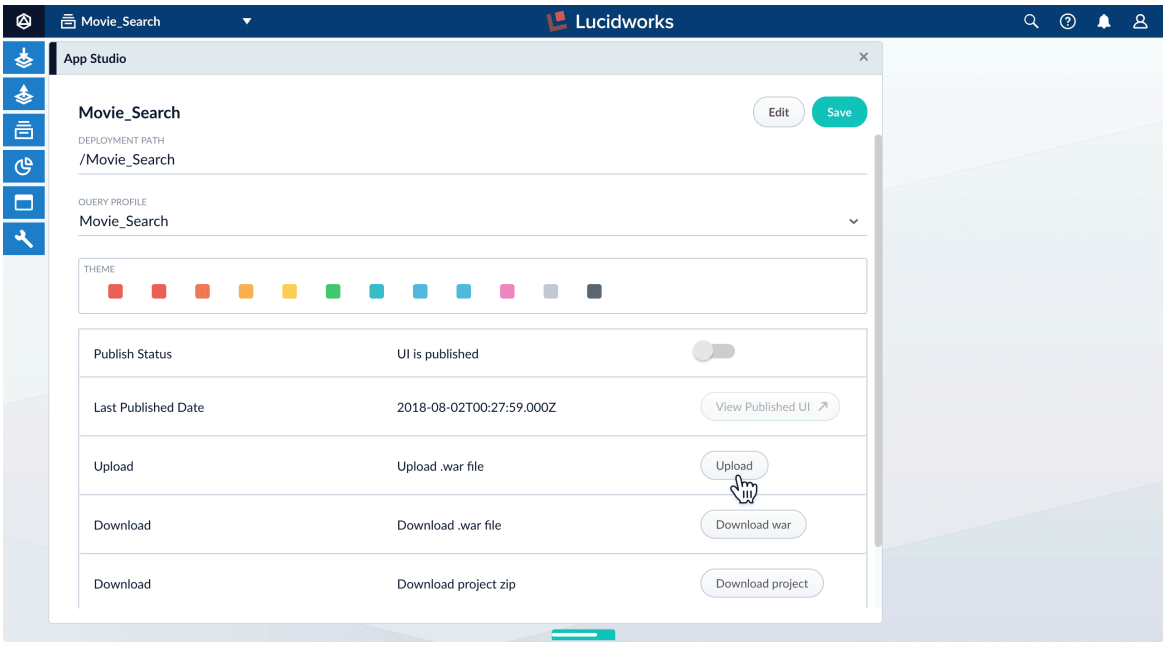
If you upload the file into a Fusion app other than the one in which it was created, the new Fusion app must include a query profile and data fields whose names are identical to the ones already configured in the search interface.

1. From the App Studio menu, select your project:



The project configuration panel appears.

2. Click **Upload**.



Fusion prompts you to confirm that you want to upload a project that will delete the existing one.

3. Click **Yes, continue**.
4. Navigate to your WAR file and select it.
   After the upload is complete, you can edit or publish the uploaded project.

UPLOADING WITH THE REST API

Use the `webapps/{id}/war` *endpoint* of the Webapps API to upload a WAR file, as in this example:

```
curl -u USERNAME:PASSWORD -X PUT
http://localhost:8764/api/webapps/Movie_Search/war -F
'file=@movies.war'
```

Running a dedicated node for a search interface

When deploying Fusion nodes dedicated to serving one or more search interfaces, only the following Fusion services need to run:

- `agent`
- `api`
- `log-shipper`
- `webapps`

When starting a dedicated node, you can minimize the number of running services by starting it like this.

Unix and MacOS:

```
cd fusion/latest.x/bin ./agent start ./api start ./log-shipper start
./webapps start
```

Windows:

```
cd fusion\latest.x\bin agent.cmd start api.cmd start log-shipper.cmd
start webapps.cmd start
```

Deploy a Search App in Fusion's Webapps Service (4.2)

How-to

> Compatible versions: 4.2
>
> Works with: *App Studio*

You can use App Studio to deploy a search app in Fusion Server. You develop your search interface, and then publish the search app to Fusion's *Webapps service*.

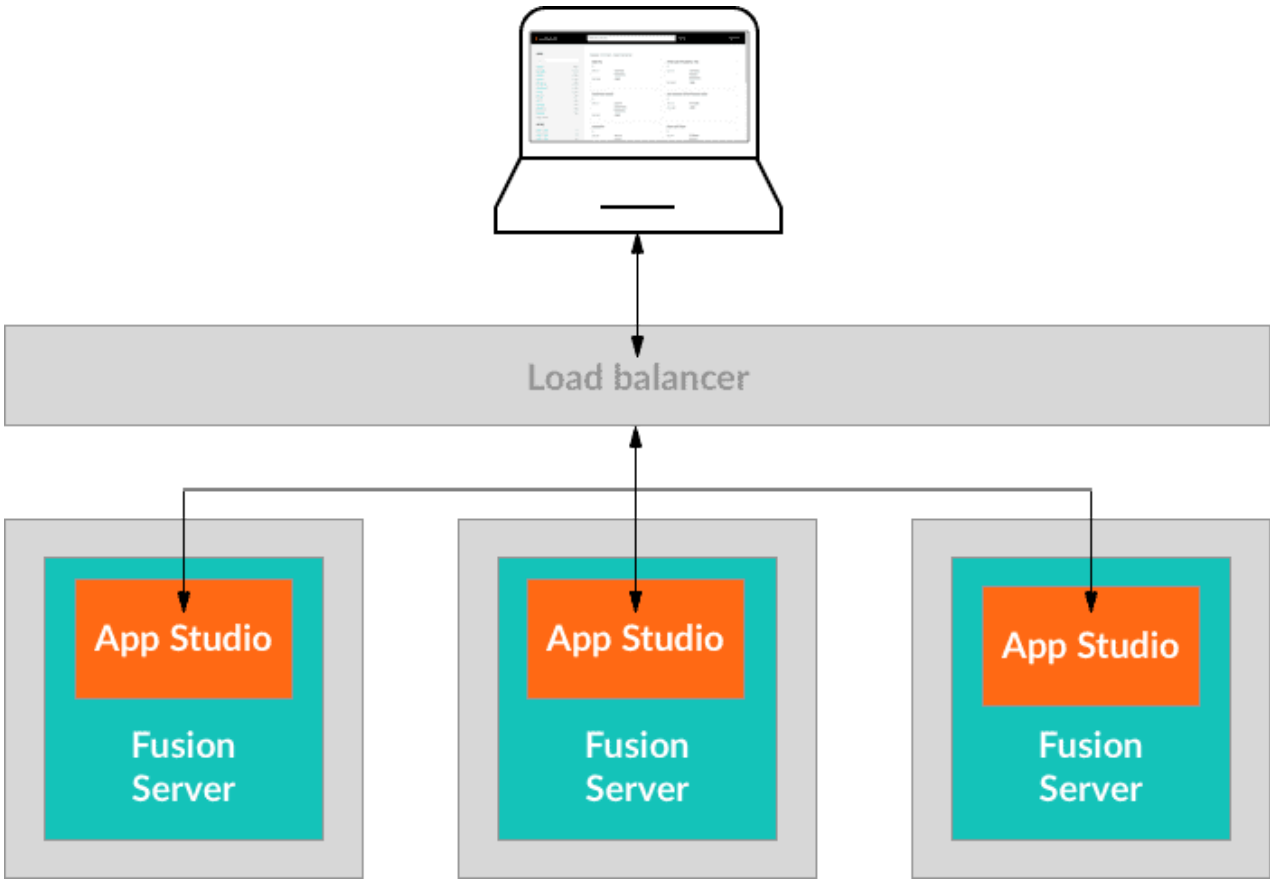
Important

Simplified instructions can be found in *Deploy App Studio*.

Note

This deployment type leverages your existing production environment, requiring no dedicated nodes for App Studio. However, you might need to deploy additional Fusion nodes if you anticipate significant traffic to your search interface.

UI deployment overview



Here is an overview of the steps:

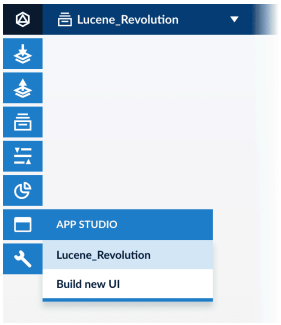
1. *Develop your search interface.*
2. Publish your UI to Fusion Server using the instructions below.

Your webapp then becomes accessible at `http://localhost:8764/webapps/<project-name>`.

Publishing a search interface to Fusion Server

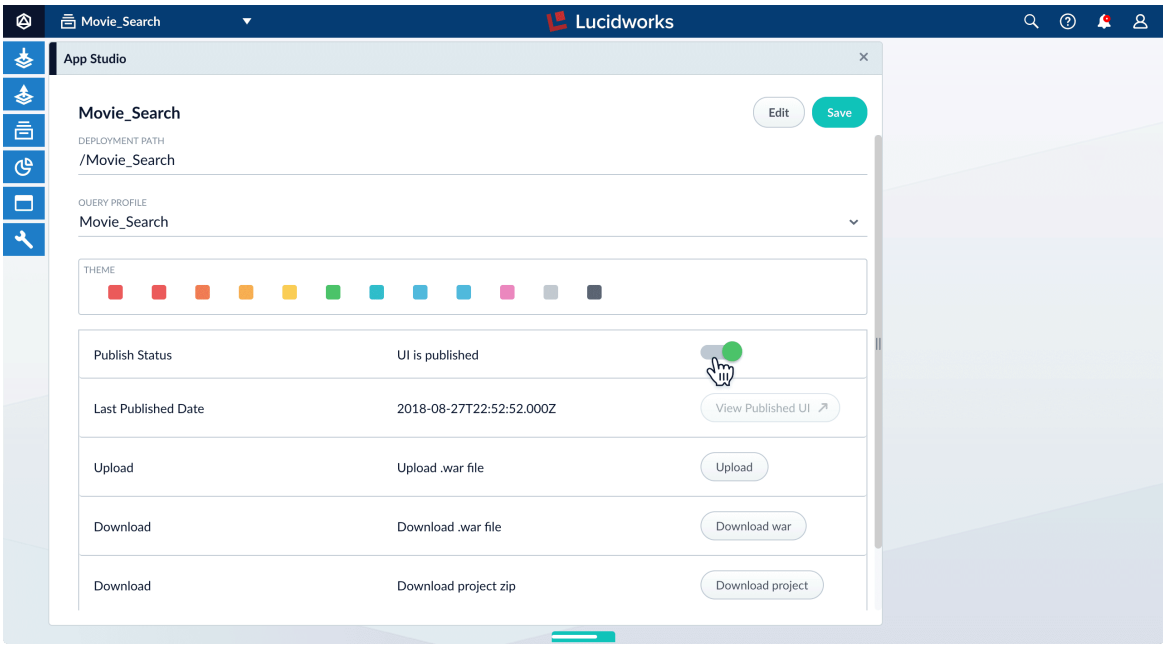
These steps publish your search interface to a Fusion instance (and its cluster) where the App Studio project already exists.

1. In the Fusion workspace, navigate to **App Studio** and select your search interface.



The configuration panel appears.

2. Next to **Publish Status**, click the toggle:



3. Click **Save**.
   The interface is only published after you click **Save**, which also enables the **View Published UI** button.
4. Click **View Published UI** to launch a new window where you can view the interface as an end user. Your published interface is available at `http://localhost:8764/webapps/<project-name>#/search`.

Migrating a search interface to other Fusion hosts

If you have published your search interface on one Fusion Host in a cluster, then it is published throughout the cluster.

There are two ways to migrate a search interface to a different Fusion cluster:

Export and import the Fusion app	Download and upload the WAR (`.war`) file
This migrates all objects in the Fusion app, such as datasources, query profiles, schedules, and so on, in addition to the search interface. The search interface will be editable on the target hosts.	This migrates only the search interface. The search interface will not be editable on the target hosts. If you are uploading the file into a Fusion app other than the one in which it was created, the new Fusion app must include a query profile and data fields whose names are identical to the ones already configured in the search interface. For instructions on downloading or building a WAR file and uploading it to Fusion, see below.

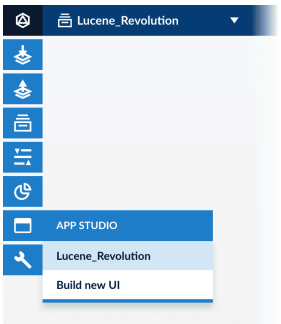
Download or build a WAR file

You can distribute a search interface as a WAR (`.war`) file.

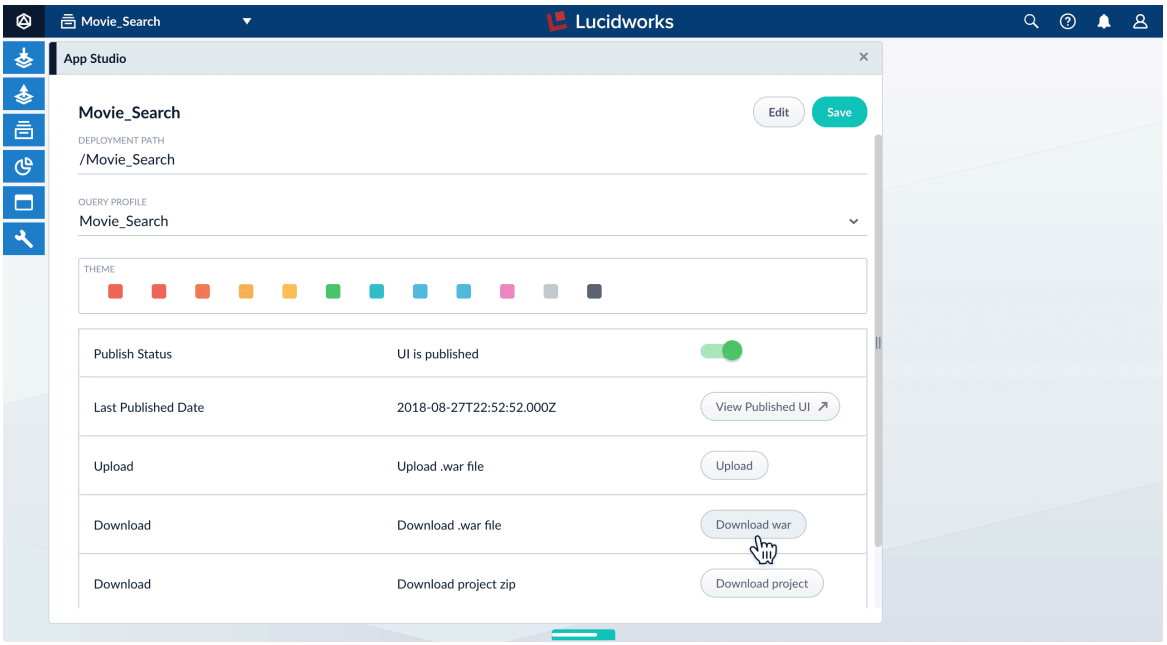
You can download a WAR file from the Fusion UI or create one from a *downloaded project*.

DOWNLOADING A WAR FILE

1. In the Fusion workspace, navigate to **App Studio** and select your search interface.



   The configuration panel appears.
2. Click **Download war**:



BUILDING A WAR FILE

A downloaded project comes with the scripts **`app-studio`** (for Unix) and **`app-studio.bat`** (for Windows) that create WAR files.

How to build a WAR file

Unix and MacOS:

1. In a shell window, switch to your project directory:

```
cd /path/to/project-directory
```

2. Create application files:

```
./app-studio package
```

Windows:

1. In a terminal window, switch to your project directory:

```
cd \path\to\project-directory
```

2. Create application files:

```
app-studio.bat package
```

Application files created:

The script creates the following files in the **`dist`** directory of your project:

- **`search-app-latest.x.jar`**
  This is a *executable Java JAR file* file.
- **`search-app-project.zip`**
  Use this file to share your project with other search interface developers.
- **`search-app.war`**
  You can upload this file to Fusion to deploy it. See the *Deployment*.

Upload the WAR file

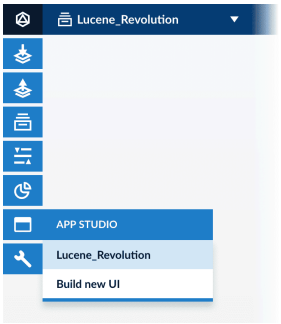
You can upload the WAR file through the Fusion UI or the REST API.

UPLOADING WITH THE FUSION UI

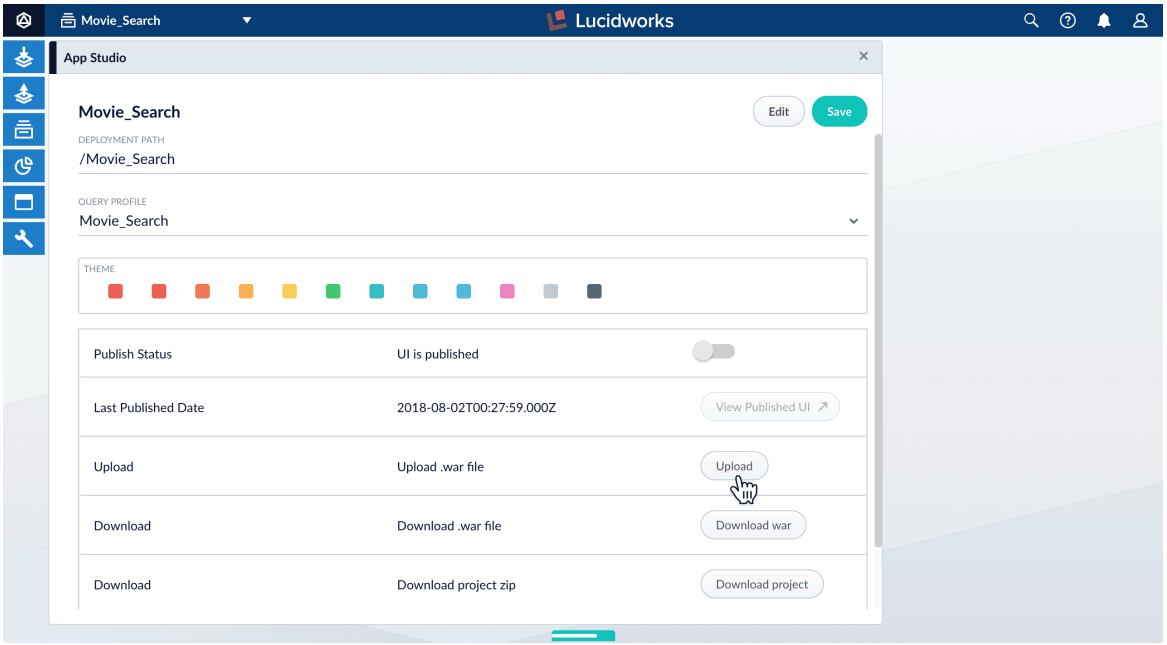
You can upload your project (as a WAR file) in the App Studio configuration panel.

> **Note**
>
> If you upload the file into a Fusion app other than the one in which it was created, the new Fusion app must include a query profile and data fields whose names are identical to the ones already configured in the search interface.

1. From the App Studio menu, select your project:



   The project configuration panel appears.
2. Click **Upload**.



   Fusion prompts you to confirm that you want to upload a project that will delete the existing one.
3. Click **Yes, continue**.
4. Navigate to your WAR file and select it.
   After the upload is complete, you can edit or publish the uploaded project.

UPLOADING WITH THE REST API

Use the `webapps/{id}/war` *endpoint* of the Webapps API to upload a WAR file, as in this example:

```
curl -u USERNAME:PASSWORD -X PUT
http://localhost:8764/api/webapps/Movie_Search/war -F
'file=@movies.war'
```

Running a dedicated node for a search interface

When deploying Fusion nodes dedicated to serving one or more search interfaces, only the following Fusion services need to run:

- `agent`
- `api`
- `log-shipper`
- `webapps`

When starting a dedicated node, you can minimize the number of running services by starting it like this.

Unix and MacOS:

```
cd fusion/latest.x/bin ./agent start ./api start ./log-shipper start
./webapps start
```

Windows:

```
cd fusion\latest.x\bin agent.cmd start api.cmd start log-shipper.cmd
start webapps.cmd start
```

Deploy a Standalone App

How-to

> Compatible versions: 4.2
>
> Works with: *App Studio*

There are two ways to deploy a standalone App Studio search interface:

- *Create an app with an embedded Tomcat web server*
  This deployment type has an embedded Tomcat web server and is ready to deploy in a production environment. Previously, this was referred to as a "self-contained deployment."
- *Create an executable Java JAR file*
  This is a JAR package. Previously, this was referred to as a "self-executing deployment."

Embedded Tomcat web server

Package your app as a standalone app for deployment to a production server. The standalone app embeds an Apache Tomcat servlet container.

Unix or MacOS:

```
./app-studio dist
```

Windows:

```
app-studio.bat dist
```

Standalone app directory

Unix or MacOS:

The standalone app for deployment is in the `app-name/search-app-standalone` directory.

```
search-app-standalone/ README.md app/ build/ config/ keystore.jks lib
search-app-start.bat search-app-stop.bat search-app-stop.ps1 search-
app.sh
```

Windows:

The standalone app for deployment is in the `app-name\search-app-standalone` directory.

```
search-app-standalone\ README.md app\ build\ config\ keystore.jks lib
search-app-start.bat search-app-stop.bat search-app-stop.ps1 search-
app.sh
```

The `search-app-project.zip` is a snapshot of your project at the time you compiled the application.

Executable Java JAR file

In this deployment type, your search interface runs within the Java runtime environment (JRE).

A *downloaded project* comes with an `app-studio` script that creates a JAR file.

How to create the JAR file

1. In a shell window, switch to your project directory:

```
cd /path/to/project-directory
```

2. Create the JAR file:
   Unix and MacOS:

```
./app-studio package
```

   Windows:

```
app-studio.bat package
```

The script creates the executable JAR file in the `dist` directory of your project:

```
search-app-latest.x.jar
```

Running an executable JAR file

```
java -jar search-app-latest.x.jar
```

Deploy an App in a Servlet Container

How-to

> Compatible versions: 4.2
>
> Works with: *App Studio*

The most common way to deploy an App Studio search application is to package it as a standard Java `.war` file, which can be deployed in any standard servlet container such as Tomcat or Jetty.

Create a WAR file

Create a WAR file suitable for deployment in a servlet container. Do either of the following:

- ***Use a script*** - Run the script `app-studio` (on Unix or MacOS) or `app-studio.bat` (on Windows) with the parameter `package` to create the WAR file.
- ***Download the WAR file from Fusion*** - If you created the search interface in Fusion, you can download the WAR file from Fusion.

Deploy the WAR file

Deploy the WAR file in the servlet container. Follow the instructions in the documentation for the servlet container.

Deploy App Studio

How-to

> Compatible versions: 4.2, 5.5, 5.9, 5.10
>
> Works with: *Fusion Server*, *App Studio*, *Fusion*

This article describes how to deploy *App Studio in Fusion 5* or *App Studio in Fusion 4*.

Important

Starting in Fusion 5.11.0, Fusion has deprecated the Webapps service. As of Fusion 5.11.0, you should instead *deploy App Studio manually*.

Tip

In Fusion 4.x, the default `FUSION_PORT` value is `8764`. In Fusion 5.x, the default value is `6764`.

1. Create UI (create a new App Studio project)

```
curl -v -u USERNAME:PASSWORD -H 'Content-type: application/json'
-X POST
'https://FUSION_HOST:FUSION_PORT/api/apps/APP_NAME/webapps' -d
'{"id": "UI_ID","name": "UI_NAME","contextPath":
"/UI_CONTEXT"}'
```

2. Execute the `command ./app-studio package` to generate the `app-studio-enterprise.war` file. The file is created in the following location: `/<$Application>/dist/app-studio-enterprise.war`
3. Deploy WAR (upload to webapps service)

```
curl -v -u USERNAME:PASSWORD -X PUT -H 'Content-type:
application/zip'
'https://FUSION_HOST:FUSION_PORT/api/apps/APP_NAME/webapps/UI_ID
/war' --data-binary @app-studio-enterprise.war
```

4. Go to `https://FUSION_HOST:FUSION_PORT/webapps/UI_CONTEXT/`.

Deploy App Studio Enterprise to a Fusion 5 Cluster (GKE)

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *App Studio*, *Fusion*

This article explains how to deploy *App Studio Enterprise* (ASE) to an existing Fusion 5 cluster in Google Kubernetes Engine (GKE) without using the Webapps service.

Before completing this guide, ensure that your `fusion.conf` file points to the IP or URI and port of the proxy service.

Run the App Studio Enterprise application locally and verify functioning security and search features with the cluster you are deploying to.

Prepare the package

1. Package your app into a single JAR file:

```
./app-studio package
```

2. App Studio Enterprise includes a `dockerfile`. Create the App Studio Enterprise Docker image:

```
docker build PATH -t APP_NAME
```

   Set or replace `APP_NAME` with the name of your application. Replace `PATH` with the path to build from.

3. You can test the Docker image locally with the following command:

```
docker run -it -p LOCAL_PORT:8080 APP_NAME
```

   Set or replace `LOCAL_PORT` with the port on your local machine that can access the app. Replace `APP_NAME` with the ASE application name.

Publish the image

You can publish the Docker image anywhere that supports Docker images. This section explains how to publish a Docker image to Google Kubernetes Engine.

1. Tag your container for the registry:

```
docker tag APP_NAME gcr.io/PROJECT_NAME/APP_NAME
```

2. Push your Docker image to the Google Container Registry:

```
docker push gcr.io/PROJECT_NAME/APP_NAME
```

3. Verify the image:

```
gcloud compute instances list
```

Deploy the app to a cluster

After publishing your ASE Docker image, deploy the image to a cluster. Your existing Fusion 5 cluster is a good choice.

1. Switch the context to your Fusion 5 cluster:

```
gcloud container clusters get-credentials CLUSTER_NAME
```

   Replace `CLUSTER_NAME` with your existing Fusion 5 cluster's name.
2. Create a deployment in your cluster using the image you published:

```
kubectl create deployment APP_NAME --
image=gcr.io/PROJECT_NAME/APP_NAME:latest
```

Create an Ingress resource

After deploying the app, create an Ingress resource for your ASE instance. If you require more Ingress rules, your Ingress resource will look different. Learn more about Ingress resources .

1. Use the following command to create a minimal Ingress resource:

```
cat <<EOF | kubectl apply -f - apiVersion:
networking.k8s.io/v1beta1 kind: Ingress metadata: name: ase-
ingress spec: backend: serviceName: $AppName servicePort: 80
EOF
```

2. Verify the Ingress resource:

```
kubectl get ingress ase-ingress
```

Deploy Fusion 5 on Amazon Elastic Kubernetes Service (EKS)

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion*

Fusion supports deployment on Amazon Elastic Kubernetes Service (EKS). This topic explains how to deploy a Fusion cluster on EKS using the `setup_f5_eks.sh` script in the `fusion-cloud-native` repository.

In addition, this topic provides information about how to *configure IAM roles for the service account*.

Prerequisites

This section covers prerequisites and background knowledge needed to help you understand the structure of this document and how the Fusion installation process works with Kubernetes.

Release Name and Namespace

Before installing Fusion, you need to choose a Kubernetes namespace to install Fusion into. Think of a K8s namespace as a virtual cluster within a physical cluster. You can install multiple instances of Fusion in the same cluster **in separate namespaces**. However, please do not install more than one Fusion release in the same namespace.

NOTE: All Fusion services must run in the same namespace, i.e. you should not try to split a Fusion cluster across multiple namespaces.

Use a short name for the namespace, containing only letters, digits, or dashes (no dots or underscores). The setup scripts in this repo use the namespace for the Helm release name by default.

Install Helm

Helm is a package manager for Kubernetes that helps you install and manage applications on your Kubernetes cluster. Regardless of which Kubernetes platform you're using, you need to install `helm` as it is required to install Fusion for any K8s platform. On MacOS, you can do:

```
brew install kubernetes-helm
```

If you already have helm installed, make sure you're using the latest version:

```
brew upgrade kubernetes-helm
```

For other OS, please refer to the Helm installation docs: https://helm.sh/docs/using_helm/

The Fusion helm chart requires that helm is greater than version `3.0.0`; check your Helm version by running `helm version --short`.

HELM USER PERMISSIONS

If you require that fusion is installed by a user with minimal permissions, instead of an admin user, then the role and cluster role that will have to be assigned to the user within the namespace that you wish to install fusion in are documented in the `install-roles` directory.

Note

When working with Kubernetes on the command-line, it's useful to create a shell alias for `kubectl`, e.g.:

```
alias k=kubectl
```

To use these role in a cluster, as an admin user first create the namespace that you wish to install fusion into:

```
k create namespace fusion-namespace
```

Apply the `role.yaml` and `cluster-role.yaml` files to that namespace

```
k apply -f cluster-role.yaml k config set-context --current --
namespace=$NAMESPACE k apply -f role.yaml
```

Then bind the rolebinding and clusterolebinding to the install user:

```
k create --namespace fusion-namespace rolebinding fusion-install-
rolebinding --role fusion-installer --user <install_user> k create
clusterrolebinding fusion-install-rolebinding --clusterrole fusion-
installer --user <install_user>
```

You will then be able to run the `helm install` command as the `<install_user>`

Clone fusion-cloud-native from Github

You should clone this repo from github as you'll need to run the scripts on your local workstation:

```
git clone https://github.com/lucidworks/fusion-cloud-native.git
```

You should get into the habit of pulling this repo for the latest changes before performing any maintenance operations on your Fusion cluster to ensure you have the latest updates to the scripts.

```
cd fusion-cloud-native git pull
```

Cloning the github repo is preferred so that you can pull in updates to the scripts, but if you are not a git user, then you can download the project: https://github.com/lucidworks/fusion-cloud-native/archive/master.zip . Once downloaded, extract the zip and cd into the `fusion-cloud-native-master` directory.

The `setup_f5_eks.sh` script provided in this repo is strictly optional. The script is mainly to help those new to Kubernetes and/or Fusion get started quickly. If you're already familiar with K8s, Helm, and EKS, then you use Helm directly to install Fusion into an existing cluster or one you create yourself using the process described *here*.

If you're new to Amazon Web Services (AWS), then please visit the Amazon Web Services Getting Started Center to set up an account.

If you're new to Kubernetes and EKS, then we recommend going through Amazon's EKS Workshop before proceeding with Fusion.

Set up the AWS CLI tools

Before launching an EKS cluster, you need to install and configure `kubectl`, `aws`, `eksctl`, `aws-iam-authenticator` using the links provided below:

Required AWS Command-line Tools:

1. kubectl: Install kubectl
2. aws: Installing the AWS CLI
3. eksctl: Getting Started with eksctl
4. aws-iam-authenticator: AWS IAM Authenticator for Kubernetes

Run `aws configure` to configure a profile for authenticating to AWS. You'll use the profile name you configure in this step, which defaults to `default`, as the `-p` argument to the `setup_f5_eks.sh` script in the next section.

Note

When working in Ubuntu, avoid using the eksctl snap version. Alternative sources can have different versions that could cause command failures. Also, always make sure you are using the latest version for each one of the required tools.

Set up Fusion on EKS

To create a cluster in EKS the following IAM policies are required:

- AmazonEC2FullAccess
- AWSCloudFormationFullAccess

EKS Permissions

eks:DeleteCluster	eks:UpdateClusterVersion	eks:ListUpdates	eks:DescribeUpdate
eks:DescribeCluster	eks:ListClusters	eks:CreateCluster	

VPC Permissions

ec2:DeleteSubnet	ec2:DeleteVpcEndpoints	ec2:CreateVpc	ec2:AttachInternetGateway
ec2:DetachInternetGateway	ec2:DisassociateSubnetCidrBlock	ec2:DescribeVpcAttribute	ec2:AssociateVpcCidrBlock
ec2:ModifySubnetAttribute	ec2:DisassociateVpcCidrBlock	ec2:CreateVpcEndpoint	ec2:DescribeVpcs
ec2:CreateInternetGateway	ec2:AssociateSubnetCidrBlock	ec2:ModifyVpcAttribute	ec2:DeleteInternetGateway
ec2:DeleteVpc	ec2:CreateSubnet	ec2:DescribeSubnets	ec2:ModifyVpcEndpoint

IAM Permissions

iam:CreateInstanceProfile	iam:DeleteInstanceProfile	iam:GetRole	iam:GetPolicyVersion
iam:UntagRole	iam:GetInstanceProfile	iam:GetPolicy	iam:TagRole
iam:RemoveRoleFromInstanceProfile	iam:DeletePolicy	iam:CreateRole	iam:DeleteRole
iam:AttachRolePolicy	iam:PutRolePolicy	iam:ListInstanceProfiles	iam:AddRoleToInstanceProfile
iam:CreatePolicy	iam:ListInstanceProfilesForRole	iam:PassRole	iam:DetachRolePolicy
iam:DeleteRolePolicy	iam:CreatePolicyVersion	iam:GetRolePolicy	iam:DeletePolicyVersion

Download and run the `setup_f5_eks.sh` script to install Fusion 5.x in an EKS cluster.

Note

This script does not support multiple node pools and should not be used for production clusters.

- To create a new cluster and install Fusion, run the following command:

```
./setup_f5_eks.sh -c my-eks-cluster -p profile-name -n fusion-namespace --create demo
```

  - Replace **`my-eks-cluster`**, **`profile-name`**, and **`fusion-namespace`** with your cluster, profile, and namespace values.
  - Pass the **`--create`** option with either **`demo`** or **`multi_az`**.

- To use an existing cluster and install Fusion, run the following command:

```
./setup_f5_eks.sh -c cluster-name -p profile-name
```

  - Replace `cluster-name` with the name of the cluster you already created.
  - Replace `profile-name` with the name of your profile.

The profile is automatically set to `default` if you ran the AWS configure command without giving the profile a name.

Use the `--help` option to see full script usage.

Warning

If using Helm V2, the `setup_f5_eks.sh` script installs Helm's `tiller` component into your EKS cluster with the cluster admin role. If you don't want this, then please upgrade to Helm v3.

Warning

The `setup_f5_eks.sh` script creates a service account that provides S3 read-only permissions to the created pods.

After running the `setup_f5_eks.sh` script, proceed to the *Verifying the Fusion Installation* section below.

EKS cluster overview

The EKS cluster is created using `eksctl` (https://eksctl.io/). By default it will setup the following resources in your AWS account:

- A dedicated VPC for the EKS cluster in the specified region with CIDR: `192.168.0.0/16`
- 3 Public and 3 Private subnets within the created VPC, each with a `/19` CIDR range, along with the corresponding route tables.
- A NAT gateway in each Public subnet
- An Auto Scaling Group of the instance type specified by the script, which defaults to `m5.2xlarge`, with 3 instances spanning the public subnets.

See https://eksctl.io/usage/vpc-networking/ for more information on the networking setup.

EKS Ingress

The `setup_f5_eks.sh` script exposes the Fusion proxy service on an external DNS name provided by an ELB over HTTP. This is done for demo or getting started purposes. However, you're strongly encouraged to configure a K8s Ingress with TLS termination in front of the proxy service. See: https://aws.amazon.com/premiumsupport/knowledge-center/terminate-https-traffic-eks-acm/

Our EKS script creates a classic ELB for exposing fusion proxy service. In case you need to change this behavior and use AWS Load Balancer Controller instead you can use the following parameters when running the `setup_f5_eks.sh` script:

```
--deploy-alb # Tells the script to deploy an ALB
```

By default the `kube-system` namespace is being used for installing the `aws-load-balancer-controller` because pods `priorityClassName` is set to `system-cluster-critical`.

In case you need to deploy an internal ALB you can use the `--internal-alb` option. This will create the nodes in the internal subnets. Fusion will be reachable from an AWS instance located in any of the external subnets on the same VPC. To use an ALB also an ingress with a DNS name is required, you can use the `-h` option to create an ingress with the required DNS name.

Finally, use Route 53 or your DNS provider for creating an A ALIAS DNS record for your DNS name pointing to the ingress ADRESS. You can get the address listing the ingress using the command `kubectl get ing`.

Provide access to the EKS cluster to other users

Initially, only the user that created the Amazon EKS cluster has `system:masters` permissions to configure the cluster. In order to extend the permissions, a `ConfigMap` should be created to allow access to IAM users or roles.

For providing these permissions, use the following yaml file as a template, replacing the required values:

aws-auth.yaml

```
apiVersion: v1 kind: ConfigMap metadata: name: aws-auth namespace:
kube-system data: mapRoles: | - rolearn: <node_instance_role_arn>
username: system:node:{{EC2PrivateDNSName}} groups: -
system:bootstrappers - system:nodes mapUsers: | - userarn:
arn:aws:iam::<account_id>:user/<username> username: <username> groups:
- system:masters
```

Use the following command for applying the yaml file: `kubectl apply -f aws-auth.yaml`

Remove EKS cluster

In case you have deployed an ALB ingress controller, you would need to remove the policy that was created for managing the ALB before removing the cluster. You can use the following command for it:

```
aws iam --profile <profile-name> delete-policy --policy-arn
arn:aws:iam::<account_id>:policy/eksctl-<cluster-name>-alb-policy
```

Also you can remove it manually using the AWS IAM console, searching for `eksctl-<cluster-name>-alb-policy`.

After that you should remove the ALB with `helm delete`, list the current releases with `helm list`.

The EKS cluster is created using Cloudformation stacks so you need to remove them to delete the cluster, you can check them in the AWS Cloudformation Console, check for the following stacks:

- eksctl-<cluster-name>-nodegroup-standard-workers
- eksctl-<cluster-name>-cluster

The `eksctl-<cluster-name>-nodegroup-standard-workers` stack should be the first to be removed. After that we can remove the `eksctl-<cluster-name>-cluster` stack.

Also you can use the following commands>:

```
aws cloudformation --profile <profile-name> delete-stack --stack-name
eksctl-<cluster-name>-nodegroup-standard-workers aws cloudformation --
profile <profile-name> delete-stack --stack-name eksctl-<cluster-
name>-cluster
```

Verifying the Fusion Installation

In this section, we provide some tips on how to verify the Fusion installation.

Tip

Check if the Fusion Admin UI is available at `https://<fusion-host>:6764/admin/`.

Let's review some useful kubectl commands.

Enhance the K8s Command-line Experience

Here is a list of tools we found useful for improving your command-line experience with Kubernetes:

- krew (kubectl plugin mgr): https://github.com/kubernetes-sigs/krew/
- kube-ps1 (show current context on command line prompt): https://github.com/jonmosco/kube-ps1
- kubectx / kubens (switch between clusters / namespaces): https://github.com/ahmetb/kubectx

Useful kubectl commands

kubectl reference: https://kubernetes.io/docs/reference/generated/kubectl/kubectl-commands

Set the namespace for `kubectl` if not using the default:

```
kubectl config set-context --current --namespace=<NAMESPACE>
```

This saves you from having to pass `-n` *with every command.*

Get a list of running pods: `k get pods`

Get logs for a pod using a label: `k logs -l app.kubernetes.io/component=query-pipeline`

Get pod deployment spec and details: `k get pods <pod_id> -o yaml`

Get details about a pod events: `k describe po <pod_id>`

Port forward to a specific pod: `k port-forward <pod_id> 8983:8983`

SSH into a pod: `k exec -it <pod_id> -- /bin/bash`

CPU/Memory usage report for pods: `k top pods`

Forcefully kill a pod: `k delete po <pod_id> --force --grace-period 0`

Scale up (or down) a deployment: `k scale deployment.v1.apps/<id> --replicas=N`

Get a list of pod versions: `k get po -o jsonpath='{..image}' | tr -s '.' '\n' | sort | uniq`

Check Fusion Pods and Services

Once the install script completes, you can check that all pods and services are available using:

```
kubectl get pods
```

If all goes well, you should see a list of pods similar to:

```
NAME READY STATUS RESTARTS AGE seldon-controller-manager-6675874894-
qxwrv 1/1 Running 0 8m45s f5-admin-ui-74d794f4f8-m5jms 1/1 Running 0
8m45s f5-ambassador-fd6b9b5dc-7ghf6 1/1 Running 0 8m43s f5-api-
gateway-6b9998b9c-tmchk 1/1 Running 0 8m45s f5-auth-ui-7565564b4c-
rdc74 1/1 Running 0 8m42s f5-classic-rest-service-0 1/1 Running 3
8m44s f5-devops-ui-77bb867ffb-fbzxd 1/1 Running 0 8m42s f5-fusion-
admin-78b8f8fc7f-4d7l8 1/1 Running 0 8m42s f5-fusion-indexing-
599c8d448-xzsvm 1/1 Running 0 8m44s f5-insights-665fd9f6fc-g5psw 1/1
Running 0 8m43s f5-job-launcher-84dd4c5c96-p8528 1/1 Running 0 8m44s
f5-job-rest-server-6d44d964b8-xtnxw 1/1 Running 0 8m45s f5-logstash-0
1/1 Running 0 8m45s f5-ml-model-service-6987dc94c9-9ppp8 2/2 Running 1
8m45s f5-monitoring-grafana-5d499dbb58-pzw72 1/1 Running 0 10m f5-
monitoring-prometheus-kube-state-metrics-54d6678dv9h7h 1/1 Running 0
10m f5-monitoring-prometheus-pushgateway-7d65c65b85-vwrwf 1/1 Running
0 10m f5-monitoring-prometheus-server-0 2/2 Running 0 10m f5-pm-ui-
86cbc5bb65-nd2n8 1/1 Running 0 8m44s f5-pulsar-bookkeeper-0 1/1
Running 0 8m45s f5-pulsar-broker-b56cc776f-56msx 1/1 Running 0 8m45s
f5-query-pipeline-5d75d7d5f4-l2mdf 1/1 Running 0 8m43s f5-connectors-
7bb6cfc65f-7wfs2 1/1 Running 0 8m42s f5-connectors-backend-987fdc648-
dldwv 1/1 Running 0 8m45s f5-rules-ui-6b9d55b78f-9hzzj 1/1 Running 0
8m43s f5-solr-0 1/1 Running 0 8m44s f5-solr-exporter-c4687c785-jsm7x
1/1 Running 0 8m45s f5-ui-6cdbcc68c6-rj9cq 1/1 Running 0 8m45s f5-
webapps-6d6bb9bfd-hm4qx 1/1 Running 0 8m45s f5-workflow-controller-
7b66679fb7-sjbvp 1/1 Running 0 8m44s f5-zookeeper-0 1/1 Running 0
8m45s
```

The number of pods per deployment / statefulset will vary based on your cluster size and replicaCount settings in your custom values YAML file. Also, don't worry if you see some pods having been restarted as that just means they were too slow to come up and Kubernetes killed and restarted them. You do want to see at least one pod running for every service. If a pod is not running after waiting a sufficient amount of time, use `kubectl logs <pod_id>` to see the logs for that pod; to see the logs for previous versions of a pod, use: `kubectl logs <pod_id> -p`. You can also look at the actions Kubernetes performed on the pod using `kubectl describe po <pod_id>`.

To see a list of Fusion services, do:

```
kubectl get svc
```

For an overview of the various Fusion 5 microservices, see: Fusion microservices .

Once you're ready to build a Fusion cluster for production, please see see more information at Fusion 5 Survival Guide .

Upgrading with Zero Downtime

One of the most powerful features provided by Kubernetes and a cloud-native microservices architecture is the ability to do a rolling update on a live cluster. Fusion 5 allows customers to upgrade from Fusion 5.x.y to a later 5.x.z version on a live cluster with zero downtime or disruption of service.

When Kubernetes performs a rolling update to an individual microservice, there will be a mix of old and new services in the cluster concurrently (only briefly in most cases) and requests from other services will be routed to both versions. Consequently, Lucidworks ensures all changes we make to our service do not break the API interface exposed to other services in the same 5.x line of releases. We also ensure stored configuration remains compatible in the same 5.x release line.

Lucidworks releases minor updates to individual services frequently, so our customers can pull in those upgrades using Helm at their discretion.

To upgrade your cluster at any time, use the `--upgrade` option with our setup scripts in this repo.

The scripts in this repo automatically pull in the latest chart updates from our Helm repository and deploy any updates needed by doing a diff of your current installation and the latest release from Lucidworks. To see what would be upgraded, you can pass the `--dry-run` option to the script.

Grafana Dashboards

Get the initial Grafana password from a K8s secret by doing:

```
kubectl get secret --namespace "${NAMESPACE}" ${RELEASE}-monitoring-
grafana \ -o jsonpath="{.data.admin-password}" | base64 --decode ;
echo
```

With Grafana, you can either setup a temporary port-forward to a Grafana pod or expose Grafana on an external IP using a K8s LoadBalancer. To define a LoadBalancer, do (replace ${RELEASE} with your Helm release label):

```
kubectl expose deployment ${RELEASE}-monitoring-grafana --
type=LoadBalancer --name=grafana --port=3000 --target-port=3000
```

You can use `kubectl get services --namespace <namespace>` to determine when the load balancer is setup and its IP address. Direct your browser to http://<GrafanaIP>:3000 and enter the username `admin@localhost` and the password that was returned in the previous step.

This will log you into the application. It is recommended that you create another administrative user with a more desirable password.

The dashboards and datasoure will be setup for you in grafana, simply navigate to **Dashboards** → **Manage** to view the vailable dashboards

Configure IAM roles for the service account

Configuring IAM roles lets you utilize the Amazon Web Services Security Token Service (AWS STS) for short-term authentication credentials to access services like the Amazon S3 simple storage service.

To configure IAM roles, your user account must be granted **admin** permissions or **IAM:FullAccess**. Complete the following steps:

1. To create the OpenID Connect (OIDC) provider, run the following command:

```
eksctl utils associate-iam-oidc-provider --cluster cluster_name
--approve
```

2. To create an IAM role for the service account associated with the plugin-pod, run the following command:

```
eksctl create iamserviceaccount --name f5-connector-plugin --
namespace default --cluster cluster_name --attach-policy-arn
arn:aws:iam::aws:policy/AmazonS3ReadOnlyAccess --approve --
override-existing-serviceaccounts
```

This command:

- Creates an IAM role and attaches the target policy.
- Updates the existing Kubernetes **f5-connector-plugin** service account and annotates it with the IAM role.
- Uses the existing **policy/AmazonS3ReadOnlyAccess** policy.

If the IAM role was already created without the command, and you want to associate the service account, run the following command:

```
kubectl annotate serviceaccount -n default f5-connector-plugin
eks.amazonaws.com/role-arn=arn:aws:iam::411271863668:role/FUS_ROLE --
overwrite=true
```

Note

To utilize this feature in Fusion 5.3 and later, create a data source with the settings in **S3 Authentication Settings > AWS Instance Credentials Authentication Settings**. For detailed installation information, *AWS S3 V2 connector*.

For more information, see:

- Welcome to the AWS Security Token Service API Reference
- Temporary security credentials in IAM
- IAM roles for service accounts
- What is Amazon EKS?
- Fine-grained IAM roles for service accounts

More deployment options

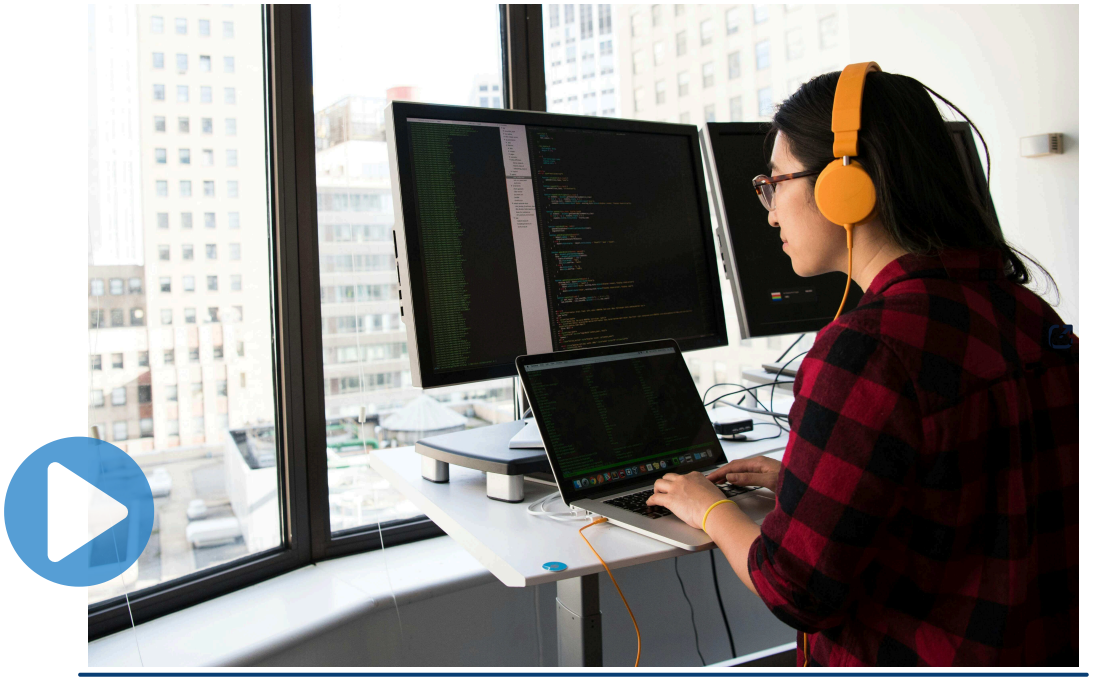
- *How to deploy Fusion 5 in Google Kubernetes Engine*
- *How to deploy Fusion 5 in Azure Kubernetes Service*
- *How to deploy Fusion 5 on other Kubernetes platforms*

Additional resources

LucidAcademy

Lucidworks offers free training to help you get started.

The **Course** for **Deploying Fusion 5** focuses on the prerequisite software needed to deploy Fusion, the necessary setup steps, and the physical act of deployment:



Visit the LucidAcademy to see the full training catalog.

Deploy Fusion 5 on Azure Kubernetes Service (AKS)

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion*

Fusion supports deployment on Azure Kubernetes Service (AKS). This topic explains how to deploy a Fusion cluster on AKS using the `setup_f5_aks.sh` script in the `fusion-cloud-native` repository.

Important

The **`setup_f5_aks.sh`** script is the basic foundation for getting started and proof-of-concept purposes. For information about custom values in a production-ready environment, see *Custom values YAML file*.

Prerequisites

This section covers prerequisites and background knowledge needed to help you understand the structure of this document and how the Fusion installation process works with Kubernetes.

Release Name and Namespace

Before installing Fusion, you need to choose a Kubernetes namespace to install Fusion into. Think of a K8s namespace as a virtual cluster within a physical cluster. You can install multiple instances of Fusion in the same cluster **in separate namespaces**. However, please do not install more than one Fusion release in the same namespace.

NOTE: All Fusion services must run in the same namespace, i.e. you should not try to split a Fusion cluster across multiple namespaces.

Use a short name for the namespace, containing only letters, digits, or dashes (no dots or underscores). The setup scripts in this repo use the namespace for the Helm release name by default.

Install Helm

Helm is a package manager for Kubernetes that helps you install and manage applications on your Kubernetes cluster. Regardless of which Kubernetes platform you're using, you need to install **`helm`** as it is required to install Fusion for any K8s platform. On MacOS, you can do:

```
brew install kubernetes-helm
```

If you already have helm installed, make sure you're using the latest version:

```
brew upgrade kubernetes-helm
```

For other OS, please refer to the Helm installation docs: https://helm.sh/docs/using_helm/

The Fusion helm chart requires that helm is greater than version **`3.0.0`**; check your Helm version by running **`helm version --short`**.

HELM USER PERMISSIONS

If you require that fusion is installed by a user with minimal permissions, instead of an admin user, then the role and cluster role that will have to be assigned to the user within the namespace that you wish to install fusion in are documented in the **`install-roles`** directory.

Note

When working with Kubernetes on the command-line, it's useful to create a shell alias for `kubectl`, e.g.:

```
alias k=kubectl
```

To use these role in a cluster, as an admin user first create the namespace that you wish to install fusion into:

```
k create namespace fusion-namespace
```

Apply the `role.yaml` and `cluster-role.yaml` files to that namespace

```
k apply -f cluster-role.yaml k config set-context --current --
namespace=$NAMESPACE k apply -f role.yaml
```

Then bind the rolebinding and clusterolebinding to the install user:

```
k create --namespace fusion-namespace rolebinding fusion-install-
rolebinding --role fusion-installer --user <install_user> k create
clusterrolebinding fusion-install-rolebinding --clusterrole fusion-
installer --user <install_user>
```

You will then be able to run the `helm install` command as the `<install_user>`

Clone fusion-cloud-native from Github

You should clone this repo from github as you'll need to run the scripts on your local workstation:

```
git clone https://github.com/lucidworks/fusion-cloud-native.git
```

You should get into the habit of pulling this repo for the latest changes before performing any maintenance operations on your Fusion cluster to ensure you have the latest updates to the scripts.

```
cd fusion-cloud-native git pull
```

Cloning the github repo is preferred so that you can pull in updates to the scripts, but if you are not a git user, then you can download the project: https://github.com/lucidworks/fusion-cloud-native/archive/master.zip . Once downloaded, extract the zip and cd into the `fusion-cloud-native-master` directory.

The `setup_f5_aks.sh` script provided in this repo is strictly optional. The script is mainly to help those new to Kubernetes and/or Fusion get started quickly. If you're already familiar with K8s, Helm, and AKS, then you use Helm directly to install Fusion into an existing cluster or one you create yourself using the process described *here*.

If you're new to Azure, then please visit https://azure.microsoft.com/en-us/free/search/ to set up an account.

Set up the AKS CLI tools

Before launching an AKS cluster, you need to install and configure `kubectl` and `az` using the links provided below:

Required AKS Command-line Tools:

1. `kubectl` : Install kubectl
2. `az` : Installing the Azure CLI

To confirm your account access and command-line tools are set up correctly, run the `az login` command (`az login --help` to see available options).

Azure Prerequisites

To launch a cluster in AKS (or pretty much do anything with Azure) you need to setup a Resource Group. Resource Groups are a way of organizing and managing related resources in Azure. For more information about resource groups, see https://docs.microsoft.com/en-us/azure/azure-resource-manager/resource-group-overview#resource-groups .

You also need to choose a location where you want to spin up your AKS cluster, such as `westus2` . For a list of locations you can choose, see https://azure.microsoft.com/en-us/global-infrastructure/locations/ .

Use the Azure console in your browser to create a resource group, or simply do:

```
az group create -g $AZURE_RESOURCE_GROUP -l $AZURE_LOCATION
```

To recap, you should have the following requirements in place:

1. Azure Account set up.
2. `azure-cli` (`az`) command-line tools installed.
3. `az` login working.
4. Created an Azure Resource Group and selected a location to launch the cluster.

Set up Fusion on AKS

Download and run the `setup_f5_aks.sh` script to install Fusion 5.x in a AKS cluster. To create a new cluster and install Fusion, simply do:

```
./setup_f5_aks.sh -c <cluster_name> -p <aks_resource_group>
```

If you don't want the script to create a cluster, then you need to create a cluster before running the script and simply pass the name of the existing cluster using the `-c` parameter.

Use the `--help` option to see full script usage.

By default, our script installs Fusion into the default namespace; think of a K8s namespace as a virtual cluster within a physical cluster. You can install multiple instances of Fusion in the same cluster in separate namespaces. However, please do not install more than one Fusion release in the same namespace.

You can override the namespace using the `-n` option. In addition, our script uses f5 for the Helm release name; you can customize this using the `-r` option. Helm uses the release name you provide to track a specific instance of an installation, allowing you to perform updates and rollback changes for that specific release only.

You can also pass the `--preview` option to the script, which enables soon-to-be-released features for AKS, such as deploying a multi-zone cluster across 3 availability zones for higher availability guarantees. For more information about the Availability Zone feature, see https://docs.microsoft.com/en-us/azure/aks/availability-zones .

It takes a while for AKS to spin up the new cluster. The cluster will have three Standard_D4_v3 nodes which have 4 CPU cores and 16 GB of memory. Behind the scenes, our script calls the `az aks create` command.

Warning

If using Helm V2, the `setup_f5_aks.sh` script installs Helm's `tiller` component into your AKS cluster with the cluster admin role. If you don't want this, then please upgrade to Helm v3.

After running the `setup_f5_aks.sh` script, proceed to *Verifying the Fusion Installation*.

AKS Ingress

The `setup_f5_aks.sh` script exposes the Fusion proxy service on an external IP over HTTP. This is done for demo or getting started purposes. However, you're strongly encouraged to configure a K8s Ingress with TLS termination in front of the proxy service.

Use the `-t` and `-h <hostname>` options to have our script create an Ingress with a TLS certificate issued by Let's Encrypt.

Upgrades and Ingress

Important

If you used the `-t -h <hostname>` options when installing your cluster, our script created an additional values yaml file named `tls-values.yaml`.

To make things easier for you when upgrading, you should add the settings from this file into your main custom values yaml file. For example:

```
api-gateway: service: type: "NodePort" ingress: enabled: true host: "
<hostname>" tls: enabled: true annotations:
"networking.gke.io/managed-certificates": "<RELEASE>-managed-
certificate" "kubernetes.io/ingress.class": "gce"
```

This way, you don't have to remember to pass the additional `tls-values.yaml` file when upgrading.

Verifying the Fusion Installation

In this section, we provide some tips on how to verify the Fusion installation.

Tip

Check if the Fusion Admin UI is available at `https://<fusion-host>:6764/admin/`.

Let's review some useful kubectl commands.

Enhance the K8s Command-line Experience

Here is a list of tools we found useful for improving your command-line experience with Kubernetes:

- krew (kubectl plugin mgr): https://github.com/kubernetes-sigs/krew/
- kube-ps1 (show current context on command line prompt): https://github.com/jonmosco/kube-ps1
- kubectx / kubens (switch between clusters / namespaces): https://github.com/ahmetb/kubectx

Useful kubectl commands

kubectl reference: https://kubernetes.io/docs/reference/generated/kubectl/kubectl-commands

Set the namespace for `kubectl` if not using the default:

```
kubectl config set-context --current --namespace=<NAMESPACE>
```

This saves you from having to pass `-n` *with every command.*

Get a list of running pods: `k get pods`

Get logs for a pod using a label: `k logs -l app.kubernetes.io/component=query-pipeline`

Get pod deployment spec and details: `k get pods <pod_id> -o yaml`

Get details about a pod events: `k describe po <pod_id>`

Port forward to a specific pod: `k port-forward <pod_id> 8983:8983`

SSH into a pod: `k exec -it <pod_id> -- /bin/bash`

CPU/Memory usage report for pods: `k top pods`

Forcefully kill a pod: `k delete po <pod_id> --force --grace-period 0`

Scale up (or down) a deployment: `k scale deployment.v1.apps/<id> --replicas=N`

Get a list of pod versions: `k get po -o jsonpath='{..image}' | tr -s '.' '\n' | sort | uniq`

Check Fusion Pods and Services

Once the install script completes, you can check that all pods and services are available using:

```
kubectl get pods
```

If all goes well, you should see a list of pods similar to:

```
NAME READY STATUS RESTARTS AGE seldon-controller-manager-6675874894-
qxwrv 1/1 Running 0 8m45s f5-admin-ui-74d794f4f8-m5jms 1/1 Running 0
8m45s f5-ambassador-fd6b9b5dc-7ghf6 1/1 Running 0 8m43s f5-api-
gateway-6b9998b9c-tmchk 1/1 Running 0 8m45s f5-auth-ui-7565564b4c-
rdc74 1/1 Running 0 8m42s f5-classic-rest-service-0 1/1 Running 3
8m44s f5-devops-ui-77bb867ffb-fbzxd 1/1 Running 0 8m42s f5-fusion-
admin-78b8f8fc7f-4d7l8 1/1 Running 0 8m42s f5-fusion-indexing-
599c8d448-xzsvm 1/1 Running 0 8m44s f5-insights-665fd9f6fc-g5psw 1/1
Running 0 8m43s f5-job-launcher-84dd4c5c96-p8528 1/1 Running 0 8m44s
f5-job-rest-server-6d44d964b8-xtnxw 1/1 Running 0 8m45s f5-logstash-0
1/1 Running 0 8m45s f5-ml-model-service-6987dc94c9-9ppp8 2/2 Running 1
8m45s f5-monitoring-grafana-5d499dbb58-pzw72 1/1 Running 0 10m f5-
monitoring-prometheus-kube-state-metrics-54d6678dv9h7h 1/1 Running 0
10m f5-monitoring-prometheus-pushgateway-7d65c65b85-vwrwf 1/1 Running
0 10m f5-monitoring-prometheus-server-0 2/2 Running 0 10m f5-pm-ui-
86cbc5bb65-nd2n8 1/1 Running 0 8m44s f5-pulsar-bookkeeper-0 1/1
Running 0 8m45s f5-pulsar-broker-b56cc776f-56msx 1/1 Running 0 8m45s
f5-query-pipeline-5d75d7d5f4-l2mdf 1/1 Running 0 8m43s f5-connectors-
7bb6cfc65f-7wfs2 1/1 Running 0 8m42s f5-connectors-backend-987fdc648-
dldwv 1/1 Running 0 8m45s f5-rules-ui-6b9d55b78f-9hzzj 1/1 Running 0
8m43s f5-solr-0 1/1 Running 0 8m44s f5-solr-exporter-c4687c785-jsm7x
1/1 Running 0 8m45s f5-ui-6cdbcc68c6-rj9cq 1/1 Running 0 8m45s f5-
webapps-6d6bb9bfd-hm4qx 1/1 Running 0 8m45s f5-workflow-controller-
7b66679fb7-sjbvp 1/1 Running 0 8m44s f5-zookeeper-0 1/1 Running 0
8m45s
```

The number of pods per deployment / statefulset will vary based on your cluster size and replicaCount settings in your custom values YAML file. Also, don't worry if you see some pods having been restarted as that just means they were too slow to come up and Kubernetes killed and restarted them. You do want to see at least one pod running for every service. If a pod is not running after waiting a sufficient amount of time, use `kubectl logs <pod_id>` to see the logs for that pod; to see the logs for previous versions of a pod, use: `kubectl logs <pod_id> -p`. You can also look at the actions Kubernetes performed on the pod using `kubectl describe po <pod_id>`.

To see a list of Fusion services, do:

```
kubectl get svc
```

For an overview of the various Fusion 5 microservices, see: Fusion microservices .

Once you're ready to build a Fusion cluster for production, please see see more information at Fusion 5 Survival Guide .

Upgrading with Zero Downtime

One of the most powerful features provided by Kubernetes and a cloud-native microservices architecture is the ability to do a rolling update on a live cluster. Fusion 5 allows customers to upgrade from Fusion 5.x.y to a later 5.x.z version on a live cluster with zero downtime or disruption of service.

When Kubernetes performs a rolling update to an individual microservice, there will be a mix of old and new services in the cluster concurrently (only briefly in most cases) and requests from other services will be routed to both versions. Consequently, Lucidworks ensures all changes we make to our service do not break the API interface exposed to other services in the same 5.x line of releases. We also ensure stored configuration remains compatible in the same 5.x release line.

Lucidworks releases minor updates to individual services frequently, so our customers can pull in those upgrades using Helm at their discretion.

To upgrade your cluster at any time, use the `--upgrade` option with our setup scripts in this repo.

The scripts in this repo automatically pull in the latest chart updates from our Helm repository and deploy any updates needed by doing a diff of your current installation and the latest release from Lucidworks. To see what would be upgraded, you can pass the `--dry-run` option to the script.

Grafana Dashboards

Get the initial Grafana password from a K8s secret by doing:

```
kubectl get secret --namespace "${NAMESPACE}" ${RELEASE}-monitoring-
grafana \ -o jsonpath="{.data.admin-password}" | base64 --decode ;
echo
```

With Grafana, you can either setup a temporary port-forward to a Grafana pod or expose Grafana on an external IP using a K8s LoadBalancer. To define a LoadBalancer, do (replace ${RELEASE} with your Helm release label):

```
kubectl expose deployment ${RELEASE}-monitoring-grafana --
type=LoadBalancer --name=grafana --port=3000 --target-port=3000
```

You can use `kubectl get services --namespace <namespace>` to determine when the load balancer is setup and its IP address. Direct your browser to http://<GrafanaIP>:3000 and enter the username `admin@localhost` and the password that was returned in the previous step.

This will log you into the application. It is recommended that you create another administrative user with a more desirable password.

The dashboards and datasoure will be setup for you in grafana, simply navigate to `Dashboards` → `Manage` to view the vailable dashboards

More deployment options

- *How to deploy Fusion 5 in Google Kubernetes Engine*
- *How to deploy Fusion 5 in Amazon Elastic Kubernetes Service*
- *How to deploy Fusion on 5 other Kubernetes platforms*

Frequently Asked Questions

Can the stateful database services, for example, MySQL, be supported by an Azure PaaS service?

This option is not supported. In theory, it may be possible to implement this function.

Is it possible to use cross-zone storage solutions rather than volumes, such as Microsoft Azure file storage, for stateful services?

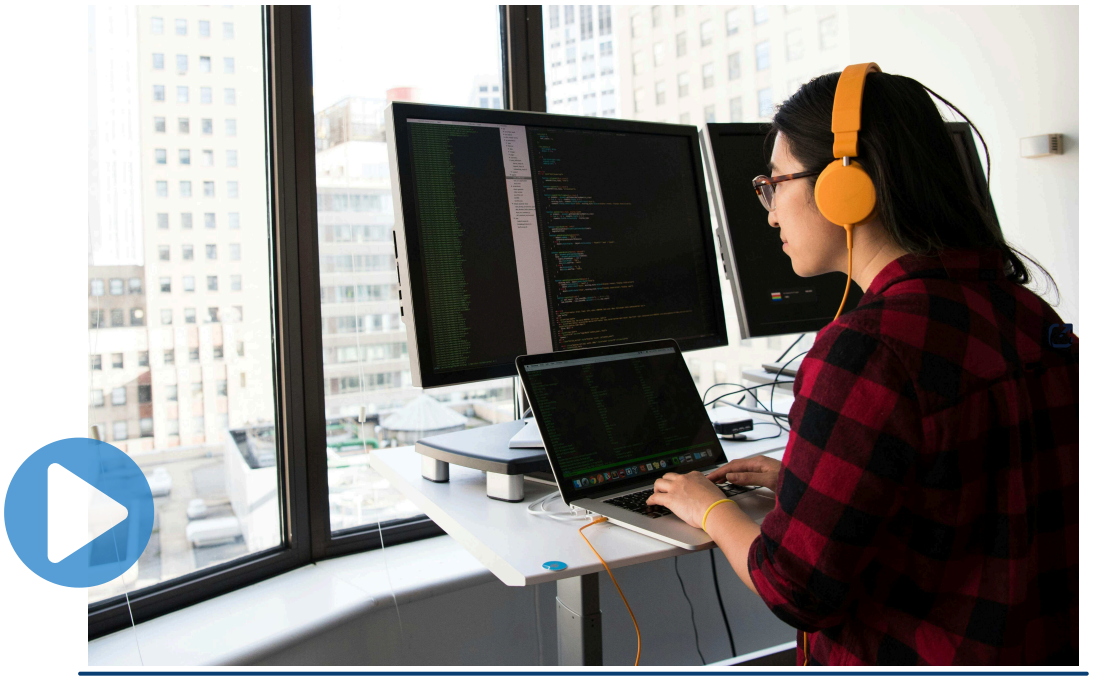
While in theory it may be possible to implement this functionality, the configuration has not been tested by Lucidworks and is not supported.

Additional resources

LucidAcademy

Lucidworks offers free training to help you get started.

The **Course** for **Deploying Fusion 5** focuses on the prerequisite software needed to deploy Fusion, the necessary setup steps, and the physical act of deployment:



Visit the LucidAcademy to see the full training catalog.

Deploy Fusion 5 on Google Kubernetes Engine (GKE)

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion*

Fusion supports deployment on Google Kubernetes Engine (GKE). This topic explains how to deploy a Fusion cluster on GKE using the `setup_f5_gke.sh` script in the `fusion-cloud-native` repository.

Prerequisites

This section covers prerequisites and background knowledge needed to help you understand the structure of this document and how the Fusion installation process works with Kubernetes.

Release Name and Namespace

Before installing Fusion, you need to choose a Kubernetes namespace to install Fusion into. Think of a K8s namespace as a virtual cluster within a physical cluster. You can install multiple instances of Fusion in the same cluster **in separate namespaces**. However, please do not install more than one Fusion release in the same namespace.

NOTE: All Fusion services must run in the same namespace, i.e. you should not try to split a Fusion cluster across multiple namespaces.

Use a short name for the namespace, containing only letters, digits, or dashes (no dots or underscores). The setup scripts in this repo use the namespace for the Helm release name by default.

Install Helm

Helm is a package manager for Kubernetes that helps you install and manage applications on your Kubernetes cluster. Regardless of which Kubernetes platform you're using, you need to install `helm` as it is required to install Fusion for any K8s platform. On MacOS, you can do:

```
brew install kubernetes-helm
```

If you already have helm installed, make sure you're using the latest version:

```
brew upgrade kubernetes-helm
```

For other OS, please refer to the Helm installation docs: https://helm.sh/docs/using_helm/

The Fusion helm chart requires that helm is greater than version `3.0.0`; check your Helm version by running `helm version --short`.

HELM USER PERMISSIONS

If you require that fusion is installed by a user with minimal permissions, instead of an admin user, then the role and cluster role that will have to be assigned to the user within the namespace that you wish to install fusion in are documented in the `install-roles` directory.

Note

When working with Kubernetes on the command-line, it's useful to create a shell alias for `kubectl`, e.g.:

```
alias k=kubectl
```

To use these role in a cluster, as an admin user first create the namespace that you wish to install fusion into:

```
k create namespace fusion-namespace
```

Apply the `role.yaml` and `cluster-role.yaml` files to that namespace

```
k apply -f cluster-role.yaml k config set-context --current --
namespace=$NAMESPACE k apply -f role.yaml
```

Then bind the rolebinding and clusterolebinding to the install user:

```
k create --namespace fusion-namespace rolebinding fusion-install-
rolebinding --role fusion-installer --user <install_user> k create
clusterrolebinding fusion-install-rolebinding --clusterrole fusion-
installer --user <install_user>
```

You will then be able to run the `helm install` command as the `<install_user>`

Clone fusion-cloud-native from Github

You should clone this repo from github as you'll need to run the scripts on your local workstation:

```
git clone https://github.com/lucidworks/fusion-cloud-native.git
```

You should get into the habit of pulling this repo for the latest changes before performing any maintenance operations on your Fusion cluster to ensure you have the latest updates to the scripts.

```
cd fusion-cloud-native git pull
```

Cloning the github repo is preferred so that you can pull in updates to the scripts, but if you are not a git user, then you can download the project: https://github.com/lucidworks/fusion-cloud-native/archive/master.zip . Once downloaded, extract the zip and cd into the `fusion-cloud-native-master` directory.

The `setup_f5_gke.sh` script provided in this repo is strictly optional. The script is mainly to help those new to Kubernetes and/or Fusion get started quickly. If you're already familiar with K8s, Helm, and GKE, then you can skip the script and just use Helm directly to install Fusion into an existing cluster or one you create yourself using the process described *here*.

Set up the Google Cloud SDK (one time only)

If you've already installed the `gcloud` command-line tools, you can skip to *Create a Fusion cluster in GKE*.

These steps set up your local Google Cloud SDK environment so that you're ready to use the command-line tools to manage your Fusion deployment.

Usually, you only need to perform these setup steps once. After that, you're ready to *create a cluster*.

For a nice getting started tutorial for GKE, see: Deploy an app to a GKE cluster .

How to set up the Google Cloud SDK

1. Enable the Kubernetes Engine API .
2. Log in to Google Cloud: `gcloud auth login`
3. Set up the Google Cloud SDK:
    a. `gcloud config set compute/zone <zone-name>`
    If you are working with regional clusters instead of zone clusters, use `gcloud config set compute/region <region-name>` instead.
    b. `gcloud config set core/account <email address>`
    c. *New GKE projects only:* `gcloud projects create <new-project-name>`
    If you have already created a project, for example in the Google Cloud Platform console , then skip to the next step.
    d. `gcloud config set project <project-name>`

Make sure you install the Kubernetes command-line tool `kubectl` using:

```
gcloud components install kubectl gcloud components update
```

Create a single-node demo cluster

Run the `setup_f5_gke.sh` script to install Fusion 5.x in a GKE cluster. To create a new, single-node **demo** cluster and install Fusion, simply do:

```
./setup_f5_gke.sh -c <cluster_name> -p <gcp_project_id> --create demo
```

Use the `--help` option to see script usage. If you want the script to create a cluster for you, then you need to pass the `--create` option with either `demo` or `multi_az`. If you don't want the script to create a cluster, then you need to create a cluster before running the script and simply pass the name of the existing cluster using the `-c` parameter.

If you pass `--create demo` to the script, then we create a single node GKE cluster (defaults to using `n1-standard-8` node type). The minimum node type you'll need for a 1 node cluster is an `n1-standard-8` (on GKE) which has 8 CPU and 30 GB of memory. This is cutting it very close in terms of resources as you also need to host all of the Kubernetes system pods on this same node. Obviously, this works for kicking the tires on Fusion 5.1 but is not sufficient for production workloads.

You can change the instance type using the `-i` parameter; see: https://cloud.google.com/compute/docs/regions-zones/#available for an list of which machine types are available in your desired region.

Note: If not provided the script generates a custom values file named `gke_<cluster>_<namespace>_fusion_values.yaml` *which you can use to customize the Fusion chart.*

WARNING If using Helm V2, the `setup_f5_gke.sh` script installs Helm's `tiller` component into your GKE cluster with the cluster admin role. If you don't want this, then please upgrade to Helm v3.

If you see an error similar to the following, then wait a few seconds and try running the `setup_f5_gke.sh` script again with the same arguments as this is usually a transient issue:

```
Error: could not get apiVersions from Kubernetes: unable to retrieve
the complete list of server APIs: metrics.k8s.io/v1beta1: the server
is currently unable to handle the request
```

After running the `setup_f5_gke.sh` script, proceed to the *Verifying the Fusion Installation* section below.

When you're ready to deploy Fusion to a production-like environment, see more information at Fusion 5 Survival Guide.

Create a three-node regional cluster to withstand a zone outage

With a three-node regional cluster, nodes are deployed across three separate availability zones.

```
./setup_f5_gke.sh -c <cluster> -p <project> -n <namespace> --region
<region-name> --create multi_az
```

- `<cluster>` value should be the name of a non-existent cluster; the script will create the new cluster.
- `<project>` must match the name of an existing project in GKE. Run `gcloud config get-value project` to get this value, or see the *GKE setup instructions*.
- `<namespace>` Kubernetes namespace to install Fusion into, defaults to `default` with release `f5`
- `<region-name>` value should be the name of a GKE region, defaults to `us-west1`. Run `gcloud config get-value compute/zone` to get this value, or see the *GKE setup instructions* to set the value.

In this configuration, Kubernetes deploys a ZooKeeper and Solr pod on each of the three nodes, which allows the cluster to retain ZK quorum and remain operational after losing one node, such as during an outage in one availability zone.

When running in a multi-zone cluster, each Solr node has the `solr_zone` system property set to the zone it is running in, such as `-Dsolr_zone=us-west1-a`.

After running the `setup_f5_gke.sh` script, proceed to the *Verifying the Fusion Installation* section below.

When you're ready to deploy Fusion to a production-like environment, see more information at Fusion 5 Survival Guide.

GKE Ingress and TLS

The Fusion proxy service provides authentication and serves as an API gateway for accessing all other Fusion services. It's typical to use an Ingress for TLS termination in front of the proxy service.

The `setup_f5_gke.sh` supports creating an Ingress with an TLS cert for a domain you own by passing: `-t -h <hostname>`

After the script runs, you need to create an A record in GCP's DNS service to map your domain name to the Ingress IP. Once this occurs, our script setup uses Let's Encrypt to issue a TLS cert for your Ingress.

To see the status of the Let's Encrypt issued certificate, do:

```
kubectl get managedcertificates -n <namespace> -o yaml
```

Please refer to the Kubernetes documentation on configuring an Ingress for GKE:
Setting up HTTP Load Balancing with Ingress

Note

The GCP Ingress defaults to a 30 second timeout, which can lead to false negatives for long running requests such as importing apps. To configure the timeout for the backend in kubernetes:

Create a BackendConfig object in your namespace:

```
--- apiVersion: cloud.google.com/v1beta1 kind: BackendConfig metadata:
name: backend_config_name spec: timeoutSec: 120 connectionDraining:
drainingTimeoutSec: 60
```

Then make sure that the following entries are in the right place in your values.yaml file:

```
api-gateway: service: annotations: beta.cloud.google.com/backend-
config: '{"ports": {"6764":"backend_config_name"}}'
```

and upgrade your release to apply the configuration changes

Ingresses and externalTrafficPolicy

When running a fusion cluster behind an externally controlled LoadBalancer it can be advantageous to configure the **`externalTrafficPolicy`** of the **`proxy`** service to **`Local`**. This preserves the client source IP and avoids a second hop for LoadBalancer and NodePort type services, but risks potentially imbalanced traffic spreading. Although when running in a cluster with a dedicated pool for spark jobs that can scale up and down freely it can prevent unwanted request failures. This behaviour can be altered with the **`api-gateway.service.externalTrafficPolicy`** value, which is set to **`Local`** if the example values file is used.

You must use `externalTrafficPolicy = Local` *for the Trusted HTTP Realm to work correctly.*

If you are already using a custom 'values.yaml' file, create an entry for **`externalTrafficPolicy`** under **`api-gateway`** service.

```
api-gateway: service: externalTrafficPolicy: Local
```

Considerations when using the nginx ingress controller

If you are using the **`nginx`** ingress controller to fulfil your ingress definitions there are a couple of options that are recommended to be set in the configmap:

```
enable-underscores-in-headers: "true" # Fusion can return some headers
that have underscores, these have to be explicitly enabled in nginx
proxy-body-size: "0" # By default nginx places a maximum size on
request bodies, either increase as needed or disable by setting to 0
proxy-read-timeout: "300" # Increases the timeout for potential slow
queries.
```

Custom values

There are some example values files that can be used as a starting point for resources, affinity and replica count configuration in the **`example-values`** folder. These can be passed to the install script using the **`--values`** option, for example:

```
./setup_f5_gke.sh -c <cluster> -p <project> -r <release> -n
<namespace> \ --values example-values/affinity.yaml --values example-
values/resources.yaml --values example-values/replicas.yaml
```

The **`--values`** option can be passed multiple times, if the same configuration property is contained within multiple **`values`** files then the values from the latest file passed as a **`--values`** option are used.

CONNECTORS CUSTOM VALUES

If you are using Fusion 5.9 or later, you can specify resources and replica count per connector. This allows you to set different resource limits for each connector. If you do not set custom values for a connector, that connector uses the default values.

Set each connector's resource values in the `connector-plugin` section under `pluginValues`. The `pluginValues` section is a list of plugins and its resources. The following sample shows an example.

```
pluginValues: - id: "plugin-id"  1  resources:  2  limits: cpu: "2"
memory: "3Gi" requests: cpu: "250m" memory: "2Gi" replicaCount: 1  3
```

1	The plugin ID. The plugin ID must match the plugin ID on the plugin ZIP file. without the `lucidworks.` prefix. For example, if the plugin ID on the plugin ZIP file is `lucidworks.sharepoint-optimized`, the plugin ID is `sharepoint-optimized`.
2	The resources settings. You may specify the limits, the requests, and the CPU and memory for each.
3	The number of replicas per connector. This value is 1 by default.

IMPORTANT After editing the `connector-plugin` section, you must reinstall the affected connector.

Upgrades and Ingress

IMPORTANT If you used the `-t -h <hostname>` options when installing your cluster, our script created an additional values yaml file named `tls-values.yaml`.

To make things easier for you when upgrading, you should add the settings from this file into your main custom values yaml file, e.g.:

```
api-gateway: service: type: "NodePort" ingress: enabled: true host: "
<hostname>" tls: enabled: true annotations:
"networking.gke.io/managed-certificates": "<RELEASE>-managed-
certificate" "kubernetes.io/ingress.class": "gce"
```

This way you don't have to remember to pass the additional `tls-values.yaml` file when upgrading.

If you're not running on a managed K8s platform like GKE, AKS, or EKS, you can use Helm to install the Fusion chart to an existing Kubernetes cluster.

Fusion version 5.5 now includes support for the Rancher Kubernetes Engine (RKE) platform. Before deploying Fusion to RKE, you must download and install the RKE software. After configuring your cluster, you can proceed with the Helm v3 installation.

Note

You must have a working cluster configured before performing the Helm v3 installation.

Use Helm v3 to Install Fusion

You should upgrade to the latest version of Helm v3 for working with Fusion. If you need to keep Helm V2 for other clusters, ensure Helm V3 is ahead of Helm V2 in your working shell's PATH before proceeding.

CUSTOMIZE FUSION CHART SETTINGS

Fusion aims to be well-configured out-of-the-box, but you can customize any of the built-in settings using a custom values `YAML` file. If you use one of our setup scripts, such as `setup_f5_gke.sh`, then it will create a custom values YAML file for you the first time you run it using the `customize_fusion_values.yaml.example` as a template.

If you're working with Helm directly and not using one of our setup scripts, then run the `customize_fusion_values.sh` script to create a custom values YAML file from our `customize_fusion_values.yaml.example` template as a starting point:

```
./customize_fusion_values.sh -c <cluster> -n <namespace> \ --provider
<provider> --num-solr 1 --node-pool "<node_pool>"
```

Note

Pass `--help` for usage details.

In this example:

- `<provider>` is the K8s platform you're running on, such as `gke`
- `<cluster>` is the name of your cluster
- `<namespace>` is the K8s namespace where you plan to install Fusion

Note

The `--node-pool` option specifies the node selector label for determining which nodes to run Fusion pods. You can pass `"{}"` to let Kubernetes decide which nodes to schedule pods on.

This file is referred to as `${MY_VALUES}` in the commands belo. Replace the filename with the correct filename for your environment. Keep this file handy, as you'll need it to customize Fusion settings and upgrade to a newer version.

Review the settings in the custom values YAML file to ensure the defaults are appropriate for your environment, including the number of Solr and Zookeeper replicas.

Add the Lucidworks Helm repo:

```
helm repo add lucidworks https://charts.lucidworks.com
```

The `customize_fusion_values.sh` script creates an upgrade script to install/upgrade Fusion into Kubernetes using Helm. Look in the directory where you ran `customize_fusion_values.sh` for a script named like: `<provider>_<cluster>_<namespace>_upgrade_fusion.sh`. Run this script to install Fusion.

RedHat OpenShift

We can deploy Fusion in an existing OpenShift cluster. This cluster should be created using OpenShift Infrastructure Provider. A Red Hat Customer Portal account is required. OpenShift Online services are not supported.

The easiest way to install on OpenShift is to run the `setup_f5_k8s.sh` script for your **existing** cluster; use the `--help` option to see script usage. For instance, the following command will install Fusion 5 into the specified namespace (`-n`) and OpenShift cluster (`-c`):

```
./setup_f5_k8s.sh -c <CLUSTER> -n <NAMESPACE> --provider oc
```

Tip: `kubectl` should work with your OpenShift cluster (see: https://docs.openshift.com/container-platform/4.1/cli_reference/usage-oc-kubectl.html) and Lucidworks recommends installing the latest `kubectl` for your workstation instead of using `oc` for installing Fusion 5. However, if you do not have `kubectl` installed, then you'll need to update the upgrade script created by `setup_f5_k8s.sh` to use `oc` instead of `kubectl` (search and replace on the BASH script using a text editor).

When you're ready to deploy Fusion to a production-like environment, see more information at Fusion 5 Survival Guide.

Lucidworks recommends using Helm v3, but in case Tiller is required for Helm v2, the cluster security needs to be relaxed to allow images to run with different UIDs:

```
oc adm policy add-scc-to-group anyuid system:authenticated
```

Verifying the Fusion Installation

In this section, we provide some tips on how to verify the Fusion installation.

Tip

Check if the Fusion Admin UI is available at `https://<fusion-host>:6764/admin/`.

Let's review some useful kubectl commands.

Enhance the K8s Command-line Experience

Here is a list of tools we found useful for improving your command-line experience with Kubernetes:

- krew (kubectl plugin mgr): https://github.com/kubernetes-sigs/krew/
- kube-ps1 (show current context on command line prompt): https://github.com/jonmosco/kube-ps1
- kubectx / kubens (switch between clusters / namespaces): https://github.com/ahmetb/kubectx

Useful kubectl commands

kubectl reference: https://kubernetes.io/docs/reference/generated/kubectl/kubectl-commands

Set the namespace for `kubectl` if not using the default:

```
kubectl config set-context --current --namespace=<NAMESPACE>
```

This saves you from having to pass `-n` *with every command.*

Get a list of running pods: **`k get pods`**

Get logs for a pod using a label: **`k logs -l app.kubernetes.io/component=query-pipeline`**

Get pod deployment spec and details: **`k get pods <pod_id> -o yaml`**

Get details about a pod events: **`k describe po <pod_id>`**

Port forward to a specific pod: **`k port-forward <pod_id> 8983:8983`**

SSH into a pod: **`k exec -it <pod_id> -- /bin/bash`**

CPU/Memory usage report for pods: **`k top pods`**

Forcefully kill a pod: **`k delete po <pod_id> --force --grace-period 0`**

Scale up (or down) a deployment: **`k scale deployment.v1.apps/<id> --replicas=N`**

Get a list of pod versions: **`k get po -o jsonpath='{..image}' | tr -s '.' '\n' | sort | uniq`**

Check Fusion Pods and Services

Once the install script completes, you can check that all pods and services are available using:

```
kubectl get pods
```

If all goes well, you should see a list of pods similar to:

```
NAME READY STATUS RESTARTS AGE seldon-controller-manager-6675874894-
qxwrv 1/1 Running 0 8m45s f5-admin-ui-74d794f4f8-m5jms 1/1 Running 0
8m45s f5-ambassador-fd6b9b5dc-7ghf6 1/1 Running 0 8m43s f5-api-
gateway-6b9998b9c-tmchk 1/1 Running 0 8m45s f5-auth-ui-7565564b4c-
rdc74 1/1 Running 0 8m42s f5-classic-rest-service-0 1/1 Running 3
8m44s f5-devops-ui-77bb867ffb-fbzxd 1/1 Running 0 8m42s f5-fusion-
admin-78b8f8fc7f-4d7l8 1/1 Running 0 8m42s f5-fusion-indexing-
599c8d448-xzsvm 1/1 Running 0 8m44s f5-insights-665fd9f6fc-g5psw 1/1
Running 0 8m43s f5-job-launcher-84dd4c5c96-p8528 1/1 Running 0 8m44s
f5-job-rest-server-6d44d964b8-xtnxw 1/1 Running 0 8m45s f5-logstash-0
1/1 Running 0 8m45s f5-ml-model-service-6987dc94c9-9ppp8 2/2 Running 1
8m45s f5-monitoring-grafana-5d499dbb58-pzw72 1/1 Running 0 10m f5-
monitoring-prometheus-kube-state-metrics-54d6678dv9h7h 1/1 Running 0
10m f5-monitoring-prometheus-pushgateway-7d65c65b85-vwrwf 1/1 Running
0 10m f5-monitoring-prometheus-server-0 2/2 Running 0 10m f5-pm-ui-
86cbc5bb65-nd2n8 1/1 Running 0 8m44s f5-pulsar-bookkeeper-0 1/1
Running 0 8m45s f5-pulsar-broker-b56cc776f-56msx 1/1 Running 0 8m45s
f5-query-pipeline-5d75d7d5f4-l2mdf 1/1 Running 0 8m43s f5-connectors-
7bb6cfc65f-7wfs2 1/1 Running 0 8m42s f5-connectors-backend-987fdc648-
dldwv 1/1 Running 0 8m45s f5-rules-ui-6b9d55b78f-9hzzj 1/1 Running 0
8m43s f5-solr-0 1/1 Running 0 8m44s f5-solr-exporter-c4687c785-jsm7x
1/1 Running 0 8m45s f5-ui-6cdbcc68c6-rj9cq 1/1 Running 0 8m45s f5-
webapps-6d6bb9bfd-hm4qx 1/1 Running 0 8m45s f5-workflow-controller-
7b66679fb7-sjbvp 1/1 Running 0 8m44s f5-zookeeper-0 1/1 Running 0
8m45s
```

The number of pods per deployment / statefulset will vary based on your cluster size and replicaCount settings in your custom values YAML file. Also, don't worry if you see some pods having been restarted as that just means they were too slow to come up and Kubernetes killed and restarted them. You do want to see at least one pod running for every service. If a pod is not running after waiting a sufficient amount of time, use `kubectl logs <pod_id>` to see the logs for that pod; to see the logs for previous versions of a pod, use: `kubectl logs <pod_id> -p`. You can also look at the actions Kubernetes performed on the pod using `kubectl describe po <pod_id>`.

To see a list of Fusion services, do:

```
kubectl get svc
```

For an overview of the various Fusion 5 microservices, see: Fusion microservices .

Once you're ready to build a Fusion cluster for production, please see see more information at Fusion 5 Survival Guide .

Upgrading with Zero Downtime

One of the most powerful features provided by Kubernetes and a cloud-native microservices architecture is the ability to do a rolling update on a live cluster. Fusion 5 allows customers to upgrade from Fusion 5.x.y to a later 5.x.z version on a live cluster with zero downtime or disruption of service.

When Kubernetes performs a rolling update to an individual microservice, there will be a mix of old and new services in the cluster concurrently (only briefly in most cases) and requests from other services will be routed to both versions. Consequently, Lucidworks ensures all changes we make to our service do not break the API interface exposed to other services in the same 5.x line of releases. We also ensure stored configuration remains compatible in the same 5.x release line.

Lucidworks releases minor updates to individual services frequently, so our customers can pull in those upgrades using Helm at their discretion.

To upgrade your cluster at any time, use the `--upgrade` option with our setup scripts in this repo.

The scripts in this repo automatically pull in the latest chart updates from our Helm repository and deploy any updates needed by doing a diff of your current installation and the latest release from Lucidworks. To see what would be upgraded, you can pass the `--dry-run` option to the script.

Grafana Dashboards

Get the initial Grafana password from a K8s secret by doing:

```
kubectl get secret --namespace "${NAMESPACE}" ${RELEASE}-monitoring-
grafana \ -o jsonpath="{.data.admin-password}" | base64 --decode ;
echo
```

With Grafana, you can either setup a temporary port-forward to a Grafana pod or expose Grafana on an external IP using a K8s LoadBalancer. To define a LoadBalancer, do (replace ${RELEASE} with your Helm release label):

```
kubectl expose deployment ${RELEASE}-monitoring-grafana --
type=LoadBalancer --name=grafana --port=3000 --target-port=3000
```

You can use `kubectl get services --namespace <namespace>` to determine when the load balancer is setup and its IP address. Direct your browser to http://<GrafanaIP>:3000 and enter the username `admin@localhost` and the password that was returned in the previous step.

This will log you into the application. It is recommended that you create another administrative user with a more desirable password.

The dashboards and datasoure will be setup for you in grafana, simply navigate to `Dashboards` → `Manage` to view the vailable dashboards

More deployment options

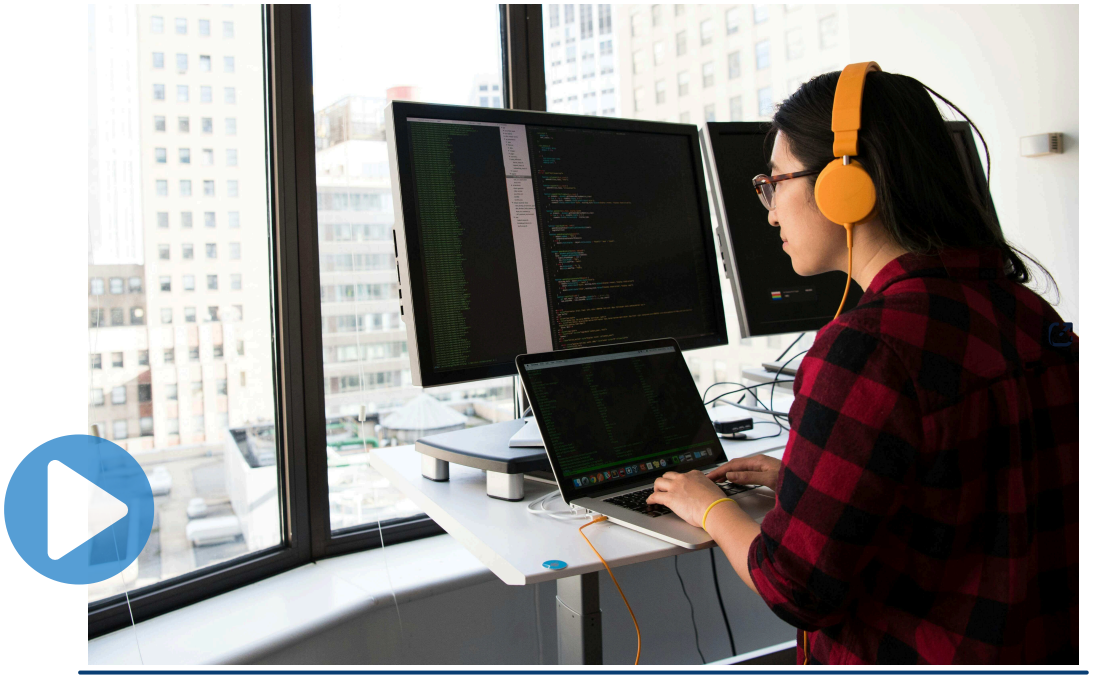
- *How to deploy Fusion 5 on Amazon Elastic Kubernetes Service*
- *How to deploy Fusion 5 on Azure Kubernetes Service*
- *How to deploy Fusion on 5 other Kubernetes platforms*

Additional resources

LucidAcademy

Lucidworks offers free training to help you get started.

The **Course** for **Deploying Fusion 5** focuses on the prerequisite software needed to deploy Fusion, the necessary setup steps, and the physical act of deployment:



Visit the LucidAcademy to see the full training catalog.

Deploy Fusion 5 on Other Kubernetes Platforms

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion*

The `setup_f5_k8s.sh` script in the `fusion-cloud-native` repository provides deployment support for any Kubernetes platform, including on-premise, private cloud, public cloud, and hybrid platforms.

This script is used by the `setup_f5_gke.sh`, `setup_f5_eks.sh`, and `setup_f5_aks.sh` scripts, which provide additional platform-specific support for Google Kubernetes Engine (GKE), Amazon Elastic Kubernetes Service (EKS), and Azure Kubernetes Service (AKS).

This topic explains how to deploy a Fusion cluster in Kubernetes using the `setup_f5_k8s.sh` script in the `fusion-cloud-native` repository.

If you're deploying on-premises or using a localized repository, you'll need to *use a private repository for Docker images*.

Prerequisites

This section covers prerequisites and background knowledge needed to help you understand the structure of this document and how the Fusion installation process works with Kubernetes.

Release Name and Namespace

Before installing Fusion, you need to choose a Kubernetes namespace to install Fusion into. Think of a K8s namespace as a virtual cluster within a physical cluster. You can install multiple instances of Fusion in the same cluster **in separate namespaces**. However, please do not install more than one Fusion release in the same namespace.

NOTE: All Fusion services must run in the same namespace, i.e. you should not try to split a Fusion cluster across multiple namespaces.

Use a short name for the namespace, containing only letters, digits, or dashes (no dots or underscores). The setup scripts in this repo use the namespace for the Helm release name by default.

Install Helm

Helm is a package manager for Kubernetes that helps you install and manage applications on your Kubernetes cluster. Regardless of which Kubernetes platform you're using, you need to install `helm` as it is required to install Fusion for any K8s platform. On MacOS, you can do:

```
brew install kubernetes-helm
```

If you already have helm installed, make sure you're using the latest version:

```
brew upgrade kubernetes-helm
```

For other OS, please refer to the Helm installation docs: https://helm.sh/docs/using_helm/

The Fusion helm chart requires that helm is greater than version `3.0.0`; check your Helm version by running `helm version --short`.

HELM USER PERMISSIONS

If you require that fusion is installed by a user with minimal permissions, instead of an admin user, then the role and cluster role that will have to be assigned to the user within the namespace that you wish to install fusion in are documented in the `install-roles` directory.

Note

When working with Kubernetes on the command-line, it's useful to create a shell alias for `kubectl`, e.g.:

```
alias k=kubectl
```

To use these role in a cluster, as an admin user first create the namespace that you wish to install fusion into:

```
k create namespace fusion-namespace
```

Apply the `role.yaml` and `cluster-role.yaml` files to that namespace

```
k apply -f cluster-role.yaml k config set-context --current --
namespace=$NAMESPACE k apply -f role.yaml
```

Then bind the rolebinding and clusterolebinding to the install user:

```
k create --namespace fusion-namespace rolebinding fusion-install-
rolebinding --role fusion-installer --user <install_user> k create
clusterrolebinding fusion-install-rolebinding --clusterrole fusion-
installer --user <install_user>
```

You will then be able to run the `helm install` command as the `<install_user>`

Clone fusion-cloud-native from Github

You should clone this repo from github as you'll need to run the scripts on your local workstation:

```
git clone https://github.com/lucidworks/fusion-cloud-native.git
```

You should get into the habit of pulling this repo for the latest changes before performing any maintenance operations on your Fusion cluster to ensure you have the latest updates to the scripts.

```
cd fusion-cloud-native git pull
```

Cloning the github repo is preferred so that you can pull in updates to the scripts, but if you are not a git user, then you can download the project: https://github.com/lucidworks/fusion-cloud-native/archive/master.zip . Once downloaded, extract the zip and cd into the `fusion-cloud-native-master` directory.

Deployment

If you're not running on a managed K8s platform like GKE, AKS, or EKS, you can use Helm to install the Fusion chart to an existing Kubernetes cluster.

Fusion version 5.5 now includes support for the Rancher Kubernetes Engine (RKE) platform. Before deploying Fusion to RKE, you must download and install the RKE software. After configuring your cluster, you can proceed with the Helm v3 installation.

Note

You must have a working cluster configured before performing the Helm v3 installation.

Use Helm v3 to Install Fusion

You should upgrade to the latest version of Helm v3 for working with Fusion. If you need to keep Helm V2 for other clusters, ensure Helm V3 is ahead of Helm V2 in your working shell's PATH before proceeding.

CUSTOMIZE FUSION CHART SETTINGS

Fusion aims to be well-configured out-of-the-box, but you can customize any of the built-in settings using a custom values `YAML` file. If you use one of our setup scripts, such as `setup_f5_gke.sh`, then it will create a custom values YAML file for you the first time you run it using the `customize_fusion_values.yaml.example` as a template.

If you're working with Helm directly and not using one of our setup scripts, then run the `customize_fusion_values.sh` script to create a custom values YAML file from our `customize_fusion_values.yaml.example` template as a starting point:

```
./customize_fusion_values.sh -c <cluster> -n <namespace> \ --provider
<provider> --num-solr 1 --node-pool "<node_pool>"
```

Note

Pass `--help` for usage details.

In this example:

- `<provider>` is the K8s platform you're running on, such as `gke`
- `<cluster>` is the name of your cluster

- `<namespace>` is the K8s namespace where you plan to install Fusion

> **Note**
>
> The `--node-pool` option specifies the node selector label for determining which nodes to run Fusion pods. You can pass `"{}"` to let Kubernetes decide which nodes to schedule pods on.

This file is referred to as `${MY_VALUES}` in the commands belo. Replace the filename with the correct filename for your environment. Keep this file handy, as you'll need it to customize Fusion settings and upgrade to a newer version.

Review the settings in the custom values YAML file to ensure the defaults are appropriate for your environment, including the number of Solr and Zookeeper replicas.

Add the Lucidworks Helm repo:

```
helm repo add lucidworks https://charts.lucidworks.com
```

The `customize_fusion_values.sh` script creates an upgrade script to install/upgrade Fusion into Kubernetes using Helm. Look in the directory where you ran `customize_fusion_values.sh` for a script named like: `<provider>_<cluster>_<namespace>_upgrade_fusion.sh`. Run this script to install Fusion.

RedHat OpenShift

We can deploy Fusion in an existing OpenShift cluster. This cluster should be created using OpenShift Infrastructure Provider. A Red Hat Customer Portal account is required. OpenShift Online services are not supported.

The easiest way to install on OpenShift is to run the `setup_f5_k8s.sh` script for your **existing** cluster; use the `--help` option to see script usage. For instance, the following command will install Fusion 5 into the specified namespace (`-n`) and OpenShift cluster (`-c`):

```
./setup_f5_k8s.sh -c <CLUSTER> -n <NAMESPACE> --provider oc
```

Tip: `kubectl` should work with your OpenShift cluster (see: https://docs.openshift.com/container-platform/4.1/cli_reference/usage-oc-kubectl.html) and Lucidworks recommends installing the latest `kubectl` for your workstation instead of using `oc` for installing Fusion 5. However, if you do not have `kubectl` installed, then you'll need to update the upgrade script created by `setup_f5_k8s.sh` to use `oc` instead of `kubectl` (search and replace on the BASH script using a text editor).

When you're ready to deploy Fusion to a production-like environment, see more information at Fusion 5 Survival Guide.

Lucidworks recommends using Helm v3, but in case Tiller is required for Helm v2, the cluster security needs to be relaxed to allow images to run with different UIDs:

```
oc adm policy add-scc-to-group anyuid system:authenticated
```

Verifying the Fusion Installation

In this section, we provide some tips on how to verify the Fusion installation.

Tip

Check if the Fusion Admin UI is available at `https://<fusion-host>:6764/admin/`.

Let's review some useful kubectl commands.

Enhance the K8s Command-line Experience

Here is a list of tools we found useful for improving your command-line experience with Kubernetes:

- krew (kubectl plugin mgr): https://github.com/kubernetes-sigs/krew/
- kube-ps1 (show current context on command line prompt): https://github.com/jonmosco/kube-ps1
- kubectx / kubens (switch between clusters / namespaces): https://github.com/ahmetb/kubectx

Useful kubectl commands

kubectl reference: https://kubernetes.io/docs/reference/generated/kubectl/kubectl-commands

Set the namespace for `kubectl` if not using the default:

```
kubectl config set-context --current --namespace=<NAMESPACE>
```

This saves you from having to pass `-n` *with every command.*

Get a list of running pods: `k get pods`

Get logs for a pod using a label: `k logs -l app.kubernetes.io/component=query-pipeline`

Get pod deployment spec and details: `k get pods <pod_id> -o yaml`

Get details about a pod events: `k describe po <pod_id>`

Port forward to a specific pod: `k port-forward <pod_id> 8983:8983`

SSH into a pod: `k exec -it <pod_id> -- /bin/bash`

CPU/Memory usage report for pods: `k top pods`

Forcefully kill a pod: `k delete po <pod_id> --force --grace-period 0`

Scale up (or down) a deployment: `k scale deployment.v1.apps/<id> --replicas=N`

Get a list of pod versions: `k get po -o jsonpath='{..image}' | tr -s '.' '\n' | sort | uniq`

Check Fusion Pods and Services

Once the install script completes, you can check that all pods and services are available using:

```
kubectl get pods
```

If all goes well, you should see a list of pods similar to:

```
NAME READY STATUS RESTARTS AGE seldon-controller-manager-6675874894-
qxwrv 1/1 Running 0 8m45s f5-admin-ui-74d794f4f8-m5jms 1/1 Running 0
8m45s f5-ambassador-fd6b9b5dc-7ghf6 1/1 Running 0 8m43s f5-api-
gateway-6b9998b9c-tmchk 1/1 Running 0 8m45s f5-auth-ui-7565564b4c-
rdc74 1/1 Running 0 8m42s f5-classic-rest-service-0 1/1 Running 3
8m44s f5-devops-ui-77bb867ffb-fbzxd 1/1 Running 0 8m42s f5-fusion-
admin-78b8f8fc7f-4d7l8 1/1 Running 0 8m42s f5-fusion-indexing-
599c8d448-xzsvm 1/1 Running 0 8m44s f5-insights-665fd9f6fc-g5psw 1/1
Running 0 8m43s f5-job-launcher-84dd4c5c96-p8528 1/1 Running 0 8m44s
f5-job-rest-server-6d44d964b8-xtnxw 1/1 Running 0 8m45s f5-logstash-0
1/1 Running 0 8m45s f5-ml-model-service-6987dc94c9-9ppp8 2/2 Running 1
8m45s f5-monitoring-grafana-5d499dbb58-pzw72 1/1 Running 0 10m f5-
monitoring-prometheus-kube-state-metrics-54d6678dv9h7h 1/1 Running 0
10m f5-monitoring-prometheus-pushgateway-7d65c65b85-vwrwf 1/1 Running
0 10m f5-monitoring-prometheus-server-0 2/2 Running 0 10m f5-pm-ui-
86cbc5bb65-nd2n8 1/1 Running 0 8m44s f5-pulsar-bookkeeper-0 1/1
Running 0 8m45s f5-pulsar-broker-b56cc776f-56msx 1/1 Running 0 8m45s
f5-query-pipeline-5d75d7d5f4-l2mdf 1/1 Running 0 8m43s f5-connectors-
7bb6cfc65f-7wfs2 1/1 Running 0 8m42s f5-connectors-backend-987fdc648-
dldwv 1/1 Running 0 8m45s f5-rules-ui-6b9d55b78f-9hzzj 1/1 Running 0
8m43s f5-solr-0 1/1 Running 0 8m44s f5-solr-exporter-c4687c785-jsm7x
1/1 Running 0 8m45s f5-ui-6cdbcc68c6-rj9cq 1/1 Running 0 8m45s f5-
webapps-6d6bb9bfd-hm4qx 1/1 Running 0 8m45s f5-workflow-controller-
7b66679fb7-sjbvp 1/1 Running 0 8m44s f5-zookeeper-0 1/1 Running 0
8m45s
```

The number of pods per deployment / statefulset will vary based on your cluster size and replicaCount settings in your custom values YAML file. Also, don't worry if you see some pods having been restarted as that just means they were too slow to come up and Kubernetes killed and restarted them. You do want to see at least one pod running for

every service. If a pod is not running after waiting a sufficient amount of time, use `kubectl logs <pod_id>` to see the logs for that pod; to see the logs for previous versions of a pod, use: `kubectl logs <pod_id> -p`. You can also look at the actions Kubernetes performed on the pod using `kubectl describe po <pod_id>`.

To see a list of Fusion services, do:

```
kubectl get svc
```

For an overview of the various Fusion 5 microservices, see: Fusion microservices.

Once you're ready to build a Fusion cluster for production, please see see more information at Fusion 5 Survival Guide.

Upgrading with Zero Downtime

One of the most powerful features provided by Kubernetes and a cloud-native microservices architecture is the ability to do a rolling update on a live cluster. Fusion 5 allows customers to upgrade from Fusion 5.x.y to a later 5.x.z version on a live cluster with zero downtime or disruption of service.

When Kubernetes performs a rolling update to an individual microservice, there will be a mix of old and new services in the cluster concurrently (only briefly in most cases) and requests from other services will be routed to both versions. Consequently, Lucidworks ensures all changes we make to our service do not break the API interface exposed to other services in the same 5.x line of releases. We also ensure stored configuration remains compatible in the same 5.x release line.

Lucidworks releases minor updates to individual services frequently, so our customers can pull in those upgrades using Helm at their discretion.

To upgrade your cluster at any time, use the `--upgrade` option with our setup scripts in this repo.

The scripts in this repo automatically pull in the latest chart updates from our Helm repository and deploy any updates needed by doing a diff of your current installation and the latest release from Lucidworks. To see what would be upgraded, you can pass the `--dry-run` option to the script.

Grafana Dashboards

Get the initial Grafana password from a K8s secret by doing:

```
kubectl get secret --namespace "${NAMESPACE}" ${RELEASE}-monitoring-
grafana \ -o jsonpath="{.data.admin-password}" | base64 --decode ;
echo
```

With Grafana, you can either setup a temporary port-forward to a Grafana pod or expose Grafana on an external IP using a K8s LoadBalancer. To define a LoadBalancer, do (replace ${RELEASE} with your Helm release label):

```
kubectl expose deployment ${RELEASE}-monitoring-grafana --
type=LoadBalancer --name=grafana --port=3000 --target-port=3000
```

You can use `kubectl get services --namespace <namespace>` to determine when the load balancer is setup and its IP address. Direct your browser to http://<GrafanaIP>:3000 and enter the username `admin@localhost` and the password that was returned in the previous step.

This will log you into the application. It is recommended that you create another administrative user with a more desirable password.

The dashboards and datasoure will be setup for you in grafana, simply navigate to `Dashboards` → `Manage` to view the vailable dashboards

More deployment options

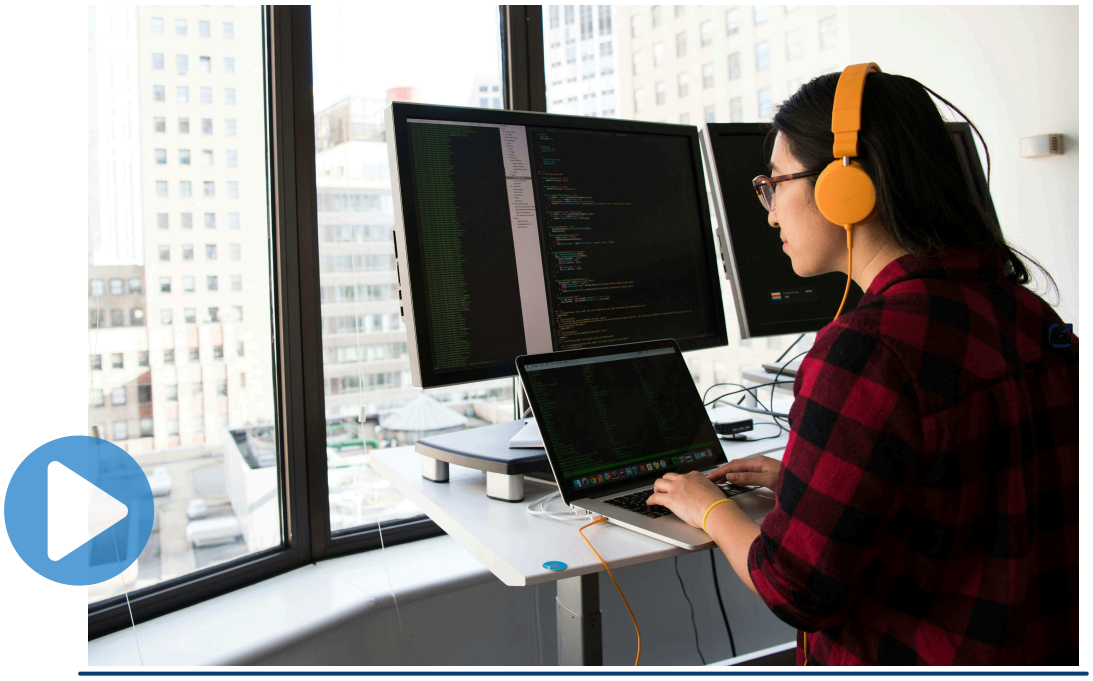
- *How to deploy Fusion 5 in Google Kubernetes Engine*
- *How to deploy Fusion 5 in Amazon Elastic Kubernetes Service*
- *How to deploy Fusion 5 in Azure Kubernetes Service*

Additional resources

LucidAcademy

Lucidworks offers free training to help you get started.

The **Course** for **Deploying Fusion 5** focuses on the prerequisite software needed to deploy Fusion, the necessary setup steps, and the physical act of deployment:



Visit the LucidAcademy to see the full training catalog.

Deploy Fusion at Scale

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion*

Before you begin, see *Fusion Server Deployment* to understand the architecture and requirements.

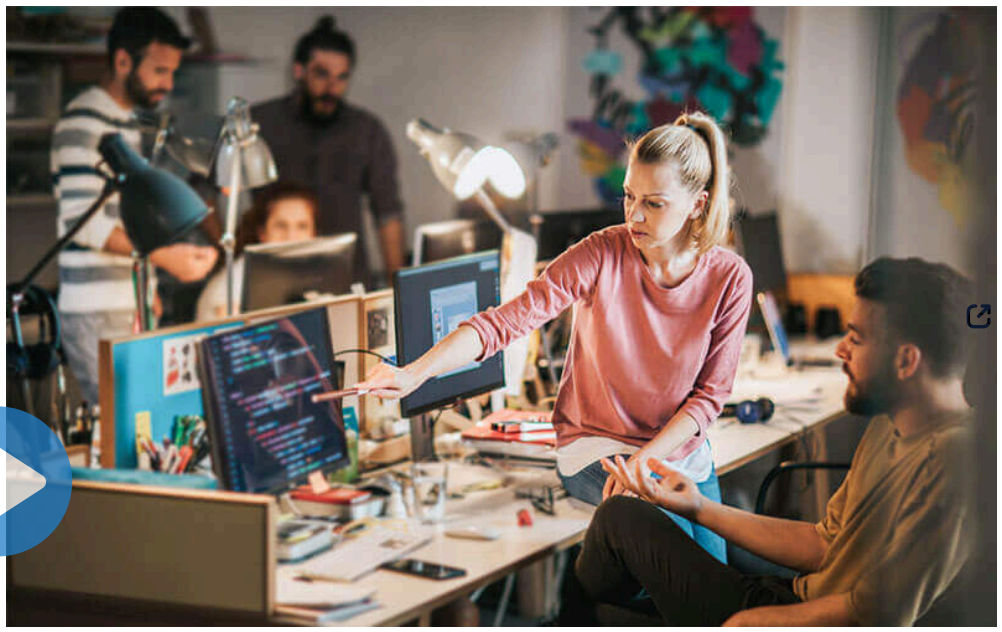
This article explains how to plan and execute a Fusion deployment at the scale required for staging or production.

While the `setup_f5_*.sh` scripts are handy for getting started and proof-of-concept purposes, this article covers the planning process for building a production-ready environment.

LucidAcademy

Lucidworks offers free training to help you get started.

The **Course** for **Preparing for Fusion Implementation** focuses on the key elements for a successful implementation, defining your business requirements, preparing clean data, and involving the right personnel:



Visit the LucidAcademy to see the full training catalog.

Prerequisites

You must meet the following prerequisites before you can customize your Fusion cluster:

- A local copy of the fusion-cloud-native repository. This must be up-to-date with the latest master branch.
- Any cloud provider-specific **command line tools**, such as `gcloud` or `aws`, and `kubectl`. See the platform-specific instructions linked above, or check with your cloud provider.
- Helm v3
    - To install on a Mac:

```
brew upgrade kubernetes-helm
```

    - For other operating systems, download from Helm Releases.
    - Verify your installation:

```
helm version --short v3.0.0+ge29ce2a
```

- Kubernetes namespace
    - Collect the following information about your Kubernetes environment:
        - **CLUSTER**: Cluster name (passed to our setup scripts using the `-c` arg)
        - **NAMESPACE**: Kubernetes namespace where to install Fusion; a namespace should only contain lowercase letters (a-z), digits (0-9), or dash. No periods or underscores allowed.

- *(optional)* Clarify your organization's DockerHub policy. The Fusion Helm chart points to public Docker images on DockerHub. Your organization may not allow Kubernetes to pull images directly from DockerHub or may require extra security scanning before loading images into production clusters. Consult your Kubernetes and Docker admin team to find how to get the Fusion images loaded into a registry that's accessible to your cluster. You can update the image for each service using the *custom values YAML file*.

Tip

Kubernetes namespace tips

- Fusion 5 service discovery requires all services for the same release be deployed in the same namespace. Moreover, you should only run one instance of Fusion in a namespace. If you need multiple instances of Fusion running in the same Kubernetes cluster, then you need to deploy them in separate namespaces.
- If your organization requires CPU / Memory quotas for namespaces, you can start with a minimum of 12 CPU and 45GB of RAM (such as 3 x n1-standard-4 on GKE), but you will need to increase the quotas once you start load testing Fusion with production workloads and real datasets.
- Fusion requires at least 3 ZooKeeper nodes and 2 Solr nodes to achieve high availability.

Custom values YAML file

1. Clone the `fusion-cloud-native` repository: `git clone https://github.com/lucidworks/fusion-cloud-native`
2. Run the `customize_fusion_values.sh` script.

```
./customize_fusion_values.sh --provider <provider> -c <cluster>
-n <namespace> \ --num-solr 3 \ --solr-disk-gb 100 \ --node-pool
<node_selector> \ --prometheus true \ --with-resource-limits \ -
-with-affinity-rules
```

Tip

Pass the `--help` parameter to see script usage details.

The script creates the following files:

File	Description
`<provider>_<cluster>_<namespace>_fusion_values.yaml`	Main custom values YAML used to override Helm chart defaults for Fusion microservices.
`<provider>_<cluster>_<namespace>_monitoring_values.yaml`	Custom values yaml used to configure Prometheus and Grafana.
`<provider>_<cluster>_<namespace>_fusion_resources.yaml`	Resource requests and limits for all Microservices.
`<provider>_<cluster>_<namespace>_fusion_affinity.yaml`	Pod affinity rules to ensure mulitple replicas for a single service are evenly distributed across zones and nodes.
`<provider>_<cluster>_<namespace>_upgrade_fusion.sh`	Script used to install and/or upgrade Fusion using the aforementioned custom values YAML files.

For an explanation of these placeholder values, see *Configuration Values* below.

3. Add the new files to version control. You will make changes to it over time as you fine-tune your Fusion installation. You will also need it to perform upgrades. If you try to upgrade your Fusion installation and don't provide the custom values YAML, your deployment will revert to chart defaults.

Review the `<provider>_<cluster>_<release>_fusion_values.yaml` file to familiarize yourself with its structure and contents. Notice it contains a separate section for each of the Fusion microservices. The example configuration of the `query-pipeline` service below illustrates some important concepts about the custom values YAML file.

```
query-pipeline: 1 enabled: true 2 nodeSelector: 3
cloud.google.com/gke-nodepool: default-pool javaToolOptions: "..." 4
pod: 5 annotations: prometheus.io/port: "8787" prometheus.io/scrape:
"true" prometheus.io/path: "/actuator/prometheus"
```

1	Service-specific setting overrides under the top-level heading
2	Every Fusion service has an implicit enabled flag that defaults to true, set to false to remove this service from your cluster
3	Node selector identifies the label find nodes to schedule pods on
4	Used to pass JVM options to the service
5	Pod annotations to allow Prometheus to scrape metrics from the service

Once we go through all of the configuration topics in this topic, you'll have a well-configured custom values YAML file for your Fusion 5 installation. You'll then use this file during the Helm v3 installation at the end of this topic.

Deployment-specific values

The script creates a custom values YAML file using the naming convention: `<provider>_<cluster>_<namespace>_fusion_values.yaml`. For example, `gke_search_f5_fusion_values.yaml`.

Parameter	Description
`<provider>`	The K8s platform you're running on, such as `gke`.
`<cluster>`	The name of your cluster.
`<namespace>`	The K8s namespace where you want to install Fusion.
`<node_selector>`	Specifies a `nodeSelector` label to find nodes to schedule Fusion pods on.

Warning

Providing the correct `--node-pool <node_selector>` label is very important. Using the wrong value will cause your pods to be stuck in the `pending` state. If you're not sure about the correct value for your cluster, pass `'{}'` to let Kubernetes decide which nodes to schedule Fusion pods on.

Default `nodeSelector` labels are provider-specific. The `fusion-cloud-native` scripts use the following defaults for GKE and EKS:

Provider	Default node selector
GKE	cloud.google.com/gke-nodepool: default-pool
EKS	alpha.eksctl.io/nodegroup-name: standard-workers

Caution

If you are deploying Fusion 5.9.12, add the following to your `values.yaml` file to avoid a known issue that prevents the `kuberay-operator` pod from launching successfully:

```
kuberay-operator: crd: create: true
```

Flags

The script provides flags for additional configuration:

Flag	Description
`--node-pool`	Add a Fusion specific label to your nodes.
`--with-resource-limits`	Configure resource requests/limits.
`--with-replicas`	Configure replica counts.
`--with-affinity-rules`	Configure pod affinity rules for Fusion services.

Use `--node-pool` to add a Fusion specific label to your nodes by doing:

```
kubectl label <NODE_ID> fusion_node_type=<NODE_LABEL>
```

Then, pass `--node-pool 'fusion_node_type: <NODE_LABEL>'`.

Configure Solr sizing

When you're ready to build a production-ready setup for Fusion 5, you need to customize the Fusion Helm chart to ensure Fusion is well-configured for production workloads.

You'll be able to scale the number of nodes for Solr up and down after building the cluster, but you need to establish the initial size of the nodes (memory and CPU) and the size and type of disks you need.

See the example config below to learn which parameters to change in the custom values YAML file.

```
solr: resources: # Set resource limits for Solr to help K8s pod
scheduling; limits: # these limits are not just for the Solr process
in the pod, cpu: "7700m" # so allow ample memory for loading index
files into the OS cache (mmap) memory: "26Gi" requests: cpu: "7000m"
memory: "25Gi" logLevel: WARN nodeSelector: fusion_node_type: search #
Run this Solr StatefulSet in the "search" node pool exporter: enabled:
true # Enable the Solr metrics exporter (for Prometheus) and #
schedule on the default node pool (system partition) podAnnotations:
prometheus.io/scrape: "true" prometheus.io/port: "9983"
prometheus.io/path: "/metrics" nodeSelector: cloud.google.com/gke-
nodepool: default-pool image: tag: 8.4.1 updateStrategy: type:
"RollingUpdate" javaMem: "-Xmx3g -Dfusion_node_type=system" #
Configure memory settings for Solr solrGcTune: "-XX:+UseG1GC -XX:-
OmitStackTraceInFastThrow -XX:+UseStringDeduplication -
XX:+PerfDisableSharedMem -XX:+ParallelRefProcEnabled -
XX:MaxGCPauseMillis=150 -XX:+UseLargePages -XX:+AlwaysPreTouch"
volumeClaimTemplates: storageSize: "100Gi" # Size of the Solr disk
replicaCount: 6 # Number of Solr pods to run in this StatefulSet
zookeeper: nodeSelector: cloud.google.com/gke-nodepool: default-pool
replicaCount: 3 # Number of Zookeepers persistence: size: 20Gi
resources: {} env: ZK_HEAP_SIZE: 1G ZOO_AUTOPURGE_PURGEINTERVAL: 1
```

To be clear, you can tune GC settings and number of replicas after the cluster is built. But changing the size of the persistent volumes is more complicated so you should try to pick a good size initially.

Configure storage class for Solr pods (optional)

If you wish to run with a storage class other than the default you can create a storage class for your Solr pods before you install. For example, to create regional disks in GCP you can create a file called **`storageClass.yaml`** with the following contents:

```
kind: StorageClass apiVersion: storage.k8s.io/v1 metadata: name: solr-
gke-storage-regional provisioner: kubernetes.io/gce-pd parameters:
type: pd-standard replication-type: regional-pd zones: us-west1-b, us-
west1-c
```

and then provision into your cluster by calling:

```
kubectl apply -f storageClass.yaml
```

to then have Solr use the storage class by adding the following to the custom values YAML:

```
solr: volumeClaimTemplates: storageClassName: solr-gke-storage-
regional storageSize: 250Gi
```

Note

We're not advocating that you must use regional disks for Solr storage, as that would be redundant with Solr replication. We're just using this as an example of how to configure a custom storage class for Solr disks if you see the need. For instance, you could use regional disks without Solr replication for write-heavy type collections.

Configure multiple node pools

Lucidworks recommends isolating search workloads from analytics workloads using multiple node pools. The included scripts do not do this for you; this is a manual process.

See the example script for GKE, see create_gke_cluster_node_pools.sh .

In the custom values YAML file, you can add additional Solr StatefulSets by adding their names to the list under the `nodePools` property. If any property for that statefulset needs to be changed from the default set of values, then it can be set directly on the object representing the node pool, any properties that are omitted are defaulted to the base value. See the following example (additional whitespace added for display purposes only):

```
solr: nodePools: - name: ""  1  - name: "analytics"  2  javaMem: "-
Xmx6g" replicaCount: 6 storageSize: "100Gi" nodeSelector:
fusion_node_type: analytics  3  resources: requests: cpu: 2 memory:
12Gi limits: cpu: 3 memory: 12Gi - name: "search"  4  javaMem: "-Xms11g
-Xmx11g" replicaCount: 12 storageSize: "50Gi" nodeSelector:
fusion_node_type: search  5  resources: limits: cpu: "7700m" memory:
"26Gi" requests: cpu: "7000m" memory: "25Gi" nodeSelector:
cloud.google.com/gke-nodepool: default-pool  6  ...
```

1	The empty string `""` is the suffix for the default partition.
2	Overrides the settings for the **analytics** Solr pods.
3	Assigns the **analytics** Solr pods to the node pool and attaches the label `fusion_node_type=analytics`. You can use the `fusion_node_type` property in Solr auto-scaling policies to govern replica placement during collection creation.
4	Overrides the settings for the **search** Solr pods
5	Assigns the **search** Solr pods to the node pool and attaches the label `fusion_node_type=search`.
6	Sets the default settings for all Solr pods, if not specifically overridden in the `nodePools` section above.

Warning

Do not edit the **`nodePools`** value **`""`**.

In the example above, the **analytics** partition **`replicaCount`**, or number of Solr pods, is six. The **search** partition **`replicaCount`** is twelve.

Each nodePool is automatically be assigned the -Dfusion_node_type property of **`<search>`**, **`<system>`**, or **`<analytics>`**. This value matches the name of the nodePool. For example, **`-Dfusion_node_type=<search>`**.

The Solr pods have a **`fusion_node_type`** system property, as shown below:

Args

```
-DSTOP.KEY=solrrocks
-DSTOP.PORT=7983
-Dfusion_node_type=search
-Dhost=search-f5-solr-5.search-f5-solr-headless
-Djetty.home=/opt/solr/server
-Djetty.port=8983
-Dlog4j.configurationFile=file:/var/solr/log4j2.xml
-Dsolr.data.home=
-Dsolr.default.confdir=/opt/solr/server/solr/configsets/_default/conf
-Dsolr.install.dir=/opt/solr
-Dsolr.jetty.https.port=8983
-Dsolr.log.dir=/var/solr/logs
-Dsolr.log.level=INFO
-Dsolr.solr.home=/var/solr/data
-Dsolr_zone=us-west1-a
```

Solr auto-scaling policy

```
This section applies only to Fusion 5.7 and earlier. For Fusion 5.8 and
later, use link:https://solr.apache.org/guide/solr/latest/configuration-
guide/replica-placement-plugins.html[replica placement plugins^] to
control how replicas are placed in Solr.
```

You can configure a custom Solr auto-scaling policy in the custom values YAML file under the **`fusion-admin`** section, as shown below:

```
fusion-admin: ... solrAutocalingPolicyJson: { "set-cluster-policy": [
{"node": "#ANY", "shard": "#EACH", "replica":"<2"}, {"replica":
"#EQUAL", "sysprop.solr_zone": "#EACH", "strict" : false} ] }
```

You can use an auto-scaling policy to govern how the shards and replicas for Fusion system and application-specific collections are configured. If your cluster defines the **search**, **analytics**, and **system** node pools, use the `policy.json` file provided in the fusion-cloud-native repo as a starting point. The Fusion Admin service applies the policy from the custom values YAML file to Solr, before creating system collections during initialization.

Pod network policy

A Kubernetes network policy governs how groups of pods communicate with each other and other network endpoints. With Fusion, all incoming traffic flows through the API Gateway service. All Fusion services in the same namespace expect an internal JWT, which is supplied by the Gateway, as part of the request. As a result, Fusion services enforce a basic level of API security and don't need an additional network policy to protect them from other pods in the cluster.

To install the network policy for Fusion services, pass `--set global.networkPolicyEnabled=true` when installing the Fusion Helm chart.

On-premises private Docker registries

For on-premises Kubernetes deployments, your organization may not allow Kubernetes to pull Fusion's Docker images from DockerHub. See the instructions below for details on using a private Docker registry with Fusion 5.1.1. These are general instructions that may need to be adapted to work within your organization's security policies:

1. Transfer the public images from DockerHub to your private Docker registry.
2. Establish a workstation that has access to DockerHub. This workstation must connect to your internal Docker registry, most likely via VPN connection. In this example, the workstation is referred to as `envoy`.
3. Install Docker on `envoy`. You need at least 100GB of free disk for Docker.
4. Pull all of the images from DockerHub to `envoy`'s local registry. For example, to pull the query pipeline image, run `docker pull lucidworks/query-pipeline:5.1.1`. This will take a long time. See the list of the images required below to pull for the Fusion 5.1.1 chart:

```
apachepulsar/pulsar-all:2.5.0 argoproj/argoui:v2.4.3
argoproj/workflow-controller:v2.4.3 bitnami/kubectl:1.15-debian-
9 busybox:1.31.1 busybox:latest curlimages/curl:7.68.0
docker.io/seldonio/seldon-core-operator:1.0.1
grafana/grafana:6.6.2 jimmidyson/configmap-reload:v0.3.0
lucidworks/admin-ui:5.1.1 lucidworks/api-gateway:5.1.1
lucidworks/auth-ui:5.1.0 lucidworks/classic-rest-service:5.1.1
lucidworks/devops-ui:5.1.0 lucidworks/fusion-api:5.1.1
lucidworks/fusion-indexing:5.1.1 lucidworks/fusion-
logstash:5.1.0 lucidworks/insights:5.1.0 lucidworks/job-
launcher:5.1.1 lucidworks/job-rest-server:5.1.1 lucidworks/ml-
model-service:5.1.0 lucidworks/ml-python-image:5.1.0
lucidworks/pm-ui:5.1.0 lucidworks/query-pipeline:5.1.1
lucidworks/rest-service:5.1.1 lucidworks/rpc-service:5.1.1
lucidworks/rules-ui:5.1.0 lucidworks/webapps:5.1.0
prom/prometheus:v2.16.0 prom/pushgateway:v1.0.1
quay.io/coreos/kube-state-metrics:v1.9.5
quay.io/datawire/ambassador:0.86.1 seldonio/seldon-core-
operator:1.0.1 solr:8.4.1 zookeeper:3.5.6
```

See **`docker pull --help`** for more information about pulling Docker images.

5. Establish a connection from **`envoy`** to the private Docker registry, most likely via a VPN connection. In this example, the private Docker registry is referred to as **`<internal-private-registry>`**.
6. Push the images from **`envoy`**'s Docker registry to the private registry. This will take a long time.
    a. You'll need to re-tag all images for the internal registry. For example, to tag the query-pipeline image, run:

```
docker tag lucidworks/query-pipeline:5.1.1 <internal-
private-registry>/query-pipeline:5.1.1
```

    b. Push each image to the internal repo:

```
docker push <internal-private-registry>/query-
pipeline:5.1.1
```

7. Install the Docker registry secret in Kubernetes. Create the Docker registry secret in the Kubernetes namespace where you want to install Fusion:

```
SECRET_NAME=<internal-private-secret> REPO=<internal-private-
registry> kubectl create secret docker-registry "${SECRET_NAME}"
\ --namespace "${NAMESPACE}" \ --docker-server="${REPO}" \ --
docker-username=${REPO_USER} \ --docker-password=${REPO_PASS} \
--docker-email=${REPO_USER}
```

   For details, see the Kubernetes article Pull an Image from a Private Registry .

8. Update the custom values YAML for your cluster to point to your private registry and secret to allow Kubernetes to pull images. For example:

```
query-pipeline: image: imagePullSecrets: - name: <internal-
private-secret> repository: <internal-private-registry>
```

   Repeat the process for all Fusion services.

Customize Helm Chart

Every Fusion service allows you to override the `imagePullSecrets` setting using custom values YAML. However, other 3rd party services — including Zookeeper, Pulsar, Prometheus, and Grafana — don't allow you to supply the pull secret using the custom values YAML.

To patch the default service account for your namespace and add the pull secret, run the following:

```
kubectl patch sa default -n $NAMESPACE \ -p '"imagePullSecrets":
[{"name": "<internal-private-secret>" }]'
```

In Windows using PowerShell or another CLI, you might have to escape the double quotes with a backslash (`\`) or reverse the order of single and double quotes:

```
kubectl patch sa default -n $NAMESPACE \ -p "'imagePullSecrets':
[{'name': '<internal-private-secret>'}]"
```

Important

Replace `<internal-private-secret>` with the name of the secret you created in the steps above.

This allows the default service account to pull images from the private registry without specifying the pull secret on the resources directly.

Add additional trusted certificate(s) to Fusion's indexing and querying services (optional)

You can add custom trusted certificates to support Fusion's indexing and querying services. You may want to use custom trusted certificates if, for example, you have specific security requirements for data handling or need to support an existing infrastructure and its security needs. This method involves updating your Helm chart.

If you want to add custom trusted certificates for both the indexing and querying services, follow these instructions twice: once for the indexing service, and once for the querying service. To add different certificates to the indexing and querying services, create one YAML file with the indexing service certificates and one YAML file for the querying service certificates before following these instructions.

Tip

You may use the same YAML file if you want to use the same certificates for both services.

To add custom trusted certificates:

1. Create a new YAML file for your custom trusted certificates. The `customcerts.yaml` file is the example file in these instructions.
2. Add the custom certificate(s) in the YAML file created in the previous step. For example:

```
trustedCertificates: enabled: true files: some.cert: |-
      -----BEGIN CERTIFICATE-----

MIIDeTCCAmGgAwIBAgIJAPziuikCTox4MA0GCSqGSIb3DQEBCwUAMGIxCzAJBgNV
      (...)
      EVA0pmzIzgBg+JIe3PdRy27T0asgQW/F4TY61Yk=
      -----END CERTIFICATE-----
other.cert: |-
      -----BEGIN CERTIFICATE-----

MIIDeTCCAmGgAwIBAgIJAPziuikCTox4MA0GCSqGSIb3DQEBCwUAMGIxCzAJBgNV
      (...)
      EVA0pmzIzgBg+JIe3PdRy27T0asgQW/F4TY61Yk=
      -----END CERTIFICATE---------
```

3. Update the indexing or querying service by running the following Helm command. Replace **`EXAMPLE-VALUES-FILE.yaml`** with your previous values file.
   ```
   helm upgrade --install --namespace ${EXAMPLE-NAMESPACE} ${HELM-RELEASE} ${HELM-CHART-PATH} --values EXAMPLE-VALUES-FILE.yaml --values customcerts.yaml
   ```
4. Verify the indexing or querying pod has a new **`init-container`** with the name **`import-certs`**.

Add additional trusted certificate(s) for connectors to allow crawling of web resources with SSL/TLS enabled (optional)

To crawl a datasource which for some reason is using a self-signed certificate, add arbitrary certificates to connectors. For example:

Fusion 5.3x, 5.2x, and 5.1x

```
classic-rest-service: trustedCertificates: enabled: true files:
some.cert: |-
        -----BEGIN CERTIFICATE-----

MIIDeTCCAmGgAwIBAgIJAPziuikCTox4MA0GCSqGSIb3DQEBCwUAMGIxCzAJBgNV
        (...)
        EVA0pmzIzgBg+JIe3PdRy27T0asgQW/F4TY61Yk=
        -----END CERTIFICATE-----
other.cert: |-
        -----BEGIN CERTIFICATE-----

MIIDeTCCAmGgAwIBAgIJAPziuikCTox4MA0GCSqGSIb3DQEBCwUAMGIxCzAJBgNV
        (...)
        EVA0pmzIzgBg+JIe3PdRy27T0asgQW/F4TY61Yk=
        -----END CERTIFICATE---------
connector-plugin-service: trustedCertificates: enabled: true files:
some.cert: |-
        -----BEGIN CERTIFICATE-----

MIIDeTCCAmGgAwIBAgIJAPziuikCTox4MA0GCSqGSIb3DQEBCwUAMGIxCzAJBgNV
        (...)
        EVA0pmzIzgBg+JIe3PdRy27T0asgQW/F4TY61Yk=
        -----END CERTIFICATE-----
other.cert: |-
        -----BEGIN CERTIFICATE-----

MIIDeTCCAmGgAwIBAgIJAPziuikCTox4MA0GCSqGSIb3DQEBCwUAMGIxCzAJBgNV
        (...)
        EVA0pmzIzgBg+JIe3PdRy27T0asgQW/F4TY61Yk=
        -----END CERTIFICATE---------
```

Fusion 5.4 and later

```
classic-rest-service: trustedCertificates: enabled: true files:
some.cert: |-
        -----BEGIN CERTIFICATE-----

MIIDeTCCAmGgAwIBAgIJAPziuikCTox4MA0GCSqGSIb3DQEBCwUAMGIxCzAJBgNV
        (...)
        EVA0pmzIzgBg+JIe3PdRy27T0asgQW/F4TY61Yk=
        -----END CERTIFICATE-----
other.cert: |-
        -----BEGIN CERTIFICATE-----

MIIDeTCCAmGgAwIBAgIJAPziuikCTox4MA0GCSqGSIb3DQEBCwUAMGIxCzAJBgNV
        (...)
        EVA0pmzIzgBg+JIe3PdRy27T0asgQW/F4TY61Yk=
        -----END CERTIFICATE---------
connector-plugin: trustedCertificates: enabled: true files: some.cert:
|-
        -----BEGIN CERTIFICATE-----

MIIDeTCCAmGgAwIBAgIJAPziuikCTox4MA0GCSqGSIb3DQEBCwUAMGIxCzAJBgNV
        (...)
        EVA0pmzIzgBg+JIe3PdRy27T0asgQW/F4TY61Yk=
        -----END CERTIFICATE-----
other.cert: |-
        -----BEGIN CERTIFICATE-----

MIIDeTCCAmGgAwIBAgIJAPziuikCTox4MA0GCSqGSIb3DQEBCwUAMGIxCzAJBgNV
        (...)
        EVA0pmzIzgBg+JIe3PdRy27T0asgQW/F4TY61Yk=
        -----END CERTIFICATE---------
```

Generating the certificate on linux command line

Use the following command to generate a `.crt` file in `$fusion_home/apps/jetty/connectors/etc/yourcertname.crt`:

```
openssl s_client -servername remote.server.net -connect remote.server.net:443
</dev/null | sed -ne '/-BEGIN CERTIFICATE-/,/-END CERTIFICATE-/p'
>$fusion_home/apps/jetty/connectors/etc/yourcertname.crt
```

Generating the certificate using Firefox web browser

1. Navigate to the Sharepoint host.
2. Click the in the address bar, then click the icon.
3. Next, navigate to **More Information** > **View Certificate** > **Export**.
   Save the file to the following folder:

```
$fusion_home/apps/jetty/connectors/etc/yourcertname.crt
```

Generating the certificate using Chrome web browser

1. Navigate to **Chrome menu** > **More Tools** > **Developer Tools** > **Security Tab**.
   This will display the **Security overview**.
2. Click the **View certificate** button.
3. Save the file to the following folder:

```
$fusion_home/apps/jetty/connectors/etc/yourcertname.crt
```

Generating the certificate using powershell

Use the following script to generate a `.crt` file in `$fusion_home\apps\jetty\connectors\etc\yourcertname.crt`:

```
$fusion_home = c:\your\fusion\install\directory $webRequest =
[Net.WebRequest]::Create("https://your-hostname") try {
$webRequest.GetResponse() } catch {} $cert =
$webRequest.ServicePoint.Certificate $bytes =
$cert.Export([Security.Cryptography.X509Certificates.X509ContentType]:
:Cert) set-content -value $bytes -encoding byte -path
"$fusion_home\apps\jetty\connectors\etc\yourcertname.binary.crt"
certutil -encode
"$fusion_home\apps\jetty\connectors\etc\yourcertname.binary.crt"
"$fusion_home\apps\jetty\connectors\etc\yourcertname.crt" rm
"$fusion_home\apps\jetty\connectors\etc\yourcertname.binary.crt" -f
```

Install Fusion 5 on Kubernetes

At this point, you're ready to install Fusion 5 using the custom values YAML files and upgrade script. If you used the `customize_fusion_values.sh` script, run it using BASH:

```
./gke_search_f5_upgrade_fusion.sh
```

Once the installation is complete, verify your Fusion installation is running correctly.

Monitoring Fusion with Prometheus and Grafana

Lucidworks recommends using Prometheus and Grafana for monitoring the performance and health of your Fusion cluster. Your operations team may already have these services installed. If not, install them into the Fusion namespace.

Note

The *Custom values YAML file shown above* activates the Solr metrics exporter service and adds pod annotations so Prometheus can scrape metrics from Fusion services.

1. Run the `customize_fusion_values.sh` script with the `--prometheus true` option. This creates an extra custom values YAML file for installing Prometheus and Grafana, `<provider>_<cluster>_<namespace>_monitoring_values.yaml`. For example: `gke_search_f5_monitoring_values.yaml`.
2. Commit the YAML file to version control.
3. Review its contents to ensure that the settings suit your needs. For example, decide how long you want to keep metrics. The default is 36 hours.
   See the Prometheus documentation and Grafana documentation for details.
4. Run the `install_prom.sh` script to install Prometheus & Grafana in your namespace. Include the provider, cluster name, namespace, and helm release as in the example below:

```
./install_prom.sh --provider gke -c search -n f5 -r 5-5-1
```

Tip

Pass the `--help` parameter to see script usage details.

The Grafana dashboards from monitoring/grafana are installed automatically by the `install_prom.sh` script.

Deploy the sentiment-general Model

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion*

This topic explains how to deploy the sentiment-general pre-trained *sentiment prediction* model. This is a general-purpose sentiment prediction model, trained on short sentences. It is suitable for short texts and for intent prediction.

See also the *sentiment-reviews model deployment*.

1. Install the model in Fusion

1. Navigate to **Collections** > **Jobs**.
2. Select **New** > **Create Seldon Core Model Deployment**.
3. Configure the job as follows:
    - **Job ID.** The ID for this job, such as `deploy-sentiment-general`.
    - **Model Name.** The model name of the Seldon Core deployment that will be referenced in the Machine Learning pipeline stage configurations, such as `sentiment-general`.
    - **Docker Repository.** The value is `lucidworks`.
    - **Image Name.** The value is `sentiment-general:v1.0`.
    - **Kubernetes Secret Name for Model Repo.** This value is left empty.
    - **Output Column Names for Model.** The value is `[label, score]`.
4. Click **Save**.
5. Click **Run > Start**.

Note

If you are running a private Docker repository that virtualizes external images, enter your private registry into the **Docker Repository** field and `lucidworks/sentiment-general:v1.0` into the **Image Name** field.

2. Configure the Machine Learning pipeline stages

In your index or query pipelines add Machine Learning stage and specify sentiment-general in the Model ID field (or a custom model name that was used during deployment).

2.1. Configure the Machine Learning index stage

1. In your index pipeline, click **Add a Stage** > **Machine Learning**.
2. In the **Model ID** field, enter the model name you configured above, such as `sentiment-general`.
3. In the **Model input transformation script** field, enter the following:

```
var modelInput = new java.util.HashMap() modelInput.put("text",
doc.getFirstFieldValue("text")) modelInput
```

4. In the **Model output transformation script** field, enter the following:

```
doc.addField("sentiment_label_s", modelOutput.get("label")[0])
doc.addField("sentiment_score_d", modelOutput.get("score")[0])
```

5. Save the pipeline.

2.2. Configure the Machine Learning query stage

1. In your query pipeline, click **Add a Stage** > **Machine Learning**.
2. In the **Model ID** field, enter the model name you configured above, such as sentiment-general.
3. In the **Model input transformation script** field, enter the following:

```
var modelInput = new java.util.HashMap() modelInput.put("text",
request.getFirstParam("q")) modelInput
```

4. In the **Model output transformation script** field, enter the following:

```
// To put into request request.putSingleParam("sentiment_label",
modelOutput.get("label")[0])
request.putSingleParam("sentiment_score",
modelOutput.get("score")[0]) // To put into query context
context.put("sentiment_label", modelOutput.get("label")[0])
context.put("sentiment_score", modelOutput.get("score")[0]) //
To put into response documents. NOTE: This can be done only
after Solr Query stage var docs =
response.get().getInnerResponse().getDocuments(); var ndocs =
new java.util.ArrayList(); for (var i=0; i<docs.length;i++){ var
doc = docs[i]; doc.putField("query_sentiment_label",
modelOutput.get("label")[0])
doc.putField("query_sentiment_score", modelOutput.get("score")
[0]) ndocs.add(doc); }
response.get().getInnerResponse().updateDocuments(ndocs);
```

5. Save the pipeline.

Model output

Both of the pre-trained models output the following:

- a **label**, `negative` or `positive`
- a **score** from `-2` to `2`

The sentiment-reviews:v1.0 model also optionally outputs tokens and their corresponding attention weights, that is, the weight that each token carries in the sentiment prediction. The total of all attention weights is always 1, that is, each value represents a percentage of the total weight. In the example below, "awesome" has the highest weight because it expresses the strongest sentiment compared to other tokens in the string:

```
// Input text = "That is awesome!" // Output sentiment_label =
‘positive’ sentiment_score = 1.998 sentiment_attention_tokens =
['That', "'", 's', 'awesome', '!'] sentiment_attention_weights =
[0.154, 0.078, 0.069, 0.444, 0.255]
```

Deploy the sentiment-general Model

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

This topic explains how to deploy the sentiment-general pre-trained *sentiment prediction* model. This is a general-purpose sentiment prediction model, trained on short sentences. It is suitable for short texts and for intent prediction.

See also the *sentiment-reviews model deployment*.

1. Install the model in Managed Fusion

1. Navigate to **Collections** > **Jobs**.
2. Select **New** > **Create Seldon Core Model Deployment**.
3. Configure the job as follows:
    - **Job ID.** The ID for this job, such as `deploy-sentiment-general`.
    - **Model Name.** The model name of the Seldon Core deployment that will be referenced in the Machine Learning pipeline stage configurations, such as `sentiment-general`.
    - **Docker Repository.** The value is `lucidworks`.
    - **Image Name.** The value is `sentiment-general:v1.0`.
    - **Kubernetes Secret Name for Model Repo.** This value is left empty.
    - **Output Column Names for Model.** The value is `[label, score]`.
4. Click **Save**.
5. Click **Run > Start**.

2. Configure the Machine Learning pipeline stages

In your index or query pipelines add Machine Learning stage and specify sentiment-general in the Model ID field (or a custom model name that was used during deployment).

2.1. Configure the Machine Learning index stage

1. In your index pipeline, click **Add a Stage** > **Machine Learning**.
2. In the **Model ID** field, enter the model name you configured above, such as `sentiment-general`.
3. In the **Model input transformation script** field, enter the following:

```
var modelInput = new java.util.HashMap() modelInput.put("text",
doc.getFirstFieldValue("text")) modelInput
```

4. In the **Model output transformation script** field, enter the following:

```
doc.addField("sentiment_label_s", modelOutput.get("label")[0])
doc.addField("sentiment_score_d", modelOutput.get("score")[0])
```

5. Save the pipeline.

2.2. Configure the Machine Learning query stage

1. In your query pipeline, click **Add a Stage** > **Machine Learning**.
2. In the **Model ID** field, enter the model name you configured above, such as sentiment-general.
3. In the **Model input transformation script** field, enter the following:

```
var modelInput = new java.util.HashMap() modelInput.put("text",
request.getFirstParam("q")) modelInput
```

4. In the **Model output transformation script** field, enter the following:

```
// To put into request request.putSingleParam("sentiment_label",
modelOutput.get("label")[0])
request.putSingleParam("sentiment_score",
modelOutput.get("score")[0]) // To put into query context
context.put("sentiment_label", modelOutput.get("label")[0])
context.put("sentiment_score", modelOutput.get("score")[0]) //
To put into response documents. NOTE: This can be done only
after Solr Query stage var docs =
response.get().getInnerResponse().getDocuments(); var ndocs =
new java.util.ArrayList(); for (var i=0; i<docs.length;i++){ var
doc = docs[i]; doc.putField("query_sentiment_label",
modelOutput.get("label")[0])
doc.putField("query_sentiment_score", modelOutput.get("score")
[0]) ndocs.add(doc); }
response.get().getInnerResponse().updateDocuments(ndocs);
```

5. Save the pipeline.

Model output

Both of the pre-trained models output the following:

- a **label**, `negative` or `positive`
- a **score** from `-2` to `2`

The sentiment-reviews:v1.0 model also optionally outputs tokens and their corresponding attention weights, that is, the weight that each token carries in the sentiment prediction. The total of all attention weights is always 1, that is, each value represents a percentage of the total weight. In the example below, "awesome" has the highest weight because it expresses the strongest sentiment compared to other tokens in the string:

```
// Input text = "That is awesome!" // Output sentiment_label =
'positive' sentiment_score = 1.998 sentiment_attention_tokens =
['That', "'", 's', 'awesome', '!'] sentiment_attention_weights =
[0.154, 0.078, 0.069, 0.444, 0.255]
```

Deploy the sentiment-reviews Model

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion*

This topic explains how to deploy the sentiment-reviews pre-trained *sentiment prediction* model. This model is trained on a variety of customer reviews and optimized for longer texts. It also supports attention weights output that can be used for highlighting the tokens that provide stronger sentiment; see *Model output* below for an example.

See also the *sentiment-general model deployment*.

1. Install the model in Fusion

1. Navigate to **Collections** > **Jobs**.
2. Select **New** > **Create Seldon Core Model Deployment**.
3. Configure the job as follows:
    - **Job ID.** The ID for this job, such as `deploy-sentiment-reviews`
    - **Model Name.** The model name of the Seldon Core deployment that will be referenced in the Machine Learning pipeline stage configurations, such as `sentiment-reviews`.
    - **Docker Repository.** The value is `lucidworks`.
    - **Image Name.** The value is `sentiment-reviews:v1.0`.
    - **Kubernetes Secret Name for Model Repo.** The value is left empty.
    - **Output Column Names for Model.** The value is `[label, score, tokens, attention_weights]`.
4. Click **Save**.
5. Click **Run** > **Start**.

2. Configure the Machine Learning pipeline stages

You can put your sentiment prediction model to work using the Machine Learning index stage or Machine Learning query stage. You will specify the same **Model Name** that you used when you installed the model above.

Generally, you only need to apply the model in the index pipeline, in order to perform sentiment prediction on your content. Optionally, you can configure the query pipeline in a similar way, to perform sentiment prediction on incoming queries and outgoing responses and apply special treatment depending on the prediction.

2.1. Configure the Machine Learning index stage

1. In your index pipeline, click **Add a Stage** > **Machine Learning**.
2. In the **Model ID** field, enter the model name you configured above, such as `sentiment-reviews`.
3. In the **Model input transformation script** field, enter one of the following, depending on whether you want to output attention weights:

Without attention weights	With attention weights
`var modelInput = new java.util.HashMap() modelInput.put("text", doc.getFirstFieldValue ("text")) modelInput`	`var modelInput = new java.util.HashMap() modelInput.put("text", doc.getFirstFieldValue("text") ) modelInput.put("attention_outp ut", "true") modelInput`

4. In the **Model output transformation script** field, enter the following:

Without attention weights	With attention weights
`doc.addField("sentiment_label_s", modelOutput.get("label")[0]) doc.addField("sentiment_score_d", modelOutput.get("score")[0])`	`doc.addField("sentiment_label_s", modelOutput.get("label")[0]) doc.addField("sentiment_score_d", modelOutput.get("score")[0]) doc.addField("sentiment_attention_tokens_ss", modelOutput.get("tokens")) doc.addField("sentiment_attention_weights_ds", modelOutput.get("attention_weights"))`

5. Save the pipeline.

2.2. Optional: Configure the Machine Learning query stage

1. In your query pipeline, click **Add a Stage** > **Machine Learning**.
2. In the **Model ID** field, enter the model name you configured above, such as sentiment-reviews.
3. In the **Model input transformation script** field, enter the following:

Without attention weights	With attention weights
`var modelInput = new java.util.HashMap() modelInput.put("text", request.getFirstParam("q")) modelInput`	`var modelInput = new java.util.HashMap() modelInput.put("text", request.getFirstParam("q")) modelInput.put("attention_output", "true") modelInput`

4. In the **Model output transformation script** field, enter the following, noting the sections that need to be uncommented if you are using attention weights:

```
// To put into request request.putSingleParam("sentiment_label",
modelOutput.get("label")[0])
request.putSingleParam("sentiment_score",
modelOutput.get("score")[0]) // With attention output also
uncomment this //
request.putSingleParam("sentiment_attention_tokens",
modelOutput.get("tokens")) //
request.putSingleParam("sentiment_attention_weights",
modelOutput.get("attention_weights")) // To put into query
context context.put("sentiment_label", modelOutput.get("label")
[0]) context.put("sentiment_score", modelOutput.get("score")[0])
// With attention output also uncomment this //
context.put("sentiment_attention_tokens",
modelOutput.get("tokens")) //
context.put("sentiment_attention_weights",
modelOutput.get("attention_weights")) // To put into response
documents (can be done only after Solr Query stage) var docs =
response.get().getInnerResponse().getDocuments(); var ndocs =
new java.util.ArrayList(); var attention_tokens =
modelOutput.get("tokens") var attention_weights =
modelOutput.get("attention_weights") var attention_tokens_arr =
new java.util.ArrayList(attention_tokens.size()); var
attention_weights_arr = new
java.util.ArrayList(attention_weights.size()); for ( i = 0; i <
attention_tokens.size(); i++) {
attention_tokens_arr.add(attention_tokens[i])
attention_weights_arr.add(attention_weights[i]) } for (var i=0;
i<docs.length;i++){ var doc = docs[i];
doc.putField("query_sentiment_label", modelOutput.get("label")
[0]) doc.putField("query_sentiment_score",
modelOutput.get("score")[0]) // With attention output also
uncomment this //
doc.putField("query_sentiment_attention_tokens",
```

```
attention_tokens_arr) //
doc.putField("query_sentiment_attention_weights",
attention_weights_arr) ndocs.add(doc); }
response.get().getInnerResponse().updateDocuments(ndocs);
```

5. Save the pipeline.

Model output

Both of the pre-trained models output the following:

- a **label**, `negative` or `positive`
- a **score** from `-2` to `2`

The sentiment-reviews:v1.0 model also optionally outputs tokens and their corresponding attention weights, that is, the weight that each token carries in the sentiment prediction. The total of all attention weights is always 1, that is, each value represents a percentage of the total weight. In the example below, "awesome" has the highest weight because it expresses the strongest sentiment compared to other tokens in the string:

```
// Input text = "That is awesome!" // Output sentiment_label =
‘positive’ sentiment_score = 1.998 sentiment_attention_tokens =
['That', "'", 's', 'awesome', '!'] sentiment_attention_weights =
[0.154, 0.078, 0.069, 0.444, 0.255]
```

Deploy the sentiment-reviews Model

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

This topic explains how to deploy the sentiment-reviews pre-trained *sentiment prediction* model. This model is trained on a variety of customer reviews and optimized for longer texts. It also supports attention weights output that can be used for highlighting the tokens that provide stronger sentiment; see *Model output* below for an example.

See also the *sentiment-general model deployment*.

1. Install the model in Managed Fusion

1. Navigate to **Collections** > **Jobs**.
2. Select **New** > **Create Seldon Core Model Deployment**.
3. Configure the job as follows:
    - **Job ID.** The ID for this job, such as `deploy-sentiment-reviews`
    - **Model Name.** The model name of the Seldon Core deployment that will be referenced in the Machine Learning pipeline stage configurations, such as `sentiment-reviews`.
    - **Docker Repository.** The value is `lucidworks`.
    - **Image Name.** The value is `sentiment-reviews:v1.0`.
    - **Kubernetes Secret Name for Model Repo.** The value is left empty.
    - **Output Column Names for Model.** The value is `[label, score, tokens, attention_weights]`.
4. Click **Save**.
5. Click **Run** > **Start**.

2. Configure the Machine Learning pipeline stages

You can put your sentiment prediction model to work using the Machine Learning index stage or Machine Learning query stage. You will specify the same **Model Name** that you used when you installed the model above.

Generally, you only need to apply the model in the index pipeline to perform sentiment prediction on your content. Optionally, you can configure the query pipeline in a similar way, to perform sentiment prediction on incoming queries and outgoing responses and apply special treatment depending on the prediction.

2.1. Configure the Machine Learning index stage

1. In your index pipeline, click **Add a Stage** > **Machine Learning**.
2. In the **Model ID** field, enter the model name you configured above, such as `sentiment-reviews`.
3. In the **Model input transformation script** field, enter one of the following, depending on whether you want to output attention weights:

Without attention weights	With attention weights
`var modelInput = new java.util.HashMap() modelInput.put("text", doc.getFirstFieldValue ("text")) modelInput`	`var modelInput = new java.util.HashMap() modelInput.put("text", doc.getFirstFieldValue("text") ) modelInput.put("attention_outp ut", "true") modelInput`

4. In the **Model output transformation script** field, enter the following:

Without attention weights	With attention weights
`doc.addField("sentiment_label_s", modelOutput.get("label")[0]) doc.addField("sentiment_score_d", modelOutput.get("score")[0])`	`doc.addField("sentiment_label_s", modelOutput.get("label")[0]) doc.addField("sentiment_score_d", modelOutput.get("score")[0]) doc.addField("sentiment_attention_tokens_ss", modelOutput.get("tokens")) doc.addField("sentiment_attention_weights_ds", modelOutput.get("attention_weights"))`

5. Save the pipeline.

2.2. Optional: Configure the Machine Learning query stage

1. In your query pipeline, click **Add a Stage** > **Machine Learning**.
2. In the **Model ID** field, enter the model name you configured above, such as sentiment-reviews.
3. In the **Model input transformation script** field, enter the following:

Without attention weights	With attention weights
`var modelInput = new java.util.HashMap() modelInput.put("text", request.getFirstParam("q")) modelInput`	`var modelInput = new java.util.HashMap() modelInput.put("text", request.getFirstParam("q")) modelInput.put("attention_output", "true") modelInput`

4. In the **Model output transformation script** field, enter the following, noting the sections that need to be uncommented if you are using attention weights:

```
// To put into request request.putSingleParam("sentiment_label",
modelOutput.get("label")[0])
request.putSingleParam("sentiment_score",
modelOutput.get("score")[0]) // With attention output also
uncomment this //
request.putSingleParam("sentiment_attention_tokens",
modelOutput.get("tokens")) //
request.putSingleParam("sentiment_attention_weights",
modelOutput.get("attention_weights")) // To put into query
context context.put("sentiment_label", modelOutput.get("label")
[0]) context.put("sentiment_score", modelOutput.get("score")[0])
// With attention output also uncomment this //
context.put("sentiment_attention_tokens",
modelOutput.get("tokens")) //
context.put("sentiment_attention_weights",
modelOutput.get("attention_weights")) // To put into response
documents (can be done only after Solr Query stage) var docs =
response.get().getInnerResponse().getDocuments(); var ndocs =
new java.util.ArrayList(); var attention_tokens =
modelOutput.get("tokens") var attention_weights =
modelOutput.get("attention_weights") var attention_tokens_arr =
new java.util.ArrayList(attention_tokens.size()); var
attention_weights_arr = new
java.util.ArrayList(attention_weights.size()); for ( i = 0; i <
attention_tokens.size(); i++) {
attention_tokens_arr.add(attention_tokens[i])
attention_weights_arr.add(attention_weights[i]) } for (var i=0;
i<docs.length;i++){ var doc = docs[i];
doc.putField("query_sentiment_label", modelOutput.get("label")
[0]) doc.putField("query_sentiment_score",
modelOutput.get("score")[0]) // With attention output also
uncomment this //
doc.putField("query_sentiment_attention_tokens",
```

```
attention_tokens_arr) //
doc.putField("query_sentiment_attention_weights",
attention_weights_arr) ndocs.add(doc); }
response.get().getInnerResponse().updateDocuments(ndocs);
```

5. Save the pipeline.

Model output

Both of the pre-trained models output the following:

- a **label**, `negative` or `positive`
- a **score** from `-2` to `2`

The sentiment-reviews:v1.0 model also optionally outputs tokens and their corresponding attention weights, that is, the weight that each token carries in the sentiment prediction. The total of all attention weights is always 1, that is, each value represents a percentage of the total weight. In the example below, "awesome" has the highest weight because it expresses the strongest sentiment compared to other tokens in the string:

```
// Input text = "That is awesome!" // Output sentiment_label =
‘positive’ sentiment_score = 1.998 sentiment_attention_tokens =
['That', "'", 's', 'awesome', '!'] sentiment_attention_weights =
[0.154, 0.078, 0.069, 0.444, 0.255]
```

Deploying a WAR File

How-to

> Compatible versions: 4.2, 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *App Studio*

Appkit runs in most Java servlet containers including Apache Tomcat, Jetty and GlassFish. The majority of our customers use Tomcat 8 for production systems.

If using Tomcat, see the *troubleshooting page* for information on configuring the use of UTF-8 character encoding. (This resolves the most common issues with special characters.)

Follow these steps to deploy an Appkit .war file:

1. Ensure that Java 8 JDK Java 1.8 is installed.

   Important

   We do not recommend OpenJDK, which is installed by default in various Linux distributions. We recommend using the Oracle JDK only.

2. Install your servlet container of choice, such as Tomcat 8.
3. You will have been provided a web application archive (.war file) that contains an Appkit web application (such as `twigkit-search.war`). If using Tomcat, copy the provided .war file into Tomcat's webapps folder.
4. Start the servlet container.
5. Several seconds later, a folder with the same name as the .war file will appear in the file system (for example, twigkit-search). You can then access the application in your browser at `http://localhost:8080/{contextPath}`, where `{contextPath}` is the name of the .war file (for example, `http://localhost:8080/twigkit-search/`). You can also access the Appkit logs in the file system at `{tomcat}/logs/catalina.out`.

Congratulations, your Appkit application has been successfully installed.

Develop a Custom Connector

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion Connectors*, *Fusion*

Java SDK configuration

To build a valid connector configuration, you must:

- Define an interface.
- Extend `ConnectorConfig`.
- Apply a few annotations.
- Define connector methods and annotations.

All methods that are annotated with `@Property` are considered to be configuration properties. For example, `@Property() String name();` results in a String property called `name`. This property would then be present in the generated schema.

Here is an example of the most basic configuration, along with required annotations:

```
@RootSchema( title = "My Connector", description = "My Connector
description", category = "My Category" ) public interface MyConfig
extends ConnectorConfig<MyConfig.Properties> { @Property( title =
"Properties", required = true ) public Properties properties(); /** *
Connector specific settings */ interface Properties extends
FetcherProperties { @Property( title = "My custom property",
description = "My custom property description" ) public Integer
myCustomProperty(); } }
```

The metadata defined by `@RootSchema` is used by Fusion when showing the list of available connectors. The `ConnectorConfig` base interface represents common, top-level settings required by all connectors. The `type` parameter of the `ConnectorConfig` class indicates the interface to use for custom properties.

Once a connector configuration has been defined, it can be associated with the `ConnectorPlugin` class. From that point, the framework takes care of providing the configuration instances to your connector. It also generates the schema, and sends it along to Fusion when it connects to Fusion.

Schema metadata can be applied to properties using additional annotations. For example, applying limits to the min/max length of a string, or describing the types of items in an array.

Nested schema metadata can also be applied to a single field by using "stacked" schema based annotations:

```
interface MySetConfig extends Model {
@SchemaAnnotations.Property(title = "My Set")
@SchemaAnnotations.ArraySchema(defaultValue = "[\"a\"]")
@SchemaAnnotations.StringSchema(defaultValue = "some-set-value",
minLength = 1, maxLength = 1) Set<String> mySet(); }
```

Plugin client

The Fusion connector plugin client provides a wrapper for the Fusion Java plugin-sdk so that plugins do not need to directly talk with gRPC code. Instead, they can use high-level interfaces and base classes, like Connector and Fetcher.

The plugin client also provides a standalone "runner" that can host a plugin that was built from the Fusion Java Connector SDK. It does this by loading the plugin zip file, then calling on the wrapper to provide the framework interactions.

Standalone Connector Plugin Application

The second goal of the plugin-client is to allow Java SDK plugins to run remotely. The instructions for deploying a connector using this method are provided below.

LOCATING THE UBERJAR

The uberjar is located in this location in the Fusion file system:

```
$FUSION_HOME/apps/connectors/connectors-rpc/client/connector-plugin-client-<version>-uberjar.jar
```

where `$FUSION_HOME` is your Fusion installation directory and `<version>` is your Fusion version number.

STARTING THE HOST

To start the host app, you need a Fusion SDK-based connector, built into the standard packaging format as a `.zip` file. This `zip` must contain only one connector plugin.

Here is an example of how to start up using the web connector:

```
java -jar $FUSION_HOME/apps/connectors/connectors-rpc/client/connector-plugin-client-<version>-uberjar.jar fusion-connectors/build/plugins/connector-web-4.0.0-SNAPSHOT.zip
```

To run the client with remote debugging enabled:

```
java -agentlib:jdwp=transport=dt_socket,server=y,suspend=n,address=5010 -jar $FUSION_HOME/apps/connectors/connectors-rpc/client/connector-plugin-client-<version>-uberjar.jar fusion-connectors/build/plugins/connector-web-4.0.0-SNAPSHOT.zip
```

Java SDK security

See the main custom connector security overview for instructions on how to configure SSL/TLS support for a Fusion custom connector.

Java Plugin Client

The information here is specific to running a Java SDK plugin outside of Fusion.

The plugin-client supports several variations of SSL/TLS auth. The examples below show the relevant Java properties.

Example with Mutual TLS auth and private key passwords

```
–Dcom.lucidworks.apollo.app.hostname=myhost –
Dcom.lucidworks.fusion.tls.trustCertCollection=./sslcerts/ca.crt –
Dcom.lucidworks.fusion.tls.client.certChain=./sslcerts/client.crt –
Dcom.lucidworks.fusion.tls.client.privateKey=./sslcerts/client.pem –
Dcom.lucidworks.fusion.tls.client.privateKeyPassword=password123 –
Dcom.lucidworks.fusion.tls.enabled=true</td>
```

Example without TLS auth and no private key passwords:

```
–Dcom.lucidworks.apollo.app.hostname=myhost –
Dcom.lucidworks.fusion.tls.trustCertCollection=./sslcerts/ca.crt –
Dcom.lucidworks.fusion.tls.enabled=true
```

Simple Connector

Connector Description

This connector generates a configurable number of documents, all with random titles and body fields.

Quick start

1. Clone the repo:

```
git clone https://github.com/lucidworks/connectors-sdk-
resources.git
cd connectors-sdk-resources/java-sdk/connectors/
./gradlew assemblePlugins
```

2. This produces one zip file, named `simple-connector.zip`, located in the `build/plugins` directory. This artifact is now ready to be uploaded directly to Fusion as a connector plugin.
3. See the following *instructions* on how to build, deploy, and run the plugin.

Connector properties

RANDOM CONTENT GENERATOR PROPERTIES

Property Name	Property description
Total	The total number of documents to generate
Minimum number of sentences	The minimum number of sentences to generate per document, the random generator will use this value as lower bound to calculate a random number of sentences
maximum number of sentences	The maximum number of sentences to generate per document, the random generator will use this value as upper bound to calculate a random number of sentences

How to use the connector

- Create a configuration with the properties listed above.
- After the first job is completed, the connector will index the same number of documents as defined in the `Random content generator properties.Total` property.

Develop a Custom Connector

How-to

> Compatible versions: 4.2
>
> Works with: *Fusion Connectors*, *Fusion Server*

Fusion 4.2.6

Java SDK configuration

To build a valid connector configuration, you must:

- Define an interface.
- Extend `ConnectorConfig`.
- Apply a few annotations.
- Define connector "getter" methods and annotations.

All methods that are annotated with `@Property` and start with "get" are considered to be configuration properties. For example, `@Property() String getName();` results in a String property called `name`. This property would then be present in the generated schema.

Here is an example of the most basic configuration, along with required annotations and a sample "getter" method:

```
@RootSchema( name = "com.my.connector", title = "My Connector",
description = "My Connector description", category = "My Category" )
public interface MyConfig extends ConnectorConfig<MyConfig.Properties>
{ @Property( title = "Properties", required = true ) public Properties
getProperties(); /** * Connector specific settings */ interface
Properties extends FetcherProperties { @Property( title = "My custom
property", description = "My custom property description" ) public
Integer getMyCustomProperty(); } }
```

The metadata defined by `@RootSchema` is used by Fusion when showing the list of available connectors. The `ConnectorConfig` base interface represents common, top-level settings required by all connectors. The `type` parameter of the `ConnectorConfig` class indicates the interface to use for custom properties.

Once a connector configuration has been defined, it can be associated with the `ConnectorPlugin` class. From that point, the framework takes care of providing the configuration instances to your connector. It also generates the schema, and sends it along to Fusion when it connects to Fusion.

About com.lucidworks.schema

The `com.lucidworks.schema` project (included in the connectors SDK) aims to simplify the process of creating JSON Schemas. It also provides utilities for building type-safe POJOs from schema definitions.

The basic idea here is that you create an interface that extends `Model`, add property getters, and then add a few annotations. Once your interface has been built, you can generate an `ObjectType` instance, which is the object that contains all of the JSON schema metadata. By then combining that schema object with a `Map<String,Object>`, you can create instances of your interface to use as configuration objects.

The configuration objects are based on `java.lang.reflect.Proxy`, which is what the `ModelGenerator` returns. Most method calls to these proxy instances result in method calls to the underlying `Map<String,Object>`. For example, if the interface defines a `getId` method, then a runtime call results in a call to the `Map<String,Object>`: `data.get("id")`.

A few special cases exist:

- `toString` is proxied directly to the `Map<String,Object>`
- `getClass` returns the class of the interface provided to `ModelGenerator#generate()`
- `_data` returns the underlying `Map<String,Object>` object

- If the method starts with `set`, the arguments are captured and sent to the underlying map via `put()`

Here is a simple example:

```
interface MyConfig extends Model { @Property
@StringSchema(minLength=1) String getId(); }
```

Create the schema:

```
class Runner { public static void main(String[] args){ ObjectType
schema = SchemaGenerator.generate(MyConfig.class); } }
```

Generate an instance of the `MyConfig` class:

```
class Runner { public static void main(String[] args){
Map<String,Object> data = new java.util.HashMap<>(); data.put("id",
100); MyConfig config = ModelGenerator.generate(MyConfig.class, data);
System.out.println("id -> " + config.getId()); } }
```

Schema metadata can be applied to properties using additional annotations. For example, applying limits to the min/max length of a string, or describing the types of items in an array.

Nested schema metadata can also be applied to a single field by using "stacked" schema based annotations:

```
interface MySetConfig extends Model {
@SchemaAnnotations.Property(title = "My Set")
@SchemaAnnotations.ArraySchema(defaultValue = "[\"a\"]")
@SchemaAnnotations.StringSchema(defaultValue = "some-set-value",
minLength = 1, maxLength = 1) Set<String> getMySet(); }
```

Plugin client

The Fusion connector plugin client provides a wrapper for the Fusion Java plugin-sdk so that plugins do not need to directly talk with gRPC code. Instead, they can use high-level interfaces and base classes, like Connector and

Fetcher.

The plugin client also provides a standalone "runner" that can host a plugin that was built from the Fusion Java Connector SDK. It does this by loading the plugin zip file, then calling on the wrapper to provide the gRPC interactions.

Java SDK / gRPC Wrapper

One of the primary goals of the plugin-client is to isolate plugin code from the underlying framework details. Specifically, the underlying message formats (protocol buffers) and native gRPC code. This makes it possible to make some changes to the base support layer, without having to make changes to the Java plugin implementation.

Standalone Connector Plugin Application

The second goal of the plugin-client is to allow Java SDK plugins to run remotely. The instructions for deploying a connector using this method are provided below.

LOCATING THE UBERJAR

The uberjar is located in this location in the Fusion file system:

```
$FUSION_HOME/apps/connectors/connectors-rpc/client/connector-plugin-client-<version>-uberjar.jar
```

where `$FUSION_HOME` is your Fusion installation directory and `<version>` is your Fusion version number.

STARTING THE HOST

To start the host app, you need a Fusion SDK-based connector, built into the standard packaging format as a `.zip` file. This `zip` must contain only one connector plugin.

Here is an example of how to start up using the web connector:

```
java -jar $FUSION_HOME/apps/connectors/connectors-rpc/client/connector-plugin-client-<version>-uberjar.jar fusion-connectors/build/plugins/connector-web-4.0.0-SNAPSHOT.zip
```

To run the client with remote debugging enabled:

```
java -agentlib:jdwp=transport=dt_socket,server=y,suspend=n,address=5010 -jar $FUSION_HOME/apps/connectors/connectors-rpc/client/connector-plugin-client-<version>-uberjar.jar fusion-connectors/build/plugins/connector-web-4.0.0-SNAPSHOT.zip
```

Java SDK security

Fusion connectors support SSL/TLS.

Java Plugin Client

The information here is specific to running a Java SDK plugin outside of Fusion.

The Fusion Connector Plugin Client supports several variations of SSL/TLS auth. The examples below show the relevant Java properties.

Example with Mutual TLS auth and private key passwords

```
–Dcom.lucidworks.apollo.app.hostname=myhost –
Dcom.lucidworks.fusion.tls.trustCertCollection=./sslcerts/ca.crt –
Dcom.lucidworks.fusion.tls.client.certChain=./sslcerts/client.crt –
Dcom.lucidworks.fusion.tls.client.privateKey=./sslcerts/client.pem –
Dcom.lucidworks.fusion.tls.client.privateKeyPassword=password123 –
Dcom.lucidworks.fusion.tls.enabled=true</td>
```

Example without TLS auth and no private key passwords:

```
–Dcom.lucidworks.apollo.app.hostname=myhost –
Dcom.lucidworks.fusion.tls.trustCertCollection=./sslcerts/ca.crt –
Dcom.lucidworks.fusion.tls.enabled=true
```

Random Connector

This connector generates a configurable number of documents, all with random titles and body fields.

Quick start

Note

This quick start assumes that Fusion is installed on the `/opt` path.

1. Clone the repo:

```
git clone https://github.com/lucidworks/connectors-sdk-resources.git
cd connectors-sdk-resources/java-sdk/connectors/ ./gradlew
assemblePlugins
```

2. This produces one zip file, named `random-connector.zip`, located in the `build/plugins` directory. This artifact is now ready to be uploaded directly to Fusion as a connector plugin.
3. See the following instructions on how to build, deploy, and run the plugin.

Develop an App with App Studio

How-to

> Compatible versions: 4.2
>
> Works with: *App Studio*

When you first create a UI using the setup wizard, it configures the default search view. In version 4.0, the view is located at `app-studio/app/views/search.html`.

App Studio supports further customization using an array of markup elements, or tags, that handle all the complexities of connecting to Fusion and visualizing the response to your queries.

These topics explain the details of how to customize your search interface:

- *Introduction to App Studio Markup*
- *Create a New Web App*
- *The Code Editor*
- *Downloading a Project* (version 4.1 and later)
- *Search Queries*
- *Search Filters*
- *Configuring Autocomplete*
- *Localize an App* (version 4.2 and later)
- *Demo Deployment*

You can edit the default view in your browser using the *code editor*.

You can *download the project* to develop it using your own IDE, or to store it in a source control repository (version 4.1 and later).

You can edit the default view using any text editor or IDE (version 4.0)

- In *standalone mode*, your changes become available as soon as you save the file. Just reload the page in your browser.
- When your search UI is published as a *Fusion Web app*, you must re-publish the app to make your changes available.
- You can get detailed information about every App Studio markup tag in *Reference*.

Develop and deploy a chunking machine learning model with Ray

How-to

> Compatible versions: 5.9
>
> Works with: *Fusion*

Note

This feature is only available in Fusion 5.9.x for versions 5.9.14 and later.

This tutorial walks you through deploying your own chunking model to Fusion with Ray.

Prerequisites

- A Fusion instance with an app and data to index
- An understanding of Python and the ability to write Python code
- Docker installed locally, plus a private or public Docker repository
- Ray installed locally: `pip install ray[serve]`
- Code editor; you can use any editor, but Visual Studio Code is used in this example
- Model: Snowflake/snowflake-arctic-embed-xs
- Docker image for chunking on indexing: ray-chunking-snowflake-arctic-embed-xs
- Docker image for chunking on querying: ray-snowflake-arctic-embed-xs
- The chunking query parsers are added to your `solrconfig.xml` file, if not present:

```
<!-- FUSION NOTES: These query parsers are used with Solr-
based vector search --> <queryParser name="xvecSim"
class="org.apache.solr.lwbackported.XVecSimQParserPlugin"/>
<queryParser name="_lw_chunk_wrap"
class="org.apache.solr.lw.ParentAndAllKidsWrapperQParserPlugin
"/> <queryParser name="neuralHybrid"
class="org.apache.solr.lw.NeuralHybridQParserPlugin"/>
```

- The vector definitions are added to your `managed-schema.xml` file. See *Vector definitions* for the full definitions.

Tips

- Always test your Python code locally before uploading to Docker and then Fusion. This simplifies troubleshooting significantly.
- Once you've created your Docker you can also test locally by doing `docker run` with a specified port, like 8000, which you can then `curl` to confirm functionality in Fusion. See the testing example below.
- If you run into an issue with the model not deploying and you're using the 'real' example, there is a very good chance you haven't allocated enough memory or CPU in your job spec or in the Ray-Argo config. You can increase the resources. To edit the ConfigMap, run `kubectl edit configmap argo-deploy-ray-model-workflow -n <namespace>` and then find the `ray-head` container in the artisanal escaped YAML and change the memory limit. Exercise caution when editing because it can break the YAML. Just delete and replace a single character at a time without changing any formatting.
    - For additional guidance, see the testing locally snowflake-arctic-embed-xs_chunking-ray example.

LucidAcademy

Lucidworks offers free training to help you get started.

The **Course** for **Intro to Machine Learning in Fusion** focuses on using machine learning to to infer the goals of customers and users in order to deliver a more sophisticated search experience:

Visit the LucidAcademy to see the full training catalog.

Local testing example

1. Docker command:

```
docker run -p 127.0.0.1:8000:8000 DOCKER_IMAGE
```

2. Curl to hit Docker:

```
curl -i -X POST http://127.0.0.1:8000 -H 'Content-Type:
application/json' -d '{"text": "I love Fusion", "quantize":True,
"include_text_chunks":True}'
```

3. Curl model in Fusion:

```
curl -u $FUSION_USER:$FUSION_PASSWORD -X POST -H 'Content-Type:
application/json' -d '{"text": "I love Fusion", "quantize":True,
"include_text_chunks":False}'
https://FUSION_HOST:6764/api/ai/ml-models/MODEL_NAME/prediction
```

4. See all your deployed models:

```
https://FUSION_HOST.com/api/ai/ml-models
```

5. Check the Ray UI to see Replica State, Resources, and Logs.
   If you are getting an internal model error, the best way to see what is going on is to query via port-forwarding the model. The `MODEL_DEPLOYMENT` in the command below can be found with `kubectl get svc -n NAMESPACE`. It will have the same name as set in the model name in the **Create Ray Model Deployment** job.

```
kubectl -n NAMESPACE port-forward svc/MODEL_DEPLOYMENT-head-svc
8000:8265
```

Download the model and choose a chunking strategy

This tutorial uses the Snowflake/snowflake-arctic-embed-xs model from Hugging Face, but any pre-trained model from huggingface.co works with this tutorial.

For the chunking strategy you can start with LangChain or LlamaIndex.

If you want to use your own model instead, you can do so, but your model must have been trained and then saved though a function similar to the PyTorch's `torch.save(model, PATH)` function. See Saving and Loading Models in the PyTorch documentation.

Create the index model

The next step is to format a Python class which will be invoked by Fusion to get the results from your index model. The skeleton below represents the format that you should follow. This is distinct from *the standard example without chunking* because the format to output is more complex.

The model's return value must be a dictionary with a key named `response`. The value associated with this key must be a JSON string. When parsed, this JSON string is a dictionary that contains two primary keys:

- `spans`: A list of lists, where each inner list represents `[start_index, end_index]` pairs for each text chunk
- `vectors`: A list of dictionaries. Each dictionary in this list must have a key named `vector`, and the value is a list of numbers representing an embedding vector with the shape of (1, DIM), where DIM

(vector dimension) is a consistent integer. This format is required for the *Local Chunker Index Stage* to handle the vector encoding.

Optionally you can pass the actual chunks in a list of strings. However, this is not recommended due to the Solr storage impact of saving a very large document twice. See also Getting Started in the Ray Serve documentation.

```
import json from typing import Any, Dict, List from ray import serve from starlette.requests import Request from torch import Tensor # These defaults are for the ray serve deployment # when running simply from docker. The 'Create Ray Model Deployment' # job can override these replicas and resources if needed. @serve.deployment(num_replicas=1, ray_actor_options={"num_cpus": 1}) class Deployment(object): def __init__(self): """ Add any initialization parameters. Generally this is where you would load your model. This method will be called once when the deployment is created. """ print("Initializing") self.model = load_model() # faux code # This can be named as any method which takes a dictionary as input and returns a dictionary # as output. In this example, we are using the encode method to encode the # input text into a vector. def main(self, input_dict: Dict[str, Any]) -> Dict[str, Any]: """ This method will be called when the deployment is queried. It will receive the input data and should return the output data. """ text = input_dict["text"] chunks = self.chunk_text(text) # faux code spans = self.get_spans(text) # faux code embeddings = self.model.encode(chunks) # faux code output = self.create_output_dict(spans=spans, vectors=embeddings, chunks=chunks) return output # This shows the required structure of the output dictionary. # The 'spans' and 'vectors' keys are required for the `Local Chunker` stage. def create_output_dict( self, spans: List[List[int]], vectors: Tensor, chunks: List[str], ) -> Dict[str, Any]: vectors = self.format_vectors(vectors=vectors, chunks_len=len(chunks)) output_dict = { "spans": spans, # required "vectors": vectors, # required "chunks": chunks, # optional } return { "response": json.dumps(output_dict), # This is required because the Fusion logic doesn't handle nested structures } # This converts the vectors to the required format # for the Local Chunking stage. The
```

```
structure # is a list of dictionaries,each containing a # 'vector'
key. # [{"vector": [0.1, 0.2, ...]}, {"vector": [0.3, 0.4, ...]}, ...]
@staticmethod def format_vectors( vectors: Tensor, chunks_len: int, )
-> List[Dict[str, Any]]: vectors = ( vectors.squeeze().tolist() if
chunks_len > 1 else [vectors.squeeze().tolist()] ) # Convert the
tensor to a list of dictionaries formatted_vectors = [] for vector in
vectors: formatted_vectors.append({"vector": vector}) return
formatted_vectors async def __call__(self, http_request: Request) ->
Dict[str, Any]: input_dict: Dict[str, Any] = await http_request.json()
return self.main( input_dict=input_dict ) # This will be the function
you defined above, in this case main app = Deployment.bind()
```

A real instance of this class with the **`snowflake-arctic-embed-xs`** model is as follows:

Note

This code pulls from Hugging Face. To have the model load in the image without pulling from Hugging Face or other external sources, download the model weights into a folder name and change the model name to the folder name preceded by `./`.

```
"""
Example is based on:
    - https://huggingface.co/Snowflake/snowflake-arctic-embed-xs
    - https://docs.ray.io/en/latest/serve/getting_started.html
""" import json import sys from time import time from typing import
Any, Dict, List, Optional import torch import torch.nn.functional as F
from langchain_text_splitters import CharacterTextSplitter from ray
import serve from starlette.requests import Request from
starlette.responses import JSONResponse from torch import Tensor from
transformers import AutoModel, AutoTokenizer HUB_MODEL_NAME =
"Snowflake/snowflake-arctic-embed-xs" # Truncate to 150k characters to
avoid timeout errors from model # and 10 Chunk limit to protect Fusion
document count CHAR_TRUNCATION = 150_000 MAX_CHUNKS = 10
@serve.deployment(num_replicas=1, ray_actor_options={"num_cpus": 1})
class Deployment(object): def __init__(self): from loguru import
logger # Initializing logger self.logger = logger self.logger.remove()
self.logger.add(sys.stdout, level="INFO", serialize=False,
colorize=True) # Initializing model self.logger.info("Loading
model...") self.tokenizer =
AutoTokenizer.from_pretrained(HUB_MODEL_NAME) self.model =
AutoModel.from_pretrained(HUB_MODEL_NAME) self.model.eval()
self.vector_size = 384 self.logger.info("Model initialization
finished!") # snowflake-arctic-embed-xs specific prefix (NOT for other
models) self.query_prefix = "Represent this sentence for searching
relevant passages:" self.passage_prefix = "" def main(self,
input_dict: Dict[str, Any]) -> Dict[str, Any]: try: timings_dict = {}
_start_time = time() # Extracting text from input text =
input_dict.get("text", None) include_text_chunks = self.obj_to_bool(
input_dict.get("include_text_chunks", False) ) if text is None: #
Check if text is None and return self.logger.error("No text provided
in the input dictionary.") return JSONResponse( {"error": "No
`text`input key provided"}, status_code=400 ) elif ( text.isspace() or
len(text) == 0 ): # Check if the text is empty and return
```

```
self.logger.warning("Empty text provided. Returning empty output.")
return self.create_output_dict( spans=[], vectors=[], chunks=[],
include_text_chunks=include_text_chunks, ) elif len(text) >
CHAR_TRUNCATION: # Check if the text is too long text =
text[:CHAR_TRUNCATION] self.logger.warning( f"Input text truncated to
{CHAR_TRUNCATION} characters" f" to avoid Fusion stage timeout
errors." ) # Log the length of the text self.logger.debug(f"Text len:
{len(text)}") # Check DataType dataType = input_dict.get("dataType",
"passage") # Check if quantization is set quantize_bool =
self.obj_to_bool(input_dict.get("quantize", False)) # Check if chunks
should be included # chunking the text chunks = self.chunk_text(text)
initial_chunk_count = len(chunks) # Limit the number of chunks to
Expolding Fusion Document Count if initial_chunk_count > MAX_CHUNKS:
chunks = chunks[:MAX_CHUNKS] self.logger.warning( f"Input text chunked
into {initial_chunk_count} chunks. " f"Only the first {MAX_CHUNKS}
chunks will be processed." ) timings_dict["chunking_time"] = time()
spans = self.get_chunk_spans(text=text, chunks=chunks)
timings_dict["span_extraction_time"] = time() # encoding the chunks
vectors = self.encode_chunks(chunks, dataType=dataType)
timings_dict["encoding_time"] = time() # Check if quantization is set
if quantize_bool: vectors = self.quantize_vectors(embeddings=vectors)
timings_dict["quantization_time"] = time() # Create output dictionary
output_dict = self.create_output_dict( spans=spans, vectors=vectors,
chunks=chunks, include_text_chunks=include_text_chunks, )
self.log_response_timings( action_name="chunk_via_ray",
start_time=_start_time, timings_dict=timings_dict, ) return
output_dict except Exception as e: self.logger.error(f"An error
occurred: {e}") return JSONResponse({"error": str(e)},
status_code=500) def encode_chunks(self, chunks: List[str], dataType:
str) -> Tensor: # Add prefix to the chunks based on the dataType if
dataType == "passage": chunks = [f"{self.passage_prefix}
{text}".strip() for text in chunks] elif dataType == "query": chunks =
[f"{self.query_prefix} {text}".strip() for text in chunks] #
Tokenization tokenized_texts = self.tokenizer( chunks, max_length=512,
```

```
padding=True, truncation=True, return_tensors="pt", ) # Encoding
(Model Specific Please check the model documentation) with
torch.inference_mode(): # Vectorization model_output =
self.model(**tokenized_texts) embeddings =
self.cls_pooling(model_output.last_hidden_state) embeddings =
F.normalize(embeddings, p=2, dim=1) return embeddings def
create_output_dict( self, spans: List[List[int]], vectors: Tensor,
chunks: Optional[List[str]], include_text_chunks: bool = False, ) ->
Dict[str, Any]: vectors = self.format_vectors(vectors=vectors,
chunks_len=len(chunks)) output_dict = { "spans": spans, "vectors":
vectors, } if include_text_chunks: output_dict["chunks"] = chunks
return { "chunkedData": json.dumps(output_dict), } @staticmethod def
format_vectors(vectors: Tensor, chunks_len: int) -> List[Dict[str,
Any]]: vectors = vectors if chunks_len > 1 else [vectors] # Convert
the tensor to a list of dictionaries formatted_vectors = [] for vector
in vectors: formatted_vectors.append({"vector":
vector.squeeze().tolist()}) return formatted_vectors @staticmethod def
cls_pooling(encoded: torch.Tensor) -> torch.Tensor: return encoded[:,
0, :] @staticmethod def chunk_text(text: str) -> List[str]: separator
= "\n\n" if text.count("\n\n") > 0 else "\n" # Initialize the text
splitter text_splitter = CharacterTextSplitter( separator=separator,
chunk_size=512, chunk_overlap=0, length_function=len, ) # Split the
text into chunks chunks = text_splitter.split_text(text) return chunks
@staticmethod def get_chunk_spans(text: str, chunks: List[str]) ->
List[List[int]]: # Initialize spans list spans = [] for chunk in
chunks: start = text.find(chunk) end = start + len(chunk)
spans.append([start, end]) return spans @staticmethod def
quantize_vectors(embeddings: Tensor) -> Tensor: min_val =
torch.min(embeddings, dim=1, keepdim=True).values max_val =
torch.max(embeddings, dim=1, keepdim=True).values scale = (max_val -
min_val).clamp(min=1e-8) normalized = (embeddings - min_val) / scale #
Makes signed byte compatible quantized = normalized * 255 - 128
quantized = torch.round(quantized).clamp(-128, 127).to(torch.int8)
return quantized @staticmethod def obj_to_bool(s: Any) -> bool: if
```

```
isinstance(s, bool): return s elif isinstance(s, int): return s != 0
elif isinstance(s, str): return str(s).strip().lower() == "true" else:
return False def log_response_timings( self, action_name: str,
start_time: float, timings_dict: Optional[Dict[str, float]] = None, )
-> None: timings_str = ( f"Time taken to {action_name} input: {(time()
- start_time) * 1000:.1f}ms" ) if timings_dict is not None:
timings_dict_str = {} previous_time = start_time for k, v in
timings_dict.items(): timings_dict_str[k] = f"{(v - previous_time) *
1000:.1f}ms" previous_time = v timings_str += f" {timings_dict_str}"
self.logger.info(timings_str) async def __call__(self, http_request:
Request) -> Dict[str, Any]: try: input_dict: Dict[str, Any] = await
http_request.json() except UnicodeDecodeError: body_bytes = await
http_request.body() try: decoded = body_bytes.decode("utf-8",
errors="replace") input_dict = json.loads(decoded) except
json.JSONDecodeError: return JSONResponse({"error": "Invalid JSON"},
status_code=400) return self.main(input_dict=input_dict) app =
Deployment.bind()
```

In the preceding code, logging has been added for debugging purposes.

The preceding code example contains the following functions:

- **`__call__`**
  This function is non-negotiable.
- **`__init__`**
  The **`__init__`** function is where models, tokenizers, vectorizers, and the like should be set to self for invoking. It is recommended that you include your model's trained parameters directly into the Docker container rather than reaching out to external storage inside **`__init__`**.
- **`main`**
  The **`main`** function is where the field or query that is passed from Fusion to the model is processed. Alternatively, you can process this in the **`call`** function but it is cleaner not to. The **`main`** function can handle any text processing needed for the model to accept input invoked in its **`model.predict()`** or equivalent function which gets the expected model result.

The model's return value must be a dictionary with a key named **`response`**. The value associated with this key must be a JSON string. When parsed, this JSON string is a dictionary that contains two primary keys:

- **`spans`**: A list of lists, where each inner list represents **`[start_index, end_index]`** pairs for each text chunk
- **`vectors`**: A list of dictionaries. Each dictionary in this list must have a key named **`vector`**, and the value is a list of numbers representing an embedding vector with the shape of (1, DIM), where DIM (vector dimension) is a consistent integer. This format is required for the *Local Chunker Index Stage* to handle the vector encoding.

If the output needs additional manipulation, that should be done before the result is returned.

Note

Use the *exact* name of the class when naming this file. In the preceding example, the Python file is named `deployment.py` and the class name is `Deployment()`.

Create a Dockerfile

The next step is to create a Dockerfile. The Dockerfile should follow this general outline; read the comments for additional details:

```
#It is important that python version is 3.x-slim FROM python:3.10-slim
# Install dependencies RUN apt-get update && apt-get install -y wget #
Create working app directory RUN mkdir -p /app WORKDIR /app # Copy the
requirements file and install the dependencies COPY requirements.txt
/app RUN pip install -r requirements.txt --no-cache-dir # Copy source
code COPY deployment.py /app # Expose serving port for HTTP
communication with Fusion EXPOSE 8000 # The end of the command follows
module:application and the below value should be set in the RAY
DEPLOYMENT IMPORT PATH field in 'Create Ray Model Deployment' job CMD
exec serve run deployment:app
```

Create a requirements file

The `requirements.txt` file is a list of installs for the Dockerfile to run to ensure the Docker container has the right resources to run the model. For the `snowflake-arctic-embed-xs` model, the requirements are as follows:

```
torch==2.6.0 -f https://download.pytorch.org/whl/torch_stable.html
transformers==4.51.3 ray[serve]==2.46.0 loguru=0.7.2
```

Any recent ray[serve] version should work, but the tested value and known supported version for Fusion 5.9.14 is 2.46.0. In general, if an item was used in an `import` statement in your Python file, it should be included in the requirements file. Check your Fusion version's release notes for the tested and verified version of ray[serve].

To populate the requirements, use the following command in the terminal, inside the directory that contains your code:

```
pip freeze > requirements.txt
```

Build and push the Docker image

After creating the **`deployment.py`**, Dockerfile, and **`requirements.txt`** files, you need to run a few Docker commands. Run the following commands in order:

```
DOCKER_BUILDKIT=1 docker build . -t [DOCKERHUB-USERNAME]/[REPOSITORY]:
[VERSION-TAG]
```

```
docker push [DOCKERHUB USERNAME]/[REPOSITORY]:[VERSION-TAG]
```

Using the example model, the terminal commands would be as follows:

```
DOCKER_BUILDKIT=1 docker build . -t jstrmec/ray-chunking-snowflake-
arctic-embed-xs:0.1
```

```
docker push jstrmec/ray-chunking-snowflake-arctic-embed-xs:0.1
```

This repository is public and you can visit it here: ray-chunking-snowflake-arctic-embed-xs

Create the query model

The chunking is complex and does a lot of particular things. To stabilize your Fusion environment and to simplify indexing and querying, this tutorial creates a separate model for querying. The query model code goes into less detail. For further explanation see *Develop and deploy a machine learning model with Ray*, which deploys a querying model.

A real instance of this class with the **`snowflake-arctic-embed-xs`** model is as follows:

```
"""
Example is based on:
    - https://huggingface.co/Snowflake/snowflake-arctic-embed-xs
    - https://docs.ray.io/en/latest/serve/getting_started.html
""" import json import sys from time import time from typing import
Any, Dict, Optional import torch import torch.nn.functional as F from
ray import serve from starlette.requests import Request from
starlette.responses import JSONResponse from torch import Tensor from
transformers import AutoModel, AutoTokenizer HUB_MODEL_NAME =
"Snowflake/snowflake-arctic-embed-xs" # Truncate to 150k characters to
avoid timeout errors from model CHAR_TRUNCATION = 150_000
@serve.deployment(num_replicas=1, ray_actor_options={"num_cpus": 1})
class Deployment(): def __init__(self): from loguru import logger #
Initializing logger self.logger = logger self.logger.remove()
self.logger.add(sys.stdout, level="INFO", serialize=False,
colorize=True) # Initializing model self.logger.info("Loading
model...") self.tokenizer =
AutoTokenizer.from_pretrained(HUB_MODEL_NAME) self.model =
AutoModel.from_pretrained(HUB_MODEL_NAME) self.model.eval()
self.logger.info("Model initialization finished!") # snowflake-arctic-
embed-xs specific prefix (NOT for other models) self.query_prefix =
"Represent this sentence for searching relevant passages:"
self.passage_prefix = "" @staticmethod def
quantize_vectors(embeddings: Tensor) -> Tensor: min_val =
torch.min(embeddings, dim=1, keepdim=True).values max_val =
torch.max(embeddings, dim=1, keepdim=True).values scale = (max_val -
min_val).clamp(min=1e-8) normalized = (embeddings - min_val) / scale #
Makes signed byte compatible quantized = normalized * 255 - 128
quantized = torch.round(quantized).clamp(-128, 127).to(torch.int8)
return quantized def encode(self, input_dict: Dict[str, Any]) ->
Dict[str, Any]: try: timings_dict = {} _start_time = time() #
Extracting text from input text = input_dict.get("text", None)
validated_text = self.validate_and_truncate_text(text) if
```

```
isinstance(validated_text, JSONResponse): return validated_text text =
validated_text # Check DataType if it is passage or query # and add
prefix accordingly dataType = input_dict.get("dataType", None) if
dataType == "passage": text = f"{self.passage_prefix} {text}".strip()
elif dataType == "query": text = f"{self.query_prefix} {text}".strip()
# Check if quantization is set quantize_bool =
self.obj_to_bool(input_dict.get("quantize", False)) # Tokenization
tokenized_texts = self.tokenizer( text, max_length=512, padding=True,
truncation=True, return_tensors="pt", )
timings_dict["tokenizing_time"] = time() # Encoding with
torch.inference_mode(): # Forward pass of the model model_output =
self.model(**tokenized_texts) embeddings =
self.cls_pooling(model_output.last_hidden_state)
timings_dict["encoding_time"] = time() # Normalizing embeddings
embeddings = F.normalize(embeddings, p=2, dim=1)
timings_dict["normalizing_time"] = time() # Check if quantization is
set if quantize_bool: # Quantize the embeddings embeddings =
self.quantize_vectors(embeddings) timings_dict["quantizing_time"] =
time() # Converting into output format output_dict = {"vector":
embeddings.squeeze().tolist()} self.log_response_timings(
action_name="encode_via_ray", start_time=_start_time,
timings_dict=timings_dict, ) return output_dict except Exception as e:
self.logger.error(f"An error occurred: {e}") return
JSONResponse({"error": str(e)}, status_code=500) def
validate_and_truncate_text(self, text: Optional[str]) -> str |
JSONResponse: if text is None: self.logger.error("No text provided in
the input dictionary.") return JSONResponse( {"error": "No `text`
input key provided"}, status_code=400 ) elif text.isspace() or
len(text) == 0: self.logger.warning("Empty text provided. Returning
empty output.") return JSONResponse({"error": "No text provided"},
status_code=400) elif len(text) > CHAR_TRUNCATION:
self.logger.warning( f"Input text truncated to {CHAR_TRUNCATION}
characters " f"to avoid Fusion stage timeout errors." ) return
text[:CHAR_TRUNCATION] return text @staticmethod def
```

```
cls_pooling(encoded: torch.Tensor) -> torch.Tensor: return encoded[:,
0, :] @staticmethod def obj_to_bool(s: Any) -> bool: if isinstance(s,
bool): return s elif isinstance(s, int): return s != 0 elif
isinstance(s, str): return str(s).strip().lower() == "true" else:
return False def log_response_timings( self, action_name: str,
start_time: float, timings_dict: Optional[Dict[str, float]] = None, )
-> None: timings_str = ( f"Time taken to {action_name} input: {(time()
- start_time) * 1000:.1f}ms" ) if timings_dict is not None:
timings_dict_str = {} previous_time = start_time for k, v in
timings_dict.items(): timings_dict_str[k] = f"{(v - previous_time) *
1000:.1f}ms" previous_time = v timings_str += f" {timings_dict_str}"
self.logger.info(timings_str) async def __call__(self, http_request:
Request) -> Dict[str, Any]: try: input_dict: Dict[str, Any] = await
http_request.json() except UnicodeDecodeError: body_bytes = await
http_request.body() try: decoded = body_bytes.decode("utf-8",
errors="replace") input_dict = json.loads(decoded) except
json.JSONDecodeError: return JSONResponse({"error": "Invalid JSON"},
status_code=400) return self.encode(input_dict=input_dict) app =
Deployment.bind()
```

Note

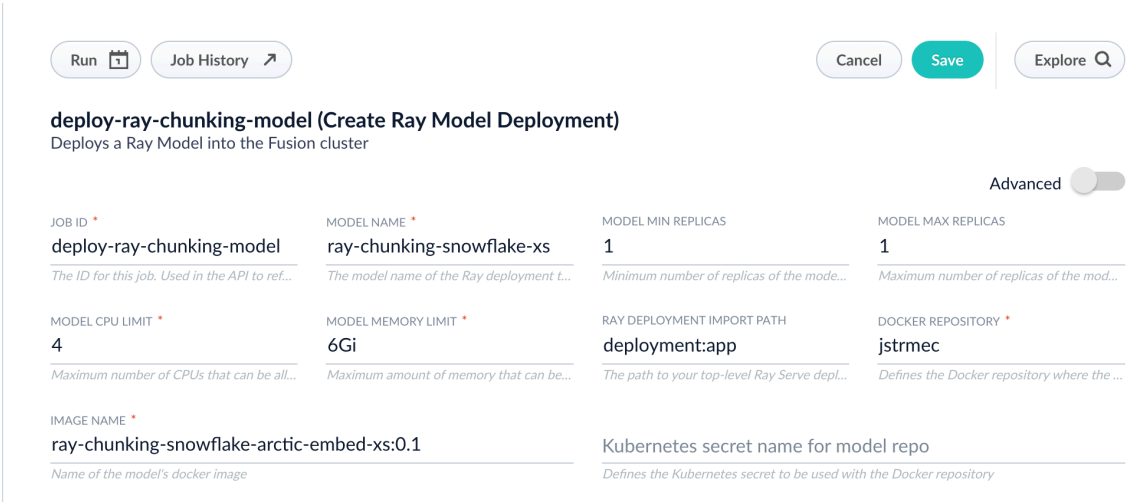
This code pulls from Hugging Face. To have the model load in the image without pulling from Hugging Face or other external sources, download the model weights into a folder name and change the model name to the folder name preceded by `./`.

This repository is public and you can visit it here: ray-snowflake-arctic-embed-xs

Deploy the models in Fusion

Now you can go to Fusion to deploy your model. You must deploy the indexing model and the querying model.

1. In Fusion, navigate to **Collections** > **Jobs**.
2. Add a job by clicking the **Add+** Button and selecting **Create Ray Model Deployment**.
3. Fill in each of the text fields. Chunking will need a higher memory and CPU limit requirement than the default:



Parameter	Description
Job ID	A string used by the Fusion API to reference the job after its creation.
Model name	A name for the deployed model. This is used to generate the deployment name in Ray. It is also the name that you reference as a `model-id` when making predictions with the ML Service.
Model min replicas	The minimum number of load-balanced replicas of the model to deploy.
Model max replicas	The maximum number of load-balanced replicas of the model to deploy. Specify multiple replicas for a higher-volume intake.
Model CPU limit	The number of CPUs to allocate to a single model replica.
Model memory limit	The maximum amount of memory to allocate to a single model replica.
Ray Deployment Import Path	The path to your top-level Ray Serve deployment (or the same path passed to `serve run`). For example, `deployment:app`
Docker Repository	The public or private repository where the Docker image is located. If you're using Docker Hub, fill in the Docker Hub username here.
Image name	The name of the image. For example, `ray-chunking-snowflake-arctic-embed-xs:0.1`.
Kubernetes secret	If you're using a private repository, supply the name of the Kubernetes secret used for access.

4. Click **Advanced** to view and configure advanced details:

Parameter	Description
Additional parameters.	This section lets you enter parameter `name:parameter` value options to be injected into the training JSON map at runtime. The values are inserted as they are entered, so you must surround string values with `"`. This is the sparkConfig field in the configuration file.
Write Options.	This section lets you enter parameter `name:parameter` value options to use when writing output to Solr or other sources. This is the writeOptions field in the configuration file.
Read Options.	This section lets you enter parameter `name:parameter` value options to use when reading input from Solr or other sources. This is the readOptions field in the configuration file.

5. Click **Save**, then **Run** and **Start**.
6. Repeat these steps for the querying model.

When the job finishes successfully, you can proceed to the next section.

Now that the models are in Fusion, you can use them in the Machine Learning or Ray / Seldon Vectorize index and query stages.

Configure the Fusion pipelines

Your real-world pipeline configuration depends on your use case and model, but for our example we will configure the index pipeline and then the query pipeline.

Configure the index pipeline

The index pipeline requires at least two additional stages: the Machine Learning stage and the Local Chunker stage.

Create the Machine Learning stage first. To create the Machine Learning stage:

1. Create a new index pipeline or load an existing one for editing.
2. Click **Add a Stage** and then **Machine Learning**.
3. In the new Machine Learning stage, fill in these fields:
    - The model ID
    - The model input

```
var modelInput = new java.util.HashMap() var text =
doc.getFirstFieldValue("embedding_t");
modelInput.put("text", text)
modelInput.put("dataType","passage")
modelInput.put("quantize", "false") // if this is true
it must be true on query-side too
modelInput.put("include_text_chunks", "true")
modelInput
```

    - The model output

```
context.put("chunkedData",
modelOutput.get("response"));
```

4. Save the stage.

To create the Local Chunker stage:

1. In the same existing index pipeline, click **Add a Stage** and then **Local Chunker**.
2. In the new stage, fill in these fields:
    - The **Input Context Variable** is `<ctx.chunkedData>`
    - The **Destination Field Name and Context Output** is `ray_chunk_vector_384v`

Configure the query pipeline

The query pipeline requires at least two additional stages: the Chunking Neural Hybrid Query stage and either the Machine Learning stage with a context key vector or the Ray/Seldon Vectorize Query stage. If you followed the full tutorial, use the Machine Learning stage.

To set up the Machine Learning query stage:

1. Create a new query pipeline or load an existing one for editing.
2. Click **Add a Stage** and then **Machine Learning**.
3. In the new stage, fill in these fields:
    - The model ID
    - The model input as shown below:

```
var modelInput = new java.util.HashMap() var text =
request.getFirstParam("q") modelInput.put("text", text)
modelInput.put("dataType", "query")
modelInput.put("quantize", "false") // should only be
true if done on index-side too modelInput
```

    - The model output as shown below:

```
context.put("vector",
modelOutput.get("vector").toString());
```

To set up the Chunking Neural Hybrid Query stage:

1. In the same existing query pipeline, click **Add a Stage** and then **Chunking Neural Hybrid Query**.

2. In the new stage, fill in the required fields.

You have now successfully uploaded indexing and querying Ray models to Fusion and deployed it, and you are now ready to query. If you are getting the same results every time you query, double check that your vectors are correct and check that the chunking query parsers are defined in your `solrconfig.xml` file as described in the prerequisites of this tutorial.

Develop and Deploy a Machine Learning Model

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion*

This tutorial walks you through deploying your own model to Fusion with Seldon Core.

Prerequisites

- A Fusion instance with an app and indexed data
- An understanding of Python and the ability to write Python code
- Docker installed locally, plus a private or public Docker repository
- Seldon-core installed locally: `pip install seldon-core`
- Code editor; you can use any editor, but Visual Studio Code is used in the example
- Model: paraphrase-multilingual-MiniLM-L12-v2 from Hugging Face
- Docker image: example_sbert_model

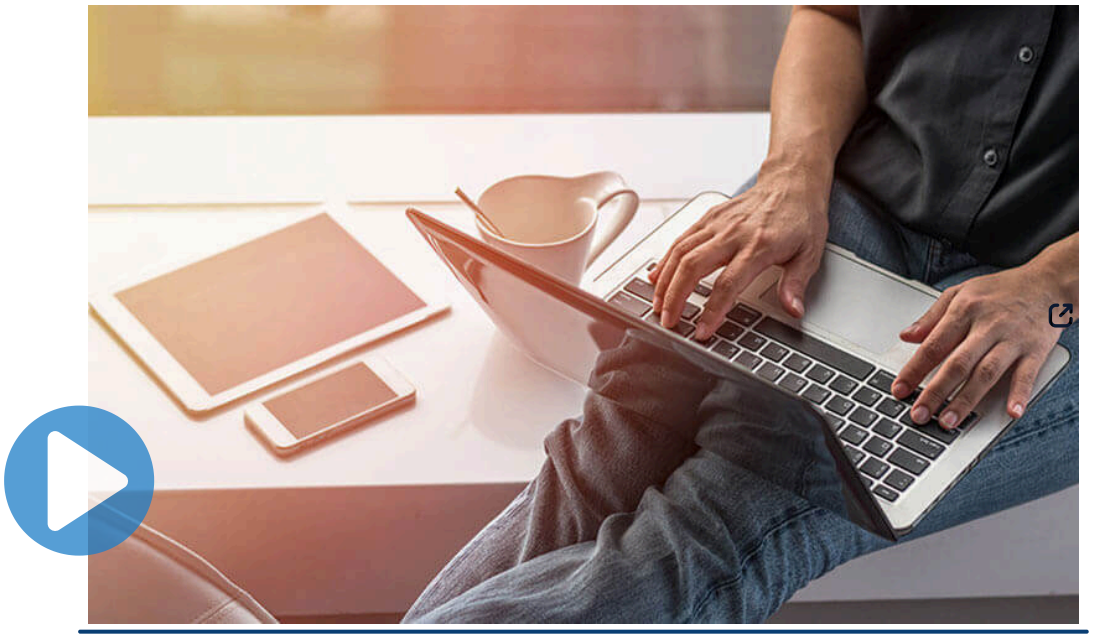
Tips

- Always test your Python code locally before uploading to Docker and then Fusion. This simplifies troubleshooting significantly.
- Once you've created your Docker you can also test locally by doing `docker run` with a specified port, like 9000, which you can then `curl` to confirm functionality in Fusion. See the testing example below.

LucidAcademy

Lucidworks offers free training to help you get started.

The **Course** for **Intro to Machine Learning in Fusion** focuses on using machine learning to to infer the goals of customers and users in order to deliver a more sophisticated search experience:



Visit the LucidAcademy to see the full training catalog.

Local testing example

The examples in this section use the following models:

- multilingual miniLM model
- e5-model

1. Docker command:

```
docker run -p 127.0.0.1:9000:9000 <your-docker-image>
```

2. Curl to hit Docker:

```
curl -X POST -H 'Content-Type: application/json' -d
'{"data": { "ndarray": ["Sentence to test"], "names":
["text"]} }'
https://localhost:9000/api/v1.0/predictions
```

3. Curl model in Fusion:

```
curl -u $FUSION_USER:$FUSION_PASSWORD -X POST -H
'Content-Type: application/json' -d '{"text": "i love
fusion"}' https://<your-
fusion>.lucidworks.com:6764/api/ai/ml-models/<your-
model>/prediction
```

4. See all your deployed models:

```
https://<your-fusion-host>/api/ai/ml-models
```

1. Download the model

This tutorial uses the paraphrase-multilingual-MiniLM-L12-v2 model from Hugging Face, but any pre-trained model from https://huggingface.co will work with this tutorial.

If you want to use your own model instead, you can do so, but your model must have been trained and then saved though a function similar to the PyTorch's `torch.save(model, PATH)` function. See Saving and Loading Models in the PyTorch documentation.

2. Format a Python class

The next step is to format a Python class which will be invoked by Fusion to get the results from your model. The skeleton below represents the format that you should follow. See also

Packaging a Python model for Seldon Core using Docker in the Seldon Core documentation.

```
class MyModel(object): """
    Model template. You can load your model parameters in __init__
from a
    location accessible at runtime
    """ def __init__(self): """
        Add any initialization parameters. These will be passed at
runtime
        from the graph definition parameters defined in your
seldondeployment
        kubernetes resource manifest.
        """ print("Initializing") def
predict(self,X,features_names,**kwargs): """
        Return a prediction.

        Parameters
        ----------
        X : array-like
        feature_names : array of feature names (optional)
        """ print("Predict called - will run identity function")
return X def class_names(self): return ["X_name"]
```

A real instance of this class with the Paraphrase Multilingual MiniLM L12 v2 model is as follows:

```
import logging import os from transformers import AutoTokenizer,
AutoModel from torch.nn import functional as F from typing import
Iterable import numpy as np import torch log = logging.getLogger()
class mini(): def __init__(self): self.tokenizer =
AutoTokenizer.from_pretrained('sentence-transformers/paraphrase-
multilingual-MiniLM-L12-v2') self.model=
AutoModel.from_pretrained('sentence-transformers/paraphrase-
multilingual-MiniLM-L12-v2') #Mean Pooling def mean_pooling(self,
model_output, attention_mask): token_embeddings = model_output[0]
#First element of model_output contains all token embeddings
input_mask_expanded =
attention_mask.unsqueeze(-1).expand(token_embeddings.size()).float()
return torch.sum(token_embeddings * input_mask_expanded, 1) /
torch.clamp(input_mask_expanded.sum(1), min=1e-9) def predict(self,
X:np.ndarray, names=None, **kwargs): # In Fusion there are several
variables passed in the numpy array with the Milvus Query stage, #
Encode to Milvus index stage, and Vectorize Seldon index and query
stage: # [pipeline, bool, and text]. Text is what variable will be
encoded, so that is what will be set to 'text' # When using the
Machine Learning stage, the input map keys should match what what is
in this file. model_input = dict(zip(names, X)) text =
model_input["text"] with torch.inference_mode(): # Allows torch to run
more quickly # Tokenize sentences encoded_input = self.tokenizer(text,
padding=True, truncation=True, return_tensors='pt') log.debug('encoded
input',str(encoded_input)) model_output = self.model(**encoded_input)
log.debug('model output',str(model_output)) # Perform pooling. In this
case, max pooling. sentence_embeddings =
self.mean_pooling(model_output, encoded_input['attention_mask']) #
Normalize embeddings, because Fusion likes it that way.
sentence_embeddings = F.normalize(sentence_embeddings, p=2, dim=-1) #
Fixing the shape of the emebbedings to match (1, 384). final =
[sentence_embeddings.squeeze().cpu().detach().numpy().tolist()] return
final def class_names(self) -> Iterable[str]: return ["vector"]
```

In the above code, an additional function has been added in the class; this is completely fine to do. Logging has also been added for debugging purposes.

Two functions are non-negotiable:

- `init`
  The `init` function is where models, tokenizers, vectorizers, and the like should be set to self for invoking. It is recommended that you include your model's trained parameters directly into the Docker container rather than reaching out to external storage inside `init`.
- `predict`
  The `predict` function processes the field or query that Fusion passes to the model. The `predict` function must be able to handle any text processing needed for the model to accept input invoked in its `model.evaluate()`, `model.predict()`, or equivalent function to get the expected model result.
  If the output needs additional manipulation, that should be done before the result is returned. For embedding models the return value must have the shape of (1, DIM), where DIM (dimension) is a consistent integer, to enable Fusion to handle the vector encoding into Milvus or Solr.

Note

Use the *exact* name of the class when naming this file. For the example, above the Python file is named `mini.py` and the class name is `mini()`.

3. Create a Dockerfile

The next step is to create a Dockerfile. The Dockerfile should follow this general outline; read the comments for additional details:

```
#It is important that python version is 3.x-slim for seldon-core FROM
python:3.9-slim # Whatever directory(folder)the python file for your
python class, Dockerfile, and # requirements.txt is in should be
copied then denoted as the work directory. COPY . /app WORKDIR /app #
The requirements file for the Docker container COPY requirements.txt
requirements.txt RUN pip install --no-cache-dir -r requirements.txt #
Copy source code COPY . . # GRPC - Allows Fusion to do a Remote
Procedure Call EXPOSE 5000 # Define environment variable for seldon-
core # !!!MODEL_NAME must be the EXACT same as the python file &
python class name!!! ENV MODEL_NAME mini ENV SERVICE_TYPE MODEL ENV
PERSISTENCE 0 # Changing active directory folder (same one as above on
lines 5 & 6) to default user, required for Fusion RUN chown -R 8888
/app # Command to wrap python class with seldon-core to allow it to be
usable in Fusion CMD ["sh", "-c", "seldon-core-microservice
$MODEL_NAME --service-type $SERVICE_TYPE --persistence $PERSISTENCE"]
# You can use the following if You need shell features like
environment variable expansion or # You need to use shell constructs
like pipes, redirects, etc. # See
https://docs.docker.com/reference/dockerfile/#cmd for more details. #
CMD exec seldon-core-microservice $MODEL_NAME --service-type
$SERVICE_TYPE --persistence $PERSISTENCE
```

4. Create a requirements file

The **`requirements.txt`** file is a list of installs for the **`Dockerfile`** to run to ensure the Docker container has the right resources to run the model. For the Paraphrase Mulitilingual MiniLM L12 v2 model, the requirements are as follows:

```
seldon-core torch transformers numpy
```

In general, if an item was used in an **`import`** statement in your Python file, it should be included in the requirements file.

An easy way to populate the requirements is by using in the following command in the terminal, inside the directory that contains your code:

```
pip freeze > requirements.txt
```

Note

If you use `pip freeze`, you must manually add `seldon-core` to the requirements file because it is not invoked in the Python file but is required for containerization.

5. Build and push the Docker image

After creating the `<your_model>.py`, `Dockerfile`, and `requirements.txt` files, you need to run a few Docker commands. Run the commands below in order:

```
DOCKER_DEFAULT_PLATFORM=linux/amd64 docker build . -t [DOCKERHUB-
USERNAME]/[REPOSITORY]:[VERSION-TAG]
```

```
docker push [DOCKERHUB USERNAME]/[REPOSITORY]:[VERSION-TAG]
```

Using the example model, the terminal commands would be as follows:

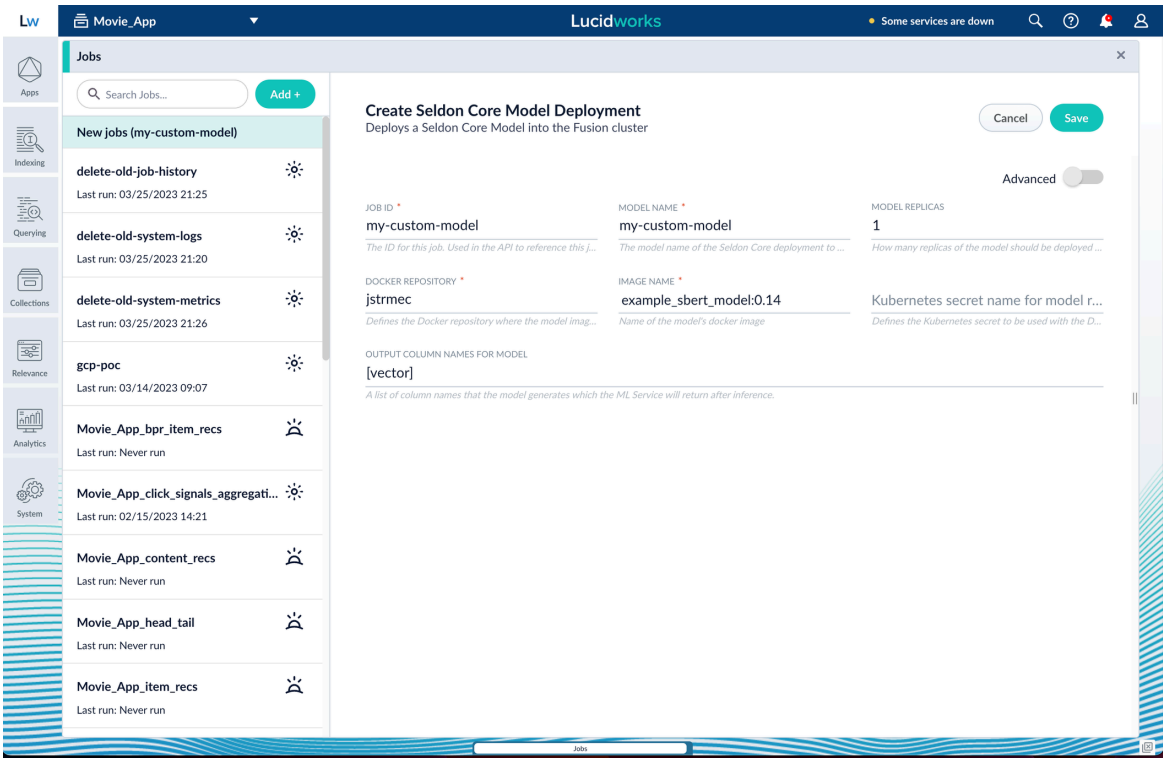
```
DOCKER_DEFAULT_PLATFORM=linux/amd64 docker build . -t
jstrmec/example_sbert_model:0.14; docker push
jstrmec/example_sbert_model:0.14
```

This repository is public and you can visit it here: example_sbert_model

6. Deploy the model in Fusion

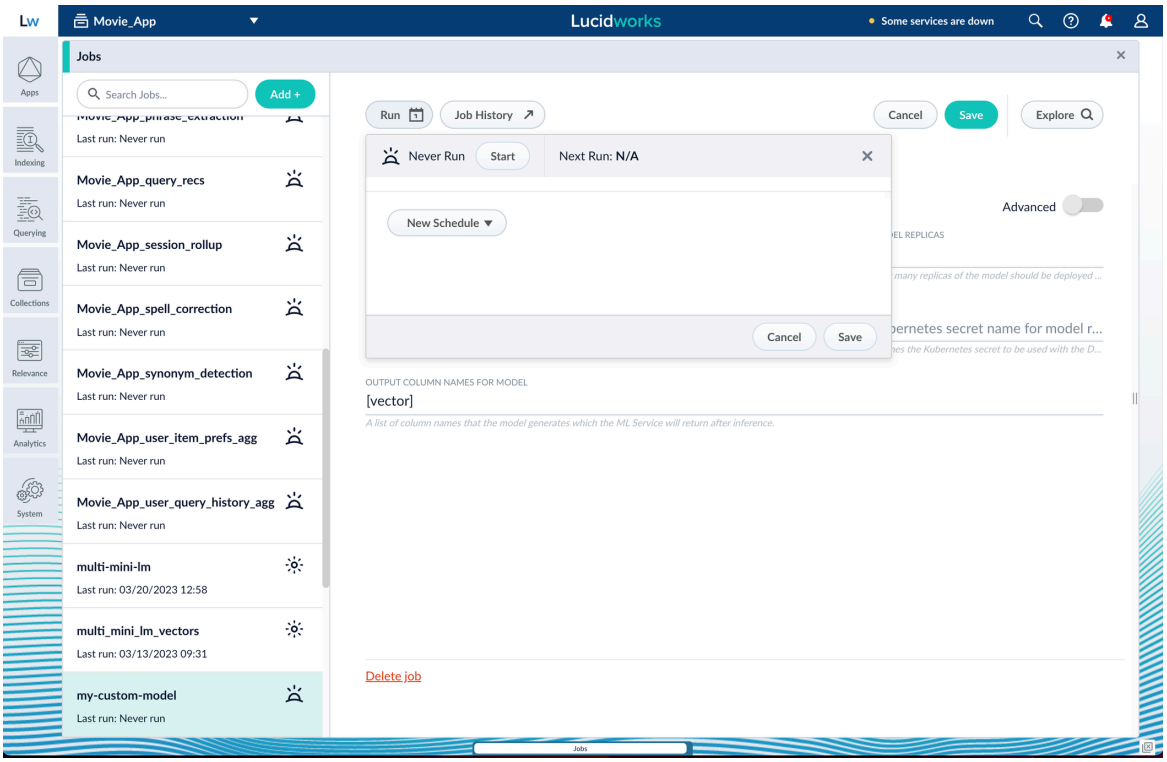
Now you can go to Fusion to deploy your model.

1. In Fusion, navigate to **Collections** > **Jobs**.
2. Add a job by clicking the **Add+** Button and selecting **Create Seldon Core Model Deployment**.
3. Fill in each of the text fields:



Parameter	Description
Job ID	A string used by the Fusion API to reference the job after its creation.
Model name	A name for the deployed model. This is used to generate the deployment name in Seldon Core. It is also the name that you reference as a `model-id` when making predictions with the ML Service.
Model replicas	The number of load-balanced replicas of the model to deploy; specify multiple replicas for a higher-volume intake.
Docker Repository	The public or private repository where the Docker image is located. If you're using Docker Hub, fill in the Docker Hub username here.
Image name	The name of the image with an optional tag. If no tag is given, `latest` is used.
Kubernetes secret	If you're using a private repository, supply the name of the Kubernetes secret used for access.
Output columns	A list of column names that the model's `predict` method returns.

4. Click **Save**, then **Run** and **Start**.



When the job finishes successfully, you can proceed to the next section.

Now that the model is in Fusion, it can be utilized in either index or query pipelines, depending on the model's purpose. In this case the model is a word vectorizer or semantic vector search implementation, so both pipelines must invoke the model.

7. Apply an API key to the deployment

These steps are only needed if your model utilizes any kind of secret, such as an API key. If not, skip this section and proceed to the next.

1. Create and modify a `<seldon_model_name>_sdep.yaml` file.
   In the first line, `kubectl get sdep` gets the details for the currently running Seldon Deployment job and saves those details to a YAML file. `kubectl apply -f open_sdep.yaml` adds the key to the Seldon Deployment job the next time it launches.

```
kubectl get sdep <seldon_model_name> -o yaml >
<seldon_model_name>_sdep.yaml # Modify
<seldon_model_name>_sdep.yaml to add - env: - name: API_KEY
value: "your-api-key-here" kubectl apply -f
<seldon_model_name>_sdep.yaml
```

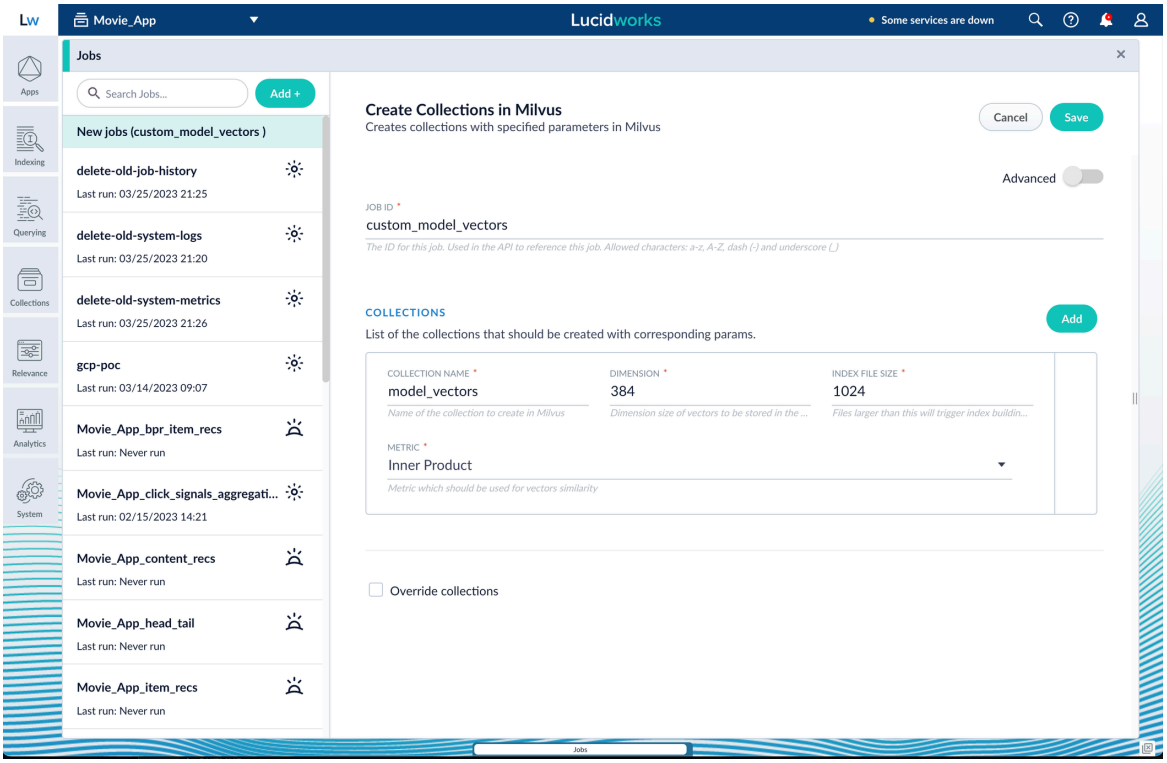
2. Delete `sdep` before redeploying the model. The currently running Seldon Deployment job does not have the key applied to it. Delete it before redeploying and the new job will have the key.

```
kubectl delete sdep <seldon_model_name>
```

3. Lastly, you can *encode into Milvus*.

8. Create a Milvus collection

1. In Fusion, navigate to **Collections** > **Jobs**.
2. Click the **Add+** Button and select **Create Collections in Milvus**.
   This job creates a collection in Milvus for storing the vectors sent to it. The job is needed because a collection does not automatically spawn at indexing or query time if it does not already exist.
3. Name the job and the collection.
4. Click **Add** on the right side of the job panel.
   The key to creating the collection is the **Dimension** text field; this must exactly match the shape value your output prediction has. In our example the shape is (1,384), so 384 will be in the collections **Dimension** field:



   The **Metric** field should typically be left at the default of `Inner Product`, but this also depends on use case and model type.
5. Click **Save**, then **Run** and **Start**.

9. Configure the Fusion pipelines

Your real-world pipeline configuration depends on your use case and model, but for our example we will configure the index pipeline and then the query pipeline.

Configure the index pipeline

1. Create a new index pipeline or load an existing one for editing.
2. Click **Add a Stage** and then **Encode to Milvus**.
3. In the new stage, fill in these fields:
   - The name of your model

- The output name you have for your model job
- The field you'd like to encode
- The collection name

4. Save the stage in the pipeline and index your data with it.

Configure the query pipeline

1. Create a new query pipeline or load an existing one for editing.
2. Click **Add a Stage** and then **Milvus Query**.
3. Fill in the configuration fields, then save the stage.
4. Add a **Milvus Ensemble Query** stage.
   This stage is necessary to have the Milvus collection scores taken into account in ranking and to weight multiple collections. The **Milvus Results Context Key** from the Milvus Query Stage is used in this stage to preform math on the Milvus result scores. One (1) is a typical multiplier for the Milvus results but any number can be used.
5. Save the stage and then run a query by typing a search term.
6. To verify the Milvus results are correct, use the **Compare+** button to see another pipeline without the model implementation and compare the number of results.

You have now successfully uploaded a Seldon Core model to Fusion and deployed it.

Develop and Deploy a Machine Learning Model

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

This tutorial walks you through deploying your own model to Managed Fusion with Seldon Core.

Prerequisites

- A Managed Fusion instance with an app and indexed data
- An understanding of Python and the ability to write Python code
- Docker installed locally, plus a private or public Docker repository
- Seldon-core installed locally: `pip install seldon-core`
- Code editor; you can use any editor, but Visual Studio Code is used in the example
- Model: paraphrase-multilingual-MiniLM-L12-v2 from Hugging Face
- Docker image: example_sbert_model

Tips

- Always test your Python code locally before uploading to Docker and then Managed Fusion. This simplifies troubleshooting significantly.
- Once you've created your Docker you can also test locally by doing `docker run` with a specified port, like 9000, which you can then `curl` to confirm functionality in Managed Fusion. See the testing example below.

Local testing example

The examples in this section use the following models:

- multilingual miniLM model
- e5-model

1. Docker command:

```
docker run -p 127.0.0.1:9000:9000 <your-docker-image>
```

2. Curl to hit Docker:

```
curl -X POST -H 'Content-Type: application/json' -d
'{"data": { "ndarray": ["Sentence to test"], "names":
["text"]} }' https://localhost:9000/api/.0/predictions
```

3. Curl model in Managed Fusion:

```
curl -u $FUSION_USER:$FUSION_PASSWORD -X POST -H
'Content-Type: application/json' -d '{"text": "i love
fusion"}' https://<your-
fusion>.lucidworks.com:6764/api/ai/ml-models/<your-
model>/prediction
```

4. See all your deployed models:

```
https://<your-fusion-host>/api/ai/ml-models
```

1. Download the model

This tutorial uses the paraphrase-multilingual-MiniLM-L12-v2 model from Hugging Face, but any pre-trained model from https://huggingface.co will work with this tutorial.

If you want to use your own model instead, you can do so, but your model must have been trained and then saved though a function similar to the PyTorch's `torch.save(model, PATH)` function. See Saving and Loading Models in the PyTorch documentation.

2. Format a Python class

The next step is to format a Python class which will be invoked by Managed Fusion to get the results from your model. The skeleton below represents the format that you should follow. See also Packaging a Python model for Seldon Core using Docker in the Seldon Core documentation.

```
class MyModel(object): """
    Model template. You can load your model parameters in __init__
from a
    location accessible at runtime
    """ def __init__(self): """
        Add any initialization parameters. These will be passed at
runtime
        from the graph definition parameters defined in your
seldondeployment
        kubernetes resource manifest.
        """ print("Initializing") def
predict(self,X,features_names,**kwargs): """
        Return a prediction.

        Parameters
        ----------
        X : array-like
        feature_names : array of feature names (optional)
        """ print("Predict called - will run identity function")
return X def class_names(self): return ["X_name"]
```

A real instance of this class with the Paraphrase Multilingual MiniLM L12 v2 model is as follows:

```
import logging import os from transformers import AutoTokenizer,
AutoModel from torch.nn import functional as F from typing import
Iterable import numpy as np import torch log = logging.getLogger()
class mini(): def __init__(self): self.tokenizer =
AutoTokenizer.from_pretrained('sentence-transformers/paraphrase-
multilingual-MiniLM-L12-v2') self.model=
AutoModel.from_pretrained('sentence-transformers/paraphrase-
multilingual-MiniLM-L12-v2') #Mean Pooling def mean_pooling(self,
model_output, attention_mask): token_embeddings = model_output[0]
#First element of model_output contains all token embeddings
input_mask_expanded =
attention_mask.unsqueeze(-1).expand(token_embeddings.size()).float()
return torch.sum(token_embeddings * input_mask_expanded, 1) /
torch.clamp(input_mask_expanded.sum(1), min=1e-9) def predict(self,
X:np.ndarray, names=None, **kwargs): # In Managed Fusion there are
several variables passed in the numpy array with the Milvus Query
stage, # Encode to Milvus index stage, and Vectorize Seldon index and
query stage: # [pipeline, bool, and text]. Text is what variable will
be encoded, so that is what will be set to 'text' # When using the
Machine Learning stage, the input map keys should match what what is
in this file. model_input = dict(zip(names, X)) text =
model_input["text"] with torch.inference_mode(): # Allows torch to run
more quickly # Tokenize sentences encoded_input = self.tokenizer(text,
padding=True, truncation=True, return_tensors='pt') log.debug('encoded
input',str(encoded_input)) model_output = self.model(**encoded_input)
log.debug('model output',str(model_output)) # Perform pooling. In this
case, max pooling. sentence_embeddings =
self.mean_pooling(model_output, encoded_input['attention_mask']) #
Normalize embeddings, because Managed Fusion likes it that way.
sentence_embeddings = F.normalize(sentence_embeddings, p=2, dim=-1) #
Fixing the shape of the emebbedings to match (1, 384). final =
[sentence_embeddings.squeeze().cpu().detach().numpy().tolist()] return
final def class_names(self) -> Iterable[str]: return ["vector"]
```

In the above code, an additional function has been added in the class; this is completely fine to do. Logging has also been added for debugging purposes.

Two functions are non-negotiable:

- `init`
  The `init` function is where models, tokenizers, vectorizers, and the like should be set to self for invoking. It is recommended that you include your model's trained parameters directly into the Docker container rather than reaching out to external storage inside `init`.
- `predict`
  The `predict` function processes the field or query that Managed Fusion passes to the model. The `predict` function must be able to handle any text processing needed for the model to accept input invoked in its `model.evaluate()`, `model.predict()`, or equivalent function to get the expected model result.
  If the output needs additional manipulation, that should be done before the result is returned. For embedding models the return value must have the shape of (1, DIM), where DIM (dimension) is a consistent integer, to enable Managed Fusion to handle the vector encoding into Milvus or Solr.

Note

Use the *exact* name of the class when naming this file. For the example, above the Python file is named `mini.py` and the class name is `mini()`.

3. Create a Dockerfile

The next step is to create a Dockerfile. The Dockerfile should follow this general outline; read the comments for additional details:

```
#It is important that python version is 3.x-slim for seldon-core FROM
python:3.9-slim # Whatever directory(folder)the python file for your
python class, Dockerfile, and # requirements.txt is in should be
copied then denoted as the work directory. COPY . /app WORKDIR /app #
The requirements file for the Docker container COPY requirements.txt
requirements.txt RUN pip install --no-cache-dir -r requirements.txt #
Copy source code COPY . . # GRPC - Allows Managed Fusion to do a
Remote Procedure Call EXPOSE 5000 # Define environment variable for
seldon-core # !!!MODEL_NAME must be the EXACT same as the python file
& python class name!!! ENV MODEL_NAME mini ENV SERVICE_TYPE MODEL ENV
PERSISTENCE 0 # Changing active directory folder (same one as above on
lines 5 & 6) to default user, required for Managed Fusion RUN chown -R
8888 /app # Command to wrap python class with seldon-core to allow it
to be usable in Managed Fusion CMD ["sh", "-c", "seldon-core-
microservice $MODEL_NAME --service-type $SERVICE_TYPE --persistence
$PERSISTENCE"] # You can use the following if You need shell features
like environment variable expansion or # You need to use shell
constructs like pipes, redirects, etc. # See
https://docs.docker.com/reference/dockerfile/#cmd for more details. #
CMD exec seldon-core-microservice $MODEL_NAME --service-type
$SERVICE_TYPE --persistence $PERSISTENCE
```

4. Create a requirements file

The **`requirements.txt`** file is a list of installs for the **`Dockerfile`** to run to ensure the Docker container has the right resources to run the model. For the Paraphrase Mulitilingual MiniLM L12 v2 model, the requirements are as follows:

```
seldon-core torch transformers numpy
```

In general, if an item was used in an **`import`** statement in your Python file, it should be included in the requirements file.

An easy way to populate the requirements is by using in the following command in the terminal, inside the directory that contains your code:

```
pip freeze > requirements.txt
```

Note

If you use `pip freeze`, you must manually add `seldon-core` to the requirements file because it is not invoked in the Python file but is required for containerization.

5. Build and push the Docker image

After creating the `<your_model>.py`, `Dockerfile`, and `requirements.txt` files, you need to run a few Docker commands. Run the commands below in order:

```
DOCKER_DEFAULT_PLATFORM=linux/amd64 docker build . -t [DOCKERHUB-USERNAME]/[REPOSITORY]:[VERSION-TAG]
```

```
docker push [DOCKERHUB USERNAME]/[REPOSITORY]:[VERSION-TAG]
```

Using the example model, the terminal commands would be as follows:

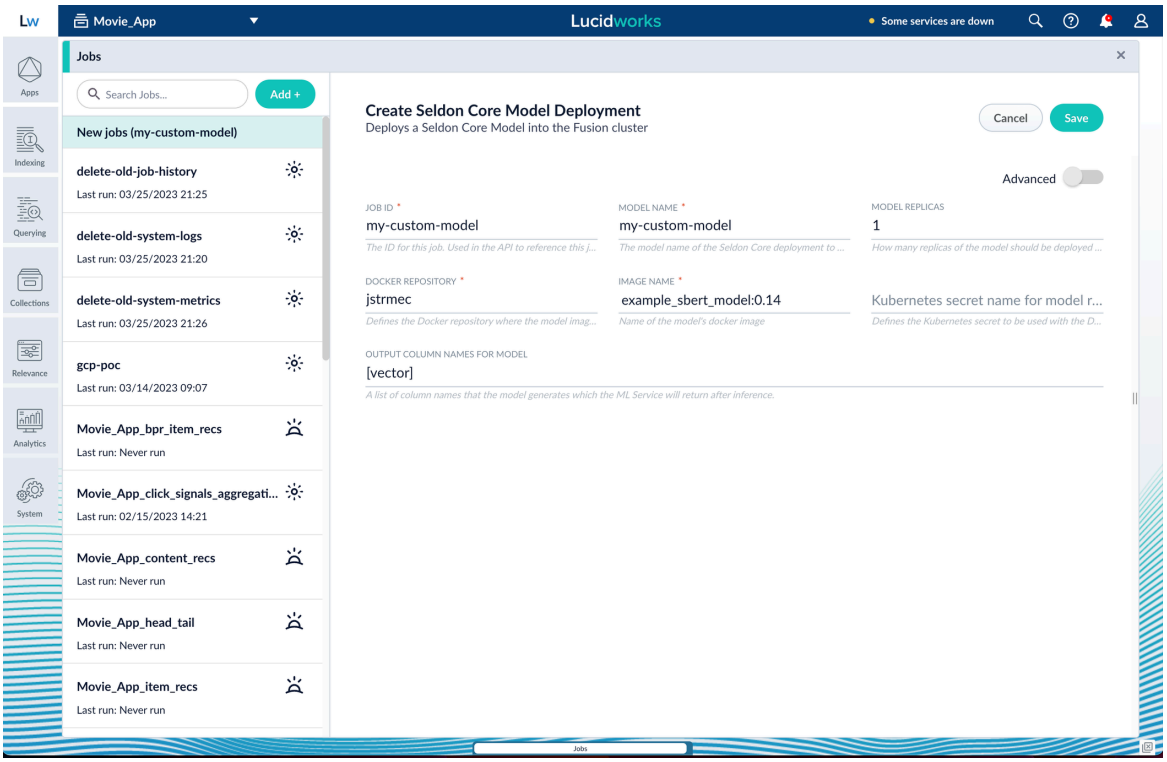
```
DOCKER_DEFAULT_PLATFORM=linux/amd64 docker build . -t jstrmec/example_sbert_model:0.14; docker push jstrmec/example_sbert_model:0.14
```

This repository is public and you can visit it here: example_sbert_model

6. Deploy the model in Managed Fusion

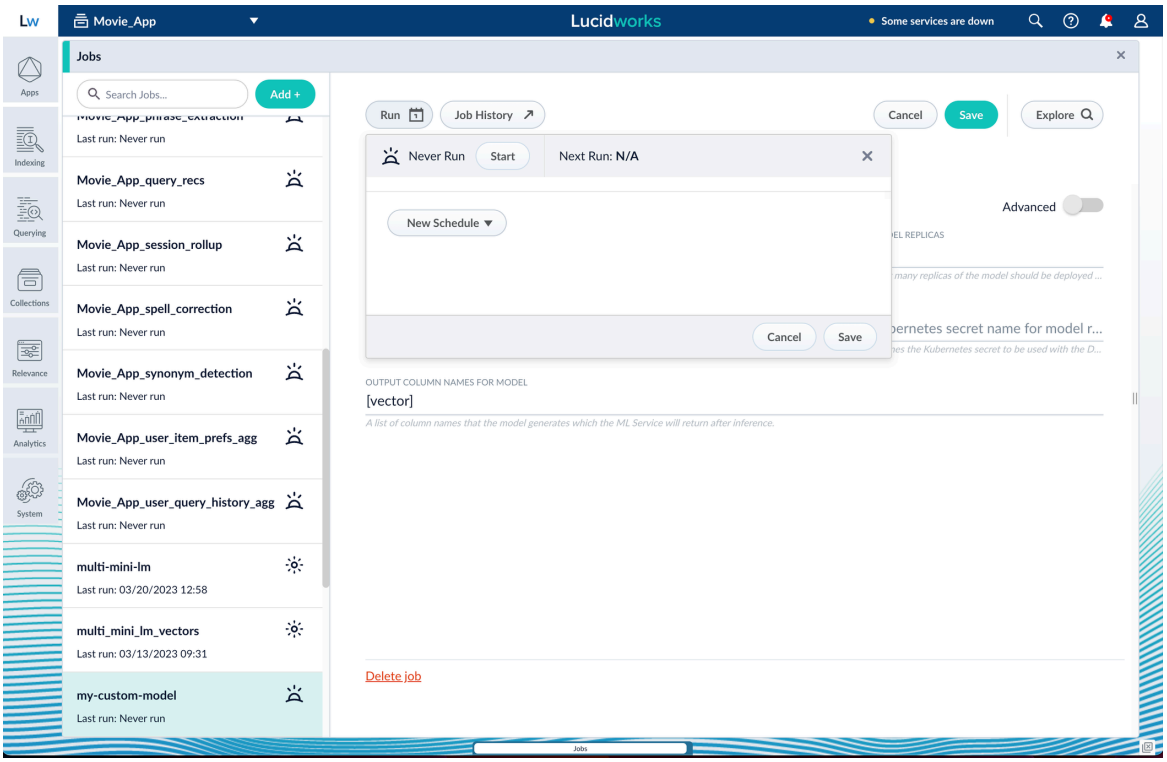
Now you can go to Managed Fusion to deploy your model.

1. In Managed Fusion, navigate to **Collections** > **Jobs**.
2. Add a job by clicking the **Add+** Button and selecting **Create Seldon Core Model Deployment**.
3. Fill in each of the text fields:



Parameter	Description
Job ID	A string used by the Managed Fusion API to reference the job after its creation.
Model name	A name for the deployed model. This is used to generate the deployment name in Seldon Core. It is also the name that you reference as a `model-id` when making predictions with the ML Service.
Model replicas	The number of load-balanced replicas of the model to deploy; specify multiple replicas for a higher-volume intake.
Docker Repository	The public or private repository where the Docker image is located. If you're using Docker Hub, fill in the Docker Hub username here.
Image name	The name of the image with an optional tag. If no tag is given, `latest` is used.
Kubernetes secret	If you're using a private repository, supply the name of the Kubernetes secret used for access.
Output columns	A list of column names that the model's `predict` method returns.

4. Click **Save**, then **Run** and **Start**.



When the job finishes successfully, you can proceed to the next section.

Now that the model is in Managed Fusion, it can be utilized in either index or query pipelines, depending on the model's purpose. In this case the model is a word vectorizer or semantic vector search implementation, so both pipelines must invoke the model.

7. Apply an API key to the deployment

These steps are only needed if your model utilizes any kind of secret, such as an API key. If not, skip this section and proceed to the next.

1. Create and modify a `<seldon_model_name>_sdep.yaml` file.
   In the first line, `kubectl get sdep` gets the details for the currently running Seldon Deployment job and saves those details to a YAML file. `kubectl apply -f open_sdep.yaml` adds the key to the Seldon Deployment job the next time it launches.

```
kubectl get sdep <seldon_model_name> -o yaml >
<seldon_model_name>_sdep.yaml # Modify
<seldon_model_name>_sdep.yaml to add - env: - name: API_KEY
value: "your-api-key-here" kubectl apply -f
<seldon_model_name>_sdep.yaml
```

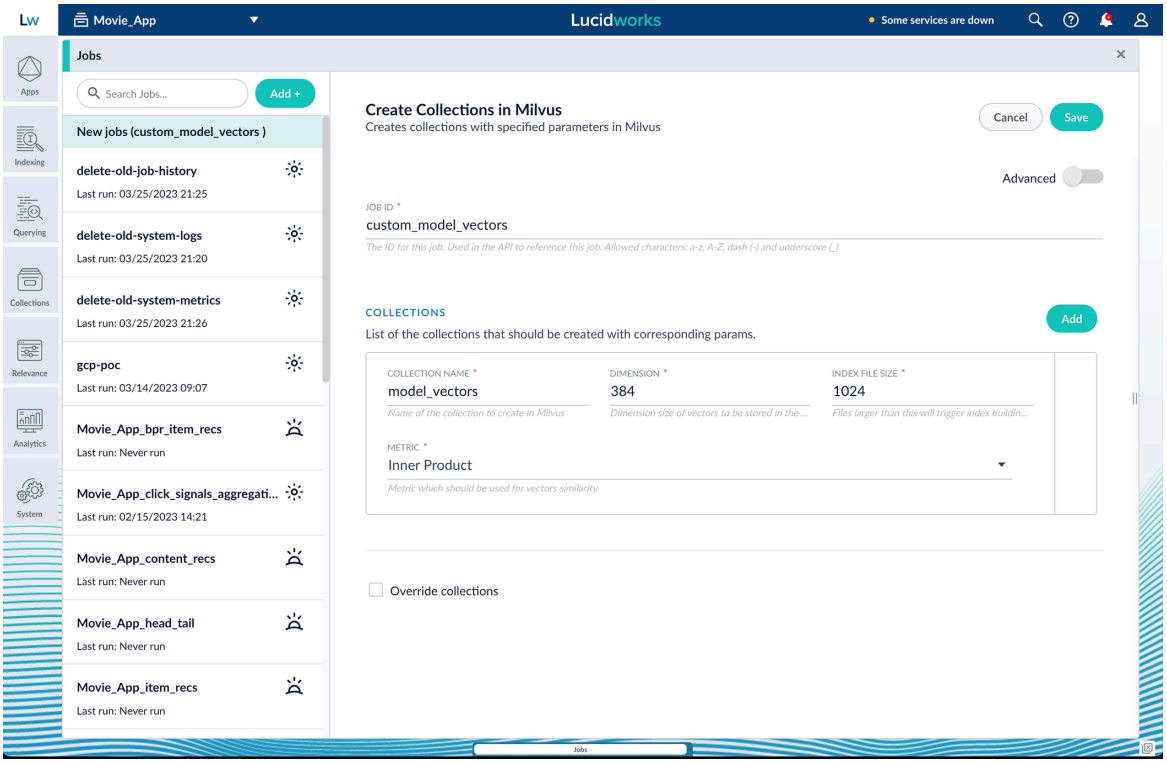
2. Delete `sdep` before redeploying the model. The currently running Seldon Deployment job does not have the key applied to it. Delete it before redeploying and the new job will have the key.

```
kubectl delete sdep <seldon_model_name>
```

3. Lastly, you can *encode into Milvus*.

8. Create a Milvus collection

1. In Managed Fusion, navigate to **Collections** > **Jobs**.
2. Click the **Add+** Button and select **Create Collections in Milvus**.
   This job creates a collection in Milvus for storing the vectors sent to it. The job is needed because a collection does not automatically spawn at indexing or query time if it does not already exist.
3. Name the job and the collection.
4. Click **Add** on the right side of the job panel.
   The key to creating the collection is the **Dimension** text field; this must exactly match the shape value your output prediction has. In our example the shape is (1,384), so 384 will be in the collections **Dimension** field:



   The **Metric** field should typically be left at the default of `Inner Product`, but this also depends on use case and model type.
5. Click **Save**, then **Run** and **Start**.

9. Configure the Managed Fusion pipelines

Your real-world pipeline configuration depends on your use case and model, but for our example we will configure the index pipeline and then the query pipeline.

Configure the index pipeline

1. Create a new index pipeline or load an existing one for editing.
2. Click **Add a Stage** and then **Encode to Milvus**.
3. In the new stage, fill in these fields:
   - The name of your model

- The output name you have for your model job
- The field you'd like to encode
- The collection name

4. Save the stage in the pipeline and index your data with it.

Configure the query pipeline

1. Create a new query pipeline or load an existing one for editing.
2. Click **Add a Stage** and then **Milvus Query**.
3. Fill in the configuration fields, then save the stage.
4. Add a **Milvus Ensemble Query** stage.
   This stage is necessary to have the Milvus collection scores taken into account in ranking and to weight multiple collections. The **Milvus Results Context Key** from the Milvus Query Stage is used in this stage to preform math on the Milvus result scores. One (1) is a typical multiplier for the Milvus results but any number can be used.
5. Save the stage and then run a query by typing a search term.
6. To verify the Milvus results are correct, use the **Compare+** button to see another pipeline without the model implementation and compare the number of results.

You have now successfully uploaded a Seldon Core model to Managed Fusion and deployed it.

Develop and deploy a machine learning model with Ray

How-to

> Compatible versions: 5.9
>
> Works with: *Fusion*, *Managed Fusion*

This tutorial walks you through deploying your own model to Fusion with Ray.

> **Note**
>
> This feature is only available in Fusion 5.9.x for versions 5.9.12 and later.

Prerequisites

- A Fusion instance with an app and indexed data.
- An understanding of Python and the ability to write Python code.
- Docker installed locally, plus a private or public Docker repository.
- Ray installed locally: `pip install ray[serve]`.
- Code editor; you can use any editor, but Visual Studio Code is used in this example.
- Model: intfloat/e5-small-v2
- Docker image: e5-small-v2-ray

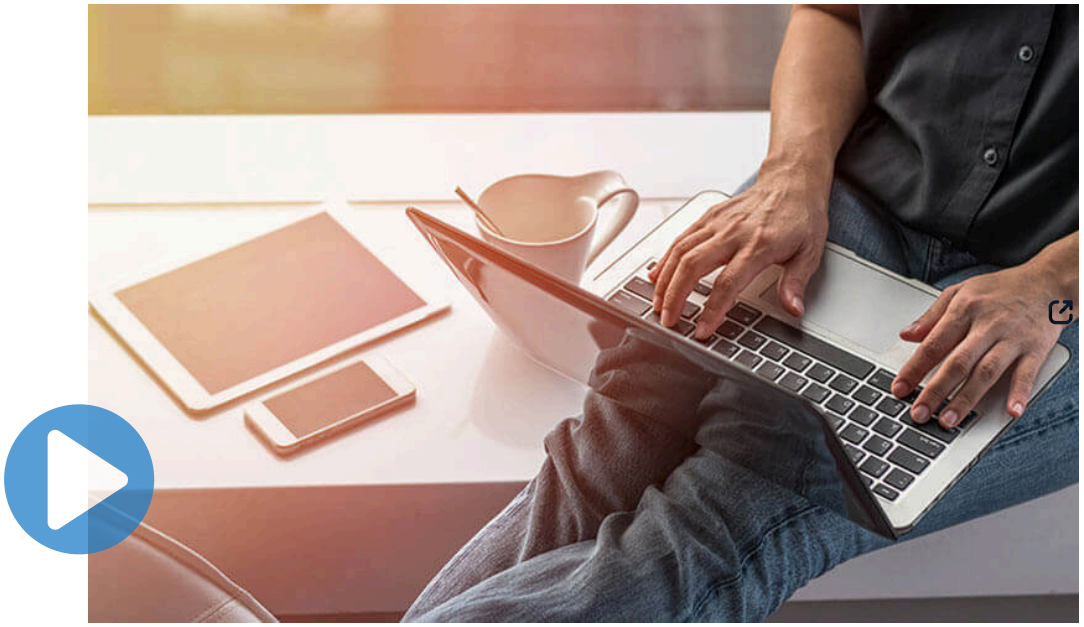
Tips

- Always test your Python code locally before uploading to Docker and then Fusion. This simplifies troubleshooting significantly.
- Once you've created your Docker you can also test locally by doing `docker run` with a specified port, like 9000, which you can then `curl` to confirm functionality in Fusion. See the testing example below.
- If you previously deployed a model with Seldon, you can deploy the same model with Ray after making a few changes to your Docker image as explained in this topic. To avoid conflicts, deploy the model with a different name. When you have verified that the model is working after deployment with Ray, you can delete the Seldon model using the *Delete Seldon Core Model Deployment job*.
- If you run into an issue with the model not deploying and you're using the 'real' example, there is a very good chance you haven't allocated enough memory or CPU in your job spec or in the Ray-Argo config. It's easy to increase the resources. To edit the ConfigMap, run `kubectl edit configmap argo-deploy-ray-model-workflow -n <namespace>` and then find the `ray-head` container in the artisanal escaped YAML and change the memory limit. Exercise caution when editing because it can break the YAML. Just delete and replace a single character at a time without changing any formatting.
    - For additional guidance, see the testing locally e5-model example.

LucidAcademy

Lucidworks offers free training to help you get started.

The **Course** for **Intro to Machine Learning in Fusion** focuses on using machine learning to to infer the goals of customers and users in order to deliver a more sophisticated search experience:



Visit the LucidAcademy to see the full training catalog.

Local testing example

1. Docker command:

```
docker run -p 127.0.0.1:9000:9000 DOCKER_IMAGE
```

2. Curl to hit Docker:

```
curl -i -X POST http://127.0.0.1:8000 -H 'Content-Type:
application/json' -d '{"text": "The quick brown fox jumps over
the lazy dog."}'
```

3. Curl model in Fusion:

```
curl -u $FUSION_USER:$FUSION_PASSWORD -X POST -H 'Content-Type:
application/json' -d '{"text": "i love fusion"}'
https://FUSION_HOST.com:6764/api/ai/ml-
models/MODEL_NAME/prediction
```

4. See all your deployed models:

```
https://FUSION_HOST.com/api/ai/ml-models
```

5. Check the Ray UI to see Replica State, Resources, and Logs.
   If you are getting an internal model error, the best way to see what is going on is to query via port-forwarding the model. The `MODEL_DEPLOYMENT` in the command below can be found with `kubectl get svc -n NAMESPACE`. It will have the same name as set in the model name in the **Create Ray Model Deployment** job.

```
kubectl -n NAMESPACE port-forward svc/MODEL_DEPLOYMENT-head-svc
8000:8000
```

   Once port-forwarding is successful, you can use the below cURL command to see the issue. At that point your worker logs should show helpful error messages.

```
curl --location 'http://127.0.0.1:8000/' \ --header 'charset:
utf-8' \ --header 'Content-Type: application/json' \ --data
'{"text": "i love fusion"}'
```

1. Download the model

This tutorial uses the `e5-small-v2` model from Hugging Face, but any pre-trained model from https://huggingface.co will work with this tutorial.

If you want to use your own model instead, you can do so, but your model must have been trained and then saved though a function similar to the PyTorch's `torch.save(model, PATH)` function. See Saving and Loading Models in the PyTorch documentation.

2. Format a Python class

The next step is to format a Python class which will be invoked by Fusion to get the results from your model. The skeleton below represents the format that you should follow. See also Getting Started in the Ray Serve documentation.

```
from ray import serve from starlette.requests import Request # These defaults are for the ray serve deployment # when running simply from docker. The 'Create Ray Model Deployment' # job can override these replicas and resources if needed. @serve.deployment(num_replicas=1, ray_actor_options={"num_cpus": 1}) class Deployment(object): def __init__(self): """
        Add any initialization parameters. Generally this is where you would load
        your model. This method will be called once when the deployment is created.
        """ print("Initializing") self.model = load_model() #faux code # This can be named as any method which takes a dictionary as input and returns a dictionary # as output. In this example, we are using the encode method to encode the # input text into a vector. def encode(self, input_dict: Dict[str, Any]) -> Dict[str, Any]: """
        This method will be called when the deployment is queried. It will receive
        the input data and should return the output data.
        """ text = input_dict["text"] embeddings = self.model.encode #faux code return { "vector": embeddings } # To use the 'Ray / Seldon Vectorize Field' stage, the output key should be `vector`, if using the 'Machine Learning' stage you must ensure the output key matches the output key in the 'Machine Learning' stage async def __call__(self, http_request: Request) -> Dict[str, Any]: input_dict: Dict[str, Any] = await http_request.json() return self.encode(input_dict=input_dict) # This will be the function you defined above, in this case encode app = Deployment.bind()
```

A real instance of this class with the **`e5-small-v2`** model is as follows:

Note

This code pulls from Hugging Face. To have the model load in the image without pulling from Hugging Face or other external sources, download the model weights into a folder name and change the model name to the folder name preceded by `./`.

```
import json import sys from time import time from typing import Any,
Dict import torch import torch.nn.functional as F from ray import
serve from starlette.requests import Request from starlette.responses
import JSONResponse from torch import Tensor from transformers import
AutoModel, AutoTokenizer HUB_MODEL_NAME = "intfloat/e5-small-v2"
@serve.deployment(num_replicas=1, ray_actor_options={"num_cpus": 1})
class Deployment(object): def __init__(self): from loguru import
logger self.logger = logger # Initializing logger self.logger.remove()
self.logger.add(sys.stdout, level="INFO", serialize=False,
colorize=True) # Initializing model self.logger.info("Loading
model...") self.tokenizer =
AutoTokenizer.from_pretrained(HUB_MODEL_NAME) self.model =
AutoModel.from_pretrained(HUB_MODEL_NAME) self.model.eval()
self.logger.info("Model initialization finished!") def encode(self,
input_dict: Dict[str, Any]) -> Dict[str, Any]: _start_time = time() #
Extracting text from input text = input_dict["text"] # Tokenization
tokenized_texts = self.tokenizer( text, max_length=512, padding=True,
truncation=True, return_tensors="pt", ) # Encoding with
torch.inference_mode(): # Forward pass of the model outputs =
self.model(**tokenized_texts) # Average pooling the last hidden states
embeddings = self.average_pool( outputs.last_hidden_state,
tokenized_texts["attention_mask"] ) # Normalizing embeddings
embeddings = F.normalize(embeddings, p=2, dim=1) # Converting into
output format output_dict = {"vector": embeddings.squeeze().tolist()}
prediction_time = (time() - _start_time) * 1000
self.logger.info(f"Time taken to make a prediction:
{prediction_time:.0f}ms") return output_dict async def __call__(self,
http_request: Request) -> Dict[str, Any]: try: input_dict: Dict[str,
Any] = await http_request.json() except UnicodeDecodeError: body_bytes
= await http_request.body() try: decoded = body_bytes.decode("utf-8",
errors="replace") input_dict = json.loads(decoded) except
json.JSONDecodeError: return JSONResponse({"error": "Invalid JSON"},
status_code=400) return self.encode(input_dict=input_dict)
```

```
@staticmethod def average_pool(last_hidden_states: Tensor,
attention_mask: Tensor) -> Tensor: last_hidden =
last_hidden_states.masked_fill( ~attention_mask[..., None].bool(), 0.0
) return last_hidden.sum(dim=1) / attention_mask.sum(dim=1)[..., None]
app = Deployment.bind()
```

In the preceding code, logging has been added for debugging purposes.

The preceding code example contains the following functions:

- **`__call__`**
  This function is non-negotiable.
- **`init`**
  The **`init`** function is where models, tokenizers, vectorizers, and the like should be set to self for invoking. It is recommended that you include your model's trained parameters directly into the Docker container rather than reaching out to external storage inside **`init`**.
- **`encode`**
  The **`encode`** function is where the field or query that is passed to the model from Fusion is processed. Alternatively, you can process it all in the ***`call`*** function, but it is cleaner not to. The **`encode`** function can handle any text processing needed for the model to accept input invoked in its **`model.predict()`** or equivalent function which gets the expected model result.

If the output needs additional manipulation, that should be done before the result is returned. For embedding models, the return value must have the shape of (1, DIM), where DIM (vector dimension) is a consistent integer, to enable Fusion to handle the vector encoding into Ray.

Note

Use the *exact* name of the class when naming this file. In the preceding example, the Python file is named **`deployment.py`** and the class name is **`Deployment()`**.

3. Create a Dockerfile

The next step is to create a Dockerfile. The Dockerfile should follow this general outline; read the comments for additional details:

```
#It is important that python version is 3.x-slim FROM python:3.10-slim
# Install dependencies RUN apt-get update && apt-get install -y wget #
Create working app directory RUN mkdir -p /app WORKDIR /app # Copy the
requirements file and install the dependencies COPY requirements.txt
/app RUN pip install -r requirements.txt --no-cache-dir # Copy source
code COPY deployment.py /app # Expose serving port for HTTP
communication with Fusion EXPOSE 8000 # The end of the command follows
module:application and the below value should be set in the RAY
DEPLOYMENT IMPORT PATH field in 'Create Ray Model Deployment' job CMD
exec serve run deployment:app
```

4. Create a requirements file

The `requirements.txt` file is a list of installs for the `Dockerfile` to run to ensure the Docker container has the right resources to run the model. For the `e5-small-v2` model, the requirements are as follows:

```
torch -f https://download.pytorch.org/whl/torch_stable.html # Make
sure that we download CPU version of PyTorch transformers loguru
ray[serve]==2.42.1
```

Any recent ray[serve] version should work, but the tested value and known supported version is 2.42.1. In general, if an item was used in an `import` statement in your Python file, it should be included in the requirements file.

To populate the requirements, use the following command in the terminal, inside the directory that contains your code:

```
pip freeze > requirements.txt
```

5. Build and push the Docker image

After creating the `MODEL_NAME.py`, `Dockerfile`, and `requirements.txt` files, you need to run a few Docker commands. Run the following commands in order:

```
DOCKER_BUILDKIT=1 docker build . -t [DOCKERHUB-USERNAME]/[REPOSITORY]:
[VERSION-TAG]
```

```
docker push [DOCKERHUB USERNAME]/[REPOSITORY]:[VERSION-TAG]
```

Using the example model, the terminal commands would be as follows:

```
DOCKER_BUILDKIT=1 docker build . -t jstrmec/e5-small-v2-ray:0.1
```

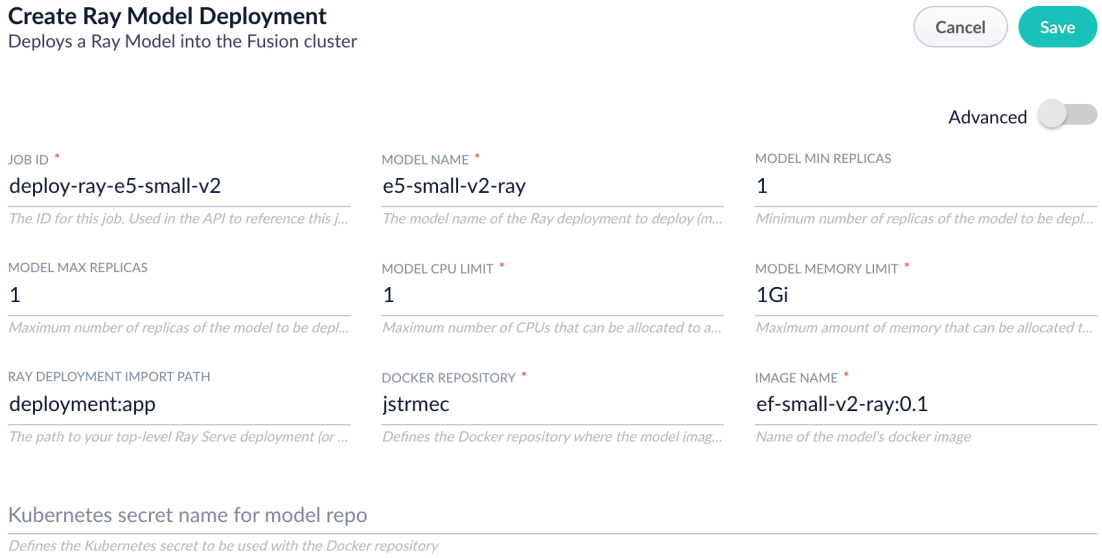
```
docker push jstrmec/e5-small-v2-ray:0.1
```

This repository is public and you can visit it here: e5-small-v2-ray

6. Deploy the model in Fusion

Now you can go to Fusion to deploy your model.

1. In Fusion, navigate to **Collections** > **Jobs**.
2. Add a job by clicking the **Add+** Button and selecting **Create Ray Model Deployment**.
3. Fill in each of the text fields:

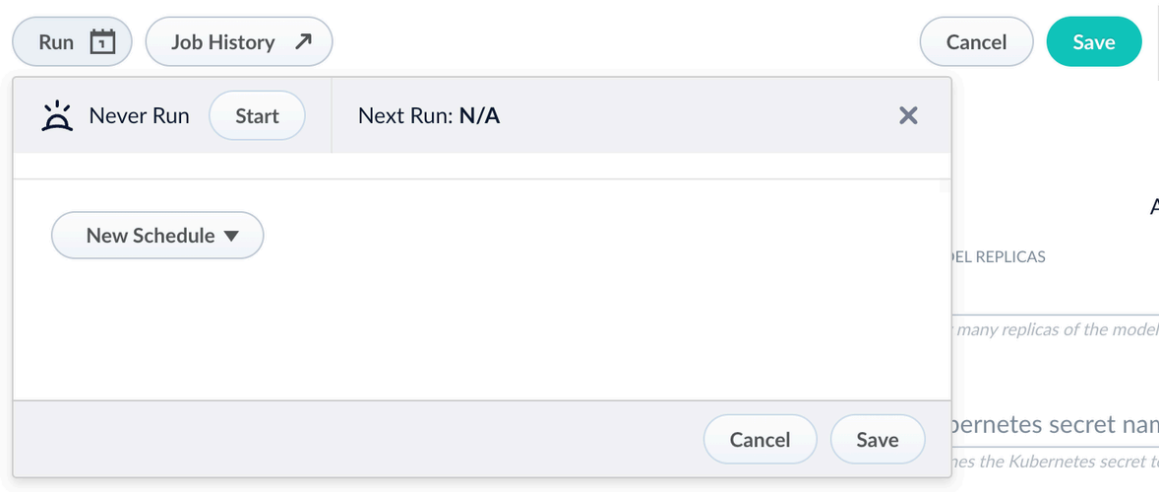


Parameter	Description
Job ID	A string used by the Fusion API to reference the job after its creation.
Model name	A name for the deployed model. This is used to generate the deployment name in Ray. It is also the name that you reference as a `model-id` when making predictions with the ML Service.
Model min replicas	The minimum number of load-balanced replicas of the model to deploy.
Model max replicas	The maximum number of load-balanced replicas of the model to deploy. Specify multiple replicas for a higher-volume intake.
Model CPU limit	The number of CPUs to allocate to a single model replica.
Model memory limit	The maximum amount of memory to allocate to a single model replica.
Ray Deployment Import Path	The path to your top-level Ray Serve deployment (or the same path passed to `serve run`). For example, `deployment:app`
Docker Repository	The public or private repository where the Docker image is located. If you're using Docker Hub, fill in the Docker Hub username here.
Image name	The name of the image. For example, `e5-small-v2-ray:0.1`.
Kubernetes secret	If you're using a private repository, supply the name of the Kubernetes secret used for access.

4. Click **Advanced** to view and configure advanced details:

Parameter	Description
Additional parameters.	This section lets you enter parameter `name:parameter` value options to be injected into the training JSON map at runtime. The values are inserted as they are entered, so you must surround string values with `"`. This is the sparkConfig field in the configuration file.
Write Options.	This section lets you enter parameter `name:parameter` value options to use when writing output to Solr or other sources. This is the writeOptions field in the configuration file.
Read Options.	This section lets you enter parameter `name:parameter` value options to use when reading input from Solr or other sources. This is the readOptions field in the configuration file.

5. Click **Save**, then **Run** and **Start**.



When the job finishes successfully, you can proceed to the next section.

Now that the model is in Fusion, you can use it in the Machine Learning or Ray / Seldon Vectorize index and query stages.

7. Configure the Fusion pipelines

Your real-world pipeline configuration depends on your use case and model, but for our example we will configure the index pipeline and then the query pipeline.

Configure the index pipeline

1. Create a new index pipeline or load an existing one for editing.
2. Click **Add a Stage** and then **Machine Learning**.
3. In the new stage, fill in these fields:
    - The model ID
    - The model input
    - The model output
4. Save the stage in the pipeline and index your data with it.

Configure the query pipeline

1. Create a new query pipeline or load an existing one for editing.
2. Click **Add a Stage** and then **Machine Learning**
3. In the new stage, fill in these fields:
    - The model ID
    - The model input

- The model output

4. Save the stage and then run a query by typing a search term.
5. To verify the Ray results are correct, use the **Compare+** button to see another pipeline without the model implementation and compare the number of results.

You have now successfully uploaded a Ray model to Fusion and deployed it.

Develop and Deploy with App Studio

How-to

> Compatible versions: 4.2
>
> Works with: *App Studio*

This topic shows you the App Studio workflow from initial app development through deployment into Fusion.

> **Note**
>
> This procedure does not apply to App Studio Enterprise (ASE).

Development tools and workflow

In building out a search application, App Studio developers use the following tools:

- an IDE (Intellij, eclipse, VS Code, and so on)
- Maven for server-side dependency management and its integrated Jetty plugin
- NPM 9.0.2 for client-side dependency management
- Java 1.8 JDK

The high-level development workflow looks like this:

1. Clone the repository of our project on our local system.
2. Make additions and edits to the code which we then commit to the repo.
3. Use the integrated Maven Jetty plugin to serve up our local application at `http://localhost:8080`, where we can see most changes made to the code in real time.

1. Build a profile (optional)

For each environment, we can build out a profile to overlay specific parameters to this environment:

Example command: `mvn clean jetty:run -Dtwigkit.profile=profileName`

In App Studio apps, we can have different profiles for each environment (Dev, QA and Prod) as well as local development. These profiles contain configuration files that should overwrite the files with the corresponding names and directory locations in the main resources directory.

For more details, see "Package the web application for different runtime environments - build profiles" at https://dev.twigkit.net/docs/Getting_started/Installation/Building_from_source/?q=profile.

2. Configure the profile (if any)

Following are the files you may want to specify for a given environment in each profile:

- `message.service.fusion.conf`
  This config file is responsible for the configuration of the way we send *signals* to the Fusion endpoint. The important property here is `query-profile`, which should point to the query profile that manages the main collection to which we want to send signals. For example, if we specify a query profile that points to the `myCollection` collection, then App Studio sends signals to the `myCollection_signals` collection (if it exists).
- `platforms.fusion.fusion.conf`
  This is our base Fusion platform configuration file. It manages all the configuration settings that are common amongst each Fusion platform. These settings are inherited by any config file in the `platforms.fusion` directory, as this file has the same name as the directory it resides in.
- `platforms.fusion.data.conf`
  This is our configuration file for the main platform, by default. The only required parameter is the `query-profile` to send these queries to. This is included in the profiles as during the course of development, there were some differences across environments in the name of this collection, so we set them individually for each environment.
- `security.kerberos.conf`
  This is an example of a security configuration file. In this case, the file in which Kerberos security is configured. The parameters set here are referenced in our configuration of `resources/spring-security.xml` when the app is loaded. The main parameters to understand are `service-principal` and `keytab-location`. The service principal should be the one used when creating the keytab file. The keytab location is the location in which the keytab resides on the server where App Studio is running.
- `security.security.conf`

The function of this file is to set general parameters related to security. The `type` parameter should be the type of security your app should use (`kerberos`, `spring_security`, `saml`, and so on). The `password` parameter is the key that is used when *hashing passwords using the* `twigcrypt` *utility*. It is to be used in combination with a 'secret seed' in order to generate the hash.

- `services.api.fusion.conf`
  This is the main file in which the connection to the Fusion API is configured. The host, port and protocol values are all given independently. The credentials given are that of a service account that has `POST` rights to enable the sending of signals to Fusion.
- `logback.xml`
  In this file, we can add additional `<logger>` tags to print out additional logging on specific classes in the web apps or local logs. The main logging comes from the root package `twigkit`.

3. Package the app

To deploy our code to the server, we use Maven to create a `WAR` file:

- If you are using a build profile as described above, run `mvn clean package -Dtwigkit.profile=profileName`.
- If you are not using a build profile, run `mvn clean package`.

You can watch the logs for any compilation errors.

Tip

Rename the resulting `WAR` file to `ROOT.war`, if it is not already named this way.

4. Deploy the app

Before you deploy an app, make sure that both of these prerequisites are met:

- Fusion is installed.
- A Fusion app is created.

Deploy the web app as follows:

1. Create a new App Studio context in Fusion by executing the following command as a user who has admin rights:

```
curl -v -u USERNAME:PASSWORD -H 'Content-type: application/json'
-X POST 'http://localhost:8764/api/apollo/apps/my-fusion-app-
name/webapps' -d '{"id": "myasapp","name": "My AS
App","contextPath": "/"}'
```

2. Upload the WAR to the Fusion Webapps service:

```
curl -v -u USERNAME:PASSWORD -H 'Content-type: application/zip'
-X PUT 'http://localhost:8764/api/apollo/apps/my-fusion-app-
name/webapps/myasapp/war' --data-binary @ROOT.war
```

This puts the WAR file into the *Fusion blob store*, and tells the Webapps service that this is the file that should be started in the specified context.

Download an App Studio Project (4.1 and 4.2)

How-to

> Compatible versions: 4.2
>
> Works with: *App Studio*

If you prefer to develop your search interface using your own tools, you can download it as a `.zip` file.

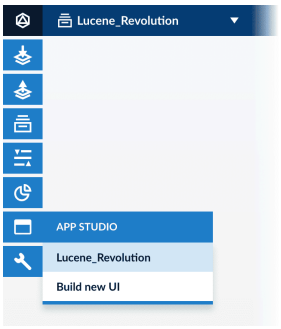
The project directory includes tools for running a demo deployment or compiling an application for a *production deployment*.

Downloading a project from Fusion

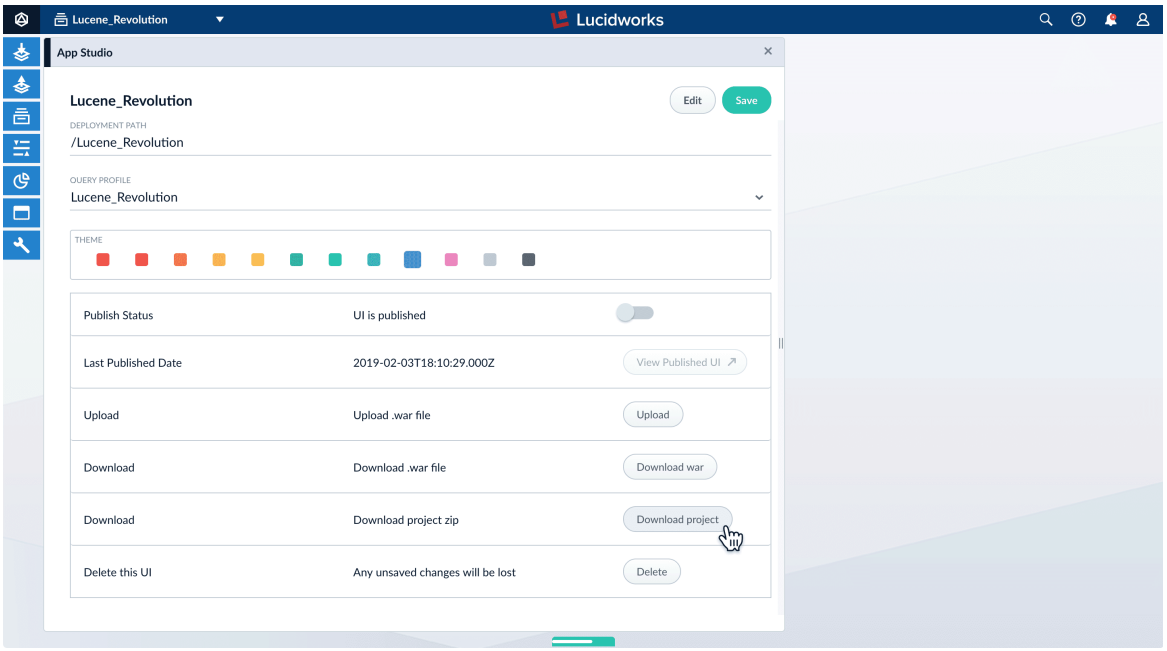
You can download your App Studio project as a `.zip` file from its configuration panel in App Studio.

> **Note**
>
> This option is unavailable after you have uploaded a WAR file for the project.

1. From the App Studio menu, select your project:



   The project configuration panel appears.
2. Click **Download project**:



   Your browser downloads the project as a `.zip` file.

Deploying a project

There are several ways to deploy a project. For information, see *Deployment Overview*.

Dynamically Adding Icons to Fields

How-to

> Compatible versions: 4.2, 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *App Studio*

Dynamically Adding An Icon To A Field

Sometimes you might want to dynamically prepend a field with an image or an icon depending on the value of a field or variable.

This can easily be accomplished using a `class` attribute referencing a little bit of CSS, defined in your styles/includes/custom.less.

For example, to dynamically add a 'type' image before a field:

```
[class*="field-icon-"] { position: relative; margin-left: 2em; }
[class*="field-icon-"]:before { position: absolute; left: -30px;
bottom: 14px; background-size: 20px 20px; display: inline-block;
width: 20px; height: 20px; content: " "; } .field-icon-pdf:before {
background-image: url('../assets/icon-pdf.png'); } .field-icon-
doc:before { background-image: url('../assets/icon-doc.png'); }
```

And in your view, use:

```
<search:field name="type" ng-class="field-icon-{{result |
fields:'type' | actual}}"></search:field>
```

The CSS can easily be extended or adapted to add more icon types or change the position and size of the icon.

Edit parameters in the Query Workbench

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

The Query Workbench allows you to edit the search parameters.

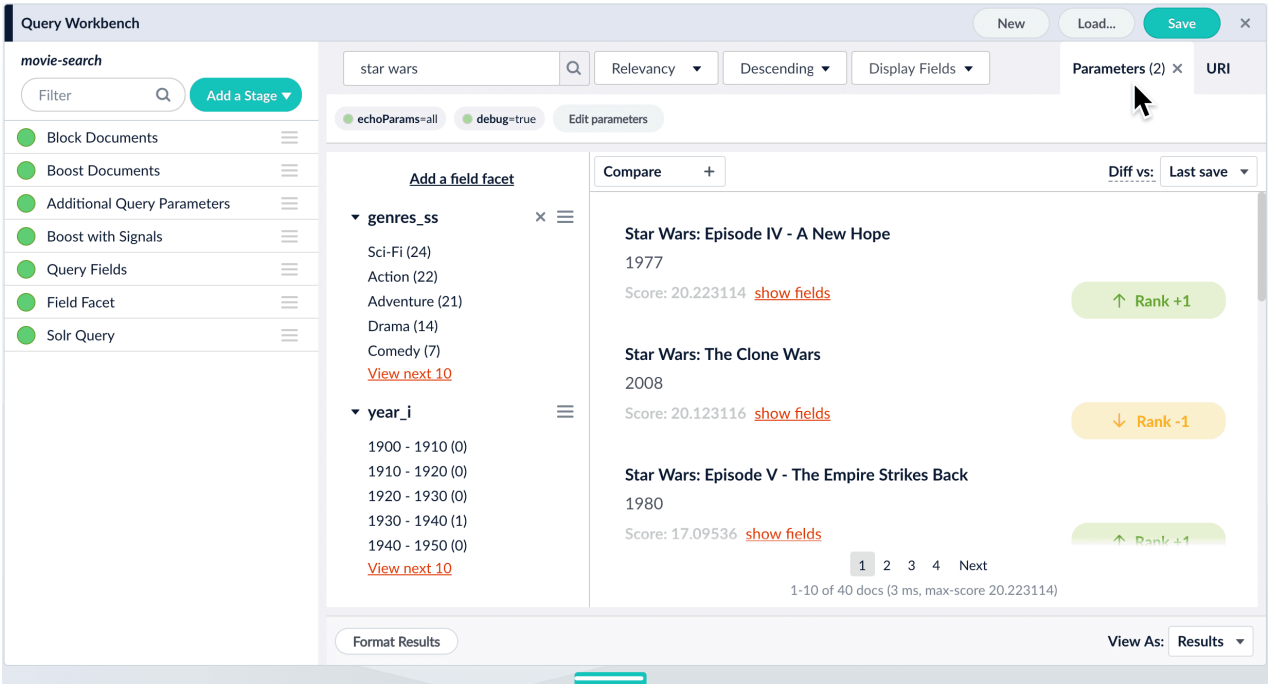
1. Click **Parameters** to display the current search parameters.
2. Click **Edit parameters**.
3. In the Parameters and Values screen, edit the parameters and click **Close**.

Edit Parameters in the Query Workbench

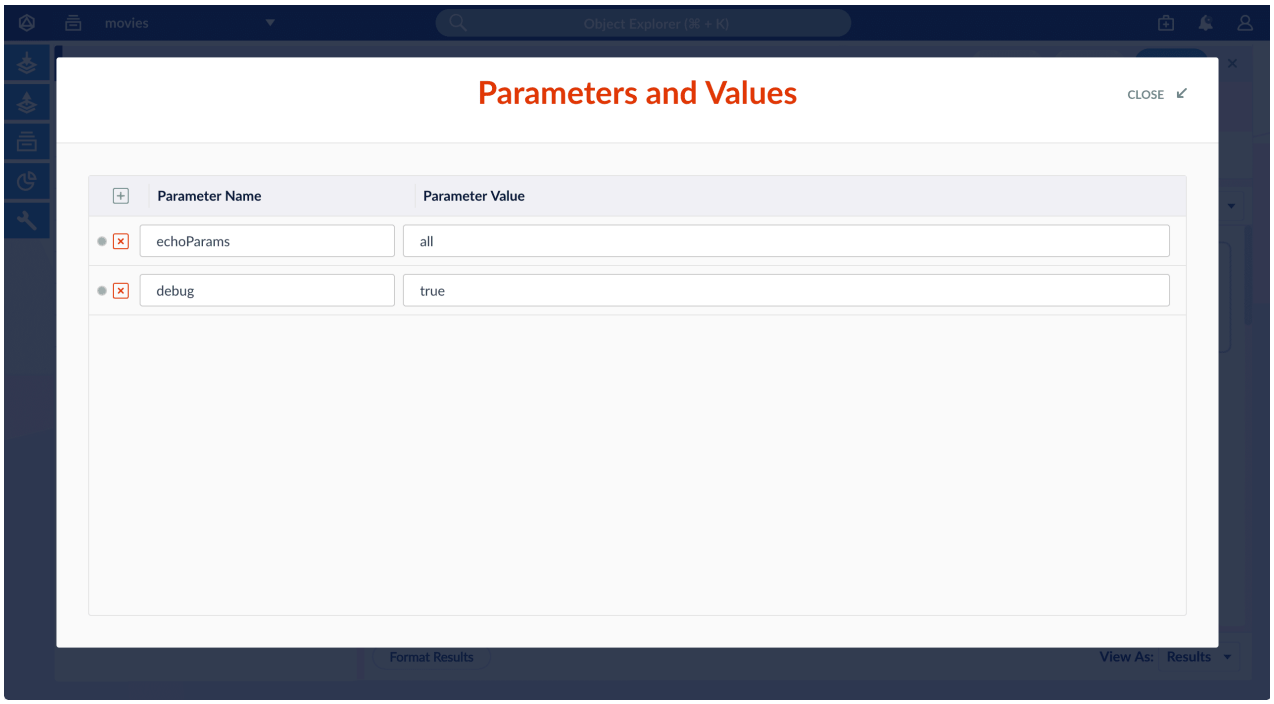
How-to

> Compatible versions: 4.2, 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion Server*, *Fusion*

The Query Workbench allows you to edit the search parameters. Click **Parameters** to display the current search parameters and the **Edit parameters** button:



Then, click **Edit parameters** to open the "Parameters and Values" window where you can add, modify, or remove search parameters:

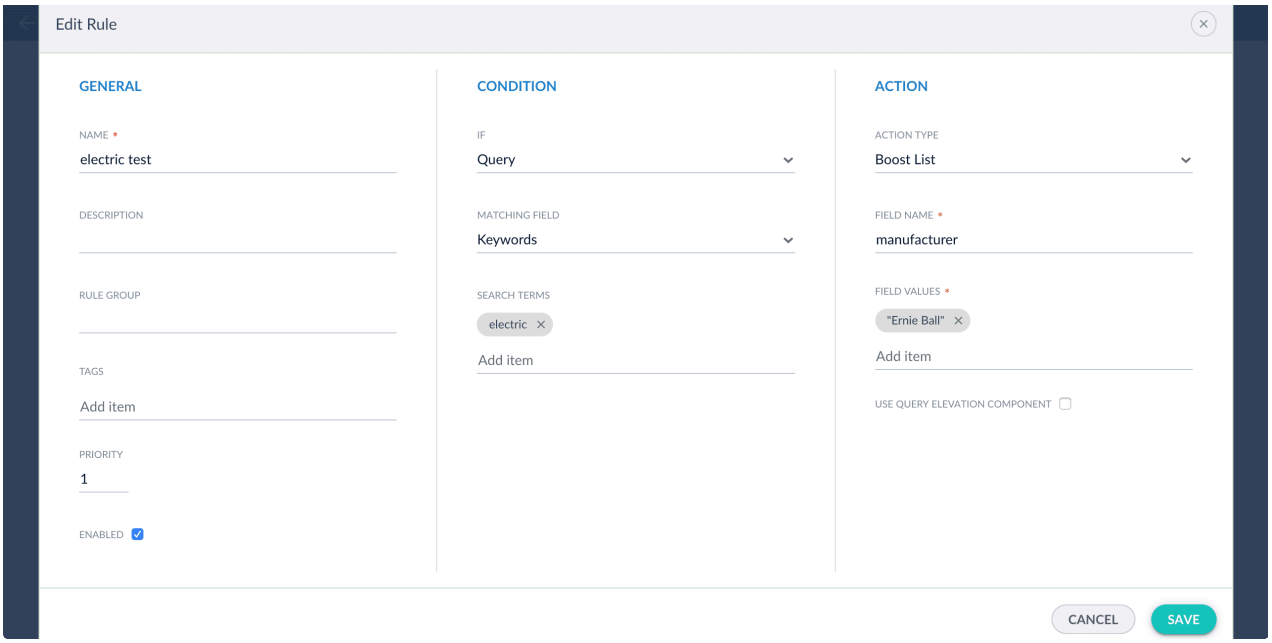


Edit Rules in the Rules Simulator

How-to

> Compatible versions: 4.2
>
> Works with: *Fusion AI*

You can edit any rule in the list of triggered rules, including disabling it, by clicking the Edit icon next to the rule.



De-select "Enabled" in the Edit Rule window to turn off this rule. See *Rule conditions* and *Rule actions* for details about the conditions and actions that you can modify when editing a rule.

Tip

Only rules triggered by the current query are available to edit in the Simulator. To edit other rules, see *Business Rules*.

Edit, Enable, or Disable a Rule

How-to

> Compatible versions: 4.2
>
> Works with: *Fusion AI*

1. In the Fusion UI, navigate to **Relevance** > **Query Rewriting**.
2. Under **Business Rules**, click **View**.
3. Hover over the rule you want to edit.
4. Click the icon next to the rule.

Publishing your changes

How to publish updated business rules

1. In the **Business Rules** screen, click the **PUBLISH** button.
   Fusion prompts you to confirm that you want to publish your changes.
2. Click **PUBLISH**.

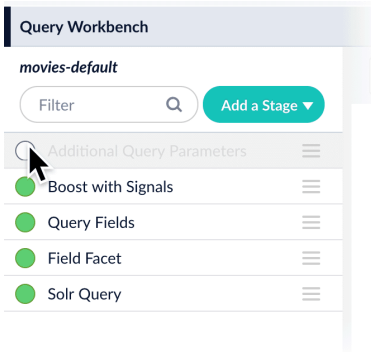
> **Tip**
>
> You can un-publish a query rewrite by changing its status to "denied", then clicking **PUBLISH**.

Enable and disable query pipeline stages

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

By default, every stage in a query pipeline is enabled. While working with a query pipeline, it can be helpful to disable stages without removing them completely. This allows you to preserve a stage's configuration while observing how the search results change in its absence. You can re-enable the stage at any time. When you save a query pipeline, the enabled/disabled state of each stage is also saved.



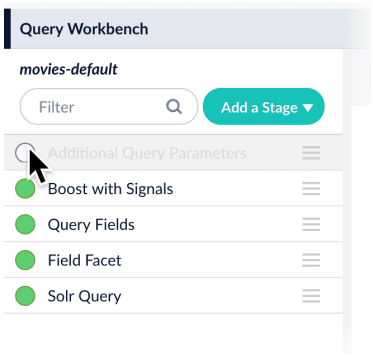
Click the circle next to any stage to enable or disable it.

Enable and Disabling Query Pipeline Stages

How-to

> Compatible versions: 4.2, 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion Server*, *Fusion*

By default, every stage in a query pipeline is enabled. While working with a query pipeline, it can be helpful to disable stages without removing them completely. This allows you to preserve a stage's configuration while observing how the search results change in its absence. You can re-enable the stage at any time. When you save a query pipeline, the enabled/disabled state of each stage is also saved.



Click the circle next to any stage in order to enable or disable it.

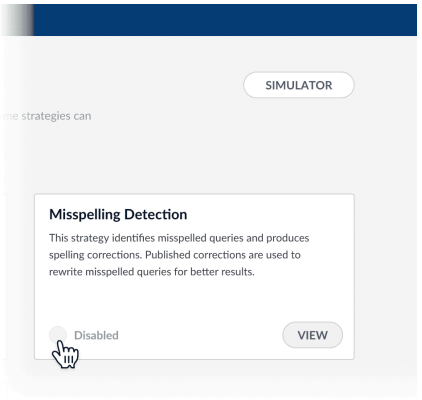
Enable and Disabling Query Rewriting Strategies

How-to

> Compatible versions: 4.2
>
> Works with: *Fusion AI*, *Fusion Server*, *Fusion*

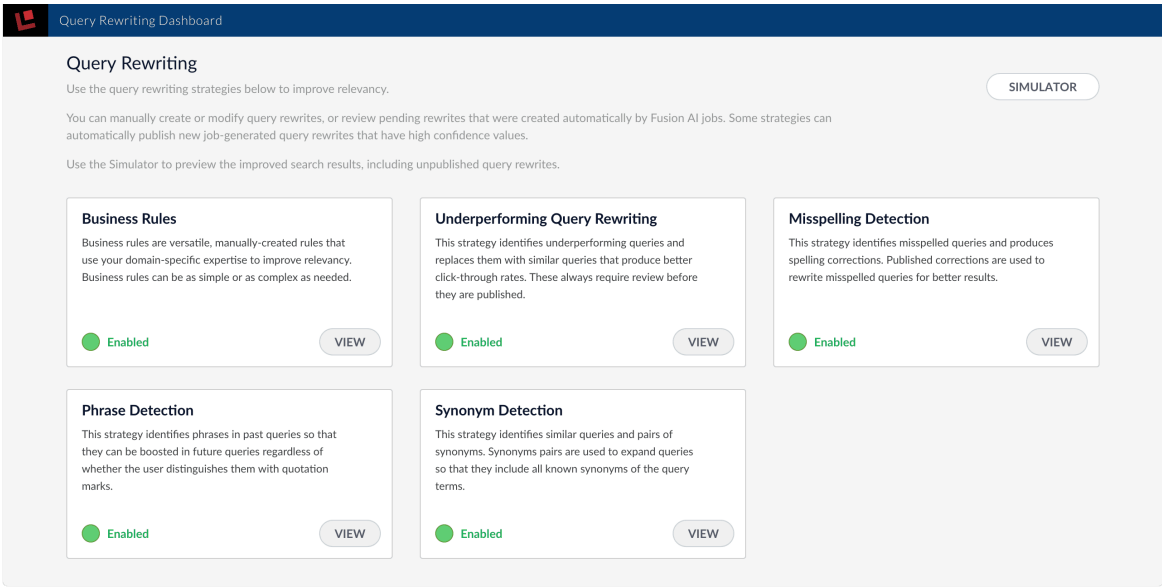
By default, all query rewriting strategies are enabled, and all of the data in the `_query_rewrite_staging` collection is applied in the Simulator. To see how relevancy is affected by individual strategies, you can selectively enable or disable strategies in the query rewriting dashboard.

How to enable or disable query rewriting strategies

1. 

From the Simulator, click **Query Rewriting Dashboard**.

   The query rewriting dashboard appears:



2. Click **Enabled** to disable an active strategy, or click **Disabled** to enable an inactive strategy.
3. Click **Simulator** to return to the Simulator and see how your changes affect search results.

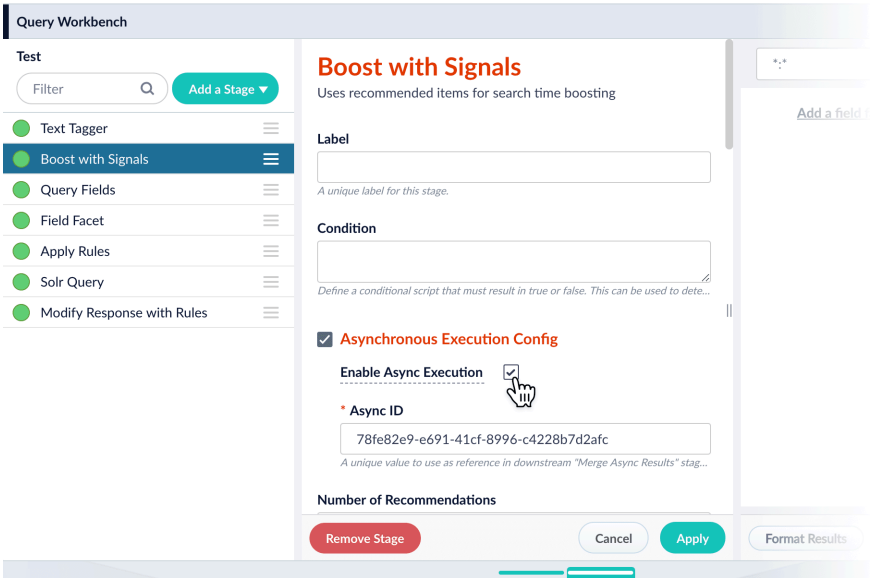
Enable Asynchronous Query Pipeline Processing

How-to

> Compatible versions: 4.2, 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion Server*, *Fusion*

1. In the *Query Workbench*, open the query pipeline you want to modify.
2. Click **Add a Stage** and select a stage that supports asynchronous processing:
    - Active Directory Security Trimming
    - Security Trimming
    - Boost with Signals
    - Apply Rules
    - JDBC Lookup
    - LWAI Vectorize Query
    - Solr Subquery
    - LWAI Prediction Query

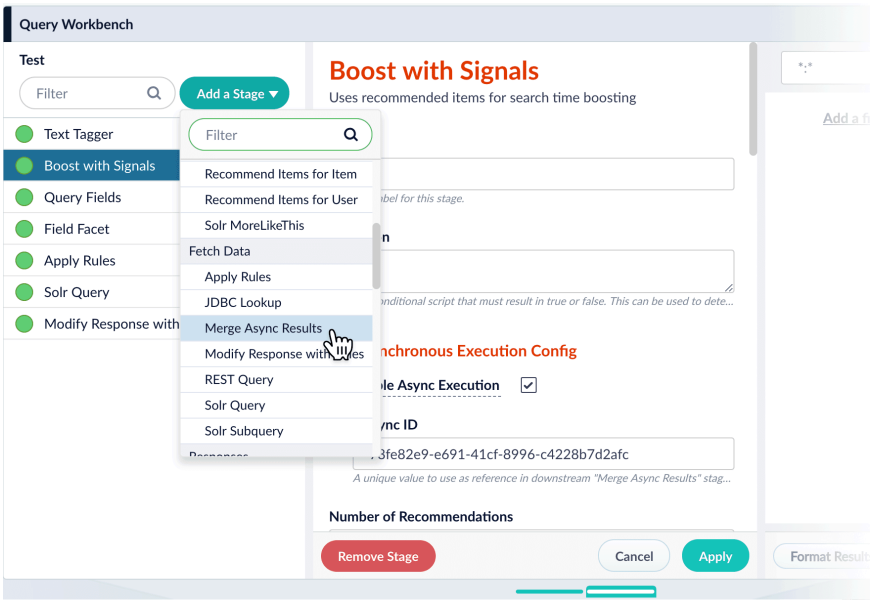
3. In the stage configuration panel, select **Asynchronous Execution Config**.
4. Select **Enable Async Execution**.



Tip

Fusion automatically assigns an **Async ID** value to this stage. Change this to a more memorable string that describes the asynchronous stages you are merging, such as `signals` or `access_control`.

5. Copy the **Async ID** value.
6. Click **Apply**.
   Verify that the stage is in the correct position in your pipeline.
7. Click **Add a Stage** and select the *Merge Async Results stage*.

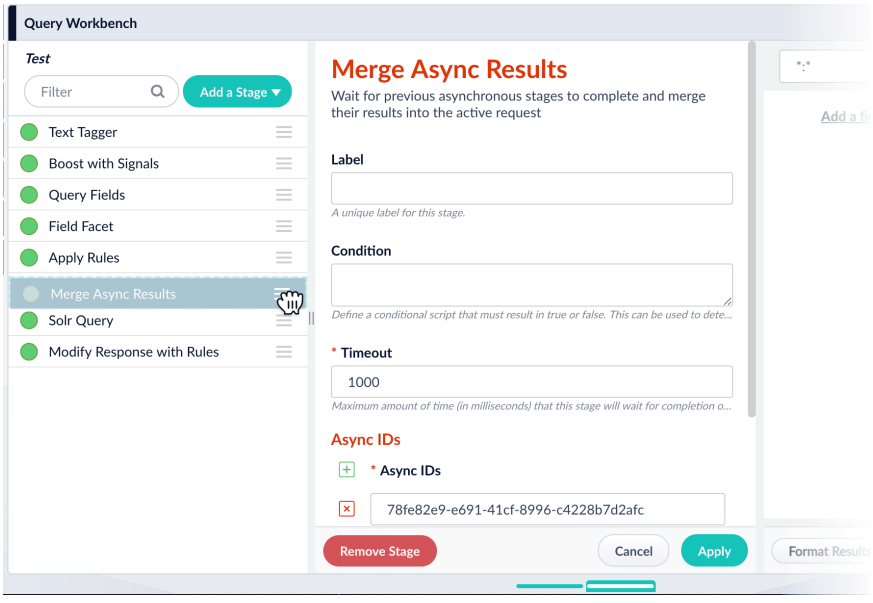


8. In the stage configuration panel, click the Add button next to **Async IDs**.

9. Paste the ID from the asynchronous query stage into the new ID field.

> **Tip**
>
> When using multiple asynchronous stages, the order in which IDs are entered here is the order in which they are joined back into the pipeline.

10. Click **Apply**.
    The Merge Async Results stage is now the first stage in the pipeline.
11. Drag the Merge Async Results stage down so that it appears immediately before the Solr Query stage:



12. Click **Save**.

For more information, see *Fusion asynchronous query pipeline processing*.

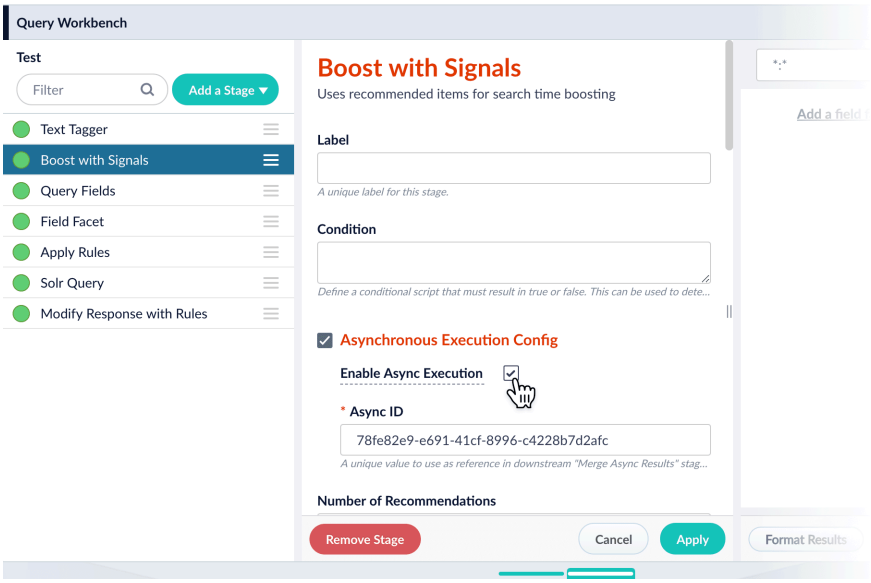
Enable Asynchronous Query Pipeline Processing

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

1. In the *Query Workbench*, open the query pipeline you want to modify.
2. Click **Add a Stage** and select a stage that supports asynchronous processing, or select the stage in your pipeline if it has already been added. The stages that support asynchronous processing are:
    - Active Directory Security Trimming
    - Security Trimming
    - Boost with Signals
    - Apply Rules
    - JDBC Lookup
    - LWAI Vectorize Query
    - Solr Subquery
    - LWAI Prediction Query

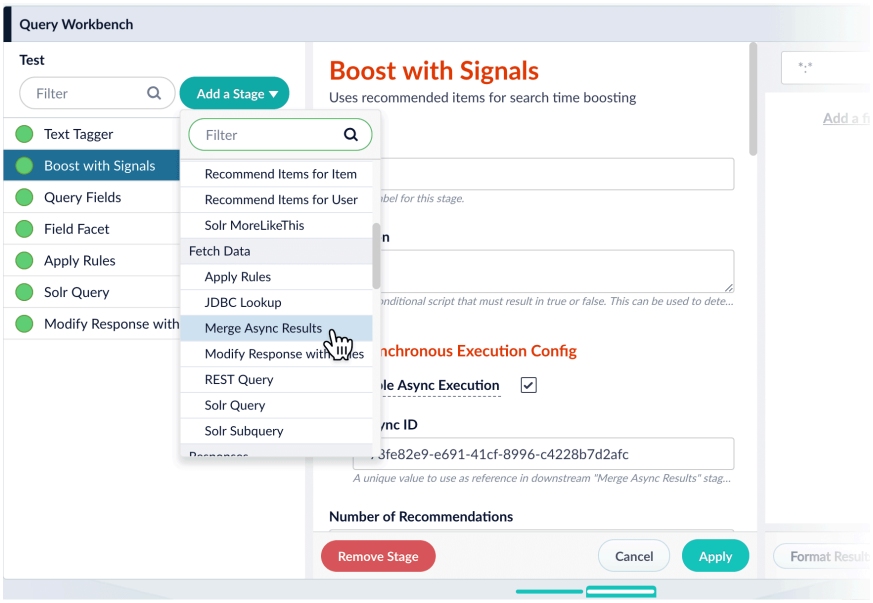
3. In the stage configuration panel, select **Asynchronous Execution Config**.
4. Select **Enable Async Execution**.



Tip

Managed Fusion automatically assigns an **Async ID** value to this stage. Change this to a more memorable string that describes the asynchronous stages you are merging, such as `signals` or `access_control`.

5. Copy the **Async ID** value.
6. Click **Apply**.
   Verify that the stage is in the correct position in your pipeline.
7. Click **Add a Stage** and select the *Merge Async Results stage*.

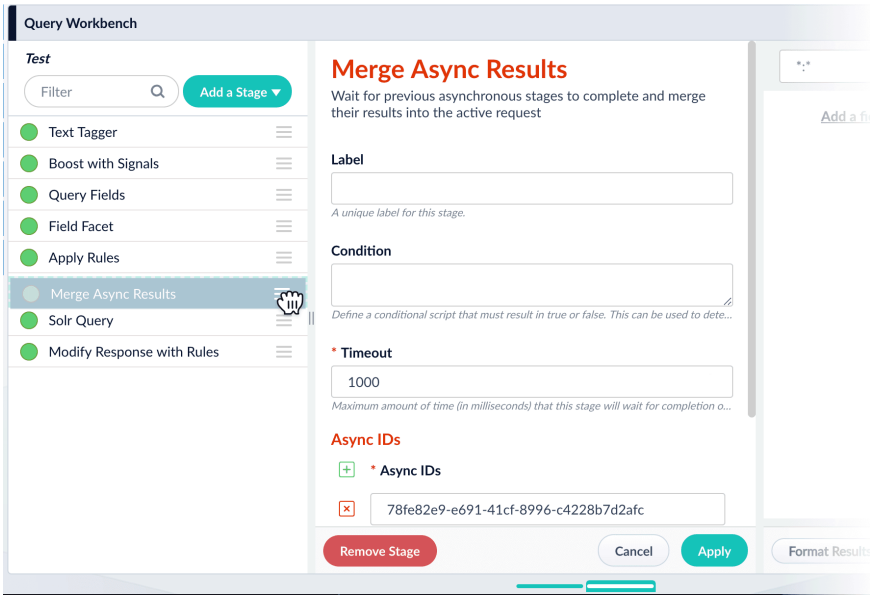


8. In the stage configuration panel, click the Add button next to **Async IDs**.

9. Paste the ID from the asynchronous query stage into the new ID field.

> **Tip**
>
> When using multiple asynchronous stages, the order in which IDs are entered here is the order in which they are joined back into the pipeline.

10. Click **Apply**.
    The Merge Async Results stage is now the first stage in the pipeline.
11. Drag the Merge Async Results stage down so that it appears immediately before the Solr Query stage:



12. Click **Save**.

For more information, see *Managed Fusion asynchronous query pipeline processing*.

Enable Jupyter

How-to

> Compatible versions: 5.5
>
> Works with: *Fusion*

Jupyter can be enabled with the Fusion Helm chart. It is not enabled by default.

How to enable Jupyter

1. In your custom values YAML file, locate `fusion-jupyter` and set `enabled` to "true":

```
fusion-jupyter: enabled: true
```

2. Run the setup script using the `--upgrade` option, specifying your custom values YAML file using the `--values` option.
   The example below uses the setup script for Google Kubernetes Engine (GKE):

```
./setup_f5_gke.sh -c mycluster -p myproject -n mynamespace --
region us-west1-c --upgrade --values
gke_mycluster_mynamespace_fusion_values.yaml
```

   Be sure to specify your version of Fusion.
3. Connect to the Jupyter UI at `https://FUSION_HOST:6764/jupyter`.

Enable or disable signals

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

You can enable and disable signals using the *Managed Fusion UI* or the *REST API*.

Tip

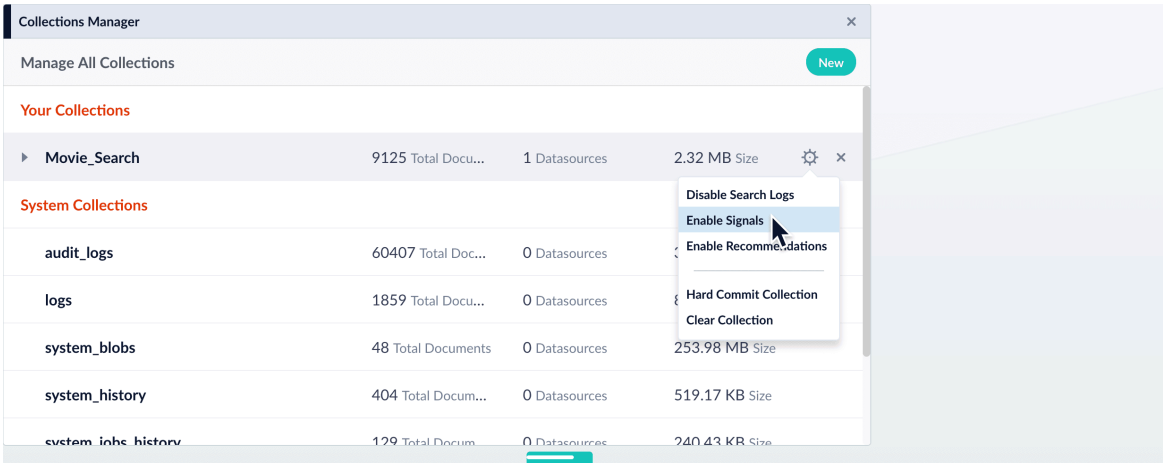
When you disable signals, the *aggregation jobs* are deleted, but the `_signals` and `_signals_aggr` collections are not, your legacy signal data remains intact.

Using the UI

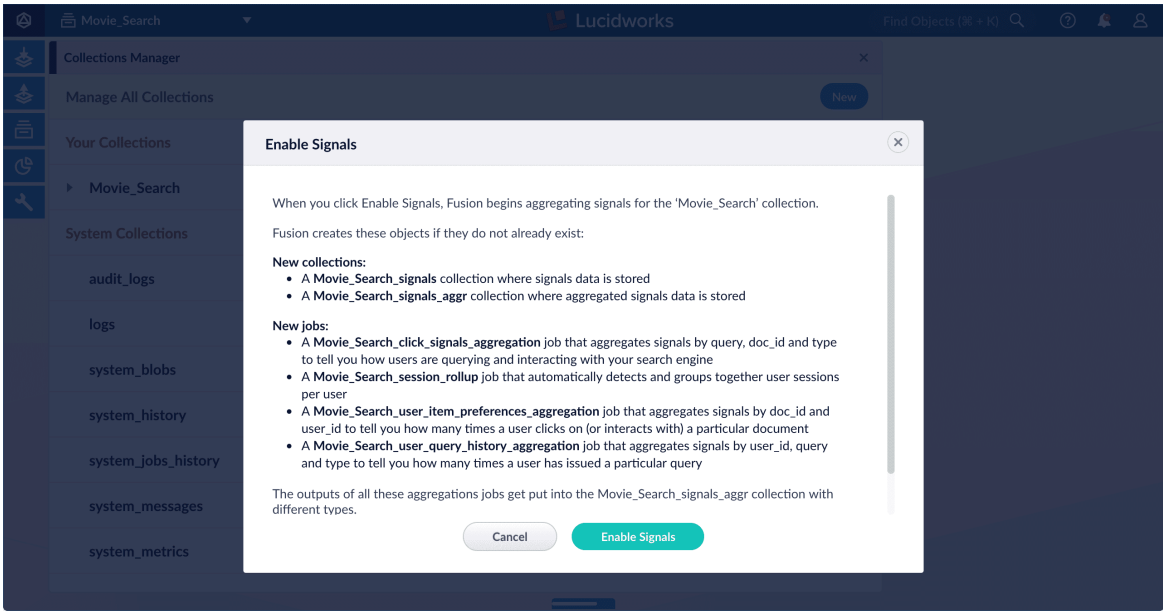
When you create a collection using the Fusion UI, signals are enabled and a signals collection created by default. You can also enable and disable signals for existing collections using the Collections Manager.

Enable signals for a collection

1. In the Fusion workspace, navigate to **Collections** > **Collections Manager**.
2. Hover over the primary collection for which you want to enable signals.
3. Click **Configure** to open the drop-down menu.



4. Click **Enable Signals**.
   The **Enable Signals** window appears, with a list of collections and jobs that are created when you enable signals.



5. Click **Enable Signals**.

Disable signals for a collection

1. In the Fusion workspace, navigate to **Collections** > **Collections Manager**.
2. Hover over the primary collection for which you want to disable signals.
3. Click **Configure** to open the drop-down menu.
4. Click **Disable Signals**.
   The **Disable Signals** window appears, with a list of jobs that are created when you enable signals.
5. Click **Disable Signals**.
   Your `_signals` and `_signals_aggr` collections remain intact so that you can access your legacy signals data.

Using the Collection Features API

Using the API, the `/collections/{collection}/features/{feature}` endpoint enables or disables signals for any collection:

Check whether signals are enabled for a collection

```
curl -u USERNAME:PASSWORD https://EXAMPLE_COMPANY.b.lucidworks.cloud:{api-port}/api/collections/COLLECTION_NAME/features/signals
```

Enable signals for a collection

```
curl -u USERNAME:PASSWORD -X PUT -H "Content-type: application/json" -d '{"enabled" : true}' https://https://EXAMPLE_COMPANY.b.lucidworks.cloud:{api-port}/api/collections/COLLECTION_NAME/features/signals
```

Disable signals for a collection

```
curl -u USERNAME:PASSWORD -X PUT -H "Content-type: application/json" -d '{"enabled" : false}' https://https://EXAMPLE_COMPANY.b.lucidworks.cloud:{api-port}/api/collections/COLLECTION_NAME/features/signals
```

Enable or Disable Signals

How-to

> Compatible versions: 4.2, 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion AI*, *Fusion Server*, *Fusion*

You can enable and disable signals using the *Fusion UI* or the *REST API*.

Tip

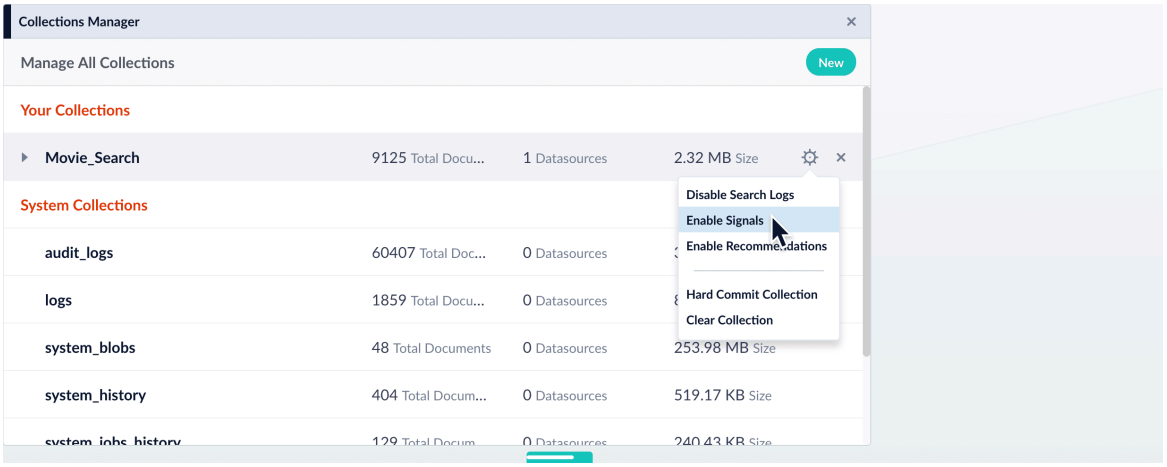
When you disable signals, the *aggregation jobs* are deleted, but the `_signals` and `_signals_aggr` collections are not, your legacy signal data remains intact.

Using the UI

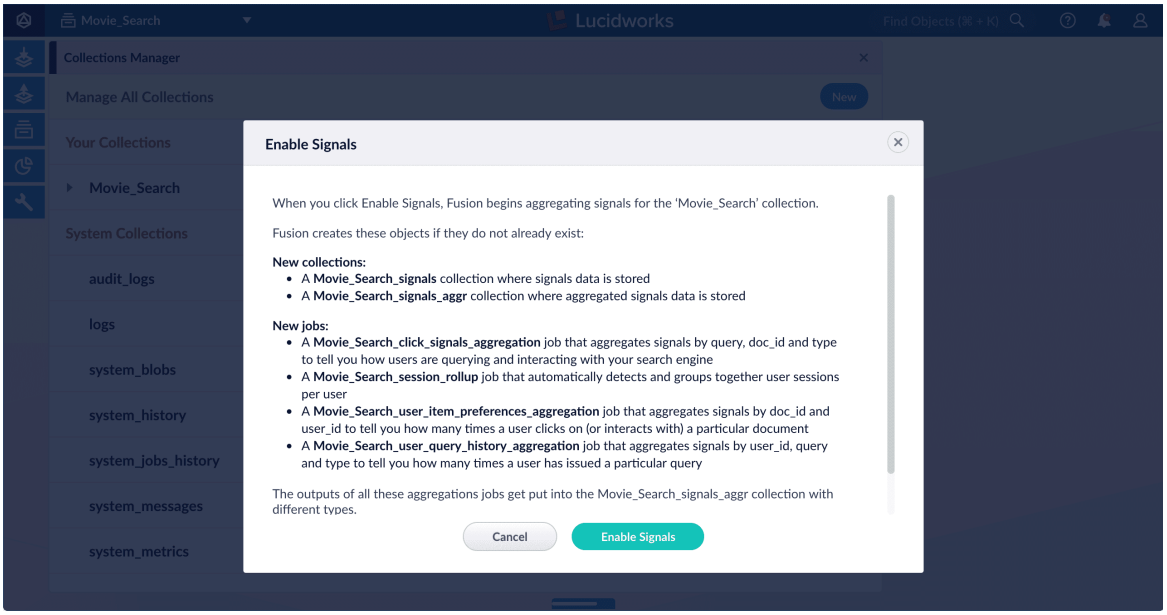
When you create a collection using the Fusion UI, signals are enabled and a signals collection created by default. You can also enable and disable signals for existing collections using the Collections Manager.

Enable signals for a collection

1. In the Fusion workspace, navigate to **Collections** > **Collections Manager**.
2. Hover over the primary collection for which you want to enable signals.
3. Click **Configure** to open the drop-down menu.



4. Click **Enable Signals**.
   The **Enable Signals** window appears, with a list of collections and jobs that are created when you enable signals.



5. Click **Enable Signals**.

Disable signals for a collection

1. In the Fusion workspace, navigate to **Collections** > **Collections Manager**.
2. Hover over the primary collection for which you want to disable signals.
3. Click **Configure** to open the drop-down menu.
4. Click **Disable Signals**.
   The **Disable Signals** window appears, with a list of jobs that are created when you enable signals.
5. Click **Disable Signals**.
   Your `_signals` and `_signals_aggr` collections remain intact so that you can access your legacy signals data.

Using the Collection Features API

Using the API, the `/collections/{collection}/features/{feature}` endpoint enables or disables signals for any collection:

Check whether signals are enabled for a collection

```
curl -u USERNAME:PASSWORD http://localhost:{api-
port}/api/collections/COLLECTION_NAME/features/signals
```

Enable signals for a collection

```
curl -u USERNAME:PASSWORD -X PUT -H "Content-type: application/json" -
d '{"enabled" : true}' http://localhost:{api-
port}/api/collections/COLLECTION_NAME/features/signals
```

Disable signals for a collection

```
curl -u USERNAME:PASSWORD -X PUT -H "Content-type: application/json" -
d '{"enabled" : false}' http://localhost:{api-
port}/api/collections/COLLECTION_NAME/features/signals
```

Enable Read-Only Root File System

How-to

> Compatible versions: 5.9
>
> Works with: *Fusion*

This topic describes how to configure a read-only root file system for a Fusion deployment. Read-only mode safeguards your file system against unauthorized changes, such as by malicious software or other attacks. Enabling this feature entails configuring the `readOnlyRootFilesystem` attribute in the Fusion Helm chart.

Note

This feature is only available in Fusion 5.9.9 and later versions of Fusion 5.9.

Beginning with Fusion 5.9.10, all of *Fusion's services* are designed to work with a read-only root file system, but some external services may require write access to the file system. In that case, you can configure a read-only root file system and mount a separate writable file system for these services.

The examples below show how to configure a read-only root file system for various Fusion services.

Solr

```
fusion: solr: containerSecurityContext: readOnlyRootFilesystem: true
```

With optional services:

```
fusion: solr: exporter: enabled: true securityContext:
readOnlyRootFilesystem: true containerSecurityContext:
readOnlyRootFilesystem: true
```

Zookeeper

```
fusion: zookeeper: containerSecurityContext: readOnlyRootFilesystem:
true
```

With optional services:

```
fusion: zookeeper: containerSecurityContext: readOnlyRootFilesystem:
true exporters: jmx: enabled: true zookeeper: enabled: true jobs:
chroots: enabled: true config: create: – /root-read-only
```

Kafka

```
fusion: kafka: containerSecurityContext: readOnlyRootFilesystem: true
```

With optional services:

```
fusion: kafka: containerSecurityContext: readOnlyRootFilesystem: true
volumePermissions: enabled: true containerSecurityContext:
readOnlyRootFilesystem: true metrics: kafka: containerSecurityContext:
readOnlyRootFilesystem: true enabled: true jmx: enabled: true
containerSecurityContext: enabled: true readOnlyRootFilesystem: true
externalAccess: enabled: true autoDiscovery: enabled: true
containerSecurityContext: enabled: true readOnlyRootFilesystem: true
provisioning: enabled: true topics: - name: test partitions: 1
replicationFactor: 1 ##
https://kafka.apache.org/documentation/#topicconfigs config:
max.message.bytes: 64000 flush.messages: 1 containerSecurityContext:
readOnlyRootFilesystem: true enabled: true
```

ML Model Service

```
fusion: ml-model-service: ambassador: containerSecurityContext:
readOnlyRootFilesystem: true milvus: mysql: containerSecurityContext:
readOnlyRootFilesystem: true securityContext: readOnlyRootFilesystem:
true With optional services: fusion: ml-model-service: ambassador:
containerSecurityContext: readOnlyRootFilesystem: true
prometheusExporter: enabled: true securityContext:
readOnlyRootFilesystem: true milvus: mysql: metrics: enabled: true
securityContext: readOnlyRootFilesystem: true
containerSecurityContext: readOnlyRootFilesystem: true cluster:
enabled: false mishards: securityContext: readOnlyRootFilesystem: true
admin: enabled: true securityContext: readOnlyRootFilesystem: true
securityContext: readOnlyRootFilesystem: true
```

Argo

```
fusion: argo: minio: containerSecurityContext: readOnlyRootFilesystem:
true extraVolumes: - name: config-dir emptyDir: {} - name: certs-dir
emptyDir: {} extraVolumeMounts: - name: config-dir mountPath: /.minio
- name: certs-dir mountPath: /etc/minio/certs server: securityContext:
readOnlyRootFilesystem: true controller: securityContext:
readOnlyRootFilesystem: true mainContainer: securityContext:
readOnlyRootFilesystem: true executor: securityContext:
readOnlyRootFilesystem: true
```

Seldon Core Operator

```
fusion: seldon-core-operator: containersSecurityContext:
readOnlyRootFilesystem: true
```

Argo Common Workflows

```
fusion: argo-common-workflows: containerSecurityContext:
readOnlyRootFilesystem: true
```

Question Answering

```
fusion: question-answering: containersSecurityContext:
readOnlyRootFilesystem: true
```

Classification

```
fusion: classification: containersSecurityContext:
readOnlyRootFilesystem: true
```

All-in-one values.yaml example

```
fusion: solr: containerSecurityContext: readOnlyRootFilesystem: true
solr-managed: containerSecurityContext: readOnlyRootFilesystem: true
zookeeper: containerSecurityContext: readOnlyRootFilesystem: true
kafka: containerSecurityContext: readOnlyRootFilesystem: true ml-
model-service: ambassador: containerSecurityContext:
readOnlyRootFilesystem: true milvus: mysql: containerSecurityContext:
readOnlyRootFilesystem: true securityContext: readOnlyRootFilesystem:
true argo: minio: containerSecurityContext: readOnlyRootFilesystem:
true extraVolumes: - name: config-dir emptyDir: {} - name: certs-dir
emptyDir: {} extraVolumeMounts: - name: config-dir mountPath: /.minio
- name: certs-dir mountPath: /etc/minio/certs server: securityContext:
readOnlyRootFilesystem: true controller: securityContext:
readOnlyRootFilesystem: true mainContainer: securityContext:
readOnlyRootFilesystem: true executor: securityContext:
readOnlyRootFilesystem: true seldon-core-operator:
containersSecurityContext: readOnlyRootFilesystem: true argo-common-
workflows: containerSecurityContext: readOnlyRootFilesystem: true
question-answering: containersSecurityContext: readOnlyRootFilesystem:
true classification: containersSecurityContext:
readOnlyRootFilesystem: true
```

All-in-one values.yaml with optional services example

```
fusion: argo: minio: persistence: enabled: false
containerSecurityContext: readOnlyRootFilesystem: true extraVolumes: -
name: config-dir emptyDir: {} - name: certs-dir emptyDir: {}
extraVolumeMounts: - name: config-dir mountPath: /.minio - name:
certs-dir mountPath: /etc/minio/certs server: securityContext:
readOnlyRootFilesystem: true controller: securityContext:
readOnlyRootFilesystem: true mainContainer: securityContext:
readOnlyRootFilesystem: true executor: securityContext:
readOnlyRootFilesystem: true solr: exporter: enabled: true
securityContext: readOnlyRootFilesystem: true
containerSecurityContext: readOnlyRootFilesystem: true # tls: #
enabled: true solr-managed: exporter: enabled: true securityContext:
readOnlyRootFilesystem: true enableExternalFiles: true enabledStorage:
- gcs processRaw: image: repository: fusion-dev-docker.ci-
artifactory.lucidworks.com cloudRoot: gs://lw-managed-fusion-data/tmp
gcs: secret: gcs-key secretFieldName: key.json securityContext:
readOnlyRootFilesystem: true containerSecurityContext:
readOnlyRootFilesystem: true # tls: # enabled: true zookeeper:
containerSecurityContext: readOnlyRootFilesystem: true exporters: jmx:
enabled: true zookeeper: enabled: true jobs: chroots: # enabled: true
config: create: - /root-read-only kafka: containerSecurityContext:
readOnlyRootFilesystem: true volumePermissions: enabled: true image:
repository: "fusion-dev-docker.ci-artifactory.lucidworks.com/os-shell"
containerSecurityContext: readOnlyRootFilesystem: true metrics: kafka:
containerSecurityContext: readOnlyRootFilesystem: true enabled: true
jmx: enabled: true containerSecurityContext: enabled: true
readOnlyRootFilesystem: true externalAccess: enabled: true
autoDiscovery: enabled: true containerSecurityContext: enabled: true
readOnlyRootFilesystem: true provisioning: enabled: true topics: -
name: test partitions: 1 replicationFactor: 1 ##
https://kafka.apache.org/documentation/#topicconfigs config:
max.message.bytes: 64000 flush.messages: 1 containerSecurityContext:
```

```
readOnlyRootFilesystem: true enabled: true ml-model-service: enabled:
true ambassador: containerSecurityContext: readOnlyRootFilesystem:
true prometheusExporter: enabled: true securityContext:
readOnlyRootFilesystem: true milvus: mysql: metrics: enabled: true
securityContext: readOnlyRootFilesystem: true
containerSecurityContext: readOnlyRootFilesystem: true cluster:
enabled: false mishards: securityContext: readOnlyRootFilesystem: true
admin: enabled: true securityContext: readOnlyRootFilesystem: true
securityContext: readOnlyRootFilesystem: true seldon-core-operator:
containersSecurityContext: readOnlyRootFilesystem: true argo-common-
workflows: containerSecurityContext: readOnlyRootFilesystem: true
question-answering: containersSecurityContext: readOnlyRootFilesystem:
true classification: containersSecurityContext:
readOnlyRootFilesystem: true
```

Supported services

The table below lists the services that support a read-only root file system, the ones that have it enabled by default, and the Fusion release in which support was added:

Chart Name	Pod Name	Container Name	Supported	Default enabled	Supported version
admin-ui	admin-ui	admin-ui	✅	✅	5.9.9+
api-gateway	api-gateway	init/api-gateway	✅	✅	5.9.9+
api-gateway	api-gateway	api-gateway	✅	✅	5.9.9+
api-gateway	api-gateway	generate-jks	✅	✅	5.9.9+
apps-manager	apps-manager	apps-manager	✅	✅	5.9.9+
argo	argo-server	argo-server	✅	❌	5.9.10+
argo	argo-executor	executor	✅	❌	5.9.10+
argo	argo-mainContainer	mainContainer	✅	❌	5.9.10+
argo	argo-controller	controller	✅	✅	5.9.10+
argo/minio	minio	minio	✅	❌	5.9.10+
argo/minio	minio	minio	✅	❌	5.9.10+
argo/minio	make-bucket-job	minio-mc	✅	❌	5.9.10+
argo-common-workflows	delete-model	init/main/wait	✅	❌	5.9.10+
argo-common-workflows	deploy-model	init/main/wait	✅	❌	5.9.10+
argo-common-workflows	milvus-maintenance	init/main/wait	✅	❌	5.9.10+
argo-common-workflows	upload-model-to-cloud	init/main/wait	✅	❌	5.9.10+
async-parsing	async-parsing	tika-server	✅	✅	5.9.9+
async-parsing	async-parsing	async-parsing	✅	✅	5.9.9+
auth-ui	auth-ui	auth-ui	✅	✅	5.9.9+
classic-rest-service	classic-rest-service	init/import-certs	✅	✅	5.9.10+
classic-rest-	classic-rest-service	classic-rest-service	✅	✅	5.9.9+

Chart Name	Pod Name	Container Name	Supported	Default enabled	Supported version
service					
classification	argo/classification	init/wait/main	✅	❌	5.9.10+
connector-plugin	connector-plugin	init/import-certs	✅	❌	5.9.10+
connector-plugin	connector-plugin	connector-plugin	✅	✅	5.9.9+
connectors	connectors	connectors	✅	✅	5.9.9+
connectors-backend	CRD		✅	✅	5.9.9+
connectors-backend	connectors-backend	connectors-backend	✅	✅	5.9.9+
fusion-admin	fusion-admin	admin	✅	✅	5.9.9+
fusion-commons		check-admin	✅	✅	5.9.9+
fusion-commons		check-api-gateway	✅	✅	5.9.9+
fusion-commons		check-indexing	✅	✅	5.9.9+
fusion-commons		check-kafka	✅	✅	5.9.9+
fusion-commons		check-logstash	✅	✅	5.9.9+
fusion-commons		check-pulsar	✅	✅	5.9.9+
fusion-commons		setup-keystore-and-properties	✅	✅	5.9.9+
fusion-commons		check-zk	✅	✅	5.9.9+
fusion-config-sync	fusion-config-sync	fusion-config-sync	✅	✅	5.9.9+
fusion-data-augmentation	argo/data-augmentation/volume-fix	init/main/wait	✅	❌	5.9.10+
fusion-data-augmentation	argo/data-augmentation/init-workspace	init/main/wait	✅	❌	5.9.10+

Chart Name	Pod Name	Container Name	Supported	Default enabled	Supported version
fusion-data-augmentation	argo/data-augmentation/write-job-configs	init/main/wait	✅	❌	5.9.10+
fusion-data-augmentation	argo/data-augmentation/write-io-configs	init/main/wait	✅	❌	5.9.10+
fusion-data-augmentation	argo/data-augmentation/add-zkhost	init/main/wait	✅	❌	5.9.10+
fusion-data-augmentation	argo/data-augmentation/pull-data-training-and-metadata	init/main/wait	✅	❌	5.9.10+
fusion-data-augmentation	argo/data-augmentation/pull-data-training-and-metadata-cloud	init/main/wait	✅	❌	5.9.10+
fusion-data-augmentation	argo/data-augmentation/volume-fix2	init/main/wait	✅	❌	5.9.10+
fusion-data-augmentation	argo/data-augmentation/synonym-list	init/main/wait	✅	❌	5.9.10+
fusion-data-augmentation	argo/data-augmentation/download-synonym-dictionary	init/main/wait	✅	❌	5.9.10+
fusion-data-augmentation	argo/data-augmentation/keystroke-list	init/main/wait	✅	❌	5.9.10+
fusion-data-augmentation	argo/data-augmentation/download-keystroke-blob	init/main/wait	✅	❌	5.9.10+
fusion-data-augmentation	argo/data-augmentation/augment	init/main/wait	✅	❌	5.9.10+
fusion-data-augmentation	argo/data-augmentation/push-augmented-data	init/main/wait	✅	❌	5.9.10+
fusion-data-augmentation	argo/data-augmentation/push-augmented-data-cloud	init/main/wait	✅	❌	5.9.10+
fusion-indexing	fusion-indexing	fusion-indexing	✅	✅	5.9.9+
fusion-resources	fusion-resources-secret-hook	setup-keystore-and-properties	✅	✅	5.9.10+

Chart Name	Pod Name	Container Name	Supported	Default enabled	Supported version
insights	insights	insights	✅	✅	5.9.9+
job-launcher	job-launcher	job-launcher	✅	✅	5.9.9+
job-launcher	job-launcher	kubectl-runner	✅	✅	5.9.9+
job-launcher	job-launcher-spark-cleanup	kubectl-runner	✅	✅	5.9.9+
job-launcher	spark-kubernetes-driver	spark-kubernetes-driver	✅	❌	5.9.10+
job-launcher	spark-kubernetes-executor	spark-kubernetes-executor	✅	❌	5.9.10+
job-rest-server	job-rest-server	job-rest-server	✅	✅	5.9.9+
kafka	kafka-metrics	kafka-exporter-archived	✅	❌	5.9.10+
kafka	kafka-provisioning	init/wait-for-available-kafka	✅	❌	5.9.10+
kafka	kafka-provisioning	kafka-provisioning	✅	❌	5.9.10+
kafka	kafka	kafka	✅	❌	5.9.10+
kafka	kafka	jmx-exporter	✅	❌	5.9.10+
kafka	kafka	init/check-zk	✅	✅	5.9.9+
kafka	kafka	init/auto-discovery	✅	❌	5.9.10+
kafka	kafka	init/volume-permissions	✅	❌	5.9.10+
lwai-gateway	lwai-gateway	lwai-gateway	✅	✅	5.9.9+
ml-model-service	ml-model-service	java-service	✅	✅	5.9.9+
ml-model-service	ml-model-service-namespace-hook	kubectl-runner	✅	✅	5.9.9+

Chart Name	Pod Name	Container Name	Supported	Default enabled	Supported version
ml-model-service/ambassador	ambassador	ambassador	✅	❌	5.9.10+
ml-model-service/ambassador	ambassador	prometheus-exporter	✅	❌	5.9.10+
ml-model-service/milvus	milvus-writable	milvus	✅	❌	5.9.10+
ml-model-service/milvus	milvus-writable	init/wait-for-mysql	✅	✅	5.9.10+
ml-model-service/milvus	milvus-writable	init/create-for-share-storage	✅	❌	5.9.10+
ml-model-service/milvus	milvus-admin	admin	✅	❌	5.9.10+
ml-model-service/milvus	milvus-admin	init/wait-for-milvus	✅	✅	5.9.10+
ml-model-service/milvus	milvus-mishards	init/wait-for-mysql	✅	✅	5.9.10+
ml-model-service/milvus	milvus-mishards	init/wait-for-mysql	✅	✅	5.9.10+
ml-model-service/milvus	milvus-mishards	mishards	✅	❌	5.9.10+
ml-model-service/milvus/mysql	mysql	init/remove-lost-found	✅	✅	5.9.10+
ml-model-service/milvus/mysql	mysql	mysql	✅	❌	5.9.10+
pm-ui	pm-ui	pm-ui	✅	✅	5.9.9+
query-pipeline	query-pipeline	query-pipeline	✅	✅	5.9.9+
question-answering	argo/qna-coldstart/init-workspace	init/wait/main	✅	❌	5.9.10+

Chart Name	Pod Name	Container Name	Supported	Default enabled	Supported version
question-answering	argo/qna-coldstart/write-job-configs	init/wait/main	✅	❌	5.9.10+
question-answering	argo/qna-coldstart/write-io-configs	init/wait/main	✅	❌	5.9.10+
question-answering	argo/qna-coldstart/write-io-configs	init/wait/main	✅	❌	5.9.10+
question-answering	argo/qna-coldstart/add-zkHost	init/wait/main	✅	❌	5.9.10+
question-answering	argo/qna-coldstart/pull-data	init/wait/main	✅	❌	5.9.10+
question-answering	argo/qna-coldstart/pull-data-cloud	init/wait/main	✅	❌	5.9.10+
question-answering	argo/qna-coldstart/train	init/wait/main	✅	❌	5.9.10+
question-answering	argo/qna-coldstart/list-workspace	init/wait/main	✅	❌	5.9.10+
question-answering	argo/qna-coldstart/post-model	init/wait/main	✅	❌	5.9.10+
question-answering	argo/qna-coldstart/apply-seldon-deployment	init/wait/main	✅	❌	5.9.10+
question-answering	argo/qna-supervised/init-workspace	init/wait/main	✅	❌	5.9.10+
question-answering	argo/qna-supervised/write-job-configs	init/wait/main	✅	❌	5.9.10+
question-answering	argo/qna-supervised/write-io-configs	init/wait/main	✅	❌	5.9.10+
question-answering	argo/qna-supervised/add-zkhost	init/wait/main	✅	❌	5.9.10+
question-answering	argo/qna-supervised/pull-qa-data-cloud	init/wait/main	✅	❌	5.9.10+
question-answering	argo/qna-supervised/pull-qa-data	init/wait/main	✅	❌	5.9.10+
question-answering	argo/qna-supervised/train	init/wait/main	✅	❌	5.9.10+

Chart Name	Pod Name	Container Name	Supported	Default enabled	Supported version
question-answering	argo/qna-supervised/train-with-texts	init/wait/main	✅	❌	5.9.10+
question-answering	argo/qna-supervised/list-workspace	init/wait/main	✅	❌	5.9.10+
question-answering	argo/qna-supervised/post-model	init/wait/main	✅	❌	5.9.10+
question-answering	argo/qna-supervised/apply-seldon-deployment	init/wait/main	✅	❌	5.9.10+
question-answering	argo/qna-evaluation/init-workspace	init/wait/main	✅	❌	5.9.10+
question-answering	argo/qna-evaluation/write-job-configs	init/wait/main	✅	❌	5.9.10+
question-answering	argo/qna-evaluation/write-io-configs	init/wait/main	✅	❌	5.9.10+
question-answering	argo/qna-evaluation/add-zkhost	init/wait/main	✅	❌	5.9.10+
question-answering	argo/qna-evaluation/list-workspace	init/wait/main	✅	❌	5.9.10+
question-answering	argo/qna-evaluation/pull-eval-data	init/wait/main	✅	❌	5.9.10+
question-answering	argo/qna-evaluation/pull-eval-data-cloud	init/wait/main	✅	❌	5.9.10+
question-answering	argo/qna-evaluation/evaluate	init/wait/main	✅	❌	5.9.10+
question-answering	argo/qna-evaluation/push-eval-results-cloud	init/wait/main	✅	❌	5.9.10+
question-answering	argo/qna-evaluation/push-eval-results	init/wait/main	✅	❌	5.9.10+
recommender	argo/item-recommender-user	init/wait/main	✅	❌	5.9.10+
recommender	argo/item-recommender-user/init-workspace	init/wait/main	✅	❌	5.9.10+

Chart Name	Pod Name	Container Name	Supported	Default enabled	Supported version
recommender	argo/item-recommender-user/write-job-configs	init/wait/main	✅	❌	5.9.10+
recommender	argo/item-recommender-user/write-io-configs	init/wait/main	✅	❌	5.9.10+
recommender	argo/item-recommender-user/add-zkhost	init/wait/main	✅	❌	5.9.10+
recommender	argo/item-recommender-user/pull-data-training-and-metadata	init/wait/main	✅	❌	5.9.10+
recommender	argo/item-recommender-user/pull-data-training-and-metadata-cloud	init/wait/main	✅	❌	5.9.10+
recommender	argo/item-recommender-user/train-with-metadata	init/wait/main	✅	❌	5.9.10+
recommender	argo/item-recommender-user/train-without-metadata	init/wait/main	✅	❌	5.9.10+
recommender	argo/item-recommender-user/push-recommendations	init/wait/main	✅	❌	5.9.10+
recommender	argo/item-recommender-user/push-recommendations-cloud	init/wait/main	✅	❌	5.9.10+
recommender	argo/item-recommender-user/add-default-exclude-query	init/wait/main	✅	❌	5.9.10+
recommender	argo/item-recommender-user/init-workspace	init/wait/main	✅	❌	5.9.10+
recommender	argo/item-recommender-content/copy-model	init/wait/main	✅	❌	5.9.10+
recommender	argo/item-recommender-content/write-job-configs	init/wait/main	✅	❌	5.9.10+
recommender	argo/item-recommender-content/write-io-configs	init/wait/main	✅	❌	5.9.10+
recommender	argo/item-recommender-content/add-zkhost	init/wait/main	✅	❌	5.9.10+
recommender	argo/item-recommender-content/pull-data	init/wait/main	✅	❌	5.9.10+
recommender	argo/item-recommender-content/pull-data-cloud	init/wait/main	✅	❌	5.9.10+

Chart Name	Pod Name	Container Name	Supported	Default enabled	Supported version
recommender	argo/item-recommender-content/train	init/wait/main	✅	❌	5.9.10+
recommender	argo/item-recommender-content/push-content	init/wait/main	✅	❌	5.9.10+
recommender	argo/item-recommender-content/push-content-cloud	init/wait/main	✅	❌	5.9.10+
recommender	argo/item-recommender-content/add-default-exclude-query	init/wait/main	✅	❌	5.9.10+
recommender	argo/item-recommender-content/delete-old-content-recommendations	init/wait/main	✅	❌	5.9.10+
reverse-search	reverse-search	init/set-reverse-search-zone	✅	✅	5.9.10+
reverse-search	reverse-search	init/check-zk	✅	✅	5.9.9+
reverse-search	reverse-search	init/enable-tls-in-reverse-search	✅	✅	5.9.10+
reverse-search	reverse-search	reverse-search	✅	❌	5.9.10+
rules-ui	rules-ui	rules-ui	✅	✅	5.9.9+
seldon-core-operator	seldon-controller-manager	manager	✅	❌	5.9.10+
seldon-core-operator	seldon-spartakus-volunteer	seldon-spartakus-volunteer	✅	❌	5.9.9+
seldon-core-operator	crd/SeldonDeployment		✅ / ❌	❌	5.9.9+
solr	solr	init/set-solr-zone	✅	✅	5.9.10+
solr	solr	init/enable-tls-in-solr	✅	✅	5.9.10+
solr	configset-bootstrap	configset-bootstrap	✅	✅	5.9.10+
solr	solr	solr	✅	❌	5.9.10+
solr	solr-exporter	exporter	✅	❌	5.9.10+

Chart Name	Pod Name	Container Name	Supported	Default enabled	Supported version
`solr`	`solr-exporter`	`init/solr-init`	✅	✅	5.9.10+
`solr-managed`	`configset-bootstrap`	`configset-bootstrap`	✅	✅	5.9.10+
`solr-managed`	`exporter`	`exporter`	✅	❌	5.9.10+
`solr-managed`	`exporter`	`init/solr-init`	✅	✅	5.9.10+
`solr-managed`	`solr`	`init/set-solr-zone`	✅	✅	5.9.10+
`solr-managed`	`solr`	`init/enable-tls-in-solr`	✅	✅	5.9.10+
`solr-managed`	`solr`	`solr`	✅	❌	5.9.10+
`solr-managed`	`solr`	`process-raw`	✅	❌	5.9.10+
`solr-backup-runner`	`solr-backup-runner-backup`	`solr-backups`	✅	❌	5.9.10+
`solr-backup-runner`	`solr-backup-runner-prune`	`solr-prune`	✅	❌	5.9.10+
`templating`	`templating`	`templating`	✅	✅	5.9.9+
`webapps`	`webapps`	`webapps`	✅	✅	5.9.9+
`zookeeper`	`zookeeper`	`zookeeper`	✅	❌	5.9.10+
`zookeeper`	`zookeeper`	`jmx-exporter`	✅	❌	5.9.10+
`zookeeper`	`zookeeper`	`zookeeper-exporter`	✅	❌	5.9.10+
`zookeeper`	`zookeeper-chroots`	`main`	✅	❌	5.9.10+

Enable Recommendations

How-to

> Compatible versions: 4.2
>
> Works with: *Fusion AI*, *Fusion Server*

When you enable recommendations on a per-collection basis:

A set of default objects is created, including jobs you can schedule and pipelines you can configure to fetch the recommendation data.

Signals are enabled.

How to enable recommendations for a collection

1. From the collection picker, select the collection for which you want to enable recommendations, if it is not selected already.
2. Navigate to **Relevance > Recommendations**.
   The Enable Recommendations window appears, displaying the list of objects that will be created when you enable recommendations.
   If recommendations are already enabled for this collection, then the Object Explorer appears, displaying the recommendations-related objects in this app.
3. Click Enable Recommendations.

It may take a few moments for Fusion to create the default objects.

For more information, see *Getting Started with Recommendations and Boosting*.

Enable Social Features

How-to

> Compatible versions: 4.2
>
> Works with: *App Studio*

Prior to using social and collaboration tools, you must configure them in Fusion and in App Studio, and then enable them.

Configure Fusion

How to configure Fusion (*Fusion Server 4.1 and earlier*)

1. In the `fusion-app-name` Fusion app, create an `fusion-app-name_user_data` Fusion collection with 1 shard and 2 replicas.
2. Update the `managed-schema.xml` to include the new social fields:

```
<field indexed="true" multiValued="false" name="type"
required="false" stored="true" type="string"/> <field
indexed="true" multiValued="false" name="user_id"
required="false" stored="true" type="string"/> <field
indexed="true" multiValued="false" name="full_name"
required="false" stored="true" type="string"/> <field
indexed="true" multiValued="false" name="anonymous"
required="false" stored="true" type="boolean"/> <field
indexed="true" multiValued="false" name="created"
required="false" stored="true" type="pdate"/> <field
indexed="true" multiValued="false" name="created_epoch"
required="false" stored="true" type="plong"/> <field
indexed="true" multiValued="false" name="name" required="false"
stored="true" type="string"/> <field indexed="true"
multiValued="false" name="path" required="false" stored="true"
type="string"/> <field indexed="true" multiValued="false"
name="private" required="false" stored="true" type="boolean"/>
<field indexed="true" multiValued="false" name="query_url"
required="false" stored="true" type="string"/> <field
indexed="true" multiValued="false" name="collection"
required="false" stored="true" type="string"/> <field
indexed="true" multiValued="false" name="target"
required="false" stored="true" type="string"/> <field
indexed="true" multiValued="false" name="text" required="false"
stored="true" type="string"/> <field indexed="true"
multiValued="false" name="title" required="false" stored="true"
type="string"/> <field indexed="true" multiValued="false"
name="description" required="false" stored="true"
type="string"/> <field indexed="true" multiValued="false"
name="url" required="false" stored="true" type="string"/> <field
indexed="true" multiValued="false" name="accessibility"
required="false" stored="true" type="string"/> <field
indexed="true" multiValued="false" name="namespace"
```

```
required="false" stored="true" type="string"/> <field
indexed="true" multiValued="false" name="creator__anonymous"
required="false" stored="true" type="boolean"/> <field
indexed="true" multiValued="false" name="creator__full_name"
required="false" stored="true" type="string"/> <field
indexed="true" multiValued="false" name="creator__id"
required="false" stored="true" type="plong"/> <field
indexed="true" multiValued="false" name="creator__user_id"
required="false" stored="true" type="string"/> <field
indexed="true" multiValued="false" name="topic__id"
required="false" stored="true" type="plong"/> <field
indexed="true" multiValued="false" name="topic__title"
required="false" stored="true" type="string"/> <field
indexed="true" multiValued="false" name="topic__private"
required="false" stored="true" type="boolean"/> <field
indexed="true" multiValued="false" name="topic__path"
required="false" stored="true" type="string"/> <field
indexed="true" multiValued="false" name="topic__namespace"
required="false" stored="true" type="string"/> <field
indexed="true" multiValued="false" name="topic__description"
required="false" stored="true" type="string"/> <field
indexed="true" multiValued="false" name="topic__accessibility"
required="false" stored="true" type="string"/> <field
indexed="true" multiValued="false" name="topic__collection"
required="false" stored="true" type="string"/> <field
indexed="true" multiValued="false" name="topic__created"
required="false" stored="true" type="pdate"/> <field
indexed="true" multiValued="false" name="topic__created_epoch"
required="false" stored="true" type="plong"/> <field
indexed="true" multiValued="false"
name="topic__creator__anonymous" required="false" stored="true"
type="boolean"/> <field indexed="true" multiValued="false"
name="topic__creator__full_name" required="false" stored="true"
type="string"/> <field indexed="true" multiValued="false"
```

```
name="topic__creator__id" required="false" stored="true"
type="plong"/> <field indexed="true" multiValued="false"
name="topic__creator__user_id" required="false" stored="true"
type="string"/>
```

Configure App Studio

Configure App Studio to support social features.

Upgrade to the latest version of Appkit

Upgrade to the latest version of Appkit.

Enable social features

1. Create the file **`social.conf`** in the **`conf/social`** directory and add the following parameter to it:

```
platform: platforms.fusion.social
```

2. Create the file **`social.conf`** in the **`conf/platforms/Fusion`** directory and add the following parameters to it:

```
# Required for query/index pipeline collection: fusion-app-
name_user_data # Required for query/index profiles index-
profile: fusion-app-name_user_data query-profile: fusion-app-
name_user_data # Allow Fusion platform to store Social entities
readOnly: false webservice-enabled: false
```

Enable Superset Fusion 5.4 to 5.5

How-to

> Compatible versions: 5.5
>
> Works with: *Fusion*

Superset can be enabled with the Fusion Helm chart. It is not enabled by default.

> **Note**
>
> Managed Fusion users should reach out to our Customer Support team to discuss Superset. These instructions are for on-prem Fusion users.

You can choose to deploy Superset with a dedicated gateway for extra security. If you choose this option, you log in to the gateway before logging in to Superset.

See *Use Superset with Fusion SQL* for a detailed tutorial.

How to enable Superset

1. In your custom values YAML file, add *one* of the following, depending on whether you want the load balancer at the gateway or directly on the Superset container:

Load balancer at the gateway	**Load balancer on the Superset container**
Use this option for the best security.	Use this option to bypass the gateway login.
Configuration: `fusion-superset: enabled: true proxy: service: type: LoadBalancer`	**Configuration:** `fusion-superset: enabled: true service: type: LoadBalancer`

   For security, `ingress.tls.enabled: true` is also strongly recommended. Enabling TLS ensures that the basic authentication credentials sent to the gateway cannot be viewed in transit. If you enable TLS, you must create a Kubernetes TLS secret named `<namespace>-fusion-sql-tls-secret` that holds the certificate and key needed for Kubernetes to serve TLS traffic.
2. Run the *setup script* using the `--upgrade` option, specifying your custom values YAML file using the `--values` option.
   The example below uses the setup script for Google Kubernetes Engine (GKE):

```
./setup_f5_gke.sh -c mycluster -p myproject -n mynamespace --
region us-west1-c --upgrade --values
gke_mycluster_mynamespace_fusion_values.yaml
```

3. *If you configured the load balancer at the gateway in step 1*, retrieve the password for the `superset_gateway` service:

a. Run `kubectl --namespace <namespace> get secret superset-proxy-service-creds -o jsonpath='{.data.password}' | base64 -d`

```
GrDOQXzJ2hUgWrlGLLB2yh1kEjhWZABX
```

b. Copy the password. The username is `superset_gateway`.

4. Initialize the Superset admin user:
   a. *Download the initialization script*.
   b. Run the script:

```
./superset_init.sh <namespace> <superset-pod> <admin-password>
```

   This initializes the `admin` user with the password you specify (`<admin-password>`).

5. Get the IP and port of the Superset load balancer:

```
kubectl get services --namespace=<namespace>
```

6. Connect to the Superset load balancer at `https://<IP-address>:<port>/` and log in.
   - *If you configured the load balancer at the gateway*, log in to the gateway using the `superset_gateway` username and the password you retrieved above.
   - Log in to Superset as `admin`, with the password you set when you initialized the admin user above.
     From here, you can create additional Superset users. There is no authentication timeout, so your session persists indefinitely.

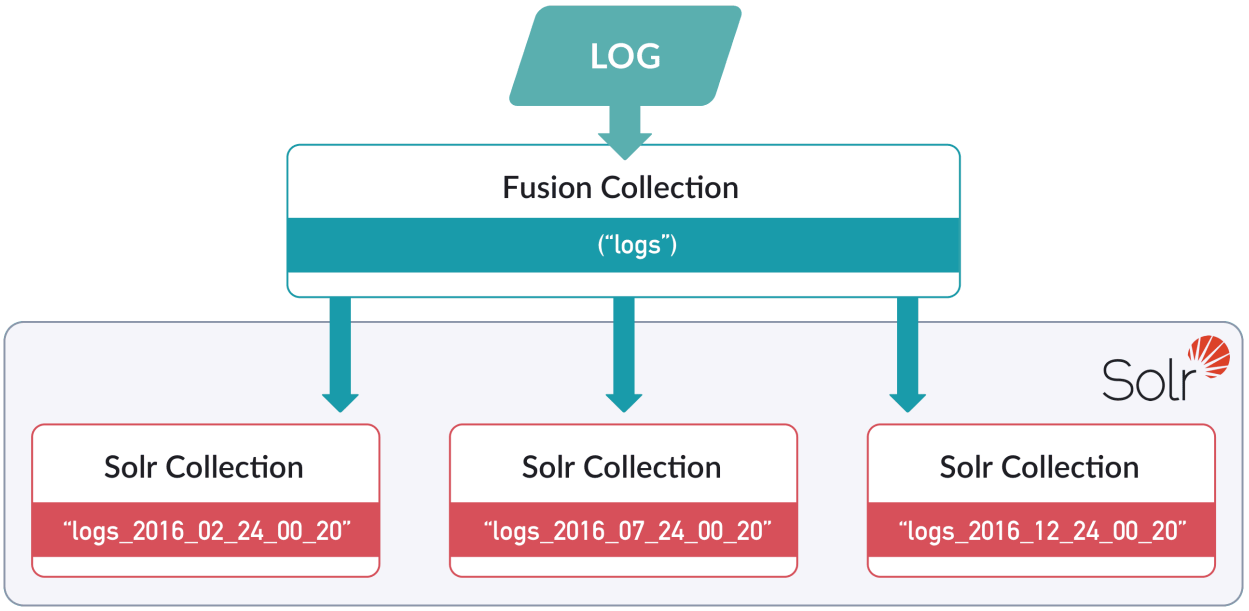
See *Use Superset with Fusion SQL* for a detailed tutorial.

Enable Time-based Partitioning

How-to

> Compatible versions: 4.2
>
> Works with: *Fusion Server*, *Fusion*

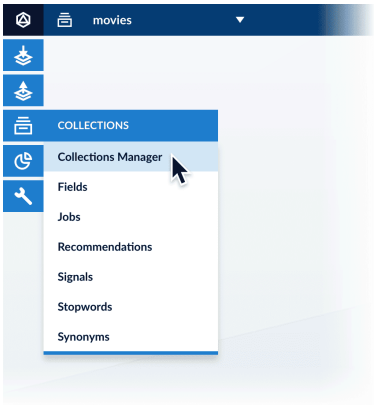
A Fusion collection can be configured to map to multiple Solr collections, known as partitions in this context, where each partition contains data from a specific time range. An example is time-based partitioning for logs:



- In the UI, you can only enable time-based partitioning for *new* collections.
- In the API, you can only enable time-based partitioning for *existing* collections.

Enablement using the Fusion UI

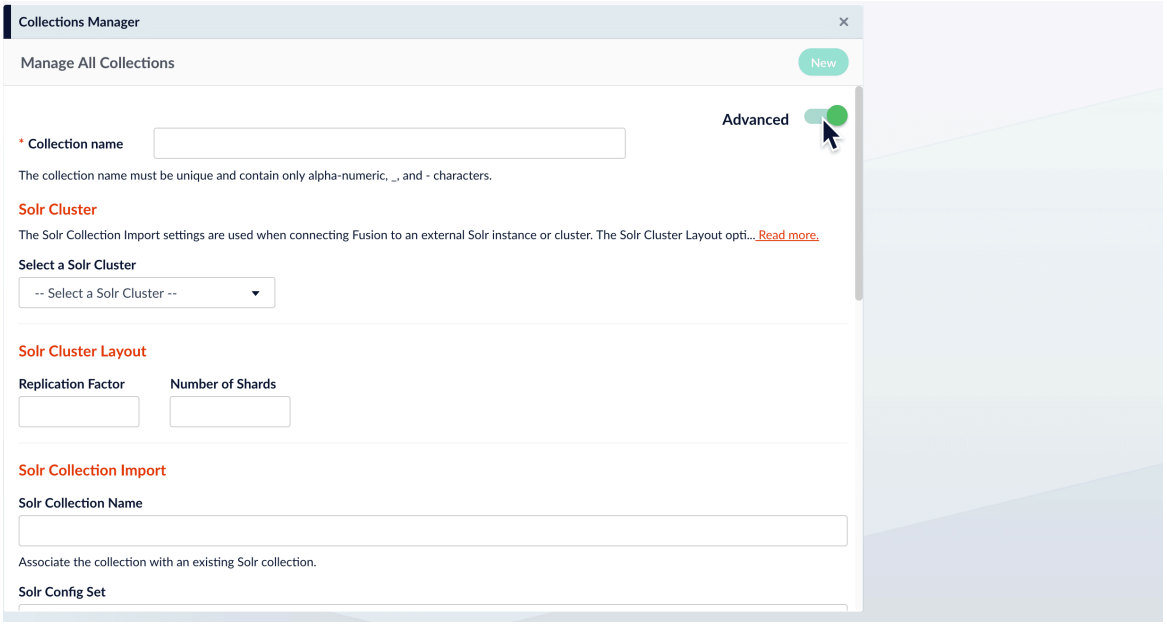
1. Open the Collections Manager:



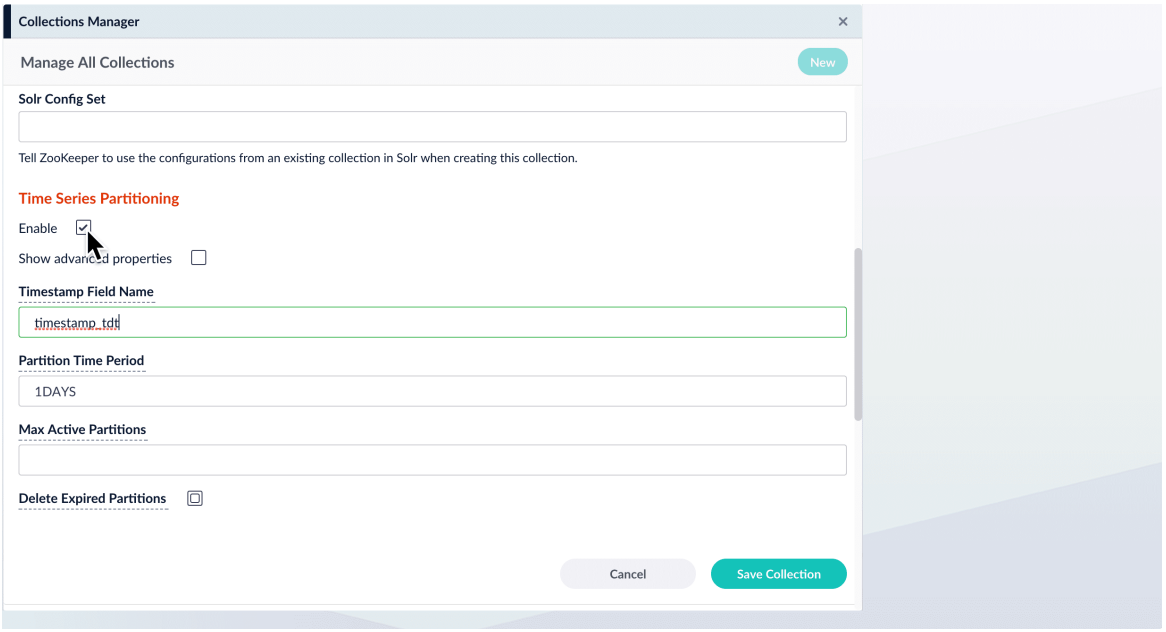
2. Click **New**.

> **Note**
>
> In the UI, you can only enable time-based partitioning for *new* collections. To enable it for an existing collection, *use the API*.

3. Click **Advanced**.



4. Scroll down to "Time Series Partitioning".
5. Click **Enable**.



When you enable this option, Fusion displays the *time series partitioning configuration options*.

6. Save the collection.

Enablement using the API

Use the *Collection Features API* to enable time-based partitioning for an existing collection.

Enable time-based partitioning using the default configuration:

```
curl –X PUT –H 'Content–type: application/json' –d '{"enabled": true}'
http://localhost:8764/api/collections/COLLECTION_NAME/features/partiti
onByTime
```

No response is returned.

Submit an empty request to the same endpoint to verify that time-based partitioning is enabled:

```
curl –X GET
http://localhost:8764/api/collections/COLLECTION_NAME/features/partiti
onByTime
```

Response:

```
{ "name" : "partitionByTime", "collectionId" : "COLLECTION_NAME",
"params" : { }, "enabled" : true }
```

Enable Transport Layer Security (TLS) for Fusion Microservices

How-to

> Compatible versions: 5.9, 5.11, 5.12
>
> Works with: *Fusion*

Important

This feature is only available in Fusion releases 5.9.x starting with 5.9.2, and 5.11.0 and later.

This article describes how to deploy Fusion with Transport Layer Security (TLS) enabled for *Fusion microservices*.

When enabled, Fusion generates a TLS certificate for each pod when the pod starts. This allows Fusion to use the Kubernetes endpoints API to reach each pod by its IP address and perform load balancing, circuit breaking, and retries in the Fusion microservices.

In order to facilitate the TLS operations, Fusion utilizes Jetstack's **cert-manager** add-on to provision a certificate for each pod. This certificate contains the pods' IP address.

Warning

It is *not* possible to update an existing cluster enable or disable TLS. These instructions apply to new deployments only.

Install Jetstack cert-manager

1. Add the Jetstack helm repo.

```
helm repo add jetstack https://charts.jetstack.io
```

2. Update the local cache.

```
helm repo update
```

3. Create the CRDs required for Jetstack.
   For Jetstack v1.12.4:

```
kubectl apply -f https://github.com/cert-manager/cert-
manager/releases/download/v1.12.4/cert-manager.crds.yaml
```

   For Jetstack v1.13.1:

```
kubectl apply -f https://github.com/cert-manager/cert-
manager/releases/download/v1.13.1/cert-manager.crds.yaml
```

4. Create the namespace for **cert-manager**.

```
kubectl create namespace "cert-manager"
```

5. Install **cert-manager** into the namespace.
ForJetstack v1.12.4:

```
helm upgrade --install --namespace "cert-manager" cert-manager
jetstack/cert-manager --version 1.12.4 --set 'extraArgs[0]=--
enable-certificate-owner-ref=true'
```

For Jetstack v1.13.1:

```
helm upgrade --install --namespace "cert-manager" cert-manager
jetstack/cert-manager --version 1.13.1 --set 'extraArgs[0]=--
enable-certificate-owner-ref=true'
```

Important

You must only complete this process once per Fusion cluster. All namespaces in the cluster are affected by this process.

Prepare the namespace for Fusion

1. Create the namespace to install Fusion into.

```
kubectl create namespace ${KUBE_NAMESPACE}
```

2. Create the Root CA certificate for the namespace that will be used to sign all certificates in the namespace.

```
cat <<EOF | cfssl genkey -initca - | cfssljson -bare ca {
"hosts": [ ], "key": { "algo": "rsa", "size": 4096 }, "names": [
{ "C": "US", "L": "San Francisco", "O": "Lucidworks", "OU":
"Engineering", "ST": "California" } ] } EOF kubectl --namespace
"${KUBE_NAMESPACE}" create secret generic cert-manager-ca --
from-literal=tls.crt="$(cat ca.pem)" --from-
literal=tls.key="$(cat ca-key.pem)"
```

3. Create a cert-manager issuer to sign CSRs in the namespace.
For Jetstack v1.12.4:

```
cat > ca-issuer.yaml <<EOF apiVersion: cert-manager.io/v1 kind:
Issuer metadata: name: ${KUBE_NAMESPACE}-ca-issuer spec: ca:
secretName: cert-manager-ca EOF kubectl --namespace
"${KUBE_NAMESPACE}" apply -f ca-issuer.yaml
```

For Jetstack v1.13.1:

```
cat > ca-issuer.yaml <<EOF apiVersion: cert-manager.io/v1alpha2
kind: Issuer metadata: name: ${KUBE_NAMESPACE}-ca-issuer spec:
ca: secretName: cert-manager-ca EOF kubectl --namespace
"${KUBE_NAMESPACE}" apply -f ca-issuer.yaml
```

4. Install Fusion with the following parameters:

```
helm install... --set global.tlsEnabled=true --set
global.tlsIssuerRef=${KUBE_NAMESPACE}-ca-issuer --set
global.zkPort=2281 --set global.kafkaPort=9092 --set
kafka.auth.clientProtocol=tls --set global.zkReplicaCount=3
```

Enable User-for-item recommendations

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

To enable this recommendation method, you only need to create a new collection and modify the configuration of the default recommender job, as explained below.

How to enable users-for-item recommendations

1. Navigate to **Collections** > **Collections Manager**.
2. Click **New**
3. Enter a collection name, such as `MAIN_COLLECTION_users_for_item_recommendations`.
4. Click **Save Collection**.

> **Tip**
>
> You can safely disable signals for this new collection.

5. In your main collection, verify that signals and recommendations are *enabled*.
6. Navigate to **Collections** > **Jobs**.
7. Open the `COLLECTION_NAME_item_recommendations` job.
   Alternatively, you can create *BPR Recommender jobs*.
8. Click **Advanced**.
9. In the **Users-for-items Recommendation Collection** field, enter the collection name `COLLECTION_NAME_users_for_item_recommendations`.
10. Click **Save**.

Enable User-for-item Recommendations

How-to

> Compatible versions: 4.2, 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion AI*, *Fusion Server*, *Fusion*

To enable this recommendation method, you only need to create a new collection and modify the configuration of the default recommender job, as explained below.

How to enable users-for-item recommendations

1. Navigate to **Collections** > **Collections Manager**.
2. Click **New**
3. Enter a collection name, such as `<main-collection>_users_for_item_recommendations`.
4. Click **Save Collection**.

> **Tip**
>
> You can safely disable signals for this new collection.

5. In your main collection, verify that signals and recommendations are *enabled*.
6. Navigate to **Collections** > **Jobs**.
7. Open the `COLLECTION_NAME_item_recommendations` job.
   Alternatively, you can create the following for these recommendations based on the Fusion release:
   - Fusion 4.x - *ALS Recommender job*
   - Fusion 5.1 - *ALS Recommender job*
   - Fusion 5.2 - *ALS Recommender job* or *BPR Recommender job*
   - Fusion 5.3 and later - *BPR Recommender job*
8. Click **Advanced**.
9. In the **Users-for-items Recommendation Collection** field, enter the collection name `COLLECTION_NAME_users_for_item_recommendations`.
10. Click **Save**.

Encrypt Sensitive Values

How-to

> Compatible versions: 4.2
>
> Works with: *App Studio*

In a downloaded project, the `bin/twigcrypt/twigcrypt.sh` utility is available to encrypt sensitive string values, such a passwords, at the command line. It uses a two-way encryption mechanism, so anywhere in the code this is used the value can be decrypted.

To encrypt a value, run the following (note the single quotes around `yourSensitiveValue`):

```
./twigcrypt.sh yourSecretSeed 'yourSensitiveValue'
```

This outputs an encrypted string. You must copy the whole string and paste into your configuration file.

For example, in your `src/main/resources/conf/platforms/fusion/fusion.conf` file, add:

```
username:jbloggs password:Enc(ABC123==)
```

Then you must also configure the seed in the application's security configuration, in `src/main/resources/conf/security/security.conf`:

```
password: yourSecretSeed
```

Wherever this configuration parameter is used, it will be decrypted back to plain text at the time it is used.

Note

Lucidworks recommends using a randomly-generated **alphanumeric** seed (special characters can cause problems).

Adding sensitive configuration parameters to Spring Security XML files

Two options for handling sensitive configuration parameters with Appkit and App Studio Spring Security XML files are:

- Environment variables
- Configuration parameters

Passing environment variables

Assuming there is already a configuration file wired up for property lookup, you can use the following property using the environment variable `USER_TEST`:

```
my property: \#{systemEnvironment['USER_TEST']}
```

Passing in configuration parameters

Use a system property passed at startup with `-Duser_test=myvalue`:

```
my property: \#{systemProperties['user_test']}
```

Evaluate a Smart Answers Query Pipeline

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion*

The *Smart Answers Evaluate Pipeline job* (*Evaluate QnA Pipeline job* in Fusion 5.1 and 5.2) evaluates the rankings of results from any *Smart Answers* pipeline and finds the best set of weights in the ensemble score. This topic explains how to set up the job.

Before beginning this procedure, prepare a machine learning model using either the *Supervised method* or the *Cold start method*, or by selecting one of the *pre-trained cold start models*, then *Configure your pipelines*.

The input for this job is a set of test queries and the text or ID of the correct responses. At least 100 entries are needed to obtain useful results. The job compares the test data with Fusion's actual results and computes variety of the ranking metrics to provide insights of how well the pipeline works. It is also useful to use to compare with other setups or pipelines.

1. Prepare test data

1. Format your test data as query/response pairs, that is, a query and its corresponding answer in each row. You can do this in any format that Fusion support, but parquet file would be preferable to reduce the amount of possible encoding issues. The response value can be either the document ID of the correct answer in your Fusion index (preferable), or the text of the correct answer.

   Note

   If you use answer text instead of an ID, make sure that the answer text in the evaluation file is formatted identically to the answer text in Fusion.

   If there are multiple possible answers for a unique question, then repeat the questions and put the pair into different rows to make sure each row has exactly one query and one response.
2. If you wish to index test data into Fusion, create a collection for your test data, such as `sa_test_input` and index the test data into that collection.

2. Configure the evaluation job

1. If you wish to save the job output in Fusion, create a collection for your evaluation data such as `sa_test_output`.
2. Navigate to **Collections** > **Jobs**.
3. Select **New** > **Smart Answers Evaluate Pipeline** (**Evaluate QnA Pipeline** in Fusion 5.1 and 5.2).
4. Enter a **Job ID**, such as `sa-pipeline-evaluator`.
5. Enter the name of your test data collection (such as `sa_test_input`) in the **Input Evaluation Collection** field.
6. Enter the name of your output collection (such as `sa_test_output`) in the **Output Evaluation Collection** field.

   Note

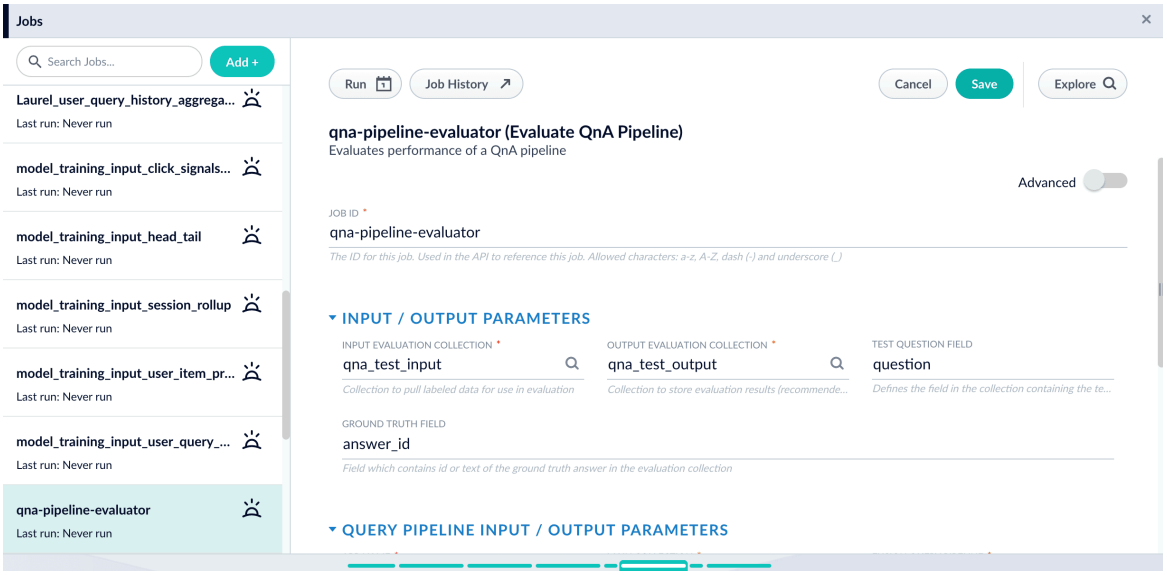
   In Fusion 5.3 and later, you can also configure this job to read from or write to cloud storage. See *Configure An Argo-Based Job to Access GCS* and *Configure An Argo-Based Job to Access S3*.

7. Enter the name of the **Test Question Field** in the input collection.
8. Enter the name of the answer field as the **Ground Truth Field**.
9. Enter the **App Name** of the Fusion app where the main Smart Answers content is indexed.
10. In the **Main Collection** field, enter the name of the Fusion collection that contains your Smart Answers content.
11. In the **Fusion Query Pipeline** field, enter the name of the Smart Answers query pipeline you want to evaluate.
12. In the **Answer Or ID Field In Fusion** field, enter the name of the field that Fusion will return containing the answer text or answer ID.
13. Optionally, you can configure the **Return Fields** to pass from Smart Answers collection into the evaluation output.

> **Tip**
>
> Check the Query Workbench to see which fields are available to be returned.

14. Configure the **Metrics** parameters:
    - **Solr Scale Function**
      Specify the function used in the Compute Mathematical Expression stage of the query pipeline, one of the following: * `max` * `log10` * `pow0.5`
    - **List of Ranking Scores For Ensemble**
      To find the best weights for different ranking scores, list the names of the ranking score fields, separated by commas. Different ranking scores might include Solr score, query-to-question distance, or query-to-answer distance from the Compute Mathematical Expression pipeline stage.
    - **Target Metric To Use For Weight Selection**
      The target ranking metric to optimize during weights selection. The default is `mrr@3`.

15. Optionally, *read about the advanced parameters* and consider whether to configure them as well.
    For example, **Sampling proportion** and **Sampling seed** provide a way to run the job only on a sample of the test data.
16. Click **Save**.



17. Click **Run** > **Start**.

3. Examine the output

The job provides a variety of metrics (controlled by the **Metrics list** advanced parameter) at different positions (controlled by the **Metrics@k list** advanced parameter) for the chosen final ranking score (specified in **Ranking score** parameter).

Example: Pipeline evaluation metrics

```
map@1:          0.454545
mrr@1:          0.454545
precision@1:    0.454545
recall@1:       0.454545
map@3:          0.606061
mrr@3:          0.606061
precision@3:    0.272727
recall@3:       0.818182
map@5:          0.646970
mrr@5:          0.646970
precision@5:    0.200000
recall@5:       1.000000
roc_auc:        0.981011
```

Example: recall@1,3,5 for different weights and distances

```
map@1:          0.454545
mrr@1:          0.454545
precision@1:    0.454545
recall@1:       0.454545
map@3:          0.606061
mrr@3:          0.606061
precision@3:    0.272727
recall@3:       0.818182
map@5:          0.646970
mrr@5:          0.646970
precision@5:    0.200000
recall@5:       1.000000
roc_auc:        0.981011
```

In addition to metrics, a results evaluation file is indexed to the specified output evaluation collection. It provides the correct answer position for each test question as well as the top returned results for each field specified in **Return fields** parameter.

See *See A Smart Answers Supervised Job's Status and Output* for more details about how to access this job's output.

Evaluate a Smart Answers Query Pipeline

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

The *Smart Answers Evaluate Pipeline job* evaluates the rankings of results from any *Smart Answers* pipeline and finds the best set of weights in the ensemble score. This topic explains how to set up the job.

Before beginning this procedure, prepare a machine learning model using either the *Supervised method* or the *Cold start method*, or by selecting one of the *pre-trained cold start models*, then *Configure your pipelines*.

The input for this job is a set of test queries and the text or ID of the correct responses. At least 100 entries are needed to obtain useful results. The job compares the test data with Managed Fusion's actual results and computes variety of the ranking metrics to provide insights of how well the pipeline works. It is also useful to use to compare with other setups or pipelines.

1. Prepare test data

1. Format your test data as query/response pairs, that is, a query and its corresponding answer in each row. You can do this in any format that Managed Fusion supports, but parquet file would be preferable to reduce the amount of possible encoding issues. The response value can be either the document ID of the correct answer in your Managed Fusion index (preferable), or the text of the correct answer.

   Note

   If you use answer text instead of an ID, make sure that the answer text in the evaluation file is formatted identically to the answer text in Managed Fusion.

   If there are multiple possible answers for a unique question, then repeat the questions and put the pair into different rows to make sure each row has exactly one query and one response.
2. If you wish to index test data into Managed Fusion, create a collection for your test data, such as `sa_test_input` and index the test data into that collection.

2. Configure the evaluation job

1. If you wish to save the job output in Managed Fusion, create a collection for your evaluation data such as `sa_test_output`.
2. Navigate to **Collections** > **Jobs**.
3. Select **New** > **Smart Answers Evaluate Pipeline**.
4. Enter a **Job ID**, such as `sa-pipeline-evaluator`.
5. Enter the name of your test data collection (such as `sa_test_input`) in the **Input Evaluation Collection** field.
6. Enter the name of your output collection (such as `sa_test_output`) in the **Output Evaluation Collection** field.
7. Enter the name of the **Test Question Field** in the input collection.
8. Enter the name of the answer field as the **Ground Truth Field**.
9. Enter the **App Name** of the Managed Fusion app where the main Smart Answers content is indexed.
10. In the **Main Collection** field, enter the name of the Managed Fusion collection that contains your Smart Answers content.
11. In the **Fusion Query Pipeline** field, enter the name of the Smart Answers query pipeline you want to evaluate.
12. In the **Answer Or ID Field In Fusion** field, enter the name of the field that Managed Fusion will return containing the answer text or answer ID.
13. Optionally, you can configure the **Return Fields** to pass from Smart Answers collection into the evaluation output.

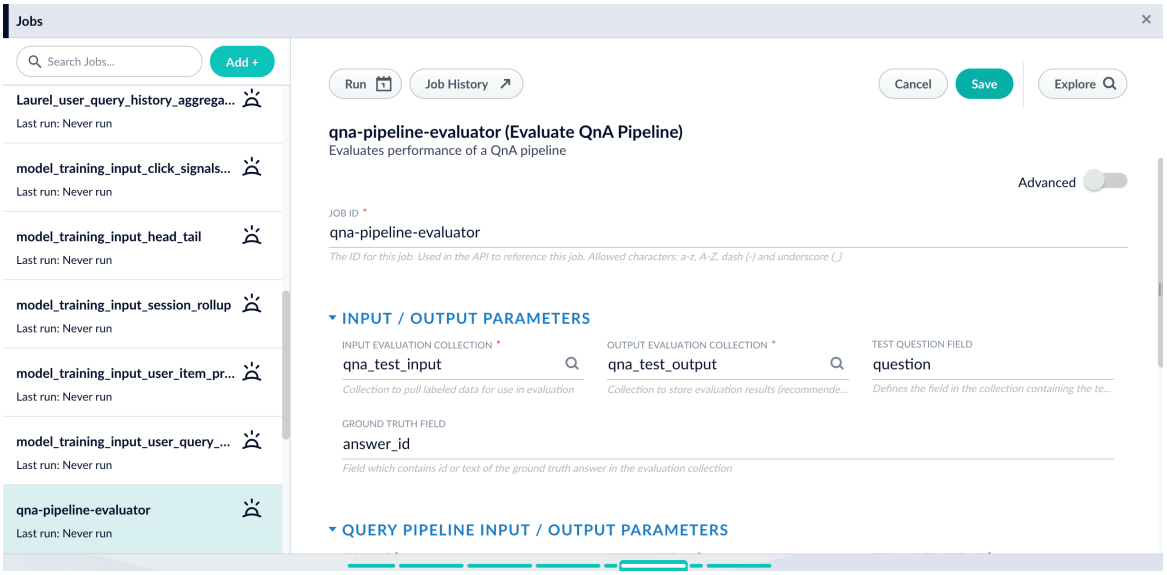
    Tip

    Check the Query Workbench to see which fields are available to be returned.

14. Configure the **Metrics** parameters:

- **Solr Scale Function**
  Specify the function used in the Compute Mathematical Expression stage of the query pipeline, one of the following: * `max` * `log10` * `pow0.5`
- **List of Ranking Scores For Ensemble**
  To find the best weights for different ranking scores, list the names of the ranking score fields, separated by commas. Different ranking scores might include Solr score, query-to-question distance, or query-to-answer distance from the Compute Mathematical Expression pipeline stage.
- **Target Metric To Use For Weight Selection**
  The target ranking metric to optimize during weights selection. The default is `mrr@3`.

15. Optionally, *read about the advanced parameters* and consider whether to configure them as well.
    For example, **Sampling proportion** and **Sampling seed** provide a way to run the job only on a sample of the test data.
16. Click **Save**.



17. Click **Run** > **Start**.

3. Examine the output

The job provides a variety of metrics (controlled by the **Metrics list** advanced parameter) at different positions (controlled by the **Metrics@k list** advanced parameter) for the chosen final ranking score (specified in **Ranking score** parameter).

Example: Pipeline evaluation metrics

```
map@1:          0.454545
mrr@1:          0.454545
precision@1:    0.454545
recall@1:       0.454545
map@3:          0.606061
mrr@3:          0.606061
precision@3:    0.272727
recall@3:       0.818182
map@5:          0.646970
mrr@5:          0.646970
precision@5:    0.200000
recall@5:       1.000000
roc_auc:        0.981011
```

Example: recall@1,3,5 for different weights and distances

```
map@1:          0.454545
mrr@1:          0.454545
precision@1:    0.454545
recall@1:       0.454545
map@3:          0.606061
mrr@3:          0.606061
precision@3:    0.272727
recall@3:       0.818182
map@5:          0.646970
mrr@5:          0.646970
precision@5:    0.200000
recall@5:       1.000000
roc_auc:        0.981011
```

In addition to metrics, a results evaluation file is indexed to the specified output evaluation collection. It provides the correct answer position for each test question as well as the top returned results for each field specified in **Return fields** parameter.

Export a Fusion App

How-to

> Compatible versions: 4.2
>
> Works with: *Fusion Server*

To export a Fusion app, you can use either the Fusion UI or the *Objects API*.

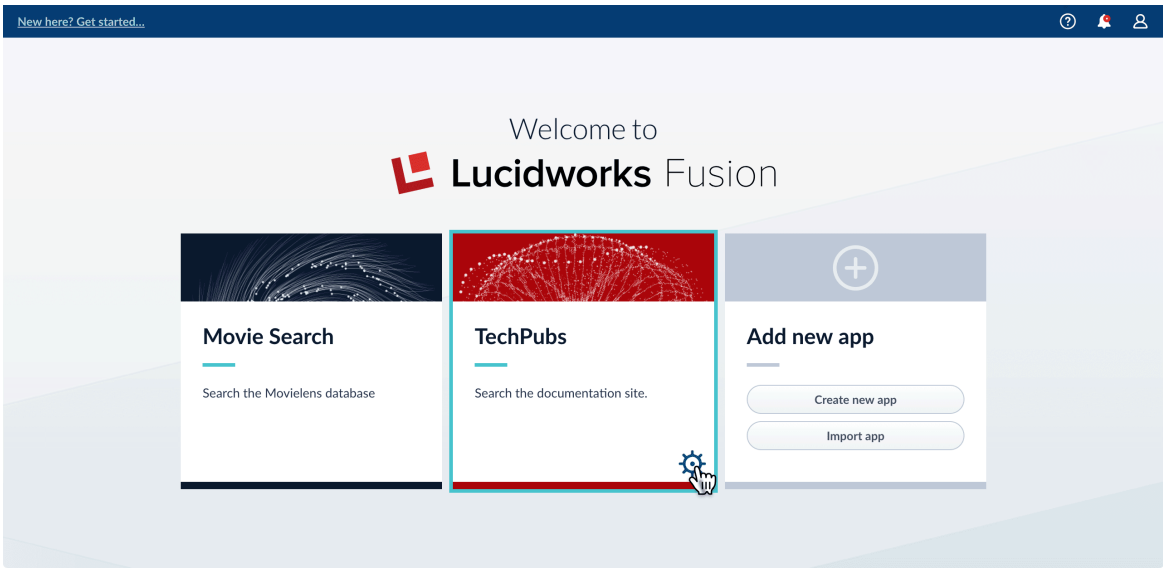
Important

When upgrading from one Fusion version to a later one, you must use the *migrator* to migrate objects. The migrator automates the process of translating relevant objects to the new version. Exporting an object from one Fusion version and importing it into a different Fusion version *isn't* supported.

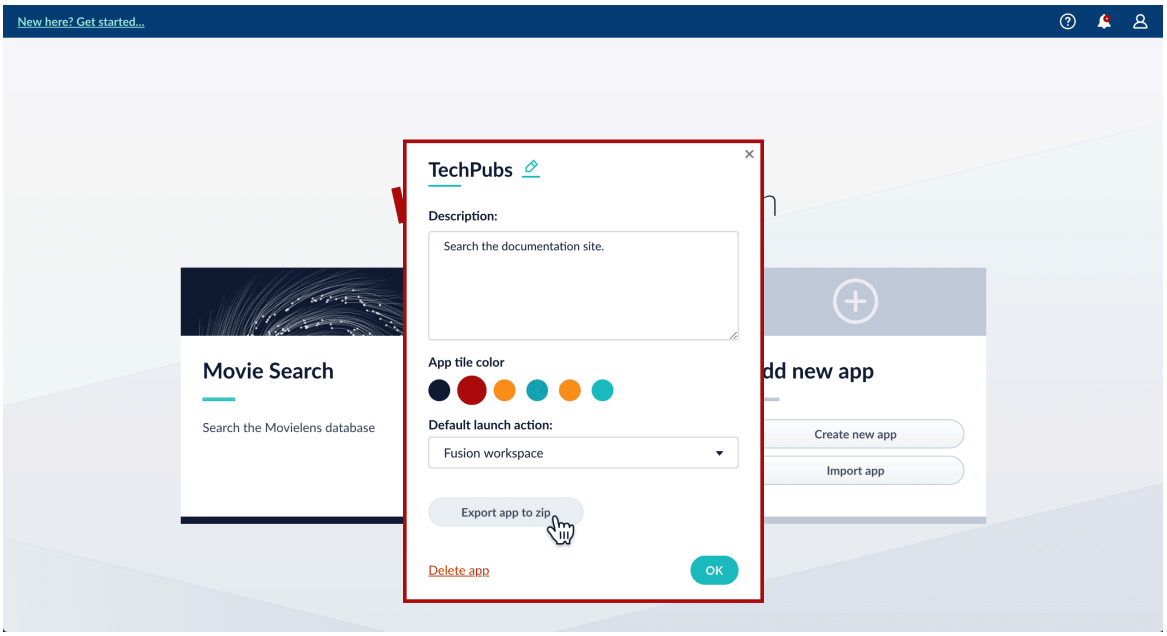
Export an app with the Fusion UI

How to export an app with the Fusion UI

1. Navigate to the launcher.
2. Hover over the app you want to export and click the Configure icon:



3. In the app config window, click **Export app to zip**:



See *import* for information to import the downloaded zip file into other instances of Fusion 4.x Server.

Export an app with the Objects API

The examples below show how to export one or more apps:

Export all apps

```
curl -u USERNAME:PASSWORD http://localhost:8764/api/objects/export?
type=app > all-apps.zip
```

Get all app IDs, then export one app by ID

```
> curl -u USERNAME:PASSWORD http://localhost:8764/api/apps [ { "id" :
"movies", "name" : "Movies", "description" : "Search the movielens
database.", "dataUri" : "/App-Tile-01-460x160.png", "properties" : {
"headerImageName" : "headerImage1", "tileColor" : "apps-darkblue",
"previousCollectionId" : "movies" } }, { "id" : "tech-pubs", "name" :
"TechPubs", "description" : "Search the documentation.", "dataUri" :
"/App-Tile-02-460x160.png", "properties" : { "headerImageName" :
"headerImage2", "tileColor" : "apps-darkblue", "previousCollectionId"
: "tech-pubs" } } ] > curl -u USERNAME:PASSWORD
http://localhost:8764/api/objects/export?app.ids=tech-pubs >
techpubs.zip
```

Note

When you export an app that includes objects that are shared with other apps, then all apps linked to that object are also exported.

Export two apps by ID

```
curl -u USERNAME:PASSWORD http://localhost:8764/api/objects/export?
app.ids=tech-pubs,movies > two-apps.zip
```

Tip

For the **app** object type, the **deep** parameter is not used. Linked objects are always included when exporting apps.

Export a Fusion App

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion*

To export a Fusion app, you can use either the Fusion UI or the *Objects API*.

Important

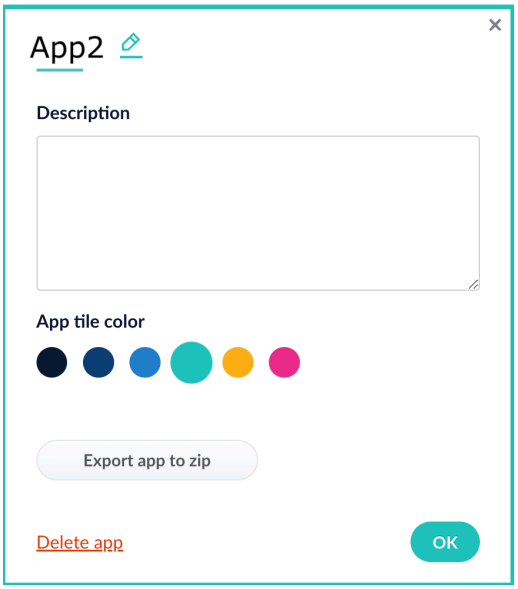
To upgrade from one Fusion version to a 5.x version, see *Fusion 5 Upgrades* to migrate objects.

Export an app with the Fusion UI

How to export an app with the Fusion UI

1. Navigate to the launcher.
2. Hover over the app you want to export and click the Configure icon:

3. In the app config window, click **Export app to zip**:



See *import* for information to import the downloaded zip file into other instances of Fusion 5.x.

Export an app with the Objects API

The examples below show how to export one or more apps:

Export all apps

```
curl -u USERNAME:PASSWORD http://localhost:6764/api/objects/export?
type=app > all-apps.zip
```

Get all app IDs, then export one app by ID

```
> curl -u USERNAME:PASSWORD http://localhost:6764/api/apps [ { "id" :
"movies", "name" : "Movies", "description" : "Search the movielens
database.", "dataUri" : "/App-Tile-01-460x160.png", "properties" : {
"headerImageName" : "headerImage1", "tileColor" : "apps-darkblue",
"previousCollectionId" : "movies" } }, { "id" : "tech-pubs", "name" :
"TechPubs", "description" : "Search the documentation.", "dataUri" :
"/App-Tile-02-460x160.png", "properties" : { "headerImageName" :
"headerImage2", "tileColor" : "apps-darkblue", "previousCollectionId"
: "tech-pubs" } } ] > curl -u USERNAME:PASSWORD
http://localhost:6764/api/objects/export?app.ids=tech-pubs >
techpubs.zip
```

Note

When you export an app that includes objects that are shared with other apps, then all apps linked to that object are also exported.

Export two apps by ID

```
curl -u USERNAME:PASSWORD http://localhost:6764/api/objects/export?
app.ids=tech-pubs,movies > two-apps.zip
```

Tip

For the **app** object type, the **deep** parameter is not used. Linked objects are always included when exporting apps.

Export a Managed Fusion App

How-to

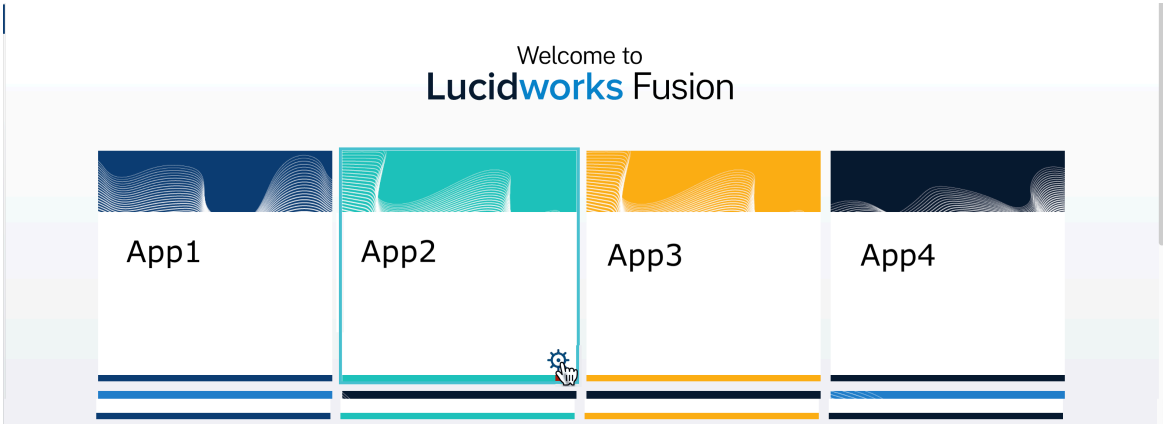
> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

To export a Managed Fusion app, you can use either the Managed Fusion UI or the *Objects API*.

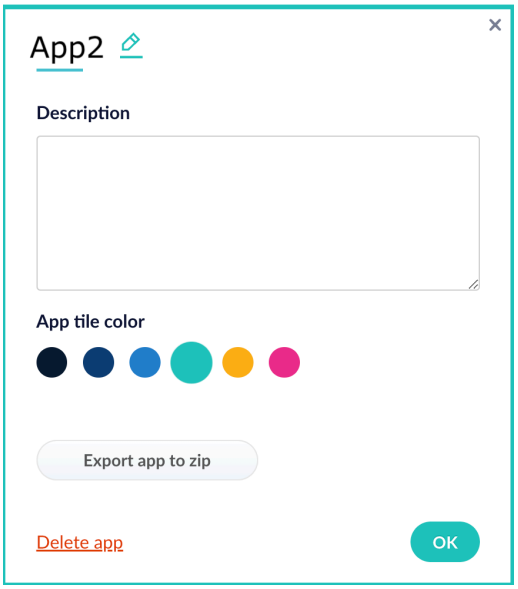
Export an app with the Managed Fusion UI

How to export an app with the Managed Fusion UI

1. Navigate to the launcher.
2. Hover over the app you want to export and click the Configure icon:



3. In the app config window, click **Export app to zip**:



Export an app with the Objects API

The examples below show how to export one or more apps:

Export all apps

```
curl -u USERNAME:PASSWORD
https://EXAMPLE_COMPANY.b.lucidworks.cloud/api/objects/export?type=app
> all-apps.zip
```

Get all app IDs, then export one app by ID

```
> curl -u USERNAME:PASSWORD
https://EXAMPLE_COMPANY.b.lucidworks.cloud/api/apps [ { "id" :
"movies", "name" : "Movies", "description" : "Search the movielens
database.", "dataUri" : "/App-Tile-01-460x160.png", "properties" : {
"headerImageName" : "headerImage1", "tileColor" : "apps-darkblue",
"previousCollectionId" : "movies" } }, { "id" : "tech-pubs", "name" :
"TechPubs", "description" : "Search the documentation.", "dataUri" :
"/App-Tile-02-460x160.png", "properties" : { "headerImageName" :
"headerImage2", "tileColor" : "apps-darkblue", "previousCollectionId"
: "tech-pubs" } } ] > curl -u USERNAME:PASSWORD
https://EXAMPLE_COMPANY.b.lucidworks.cloud/api/objects/export?
app.ids=tech-pubs > techpubs.zip
```

Note

When you export an app that includes objects that are shared with other apps, then all apps linked to that object are also exported.

Export two apps by ID

```
curl -u USERNAME:PASSWORD
https://EXAMPLE_COMPANY.b.lucidworks.cloud/api/objects/export?
app.ids=tech-pubs,movies > two-apps.zip
```

Tip

For the **app** object type, the **deep** parameter is not used. Linked objects are always included when exporting apps.

Export Data from the DevOps Center

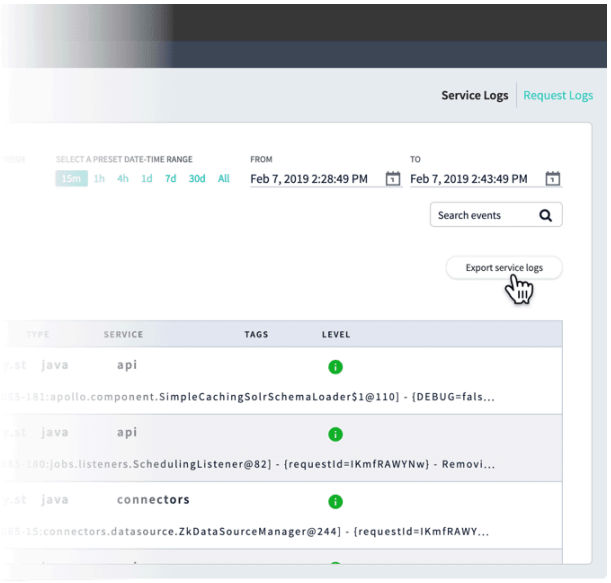
How-to

> Compatible versions: 4.2, 5.5
>
> Works with: *Fusion Server*, *Fusion*

The DevOps Center can export metrics and events in CSV format for the currently-selected time period. Exported metrics and events are useful for collaborating with other Fusion administrators, using third-party analysis tools, or sending troubleshooting data to the Lucidworks support team.

How to export data from the DevOps Center

1. In the Fusion UI, navigate to **System** > **DevOps Center** > **Log Viewer**.
2. Select the time period that is relevant to the incident you are interested in.
   Log files will be exported according to the tab you are viewing.



Click **Export service logs** in the Service Logs tab or **Export request logs** in the Request Logs tab to download a `.csv` file containing the logs (shown here with example timestamps):

- `serviceLogs[2019-02-07T22:28:49.000Z-2019-02-07T22:43:49.695Z].csv`
- `requestLogs[2019-02-07T22:28:49.000Z-2019-02-07T22:43:49.695Z].csv`

Export Data from the DevOps Center

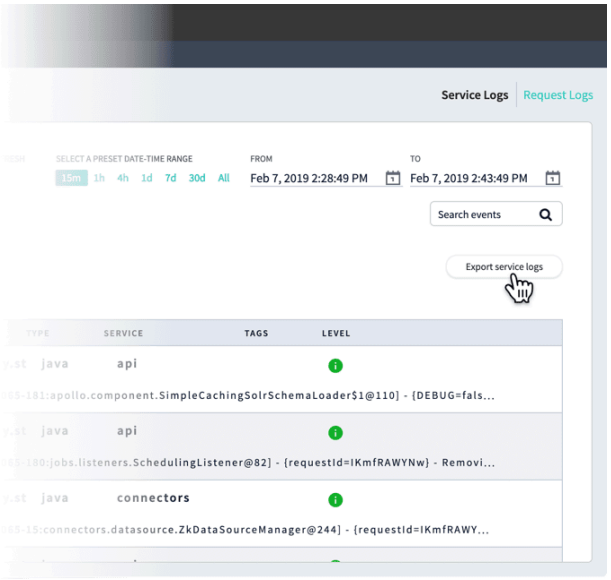
How-to

> Compatible versions: 5.5
>
> Works with: *Managed Fusion*

The DevOps Center can export metrics and events in CSV format for the currently-selected time period. Exported metrics and events are useful for collaborating with other Managed Fusion administrators, using third-party analysis tools, or sending troubleshooting data to the Lucidworks support team.

How to export data from the DevOps Center

1. In the Managed Fusion UI, navigate to **System** > **DevOps Center** > **Log Viewer**.
2. Select the time period that is relevant to the incident you are interested in.
   Log files will be exported according to the tab you are viewing.



Click **Export service logs** in the Service Logs tab or **Export request logs** in the Request Logs tab to download a `.csv` file containing the logs (shown here with example timestamps):

- `serviceLogs[2019-02-07T22:28:49.000Z-2019-02-07T22:43:49.695Z].csv`
- `requestLogs[2019-02-07T22:28:49.000Z-2019-02-07T22:43:49.695Z].csv`

Extract content from Web pages

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

The Web connector retrieves data from a Web site using HTTP and starting from a specified URL.

Managed Fusion releases 5.5 - 5.6.0 use the *Web V1 connector*. Managed Fusion releases 5.6.1 and later use the *Web V2 connector*.

The connector supports several approaches to extracting and filtering content from pages. When analyzing the HTML of a page, the connector can specifically include or exclude elements based on the HTML tag, the tag ID, or the tag class (such as a `div` tag, or the `#content` tag ID).

Specific tags can be selected to become fields of the document if needed. For example, all content from `<h1>` tags can be pulled into an `h1` field, and with field mapping be transformed into document titles.

For other advanced capabilities, you can use jsoup selectors to find elements in the content to include or exclude from the content.

While field mapping is generally a function of the index pipeline, you can define some initial mappings to occur during the crawl. The "initial mappings" property for each web datasource is predefined with three mappings: to move `fetchedDates` to a `fetchedDates_dts` field, to move `lastModified` to a `lastModified_dt` field, and to move `length` to a `length_l` field.

Finally, the crawler can deduplicate crawled content. You can define a specific field to use for this deduplication (such as title, or another field), or you can use the full raw content as the default. In the Managed Fusion UI, when you are defining your datasource, toggle **Advanced** to access the **Dedupe** settings.

Extract Content from Web Pages

How-to

> Compatible versions: 4.2, 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion Connectors*, *Fusion Server*, *Fusion*

The Web V1 connector retrieves data from a Web site using HTTP and starting from a specified URL.

The connector supports several approaches to extracting and filtering content from pages. When analyzing the HTML of a page, the connector can specifically include or exclude elements based on the HTML tag, the tag ID, or the tag class (such as a `div` tag, or the `#content` tag ID).

Specific tags can be selected to become fields of the document if needed. For example, all content from `<h1>` tags can be pulled into an `h1` field, and with field mapping be transformed into document titles.

For other advanced capabilities, you can use jsoup selectors to find elements in the content to include or exclude from the content.

While field mapping is generally a function of the index pipeline, you can define some initial mappings to occur during the crawl. The "initial mappings" property for each web datasource is predefined with three mappings: to move `fetchedDates` to a `fetchedDates_dts` field, to move `lastModified` to a `lastModified_dt` field, and to move `length` to a `length_l` field.

Finally, the crawler can deduplicate crawled content. You can define a specific field to use for this deduplication (such as title, or another field), or you can use the full raw content as the default. In the Fusion UI, when you are defining your datasource, toggle **Advanced** to access the **Dedupe** settings.

Extract short answers from longer documents

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

This topic explains how to deploy and configure the transformer-based deep learning model for short answer extraction with *Smart Answers*. This model is useful for analyzing long documents and extracting just a paragraph, a sentence, or a few words that answer the question.

Tip

This model is trained on the SQuAD2.0 dataset which consists of questions about Wikipedia articles and answers gleaned from those articles. Therefore, this model is most effective with Wikipedia-like content and may produce uneven results when applied to more informal content such as message boards.

Note

The out-of-the-box (OOTB) model only supports English content.

1. Deploy the model in Fusion

1. Navigate to **Collections** > **Jobs**.
2. Select **New** > **Create Seldon Core Model Deployment**.
3. Configure the job as follows:
    - **Job ID.** the ID for this job, such as `deploy-answer-extractor`.
    - **Model Name.** model name of the Seldon Core deployment that will be referenced in the Machine Learning pipeline stage configurations, such as `answer-extractor`.
    - **Docker Repository.** `lucidworks`
    - **Image Name.** `answer-extractor:v1.1`
    - **Kubernetes Secret Name for Model Repo.** (empty)
    - **Output Column Names for Model.** `[answer,score,start,end]`
4. Click **Save**.
5. Click **Run** > **Start**.

2. Configure the Machine Learning query stage

This model provides the best results when used with one of the *question-answering query pipelines*. The default query pipeline is called `APP_NAME-smart-answers`.

Starting with one of those pipelines, add a new Machine Learning stage to the end of the pipeline and configure it as described below.

How to configure short answer extraction in the query pipeline

1. Make sure you have performed the *basic configuration* of your query pipeline.
2. In the query pipeline, click **Add a Stage** > **Machine Learning**.
3. In the **Model ID** field, enter the model name you configured above, such as answer-extractor.
4. In the **Model input transformation script** field, enter the following:

```
var textFieldToExtract = "answer_t" var numDocsToExtract = 3
responses = new java.util.ArrayList(); var docs =
response.get().getInnerResponse().getDocuments(); for (var i=0;
i<numDocsToExtract; i++) {
responses.add(docs[i].getField(textFieldToExtract)) } var
modelInput = new java.util.HashMap() modelInput.put("question",
request.getFirstParam("q")) modelInput.put("context", responses)
modelInput.put("topk", 3)
modelInput.put("handle_impossible_answer", 'false') modelInput
```

Configure the parameters in the script as follows:

- `question` (Required). The name of the field containing the questions.

  Note

  Make sure that the question is provided as it was originally entered by user. If you have previous stages that augments question (like stopwords removing or synonyms expansion), it is better to copy original question and use it for the answer extraction without additional modifications.

- `context` (Required). A string or list of contexts; by default this is the first `num_docs_to_extract` documents in the output of the previous stage in the pipeline. If only one question is present with multiple contexts, that question will be applied to every context and vice versa for 1 context and multiple questions. If a list of questions and contexts is passed, a 1:1 mapping of questions and contexts will be created in the order in which they're passed.
- `topk`. The number of answers to return (will be chosen by order of likelihood). Default: 1
- `handle_impossible_answer`. Whether or not to deal with a question that has no answer in the context.
  If true, an empty string is returned. If false, the most probable (`topk`) answer(s) are returned regardless of how low the probability score is. Default: True

  Tip

  Experiment with this parameter to see what value returns the most acceptable answers.

For advanced use cases, you can add the following parameters to the script to override their defaults:

- `batch_size`. How many samples to process at a time. Reducing this number will reduce memory usage but increase execution time, while increasing it will increase memory usage and decrease execution time to a certain extent. Default: 8
- `max_context_len`. If set to greater than 0, truncate contexts to this length in characters. Default: 5000
- `max_answer_len`. The maximum length of predicted answers (for example, only answers with a shorter length are considered). Default: 15
- `max_question_len`. The maximum length of the question after tokenization. It will be truncated if needed. Default: 64

- `doc_stride`. If the context is too long to fit with the question for the model, it will be split in several chunks with some overlap. This argument controls the size of that overlap. Default: 128
- `max_seq_len`. The maximum length of the total sentence (context + question) after tokenization. The context will be split in several chunks (using doc_stride) if needed. Default: 384

5. In the **Model output transformation script** field, enter the following:

```
// Parse raw output from model var jsonOutput =
JSON.parse(modelOutput.get("_rawJsonResponse")) var parsedOutput
= {}; for (var i=0; i<jsonOutput["names"].length;i++){
parsedOutput[jsonOutput["names"][i]] = jsonOutput["ndarray"][i]
} // Get response documents var docs =
response.get().getInnerResponse().getDocuments(); var ndocs =
new java.util.ArrayList(); // Add extracted answers to the
response docs for (var i=0; i <
parsedOutput["answer"].length;i++){ var doc = docs[i];
doc.putField("extracted_answer", new
java.util.ArrayList(parsedOutput["answer"][i]))
doc.putField("extracted_score", new
java.util.ArrayList(parsedOutput["score"][i]))
doc.putField("extracted_start", new
java.util.ArrayList(parsedOutput["start"][i]))
doc.putField("extracted_end", new
java.util.ArrayList(parsedOutput["end"][i])) ndocs.add(doc); }
response.get().getInnerResponse().updateDocuments(ndocs);
```

6. Save the pipeline.

Model output

The model adds the following fields to the query pipeline output:

- `answer`. The short answer extracted from the context. This may be blank if `handle_impossible_answers=True` and `topk=1`.
- `score`. The score for the extracted answers.
- `start`. The start index of the extracted answer in the provided context.
- `end`. The end index of the extracted answer in the provided context.

Extract Short Answers from Longer Documents

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion*

This topic explains how to deploy and configure the transformer-based deep learning model for short answer extraction with *Smart Answers*. This model is useful for analyzing long documents and extracting just a paragraph, a sentence, or a few words that answer the question.

Tip

This model is trained on the SQuAD2.0 dataset which consists of questions about Wikipedia articles and answers gleaned from those articles. Therefore, this model is most effective with Wikipedia-like content and may produce uneven results when applied to more informal content such as message boards.

Note

The out-of-the-box (OOTB) model only supports English content.

Hardware recommendations

When creating a nodePool to perform short answer extraction, use a configuration that meets these guidelines in order to achieve the best performance:

- It is strongly recommended to use the latest possible Intel CPU architecture; Intel CascadeLake or higher architectures are recommended.
- Large core count is also recommended: 12-16 cores with 32G of RAM.

1. Deploy the model in Fusion

1. Navigate to **Collections** > **Jobs**.
2. Select **New** > **Create Seldon Core Model Deployment**.
3. Configure the job as follows:
    - **Job ID.** the ID for this job, such as `deploy-answer-extractor`.
    - **Model Name.** model name of the Seldon Core deployment that will be referenced in the Machine Learning pipeline stage configurations, such as `answer-extractor`.
    - **Docker Repository.** `lucidworks`
    - **Image Name.** `answer-extractor:v1.1`
    - **Kubernetes Secret Name for Model Repo.** (empty)
    - **Output Column Names for Model.** `[answer,score,start,end]`
4. Click **Save**.
5. Click **Run** > **Start**.

2. Configure the Machine Learning query stage

This model provides the best results when used with one of the *question-answering query pipelines*.

- In Fusion 5.3 and later, the default query pipeline is called `APP_NAME-smart-answers`.
- In Fusion 5.1 and 5.2, there are two: `APP_NAME-question-answering` and `APP_NAME-question-answering-dual-fields`.

Starting with one of those pipelines, add a new Machine Learning stage to the end of the pipeline and configure it as described below.

How to configure short answer extraction in the query pipeline

1. Make sure you have performed the *basic configuration* of your query pipeline.
2. In the query pipeline, click **Add a Stage** > **Machine Learning**.
3. In the **Model ID** field, enter the model name you configured above, such as answer-extractor.
4. In the **Model input transformation script** field, enter the following:

```
var textFieldToExtract = "answer_t" var numDocsToExtract = 3
responses = new java.util.ArrayList(); var docs =
response.get().getInnerResponse().getDocuments(); for (var i=0;
i<numDocsToExtract; i++) {
responses.add(docs[i].getField(textFieldToExtract)) } var
modelInput = new java.util.HashMap() modelInput.put("question",
request.getFirstParam("q")) modelInput.put("context", responses)
modelInput.put("topk", 3)
modelInput.put("handle_impossible_answer", 'false') modelInput
```

Configure the parameters in the script as follows:

- `question` (Required). The name of the field containing the questions.

  Note

  Make sure that the question is provided as it was originally entered by user. If you have previous stages that augments question (like stopwords removing or synonyms expansion), it is better to copy original question and use it for the answer extraction without additional modifications.

- `context` (Required). A string or list of contexts; by default this is the first `num_docs_to_extract` documents in the output of the previous stage in the pipeline. If only one question is present with multiple contexts, that question will be applied to every context and vice versa for 1 context and multiple questions. If a list of questions and contexts is passed, a 1:1 mapping of questions and contexts will be created in the order in which they're passed.
- `topk`. The number of answers to return (will be chosen by order of likelihood). Default: 1
- `handle_impossible_answer`. Whether or not to deal with a question that has no answer in the context.
  If true, an empty string is returned. If false, the most probable (`topk`) answer(s) are returned regardless of how low the probability score is. Default: True

  Tip

  Experiment with this parameter to see what value returns the most acceptable answers.

For advanced use cases, you can add the following parameters to the script to override their defaults:

- `batch_size`. How many samples to process at a time. Reducing this number will reduce memory usage but increase execution time, while increasing it will increase memory usage and decrease execution time to a certain extent. Default: 8
- `max_context_len`. If set to greater than 0, truncate contexts to this length in characters. Default: 5000
- `max_answer_len`. The maximum length of predicted answers (for example, only answers with a shorter length are considered). Default: 15
- `max_question_len`. The maximum length of the question after tokenization. It will be truncated if needed. Default: 64

- `doc_stride`. If the context is too long to fit with the question for the model, it will be split in several chunks with some overlap. This argument controls the size of that overlap. Default: 128
- `max_seq_len`. The maximum length of the total sentence (context + question) after tokenization. The context will be split in several chunks (using doc_stride) if needed. Default: 384

5. In the **Model output transformation script** field, enter the following:

```
// Parse raw output from model var jsonOutput =
JSON.parse(modelOutput.get("_rawJsonResponse")) var parsedOutput
= {}; for (var i=0; i<jsonOutput["names"].length;i++){
parsedOutput[jsonOutput["names"][i]] = jsonOutput["ndarray"][i]
} // Get response documents var docs =
response.get().getInnerResponse().getDocuments(); var ndocs =
new java.util.ArrayList(); // Add extracted answers to the
response docs for (var i=0; i <
parsedOutput["answer"].length;i++){ var doc = docs[i];
doc.putField("extracted_answer", new
java.util.ArrayList(parsedOutput["answer"][i]))
doc.putField("extracted_score", new
java.util.ArrayList(parsedOutput["score"][i]))
doc.putField("extracted_start", new
java.util.ArrayList(parsedOutput["start"][i]))
doc.putField("extracted_end", new
java.util.ArrayList(parsedOutput["end"][i])) ndocs.add(doc); }
response.get().getInnerResponse().updateDocuments(ndocs);
```

6. Save the pipeline.

Model output

The model adds the following fields to the query pipeline output:

- `answer`. The short answer extracted from the context. This may be blank if `handle_impossible_answers=True` and `topk=1`.
- `score`. The score for the extracted answers.
- `start`. The start index of the extracted answer in the provided context.
- `end`. The end index of the extracted answer in the provided context.

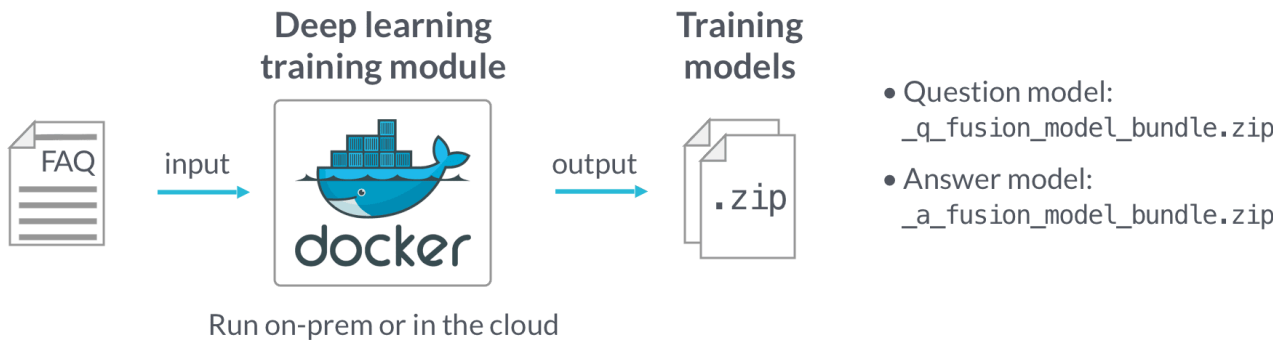
FAQ Solution Part 1

Model training (Fusion 4.2 only)

How-to

> Compatible versions: 4.2
>
> Works with: *Fusion AI*

The *FAQ solution for Smart Answers* begins with training a model using your existing data, as explained in this topic. Once you have trained a model, you can *deploy it*.



See also *Advanced Model Training Configuration for Smart Answers*.

Prerequisites

You need a Docker Hub account with access to the `lucidworks/ai-lab` repository. Contact Lucidworks AI Labs with your Docker Hub account name to gain access to this repository.

Hardware requirements and instance setup

The Docker training module can run on either a CPU or a GPU.

CPU is sufficient for most use cases.

GPU is 2-8 times faster. GPU is recommended if want to try our model auto-tune feature on a large dataset (e.g. over 20k QA pairs) or if the training data size is over 1 million entries for any of the training scenarios.

You can run the Docker image either on-prem or in the cloud. We recommend a minimum of 32GB RAM and 32 cores for a CPU server or a cloud instance. If you choose a GPU machine, one GPU with 11GB memory is sufficient for most training use cases.

Below are examples of the AWS instance types that we recommend:

CPU:

- AMI: Ubuntu Server 18.04 LTS (HVM), SSD Volume Type
- Instance Type: c5.9xlarge (36 vCPUs, 72 GB RAM)
- Recommended storage volume: Suggested 150GB plus 2.5 times the total input data size
- Networking
    - Allow inbound port 4440
    - Allow inbound port 5550

GPU:

- AMI: Deep Learning AMI (Ubuntu) Version 22.0
- Instance Type: p2.xlarge (11.75 ECUs, 4 vCPUs, 61 GB RAM, EBS only)
- Recommended storage volume: Suggested 150GB plus 2.5 times of total input data size.
- Networking
    - Allow inbound port 4440
    - Allow inbound port 5550

Below are examples of the Azure GPU instance type that we recommend:

- Deep Learning Virtual Machine
- Linux OS
- VM Size: 1x Standard NC6 (6 vcpus, 56 GB memory)
- Recommended OS Disk Size: Suggested 150GB plus 2.5 times the total input data size
- Networking
    - Allow inbound port 4440

- Allow inbound port 5550

Docker image setup

1. Determine the instance's external hostname.
   Refer to your cloud provider's documentation for instructions on how to do this.
2. Create a directory, such as `/opt/faq_training`, and ensure it meets *disk space requirements*. Also ensure that it is writable by the `docker` Linux user account.
   The training Docker container requires this directory to read input files, write temporary files, and write the output (deep learning models).
3. Run `docker login` and enter your account credentials.
4. Pull the `lucidworks/ai-lab:latest` image:

```
docker pull lucidworks/ai_lab:latest
```

5. Run the Docker image:
   For GPU instance:

```
docker run -d --rm \ -e "RD_HOSTNAME=<external hostname here>" \
-e "ENABLE_GPU=1" \ --name deep_faq \ -v <working directory
here>:/src/DL/working_dir \ --net=host \          --
runtime=nvidia \ lucidworks/ai_lab:latest
```

   For CPU-only instance:

```
docker run -d --rm \ -e "RD_HOSTNAME=<external hostname here>" \
--name deep_faq \ -v <working directory
here>:/src/DL/working_dir \ --net=host \
lucidworks/ai_lab:latest
```

   This starts the Docker container in the background.
6. Wait about one minute for the container to initialize.
   In your working directory, you should see these directories:

```
conf data input models output rundeck
```

- The `input` directory is where all training data and input files should be placed.
- The `output` directory contains the models that result from the training.
- All other directories are used by the model training container to save persistent files or temporary files.

> **Tip**
>
> You can clear the `data` folder periodically to save disk space.

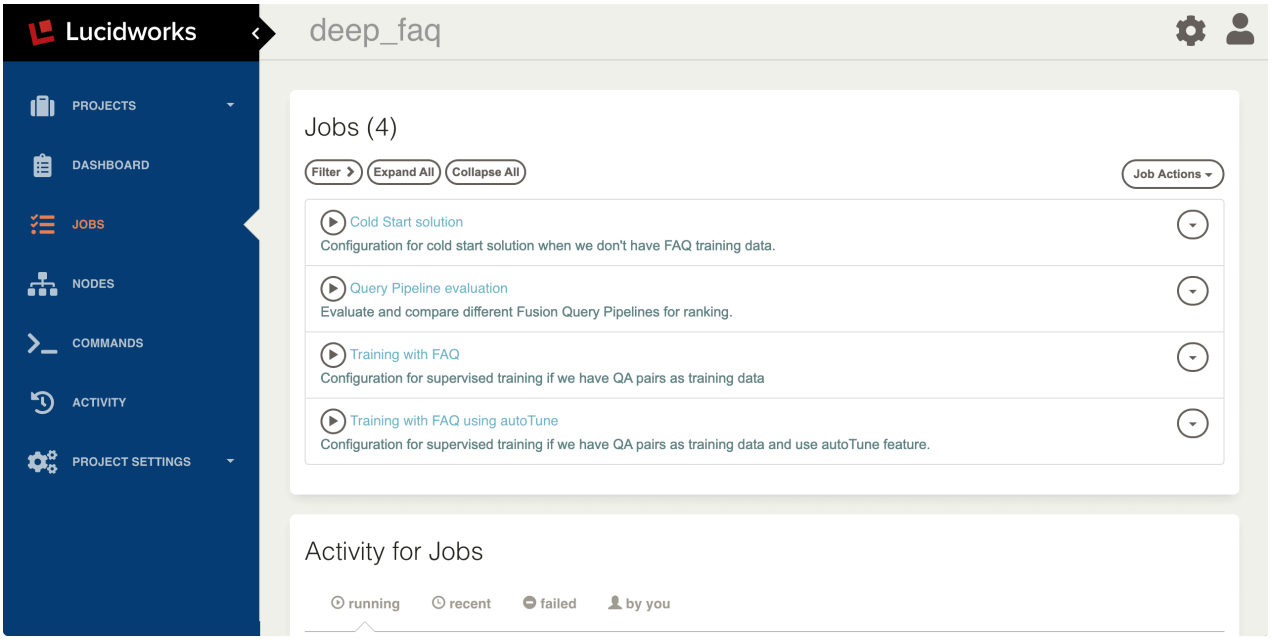
7. Point your browser to `http://<external hostname>:4440`.
   This redirects you to a Rundeck login screen.
8. Log in with your username and password.

Running Jobs

We use https://www.rundeck.com/open source[Rundeck^] to control job runs.

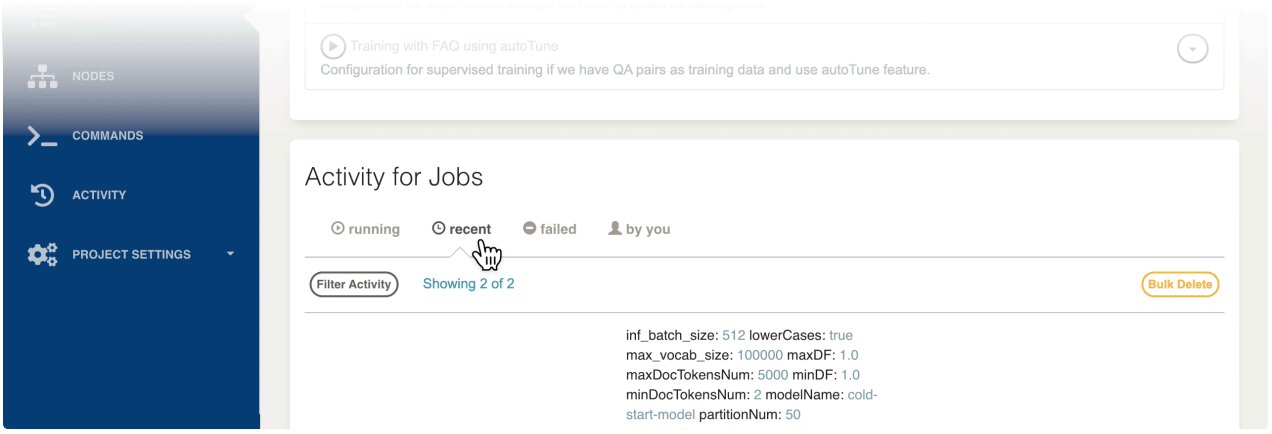
1. When Rundeck has started, click on the `deep_faq` project.
   This opens the **JOBS** panel, where you will see three jobs for different training scenarios and one job for query pipeline evaluation.
   - Click on a job to configure job parameters.
   - Click **Run Job Now** to start the job.
   - You can monitor the progress of a job by clicking on the **running** button under the **Activity for Jobs** section as shown in the screenshot below.

> **Note**
>
> If similar jobs have run before, then Rundeck estimates the total run time and displays a progress bar.



You can click on the running job to enter the activity control page, where you can kill the job or watch the log output.

If you want to reuse the previously-configured job, click the **recent** button under **Activity for Jobs**.



Select a recent job configuration that you want to modify, then click the button to enter the configuration page with previously-entered values.

For more details about Rundeck job control, refer to the Rundeck documentation .

Input data format

- FAQ data format
  The expected FAQ input is Q&A pairs, that is, a question and its corresponding answer in each row. If there are multiple possible answers for a unique question, then repeat the questions and put the Q&A pair into different rows to make sure each row has one question and one answer. Specify the question and answer field names in the **Question field name** and **Answer field name** parameters in the job configuration, respectively.

  Tip

  Be sure that your data has the correct JSON format such as in this example:

```
[{"question":"How to transfer personal auto lease to business auto lease?","answer":"I would approach the lender that you are getting the lease from..."} + {"question":"How to transfer personal auto lease to business auto lease?","answer":"See what the contract says about transfers or subleases..."}]
```

- Content document input data format
  If you choose to generate customized embeddings (as described *below*), then put the documents from which you want to learn the vocabulary into one or multiple fields and specify the field name in the **Field which contains the content documents** parameter.

Important

For both methods, we only support Parquet and JSON files. Make sure your data filename ends with either `.parquet` or `.json`.

Training scenarios and important configuration parameters

There are three main scenarios supported by three different jobs in the `deep_faq` project:

- If you have FAQ records available for training and want to use our auto-tune feature to automatically find the best model through hyper-parameter search, click on the **Training with FAQ using autoTune** job.

  Note

  We recommend using GPU for auto-tune if the FAQ data size is over 20k. GPU training speed for auto-tune can be 2 to 6 times faster than CPU. Specify whether you have GPU using the **GPU available** parameter.

- If you have FAQ records available for training and want to tune the model yourself, click on the **Training with FAQ** job.
  Only one set of hyper-parameters, which are specified in the UI, will be used to run the model. If you have a big dataset but do not have GPU, you can consider running this job using default parameter settings, or run autoTune mode once to help find best parameters, then use the found parameter in this job in future model updates.
- If you don't have FAQ records and want to use our cold start solution, click on the **Cold Start solution** job.
  This job can learn about the vocabulary that you want to search against through Word2vec. And we can combine Solr with Word2vec at query time using our query pipeline.

Input data parameters

- For training on FAQ data, these parameters are required:
  - QA pair file name
  - Question field name
  - Answer field name

- For cold start solution input, these parameters are required:
  - Content documents file name
  - Field which contains the content documents

For each of the *three scenarios above*, there are two vocabulary learning options:

- You can use the pre-trained word embeddings shipped with our training module.
  In the FAQ solution, these pre-trained word embeddings transfer words into embeddings for initialization of the deep learning model.
  In cold start mode, it gets the Word2vec vectors from the pre-trained file directly. The pre-trained word embeddings are built on a large corpus from the Internet, covering common words we use daily.
- If there are many domain-specific words or special tokens in your documents, we recommend training Word2vec using your own data.
  Set the **Generate customized embeddings** parameter to true to train from your own data. This uses the data and field specified in **Content documents file name** and **Field which contains the content documents** to train Word2vec vectors. This dataset can be the same as the FAQ input. If that's the case, then set the **Content documents file name** parameter to blank. Otherwise you can specify a different dataset, especially when the FAQ dataset is small.

Tip

When you use the pre-trained embeddings, the log shows the percentage of covered vocabulary words (search for vocab coverage by embeddings in the log). If the number of covered vocabulary words is less than 80%, then you should train Word2vec from your own data.

Note

For all training scenarios, you must provide a **Model name** for model tracking and version control purposes. If you use the same model name across different runs, the new model will replace the old model with the same name.

See also the *Advanced Model Training Configuration for Smart Answers*.

Result models

Track the running steps in the log, which provides information on data pre-processing, training steps, evaluations, and model generation in the end.

After training is finished, the final models and their associated files are saved in zip file format, which can be downloaded from `http://<external hostname>:5550/{modelName}`.

Download the two zipfiles ending with `_a_fusion_model_bundle.zip` and `_q_fusion_model_bundle.zip` to be used in the *model deployment* phase.

FAQ Solution Part 2

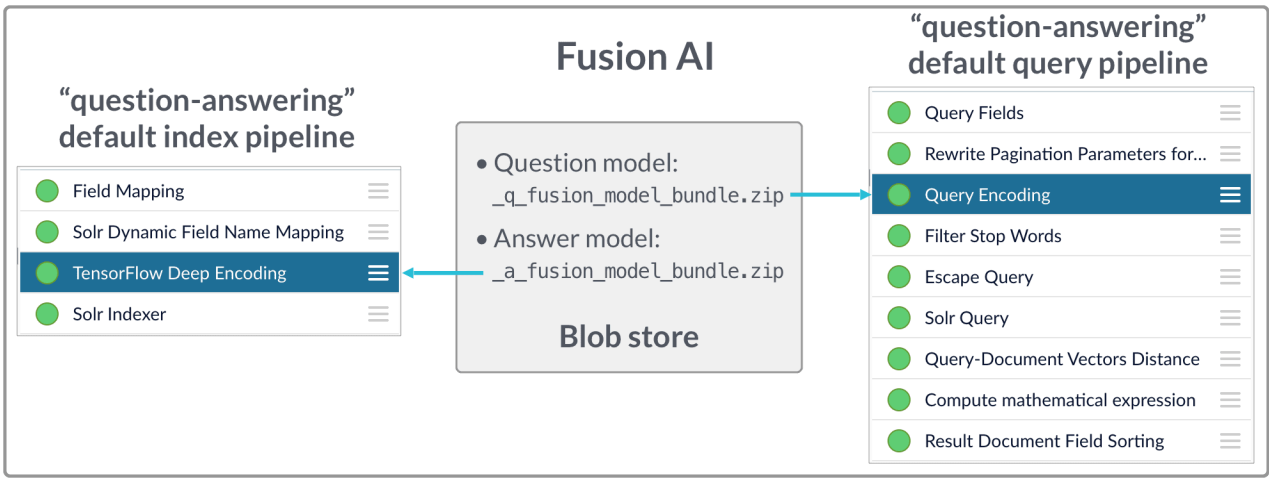
Model Deployment (Fusion 4.2 only)

How-to

> Compatible versions: 4.2
>
> Works with: *Fusion AI*

The Smart Answers deployment procedure for the FAQ solution and the cold start solution is the same:

1. *Upload the question and answer models* to the blob store.
2. *Configure the* `question-answering` *index pipeline* so that it uses the answer model.
3. *Configure the* `question-answering` *query pipeline* so that it uses the question model.



See also *Best Practices*.

How to upload the models to the blob store

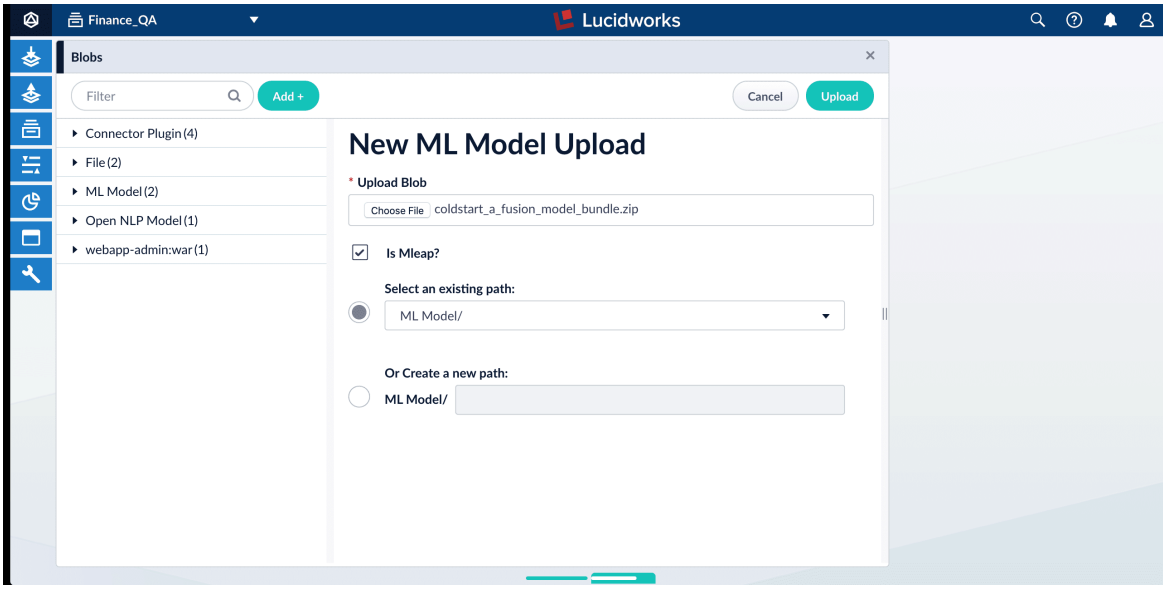
Two model files are generated by the *model training* Docker job:

- `x_a_fusion_model_bundle.zip` - The answer model.
- `x_q_fusion_model_bundle.zip` - The question model.

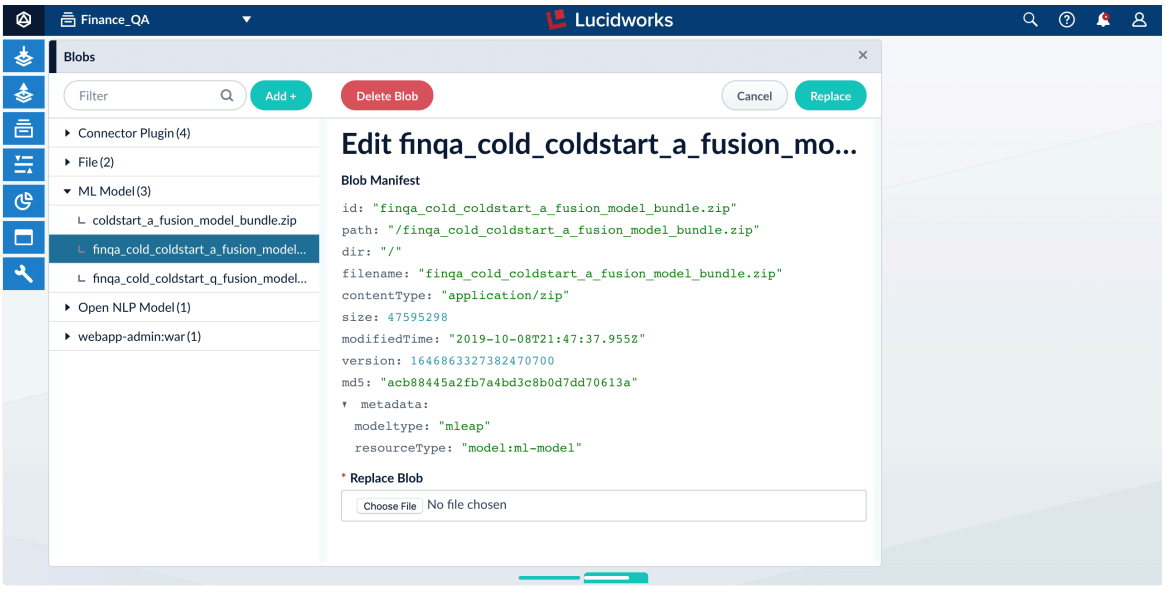
Both files must be deployed to the Fusion *blob store*.

How to upload the deep-learning models to the blob store

1. In the Fusion UI, navigate to **System** > **Blobs**.
2. Click **Add** > **ML Model**.
3. Click **Choose File** and select one of the model files.
4. Make sure the **Is Mleap?** checkbox is selected.



5. Click **Upload**.



6. Repeat steps 2-5 for the other model file.

Fusion configuration overview

- Fusion index pipeline
  The index pipeline uses the trained model in `x_a_fusion_model_bundle.zip` to generate dense vectors for the documents to be indexed.
- Fusion query pipeline
  The query pipeline uses the trained model in `x_q_fusion_model_bundle.zip` to generate dense vectors for incoming questions on the fly, then compare those with the indexed dense vectors for answers to find answers, or with the indexed dense vectors for historical questions to find similar questions.

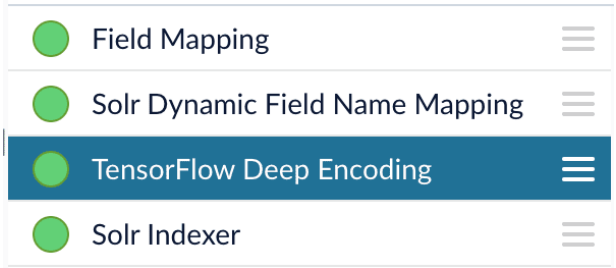
Another option is to have two separate index stages, one for questions and another for answers. Then, at query time, two query stages compute query-to-question distance and query-to-answer distance. Both scores are ensembled into a final similarity score. The two options are illustrated in *Pipeline setup examples*.

How to configure the default pipelines

If you have an AI license, then the following default index and query pipelines are included in any newly-created Fusion app:

Default index pipelines		Default query pipelines	
`question-answering`	For encoding one field.	`question-answering`	Calculates vectors distances between an encoded query and one document vector field. Should be used together with `question-answering` index pipeline.
`question-answering-dual-fields`	For encoding two fields (question and answer pairs, for example).	`question-answering-dual-fields`	Calculates vectors distances between an encoded query and two document vector fields. After that, scores are ensembled. Should be used together with the `question-answering-dual-fields` index pipeline.
See *Configure the index pipeline* below.		See *Configure the query pipeline* below.	

Configure the index pipeline



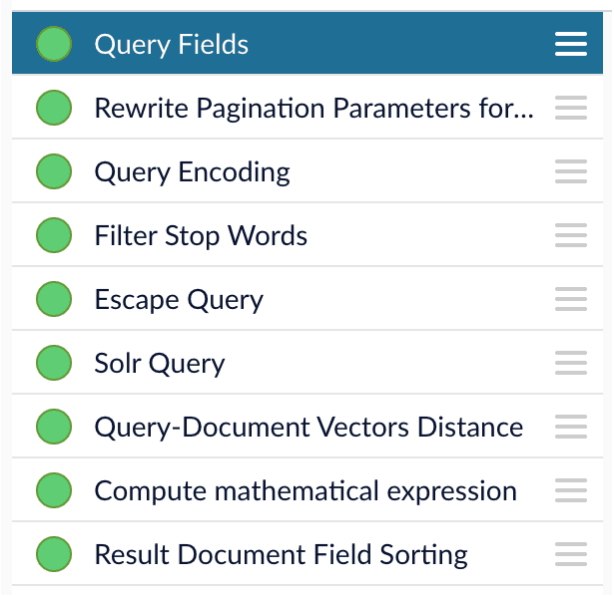
1. Open the Index Workbench.
2. Load or create your datasource using the default question-answering index pipeline.
3. In the *TensorFlow Deep Encoding stage*, change the value of **TensorFlow Deep Learning Encoder Model ID** to the model ID of the `x_a_fusion_model_bundle.zip` model that was uploaded to the blob store.

   Tip

   The `a` (answer) model allows you to encode longer text.

4. Change **Document Feature Field** to the document field name to be processed and encoded into dense vectors.
5. Save the datasource.
6. Index your data.

Configure the query pipeline



1. Open the Query Workbench.
2. Load or create your datasource using one of the default question-answering query pipelines.
3. In the *Query Fields stage*, update **Return Fields** to return additional fields that should be displayed with each answer, such as fields corresponding to title, text, or ID.
   It is recommended that you remove the asterisk (`*`) field and specify each individual field you want to return, as returning too many fields will affect runtime performance.
   Do not remove `compressed_document_vector_s` , `document_clusters_ss` , and `score` as these fields are necessary for later stages
4. In the *TensorFlow Deep Encoding stage*, change **TensorFlow Deep Learning Encoder Model ID** value to the model ID of the `x_q_fusion_model_bundle.zip` model that was uploaded to the blob store.

   Tip

   The (`q`) (question) model is slightly more efficient for short natural-language questions.

5. Save the query pipeline.

Pipeline Setup Examples

- *Example 1: Index and retrieve question and answer separately*
- *Example 2: Index and retrieve question and answer together*

Example 1: Index and retrieve question and answer separately

Based on your search Web page design, you can put best-matched questions and answers in separate sections, or if you only want to retrieve answers and serve to chatbot app, please index them separately in different documents.

For example, in the picture below, we construct the input file for the index pipeline such that the text part of the question/answer is stored in `text_t` , and we add an additional field `type_s` whose value is "question" or "answer" to separate the two types.

In the TensorFlow Deep Encoding stage, we specify **Document Feature Field** as `text_t` so that `compressed_document_vector_s` is generated based on this field.

At search time, we can apply a filter query on the `type_s` field to return either a question or an answer.

You can achieve a similar result by using the default `question-answering` index and query pipelines.

Example 2: Index and retrieve question and answer together.

If you prefer to show question and answer together in one document (that is, treat the question as the title and the answer as the description), you can index them together in the same document. It's similar to the `question-answering-dual-fields` index and query pipelines default setup.

For example, in the picture below, we added two TensorFlow Deep Encoding stages and named them Answers Encoding and Questions Encoding respectively. In the Questions Encoding stage, we specify **Document Feature Field** to be question_t, and changed the default values for **Vector Field**, **Clusters Field** and **Distances Field** to question_vector_ds, question_clusters_ss and question_distances_ds respectively. In the Answers Encoding stage, we specify **Document Feature Field** to be answer_t, and changed the default values for **Vector Field**, **Clusters Field** and **Distances Field** to answer_vector_ds, answer_clusters_ss and answer_distances_ds respectively. (Detailed information of the above field setup please refer to the "Appendix C: Detailed Pipeline Setup" section.)

Since we have two dense vectors generated in the index (compressed_question_vector_s and compressed_answer_vector_s), at query time, we need to compute query to question distance and query to answer distance. This can be setup as the picture shown below. We added two **Vectors distance per Query/Document** stages and named them QQ Distance and QA Distance respectively. In the QQ Distance stage, we changed the default values for **Document Vector Field** and **Document Vectors Distance Field** to compressed_question_vector_s and qq_distance respectively. In the QA Distance stage, we changed the default values for **Document Vector Field** and **Document Vectors Distance Field** to compressed_answer_vector_s and qa_distance respectively. (Detailed information of the above field setup please refer to the "Appendix C: Detailed Pipeline Setup" section.)

Now we have two distances (query to question distance and query to answer distance), we can ensemble them together with Solr score to get a final ranking score. This is recommended especially when you have limited FAQ dataset and want to utilize both question and answer information. This ensemble can be done in the **Compute mathematical expression** stage as shown below.

Query pipeline evaluation

Stages and configurations of the query pipeline will impact ranking results. For example, we may need to make decisions about whether to use clustering option in the *TensorFlow Deep Encoding stage*, what should be the weights of each ranking score in the **Compute mathematical expression** stage, whether adding other Fusion stages (such as text tagger, ML stage) would help. Comparing two different query pipelines with different models and setups may be needed as well.

The **Query Pipeline evaluation** job in the training docker image can help evaluate the rankings of Fusion returned results from different pipeline setups, additionally help search for the best set of weights in the ensemble score.

In order to evaluate the rankings, we need to know the ground truth answers for some testing questions. For example, user can provide around 200 testing questions, and for each question, we need to know which indexed answers are the correct answers in Fusion, so that we can compute positions of correct answers in the current ranking. User needs to provide a file in CSV format which contains the testing questions (**Testing query field** parameter in the job) and correct answer text or id (**Ground truth field** parameter in the job), put this file in the mapped input folder where we put other training data files, then specify the file name in the **Evaluation file name** parameter. NOTE: if there are multiple matching answers for a question, please put question and answer pairs in different rows. In Fusion, there should be a returned field which contains values that match the answer text or id, please specify this field in the **Answer or id field in Fusion** parameter. NOTE: if use answer text, please make sure the formatting of answer text in the evaluation file is the same as the answer text in Fusion so that we can find matches. Several Fusion access parameters are also needed in order to grab results from Fusion in batch, such as login username, password, host ip, app name, collection and pipeline name.

The job will provide a variety of metrics (controlled by **Metrics list *parameter) at different positions (controlled by *Metrics@k list** parameter) in the logs for the chosen final ranking score (specified in **Ranking score** parameter). For example, if choose **Ranking score** as "ensemble_score", the program will rank results by the ensemble score returned in **Compute mathematical expression** stage. If choose **Ranking score** as "score" (default field name for Solr score in Fusion), then the ranking evaluation will be based only on Solr score. In addition to the metrics in logs, a CSV results evaluation file will be generated. It will provide correct answer positions for each testing question as well as top returned results for each field specified in **Return fields** parameter. User can specify the file name for this results file in the **Ranking results file name** parameter, then download the file from `http://<external hostname>:5550/evaluation`.

Another function of this job is to help choose weights for different ranking scores such as Solr score, query to question distance, query to answer distance in **Compute mathematical expression** stage. If interested in performing this weights selection, please choose **Whether perform weights selection *parameter to true, list the set of score names in *List of ranking scores for ensemble** parameter. Since we can use different scaling methods for Solr score in the stage, please choose which Solr scale function you used in the stage in **Solr scale function** parameter. **Target metric to use for weight selection** parameter allows to specify metric that should be optimized during weights selection, for example `recall@3`. Metric values at different positions for different weights combinations will be shown in the log, sorted descendingly based on metric specified above. NOTE: Weights

selection can take a while to run for big evaluation datasets, thus if only interested in comparing pipelines, please turn this function off by specifying **Whether perform weights selection** parameter to false.

There are a few additional advanced parameters that might be useful but not required to provide. **Additional query parameters** allows to provide extra query parameters like `rowsFromSolrToRerank` in a dictionary format. **Sampling proportion** and **Sampling seed** provides a possibility to run evaluation job only on a sample of data.

Tip: Please make sure the Ranking score, answer text or id, and list of ranking scores for ensemble (if perform weights selection) are returned by Fusion by checking the fields returned in the Query Workbench. User can setup the returned fields in Return Fields section of Query Fields stage.

Fetch Azure Groups with the AD Connector for ACLs

How-to

> Works with: *Fusion Connectors*

In Fusion 4.2.6 and later 4.x.x versions, the *Active Directory Connectors for ACLs* has the ability to fetch Azure groups and index them into the ACL collection. The connector utilizes the Microsoft Graph API's group-list service .

The resulting ACL document contains IDs that are GUID strings, such as `45b7d2e7-b882-4a80-ba97-10b7a63b8fa4`, and they have `inbound_ss` relationship to the `ldapGroup-sid` record's SID identifier.

Configuration parameters

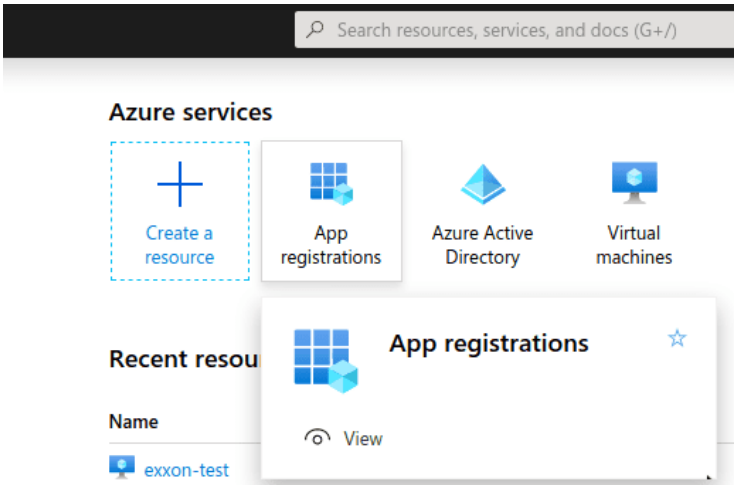
In order to crawl Azure groups from the AD Connector for ACLs, you will need the values for the following parameters:

- Azure AD Tenant ID
- Azure AD Client ID
- Azure AD Client Secret

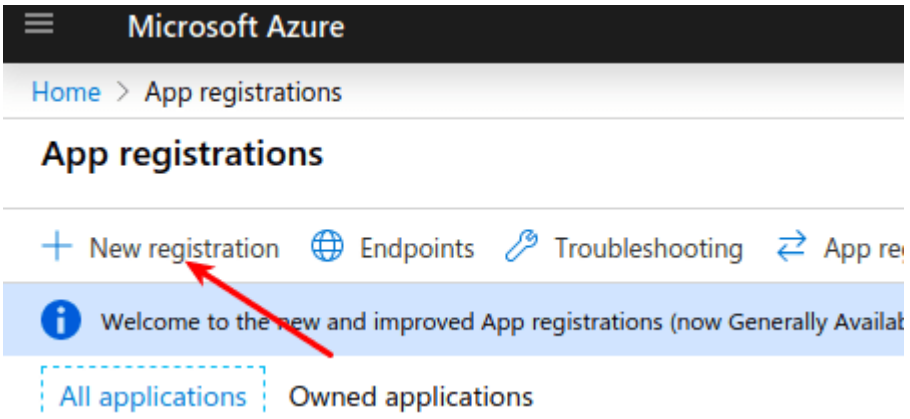
Find your Tenant ID and Client ID

Begin by registering your application:

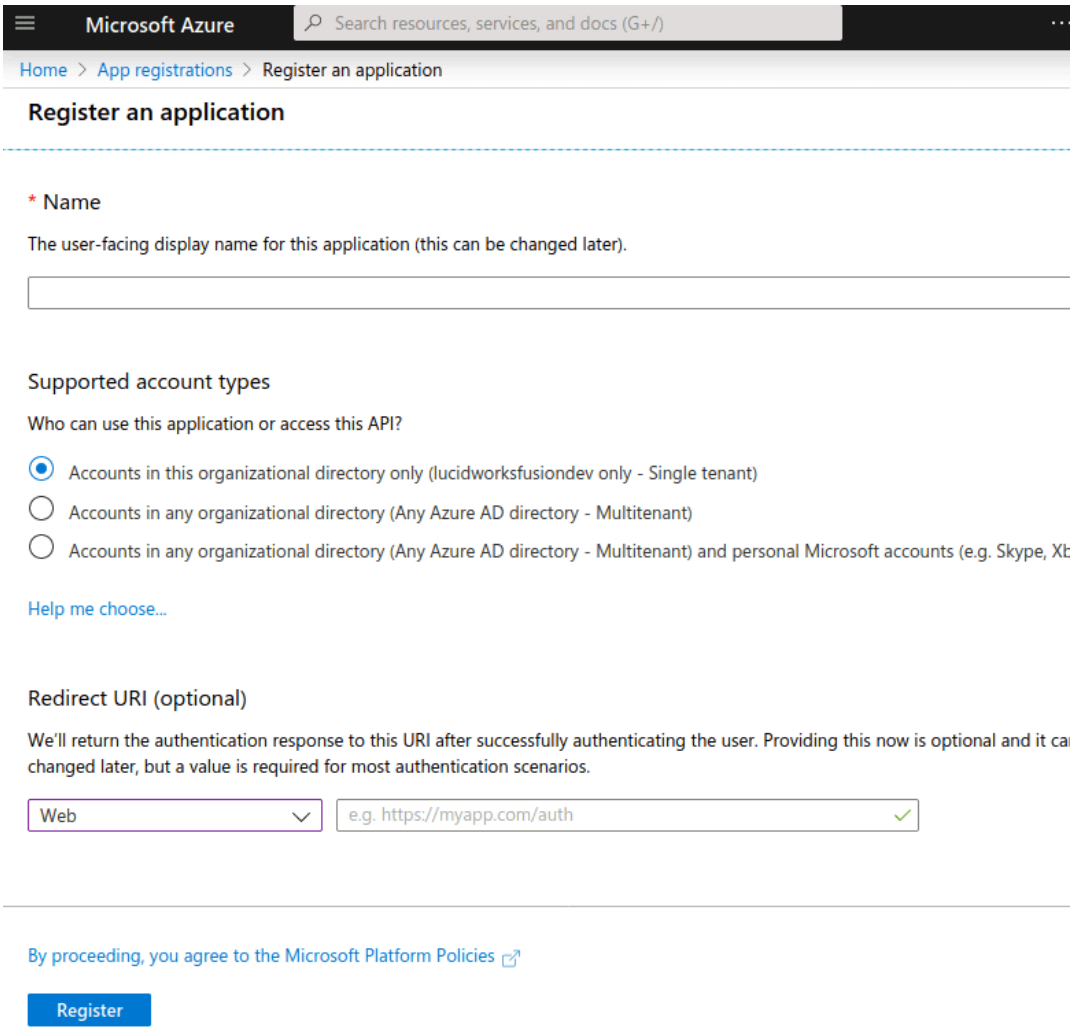
1. Visit the Azure portal .
2. Click **App registrations**.



3. Click **New registration**.



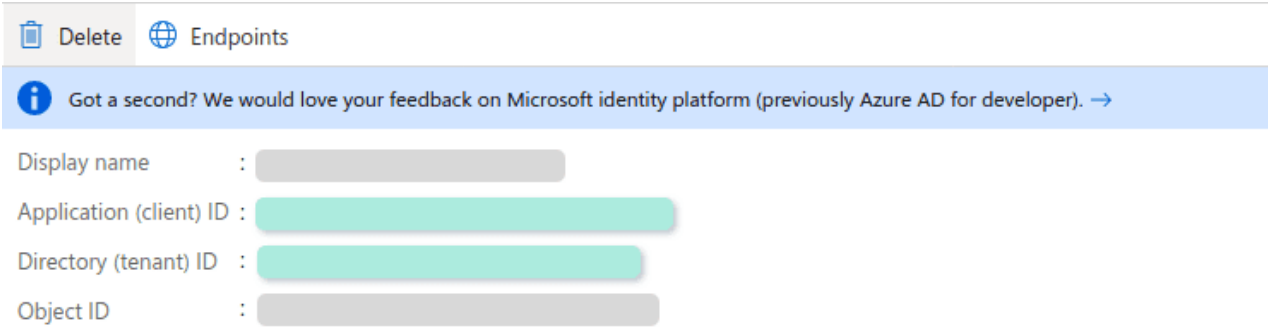
The new application registration screen will appear:



4. Enter a name for the application.
5. Choose the *Single Tenant* supported account type.
6. Leave the *Redirect URI* value blank.
7. Click **Register**.

The screen that follows displays the values for:

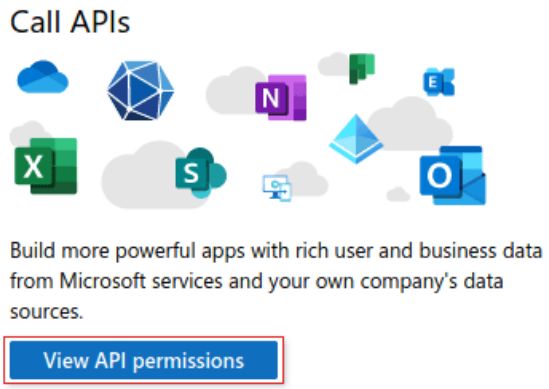
- **Azure AD Tentant ID.** Listed as "Directory (tenant) ID"
- **Azure AD Client ID.** Listed as "Application (client) ID"



Enter these values in the Fusion UI's connector configuration.

Configure your application permissions

1. Click **View API permissions**.

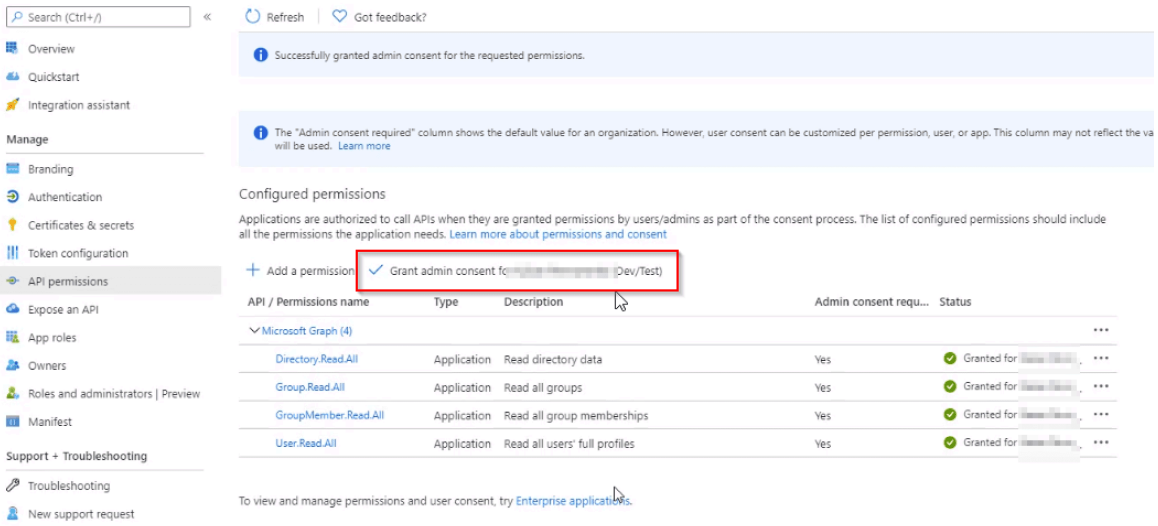


2. Add the following as *Application permissions* under **Microsoft Graph**.

> **Warning**
>
> You must use ***Application*** *permissions*. Failure to use *Application permissions* will result in `403` errors from the Graph API when attempting to crawl the Azure groups. Do not use the ***Delegated*** *permissions* option.

   a. `GroupMember.Read.All`
   Allows the application to read general information about associated groups, such as the list of members and basic group properties. See Microsoft Graph Group Application permissions for more information.
   b. `User.Read.All`
   Allows the application to read all user profile information, such as properties and group membership. See Microsoft Graph User Application permissions for more information.

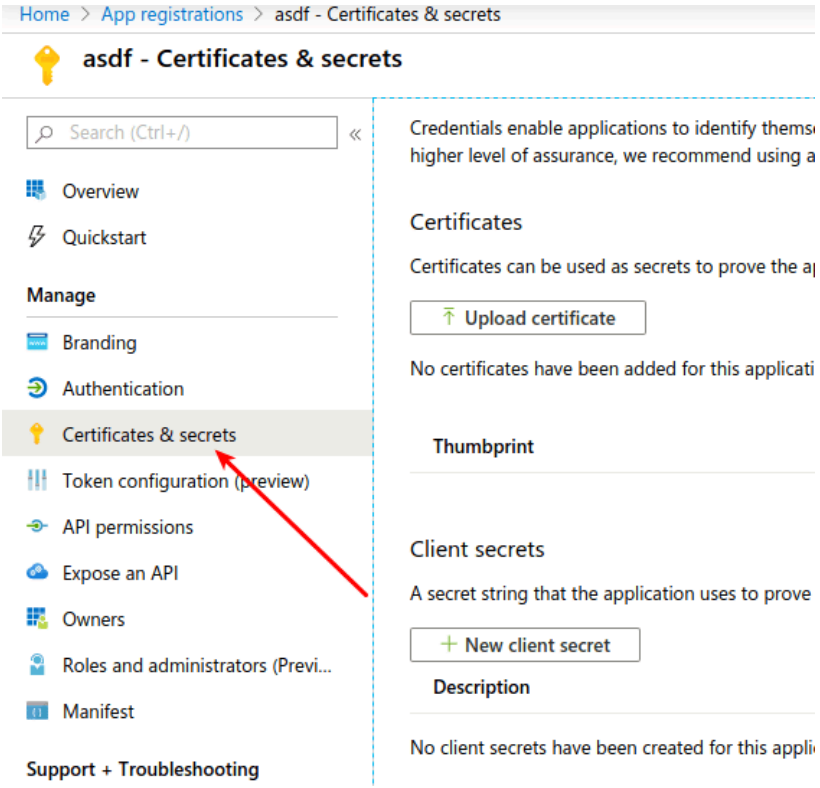
3. Select **Grant admin consent**.



Find your Client Secret

1. Click **Clients & Secrets**.

2. Create a **New client secret**.



This value is used as the **Azure AD Client Secret** in the Fusion UI's connector configuration.

More information

- *AD Connector for ACLs Concepts*
- *AD Connector for ACLs Configuration Reference*

Troubleshooting

Use the following script to diagnose Azure credentials issues:

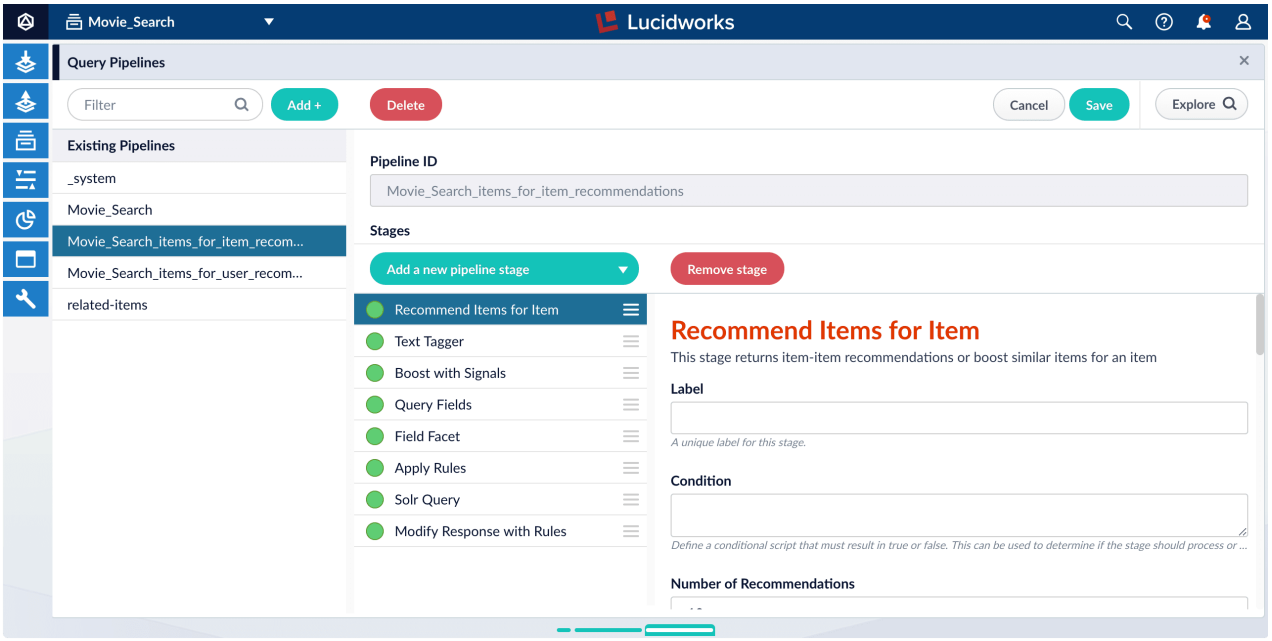
```
param ( [Parameter(Mandatory)]$AzureTenantId,
[Parameter(Mandatory)]$AzureClientId,
[Parameter(Mandatory)]$AzureClientSecret ) # first we get the bearer
token $headers = New-Object
"System.Collections.Generic.Dictionary[[String],[String]]"
$headers.Add("Content-Type", "application/x-www-form-urlencoded")
$body =
"client_id=${AzureClientId}&client_secret=${AzureClientSecret}&scope=h
ttps%3A%2F%2Fgraph.microsoft.com%2F.default&grant_type=client_credenti
als" $response = Invoke-RestMethod
"https://login.microsoftonline.com/${AzureTenantId}/oauth2/v2.0/token"
-Method 'POST' -Headers $headers -Body $body $access_token =
$response.access_token Write-Host "Successfully got access token
${access_token}" # now that we got the token, use it to get the groups
$headers2 = New-Object
"System.Collections.Generic.Dictionary[[String],[String]]"
$headers2.Add("Authorization", "Bearer ${access_token}") $response2 =
Invoke-RestMethod 'https://graph.microsoft.com/v1.0/groups' -Method
'GET' -Headers $headers2 $response2 | ConvertTo-Json
```

Fetch Items-for-Item Recommendations (Collaborative/ALS Method)

How-to

> Compatible versions: 4.2
>
> Works with: *Fusion AI*, *Fusion Server*, *Fusion*

If you have enabled signals and recommendations for a collection, then the `_items_for_item_recommendations` query pipeline is created by default and configured to fetch items-for-item recommendations using the output from the *ALS Recommender job*. It is similar to the default query pipeline that fetches content from your main collection, but it has an additional *Recommend Items for Item* stage:



This pipeline is a template that you can use two different ways:

- *Use only the Recommend Items for Item stage*
  This method returns only the document IDs of the recommended items. Your search application must perform additional queries to retrieve the desired fields for those items, such as their names, images, categories, and so on.
- *Query Solr directly*
  This method returns complete documents about the recommended items from the `_items_for_item_recommendations` collection. The fields that are included in these recommender documents are configured in the recommender job's `itemMetadataJoinField` /**Item Metadata Join Field** and `itemMetadataFields` /**Item Metadata Fields** fields; see *Item metadata settings*.

Note

In Fusion 5.2, the BPR method is recommended and the ALS method is deprecated. See *Fetch Items-for-Item Recommendations (Collaborative/BPR Method)*. You can also *Fetch Items-for-Item Recommendations (Content-Based Method)*.

Fetching recommendations from App Studio

App Studio can only access user-created collections; it cannot access system collections such as the default collections that Fusion creates for recommendations. If you are using App Studio to create your front-end search application, you must:

- Create a new collection for item-for-item recommendations
- Configure the recommender job to send output to the user-created collection instead of the system collection
- Direct your queries to the user-created collection

Once this is done, you can fetch recommendations as usual, using either of the methods explained below.

Fetching with the Recommend Items for Item stage

With this method, we use only one query pipeline stage: the *Recommend Items for Item* query stage. This stage works with the output from the *ALS Recommender* job.

This method returns only the document IDs of the recommended items. Your search application must perform additional queries to retrieve the desired fields for those items, such as their names, images, categories, and so on.

How to fetch recommendations using the Recommend Items for Item stage

1. From your `_items_for_item_recommendations` collection, navigate to **Querying** > **Query Workbench**.

   Tip

   If this collection is empty, you must run one of the *items-for-item recommender jobs*.

2. Click **Load...** and open the `_items_for_item_recommendations` query pipeline, if it is not open already.
3. Disable all of the pipeline stages *except* Recommend Items for Item.
4. Verify that the following fields are correctly configured in the Recommend Items for Item query stage:
   - `numRecommendations` **/Number of Recommendations**
     This is the number of recommendations to return. It should be less than or equal to the value of the `numSims` **/Number of Item Similarities to Compute** parameter in the *model tuning parameters* of the `COLLECTION_NAME_item_recommendations` job configuration.
   - `modelID` **/Model ID**
     This must match the `modelId` **/Recommender Model ID** value in the recommender job's *model settings*.
   - `collection` **/Recommendation Collection**
     This should be the collection specified in the `outputItemSimCollection` **/Item-to-item Similarity Collection** parameter of the recommender job's *Input/output settings*.
   - `resultsLocation` **/Results Location**
     Select the **As Response** value for this field.
   - There are several fields that specify the names of fields that should be present in the documents in the `_items_for_item_recommendations` collection. Verify that these values match the field names in that collection.
   - To use live signals in addition to historical signals, enable **Estimate Recent Results**.

   Be sure to click **Apply** after changing any of the stage configuration parameters.

   Note

   At this point, the results panel should display "No Search Results". This is normal; we will see results at a later step.

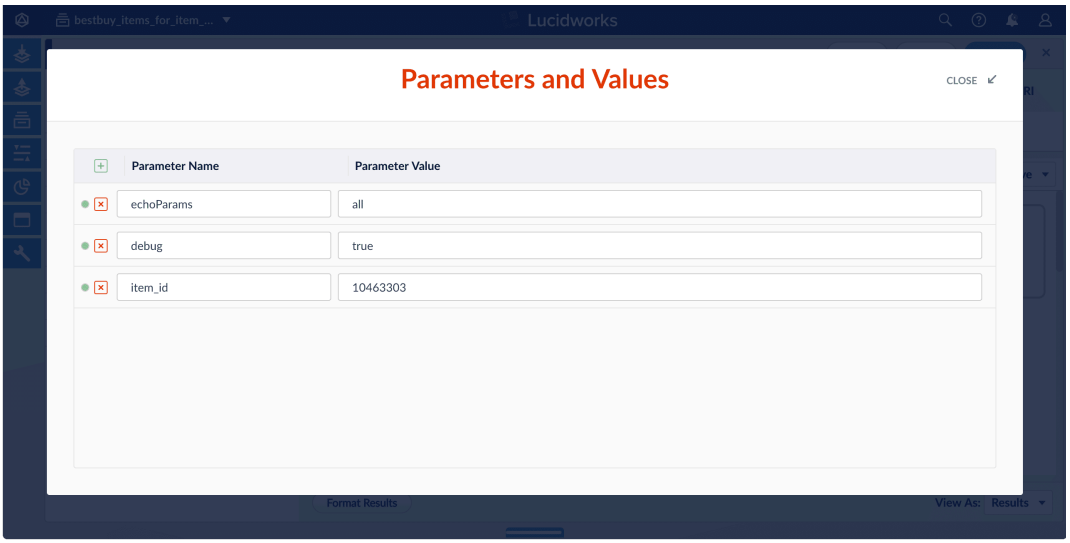
5. Click **Save**.

   Tip

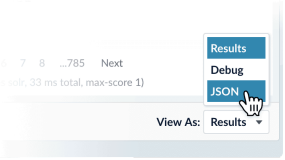
   Save the modified pipeline as a new pipeline with a different name, to distinguish it from the default pipeline.

6. Test the pipeline configuration:

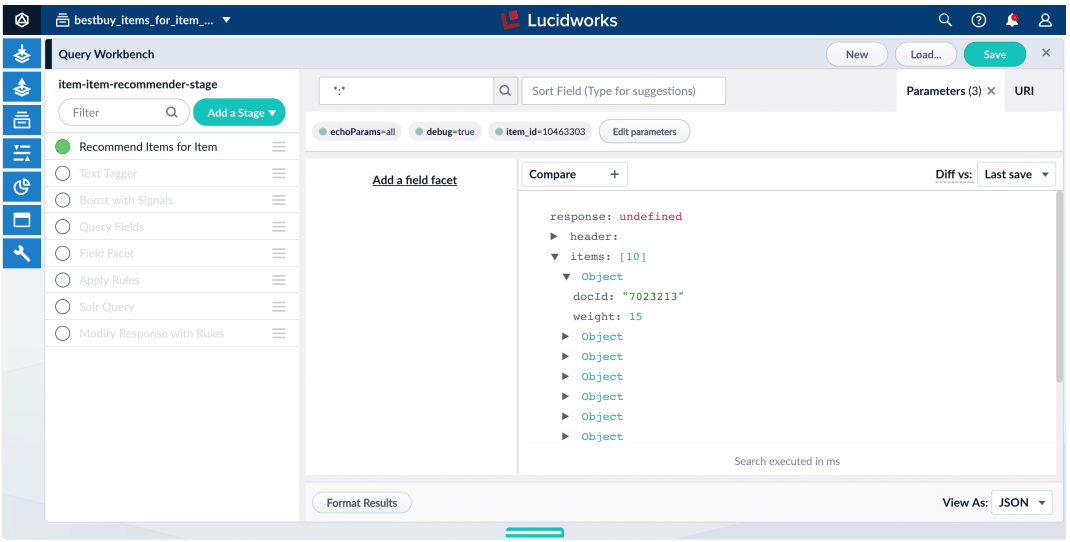
1. Select an item from your main collection to use for testing.
    a. Navigate to your main collection and open the Query Workbench.
    b. Search for an item and copy the value of its `id` field.
    c. Return to the `_items_for_item_recommendations` collection and the Query Workbench.
2. Click **Parameters**.
3. Click **Edit Parameters**.
   The **Parameters and Values** window appears.
4. Click the Add icon.
5. Enter the parameter name `item_id` and the value that you copied from the `id` field of the item in your main collection.



6. Click **Close** to close the **Parameters and Values** window.
7. In the lower right, select **View As: JSON**:



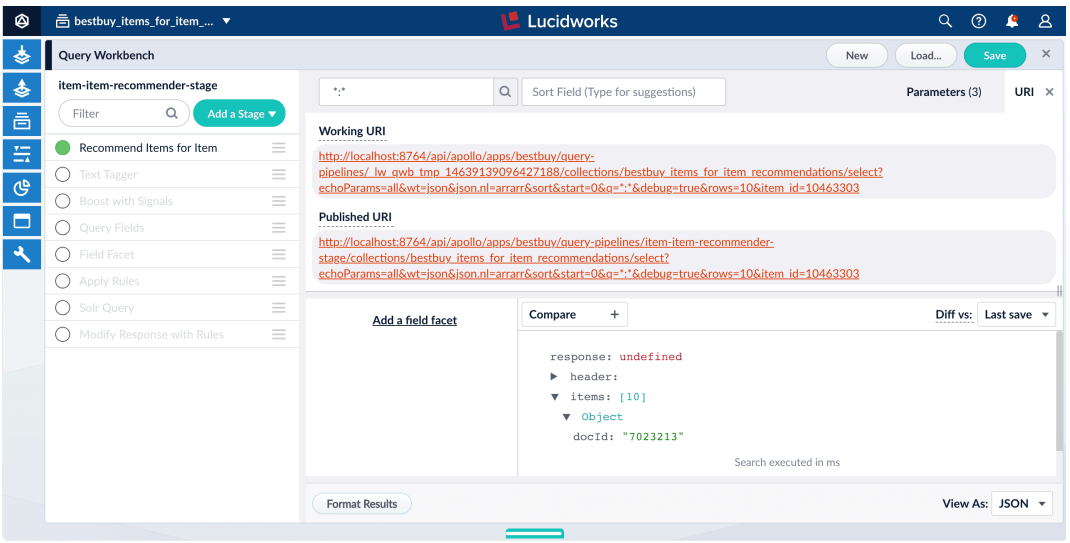
Notice that the search results contain only a document ID and a weight, as you can see when you expand the `items` field and any of its `Object` fields:



Tip

If the `items` field is empty, choose another item from your main collection and update the `item_id` query parameter to match it. Some items in your main collection may have no recommendations.

7. Get the query URI that your search application can use to retrieve recommendations from your modified pipeline:
    a. Click **URI**.
    The Query Workbench displays a **Working URI** and a **Published URI**.



    b. Click the **Published URI** to copy it to your clipboard.
    This is the URI and parameters that your search application should use to query for additional recommendations.

Tips:

- Each time you query for recommendations, replace the `item_id` parameter value with the ID of the item for which you want recommendations.
- Replace the Fusion hostname as needed, depending on your production environment.

- As shown above, Fusion returns an array of document IDs and weights. Your search application must then query the main collection to retrieve the details about each of the recommended documents. That is, for each `docId` value returned from the recommendations collection, query for the corresponding `id` value in the main collection, and order the set of results according to the `weight` value from the recommendations collection.

Querying Solr directly

With this method, we use only the *Solr Query* pipeline stage.

This method returns only the document IDs of the recommended items. Your search application must perform additional queries to retrieve the desired fields for those items, such as their names, images, categories, and so on.

How to fetch recommendations using the Solr Query stage

1. From your `_items_for_item_recommendations` collection, navigate to **Querying** > **Query Workbench**.
2. Click **Load**.
3. Select the `_items_for_item_recommendations` query pipeline.
4. Disable all stages in the pipeline *except* the Solr Query stage.
5. Click **Save**.

> **Tip**
>
> Save the modified pipeline as a new pipeline with a different name, to distinguish it from the default pipeline.

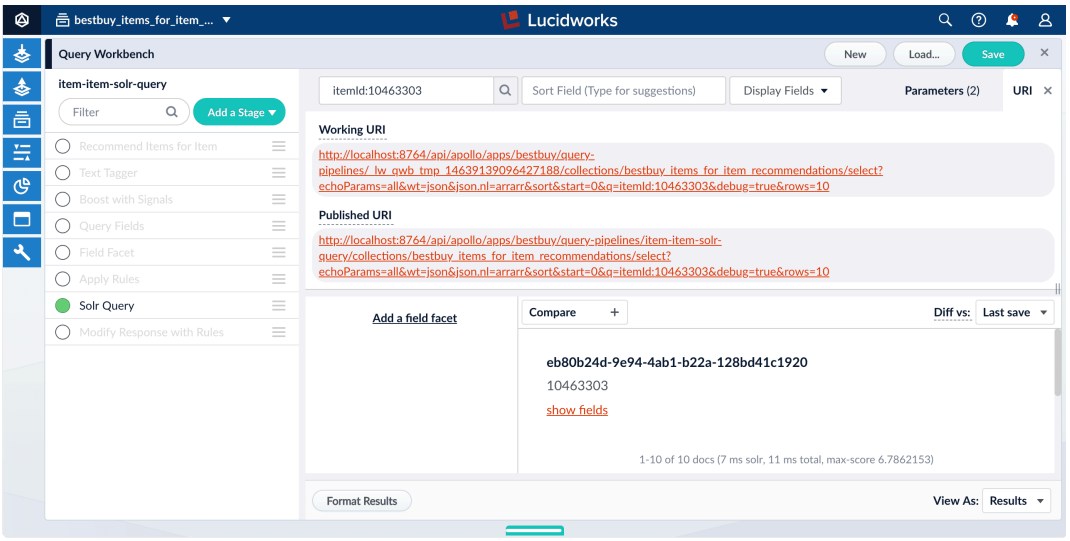
6. Test the pipeline configuration:
    1. Select an item from your main collection to use for testing.
        a. Navigate to your main collection and open the Query Workbench.
        b. Search for an item and copy the value of its `id` field.
        c. Return to the `_items_for_item_recommendations` collection and the Query Workbench.
    2. In the search field, enter the parameter name `itemId` and the value that you copied from the `id` field of the item in your main collection.
       For example, enter `itemId:10463303` (substituting the value from your main collection) and click the Search button.
       The number of results should be the same as the `numRecs` /**Number of User Recommendations to Compute** value in the `_item_recommendations` job (the default is 10).

       > **Tip**
       >
       > If the results panel displays "No Search Results", choose another item from your main collection and update the `item_id` query parameter to match it. Some items in your main collection may have no recommendations.

    3. In any of the search results, click **show fields**.
       Notice these important fields:
        - `itemId` is the original item to which the recommendation pertains.
        - `otherItemId` is the recommended item.
        - `sim` is the similarity score, or the estimated likelihood that the recommended item is related to the original item.

7. Get the query URI that your search application can use to retrieve recommendations from your modified pipeline:

a. Click **URI**.
The Query Workbench displays a **Working URI** and a **Published URI**.



b. Click the **Published URI** to copy it to your clipboard.
This is the URI and parameters that your search application should use to query for additional recommendations.

Tips:

- Each time you query for recommendations, replace the `itemId` parameter value with the ID of the item for which you want recommendations.
- Replace the Fusion hostname as needed, depending on your production environment.
- As shown above, Fusion returns a set of results that each include an `otherItemId` and a `sim` score. Your search application must then query the main collection to retrieve the details about each of the recommended documents.
  That is, for each `otherItemId` value returned from the recommendations collection, query for the corresponding `id` value in the main collection, and order the set of results according to the `sim` value from the recommendations collection.

Tip

You can eliminate these additional queries by configuring the join field and item metadata fields in the `_item_recommendations` job's *Item metadata settings*. This copies the specified metadata fields from the main collection into the recommendations collection so that they can be retrieved in a single query. For example, if you specify `name`, `url`, `category`, `image`, or similar fields in the job configuration, then those are returned in the recommender results without the need for additional queries.

Fetch Items-for-Item Recommendations (Collaborative/BPR Method)

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion*

In Fusion 5.2 and later, you can download and import a query pipeline that works out of the box with the default *BPR Recommender* job.

There are two separate pipelines attached below which work in different ways to query and return recommendations.

Query for recommendations only

1. *Download the* `APPName_item_item_rec_pipelines_bpr.json` *file.*
2. Rename the file to replace `APPName` with the name of your Fusion app, such as `ProductCatalog_item_item_rec_pipelines_bpr.json`.
3. Open the JSON file, replace all instances of `APPName` with the name of your Fusion app, such as `ProductCatalog`, and save it.
4. Import the JSON file into your Fusion instance using the Query Pipelines REST API:

```
curl -u USERNAME:PASSWORD <fusion_proxy>/api/query-pipelines -
XPOST -H 'content-type:application/json' -
d@<path/to/filename.json>
```

5. In the Fusion UI, navigate to **Query** > **Query Pipelines** to verify that the new pipeline is available.

> **Note**
>
> This pipeline should be used to query the collection where the recommendations are stored. It makes a query against the `itemId` field and only returns the recommended `itemId` values. To get the actual items, you need to make a second query to the respective catalog collection with the returned `itemId` values.

No additional configuration is needed to use this pipeline with the default BPR job configuration.

Boost recommended products from catalog

1. *Download the* `APPName_items_for_item_bpr_boost.json` *file*
2. Rename the file to replace `APPName` with the name of your Fusion app, such as `ProductCatalog_item_item_rec_pipelines_bpr.json`.
3. Open the JSON file, replace all instances of `APPName` with the name of your Fusion app, such as `ProductCatalog`.
4. Fill in the `collection` name field in the first `Recommend Items for Item` stage and save the file.
5. Import the JSON file into your Fusion instance using the Query Pipelines REST API:

```
curl -u USERNAME:PASSWORD <fusion_proxy>/api/query-pipelines -
XPOST -H 'content-type:application/json' -
d@<path/to/filename.json>
```

6. In the Fusion UI, navigate to **Query** > **Query Pipelines** to verify that the new pipeline is available.

Note

This pipeline queries the recommendations collection and then makes a subsequent query to the actual catalog collection boosting the recommended items and returning the actual items from the catalog. This pipeline will therefore also return recommendations even if none were generated/available.

Note

This pipeline expects a request parameter called `id=<itemId>` to be appended to the request in order to work. An example query URL to this pipeline would look like `https://example-recs-fusion.com/api/query-pipelines/APPName_items_for_item_bpr_boost/collections/catalog/select?q=:&id=SomeItemId`

Tip

If the pipeline does not appear in the **Query Pipelines** panel, you may need to attach it to your app like this: Go to **System** > **Object Explorer**, click the **In No Apps** filter, hover over the pipeline, click the icon, and select **Add to this app**.

No additional configuration is needed to use this pipeline with the default BPR job configuration.

See also

- *Fetch Content-Based Items-for-Item Recommendations*
- *Items-For-Item Recommendations*
- *Getting Started with Recommendations and Boosting*

Fetch Items-for-Item Recommendations (Collaborative/BPR Method)

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

In Managed Fusion, you can download and import a query pipeline that works out of the box with the default *BPR Recommender* job.

There are two separate pipelines attached below which work in different ways to query and return recommendations.

Query for recommendations only

1. *Download the* `APPName_item_item_rec_pipelines_bpr.json` *file.*
2. Rename the file to replace `APPName` with the name of your Managed Fusion app, such as `ProductCatalog_item_item_rec_pipelines_bpr.json`.
3. Open the JSON file, replace all instances of `APPName` with the name of your Managed Fusion app, such as `ProductCatalog`, and save it.
4. Import the JSON file into your Managed Fusion instance using the Query Pipelines REST API:

```
curl -u USERNAME:PASSWORD
<https://EXAMPLE_COMPANY.b.lucidworks.cloud/api/query-pipelines
-XPOST -H 'content-type:application/json' -
d@<path/to/filename.json>
```

5. In the Managed Fusion UI, navigate to **Query** > **Query Pipelines** to verify that the new pipeline is available.

Note

This pipeline should be used to query the collection where the recommendations are stored. It makes a query against the `itemId` field and only returns the recommended `itemId` values. To get the actual items, you need to make a second query to the respective catalog collection with the returned `itemId` values.

No additional configuration is needed to use this pipeline with the default BPR job configuration.

Boost recommended products from catalog

1. *Download the* `APPName_items_for_item_bpr_boost.json` *file*
2. Rename the file to replace `APPName` with the name of your Managed Fusion app, such as `ProductCatalog_item_item_rec_pipelines_bpr.json`.
3. Open the JSON file, replace all instances of `APPName` with the name of your Managed Fusion app, such as `ProductCatalog`.
4. Fill in the `collection` name field in the first `Recommend Items for Item` stage and save the file.
5. Import the JSON file into your Managed Fusion instance using the Query Pipelines REST API:

```
curl -u USERNAME:PASSWORD
https://EXAMPLE_COMPANY.b.lucidworks.cloud/api/query-pipelines -
XPOST -H 'content-type:application/json' -
d@<path/to/filename.json>
```

6. In the Managed Fusion UI, navigate to **Query** > **Query Pipelines** to verify that the new pipeline is available.

Note

This pipeline queries the recommendations collection and then makes a subsequent query to the actual catalog collection boosting the recommended items and returning the actual items from the catalog. This pipeline will therefore also return recommendations even if none were generated/available.

Note

This pipeline expects a request parameter called `id=<itemId>` to be appended to the request in order to work. An example query URL to this pipeline would look like `https://EXAMPLE_COMPANY.b.lucidworks.cloud/api/query-pipelines/APPName_items_for_item_bpr_boost/collections/catalog/select?q=:&id=SomeItemId`

Tip

If the pipeline does not appear in the **Query Pipelines** panel, you may need to attach it to your app like this: Go to **System** > **Object Explorer**, click the **In No Apps** filter, hover over the pipeline, click the icon, and select **Add to this app**.

No additional configuration is needed to use this pipeline with the default BPR job configuration.

See also

- *Fetch Content-Based Items-for-Item Recommendations*
- *Items-For-Item Recommendations*
- *Get Started with Recommendations and Boosting*

Fetch Items-for-Item Recommendations (Content-Based Method)

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion*

In Fusion 5.2 and later, you can download and import a query pipeline that works out of the box to fetch the items-for-item recommendations generated by the default *Content-Based Recommender* job.

There are two separate pipelines attached below which work in different ways to query and return recommendations.

Query for recommendations only

1. *Download the* `APPName_item_item_rec_pipelines_content.json` *file.*
2. Rename the file to replace `APPName` with the name of your Fusion app, such as `ProductCatalog_item_item_rec_pipelines_bpr.json`.
3. Open the JSON file, replace all instances of `APPName` with the name of your Fusion app, such as `ProductCatalog`, and save it.
4. Import the JSON file into your Fusion instance using the Query Pipelines REST API:

```
curl -u USERNAME:PASSWORD <fusion_proxy>/api/query-pipelines -
XPOST -H 'content-type:application/json' -
d@<path/to/filename.json>
```

5. In the Fusion UI, navigate to **Query** > **Query Pipelines** to verify that the new pipeline is available.

Note

This pipeline should be used to query the collection where the recommendations are stored. It makes a query against the `itemId` field and returns the recommended `itemId` values. To get the actual items, you need to make a second query to the respective catalog collection with the returned `itemId` values.

No additional configuration is needed to use this pipeline with the default BPR job configuration.

Boost recommended products from catalog

1. *Download the* `APPName_content_boost.json` *file*
2. Rename the file to replace `APPName` with the name of your Fusion app, such as `ProductCatalog_item_item_rec_pipelines_bpr.json`.
3. Open the JSON file, replace all instances of `APPName` with the name of your Fusion app, such as `ProductCatalog`.
4. Fill in the `collection` name field in the first `Recommend Items for Item` stage and save the file.
5. Import the JSON file into your Fusion instance using the Query Pipelines REST API:

```
curl -u USERNAME:PASSWORD <fusion_proxy>/api/query-pipelines -
XPOST -H 'content-type:application/json' -
d@<path/to/filename.json>
```

6. In the Fusion UI, navigate to **Query** > **Query Pipelines** to verify that the new pipeline is available.

Note

This pipeline queries the recommendations collection and then makes a subsequent query to the actual catalog collection boosting the recommended items and returning the actual items from the catalog. This pipeline will therefore also return recommendations even if none were generated/available.

Note

This pipeline expects a request parameter called `id=itemId` to be appended to the request in order to work. An example query URL to this pipeline would look like `https://example-recs-fusion.com/api/query-pipelines/APPName_items_for_user_bpr_boost/collections/catalog/select?q=:&id=SomeItemId`

Tip

If the pipeline does not appear in the **Query Pipelines** panel, you may need to attach it to your app like this: Go to **System** > **Object Explorer**, click the **In No Apps** filter, hover over the pipeline, click the icon, and select **Add to this app**.

See also

- *Fetch Items-for-Item Recommendations (Collaborative/BPR Method)*
- *Items-For-Item Recommendations*
- *Content-Based Recommender job*
- *Getting Started with Recommendations and Boosting*

Fetch Items-for-Item Recommendations (Content-Based Method)

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

You can download and import a query pipeline that works out of the box to fetch the items-for-item recommendations generated by the default *Content-Based Recommender* job.

There are two separate pipelines attached below which work in different ways to query and return recommendations.

Query for recommendations only

1. *Download the* `APPName_item_item_rec_pipelines_content.json` *file.*
2. Rename the file to replace `APPName` with the name of your Managed Fusion app, such as `ProductCatalog_item_item_rec_pipelines_bpr.json`.
3. Open the JSON file, replace all instances of `APPName` with the name of your Managed Fusion app, such as `ProductCatalog`, and save it.
4. Import the JSON file into your Managed Fusion instance using the Query Pipelines REST API:

```
curl –u USERNAME:PASSWORD
https://EXAMPLE_COMPANY.b.lucidworks.cloud/api/query–pipelines –
XPOST –H 'content–type:application/json' –
d@<path/to/filename.json>
```

5. In the Managed Fusion UI, navigate to **Query** > **Query Pipelines** to verify that the new pipeline is available.

Note

This pipeline should be used to query the collection where the recommendations are stored. It makes a query against the `itemId` field and returns the recommended `itemId` values. To get the actual items, you need to make a second query to the respective catalog collection with the returned `itemId` values.

No additional configuration is needed to use this pipeline with the default BPR job configuration.

Boost recommended products from catalog

1. *Download the* `APPName_content_boost.json` *file*
2. Rename the file to replace `APPName` with the name of your Managed Fusion app, such as `ProductCatalog_item_item_rec_pipelines_bpr.json`.
3. Open the JSON file, replace all instances of `APPName` with the name of your Managed Fusion app, such as `ProductCatalog`.
4. Fill in the `collection` name field in the first `Recommend Items for Item` stage and save the file.
5. Import the JSON file into your Managed Fusion instance using the Query Pipelines REST API:

```
curl –u USERNAME:PASSWORD
https://EXAMPLE_COMPANY.b.lucidworks.cloud/api/query–pipelines –
XPOST –H 'content–type:application/json' –
d@<path/to/filename.json>
```

6. In the Managed Fusion UI, navigate to **Query** > **Query Pipelines** to verify that the new pipeline is available.

Note

This pipeline queries the recommendations collection and then makes a subsequent query to the actual catalog collection boosting the recommended items and returning the actual items from the catalog. This pipeline will therefore also return recommendations even if none were generated/available.

Note

This pipeline expects a request parameter called `id=itemId` to be appended to the request. An example query URL to this pipeline would look like `https://EXAMPLE_COMPANY.b.lucidworks.cloud/api/query-pipelines/APPName_items_for_user_bpr_boost/collections/catalog/select?q=:&id=SomeItemId`

Tip

If the pipeline does not appear in the **Query Pipelines** panel, you may need to attach it to your app like this: Go to **System** > **Object Explorer**, click the **In No Apps** filter, hover over the pipeline, click the icon, and select **Add to this app**.

See also

- *Fetch Items-for-Item Recommendations (Collaborative/BPR Method)*
- *Items-For-Item Recommendations*
- *Content-Based Recommender job*
- *Get Started with Recommendations and Boosting*

Fetch Items-for-User Recommendations (ALS Method)

How-to

> Compatible versions: 4.2
>
> Works with: *Fusion AI*, *Fusion Server*, *Fusion*

For items-for-user recommendations generated by the ALS recommender job, a `_items_for_user_recommendations` query pipeline is created by default and configured to fetch items-for-user recommendations. It is similar to the default query pipeline that fetches content from your main collection, but it has an additional *Recommend Items for User* stage.

This pipeline is a template that you can use two different ways:

- *Use only the Recommend Items for User stage*
  This method returns the set of recommended items for the specified user.
- *Query Solr directly*
  This method returns complete documents about the recommended items from the `_items_for_user_recommendations` collection. The fields that are included in these recommender documents are configured in the recommender job's `itemMetadataJoinField` /**Item Metadata Join Field** and `itemMetadataFields` /**Item Metadata Fields** fields; see *Item metadata settings* above.

Fetching recommendations from App Studio

App Studio can only access user-created collections; it cannot access system collections such as the default collections that Fusion creates for recommendations. If you are using App Studio to create your front-end search application, you must:

- Create a new collection for item-for-item recommendations
- Configure the recommender job to send output to the user-created collection instead of the system collection
- Direct your queries to the user-created collection

Once this is done, you can fetch recommendations as usual, using either of the methods explained below.

Note

In Fusion 5.2, the BPR method is recommended and the ALS method is deprecated. See *Fetch Items-for-User Recommendations (Collaborative/BPR Method)*.

Fetching with the Recommend Items for User stage

With this method, we use only one query pipeline stage: the *Recommend Items for User* query stage.

This method returns only the document IDs of the recommended items. Your search application must perform additional queries to retrieve the desired fields for those items, such as their names, images, categories, and so on.

How to fetch recommendations using the Recommend Items for User stage

1. From your `_items_for_user_recommendations` collection, navigate to **Querying** > **Query Workbench**.

   Tip

   If this collection is empty, you must run one of the *items-for-user recommender jobs*.

2. Click **Load...** and open the `_items_for_user_recommendations` query pipeline, if it is not open already.
3. Disable all of the pipeline stages *except* Recommend Items for User.
4. Verify that the following fields are correctly configured in the Recommend Items for User query stage:

- `numRecommendations` /**Number of Recommendations**
  This is the number of recommendations to return. It should be less than or equal to the value of the `numSims` /**Number of Item Similarities to Compute** parameter in the *model tuning parameters* of the `COLLECTION_NAME_item_recommendations` job configuration.
- `modelID` /**Model ID**
  This must match the `modelId` /**Recommender Model ID** value in the recommender job's *model settings*.
- `collection` /**Recommendation Collection**
  This should be the collection specified in the `outputItemSimCollection` /**Item-to-item Similarity Collection** parameter of the recommender job's *Input/output settings*.
- `resultsLocation` /**Results Location**
  Select the **As Response** value for this field.
- There are several fields that specify the names of fields that should be present in the documents in the `_items_for_user_recommendations` collection. Verify that these values match the field names in that collection.
- To use live signals in addition to historical signals, enable **Estimate Recent Results**.

Be sure to click **Apply** after changing any of the stage configuration parameters.

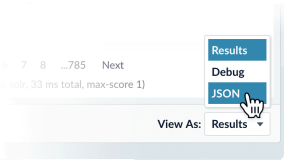
5. Click **Save**.

> **Tip**
>
> Save the modified pipeline as a new pipeline with a different name, to distinguish it from the default pipeline.

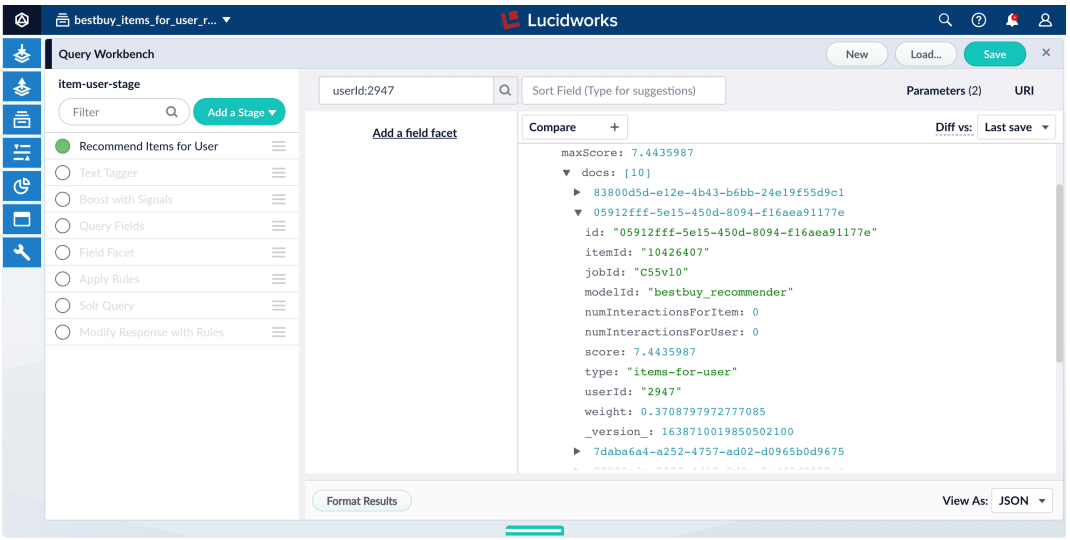
6. Test the pipeline configuration:

1. Select a user ID for testing by clicking **show fields** on any result in the `_items_for_user_recommendations` collection and copying the value of its `userId` field.
2. In the search field, enter the parameter name `userId` and the value that you copied. For example, enter `userId:5049` (substituting the value from your collection) and click the Search button.

   The number of results should be the same as the `numRecs` **/Number of User Recommendations to Compute** value in the `_item_recommendations` job (the default is 10).
3. In the lower right, select **View As: JSON**:



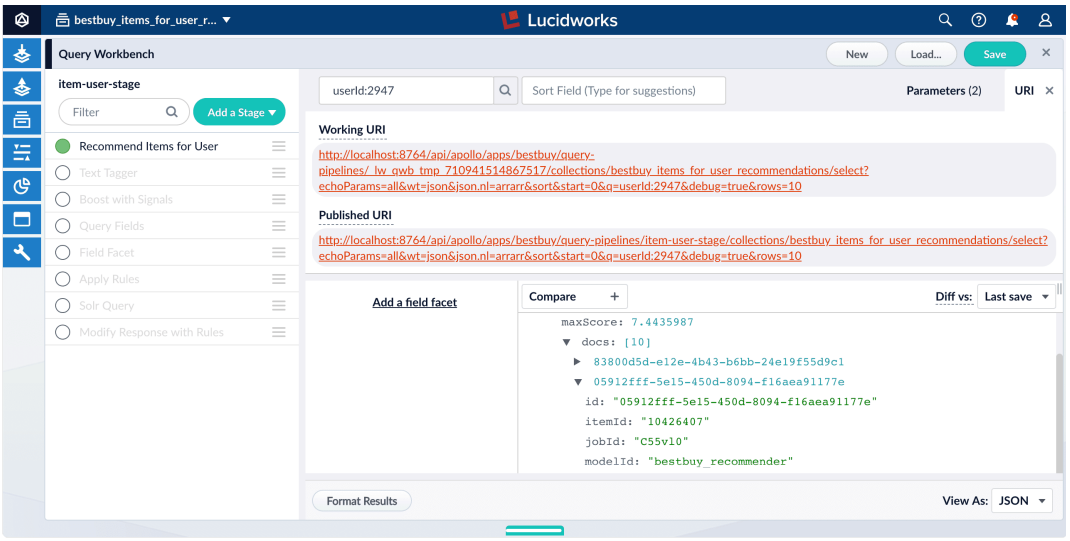
4. Expand the `items` field and any of its `Object` fields:



   Notice these important fields:
   - `userId` is the user to whom the recommendation pertains.
   - `itemId` is the recommended item.
   - `weight` is the estimated likelihood that the recommended item is of interest to the specified user.

7. Get the query URI that your search application can use to retrieve recommendations from your modified pipeline:

a. Click **URI**.
The Query Workbench displays a **Working URI** and a **Published URI**.



b. Click the **Published URI** to copy it to your clipboard.
This is the URI and parameters that your search application should use to query for additional recommendations.

Tips:

- Each time you query for recommendations, replace the `userId` parameter value with the ID of the user for whom you want recommendations.
- Replace the Fusion hostname as needed, depending on your production environment.
- As shown above, Fusion returns an array of item IDs and weights. Your search application must then query the main collection to retrieve the details about each of the recommended items.
That is, for each `itemId` value returned from the recommendations collection, query for the corresponding `id` value in the main collection, and order the set of results according to the `weight` value from the recommendations collection.

Tip

You can eliminate these additional queries by configuring the join field and item metadata fields in the `_item_recommendations` job's *Item metadata settings*. This copies the specified metadata fields from the main collection into the recommendations collection so that they can be retrieved in a single query. For example, if you specify `name`, `url`, `category`, `image`, or similar fields in the job configuration, then those are returned in the recommender results without the need for additional queries.

Querying Solr directly

With this method, we use only the *Solr Query* pipeline stage.

This method returns only the document IDs of the recommended items. Your search application must perform additional queries to retrieve the desired fields for those items, such as their names, images, categories, and so on.

How to fetch recommendations using the Solr Query stage

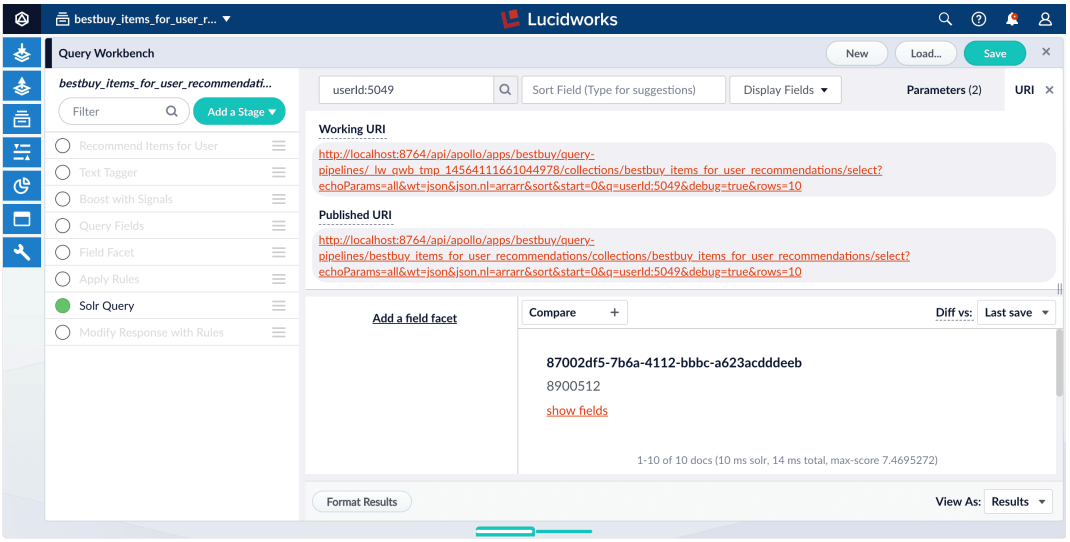
1. From your `_items_for_user_recommendations` collection, navigate to **Querying** > **Query Workbench**.
2. Click **Load**.
3. Select the `_items_for_user_recommendations` query pipeline.
4. Disable all stages in the pipeline *except* the Solr Query stage.

5. Click **Save**.

> **Tip**
>
> Save the modified pipeline as a new pipeline with a different name, to distinguish it from the default pipeline.

6. Test the pipeline configuration:
    1. Select a user ID for testing by clicking **show fields** on any result in the `_items_for_user_recommendations` collection and copying the value of its `userId` field.
    2. In the search field, enter the parameter name `userId` and the value that you copied. For example, enter `userId:5049` (substituting the value from your collection) and click the Search button.
       The number of results should be the same as the `numRecs` /**Number of User Recommendations to Compute** value in the `_item_recommendations` job (the default is 10).
    3. In any of the search results, click **show fields**.
       Notice these important fields:
        - `userId` is the user to whom the recommendation pertains.
        - `itemId` is the recommended item.
        - `weight` is the estimated likelihood that the recommended item is of interest to the specified user.

7. Get the query URI that your search application can use to retrieve recommendations from your modified pipeline:
    a. Click **URI**.
       The Query Workbench displays a **Working URI** and a **Published URI**.



    b. Click the **Published URI** to copy it to your clipboard.
       This is the URI and parameters that your search application should use to query for additional recommendations.

Tips:

- Each time you query for recommendations, replace the `userId` parameter value with the ID of the user for whom you want recommendations.
- Replace the Fusion hostname as needed, depending on your production environment.
- As shown above, Fusion returns a set of results that each include an `itemId` and a `weight`. Your search application must then query the main collection to retrieve the details about each of the recommended items.

That is, for each `itemId` value returned from the recommendations collection, query for the corresponding `id` value in the main collection, and order the set of results according to the `weight` value from the recommendations collection.

Tip

You can eliminate these additional queries by configuring the join field and item metadata fields in the `_item_recommendations` job's *Item metadata settings*. This copies the specified metadata fields from the main collection into the recommendations collection so that they can be retrieved in a single query. For example, if you specify `name`, `url`, `category`, `image`, or similar fields in the job configuration, then those are returned in the recommender results without the need for additional queries.

Fetch Items-for-User Recommendations (Collaborative/BPR Method)

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion*

In Fusion 5.2 and later, you can download and import a query pipeline that works out of the box to fetch items-for-user recommendations generated by the default *BPR Recommender* job.

There are two separate pipelines attached below which work in different ways to query and return recommendations.

Query for recommendations only

1. *Download the* `APPName_item_user_rec_pipelines_bpr.json` *file.*
2. Rename the file to replace `APPName` with the name of your Fusion app, such as `ProductCatalog_item_item_rec_pipelines_bpr.json`.
3. Open the JSON file, replace all instances of `APPName` with the name of your Fusion app, such as `ProductCatalog`, and save it.
4. Import the JSON file into your Fusion instance using the Query Pipelines REST API:

```
curl -u USERNAME:PASSWORD <fusion_proxy>/api/query-pipelines -
XPOST -H 'content-type:application/json' -
d@<path/to/filename.json>
```

5. In the Fusion UI, navigate to **Query** > **Query Pipelines** to verify that the new pipeline is available.

Note

This pipeline should be used to query the collection where the recommendations are stored. It makes a query against the `userId` field and only returns the recommended `itemId` values. To get the actual items, you need to make a second query to the respective catalog collection with the returned `itemId` values.

No additional configuration is needed to use this pipeline with the default BPR job configuration.

Boost recommended products from catalog

1. *Download the* `APPName_items_for_user_bpr_boost.json` *file*
2. Rename the file to replace `APPName` with the name of your Fusion app, such as `ProductCatalog_items_for_user_bpr_boost.json`.
3. Open the JSON file, replace all instances of `APPName` with the name of your Fusion app, such as `ProductCatalog`.
4. Fill in the `collection` name field in the first `Recommend Items for User` stage and save the file.
5. Import the JSON file into your Fusion instance using the Query Pipelines REST API:

```
curl -u USERNAME:PASSWORD <fusion_proxy>/api/query-pipelines -
XPOST -H 'content-type:application/json' -
d@<path/to/filename.json>
```

6. In the Fusion UI, navigate to **Query** > **Query Pipelines** to verify that the new pipeline is available.

Note

This pipeline queries the recommendations collection and then makes a subsequent query to the actual catalog collection boosting the recommended items and returning the actual items from the catalog. This pipeline will therefore also return recommendations even if none were generated/available.

Note

This pipeline expects a request parameter called `id=<userId>` to be appended to the request in order to work. An example query URL to this pipeline would look like `https://example-recs-fusion.com/api/query-pipelines/APPName_items_for_user_bpr_boost/collections/catalog/select?q=:&id=SomeUserId`

Tip

If the pipeline does not appear in the **Query Pipelines** panel, you may need to attach it to your app like this: Go to **System** > **Object Explorer**, click the **In No Apps** filter, hover over the pipeline, click the icon, and select **Add to this app**.

See also

- *Items-For-User Recommendations*
- *BPR Recommender job*
- *Getting Started with Recommendations and Boosting*

Fetch Items-for-User Recommendations (Collaborative/BPR Method)

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

You can download and import a query pipeline that works out of the box to fetch items-for-user recommendations generated by the default *BPR Recommender* job.

There are two separate pipelines attached below which work in different ways to query and return recommendations.

Query for recommendations only

1. *Download the* `APPName_item_user_rec_pipelines_bpr.json` *file.*
2. Rename the file to replace `APPName` with the name of your Fusion app, such as `ProductCatalog_item_item_rec_pipelines_bpr.json`.
3. Open the JSON file, replace all instances of `APPName` with the name of your Fusion app, such as `ProductCatalog`, and save it.
4. Import the JSON file into your Fusion instance using the Query Pipelines REST API:

```
curl -u USERNAME:PASSWORD <fusion_proxy>/api/query-pipelines -
XPOST -H 'content-type:application/json' -
d@<path/to/filename.json>
```

5. In the Fusion UI, navigate to **Query** > **Query Pipelines** to verify that the new pipeline is available.

Note

This pipeline should be used to query the collection where the recommendations are stored. It makes a query against the `userId` field and only returns the recommended `itemId` values. To get the actual items, you need to make a second query to the respective catalog collection with the returned `itemId` values.

No additional configuration is needed to use this pipeline with the default BPR job configuration.

Boost recommended products from catalog

1. *Download the* `APPName_items_for_user_bpr_boost.json` *file*
2. Rename the file to replace `APPName` with the name of your Fusion app, such as `ProductCatalog_items_for_user_bpr_boost.json`.
3. Open the JSON file, replace all instances of `APPName` with the name of your Fusion app, such as `ProductCatalog`.
4. Fill in the `collection` name field in the first `Recommend Items for User` stage and save the file.
5. Import the JSON file into your Fusion instance using the Query Pipelines REST API:

```
curl -u USERNAME:PASSWORD <fusion_proxy>/api/query-pipelines -
XPOST -H 'content-type:application/json' -
d@<path/to/filename.json>
```

6. In the Fusion UI, navigate to **Query** > **Query Pipelines** to verify that the new pipeline is available.

Note

This pipeline queries the recommendations collection and then makes a subsequent query to the actual catalog collection boosting the recommended items and returning the actual items from the catalog. This pipeline will therefore also return recommendations even if none were generated/available.

Note

This pipeline expects a request parameter called `id=<userId>` to be appended to the request in order to work. An example query URL to this pipeline would look like `https://example-recs-fusion.com/api/query-pipelines/APPName_items_for_user_bpr_boost/collections/catalog/select?q=:&id=SomeUserId`

Tip

If the pipeline does not appear in the **Query Pipelines** panel, you may need to attach it to your app like this: Go to **System** > **Object Explorer**, click the **In No Apps** filter, hover over the pipeline, click the icon, and select **Add to this app**.

See also

- *Items-For-User Recommendations*
- *BPR Recommender job*
- *Getting Started with Recommendations and Boosting*

Fetch Results

How-to

> Works with: *Never Null*

For any primary query to your search platform, you can perform a secondary query to Never Null and get results that enhance your own. This topic shows you how to construct and send a secondary query to Never Null's REST API using the Solr query language.

To use the API, you need these items from Lucidworks:

- Your short customer name
- Your query profile name
- Your credentials

Note

If your content was recently ingested for the first time, Lucidworks will inform you when the initial model training is complete and your Never Null data is ready to query.

Authentication

Before you can begin fetching Never Null results, you must log in to the service as shown below:

```
curl -i -H "content-type:application/json" -X POST -d
'{"username":"username","password":"password"}'
'https://name.b.lucidworks.com/api/session?realmName=native'
```

Never Null returns a cookie, as in the example below:

```
HTTP/1.1 201 Created Date: Mon, 03 May 2021 22:37:57 GMT Content-
Length: 0 Connection: keep-alive Cache-Control: no-cache, no-store,
max-age=0, must-revalidate Pragma: no-cache Expires: 0 X-Content-Type-
Options: nosniff Strict-Transport-Security: max-age=15724800;
includeSubDomains X-Frame-Options: SAMEORIGIN X-XSS-Protection: 1 ;
mode=block Referrer-Policy: no-referrer Set-Cookie:
id=eyJhbGciOiJSUzI1NiJ9.eyJzdWIiOiJhZG1pbiIsInBlcm1pc3Npb25zIjpbXSwic2
NvcGUiOlsib3BlbmlkIiwiZW1haWwiLCJwcm9maWxlIl0sImlzcyI6Imh0dHA6XC9cL3By
b3h5OjY3NjRcL29hdXRoMlwvZGVmYXVsdCIsInJlYWxtIjoibmF0aXZlIiwiZXhwIjoxNj
IwMDgzMjM2LCJ1c2VySWQiOiIwMzhkM2ZmYy1jNDJhLTQ5NGItYTY1ZS04ZGExNzFkMDc4
MGQiLCJpYXQiOjE2MjAwODE0MzYsInBlcm1pc3Npb25zX3ZzIjo0Mjk0OTY4MTA5LCJhdX
Rob3JpdGllcyI6WyJhZG1pbiJdfQ.e3ipLvtx0H4do4Ads5mD0ABV2sn6hB0QBZXxSH4b2
DxJe5sbvx3OWBhqw3AgXp7ilnNSlCuYfwG7VjhpDGpGanc68eihUkuNzRjIX1THZAaJK3Z
miebUH4ey7tFnavHp17foPtNGQTMAtui9svEdUiJAcP8YkFC1-hpIozaD2b-
EFQjCoCxfif3yOtyBIyfG80o9EZdRF7JSSSOdZ5w3wbOsFDKul0w61Zq_5kk_Z4lw7fhvJ
AneMst_gLTNscenj7daBbxQ0Ved0XteLD6Ob3HxWTA1R2HvkD1JETtZr6v4okXY30-
h5ES0aO3UnXadWfDaOe24HqNb3-e0bSnjGg; Path=/; Max-Age=1759;
Expires=Mon, 03 May 2021 23:07:16 GMT; HttpOnly
```

Query URL format

The request URL includes the short customer name and query profile name assigned to you by Lucidworks, along with a query string:

```
https://name.b.lucidworks.com/api/query/profileName?q=QUERY-STRING...
```

Response format

The response is a JSON response where each item includes these fields:

- Product ID
- Short product description
- Similarity score
- Additional objects that reflect the input parameters

Example request and response

As "bigbox", query a profile named 'bigbox_en' for the misspelled term "dvdplayr".

Request

```
curl -u USERNAME:PASSWORD
https://bigbox.b.lucidworks.cloud/api/query/bigbox_en?q=dvdplayr
```

Response


```
{ "response" : { "docs" : [ { "description_s" : "Dynex™ - Refurbished
9\" Portable DVD Player", "id" : "1559547", "score" : 0.685536 }, {
"description_s" : "Dynex™ - Refurbished 7\" Portable DVD Player", "id"
: "1559538", "score" : 0.659592 }, { ... } }
```

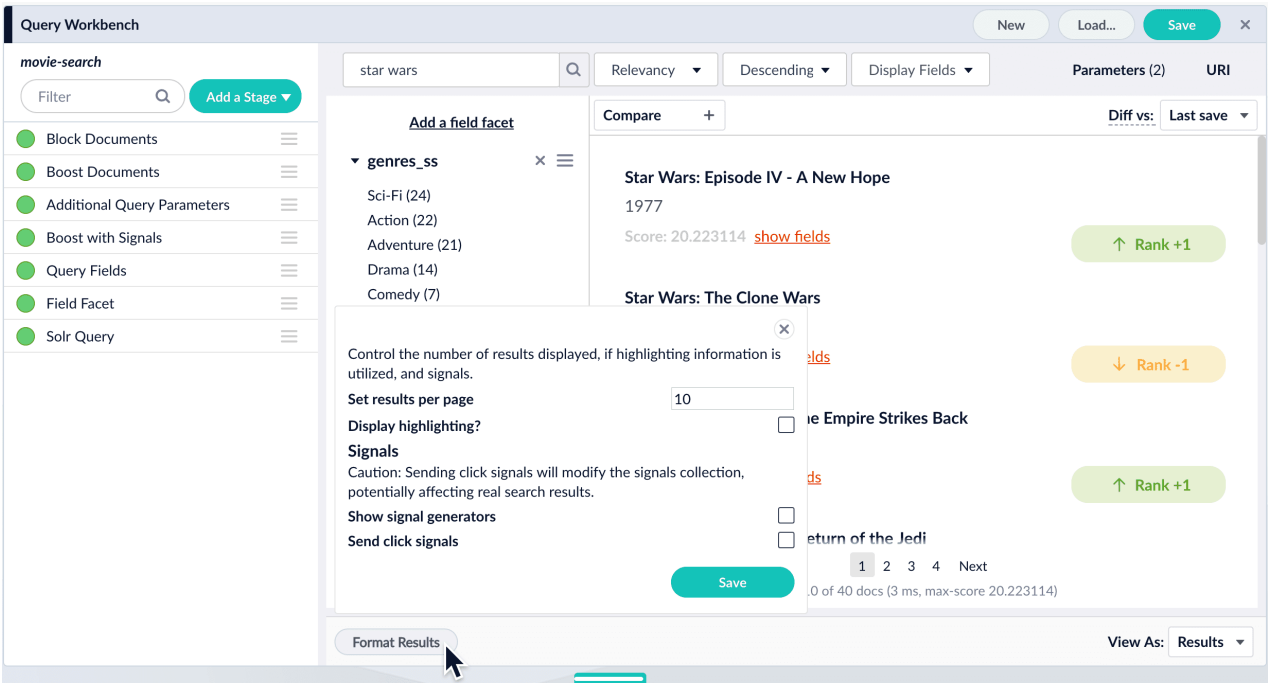
Format search results in the Query Workbench

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

Ultimately, your search application formats the search results that end users see. The Query Workbench provides some formatting options for the preview panel.

At the bottom of the screen, click **Format Results** to configure how results are displayed in the Query Workbench.



These options only affect how the Query Workbench displays results. They have no effect on how your search application displays them.

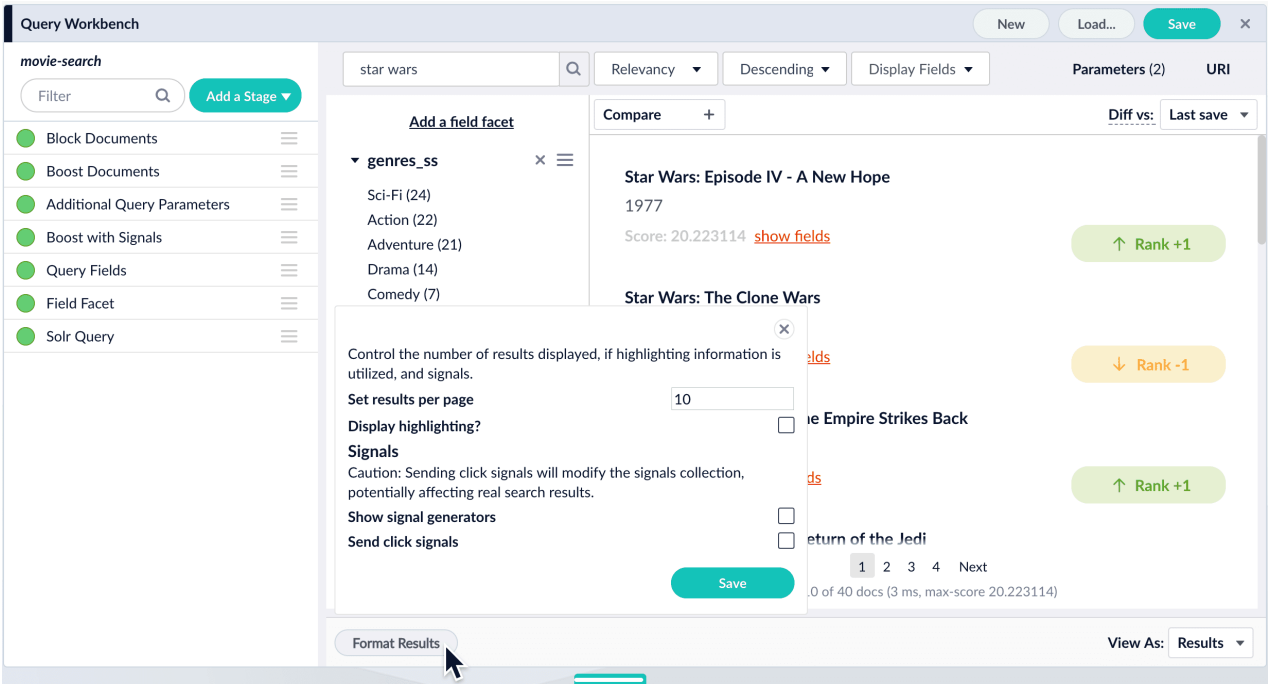
Format Search Results in the Query Workbench

How-to

> Compatible versions: 4.2, 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion Server*, *Fusion*

Ultimately, your search application will format the search results that end users see. The Query Workbench provides some formatting options for the preview panel.

At the bottom of the screen, click **Format Results** to configure how results are displayed in the QWB:



These options only affect how the Query Workbench displays results; they have no effect on how your search application displays them.

Fusion 4.x.x Zookeeper Ensemble Setup

How-to

> Compatible versions: 4.2
>
> Works with: *Fusion Server*

Configuration example

This example involves three servers:

- host1.devin3.ss.com
- host2.devin3.ss.com
- host3.devin3.ss.com

The sample location for `$FUSION` is `/opt/lw/fusion/4.1.0`. The actual path is based on the installation location you select.

Required files

ZOO.CFG

This file *must* be identical on all three servers. The location of the file is:

`$FUSION/conf/zookeeper/zoo.cfg`

1. Add the following after the **clientPort** entry:
   - **server.1=host1.devin3.ss.com:2888:3888**
   - **server.2=host2.devin3.ss.com:2888:3888**
   - **server.3=host3.devin3.ss.com:2888:3888**
2. Modify the **dataDir** entry to the following value:
   dataDir=/opt/lw/fusion/4.1.0/data/zookeeper
3. Copy this file, or make the same changes, to all three servers.

FUSION.PROPERTIES

This file *must* be identical on all three servers. The location of the file is:

`$FUSION/conf/fusion.properties`

1. Remove the comment from the **default.zk.connect** entry and enter the following value:
   default.zk.connect=host1.devin3.ss.com:9983,host2.devin3.ss.com:9983,host3.devin3.ss.com:9983
2. Modify the **group.default** entry to ensure the *only* value:
   group.default = zookeeper
3. Copy this file, or make the same changes, to all three servers.

MYID

The `myid` file designates a specific Zookeeper service by ID number.

1. Create a `myid` file on the following server with the specified path:
   `host1.devin3.ss.com: $FUSION/data/zookeeper/myid`

   Note

   Verify the only content in this myid file is the number **1**.

2. Create a `myid` file on the following server with the specified path:
   `host2.devin3.ss.com: $FUSION/data/zookeeper/myid`

   Note

   Verify the only content in this myid file is the number **2**.

3. Create a `myid` file on the following server with the specified path:
   `host3.devin3.ss.com: $FUSION/data/zookeeper/myid`

> **Note**
>
> Verify the only content in this myid file is the number **3**.

Verify the Zookeeper ensemble configuration

1. Access the host1.devin3.ss.com server and navigate to the `dist` installation folder.
2. Run the following command:
   `.fusion start`

   > **Note**
   >
   > Verify the only service that executes is Zookeeper (based on the **group.default** entry in the `fusion.properties` file.)

3. Access the host2.devin3.ss.com server and navigate to the `dist` installation folder.
4. Run the following command:
   `.fusion start`

   > **Note**
   >
   > Verify the only service that executes is Zookeeper (based on the **group.default** entry in the `fusion.properties` file.)

5. Access the host3.devin3.ss.com server and navigate to the `dist` installation folder.
6. Run the following command:
   `.fusion start`

   > **Note**
   >
   > Verify the only service that executes is Zookeeper (based on the **group.default** entry in the `fusion.properties` file.)

7. Enter the following command on the host1.devin3.ss.com server to verify the status of the Zookeeper service:
   `echo stat | nc host1.devin3.ss.com 9983`

   > **Important**
   >
   > The Zookeeper service must be running on all three servers to confirm the status of the service on any of the servers.

   The output of the command may contain a significant amount of information. Verify the **mode** value is **leader**. The **mode** value on the other two servers is **follower**.
8. Enter the following command on the host1.devin3.ss.com server to verify the status of the Zookeeper service:
   `echo stat | nc host2.devin3.ss.com 9983`
9. Enter the following command on the host1.devin3.ss.com server to verify the status of the Zookeeper service:
   `echo stat | nc host3.devin3.ss.com 9983`
10. Review the log files as needed. To review the log, enter:

```
tail -f /opt/lw/fusion/4.1.0/var/log/zookeeper/zookeeper.log
```

Fusion Impersonation via a Service Account

How-to

> Compatible versions: 4.2, 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *App Studio*

To securely access Fusion via Appkit, a user has several options. They can either adopt *Appkit's Fusion security provider* making use of session cookies to send repeated requests back to Fusion, or they can access Fusion via a service account if they are planning to use *query pipelines*. In this section, we describe how to set up the latter.

1 Set up a service account on the Fusion server

If a user wants to query Fusion through a pipeline by impersonating another user via a service account, they must first ensure such an account has been set up on the Fusion server. This account should have access rights to the required resources.

2 Add Security Trimming query stages to required pipelines

Query pipelines can have additional security filtering applied to ensure specific users do or do not have access to specific resources. This filtering can be set up by adding a Security Trimming stage to the query pipeline via the Fusion UI. The User ID key that is used in the security trimming stage to filter on results is supported in Appkit via the `user-id` attribute, which is described in the next section.

3 Update fusion.conf

To query Fusion using this approach, this attributes must be added to the platform `fusion.conf` file in `src/main/resources/conf/platforms/fusion/`:

```
impersonate: true userName: joebloggs password: password user-id:
username
```

Here, the `impersonate` attribute informs Appkit that users will be querying Fusion pipelines via a service account. Below that, both the `userName` and the `password` are the credentials for the service account that Fusion will authenticate against. The last attribute `user-id` is optional and by default takes the value of `username`. This is the parameter that will be appended to the query string and filtered on in the security trimming stage. For example, the complete query URL might appear as:

```
http://localhost:8764/api/apollo/query-pipelines/test-
default/collections/test/select?
&wt=json&q=*:*&debug=false&fl=x,y,z&start=0&username=joe@bloggs.com'
```

Fusion Query Terms and Suggestions

How-to

> Compatible versions: 4.2
>
> Works with: *App Studio*

Appkit supports the ability to generate suggestions as-you-type for successive query terms with Fusion via our *Solr suggestion services*.

Accessing the Suggest Component

To access the Solr suggester in a Fusion application, refer to the section *Suggester*.

For reference, here is an example of how the Appkit configuration might appear for this particular service:

```
name: twigkit.search.solr.suggestions.SolrSuggestionsParserService
source: platforms.fusion suggester: titleMatchSuggester parenthesis:
true title: Staff custom: count[5] doctype: howto
```

Here, the `source` defines the location of the fusion platform configuration. This will be used to identify the Fusion host and collection.

Accessing the Terms Component

To access the Solr Terms component in a Fusion application, refer to the section *Terms Component*.

For reference, here is an example of how the Appkit configuration might appear for this particular service:

```
name: twigkit.search.solr.suggestions.SolrQueryTermsService source:
platforms.fusion term-fields: Symbol, Project_name title: Suggested
phrase doctype: howto
```

Here, the `source` defines the location of the fusion platform configuration. This will be used to identify the Fusion host and subsequently a collection that will contain fields as listed in `term-fields`.

Ensure the terms request handler can be used with the Fusion query pipeline

The terms component and request handler are provided out-of-the-box in the `solrconfig.xml` with Fusion. However, to ensure the query pipeline you are using does allow the terms handler to be used with queries, check that the `Query Solr` stage of your query-pipeline does list `terms` as an allowed request handler.

Dynamic updates to multiple term queries

Because the terms component only allows suggestions one term at a time, in cases when multiple terms are being typed, suggestions will be given for the last term to be typed.

Related articles

- *Fusion Impersonation via a Service Account*

Get Logs for a Spark Job

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion*

See the table below for useful commands related to Spark jobs:

Description	Command
Retrieve the initial logs that contain information about the pod spin up.	`curl -X GET -u USERNAME:PASSWORD http://FUSION_HOST:FUSION_PORT/api/spark/driver/log/JOB_ID`
Retrieve the pod ID.	`k get pods -l spark-role=driver -l jobConfigId=JOB_ID`
Retrieve logs from failed jobs.	`kubectl logs DRIVER_POD_NAME`
Tail logs from running containers by using the `-f` parameter.	`kubectl logs -f POD_NAME`

Note

Spark deletes failed and successful executor pods. Fusion provides a cleanup Kubernetes cron job that removes successfully completed driver pods every 15 minutes.

Viewing the Spark UI

In the event that you need to monitor or inspect your Spark job executions, you can use port forwarding to access the Spark UI in your web browser. Port forwarding forwards your local port connection to the port of the pod that is running the Spark driver.

To view the Spark UI, find the pod that is running the Spark driver and run the following command:

```
kubectl -n namespace port-forward driver-pod 4040:4040
```

You can then access the Spark UI on `localhost:4040`

For related topics, see *Spark Operations*.

Get Logs for a Spark Job

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

See the table below for useful commands related to Spark jobs:

Description	Command
Retrieve the initial logs that contain information about the pod spin up.	`curl -X GET -u USERNAME:PASSWORD http://FUSION_HOST:FUSION_PORT/api/spark/driver/log/JOB_ID`
Retrieve the pod ID.	`k get pods -l spark-role=driver -l jobConfigId=JOB_ID`
Retrieve logs from failed jobs.	`kubectl logs DRIVER_POD_NAME`
Tail logs from running containers by using the `-f` parameter.	`kubectl logs -f POD_NAME`

Note

Spark deletes failed and successful executor pods. Fusion provides a cleanup Kubernetes cron job that removes successfully completed driver pods every 15 minutes.

Viewing the Spark UI

In the event that you need to monitor or inspect your Spark job executions, you can use port forwarding to access the Spark UI in your web browser. Port forwarding forwards your local port connection to the port of the pod that is running the Spark driver.

To view the Spark UI, find the pod that is running the Spark driver and run the following command:

```
kubectl -n namespace port-forward driver-pod 4040:4040
```

You can then access the Spark UI on `localhost:4040`

For related topics, see *Spark Operations*.

Get Started with App Studio

How-to

> Compatible versions: 4.2
>
> Works with: *App Studio*

App Studio 4.1 and later

You can get started with App Studio in just minutes when you have a *Fusion app* that includes some *indexed data*.

App Studio creates a single-page search application that you can further customize through the built-in markup editor.

To get started with App Studio, see *Create a New Web App*.

If you have created apps in prior versions of App Studio or with prior versions of Appkit, you can *upgrade those apps to use the latest version of Appkit*.

App Studio 4.0

You can get started with App Studio in just minutes by expanding the zip file, editing a few basic configuration parameters, and running the startup script. The application runs in an embedded Jetty servlet, by default on port 8080.

App Studio creates a single-page search application that you can further customize through the built-in markup editor.

These topics help you get started with App Studio:

- *Installation*
- *Start App Studio*
- *Create a New Web App*

Get the Status of App Studio

How-to

> Compatible versions: 4.2
>
> Works with: *App Studio*

App Studio has two scripts for startup and shutdown in the root directory of an App Studio project (`app-studio`):

- `app-studio.sh` . A Bash script for Linux and macOS
- `app-studio.bat` . A batch file for Windows

You also use these scripts to get the status of an App Studio server.

Linux and macOS:

```
./app-studio.sh status
```

Windows:

```
app-studio.sh status
```

Handle a Multi-Domain Active Directory

How-to

> Compatible versions: 4.2, 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion Connectors*, *Fusion Server*, *Fusion*

The Active Directory Connector for ACLs indexes Access Control List (ACL) information into a configured "sidecar" Solr collection, so that it can be used by other connectors.

For applications involving multi-domain Active Directory, you need one LDAP data source per domain. Here is an example of a multi-domain Active Directory:

Domain	Type
lucidworks.com	parent domain
na.lucidworks.com	child domain
can.lucidworks.com	child domain
sa.lucidworks.com	child domain

In this example, you must have several data sources:

```
ldap://na.lucidworks.com Base DN: DC=na,DC=lucidworks,DC=com
ldap://can.lucidworks.com Base DN: DC=can,DC=lucidworks,DC=com
ldap://sa.lucidworks.com Base DN: DC=sa,DC=lucidworks,DC=com
```

Please note that querying the Active Directory Global Catalog to have a single LDAP ACL datasource does not work, as the global catalog does not replicate the `memberOf` attribute of `person` objects. Doing so results in users not able to see the expected documents.

However, you may be able to set up Active Directory to replicate that attribute. This would allow you to use a single global catalog for your entire Active Directory forest, if desired.

Handle a Multi-Domain Active Directory

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

The Active Directory Connector for ACLs indexes Access Control List (ACL) information into a configured "sidecar" Solr collection, so that it can be used by other connectors.

For applications involving multi-domain Active Directory, you need one LDAP data source per domain. Here is an example of a multi-domain Active Directory:

Domain	Type
lucidworks.com	parent domain
na.lucidworks.com	child domain
can.lucidworks.com	child domain
sa.lucidworks.com	child domain

In this example, you must have several data sources:

```
ldap://na.lucidworks.com Base DN: DC=na,DC=lucidworks,DC=com
ldap://can.lucidworks.com Base DN: DC=can,DC=lucidworks,DC=com
ldap://sa.lucidworks.com Base DN: DC=sa,DC=lucidworks,DC=com
```

Please note that querying the Active Directory Global Catalog to have a single LDAP ACL datasource does not work, as the global catalog does not replicate the `memberOf` attribute of `person` objects. Doing so results in users not able to see the expected documents.

However, you may be able to set up Active Directory to replicate that attribute. This would allow you to use a single global catalog for your entire Active Directory forest, if desired.

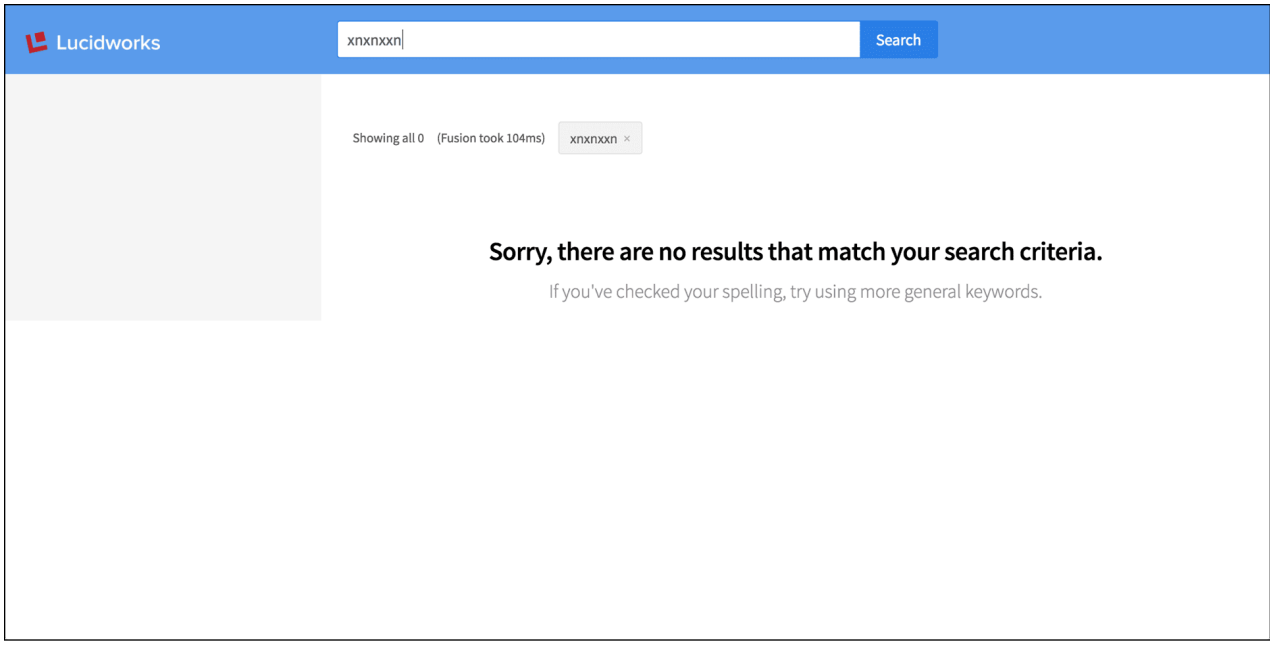
Handle the No-Results Case

How-to

> Compatible versions: 4.2
>
> Works with: *App Studio*

In the case of no search results, the `search:no-results` tag displays a default message:

```
<search:no-results response="response"></search:no-results>
```



Identify trending items or queries

How-to

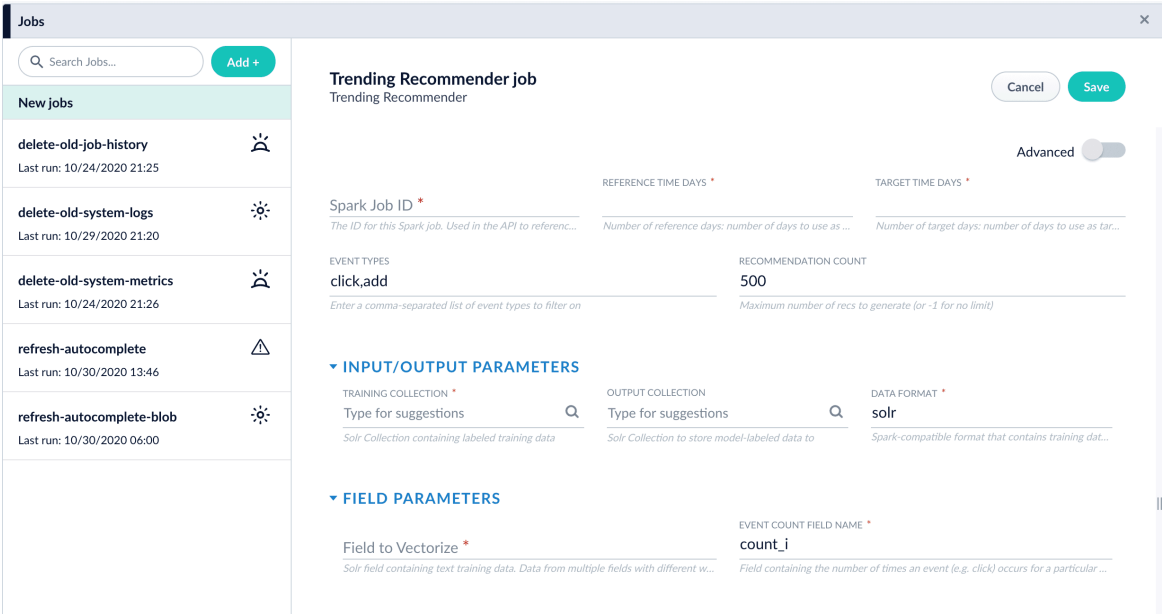
> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

The Trending Recommender job analyzes signals to measure customer engagement over time. Use this job to identify spikes in popularity for specific items or queries, then display those items to your users or analyze the trends for business purposes. You can configure any time window, such as daily, weekly, or monthly.

For complete details about the job's configuration options, see *Trending Recommender Jobs*.

How to identify trending items or queries

1. Navigate to **Collections** > **Jobs** > **Add +** > **Trending Recommender**.



2. Configure the job:

a. Enter an ID for this job.
b. In the **Reference Time Days** field, enter the number of days to use as a *baseline* for identifying trends, starting from today.
   For example, enter 21 days to analyze three weeks of signals data to use as a baseline.
c. In the **Target Time Days** field, enter the number of days to use as a *target* for identifying trends, starting from today.
   For example, enter 7 days to get documents or products whose popularity has spiked in the past week.

> **Tip**
>
> **Reference Time Days** and **Target Time Days** do not overlap. For example, with the values suggested above, a total of 28 days of signals are analyzed, and the first 21 days are compared to the last 7 days.

d. If you want to identify trending queries instead of trending items, change the value of the **Document ID Field** from `doc_id_s` to `query_s`.

> **Note**
>
> This field must be present in your signals. See the required signals fields in the *Trending Recommender Jobs* reference topic.

e. In the **Training Collection** field, enter the Solr collection or cloud path where signals are stored (the `COLLECTION_NAME_signals` collection by default).
f. In the **Output Collection** field, enter the Solr collection or cloud path where trend analysis data will be stored.
g. If you are using a format other than `solr`, enter it in the **Data Format** field.
h. In the **Solr Fields to Read** field, enter one or more field names containing text training data.

> **Tip**
>
> You can enter multiple field names with weights, as in `field1:weight1,field2:weight2…`.

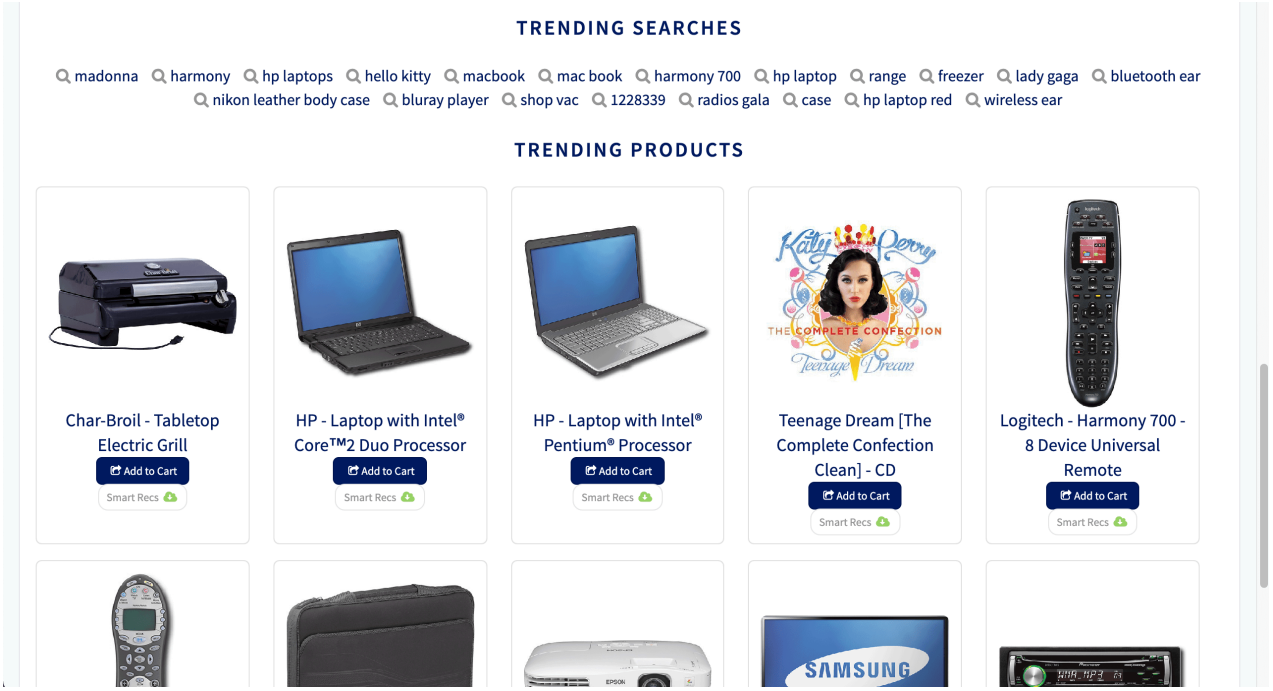
i. In the **Event Count Field Name** field, enter the name of the event count field in your training data, usually `count_i`.

3. Click **Save**.
4. Click **Run** > **Start** to run the job.
   The job outputs documents similar to this example:

```
{ "doc_id":250, "ref_hits":1, "ref_rank":1, "trgt_hits":1,
"trgt_rank":1, "vol_diff":0.5, "average_weekly_vol":0.5,
"hit_vol_ratio":2.0, "combine_score":1.0, "vol_diff_ratio":1.0,
"ref_wt_vol_diff_ratio":1.0, "vol_diff_wt_vol_diff_ratio":0.5,
"log_diff_wt_ratio":1.3068528194400546, "trend-
type":"prds_weekly", "id":"284f930d-d750-49a2-90ac-
be4692bddda9", "_version_":1682995654191742976 }
```

5. Configure a query pipeline to retrieve trending items from the job's output collection for display or further analysis. For more information about pipelines created when recommendations are enabled, see *enable recommendations*.

> **Tip**
>
> Search on the `log_diff_wt_ratio` field to find the top trending items in the output collection.



Identify Trending Items or Queries

How-to

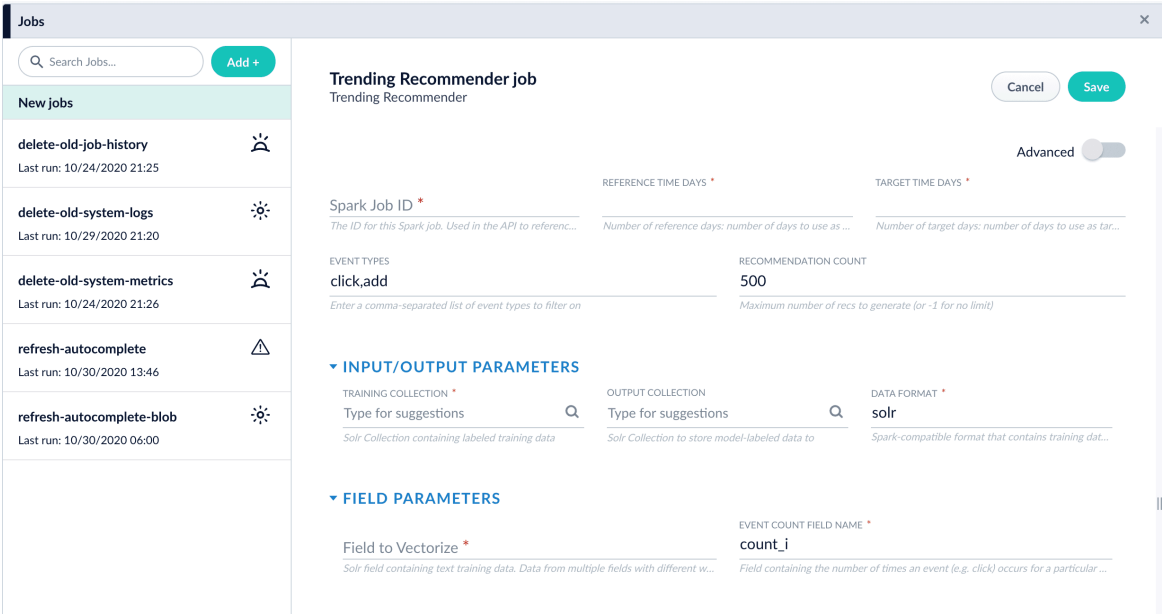
> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion*

The Trending Recommender job analyzes signals to measure customer engagement over time. Use this job to identify spikes in popularity for specific items or queries, then display those items to your users or analyze the trends for business purposes. You can configure any time window, such as daily, weekly, or monthly.

For complete details about the job's configuration options, see *Trending Recommender Jobs*.

How to identify trending items or queries

1. Navigate to **Collections** > **Jobs** > **Add +** > **Trending Recommender**.



2. Configure the job:

a. Enter an ID for this job.
b. In the **Reference Time Days** field, enter the number of days to use as a *baseline* for identifying trends, starting from today.
   For example, enter 21 days to analyze three weeks of signals data to use as a baseline.
c. In the **Target Time Days** field, enter the number of days to use as a *target* for identifying trends, starting from today.
   For example, enter 7 days to get documents or products whose popularity has spiked in the past week.

> **Tip**
>
> **Reference Time Days** and **Target Time Days** do not overlap. For example, with the values suggested above, a total of 28 days of signals are analyzed, and the first 21 days are compared to the last 7 days.

d. If you want to identify trending queries instead of trending items, change the value of the **Document ID Field** from `doc_id_s` to `query_s`.

> **Note**
>
> This field must be present in your signals. See the required signals fields in the *Trending Recommender Jobs* reference topic.

e. In the **Training Collection** field, enter the Solr collection or cloud path where signals are stored (the `COLLECTION_NAME_signals` collection by default).
f. In the **Output Collection** field, enter the Solr collection or cloud path where trend analysis data will be stored.
g. If you are using a format other than `solr`, enter it in the **Data Format** field.
h. In the **Solr Fields to Read** field, enter one or more field names containing text training data.

> **Tip**
>
> You can enter multiple field names with weights, as in `field1:weight1,field2:weight2…`.

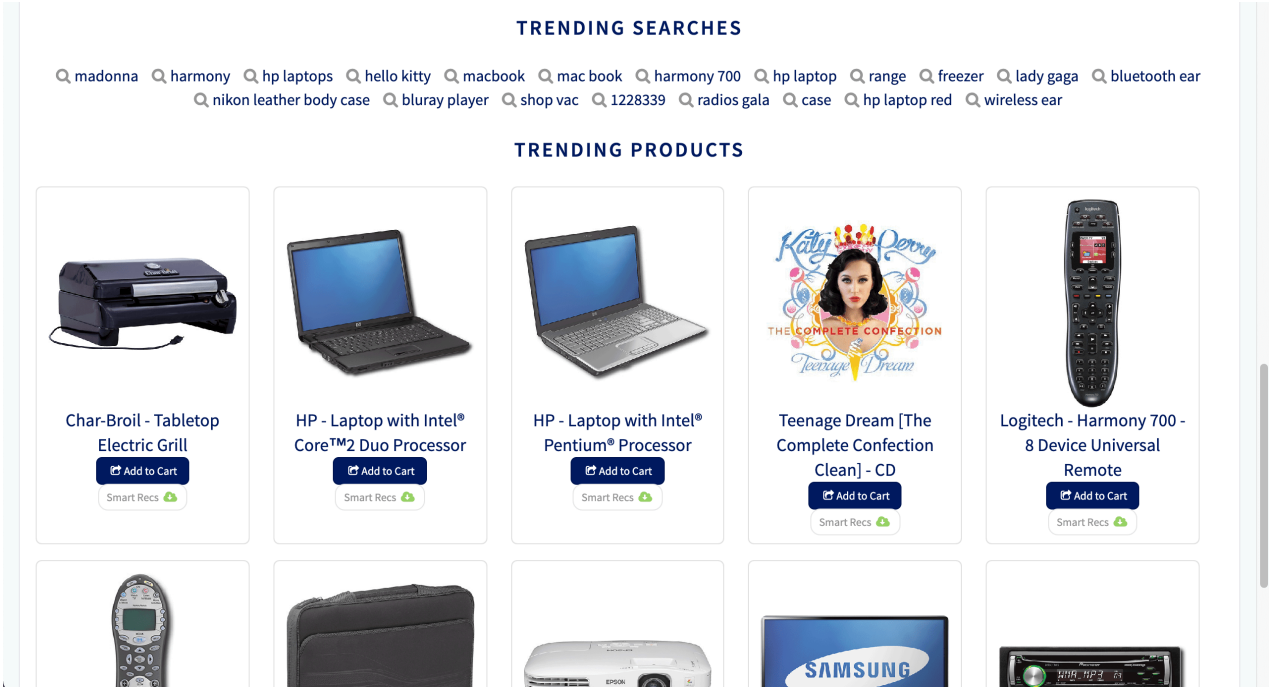
i. In the **Event Count Field Name** field, enter the name of the event count field in your training data, usually `count_i`.

3. Click **Save**.
4. Click **Run** > **Start** to run the job.
   The job outputs documents similar to this example:

```
{ "doc_id":250, "ref_hits":1, "ref_rank":1, "trgt_hits":1,
"trgt_rank":1, "vol_diff":0.5, "average_weekly_vol":0.5,
"hit_vol_ratio":2.0, "combine_score":1.0, "vol_diff_ratio":1.0,
"ref_wt_vol_diff_ratio":1.0, "vol_diff_wt_vol_diff_ratio":0.5,
"log_diff_wt_ratio":1.3068528194400546, "trend-
type":"prds_weekly", "id":"284f930d-d750-49a2-90ac-
be4692bddda9", "_version_":1682995654191742976 }
```

5. Configure a query pipeline to retrieve trending items from the job's output collection for display or further analysis. For information about pipelines created when recommendations are enabled, see *enable recommendations*.

> **Tip**
>
> Search on the `log_diff_wt_ratio` field to find the top trending items in the output collection.



Import a Fusion App

How-to

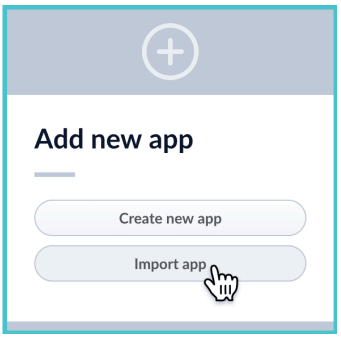
> Compatible versions: 4.2
>
> Works with: *Fusion Server*

To import an app, you can use either the Fusion UI or the *Objects API*.

Import an app with the Fusion UI

How to import an app with the Fusion UI

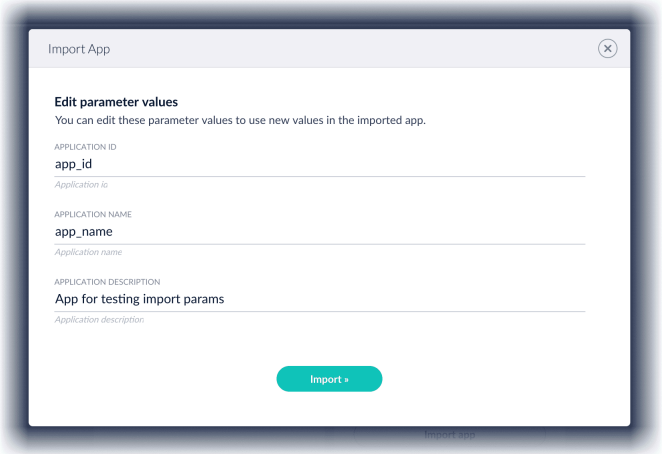
1. Navigate to the launcher.
2. Click **Import app**.



3. Under **Data File**, click **Choose File** and select the zip file containing the app you want to import.
4. If your app has *usernames and passwords* in a separate file, select it under **Variables File**.
   If the Variables File is needed, it *must* be a separate file that is not in a .zip file. It is a .json map of variables to values. The following is an example:

```
{ "secret.dataSources.Inventory_BR_S3_DS.password":"SOMETHING",
"secret.dataSources.LLM_A_BR_S3_DS.password":"FmJSaDE9Tj5REDACTED",
"secret.dataSources.LLM_BR_Inventory_S3_DS.password":"FmJSaDE9Tj5GzIVv
ethAC4Huh",
"secret.dataSources.LLM_BR_Load_S3_DS.password":"FmJSaDE9Tj5GzIVvethAC
4" }
```

5. In Fusion 4.1 and later, you can sometimes edit parameter values to use the new values in the imported app. If this is the case, Fusion displays a dialog box that lets you edit the parameter values.



Make desired changes, and then click **Import**.

Import an app with the Objects API

Like any other Fusion object, an exported app is contained in a zip file which Fusion can consume upon import. The zip file does not need to be expanded first.

Import objects from a file and stop if there are conflicts

```
curl -u USERNAME:PASSWORD -H "Content-Type:multipart/form-data" -X
POST -F 'importData=@/Users/admin/Fusion/export.zip'
http://localhost:8764/api/objects/import?importPolicy=abort
```

Import objects, substitute the password variables, and merge any conflicts

```
curl -u USERNAME:PASSWORD -H "Content-Type:multipart/form-data" -X
POST -F 'importData=@/Users/admin/Fusion/export.zip' -F
'variableValues=@password_file.json'
http://localhost:8764/api/objects/import?importPolicy=merge
```

Note

`password_file.json` must contain plaintext passwords.

Import a zip file of Fusion objects and merge any conflicts

```
curl -u USERNAME:PASSWORD -H "Content-Type:multipart/form-data" -X
POST -F 'importData=@/Users/admin/Fusion/techpubs.zip'
http://localhost:8764/api/objects/import?importPolicy=merge
```

Change an app's default launch action

Important

This is only available in Fusion 4.1 and Fusion 4.2.

The default launch action is the action that Fusion takes when you click an app's tile or name in the Fusion Launcher:

- After you create an app, the default launch action is to open the app in the Fusion workspace.
- If you create and publish a UI for the app, then you can change the default launch action to open the app in the browser.

To change an app's default launch action in the Fusion UI:

1. From the Launcher, hover over an app tile, and then click the Configure icon.

2. Under **Default launch action**, use the drop-down list to select a launch action:
   - Select **Fusion workspace** to open the app in the Fusion workspace.
   - Select an app name to open that app in the browser.

3. Click **OK**.

Import a Fusion App

How-to

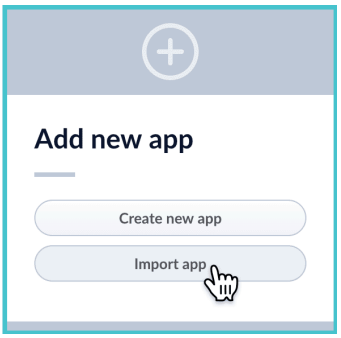
> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion*

To import an app, you can use either the Fusion UI or the *Objects API*.

Import an app with the Fusion UI

How to import an app with the Fusion UI

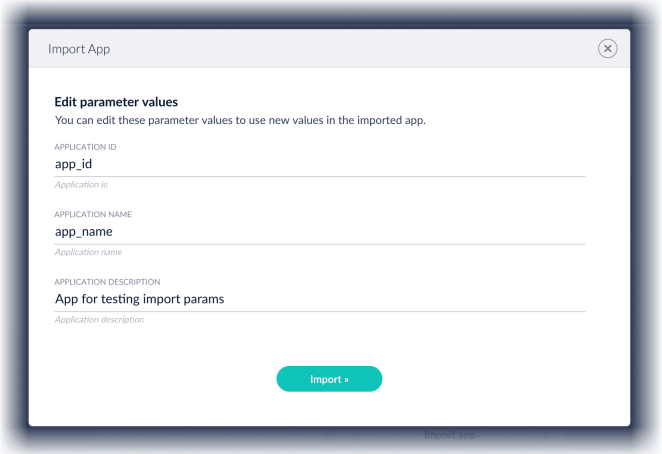
1. Navigate to the launcher.
2. Click **Import app**.



3. Under **Data File**, click **Choose File** and select the zip file containing the app you want to import.
4. If your app has *usernames and passwords* in a separate file, select it under **Variables File**.
   If the Variables File is needed, it *must* be a separate file that is not in a .zip file. It is a .json map of variables to values. The following is an example:

```
{ "secret.dataSources.Inventory_BR_S3_DS.password":"SOMETHING",
"secret.dataSources.LLM_A_BR_S3_DS.password":"FmJSaDE9Tj5REDACTED",
"secret.dataSources.LLM_BR_Inventory_S3_DS.password":"FmJSaDE9Tj5GzIVv
ethAC4Huh",
"secret.dataSources.LLM_BR_Load_S3_DS.password":"FmJSaDE9Tj5GzIVvethAC
4" }
```

5. You can sometimes edit parameter values to use the new values in the imported app. If this is the case, Fusion displays a dialog box that lets you edit the parameter values.



Make desired changes, and then click **Import**.

Import an app with the Objects API

Like any other Fusion object, an exported app is contained in a zip file which Fusion can consume upon import. The zip file does not need to be expanded first.

Import objects from a file and stop if there are conflicts

```
curl -u USERNAME:PASSWORD -H "Content-Type:multipart/form-data" -X
POST -F 'importData=@/Users/admin/Fusion/export.zip'
http://localhost:6764/api/objects/import?importPolicy=abort
```

Import objects, substitute the password variables, and merge any conflicts

```
curl -u USERNAME:PASSWORD -H "Content-Type:multipart/form-data" -X
POST -F 'importData=@/Users/admin/Fusion/export.zip' -F
'variableValues=@password_file.json'
http://localhost:6764/api/objects/import?importPolicy=merge
```

Note

`password_file.json` must contain plaintext passwords.

Import a zip file of Fusion objects and merge any conflicts

```
curl -u USERNAME:PASSWORD -H "Content-Type:multipart/form-data" -X
POST -F 'importData=@/Users/admin/Fusion/techpubs.zip'
http://localhost:6764/api/objects/import?importPolicy=merge
```

Import a Managed Fusion App

How-to

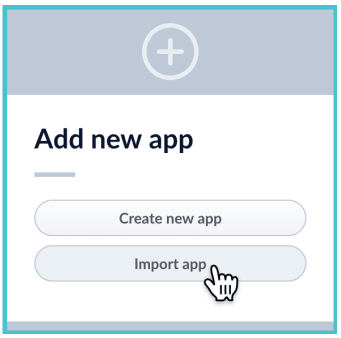
> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

To import an app, you can use either the Managed Fusion UI or the *Objects API*.

Import an app with the Managed Fusion UI

How to import an app with the Managed Fusion UI

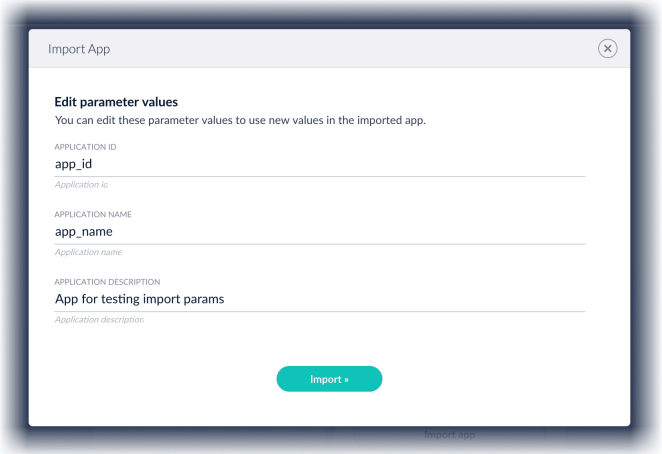
1. Sign in to Managed Fusion.
2. In the Add new app screen, click **Import app**.



3. Under **Data File**, click **Choose File** and select the zip file containing the app you want to import.
4. If your app has *usernames and passwords* in a separate file, select it under **Variables File**.
   If the Variables File is needed, it *must* be a separate file that is not in a .zip file. It is a .json map of variables to values. The following is an example:

```
{ "secret.dataSources.Inventory_BR_S3_DS.password":"SOMETHING",
"secret.dataSources.LLM_A_BR_S3_DS.password":"FmJSaDE9Tj5REDACTED",
"secret.dataSources.LLM_BR_Inventory_S3_DS.password":"FmJSaDE9Tj5GzIVv
ethAC4Huh",
"secret.dataSources.LLM_BR_Load_S3_DS.password":"FmJSaDE9Tj5GzIVvethAC
4" }
```

5. You can sometimes edit parameter values to use the new values in the imported app. If this is the case, Managed Fusion displays a dialog box that lets you edit the parameter values.



Make desired changes, and then click **Import**.

Import an app with the Objects API

Like any other Managed Fusion object, an exported app is contained in a zip file which Managed Fusion can consume upon import. The zip file does not need to be expanded first.

Import objects from a file and stop if there are conflicts

```
curl -u USERNAME:PASSWORD -H "Content-Type:multipart/form-data" -X
POST -F 'importData=@/Users/admin/Fusion/export.zip'
http://localhost:6764/api/objects/import?importPolicy=abort
```

Import objects, substitute the password variables, and merge any conflicts

```
curl -u USERNAME:PASSWORD -H "Content-Type:multipart/form-data" -X
POST -F 'importData=@/Users/admin/Fusion/export.zip' -F
'variableValues=@password_file.json'
http://localhost:6764/api/objects/import?importPolicy=merge
```

Note

`password_file.json` must contain plaintext passwords.

Import a zip file of Managed Fusion objects and merge any conflicts

```
curl -u USERNAME:PASSWORD -H "Content-Type:multipart/form-data" -X
POST -F 'importData=@/Users/admin/Fusion/techpubs.zip'
http://localhost:6764/api/objects/import?importPolicy=merge
```

Import Data with Hive

How-to

> Compatible versions: 4.2, 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion Connectors*, *Fusion Server*, *Fusion*

Fusion ships with a Serializer/Deserializer (SerDe) for Hive, included in the distribution as `lucidworks-hive-serde-v2.2.6.jar` in `$FUSION_HOME/apps/connectors/resources/lucid.hadoop/jobs`.

Important

For Fusion 4.1.x and 4.2.x, the preferred method of importing data with Hive is to use the *Parallel Bulk Loader*. The import procedure does not apply to Fusion 5.x.

Features

- Index Hive table data to Solr.
- Read Solr index data to a Hive table.
- Kerberos support for securing communication between Hive and Solr.
- As of v2.2.4 of the SerDe, integration with Lucidworks Fusion is supported.
    - Fusion's index pipelines can be used to index data to Fusion.
    - Fusion's query pipelines can be used to query Fusion's Solr instance for data to insert into a Hive table.

Add the SerDe Jar to Hive Classpath

In order for the Hive SerDe to work with Solr, the SerDe jar must be added to Hive's classpath using the `hive.aux.jars.path` capability. There are several options for this, described below.

It's considered a best practice to use a single directory for all auxiliary jars you may want to add to Hive so you only need to define a single path. However, you must then copy any jars you want to use to that path.

Note

The following options all assume you have created such a directory at `/usr/hive/auxlib`; if you use another path, update the path in the examples accordingly.

1. If you use Hive with Ambari (as with the Hortonworks HDP distribution), go to **Hive** **Configs** **Advanced**, and scroll down to **Advanced hive-env** **hive-env template**. Find the section where the `HIVE_AUX_JARS_PATH` is defined, and add the path to each line which starts with `export`. What you want will end up looking like:

```
# Folder containing extra libraries required for hive
compilation/execution can be controlled by: if [
"${HIVE_AUX_JARS_PATH}" != "" ]; then if [ -f
"${HIVE_AUX_JARS_PATH}" ]; then export
HIVE_AUX_JARS_PATH=${HIVE_AUX_JARS_PATH},/usr/hive/auxlib elif [
-d "/usr/hdp/current/hive-webhcat/share/hcatalog" ]; then export
HIVE_AUX_JARS_PATH=/usr/hdp/current/hive-
webhcat/share/hcatalog/hive-hcatalog-core.jar,/usr/hive/auxlib
fi elif [ -d "/usr/hdp/current/hive-webhcat/share/hcatalog" ];
then export HIVE_AUX_JARS_PATH=/usr/hdp/current/hive-
webhcat/share/hcatalog/hive-hcatalog-core.jar,/usr/hive/auxlib
fi
```

2. If not using Ambari or similar cluster management tool, you can add the jar location to `hive/conf/hive-site.xml`:

```
<property> <name>hive.aux.jars.path</name>
<value>/usr/hive/auxlib</value> </property>
```

3. Another option is to launch Hive with the path defined with the `auxpath` variable:

```
hive --auxpath /usr/hive/auxlib
```

There are also other approaches that could be used. Keep in mind, though, that the jar **must** be loaded into the classpath, adding it with the `ADD JAR` function is not sufficient.

Indexing Data to Fusion

If you use Lucidworks Fusion, you can index data from Hive to Solr via Fusion's index pipelines. These pipelines allow you several options for further transforming your data.

Tip

If you are using Fusion v3.0.x, you already have the Hive SerDe in Fusion's `./apps/connectors/resources/lucid.hadoop/jobs` directory. The SerDe jar that supports Fusion is v2.2.4 or higher. This was released with Fusion 3.0.

If you are using Fusion 3.1.x and higher, you will need to download the Hive SerDe from http://lucidworks.com/connectors/ . Choose the proper Hadoop distribution and the resulting .zip file will include the Hive SerDe.

A 2.2.4 or higher jar built from this repository will also work with Fusion 2.4.x releases.

This is an example Hive command to create an external table to index documents in Fusion and to query the table later.

```
hive> CREATE EXTERNAL TABLE fusion (id string, field1_s string,
field2_i int) STORED BY
'com.lucidworks.hadoop.hive.FusionStorageHandler' LOCATION
'/tmp/fusion' TBLPROPERTIES('fusion.endpoints' =
'http://localhost:8764/api/apollo/index-
pipelines/<pipeline>/collections/<collection>/index',
'fusion.fail.on.error' = 'false', 'fusion.buffer.timeoutms' = '1000',
'fusion.batchSize' = '500', 'fusion.realm' = 'KERBEROS', 'fusion.user'
= 'fusion-indexer@FUSIONSERVER.COM', 'java.security.auth.login.config'
= '/path/to/JAAS/file', 'fusion.jaas.appname' = 'FusionClient',
'fusion.query.endpoints' = 'http://localhost:8764/api/apollo/query-
pipelines/pipeline-id/collections/collection-id', 'fusion.query' =
'*:*');
```

In this example, we have created an external table named "fusion", and defined a custom storage handler (`STORED BY 'com.lucidworks.hadoop.hive.FusionStorageHandler'`) that a class included with the Hive SerDe jar designed for use with Fusion.

Note that all of the same caveats about field types discussed in the section *[Defining Fields for Solr]* apply to Fusion as well. In Fusion, however, you have the option of using an index pipeline to perform specific field mapping instead of using dynamic fields.

The LOCATION indicates the location in HDFS where the table data will be stored. In this example, we have chosen to use `/tmp/fusion` .

In the section TBLPROPERTIES, we define several properties for Fusion so the data can be indexed to the right Fusion installation and collection:

`fusion.endpoints`
: The full URL to the index pipeline in Fusion. The URL should include the pipeline name and the collection data will be indexed to.

fusion.fail.on.error

If `true`, when an error is encountered, such as if a row could not be parsed, indexing will stop. This is `false` by default.

fusion.buffer.timeoutms

The amount of time, in milliseconds, to buffer documents before sending them to Fusion. The default is 1000. Documents will be sent to Fusion when either this value or `fusion.batchSize` is met.

fusion.batchSize

The number of documents to batch before sending the batch to Fusion. The default is 500. Documents will be sent to Fusion when either this value or `fusion.buffer.timeoutms` is met.

fusion.realm

This is used with `fusion.user` and `fusion.password` to authenticate to Fusion for indexing data. Two options are supported, `KERBEROS` or `NATIVE`.

Kerberos authentication is supported with the additional definition of a JAAS file. The properties `java.security.auth.login.config` and `fusion.jaas.appname` are used to define the location of the JAAS file and the section of the file to use.

Native authentication uses a Fusion-defined username and password. This user must exist in Fusion, and have the proper permissions to index documents.

fusion.user

The Fusion username or Kerberos principal to use for authentication to Fusion. If a Fusion username is used (`'fusion.realm' = 'NATIVE'`), the `fusion.password` must also be supplied.

fusion.password

This property is not shown in the example above. The password for the `fusion.user` when the `fusion.realm` is `NATIVE`.

java.security.auth.login.config

This property defines the path to a JAAS file that contains a service principal and keytab location for a user who is authorized to read from and write to Fusion and Hive.

The JAAS configuration file **must** be copied to the same path on every node where a Node Manager is running (i.e., every node where map/reduce tasks are executed). Here is a sample section of a JAAS file:

```
Client { 1 com.sun.security.auth.module.Krb5LoginModule required
useKeyTab=true keyTab="/data/fusion-indexer.keytab" 2
storeKey=true useTicketCache=true debug=true principal="fusion-
indexer@FUSIONSERVER.COM"; 3 };
```

1	The name of this section of the JAAS file. This name will be used with the `fusion.jaas.appname` parameter.
2	The location of the keytab file.
3	The service principal name. This should be a different principal than the one used for Fusion, but must have access to both Fusion and Hive. This name is used with the `fusion.user` parameter described above.

fusion.jaas.appname

Used only when indexing to or reading from Fusion when it is secured with Kerberos.

This property provides the name of the section in the JAAS file that includes the correct service principal and keytab path.

`fusion.query.endpoints`
The full URL to a query pipeline in Fusion. The URL should include the pipeline name and the collection data will be read from. You should also specify the request handler to be used.

If you do not intend to query your Fusion data from Hive, you can skip this parameter.

`fusion.query`
The query to run in Fusion to select records to be read into Hive. This is `*:*` by default, which selects all records in the index.

If you do not intend to query your Fusion data from Hive, you can skip this parameter.

Query and Insert Data to Hive

Once the table is configured, any syntactically correct Hive query will be able to query the index.

For example, to select three fields named "id", "field1_s", and "field2_i" from the "solr" table, you would use a query such as:

```
hive> SELECT id, field1_s, field2_i FROM solr;
```

Replace the table name as appropriate to use this example with your data.

To join data from tables, you can make a request such as:

```
hive> SELECT id, field1_s, field2_i FROM solr left JOIN sometable
right WHERE left.id = right.id;
```

And finally, to insert data to a table, simply use the Solr table as the target for the Hive INSERT statement, such as:

```
hive> INSERT INTO solr SELECT id, field1_s, field2_i FROM sometable;
```

Example Indexing Hive to Solr

Solr includes a small number of sample documents for use when getting started. One of these is a CSV file containing book metadata. This file is found in your Solr installation, at `$SOLR_HOME/example/exampledocs/books.csv`.

Using the sample `books.csv` file, we can see a detailed example of creating a table, loading data to it, and indexing that data to Solr.

```
CREATE TABLE books (id STRING, cat STRING, title STRING, price FLOAT,
in_stock BOOLEAN, author STRING, series STRING, seq INT, genre STRING)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ','; 1 LOAD DATA LOCAL
INPATH '/solr/example/exampledocs/books.csv' OVERWRITE INTO TABLE
books; 2 CREATE EXTERNAL TABLE solr (id STRING, cat_s STRING, title_s
STRING, price_f FLOAT, in_stock_b BOOLEAN, author_s STRING, series_s
STRING, seq_i INT, genre_s STRING) 3 STORED BY
'com.lucidworks.hadoop.hive.LWStorageHandler' 4 LOCATION '/tmp/solr'
5 TBLPROPERTIES('solr.zkhost' =
'zknode1:2181,zknode2:2181,zknode3:2181/solr', 'solr.collection' =
'gettingstarted', 'solr.query' = '*:*', 6 'lww.jaas.file' =
'/data/jaas-client.conf'); 7 INSERT OVERWRITE TABLE solr SELECT b.*
FROM books b;
```

1	Define the table `books`, and provide the field names and field types that will make up the table.
2	Load the data from the `books.csv` file.
3	Create an external table named `solr`, and provide the field names and field types that will make up the table. These will be the same field names as in your local Hive table, so we can index all of the same data to Solr.
4	Define the custom storage handler provided by the `lucidworks-hive-serde-v2.2.6.jar`.
5	Define storage location in HDFS.
6	The query to run in Solr to read records from Solr for use in Hive.
7	Define the location of Solr (or ZooKeeper if using SolrCloud), the collection in Solr to index the data to, and the query to use when reading the table. This example also refers to a JAAS configuration file that will be used to authenticate to the Kerberized Solr cluster.

Import Data with Pig

How-to

> Compatible versions: 4.2, 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion Connectors*, *Fusion Server*, *Fusion*

You can use Pig to import data into Fusion, using the `lucidworks-pig-functions-v2.2.6.jar` file found in `$FUSION_HOME/apps/connectors/resources/lucid.hadoop/jobs`.

Available Functions

The Pig functions included in the `lucidworks-pig-functions-v2.2.6.jar` are three UserDefined Functions (UDF) and two Store functions. These functions are:

- `com/lucidworks/hadoop/pig/SolrStoreFunc.class`
- `com/lucidworks/hadoop/pig/FusionIndexPipelinesStoreFunc.class`
- `com/lucidworks/hadoop/pig/EpochToCalendar.class`
- `com/lucidworks/hadoop/pig/Extract.class`
- `com/lucidworks/hadoop/pig/Histogram.class`

Using The Functions

Register the Functions

There are two approaches to using functions in Pig: `REGISTER` them in the script, or load them with your Pig command line request.

If using `REGISTER`, the Pig function jars must be put in HDFS in order to be used by your Pig script. It can be located anywhere in HDFS; you can either supply the path in your script or use a variable and define the variable with `-p` property definition.

The example below uses the second approach, loading the jars with the `-Dpig.additional.jars` system property when launching the script. With this approach, the jars can be located anywhere on the machine where the script will be run.

Indexing Data to Fusion

When indexing data to Fusion, there are several parameters to pass with your script in order to output data to Fusion for indexing.

These parameters can be made into variables in the script, with the proper values passed on the command line when the script is initiated. The example script below shows how to do this for Solr. The theory is the same for Fusion, only the parameter names would change as appropriate:

`fusion.endpoints`
: The full URL to the index pipeline in Fusion. The URL should include the pipeline name and the collection data will be indexed to.

`fusion.fail.on.error`
: If `true`, when an error is encountered, such as if a row could not be parsed, indexing will stop. This is `false` by default.

`fusion.buffer.timeoutms`
: The amount of time, in milliseconds, to buffer documents before sending them to Fusion. The default is 1000. Documents will be sent to Fusion when either this value or `fusion.batchSize` is met.

`fusion.batchSize`
: The number of documents to batch before sending the batch to Fusion. The default is 500. Documents will be sent to Fusion when either this value or `fusion.buffer.timeoutms` is met.

`fusion.realm`
: This is used with `fusion.user` and `fusion.password` to authenticate to Fusion for indexing data. Two options are supported, `KERBEROS` or `NATIVE`.

  Kerberos authentication is supported with the additional definition of a JAAS file. The properties `java.security.auth.login.config` and `fusion.jaas.appname` are used to define the location of the JAAS file and the section of the file to use. These are described in more detail below.

  Native authentication uses a Fusion-defined username and password. This user must exist in Fusion, and have the proper permissions to index documents.

`fusion.user`

The Fusion username or Kerberos principal to use for authentication to Fusion.

If a Fusion username is used (`'fusion.realm' = 'NATIVE'`), the `fusion.password` must also be supplied.

`fusion.pass`

This property is not shown in the example above. The password for the `fusion.user` when the `fusion.realm` is `NATIVE` .

INDEXING TO A KERBERIZED FUSION INSTALLATION

When Fusion is secured with Kerberos, Pig scripts must include the full path to a JAAS file that includes the service principal and the path to a keytab file that will be used to index the output of the script to Fusion.

Additionally, a Kerberos ticket must be obtained on the server for the principal using `kinit` .

`java.security.auth.login.config`

This property defines the path to a JAAS file that contains a service principal and keytab location for a user who is authorized to write to Fusion.

The JAAS configuration file **must** be copied to the same path on every node where a Node Manager is running (i.e., every node where map/reduce tasks are executed). Here is a sample section of a JAAS file:

```
Client {  1  com.sun.security.auth.module.Krb5LoginModule required
useKeyTab=true keyTab="/data/fusion-indexer.keytab"  2
storeKey=true useTicketCache=true debug=true principal="fusion-
indexer@FUSIONSERVER.COM";  3  };
```

1	The name of this section of the JAAS file. This name will be used with the `fusion.jaas.appname` parameter.
2	The location of the keytab file.
3	The service principal name. This should be a different principal than the one used for Fusion, but must have access to both Fusion and Pig. This name is used with the `fusion.user` parameter described above.

`fusion.jaas.appname`

Used only when indexing to or reading from Fusion when it is secured with Kerberos.

This property provides the name of the section in the JAAS file that includes the correct service principal and keytab path.

Sample CSV Script

The following Pig script will take a simple CSV file and index it to Solr.

```
set solr.zkhost '$zkHost'; set solr.collection '$collection'; 1 A =
load '$csv' using PigStorage(',') as
(id_s:chararray,city_s:chararray,country_s:chararray,code_s:chararray,
code2_s:chararray,latitude_s:chararray,longitude_s:chararray,flag_s:ch
ararray); 2 --dump A; B = FOREACH A GENERATE $0 as id, 'city_s', $1,
'country_s', $2, 'code_s', $3, 'code2_s', $4, 'latitude_s', $5,
'longitude_s', $6, 'flag_s', $7; 3 ok = store B into 'SOLR' using
com.lucidworks.hadoop.pig.SolrStoreFunc(); 4
```

This relatively simple script is doing several things that help to understand how the Solr Pig functions work.

1	This and the line above define parameters that are needed by `SolrStoreFunc` to know where Solr is. `SolrStoreFunc` needs the properties `solr.zkhost` and `solr.collection`, and these lines are mapping the `zkhost` and `collection` parameters we will pass when invoking Pig to the required properties.
2	Load the CSV file, the path and name we will pass with the `csv` parameter. We also define the field names for each column in CSV file, and their types.
3	For each item in the CSV file, generate a document id from the first field (`$0`) and then define each field name and value in `name, value` pairs.
4	Load the documents into Solr, using the `SolrStoreFunc`. While we don't need to define the location of Solr here, the function will use the `zkhost` and `collection` properties that we will pass when we invoke our Pig script.

Warning

When using `SolrStoreFunc`, the document ID **must** be the first field.

When we want to run this script, we invoke Pig and define several parameters we have referenced in the script with the `-p` option, such as in this command:

```
./bin/pig -Dpig.additional.jars=/path/to/lucidworks-pig-functions-
v2.2.6.jar -p csv=/path/to/my/csv/airports.dat -p
zkHost=zknode1:2181,zknode2:2181,zknode3:2181/solr -p
collection=myCollection ~/myScripts/index-csv.pig
```

The parameters to pass are:

`CSV`

The path and name of the CSV file we want to process.

`zkhost`

The ZooKeeper connection string for a SolrCloud cluster, in the form of `zkhost1:port,zkhost2:port,zkhost3:port/chroot`. In the script, we mapped this to the `solr.zkhost` property, which is required by the `SolrStoreFunc` to know where to send the output documents.

`collection`

The Solr collection to index into. In the script, we mapped this to the `solr.collection` property, which is required by the `SolrStoreFunc` to know the Solr collection the documents should be indexed to.

Tip

The `zkhost` parameter above is only used if you are indexing to a SolrCloud cluster, which uses ZooKeeper to route indexing and query requests.

If, however, you are not using SolrCloud, you can use the `solrUrl` parameter, which takes the location of a standalone Solr instance, in the form of `http://host:port/solr`.

In the script, you would change the line that maps `solr.zkhost` to the `zkhost` property to map `solr.server.url` to the `solrUrl` property. For example:

```
`set solr.server.url '$solrUrl';`
```

Import Data with the REST API

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

It is often possible to get documents into Managed Fusion by configuring a datasource with the appropriate connector: *Connectors Configuration Reference*.

But if there are obstacles to using connectors, it can be simpler to index documents with a REST API call to an index profile or pipeline.

Push documents to Managed Fusion using index profiles

Index profiles allow you to send documents to a consistent endpoint (the profile alias) and change the backend index pipeline as needed. The profile is also a simple way to use one pipeline for multiple collections without any one collection "owning" the pipeline. See *Managed Fusion Index Profiles*.

You can send documents directly to an index using the *Index REST API*. The request path is:

```
/api/apps/APP_NAME/index/INDEX_PROFILE
```

These requests are sent as a POST request. The request header specifies the format of the contents of the request body. Create an index profile in the Managed Fusion UI.

To send a streaming list of JSON documents, you can send the JSON file that holds these objects to the API listed above with `application/json` as the content type. If your JSON file is a list or array of many items, the endpoint operates in a streaming way and indexes the docs as necessary.

Send data to an index profile that is part of an app

Accessing an index profile through an app lets a Managed Fusion admin secure and manage all objects on a per-app basis. Security is then determined by whether a user can access an app. This is the recommended way to manage permissions in Managed Fusion.

The syntax for sending documents to an index profile that is part of an app is as follows:

```
curl -u USERNAME:PASSWORD -X POST -H 'content-type: application/json'
https://EXAMPLE_COMPANY.b.lucidworks.cloud/api/apps/APP_NAME/index/IND
EX_PROFILE --data-binary @my-json-data.json
```

Note

Spaces in an app name become underscores. Spaces in an index profile name become hyphens.

To prevent the terminal from displaying all the data and metadata it indexes — useful if you are indexing a large file — you can optionally append `?echo=false` to the URL.

Be sure to set the content type header properly for the content being sent. Some frequently used content types are:

- Text: `application/json` , `application/xml`
- PDF documents: `application/pdf`
- MS Office:
  - DOCX: `application/vnd.openxmlformats-officedocument.wordprocessingml.document`
  - XLSX: `application/vnd.openxmlformats-officedocument.spreadsheetml.sheet`

- PPTX: `application/vnd.vnd.openxmlformats-officedocument.presentationml.presentation`
- More types: http://filext.com/faq/office_mime_types.php

Example: Send JSON data to an index profile under an app

In `$FUSION_HOME/apps/solr-dist/example/exampledocs` you can find a few sample documents. This example uses one of these, `books.json`.

To push JSON data to an index profile under an app:

1. Create an index profile. In the Managed Fusion UI, click **Indexing > Index Profiles** and follow the prompts.
2. From the directory containing `books.json`, enter the following, substituting your values for username, password, and index profile name:

```
curl -u USERNAME:PASSWORD -X POST -H 'content-type:
application/json'
https://EXAMPLE_COMPANY.b.lucidworks.cloud/api/apps/APP_NAME/ind
ex/INDEX_PROFILE?echo=false --data-binary @books.json
```

3. Test that your data has made it into Managed Fusion:
   a. Log into the Managed Fusion UI.
   b. Navigate to the app where you sent your data.
   c. Navigate to the Query Workbench.
   d. Search for `*:*`.
   e. Select relevant Display Fields, for example `author` and `name`.

Example: Send JSON data without defining an app

In most cases it is best to delegate permissions on a per-app basis. But if your use case requires it, you can push data to Managed Fusion without defining an app.

To send JSON data without app security, issue the following curl command:

```
curl -u USERNAME:PASSWORD -X POST -H 'content-type: application/json'
https://EXAMPLE_COMPANY.b.lucidworks.cloud/api/index/INDEX_PROFILE --
data-binary @my-json-data.json
```

Example: Send XML data to an index profile with an app

To send XML data to an app, use the following:

```
curl -u USERNAME:PASSWORD -X POST -H 'content-type: application/xml'
https://EXAMPLE_COMPANY.b.lucidworks.cloud/api/apps/APP_NAME/index/IND
EX_PROFILE --data-binary @my-xml-file.xml
```

You can also create documents using the *PipelineDocument* JSON notation.

Send documents to an index pipeline

Although sending documents to an index profile is recommended, if your use case requires it, you can send documents directly to an *index pipeline*.

Specify a parser

When you push data to a pipeline, you can specify the name of the parser by adding a parserId querystring parameter to the URL. For example: `https://EXAMPLE_COMPANY.b.lucidworks.cloud/api/index-pipelines/INDEX_PIPELINE/collections/COLLECTION_NAME/index?parserId=PARSER`.

If you do not specify a parser, and you are indexing outside of an app (`https://EXAMPLE_COMPANY.b.lucidworks.cloud/api/index-pipelines/…`), then the `_system` parser is used.

If you do not specify a parser, and you are indexing in an app context (`https://EXAMPLE_COMPANY.b.lucidworks.cloud/api/apps/APP_NAME/index-pipelines/…`), then the parser with the same name as the app is used.

Indexing CSV Files

In the usual case, to index a CSV or TSV file, the file is split into records, one per row, and each row is indexed as a separate document.

Import Data with the REST API

How-to

> Compatible versions: 4.2, 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion Server*, *Fusion*

It is often possible to get documents into Fusion Server by configuring a datasource with the appropriate connector.

- *Fusion 4.x Connectors*
- *Fusion 5.x Connectors*

But if there are obstacles to using connectors, it can be simpler to index documents with a REST API call to an index profile or pipeline.

Push documents to Fusion using index profiles

Index profiles allow you to send documents to a consistent endpoint (the profile alias) and change the backend index pipeline as needed. The profile is also a simple way to use one pipeline for multiple collections without any one collection "owning" the pipeline.

- *Fusion 4.x Index Profiles*
- *Fusion 5.x Index Profiles*

You can send documents directly to an index using the *Index REST API*. The request path is:

```
/api/apps/APP_NAME/index/INDEX_PROFILE
```

These requests are sent as a POST request. The request header specifies the format of the contents of the request body. Create an index profile in the Fusion UI.

To send a streaming list of JSON documents, you can send the JSON file that holds these objects to the API listed above with `application/json` as the content type. If your JSON file is a list or array of many items, the endpoint operates in a streaming way and indexes the docs as necessary.

Send data to an index profile that is part of an app

Accessing an index profile through an app lets a Fusion admin secure and manage all objects on a per-app basis. Security is then determined by whether a user can access an app. This is the recommended way to manage permissions in Fusion.

The syntax for sending documents to an index profile that is part of an app is as follows:

```
curl –u USERNAME:PASSWORD –X POST –H 'content–type: application/json'
https://FUSION_HOST:FUSION_PORT/api/apps/APP_NAME/index/INDEX_PROFILE
––data–binary @my–json–data.json
```

Note

Spaces in an app name become underscores. Spaces in an index profile name become hyphens.

To prevent the terminal from displaying all the data and metadata it indexes — useful if you are indexing a large file — you can optionally append `?echo=false` to the URL.

Be sure to set the content type header properly for the content being sent. Some frequently used content types are:

- Text: `application/json`, `application/xml`
- PDF documents: `application/pdf`

- MS Office:
    - DOCX: `application/vnd.openxmlformats-officedocument.wordprocessingml.document`
    - XLSX: `application/vnd.openxmlformats-officedocument.spreadsheetml.sheet`
    - PPTX: `application/vnd.vnd.openxmlformats-officedocument.presentationml.presentation`
    - More types: http://filext.com/faq/office_mime_types.php

Example: Send JSON data to an index profile under an app

In `$FUSION_HOME/apps/solr-dist/example/exampledocs` you can find a few sample documents. This example uses one of these, `books.json`.

To push JSON data to an index profile under an app:

1. Create an index profile. In the Fusion UI, click **Indexing > Index Profiles** and follow the prompts.
2. From the directory containing `books.json`, enter the following, substituting your values for username, password, and index profile name:

```
curl -u USERNAME:PASSWORD -X POST -H 'content-type:
application/json'
https://FUSION_HOST:FUSION_PORT/api/apps/APP_NAME/index/INDEX_PR
OFILE?echo=false --data-binary @books.json
```

3. Test that your data has made it into Fusion:
    a. Log into the Fusion UI.
    b. Navigate to the app where you sent your data.
    c. Navigate to the Query Workbench.
    d. Search for `*:*`.
    e. Select relevant Display Fields, for example `author` and `name`.

Example: Send JSON data without defining an app

In most cases it is best to delegate permissions on a per-app basis. But if your use case requires it, you can push data to Fusion without defining an app.

To send JSON data without app security, issue the following curl command:

```
curl -u USERNAME:PASSWORD -X POST -H 'content-type: application/json'
https://FUSION_HOST:FUSION_PORT/api/index/INDEX_PROFILE --data-binary
@my-json-data.json
```

Example: Send XML data to an index profile with an app

To send XML data to an app, use the following:

```
curl –u USERNAME:PASSWORD –X POST –H 'content–type: application/xml'
https://FUSION_HOST:FUSION_PORT/api/apps/APP_NAME/index/INDEX_PROFILE
––data–binary @my–xml–file.xml
```

In Fusion 5.3.0+, documents can be created on the fly using the *PipelineDocument* JSON notation.

Remove documents

Example 1

The following example removes content:

```
curl –u USERNAME:PASSWORD –X POST –H 'content–type:
application/vnd.lucidworks–document'
https://FUSION_HOST:FUSION_PORT/api/apps/APP_NAME/index/INDEX_PROFILE
––data–binary @del–json–data.json
```

Example 2

A more specific example removes data from `books.json`. To delete "The Lightning Thief" and "The Sea of Monsters" from the index, use their id values in the JSON file.

The `del–json–data.json` file to delete the two books:

```
[{ "id": "978–0641723445","commands": [{"name": "delete","params":
{}}]},{ "id": "978–1423103349","commands": [{"name":
"delete","params": {}}, {"name": "commit","params": {}}]}]
```

The `?echo=false` can be used to turn off the response to the terminal.

Example 3

Another example to delete items using the Push API is:

```
curl -u admin:XXX -X POST
'http://FUSION_HOST:FUSION_PORT/api/apps/APP/index/INDEX' -H 'Content-
Type: application/vnd.lucidworks-document' -d '[ { "id": "1663838589-
44", "commands": [ { "name": "delete", "params": {} }, { "name":
"commit", "params": {} } ] }, ... ]'
```

Send documents to an index pipeline

Although sending documents to an index profile is recommended, if your use case requires it, you can send documents directly to an index pipeline.

For more information about index pipeline REST API reference documentation, select the link for your Fusion release:

- *Fusion 4.x Index Pipelines API*
- *Fusion 5.x Index Pipelines API*

Specify a parser

When you push data to a pipeline, you can specify the name of the parser by adding a parserId querystring parameter to the URL. For example: `https://FUSION_HOST:FUSION_PORT/api/index-pipelines/INDEX_PIPELINE/collections/COLLECTION_NAME/index?parserId=PARSER`.

If you do not specify a parser, and you are indexing outside of an app (`https://FUSION_HOST:FUSION_PORT/api/index-pipelines/…`), then the `_system` parser is used.

If you do not specify a parser, and you are indexing in an app context (`https://FUSION_HOST:FUSION_PORT/api/apps/APP_NAME/index-pipelines/…`), then the parser with the same name as the app is used.

Indexing CSV Files

In the usual case, to index a CSV or TSV file, the file is split into records, one per row, and each row is indexed as a separate document.

Import Signals

How-to

> Compatible versions: 4.2
>
> Works with: *Fusion AI*, *Fusion Server*

Normally, signals are indexed as streaming data during the natural activity of users. This topic describes how to load historical signals data in batches, in Parquet format, using Spark shell.

Note

Fusion's performance may be affected during this resource-intensive operation. Be sure to allocate sufficient memory for the Spark, Solr, and connectors services.

How to load Parquet files in Spark shell

1. Customize the code below by replacing the following strings:
   - `path_of_folder`. The absolute path to the folder containing your Parquet files.
   - `collection_name_signals`. The name of the signals collection where you want to load these signals.
   - `localhost:9983/lwfusion/4.2.2/solr` - You can verify the correct path by going to the Solr console at `http://fusion_host:8983/solr/#/` and looking for the value of `DzkHost`.

```
val parquetFilePath = "path_of_folder" val signals =
spark.read.parquet(parquetFilePath) val collectionName =
"collection_name_signals" val zkhostName =
"localhost:9983/lwfusion/4.2.1/solr" var connectionMap =
Map("collection" -> collectionName, "zkhost" -> zkhostName,
"commit_within" -> "5000", "batch_size" -> "10000")
signals.write.format("solr").options(connectionMap).save()
```

   For information about `commit_within` and `batch_size`, see https://github.com/lucidworks/spark-solr#commit_within.
2. Launch the Spark shell:

```
https://FUSION_HOST:FUSION_PORT/bin/spark-shell
```

3. At the `scala>` prompt, enter paste mode:

```
:paste
```

4. Paste your modified code from step 1.
5. Exit paste mode by pressing `CTRL-d`.

6. When the operation is finished, navigate to **Collections** > **Collections Manager** to verify that the number of documents in the specified signals collection has increased as expected.

Import Solr Collections with the Parallel Bulk Loader

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

You can use the *Parallel Bulk Loader* to copy Solr collections from one collection to another. This is helpful for copying collections from a production environment to a testing or development environment and using real data in the development and testing process.

Parallel Bulk Loader Job Configuration

In the Managed Fusion UI:

1. Navigate to **Collections > Jobs**.
2. Click Add and select Parallel Bulk Loader from the menu.
3. Enter a name for your job, and enter `solr` as the format.
4. Set the parameter name and value for the collection you want to import.
    - Parameter name: `collection`
    - Parameter value: the name of the collection to import from
5. If the source collection is in a different Managed Fusion app, add an additional parameter name and value pair.
    - Parameter name: `zkHost`
    - Parameter value: the location of zkHost
6. Enter the output collection, or where you want to import the collection to.
    - Output collection: the name of the output collection
    - Send to index pipeline: `_system`
7. Save your job.

When you're ready, *run the job* or *schedule the job* to run at an interval of your choice.

Parallel Bulk Loader JSON Configuration

Use this sample JSON payload to load the Parallel Bulk Loader to the *Spark Jobs API*:

```
{ "type" : "parallel-bulk-loader", "id" : "copy_sigs_with_timestamp",
"format" : "solr", "readOptions" : [ { "key" : "collection", "value" :
"SOURCE_COLLECTION" }, { "key" : "query", "value" : "timestamp_tdt:
{$lastTimestamp*}* TO *]" } ], "outputCollection" :
"OUTPUT_COLLECTION", "timestampFieldName" : "timestamp_tdt",
"defineFieldsUsingInputSchema" : true }
```

Replace `SOURCE_COLLECTION` with the name of the collection being exported. Replace `OUTPUT_COLLECTION` with the name of the destination collection.

See *Parallel Bulk Loader* for full configuration options.

Use the following Spark Jobs API call to run the job. Replace `data.json` with your JSON payload file.

```
curl -X POST \ -u USERNAME:PASSWORD \
'https://FUSION_HOST:6764/api/apps/Documentation/spark/configurations'
\ -H 'accept: */*' \ -H 'Content-Type: application/json' \ -d
'data.json'
```

Import Solr Collections with the Parallel Bulk Loader

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion*

You can use the *Parallel Bulk Loader* to copy Solr collections from one collection to another. This is helpful for copying collections from a production environment to a testing or development environment and using real data in the development and testing process.

Parallel Bulk Loader Job Configuration

In the Fusion UI:

1. Navigate to **Collections > Jobs**.
2. Click Add and select Parallel Bulk Loader from the menu.
3. Enter a name for your job, and enter `solr` as the format.
4. Set the parameter name and value for the collection you want to import.
    - Parameter name: `collection`
    - Parameter value: the name of the collection to import from
5. If the source collection is in a different Fusion app, add an additional parameter name and value pair.
    - Parameter name: `zkHost`
    - Parameter value: the location of zkHost
6. Enter the output collection, or where you want to import the collection to.
    - Output collection: the name of the output collection
    - Send to index pipeline: `_system`
7. Save your job.

When you're ready, *run the job* or *schedule the job* to run at an interval of your choice.

Parallel Bulk Loader JSON Configuration

Use this sample JSON payload to load the Parallel Bulk Loader to the *Spark Jobs API*:

```
{ "type" : "parallel-bulk-loader", "id" : "copy_sigs_with_timestamp",
"format" : "solr", "readOptions" : [ { "key" : "collection", "value" :
"SOURCE_COLLECTION" }, { "key" : "query", "value" : "timestamp_tdt:
{$lastTimestamp*}* TO *]" } ], "outputCollection" :
"OUTPUT_COLLECTION", "timestampFieldName" : "timestamp_tdt",
"defineFieldsUsingInputSchema" : true }
```

Replace `SOURCE_COLLECTION` with the name of the collection being exported. Replace `OUTPUT_COLLECTION` with the name of the destination collection.

See *Parallel Bulk Loader* for full configuration options.

Use the following Spark Jobs API call to run the job. Replace `data.json` with your JSON payload file.

```
curl -X POST \ -u USERNAME:PASSWORD \
'https://FUSION_HOST:6764/api/apps/Documentation/spark/configurations'
\ -H 'accept: */*' \ -H 'Content-Type: application/json' \ -d
'data.json'
```

Import with the Bulk Loader

How-to

> Compatible versions: 4.2
>
> Works with: *Fusion AI*, *Fusion Server*

Create and run Parallel Bulk Loader jobs

Use the *Jobs manager* to create and run Parallel Bulk Loader jobs. You can also use the *Scheduler* to schedule jobs.

In the procedures, select **Parallel Bulk Loader** as the job type and *configure the job as needed*.

Configuration settings for the Parallel Bulk Loader job

This section provides configuration settings for the Parallel Bulk Loader job. Also see configuration properties in the *Jobs Configuration Reference*.

Read settings

Setting	Description
`format`	Unique identifier of the data source provider. Spark scans the job's `classpath` for a class named `DefaultSource` in the `<format>` package. For example, for the `solr` format, we provide the `solr.DefaultSource` class in our `spark-solr` repository :
`path` (optional)	Comma-delimited list of paths to load. Some data sources, such as parquet, require a path. Others, such as Solr, do not. Refer to the documentation for your data source to determine if you need to provide a path.
`readOptions`	Options passed to the Spark SQL data source to configure the read operation. Options differ for every data source. Refer to the specific data source documentation for more information.
`sparkConfig` (optional)	List of Spark configuration settings needed to run the Parallel Bulk Loader.

`shellOptions`	Behind the scenes, the Parallel Bulk Loader job submits a Scala script to the Fusion Spark shell. The `shellOptions` setting lets you pass any additional options needed by the Spark shell. The two most common options are `--packages` and `--repositories` :
	
	`--packages`
	
	Comma-separated list of Maven coordinates of JAR files to include on the driver and executor classpaths. Spark searches the local Maven repository, and then Maven central and any additional remote repositories given in the config. The format for the coordinates should be `groupId:artifactId:version` . The HBase example below demonstrates the use of the packages option for loading the `com.hortonworks:shc-core:1.1.1-2.1-s_2.11` package.
	
	Tip
	
	Use the **https://spark-packages.org/** site to find interesting packages to add to your Parallel Bulk Loader jobs.
	
	`--repositories`
	
	Comma-separated list of additional remote Maven repositories to search for the Maven coordinates given in the `packages` config setting. The *Index HBase tables* example below demonstrates the use of the `repositories` option for loading the `com.hortonworks:shc-core:1.1.1-2.1-s_2.11` package from the Hortonworks repository.

Setting	Description

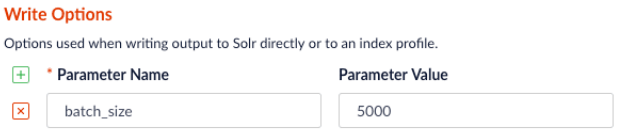
`timestampFieldName`	For datasources that support time-based filters, the Parallel Bulk Loader computes the timestamp of the last document written to Solr and the current timestamp of the Parallel Bulk Loader job. For example, the HBase data source lets you filter the read between a `MIN_STAMP` and `MAX_STAMP`, for example:
	
	`val timeRangeOpts = Map(HBaseRelation.MIN_STAMP -> minStamp.toString, HBaseRelation.MAX_STAMP -> maxStamp.toString)`
	
	This lets Parallel Bulk Loader jobs run on schedules, and pull only the newest rows from the underlying datasources.
	
	To support timestamp based filtering, the Parallel Bulk Loader provides two simple macros:
	
	`$lastTimestamp(format)`
	
	`$nowTimestamp(format)`
	
	The `format` argument is optional. If not supplied, then an ISO-8601 date/time string is used. The `timestampFieldName` setting is used to determine the value of `lastTimestamp`, using a Top 1 query to Solr to get the max timestamp. You can also pass `$lastTimestamp(EPOCH)` or `$lastTimestamp(EPOCH_MS)` to get the timestamp in seconds or milliseconds.
	
	See the *Index HBase tables* example below for an example of using this configuration property.

Transformation settings

Setting	Description
`transformScala`	Sometimes, you can write a small script to transform input data into the correct form for indexing. But at other times, you might need the full power of the Spark API to transform data into an indexable form. The `transformScala` option lets you filter and/or transform the input DataFrame any way you would like. You can even define UDFs to use during your transformation. For an example of using Scala to transform the input DataFrame before indexing in Solr, see the *Read from Parquet* example. Another powerful use of the `transformScala` option is that you can pull in advanced libraries, such as Spark NLP (from John Snow Labs) to do NLP work on your content before indexing. See the *Use NLP during indexing* example. Your Scala script can do other things but, at a minimum, it must define the following function that the Parallel Bulk Loader invokes: `def transform(inputDF: Dataset[Row]) : Dataset[Row] = { // do transformations and/or filter the inputDF here } Your script can rely on the following vals: spark: SparkSession sc: SparkContext fusionZKConn: ZKConnection // needed to access Fusion API solrCollection: SolrCollection // output collection jobId: Loader job config ID Also, the following classes have already been imported: import org.apache.spark.SparkContext._ import spark.implicits._ import spark.sql import org.apache.spark.sql.functions._ import com.lucidworks.spark.util.{SparkShellSupport => _lw} import com.lucidworks.spark.job.sql.SparkSQLLoader import com.lucidworks.spark.ml.recommenders.SolrCollection import com.lucidworks.spark.ZKConnection import org.apache.spark.sql.{Dataset, Row}`
`transformSql`	The `transformSql` option lets you write a SQL query to transform the input DataFrame. The SQL is executed after the `transformScala` script (if both are defined). The input DataFrame is exposed to your SQL as the `_input` view. See the *Clean up data with SQL transformations* example below for an example of using SQL to transform the input before indexing in Solr. This option also lets you leverage the UDF/UDAF functions provided by Spark SQL.

Setting	Description
`mlModelId`	If you have a Spark ML PipelineModel loaded into the blob store, you can supply the blob ID to the Parallel Bulk Loader and it will: 1. Load the model from the blob store. 2. Transform the input DataFrame (after the Scala transform but before the SQL transform). 3. Add the predicted output field (specified in the model metadata stored in the blob store) to the projected fields list. This lets you use Spark ML models to make predictions in a more scalable, performant manner than what can be achieved with a Machine Learning index stage.

Output settings

Setting	Description
`outputCollection`	Name of the Fusion collection to write to. The Parallel Bulk Loader uses the Collections API to resolve the underlying Solr collection at runtime.
`outputIndexPipeline`	Name of a Fusion index pipeline to which to send documents, instead of directly indexing to Solr. This option lets you perform additional ETL (extract, transform, and load) processing on the documents before they are indexed in Solr. If you need to write to time-partitioned indexes, then you must use an index pipeline, because writing directly to Solr is not partition aware.
`defineFieldsUsingInputSchema`	Flag to indicate if the Parallel Bulk Loader should use the input schema to create fields in Solr, after applying the Scala and/or SQL transformations. If `false`, then the Parallel Bulk Loader relies on the Fusion index pipeline and/or Solr field guessing to create the fields. If `true`, only fields that do not exist already in Solr are created. Consequently, if there is a type mismatch between an existing field in Solr and the input schema, you will need to use a transformation to rename the field in the input schema.
`clearDatasource`	If checked, the Parallel Bulk Loader deletes any existing documents in the output collection that match the query `_lw_loader_id_s:<JOB>`. Consequently, the Parallel Bulk Loader adds two metadata fields to each row: `_lw_loader_id_s` and `_lw_loader_job_id_s`.
`atomicUpdates`	Flag to send documents directly to Solr as atomic updates instead of as new documents. This option is not supported when using an index profile. Also note that the Parallel Bulk Loader tracking fields `_lw_loader_id_s` and `_lw_loader_job_id_s` are not sent when using atomic updates, so the clear datasource option does not work with documents created using atomic updates.
`outputOptions`	Options used when writing directly to Solr. See Spark-Solr: https://github.com/lucidworks/spark-solr#index-parameters For example, if your docs are relatively small, you might want to increase the `batch_size` (2000 default) as shown below: 
`outputPartitions`	Coalesce the DataFrame into N partitions before writing to Solr. This can help spread the indexing work out across more executors that are available in Spark, or limit the parallelism when writing to Solr.

Tune performance

As the name of the Parallel Bulk Loader job implies, it is designed to ingest large amounts of data into Fusion by parallelizing the work across your Spark cluster. To achieve scalability, you might need to increase the amount of memory and/or CPU resources allocated to the job.

By default, *Fusion's Spark configuration settings* control the resources allocated to Parallel Bulk Loader jobs.

You can pass these properties in the job configuration to override the default Spark shell options:

Parameter Name	Description and Default
`--driver-cores`	Cores for the driver Default: `1`
`--driver-memory`	Memory for the driver (for example, `1000M` or `2G`) Default: `1024M`
`--executor-cores`	Cores per executor Default: 1 in YARN mode, or all available cores on the worker in standalone mode
`--executor-memory`	Memory per executor (for example, `1000M` or `2G`) Default: `1G`
`--total-executor-cores`	Total cores for all executors Default: Without setting this parameter, the total cores for all executors is the number of executors in YARN mode, or all available cores on all workers in standalone mode.

Examples

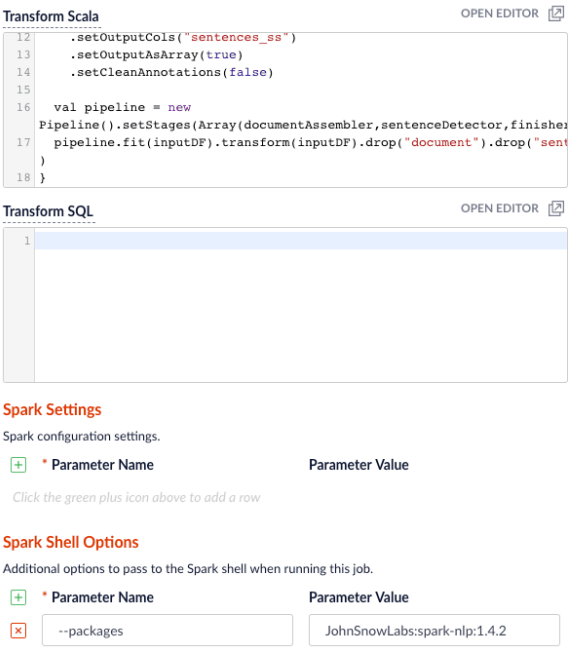
Here we provide screenshots and example JSON job definitions to illustrate key points about how to load from different data sources.

Use NLP during indexing

In this example, we leverage the John Snow labs NLP library during indexing. This is just quick-and-dirty to show the concept.

Also see:

- https://github.com/JohnSnowLabs/spark-nlp
- https://databricks.com/blog/2017/10/19/introducing-natural-language-processing-library-apache-spark.html



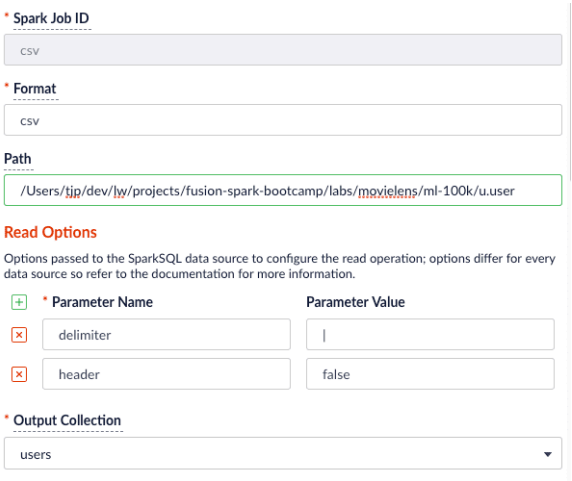
Use this transform Scala script:

```
import com.johnsnowlabs.nlp._ import com.johnsnowlabs.nlp.annotators._
import org.apache.spark.ml.Pipeline import
com.johnsnowlabs.nlp.annotators.sbd.pragmatic.SentenceDetector def
transform(inputDF: Dataset[Row]) : Dataset[Row] = { val
documentAssembler = new
DocumentAssembler().setInputCol("plot_txt_en").setOutputCol("document"
) val sentenceDetector = new
SentenceDetector().setInputCols(Array("document")).setOutputCol("sente
nces") val finisher = new Finisher() .setInputCols("sentences")
.setOutputCols("sentences_ss") .setOutputAsArray(true)
.setCleanAnnotations(false) val pipeline = new
Pipeline().setStages(Array(documentAssembler,sentenceDetector,finisher
))
pipeline.fit(inputDF).transform(inputDF).drop("document").drop("senten
ces") }
```

Be sure to add the `JohnSnowLabs:spark-nlp:1.4.2` package using Spark Shell Options.

Clean up data with SQL transformations

Fusion has a Local Filesystem connector that can handle files such as CSV and JSON files. Using the Parallel Bulk Loader lets you leverage features that are not in the Local Filesystem connector, such as using SQL to clean up the input data.



Use the following SQL to clean up the input data before indexing:

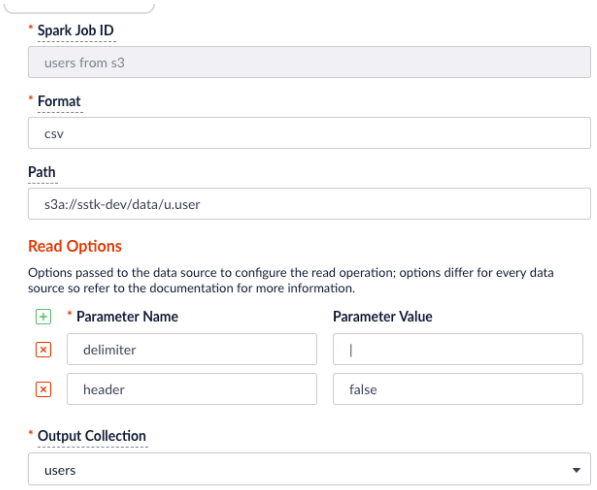
```
SELECT _c0 as user_id, CAST(_c1 as INT) as age, _c2 as gender, _c3 as
occupation, _c4 as zip_code FROM _input Job JSON: { "type" :
"parallel-bulk-loader", "id" : "csv", "format" : "csv", "path" :
"/Users/tjp/dev/lw/projects/fusion-spark-bootcamp/labs/movielens/ml-
100k/u.user", "readOptions" : [ { "key" : "delimiter", "value" : "|"
}, { "key" : "header", "value" : "false" } ], "outputCollection" :
"users", "clearDatasource" : false, "defineFieldsUsingInputSchema" :
true, "atomicUpdates" : false, "transformSql" : "SELECT _c0 as
user_id, \n CAST(_c1 as INT) as age, \n _c2 as gender,\n _c3 as
occupation,\n _c4 as zip_code \n FROM _input" }
```

Read from S3

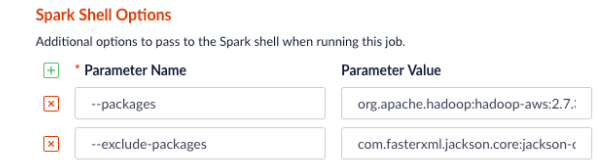
It is easy to read from an S3 bucket without pulling data down to your local workstation first. To avoid exposing your AWS credentials, add them to a file named `core-site.xml` in the `apps/spark-dist/conf` directory, such as:

```
<configuration> <property> <name>fs.s3a.access.key</name> <value>???
</value> </property> <property> <name>fs.s3a.secret.key</name>
<value>???</value> </property> </configuration>
```

Then you can load files using the S3a protocol, such as: `s3a://sstk-dev/data/u.user`. If you are running a Fusion cluster then each instance of Fusion will need a `core-site.xml` file. S3a is the preferred protocol for reading data into Spark because it uses Amazon's libraries to read from S3 instead of the legacy Hadoop libraries. If you need other S3 protocols (for example, s3 or s3n) you will need to add the equivalent properties to `core-site.xml`.



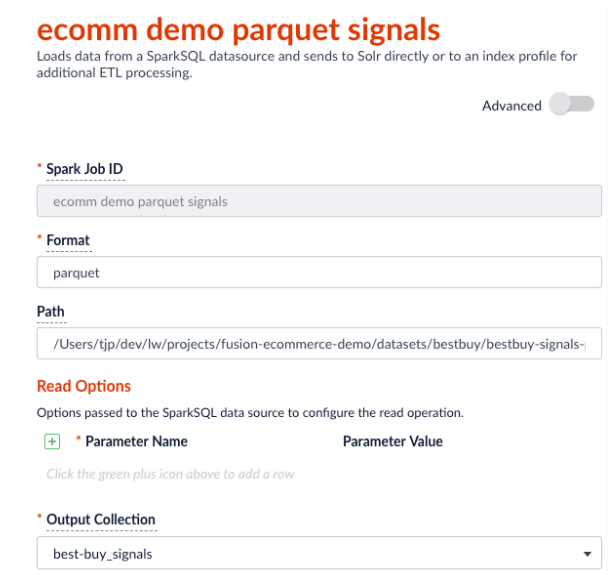
You will need to add the `org.apache.hadoop:hadoop-aws:2.7.3` package to the job using the `--packages` Spark option. Also, you will need to exclude the `com.fasterxml.jackson.core:jackson-core,joda-time:joda-time` packages using the `--exclude-packages` option.



You can also read from Google Cloud Storage (GCS), but you will need a few more properties in your `core-site.xml`; see Installing the Cloud Storage connector.

Read from Parquet

Reading from parquet files is built into Spark using the "parquet" format. For additional read options, see Configuration of Parquet.



Job JSON:

```
{ "type" : "parallel-bulk-loader", "id" : "ecomm demo parquet
signals", "format" : "parquet", "path" : "./part-00000-c1951958-98ae-
4f2a-b7b4-2e3a69fcf403-c000.snappy.parquet", "outputCollection" :
"best-buy_signals", "clearDatasource" : false,
"defineFieldsUsingInputSchema" : true, "atomicUpdates" : false }
```

This example also uses the **`transformScala`** option to filter and transform the input DataFrame into a better form for indexing using the following Scala script:

```
import java.util.Calendar import java.util.Locale import
java.util.TimeZone def transform(inputDF: Dataset[Row]) : Dataset[Row]
= { // do transformations and/or filter the inputDF here val signalsDF
= inputDF.filter((unix_timestamp($"timestamp_tdt", "MM/dd/yyyy
HH:mm:ss.SSS") < 1325376000)) val now = System.currentTimeMillis() val
maxDate = signalsDF.agg(max("timestamp_tdt")).take(1)
(0).getAs[java.sql.Timestamp](0).getTime val diff = now - maxDate val
add_time = udf((t: java.sql.Timestamp, diff : Long) => new
java.sql.Timestamp(t.getTime + diff)) val day_of_week = udf((t:
java.sql.Timestamp) => { val calendar =
Calendar.getInstance(TimeZone.getTimeZone("UTC"))
calendar.setTimeInMillis(t.getTime)
calendar.getDisplayName(Calendar.DAY_OF_WEEK, Calendar.LONG,
Locale.getDefault) }) //Remap some columns to bring the timestamps
current signalsDF .withColumnRenamed("timestamp_tdt",
"orig_timestamp_tdt").withColumn("timestamp_tdt",
add_time($"orig_timestamp_tdt", lit(diff))) .withColumn("date",
$"timestamp_tdt") .withColumn("tx_timestamp_txt",
date_format($"timestamp_tdt", "E YYYY-MM-d HH:mm:ss.SSS Z"))
.withColumn("param.query_time_dt", $"timestamp_tdt")
.withColumn("date_day", date_format(date_sub($"date", 0), "YYYY-MM-
d'T'HH:mm:ss.SSS'Z'")) .withColumn("date_month",
date_format(trunc($"date", "mm"), "YYYY-MM-d'T'HH:mm:ss.SSS'Z'"))
.withColumn("date_year", date_format(trunc($"date", "yyyy"), "YYYY-MM-
d'T'HH:mm:ss.SSS'Z'")) .withColumn("day_of_week",
day_of_week($"date")) }
```

Read from JDBC tables

You can use the Parallel Bulk Loader to parallelize reads from JDBC tables, if the tables have numeric columns that can be partitioned into relatively equal partition sizes. In the example below, we partition the employees table into 4 partitions using the `emp_no` column (`int`). Behind the scenes, Spark sends four separate queries to the database and processes the result sets in parallel.

LOAD THE JDBC DRIVER JAR FILE INTO THE BLOB STORE

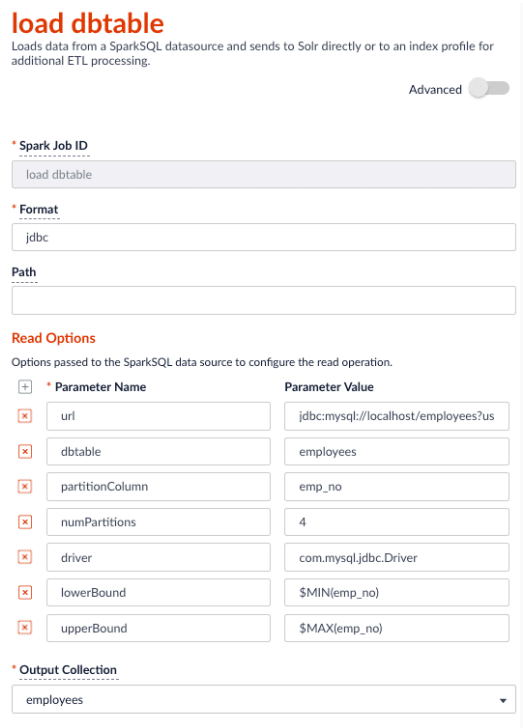
Before you ingest from a JDBC data source, you need to use the Fusion Admin UI to *upload the JDBC driver JAR file into the blob store.*

Alternatively, you can add the JAR file to the Fusion blob store with `resourceType=spark:jar`; for example:

```
curl -XPUT -H "Content-type:application/octet-stream"
"http://<FUSION_HOST>/api/blobs/mysql_jdbc_jar?resourceType=spark:jar"
--data-binary @mysql-connector-java-5.1.45-bin.jar
```

At runtime, Fusion's Spark job management framework knows how to add any JAR files with `resourceType=spark:jar` from the blob store to the appropriate classpaths before running a Parallel Bulk Loader job.

READ FROM A TABLE



For more information on reading from JDBC-compliant databases, see:

- http://spark.apache.org/docs/latest/sql-programming-guide.html#jdbc-to-other-databases
- https://medium.com/@radek.strnad/tips-for-using-jdbc-in-apache-spark-sql-396ea7b2e3d3

Job JSON:

```
{ "type" : "parallel-bulk-loader", "id" : "load dbtable", "format" :
"jdbc", "readOptions" : [ { "key" : "url", "value" :
"jdbc:mysql://localhost/employees?user=?&password=?" }, { "key" :
"dbtable", "value" : "employees" }, { "key" : "partitionColumn",
"value" : "emp_no" }, { "key" : "numPartitions", "value" : "4" }, {
"key" : "driver", "value" : "com.mysql.jdbc.Driver" }, { "key" :
"lowerBound", "value" : "$MIN(emp_no)" }, { "key" : "upperBound",
"value" : "$MAX(emp_no)" } ], "outputCollection" : "employees",
"clearDatasource" : false, "defineFieldsUsingInputSchema" : true,
"atomicUpdates" : false }
```

Notice the use of the `$MIN(emp_no)` and `$MAX(emp_no)` macros in the read options. These are macros offered by the Parallel Bulk Loader to help configure parallel reads of JDBC tables. Behind the scenes, the macros are translated into SQL queries to get the MAX and MIN values of the specified field, which Spark uses to compute splits for partitioned queries. As mentioned above, the field must be numeric and must have a relatively balanced distribution of values between MAX and MIN; otherwise, you are unlikely to see much performance benefit to partitioning.

Index HBase tables

To index an HBase table, use the Hortonworks connector .

Note

The Parallel Bulk Loader lets us replace the HBase Indexer.

You will need to add an `hbase-site.xml` (and possibly `core-site.xml`) to `apps/spark-dist/conf` in Fusion, for example:

```
<configuration> <property>
<name>hbase.defaults.for.version.skip</name> <value>true</value>
</property> <property> <name>hbase.zookeeper.quorum</name>
<value>localhost:2181</value> </property> <property>
<name>zookeeper.znode.parent</name> <value>/hbase</value> </property>
</configuration>
```

For this example, we will create a test table in HBase. If you already have a table in HBase, feel free to use that table instead.

1. Launch the HBase shell and create a table named `fusion_nums` with a single column family named `lw`:

```
create 'fusion_nums', 'lw'
```

2. Do a list command to see your table:

```
hbase(main):002:0> list TABLE fusion_nums 1 row(s) in 0.0250
seconds => ["fusion_nums"]
```

3. Fill the table with some data:

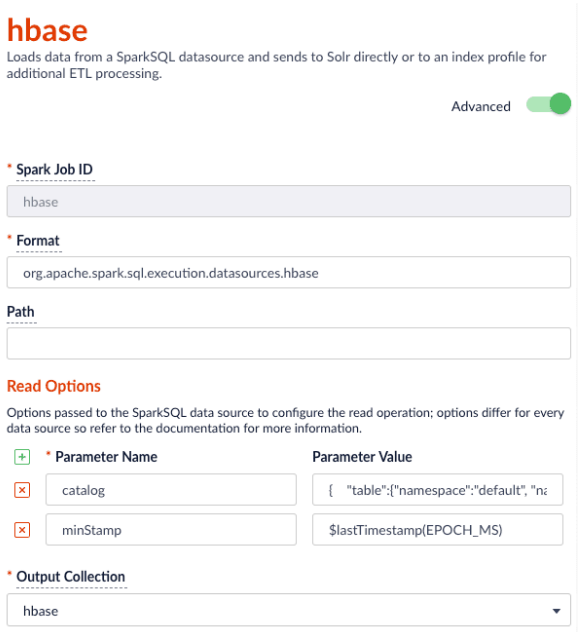
```
for i in '1'..'100' do for j in '1'..'2' do put 'fusion_nums',
"row#{i}", "lw:c#{j}", "#{i}#{j}" end end
```

4. Scan the fusion_nums table to see your data:

```
scan 'fusion_nums'
```

The HBase connector requires a catalog read option that defines the columns you want to read and how to map them into a Spark DataFrame. For our sample table, the following suffices:

```
{ "table":{"namespace":"default", "name":"fusion_nums"},
"rowkey":"key", "columns":{ "id":{"cf":"rowkey", "col":"key",
"type":"string"}, "lw_c1_s":{"cf":"lw", "col":"c1", "type":"string"},
"lw_c2_s":{"cf":"lw", "col":"c2", "type":"string"} } }
```



Notice the use of the `$lastTimestamp` macro in the read options. This lets us filter rows read from HBase using the timestamp of the last document the Parallel Bulk Loader wrote to Solr, that is, to get the newest updates from HBase only (incremental updates). Most Spark data sources provide a way to filter results based on timestamp.

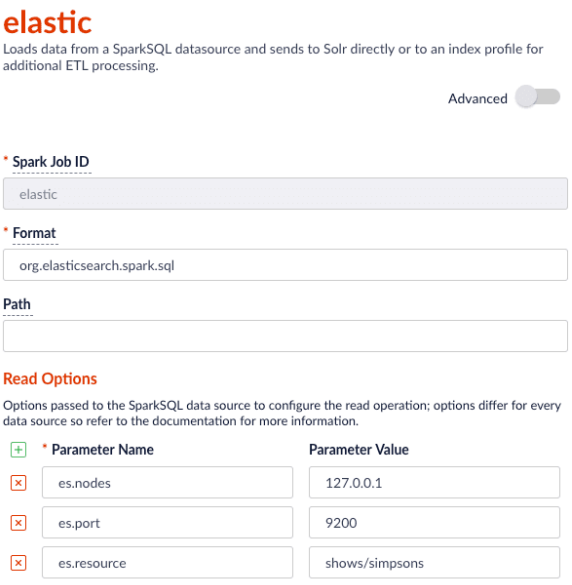
Job JSON:

```
{ "type" : "parallel-bulk-loader", "id" : "hbase", "format" :
"org.apache.spark.sql.execution.datasources.hbase", "readOptions" : [
{ "key" : "catalog", "value" : "{    \"table\":
{\"namespace\":\"default\", \"name\":\"fusion_nums\"},
\"rowkey\":\"key\",    \"columns\":{      \"id\":{\"cf\":\"rowkey\",
\"col\":\"key\", \"type\":\"string\"},      \"lw_c1_s\":
{\"cf\":\"lw\", \"col\":\"c1\", \"type\":\"string\"},
\"lw_c2_s\":{\"cf\":\"lw\", \"col\":\"c2\", \"type\":\"string\"}    }
}" }, { "key" : "minStamp", "value" : "$lastTimestamp(EPOCH_MS)" } ],
"outputCollection" : "hbase", "timestampFieldName" : "timestamp_tdt",
"clearDatasource" : false, "defineFieldsUsingInputSchema" : true,
"atomicUpdates" : false, "shellOptions" : [ { "key" : "--packages",
"value" : "com.hortonworks:shc-core:1.1.1-2.1-s_2.11" }, { "key" : "--
repositories", "value" :
"http://repo.hortonworks.com/content/repositories/releases" } ] }
```

Index Elastic data

With Elasticsearch 6.2.2 using the `org.elasticsearch:elasticsearch-spark-20_2.11:6.2.1` package, here is a Scala script to run in `bin/spark-shell` to index some test data:

```
import spark.implicits._ case class SimpsonCharacter(name: String,
actor: String, episodeDebut: String) val simpsonsDF = sc.parallelize(
SimpsonCharacter("Homer", "Dan Castellaneta", "Good Night") ::
SimpsonCharacter("Marge", "Julie Kavner", "Good Night") ::
SimpsonCharacter("Bart", "Nancy Cartwright", "Good Night") ::
SimpsonCharacter("Lisa", "Yeardley Smith", "Good Night") ::
SimpsonCharacter("Maggie", "Liz Georges and more", "Good Night") ::
SimpsonCharacter("Sideshow Bob", "Kelsey Grammer", "The Telltale
Head") :: Nil).toDF() val writeOpts = Map("es.nodes" -> "127.0.0.1",
"es.port" -> "9200", "es.index.auto.create" -> "true",
"es.resouce.auto.create" -> "shows/simpsons")
simpsonsDF.write.format("org.elasticsearch.spark.sql").mode("Overwrite
").save("shows/simpsons")
```



Job JSON:

```
{ "type" : "parallel-bulk-loader", "id" : "elastic", "format" :
"org.elasticsearch.spark.sql", "readOptions" : [ { "key" : "es.nodes",
"value" : "127.0.0.1" }, { "key" : "es.port", "value" : "9200" }, {
"key" : "es.resource", "value" : "shows/simpsons" } ],
"outputCollection" : "hbase_signals_aggr", "clearDatasource" : false,
"defineFieldsUsingInputSchema" : true, "atomicUpdates" : false,
"shellOptions" : [ { "key" : "--packages", "value" :
"org.elasticsearch:elasticsearch-spark-20_2.11:6.2.2" } ] }
```

Read from Couchbase

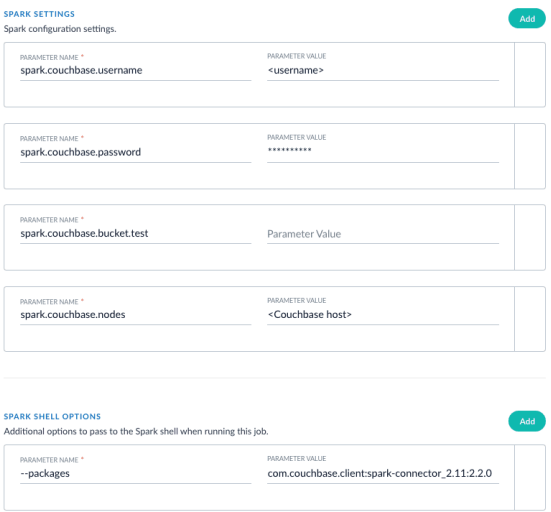
To index a Couchbase bucket, use the official Couchbase Spark connector found here.

For example, we will create a test bucket in Couchbase. If you already have a bucket in Couchbase, feel free to use that and skip to the test data setup section. This test was performed using Couchbase Server 6.0.0.

1. Create a bucket test in the Couchbase admin UI. Give access to a system account user to use in the Parallel Bulk Loader job config.
2. Connect to Couchbase using the command line client cbq. For example, `cbq -e=http://<host>:8091 -u <user> -p <password>`. Ensure the provided user is an authorized user of the test bucket.
3. Create a primary index on the test bucket: `CREATE PRIMARY INDEX 'test-primary-index' ON 'test' USING GSI;`.
4. Insert some data: `INSERT INTO 'test' ( KEY, VALUE ) VALUES ( "1", { "id": "01", "field1": "a value", "field2": "another value"} ) RETURNING META().id as docid, *;`.
5. Verify you can query the document just inserted: `select * from 'test';`.

To ingest from this bucket with the Parallel Bulk Loader, use the Couchbase Spark connector by specifying the format `com.couchbase.spark.sql.DefaultSource`. Then specify the `com.couchbase.client:spark-connector_2.11:2.2.0` package as the `spark shell --packages` option, as well as a few spark settings that direct the connector to a particular Couchbase server and bucket to connect to using the provided credentials. See here for all of the available Spark configuration settings for the Couchbase Spark connector.

Putting it all together:



XML setup

XML is a supported format that requires settings for `format` and `--packages`. In addition, you must specify the filepath in the `readOptions` section. For example:

```
{ "type": "parallel-bulk-loader", "id": "pbl", "format":
"com.databricks.spark.xml", "path": "this-is-ignored", "readOptions":
[ { "key": "rowTag", "value": "tag_name_representing_record" }, {
"key": "path", "value": "/home/user/file.xml.gz" } ],
"outputCollection": "output_collection_name", "clearDatasource": true,
"defineFieldsUsingInputSchema": true, "atomicUpdates": false,
"transformScala": "", "shellOptions": [ { "key": "--packages",
"value": "com.databricks:spark-xml_2.11:0.5.0" } ], "cacheAfterRead":
false }
```

Import with the Bulk Loader

How-to

> Compatible versions: 5.5, 5.9, 5.10
>
> Works with: *Managed Fusion*

Create and run Parallel Bulk Loader jobs

Use the *Jobs manager* to create and run Parallel Bulk Loader jobs. You can also use the *Scheduler* to schedule jobs.

In the procedures, select **Parallel Bulk Loader** as the job type and configure the job using the following information.

Configuration settings for the Parallel Bulk Loader job

This section provides configuration settings for the Parallel Bulk Loader job. Also see configuration properties in the *Configuration reference*.

Read settings

Setting	Description
`format`	Unique identifier of the data source provider. Spark scans the job's `classpath` for a class named `DefaultSource` in the `<format>` package. For example, for the `solr` format, we provide the `solr.DefaultSource` class in our `spark-solr` repository :
`path` (optional)	Comma-delimited list of paths to load. Some data sources, such as parquet, require a path. Others, such as Solr, do not. Refer to the documentation for your data source to determine if you need to provide a path.
`readOptions`	Options passed to the Spark SQL data source to configure the read operation. Options differ for every data source. Refer to the specific data source documentation for more information.
`sparkConfig` (optional)	List of Spark configuration settings needed to run the Parallel Bulk Loader.

`shellOptions`	Behind the scenes, the Parallel Bulk Loader job submits a Scala script to the Fusion Spark shell. The `shellOptions` setting lets you pass any additional options needed by the Spark shell. The two most common options are `--packages` and `--repositories` :
	
	`--packages`
	
	Comma-separated list of Maven coordinates of JAR files to include on the driver and executor classpaths. Spark searches the local Maven repository, and then Maven central and any additional remote repositories given in the config. The format for the coordinates should be `groupId:artifactId:version` . The HBase example below demonstrates the use of the packages option for loading the `com.hortonworks:shc-core:1.1.1-2.1-s_2.11` package.
	
	Tip
	
	Use the **https://spark-packages.org/** site to find interesting packages to add to your Parallel Bulk Loader jobs.
	
	`--repositories`
	
	Comma-separated list of additional remote Maven repositories to search for the Maven coordinates given in the `packages` config setting. The *Index HBase tables* example below demonstrates the use of the `repositories` option for loading the `com.hortonworks:shc-core:1.1.1-2.1-s_2.11` package from this repository .

Setting	Description

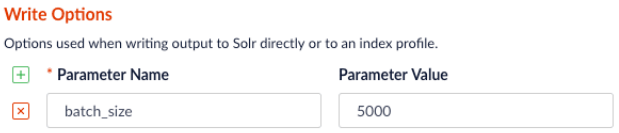
`timestampFieldName`	For datasources that support time-based filters, the Parallel Bulk Loader computes the timestamp of the last document written to Solr and the current timestamp of the Parallel Bulk Loader job. For example, the HBase data source lets you filter the read between a `MIN_STAMP` and `MAX_STAMP`, for example:
	
	`val timeRangeOpts = Map(HBaseRelation.MIN_STAMP -> minStamp.toString, HBaseRelation.MAX_STAMP -> maxStamp.toString)`
	
	This lets Parallel Bulk Loader jobs run on schedules, and pull only the newest rows from the underlying datasources.
	
	To support timestamp based filtering, the Parallel Bulk Loader provides two simple macros:
	
	`$lastTimestamp(format)`
	
	`$nowTimestamp(format)`
	
	The `format` argument is optional. If not supplied, then an ISO-8601 date/time string is used. The `timestampFieldName` setting is used to determine the value of `lastTimestamp`, using a Top 1 query to Solr to get the max timestamp. You can also pass `$lastTimestamp(EPOCH)` or `$lastTimestamp(EPOCH_MS)` to get the timestamp in seconds or milliseconds.
	
	See the *Index HBase tables* example below for an example of using this configuration property.

Transformation settings

Setting	Description
`transformScala`	Sometimes, you can write a small script to transform input data into the correct form for indexing. But at other times, you might need the full power of the Spark API to transform data into an indexable form. The `transformScala` option lets you filter and/or transform the input DataFrame any way you would like. You can even define UDFs to use during your transformation. For an example of using Scala to transform the input DataFrame before indexing in Solr, see the *Read from Parquet* example. Another powerful use of the `transformScala` option is that you can pull in advanced libraries, such as Spark NLP (from John Snow Labs) to do NLP work on your content before indexing. See the *Use NLP during indexing* example. Your Scala script can do other things but, at a minimum, it must define the following function that the Parallel Bulk Loader invokes: `def transform(inputDF: Dataset[Row]) : Dataset[Row] = { // do transformations and/or filter the inputDF here } Your script can rely on the following vals: spark: SparkSession sc: SparkContext fusionZKConn: ZKConnection // needed to access Fusion API solrCollection: SolrCollection // output collection jobId: Loader job config ID Also, the following classes have already been imported: import org.apache.spark.SparkContext._ import spark.implicits._ import spark.sql import org.apache.spark.sql.functions._ import com.lucidworks.spark.util.{SparkShellSupport => _lw} import com.lucidworks.spark.job.sql.SparkSQLLoader import com.lucidworks.spark.ml.recommenders.SolrCollection import com.lucidworks.spark.ZKConnection import org.apache.spark.sql.{Dataset, Row}`
`transformSql`	The `transformSql` option lets you write a SQL query to transform the input DataFrame. The SQL is executed after the `transformScala` script (if both are defined). The input DataFrame is exposed to your SQL as the `_input` view. See the *Clean up data with SQL transformations* example below for an example of using SQL to transform the input before indexing in Solr. This option also lets you leverage the UDF/UDAF functions provided by Spark SQL.

Setting	Description
`mlModelId`	If you have a Spark ML PipelineModel loaded into the blob store, you can supply the blob ID to the Parallel Bulk Loader and it will: 1. Load the model from the blob store. 2. Transform the input DataFrame (after the Scala transform but before the SQL transform). 3. Add the predicted output field (specified in the model metadata stored in the blob store) to the projected fields list. This lets you use Spark ML models to make predictions in a more scalable, performant manner than what can be achieved with a Machine Learning index stage.

Output settings

Setting	Description
`outputCollection`	Name of the Fusion collection to write to. The Parallel Bulk Loader uses the Collections API to resolve the underlying Solr collection at runtime.
`outputIndexPipeline`	Name of a Fusion index pipeline to which to send documents, instead of directly indexing to Solr. This option lets you perform additional ETL (extract, transform, and load) processing on the documents before they are indexed in Solr. If you need to write to time-partitioned indexes, then you must use an index pipeline, because writing directly to Solr is not partition aware.
`defineFieldsUsingInputSchema`	Flag to indicate if the Parallel Bulk Loader should use the input schema to create fields in Solr, after applying the Scala and/or SQL transformations. If `false`, then the Parallel Bulk Loader relies on the Fusion index pipeline and/or Solr field guessing to create the fields. If `true`, only fields that do not exist already in Solr are created. Consequently, if there is a type mismatch between an existing field in Solr and the input schema, you will need to use a transformation to rename the field in the input schema.
`clearDatasource`	If checked, the Parallel Bulk Loader deletes any existing documents in the output collection that match the query `_lw_loader_id_s:<JOB>`. Consequently, the Parallel Bulk Loader adds two metadata fields to each row: `_lw_loader_id_s` and `_lw_loader_job_id_s`.
`atomicUpdates`	Flag to send documents directly to Solr as atomic updates instead of as new documents. This option is not supported when using an index profile. Also note that the Parallel Bulk Loader tracking fields `_lw_loader_id_s` and `_lw_loader_job_id_s` are not sent when using atomic updates, so the clear datasource option does not work with documents created using atomic updates.
`outputOptions`	Options used when writing directly to Solr. See Spark-Solr: https://github.com/lucidworks/spark-solr#index-parameters For example, if your docs are relatively small, you might want to increase the `batch_size` (2000 default) as shown below: 
`outputPartitions`	Coalesce the DataFrame into N partitions before writing to Solr. This can help spread the indexing work out across more executors that are available in Spark, or limit the parallelism when writing to Solr.

Tune performance

As the name of the Parallel Bulk Loader job implies, it is designed to ingest large amounts of data into Fusion by parallelizing the work across your Spark cluster. To achieve scalability, you might need to increase the amount of memory and/or CPU resources allocated to the job.

You can pass these properties in the job configuration to override the default Spark shell options:

Parameter Name	Description and Default
`--driver-cores`	Cores for the driver Default: `1`
`--driver-memory`	Memory for the driver (for example, `1000M` or `2G`) Default: `1024M`
`--executor-cores`	Cores per executor Default: 1 in YARN mode, or all available cores on the worker in standalone mode
`--executor-memory`	Memory per executor (for example, `1000M` or `2G`) Default: `1G`
`--total-executor-cores`	Total cores for all executors Default: Without setting this parameter, the total cores for all executors is the number of executors in YARN mode, or all available cores on all workers in standalone mode.

Examples

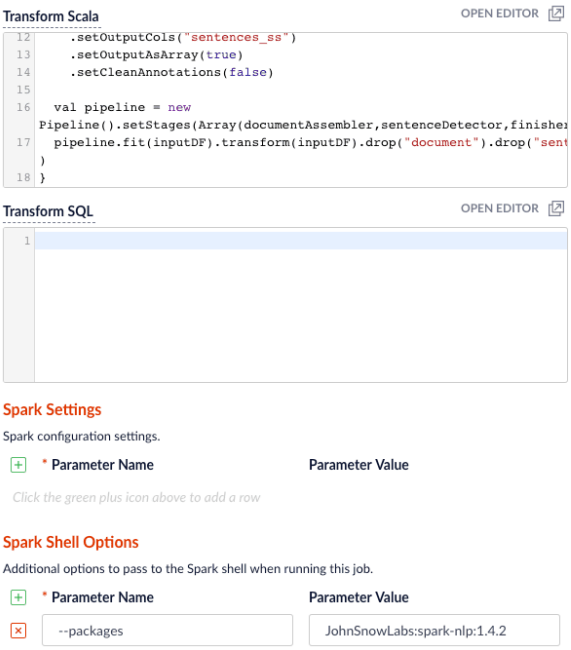
Here we provide screenshots and example JSON job definitions to illustrate key points about how to load from different data sources.

Use NLP during indexing

In this example, we leverage the John Snow labs NLP library during indexing. This is just quick-and-dirty to show the concept.

Also see:

- https://github.com/JohnSnowLabs/spark-nlp
- https://databricks.com/blog/2017/10/19/introducing-natural-language-processing-library-apache-spark.html



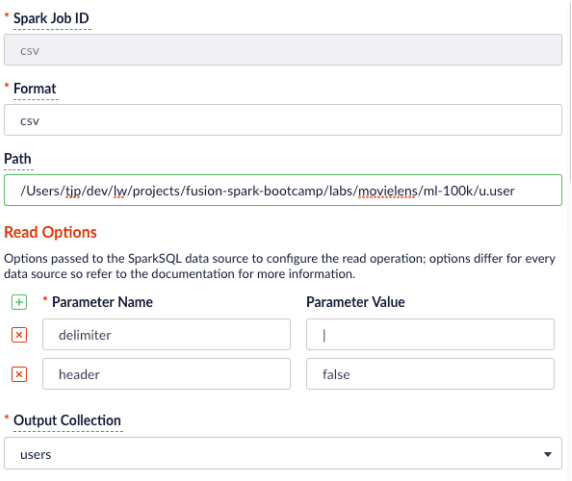
Use this transform Scala script:

```
import com.johnsnowlabs.nlp._ import com.johnsnowlabs.nlp.annotators._
import org.apache.spark.ml.Pipeline import
com.johnsnowlabs.nlp.annotators.sbd.pragmatic.SentenceDetector def
transform(inputDF: Dataset[Row]) : Dataset[Row] = { val
documentAssembler = new
DocumentAssembler().setInputCol("plot_txt_en").setOutputCol("document"
) val sentenceDetector = new
SentenceDetector().setInputCols(Array("document")).setOutputCol("sente
nces") val finisher = new Finisher() .setInputCols("sentences")
.setOutputCols("sentences_ss") .setOutputAsArray(true)
.setCleanAnnotations(false) val pipeline = new
Pipeline().setStages(Array(documentAssembler,sentenceDetector,finisher
))
pipeline.fit(inputDF).transform(inputDF).drop("document").drop("senten
ces") }
```

Be sure to add the `JohnSnowLabs:spark-nlp:1.4.2` package using Spark Shell Options.

Clean up data with SQL transformations

Fusion has a Local Filesystem connector that can handle files such as CSV and JSON files. Using the Parallel Bulk Loader lets you leverage features that are not in the Local Filesystem connector, such as using SQL to clean up the input data.



Use the following SQL to clean up the input data before indexing:

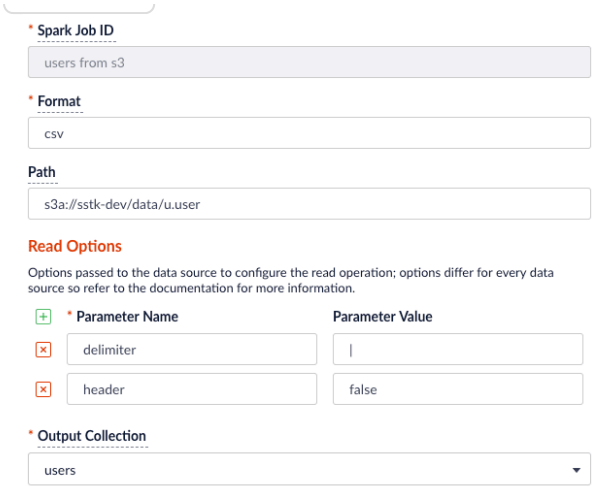
```
SELECT _c0 as user_id, CAST(_c1 as INT) as age, _c2 as gender, _c3 as
occupation, _c4 as zip_code FROM _input Job JSON: { "type" :
"parallel-bulk-loader", "id" : "csv", "format" : "csv", "path" :
"/Users/tjp/dev/lw/projects/fusion-spark-bootcamp/labs/movielens/ml-
100k/u.user", "readOptions" : [ { "key" : "delimiter", "value" : "|"
}, { "key" : "header", "value" : "false" } ], "outputCollection" :
"users", "clearDatasource" : false, "defineFieldsUsingInputSchema" :
true, "atomicUpdates" : false, "transformSql" : "SELECT _c0 as
user_id, \n CAST(_c1 as INT) as age, \n _c2 as gender,\n _c3 as
occupation,\n _c4 as zip_code \n FROM _input" }
```

Read from S3

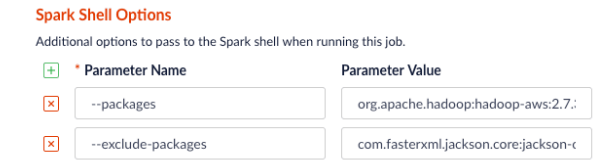
It is easy to read from an S3 bucket without pulling data down to your local workstation first. To avoid exposing your AWS credentials, add them to a file named `core-site.xml` in the `apps/spark-dist/conf` directory, such as:

```
<configuration> <property> <name>fs.s3a.access.key</name> <value>???
</value> </property> <property> <name>fs.s3a.secret.key</name>
<value>???</value> </property> </configuration>
```

Then you can load files using the S3a protocol, such as: `s3a://sstk-dev/data/u.user`. If you are running a Fusion cluster then each instance of Fusion will need a `core-site.xml` file. S3a is the preferred protocol for reading data into Spark because it uses Amazon's libraries to read from S3 instead of the legacy Hadoop libraries. If you need other S3 protocols (for example, s3 or s3n) you will need to add the equivalent properties to `core-site.xml`.



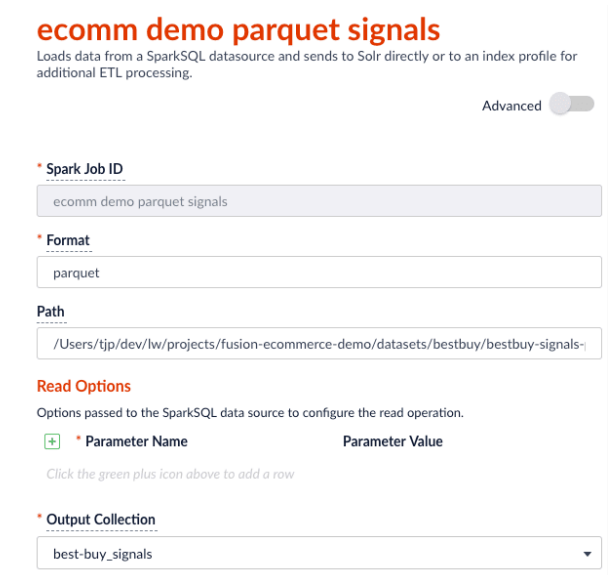
You will need to add the `org.apache.hadoop:hadoop-aws:2.7.3` package to the job using the `--packages` Spark option. Also, you will need to exclude the `com.fasterxml.jackson.core:jackson-core,joda-time:joda-time` packages using the `--exclude-packages` option.



You can also read from Google Cloud Storage (GCS), but you will need a few more properties in your `core-site.xml`; see Installing the Cloud Storage connector.

Read from Parquet

Reading from parquet files is built into Spark using the "parquet" format. For additional read options, see Configuration of Parquet.



Job JSON:

```
{ "type" : "parallel-bulk-loader", "id" : "ecomm demo parquet
signals", "format" : "parquet", "path" : "./part-00000-c1951958-98ae-
4f2a-b7b4-2e3a69fcf403-c000.snappy.parquet", "outputCollection" :
"best-buy_signals", "clearDatasource" : false,
"defineFieldsUsingInputSchema" : true, "atomicUpdates" : false }
```

This example also uses the **`transformScala`** option to filter and transform the input DataFrame into a better form for indexing using the following Scala script:

```
import java.util.Calendar import java.util.Locale import
java.util.TimeZone def transform(inputDF: Dataset[Row]) : Dataset[Row]
= { // do transformations and/or filter the inputDF here val signalsDF
= inputDF.filter((unix_timestamp($"timestamp_tdt", "MM/dd/yyyy
HH:mm:ss.SSS") < 1325376000)) val now = System.currentTimeMillis() val
maxDate = signalsDF.agg(max("timestamp_tdt")).take(1)
(0).getAs[java.sql.Timestamp](0).getTime val diff = now - maxDate val
add_time = udf((t: java.sql.Timestamp, diff : Long) => new
java.sql.Timestamp(t.getTime + diff)) val day_of_week = udf((t:
java.sql.Timestamp) => { val calendar =
Calendar.getInstance(TimeZone.getTimeZone("UTC"))
calendar.setTimeInMillis(t.getTime)
calendar.getDisplayName(Calendar.DAY_OF_WEEK, Calendar.LONG,
Locale.getDefault) }) //Remap some columns to bring the timestamps
current signalsDF .withColumnRenamed("timestamp_tdt",
"orig_timestamp_tdt").withColumn("timestamp_tdt",
add_time($"orig_timestamp_tdt", lit(diff))) .withColumn("date",
$"timestamp_tdt") .withColumn("tx_timestamp_txt",
date_format($"timestamp_tdt", "E YYYY-MM-d HH:mm:ss.SSS Z"))
.withColumn("param.query_time_dt", $"timestamp_tdt")
.withColumn("date_day", date_format(date_sub($"date", 0), "YYYY-MM-
d'T'HH:mm:ss.SSS'Z'")) .withColumn("date_month",
date_format(trunc($"date", "mm"), "YYYY-MM-d'T'HH:mm:ss.SSS'Z'"))
.withColumn("date_year", date_format(trunc($"date", "yyyy"), "YYYY-MM-
d'T'HH:mm:ss.SSS'Z'")) .withColumn("day_of_week",
day_of_week($"date")) }
```

Read from JDBC tables

You can use the Parallel Bulk Loader to parallelize reads from JDBC tables, if the tables have numeric columns that can be partitioned into relatively equal partition sizes. In the example below, we partition the employees table into 4 partitions using the `emp_no` column (`int`). Behind the scenes, Spark sends four separate queries to the database and processes the result sets in parallel.

LOAD THE JDBC DRIVER JAR FILE INTO THE BLOB STORE

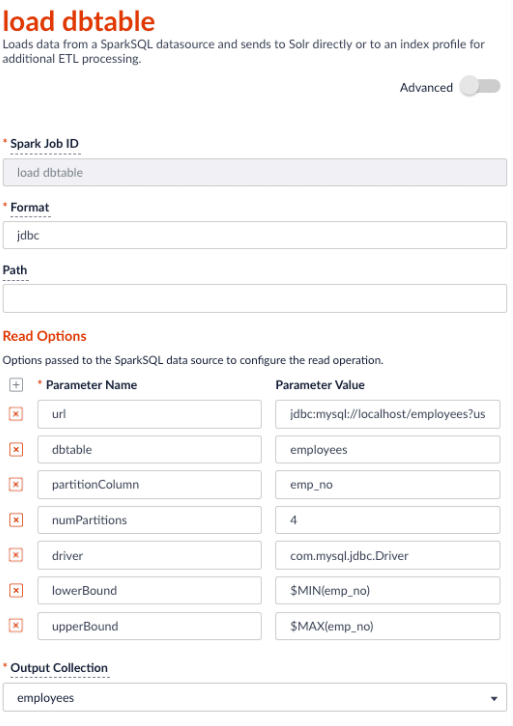
Before you ingest from a JDBC data source, you need to use the Fusion Admin UI to *upload the JDBC driver JAR file into the blob store.*

Alternatively, you can add the JAR file to the Fusion blob store with `resourceType=spark:jar`; for example:

```
curl -XPUT -H "Content-type:application/octet-stream"
"http://<FUSION_HOST>/api/blobs/mysql_jdbc_jar?resourceType=spark:jar"
--data-binary @mysql-connector-java-5.1.45-bin.jar
```

At runtime, Fusion's Spark job management framework knows how to add any JAR files with `resourceType=spark:jar` from the blob store to the appropriate classpaths before running a Parallel Bulk Loader job.

READ FROM A TABLE



For more information on reading from JDBC-compliant databases, see:

- http://spark.apache.org/docs/latest/sql-programming-guide.html#jdbc-to-other-databases
- https://medium.com/@radek.strnad/tips-for-using-jdbc-in-apache-spark-sql-396ea7b2e3d3

Job JSON:

```
{ "type" : "parallel-bulk-loader", "id" : "load dbtable", "format" :
"jdbc", "readOptions" : [ { "key" : "url", "value" :
"jdbc:mysql://localhost/employees?user=?&password=?" }, { "key" :
"dbtable", "value" : "employees" }, { "key" : "partitionColumn",
"value" : "emp_no" }, { "key" : "numPartitions", "value" : "4" }, {
"key" : "driver", "value" : "com.mysql.jdbc.Driver" }, { "key" :
"lowerBound", "value" : "$MIN(emp_no)" }, { "key" : "upperBound",
"value" : "$MAX(emp_no)" } ], "outputCollection" : "employees",
"clearDatasource" : false, "defineFieldsUsingInputSchema" : true,
"atomicUpdates" : false }
```

Notice the use of the `$MIN(emp_no)` and `$MAX(emp_no)` macros in the read options. These are macros offered by the Parallel Bulk Loader to help configure parallel reads of JDBC tables. Behind the scenes, the macros are translated into SQL queries to get the MAX and MIN values of the specified field, which Spark uses to compute splits for partitioned queries. As mentioned above, the field must be numeric and must have a relatively balanced distribution of values between MAX and MIN; otherwise, you are unlikely to see much performance benefit to partitioning.

Index HBase tables

To index an HBase table, use the Hortonworks connector .

Note

The Parallel Bulk Loader lets us replace the HBase Indexer.

You will need to add an `hbase-site.xml` (and possibly `core-site.xml`) to `apps/spark-dist/conf` in Fusion, for example:

```
<configuration> <property>
<name>hbase.defaults.for.version.skip</name> <value>true</value>
</property> <property> <name>hbase.zookeeper.quorum</name>
<value>localhost:2181</value> </property> <property>
<name>zookeeper.znode.parent</name> <value>/hbase</value> </property>
</configuration>
```

For this example, we will create a test table in HBase. If you already have a table in HBase, feel free to use that table instead.

1. Launch the HBase shell and create a table named `fusion_nums` with a single column family named `lw`:

```
create 'fusion_nums', 'lw'
```

2. Do a list command to see your table:

```
hbase(main):002:0> list TABLE fusion_nums 1 row(s) in 0.0250
seconds => ["fusion_nums"]
```

3. Fill the table with some data:

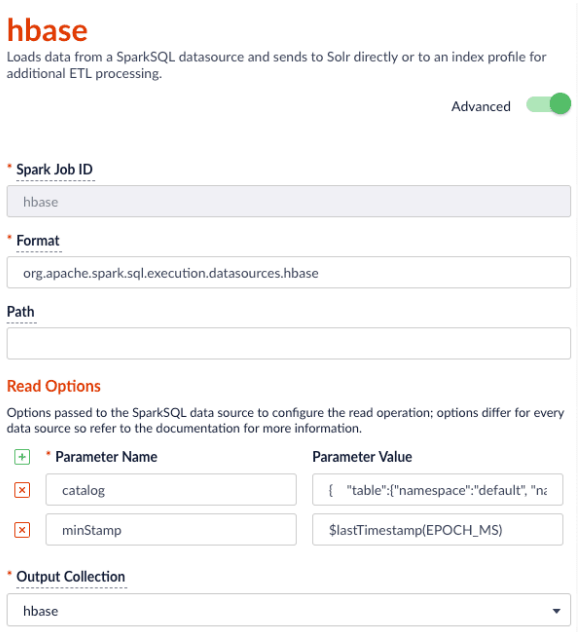
```
for i in '1'..'100' do for j in '1'..'2' do put 'fusion_nums',
"row#{i}", "lw:c#{j}", "#{i}#{j}" end end
```

4. Scan the fusion_nums table to see your data:

```
scan 'fusion_nums'
```

The HBase connector requires a catalog read option that defines the columns you want to read and how to map them into a Spark DataFrame. For our sample table, the following suffices:

```
{ "table":{"namespace":"default", "name":"fusion_nums"},
"rowkey":"key", "columns":{ "id":{"cf":"rowkey", "col":"key",
"type":"string"}, "lw_c1_s":{"cf":"lw", "col":"c1", "type":"string"},
"lw_c2_s":{"cf":"lw", "col":"c2", "type":"string"} } }
```



Notice the use of the `$lastTimestamp` macro in the read options. This lets us filter rows read from HBase using the timestamp of the last document the Parallel Bulk Loader wrote to Solr, that is, to get the newest updates from HBase only (incremental updates). Most Spark data sources provide a way to filter results based on timestamp.

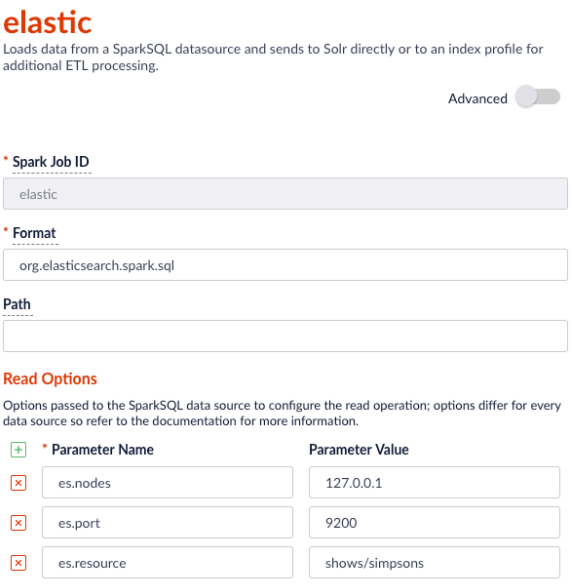
Job JSON:

```
{ "type" : "parallel-bulk-loader", "id" : "hbase", "format" :
"org.apache.spark.sql.execution.datasources.hbase", "readOptions" : [
{ "key" : "catalog", "value" : "{    \"table\":
{\"namespace\":\"default\", \"name\":\"fusion_nums\"},
\"rowkey\":\"key\",    \"columns\":{      \"id\":{\"cf\":\"rowkey\",
\"col\":\"key\", \"type\":\"string\"},      \"lw_c1_s\":
{\"cf\":\"lw\", \"col\":\"c1\", \"type\":\"string\"},
\"lw_c2_s\":{\"cf\":\"lw\", \"col\":\"c2\", \"type\":\"string\"}    }
}" }, { "key" : "minStamp", "value" : "$lastTimestamp(EPOCH_MS)" } ],
"outputCollection" : "hbase", "timestampFieldName" : "timestamp_tdt",
"clearDatasource" : false, "defineFieldsUsingInputSchema" : true,
"atomicUpdates" : false, "shellOptions" : [ { "key" : "--packages",
"value" : "com.hortonworks:shc-core:1.1.1-2.1-s_2.11" }, { "key" : "--
repositories", "value" :
"http://repo.hortonworks.com/content/repositories/releases" } ] }
```

Index Elastic data

With Elasticsearch 6.2.2 using the `org.elasticsearch:elasticsearch-spark-20_2.11:6.2.1` package, here is a Scala script to run in `bin/spark-shell` to index some test data:

```
import spark.implicits._ case class SimpsonCharacter(name: String,
actor: String, episodeDebut: String) val simpsonsDF = sc.parallelize(
SimpsonCharacter("Homer", "Dan Castellaneta", "Good Night") ::
SimpsonCharacter("Marge", "Julie Kavner", "Good Night") ::
SimpsonCharacter("Bart", "Nancy Cartwright", "Good Night") ::
SimpsonCharacter("Lisa", "Yeardley Smith", "Good Night") ::
SimpsonCharacter("Maggie", "Liz Georges and more", "Good Night") ::
SimpsonCharacter("Sideshow Bob", "Kelsey Grammer", "The Telltale
Head") :: Nil).toDF() val writeOpts = Map("es.nodes" -> "127.0.0.1",
"es.port" -> "9200", "es.index.auto.create" -> "true",
"es.resouce.auto.create" -> "shows/simpsons")
simpsonsDF.write.format("org.elasticsearch.spark.sql").mode("Overwrite
").save("shows/simpsons")
```



Job JSON:

```
{ "type" : "parallel-bulk-loader", "id" : "elastic", "format" :
"org.elasticsearch.spark.sql", "readOptions" : [ { "key" : "es.nodes",
"value" : "127.0.0.1" }, { "key" : "es.port", "value" : "9200" }, {
"key" : "es.resource", "value" : "shows/simpsons" } ],
"outputCollection" : "hbase_signals_aggr", "clearDatasource" : false,
"defineFieldsUsingInputSchema" : true, "atomicUpdates" : false,
"shellOptions" : [ { "key" : "--packages", "value" :
"org.elasticsearch:elasticsearch-spark-20_2.11:6.2.2" } ] }
```

Read from Couchbase

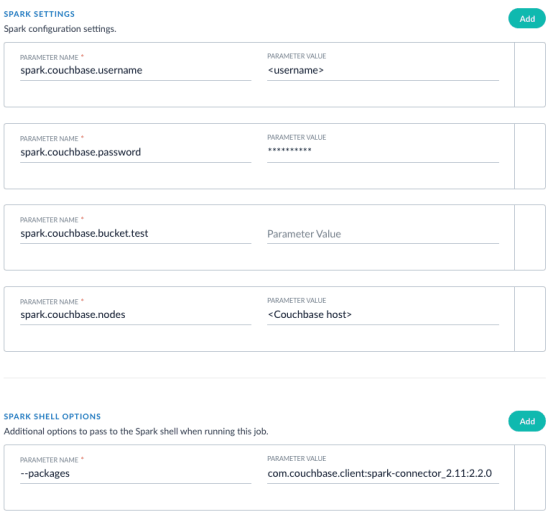
To index a Couchbase bucket, use the official Couchbase Spark connector found here.

For example, we will create a test bucket in Couchbase. If you already have a bucket in Couchbase, feel free to use that and skip to the test data setup section. This test was performed using Couchbase Server 6.0.0.

1. Create a bucket test in the Couchbase admin UI. Give access to a system account user to use in the Parallel Bulk Loader job config.
2. Connect to Couchbase using the command line client cbq. For example, `cbq -e=http://<host>:8091 -u <user> -p <password>`. Ensure the provided user is an authorized user of the test bucket.
3. Create a primary index on the test bucket: `CREATE PRIMARY INDEX 'test-primary-index' ON 'test' USING GSI;`.
4. Insert some data: `INSERT INTO 'test' ( KEY, VALUE ) VALUES ( "1", { "id": "01", "field1": "a value", "field2": "another value"} ) RETURNING META().id as docid, *;`.
5. Verify you can query the document just inserted: `select * from 'test';`.

To ingest from this bucket with the Parallel Bulk Loader, use the Couchbase Spark connector by specifying the format `com.couchbase.spark.sql.DefaultSource`. Then specify the `com.couchbase.client:spark-connector_2.11:2.2.0` package as the `spark shell --packages` option, as well as a few spark settings that direct the connector to a particular Couchbase server and bucket to connect to using the provided credentials. See here for all of the available Spark configuration settings for the Couchbase Spark connector.

Putting it all together:



XML setup

XML is a supported format that requires settings for `format` and `--packages`. In addition, you must specify the filepath in the `readOptions` section. For example:

```
{ "type": "parallel-bulk-loader", "id": "pbl", "format":
"com.databricks.spark.xml", "path": "this-is-ignored", "readOptions":
[ { "key": "rowTag", "value": "tag_name_representing_record" }, {
"key": "path", "value": "/home/user/file.xml.gz" } ],
"outputCollection": "output_collection_name", "clearDatasource": true,
"defineFieldsUsingInputSchema": true, "atomicUpdates": false,
"transformScala": "", "shellOptions": [ { "key": "--packages",
"value": "com.databricks:spark-xml_2.11:0.5.0" } ], "cacheAfterRead":
false }
```

Import with the Bulk Loader

How-to

> Compatible versions: 5.5, 5.9, 5.10
>
> Works with: *Fusion*

Create and run Parallel Bulk Loader jobs

Use the *Jobs manager* to create and run Parallel Bulk Loader jobs. You can also use the *Scheduler* to schedule jobs.

In the procedures, select **Parallel Bulk Loader** as the job type and configure the job using the following information.

Configuration settings for the Parallel Bulk Loader job

This section provides configuration settings for the Parallel Bulk Loader job. Also see configuration properties in the *Configuration reference*.

Read settings

Setting	Description
`format`	Unique identifier of the data source provider. Spark scans the job's `classpath` for a class named `DefaultSource` in the `<format>` package. For example, for the `solr` format, we provide the `solr.DefaultSource` class in our `spark-solr` repository :
`path` (optional)	Comma-delimited list of paths to load. Some data sources, such as parquet, require a path. Others, such as Solr, do not. Refer to the documentation for your data source to determine if you need to provide a path.
`readOptions`	Options passed to the Spark SQL data source to configure the read operation. Options differ for every data source. Refer to the specific data source documentation for more information.
`sparkConfig` (optional)	List of Spark configuration settings needed to run the Parallel Bulk Loader.

`shellOptions`	Behind the scenes, the Parallel Bulk Loader job submits a Scala script to the Fusion Spark shell. The `shellOptions` setting lets you pass any additional options needed by the Spark shell. The two most common options are `--packages` and `--repositories` :
	
	`--packages`
	
	Comma-separated list of Maven coordinates of JAR files to include on the driver and executor classpaths. Spark searches the local Maven repository, and then Maven central and any additional remote repositories given in the config. The format for the coordinates should be `groupId:artifactId:version` . The HBase example below demonstrates the use of the packages option for loading the `com.hortonworks:shc-core:1.1.1-2.1-s_2.11` package.
	
	Tip
	
	Use the **https://spark-packages.org/** site to find interesting packages to add to your Parallel Bulk Loader jobs.
	
	`--repositories`
	
	Comma-separated list of additional remote Maven repositories to search for the Maven coordinates given in the `packages` config setting. The *Index HBase tables* example below demonstrates the use of the `repositories` option for loading the `com.hortonworks:shc-core:1.1.1-2.1-s_2.11` package from this repository .

Setting	Description

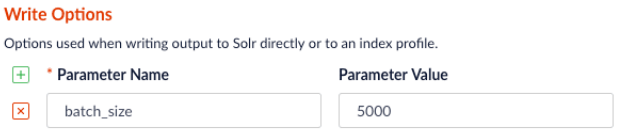
`timestampFieldName`	For datasources that support time-based filters, the Parallel Bulk Loader computes the timestamp of the last document written to Solr and the current timestamp of the Parallel Bulk Loader job. For example, the HBase data source lets you filter the read between a `MIN_STAMP` and `MAX_STAMP`, for example:
	
	`val timeRangeOpts = Map(HBaseRelation.MIN_STAMP -> minStamp.toString, HBaseRelation.MAX_STAMP -> maxStamp.toString)`
	
	This lets Parallel Bulk Loader jobs run on schedules, and pull only the newest rows from the underlying datasources.
	
	To support timestamp based filtering, the Parallel Bulk Loader provides two simple macros:
	
	`$lastTimestamp(format)`
	
	`$nowTimestamp(format)`
	
	The `format` argument is optional. If not supplied, then an ISO-8601 date/time string is used. The `timestampFieldName` setting is used to determine the value of `lastTimestamp`, using a Top 1 query to Solr to get the max timestamp. You can also pass `$lastTimestamp(EPOCH)` or `$lastTimestamp(EPOCH_MS)` to get the timestamp in seconds or milliseconds.
	
	See the *Index HBase tables* example below for an example of using this configuration property.

Transformation settings

Setting	Description
`transformScala`	Sometimes, you can write a small script to transform input data into the correct form for indexing. But at other times, you might need the full power of the Spark API to transform data into an indexable form. The `transformScala` option lets you filter and/or transform the input DataFrame any way you would like. You can even define UDFs to use during your transformation. For an example of using Scala to transform the input DataFrame before indexing in Solr, see the *Read from Parquet* example. Another powerful use of the `transformScala` option is that you can pull in advanced libraries, such as Spark NLP (from John Snow Labs) to do NLP work on your content before indexing. See the *Use NLP during indexing* example. Your Scala script can do other things but, at a minimum, it must define the following function that the Parallel Bulk Loader invokes: `def transform(inputDF: Dataset[Row]) : Dataset[Row] = { // do transformations and/or filter the inputDF here } Your script can rely on the following vals: spark: SparkSession sc: SparkContext fusionZKConn: ZKConnection // needed to access Fusion API solrCollection: SolrCollection // output collection jobId: Loader job config ID Also, the following classes have already been imported: import org.apache.spark.SparkContext._ import spark.implicits._ import spark.sql import org.apache.spark.sql.functions._ import com.lucidworks.spark.util.{SparkShellSupport => _lw} import com.lucidworks.spark.job.sql.SparkSQLLoader import com.lucidworks.spark.ml.recommenders.SolrCollection import com.lucidworks.spark.ZKConnection import org.apache.spark.sql.{Dataset, Row}`
`transformSql`	The `transformSql` option lets you write a SQL query to transform the input DataFrame. The SQL is executed after the `transformScala` script (if both are defined). The input DataFrame is exposed to your SQL as the `_input` view. See the *Clean up data with SQL transformations* example below for an example of using SQL to transform the input before indexing in Solr. This option also lets you leverage the UDF/UDAF functions provided by Spark SQL.

Setting	Description
`mlModelId`	If you have a Spark ML PipelineModel loaded into the blob store, you can supply the blob ID to the Parallel Bulk Loader and it will: 1. Load the model from the blob store. 2. Transform the input DataFrame (after the Scala transform but before the SQL transform). 3. Add the predicted output field (specified in the model metadata stored in the blob store) to the projected fields list. This lets you use Spark ML models to make predictions in a more scalable, performant manner than what can be achieved with a Machine Learning index stage.

Output settings

Setting	Description
`outputCollection`	Name of the Fusion collection to write to. The Parallel Bulk Loader uses the Collections API to resolve the underlying Solr collection at runtime.
`outputIndexPipeline`	Name of a Fusion index pipeline to which to send documents, instead of directly indexing to Solr. This option lets you perform additional ETL (extract, transform, and load) processing on the documents before they are indexed in Solr. If you need to write to time-partitioned indexes, then you must use an index pipeline, because writing directly to Solr is not partition aware.
`defineFieldsUsingInputSchema`	Flag to indicate if the Parallel Bulk Loader should use the input schema to create fields in Solr, after applying the Scala and/or SQL transformations. If `false`, then the Parallel Bulk Loader relies on the Fusion index pipeline and/or Solr field guessing to create the fields. If `true`, only fields that do not exist already in Solr are created. Consequently, if there is a type mismatch between an existing field in Solr and the input schema, you will need to use a transformation to rename the field in the input schema.
`clearDatasource`	If checked, the Parallel Bulk Loader deletes any existing documents in the output collection that match the query `_lw_loader_id_s:<JOB>`. Consequently, the Parallel Bulk Loader adds two metadata fields to each row: `_lw_loader_id_s` and `_lw_loader_job_id_s`.
`atomicUpdates`	Flag to send documents directly to Solr as atomic updates instead of as new documents. This option is not supported when using an index profile. Also note that the Parallel Bulk Loader tracking fields `_lw_loader_id_s` and `_lw_loader_job_id_s` are not sent when using atomic updates, so the clear datasource option does not work with documents created using atomic updates.
`outputOptions`	Options used when writing directly to Solr. See Spark-Solr: https://github.com/lucidworks/spark-solr#index-parameters For example, if your docs are relatively small, you might want to increase the `batch_size` (2000 default) as shown below: 
`outputPartitions`	Coalesce the DataFrame into N partitions before writing to Solr. This can help spread the indexing work out across more executors that are available in Spark, or limit the parallelism when writing to Solr.

Tune performance

As the name of the Parallel Bulk Loader job implies, it is designed to ingest large amounts of data into Fusion by parallelizing the work across your Spark cluster. To achieve scalability, you might need to increase the amount of memory and/or CPU resources allocated to the job.

For Spark configuration information, see *Spark operations*.

You can pass these properties in the job configuration to override the default Spark shell options:

Parameter Name	Description and Default
`--driver-cores`	Cores for the driver Default: `1`
`--driver-memory`	Memory for the driver (for example, `1000M` or `2G`) Default: `1024M`
`--executor-cores`	Cores per executor Default: 1 in YARN mode, or all available cores on the worker in standalone mode
`--executor-memory`	Memory per executor (for example, `1000M` or `2G`) Default: `1G`
`--total-executor-cores`	Total cores for all executors Default: Without setting this parameter, the total cores for all executors is the number of executors in YARN mode, or all available cores on all workers in standalone mode.

Examples

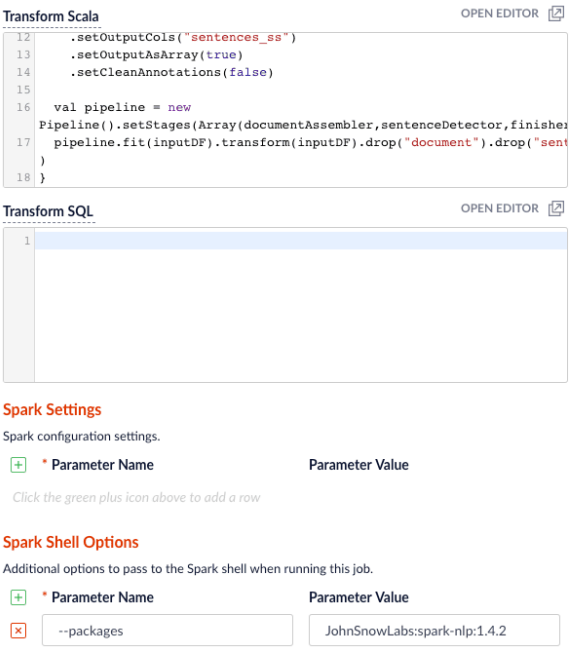
Here we provide screenshots and example JSON job definitions to illustrate key points about how to load from different data sources.

Use NLP during indexing

In this example, we leverage the John Snow labs NLP library during indexing. This is just quick-and-dirty to show the concept.

Also see:

- https://github.com/JohnSnowLabs/spark-nlp
- https://databricks.com/blog/2017/10/19/introducing-natural-language-processing-library-apache-spark.html



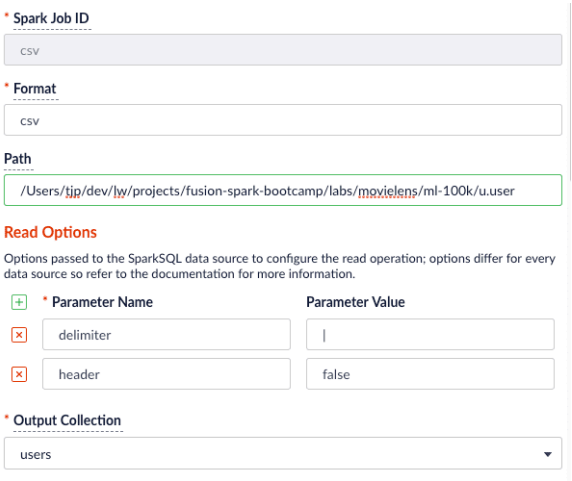
Use this transform Scala script:

```
import com.johnsnowlabs.nlp._ import com.johnsnowlabs.nlp.annotators._
import org.apache.spark.ml.Pipeline import
com.johnsnowlabs.nlp.annotators.sbd.pragmatic.SentenceDetector def
transform(inputDF: Dataset[Row]) : Dataset[Row] = { val
documentAssembler = new
DocumentAssembler().setInputCol("plot_txt_en").setOutputCol("document"
) val sentenceDetector = new
SentenceDetector().setInputCols(Array("document")).setOutputCol("sente
nces") val finisher = new Finisher() .setInputCols("sentences")
.setOutputCols("sentences_ss") .setOutputAsArray(true)
.setCleanAnnotations(false) val pipeline = new
Pipeline().setStages(Array(documentAssembler,sentenceDetector,finisher
))
pipeline.fit(inputDF).transform(inputDF).drop("document").drop("senten
ces") }
```

Be sure to add the `JohnSnowLabs:spark-nlp:1.4.2` package using Spark Shell Options.

Clean up data with SQL transformations

Fusion has a Local Filesystem connector that can handle files such as CSV and JSON files. Using the Parallel Bulk Loader lets you leverage features that are not in the Local Filesystem connector, such as using SQL to clean up the input data.



Use the following SQL to clean up the input data before indexing:

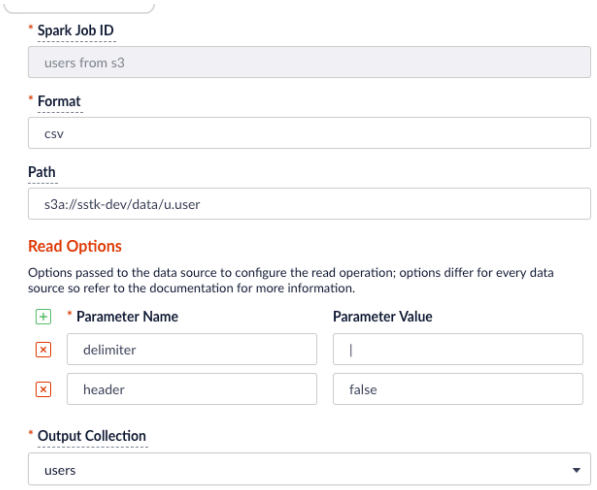
```
SELECT _c0 as user_id, CAST(_c1 as INT) as age, _c2 as gender, _c3 as
occupation, _c4 as zip_code FROM _input Job JSON: { "type" :
"parallel-bulk-loader", "id" : "csv", "format" : "csv", "path" :
"/Users/tjp/dev/lw/projects/fusion-spark-bootcamp/labs/movielens/ml-
100k/u.user", "readOptions" : [ { "key" : "delimiter", "value" : "|"
}, { "key" : "header", "value" : "false" } ], "outputCollection" :
"users", "clearDatasource" : false, "defineFieldsUsingInputSchema" :
true, "atomicUpdates" : false, "transformSql" : "SELECT _c0 as
user_id, \n CAST(_c1 as INT) as age, \n _c2 as gender,\n _c3 as
occupation,\n _c4 as zip_code \n FROM _input" }
```

Read from S3

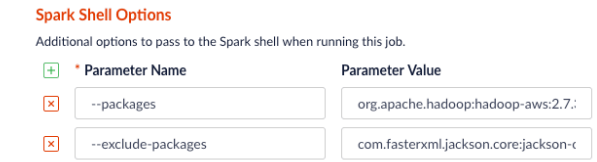
It is easy to read from an S3 bucket without pulling data down to your local workstation first. To avoid exposing your AWS credentials, add them to a file named `core-site.xml` in the `apps/spark-dist/conf` directory, such as:

```
<configuration> <property> <name>fs.s3a.access.key</name> <value>???
</value> </property> <property> <name>fs.s3a.secret.key</name>
<value>???</value> </property> </configuration>
```

Then you can load files using the S3a protocol, such as: `s3a://sstk-dev/data/u.user`. If you are running a Fusion cluster then each instance of Fusion will need a `core-site.xml` file. S3a is the preferred protocol for reading data into Spark because it uses Amazon's libraries to read from S3 instead of the legacy Hadoop libraries. If you need other S3 protocols (for example, s3 or s3n) you will need to add the equivalent properties to `core-site.xml`.



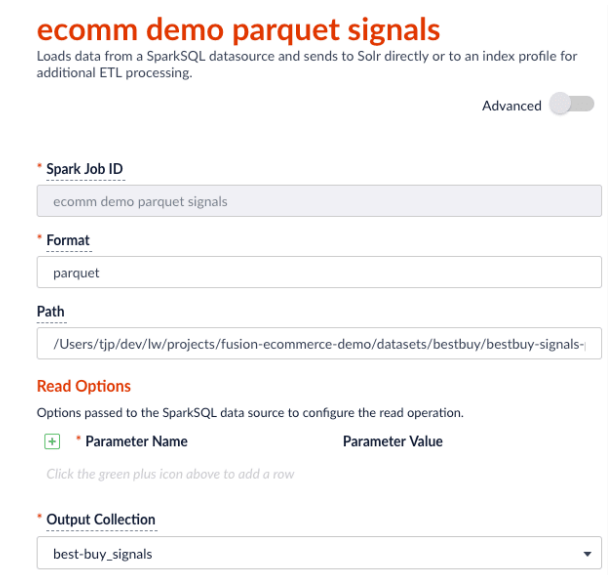
You will need to add the `org.apache.hadoop:hadoop-aws:2.7.3` package to the job using the `--packages` Spark option. Also, you will need to exclude the `com.fasterxml.jackson.core:jackson-core,joda-time:joda-time` packages using the `--exclude-packages` option.



You can also read from Google Cloud Storage (GCS), but you will need a few more properties in your `core-site.xml`; see Installing the Cloud Storage connector.

Read from Parquet

Reading from parquet files is built into Spark using the "parquet" format. For additional read options, see Configuration of Parquet.



Job JSON:

```
{ "type" : "parallel-bulk-loader", "id" : "ecomm demo parquet
signals", "format" : "parquet", "path" : "./part-00000-c1951958-98ae-
4f2a-b7b4-2e3a69fcf403-c000.snappy.parquet", "outputCollection" :
"best-buy_signals", "clearDatasource" : false,
"defineFieldsUsingInputSchema" : true, "atomicUpdates" : false }
```

This example also uses the **`transformScala`** option to filter and transform the input DataFrame into a better form for indexing using the following Scala script:

```
import java.util.Calendar import java.util.Locale import
java.util.TimeZone def transform(inputDF: Dataset[Row]) : Dataset[Row]
= { // do transformations and/or filter the inputDF here val signalsDF
= inputDF.filter((unix_timestamp($"timestamp_tdt", "MM/dd/yyyy
HH:mm:ss.SSS") < 1325376000)) val now = System.currentTimeMillis() val
maxDate = signalsDF.agg(max("timestamp_tdt")).take(1)
(0).getAs[java.sql.Timestamp](0).getTime val diff = now - maxDate val
add_time = udf((t: java.sql.Timestamp, diff : Long) => new
java.sql.Timestamp(t.getTime + diff)) val day_of_week = udf((t:
java.sql.Timestamp) => { val calendar =
Calendar.getInstance(TimeZone.getTimeZone("UTC"))
calendar.setTimeInMillis(t.getTime)
calendar.getDisplayName(Calendar.DAY_OF_WEEK, Calendar.LONG,
Locale.getDefault) }) //Remap some columns to bring the timestamps
current signalsDF .withColumnRenamed("timestamp_tdt",
"orig_timestamp_tdt").withColumn("timestamp_tdt",
add_time($"orig_timestamp_tdt", lit(diff))) .withColumn("date",
$"timestamp_tdt") .withColumn("tx_timestamp_txt",
date_format($"timestamp_tdt", "E YYYY-MM-d HH:mm:ss.SSS Z"))
.withColumn("param.query_time_dt", $"timestamp_tdt")
.withColumn("date_day", date_format(date_sub($"date", 0), "YYYY-MM-
d'T'HH:mm:ss.SSS'Z'")) .withColumn("date_month",
date_format(trunc($"date", "mm"), "YYYY-MM-d'T'HH:mm:ss.SSS'Z'"))
.withColumn("date_year", date_format(trunc($"date", "yyyy"), "YYYY-MM-
d'T'HH:mm:ss.SSS'Z'")) .withColumn("day_of_week",
day_of_week($"date")) }
```

Read from JDBC tables

You can use the Parallel Bulk Loader to parallelize reads from JDBC tables, if the tables have numeric columns that can be partitioned into relatively equal partition sizes. In the example below, we partition the employees table into 4 partitions using the `emp_no` column (`int`). Behind the scenes, Spark sends four separate queries to the database and processes the result sets in parallel.

LOAD THE JDBC DRIVER JAR FILE INTO THE BLOB STORE

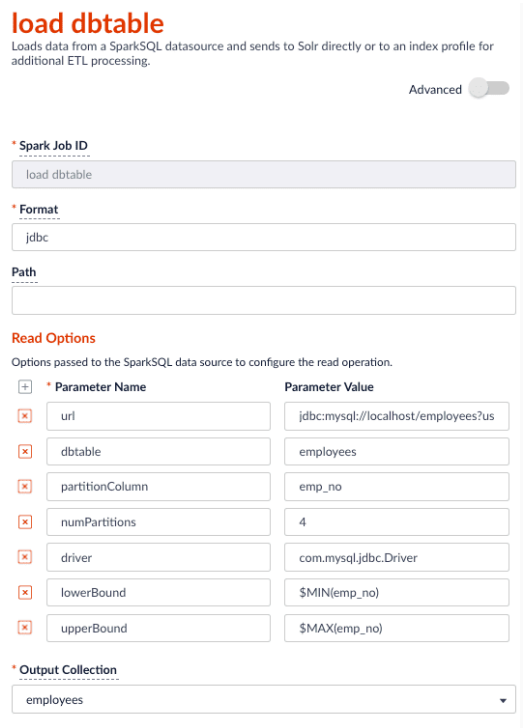
Before you ingest from a JDBC data source, you need to use the Fusion Admin UI to *upload the JDBC driver JAR file into the blob store.*

Alternatively, you can add the JAR file to the Fusion blob store with `resourceType=spark:jar`; for example:

```
curl -XPUT -H "Content-type:application/octet-stream"
"http://<FUSION_HOST>/api/blobs/mysql_jdbc_jar?resourceType=spark:jar"
--data-binary @mysql-connector-java-5.1.45-bin.jar
```

At runtime, Fusion's Spark job management framework knows how to add any JAR files with `resourceType=spark:jar` from the blob store to the appropriate classpaths before running a Parallel Bulk Loader job.

READ FROM A TABLE



For more information on reading from JDBC-compliant databases, see:

- http://spark.apache.org/docs/latest/sql-programming-guide.html#jdbc-to-other-databases
- https://medium.com/@radek.strnad/tips-for-using-jdbc-in-apache-spark-sql-396ea7b2e3d3

Job JSON:

```
{ "type" : "parallel-bulk-loader", "id" : "load dbtable", "format" :
"jdbc", "readOptions" : [ { "key" : "url", "value" :
"jdbc:mysql://localhost/employees?user=?&password=?" }, { "key" :
"dbtable", "value" : "employees" }, { "key" : "partitionColumn",
"value" : "emp_no" }, { "key" : "numPartitions", "value" : "4" }, {
"key" : "driver", "value" : "com.mysql.jdbc.Driver" }, { "key" :
"lowerBound", "value" : "$MIN(emp_no)" }, { "key" : "upperBound",
"value" : "$MAX(emp_no)" } ], "outputCollection" : "employees",
"clearDatasource" : false, "defineFieldsUsingInputSchema" : true,
"atomicUpdates" : false }
```

Notice the use of the `$MIN(emp_no)` and `$MAX(emp_no)` macros in the read options. These are macros offered by the Parallel Bulk Loader to help configure parallel reads of JDBC tables. Behind the scenes, the macros are translated into SQL queries to get the MAX and MIN values of the specified field, which Spark uses to compute splits for partitioned queries. As mentioned above, the field must be numeric and must have a relatively balanced distribution of values between MAX and MIN; otherwise, you are unlikely to see much performance benefit to partitioning.

Index HBase tables

To index an HBase table, use the Hortonworks connector .

Note

The Parallel Bulk Loader lets us replace the HBase Indexer.

You will need to add an `hbase-site.xml` (and possibly `core-site.xml`) to `apps/spark-dist/conf` in Fusion, for example:

```
<configuration> <property>
<name>hbase.defaults.for.version.skip</name> <value>true</value>
</property> <property> <name>hbase.zookeeper.quorum</name>
<value>localhost:2181</value> </property> <property>
<name>zookeeper.znode.parent</name> <value>/hbase</value> </property>
</configuration>
```

For this example, we will create a test table in HBase. If you already have a table in HBase, feel free to use that table instead.

1. Launch the HBase shell and create a table named `fusion_nums` with a single column family named `lw`:

```
create 'fusion_nums', 'lw'
```

2. Do a list command to see your table:

```
hbase(main):002:0> list TABLE fusion_nums 1 row(s) in 0.0250
seconds => ["fusion_nums"]
```

3. Fill the table with some data:

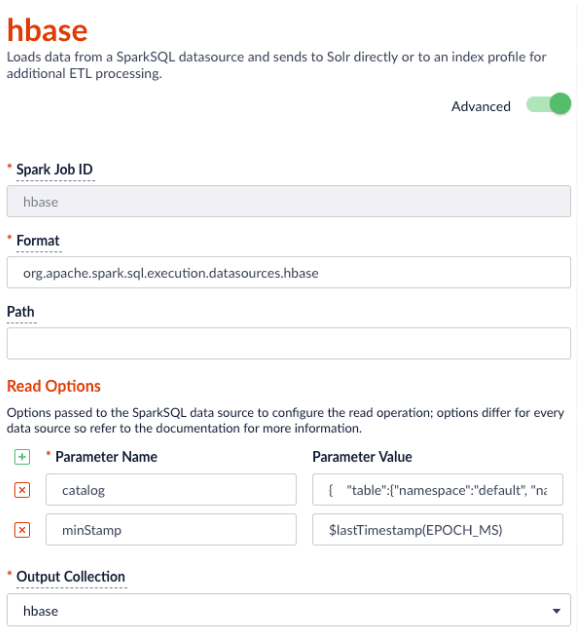
```
for i in '1'..'100' do for j in '1'..'2' do put 'fusion_nums',
"row#{i}", "lw:c#{j}", "#{i}#{j}" end end
```

4. Scan the fusion_nums table to see your data:

```
scan 'fusion_nums'
```

The HBase connector requires a catalog read option that defines the columns you want to read and how to map them into a Spark DataFrame. For our sample table, the following suffices:

```
{ "table":{"namespace":"default", "name":"fusion_nums"},
"rowkey":"key", "columns":{ "id":{"cf":"rowkey", "col":"key",
"type":"string"}, "lw_c1_s":{"cf":"lw", "col":"c1", "type":"string"},
"lw_c2_s":{"cf":"lw", "col":"c2", "type":"string"} } }
```



Notice the use of the `$lastTimestamp` macro in the read options. This lets us filter rows read from HBase using the timestamp of the last document the Parallel Bulk Loader wrote to Solr, that is, to get the newest updates from HBase only (incremental updates). Most Spark data sources provide a way to filter results based on timestamp.

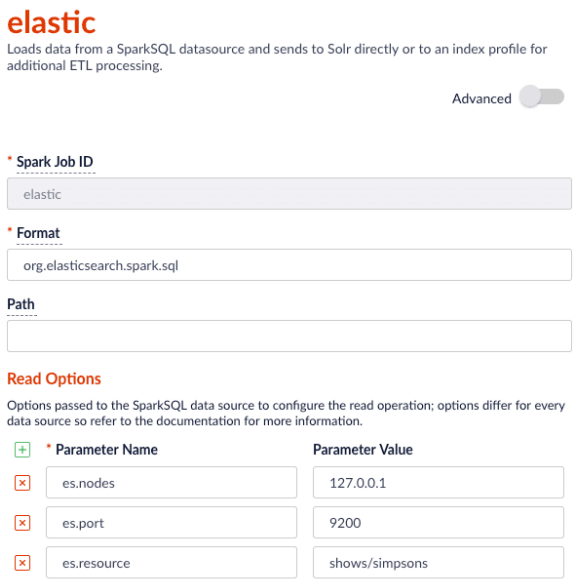
Job JSON:

```
{ "type" : "parallel-bulk-loader", "id" : "hbase", "format" :
"org.apache.spark.sql.execution.datasources.hbase", "readOptions" : [
{ "key" : "catalog", "value" : "{    \"table\":
{\"namespace\":\"default\", \"name\":\"fusion_nums\"},
\"rowkey\":\"key\",    \"columns\":{      \"id\":{\"cf\":\"rowkey\",
\"col\":\"key\", \"type\":\"string\"},      \"lw_c1_s\":
{\"cf\":\"lw\", \"col\":\"c1\", \"type\":\"string\"},
\"lw_c2_s\":{\"cf\":\"lw\", \"col\":\"c2\", \"type\":\"string\"}    }
}" }, { "key" : "minStamp", "value" : "$lastTimestamp(EPOCH_MS)" } ],
"outputCollection" : "hbase", "timestampFieldName" : "timestamp_tdt",
"clearDatasource" : false, "defineFieldsUsingInputSchema" : true,
"atomicUpdates" : false, "shellOptions" : [ { "key" : "--packages",
"value" : "com.hortonworks:shc-core:1.1.1-2.1-s_2.11" }, { "key" : "--
repositories", "value" :
"http://repo.hortonworks.com/content/repositories/releases" } ] }
```

Index Elastic data

With Elasticsearch 6.2.2 using the `org.elasticsearch:elasticsearch-spark-20_2.11:6.2.1` package, here is a Scala script to run in `bin/spark-shell` to index some test data:

```
import spark.implicits._ case class SimpsonCharacter(name: String,
actor: String, episodeDebut: String) val simpsonsDF = sc.parallelize(
SimpsonCharacter("Homer", "Dan Castellaneta", "Good Night") ::
SimpsonCharacter("Marge", "Julie Kavner", "Good Night") ::
SimpsonCharacter("Bart", "Nancy Cartwright", "Good Night") ::
SimpsonCharacter("Lisa", "Yeardley Smith", "Good Night") ::
SimpsonCharacter("Maggie", "Liz Georges and more", "Good Night") ::
SimpsonCharacter("Sideshow Bob", "Kelsey Grammer", "The Telltale
Head") :: Nil).toDF() val writeOpts = Map("es.nodes" -> "127.0.0.1",
"es.port" -> "9200", "es.index.auto.create" -> "true",
"es.resouce.auto.create" -> "shows/simpsons")
simpsonsDF.write.format("org.elasticsearch.spark.sql").mode("Overwrite
").save("shows/simpsons")
```



Job JSON:

```
{ "type" : "parallel-bulk-loader", "id" : "elastic", "format" :
"org.elasticsearch.spark.sql", "readOptions" : [ { "key" : "es.nodes",
"value" : "127.0.0.1" }, { "key" : "es.port", "value" : "9200" }, {
"key" : "es.resource", "value" : "shows/simpsons" } ],
"outputCollection" : "hbase_signals_aggr", "clearDatasource" : false,
"defineFieldsUsingInputSchema" : true, "atomicUpdates" : false,
"shellOptions" : [ { "key" : "--packages", "value" :
"org.elasticsearch:elasticsearch-spark-20_2.11:6.2.2" } ] }
```

Read from Couchbase

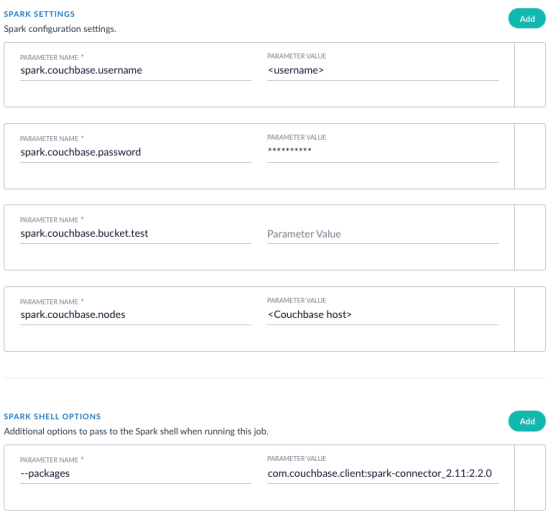
To index a Couchbase bucket, use the official Couchbase Spark connector found here.

For example, we will create a test bucket in Couchbase. If you already have a bucket in Couchbase, feel free to use that and skip to the test data setup section. This test was performed using Couchbase Server 6.0.0.

1. Create a bucket test in the Couchbase admin UI. Give access to a system account user to use in the Parallel Bulk Loader job config.
2. Connect to Couchbase using the command line client cbq. For example, `cbq -e=http://<host>:8091 -u <user> -p <password>`. Ensure the provided user is an authorized user of the test bucket.
3. Create a primary index on the test bucket: `CREATE PRIMARY INDEX 'test-primary-index' ON 'test' USING GSI;`.
4. Insert some data: `INSERT INTO 'test' ( KEY, VALUE ) VALUES ( "1", { "id": "01", "field1": "a value", "field2": "another value"} ) RETURNING META().id as docid, *;`.
5. Verify you can query the document just inserted: `select * from 'test';`.

To ingest from this bucket with the Parallel Bulk Loader, use the Couchbase Spark connector by specifying the format `com.couchbase.spark.sql.DefaultSource`. Then specify the `com.couchbase.client:spark-connector_2.11:2.2.0` package as the `spark shell --packages` option, as well as a few spark settings that direct the connector to a particular Couchbase server and bucket to connect to using the provided credentials. See here for all of the available Spark configuration settings for the Couchbase Spark connector.

Putting it all together:



XML setup

XML is a supported format that requires settings for `format` and `--packages`. In addition, you must specify the filepath in the `readOptions` section. For example:

```
{ "type": "parallel-bulk-loader", "id": "pbl", "format":
"com.databricks.spark.xml", "path": "this-is-ignored", "readOptions":
[ { "key": "rowTag", "value": "tag_name_representing_record" }, {
"key": "path", "value": "/home/user/file.xml.gz" } ],
"outputCollection": "output_collection_name", "clearDatasource": true,
"defineFieldsUsingInputSchema": true, "atomicUpdates": false,
"transformScala": "", "shellOptions": [ { "key": "--packages",
"value": "com.databricks:spark-xml_2.11:0.5.0" } ], "cacheAfterRead":
false }
```

Importing Security Certificates

How-to

> Compatible versions: 4.2, 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *App Studio*

There are numerous reasons that you might be required to import security certificates into the trusted keystore that the Java Virtual Machine (JVM) or the Appkit application is configured to use. These include:

- Resources from secure servers proxied through Appkit (such as images displayed using the Media module)
- Authentication mechanisms which require communication with secure back-ends
- Search platforms using secure protocols for communication

The JVM comes with a default keystore containing the root certificates for numerous Certificate Authorities (CAs). These are bodies which issue secure certificates, typically used during encrypted HTTPS communication over SSL.

However not all certificates are recognized by the JVM when it is acting as an HTTPS client communicating with a secured server. This will usually manifest itself as an error coming from the application such as a `java.security.cert.CertificateException` error. If the error is coming directly from the browser (easily identifiable as a non-Java exception) then it is the user's browser or operating system rejecting a certificate and not an Appkit related problem.

If it is evident that the issue is that the JVM does not trust the CA that issues your SSL certificate it does not mean that the certificate is not valid, but rather that the database of valid CAs that comes with the JVM is not complete and does not include the CA root certificate. To solve this you must import the SSL certificate into the Java keystore on each VM running Appkit.

Fusion 5.x

Using a custom certificate for the `api-gateway`

See *Create a Java Keystore for Use with Fusion* for more information about the process of adding certificates to your Java KeyStore to use with the `api-gateway`.

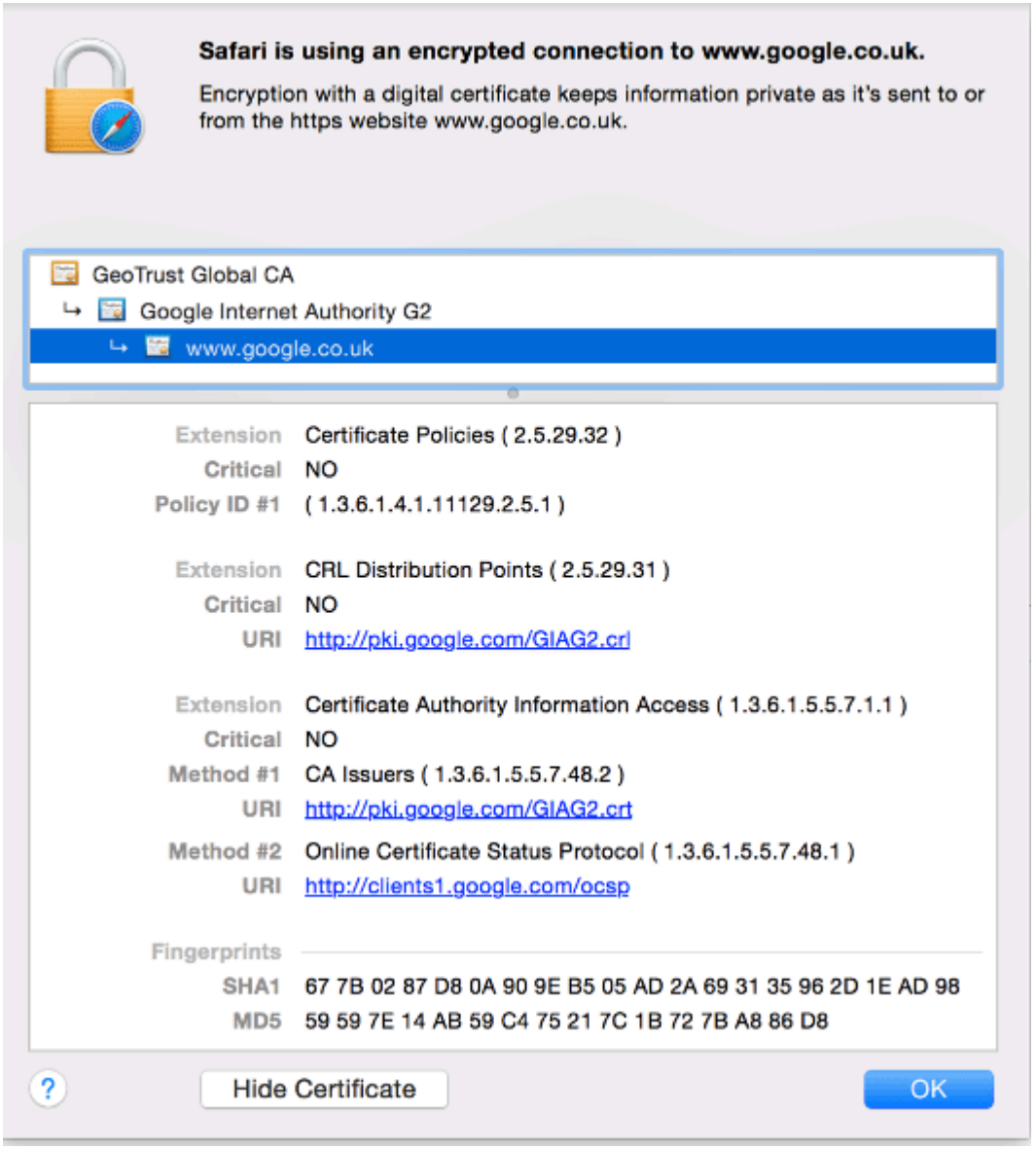
Using custom certificates for specific connectors

See the **Add additional trusted certificate(s) for connectors to allow crawling of web resources with SSL/TLS enabled** section of *Deploy Fusion at Scale* for more information about the process of adding a custom certificate for specific connectors.

Java SSL

1. Download the certificate

Download or otherwise obtain the SSL certificate CRT file for the secured server in question. If the server is accessible from a browser, an easy method to download the certificate is to use the browser's security inspection tools (usually clicking on the padlock icon in most browsers). For example, in Safari it can be done in this way for `https://www.google.co.uk`:



Download the certificate by clicking on the link to the .CRT file and save it as "c:\gsa.crt", for example.

2. Import the certificate

1. Run the Java keytool on the machine on which you are running Appkit. This can be your local development machine or a VM or server for QA or production. Assuming you have set the JAVA_HOME environment variable as indicated below:

```
%JAVA_HOME%\bin\keytool -importcert -trustcacerts -file
c:\gsa.crt -keystore %JAVA_HOME%\jre\lib\security\cacerts
```

   - Change the path to the .CRT file if needed.
   - The default password of the JVM cacerts keystore is "changeit"

2. Restart the application server (for example, Tomcat) and retry the action that requires communication over SSL.

Note

This is a purely SSL+Java configuration issue and has nothing to do with Appkit. If you Google the issue "peer not authenticated", you will find a lot of people suggesting a workaround that is to simply accept all SSL certificates in the Java HTTP client. That is a very bad idea for the reason that it bypasses SSL security and exposes the application to a MITM attack.

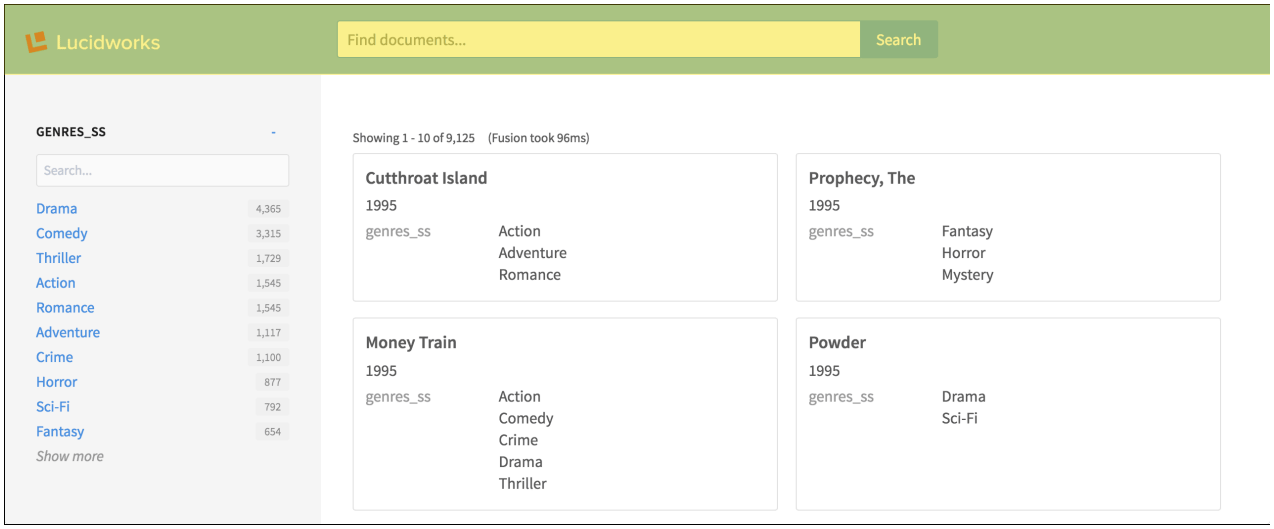
Include a Partial Page

How-to

> Compatible versions: 4.2
>
> Works with: *App Studio*

The header of the default view is a partial file that is included with the `layout:include` tag:

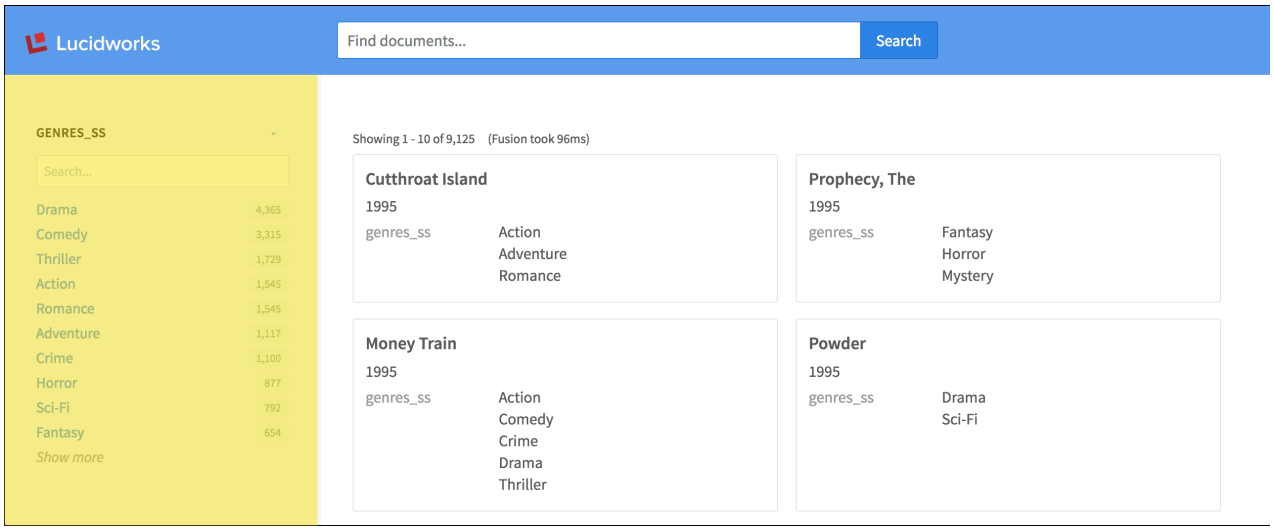
```
<layout:include file="views/partials/header.tpl.html" action="search">
</layout:include>
```



The `header.tpl.html` file, in turn, contains a number of `layout` tags, plus the important `search:box` tag.

Sidebar with facet list

The sidebar is rendered using the simple `layout:sidebar` tag. Within the sidebar, the `search:facet-list` and `search:facet` tags contain the parameters that configure the facet fields and their display.



```
<layout:block md="1-3" lg="1-4" drawer="left" id="sidebar"
styling="blocksidebar-light"> <layout:sidebar> <layout:box> <!-- List
facets and give them intuitive labels --> <search:facet-list facet-
names="genres_ss" response="response" platform="platform"
query="query" styling="facet-list facet-list-wrappedheader">
<search:facet when-field="genres_ss" show="10" search-enabled="true"
platform="platform" query="query" max-characters="40" show="12" show-
more="24" collapsible="true"></search:facet> <search:facet show="10"
max-characters="40" show="12" show-more="24" collapsible="true">
</search:facet> </search:facet-list> </layout:box> </layout:sidebar>
</layout:block>
```

Incorporate Fusion Signals into Appkit Applications

How-to

> Compatible versions: 4.2, 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *App Studio*

1 Add the Fusion Signals dependency

Signals tracking in Fusion requires the Fusion Signals module. To enable this module, insert this within the `dependencies` tag of your `pom.xml`:

```
<dependency> <groupId>twigkit</groupId>
<artifactId>twigkit.activity.fusion-signals</artifactId>
<version>${project.parent.version}</version> </dependency>
```

2 Add fusion.conf to activity tracking resources

To access this module when Appkit starts up, create a `fusion.conf` file in `resources/conf/activity/tracking/`. In that file, configure the `signals-endpoint`. This is the REST-API endpoint that will ingest signal data into a signals collection. For example:

```
signals-endpoint:
http://localhost:8764/api/apollo/signals/image_catalog signals-index-
pipeline: commit: true async: false platform-config: platforms.fusion
```

Here, `image_catalog` is the name of a primary collection that will be used to generate an auxiliary collection consisting of activity tracking data.

When accessing Fusion using a *service account*, the parameter `platform-config` is required. This is the name of the Fusion platform configuration that includes the necessary parameters required for basic authentication. For example, if the configuration file for the Fusion platform, `fusion.conf` is stored in `resources/conf/platforms/fusion/` then the `platform-config` parameter would be set as shown above. If, however, the configuration file for the Fusion platform was named `myFusion.conf` and again stored in `resources/conf/platforms/fusion/`, then the `platform-config` parameter would be `platforms.fusion.myFusion`. For quick reference, the platform configuration should contain these parameters to enable basic authentication:

```
impersonate: true userName: joebloggs password: Enc(<encoded
password>)
```

For more information about how authentication is used when accessing the Fusion Signals endpoint, see the *Authentication* section below.

Several optional parameters are also provided as shown. The first, `signals-index-pipeline`, can be used to define an index pipeline that will to be used to convert the raw JSON signal data into a set of Solr documents. As stated in the *Fusion Signals API documentation*, if no pipeline is defined, then the preconfigured `_signals_ingest` pipeline will be used. Both the remaining two parameters, `commit` and `async`, are booleans. If `commit` is set to true, a Solr commit will occur at the end of indexing allowing a persistent record of the activity to be kept. If `async` is set to true, signals will be indexed in asynchronous mode. In this mode, an autoCommit is issued with each signal and failures are not reported. The default is false.

Increase SQL Resource Allocations

How-to

> Compatible versions: 4.2, 5.5
>
> Works with: *Fusion Server*, *Fusion*

So as to not conflict with the CPU and memory settings used for Fusion driver applications (default & script), the Fusion SQL service uses a unique set of configuration properties for granting CPU and memory for executing SQL queries.

You can use the *Configurations API* to override the default values shown here.

Configuration Property and Default	Description
`fusion.sql.cores` 1	Sets the max number of cores to use across the entire cluster to execute SQL queries. Give as many as possible while still leaving CPU available for other Fusion jobs.
`fusion.sql.executor.cores` 1	Number of cores to use per executor
`fusion.sql.memory` 1g	Memory per executor to use for executing SQL queries
`fusion.sql.default.shuffle.partitions` 20	Default number of partitions when performing a distributed group-by-type operation, such as a JOIN
`fusion.sql.bucket_size_limit.threshold` 30,000,000	Threshold that determines when to use Solr streaming rollup instead of facet when computing aggregations; rollup can handle high cardinality dimensions but is much slower than using facets to compute aggregate measures.
`fusion.sql.max.no.limit.threshold` 10,000	Sets a limit for SQL queries that select all fields and all rows, that is, `select * from table-name`.
`fusion.sql.max.cache.rows` 5,000,000	Do not cache tables bigger than this threshold. If a user sends the cache-table command for large collections with row counts that exceed this value, then the cache operations will fail.
`fusion.sql.max_scan_rows` 2,000,000	Safeguard mechanism to prevent queries that request too many rows from large tables. Queries that read more than this many rows from Solr will fail; increase this threshold for larger Solr clusters that can handle streaming more rows concurrently.

Tip

The Fusion SQL service is designed for executing analytics-style queries over large data sets. You need to provide ample CPU and memory so that queries execute efficiently and can leverage Spark's in-memory caching for joins and aggregations.

Here is an example of increasing the resources for the Fusion SQL service:

```
curl -H 'Content-type:application/json' -X PUT -d '8'
"http://<FUSION_HOST>/api/configurations/fusion.sql.cores" curl -H
'Content-type:application/json' -X PUT -d '8'
"http://<FUSION_HOST>/api/configurations/fusion.sql.executor.cores"
curl -H 'Content-type:application/json' -X PUT -d '2g'
"http://<FUSION_HOST>/api/configurations/fusion.sql.memory" curl -H
'Content-type:application/json' -X PUT -d '8'
"http://<FUSION_HOST>/api/configurations/fusion.sql.default.shuffle.pa
rtitions"
```

If you change any of these settings, you must restart the Fusion SQL service with `./sql restart` (on Unix) or `sql.cmd restart` (on Windows).

The Fusion SQL service is a long-running Spark application and, as such, it holds on to the resources (CPU and memory) allocated to it using the aforementioned settings. Consequently, you might need to reconfigure the CPU and memory allocation for other Fusion Spark jobs to account for the resources given to the Fusion SQL service. In other words, any resources you give to the Fusion SQL service are no longer available for running other Fusion Spark jobs. For more information on adjusting the CPU and memory settings for Fusion Spark jobs, see the *Spark configuration settings*.

Index Binary Data from JDBC

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

The JDBC connector fetches documents from a relational database via SQL queries. Under the hood, this connector implements the Solr DataImportHandler (DIH) plugin.

The JDBC connector in Fusion does not automatically discover and index binary data you may have stored in your database (such as PDF files). However, you can configure Fusion to recognize and extract binary data correctly by modifying the datasource configuration file. This file is created when the datasource is first run, and then it is created in `https://EXAMPLE_COMPANY.b.lucidworks.cloud/data/connectors/lucid.jdbc/datasources/datasourceID/conf`. The name of the file will include the name of the datasource, as in `dataconfig_datasourceName.xml`. If you are familiar with Solr's DIH, you may recognize this as a standard `dataconfig.xml` file.

Follow these steps to modify the configuration file:

1. Add a `name` attribute for the database containing your binary data to the `dataSource` entry.
2. Set the `convertType` attribute for the `dataSource` to `false`. This prevents Fusion from treating binary data as strings.
3. Add a `FieldStreamDataSource` to stream the binary data to the Tika entity processor.
4. Specify the `dataSource` name in the `root` entity.
5. Add an entity for your `FieldStreamDataSource` using the `TikaEntityProcessor` to take the binary data from the `FieldStreamDataSource`, parse it, and specify a field for storing the processed data.
6. Reload the Solr core to apply your configuration changes.

Index Binary Data from JDBC

How-to

> Compatible versions: 4.2, 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion Connectors*, *Fusion Server*, *Fusion*

The JDBC connector fetches documents from a relational database via SQL queries. Under the hood, this connector implements the Solr DataImportHandler (DIH) plugin.

The JDBC connector in Fusion does not automatically discover and index binary data you may have stored in your database (such as PDF files). However, you can configure Fusion to recognize and extract binary data correctly by modifying the datasource configuration file. This file is created when the datasource is first run, and then it is created in `https://FUSION_HOST:FUSION_PORT/data/connectors/lucid.jdbc/datasources/datasourceID/conf`. The name of the file will include the name of the datasource, as in `dataconfig_datasourceName.xml`. If you are familiar with Solr's DIH, you will recognize this as a standard `dataconfig.xml` file.

Follow these steps to modify the configuration file:

1. Add a `name` attribute for the database containing your binary data to the `dataSource` entry.
2. Set the `convertType` attribute for the `dataSource` to `false`. This prevents Fusion from treating binary data as strings.
3. Add a `FieldStreamDataSource` to stream the binary data to the Tika entity processor.
4. Specify the `dataSource` name in the `root` entity.
5. Add an entity for your `FieldStreamDataSource` using the `TikaEntityProcessor` to take the binary data from the `FieldStreamDataSource`, parse it, and specify a field for storing the processed data.
6. Reload the Solr core to apply your configuration changes.

Install a Connector - Fusion 4.x

How-to

> Compatible versions: 4.2
>
> Works with: *Fusion Connectors*, *Fusion Server*

In Fusion 4.x, connectors are installed by uploading them to the blob store. You can use any of these methods to install a connector:

- By installing connectors *as "bootstrap plugins"*, that is, by putting them in the `bootstrap-plugins` directory during initial installation or an upgrade
- By using the *Blob Store UI* after installation or an upgrade
- By using the *Blob Store API* after installation or an upgrade.

Note

During upgrades, the migrator handles some aspects of installing connectors. Depending on the target version and the presence or absence of an Internet connection, there might be manual steps. Installing connectors during upgrades is explained where needed in the *upgrade procedures*.

After you install a connector, you can *Configure A New Datasource*.

Tip

You can view and download all current and previous V2 connector releases at plugins.lucidworks.com.

Install a connector as a bootstrap plugin

Fusion can install connectors as "bootstrap plugins." All this means is that you put the connector `zip` files in a specific directory named `bootstrap-plugins`, and Fusion installs the connectors the first time it starts during initial installation or an upgrade.

1. Download the connector zip file from *Fusion 4.x Connector Downloads*.
   Do not expand the archive; Fusion consumes it as-is. Also, do not start Fusion until instructed to do so by the installation or upgrade instructions.
2. Under the version-numbered Fusion directory, place the connector in the directory `apps/connectors/bootstrap-plugins/` (on Unix) or `\apps\connectors\bootstrap-plugins\` (on Windows).
3. Start Fusion when instructed to do so in the installation or upgrade procedure.

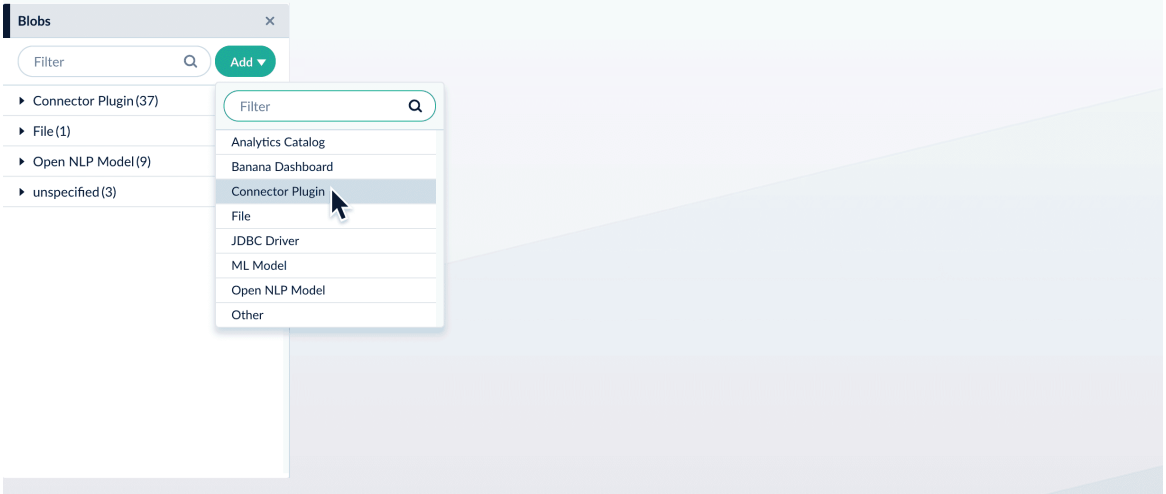
Install a connector using the Blob Store UI

1. Download the connector zip file from *Fusion 4.x Connector Downloads*.

   Note

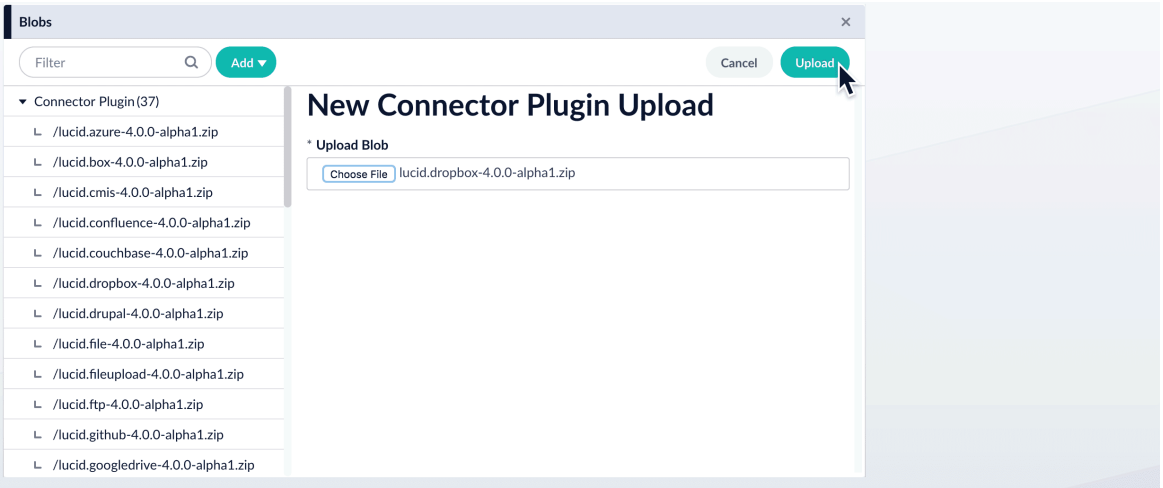
   Do not expand the archive; Fusion consumes it as-is.

2. In the Fusion workspace, navigate to **System** > **Blobs**.
3. Click **Add**.
4. Select **Connector Plugin**.

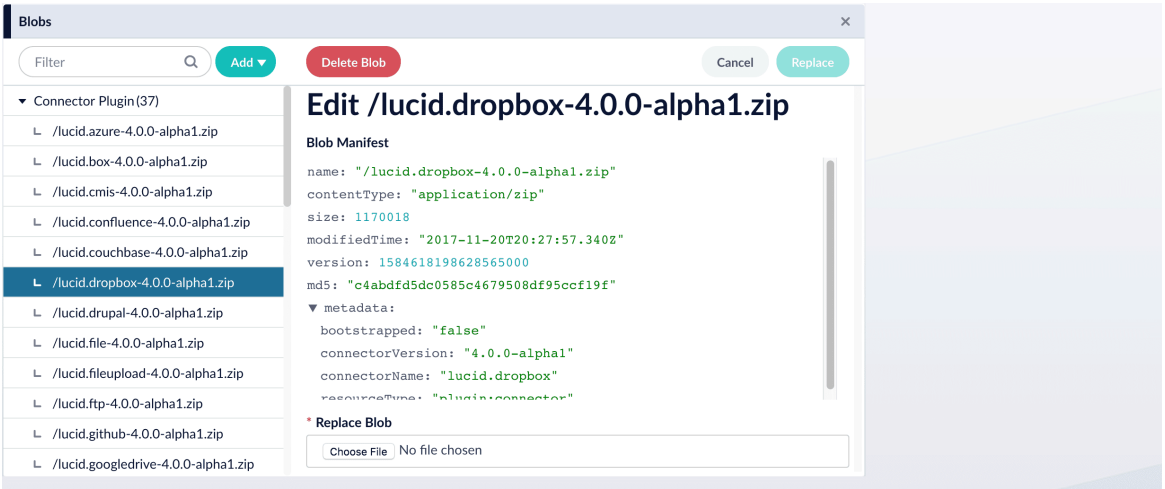


The "New Connector Plugin Upload" panel appears.

5. Click **Choose File** and select the downloaded zip file from your file system.



6. Click **Upload**.
The new connector's blob manifest appears.



From this screen you can also delete or replace the connector.

Install a connector using the Blob Store API

1. Download the connector zip file from *Fusion 4.x Connector Downloads*.

   Note

   Do not expand the archive; Fusion consumes it as-is.

2. Upload the connector zip file to Fusion's blob store.
   Specify an arbitrary blob ID, and a `resourceType` value of `plugin:connector`, as in this example:

   ```
   curl -H 'content-type:application/zip' -X PUT
   'localhost:8764/api/blobs/myplugin?
   resourceType=plugin:connector' --data-binary @myplugin.zip
   ```

   Fusion automatically publishes the event to the cluster, and the listeners perform the connector installation process on each node.

   Tip

   If the blob ID is identical to an existing one, the old connector will be uninstalled and the new connector will installed in its place. To get the list of existing blob IDs, run:
   `curl -u USERNAME:PASSWORD localhost:8764/api/blobs`

3. Look in `https://FUSION_HOST:FUSION_PORT/apps/connectors/plugins/` to verify the new connector is installed.

Reinstall a connector

To reinstall a connector for any reason, first *delete the connector* then use the preceding steps to install it again. This may take a few minutes to complete depending on how quickly the pods are deleted and recreated.

Install a Fusion 4.x Cluster

How-to

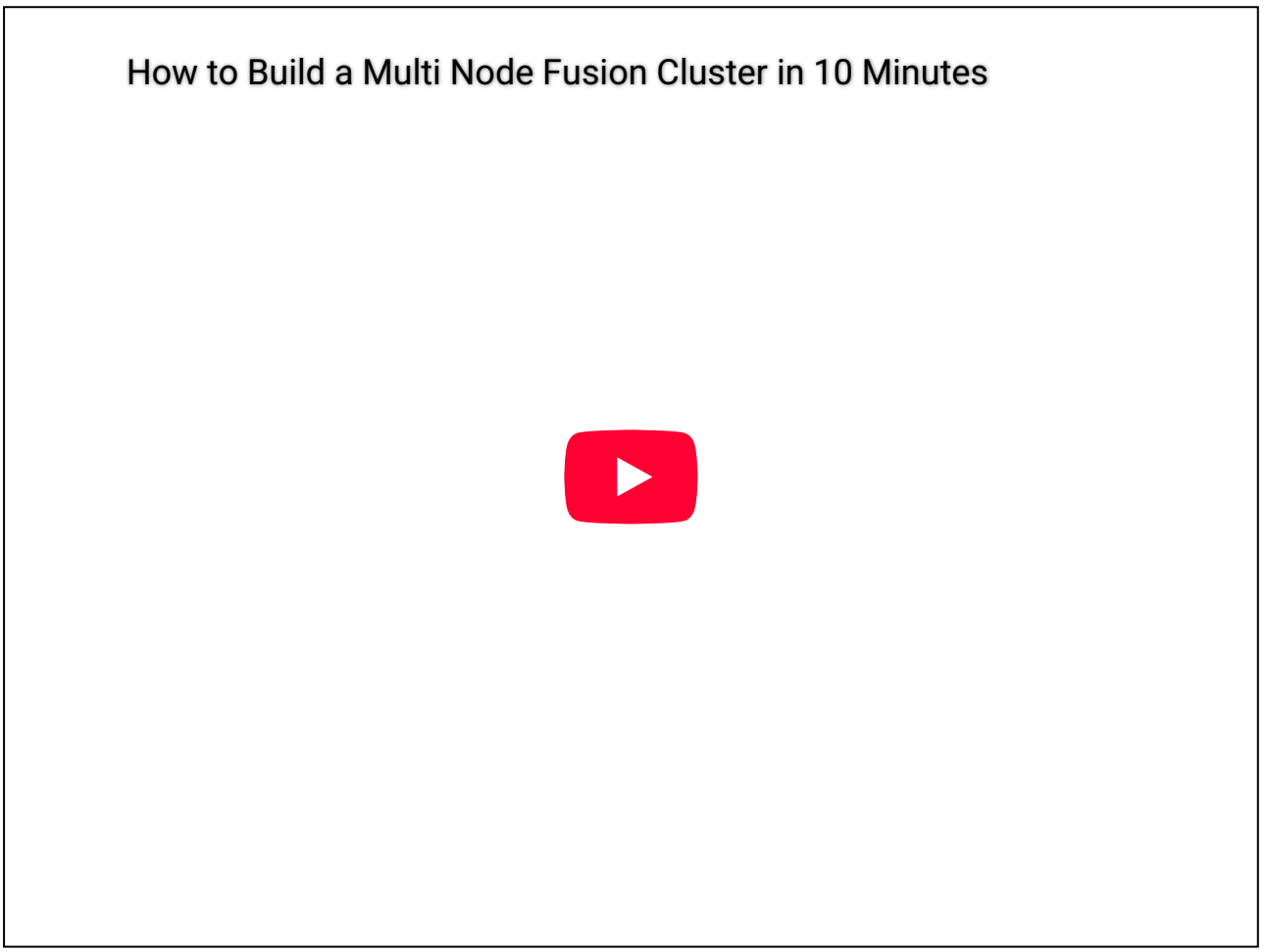
> Compatible versions: 4.2
>
> Works with: *Fusion Server*

To support enterprise search applications at any scale, install Fusion on multiple nodes (a *Fusion cluster*). Deployed in this manner, Fusion can store large amounts of data and achieve high processing throughput.

Note

Historically, "node" has also had a meaning at Lucidworks related to licensing and to the number of CPU cores. We do not mean that here. In the context of this documentation "node" is a computer, usually a physical server but also possibly a virtual machine (especially for test environments).

See also this video tutorial for building a multi-mode Fusion cluster:



To install Fusion on a single node, see *Install Fusion on a Single Node*.

Warning

Failures in the Fusion install or startup may occur if the Fusion installation directory name contains a space.

Install Fusion Server with Bundled Solr and ZooKeeper

When you install Fusion Server, the software includes bundled versions of Solr and ZooKeeper. Install Fusion Server on every node. Then, configure Fusion Server so that it does not start Solr and ZooKeeper on nodes where they are not needed.

Supported Cluster Arrangements

To satisfy processing requirements, install Fusion Server, ZooKeeper, and Solr on specific nodes.

Before proceeding, review the supported *cluster arrangements*.

Hardware and software requirements

Verify that the nodes on which you plan to install Fusion meet *hardware and software requirements*.

Next steps

Perform the installation steps in one of these articles:

- *Install a Fusion Cluster (Unix)*
- *Install a Fusion Cluster (Windows)*

If you already have a Solr deployment, use the instructions *Integrate Fusion with an Existing Solr Deployment*.

Install a Fusion 4.x Cluster (Unix)

How-to

> Compatible versions: 4.2
>
> Works with: *Fusion Server*

This article describes how to install a Fusion cluster on multiple Unix nodes. Instructions are given for each of the cluster arrangements described in *Deployment Types*.

Preliminary steps

Before proceeding to one of the sections that follow, perform these steps:

How to prepare for setting up a Fusion cluster

1. Prepare your firewall so that the Fusion nodes can communicate with each other. *The default ports list* contains a list of all ports used by Fusion. From this list, it is important that the ZooKeeper ports, Apache Ignite ports, and the Spark ports (if you are using Spark) are open between the different nodes for cross-cluster communication.
2. Fusion for Unix is distributed as a compressed archive file (`.tar.gz`). Move this file to each node that will run Fusion.

   Note

   To leverage the copies of Solr and/or ZooKeeper that are distributed with Fusion on nodes that will *not* run Fusion (as a simple means of obtaining compatible versions of the other software), also download the Fusion compressed archive file to each of those nodes. Below, you will edit configuration files so that Fusion does not run on those nodes.

3. On each node, change your working directory to the directory in which you placed the Fusion tar/zip file and unpack the archive, for example:

   ```
   $ cd /opt/lucidworks $ tar –xf fusion–version.x.tar.gz
   ```

   Warning

   Failures in the Fusion install or startup may occur if the Fusion installation directory name contains a space.

   The resulting directory is named `https://FUSION_HOST:FUSION_PORT` . You can rename this if you wish. This directory is considered your Fusion home directory. See *Directories, files, and ports* for the contents of the `https://FUSION_HOST:FUSION_PORT` directory.

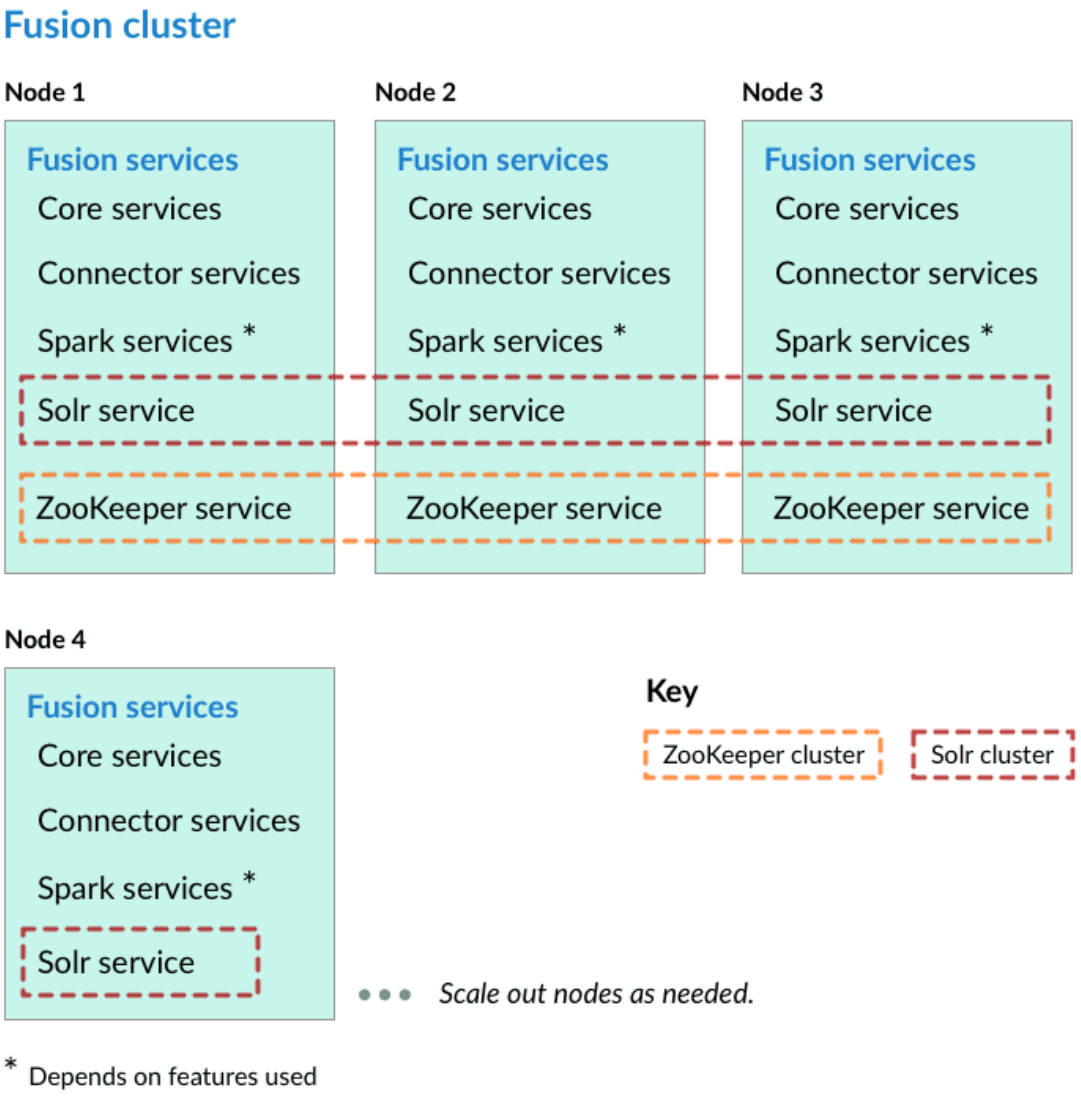
Important

In the sections that follow, for every step on multiple nodes, complete the step *on all nodes* before going to the next step. It is especially important that you *do not* start Fusion on any node until the instructions say to do so.

In the steps below, the port numbers reflect *default port numbers* and one common choice (port 2181 for nodes in an external ZooKeeper cluster). Port numbers for your nodes might differ.

Nodes running core Fusion services and Solr also run ZooKeeper

In this cluster arrangement, a ZooKeeper cluster runs on the same nodes that run core Fusion services and Solr.



How to set up a Fusion cluster

Perform the steps in the section *Preliminary steps*, and then perform these steps:

1. Assign a number to each Fusion node, starting at 1. We refer to the number we assign to each node as the *ZooKeeper myid*.
2. On each Fusion node, create a `https://FUSION_HOST:FUSION_PORT/data/zookeeper` directory, and a file called `myid` in that directory. Edit the file and save the *ZooKeeper myid* assigned for this node as the only contents.
3. On each Fusion node, open the `https://FUSION_HOST:FUSION_PORT/conf/zookeeper/zoo.cfg` file in a text editor and add the following after the `clientPort` line (change the hostnames or IP addresses to the correct ones for your servers):

```
server.1=[Hostname or IP for ZooKeeper with myid 1]:2888:3888
server.2=[Hostname or IP for ZooKeeper with myid 2]:2888:3888
server.3=[Hostname or IP for ZooKeeper with myid 3]:2888:3888
```

For example:

```
server.1=10.10.31.130:2888:3888 server.2=10.10.31.178:2888:3888
server.3=10.10.31.166:2888:3888
```

Note

Do not use `localhost` or `127.0.0.1` as the hostname/IP. Specify the hostname/IP that other nodes will use when communicating with the current node.

4. On each Fusion node, edit `default.zk.connect` in `https://FUSION_HOST:FUSION_PORT/conf/fusion.cors` (`fusion.properties` in Fusion 4.x) to point to the ZooKeeper hosts:

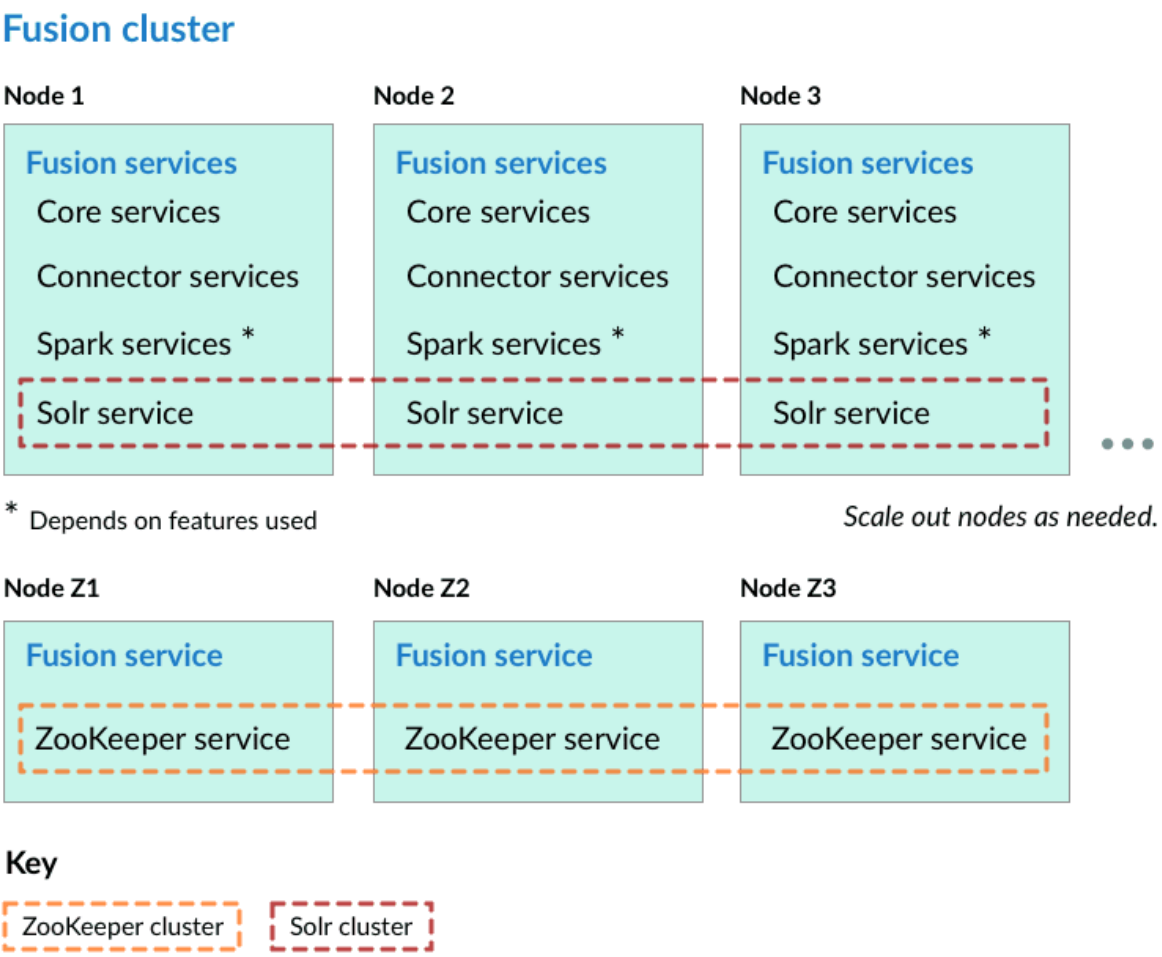
```
default.zk.connect=[ZK host 1]:9983,[ZK host 2]:9983,[ZK host
3]:9983
```

5. On each node, start ZooKeeper with `bin/zookeeper start`. Zookeeper should start without errors. If a ZooKeeper instance fails to start, check the log at `https://FUSION_HOST:FUSION_PORT/var/log/zookeeper/zookeeper.log`.
6. On each node, start the rest of Fusion using `bin/fusion start`.
7. Create an admin password and log in to Fusion at `http://FIRST_NODE_IP:8764`, where `FIRST_NODE_IP` is the IP address of your first Fusion node.
8. Verify the Solr cluster is healthy by looking at `http://ANY_NODE_IP:8983/solr/#/~cloud`, where `ANY_NODE_IP` is the IP address of a Solr node. All of the nodes should appear green.
9. If necessary, prepare high availability by setting up a load balancer in front of Fusion so that it load balances between the Fusion UI URL's at `http://NODE_IP:8764`.
   Consult your load balancer's documentation for instructions.

Nodes running ZooKeeper are not running core Fusion services or Solr

In this cluster arrangement, the ZooKeeper cluster runs on nodes in the Fusion cluster on which core Fusion services and Solr are not running.

Each node in the Fusion cluster has Fusion and Solr installed. ZooKeeper runs on Fusion cluster nodes on which neither Fusion nor Solr is running.



How to set up a Fusion cluster

Perform the steps in the section *Preliminary steps*, and then perform these steps:

1. Assign a number to each Fusion node, starting at 1. We refer to the number we assign to each node as the *ZooKeeper myid*.
2. On each Fusion node, create a `fusion\latest.x\data\zookeeper` directory, and a file called `myid` in that directory. Edit the file and save the *ZooKeeper myid* assigned for this node as the only contents.
3. On each Fusion node, open the `fusion\latest.x\conf\zookeeper\zoo.cfg` file in a text editor and add the following after the `clientPort` line (change the hostnames or IP addresses to the correct ones for your servers):

```
server.1=[Hostname or IP for ZooKeeper with myid 1]:2888:3888
server.2=[Hostname or IP for ZooKeeper with myid 2]:2888:3888
server.3=[Hostname or IP for ZooKeeper with myid 3]:2888:3888
```

For example:

```
server.1=10.10.31.130:2888:3888 server.2=10.10.31.178:2888:3888
server.3=10.10.31.166:2888:3888
```

4. Edit `conf/fusion.cors` (`fusion.properties` in Fusion 4.x) and remove `zookeeper` from the `group.default` list. This will make it so that ZooKeeper does not start when you start Fusion.
5. On each Fusion node, edit `default.zk.connect` in `https://FUSION_HOST:FUSION_PORT/conf/fusion.cors` (`fusion.properties` in Fusion 4.x) to point to the ZooKeeper hosts:

```
default.zk.connect=[ZK host 1]:2181,[ZK host 2]:2181,[ZK host 3]:2181
```

6. On each node, start ZooKeeper with `bin/zookeeper start`. Zookeeper should start without errors. If a ZooKeeper instance fails to start, check the log at `https://FUSION_HOST:FUSION_PORT/var/log/zookeeper/zookeeper.log`.
7. On each node, start the rest of Fusion using `bin/fusion start`.
8. Create an admin password and log in to Fusion at `http://FIRST_NODE_IP:8764`, where `FIRST_NODE_IP` is the IP address of your first Fusion node.
9. Verify the Solr cluster is healthy by looking at `http://ANY_NODE_IP:8983/solr/#/~cloud`, where `ANY_NODE_IP` is the IP address of a Solr node. All of the nodes should appear green.
10. If necessary, prepare high availability by setting up a load balancer in front of Fusion so that it load balances between the Fusion UI URL's at `http://NODE_IP:8764`.
Consult your load balancer's documentation for instructions.

Known issues

Metrics collection failure

When the Java virtual machine (JVM) is started, the `/tmp/.java_pid<pid>` file is created and is the socket used:

- To attach a debugger
- By the agent to connect to the service that collects Java Management Extension (JMX) metrics

A known issue in Java 8 is that the timestamp is not updated, which causes the file to be deleted in standard Linux distribution systems. For example, the `/tmp/.java_pid<pid>` is deleted after ten days on a standard Amazon Linux in EC2.

When the JVM code the agent uses cannot locate the file, then it:

- Sends a `-QUIT` message to the JVM
- Triggers a thread dump to be printed to standard out

The standard out:

- Is logged to the agent log
- Generates the "No metrics can be gathered" exception
- Prints a complete thread dump
- Sends the thread dump to system logs

Choose one of the two workarounds:

- Exclude the `agent.log` from the logstash configuration. The logshipping is turned off for the file. *The disadvantage to this option is that the metrics are missing.*
- Change the cron job in the Linux distribution that deletes the `/tmp` files older than "x" days to exclude deleting the `/tmp/.java_pid<pid>` files. If your system is running the Linux Systemd software suite on EC2, the setting is typically located in the `usr/lib/tmpfiles.d/tmp.conf` file. For Dial On Demand (DOD), remove the call that configures the JMX Metrics requirement for the debugger attachment to the Java service.

Install a Fusion 4.x Cluster (Windows)

How-to

> Compatible versions: 4.2
>
> Works with: *Fusion Server*

This article describes how to install a Fusion cluster on multiple Windows nodes. Instructions are given for each of the cluster arrangements described in *Deployment Types*.

Preliminary steps

Warning

Failures in the Fusion install or startup may occur if the Fusion installation directory contains a space.

Before proceeding to one of the sections that follow, perform these steps:

How to prepare for setting up a Fusion cluster

1. Prepare your firewall so that the Fusion nodes can communicate with each other. *The default ports list* contains a list of all ports used by Fusion. From this list, it is important that the ZooKeeper ports, Apache Ignite ports, and the Spark ports (if you are using Spark) are open between the different nodes for cross-cluster communication.
2. Fusion for Windows is distributed as a compressed archive file (`.zip`). Download the Fusion zip file for the latest version of Fusion to each node that will run Fusion, and move the file to where you would like Fusion to reside in your filesystem. It will appear as a compressed folder.

   Note

   To leverage the copies of Solr and/or ZooKeeper that are distributed with Fusion on nodes that will *not* run Fusion (as a simple means of obtaining compatible versions of the other software), also download the Fusion zip file to each of those nodes. Below, you will edit configuration files so that Fusion does not run on those nodes.

3. Unpack the archive. In most cases, you need only right-click and choose "Extract all...". If you do not see this option, check that you have permissions to extract folders on your system.
   The resulting directory is named `fusion\latest.x` . This directory is considered your Fusion home directory. You can rename it if you wish. See *Directories, Files, and Ports* for the contents of the Fusion home directory.

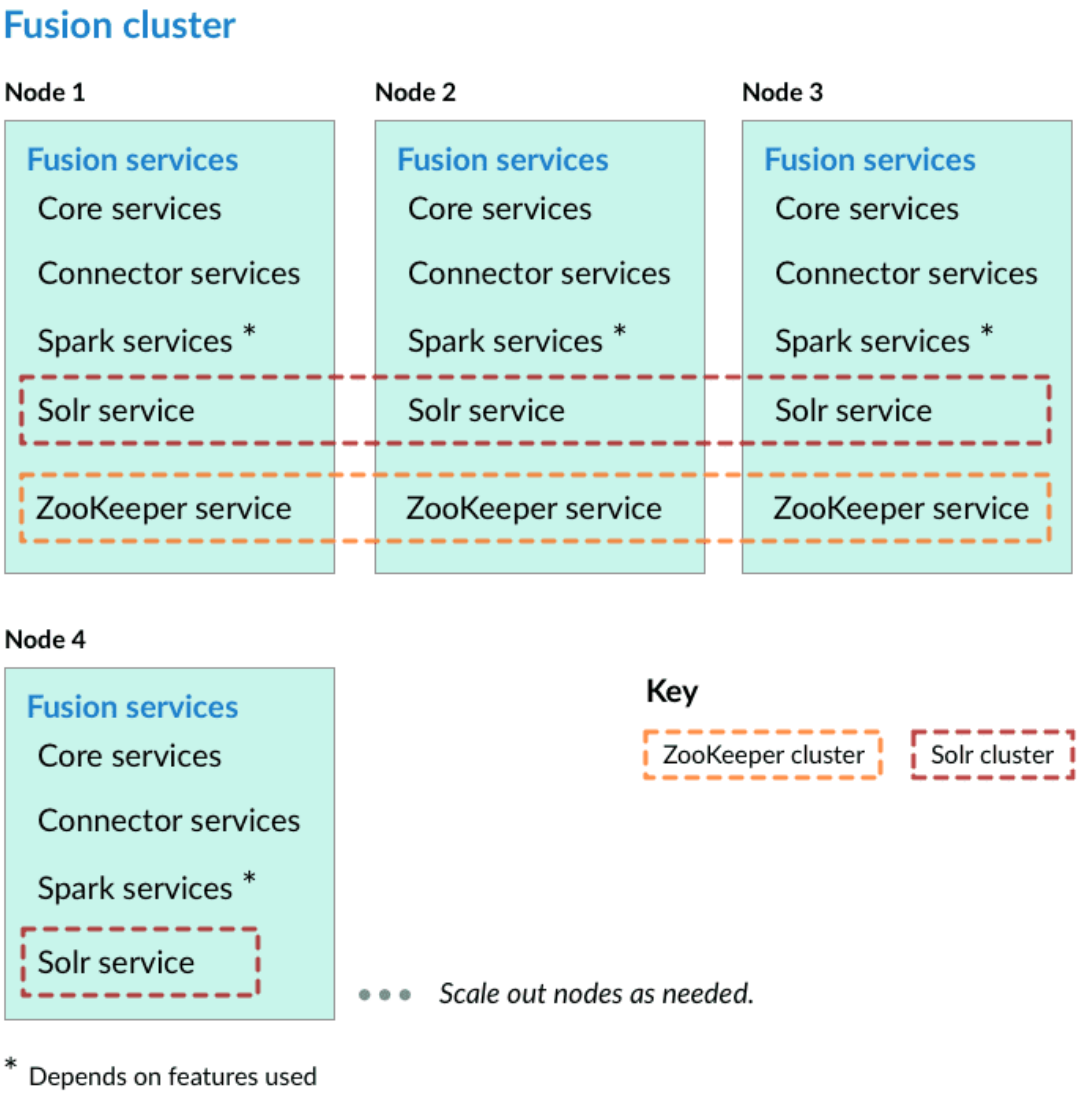
Important

In the sections that follow, for every step on multiple nodes, complete the step *on all nodes* before going to the next step. It is especially important that you *do not* start Fusion on any node until the instructions say to do so.

In the steps below, the port numbers reflect *default port numbers* and one common choice (port 2181 for nodes in an external ZooKeeper cluster). Port numbers for your nodes might differ.

Nodes running core Fusion services and Solr also run ZooKeeper

In this cluster arrangement, a ZooKeeper cluster runs on the same nodes that run core Fusion services and Solr.



How to set up a Fusion cluster

Perform the steps in the section *Preliminary steps*, and then perform these steps:

1. Assign a number to each Fusion node, starting at 1. We refer to the number we assign to each node as the *ZooKeeper myid*.
2. On each Fusion node, create a `fusion\latest.x\data\zookeeper` directory, and a file called `myid` in that directory. Edit the file and save the *ZooKeeper myid* assigned for this node as the only contents.
3. On each Fusion node, open the `fusion\latest.x\conf\zookeeper\zoo.cfg` file in a text editor and add the following after the `clientPort` line (change the hostnames or IP addresses to the correct ones for your servers):

```
server.1=[Hostname or IP for ZooKeeper with myid 1]:2888:3888
server.2=[Hostname or IP for ZooKeeper with myid 2]:2888:3888
server.3=[Hostname or IP for ZooKeeper with myid 3]:2888:3888
```

For example:

```
server.1=10.10.31.130:2888:3888 server.2=10.10.31.178:2888:3888
server.3=10.10.31.166:2888:3888
```

Note

do not use `localhost` or `127.0.0.1` as the hostname/IP. Specify the hostname/IP that other nodes will use when communicating with the current node.

4. On each Fusion node, edit `default.zk.connect` in `fusion\latest.x\conf\fusion.cors` (`fusion.properties` in Fusion 4.x) to point to the ZooKeeper hosts:

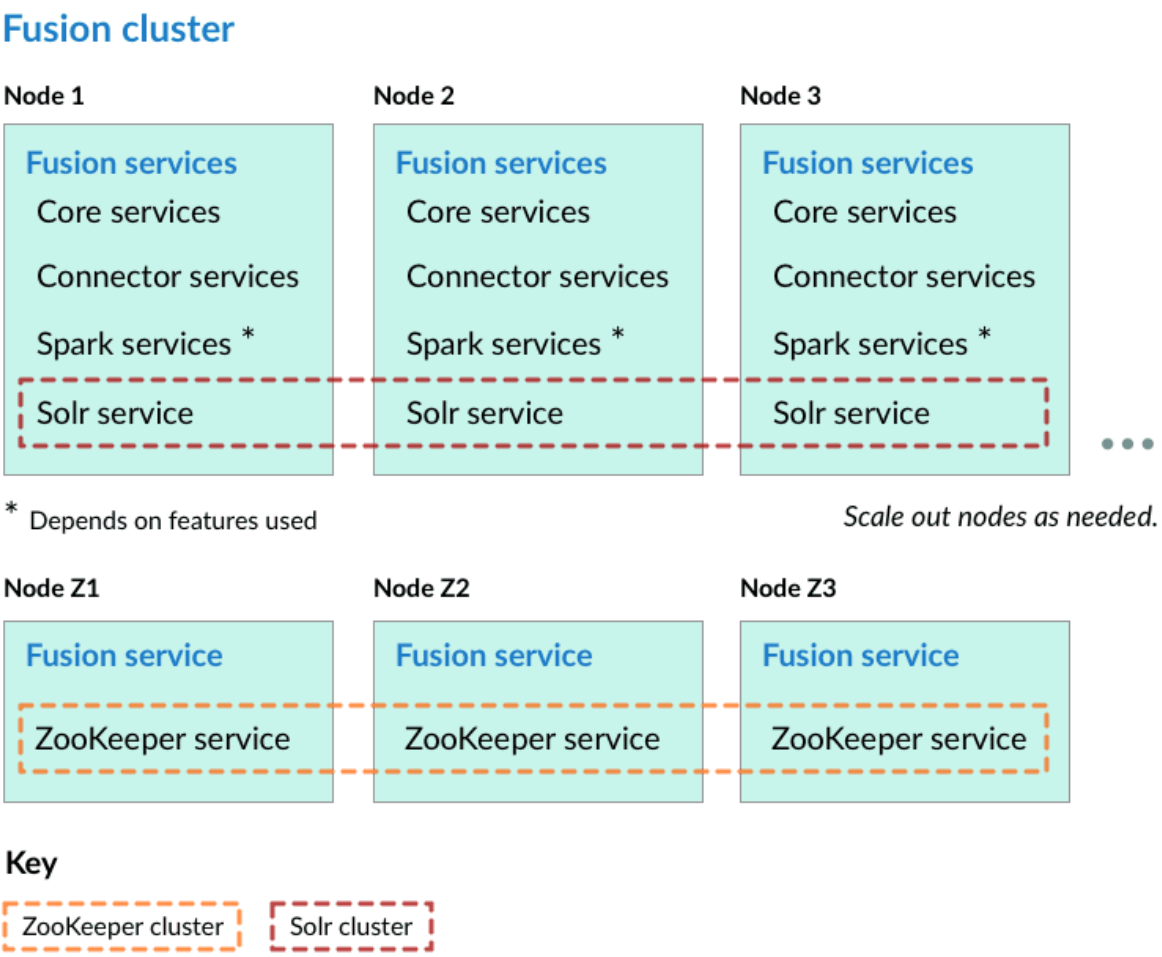
   ```
   default.zk.connect=[ZK host 1]:9983,[ZK host 2]:9983,[ZK host
   3]:9983
   ```

5. On each node, start ZooKeeper with `bin\zookeeper start`. Zookeeper should start without errors. If a ZooKeeper instance fails to start, check the log at `fusion\latest.x\var\log\zookeeper\zookeeper.log`.
6. On each node, start the rest of Fusion using `bin\fusion start`.
7. Create an admin password and log in to Fusion at `http://FIRST_NODE_IP:8764`, where `FIRST_NODE_IP` is the IP address of your first Fusion node.
8. Verify the Solr cluster is healthy by looking at `http://ANY_NODE_IP:8983/solr/#/~cloud`, where `ANY_NODE_IP` is the IP address of a Solr node. All of the nodes should appear green.
9. If necessary, prepare high availability by setting up a load balancer in front of Fusion so that it load balances between the Fusion UI URL's at `http://NODE_IP:8764`.
   Consult your load balancer's documentation for instructions.

Nodes running ZooKeeper are not running core Fusion services or Solr

In this cluster arrangement, the ZooKeeper cluster runs on nodes in the Fusion cluster on which core Fusion services and Solr are not running.

Each node in the Fusion cluster has Fusion and Solr installed. ZooKeeper runs on Fusion cluster nodes on which neither Fusion nor Solr is running.



How to set up a Fusion cluster

Perform the steps in the section *Preliminary steps*, and then perform these steps:

1. Assign a number to each Fusion node, starting at 1. We refer to the number we assign to each node as the *ZooKeeper myid*.
2. On each Fusion node, create a `fusion\latest.x\data\zookeeper` directory, and a file called `myid` in that directory. Edit the file and save the *ZooKeeper myid* assigned for this node as the only contents.
3. On each Fusion node, open the `fusion\latest.x\conf\zookeeper\zoo.cfg` file in a text editor and add the following after the `clientPort` line (change the hostnames or IP addresses to the correct ones for your servers):

```
server.1=[Hostname or IP for ZooKeeper with myid 1]:2888:3888
server.2=[Hostname or IP for ZooKeeper with myid 2]:2888:3888
server.3=[Hostname or IP for ZooKeeper with myid 3]:2888:3888
```

For example:

```
server.1=10.10.31.130:2888:3888 server.2=10.10.31.178:2888:3888
server.3=10.10.31.166:2888:3888
```

4. Edit `conf\fusion.cors` (`fusion.properties` in Fusion 4.x) and remove `zookeeper` from the `group.default` list. This will make it so that ZooKeeper does not start when you start Fusion.
5. On each Fusion node, edit `default.zk.connect` in `fusion\latest.x\conf\fusion.cors` (`fusion.properties` in Fusion 4.x) to point to the ZooKeeper hosts:

```
default.zk.connect=[ZK host 1]:2181,[ZK host 2]:2181,[ZK host 3]:2181
```

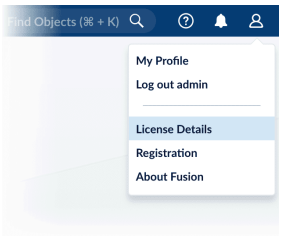
6. On each node, start ZooKeeper with `bin\zookeeper start`. Zookeeper should start without errors. If a ZooKeeper instance fails to start, check the log at `fusion\latest.x\var\log\zookeeper\zookeeper.log`.
7. On each node, start the rest of Fusion using `bin\fusion start`.
8. Create an admin password and log in to Fusion at `http://FIRST_NODE_IP:8764`, where `FIRST_NODE_IP` is the IP address of your first Fusion node.
9. Verify the Solr cluster is healthy by looking at `http://ANY_NODE_IP:8983/solr/#/~cloud`, where `ANY_NODE_IP` is the IP address of a Solr node. All of the nodes should appear green.
10. If necessary, prepare high availability by setting up a load balancer in front of Fusion so that it load balances between the Fusion UI URL's at `http://NODE_IP:8764`.
    Consult your load balancer's documentation for instructions.

Install a Fusion License

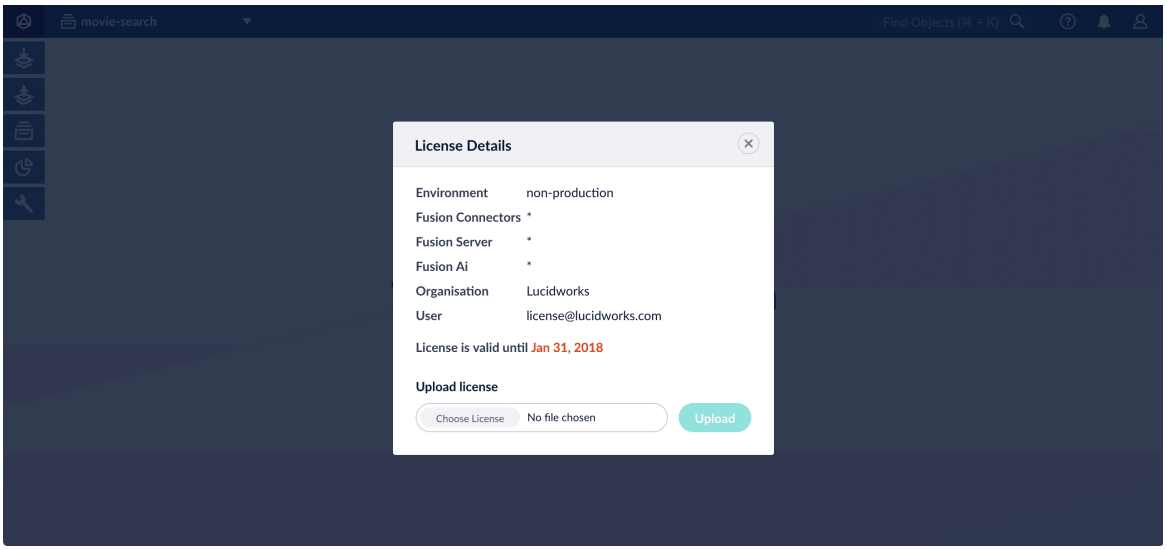
How-to

> Compatible versions: 4.2, 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion Server*, *Fusion*

1. Log in to the Fusion UI.
2. In the upper right, open the profile menu and select **License Details**:



The License Details window appears:



3. Click **Choose License** and select your license file.
4. Click **Upload**.

Install a Springboard embed

How-to

> Works with: Connected Search

Springboard embeds are drop-and-go search UI components that help you build your frontend search interface. Embeds are preconfigured and production-ready.

This article teaches you how to install *Springboard embeds* in your frontend search interface. Before you begin, *get your search application ready*. Once complete, you're ready to *install* and *customize*.

Install the embed

1. Click the Integrations icon from the *Applications UI sidebar*.
2. From the Embeds tab, click the toggle to turn your application from Private to Public. An **Allowed origins** field and the embed code appears.

Caution

The `embed-token` value used by this embed is generated when the Private toggle is switched to Public. If you switch back to Private, the token is deleted. Switching the toggle to Public again creates a new authentication token, and your embed snippet must be replaced.

3. In the **Allowed origins** field, add the domains where you want to use the embed.
4. Locate the embed you want to install.
5. Click **Copy** to copy the embed script. Paste it in the `<head>` of your HTML file.
6. Click **Copy** to copy the embed HTML object code. Paste it in `<body>` of your HTML file.
7. Build your site. The embed is visible where you placed it in your site.
8. *Customize*, test, and launch.

FAQ

Can I use embed tokens to connect to multiple Springboard applications on the same site?

No, your site can only connect to one *Springboard application* at a time with embed tokens.

This also applies to subdomains of your site. If your site, `example.com`, uses embeds with your Springboard application, you can't connect `internal.example.com` to a different Springboard application using embed tokens.

Customize the embed

If you want to customize the embed to match your company style guidelines or for another use case, you can create custom CSS rules. Follow the instructions below:

1. In the `<head>` of your HTML file, locate the embed code snippet.

```
<script type="module" id="lw-ui-lib" api-
url="https://APPLICATION_ID.applications.lucidworks.com/" embed-
token="EMBED_TOKEN" src="SCRIPT_SRC"></script>
```

2. Add a `<link>` to your custom style sheet below the embed script.

```
<script type="module" id="lw-ui-lib" api-
url="https://APPLICATION_ID.applications.lucidworks.com/" embed-
token="EMBED_TOKEN" src="SCRIPT_SRC"></script> <!-- Custom style
sheet below... --> <link rel="stylesheet"
href="CUSTOM_STYLESHEET_SRC">
```

Important

Be sure to place the custom style sheet *below* the embed script. By doing so, your custom rules take precedence over the default rules.

3. Replace the placeholder URL, `CUSTOM_STYLESHEET_SRC`, with the path to your custom style sheet.

You're now ready to create custom CSS rules for the embed.

Install App Studio

How-to

> Compatible versions: 4.2
>
> Works with: *App Studio*

Version 4.1

App Studio is licensed *with Fusion Server*.

App Studio is installed when *installing Fusion Server*.

Version 4.0

You can get started with App Studio in just minutes by expanding the zip file, running the startup script and following the steps of the setup wizard. The application runs in an embedded application server, by default on port 8080.

Licensing

App Studio comes with a trial license file, located at `app-studio/twigkit.lic`. This trial license is valid for 30 days. Contact Lucidworks to obtain a commercial license.

To install a commercial license, copy the new license file to `app-studio/twigkit.lic`, overwriting the trial license. Commercial licenses support multiple instances of App Studio. Lucidworks issues a separate license for each App Studio instance.

Requirements

App Studio is a standalone application with the following requirements:

- Oracle Java JDK 1.8 or later
- A running instance of *Fusion Server*, with one or more collections of indexed data

Installation

1. *Download App Studio*.
2. Copy the App Studio zip file to any convenient path.
3. Expand the archive: `unzip app-studio-latest.x-zip-package.zip`
   This creates an `app-studio` directory.

What is next

- See *Startup and Shutdown* for information about using the App Studio scripts.
- See *Create a New Web App* to learn how to create your first search app.

Install Fusion 4.x

How-to

> Compatible versions: 4.2
>
> Works with: *Fusion Server*

Install new instances of Fusion:

Warning

Failures in the Fusion install or startup may occur if the Fusion installation directory name contains a space.

- *Install Fusion on a Single Node*
  Start here for the simplest deployment, which is useful for evaluation, demo, and development purposes. You can install Fusion on a server, desktop, or laptop.
- *Install a Fusion Cluster (Introduction)*
  For production deployments, install Fusion on multiple servers (a *Fusion cluster*). Start with this introduction. When you install Fusion Server, the software includes bundled versions of Solr and ZooKeeper. Install Fusion on every node. Then, configure Fusion so that it does not start Solr and ZooKeeper on nodes where they are not needed.
- *Install a Fusion Cluster (Unix)*
  Install a Fusion Cluster on Unix nodes. When you install Fusion Server, the software includes bundled versions of Solr and ZooKeeper.
- *Install a Fusion Cluster (Windows)*
  Install a Fusion cluster on Windows nodes. When you install Fusion Server, the software includes bundled versions of Solr and ZooKeeper.
- *Integrate Fusion with an Existing Solr Deployment*
  Use these instructions when installing Fusion in an existing Solr environment.

Install Fusion 4.x on a Single Node

How-to

> Compatible versions: 4.2
>
> Works with: *Fusion Server*

Note

These instructions are for an initial installation of Fusion on a single node (computer). To install Fusion on multiple nodes (a *Fusion cluster*), see *Install a Fusion Cluster*. If you already have a version of Fusion installed and want to upgrade it, see the *Fusion upgrade instructions*.

You can view the application files to download at *Fusion Server 4.x File Download Links*.

Out of the box, Fusion uses the instances of Solr, ZooKeeper, and Spark that are included in the Fusion distribution. See the *Release Notes* to find out which versions of Solr, Spark, and ZooKeeper are included in each Fusion release.

To use Fusion with an existing Solr deployment, see *Integrating with existing Solr instances*.

Ports

This table lists the default port numbers used by Fusion processes. Port settings are defined in the `:fusion.properties` file in `https://FUSION_HOST:FUSION_PORT/conf/` (on Unix or macOS) or `fusion\4.2.x\conf\` (on Windows).

Port	Service
8091	Fusion agent
8763	Fusion UI service (use port 8764 to access the Fusion UI)
8764	Fusion proxy This service includes the Fusion Authorization Proxy.
8765	Fusion API Services
8766	Spark Master
8769	Spark Worker
8771	Connectors RPC Service This service can distribute connector jobs to as many Fusion nodes as you want. It uses HTTP/2 and has an SDK that you can use to *build your own connectors*.
8780	Web Apps This service delivers the UIs of Fusion apps.
8781	Log shipper *(4.1+ only)* Monitoring port that agent uses to check the health of the log shipper process. This port does not need to be accessible from other nodes.
8983	Solr This is the embedded Solr instance included in the Fusion distribution.
8984	Connectors Classic Service This service runs nondistributed connector jobs. It uses HTTP/1.1 and has no SDK.
9983	ZooKeeper The embedded ZooKeeper used by Fusion services. **Important** The ZooKeeper port is also defined in the configuration file for the embedded ZooKeeper, `https://FUSION_HOST:FUSION_PORT/conf/zookeeper/zoo.cfg` (on Unix or macOS) or `fusion\4.2.x\conf\zookeeper\zoo.cfg` (on Windows). Look for `clientPort`. If you run Fusion with the embedded ZooKeeper, remember to change the port number in both places.

Port	Service
47100-48099	Apache Ignite TCP communication port range (used by the API, Connectors Classic, Connectors RPC, and Proxy services)
48100-48199	Apache Ignite shared memory port range (used by the API, Connectors Classic, Connectors RPC, and Proxy services)
49200-49299	Apache Ignite discovery port range (used by API, Connectors Classic, Connectors RPC, and Proxy services)

Additional ports might be required. See *Port configuration* for more information or to modify the default ports before starting Fusion.

Unix installation

Fusion for Unix is distributed as a gzipped tar file.

How to install Fusion on Linux or Mac

1. Verify that the node on which you plan to install Fusion meets *hardware and software requirements*.
2. Download the Fusion tar/zip file for the latest version of Fusion and move it to where you would like it to reside in your filesystem (if you would like to use Upstart for process management, you must install Fusion in `/opt/lucidworks`).
3. Become the user that will run Fusion.

   Important

   Do not run Fusion as the root user.

4. Change your working directory to the directory in which you placed the `fusion-*version*.x.tar.gz` file, for example:

   ```
   $ cd /opt/lucidworks
   ```

   Warning

   Failures in the Fusion install or startup may occur if the Fusion installation directory name contains a space.

5. Unpack the archive with `tar -xf` (or `tar -xvf`), for example:

   ```
   $ tar -xf fusion-*version*.x.tar.gz
   ```

   The resulting directory is named `https://FUSION_HOST:FUSION_PORT`. You can rename this if you wish. This directory is considered your Fusion home directory. See *Directories, Files, and Ports* for the contents of the `https://FUSION_HOST:FUSION_PORT` directory.

Starting Fusion

All Fusion start scripts must be executed by a user who has permissions to read and write to the directories where Fusion is installed. These scripts do not need to be run as root (or sudo), nor should they be. Use a suitable user, or create a new one, and then ensure that it owns the directory where Fusion resides, (for example, `C:\lucidworks`).

Give the commands that follow from the directory `fusion/latest.x/bin`.

Start the required services that are defined in the `group.default` property.

How to start all required services

`./fusion start`

> **Tip**
>
> This is equivalent to `./fusion start default`. You can omit the group name `default`.

For information about starting groups of services or individual services, see *Start and Stop Fusion*.

Running Fusion In The Foreground

To run Fusion or any of its services in the foreground, use the `run` command-line argument in place of `start`.

Stopping Fusion

To stop Fusion or any of its services, use the `stop` command-line argument in place of `start`.

Using systemd to manage processes

On Red Hat Enterprise Linux, CentOS 7 and newer, and Ubuntu 15.04 LTS and newer, we support using the operating system-provided `systemd` for process management.

For more information about using `systemd`, see *Using systemd to manage processes*.

Using Ubuntu Upstart to manage processes

Under Ubuntu 12.04 LTS through Ubuntu 14.10, we support using Upstart for process management. This requires Fusion to be installed in the `/opt/lucidworks/` directory.

For more information about using Upstart, see *Using Ubuntu Upstart to manage processes*.

Windows installation

Fusion for Windows is distributed as a compressed zip file. To unpack the Fusion zip file on Windows, you can use a native compression utility or the freely available 7zip file archiver. Visit the 7zip download page for the latest version.

How to install Fusion on Windows

1. Verify that the node on which you plan to install Fusion meets *hardware and software requirements*.
2. Download the zip file for the latest version of Fusion and move it to where you would like Fusion to reside in your filesystem. It will appear as a compressed folder.
3. Unpack the archive. In most cases, you need only right-click and choose "Extract all...". If you do not see this option, check that you have permissions to extract folders on your system.
   The resulting directory is named `fusion\latest.x`. This directory is considered your Fusion home directory. See *Directories, Files, and Ports* for the contents of the `fusion\latest.x` directory.

To install Fusion as a set of Windows services

1. Run `bin\install-services.cmd`.
2. Enter the name of the windows user that is used to launch this service.
   Remember the username is `COMPUTERNAME\username` or `DOMAIN\username` (if your computer is part of a Windows domain).
3. Enter the user's password.
4. Enter the path to the directory containing the JDK to use for running the services.

Starting Fusion

All Fusion start scripts must be executed by a user who has permissions to read and write to the directories where Fusion is installed. Ensure that the user owns the directory where Fusion resides (for example, `C:\lucidworks`).

Give the commands that follow from the directory `fusion\latest.x\bin`.

How to start all required Fusion services as Java processes

```
fusion.cmd start
```

How to start all required Fusion services as Windows services

```
start-services.cmd
```

For information about starting groups of services or individual services, see *Start and Stop Fusion*.

Stopping Fusion

To stop Fusion or any of its services, use the `stop` command-line argument in place of `start`.

Installation with an existing Solr instance or cluster

Before you install Fusion with an existing Solr instance or cluster, confirm that the *Solr version is supported* by Fusion.

If installing Fusion to work with an existing Solr instance, either in SolrCloud mode or standalone, you should install Fusion as described above. You should start each of the services as described above.

Once Fusion installation is complete, you can register your existing Solr installation with Fusion to be able to use the two systems together. For details on how to do that, see the section *Integrate Fusion with an Existing Solr Deployment*.

Troubleshooting

For information about problems you might encounter when installing Fusion, and solutions, see *Troubleshoot When Installing Fusion*.

Install Grafana Dashboards

How-to

> Compatible versions: 5.5
>
> Works with: *Fusion*

If you're using Prometheus and Grafana to monitor the performance and health of Fusion services, Lucidworks provides pre-configured Grafana dashboards in the fusion-cloud-native repo.

Configure Grafana

If you used the `install_prom.sh` script to install Prometheus and Grafana into your Fusion namespace, retrieve the initial Grafana password from a Kubernetes secret.

1. Run the following:

```
kubectl get secret --namespace "${NAMESPACE}" ${RELEASE}-
monitoring-grafana \ -o jsonpath="{.data.admin-password}" |
base64 --decode ; echo
```

> **Tip**
>
> Replace `${RELEASE}` with your Helm release label, which is typically the same as your `${NAMESPACE}` value.

   To see the name of the Grafana deployment, run: `kubectl get deploy`.
2. With Grafana, you can either set up a temporary port-forward to a Grafana pod or expose Grafana on an external IP using a Kubernetes `LoadBalancer`. To define a `LoadBalancer`, run:

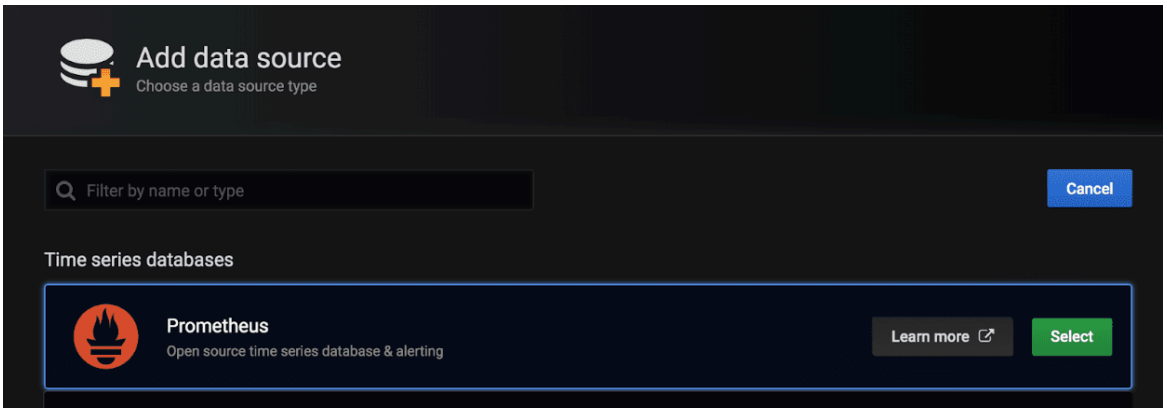
```
kubectl expose deployment ${RELEASE}-monitoring-grafana --
type=LoadBalancer --name=grafana --port=3000 --target-port=3000
```

3. Use `kubectl get services --namespace <namespace>` to verify that the load balancer is set up and has an external IP address.
4. Navigate your browser to `http://<GrafanaIP>:3000`. Log in with the username `admin@localhost` and the password that was returned in the previous step.
5. *(optional)* Create another administrative user with a more desirable password.

Configure the Prometheus datasource

If you used the `install_prom.sh` script, a default Prometheus datasource is pre-configured for you. If not, configure the Prometheus datasource in Grafana:

1. Click the gear icon on the left and navigate to **Data Sources**.
2. Click **Add Data Source**. Select **Prometheus** as the datasource type.



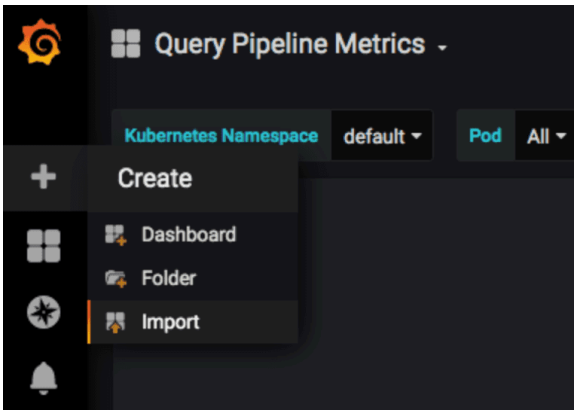
You are redirected to a page that asks for the address of the Prometheus server.

3. Enter `http://<RELEASE>-prom-prometheus-server`
4. *(optional)* Configure any additional fields, if desired.
5. Click **Save and Test**.

Import the Grafana dashboards

If you used the `install_prom.sh` script, then Fusion's default Grafana dashboards will already be imported. If not, import the dashboards from the fusion-cloud-native repo.

1. Click the gear icon on the left and select **Import**.



2. Import the Grafana dashboard JSON.
3. Click the **Load** button.

Install Istio in self-hosted Fusion

How-to

> Compatible versions: 5.9
>
> Works with: *Fusion*

Istio is a service mesh that can help you manage traffic routing between Fusion services, enable security protocols such as Transport Layer Security (TLS), and enforce policies like rate limits, access controls, and quotas. A service mesh is also useful for monitoring service health and performance when paired with an observability tool like Grafana.

If you're using *Ray models* for machine learning in Fusion or Lucidworks AI and you require in-cluster TLS, then Istio is also required.

Note

This feature is only available in Fusion 5.9.x for versions 5.9.12 and later.

Prerequisites

- A dedicated Kubernetes cluster for hosting Fusion, and admin privileges for that cluster.
- A Kubernetes namespace in that cluster where you will install Fusion.
- Helm version 3.0.0 or later.

Install Istio

Install Istio before deploying Fusion into your Kubernetes cluster. For more information, see Istio's Getting Started and Install with Istioctl guides.

1. Install the `istioctl` command line utility:

```
$ curl -sL https://istio.io/downloadIstioctl | sh -
```

2. Add `istioctl` to your path, so you can run the utility directly from your command prompt or terminal. On Mac or Linux, run the following command:

```
export PATH=$HOME/.istioctl/bin:$PATH
```

3. Perform a default installation of Istio using `istioctl`:

```
istioctl install
```

4. To check that Istio successfully installed, run the following command to check the build version:

```
istioctl bug-report version
```

5. Label the Fusion namespace with `istio-injection=enabled`:

```
kubectl label namespace FUSION_NAMESPACE istio-injection=enabled
--overwrite
```

6. Check that the Fusion namespace has the Istio label applied:

```
kubectl get namespace -L istio-injection
```

In the response, `ISTIO-INJECTION` is set to `enabled.`

```
NAME STATUS AGE ISTIO-INJECTION FUSION_NAMESPACE Active 5m9s
enabled
```

Deploy Fusion

After installing Istio, deploy Fusion. Fusion will automatically integrate with the Istio control plane. This integration allows the Istio control plane to recognize the workloads within Fusion and manipulate network traffic between them, thereby enhancing the overall functionality and routing capabilities of the Fusion instance.

- *Deploy Fusion 5 in Google Kubernetes Engine*
- *Deploy Fusion 5 in Amazon Elastic Kubernetes Service*
- *Deploy Fusion 5 in Azure Kubernetes Service*
- *Deploy Fusion 5 on Other Kubernetes Platforms*
- *Deploy Fusion at Scale*

Install or update a connector

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*, *Fusion Connectors*

When you create a new datasource that requires an uninstalled connector, Managed Fusion automatically downloads and installs the connector using the *Datasources dropdown*. You can also update the connector using the *Blob Store UI* or via the *Connector API*.

Install a connector using the Datasources dropdown

1. Sign in to Managed Fusion and click your application.
2. Click **Indexing > Datasources > Add+**.
3. In the list of connectors, scroll down to the Not Installed section and select the connector you want to install.
   Managed Fusion downloads and installs the connector. It displays in the list of installed connectors.

After you install a connector, you can *configure a new datasource*.

Tip

You can view and download all current and previous V2 connector releases at plugins.lucidworks.com .

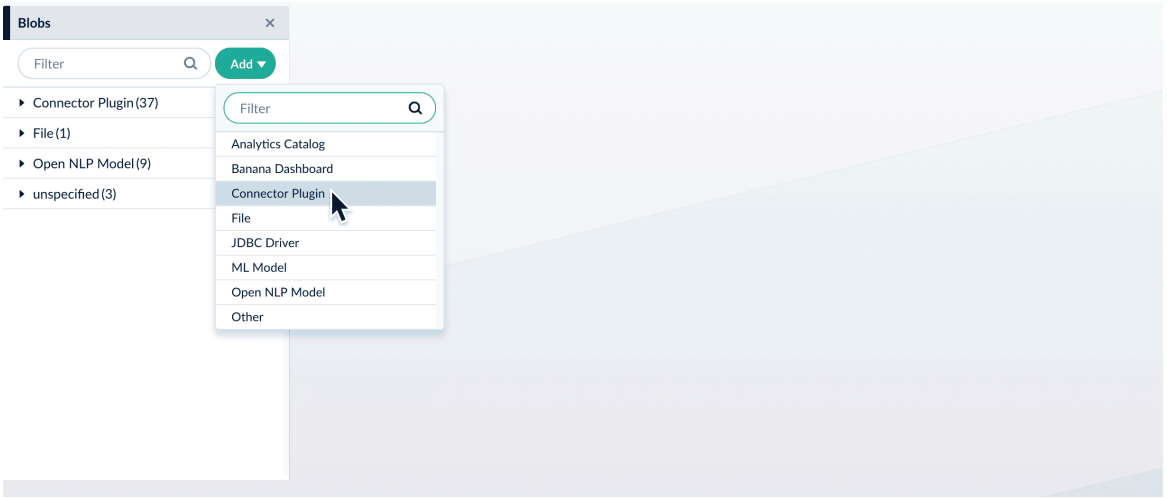
Install or update a connector using the Blob Store UI

1. Download the connector zip file from *Download V2 connectors*.

Note

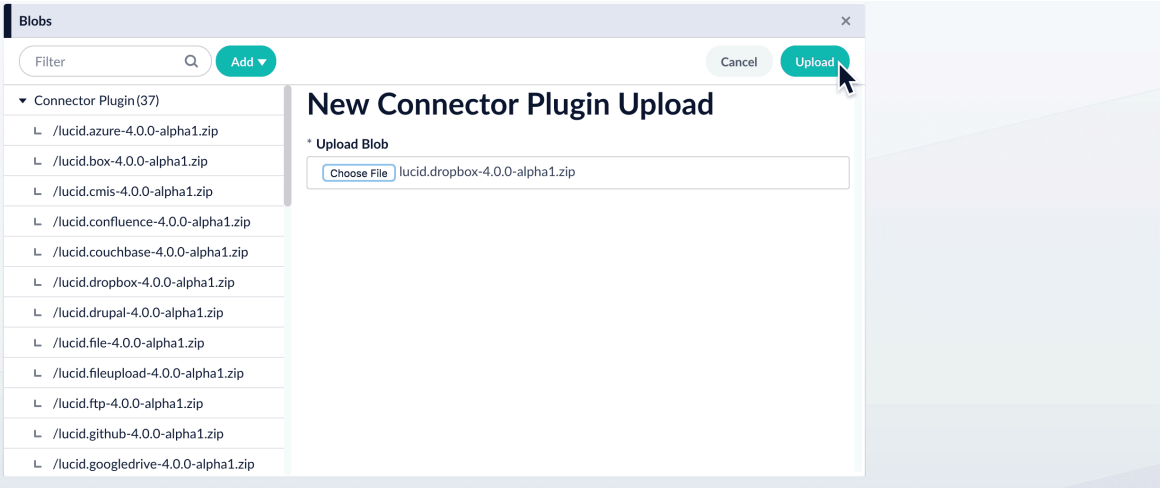
Do not expand the archive; Fusion consumes it as-is.

2. In the Fusion workspace, navigate to **System** > **Blobs**.
3. Click **Add**.
4. Select **Connector Plugin**.

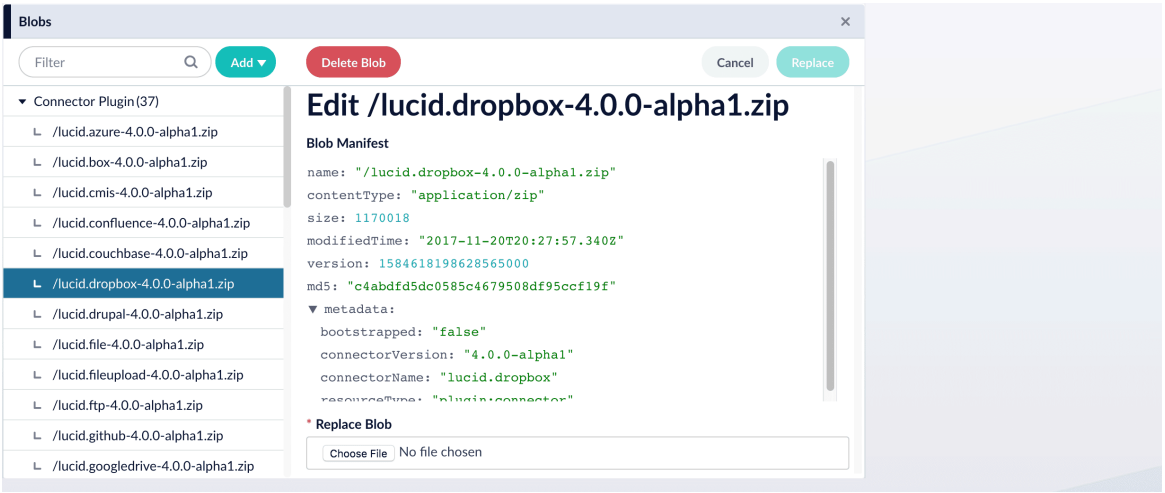


The "New Connector Plugin Upload" panel appears.

5. Click **Choose File** and select the downloaded zip file from your file system.



6. Click **Upload**.
   The new connector's blob manifest appears.



From this screen you can also delete or replace the connector.

Warning

Wait several minutes for the connector to finish uploading to the blob store before installing the connector using the *Datasources dropdown*.

Install or update a connector using the Connector API

1. Download the connector zip file.

Note

Do not expand the archive; Managed Fusion consumes it as-is.

2. Upload the connector zip file to Managed Fusion's plugins.
3. Run the following command with your information, including the `pluginId` :

```
curl -H 'content-type:application/zip' -u USERNAME:PASSWORD -X
PUT
https://EXAMPLE_COMPANY.b.lucidworks.cloud/api/connectors/plugin
s?id=lucidworks.{pluginId}' --data-binary @{plugin_path}.zip
```

Managed Fusion automatically publishes the event to the cluster, and the listeners perform the connector installation process on each node.

Tip

If the `pluginId` is identical to an existing one, the old connector is uninstalled and the new connector is installed in its place. To get the list of existing plugin IDs, run the following command: `curl -u USERNAME:PASSWORD https://EXAMPLE_COMPANY.b.lucidworks.cloud/api/connectors/plugins`

4. Verify the new connector is installed in `https://EXAMPLE_COMPANY.b.lucidworks.cloud/apps/connectors/plugins/`.

Reinstall a connector

To reinstall a connector for any reason, first *delete the connector* then use the preceding steps to install it again. This may take a few minutes to complete depending on how quickly the pods are deleted and recreated.

Install or update a connector - 5.2 and later

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion Connectors*, *Fusion*

When you create a new datasource that requires an uninstalled connector, Fusion releases 5.2 and later automatically download and install the connector using the *Datasources dropdown*. You can also update the connector using the *Blob Store UI* or via the *Connector API*.

Install a connector using the Datasources dropdown

1. In your Fusion app, navigate to **Indexing** > **Datasources**.
2. Click **Add**.
3. In the list of connectors, scroll down to the connectors marked **Not Installed** and select the one you want to install.
   Fusion automatically downloads it and moves it to the list of installed connectors.

After you install a connector, you can *Configure a New Datasource*.

Tip

You can view and download all current and previous V2 connector releases at *Download Connectors*.

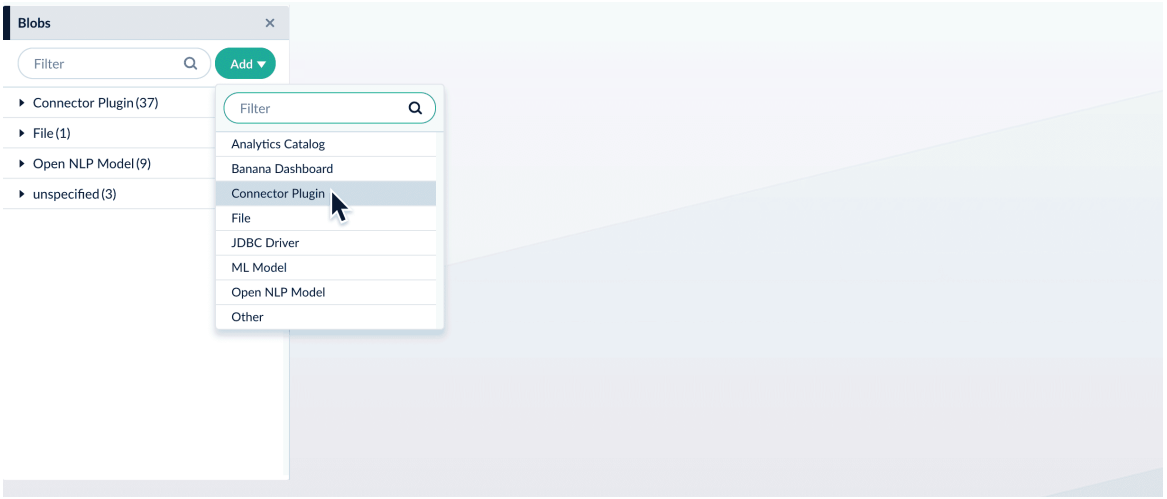
Install or update a connector using the Blob Store UI

1. Download the connector zip file from *Download V2 connectors*.

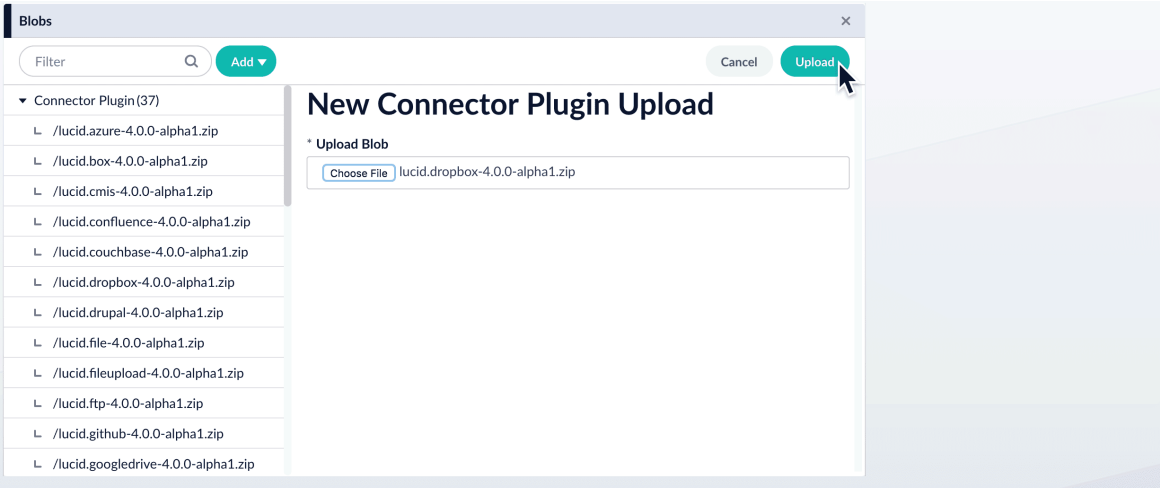
   Note

   Do not expand the archive; Fusion consumes it as-is.

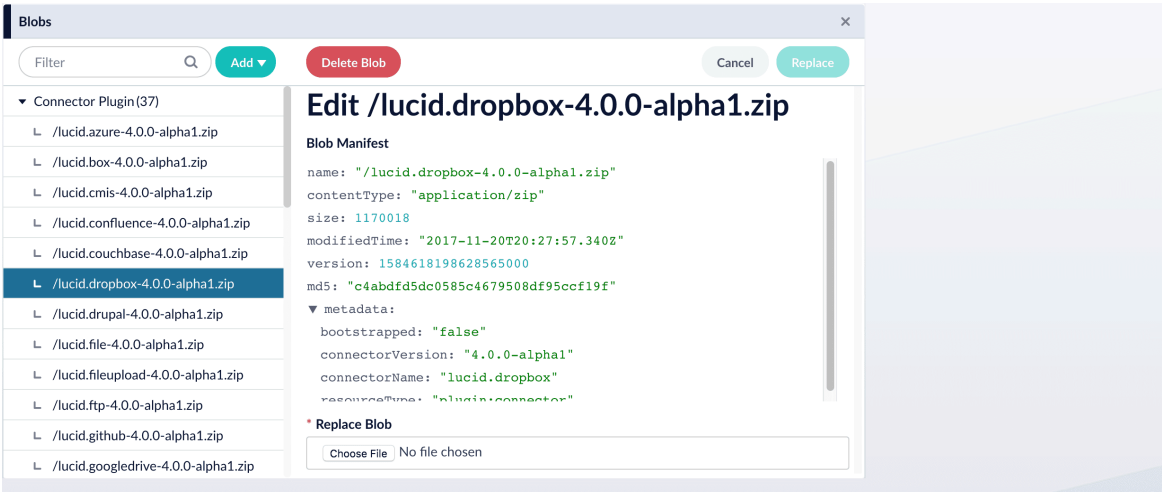
2. In your Fusion app, navigate to **System** > **Blobs**.
3. Click **Add**.
4. Select **Connector Plugin**.



   The "New Connector Plugin Upload" panel appears.
5. Click **Choose File** and select the downloaded zip file from your file system.



6. Click **Upload**.
   The new connector's blob manifest appears.



From this screen you can also delete or replace the connector.

Warning

Wait several minutes for the connector to finish uploading to the blob store before installing the connector using the *Datasources dropdown*.

Install or update a connector using the Connector API

1. Download the connector zip file from *Download V2 connectors*.

Note

Do not expand the archive; Fusion consumes it as-is.

2. Upload the connector zip file to Fusion's plugins. Specify a `pluginId` as in this example:

```
curl -H 'content-type:application/zip' -u USERNAME:PASSWORD -X
PUT 'https://FUSION_HOST:FUSION_PORT/api/connectors/plugins?
id=lucidworks.{pluginId}' --data-binary @{plugin_path}.zip
```

Fusion automatically publishes the event to the cluster, and the listeners perform the connector installation process on each node.

Tip

If the `pluginId` is identical to an existing one, the old connector will be uninstalled and the new connector will be installed in its place. To get the list of existing plugin IDs, run: `curl -u USERNAME:PASSWORD https://FUSION_HOST:FUSION_PORT/api/connectors/plugins`

3. Look in `https://FUSION_HOST:FUSION_PORT/apps/connectors/plugins/` to verify the new connector is installed.

Reinstall a connector

To reinstall a connector for any reason, first *delete the connector* then use the preceding steps to install it again. This may take a few minutes to complete depending on how quickly the pods are deleted and recreated.

Install the Jive Add-On for Fusion

How-to

> Compatible versions: 4.2, 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion Connectors*, *Fusion Server*, *Fusion*

Retrieve content from a Jive instance.

The Jive add-on for Fusion lets you use OAuth to authenticate the Fusion connector to the Jive API. This is a recommended way to use the Jive API. The add-on retrieves a client ID and secret, which are supplied to the Fusion connector. See the Jive product documentation.

If the client ID and secret cannot be retrieved or used, then Basic authentication is used as a fallback.

Install the Jive Add-on

1. Install the Fusion Jive connector using the Fusion UI.
    a. In the Fusion UI, navigate to **Indexing > Datasources**.
    b. Click **Add**.
    c. Follow the UI prompts to install the Jive connector. For more information, see *Managing Connectors in Fusion 5.x* or *Managing Connectors in Fusion 4.x*.
2. Following the instructions from the Jive product documentation, use the Jive UI to upload the add-on from your Fusion distribution.
    a. On Linux, the Jive add-on is at

```
$FUSION_HOME/apps/connectors/connectors-classic/plugins/lucid.jive/assets/lucidworks-jive-addon.zip
```

    b. On Windows, the Jive add-on is at

```
%FUSION_HOME%\apps\connectors\connectors-classic\plugins\lucid.jive\assets\lucidworks-jive-addon.zip
```

Install the Jive Add-On for Managed Fusion

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

Retrieve content from a Jive instance.

The Jive add-on for Managed Fusion lets you use OAuth to authenticate the Managed Fusion connector to the Jive API. This is a recommended way to use the Jive API. The add-on retrieves a client ID and secret, which are supplied to the Managed Fusion connector. See the Jive product documentation.

If the client ID and secret cannot be retrieved or used, then Basic authentication is used as a fallback.

Install the Jive Add-on

1. Install the Managed Fusion Jive connector using the Managed Fusion UI.
    a. In the Fusion UI, navigate to **Indexing > Datasources**.
    b. Click **Add**.
    c. Follow the UI prompts to install the Jive connector. For more information, see *Install a Connector in Managed Fusion*.
2. Following the instructions from the Jive product documentation, use the Jive UI to upload the add-on from your Fusion distribution.
    a. On Linux, the Jive add-on is at

```
$FUSION_HOME/apps/connectors/connectors-classic/plugins/lucid.jive/assets/lucidworks-jive-addon.zip
```

    b. On Windows, the Jive add-on is at

```
%FUSION_HOME%\apps\connectors\connectors-classic\plugins\lucid.jive\assets\lucidworks-jive-addon.zip
```

Install the Random Content Connector

How-to

> Compatible versions: 4.2, 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion Connectors*, *Fusion Server*, *Fusion*

The Connectors SDK comes with a sample connector to help you.

Random Content Connector

The Random Content connector generates a configurable number of documents with random titles and body fields.

This connector generates a configurable number of documents, all with random titles and body fields.

Quick start

1. Clone the repo:

```
git clone https://github.com/lucidworks/connectors-sdk-
resources.git
cd connectors-sdk-resources/java-sdk/connectors/
./gradlew assemblePlugins
```

2. This produces one zip file, named `simple-connector.zip`, located in the `build/plugins` directory. This artifact is now ready to be uploaded directly to Fusion as a connector plugin.
3. See the following *instructions* on how to build, deploy, and run the plugin.

Connector properties

RANDOM CONTENT GENERATOR PROPERTIES

Property Name	Property description
Total	The total number of documents to generate
Minimum number of sentences	The minimum number of sentences to generate per document, the random generator will use this value as lower bound to calculate a random number of sentences
maximum number of sentences	The maximum number of sentences to generate per document, the random generator will use this value as upper bound to calculate a random number of sentences

How to use the connector

- Create a configuration with the properties listed above.
- After the first job is completed, the connector will index the same number of documents as defined in the `Random content generator properties.Total` property.

Install the Random Content Connector

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

The Connectors SDK comes with a sample connector to help you.

Random Content Connector

The Random Content connector generates a configurable number of documents with random titles and body fields.

Quick start

1. Clone the repo:

```
git clone https://github.com/lucidworks/connectors-sdk-
resources.git cd connectors-sdk-resources/java-sdk/connectors/
./gradlew assemblePlugins
```

2. This produces one zip file, named `simple-connector.zip`, located in the `build/plugins` directory. This artifact is now ready to be uploaded directly to Managed Fusion as a connector plugin.

Connector properties

RANDOM CONTENT GENERATOR PROPERTIES

Property Name	Property description
Total	The total number of documents to generate
Minimum number of sentences	The minimum number of sentences to generate per document, the random generator will use this value as lower bound to calculate a random number of sentences
maximum number of sentences	The maximum number of sentences to generate per document, the random generator will use this value as upper bound to calculate a random number of sentences

How to use the connector

- Create a configuration with the properties listed above.
- After the first job is completed, the connector will index the same number of documents as defined in the `Random content generator properties.Total` property.

Install the Spark History Server

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion*

While logs from the Spark driver and executor pods can be viewed using `kubectl logs [POD_NAME]`, executor pods are deleted at their end of their execution, and driver pods are deleted by Fusion on a default schedule of every hour. In order to preserve and view Spark logs, install the Spark History Server into your Kubernetes cluster and configure Spark to write logs in a manner that suits your needs.

Spark History Server can be installed via its publicly available Helm chart. To do this, create a `values.yaml` file to configure it:

```
helm install [release-name] [chart] --namespace [fusion-namespace] --
values [spark-history-values-yaml]
```

For related topics, see *Spark Operations*.

Integrate Fusion 4.x with an Existing Solr Deployment

How-to

> Compatible versions: 4.2
>
> Works with: *Fusion Server*

If you have already implemented Solr as a standalone instance or as a SolrCloud cluster, you can add Fusion to your existing Solr deployment (if the *Solr version is supported*) and import your Solr collections into Fusion. Each Fusion collection can import one Solr collection.

- *If your existing Solr instance is running in SolrCloud mode*, you can use Fusion's UI to modify configuration files (such as schema.xml or solrconfig.xml) and create Solr collections.
- *If your existing Solr instance is running in standalone mode*, you can still connect it to Fusion. Fusion can send documents to a standalone Solr instance and query the instance. But you will not be able to use Fusion's UI to create Solr collections (Solr cores) or to modify Solr configuration files.

Prerequisites

Warning

Failures in the Fusion install or startup may occur if the Fusion installation directory contains a space.

- You have already *installed Fusion*.
- You have already installed Solr, which must meet these *Solr requirements*.
- You have already installed ZooKeeper, which must meet these *ZooKeeper requirements*.

Note

We recommend that you create an *external* ZooKeeper cluster (external to both Fusion and SolrCloud).

- Your Solr deployment must contain one or more collections (cores).
- In SolrCloud mode, Solr must be configured to use ZooKeeper.

Configure Fusion to use an existing Solr deployment

Use the *Fusion UI* to integrate Fusion with an existing Solr deployment.

Use the Fusion UI

1. Create a Fusion search cluster:
    a. In the Fusion UI, navigate to **System > Solr Clusters** and click **New Solr Cluster**.
    b. Enter this information:
        - A cluster ID of your choice
        - Whether SolrCloud is enabled
        - The connect string (to tell Fusion how to connect to the SolrCloud cluster or Solr instance)
            - For SolrCloud, this is the ZooKeeper connect string.
            - For a standalone Solr instance, this is the URL of the Solr instance.

    c. Verify that the connection is working by clicking **Cores** in the new cluster and inspecting the contents.
2. Create a Fusion *collection* that points to your Solr cluster and collection:
    a. In the UI, navigate to **Collections** and click **Add a Collection**.
    b. Enter a name for the new collection.
    c. Click **Advanced**.
    d. Select your SolrCloud cluster or Solr instance from the dropdown.
    e. Enter the name of the Solr collection to import.

Sending Documents to Solr through Fusion

You can use the Fusion *connectors* to crawl documents and index them to your existing Solr deployment.

1. Follow the steps above to create and configure a search cluster and a collection that points to Solr.
2. Define an index pipeline that ends with a *Solr Indexer stage* that sends the documents to Solr.
3. Use one of these methods to ingest your data:
    - In the collection that points to your Solr collection, define a *datasource* using the *connector* of choice.
    - Send prepared documents directly to the index pipeline for processing. See *Importing Data with the REST API*.
    - It is also possible to use a different indexing process besides a connector, such as a script that sends documents through the index pipeline.

When documents are sent to Solr, a buffering SolrServer is used. Buffering the updates reduces the number of HTTP requests made from Fusion to Solr, which can significantly affect processing time. For example, when processing simple documents, you should always try to buffer as many documents as possible to increase throughput. When processing complex documents, you should use small batch sizes. You should only turn buffering off if you are using an older version of Solr and you want Fusion to catch and document indexing errors.

Querying Solr via Fusion requests

Indexed documents are stored in Solr indexes. You can query for these documents by using query pipelines. The query pipelines let you define your query parameters - such as how many records to return, the fields you would like, how to structure facets, and so on. You also have the ability to add JavaScript to the response processing, and define landing pages or specific boost levels depending on the user's query. See *Query Pipelines*.

If you prefer, you can also use the *Solr API* and *SolrAdmin API* to query Solr directly.

Integrate Fusion with Your Logging Infrastructure

How-to

> Compatible versions: 5.5
>
> Works with: *Fusion*

Fusion services come pre-configured to write log messages to stdout and index log messages into the Fusion `system_logs` collection using a Pulsar message queue processed by the `fusion-log-forwarder` component.

The Log Viewer, integrated into the Fusion Admin UI, provides basic log analytics for the logs collected in `system_logs`. You can also use your preferred log analytics stack, such as Elastic Stack, Splunk, or DataDog.

For integration with an existing log infrastructure, you have two basic choices:

- *Disable log forwarding and scrape logs from stdout*
- *Use Logstash to send logs*

Disable Log forwarding and scrape logs from stdout

1. In your custom values YAML files, set the global logging `disablePulsar` value to `true`:

```
global: logging: disablePulsar: true
```

2. Configure your logging infrastructure to scrape logs from the stdout from each pod.

Tip

It's possible to set up Fusion to output JSON logs to stdout to ease configuration of log forwarders. This is a common pattern in Kubernetes, and most modern log analytics solutions have good integration with Kubernetes.

To do this, set the global logging `jsonOutput` value to `true`:

```
global: logging: jsonOutput: true
```

Use Logstash to send logs

Fusion can be configured to send logs directly to a user's pre-configured logstash instance. This is a good option if you use ELK, but you don't have an existing integration with Kubernetes.

To do this, set the global logging `logstashHost` value to match your `logstash_host` value:

```
global: logging: logstashHost: <logstash_host>
```

Join Signals with Item Metadata

How-to

> Compatible versions: 4.2, 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion AI*, *Fusion Server*, *Fusion*

Fusion's basic aggregation jobs aggregate using the document ID. You can also aggregate at a more coarse-grained level using other fields available for documents (item metadata), such as manufacturer or brand for products. Aggregating with item metadata is useful for building personalization boosts into your search application.

The following PUT request creates additional aggregation jobs that join signals with the primary `products` collection to compute an aggregated weight for a `manufacturer` field:

```
curl -X PUT -H "Content-type:application/json" -d '{"enabled":true,
"metadata_column":"manufacturer"}' "http://localhost:{api-
port}/api/collections/products/features/signals"
```

After performing the PUT request shown above, you will have two additional aggregation jobs in Fusion.

COLLECTION_NAME_user_METADATA_preferences_aggregation

This job computes an aggregated weight for each user/item metadata combination, e.g. user/manufacturer, found in the signals collection. Fusion computes the weight for each group using an exponential time-decay on signal count (30 day half-life) and a weighted sum based on the signal type. Use the `signalTypeWeights` parameter to set the correct signal types and weights for your dataset. Use the `primaryCollectionMetadataField` parameter to set the name of a field from the primary collection to join into the results, e.g. `manufacturer`. You can use the results of this job to boost queries based on user preferences regarding item-specific metadata such as manufacturer (e.g. Ford vs. BMW) or brand (e.g. Ralph Lauren vs. Abercrombie & Fitch).

COLLECTION_NAME_query_METADATA_preferences_aggregation

This job computes an aggregated weight for each query/item metadata combination, e.g. query/manufacturer, found in the signals collection. Fusion computes the weight for each group using an exponential time-decay on signal count (30 day half-life) and a weighted sum based on the signal type. Use the `signalTypeWeights` parameter to set the correct signal types and weights for your dataset. Use the `primaryCollectionMetadataField` parameter to set the name of a field from the primary collection to join into the results.

> **Tip**
>
> These additional item item/metadata aggregation jobs also serve as examples of how to join between the signals and primary collections to perform aggregations on fields other than the document ID. You can re-execute the same PUT request shown above using a different metadata field name in the `metadata_column` parameter.

Join Signals with Item Metadata

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

Managed Fusion's basic aggregation jobs aggregate using the document ID. You can also aggregate at a more coarse-grained level using other fields available for documents (item metadata), such as manufacturer or brand for products. Aggregating with item metadata is useful for building personalization boosts into your search application.

The following PUT request creates additional aggregation jobs that join signals with the primary `products` collection to compute an aggregated weight for a `manufacturer` field:

```
curl -X PUT -H "Content-type:application/json" -d '{"enabled":true,
"metadata_column":"manufacturer"}'
"https://EXAMPLE_COMPANY.b.lucidworks.cloud/api/collections/products/f
eatures/signals"
```

After performing the PUT request shown above, you will have two additional aggregation jobs in Managed Fusion.

COLLECTION_NAME_user_METADATA_preferences_aggregation

This job computes an aggregated weight for each user/item metadata combination, for example, user/manufacturer, found in the signals collection. Managed Fusion computes the weight for each group using an exponential time-decay on signal count (30 day half-life) and a weighted sum based on the signal type. Use the `signalTypeWeights` parameter to set the correct signal types and weights for your dataset. Use the `primaryCollectionMetadataField` parameter to set the name of a field from the primary collection to join into the results, for example, `manufacturer`. You can use the results of this job to boost queries based on user preferences regarding item-specific metadata such as manufacturer (for example, Ford vs. BMW) or brand (for example, Ralph Lauren vs. Abercrombie & Fitch).

COLLECTION_NAME_query_METADATA_preferences_aggregation

This job computes an aggregated weight for each query/item metadata combination, for example, query/manufacturer, found in the signals collection. Managed Fusion computes the weight for each group using an exponential time-decay on signal count (30 day half-life) and a weighted sum based on the signal type. Use the `signalTypeWeights` parameter to set the correct signal types and weights for your dataset. Use the `primaryCollectionMetadataField` parameter to set the name of a field from the primary collection to join into the results.

Tip

These additional item item/metadata aggregation jobs also serve as examples of how to join between the signals and primary collections to perform aggregations on fields other than the document ID. You can re-execute the same PUT request shown above using a different metadata field name in the `metadata_column` parameter.

Keep Rules Consistent Between Clusters

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion*

This article teaches you how to keep business rules consistent between clusters. Use a `PUT` API call to the *Query Rewrite* API's `/query-rewrite/instances` endpoint for bulk operations. The JSON payload sent with the `PUT` call must adhere to the following format:

```
{ "create": [<object array>], "update": [<object array>], "delete": [<item array>] }
```

- `create`. If the rule does not exist, *create it*.
- `update`. If the rule exists, *update it* to use the latest values.
- `delete`. If the rule exists, *delete it*.

Fusion generates a unique `id` value for every rule. The API operations use this value as a reference point.

The instructions in the sections below use these example values:

Parameter	Value
Development cluster	
Address	**https://FUSION_HOST_DEV:6764**
Username	**DEV_USER**
Password	**DEV_PASSWORD**
App name	**DEV_APP_NAME**
Production cluster	
Address	**https://FUSION_HOST_PROD:6764**
Username	**PROD_USER**
Password	**PROD_PASSWORD**
App name	**PROD_APP_NAME**

Note

When you use the API to conduct bulk rules operations, you must publish the changes in the UI.

1. Log into Fusion with the same user credentials that you used to make the bulk API rules operations.
2. Navigate to **Relevance > Rules** to access the rules editor.
3. Click the **Publish** button.

Export rules

In these examples, you'll export rules from an app on a development cluster. This is useful if you have a large number of rule changes you need to port from a development cluster to a production cluster.

All rules

Export all rules from the development cluster by sending a `GET` API call to the `/query-rewrite/instances` endpoint:

```
curl -X 'GET' \ -u DEV_USER : DEV_PASSWORD \
' https://FUSION_HOST_DEV:6764 /api/apps/ DEV_APP_NAME /query-
rewrite/instances' \ -H 'accept: application/json' \ > exported-
rules.json
```

This outputs your rules to a new JSON file, **exported-rules.json**. The result resembles the following:

```
[{ "type": "pinned", "id": "NqYec4R7pG", "name": "Featured - Fusion
Server 4.2", "createdOn": "2021-06-21T17:28:19.800Z", "updatedOn":
"2021-06-21T17:28:19.800Z", "enabled": true, "matching": "keywords",
"priority": 1, "search_terms": ["*:*"], ... }, { "type": "pinned",
"id": "uEvL1NBXba", "name": "Featured - Fusion Server 4.1",
"createdOn": "2021-06-21T17:29:59.863Z", "updatedOn": "2021-06-
21T17:29:59.863Z", "enabled": true, "matching": "keywords",
"priority": 1, "search_terms": ["*:*"], ... }]
```

Specific rules

You can specify which rules you want to export by adding parameters to the API endpoint. For example, if you want to export synonym rules only, send a `GET` API call to `/query-rewrite/instances?type=synonym`. If you want to export a list of rules that are disabled, use `/query-rewrite/instances?enabled=false`.

Use the same `GET` API call, but change the endpoint:

```
curl -X 'GET' \ -u DEV_USER : DEV_PASSWORD \
' https://FUSION_HOST_DEV:6764 /api/apps/ DEV_APP_NAME /query-
rewrite/instances?type=synonym' \ -H 'accept: application/json' \ >
exported-rules.json
```

Use `create`

In this example, you'll recreate the rules from your development cluster on your production cluster.

1. Manually reformat the JSON so it uses the following format:

```
{ "create": [<object array>] }
```

We are recreating the development cluster's rules on the production cluster, so we can use `create` only. Replace `[<object array>]` with the contents of **exported-rules.json**. For example:

```
{ "create": [{ "type": "pinned", "id": "NqYec4R7pG", "name":
"Featured - Fusion Server 4.2", "createdOn": "2021-06-
21T17:28:19.800Z", "updatedOn": "2021-06-21T17:28:19.800Z",
"enabled": true, "matching": "keywords", "priority": 1,
"search_terms": ["*:*"], ... }, { "type": "pinned", "id":
"uEvL1NBXba", "name": "Featured - Fusion Server 4.1",
"createdOn": "2021-06-21T17:29:59.863Z", "updatedOn": "2021-06-
21T17:29:59.863Z", "enabled": true, "matching": "keywords",
"priority": 1, "search_terms": ["*:*"], ... }, ... ]}
```

2. Save the file.
3. Send a `PUT` API call to the `/query-rewrite/instances` endpoint to recreate the rules from the development cluster on the production cluster:

```
curl -X 'PUT' \ -u PROD_USER : PROD_PASSWORD \
' https://FUSION_HOST_PROD:6764 /api/apps/ PROD_APP_NAME /query-
rewrite/instances' \ -H 'accept: application/octet-stream' \ -H
'Content-Type: application/json' \ -d @exported-rules.json
```

Tip

You may have to specify the absolute path to your rules JSON file. For example, `-d @/Users/johndoe/exported-rules.json`.

4. Verify that the `PUT` API was successful by checking the Fusion UI or *fetching the rules with the API*.

Use `update`

1. Manually reformat the JSON so it uses the following format:

```
{ "update": [<object array>] }
```

Replace `[<object array>]` with the contents of **exported-rules.json**. For example:

```
{ "update": [{ "type": "pinned", "id": "NqYec4R7pG", "name":
"Featured - Fusion Server 4.2", "createdOn": "2021-06-
21T17:28:19.800Z", "updatedOn": "2021-06-21T17:28:19.800Z",
"enabled": true, "matching": "keywords", "priority": 1,
"search_terms": ["*:*"], ... }, { "type": "pinned", "id":
"uEvL1NBXba", "name": "Featured - Fusion Server 4.1",
"createdOn": "2021-06-21T17:29:59.863Z", "updatedOn": "2021-06-
21T17:29:59.863Z", "enabled": true, "matching": "keywords",
"priority": 1, "search_terms": ["*:*"], ... }, ... ]}
```

2. Save the file.
3. Send a `PUT` API call to the `/query-rewrite/instances` endpoint to update the production cluster with the rules from the development cluster:

```
curl -X 'PUT' \ -u PROD_USER : PROD_PASSWORD \
' https://FUSION_HOST_PROD:6764 /api/apps/ PROD_APP_NAME /query-
rewrite/instances' \ -H 'accept: application/octet-stream' \ -H
'Content-Type: application/json' \ -d @exported-rules.json
```

Use `delete`

In some cases, you may want to delete rules in bulk. For example, if you *created rules* using the steps above but decide you want to undo that action.

The `delete` bulk operation uses a different format than `create` and `update`. Instead of passing the entire rules JSON as an object array, you must pass the rule `id` values as an item array.

1. Manually reformat the JSON so it uses the following format:

```
{ "delete": [<item array>] }
```

Replace `<item array>` with the `id` values of the rules you want to delete. These values must be comma separated and wrapped in double quotes (`"`). For example:

```
{ "delete": ["NqYec4R7pG", "uEvL1NBXba", ... ]}
```

2. Save the file.
3. Send a `PUT` API call to the `/query-rewrite/instances` endpoint to update the production cluster with the rules from the development cluster:

```
curl -X 'PUT' \ -u PROD_USER : PROD_PASSWORD \
' https://FUSION_HOST_PROD:6764 /api/apps/ PROD_APP_NAME /query-
rewrite/instances' \ -H 'accept: application/octet-stream' \ -H
'Content-Type: application/json' \ -d @exported-rules.json
```

Use `create` *with* `update`

You may want to determine which rules to include in the `create` array and which to include in the `update` array, but it can be difficult to determine which rules already exist. Fortunately, the operations don't conflict:

- `create`. If the rule does not exist, *create it*.
- `update`. If the rule exists, *update it* to use the latest values.

There's no need to separate the rules. You can replace `[<object array>]` for `create` and `update` with the *same* rules JSON to create the missing rules and update the existing rules.

1. Manually reformat the JSON so it uses the following format:

```
{ "create": [<object array>], "update": [<object array>] }
```

Replace `[<object array>]` with the contents of **exported-rules.json**. For example:

```
{ "create": [{ "type": "pinned", "id": "NqYec4R7pG", "name":
"Featured - Fusion Server 4.2", "createdOn": "2021-06-
21T17:28:19.800Z", "updatedOn": "2021-06-21T17:28:19.800Z",
"enabled": true, "matching": "keywords", "priority": 1,
"search_terms": ["*:*"], ... } ... ], "update": [{ "type":
"pinned", "id": "NqYec4R7pG", "name": "Featured - Fusion Server
4.2", "createdOn": "2021-06-21T17:28:19.800Z", "updatedOn":
"2021-06-21T17:28:19.800Z", "enabled": true, "matching":
"keywords", "priority": 1, "search_terms": ["*:*"], ... } ... ]
}
```


2. Save the file.
3. Send a **`PUT`** API call to the **`/query-rewrite/instances`** endpoint to update the production cluster with the rules from the development cluster:

```
curl -X 'PUT' \ -u PROD_USER : PROD_PASSWORD \
' https://FUSION_HOST_PROD:6764 /api/apps/ PROD_APP_NAME /query-
rewrite/instances' \ -H 'accept: application/octet-stream' \ -H
'Content-Type: application/json' \ -d @exported-rules.json
```

Keep Rules Consistent Between Clusters

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

This article teaches you how to keep business rules consistent between clusters. Use a `PUT` API call to the *Query Rewrite API's* `/query-rewrite/instances` endpoint for bulk operations. The JSON payload sent with the `PUT` call must adhere to the following format:

```
{ "create": [<object array>], "update": [<object array>], "delete":
[<item array>] }
```

- `create`. If the rule does not exist, *create it*.
- `update`. If the rule exists, *update it* to use the latest values.
- `delete`. If the rule exists, *delete it*.

Fusion generates a unique `id` value for every rule. The API operations use this value as a reference point.

The instructions in the sections below use these example values:

Parameter	Value
Development cluster	
Address	`https://EXAMPLE_COMPANY_PROD.b.lucidworks.cloud`
Username	`DEV_USER`
Password	`DEV_PASSWORD`
App name	`DEV_APP_NAME`
Production cluster	
Address	`https://EXAMPLE_COMPANY_PROD.b.lucidworks.cloud`
Username	`PROD_USER`
Password	`PROD_PASSWORD`
App name	`PROD_APP_NAME`

Note

When you use the API to conduct bulk rules operations, you must publish the changes in the UI.

1. Log into Fusion with the same user credentials that you used to make the bulk API rules operations.
2. Navigate to **Relevance > Rules** to access the rules editor.
3. Click the **Publish** button.

Export rules

In these examples, you'll export rules from an app on a development cluster. This is useful if you have a large number of rule changes you need to port from a development cluster to a production cluster.

All rules

Export all rules from the development cluster by sending a `GET` API call to the `/query-rewrite/instances` endpoint:

```
curl -X 'GET' \ -u DEV_USER:DEV_PASSWORD \
'https://EXAMPLE_COMPANY_PROD.b.lucidworks.cloud/api/apps/DEV_APP_NAME/query-rewrite/instances' \ -H 'accept: application/json' \ > exported-rules.json
```

This outputs your rules to a new JSON file, **exported-rules.json**. The result resembles the following:

```
[{ "type": "pinned", "id": "NqYec4R7pG", "name": "Featured - Managed Fusion", "createdOn": "2021-06-21T17:28:19.800Z", "updatedOn": "2021-06-21T17:28:19.800Z", "enabled": true, "matching": "keywords", "priority": 1, "search_terms": ["*:*"], ... }, { "type": "pinned", "id": "uEvL1NBXba", "name": "Featured - Managed fusion", "createdOn": "2021-06-21T17:29:59.863Z", "updatedOn": "2021-06-21T17:29:59.863Z", "enabled": true, "matching": "keywords", "priority": 1, "search_terms": ["*:*"], ... }]
```

Specific rules

You can specify which rules you want to export by adding parameters to the API endpoint. For example, if you want to export synonym rules only, send a `GET` API call to `/query-rewrite/instances?type=synonym`. If you want to export a list of rules that are disabled, use `/query-rewrite/instances?enabled=false`.

Use the same `GET` API call, but change the endpoint:

```
curl -X 'GET' \ -u DEV_USER:DEV_PASSWORD \
'https://EXAMPLE_COMPANY_PROD.b.lucidworks.cloud/api/apps/DEV_APP_NAME/query-rewrite/instances?type=synonym' \ -H 'accept: application/json' \ > exported-rules.json
```

Use `create`

In this example, you'll recreate the rules from your development cluster on your production cluster.

1. Manually reformat the JSON so it uses the following format:

```
{ "create": [<object array>] }
```

We are recreating the development cluster's rules on the production cluster, so we can use `create` only. Replace `[<object array>]` with the contents of **exported-rules.json**. For example:

```
{ "create": [{ "type": "pinned", "id": "NqYec4R7pG", "name":
"Featured - Managed Fusion", "createdOn": "2021-06-
21T17:28:19.800Z", "updatedOn": "2021-06-21T17:28:19.800Z",
"enabled": true, "matching": "keywords", "priority": 1,
"search_terms": ["*:*"], ... }, { "type": "pinned", "id":
"uEvL1NBXba", "name": "Featured - Managed Fusion", "createdOn":
"2021-06-21T17:29:59.863Z", "updatedOn": "2021-06-
21T17:29:59.863Z", "enabled": true, "matching": "keywords",
"priority": 1, "search_terms": ["*:*"], ... }, ... ]}
```

2. Save the file.
3. Send a `PUT` API call to the `/query-rewrite/instances` endpoint to recreate the rules from the development cluster on the production cluster:

```
curl -X 'PUT' \ -u PROD_USER : PROD_PASSWORD \
' https://EXAMPLE_COMPANY_DEV.b.lucidworks.cloud /api/apps/ PROD
_APP_NAME /query-rewrite/instances' \ -H 'accept:
application/octet-stream' \ -H 'Content-Type: application/json'
\ -d @exported-rules.json
```

Tip

You may have to specify the absolute path to your rules JSON file. For example, `-d @/Users/johndoe/exported-rules.json`.

4. Verify that the `PUT` API was successful by checking the Fusion UI or *fetching the rules with the API*.

Use `update`

1. Manually reformat the JSON so it uses the following format:

```
{ "update": [<object array>] }
```

Replace `[<object array>]` with the contents of **exported-rules.json**. For example:

```
{ "update": [{ "type": "pinned", "id": "NqYec4R7pG", "name":
"Featured – Managed", "createdOn": "2021-06-21T17:28:19.800Z",
"updatedOn": "2021-06-21T17:28:19.800Z", "enabled": true,
"matching": "keywords", "priority": 1, "search_terms": ["*:*"],
... }, { "type": "pinned", "id": "uEvL1NBXba", "name": "Featured
– Managed Fusion", "createdOn": "2021-06-21T17:29:59.863Z",
"updatedOn": "2021-06-21T17:29:59.863Z", "enabled": true,
"matching": "keywords", "priority": 1, "search_terms": ["*:*"],
... }, ... ]}
```

2. Save the file.
3. Send a `PUT` API call to the `/query-rewrite/instances` endpoint to update the production cluster with the rules from the development cluster:

```
curl -X 'PUT' \ -u PROD_USER : PROD_PASSWORD \
' https://EXAMPLE_COMPANY_DEV.b.lucidworks.cloud /api/apps/ PROD
_APP_NAME /query-rewrite/instances' \ -H 'accept:
application/octet-stream' \ -H 'Content-Type: application/json'
\ -d @exported-rules.json
```

Use `delete`

In some cases, you may want to delete rules in bulk. For example, if you *created rules* using the steps above but decide you want to undo that action.

The `delete` bulk operation uses a different format than `create` and `update`. Instead of passing the entire rules JSON as an object array, you must pass the rule `id` values as an item array.

1. Manually reformat the JSON so it uses the following format:

```
{ "delete": [<item array>] }
```

   Replace `<item array>` with the `id` values of the rules you want to delete. These values must be comma separated and wrapped in double quotes (`"`). For example:

```
{ "delete": ["NqYec4R7pG", "uEvL1NBXba", ... ]}
```

2. Save the file.
3. Send a `PUT` API call to the `/query-rewrite/instances` endpoint to update the production cluster with the rules from the development cluster:

```
curl -X 'PUT' \ -u PROD_USER : PROD_PASSWORD \
' https://EXAMPLE_COMPANY_DEV.b.lucidworks.cloud /api/apps/ PROD
_APP_NAME /query-rewrite/instances' \ -H 'accept:
application/octet-stream' \ -H 'Content-Type: application/json'
\ -d @exported-rules.json
```

Use `create` *with* `update`

You may want to determine which rules to include in the `create` array and which to include in the `update` array, but it can be difficult to determine which rules already exist. Fortunately, the operations don't conflict:

- `create`. If the rule does not exist, *create it*.
- `update`. If the rule exists, *update it* to use the latest values.

There's no need to separate the rules. You can replace `[<object array>]` for `create` and `update` with the *same* rules JSON to create the missing rules and update the existing rules.

1. Manually reformat the JSON so it uses the following format:

```
{ "create": [<object array>], "update": [<object array>] }
```

Replace `[<object array>]` with the contents of **exported-rules.json**. For example:

```
{ "create": [{ "type": "pinned", "id": "NqYec4R7pG", "name":
"Featured - Managed Fusion", "createdOn": "2021-06-
21T17:28:19.800Z", "updatedOn": "2021-06-21T17:28:19.800Z",
"enabled": true, "matching": "keywords", "priority": 1,
"search_terms": ["*:*"], ... } ... ], "update": [{ "type":
"pinned", "id": "NqYec4R7pG", "name": "Featured - Managed
Fusion", "createdOn": "2021-06-21T17:28:19.800Z", "updatedOn":
"2021-06-21T17:28:19.800Z", "enabled": true, "matching":
"keywords", "priority": 1, "search_terms": ["*:*"], ... } ... ]
}
```

2. Save the file.
3. Send a `PUT` API call to the `/query-rewrite/instances` endpoint to update the production cluster with the rules from the development cluster:

```
curl -X 'PUT' \ -u PROD_USER : PROD_PASSWORD \
' https://EXAMPLE_COMPANY_DEV.b.lucidworks.cloud /api/apps/ PROD
_APP_NAME /query-rewrite/instances' \ -H 'accept:
application/octet-stream' \ -H 'Content-Type: application/json'
\ -d @exported-rules.json
```

Launch a Remote V2 Plugin

How-to

> Compatible versions: 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

Follow this configuration to use a remote connector in Fusion 5.6 and later.

Note

A known issue is after idling for five minutes without any new connector start requests, the remote connector stops responding to server requests and must restart. This causes connector start requests to hang. A workaround is to schedule a connector to run every three minutes. It is recommended to use a connector instance that doesn't fetch any documents to keep the remote connector functional without frequent restarts.

1. Execute the following command.

```
java –jar connector–plugin–client–standalone.jar
[YAML_CONFIG_FILE]
```

2. Check that the `YAML_CONFIG_FILE` has the following format.

```
kafka–bridge: target: mynamespace–connectors–
backend.lucidworkstest.com:443 # mandatory plain–text: false #
optional proxy–server: # optional – needed when a forward proxy
server is used to provide outbound access to the standalone
connector host: host port: 111 user: user password: password
trust: # optional – needed when the client's system doesn't
trust the server's certificate cert–collection–filepath: path1
proxy: # mandatory fusion–proxy user: admin password:
password123 url: https://fusiontest.com/ # needed only when the
connector plugin requires blob store access plugin: # mandatory
path: ./fs.zip type: suffix: remote
```

   a. Use this table for descriptions of the YAML properties.

Key	Value description
`kafka-bridge.target`	The Kafka-gRPC bridge host:port pair.
`kafka-bridge.plain-text`	Enable when TLS ingress is not used.
`kafka-bridge.proxy-server.host`	The Forward Proxy Server Host.
`kafka-bridge.proxy-server.port`	The Forward Proxy Server Port.
`kafka-bridge.proxy-server.user`	The Forward Proxy Server User.
`kafka-bridge.proxy-server.password`	The Forward Proxy Server Password.
`kafka-bridge.trust.cert-collection-filepath`	The Private Key file path.
`proxy.url`	Fusion proxy url.
`proxy.user`	Fusion proxy user.
`proxy.password`	Fusion proxy password.
`plugin.path`	The path of the plugin zip file.
`plugin.type`	Optional suffix added to the plugin `id`. For example, if the plugin `id` is `lucidworks.test` and the suffix is `remote`, the plugin will be launched with `lucidworks.test.remote`.

Example `YAML_CONFIG_FILE`:

```
kafka-bridge: target: mynamespace-connectors.example.com:443 plain-text: false proxy: user: admin password: "password123" plugin: path: ./testplugin.zip
```

Lay Out a Page

How-to

> Compatible versions: 4.2
>
> Works with: *App Studio*

The `layout` tags determine the layout and structure of the page. Using these ensures that the application will work equally well on desktops, mobiles and other devices (like high-resolution screens).

App Studio uses a grid layout to control how large sections appear on the screen, and how they behave when the visible area or resolution changes.

A simple grid:

```
<layout:grid> <layout:block small="1–2" large="1–4"> Narrow section on large screens. </layout:block> <layout:block small="1–2" large="3–4"> Wide on large screens. </layout:block> </layout:grid>
```

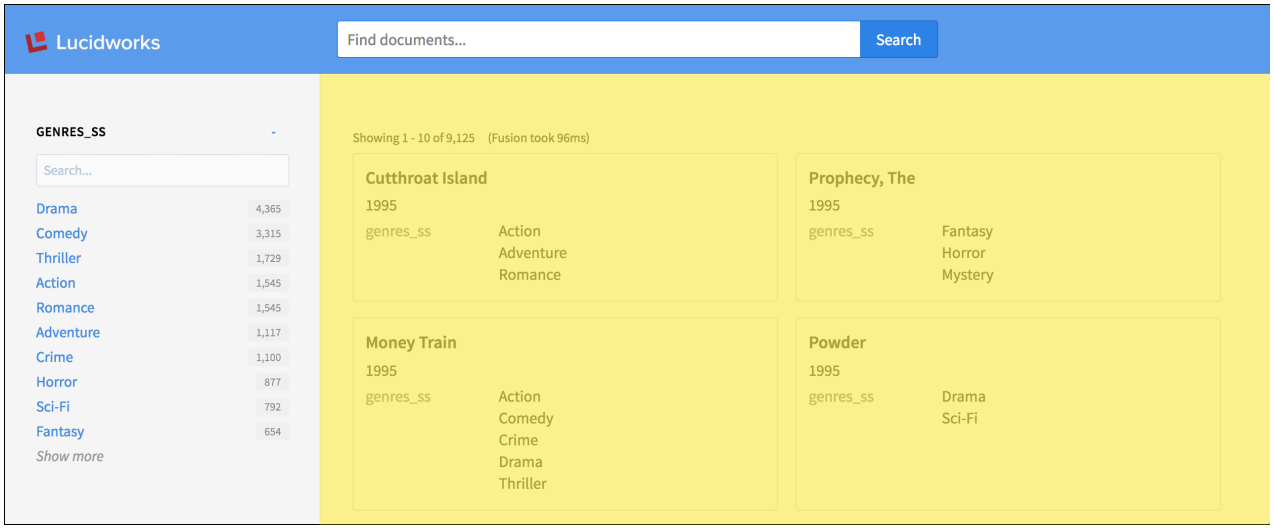
See the following reference topics for complete information about `layout` tags:

Lay Out Search Results in a Grid

How-to

> Compatible versions: 4.2
>
> Works with: *App Studio*

In the content area, `layout` tags are used again to construct the grid layout. For brevity, we will omit those from the code samples below.



The `search:result-list` tag lays out the grid of search results:

```
<search:result-list response="response" styling="cards-sm-1 cards-lg-2
cards-xl-2" platform="platform" query="query" instant-results="true">
... </search:result-list>
```

Limit the Crawl Scope for Web Sites

How-to

> Compatible versions: 4.2, 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion Connectors*, *Fusion Server*, *Fusion*

The Web V1 connector retrieves data from a Web site using HTTP and starting from a specified URL.

The connector works by going to the seed page (the `startURIs` specified in the configuration form), collecting the content for indexing, and extracting any links to other pages. It then follows those links to collect content on other pages, extracting links to those pages, and so on.

When creating a Web data source, pay attention to the **Max crawl depth** and **Restrict To Tree** parameters (`c.depth` and `c.restrictToTree` in the REST API). These properties limit the scope of your crawl to prevent an unbounded crawl that could continue for a long time, particularly if you are crawling a site with links to many pages outside the main site. An unbounded crawl can also cause memory errors in your system.

The connector keeps track of URIs it has seen, and many of the properties relate to managing the resulting database of entries. If the connector finds a standard redirect, it tracks that the redirected URI has an alias, and does not re-evaluate the URI on its next runs until the alias expiration has passed. If deduplication is enabled, documents that were found to be duplicates are also added to the alias list and are not re-evaluated until the alias expiration has passed.

Regular expressions can be used to restrict the crawl either by defining URI patterns that should be followed or URI patterns that should not be followed.

Additionally, specific patterns of the URI can be defined to define URIs that should not be followed.

Limit the Crawl Scope for Web Sites

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

The Web connector retrieves data from a Web site using HTTP and starting from a specified URL.

Managed Fusion releases 5.5 - 5.6.0 use the *Web V1 connector*. Managed Fusion releases 5.6.1 and later use the *Web V2 connector*.

The connector works by going to the seed page (the `startURIs` specified in the configuration form), collecting the content for indexing, and extracting any links to other pages. It then follows those links to collect content on other pages, extracting links to those pages, and so on.

When creating a Web data source, pay attention to the **Max crawl depth** and **Restrict To Tree** parameters (`c.depth` and `c.restrictToTree` in the REST API). These properties limit the scope of your crawl to prevent an unbounded crawl that could continue for a long time, particularly if you are crawling a site with links to many pages outside the main site. An unbounded crawl can also cause memory errors in your system.

The connector keeps track of URIs it has seen, and many of the properties relate to managing the resulting database of entries. If the connector finds a standard redirect, it tracks that the redirected URI has an alias, and does not re-evaluate the URI on its next runs until the alias expiration has passed. If deduplication is enabled, documents that were found to be duplicates are also added to the alias list and are not re-evaluated until the alias expiration has passed.

Regular expressions can be used to restrict the crawl either by defining URI patterns that should be followed or URI patterns that should not be followed.

Additionally, specific patterns of the URI can be defined to define URIs that should not be followed.

List All Docker Images and Versions

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion*

This article teaches you how to list all Docker images and their versions required by Fusion. This is helpful for *creating a private repository for Docker images*. You can compare your results with *Docker images by Fusion version*.

Prerequisites

Install the command line tools below before proceeding.

- **yq** -yq is a Python YAML/XML processor. It runs with the command `python-yq`.
  To install yq on Mac, use Homebrew: `brew install python-yq`. For other operating systems, use pip3: `pip3 install yq`.
- **jq** -jq is a command-line JSON processor.
  To install jq on Mac, use Homebrew: `brew install jq`. For other operating systems, refer to the jq download instructions.
- **Helm** -Helm helps install, manage, and upgrade complex Kubernetes applications, such as Fusion.
  To install Helm on Mac, use Homebrew: `brew install kubernetes-helm`. For other operating systems, see the Helm installation documentation.

Instructions

Option 1: Using the Helm repository

By using the Helm repository, you can specify the desired version or load the latest production version without finding the direct download URL.

1. Add the Helm repository:

```
helm repo add lucidworks https://charts.lucidworks.com
```

2. Pull your desired Helm template with `helm template`, and filter the Helm chart with `yq`:

```
helm template lucidworks/fusion --version 5.3.1 | yq -r
'..|.image? | select(.)' | sort | uniq
```

> **Tip**
>
> If you remove `--version` from the command above, the command loads the latest production release version.

Option 2: Using compressed files

If you prefer not to clone the repository, you can download individual versions directly.

1. Pull your desired Helm template with `helm template`, and filter the Helm chart with `yq`:

```
helm template https://charts.lucidworks.com/fusion-5.3.1.tgz |
yq '..|.image? | select(.)' | sort | uniq
```

Result

Regardless of the option you chose, the output should resemble the following. Be aware that this is only an example and the output may vary.

```
"alpine:3.8" "apachepulsar/pulsar-all:2.6.1" "argoproj/argocli:v2.7.6"
"argoproj/workflow-controller:v2.7.6" "bitnami/kubectl"
"bitnami/kubectl:1.15-debian-9" "busybox" "busybox:1.31.1"
"docker.io/seldonio/seldon-core-operator:1.1.0" "lucidworks/admin-
ui:5.3.1" "lucidworks/api-gateway:5.3.1" "lucidworks/auth-ui:5.3.1"
"lucidworks/check-fusion-dependency:v1.2.0" "lucidworks/classic-rest-
service:5.3.1" "lucidworks/connector-plugin-service:5.3.1"
"lucidworks/devops-ui:5.3.1" "lucidworks/fusion-api:5.3.1"
"lucidworks/fusion-indexing:5.3.1" "lucidworks/fusion-log-
forwarder:5.3.1" "lucidworks/insights:5.3.1" "lucidworks/job-
launcher:5.3.1" "lucidworks/job-rest-server:5.3.1"
"lucidworks/keytool-utils:v2.1.0" "lucidworks/ml-model-service:5.3.1"
"lucidworks/pm-ui:5.3.1" "lucidworks/query-pipeline:5.3.1"
"lucidworks/rest-service:5.3.1" "lucidworks/rpc-service:5.3.1"
"lucidworks/rules-ui:5.3.1" "lucidworks/sql-service:5.3.1"
"lucidworks/templating:5.3.1" "lucidworks/webapps:5.3.1"
"milvusdb/milvus:0.10.2-cpu-d081520-8a2393" "mysql:5.7.30"
"quay.io/datawire/ambassador:0.86.1" "solr:8.6.3" "zookeeper:3.5.7"
```

Localize an App

How-to

> Compatible versions: 4.2
>
> Works with: *App Studio*

You can localize the UI labels in an app in multiple languages of your choice.

Appkit obtains the correct labels for a language by making calls to an Appkit translation service. The service provides a label in the chosen language, with a fallback to English if no translation exists for the label in the chosen language.

Loading translations

You can use the new `<translations:localize>` JSP tag to load a set of UI label translations into an app. The tag syntax is:

```
<translations:localize dictionary="*dictionary*" locale="*locale*" />
```

- `dictionary` . Name of the resource bundle from which to load translations. This is also the first part of the name of each property file that contains translations.
  Default: `translations`
- `locale` . Locale
  Default: `en` (English)

The `dictionary` and `locale` are used to find the file as follows:

1. In `src/main/resources` , the Appkit translation service looks for a resource bundle named `dictionary` , for example, `translations` .
2. Within the resource bundle, the translation service looks for property files named `{resource}` , for example, for the files `translations_en.properties` and `translations_fr.properties` .

Enabling translations

1. Find label strings that need to be translated in the UI and identify the *key names for the labels*.
2. Create translation property files and add them under the `src/main/resources` directory. For example, for a dictionary called `translations` add `translations_en.properties` and `translations_fr.properties` .
   Edit the property files and include the label keys and translations. For example:
   In `translations_en.properties` :

   ```
   # translations_en.properties ... components.pagination.next =
   Next components.pagination.previous = Previous ...
   ```

   In `translations_fr.properties` :

   ```
   # translations_fr.properties ... components.pagination.next =
   Suivant components.pagination.previous = Précédent ...
   ```

> **Note**
>
> Some translations include variables. For example, `components.response-statistics.showing` is `Showing {first} – {last} of {total}` in English. Translations should retain the variables, though their order might differ. Labels can use `<em>` tags for emphasis, for example, `More results like <em>{result}</em>`.

3. Add the following dependency to your app's `pom.xml`:

```
<!-- Twigkit translations --> <dependency>
<groupId>twigkit</groupId>
<artifactId>twigkit.translations</artifactId>
<version>${project.parent.version}</version> </dependency>
```

4. Add this `taglib` directive at the top of your app's `index.jsp`:

```
<%@ taglib prefix="translations" uri="/twigkit/translations" %>
```

5. Add the Appkit `<translations:localize>` JSP tag inside the `<head>` element of your app's `index.jsp`:

```
<translations:localize locale="${param.locale}" />
```

6. Add this rule to the `url-rules.xml` file:

```
<!-- Language specific pages --> <rule>
<from>^/lang/(.*)/</from> <to last="true">/index.jsp?
locale=$1</to> </rule>
```

> **Note**
>
> You can modify other URL rules to suit your application logic. The
> `<translations:localize> JSP tag works the same.

7. Restart your app.
8. Verify that the translations are present by visiting pages in the app that should be in different languages.

Logging

How-to

> Compatible versions: 4.2, 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *App Studio*

Appkit provides support for tracking the activity of specific events and logging them either to the console or a file depending on how your application is set up.

Log activity events

To log activity events, perform the steps in this section.

1: Update the POM

To log activity events, first add these dependencies to your `pom.xml` file:

```
<dependency> <groupId>ch.qos.logback</groupId> <artifactId>logback-
classic</artifactId> <version>1.1.1</version> </dependency>
<dependency> <groupId>twigkit</groupId>
<artifactId>twigkit.activity.logger</artifactId>
<version>${project.parent.version}</version> </dependency>
```

2: Create an activity logging configuration file

In `conf/activity/tracking`, place a `logger.conf` file. In the file, set the property `logging-enabled` equal to `true`:

```
logging-enabled: true
```

If `logging-enabled` is set to `false`, then all logging will be disabled.

3: Create a logback.xml file

Create a `logback.xml` file in `/resources` and configure this to either output activity data to the console or a file (or both).

For example, to output logging activity to the console:

```
<configuration debug="true" scan="true"> <appender name="STDOUT"
class="ch.qos.logback.core.ConsoleAppender">
<withJansi>false</withJansi> <encoder> <pattern>%d{HH:mm:ss.SSS}
%highlight(%-5level) %cyan(%logger{36}) - %msg %n</pattern> </encoder>
</appender> <logger name="twigkit" level="INFO"/> <logger
name="twigkit.activity.subscribers" level="TRACE"/> <root
level="ERROR"> <appender-ref ref="STDOUT"/> </root> </configuration>
```

To output logging activity to a file, while all other logging is redirected to the console:

```
<configuration debug="true" scan="true"> <appender name="STDOUT"
class="ch.qos.logback.core.ConsoleAppender">
<withJansi>false</withJansi> <encoder> <pattern>%d{HH:mm:ss.SSS}
%highlight(%-5level) %cyan(%logger{36}) - %msg %n</pattern> </encoder>
</appender> <appender name="activity-log"
class="ch.qos.logback.core.rolling.RollingFileAppender">
<file>./logs/activity.log</file> <append>true</append> <rollingPolicy
class="ch.qos.logback.core.rolling.TimeBasedRollingPolicy"> <!-- Daily
rollover. Make sure the path matches the one in the file element or
else the rollover logs are placed in the working directory. -->
<fileNamePattern>./logs/activity_%d{yyyy-MM-
dd}.%i.log</fileNamePattern> <timeBasedFileNamingAndTriggeringPolicy
class="ch.qos.logback.core.rolling.SizeAndTimeBasedFNATP">
<maxFileSize>5MB</maxFileSize>
</timeBasedFileNamingAndTriggeringPolicy> <!-- keep 30 days' worth of
history --> <maxHistory>30</maxHistory> </rollingPolicy> <encoder>
<pattern>%d{HH:mm:ss.SSS} - %msg %n</pattern> </encoder> </appender>
<logger name="twigkit" level="INFO"/> <logger
name="twigkit.activity.subscribers" level="TRACE" additivity="false">
<appender-ref ref="activity-log"/> </logger> <root level="ERROR">
<appender-ref ref="STDOUT"/> </root> </configuration>
```

The location of the file is set in the `file` tag and is relative to the root of the application.

Example

A typical example of what you might see in the log is:

```
16:33:51.711 –
type:response,user:user,query:US,hits:2437,page:1,time:104
16:33:52.907 –
type:response,user:user,query:US,hits:2437,page:1,time:106
16:33:54.191 –
type:click,username:user,URL:http://twigkit.com/kapps_kyc/customers/519/,host:twigkit.com
```

Where, for example, the **`type`** refers to whether the event is a click or response, the **`user`** is the name of the user, **`query`** shows the query terms, **`hits`** is the number of hits returned in the response, **`page`** is the page number, and **`URL`** is the URL associated with the click.

Related articles

- *Tracking Fusion Signals*

Not Found

This route does not exist. *Return home*, or send us feedback (see widget) if you think this is an error.

Not Found

This route does not exist. *Return home*, or send us feedback (see widget) if you think this is an error.

Not Found

This route does not exist. *Return home*, or send us feedback (see widget) if you think this is an error.

Not Found

This route does not exist. *Return home*, or send us feedback (see widget) if you think this is an error.

Not Found

This route does not exist. *Return home*, or send us feedback (see widget) if you think this is an error.

Not Found

This route does not exist. *Return home*, or send us feedback (see widget) if you think this is an error.

Not Found

This route does not exist. *Return home*, or send us feedback (see widget) if you think this is an error.

Not Found

This route does not exist. *Return home*, or send us feedback (see widget) if you think this is an error.

Not Found

This route does not exist. *Return home*, or send us feedback (see widget) if you think this is an error.

Not Found

This route does not exist. *Return home*, or send us feedback (see widget) if you think this is an error.

Not Found

This route does not exist. *Return home*, or send us feedback (see widget) if you think this is an error.

Not Found

This route does not exist. *Return home*, or send us feedback (see widget) if you think this is an error.

Not Found

This route does not exist. *Return home*, or send us feedback (see widget) if you think this is an error.

Not Found

This route does not exist. *Return home*, or send us feedback (see widget) if you think this is an error.

Not Found

This route does not exist. *Return home*, or send us feedback (see widget) if you think this is an error.

Not Found

This route does not exist. *Return home*, or send us feedback (see widget) if you think this is an error.

Not Found

This route does not exist. *Return home*, or send us feedback (see widget) if you think this is an error.

Not Found

This route does not exist. *Return home*, or send us feedback (see widget) if you think this is an error.

Not Found

This route does not exist. *Return home*, or send us feedback (see widget) if you think this is an error.

Not Found

This route does not exist. *Return home*, or send us feedback (see widget) if you think this is an error.

Not Found

This route does not exist. *Return home*, or send us feedback (see widget) if you think this is an error.

Not Found

This route does not exist. *Return home*, or send us feedback (see widget) if you think this is an error.

Not Found

This route does not exist. *Return home*, or send us feedback (see widget) if you think this is an error.

Not Found

This route does not exist. *Return home*, or send us feedback (see widget) if you think this is an error.

Not Found

This route does not exist. *Return home*, or send us feedback (see widget) if you think this is an error.

Not Found

This route does not exist. *Return home*, or send us feedback (see widget) if you think this is an error.

Not Found

This route does not exist. *Return home*, or send us feedback (see widget) if you think this is an error.

Not Found

This route does not exist. *Return home*, or send us feedback (see widget) if you think this is an error.

Not Found

This route does not exist. *Return home*, or send us feedback (see widget) if you think this is an error.

Not Found

This route does not exist. *Return home*, or send us feedback (see widget) if you think this is an error.

Not Found

This route does not exist. *Return home*, or send us feedback (see widget) if you think this is an error.

Not Found

This route does not exist. *Return home*, or send us feedback (see widget) if you think this is an error.

Not Found

This route does not exist. *Return home*, or send us feedback (see widget) if you think this is an error.

Not Found

This route does not exist. *Return home*, or send us feedback (see widget) if you think this is an error.

Not Found

This route does not exist. *Return home*, or send us feedback (see widget) if you think this is an error.

Not Found

This route does not exist. *Return home*, or send us feedback (see widget) if you think this is an error.

Not Found

This route does not exist. *Return home*, or send us feedback (see widget) if you think this is an error.

Not Found

This route does not exist. *Return home*, or send us feedback (see widget) if you think this is an error.

Not Found

This route does not exist. *Return home*, or send us feedback (see widget) if you think this is an error.

Not Found

This route does not exist. *Return home*, or send us feedback (see widget) if you think this is an error.

Not Found

This route does not exist. *Return home*, or send us feedback (see widget) if you think this is an error.

Not Found

This route does not exist. *Return home*, or send us feedback (see widget) if you think this is an error.

Not Found

This route does not exist. *Return home*, or send us feedback (see widget) if you think this is an error.

Not Found

This route does not exist. *Return home*, or send us feedback (see widget) if you think this is an error.

Not Found

This route does not exist. *Return home*, or send us feedback (see widget) if you think this is an error.

Not Found

This route does not exist. *Return home*, or send us feedback (see widget) if you think this is an error.

Not Found

This route does not exist. *Return home*, or send us feedback (see widget) if you think this is an error.

Not Found

This route does not exist. *Return home*, or send us feedback (see widget) if you think this is an error.

Not Found

This route does not exist. *Return home*, or send us feedback (see widget) if you think this is an error.

Not Found

This route does not exist. *Return home*, or send us feedback (see widget) if you think this is an error.

Not Found

This route does not exist. *Return home*, or send us feedback (see widget) if you think this is an error.

Not Found

This route does not exist. *Return home*, or send us feedback (see widget) if you think this is an error.

Not Found

This route does not exist. *Return home*, or send us feedback (see widget) if you think this is an error.

Not Found

This route does not exist. *Return home*, or send us feedback (see widget) if you think this is an error.

Not Found

This route does not exist. *Return home*, or send us feedback (see widget) if you think this is an error.

Not Found

This route does not exist. *Return home*, or send us feedback (see widget) if you think this is an error.

Not Found

This route does not exist. *Return home*, or send us feedback (see widget) if you think this is an error.

Not Found

This route does not exist. *Return home*, or send us feedback (see widget) if you think this is an error.

Not Found

This route does not exist. *Return home*, or send us feedback (see widget) if you think this is an error.

Not Found

This route does not exist. *Return home*, or send us feedback (see widget) if you think this is an error.

Not Found

This route does not exist. *Return home*, or send us feedback (see widget) if you think this is an error.

Not Found

This route does not exist. *Return home*, or send us feedback (see widget) if you think this is an error.

Not Found

This route does not exist. *Return home*, or send us feedback (see widget) if you think this is an error.

Not Found

This route does not exist. *Return home*, or send us feedback (see widget) if you think this is an error.

Not Found

This route does not exist. *Return home*, or send us feedback (see widget) if you think this is an error.

Not Found

This route does not exist. *Return home*, or send us feedback (see widget) if you think this is an error.

Not Found

This route does not exist. *Return home*, or send us feedback (see widget) if you think this is an error.

Not Found

This route does not exist. *Return home*, or send us feedback (see widget) if you think this is an error.

Not Found

This route does not exist. *Return home*, or send us feedback (see widget) if you think this is an error.

Not Found

This route does not exist. *Return home*, or send us feedback (see widget) if you think this is an error.

Not Found

This route does not exist. *Return home*, or send us feedback (see widget) if you think this is an error.

Not Found

This route does not exist. *Return home*, or send us feedback (see widget) if you think this is an error.

Not Found

This route does not exist. *Return home*, or send us feedback (see widget) if you think this is an error.

Not Found

This route does not exist. *Return home*, or send us feedback (see widget) if you think this is an error.

Not Found

This route does not exist. *Return home*, or send us feedback (see widget) if you think this is an error.

Not Found

This route does not exist. *Return home*, or send us feedback (see widget) if you think this is an error.

Not Found

This route does not exist. *Return home*, or send us feedback (see widget) if you think this is an error.

Not Found

This route does not exist. *Return home*, or send us feedback (see widget) if you think this is an error.

Not Found

This route does not exist. *Return home*, or send us feedback (see widget) if you think this is an error.

Not Found

This route does not exist. *Return home*, or send us feedback (see widget) if you think this is an error.

Not Found

This route does not exist. *Return home*, or send us feedback (see widget) if you think this is an error.

Not Found

This route does not exist. *Return home*, or send us feedback (see widget) if you think this is an error.

Not Found

This route does not exist. *Return home*, or send us feedback (see widget) if you think this is an error.

Not Found

This route does not exist. *Return home*, or send us feedback (see widget) if you think this is an error.

Not Found

This route does not exist. *Return home*, or send us feedback (see widget) if you think this is an error.

LWAI async summarization and keyword extraction pipeline

Lucidworks AI

How-to

> Works with: Lucidworks AI

This pipeline uses the following stages:

- Field Mapping
- Solr Dynamic Field Name Mapping
- LWAI: Summarization
- LWAI: Keyword Extraction
- Solr Partial Update Indexer

Add the pipeline

1. Navigate to **Indexing > Index Pipelines**.
2. Click **Add+**.
3. Enter the **Pipeline ID**, for example `LWAI-async-summarization-and-keyword-extraction`.
4. Select the **Solr Indexer** stage and click on **Remove stage**.

Field Mapping

Configure the Field Mapping stage as follows.

1. Make sure that **Unmapped Fields** is checked.
2. Make sure that **Keep** is checked.
3. Save the pipeline.

Solr Dynamic Field Name Mapping

Configure the Solr Dynamic Field Name Mapping stage as follows.

1. Enter a **Max Length for Advanced Indexing of Text Fields**, such as `100`.
2. Save the pipeline.

LWAI: Summarization

Summarization is set up via the LWAI Prediction stage.

1. Click **Add a new pipeline stage > LWAI Prediction**.
2. Enter a **Label**, such as `LWAI: Summarization`.
3. Select the **Account Name** as defined in Lucidworks AI Gateway.
4. Select the **Use Case** as `summarization`.
5. Select the **Model** to use.
6. Enter the **Input context variable**, for example, `<doc.body_t>`.
7. Enter the **Destination Field Name & Context Output**, for example, `body_summarization`.
8. Enter values for the **Use Case Configuration**:
    a. Enter the **Parameter Name**, such as `maxWords`.
    b. Enter a **Parameter Value**, for example, `250`.
9. Make sure **Call Asynchronously?** is checked.
10. Enter the **Maximum Asynchronous Call Tries**, such as `1`.
11. Save the pipeline.

LWAI: Keyword Extraction

Keyword extraction is set up using the LWAI Prediction stage.

1. Click **Add a new pipeline stage > LWAI Prediction**.
2. Enter a **Label**, such as `LWAI: Keyword Extraction`.
3. Select the **Account Name** as defined in Lucidworks AI Gateway.
4. Select the **Use Case** as `keyword-extraction`.
5. Select the **Model** to use.
6. Enter the **Input context variable**, for example, `<doc.body_t>`.
7. Enter the **Destination Field Name & Context Output**, for example, `body_extracted_keywords`.
8. Enter values for the **Use Case Configuration**:
    a. Enter the **Parameter Name**, such as maxKeywo`rds.

    b. Enter the **Parameter Value**, for example, `50`.

9. Make sure **Call Asynchronously?** is checked.
10. Enter the **Maximum Asynchronous Call Tries**, such as `1`.
11. Save the pipeline.

Solr Partial Update Indexer

Configure the Solr Partial Update Indexer stage as follows.

1. Click **Add a new pipeline stage > Solr Partial Update Indexer**.
2. Make sure **Map to Solr Schema** is checked.
3. Make sure **Process All Pipeline Doc Fields** is checked.
4. Enter the **Solr Document ID Field Value** as `<doc.id>`.
5. Enter values under **Updates**:
    a. Enter the **Update Type** as `increment`.
    b. Enter the **Field Name** as `increment_i`.
    c. Enter a **Value**, such as `1`.
6. Make sure **Allow reserved fields** is checked.
7. Save the pipeline.

Order the stages

1. Make sure the stages are in the following order:
    a. Field Mapping
    b. Solr Dynamic Field Name Mapping
    c. LWAI: Summarization
    d. LWAI: Keyword Extraction
    e. Solr Partial Update Indexer
2. Save the pipeline.

LWAI NHS plus RAG pipeline

Lucidworks AI

How-to

> Works with: Lucidworks AI

This pipeline uses the following stages:

- Additional Query Parameters
- [LWAI] Query Rewrite
- Additional Query Parameters
- LWAI Vectorize Query
- Hybrid Query
- Solr Query
- LWAI Prediction

Add the pipeline

1. Navigate to **Querying > Query Pipelines**.
2. Click **Add+**.
3. Enter the **Pipeline ID**, for example `LWAI-NHS-plus-RAG`.
4. Remove the default stages except for Solr Query:
    a. Remove the Text Tagger stage.
    b. Remove the Boost with Signals stage.
    c. Remove the Query Fields stage.
    d. Remove the Facets stage.
    e. Remove the Apply Rules stage.
    f. Remove the Modify Response with Rules stage.

Additional Query Parameters

Configure the Additional Query Parameters stage as follows.

1. Click **Add a new pipeline stage > Additional Query Parameters**.
2. Enter names, values, and policies for the **Parameters and Values**:
    a. `orig_q` - `<request.q>` - `replace`.
    b. `rewritten_q` - `<request.q>` - `replace`.
3. Save the pipeline.

[LWAI] Query Rewrite

LWAI Query Rewrite is set up using the LWAI Prediction stage.

1. Click **Add a new pipeline stage > LWAI Prediction**.
2. Enter a **Label**, such as `[LWAI] Query Rewrite`.
3. In the **Condition** field, enter `request.getFirstFieldValue('q') != ':' && request.hasParam('memory_uuid')`.
4. Select the Lucidworks AI integration **Account Name** as defined by your Fusion Administrator.
5. Select the **Use Case**, such as `standalone-query-rewriter`.
6. Select the **Model** to use.
7. Enter the **Input context variable** as `<request.q>`.
8. Enter the **Destination Variable Name & Context Output** as `standalone`.
9. Enter the following under **Use Case Configuration**:
    a. **Parameter Name**: `memoryUuid`.
    b. **Parameter Value**: `<request.memory_uuid>`.
10. Save the pipeline.

Additional Query Parameters

Configure another Additional Query Parameters stage as follows.

1. Click **Add a new pipeline stage > *Additional Query Parameters**.
2. Enter the following under **Parameters and Values**:
    a. **Parameter Name**: `rewitten_q`.
    b. **Parameter Value**: `<ctx.lw_ai_standalone-query-rewriter_t>`.
    c. **Update Policy**: `replace`.

3. Save the pipeline.

LWAI Vectorize Query

Configure the LWAI Vectorize Query stage as follows.

1. Click **Add a new pipeline stage > LWAI Vectorize Query**.
2. In the **Label** field, enter a unique identifier for this stage.
3. In the **Condition** field, enter a script that results in true or false, which determines if the stage should process.
4. Select **Asynchronous Execution Config** if you want to run this stage asynchronously. If this field is enabled, complete the following fields:
    a. Select **Enable Async Execution**. Fusion automatically assigns an **Async ID** value to this stage. Change this to a more memorable string that describes the asynchronous stages you are merging, such as `signals` or `access_control`.
    b. Copy the **Async ID** value.

> **Note**
>
> For detailed information, see *Enable asynchronous query pipeline processing* and *Asynchronous query pipeline processing*.

5. Select the **Account Name**.
6. Select the Model to use.
7. Set the **Query Input** to `<request.rewritten_q>`.
8. Enter the **Output Context Variable** as `vector`.
9. Save the pipeline.

Hybrid Query

Configure the Hybrid Query stage as follows.

1. Click **Add a new pipeline stage > Hybrid Query**.
2. Set the **Lexical Query Input** as `<request.rewritten_q>`.
3. Enter a value for the **Lexical Query Weight**, for example, `0.3`.
4. Set the **Number of Lexical Results**, such as `1000`.
5. In the **Vector Query Field**, enter the name of the Solr field for KNN vector search.
6. Set the **Vector Input** to `<ctx.vector>`.
7. Enter a value for the **Vector Query Weight**, for example, `0.7`.
8. Check the box for **Use KNN Query**.
9. Under **Use KNN Query**, enter **Number of Vector Results**, such as `1000`.
10. Save the pipeline.

Solr Query

Configure the Solr Query stage as follows.

1. Select the **HTTP Method** as POST.
2. Make sure the **Generate Response Signal** is checked.
3. Set the **Preferred Replica Type** to pull.
4. Save the pipeline.

LWAI Prediction

Configure the LWAI Prediction stage as follows.

1. Click **Add a new pipeline stage > LWAI Prediction**.
2. In the **Label** field, enter a unique identifier for this stage.
3. In the **Condition** field, enter a script that results in true or false, which determines if the stage should process.

4. Select **Asynchronous Execution Config** if you want to run this stage asynchronously. If this field is enabled, complete the following fields:
    a. Select **Enable Async Execution**. Fusion automatically assigns an **Async ID** value to this stage. Change this to a more memorable string that describes the asynchronous stages you are merging, such as `signals` or `access_control`.
    b. Copy the **Async ID** value.

> **Note**
>
> For detailed information, see *Enable asynchronous query pipeline processing* and *Asynchronous query pipeline processing*.

5. Set the **Account Name** to the Lucidworks AI integration name as defined by your Fusion Administrator.
6. Select the **Use Case** as `rag`.
7. Select the **Model** to use.
8. Set the **Input context variable** to `<request.rewritten_q>`.
9. Make sure **Include Response Documents?** is checked.
10. Enter values into the **Use Case Configuration**:
    a. **Parameter Name**: `extractRelevantContent`.
    b. **Parameter Value**: `false`.
11. Save the pipeline.

Order the stages

1. Make sure the stages are in the following order:
    a. Additional Query Parameters
    b. [LWAI] Query Rewrite
    c. Additional Query Parameters
    d. LWAI Vectorize Query
    e. Hybrid Query
    f. Solr Query
    g. LWAI Prediction
2. Save the pipeline.

Manage Collections in the Fusion UI

How-to

> Compatible versions: 4.2, 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion Server*, *Fusion*

Collections can be created or removed using the Fusion UI or the REST API.

For information about using the REST API to manage collections, see Collections API in the REST API Reference:

- **Fusion 5.x.** *Collections API*
- **Fusion 4.x.** *Collections API*

Creating a Collection

When you create an app, by default Fusion Server creates a collection and associated objects.

To create a new collection in the Fusion UI:

1. From within an app, click **Collections > Collections Manager**.
2. At the upper right of the panel, click **New**.
3. Enter a **Collection name**. This name cannot be changed later.
4. To create the collection in the default Solr cluster and with other default settings, click **Save Collection**.

Creating a Collection with Advanced Options

To access advanced options for creating a collection in the Fusion UI:

1. From within an app, click **Collections > Collections Manager**.
2. At the upper right of the panel, click **New**.
3. Enter a **Collection name**. This name cannot be changed later.
4. Click **Advanced**.
5. Configure advanced options. The options are described below.
6. Click **Save Collection**.

Solr Cluster

By default, a new collection is associated with the Solr instance that is associated with the `default` Solr cluster.

If Fusion has multiple Solr clusters, choose from the list which cluster you want to associate your collection with. The cluster must exist first.

Solr Cluster Layout

The next section lets you define a **Replication Factor** and **Number of Shards**. Define these options only if you are creating a new collection in the Solr cluster. If you are linking Fusion to an existing Solr collection, you can skip these settings.

Solr Collection Import

Import a Solr collection to associate the new Fusion collection with an existing Solr collection. Enter a **Solr Collection Name** to associate the collection with an existing Solr collection. Then, enter a **Solr Config Set** to tell ZooKeeper to use the configurations from an existing collection in Solr when creating this collection.

Time Series Partitioning

> **Note**
>
> Available in 4.x only.

You can map a Fusion collection to multiple Solr collections, known here as partitions, where each partition contains data from a specific time range.

To configure time-based partitioning, under **Time Series Partitioning** click **Enable**.

See *Time-Based Partitioning* for more information.

Configuring Collections

The Collections menu lets you configure your existing collection, including datasources, fields, jobs, stopwords, and synonyms.

In the Fusion UI, from any app, the Collections icon displays on the left side of the screen.

Some tasks related to managing a collection are available in other menus:

- Configure a profile in **Indexing > Indexing Profiles** or **Querying > Query Profiles**.
- View reports about your collection's activity in **Analytics > Dashboards**.

Collections Manager

The Collections Manager page displays details about the collection, such as how many datasources are configured, how many documents are in the index, and how much disk space the index consumes.

This page also lets you *create a new collection*, disable search logs or signals, enable recommendations, issue a commit command to Solr, or clear a collection.

DISABLE SEARCH LOGS

When you first create a collection, the search logs are created by default. The search logs populate the panels in **Analytics > Dashboards**.

To disable search logs:

1. Hover over your collection name until the gear icon appears at the end of the line.
2. Click the gear icon.
3. Click **Disable Search Logs**.
4. On the confirmation screen, click **Disable Search Logs**.

Note that if you disable search logs, you cannot see any data for this collection in **Analytics > Dashboards**.

See Dashboards:

- **Fusion 5.x.** *Dashboards*
- **Fusion 4.x.** *Dashboards*

DISABLE SIGNALS

When you first create a collection, the signals and aggregated signals collections are created by default.

To disable signals:

1. Hover over your collection name until the gear icon appears at the end of the line.
2. Click the gear icon.
3. Click **Disable Signals**.
4. On the confirmation screen, click **Disable Signals**.

HARD COMMIT A COLLECTION

To issue a hard commit command to Solr:

1. Hover over your collection name until the gear icon appears at the end of the line.
2. Click the gear icon.
3. Click **Hard Commit Collection**.
4. On the confirmation screen, click **Hard Commit Collection**.

Read internal details about how Solr processes commits on our blog .

Datasources

To access the Datasources page, click **Indexing > Datasources**. By default, there are no datasources configured right after installation.

To add a new datasource, click **New** at the upper right of the panel.

See the *Connectors and Datasources Reference* for details on how to configure a datasource. Options vary depending on the repository you would like to index.

After you configure a datasource, it appears in a list on this screen. Click the name of a datasource to edit its properties. Click **Start** to start the datasource. Click **Stop** to stop the datasource before it completes. To the right, view information on the last completed job, including the date and time started and stopped, and the number of documents found as new, skipped, or failed.

Note

When you stop a datasource, Fusion attempts to safely close connector threads, finishing processing documents through the pipeline and indexing documents to Solr. Some connectors take longer to complete these processes than others, so might stay in a "stopping" state for several minutes.

To stop a datasource immediately, choose **Abort** instead of **Stop**.

There is also a REST API for datasources:

- **Fusion 5.x.** *Connector Datasources API*
- **Fusion 4.x.** *Connector Datasources API*

Stopwords

The Stopwords page lets you edit a stopwords list for your collection.

To add or delete stop words:

1. Click the name of the text file you wish to edit.
2. Add a new word on a new line.
3. When you are done with your changes, click **Save**.

To import a stop words list:

1. Click **System > Import Fusion Objects**.
2. Choose the file to upload.
3. Click **Import >>**.

Read more about stopwords:

- **Fusion 5.x.** *stopwords*
- **Fusion 4.x.** *stopwords*

Synonyms

Fusion has the same synonym functionality that Solr supports. This includes a list of words that are synonyms (where the synonym list expands on the terms entered by the user), as well as a full mapping of words, where a word is substituted for what the user has entered (that is, the term the user has entered is replaced by a term in the synonym list).

See more about synonyms:

- **Fusion 5.x.** *synonyms*
- **Fusion 4.x.** *synonyms*

You can edit the synonyms list for your collection.

To access the Synonyms page in the Fusion UI, in any app, click **Collections > Synonyms**.

Filter the list of synonym definitions by typing in the **Filter...** box.

To import a synonyms list:

1. From the Synonyms page, click **Import and Save**. A dialog box opens.
2. Choose the file to import.

To edit a synonyms list:

- Enter new synonym definitions one per line.
    - To enter a string of terms that expand on the terms the user entered, enter the terms separated by commas, like `Television, TV`.
    - To enter a term that should be mapped to another term, enter the terms separated by an equal sign then a right angle bracket, `⇒`, like `i-pod⇒ipod`.
- Remove a line by clicking the **x** at the end of the line.
- Once you are finished with edits, click **Save**.

To export the synonyms list, click **Export**. This downloads the list to your computer using your browser download capability.

Profiles

Profiles allow you to create an alias for an index or query pipeline. This allows you to send documents or queries to a consistent endpoint and change the underlying pipeline or collection as needed.

Read about profiles in Index Profiles and Query Profiles:

- **Fusion 5.x.**
    - *Index Profiles*
    - *Query Profiles*
- **Fusion 4.x.**
    - *Index Profiles*
    - *Query Profiles*

To access the Solr Config page, from any app, click **System > Solr Config**.

Additional resources

LucidAcademy

Lucidworks offers free training to help you get started.

The **Quick Learning** for **Collections Menu Tour** focuses on the Collections Menu features and functionality along with a brief description of each screen available in the menu:

Visit the LucidAcademy to see the full training catalog.

Manage Collections in the Managed Fusion UI

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

Collections can be created or removed using the Fusion UI or the REST API.

For information about using the REST API to manage collections, see *Collections API* in the REST API reference.

Creating a Collection

When you create an app, by default Managed Fusion creates a collection and associated objects.

To create a new collection in the Managed Fusion UI:

1. From within an app, click **Collections > Collections Manager**.
2. At the upper right of the panel, click **New**.
3. Enter a **Collection name**. This name cannot be changed later.
4. To create the collection in the default Solr cluster and with other default settings, click **Save Collection**.

Creating a Collection with Advanced Options

To access advanced options for creating a collection in the Managed Fusion UI:

1. From within an app, click **Collections > Collections Manager**.
2. At the upper right of the panel, click **New**.
3. Enter a **Collection name**. This name cannot be changed later.
4. Click **Advanced**.
5. Configure advanced options. The options are described below.
6. Click **Save Collection**.

Solr Cluster

By default, a new collection is associated with the Solr instance that is associated with the `default` Solr cluster.

If Managed Fusion has multiple Solr clusters, choose from the list which cluster you want to associate your collection with. The cluster must exist first.

Solr Cluster Layout

The next section lets you define a **Replication Factor** and **Number of Shards**. Define these options only if you are creating a new collection in the Solr cluster. If you are linking Fusion to an existing Solr collection, you can skip these settings.

Solr Collection Import

Import a Solr collection to associate the new Managed Fusion collection with an existing Solr collection. Enter a **Solr Collection Name** to associate the collection with an existing Solr collection. Then, enter a **Solr Config Set** to tell ZooKeeper to use the configurations from an existing collection in Solr when creating this collection.

Configuring Collections

The Collections menu lets you configure your existing collection, including datasources, fields, jobs, stopwords, and synonyms.

In the Managed Fusion UI, from any app, the Collections icon displays on the left side of the screen.

Some tasks related to managing a collection are available in other menus:

- Configure a profile in **Indexing > Indexing Profiles** or **Querying > Query Profiles**.
- View reports about your collection's activity in **Analytics > Dashboards**.

Collections Manager

The Collections Manager page displays details about the collection, such as how many datasources are configured, how many documents are in the index, and how much disk space the index consumes.

This page also lets you *create a new collection*, disable search logs or signals, enable recommendations, issue a commit command to Solr, or clear a collection.

DISABLE SEARCH LOGS

When you first create a collection, the search logs are created by default. The search logs populate the panels in **Analytics > Dashboards**.

To disable search logs:

1. Hover over your collection name until the gear icon appears at the end of the line.
2. Click the gear icon.
3. Click **Disable Search Logs**.
4. On the confirmation screen, click **Disable Search Logs**.

Note that if you disable search logs, you cannot see any data for this collection in **Analytics > Dashboards**.

See *Dashboards*.

DISABLE SIGNALS

When you first create a collection, the signals and aggregated signals collections are created by default.

To disable signals:

1. Hover over your collection name until the gear icon appears at the end of the line.
2. Click the gear icon.
3. Click **Disable Signals**.
4. On the confirmation screen, click **Disable Signals**.

HARD COMMIT A COLLECTION

To issue a hard commit command to Solr:

1. Hover over your collection name until the gear icon appears at the end of the line.
2. Click the gear icon.
3. Click **Hard Commit Collection**.
4. On the confirmation screen, click **Hard Commit Collection**.

Read internal details about how Solr processes commits on our blog .

Datasources

To access the Datasources page, click **Indexing > Datasources**. By default, there are no datasources configured right after installation.

To add a new datasource, click **New** at the upper right of the panel.

See the *Connectors Configuration Reference* for details on how to configure a datasource. Options vary depending on the repository you would like to index.

After you configure a datasource, it appears in a list on this screen. Click the name of a datasource to edit its properties. Click **Start** to start the datasource. Click **Stop** to stop the datasource before it completes. To the right, view information on the last completed job, including the date and time started and stopped, and the number of documents found as new, skipped, or failed.

Note

When you stop a datasource, Managed Fusion attempts to safely close connector threads, finishing processing documents through the pipeline and indexing documents to Solr. Some connectors take longer to complete these processes than others, so might stay in a "stopping" state for several minutes.

To stop a datasource immediately, choose **Abort** instead of **Stop**.

There is also a REST API for datasources. See *Connector Datasources API*.

Stopwords

The Stopwords page lets you edit a stopwords list for your collection.

To add or delete stop words:

1. Click the name of the text file you wish to edit.
2. Add a new word on a new line.
3. When you are done with your changes, click **Save**.

To import a stop words list:

1. Click **System > Import Fusion Objects**.
2. Choose the file to upload.
3. Click **Import >>**.

Read more about *stopwords*.

Synonyms

Managed Fusion has the same synonym functionality that Solr supports. This includes a list of words that are synonyms (where the synonym list expands on the terms entered by the user), as well as a full mapping of words, where a word is substituted for what the user has entered (that is, the term the user has entered is replaced by a term in the synonym list).

See more about *synonyms*.

You can edit the synonyms list for your collection.

To access the Synonyms page in the Managed Fusion UI, in any app, click **Collections > Synonyms**.

Filter the list of synonym definitions by typing in the **Filter...** box.

To import a synonyms list:

1. From the Synonyms page, click **Import and Save**. A dialog box opens.
2. Choose the file to import.

To edit a synonyms list:

- Enter new synonym definitions one per line.
    - To enter a string of terms that expand on the terms the user entered, enter the terms separated by commas, like `Television, TV`.
    - To enter a term that should be mapped to another term, enter the terms separated by an equal sign then a right angle bracket, `⇒`, like `i-pod⇒ipod`.
- Remove a line by clicking the **x** at the end of the line.
- Once you are finished with edits, click **Save**.

To export the synonyms list, click **Export**. This downloads the list to your computer using your browser download capability.

Profiles

Profiles allow you to create an alias for an index or query pipeline. This allows you to send documents or queries to a consistent endpoint and change the underlying pipeline or collection as needed.

Read about profiles in Index Profiles and Query Profiles:

- *Index Profiles*
- *Query Profiles*

To access the Solr Config page, from any app, click **System > Solr Config**.

Additional resources

LucidAcademy

Lucidworks offers free training to help you get started.

The **Quick Learning** for **Collections Menu Tour** focuses on the Collections Menu features and functionality along with a brief description of each screen available in the menu:

Visit the LucidAcademy to see the full training catalog.

Manage Dashboards

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

Set Permissions

Users who manage or use dashboards need specific *permissions* to do so.

Permissions to Manage Dashboards

The built-in roles `admin` and `developer` both have sufficient permissions to manage dashboards.

For users who manage dashboards, set these permissions in *roles*, and then assign the roles to users:

```
GET:/solr/{id}/\*:id=*_collection-name_*
GET:/solr/{id}/admin/luke:id=*_collection-name_*
GET,POST,PUT,DELETE:/solr/system_blobs/\*
GET,POST,PUT,DELETE:/collections/system_blobs
```

Permissions to Use Dashboards

The built-in role `search` *does not* have sufficient permissions to use dashboards.

For users who manage dashboards, set these permissions in *roles*, and then assign the roles to users:

```
GET:/solr/{id}/\*:id=*_collection-name_*
GET:/solr/{id}/admin/luke:id=*_collection-name_*
GET:/solr/system_blobs/\* GET:/collections/system_blobs
```

Manage Dashboards

Manage Dashboards

How-to

> Compatible versions: 4.2, 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion Server*, *Fusion*

Set Permissions

Users who manage or use dashboards need specific *permissions* to do so.

Permissions to Manage Dashboards

The built-in roles `admin` and `developer` both have sufficient permissions to manage dashboards.

For users who manage dashboards, set these permissions in *roles*, and then assign the roles to users:

```
GET:/solr/{id}/\*:id=*_collection-name_*
GET:/solr/{id}/admin/luke:id=*_collection-name_*
GET,POST,PUT,DELETE:/solr/system_blobs/\*
GET,POST,PUT,DELETE:/collections/system_blobs
```

Permissions to Use Dashboards

The built-in role `search` *does not* have sufficient permissions to use dashboards.

For users who manage dashboards, set these permissions in *roles*, and then assign the roles to users:

```
GET:/solr/{id}/\*:id=*_collection-name_*
GET:/solr/{id}/admin/luke:id=*_collection-name_*
GET:/solr/system_blobs/\* GET:/collections/system_blobs
```

Manage Dashboards

This topic details the options available to use Fusion dashboards.

Dashboards main page

When you open the Fusion dashboards, your organization's dashboards display, as well as the EXPLORE DASHBOARDS section that provides options to create or display specific types of dashboards.

You can click the icons in the top right corner of the screen to perform the following functions:

Option	Notes
Goto saved default	Open the saved default dashboard.
New	Create a new dashboard. Options are: • Time-series dashboard • Non time-series dashboard
Load	Load a dashboard using one of the following options: • Load dashboard JSON layout from file Click **Local File > Choose File > select file > Open**. Saved dashboards are listed at the bottom of the dialog box. Click a dashboard to open it. Filter dashboards by entering an exact-match string in the Filter box. If there are more that 20 saved dashboards, use the pagination controls to find a dashboard. To view the next set of pages, click the forward arrows. To view the prior set of pages, click the back arrows. Or click a page number to view the dashboards on that page. • Enter the gist URL or number (the last part of the URL) **Enter value > click Get** . A link that opens the dashboard appears below a thin line. The link displays the dashboard name. Click the link to open the dashboard. • Type to filter. Enter dashboard type to filter and display if the dashboard matches.
Save	Save options include: • Locally ◦ Export to File ◦ Set as Browser Default. Click to set the current dashboard as the default. ◦ Clear Browser Default. Click to clear the current dashboard. The default dashboard for the browser becomes the system default dashboard `default.json` (the Fusion Logs dashboard). • Solr ◦ Make Public. If this checkbox is selected, the dashboard is shared with everyone. ◦ Service Logs. Enter a value and click **Save** to save the log.

Option	Notes
Configure Dashboard	Click to configure tabs for General, Rows, Control, and Solr dashboard settings.
Delete a dashboard saved in Solr	1. Click **Load**. 2. Identify the file to delete and click **Delete**. 3. Click **OK** to confirm the deletion.
Delete a dashboard saved in a file	Navigate to the file and delete it.
Delete a dashboard saved as a Github gist	Open the gist URL and click **Delete**.

For more information, see:

- *Fusion 4.x Built-in Dashboards*
- *Fusion 5.x Built-in Dashboards*

Interact with data

The table describes options to interact with data displayed in certain dashboards.

Option	Notes
Collections	Choose the collection from which to obtain data. If a dashboard is configured to let users choose among collections, select the collection from the **Collections** drop-down list at the top right of the dashboard.
Search	Enter the value to search. • To search for a specific value, enter a term or phrase and click **Search**. • To retrieve all data, enter *.* and click **Search**.
Clear search box	Click to clear the search box and return the most general query, as if you entered *.*.
Waiting for data	Indicates the query is waiting for data to be returned. If the indicator spins for an extended time while other panels display data, the panel is not configured correctly.
Select item	Click a table row. In a graphical-visualization panel, click the data item. **Note** Not all panel types let you select data in this way.
Export data	Click to enter the number of rows to export and select the destination file type. For example, CSV, XML, or JSON.
Hide row	Click to hide the row from the display.
Expand row	Click the row to display detailed information.

Interact with specific types of panels

The table describes options for specific types of panels.

Table and text panels

Option	Notes
Show or hide the field list	Click the right or left arrow, respectively, at the top of the panel.
Sort a table column	Click the up arrow at the top of the column to sort in ascending order. Click the down arrow at the top of the column to sort in descending order.
View next or previous page of hits	Click the right or left arrow, respectively, at the bottom of the page of hits.
  View first page	Click to view the first page of hits for that particular panel.
Show record details	Click the table row and then select the display format of Table, JSON, or Raw.
Hide details	Click the up arrow at the top right of the details.

Histogram and Range Facet panels

The table describes options for the Histogram and Range Facet panels.

Option	Notes
Zoom in on a subset of data	Click the area you want to enlarge, then drag across the range of data to zoom in.
  Zoom out	Click to zoom out.

Filtering panels

The table describes filter options for panels.

Option	Notes
Edit filter	Click to edit a panel filter.
Toggle filter	Click to toggle the filter to the opposite setting (on or off).
Remove filter	Click to remove the current filter. **Caution** To display correctly, specific filters must be set for certain types of panels. Fusion also creates filters based on your interactions with data. You can add a filter back, but there is no undo. Before you remove a filter, toggle off the filter and verify the results. If the results are accurate, it is safe to remove it.

Heat Map panels

The table describes the option for heat map panels.

Option	Notes
Transpose rows and columns	Click to reverse the display of rows and columns.

Manage user roles

How-to

> Works with: Connected Search

This topic details how to manage *user roles and permissions* in Springboard.

> **Important**
>
> You must be a workspace owner to edit user roles.

1. *Sign in to Springboard*.
2. In the *Applications Manager screen*, click **Users**. For more information, see *View users*.
3. Point to the user to edit and click **View/Edit** to the right of the entry.
4. To change the user's workspace role, select the role from the **Workspace Role** dropdown menu.

   > **Note**
   >
   > Workspace owners have full access to all applications in your Springboard workspace and do not need permission to view specific applications.

5. To change a user's access to specific applications, find the application in the list. You may use the filter to find the application.
6. Select the role to give the user. See *Roles* for more details.
7. To save the changes, click **Save**. To exit without saving the edits, click **X**.

After you save the changes, the user must sign out and sign in again to view the changes.

Additional information

For conceptual information, see *Roles*.

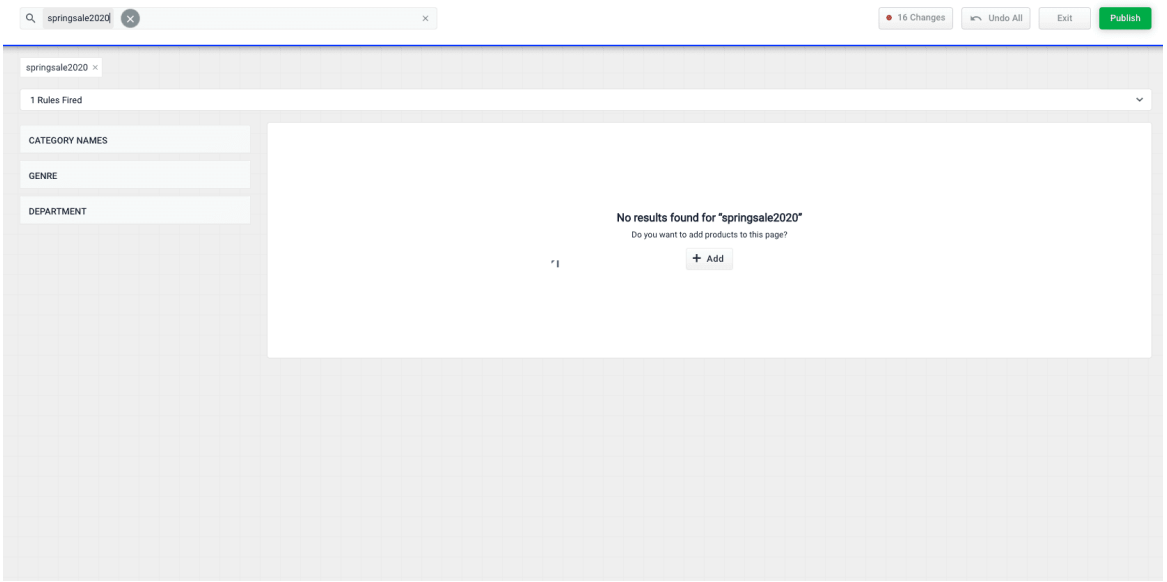
To add and remove multiple users from one application, see *Add users to an application*.

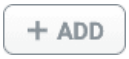
Merchandising Zero Results

How-to

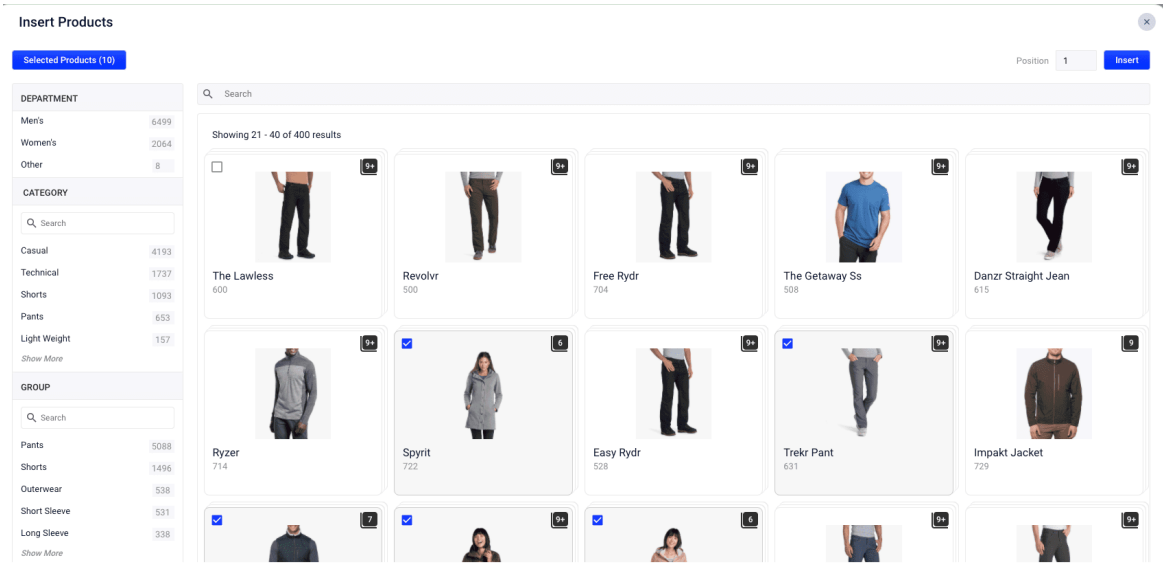
> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion*

1. Enter a search query the generates 0 results:

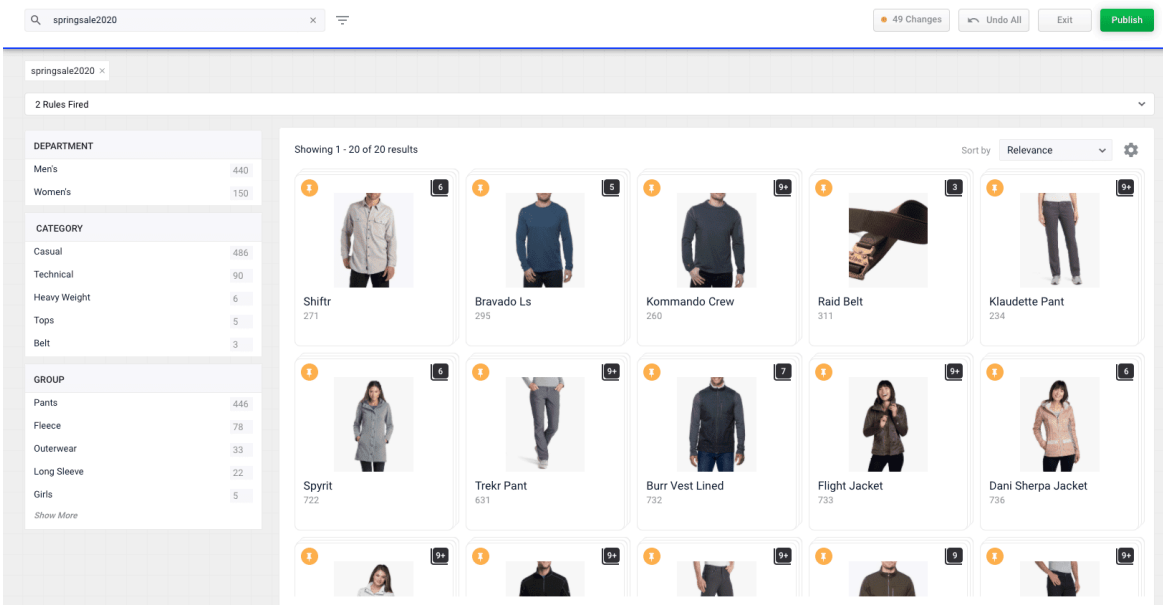


2. Click the **Add** button: 

3. Select products or *product groups* that you want to associate with the 0 results query:



4. Click the **Insert** button. The selected products are now displayed when the search query is used, resolving the 0 results query:

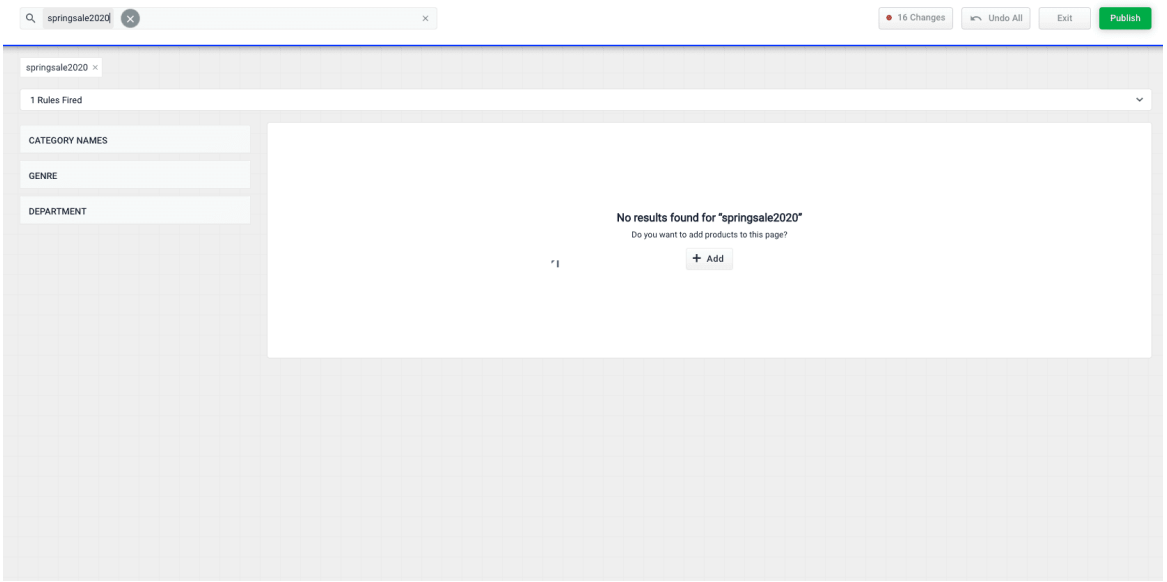

springsale2020
49 Changes
Undo All
Exit
Publish
springsale2020
2 Rules Fired
DEPARTMENT
Men's 440
Women's 150
CATEGORY
Casual 486
Technical 90
Heavy Weight 6
Tops 5
Belt 3
GROUP
Pants 446
Fleece 78
Outerwear 33
Long Sleeve 22
Girls 5
Show More
Showing 1 - 20 of 20 results
Sort by
Relevance
Shiftr 271
Bravado Ls 295
Kommando Crew 260
Raid Belt 311
Klaudette Pant 234
Spyrit 722
Trekr Pant 631
Burr Vest Lined 732
Flight Jacket 733
Dani Sherpa Jacket 736

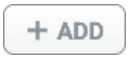
Merchandising Zero Results

How-to

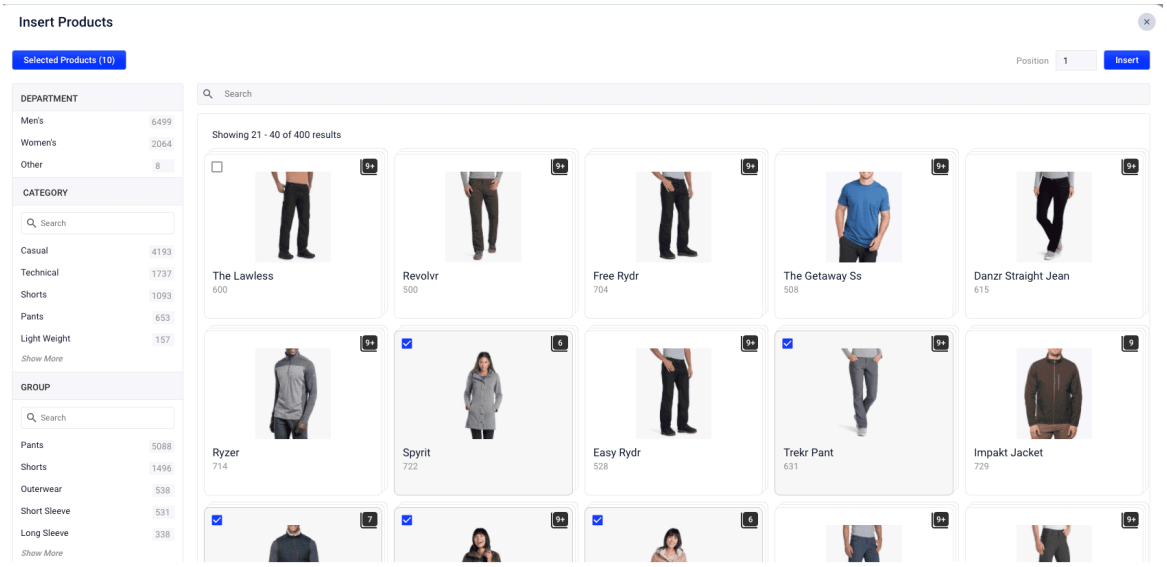
> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

1. Enter a search query the generates 0 results:

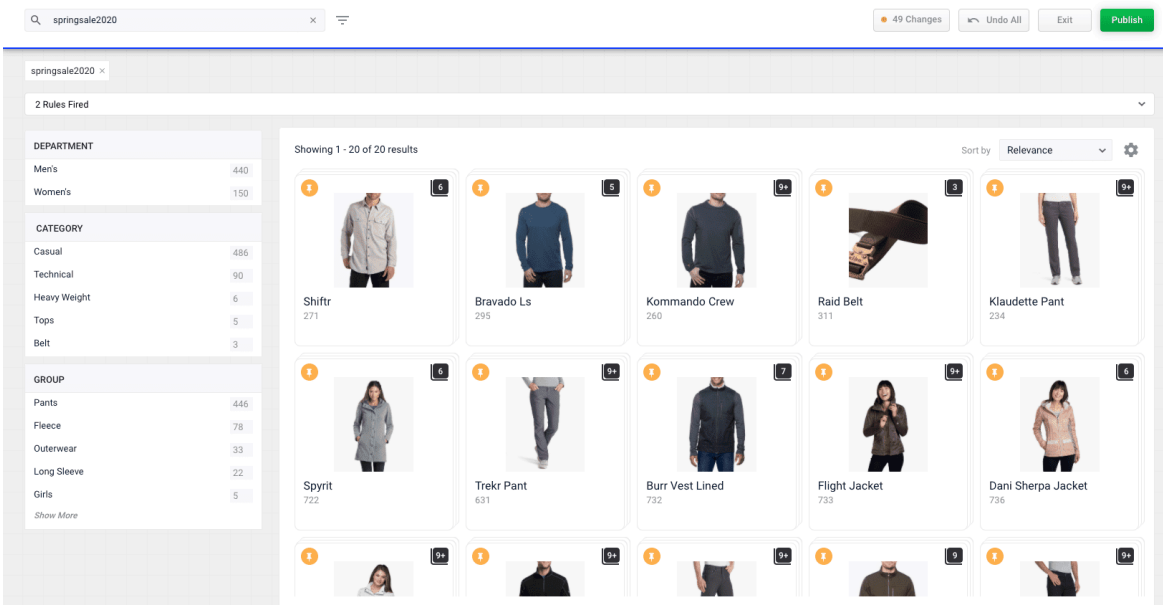


2. Click the **Add** button: 

3. Select products or product groups that you want to associate with the 0 results query:



4. Click the **Insert** button. The selected products are now displayed when the search query is used, resolving the 0 results query:


springsale2020
49 Changes
Undo All
Exit
Publish
springsale2020
2 Rules Fired
DEPARTMENT
Men's 440
Women's 150
CATEGORY
Casual 486
Technical 90
Heavy Weight 6
Tops 5
Belt 3
GROUP
Pants 446
Fleece 78
Outerwear 33
Long Sleeve 22
Girls 5
Show More
Showing 1 - 20 of 20 results
Sort by
Relevance
Shiftr 271
Bravado Ls 295
Kommando Crew 260
Raid Belt 311
Klaudette Pant 234
Spyrit 722
Trekr Pant 631
Burr Vest Lined 732
Flight Jacket 733
Dani Sherpa Jacket 736

Migrate Fusion Objects

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion*

You can use the Fusion UI and the *Objects API* to migrate collections and related objects, including your searchable data, configuration data, pipelines, aggregations, and other objects on which your collections depend. You can also migrate entire apps.

Important

To upgrade from one Fusion version to a 5.x version, see *Fusion 5 Upgrades* to migrate objects.

You might need to migrate objects in the following circumstances:

- When migrating data from development environments into testing and production environments
- To back up data, so you can restore it after an incident of data loss
- When the migrator script was not able to migrate an object automatically

Caution

When you export and import objects to migrate them, make sure you that any data gets to where it is going before deleting the sources.

Migration approaches

Several approaches are available for migrating Fusion objects. This table summarizes the approaches.

	Export an app	**Import an app**	**Export an object**	**Import an object**	**Add an object to an app**
Fusion UI	App configuration	Launcher (entire app) App configuration (combine apps)	-	-	Object Explorer
Objects API (endpoints)	GET from `export` endpoint	POST to `import` endpoint	GET from `export` endpoint	POST to `import` endpoint	-

For more information about using the Objects API to export and import objects, see *Objects API*.

The remainder of this topic describes approaches in the Fusion UI.

Use the parts of the Fusion UI indicated in the table to export and import apps and specific objects. Exporting creates a zip file. To import, you select a data file and possibly a variable file.

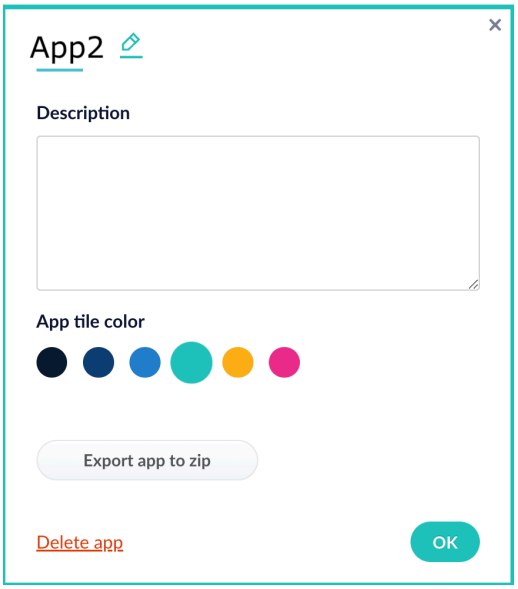
The approach with Object Explorer differs. With Object Explorer, you can add objects from other apps (or that are not linked to any apps) to the currently open app.

Export an app with the Fusion UI

How to export an app with the Fusion UI

1. Navigate to the launcher.
2. Hover over the app you want to export and click the Configure icon:

3. In the app config window, click **Export app to zip**:

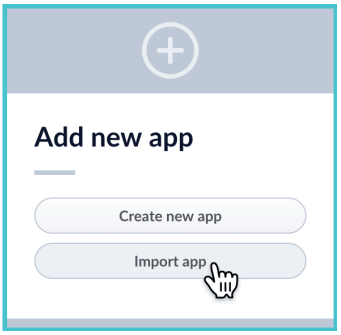


See *import* for information to import the downloaded zip file into other instances of Fusion 5.x.

Import an app with the Fusion UI

How to import an app with the Fusion UI

1. Navigate to the launcher.
2. Click **Import app**.

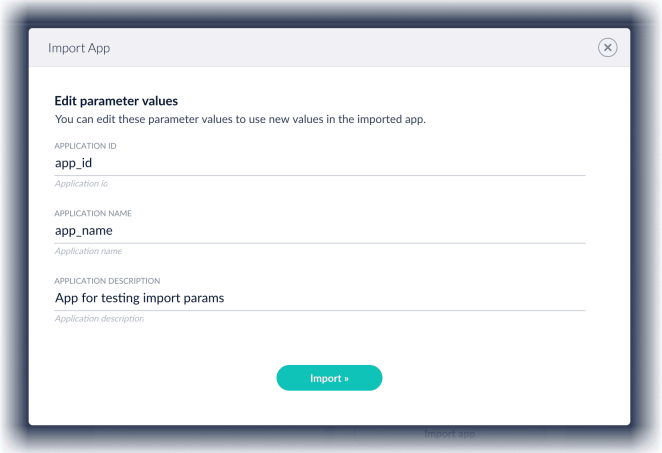


3. Under **Data File**, click **Choose File** and select the zip file containing the app you want to import.

4. If your app has *usernames and passwords* in a separate file, select it under **Variables File**.
   If the Variables File is needed, it *must* be a separate file that is not in a .zip file. It is a .json map of variables to values. The following is an example:

```
{ "secret.dataSources.Inventory_BR_S3_DS.password":"SOMETHING",
"secret.dataSources.LLM_A_BR_S3_DS.password":"FmJSaDE9Tj5REDACTED",
"secret.dataSources.LLM_BR_Inventory_S3_DS.password":"FmJSaDE9Tj5GzIVv
ethAC4Huh",
"secret.dataSources.LLM_BR_Load_S3_DS.password":"FmJSaDE9Tj5GzIVvethAC
4" }
```

5. You can sometimes edit parameter values to use the new values in the imported app. If this is the case, Fusion displays a dialog box that lets you edit the parameter values.



   Make desired changes, and then click **Import**.

Copy an app

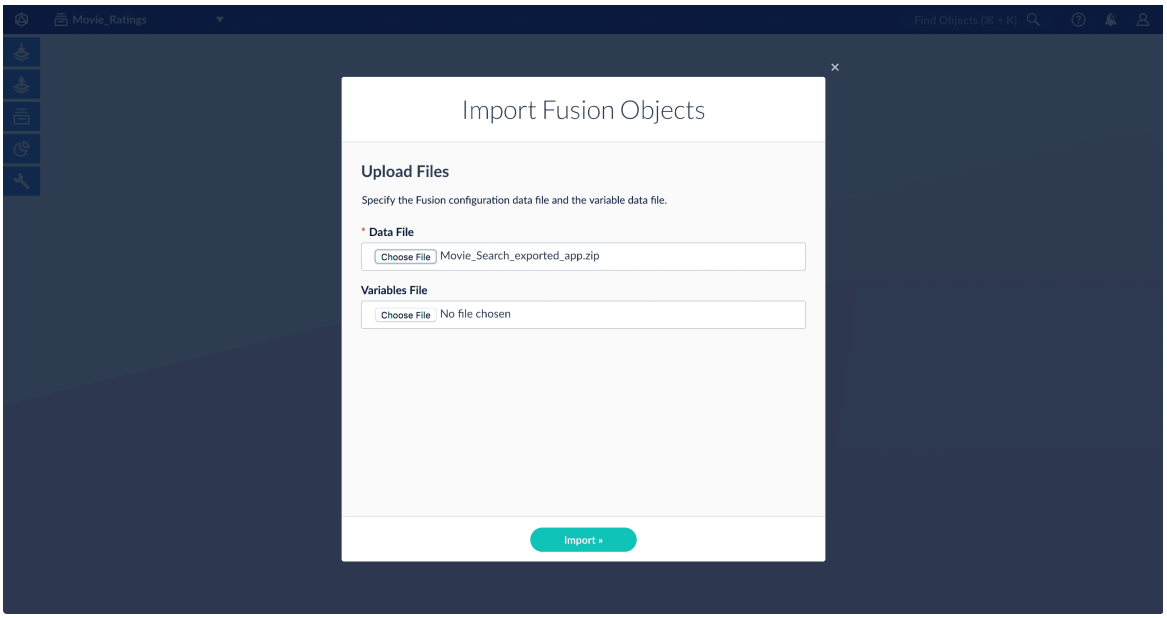
To copy an app from one deployment to a different one, *export the app* on the source deployment, and then *import the app* on the target deployment.

Import objects into an app

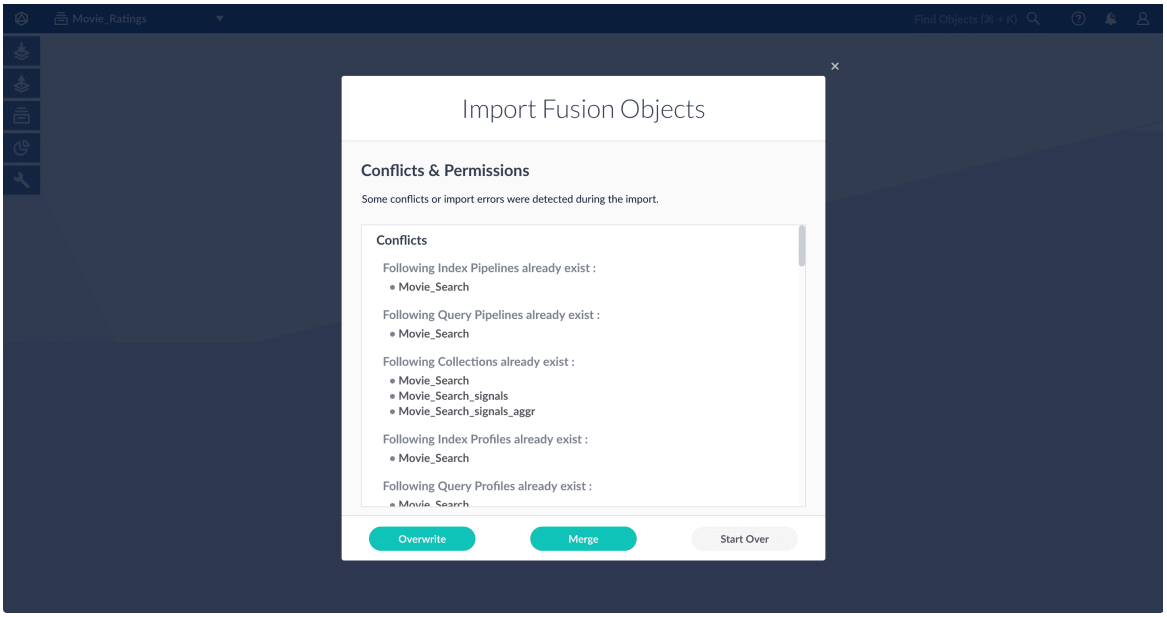
You can import objects into the currently open app.

How to import objects into an open app

1. In the Fusion launcher, click the app into which you want to import objects.
   The Fusion workspace appears.
2. Click **System > Import Fusion Objects**.
   The Import Fusion Objects window opens.

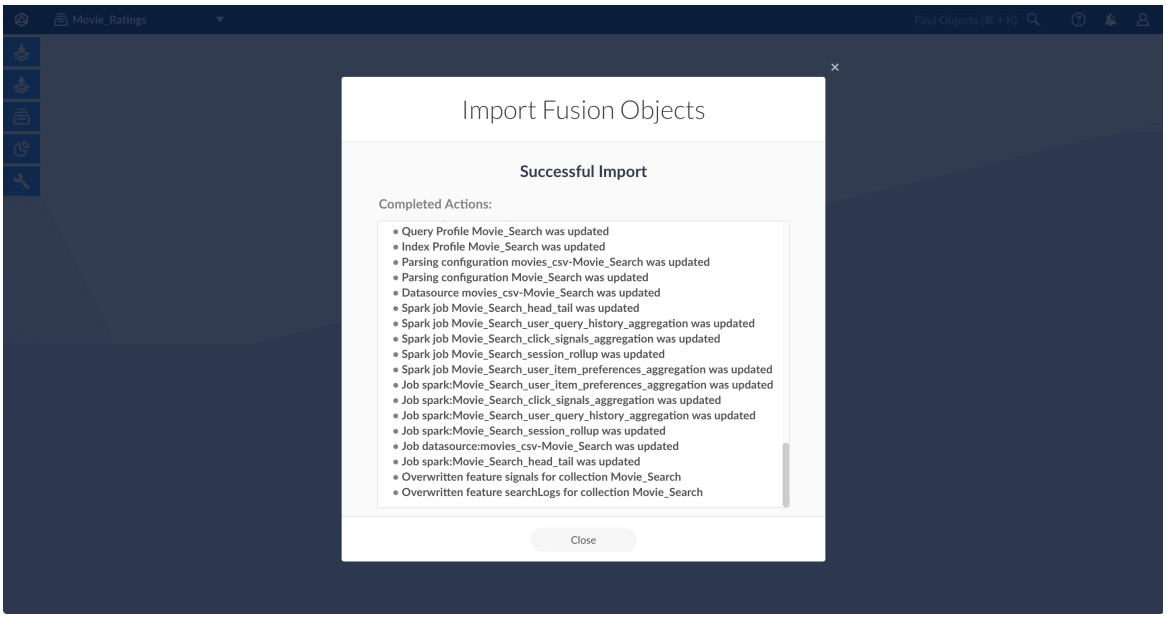


3. Select the data file from your local filesystem.
   If you are importing *usernames and passwords* in a separate file, select it under **Variables File**.
4. Click **Import**.
   If there are conflicts, Fusion prompts you to specify an import policy:



   - Click **Overwrite** to overwrite the objects on the target system with the ones in the import file.
   - Click **Merge** to skip all conflicting objects and import only the non-conflicting objects.
   - Click **Start Over** to abort the import.

Fusion confirms that the import was successful:

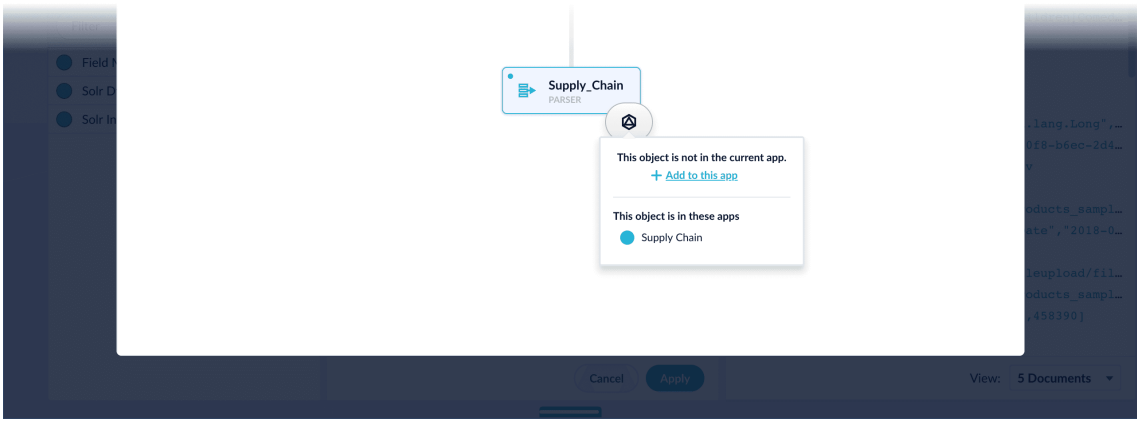


5. Click **Close** to close the Import Fusion Objects window.

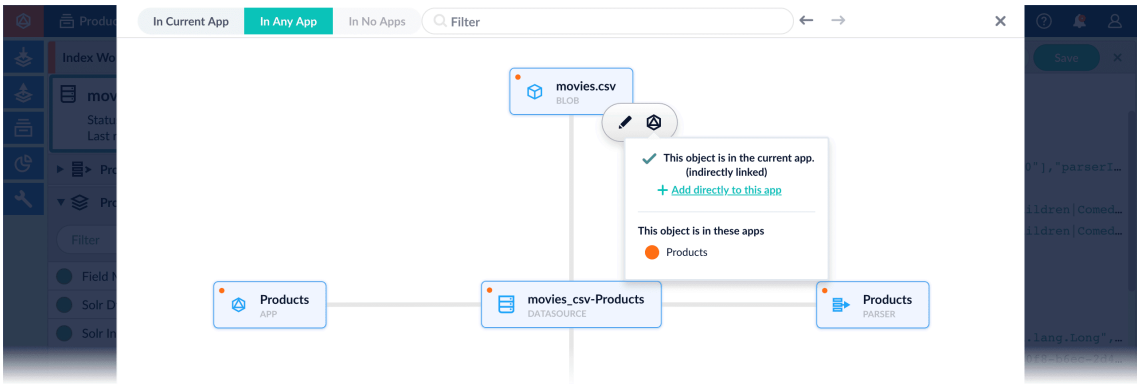
Add an object to an app

You can add objects present in other apps (or in no apps) to the open app. Some objects are linked to other apps. You can also add those directly to an app.

- **Add an object to an app** – While in the Fusion workspace for the app to which you want to add an object, open Object Explorer and click **In Any App**. Search for or browse to the object you want to add. Hover over the object, click the App icon, and then click **Add to this app**.



- **Add an object to an app directly** – In cases when an object is linked to an app, but is not linked directly to the app (it is linked via some dependency), you can add the object to an app directly.
  While in the Fusion workspace for the app to which you want to add an object directly, open Object Explorer and click **In Any App**. Search for or browse to the object you want to add. Hover over the object, click the App icon, and then click **Add to this app directly**.


In Current App
In Any App
In No Apps
Filter
movies.csv
BLOB
This object is in the current app.
(indirectly linked)
Add directly to this app
This object is in these apps
Products
Products
APP
movies_csv-Products
DATASOURCE
Products
PARSER

Migrate Fusion Objects

How-to

> Compatible versions: 4.2
>
> Works with: *Fusion Server*

You can use the Fusion UI and the *Objects API* to migrate collections and related objects, including your searchable data, configuration data, pipelines, aggregations, and other objects on which your collections depend. You can also migrate entire apps.

Important

When upgrading from one Fusion version to a later one, you must use the *migrator* to migrate objects. The migrator automates the process of translating relevant objects to the new version. Exporting an object from one Fusion version and importing it into a different Fusion version *isn't* supported.

You might need to migrate objects in the following circumstances:

- When migrating data from development environments into testing and production environments
- To back up data, so you can restore it after an incident of data loss
- When the migrator script was not able to migrate an object automatically

Caution

When you export and import objects to migrate them, make sure you that any data gets to where it is going before deleting the sources.

Migration approaches

Several approaches are available for migrating Fusion objects. This table summarizes the approaches.

	Export an app	**Import an app**	**Export an object**	**Import an object**	**Add an object to an app**
Fusion UI	App configuration	Launcher (entire app) App configuration (combine apps)	-	-	Object Explorer
Objects API (endpoints)	GET from `export` endpoint	POST to `import` endpoint	GET from `export` endpoint	POST to `import` endpoint	-

For more information about using the Objects API to export and import objects, see *Objects API*.

The remainder of this topic describes approaches in the Fusion UI.

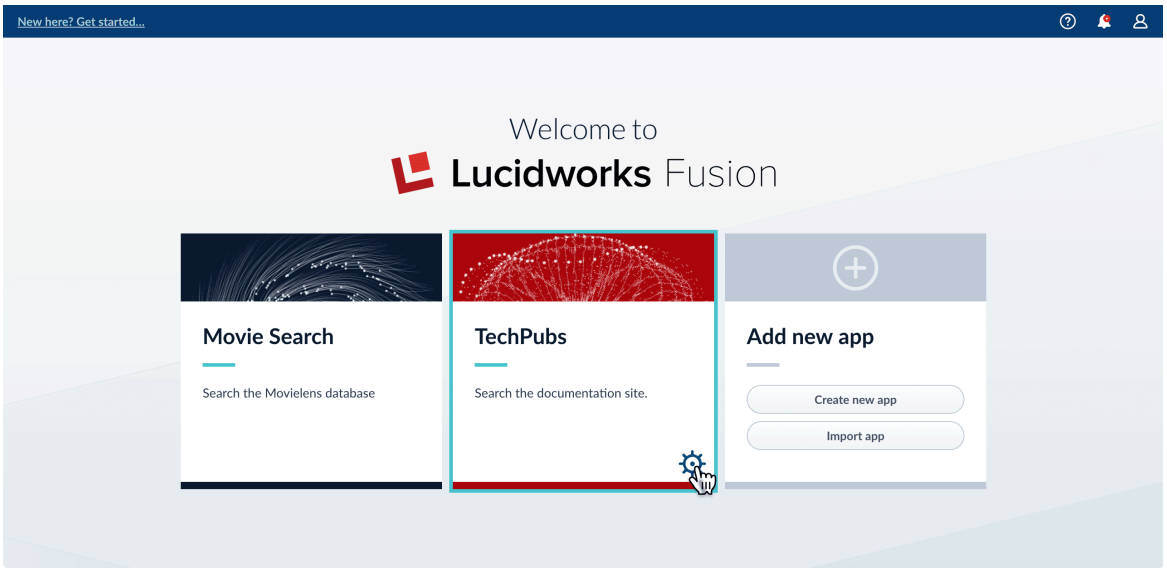
Use the parts of the Fusion UI indicated in the table to export and import apps and specific objects. Exporting creates a zip file. To import, you select a data file and possibly a variable file.

The approach with Object Explorer differs. With Object Explorer, you can add objects from other apps (or that are not linked to any apps) to the currently open app.

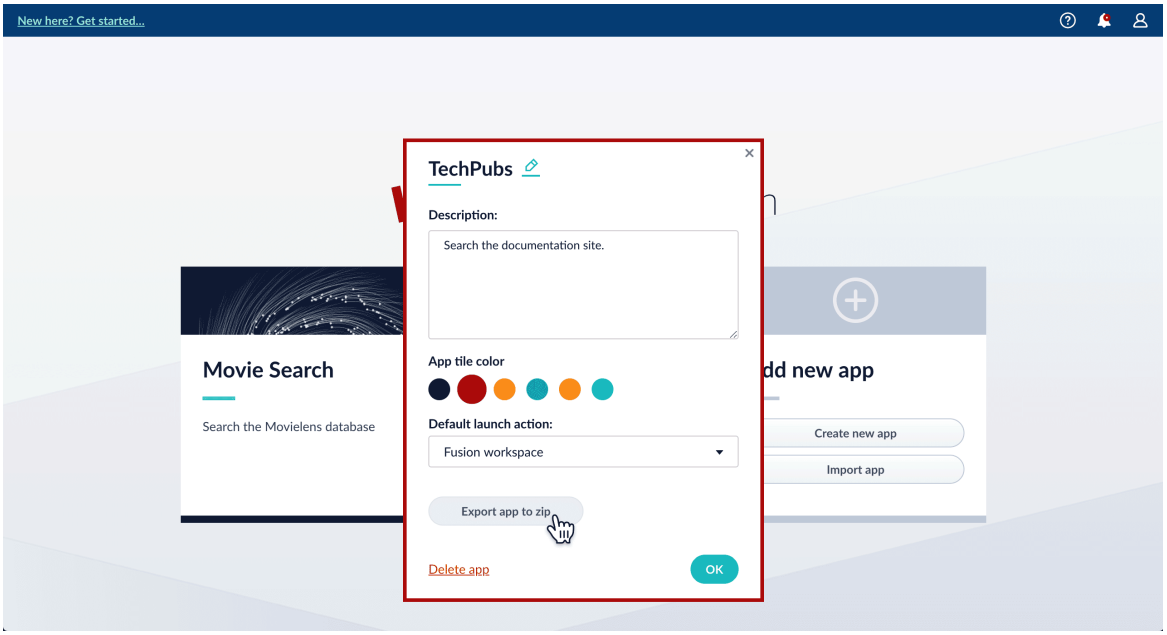
Export an app with the Fusion UI

How to export an app with the Fusion UI

1. Navigate to the launcher.
2. Hover over the app you want to export and click the Configure icon:



3. In the app config window, click **Export app to zip**:

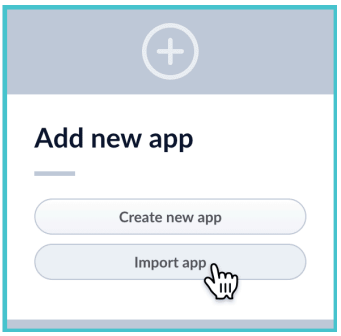


See *import* for information to import the downloaded zip file into other instances of Fusion 4.x Server.

Import an app with the Fusion UI

How to import an app with the Fusion UI

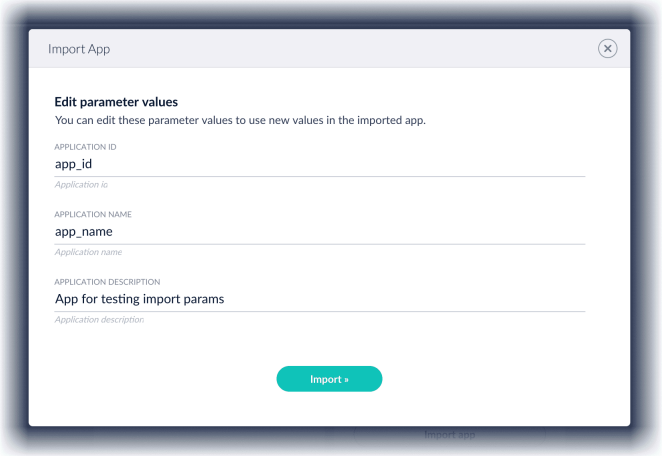
1. Navigate to the launcher.
2. Click **Import app**.



3. Under **Data File**, click **Choose File** and select the zip file containing the app you want to import.
4. If your app has *usernames and passwords* in a separate file, select it under **Variables File**.
   If the Variables File is needed, it *must* be a separate file that is not in a .zip file. It is a .json map of variables to values. The following is an example:

```
{ "secret.dataSources.Inventory_BR_S3_DS.password":"SOMETHING",
"secret.dataSources.LLM_A_BR_S3_DS.password":"FmJSaDE9Tj5REDACTED",
"secret.dataSources.LLM_BR_Inventory_S3_DS.password":"FmJSaDE9Tj5GzIVv
ethAC4Huh",
"secret.dataSources.LLM_BR_Load_S3_DS.password":"FmJSaDE9Tj5GzIVvethAC
4" }
```

5. In Fusion 4.1 and later, you can sometimes edit parameter values to use the new values in the imported app. If this is the case, Fusion displays a dialog box that lets you edit the parameter values.



   Make desired changes, and then click **Import**.

Copy an app

To copy an app from one deployment to a different one, *export the app* on the source deployment, and then *import the app* on the target deployment.

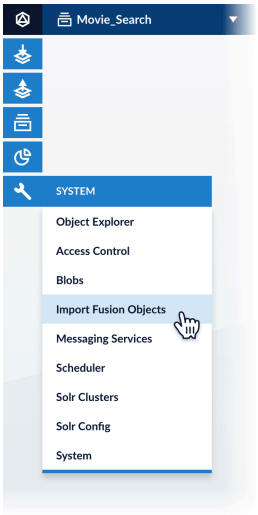
Import objects into an app

You can import objects into the currently open app.

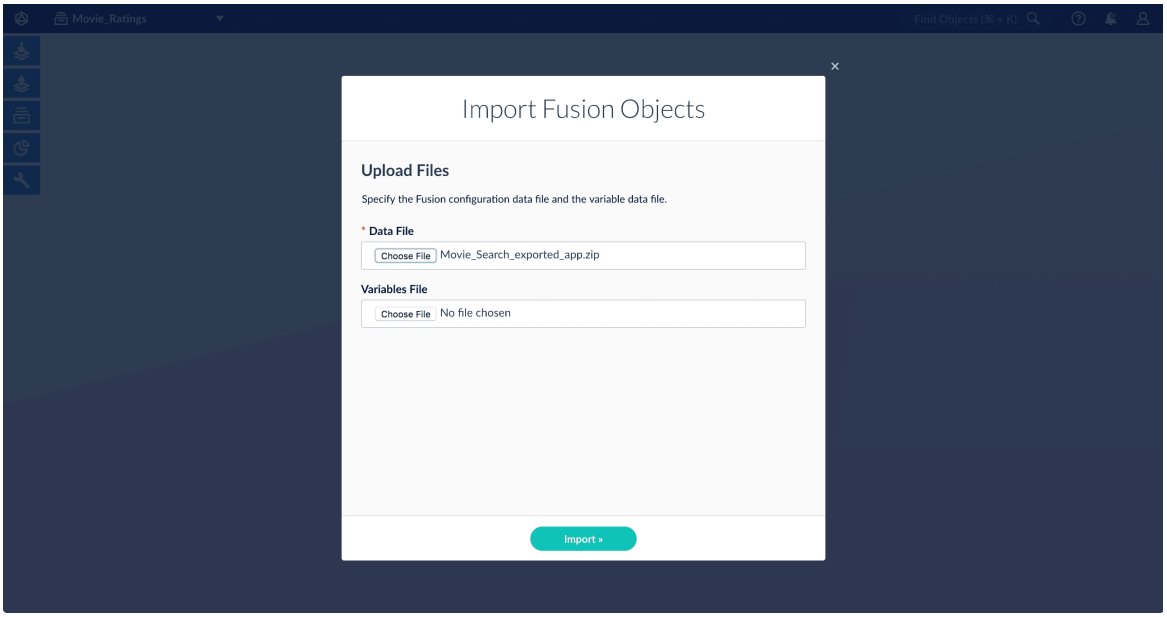
How to import objects into an open app

1. In the Fusion launcher, click the app into which you want to import objects.
   The Fusion workspace appears.

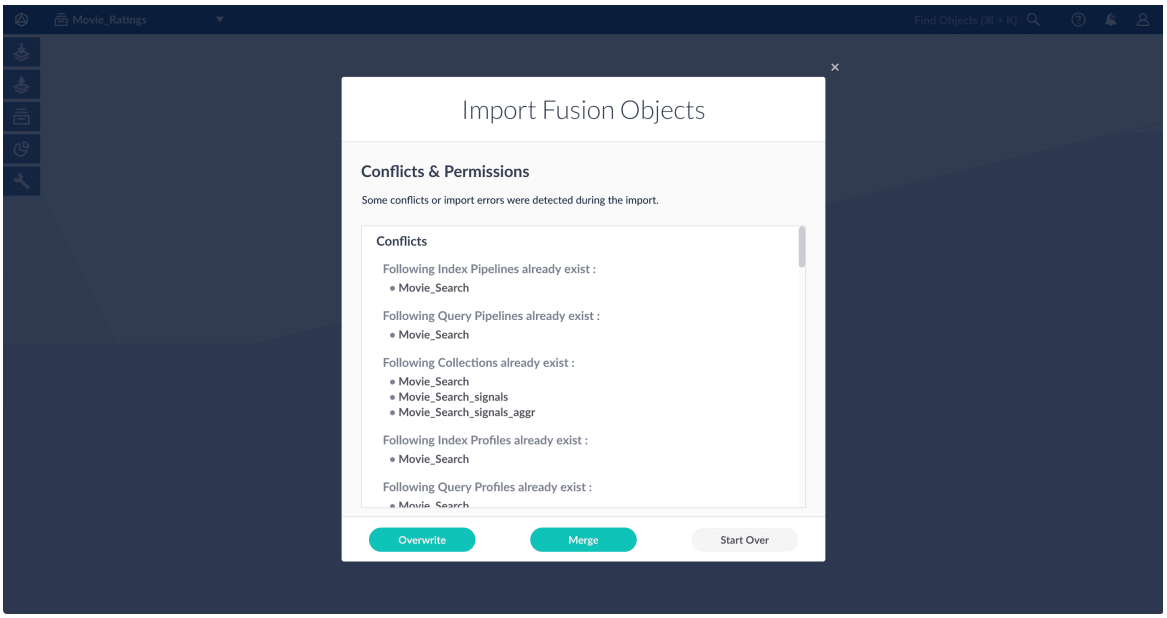
2. In the upper left, click System > **Import Fusion Objects**.



   The Import Fusion Objects window opens.

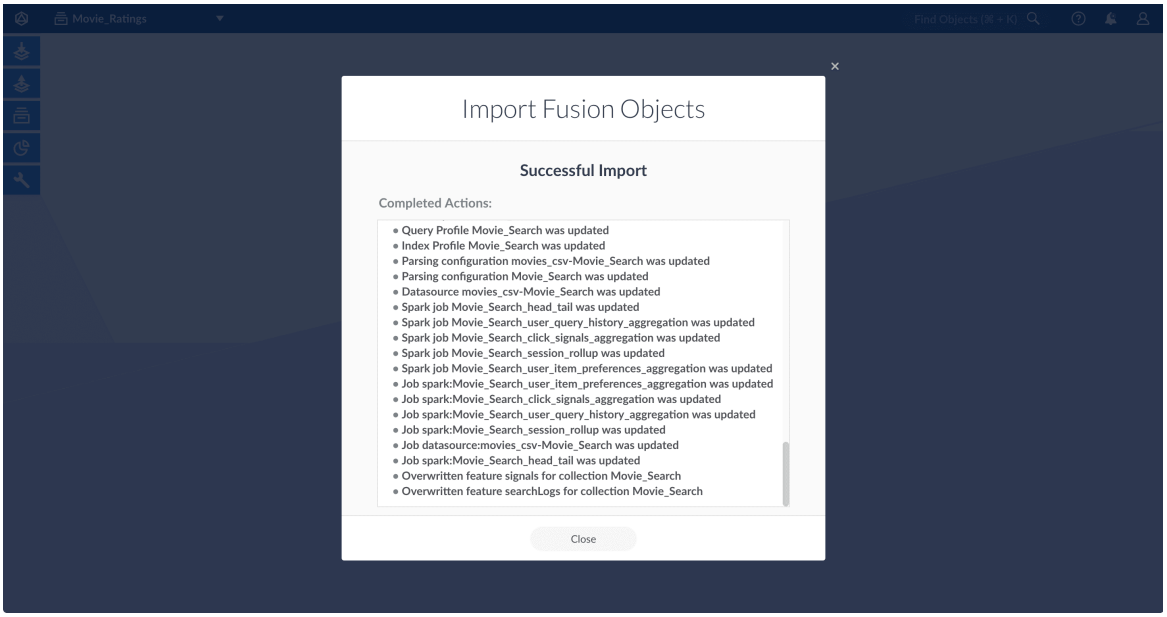


3. Select the data file from your local filesystem.
   If you are importing *usernames and passwords* in a separate file, select it under **Variables File**.
4. Click **Import**.
   If there are conflicts, Fusion prompts you to specify an import policy:



- Click **Overwrite** to overwrite the objects on the target system with the ones in the import file.
- Click **Merge** to skip all conflicting objects and import only the non-conflicting objects.
- Click **Start Over** to abort the import.

Fusion confirms that the import was successful:



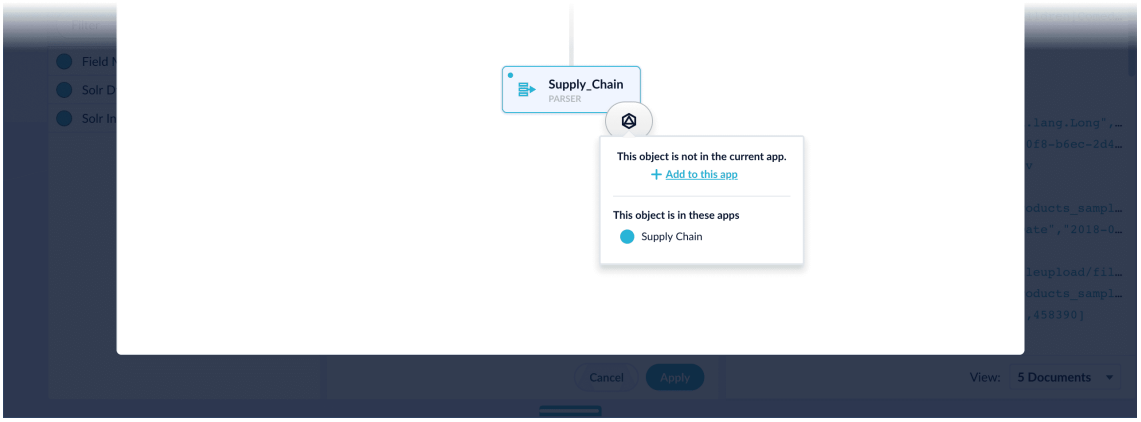
5. Click **Close** to close the Import Fusion Objects window.

Add an object to an app

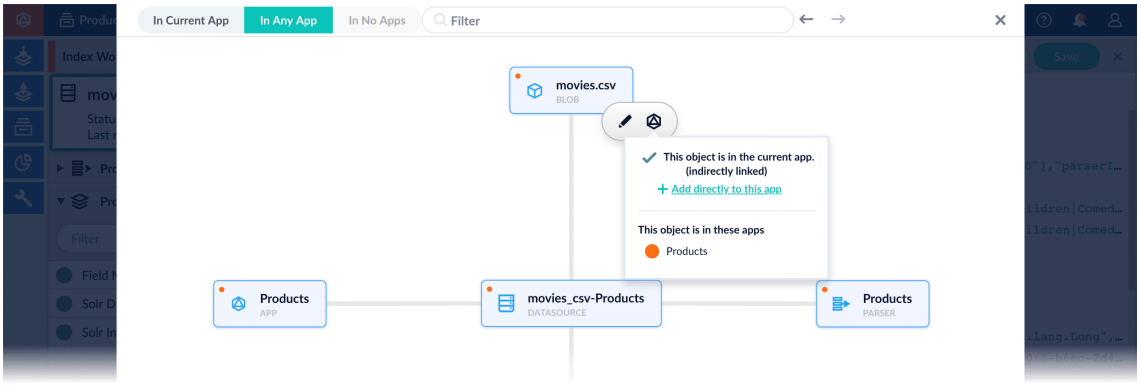
You can add objects present in other apps (or in no apps) to the open app. Some objects are linked to other apps. You can also add those directly to an app.

- **Add an object to an app** – While in the Fusion workspace for the app to which you want to add an object, open Object Explorer and click **In Any App**. Search for or browse to the object you want to add. Hover

over the object, click the App icon, and then click **Add to this app**.



- **Add an object to an app directly** – In cases when an object is linked to an app, but is not linked directly to the app (it is linked via some dependency), you can add the object to an app directly.
  While in the Fusion workspace for the app to which you want to add an object directly, open Object Explorer and click **In Any App**. Search for or browse to the object you want to add. Hover over the object, click the App icon, and then click **Add to this app directly**.



Migrate Managed Fusion Objects

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

You can use the Fusion UI and the *Objects API* to migrate collections and related objects, including your searchable data, configuration data, pipelines, aggregations, and other objects on which your collections depend. You can also migrate entire apps.

Important

To upgrade from one Fusion version to a 5.x version, see *Fusion 5 Upgrades* to migrate objects.

You might need to migrate objects in the following circumstances:

- When migrating data from development environments into testing and production environments
- To back up data, so you can restore it after an incident of data loss
- When the migrator script was not able to migrate an object automatically

Caution

When you export and import objects to migrate them, make sure you that any data gets to where it is going before deleting the sources.

Migration approaches

Several approaches are available for migrating Fusion objects. This table summarizes the approaches.

	Export an app	**Import an app**	**Export an object**	**Import an object**	**Add an object to an app**
Fusion UI	App configuration	Launcher (entire app) App configuration (combine apps)	-	-	Object Explorer
Objects API (endpoints)	GET from `export` endpoint	POST to `import` endpoint	GET from `export` endpoint	POST to `import` endpoint	-

For more information about using the Objects API to export and import objects, see *Objects API*.

The remainder of this topic describes approaches in the Fusion UI.

Use the parts of the Fusion UI indicated in the table to export and import apps and specific objects. Exporting creates a zip file. To import, you select a data file and possibly a variable file.

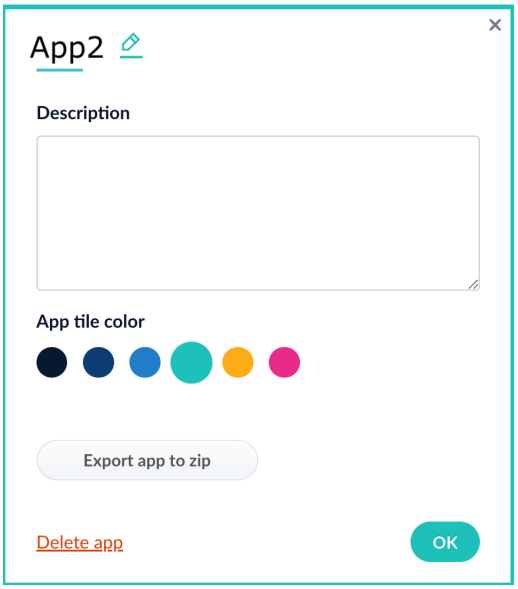
The approach with Object Explorer differs. With Object Explorer, you can add objects from other apps (or that are not linked to any apps) to the currently open app.

Export an app with the Fusion UI

How to export an app with the Fusion UI

1. Navigate to the launcher.
2. Hover over the app you want to export and click the Configure icon:

3. In the app config window, click **Export app to zip**:

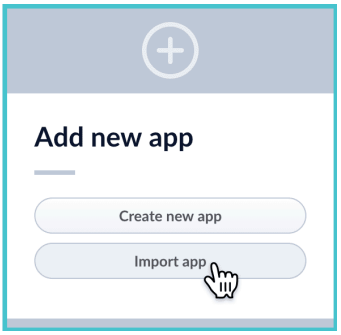


See *import* for information to import the downloaded zip file into other instances of Fusion 5.x.

Import an app with the Fusion UI

How to import an app with the Fusion UI

1. Navigate to the launcher.
2. Click **Import app**.

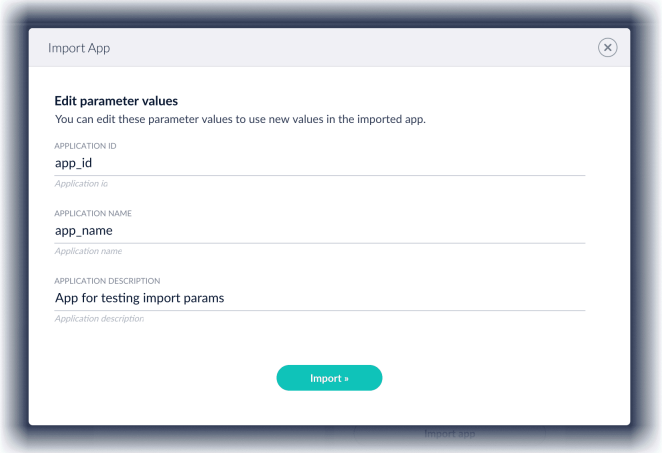


3. Under **Data File**, click **Choose File** and select the zip file containing the app you want to import.

4. If your app has *usernames and passwords* in a separate file, select it under **Variables File**.
   If the Variables File is needed, it *must* be a separate file that is not in a .zip file. It is a .json map of variables to values. The following is an example:

```
{ "secret.dataSources.Inventory_BR_S3_DS.password":"SOMETHING",
"secret.dataSources.LLM_A_BR_S3_DS.password":"FmJSaDE9Tj5REDACTED",
"secret.dataSources.LLM_BR_Inventory_S3_DS.password":"FmJSaDE9Tj5GzIVv
ethAC4Huh",
"secret.dataSources.LLM_BR_Load_S3_DS.password":"FmJSaDE9Tj5GzIVvethAC
4" }
```

5. You can sometimes edit parameter values to use the new values in the imported app. If this is the case, Fusion displays a dialog box that lets you edit the parameter values.



   Make desired changes, and then click **Import**.

Copy an app

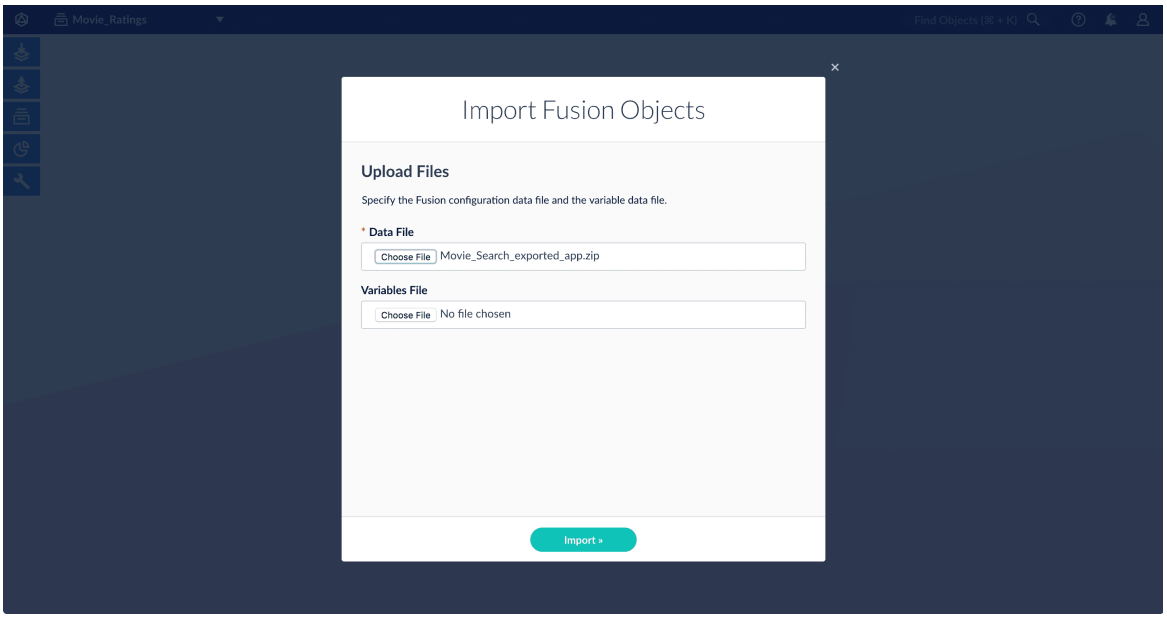
To copy an app from one deployment to a different one, *export the app* on the source deployment, and then *import the app* on the target deployment.

Import objects into an app

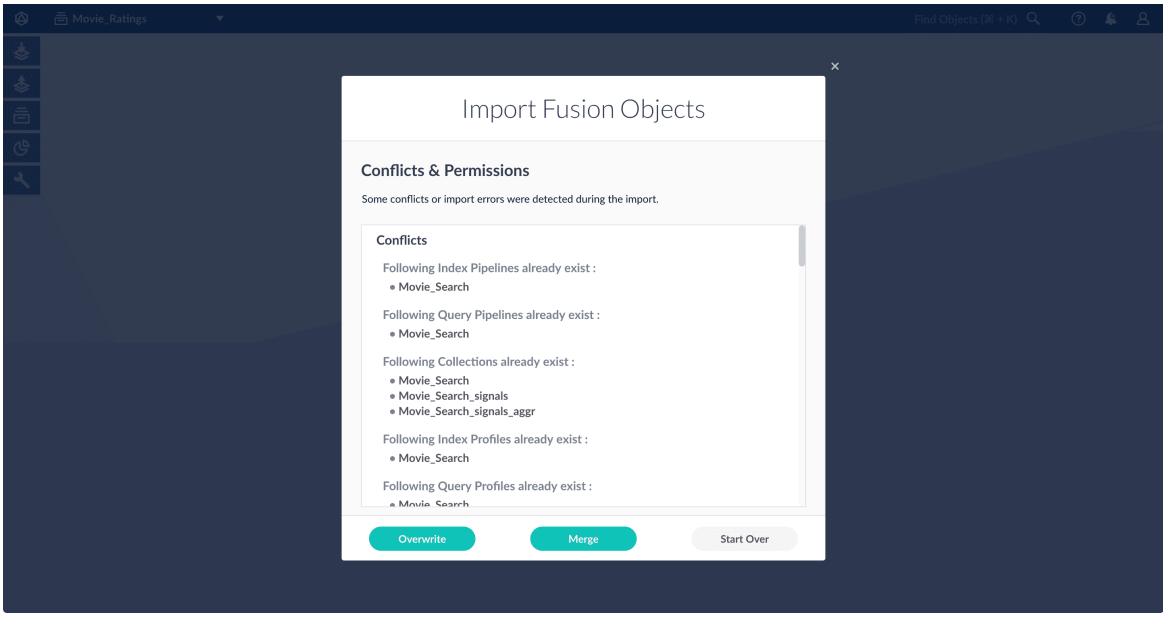
You can import objects into the currently open app.

How to import objects into an open app

1. In the Fusion launcher, click the app into which you want to import objects.
   The Fusion workspace appears.
2. Click **System > Import Fusion Objects**.
   The Import Fusion Objects window opens.

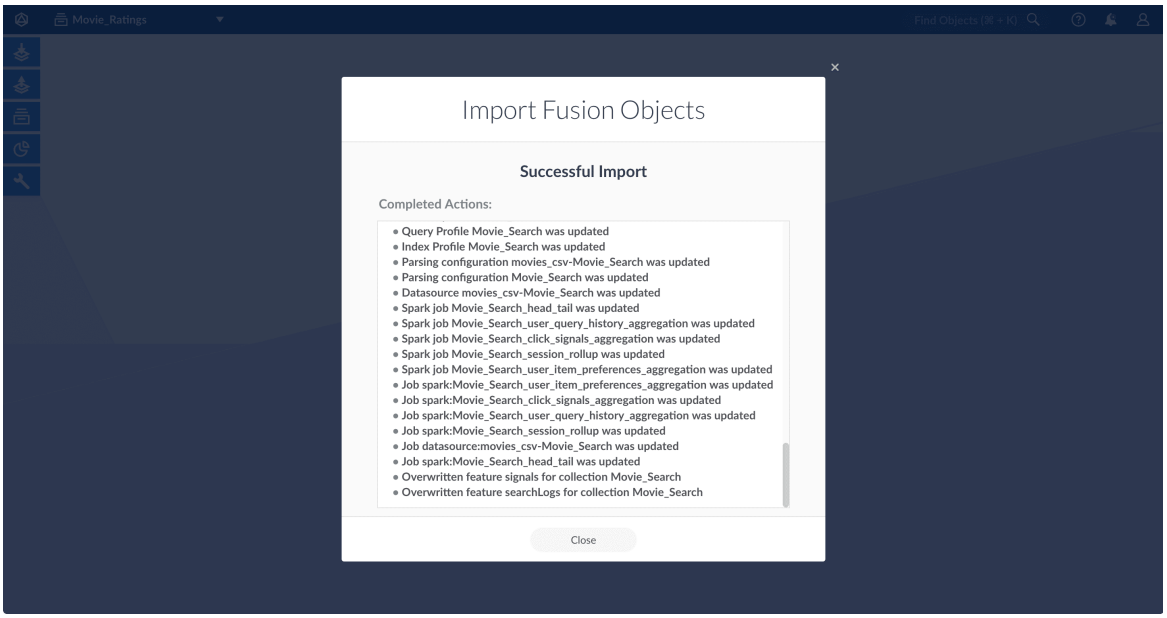


3. Select the data file from your local filesystem.
   If you are importing *usernames and passwords* in a separate file, select it under **Variables File**.
4. Click **Import**.
   If there are conflicts, Fusion prompts you to specify an import policy:



- Click **Overwrite** to overwrite the objects on the target system with the ones in the import file.
- Click **Merge** to skip all conflicting objects and import only the non-conflicting objects.
- Click **Start Over** to abort the import.

Fusion confirms that the import was successful:

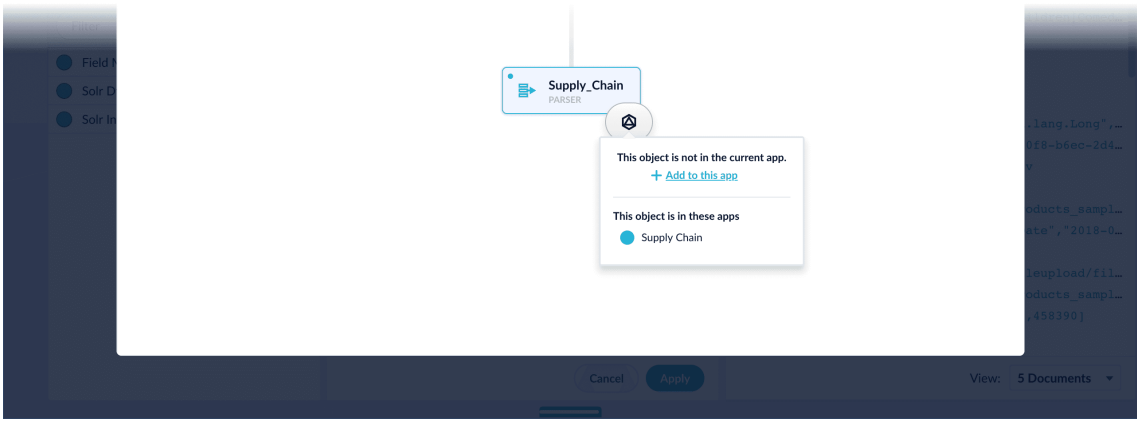


5. Click **Close** to close the Import Fusion Objects window.

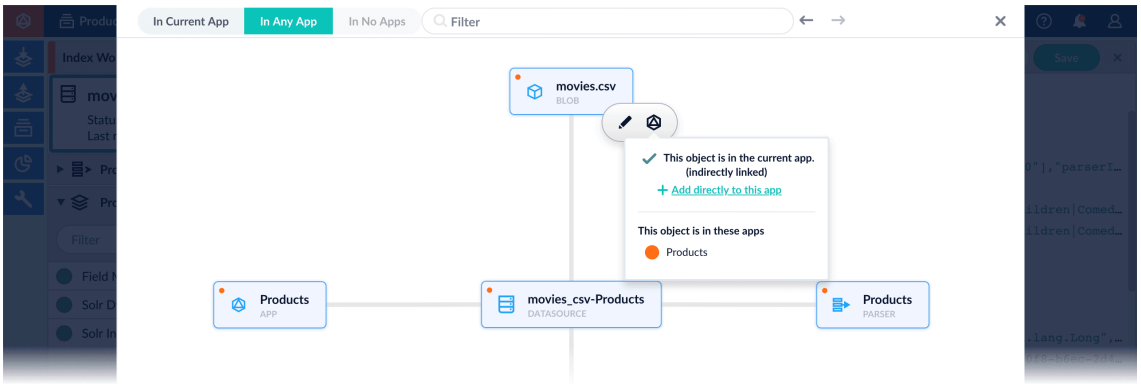
Add an object to an app

You can add objects present in other apps (or in no apps) to the open app. Some objects are linked to other apps. You can also add those directly to an app.

- **Add an object to an app** – While in the Fusion workspace for the app to which you want to add an object, open Object Explorer and click **In Any App**. Search for or browse to the object you want to add. Hover over the object, click the App icon, and then click **Add to this app**.



- **Add an object to an app directly** – In cases when an object is linked to an app, but is not linked directly to the app (it is linked via some dependency), you can add the object to an app directly.
  While in the Fusion workspace for the app to which you want to add an object directly, open Object Explorer and click **In Any App**. Search for or browse to the object you want to add. Hover over the object, click the App icon, and then click **Add to this app directly**.


In Current App
In Any App
In No Apps
Filter
movies.csv
BLOB
This object is in the current app. (indirectly linked)
Add directly to this app
This object is in these apps
Products
Products
APP
movies_csv-Products
DATASOURCE
Products
PARSER

Migrate Milvus to Solr vectors

Lucidworks AI

How-to

> Compatible versions: 5.9
>
> Works with: *Fusion*, *Managed Fusion*

With Milvus being deprecated, these are the steps to move from Milvus to Solr vectors.

1. Add Solr vector schema

The first step to migration is to ensure the Solr schema for the collections has vector field definitions. If the collection was created in Fusion 5.9.5 or later, it will automatically have the vector schemas.

Vector field definitions to add:

```
<!-- Vector search fields --> <dynamicField docValues="false"
indexed="true" multiValued="false" name="*_64v" stored="true"
type="knn_64_vector"/> <dynamicField docValues="false" indexed="true"
multiValued="false" name="*_128v" stored="true"
type="knn_128_vector"/> <dynamicField docValues="false" indexed="true"
multiValued="false" name="*_256v" stored="true"
type="knn_256_vector"/> <dynamicField docValues="false" indexed="true"
multiValued="false" name="*_384v" stored="true"
type="knn_384_vector"/> <dynamicField docValues="false" indexed="true"
multiValued="false" name="*_512v" stored="true"
type="knn_512_vector"/> <dynamicField docValues="false" indexed="true"
multiValued="false" name="*_768v" stored="true"
type="knn_768_vector"/> <dynamicField docValues="false" indexed="true"
multiValued="false" name="*_1024v" stored="true"
type="knn_1024_vector"/> <!-- Field Types to support vector search -->
<fieldType class="solr.DenseVectorField" hnswBeamWidth="200"
hnswMaxConnections="45" knnAlgorithm="hnsw" name="knn_64_vector"
similarityFunction="cosine" vectorDimension="64"/> <fieldType
class="solr.DenseVectorField" hnswBeamWidth="200"
hnswMaxConnections="45" knnAlgorithm="hnsw" name="knn_128_vector"
similarityFunction="cosine" vectorDimension="128"/> <fieldType
class="solr.DenseVectorField" hnswBeamWidth="200"
hnswMaxConnections="45" knnAlgorithm="hnsw" name="knn_256_vector"
similarityFunction="cosine" vectorDimension="256"/> <fieldType
class="solr.DenseVectorField" hnswBeamWidth="200"
hnswMaxConnections="45" knnAlgorithm="hnsw" name="knn_384_vector"
similarityFunction="cosine" vectorDimension="384"/> <fieldType
class="solr.DenseVectorField" hnswBeamWidth="200"
hnswMaxConnections="45" knnAlgorithm="hnsw" name="knn_512_vector"
similarityFunction="cosine" vectorDimension="512"/> <fieldType
class="solr.DenseVectorField" hnswBeamWidth="200"
hnswMaxConnections="45" knnAlgorithm="hnsw" name="knn_768_vector"
similarityFunction="cosine" vectorDimension="768"/> <fieldType
```

```
class="solr.DenseVectorField" hnswBeamWidth="200"
hnswMaxConnections="45" knnAlgorithm="hnsw" name="knn_1024_vector"
similarityFunction="cosine" vectorDimension="1024"/>
```

If in your Create Collections in Milvus job you have a vector dimension not included above, be sure to add it.

2. Add Ray/Seldon Vectorize stage

In the index pipeline, instead of using the Encode into Milvus stage you will need to use Ray/Seldon Vectorize Field (in Fusion 5.9.11 and earlier, this stage is called Seldon Vectorize Field). The Milvus stage field **Encode into Milvus** maps to the Vectorize stage field **Source Field**, **Encoder Output Vector** to **Model Output Vector Field**, and the other fields map to the fields of the same name. The last key field to know is **Destination Field**, which is the Solr vector field name. The field can be named anything as long as the *suffix* matches the *dimension size* of the vector your model returns in your Solr vector field definitions. For example, for a Milvus collection dimension size of `384`, the **Destination Field** should use the suffix `_384v`.

At this point you can start indexing.

On the query side, there is a Ray/Seldon Vectorize Field stage which will be set up mostly the same as on the index side. The only difference is there is no need to put in the vector field name, as it is done in the next step.

3. Add hybrid stage

The final step before you are finally migrated over to Solr vectors is adding a variation of *Fusion Neural Hybrid Query* or *Managed Fusion Hybrid Query stage* (known as Neural Hybrid Query stage in Managed Fusion 5.9.10 and later). In this stage, all you need to do is put in the **Vector Query Field** you have defined in your index pipeline.

Make sure the hybrid stage is placed before any boosting, Apply Rules, or Security Trimming stages.

At this point you can query. After determining the query works you can tune your set up before removing your Milvus collections and pipelines.

Migrate to Graph Security Trimming in Fusion 5.8 or later

How-to

> Compatible versions: 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion Connectors*, *Fusion*

This article describes how to migrate your pre-Fusion 5.8 *Graph Security Trimming query pipeline stage* setup to Fusion 5.8 or later. It applies to deployments using:

- The *SharePoint Optimized V2 connector* v1.1.0 or later
- The *LDAP ACLs V2 connector* v1.4.0 or later to crawl Active Directory in Azure
- The LDAP ACLs V2 connector v1.2.0 or later to crawl Active Directory in LDAP

To set up a new configuration, refer to *Configure Security Trimming for SharePoint Optimized V2*.

Migration

To migrate a deployment that is crawling Active Directory to Fusion 5.8 or later, follow these steps.

Update the datasource configurations

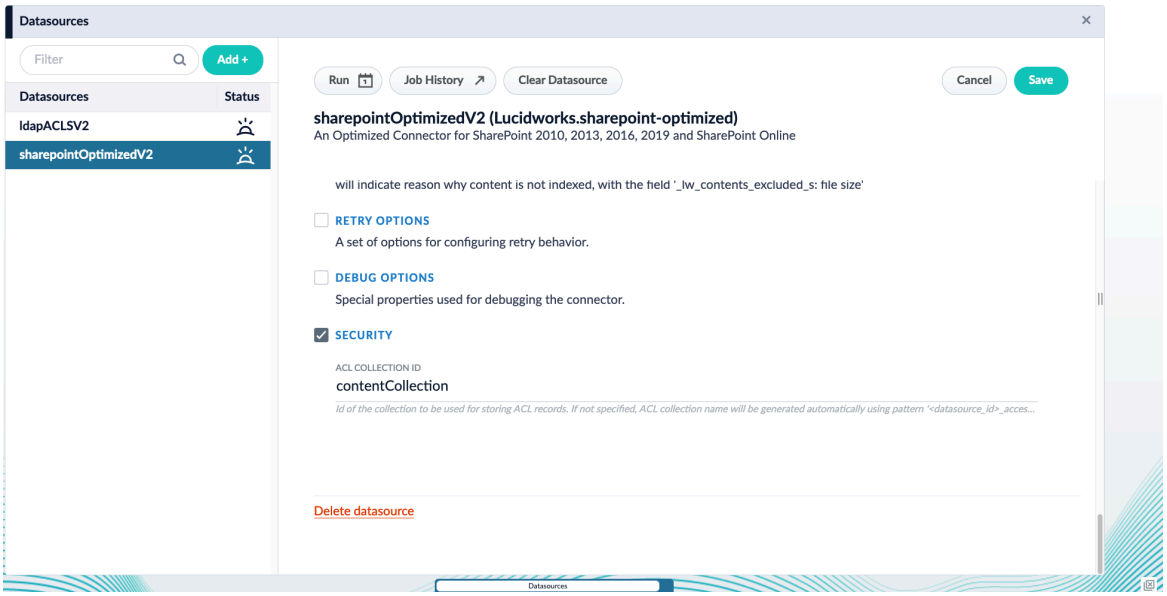
The SharePoint Optimized V2 and LDAP ACLs V2 datasources must index the content documents and ACL documents to the same collection. Ensure both datasources use the same value, `contentCollection`, for the field **ACL Collection ID**.

IF USING SHAREPOINT-OPTIMIZED AND LDAP-ACLS < V2.0.0

Update the **ACL Collection Id** in the datasource configuration.

The SharePoint-Optimized and LDAP-ACLs datasources must index their `content_documents` and `acl_documents` to the same collection. Make sure the property **Security → ACL Collection** in both datasources have the same value. In both datasources, SharePoint-Optimized and LDAP-ACLs, check the property **Security → ACL Collection Id** and make sure it points to the same content-collection.

1. Navigate to **Indexing > Datasources**.
2. Open your SharePoint Optimized V2 or LDAP ACLs V2 datasource.
3. Under **Security**, update the configuration to use `contentCollection` as the **ACL Collection ID**.



Tip

The **Security** checkbox must be checked for this field to appear.

4. Save the configuration.

Repeat this process for all required datasources.

IF USING SHAREPOINT-OPTIMIZED AND LDAP-ACLS >= V2.0.0

Recreate or update the datasources. If only updated, it is not possible to go back to the configuration of a previous plugin version.

By default, the LDAP-ACLs and SharePoint-Optimized V2 datasources will index the `content_documents` and `acl_documents` to the same collection.

1. Navigate to **Indexing > Datasources**.
2. Open your SharePoint Optimized V2 or LDAP ACLs V2 datasource.
3. Under **Graph Security Filtering Configuration**, select **Enable security trimming**.

Repeat this process for all required datasources.

Clear the datasources and perform a full crawl

1. Navigate to **Indexing > Datasources**.
2. Open your SharePoint Optimized V2 or LDAP ACLs V2 datasource.
3. Click the **Clear Datasource** button, and choose yes.
4. Navigate to **Collections > Collections Manager**.
5. Verify that the `job_state` collection is empty.
6. Return to your datasource.
7. Click **Run > Start** to reindex your data.

Repeat this process for all required datasources.

Migrate to Graph Security Trimming in Managed Fusion

How-to

> Compatible versions: 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

This article describes how to migrate your Managed Fusion releases prior to 5.8 *Graph Security Trimming query pipeline stage* setup to Managed Fusion 5.8 or later. It applies to deployments using:

- The *SharePoint Optimized V2 connector* v1.1.0 or later
- The *LDAP ACLs V2 connector* v1.4.0 or later to crawl Active Directory in Azure
- The LDAP ACLs V2 connector v1.2.0 or later to crawl Active Directory in LDAP

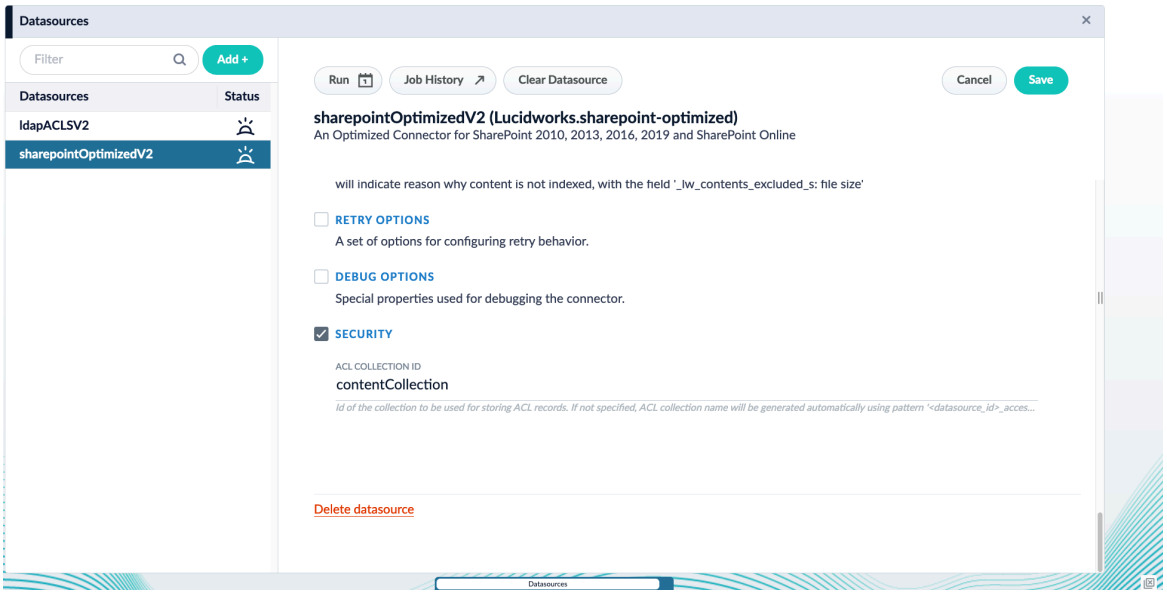
Migration

To migrate a deployment that is crawling Active Directory to Managed Fusion 5.8 or later, follow these steps.

Update the datasource configurations

The SharePoint Optimized V2 and LDAP ACLs V2 datasources must index the content documents and ACL documents to the same collection. Ensure both datasources use the same value, `contentCollection`, for the field **ACL Collection ID**.

1. Sign in to Managed Fusion and click your application.
2. Click **Indexing > Datasources**.
3. Open your SharePoint Optimized V2 or LDAP ACLs V2 datasource.
4. Under Security, update the configuration to use `contentCollection` as the ACL Collection ID.



> **Tip**
>
> The Security checkbox must be checked for this field to display.

5. Save the configuration.

Repeat this process for all required datasources.

Clear and the datasources and perform a full crawl

1. Click **Indexing > Datasources**.
2. Open your SharePoint Optimized V2 or LDAP ACLs V2 datasource.
3. Click **Clear Datasource > yes**.
4. Click **Collections > Collections Manager**.
5. Verify the `job_state` collection is empty.
6. Return to your datasource.
7. Click **Run > Start** to reindex your data.

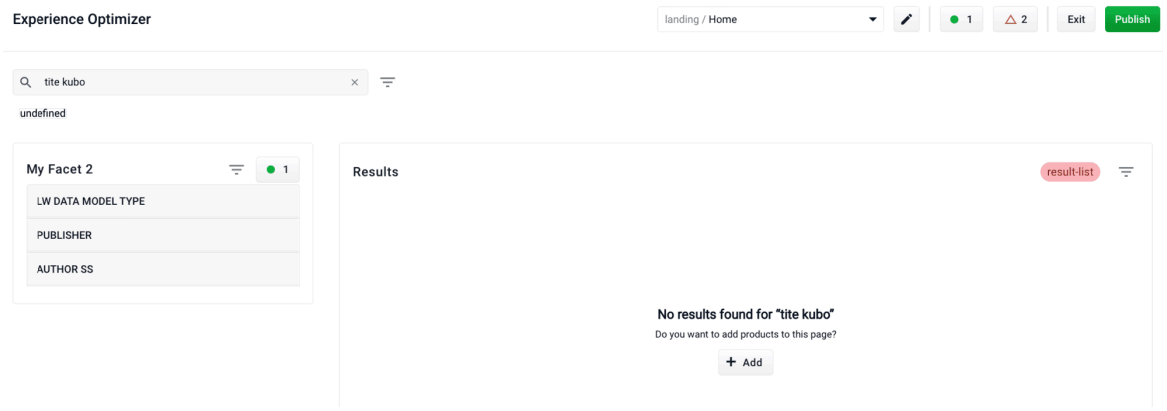
Repeat this process for all required datasources.

Optimizing Zero Results

How-to

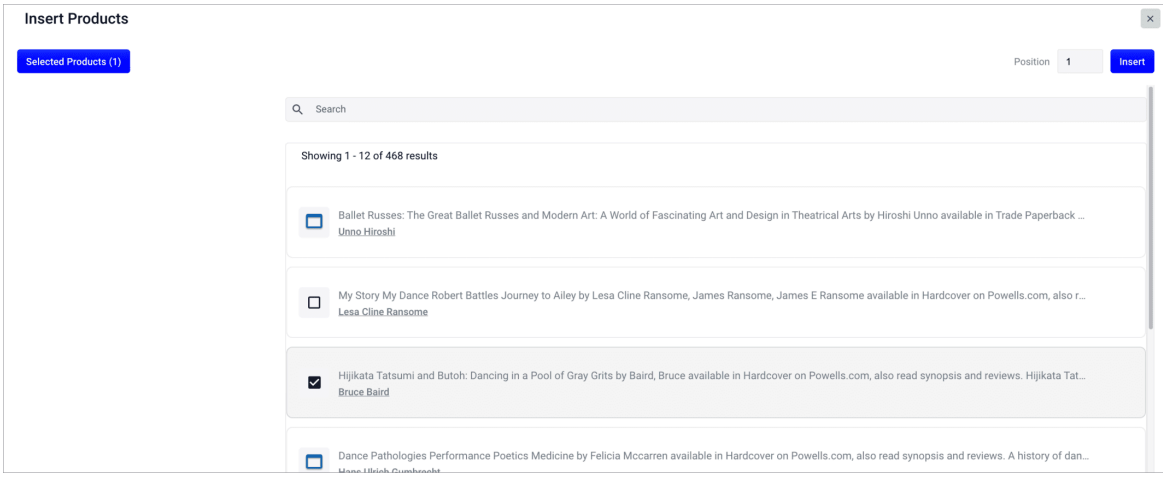
> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion*

1. Enter a search query the generates 0 results:

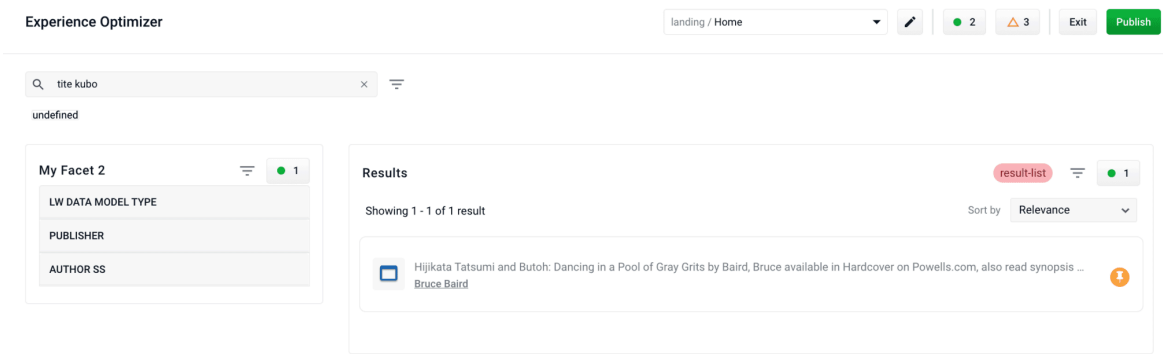


2. Click the **Add** button:

3. Select documents or *document groups* that you want to associate with the 0 results query:



4. Click the **Insert** button. The selected products are now displayed when the search query is used, resolving the 0 results query:

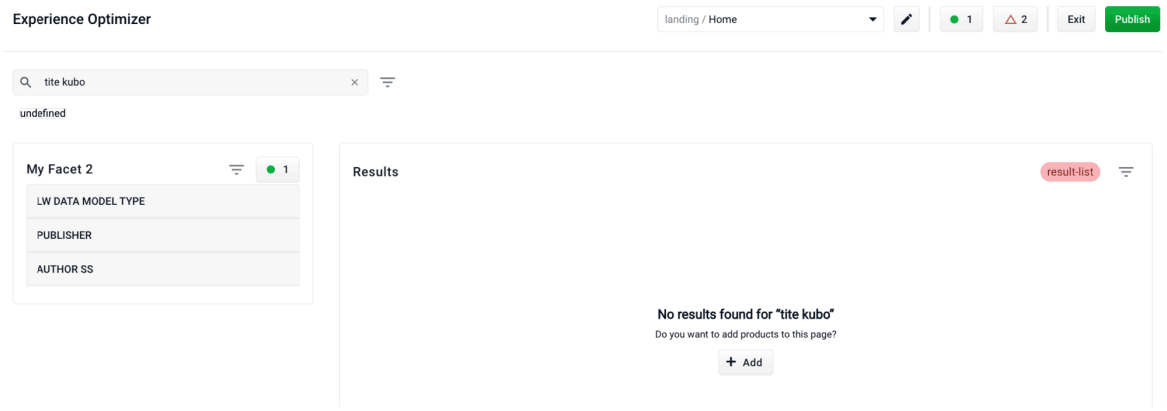


Optimizing Zero Results

How-to

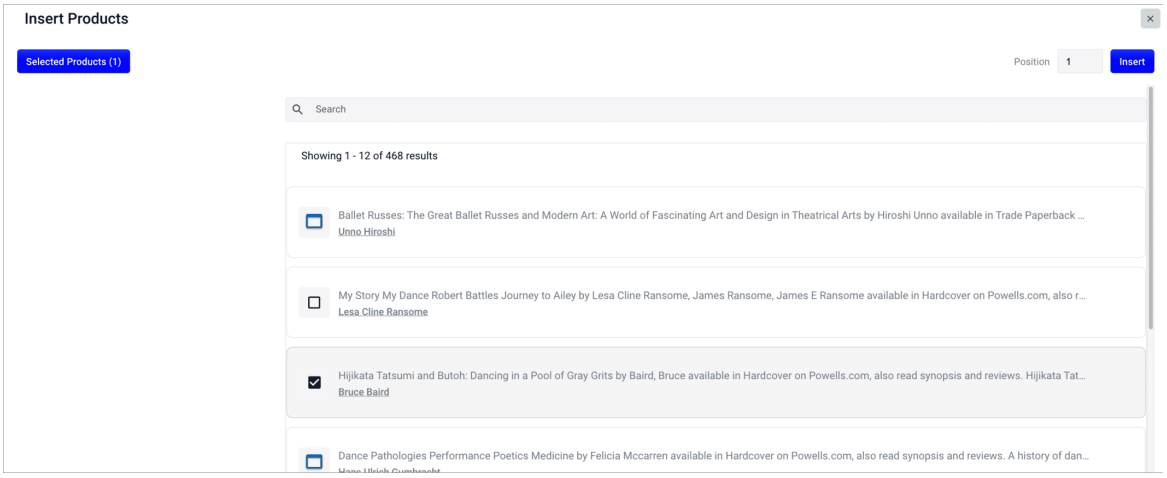
> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

1. Enter a search query the generates 0 results:

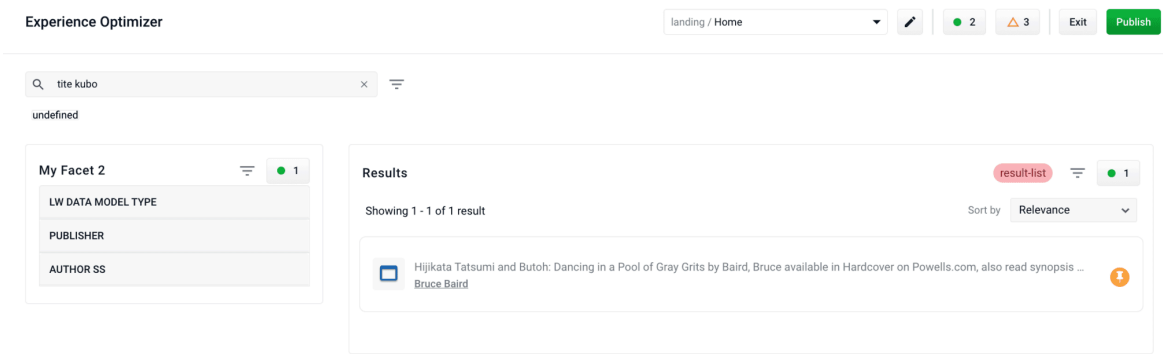


2. Click the **Add** button:

3. Select documents or *document groups* that you want to associate with the 0 results query:



4. Click the **Insert** button. The selected products are now displayed when the search query is used, resolving the 0 results query:



Package a Standalone App

How-to

> Compatible versions: 4.2
>
> Works with: *App Studio*

This topic explains how to build an app from source for *deployment as a standalone app*.

Before building and deploying an app, check the *software requirements*.

Package your app as a standalone app for deployment to a production server. The standalone app embeds an Apache Tomcat servlet container.

Unix or MacOS:

```
./app-studio dist
```

Windows:

```
app-studio.bat dist
```

Standalone app directory

Unix or MacOS:

The standalone app for deployment is in the `app-name/search-app-standalone` directory.

```
search-app-standalone/ README.md app/ build/ config/ keystore.jks lib
search-app-start.bat search-app-stop.bat search-app-stop.ps1 search-
app.sh
```

Windows:

The standalone app for deployment is in the `app-name\search-app-standalone` directory.

```
search-app-standalone\ README.md app\ build\ config\ keystore.jks lib
search-app-start.bat search-app-stop.bat search-app-stop.ps1 search-
app.sh
```

Persist Social Data

How-to

> Compatible versions: 4.2, 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *App Studio*

When using the Appkit Social module, an Appkit application can save social data (comments, bookmarks, and saved search queries) for logged in users. You can store the social data in a Fusion collection, have the application save the data in an in-memory database, the contents of which are lost when the application restarts, or configure Appkit to persist social data to a database that uses disk storage.

Requirements

- Any JDBC-compliant relational database (such as Oracle, MySQL, or SQL Server) or a Fusion deployment
- Network access from the Appkit application to the relational database or to Fusion

The database doesn't have to be empty because Appkit prefixes all of its tables with a `twigkit` identifier. But for simplicity, we recommend using a separate schema, because you probably won't must join Appkit tables with tables in other schemas.

Persist data in a relational database

Perform the tasks in this section to persist Appkit social data in a relational database.

Overview of procedure

To configure a database for persistence on disk, perform these steps, which are explained in more detail below:

1. Declare a Maven POM dependency in `pom.xml` to specify the database for storing the metadata that the Social module collects.
2. Configure a `persistence.xml` file with the appropriate elements, including those for database connections.
3. If necessary, perform other steps as required for specific databases.

The sections that follow describe these steps in more detail for the default *Derby database* and for *other databases*.

Use Derby

DECLARE A POM DEPENDENCY

To persist social data in a disk-based database, declare a POM dependency in the file `pom.xml`, which is located at the root level of a project.

By default, the Appkit Social module uses the lightweight, in-memory Derby database to persist social data, as described in this POM dependency in `pom.xml`:

```
<dependency> <groupId>twigkit</groupId>
<artifactId>twigkit.social.provider.db.derby.memory</artifactId>
<version>${project.parent.version}</version> </dependency>
```

Note

If `pom.xml` does not contain the dependency for Derby, then the app project has not been modified to support the Social module. Perform [those steps] before proceeding.

To persist social metadata in a disk-based Derby database, declare this POM dependency:

```
<dependency> <groupId>twigkit</groupId>
<artifactId>twigkit.social.db.provider.derby</artifactId>
<version>${project.parent.version}</version> </dependency>
```

CREATE AND CONFIGURE PERSISTENCE.XML

You will must create and configure a `persistence.xml` file with the appropriate elements, including those for database connections.

Create the `persistence.xml` file here:
`src/main/resources/META-INF/persistence.xml`

See the *Derby-memory* and *Derby* examples.

SPECIFY THE TYPE OF HIBERNATE ID GENERATOR MAPPINGS

Hibernate ID generator mappings changed in Hibernate 5. Ensure use of the correct mappings as follows:

- **Apps created with versions of Appkit that precede version 4.2.**
  Ensure that both pre-and post-Hibernate 5 IDs are generated using the old mapping. Add this property to the `persistence.xml` file:

  ```
  <property name="hibernate.id.new_generator_mappings"
  value="false" />
  ```

- **Apps created with Appkit 4.2 or later.**
  Apps can use new ID generator mappings. Add this property to the `persistence.xml` file:

  ```
  <property name="hibernate.id.new_generator_mappings"
  value="true" />
  ```

SPECIFY THE DERBY SYSTEM DIRECTORY

You can specify the Derby system directory, which is the directory that will contain the social database, by adding this flag to your Java options:

```
-Dderby.system.home=/my/derby/directory
```

If the system directory that you specify with `derby.system.home` does not exist at startup, Derby creates the directory automatically.

Use a different relational database

You can configure Appkit to persist social data in an on-disk JDBC-compliant relational database other than Derby (such as Oracle, MySQL, or SQL Server).

Other pre-configured versions are also available, including:

```
db.mysql db.oracle db.sqlserver
```

..1. DECLARE THE POM DEPENDENCY

To persist social metadata in a JDBC-compliant relational database, change the POM dependency to:

MySQL:

```
<dependency> <groupId>twigkit</groupId>
<artifactId>twigkit.social.provider.db.mysql</artifactId>
<version>${ppv}</version> </dependency>
```

Oracle:

```
<dependency> <groupId>twigkit</groupId>
<artifactId>twigkit.social.provider.db.oracle</artifactId>
<version>${ppv}</version> </dependency>
```

SQL Server:

```
<dependency> <groupId>twigkit</groupId>
<artifactId>twigkit.social.provider.db.sqlserver</artifactId>
<version>${ppv}</version> </dependency>
```

..2. CREATE AND CONFIGURE PERSISTENCE.XML

You will must create and configure a `persistence.xml` file with the appropriate elements, including those for database connections.

Create the `persistence.xml` file here:
`src/main/resources/META-INF/persistence.xml`

View examples of `persistence.xml` files for *MySQL*, *Oracle*, and *SQL Server*.

..3. SPECIFY THE TYPE OF HIBERNATE ID GENERATOR MAPPINGS

Hibernate ID generator mappings changed in Hibernate 5. Ensure use of the correct mappings as follows:

- **Apps created with versions of Appkit that precede version 4.2.**
  Ensure that both pre-and post-Hibernate 5 IDs are generated using the old mapping. Add this property to the `persistence.xml` file:

```
<property name="hibernate.id.new_generator_mappings"
value="false" />
```

- **Apps created with Appkit 4.2 or later.**
  Apps can use new ID generator mappings. Add this property to the `persistence.xml` file:

```
<property name="hibernate.id.new_generator_mappings"
value="true" />
```

Persist Social module data in Fusion

Perform the tasks in this section to persist Appkit Social module data in Fusion.

1. *Configure Fusion*.
2. (*Only for apps created with prior versions of App Studio or Appkit*) *Upgrade to the latest version of Appkit*.
3. *Enable social features*.

Troubleshooting database configuration

Use these tips to troubleshoot problems with database configuration.

Connection issues

Many databases require the application to manage the connections, refreshing them and maintaining enough open connections to service the load on the database. In some cases the default connection management provided is inadequate for production setups.

If you notice bad performance, connections going stale, and associated errors (usually intermittently) the default connection pooling is probably inadequate for your environment.

To remedy this situation you can use a third-party connection pooling technology. We recommend 'C3P0', which can be configured with these simple steps:

1. Add the dependency for Hibernate c3p0 to the `pom.xml` :

```
<!-- Hibernate c3p0 connection pooling --> <dependency>
<groupId>org.hibernate</groupId> <artifactId>hibernate-
c3p0</artifactId> <version>X.X.X.Final</version> </dependency>
```

The version of Hibernate c3p0 you should use depends on the version of Appkit you are using. To begin with, try Hibernate version 4.1.7.Final (legacy) or 5.2.2.Final.

2. Add the configuration for C3P0 to the **`persistence.xml`** file:

```
<!-- c3p0 connection pool settings --> <property
name="hibernate.connection.provider_class"
value="org.hibernate.connection.C3P0ConnectionProvider" />
<property name="hibernate.c3p0.max_size" value="100" />
<property name="hibernate.c3p0.min_size" value="0" /> <property
name="hibernate.c3p0.acquire_increment" value="1" /> <property
name="hibernate.c3p0.idle_test_period" value="300" /> <property
name="hibernate.c3p0.max_statements" value="0" /> <property
name="hibernate.c3p0.timeout" value="100" />
```

The settings above should be adequate for a standard Appkit application using the Social module, but they can be adjusted as desired.

Related articles

- *Enable Social Features*

Persist Social Data in a Database

How-to

> Compatible versions: 4.2
>
> Works with: *App Studio*

When using the Appkit Social module, an Appkit application can save social data (comments, bookmarks, and saved search queries) for logged in users. By default, the application saves the data in an in-memory database, the contents of which are lost when the application restarts.

You can configure Appkit to persist social data to a database that uses disk storage.

Requirements

- Any JDBC-compliant relational database (such as Oracle, MySQL, or SQL Server) or a Fusion deployment
- Network access from the Appkit application to the relational database or to Fusion

The database doesn't have to be empty because Appkit prefixes all of its tables with a `twigkit` identifier. But for simplicity, we recommend using a separate schema, because you probably won't need to join Appkit tables with tables in other schemas.

Persist data in a relational database

Perform the tasks in this section to persist Appkit social data in a relational database.

Overview of procedure

To configure a database for persistence on disk, perform the following steps, which are explained in more detail below:

1. Declare a Maven POM dependency in `pom.xml` to specify the database for storing the metadata that the Social module collects.
2. Configure a `persistence.xml` file with the appropriate elements, including those for database connections.
3. If necessary, perform other steps as required for specific databases.

The sections that follow describe these steps in more detail for the default *Derby database* and for *other databases*.

Use Derby

DECLARE A POM DEPENDENCY

To persist social data in a disk-based database, declare a POM dependency in the file `pom.xml`, which is located at the root level of a project.

By default, the Appkit Social module uses the lightweight, in-memory Derby database to persist social data, as described in the following POM dependency in `pom.xml`:

For version 4.2 and later:

```
<dependency> <groupId>twigkit</groupId>
<artifactId>twigkit.social.provider.db.derby.memory</artifactId>
<version>${project.parent.version}</version> </dependency>
```

For version 4.1 and earlier:

```
<dependency> <groupId>twigkit</groupId>
<artifactId>twigkit.social.db.derby.memory</artifactId>
<version>${project.parent.version}</version> </dependency>
```

Note

If `pom.xml` does not contain the dependency for Derby, then the app project has not been modified to support the Social module. Perform [those steps] before proceeding.

To persist social metadata in a disk-based Derby database, declare this POM dependency:

For version 4.2 and later:

```
<dependency> <groupId>twigkit</groupId>
<artifactId>twigkit.social.db.provider.derby</artifactId>
<version>${project.parent.version}</version> </dependency>
```

For version 4.1 and earlier:

```
<dependency> <groupId>twigkit</groupId>
<artifactId>twigkit.social.db.derby</artifactId>
<version>${project.parent.version}</version> </dependency>
```

CREATE AND CONFIGURE PERSISTENCE.XML

You will need to create and configure a `persistence.xml` file with the appropriate elements, including those for database connections.

Create the `persistence.xml` file here:
`src/main/resources/META-INF/persistence.xml`

See the *Derby-memory* and *Derby* examples.

SPECIFY THE TYPE OF HIBERNATE ID GENERATOR MAPPINGS

Hibernate ID generator mappings changed in Hibernate 5. Ensure use of the correct mappings as follows:

- **Apps created with versions of Appkit that precede version 4.2.**
  Ensure that both pre-and post-Hibernate 5 IDs are generated using the old mapping. Add this property to the `persistence.xml` file:

```
<property name="hibernate.id.new_generator_mappings"
value="false" />
```

- **Apps created with Appkit 4.2 or later.**
  Apps can use new ID generator mappings. Add this property to the `persistence.xml` file:

```
<property name="hibernate.id.new_generator_mappings"
value="true" />
```

SPECIFY THE DERBY SYSTEM DIRECTORY

You can specify the Derby system directory, which is the directory that will contain the social database, by adding this flag to your Java options:

```
-Dderby.system.home=/my/derby/directory
```

If the system directory that you specify with `derby.system.home` does not exist at startup, Derby creates the directory automatically.

Use a different relational database

You can configure Appkit to persist social data in an on-disk JDBC-compliant relational database other than Derby (such as Oracle, MySQL, or SQL Server).

Other pre-configured versions are also available, including:

```
db.mysql db.oracle db.sqlserver
```

..1. DECLARE THE POM DEPENDENCY

To persist social metadata in a disk-based Derby database, change the POM dependency to:

MySQL:

For version 4.2 and later:

```
<dependency> <groupId>twigkit</groupId>
<artifactId>twigkit.social.db.provider.mysql</artifactId>
<version>${project.parent.version}</version> </dependency>
```

For version 4.1 and earlier:

```
<dependency> <groupId>twigkit</groupId>
<artifactId>twigkit.social.db.mysql</artifactId>
<version>${project.parent.version}</version> </dependency>
```

Oracle:

For version 4.2 and later:

```
<dependency> <groupId>twigkit</groupId>
<artifactId>twigkit.social.db.provider.oracle</artifactId>
<version>${project.parent.version}</version> </dependency>
```

For version 4.1 and earlier:

```
<dependency> <groupId>twigkit</groupId>
<artifactId>twigkit.social.db.oracle</artifactId>
<version>${project.parent.version}</version> </dependency>
```

SQL Server:

For version 4.2 and later:

```
<dependency> <groupId>twigkit</groupId>
<artifactId>twigkit.social.db.provider.sqlserver</artifactId>
<version>${project.parent.version}</version> </dependency>
```

For version 4.1 and earlier:

```
<dependency> <groupId>twigkit</groupId>
<artifactId>twigkit.social.db.sqlserver</artifactId>
<version>${project.parent.version}</version> </dependency>
```

..2. CREATE AND CONFIGURE PERSISTENCE.XML

You will need to create and configure a `persistence.xml` file with the appropriate elements, including those for database connections.

Create the `persistence.xml` file here:
`src/main/resources/META-INF/persistence.xml`

View examples of `persistence.xml` files for *MySQL*, *Oracle*, and *SQL Server*.

..3. SPECIFY THE TYPE OF HIBERNATE ID GENERATOR MAPPINGS

Hibernate ID generator mappings changed in Hibernate 5. Ensure use of the correct mappings as follows:

- **Apps created with versions of Appkit that precede version 4.2.**
  Ensure that both pre-and post-Hibernate 5 IDs are generated using the old mapping. Add this property to the `persistence.xml` file:

  ```
  <property name="hibernate.id.new_generator_mappings"
  value="false" />
  ```

- **Apps created with Appkit 4.2 or later.**
  Apps can use new ID generator mappings. Add this property to the `persistence.xml` file:

  ```
  <property name="hibernate.id.new_generator_mappings"
  value="true" />
  ```

Troubleshooting database configuration

Use these tips to troubleshoot problems with database configuration.

Connection issues

Many databases require the application to manage the connections, refreshing them and maintaining enough open connections to service the load on the database. In some cases the default connection management provided is inadequate for production setups.

If you notice bad performance, connections going stale, and associated errors (usually intermittently) the default connection pooling is probably inadequate for your environment.

To remedy this situation you can use a third-party connection pooling technology. We recommend 'C3P0', which can be configured with the following simple steps:

1. Add the dependency for Hibernate c3p0 to the **pom.xml**:

```
<!-- Hibernate c3p0 connection pooling --> <dependency>
<groupId>org.hibernate</groupId> <artifactId>hibernate-
c3p0</artifactId> <version>X.X.X.Final</version> </dependency>
```

   The version of Hibernate c3p0 you should use depends on the version of Appkit you are using. To begin with try version 4.1.7.Final (legacy) or 5.2.2.Final.

2. Add the configuration for C3P0 to the **persistence.xml**:

```
<!-- c3p0 connection pool settings --> <property
name="hibernate.connection.provider_class"
value="org.hibernate.connection.C3P0ConnectionProvider" />
<property name="hibernate.c3p0.max_size" value="100" />
<property name="hibernate.c3p0.min_size" value="0" /> <property
name="hibernate.c3p0.acquire_increment" value="1" /> <property
name="hibernate.c3p0.idle_test_period" value="300" /> <property
name="hibernate.c3p0.max_statements" value="0" /> <property
name="hibernate.c3p0.timeout" value="100" />
```

The settings above should be adequate for a standard Appkit application using the Social module, but they can be adjusted as desired.

Persist Social Data in Fusion

How-to

> Compatible versions: 4.2
>
> Works with: *App Studio*

When using the Appkit Social module, an Appkit application can save social data (comments, bookmarks, and saved search queries) for logged in users. By default, the application saves the data in an in-memory database, the contents of which are lost when the application restarts.

You can configure Appkit to persist social data to a database that uses disk storage.

Requirements

- A Fusion deployment
- Network access from the Appkit application to Fusion

Persist Social module data in Fusion

Perform the tasks in this section to persist Appkit Social module data in Fusion.

1. *Configure Fusion.*
2. (*Only for apps created with prior versions of App Studio or Appkit*) *Upgrade to the latest version of Appkit.*
3. *Enable social features.*

Pin a Product in Experience Optimizer

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

You can pin a document so that it occupies a fixed position in the search results when your customers run a particular query. This can be helpful if you have bestsellers or popular items that you want customers to see first, for example.

Important

Boosting a document increases the document's position in the results, but takes other factors into consideration while doing so. As a result, it does not guarantee the document will occupy a specific position in the list. Pinning a document, however, places the document at a specified position regardless of other factors.

When you pin a document, either a new rule is created automatically or an existing pin rule is modified.

Use Actions

1. Hover your cursor over a document. A menu overlay will appear.
2. Hover your cursor over the **Actions** button. A dropdown menu will appear.
3. Select the **Pin** button:

Drag and Pin a document

1. Hover your cursor over a document. A menu overlay will appear.
2. Click-and-hold the drag bar.
3. Move the document to the desired location.
4. Release the click.

Insert and Pin a document

You can search for and insert an alternative document from your catalog into a list of search results, and pin it at a fixed position. This is helpful if:

- You want to include a document that is not included in the results list generated by the query.
- The results list is long and you need to search for a particular document to locate and pin it.
- You want to pin multiple products simultaneously.

When you pin a document, either a new rule is created automatically or an existing pin rule is modified.

1. Hover your cursor between two products. A **+** button will appear:

2. Click the **+** button.
3. Locate and select one or more products you want to pin.
4. Click **Insert**.

Pin a Product in Experience Optimizer

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion*

You can pin a document so that it occupies a fixed position in the search results when your customers run a particular query. This can be helpful if you have bestsellers or popular items that you want customers to see first, for example.

Important

Boosting a document increases the document's position in the results, but takes other factors into consideration while doing so. As a result, it does not guarantee the document will occupy a specific position in the list. Pinning a document, however, places the document at a specified position regardless of other factors.

When you pin a document, either a new rule is created automatically or an existing pin rule is modified.

Use Actions

1. Hover your cursor over a document. A menu overlay will appear.
2. Hover your cursor over the **Actions** button. A dropdown menu will appear.
3. Select the **Pin** button:

Drag and Pin a document

1. Hover your cursor over a document. A menu overlay will appear.
2. Click-and-hold the drag bar.
3. Move the document to the desired location.
4. Release the click.

Insert and Pin a document

You can search for and insert an alternative document from your catalog into a list of search results, and pin it at a fixed position. This is helpful if:

- You want to include a document that is not included in the results list generated by the query.
- The results list is long and you need to search for a particular document to locate and pin it.
- You want to pin multiple products simultaneously.

When you pin a document, either a new rule is created automatically or an existing pin rule is modified.

1. Hover your cursor between two products. A **+** button will appear:

2. Click the **+** button.
3. Locate and select one or more products you want to pin.
4. Click **Insert**.

Pin a product in Predictive Merchandiser

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

You can pin a product to occupy a fixed position in the search results when your customers run a particular query. This can be helpful, for example, if you have bestsellers or popular items that you want customers to see first.

Important

Boosting a product increases the product's position in the results, but takes other factors into consideration while doing so. As a result, it does not guarantee the product will occupy a specific position in the list. Pinning a product, however, places the product at a specified position regardless of other factors.

When you pin a product, either a new rule is created automatically or an existing pin rule is modified.

Use actions

1. Hold the pointer over a product.
2. In the menu that displays, hold the pointer over the **Actions** button.
3. From the menu, click Pin Product .

Drag and pin a product

1. Hold the pointer over a product.
2. In the menu that displays, click-and-hold the drag bar.
3. Move the product to the desired location.
4. Release the pointer.

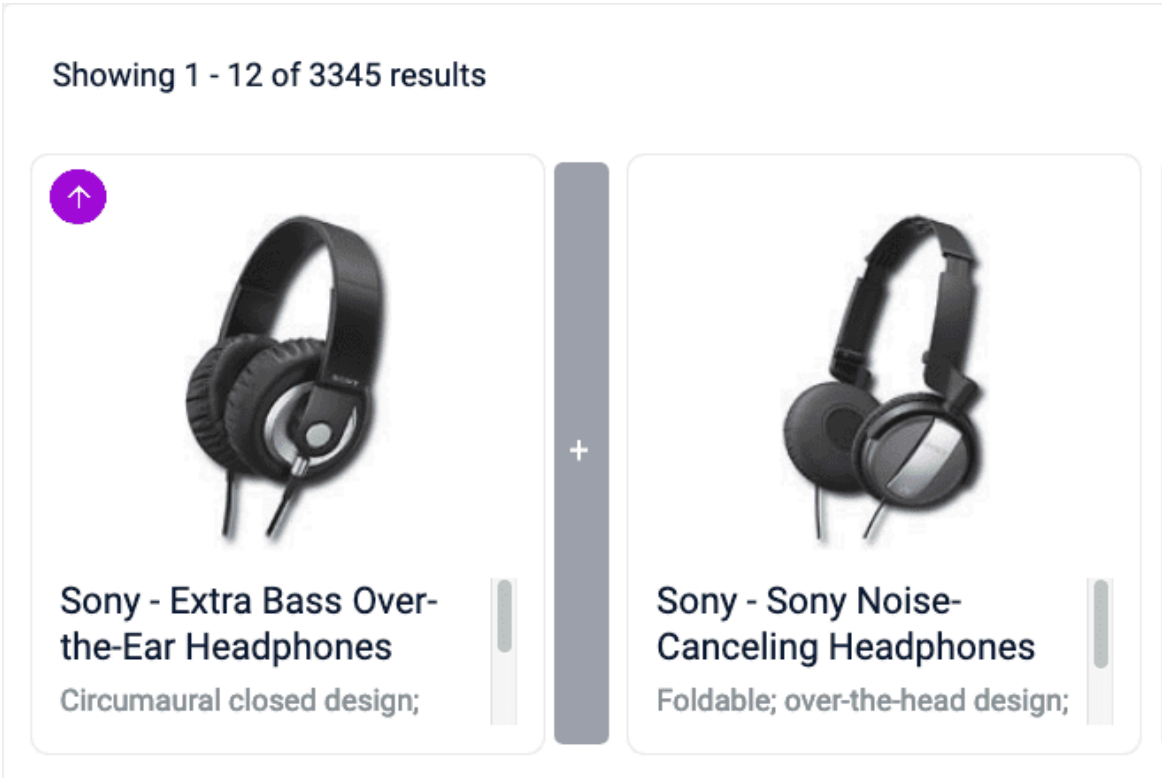
Insert and pin a product

You can search for and insert an alternative product from your catalog into a list of search results, and pin it at a fixed position. This is helpful if:

- You want to include a product that is not included in the results list generated by the query.
- The results list is long and you need to search for a particular product to locate and pin it.
- You want to pin multiple products simultaneously.

When you pin a product, either a new rule is created automatically or an existing pin rule is modified.

1. Hold the pointer between two products. A **+** button will appear:



2. Click the **+** button.
3. Locate and select one or more products you want to pin.
4. Click **Insert**

Choose pin position

1. Hold the pointer over a product.
2. Select the value of the Position parameter.
3. Enter the desired replacement value, or adjust the value using the arrows.

Pin a Product in Predictive Merchandiser

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion*

You can pin a product so that it occupies a fixed position in the search results when your customers run a particular query. This can be helpful if you have bestsellers or popular items that you want customers to see first, for example.

Important

Boosting a product increases the product's position in the results, but takes other factors into consideration while doing so. As a result, it does not guarantee the product will occupy a specific position in the list. Pinning a product, however, places the product at a specified position regardless of other factors.

When you pin a product, either a new rule is created automatically or an existing pin rule is modified.

Use Actions

1. Hover your cursor over a product. A menu overlay will appear.
2. Hover your cursor over the **Actions** button. A dropdown menu will appear.
3. Select the **Pin Product** button:

Drag and Pin a Product

1. Hover your cursor over a product. A menu overlay will appear.
2. Click-and-hold the drag bar.
3. Move the product to the desired location.
4. Release the click.

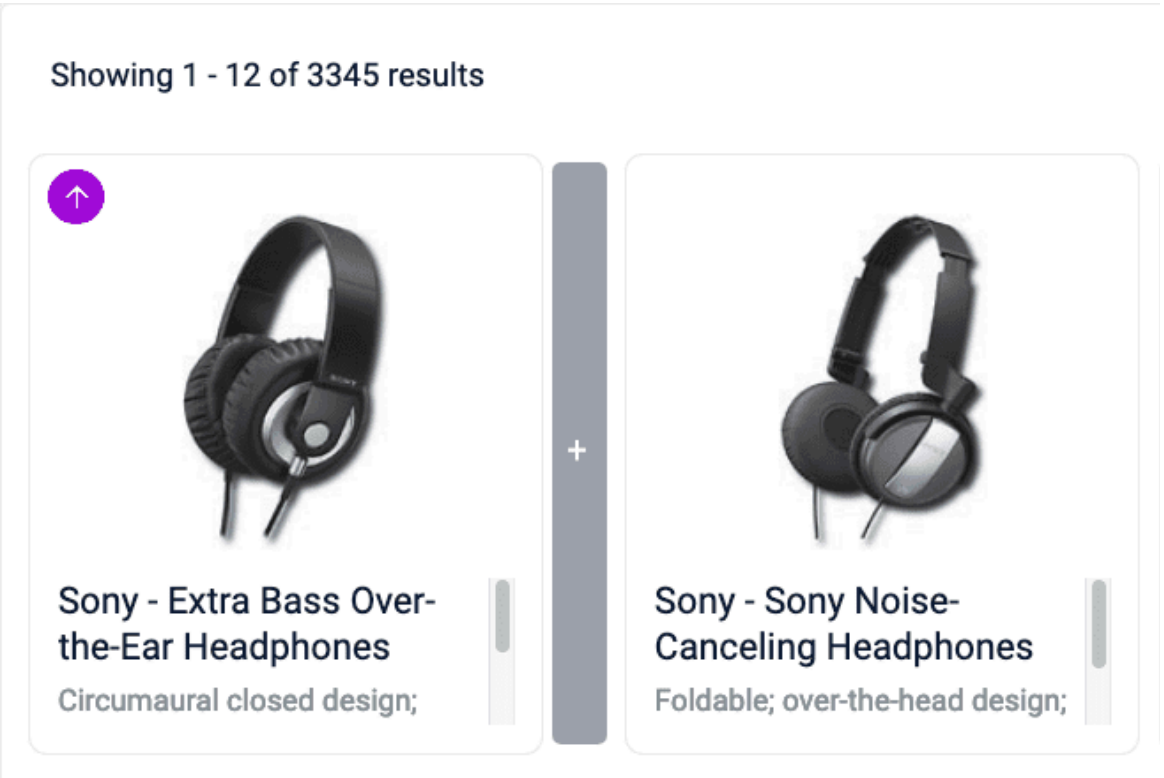
Insert and Pin a Product

You can search for and insert an alternative product from your catalog into a list of search results, and pin it at a fixed position. This is helpful if:

- You want to include a product that is not included in the results list generated by the query.
- The results list is long and you need to search for a particular product to locate and pin it.
- You want to pin multiple products simultaneously.

When you pin a product, either a new rule is created automatically or an existing pin rule is modified.

1. Hover your cursor between two products. A **+** button will appear:



2. Click the **+** button.
3. Locate and select one or more products you want to pin.
4. Click the **Insert** button:

Choose Pin Position

1. Hover your cursor over a product. A menu overlay will appear.
2. Select the value of the **Position** parameter.
3. Enter the desired replacement value, or adjust the value using the arrows.

Platform Workflow Processors

How-to

> Compatible versions: 4.2, 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *App Studio*

To process and change data on the fly in Appkit, processors are usually added either to modify *queries* or *responses*.

This is where the workflow platform comes in. It lets you configure processors through configuration files.

How to configure a workflow chain (post Appkit v3.0)

There are two steps to follow when configuring a workflow chain:

1. Add the `workflow` attribute to your normal platform configuration. This will point to the workflow processors you want to run on the platform.
2. Add configuration for each workflow processor.

Platform configuration

The normal platform configuration (for example, `platforms/gsa.conf`) would have a `workflow` attribute, paired with a comma-separated list of workflow processors to run on the platform. For example:

```
workflow:
processors.response.capitaliseFieldValuesProcessor,processors.response
.highlightFieldValuesProcessor,processors.response.copyFieldsProcesso
r
```

Each processor listed contains the name and folder path to a given processor `.conf` file within the `conf` directory - this can be organized however you want.

Processor configuration

Each of the processors would be configured with key-value pairs. For example:

CAPITALISEFIELDVALUESPROCESSOR.CONF

This, like all configuration files is made up of key-value pairs. You reference a **name** with the classpath of a given processor (either Appkit's or your own). You then pass in the attributes of this processor as key-value pairs - in this case **fields**.

```
name:
twigkit.search.processors.response.CapitaliseFieldValuesProcessor
fields: fieldA,fieldB
```

Calling the workflow platform

Since the `workflow` attribute is part of the normal platform configuration, there is no must explicitly call the workflow platform. Simply call the normal platform:

```
<search:platform var="platform" conf="platforms.gsa">
</search:platform>
```

or

```
<media:excel conf="platforms.gsa" query="query" fetch="10000"
fields="title,foo" header="myTitle,myFoo">Export</media:excel>
```

Calling the normal platform would automatically run the configured processors at runtime.

Example configuration

This example would likely be configured using this setup, with each of these broken down below this.

```
/resources/conf/platforms /gsa gsa.conf (your normal GSA conf)
/workflow workflow.conf gsaWithProcessing.conf
/resources/conf/processors /response
/capitaliseFieldValuesProcessor.conf
/highlightFieldValuesProcessor.conf /copyFieldsProcessor.conf
/processors.conf
```

Platform configuration

Each of the processors would be configured with key value pairs, like:

WORKFLOW.CONF

This file indicates to Appkit that this is a workflow configuration, and to trigger processing of that as such.

```
name: twigkit.platform.Workflow
```

GSAWITHPROCESSING.CONF

This is the part that tied together a standard existing platform configuration and one or more processor configurations.

```
platform: platforms.solr.excel workflow:
processors.response.capitaliseFieldValuesProcessor,processors.response
.highlightFieldValuesProcessor,processors.response.copyFieldsProcesso
r
```

The processor confs mentioned are the name and folder path to a given processor `.conf` file within the `conf` directory; this can be organized however you want.

Processor configuration

Each of the processors would be configured with key value pairs, much like you would pass in attributes to a normal processor tag within your JSP for example:

CAPITALISEFIELDVALUESPROCESSOR.CONF

This file, like all `.conf` files, is made up of key value pairs. You reference a **name** with the classpath of a given processor (either Appkit's or your own). You then pass in the attributes of this processor as key value pairs - in this case as **fields**.

```
name:
twigkit.search.processors.response.CapitaliseFieldValuesProcessor
fields: fieldA,fieldB
```

Calling the workflow platform

You just call the workflow platform as you would a normally configured platform for example:

```
<search:platform var="platform"
conf="platforms.workflow.gsaWithProcessing"></search:platform>
```

Or:

```
<media:excel conf="platforms.workflow.gsaWithProcessing"
query="${query}" fetch="10000" fields="title,foo"
header="myTitle,myFoo">Export</media:excel>
```

The configured processors get run at runtime.

Process sitemaps on websites

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

The Web V1 connector retrieves data from a Web site using HTTP and starting from a specified URL.

Crawling sitemaps is supported. Add the URL(s) of the sitemap to the `f.sitemapURLs` property (**Sitemap URLs** in the UI) and all of the URLs found in a sitemap are added to the list of URLs to crawl. Sitemap indexes (that is, a sitemap that points to other sitemaps) are also supported. The URLs found through each sitemap are added to the list of URLs to crawl.

To configure your datasource to crawl *only* the sitemap file, add the sitemap URL to both the `startLinks` property (because that is a required property for a datasource) and also to the `f.sitemapsURL` property so it is properly treated as a sitemap by the connector when it starts.

Incremental crawls and deleted documents

Crawling a website with a `sitemap.xml` adds the links in the sitemap to *crawlDB*.

To remove deleted documents during reindexing, you must remove the links from the `sitemap.xml`.

The *Connectors Datasources API* contains information on how to work with crawlDB and removing items.

Diagnostic logging

Diagnostic logging allows the HTML contents to print out in the logs for each step of the crawl, so you can use that to see what the browser is doing. This can be useful because you can see the browser's state if any part of a crawl fails.

To enable diagnostic logging:

1. Open the Web connector.
2. Toggle **Advanced**. You should now see new fields become active next to **Cookie Spec**.
3. Select **Diagnostic logging**.

Process sitemaps on websites

How-to

> Compatible versions: 4.2, 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion Connectors*, *Fusion Server*, *Fusion*

The Web V1 connector retrieves data from a Web site using HTTP and starting from a specified URL.

Crawling sitemaps is supported. Simply add the URL(s) of the sitemap to the `f.sitemapURLs` property (**Sitemap URLs** in the UI) and all of the URLs found in a sitemap are added to the list of URLs to crawl. Sitemap indexes (that is, a sitemap that points to other sitemaps) are also supported. The URLs found through each sitemap are added to the list of URLs to crawl.

To configure your datasource to crawl *only* the sitemap file, add the sitemap URL to both the `startLinks` property (because that is a required property for a datasource) and also to the `f.sitemapsURL` property so it is properly treated as a sitemap by the connector when it starts.

Incremental crawls and deleted documents

Crawling a website with a `sitemap.xml` adds the links in the sitemap to *crawlDB*.

To remove deleted documents during reindexing, you must remove the links from the `sitemap.xml`.

The *Connectors Datasources API* contains information on how to work with crawlDB and removing items.

Diagnostic logging

Diagnostic logging allows the HTML contents to print out in the logs for each step of the crawl, so you can use that to see what the browser is doing. This can be useful because you can see the browser's state if any part of a crawl fails.

To enable diagnostic logging:

1. Open the Web connector.
2. Toggle **Advanced**. You should now see new fields become active next to **Cookie Spec**.
3. Select **Diagnostic mode**.

Protect Against CSRF Attacks

How-to

> Compatible versions: 4.2, 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *App Studio*

To help protect against Cross-Site Request Forgery (CSRF) attacks on social web services, App Studio can ensure that all requests are tokenized. That is, each request is provided with a randomized Appkit request token.

To enable protection against CSRF in your application:

1. If it does not already exist, create a new file named `csrf.conf` and place it in `src/main/resources/conf/security`.
2. In that file, set the property `enabled` to `true`. This tokenizes the requests.
3. By default, the time-to-live of the Appkit request token is 60 minutes. You can change this by setting the property `duration` to some other value; for example, `duration: 30`. Responses received without a request token or with a request token has expired are rejected.

This configuration enables protection against CSRF attacks and sets the time-to-live of the request token to 30 minutes:

```
enabled: true duration: 30
```

Protect Against CSRF Attacks

How-to

> Compatible versions: 4.2
>
> Works with: *App Studio*

To help protect against Cross-Site Request Forgery (CSRF) attacks on social web services, App Studio can ensure that all requests are tokenized. That is, each request is provided with a randomized Appkit request token.

How to enable protection against CSRF in your application:

1. If it does not already exist, create a new file named `csrf.conf` and place it under `src/main/resources/conf/security`.
2. In that file, set the property `enabled` to `true`. This tokenizes the requests.
3. By default, the time-to-live of the Appkit request token is 60 minutes. You can change this by setting the property `duration` to some other value; for example, `duration: 30`. Responses received without a request token or with a request token has expired are rejected.

This configuration enables protection against CSRF attacks and sets the time-to-live of the request token to 30 minutes:

```
enabled: true duration: 30
```

Query rewrite jobs post-processing cleanup

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

Warning

The Synonym Detection job uses the output of the Misspelling Detection job and Phrase Extraction job.

Therefore, post processing *must* occur in the order specified in this topic for the *Synonym detection job cleanup*, *Phrase extraction job cleanup*, and *Misspelling detection job cleanup* procedures.

The *Head-Tail Analysis job cleanup* can occur in any order.

Synonym detection job cleanup

Use this job to remove low confidence synonyms.

Prerequisites

Complete this:

- **AFTER** the Misspelling Detection and Phrase Extraction jobs have successfully completed.
- **BEFORE** removing low confidence synonym suggestions generated in the *post processing* phrase extraction cleanup and misspelling detection cleanup procedures detailed later in this topic.

Remove low confidence synonym suggestions

Use either a *Synonym cleanup method 1 - API call* or the *Synonym cleanup method 2 - Managed Fusion Admin UI* to remove low confidence synonym suggestions.

SYNONYM CLEANUP METHOD 1 - API CALL

1. Open the `delete_lowConf_synonyms.json` file.

```
{ "type" : "rest-call", "id" :
"DC_Large_QR_DELETE_LOW_CONFIDENCE_SYNONYMS", "callParams" : {
"uri" : "solr://DC_Large_query_rewrite_staging/update", "method"
: "post", "queryParams" : { "wt" : "json" }, "headers" : { },
"entity" : "<root><delete><query>type:synonym AND confidence:[0
TO 0.0005]</query></delete><commit/></root>" }, "type" : "rest-
call", "type" : "rest-call" }
```

Note

REQUEST ENTITY specifies the threshold for low confidence synonyms. Edit the upper range from 0.0005 to increase or decrease the threshold based on your data.

2. Enter `<your query_rewrite_staging collection name/update>` in the **uri** field. An example URI value for an app called `DC_Large` would be `DC_Large_query_rewrite_staging/update`.
3. Change the `id` field if applicable.
4. Specify the **upper confidence level** in the **entity** field.

Note

The **entity** field specifies the threshold for low confidence synonyms. Edit the upper range to increase or decrease the threshold based on your data.

SYNONYM CLEANUP METHOD 2 - MANAGED FUSION ADMIN UI

1. Log in to Managed Fusion and select **Collections > Jobs**.
2. Select **Add+ > Custom and Other Jobs > REST Call**.
3. Enter **delete-low-confidence-synonyms** in the **ID** field.
4. Enter `<your query_rewrite_staging collection name/update>` in the **ENDPOINT URI** field. An example URI value for an app called `DC_Large` would be `DC_Large_query_rewrite_staging/update`.
5. Enter **POST** in the **CALL METHOD** field.
6. In the **QUERY PARAMETERS** section, select **+** to add a property.
7. Enter **wt** in the **Property Name** field.
8. Enter **json** in the **Property Value** field.
9. In the **REQUEST PROTOCOL HEADERS** section, select **+** to add a property.
10. Enter the following as a **REQUEST ENTITY (AS STRING)**

```
<root><delete><query>type:synonym AND confidence: [0 TO 0.0005]</query></delete>
<commit/></root>
```

Note

REQUEST ENTITY specifies the threshold for low confidence synonyms. Edit the upper range from 0.0005 to increase or decrease the threshold based on your data.

Delete all synonym suggestions

To delete *all* of the synonym suggestions, enter the following in the **REQUEST ENTITY** section:

```
<root><delete><query>type:synonym</query></delete><commit/></root>
```

Note

This entry may be helpful when tuning the synonym detection job and testing different configuration parameters.

Phrase extraction job cleanup

Use this job to remove low confidence phrase suggestions.

Prerequisites

Complete this:

- **AFTER** you complete *Synonym detection job cleanup*

Remove low confidence phrase suggestions

Use either a *Phrase cleanup method 1 - API call* or the *Phrase cleanup method 2 - Managed Fusion Admin UI* to remove low confidence phrase suggestions.

PHRASE CLEANUP METHOD 1 - API CALL

1. Open the `delete_lowConf_phrases.json` file.

```
{ "type" : "rest-call", "id" :
"DC_Large_QR_DELETE_LOW_CONFIDENCE_PHRASES", "callParams" : { "uri" :
"solr://DC_Large_query_rewrite_staging/update", "method" : "post",
"queryParams" : { "wt" : "json" }, "headers" : { }, "entity" : "
<root><delete><query>type:phrase AND confidence:[0 TO <INSERT VALUE
HERE>]</query></delete><commit/></root>" }, "type" : "rest-call",
"type" : "rest-call" }
```

2. Enter `<your query_rewrite_staging collection name/update>` in the **uri** field. An example URI value for an app called `DC_Large` would be `DC_Large_query_rewrite_staging/update`.
3. Change the **id** field if applicable.
4. Specify the **upper confidence level** in the **entity** field.

> **Note**
>
> The **entity** field specifies the threshold for low confidence phrases. Edit the upper range to increase or decrease the threshold based on your data.

PHRASE CLEANUP METHOD 2 - MANAGED FUSION ADMIN UI

1. Log in to Managed Fusion and select **Collections > Jobs**.
2. Select **Add+ > Custom and Other Jobs > REST Call**.
3. Enter **remove-low-confidence-phrases** in the **ID** field.
4. Enter `<your query_rewrite_staging collection name/update>` in the **ENDPOINT URI** field. An example URI value for an app called `DC_Large` would be `DC_Large_query_rewrite_staging/update`.
5. Enter **POST** in the **CALL METHOD** field.
6. In the **QUERY PARAMETERS** section, select **+** to add a property.
7. Enter **wt** in the **Property Name** field.
8. Enter **json** in the **Property Value** field.
9. In the **REQUEST PROTOCOL HEADERS** section, select **+** to add a property.
10. Enter the following as a **REQUEST ENTITY (AS STRING)**

```
<root><delete><query>type:phrase AND confidence: [0 TO <insert value>]</query>
</delete><commit/></root>
```

> **Note**
>
> **REQUEST ENTITY** specifies the threshold for low confidence phrases. Edit the upper range to increase or decrease the threshold based on your data.

Delete all phrase suggestions

To delete *all* of the phrase suggestions, enter the following in the **REQUEST ENTITY** section:

```
<root><delete><query>type:phrase</query></delete><commit/></root>
```

Note

This entry may be helpful when tuning the phrase extraction job and testing different configuration parameters.

Misspelling detection job cleanup

Use this job to remove low confidence spellings (also referred to as misspellings).

Prerequisites

Complete this:

- **AFTER** you complete *Synonym detection job cleanup* and *Phrase extraction job cleanup*

Remove misspelling suggestions

Use either a *Misspelling cleanup method 1 - API call* or the *Misspelling cleanup method 2 - Managed Fusion Admin UI* to remove misspelling suggestions.

MISSPELLING CLEANUP METHOD 1 - API CALL

1. Open the `delete_lowConf_misspellings.json` file.

```
{ "type" : "rest-call", "id" :
"DC_Large_QR_DELETE_LOW_CONFIDENCE_MISSPELLINGS", "callParams" : {
"uri" : "solr://DC_Large_query_rewrite_staging", "method" : "post",
"queryParams" : { "wt" : "json" }, "headers" : { }, "entity" : "<root>
<delete><query>type:spell AND confidence:[0 TO 0.5]</query></delete>
<commit/></root>" }, "type" : "rest-call", "type" : "rest-call" }
```

2. Enter `<your query_rewrite_staging collection name/update>` in the **uri** field. An example URI value for an app called `DC_Large` would be `DC_Large_query_rewrite_staging/update`.
3. Change the **id** field if applicable.
4. Specify the **upper confidence level** in the **entity** field.

Note

The **entity** field specifies the threshold for low confidence spellings. Edit the upper range to increase or decrease the threshold based on your data.

MISSPELLING CLEANUP METHOD 2 - MANAGED FUSION ADMIN UI

1. Log in to Managed Fusion and select **Collections > Jobs**.
2. Select **Add+ > Custom and Other Jobs > REST Call**.
3. Enter **remove-low-confidence-spellings** in the **ID** field.

4. Enter `<your query_rewrite_staging collection name/update>` in the **ENDPOINT URI** field. An example URI value for an app called `DC_Large` would be `DC_Large_query_rewrite_staging/update`.
5. Enter **POST** in the **CALL METHOD** field.
6. In the **QUERY PARAMETERS** section, select **+** to add a property.
7. Enter **wt** in the **Property Name** field.
8. Enter **json** in the **Property Value** field.
9. In the **REQUEST PROTOCOL HEADERS** section, select **+** to add a property.
10. Enter the following as a **REQUEST ENTITY (AS STRING)**

```
<root><delete><query>type:spell AND confidence: [0 TO 0.5]</query></delete>
<commit/></root>
```

Note

REQUEST ENTITY specifies the threshold for low confidence spellings. Edit the upper range from 0.5 to increase or decrease the threshold based on your data.

Delete all misspelling suggestions

To delete *all* of the misspelling suggestions, enter the following in the **REQUEST ENTITY** section:

```
<root><delete><query>type:spell</query></delete><commit/></root>
```

Note

This entry may be helpful when tuning the misspelling detection job and testing different configuration parameters.

Head-tail analysis job cleanup

The head-tail analysis job puts tail queries into one of multiple reason categories. For example, a tail query that includes a number might be assigned to the 'numbers' reason category. If the output in a particular category is not useful, you can remove it from the results. The examples in this section remove the numbers category.

Prerequisites

The head-tail analysis job cleanup does *not* have to occur in a specific order.

Remove head-tail analysis query suggestions

Use either a *Head-tail analysis cleanup method 1 - API call* or the *Head-tail analysis cleanup method 2 - Managed Fusion Admin UI* to remove query category suggestions.

HEAD-TAIL ANALYSIS CLEANUP METHOD 1 - API CALL

1. Open the `delete_lowConf_headTail.json` file.

```
{ "type" : "rest-call", "id" : "DC_Large_QR_HEAD_TAIL_CLEANUP",
"callParams" : { "uri" :
"solr://DC_Large_query_rewrite_staging/update", "method" : "post",
"queryParams" : { "wt" : "json" }, "headers" : { }, "entity" : "<root>
<delete><query>reason_code_s:(\"number\" \"number spelling\" \"number
rare-term\" \"question number other-specific\" \"number others\"
\"number other-specific\" \"number other-extra\" \"product number
other-specific\" \"product number other-extra\" \"product number
spelling\" \"product number others\" \"product number rare-term\"
\"product question number\" \"product number re-wording\" \"question
number other-extra\" \"number re-wording\")</query></delete><commit/>
</root>" }, "type" : "rest-call", "type" : "rest-call" }
```

2. Enter `<your query_rewrite_staging collection name/update>` in the **uri** field. An example URI value for an app called `DC_Large` would be `DC_Large_query_rewrite_staging/update`.
3. Change the **id** field if applicable.

HEAD-TAIL ANALYSIS CLEANUP METHOD 2 - MANAGED FUSION ADMIN UI

1. Log in to Managed Fusion and select **Collections > Jobs**.
2. Select **Add+ > Custom and Other Jobs > REST Call**.
3. Enter **remove-low-confidence-head-tail** in the **ID** field.
4. Enter `<your query_rewrite_staging collection name/update>` in the **ENDPOINT URI** field. An example URI value for an app called `DC_Large` would be `DC_Large_query_rewrite_staging/update`.
5. Enter **POST** in the **CALL METHOD** field.
6. In the **QUERY PARAMETERS** section, select **+** to add a property.
7. Enter **wt** in the **Property Name** field.
8. Enter **json** in the **Property Value** field.
9. In the **REQUEST PROTOCOL HEADERS** section, select **+** to add a property.
10. Enter the following as a **REQUEST ENTITY (AS STRING)**

```
<root><delete><query>reason_code_s:("number" "number spelling"
"number rare-term" "question number other-specific" "number
others" "number other-specific" "number other-extra" "product
number other-specific" "product number other-extra" "product
number spelling" "product number others" "product number rare-
term" "product question number" "product number re-wording"
"question number other-extra" "number re-wording")</query>
</delete><commit/></root>
```

Delete all head-tail suggestions

To delete *all* of the head-tail suggestions, enter the following in the **REQUEST ENTITY** section:

```
<root><delete><query>type:tail</query></delete><commit/></root>
```

Note

This entry may be helpful when tuning the head-tail job and testing different configuration parameters.

Query Rewrite Jobs Post-processing Cleanup

How-to

> Compatible versions: 4.2, 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion AI*, *Fusion Server*, *Fusion*

Warning

The Synonym Detection job uses the output of the Misspelling Detection job and Phrase Extraction job.

Therefore, post processing *must* occur in the order specified in this topic for the *Synonym detection job cleanup*, *Phrase extraction job cleanup*, and *Misspelling detection job cleanup* procedures.

The *Head-Tail Analysis job cleanup* can occur in any order.

Synonym detection job cleanup

Use this job to remove low confidence synonyms.

Prerequisites

Complete this:

- **AFTER** the Misspelling Detection and Phrase Extraction jobs have successfully completed.
- **BEFORE** removing low confidence synonym suggestions generated in the *post processing* phrase extraction cleanup and misspelling detection cleanup procedures detailed later in this topic.

Remove low confidence synonym suggestions

Use either a *Synonym cleanup method 1 - API call* or the *Synonym cleanup method 2 - Fusion Admin UI* to remove low confidence synonym suggestions.

SYNONYM CLEANUP METHOD 1 - API CALL

1. Open the `delete_lowConf_synonyms.json` file.

```
{ "type" : "rest-call", "id" :
"DC_Large_QR_DELETE_LOW_CONFIDENCE_SYNONYMS", "callParams" : { "uri" :
"solr://DC_Large_query_rewrite_staging/update", "method" : "post",
"queryParams" : { "wt" : "json" }, "headers" : { }, "entity" : "<root>
<delete><query>type:synonym AND confidence:[0 TO 0.0005]</query>
</delete><commit/></root>" }, "type" : "rest-call", "type" : "rest-
call" }
```

+ NOTE: **REQUEST ENTITY** specifies the threshold for low confidence synonyms. Edit the upper range from 0.0005 to increase or decrease the threshold based on your data.

2. Enter `<your query_rewrite_staging collection name/update>` in the **uri** field. An example URI value for an app called `DC_Large` would be `DC_Large_query_rewrite_staging/update`.
3. Change the `id` field if applicable.
4. Specify the **upper confidence level** in the **entity** field.

Note

The **entity** field specifies the threshold for low confidence synonyms. Edit the upper range to increase or decrease the threshold based on your data.

SYNONYM CLEANUP METHOD 2 - FUSION ADMIN UI

1. Log in to Fusion and select **Collections > Jobs**.
2. Select **Add+ > Custom and Other Jobs > REST Call**.
3. Enter **delete-low-confidence-synonyms** in the **ID** field.
4. Enter `<your query_rewrite_staging collection name/update>` in the **ENDPOINT URI** field. An example URI value for an app called `DC_Large` would be `DC_Large_query_rewrite_staging/update`.
5. Enter **POST** in the **CALL METHOD** field.
6. In the **QUERY PARAMETERS** section, select **+** to add a property.
7. Enter **wt** in the **Property Name** field.
8. Enter **json** in the **Property Value** field.
9. In the **REQUEST PROTOCOL HEADERS** section, select **+** to add a property.
10. Enter the following as a **REQUEST ENTITY (AS STRING)**

```
<root><delete><query>type:synonym AND confidence: [0 TO 0.0005]</query></delete>
<commit/></root>
```

Note

REQUEST ENTITY specifies the threshold for low confidence synonyms. Edit the upper range from 0.0005 to increase or decrease the threshold based on your data.

Delete all synonym suggestions

To delete *all* of the synonym suggestions, enter the following in the **REQUEST ENTITY** section:

```
<root><delete><query>type:synonym</query></delete><commit/></root>
```

Note

This entry may be helpful when tuning the synonym detection job and testing different configuration parameters.

Phrase extraction job cleanup

Use this job to remove low confidence phrase suggestions.

Prerequisites

Complete this:

- **AFTER** you complete *Synonym detection job cleanup*

Remove low confidence phrase suggestions

Use either a *Phrase cleanup method 1 - API call* or the *Phrase cleanup method 2 - Fusion Admin UI* to remove low confidence phrase suggestions.

PHRASE CLEANUP METHOD 1 - API CALL

1. Open the `delete_lowConf_phrases.json` file.

```
{ "type" : "rest-call", "id" :
"DC_Large_QR_DELETE_LOW_CONFIDENCE_PHRASES", "callParams" : { "uri" :
"solr://DC_Large_query_rewrite_staging/update", "method" : "post",
"queryParams" : { "wt" : "json" }, "headers" : { }, "entity" : "
<root><delete><query>type:phrase AND confidence:[0 TO <INSERT VALUE
HERE>]</query></delete><commit/></root>" }, "type" : "rest-call",
"type" : "rest-call" }
```

2. Enter `<your query_rewrite_staging collection name/update>` in the **uri** field. An example URI value for an app called `DC_Large` would be `DC_Large_query_rewrite_staging/update`.
3. Change the **id** field if applicable.
4. Specify the **upper confidence level** in the **entity** field.

> **Note**
>
> The **entity** field specifies the threshold for low confidence phrases. Edit the upper range to increase or decrease the threshold based on your data.

PHRASE CLEANUP METHOD 2 - FUSION ADMIN UI

1. Log in to Fusion and select **Collections > Jobs**.
2. Select **Add+ > Custom and Other Jobs > REST Call**.
3. Enter **remove-low-confidence-phrases** in the **ID** field.
4. Enter `<your query_rewrite_staging collection name/update>` in the **ENDPOINT URI** field. An example URI value for an app called `DC_Large` would be `DC_Large_query_rewrite_staging/update`.
5. Enter **POST** in the **CALL METHOD** field.
6. In the **QUERY PARAMETERS** section, select **+** to add a property.
7. Enter **wt** in the **Property Name** field.
8. Enter **json** in the **Property Value** field.
9. In the **REQUEST PROTOCOL HEADERS** section, select **+** to add a property.
10. Enter the following as a **REQUEST ENTITY (AS STRING)**

```
<root><delete><query>type:phrase AND confidence: [0 TO <insert value>]</query>
</delete><commit/></root>
```

> **Note**
>
> **REQUEST ENTITY** specifies the threshold for low confidence phrases. Edit the upper range to increase or decrease the threshold based on your data.

Delete all phrase suggestions

To delete *all* of the phrase suggestions, enter the following in the **REQUEST ENTITY** section:

```
<root><delete><query>type:phrase</query></delete><commit/></root>
```

Note

This entry may be helpful when tuning the phrase extraction job and testing different configuration parameters.

Misspelling detection job cleanup

Use this job to remove low confidence spellings (also referred to as misspellings).

Prerequisites

Complete this:

- **AFTER** you complete *Synonym detection job cleanup* and *Phrase extraction job cleanup*

Remove misspelling suggestions

Use either a *Misspelling cleanup method 1 - API call* or the *Misspelling cleanup method 2 - Fusion Admin UI* to remove misspelling suggestions.

MISSPELLING CLEANUP METHOD 1 - API CALL

1. Open the `delete_lowConf_misspellings.json` file.

```
{ "type" : "rest-call", "id" :
"DC_Large_QR_DELETE_LOW_CONFIDENCE_MISSPELLINGS", "callParams" : {
"uri" : "solr://DC_Large_query_rewrite_staging", "method" : "post",
"queryParams" : { "wt" : "json" }, "headers" : { }, "entity" : "<root>
<delete><query>type:spell AND confidence:[0 TO 0.5]</query></delete>
<commit/></root>" }, "type" : "rest-call", "type" : "rest-call" }
```

2. Enter `<your query_rewrite_staging collection name/update>` in the **uri** field. An example URI value for an app called `DC_Large` would be `DC_Large_query_rewrite_staging/update`.
3. Change the **id** field if applicable.
4. Specify the **upper confidence level** in the **entity** field.

Note

The **entity** field specifies the threshold for low confidence spellings. Edit the upper range to increase or decrease the threshold based on your data.

MISSPELLING CLEANUP METHOD 2 - FUSION ADMIN UI

1. Log in to Fusion and select **Collections > Jobs**.
2. Select **Add+ > Custom and Other Jobs > REST Call**.
3. Enter **remove-low-confidence-spellings** in the **ID** field.

4. Enter `<your query_rewrite_staging collection name/update>` in the **ENDPOINT URI** field. An example URI value for an app called `DC_Large` would be `DC_Large_query_rewrite_staging/update`.
5. Enter **POST** in the **CALL METHOD** field.
6. In the **QUERY PARAMETERS** section, select **+** to add a property.
7. Enter **wt** in the **Property Name** field.
8. Enter **json** in the **Property Value** field.
9. In the **REQUEST PROTOCOL HEADERS** section, select **+** to add a property.
10. Enter the following as a **REQUEST ENTITY (AS STRING)**

```
<root><delete><query>type:spell AND confidence: [0 TO 0.5]</query></delete>
<commit/></root>
```

> **Note**
>
> **REQUEST ENTITY** specifies the threshold for low confidence spellings. Edit the upper range from 0.5 to increase or decrease the threshold based on your data.

Delete all misspelling suggestions

To delete *all* of the misspelling suggestions, enter the following in the **REQUEST ENTITY** section:

```
<root><delete><query>type:spell</query></delete><commit/></root>
```

> **Note**
>
> This entry may be helpful when tuning the misspelling detection job and testing different configuration parameters.

Head-tail analysis job cleanup

The head-tail analysis job puts tail queries into one of multiple reason categories. For example, a tail query that includes a number might be assigned to the 'numbers' reason category. If the output in a particular category is not useful, you can remove it from the results. The examples in this section remove the numbers category.

Prerequisites

The head-tail analysis job cleanup does *not* have to occur in a specific order.

Remove head-tail analysis query suggestions

Use either a *Head-tail analysis cleanup method 1 - API call* or the *Head-tail analysis cleanup method 2 - Fusion Admin UI* to remove query category suggestions.

HEAD-TAIL ANALYSIS CLEANUP METHOD 1 - API CALL

1. Open the `delete_lowConf_headTail.json` file.

```
{ "type" : "rest-call", "id" : "DC_Large_QR_HEAD_TAIL_CLEANUP",
"callParams" : { "uri" :
"solr://DC_Large_query_rewrite_staging/update", "method" : "post",
"queryParams" : { "wt" : "json" }, "headers" : { }, "entity" : "<root>
<delete><query>reason_code_s:(\"number\" \"number spelling\" \"number
rare-term\" \"question number other-specific\" \"number others\"
\"number other-specific\" \"number other-extra\" \"product number
other-specific\" \"product number other-extra\" \"product number
spelling\" \"product number others\" \"product number rare-term\"
\"product question number\" \"product number re-wording\" \"question
number other-extra\" \"number re-wording\")</query></delete><commit/>
</root>" }, "type" : "rest-call", "type" : "rest-call" }
```

2. Enter `<your query_rewrite_staging collection name/update>` in the **uri** field. An example URI value for an app called `DC_Large` would be `DC_Large_query_rewrite_staging/update`.
3. Change the **id** field if applicable.

HEAD-TAIL ANALYSIS CLEANUP METHOD 2 - FUSION ADMIN UI

1. Log in to Fusion and select **Collections > Jobs**.
2. Select **Add+ > Custom and Other Jobs > REST Call**.
3. Enter **remove-low-confidence-head-tail** in the **ID** field.
4. Enter `<your query_rewrite_staging collection name/update>` in the **ENDPOINT URI** field. An example URI value for an app called `DC_Large` would be `DC_Large_query_rewrite_staging/update`.
5. Enter **POST** in the **CALL METHOD** field.
6. In the **QUERY PARAMETERS** section, select **+** to add a property.
7. Enter **wt** in the **Property Name** field.
8. Enter **json** in the **Property Value** field.
9. In the **REQUEST PROTOCOL HEADERS** section, select **+** to add a property.
10. Enter the following as a **REQUEST ENTITY (AS STRING)**

```
<root><delete><query>reason_code_s:("number" "number spelling"
"number rare-term" "question number other-specific" "number
others" "number other-specific" "number other-extra" "product
number other-specific" "product number other-extra" "product
number spelling" "product number others" "product number rare-
term" "product question number" "product number re-wording"
"question number other-extra" "number re-wording")</query>
</delete><commit/></root>
```

Delete all head-tail suggestions

To delete *all* of the head-tail suggestions, enter the following in the **REQUEST ENTITY** section:

```
<root><delete><query>type:tail</query></delete><commit/></root>
```

Note

This entry may be helpful when tuning the head-tail job and testing different configuration parameters.

Quickstart

How-to

> Compatible versions: 4.2, 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion Server*, *Fusion*

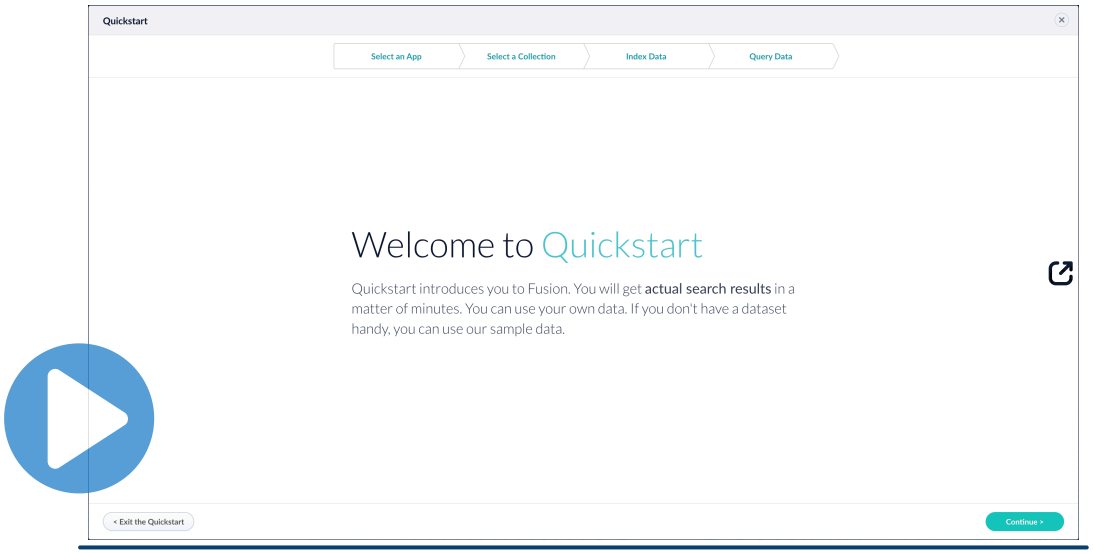
The Quickstart is a wizard that lets you explore some of Fusion's core capabilities:

- Creating or selecting an app
- Selecting a collection (where data is stored)
- Indexing data
- Searching

LucidAcademy

Lucidworks offers free training to help you get started.

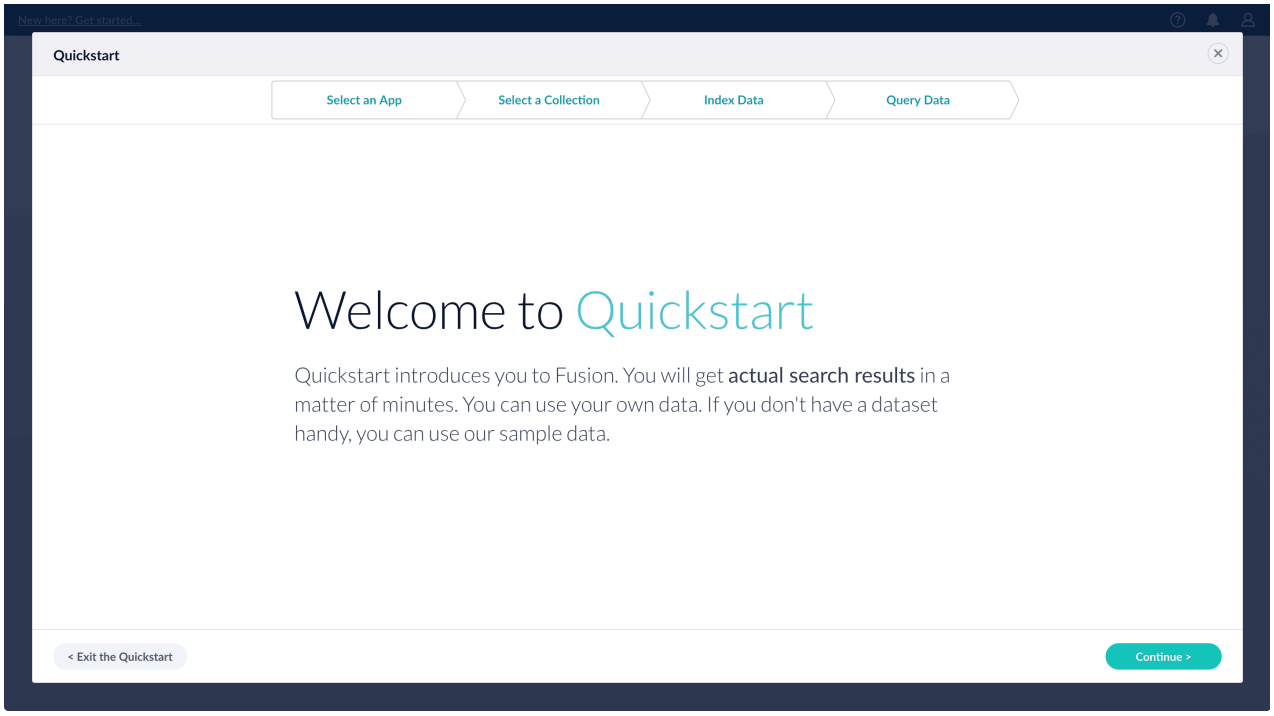
The **Course** for **Using The Quickstart Wizard** focuses on using the wizard to practice using the key functions of Fusion:



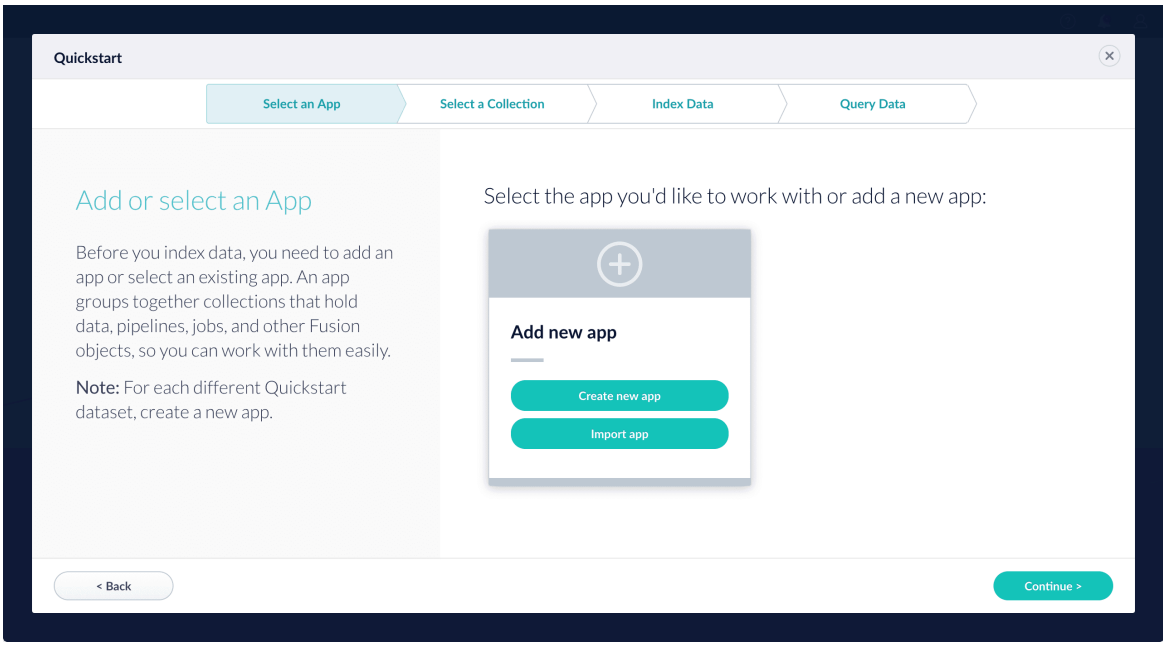
Visit the LucidAcademy to see the full training catalog.

Step through the Quickstart wizard

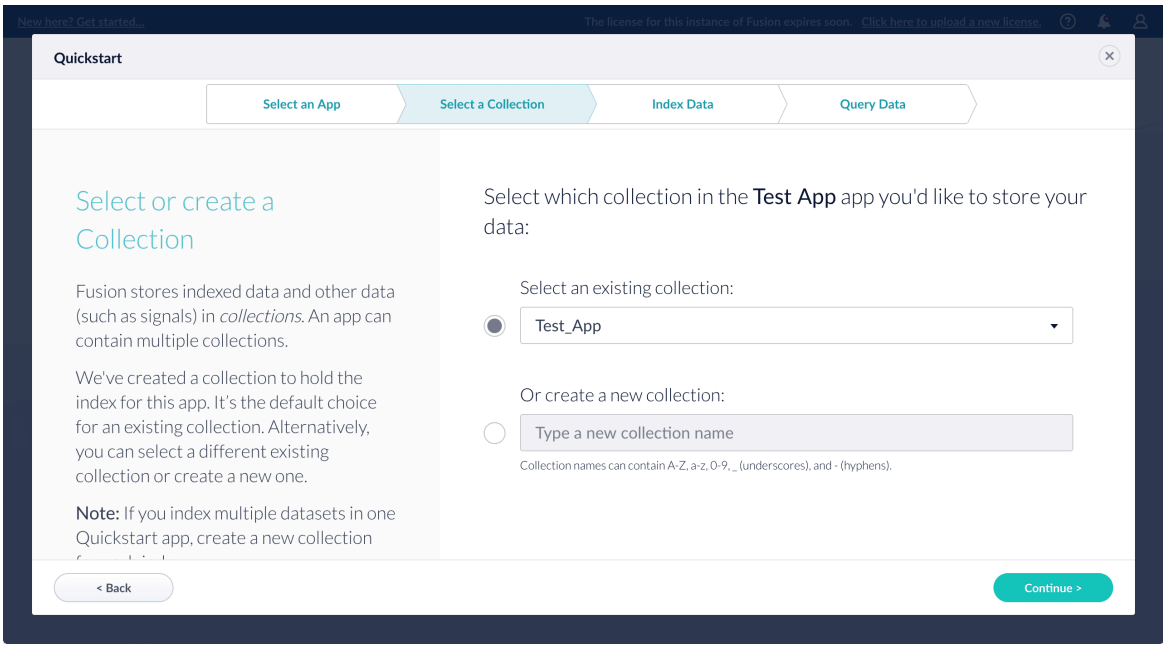
If it is not already open, open the Quickstart wizard. In the Fusion launcher, click **New here? Get started...**.



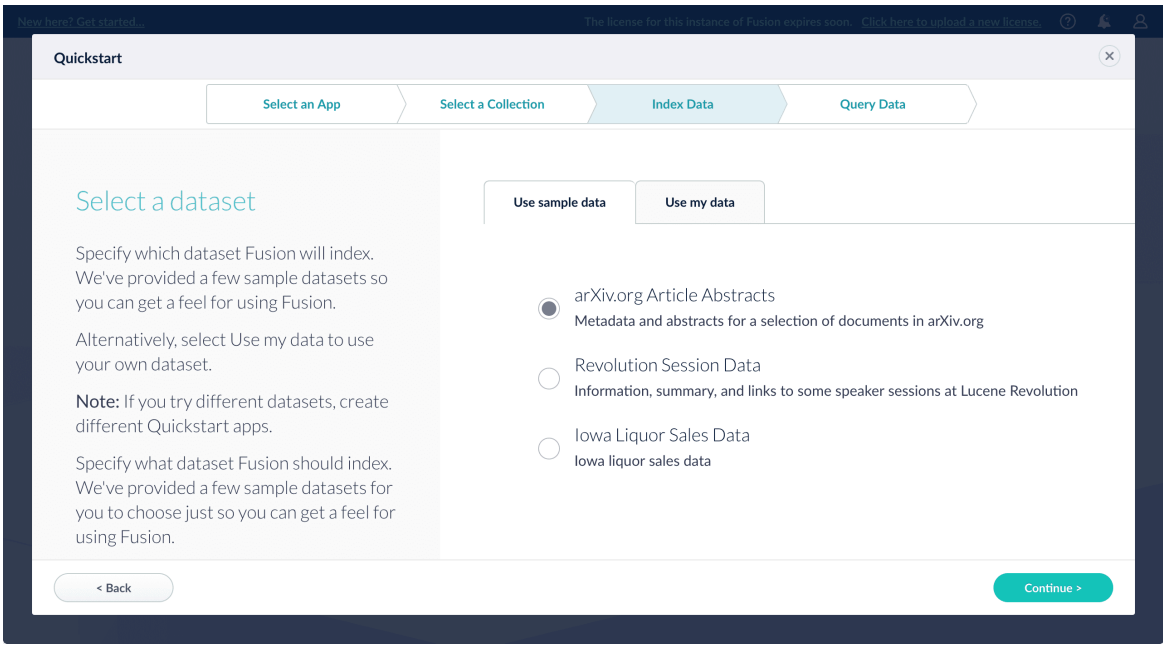
1. Click **Continue**.
   On the **Select an App** screen, you can select or create an *app* for your quickstart data.



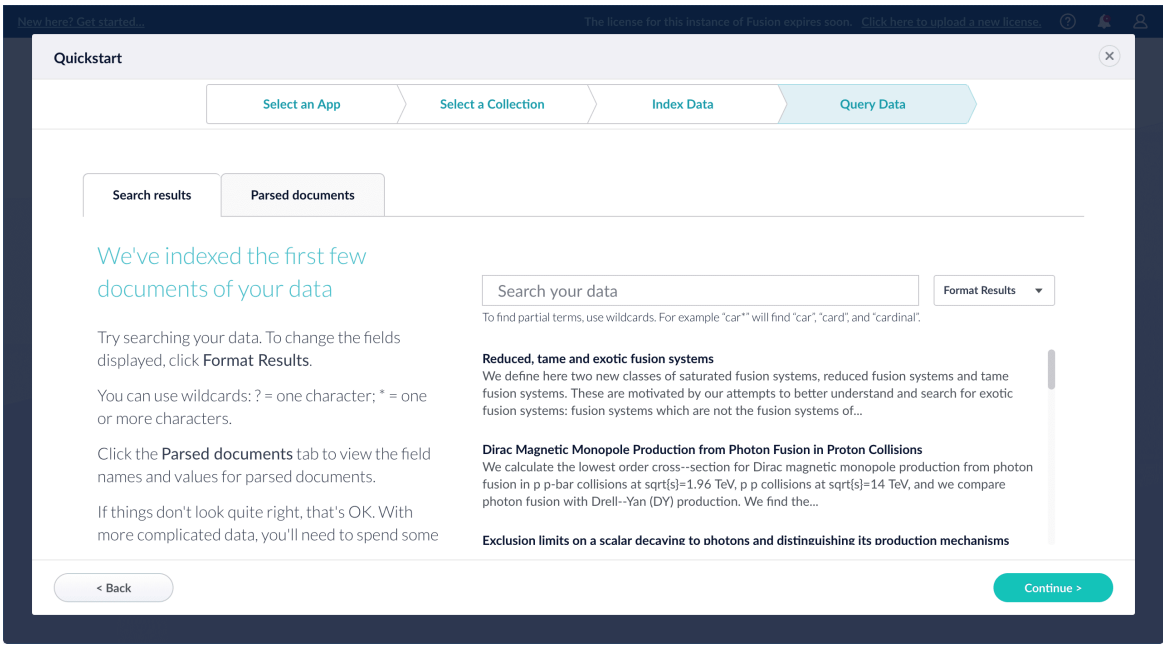
   If you click **Create new app**, Fusion prompts you to enter an app name (and optionally a description), then click **Create App** to return to the Quickstart. Your new app is selected by default.
2. After selecting an app, click **Continue**.
   On the **Select a Collection** screen, you can select or create a *collection* for your quickstart data. A collection with the same name as your app is created automatically:



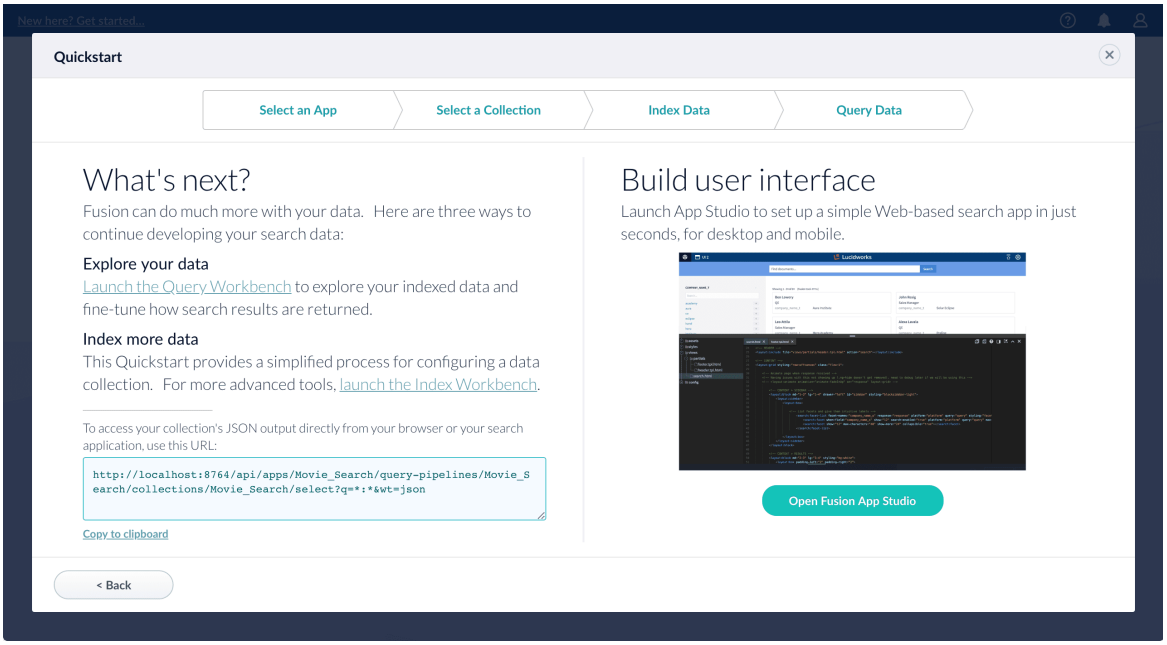
3. Click **Continue**.
On the **Index Data** screen, you can either select one of the built-in sample datasets or click **Use my data** to upload your own:



4. Click **Continue**.
On the **Query Data** screen, you can see all search results and enter your own search queries to test the indexed dataset. You can also select the display fields or view the parsed documents:



5. Click **Continue**.



From here, you can do the following:

- Open the Index Workbench to change the index pipeline.
    - *Fusion 4.x Index Workbench*
    - *Fusion 5.x Index Workbench*

- Open the Query Workbench to change the query pipeline.
    - *Fusion 4.x Query Workbench*
    - *Fusion 5.x Query Workbench*

      The workbenches are essentials tools in the Fusion workflow.

- Open *App Studio* to create a user interface for searching this collection.

Important

App Studio is only available in Fusion 4.1 and 4.2.

Quickstart

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

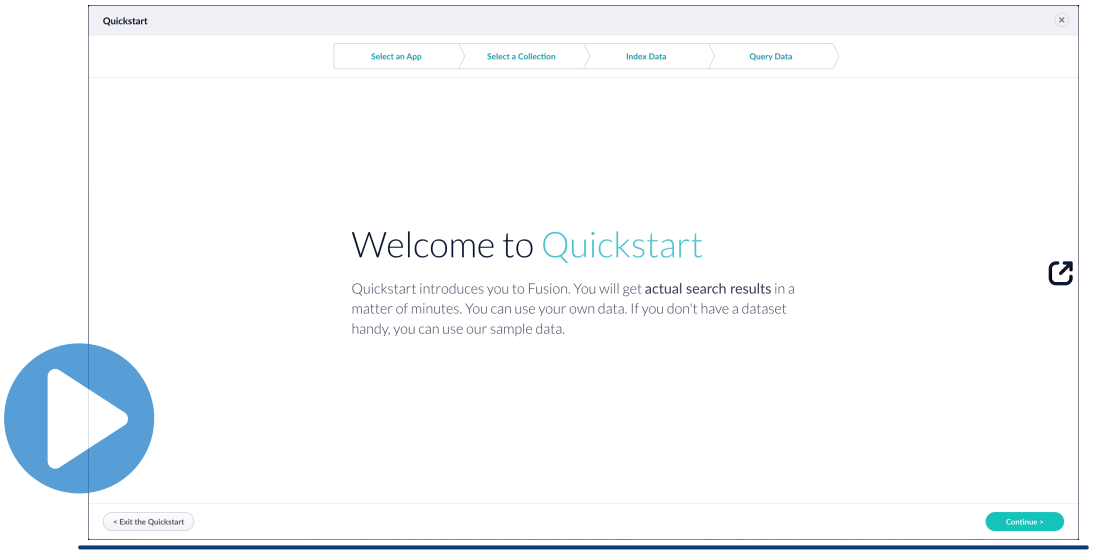
The Quickstart is a wizard that lets you explore some of Managed Fusion's core capabilities:

- Creating or selecting an app
- Selecting a collection (where data is stored)
- Indexing data
- Searching

LucidAcademy

Lucidworks offers free training to help you get started.

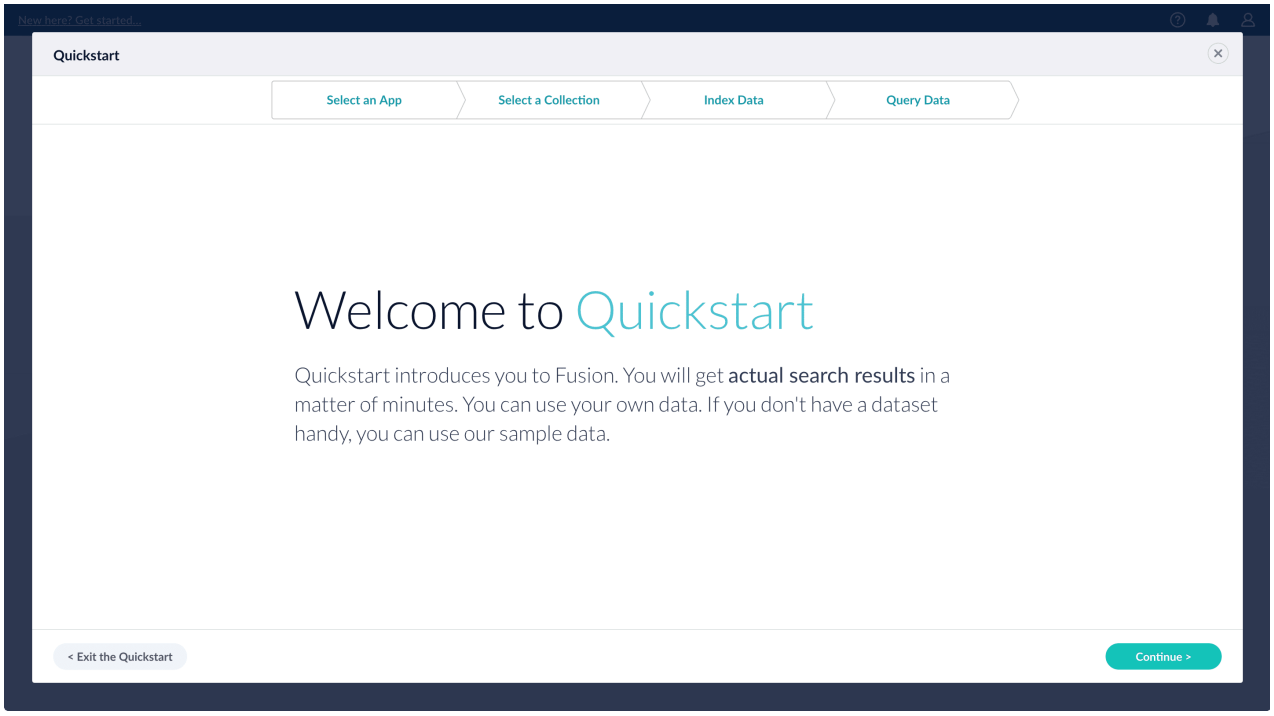
The **Course** for **Using The Quickstart Wizard** focuses on using the wizard to practice using the key functions of Fusion:



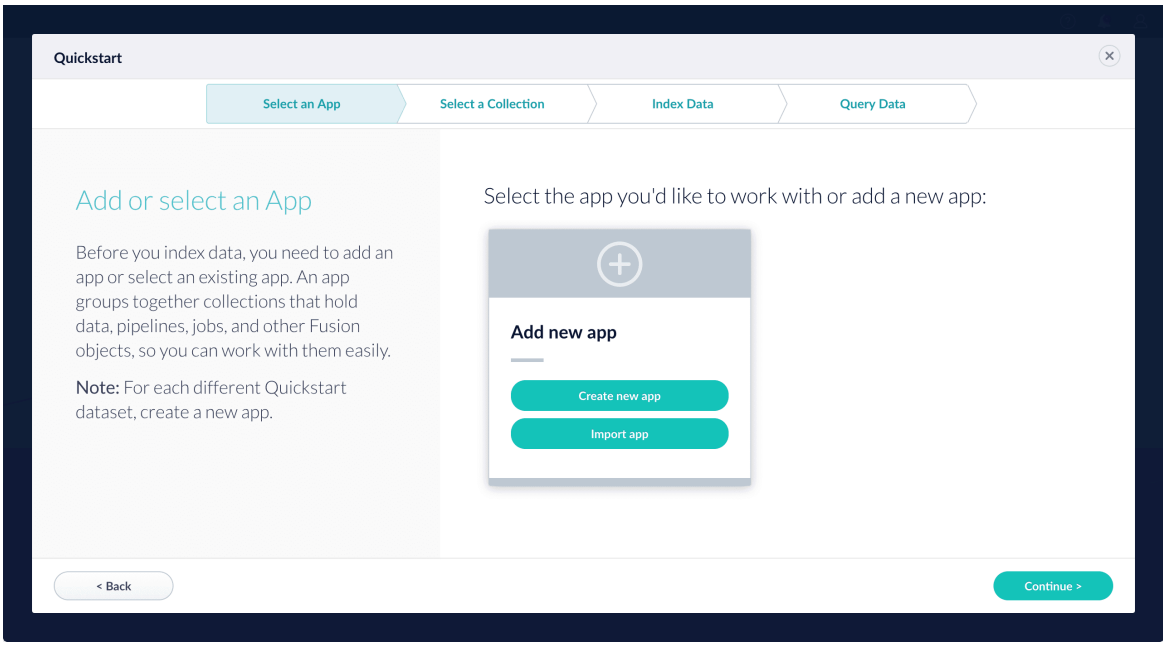
Visit the LucidAcademy to see the full training catalog.

Step through the Quickstart wizard

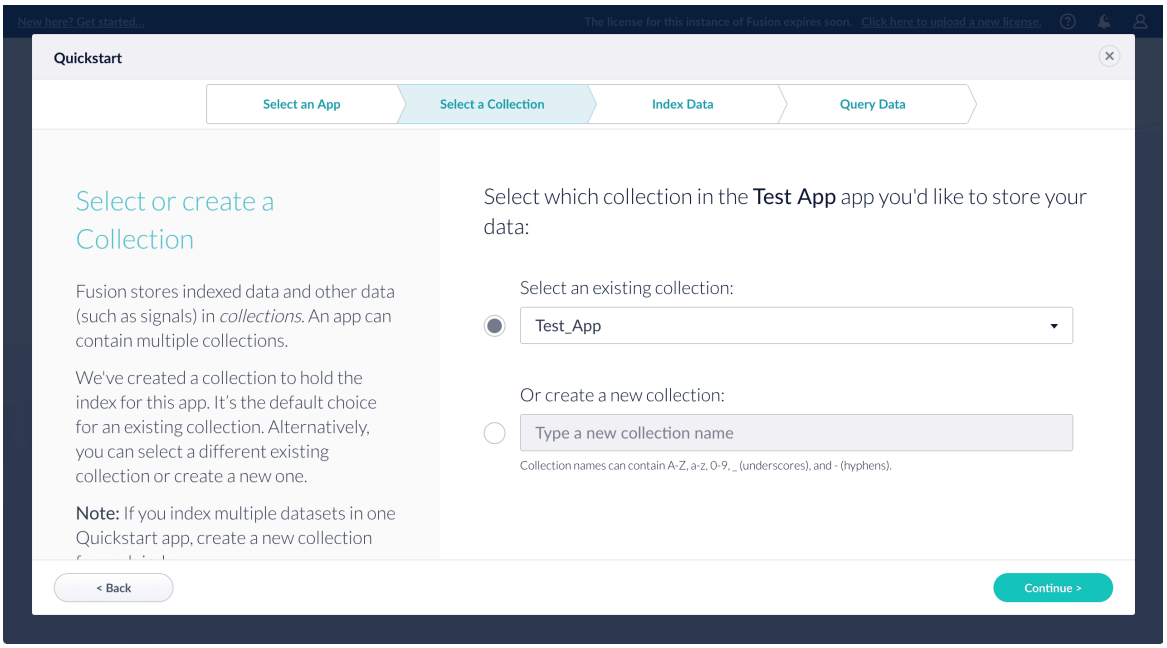
If it is not already open, open the Quickstart wizard. In the Managed Fusion launcher, click **New here? Get started...**



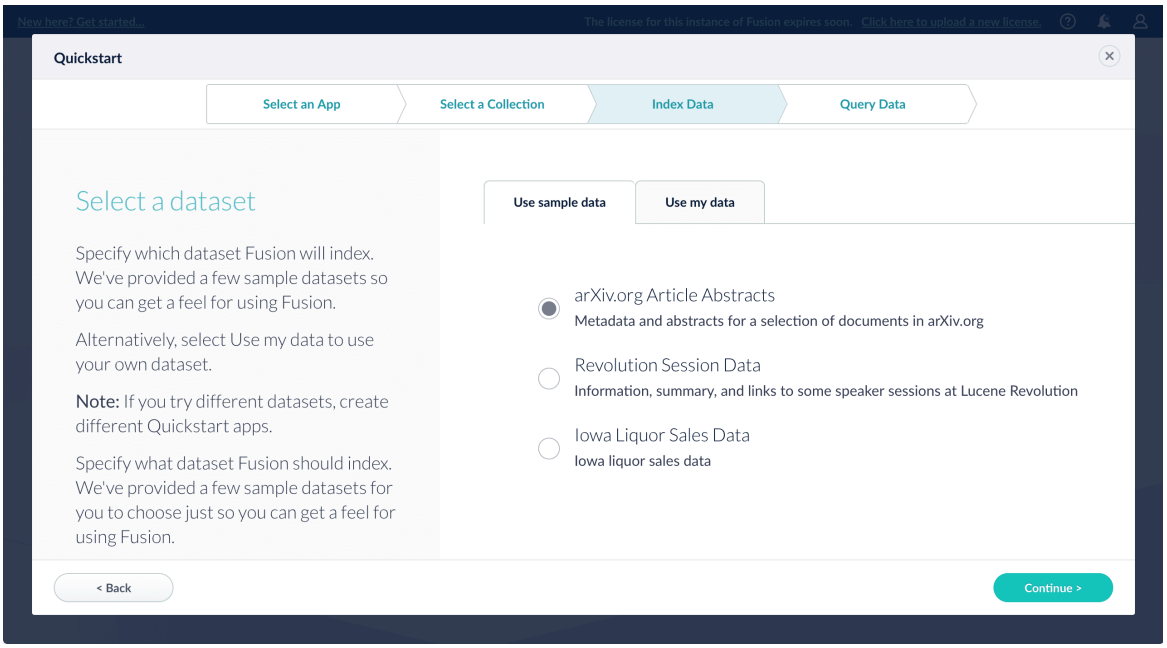
1. Click **Continue**.
   On the **Select an App** screen, you can select or create an app for your quickstart data.



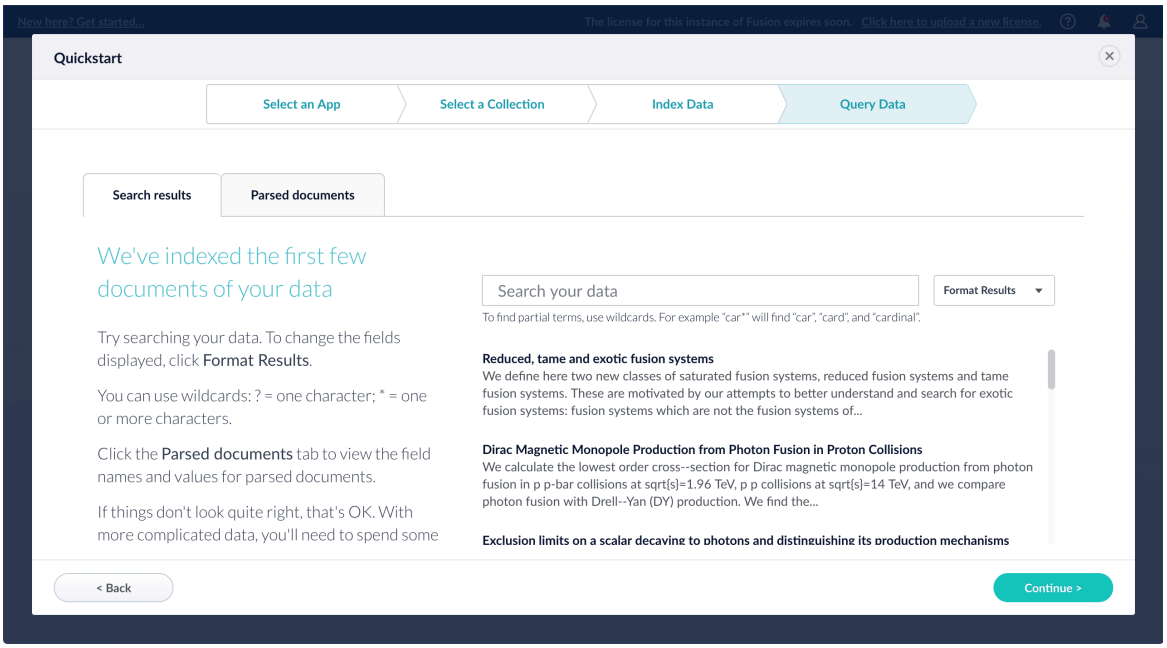
   If you click **Create new app**, Managed Fusion prompts you to enter an app name (and optionally a description), then click **Create App** to return to the Quickstart. Your new app is selected by default.
2. After selecting an app, click **Continue**.
   On the **Select a Collection** screen, you can select or create a *collection* for your quickstart data. A collection with the same name as your app is created automatically:



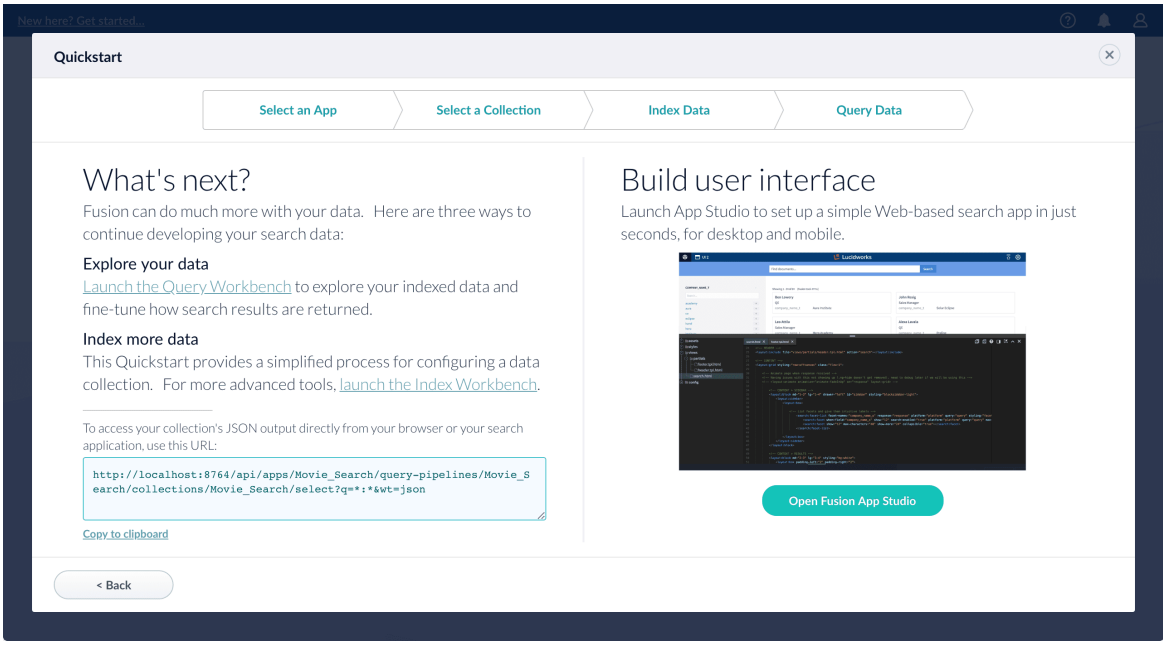
3. Click **Continue**.
On the **Index Data** screen, you can either select one of the built-in sample datasets or click **Use my data** to upload your own:



4. Click **Continue**.
On the **Query Data** screen, you can see all search results and enter your own search queries to test the indexed dataset. You can also select the display fields or view the parsed documents:



5. Click **Continue**.



From here, you can do the following:

- Open the *Index Workbench* to change the index pipeline.
- Open the *Query Workbench* to change the query pipeline.
  The workbenches are essentials tools in the Fusion workflow.

Read from the MongoDB Oplog

How-to

> Works with: *Fusion Connectors*

Retrieve data from a MongoDB instance.

You can configure the Fusion MongoDB connector to read from the MongoDB `oplog` rather than from the entire MongoDB collection. In this mode, the connector crawls the full MongoDB collection, saves a checkpoint in ZooKeeper, then continues running indefinitely, grabbing updates from the `oplog` as they happen in real time. This way, the connector can delete documents that are deleted from MongoDB.

If the connector stops for any reason, it stores a timestamp in ZooKeeper that shows what the latest update was. When the connector restarts, it continues reading from that checkpoint onward.

To read from the MongoDB oplog:

1. Make sure your connector authenticates to Mongo as a user with read `oplog` permissions. See Role-based access control in the MongoDB documentation .
2. Make sure **Process Oplog** is selected in the Fusion MongoDB connector UI.

Read from the MongoDB Oplog

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

Retrieve data from a MongoDB instance.

You can configure the Managed Fusion MongoDB connector to read from the MongoDB `oplog` rather than from the entire MongoDB collection. In this mode, the connector crawls the full MongoDB collection, saves a checkpoint in ZooKeeper, then continues running indefinitely, grabbing updates from the `oplog` as they happen in real time. This way, the connector can delete documents that are deleted from MongoDB.

If the connector stops for any reason, it stores a timestamp in ZooKeeper that shows what the latest update was. When the connector restarts, it continues reading from that checkpoint onward.

To read from the MongoDB oplog:

1. Make sure your connector authenticates to MongoDB as a user with read `oplog` permissions. See Role-based access control in the MongoDB documentation .
2. Make sure **Process Oplog** is selected in the Managed Fusion MongoDB connector UI.

Receive Technical Support

How-to

> Compatible versions: 4.2, 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion AI*, *App Studio*, *Fusion Connectors*, *Fusion Server*, *Fusion*

When you need personalized, expert assistance, the Lucidworks technical support team is available 24/7.

Before you open a support case, see the instructions below to learn how to gather data that the technical support team can use to evaluate your issue and provide a solution.

Once you have obtained the relevant data from Fusion, Submit a request .

LucidAcademy

Lucidworks offers free training to help you get started.

The **Course** for **Support Portal Overview** focuses on how to navigate the support portal, open a support ticket, identify the severity of your issue, and escalate issues as needed:

Visit the LucidAcademy to see the full training catalog.

Exporting log files from the DevOps Center

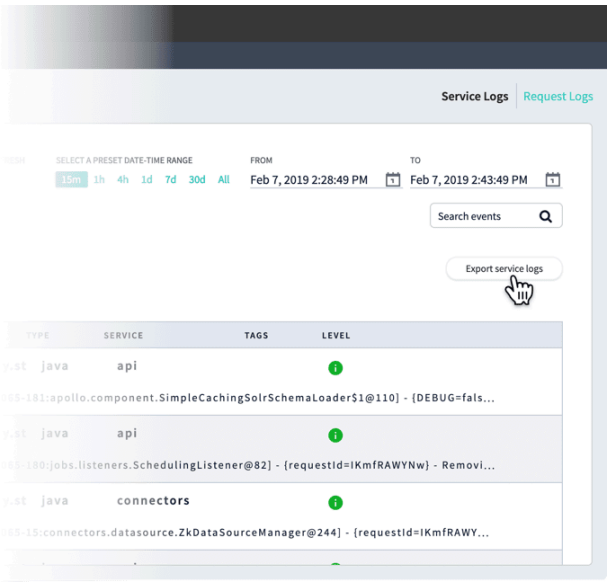
Important

This section only applies to Fusion 4.2 through 5.5.

In Fusion 4.2 through 5.5, the *DevOps Center* provides an easy way to export logs for the timeframe that pertains to your issue.

How to export data from the DevOps Center

1. In the Fusion UI, navigate to **System** > **DevOps Center** > **Log Viewer**.
2. Select the time period that is relevant to the incident you are interested in.
   Log files will be exported according to the tab you are viewing.



Click **Export service logs** in the Service Logs tab or **Export request logs** in the Request Logs tab to download a `.csv` file containing the logs (shown here with example timestamps):

- `serviceLogs[2019-02-07T22:28:49.000Z-2019-02-07T22:43:49.695Z].csv`
- `requestLogs[2019-02-07T22:28:49.000Z-2019-02-07T22:43:49.695Z].csv`

Finding log files in the filesystem

Log files are located in the `https://FUSION_HOST:FUSION_PORT/var/log` directory. Generally, the log files below are the most useful in diagnosing an issue:

- API logs
  `https://FUSION_HOST:FUSION_PORT/var/log/api/api.log` and
  `https://FUSION_HOST:FUSION_PORT/var/log/api/gc_<timestamp>.log.0.current`
- Connectors logs
  `https://FUSION_HOST:FUSION_PORT/var/log/connectors-classic/connectors-classic.log` and `https://FUSION_HOST:FUSION_PORT/var/log/connectors-rpc/connectors-rpc.log`

If it is apparent that additional components are involved, collect their logs and associated `gc` logs from the `https://FUSION_HOST:FUSION_PORT/var/log` directory.

Opening a support case

Lucidworks provides technical support to customers 24 hours a day, 7 days a week, with unlimited incidents. Once you have obtained the relevant data from Fusion as described above, you can open a support case.

Note

For detailed information about submitting a support case, see Lucidworks Customer Center Training.

To open a support case, complete the following:

1. Access the Lucidworks Submit a request page.
   You can also select **Contact Support** from any page on the Lucidworks support site.
2. Enter information in the following fields:
   - **Your email address**
   - **Subject**

- **Description**

3. In the **Severity** field, select the appropriate severity value for your issue according to the guidelines below.

Severity	Time	Definition	Example
S1	1 Hour	Severe commercial impact on Customer 's business which either makes the system inoperable or prevents content discovery.	Production search is down.
S2	12 Hours	Significant commercial impact on Customer 's business which makes one or more critical areas of functionality inoperable.	Search performance has slowed down to a crawl.
S3	24 Hours	Moderate commercial impact on Customer 's business which makes one or more non-critical areas of functionality inoperable.	Slight increase in query time.
S4	48 Hours	No commercial impact on Customer 's business.	How to implement a certain search feature.

4. Select values in the following fields:
   - **Product Line**
   - **Fusion Version** (optional)
   - **Solr Version** (optional)

5. In the **Environment** field, select one of the following:
   - **Dev/Sandbox/POC**
   - **Performance**
   - **Production**
   - **QA/UAT**
   - **Staging**

6. In the **Attachments** field, drag files or click to add *log files*.

> **Tip**
>
> You can attach only one file. Upload all of your files as one `.tar` or `.zip` archive.

7. When you have attached all pertinent data, click **Submit**.

You will receive an email message containing your case number. A Lucidworks support engineer will contact you within the time period corresponding to your issue's severity level.

Receive Technical Support

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

Technical support

When you need personalized, expert assistance, the Lucidworks technical support team is available 24/7.

Before you open a support case, see the instructions below to learn how to gather data that the technical support team can use to evaluate your issue and provide a solution.

Once you have obtained the relevant data from Fusion, Submit a request .

LucidAcademy

Lucidworks offers free training to help you get started.

The **Course** for **Support Portal Overview** focuses on how to navigate the support portal, open a support ticket, identify the severity of your issue, and escalate issues as needed:

Visit the LucidAcademy to see the full training catalog.

Exporting log files from the DevOps Center

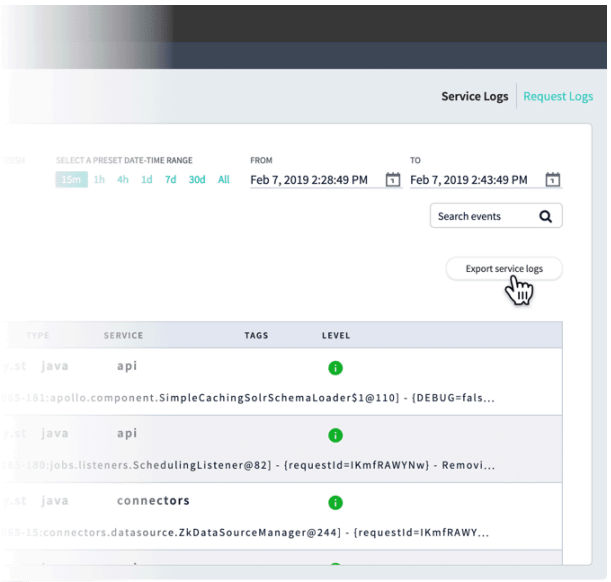
Important

This section only applies to Fusion 4.2 through 5.5.

In Fusion 4.2 through 5.5, the *DevOps Center* provides an easy way to export logs for the timeframe that pertains to your issue.

How to export data from the DevOps Center

1. In the Fusion UI, navigate to **System** > **DevOps Center** > **Log Viewer**.
2. Select the time period that is relevant to the incident you are interested in.
   Log files will be exported according to the tab you are viewing.



Click **Export service logs** in the Service Logs tab or **Export request logs** in the Request Logs tab to download a `.csv` file containing the logs (shown here with example timestamps):

- `serviceLogs[2019-02-07T22:28:49.000Z-2019-02-07T22:43:49.695Z].csv`
- `requestLogs[2019-02-07T22:28:49.000Z-2019-02-07T22:43:49.695Z].csv`

Finding log files in the filesystem

Log files are located in the `https://FUSION_HOST:FUSION_PORT/var/log` directory. Generally, the log files below are the most useful in diagnosing an issue:

- API logs
  `https://FUSION_HOST:FUSION_PORT/var/log/api/api.log` and
  `https://FUSION_HOST:FUSION_PORT/var/log/api/gc_<timestamp>.log.0.current`
- Connectors logs
  `https://FUSION_HOST:FUSION_PORT/var/log/connectors-classic/connectors-classic.log` and `https://FUSION_HOST:FUSION_PORT/var/log/connectors-rpc/connectors-rpc.log`

If it is apparent that additional components are involved, collect their logs and associated `gc` logs from the `https://FUSION_HOST:FUSION_PORT/var/log` directory.

Opening a support case

Lucidworks provides technical support to customers 24 hours a day, 7 days a week, with unlimited incidents. Once you have obtained the relevant data from Fusion as described above, you can open a support case.

Note

For detailed information about submitting a support case, see Lucidworks Customer Center Training.

To open a support case, complete the following:

1. Access the Lucidworks Submit a request page.
   You can also select **Contact Support** from any page on the Lucidworks support site.
2. Enter information in the following fields:
   - **Your email address**
   - **Subject**

- **Description**

3. In the **Severity** field, select the appropriate severity value for your issue according to the guidelines below.

Severity	Time	Definition	Example
S1	1 Hour	Severe commercial impact on Customer 's business which either makes the system inoperable or prevents content discovery.	Production search is down.
S2	12 Hours	Significant commercial impact on Customer 's business which makes one or more critical areas of functionality inoperable.	Search performance has slowed down to a crawl.
S3	24 Hours	Moderate commercial impact on Customer 's business which makes one or more non-critical areas of functionality inoperable.	Slight increase in query time.
S4	48 Hours	No commercial impact on Customer 's business.	How to implement a certain search feature.

4. Select values in the following fields:
   - **Product Line**
   - **Fusion Version** (optional)
   - **Solr Version** (optional)

5. In the **Environment** field, select one of the following:
   - **Dev/Sandbox/POC**
   - **Performance**
   - **Production**
   - **QA/UAT**
   - **Staging**

6. In the **Attachments** field, drag files or click to add *log files*.

> **Tip**
>
> You can attach only one file. Upload all of your files as one `.tar` or `.zip` archive.

7. When you have attached all pertinent data, click **Submit**.

You will receive an email message containing your case number. A Lucidworks support engineer will contact you within the time period corresponding to your issue's severity level.

Recreate a Milvus Collection

How-to

> Compatible versions: 5.5, 5.9
>
> Works with: *Fusion*

REVIEW THIS DOCUMENT FOR VALIDITY WITH MANAGED FUSION

Use this process to:

- Recreate a lost Milvus collection.
- Create a Milvus collection for a different field.
- Create a Milvus collection that uses a different encryption model.

Prerequisites

- Used the Smart Answers pipeline to index data, which created the Milvus and Solr collections.
- The created Solr collection still exists, which is used as the Source Collection.

Process

1. Create the Milvus collection. See *Create Collections in Milvus* for more information.

   Note

   If the Milvus collection still exists and you want to overwrite it, select the **Override Collections** checkbox. This option deletes the current collection, allowing you to create a new collection.

2. Edit the Smart Answers Index Pipeline as follows:
   - Disable the Solr Indexer stage.
   - If you are creating a Milvus collection for a different field, or one that uses a different encryption model, access the **Encode into Milvus** stage and:
     - Specify the **model**, the **Encoder Output Vector**, and the **Field to Encode**.
     - Verify the **Milvus Collection Name** refers to the new Milvus collection.

3. Navigate to the Datasources panel.
4. Select **Add a Solr connector** and enter the following information:
   - **Pipeline ID.** Smart Answers pipeline
   - **Solr Connection Type.** SolrCloud
   - **SolrCloud Zookeeper Host String.** Execute the following curl command. The response displays the SolrCloud Zookeeper Host String value. In the following `RESPONSE` example, the value you would set for the SolrCloud Zookeeper Host String is `ns-zookeeper-0.ns-zookeeper-headless:2181`.

     ```
     curl -u USERNAME:PASSWORD
     https://FUSION_HOST:FUSION_PORT/api/searchCluster/default
     ```

     RESPONSE

```
{ "id": "default", "connectString": "ns-zookeeper-0.ns-
zookeeper-headless:2181", "zkClientTimeout": 30000,
"zkConnectTimeout": 60000, "cloud": true,
"bufferFlushInterval": 1000, "bufferSize": 100,
"concurrency": 10, "authConfig": { "authType": "none" },
"validateCluster": true }
```

- **Source Collection.** Original collection from the initial Smart Answers pipeline indexing job

5. Navigate to Index Workbench and load the new solr-index datasource.
6. Select **Start Job** to run the Solr datasource through the updated Smart Answers pipeline.
7. When the data is reindexed, edit the Smart Answers pipeline and re-enable the Solr Indexer stage.

Refine query results with data source labels

How-to

> Works with: Connected Search

Springboard has options to manage and query the content obtained from different data source configurations. These options include labels and filters to limit the scope of a query when an application has multiple data sources.

The examples on this page demonstrate how to search content specific to English and French directories on a books website. As the website's search engineer, you can use the same fields to search content specific to directories on your website.

Example data sources configuration

In the Data Sources page for the example website's application, the data sources are configured with following information:

Data source name	Start URL	Labels	Include file types
English	`example.com/books/english`	`books-english`	PDF
French	`example.com/books/french`	`books-french`	PDF

Filters in the API calls that apply these labels generate different content specific to the two directories. Web page attributes are automatically identified during data source ingestion and can be used as facet fields in queries, such as `file.extension`.

For more information, see *Manage Springboard data sources*.

Example 1: Display content by label name

A request that uses the `books-french` label from the home page for French books to the Query API's `browse` endpoint returns that specific list of items. The website parses the JSON response and displays the list.

Request

```
{ "utcOffset": "-05:00", "sessionId": "864782f0-af36-4dee-8430-9e73d6006eaa", "filters": [ {"field": "datasourceLabels", "values": ["books-french"]} ] }
```

Response

```
{ "queryId": "fd110486-f168-47c0-a419-1518a4840589", "hits": 102, "docs": [ { "type": "page", "id": "441eb3be-7de6-470a-8141-e416a15c7db1-6a092bd4-5098-466c-94aa-40bf68294303-687088076571483713333558199345704397 31", "rank": 1, "datasourceLabels": [ "books-french" ], "fields": { "document.title": "Grisha, tome 1 : Les orphelins du royaume", "document.url": "https://example.com/french-book-reviews/95.html" } } ] }
```

Example 2: Display content by file extension

A request that uses `file.extension` returns a list of that file type. For example, the books website stores some books as PDF files along with a page to view the list of those pdf files.

Request:

```
{ "utcOffset": "-05:00", "sessionId": "864782f0-af36-4dee-8430-
9e73d6006eaa", "filters": [ {"field": "file.extension", "values":
["PDF"]} ] }
```

Response

```
{ "queryId": "fd110486-f168-47c0-a419-1518a4840589", "hits": 3155,
"docs": [ { "type": "page", "id": "441eb3be-7de6-470a-8141-
e416a15c7db1-6a092bd4-5098-466c-94aa-40bf68294303-
6870880765714837133335581993457043973 1", "rank": 1, "fields": {
"document.title": "Raggiunse 26 Edizioni In Lingua", "document.url":
"https://example.com/248.pdf" } } ] }
```

Example 3: Display content by label name and file extension

A request that uses both `file.extension` and `label` returns the specified list. For example, the list of books stored as pdf files that are written in English.

Request

```
{ "utcOffset": "-05:00", "sessionId": "864782f0-af36-4dee-8430-
9e73d6006eaa", "filters": [ {"field":"datasourceLabels","values":
["books-english"]}, {"field": "file.extension", "values": ["PDF"]}] }
```

Response


```
{ "type": "page", "id": "441eb3be-7de6-470a-8141-e416a15c7db1-
6a092bd4-5098-466c-94aa-40bf68294303-
68708807657148371333355819934570439731", "rank": 1,
"datasourceLabels": ["books"], "fields": { "document.title": "Ways and
means of using birds in combating noxious insects", "document.url":
"https://example.com/204414.pdf" } }
```

Register for Twitter Credentials

How-to

> Works with: *Fusion Connectors*

The Twitter Search connector uses Twitter's search API to query Twitter for tweets that match specific parameters. It allows querying for any keyword, location or other query terms.

In order to successfully configure either Twitter connector, you must first register your application with Twitter and accept their terms of service. The registration process provides you with the required OAuth tokens you need to access either API.

To get the tokens, follow these steps:

1. Make sure you have a Twitter account, and go to https://dev.twitter.com/ and sign in.
2. After signing in, choose **My Applications** from the pull-down menu at the upper right that shows a thumbnail of your Twitter profile picture (if you have one).
3. Choose **Create New App** and fill out the required details.
   The callback field can be skipped, but you must accept the Terms of Service.
4. To save your information, choose **Create Your Twitter Application** to register your application.
   The next page contains the Consumer Key and Consumer Secret, which you will need to configure the datasource in Fusion.
5. At the bottom of the same page, choose **Create My Access Token**.
   The next page contains the Access Token and Token Secret, which you will also need to configure the datasource in Fusion.

While you need a Twitter account to register an application, you do not use your Twitter username and password to configure this data source. The APIs will only use the Consumer Key, Consumer Secret, Access Token, and Token Secret information as authentication, so store it where you can access it while configuring the data source.

Register for X credentials

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

The X (formerly known as Twitter) Search connector uses X's search API to query X for tweets that match specific parameters. It allows querying for any keyword, location or other query terms.

To successfully configure either X connector, you must first register your application with X and accept their terms of service. The registration process provides you with the required OAuth tokens you need to access either API.

To get the tokens, follow these steps:

1. Make sure you have an X account, and go to https://dev.twitter.com/ and sign in.
2. Click your X profile picture to display the menu and click **My Applications**.
3. Click **Create New App** and fill out the required details.
   The callback field can be skipped, but you must accept the Terms of Service.
4. Click **Create Your X Application** to register your application.
5. From the page that displays, copy the Consumer Key and Consumer Secret. You need that information to configure the datasource in Managed Fusion.
6. Click **Create My Access Token**.
7. From the page that displays, copy the Access Token and Token Secret. You need that information to configure the datasource in Managed Fusion.

While you need an X account to register an application, you do not use your X username and password to configure this data source. The APIs only use the Consumer Key, Consumer Secret, Access Token, and Token Secret information as authentication, so store it where you can access it while configuring the data source.

Remotely Launch a Plugin

How-to

> Compatible versions: 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion Connectors*, *Fusion*

Follow this configuration to use a remote connector in Fusion 5.6+.

Note

A known issue is after idling for five minutes without any new connector start requests, the remote connector stops responding to server requests and must restart. This causes connector start requests to hang. A workaround is to schedule a connector to run every three minutes. It is recommended to use a connector instance that doesn't fetch any documents to keep the remote connector functional without frequent restarts.

1. Execute the following command.

```
java –jar connector–plugin–client–standalone.jar
[YAML_CONFIG_FILE]
```

2. Check that the `YAML_CONFIG_FILE` has the following format.

```
kafka–bridge: target: mynamespace–connectors–
backend.lucidworkstest.com:443 # mandatory plain–text: false #
optional proxy–server: # optional – needed when a forward proxy
server is used to provide outbound access to the standalone
connector host: host port: 111 user: user password: password
trust: # optional – needed when the client's system doesn't
trust the server's certificate cert–collection–filepath: path1
proxy: # mandatory fusion–proxy user: admin password:
password123 url: https://fusiontest.com/ # needed only when the
connector plugin requires blob store access plugin: # mandatory
path: ./fs.zip type: suffix: remote
```

   a. Use this table for descriptions of the YAML properties.

Key	Value description
`kafka-bridge.target`	The Kafka-gRPC bridge host:port pair.
`kafka-bridge.plain-text`	Enable when TLS ingress is not used.
`kafka-bridge.proxy-server.host`	The Forward Proxy Server Host.
`kafka-bridge.proxy-server.port`	The Forward Proxy Server Port.
`kafka-bridge.proxy-server.user`	The Forward Proxy Server User.
`kafka-bridge.proxy-server.password`	The Forward Proxy Server Password.
`kafka-bridge.trust.cert-collection-filepath`	The Private Key file path.
`proxy.url`	Fusion proxy url.
`proxy.user`	Fusion proxy user.
`proxy.password`	Fusion proxy password.
`plugin.path`	The path of the plugin zip file.
`plugin.type`	Optional suffix added to the plugin `id`. For example, if the plugin `id` is `lucidworks.test` and the suffix is `remote`, the plugin will be launched with `lucidworks.test.remote`.

Example `YAML_CONFIG_FILE`:

```
kafka-bridge: target: mynamespace-connectors.example.com:443 plain-text: false proxy: user: admin password: "password123" plugin: path: ./testplugin.zip
```

Removing the Security Module

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion*

These steps are required to remove the *security module* from a project.

Important

You must also remove any other module that depends on the security module such as the *Collaboration module*.

1. POM

Remove the security module dependency (and any other module that requires security) from `pom.xml`:

```
<dependency> <groupId>twigkit</groupId>
<artifactId>twigkit.security.provider.spring-security</artifactId>
<version>${project.parent.version}</version> </dependency>
```

2. Configuration

Replace the security module specified in conf/resources/security/security.conf with `type: generic`

3. Web.xml application descriptor

You must remove any reference to the Spring security filters and listeners from your `web.xml` application descriptor. For example:

```
<!-- Spring Security --> <context-param> <param-
name>contextConfigLocation</param-name> <param-value>classpath:spring-
security.xml</param-value> </context-param> <listener> <listener-
class>org.springframework.web.context.ContextLoaderListener</listener-
class> </listener> <filter> <filter-
name>springSecurityFilterChain</filter-name> <filter-
class>org.springframework.web.filter.DelegatingFilterProxy</filter-
class> </filter> <filter-mapping> <filter-
name>springSecurityFilterChain</filter-name> <url-pattern>/*</url-
pattern> </filter-mapping>
```

4. Java code

If you are using any classes from the security Module in Java code, remove references to these. For example, if the `SecurityModule` is installed in a custom module, you would remove these references:

```
import twigkit.security.springsecurity.SpringSecurityModule;
install(new SpringSecurityModule());
```

5. Security configuration files

Optionally, remove the `spring-security.xml` configuration file.

Removing the Security Module

How-to

> Compatible versions: 4.2
>
> Works with: *App Studio*

These steps are required to remove the *security module* from a project.

Important

You must also remove any other module that depends on the security module such as the *Collaboration module*.

1. POM

Remove the security module dependency (and any other module that requires security) from `pom.xml`:

```
<dependency> <groupId>twigkit</groupId>
<artifactId>twigkit.security.provider.spring-security</artifactId>
<version>${project.parent.version}</version> </dependency>
```

2. Configuration

Replace the security module specified in conf/resources/security/security.conf with `type: generic`

3. Web.xml application descriptor

You must remove any reference to the Spring security filters and listeners from your `web.xml` application descriptor. For example:

```
<!-- Spring Security --> <context-param> <param-
name>contextConfigLocation</param-name> <param-value>classpath:spring-
security.xml</param-value> </context-param> <listener> <listener-
class>org.springframework.web.context.ContextLoaderListener</listener-
class> </listener> <filter> <filter-
name>springSecurityFilterChain</filter-name> <filter-
class>org.springframework.web.filter.DelegatingFilterProxy</filter-
class> </filter> <filter-mapping> <filter-
name>springSecurityFilterChain</filter-name> <url-pattern>/*</url-
pattern> </filter-mapping>
```

4. Java code

If you are using any classes from the security Module in Java code, remove references to these. For example, if the `SecurityModule` is installed in a custom module, you would remove these references:

```
import twigkit.security.springsecurity.SpringSecurityModule;
install(new SpringSecurityModule());
```

5. Security configuration files

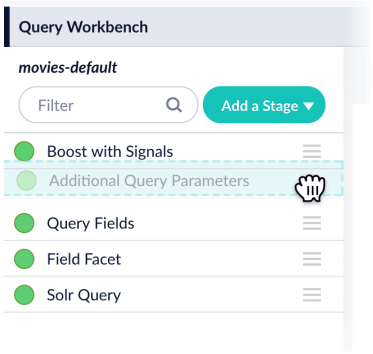
Optionally, remove the `spring-security.xml` configuration file.

Reorder Query Pipeline Stages

How-to

> Compatible versions: 4.2, 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion Server*, *Fusion*

The order of the pipeline stages matters, because the output from one stage becomes the input to the next stage. For example, the *Solr Query stage* must always come last in the sequence, so that data is indexed only after it has been processed by all other stages. Putting this stage first in the sequence means that subsequent stages have no effect on the indexed data.



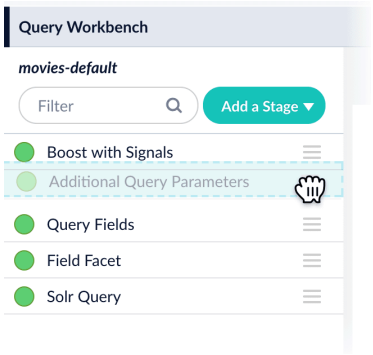
Drag any stage in the pipeline to move it up or down in the sequence of stages. The preview panel automatically updates the search results to reflect the output of the new sequence.

Reorder Query Pipeline Stages

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

The order of the pipeline stages matters, because the output from one stage becomes the input to the next stage. For example, the *Solr Query stage* must always come last in the sequence, so that data is indexed only after it has been processed by all other stages. Putting this stage first in the sequence means that subsequent stages have no effect on the indexed data.



Drag any stage in the pipeline to move it up or down in the sequence of stages. The preview panel automatically updates the search results to reflect the output of the new sequence.

Retrieve a List of Signal Types

How-to

> Compatible versions: 4.2, 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion AI*, *Fusion Server*, *Fusion*

1. In the Fusion UI, select your signals collection.
2. Open the *Query Workbench* by navigating to **Query** > **Query Workbench**.
3. Click **Add a field facet**.
4. Select the `type` field.
   The list of signal types appears in the facet panel:

Retrieve a List of Signal Types

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

1. In the Fusion UI, select your signals collection.
2. Open the *Query Workbench* by navigating to **Query** > **Query Workbench**.
3. Click **Add a field facet**.
4. Select the `type` field.
   The list of signal types appears in the facet panel:

Run a Job

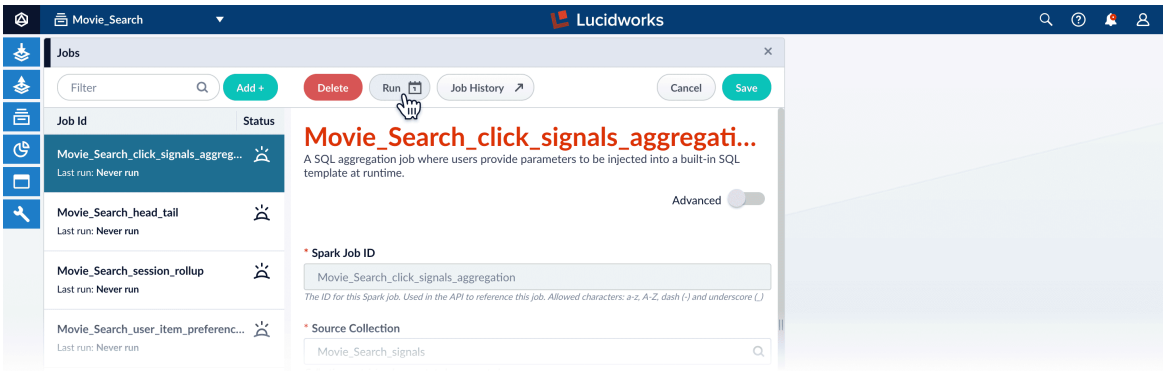
How-to

> Compatible versions: 4.2
>
> Works with: *Fusion Server*

How to run a job once

1. In the Fusion workspace, navigate to > **Jobs**.

2. Select the job from the job list.
3. Click **Run**.



4. Click **Start**.

Run App Studio in Demo Mode

How-to

> Compatible versions: 4.2
>
> Works with: *App Studio*

This deployment type is useful for:

- Evaluating App Studio
- Previewing the UI during development
- Demoing your UI

In order to run your UI in demo mode, you need a *downloaded project*.

From your project directory, run `app-studio start` or `app-studio.bat start` and go to `http://localhost:8080/` to view your demo deployment. The *code editor* is available in the demo interface; you can hide this by using the `--production` flag.

Start up in demo mode

Run the `app-studio` or `app-studio.bat` script like this:

```
app-studio start [--production] [-f] [-p port] [-m memory] [-t timeout] [-V]
```

After startup, your demo deployment is available at `http://localhost:8080/`.

`--production`	Run App Studio in "production mode", with the code editor and wizard disabled. This gives you a preview of what your UI will look like in a production deployment.
`-f`	Start App Studio in the foreground. By default, App Studio starts in the background and sends stdout/stderr to `app-studio-PORT-console.log`.
`-p <port>`	Specify the port on which to start the App Studio Web server; the default is 8080.
`-m <memory>`	Set the minimum (`-Xms`) and maximum (`-Xmx`) heap size for the JVM, such as `-m 4g` which results in `-Xms4g -Xmx4g`. By default, this script sets the heap size to 512MB.
`-t <timeout>`	Set the startup timeout in seconds; the default is 120.

Shut down in demo mode

To shut down your demo application, run `app-studio stop` or `app-studio.bat stop`:

```
app-studio stop [-p port] [-V]
```

The optional `-p` flag specifies the port to which the App Studio HTTP listener is bound.

SAML Configuration

How-to

> Compatible versions: 4.2, 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *App Studio*

As SAML is a flexible standard supporting varied security 'architectures', it is necessary to have a clear understanding of the authentication setup against which the Appkit application will be using. This said the SAML module is highly flexible, supporting a range of configurations.

In this example, we assume a fairly straightforward configuration of authentication against an IdP providing the SSO login form. The Appkit application acts as an SP that authenticates against this IdP (requiring configuration of the IdP to accept login requests for this application).

It is also assumed that the entire Appkit application is secured by the SSO and the desired behavior is to redirect the user to the IdP's login form if they are not authenticated. However it is very simple to modify the application to allow anonymous access where login is only required for extra features.

1. Change the application's security provider to SAML

Simply modify the security provider type to `saml` in `security.conf`:

```
type: saml
```

2. Use the SAML security module

Add this dependency to the application's `pom.xml` file. If an existing security provider is configured (an `artifactId` starting with `twigkit.security.provider`), replace the existing dependency with this dependency:

```
<dependency> <groupId>twigkit</groupId>
<artifactId>twigkit.security.provider.saml</artifactId>
<version>${project.parent.version}</version> </dependency>
```

3. Add a SAML security configuration file

1. Create a SAML configuration file at: `src/main/resources/conf/security/saml.conf`. Add this content:

```
sso-url: http://your-sso-host/idp-initiated-login-page slo-url:
http://your-sso-host/idp-single-logout-page entity-id: unique-
id-for-this-service-provider
```

2. Change the properties to relevant values as described here:
   `sso-url`: Configure application's login failure redirect endpoint (SSO URL).
   `slo-url`: Configure application's logout success redirect URL (SLO URL). Typically this is the URL at the IdP which triggers 'Single Logout' (SLO), logging out the user from the IdP system entirely.
   `entity-id`: Define the SP (partner) entity ID for this application (as configured in the IdP backend). If not already defined, choose a unique identifier to use to refer to this SP and provide this to the IdP.

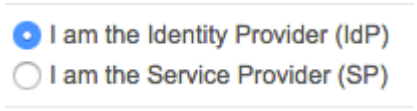
4. Add the IdP metadata to the application

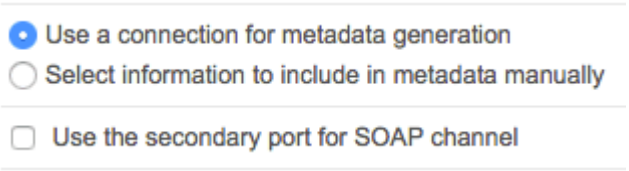
This simply requires an export of the IdP's metadata from the IdP itself. This file should then be saved as 'idp-metadata.xml' under the `src/main/resources` directory. You can if necessary also configure a custom metadata XML file location using:

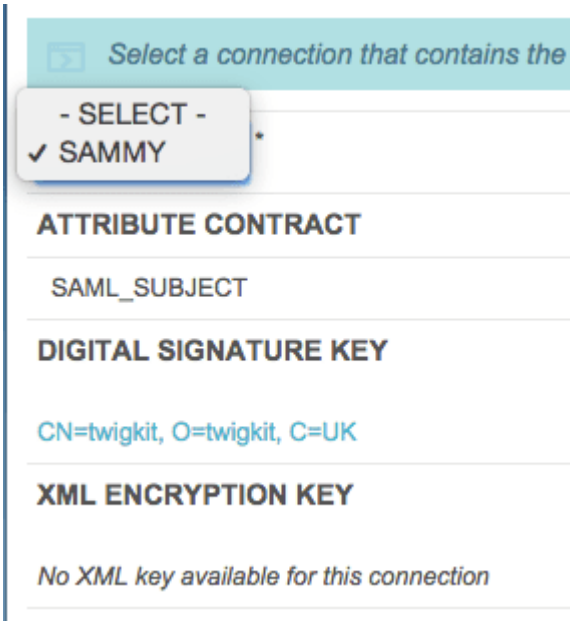
`idp-metadata-file` : path of the IdP metadata file relative to `src/main/resources` (defaults to 'idp-metadata.xml').

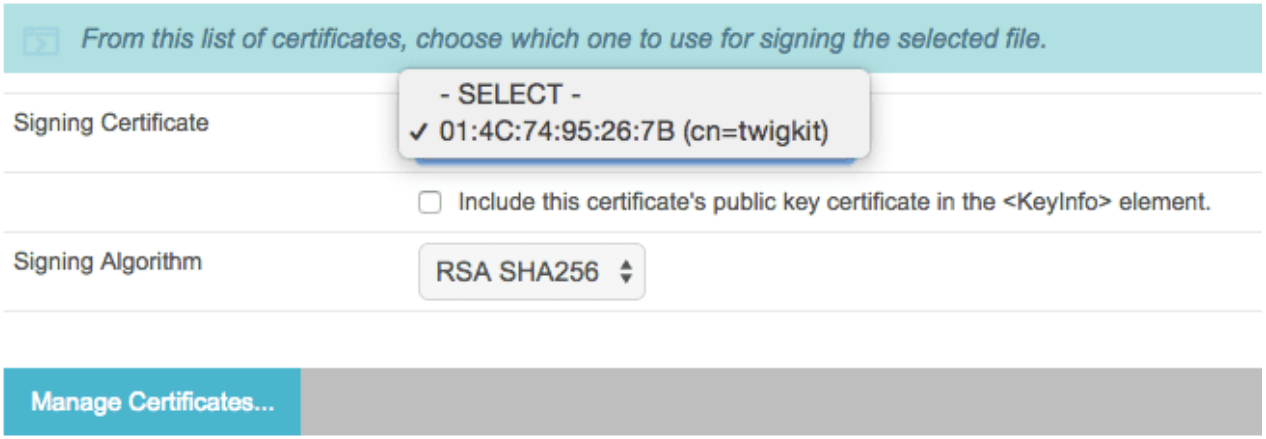
The majority of IdPs provide this metadata export functionality. In this case, we are using PingFederate (see the screenshots below).

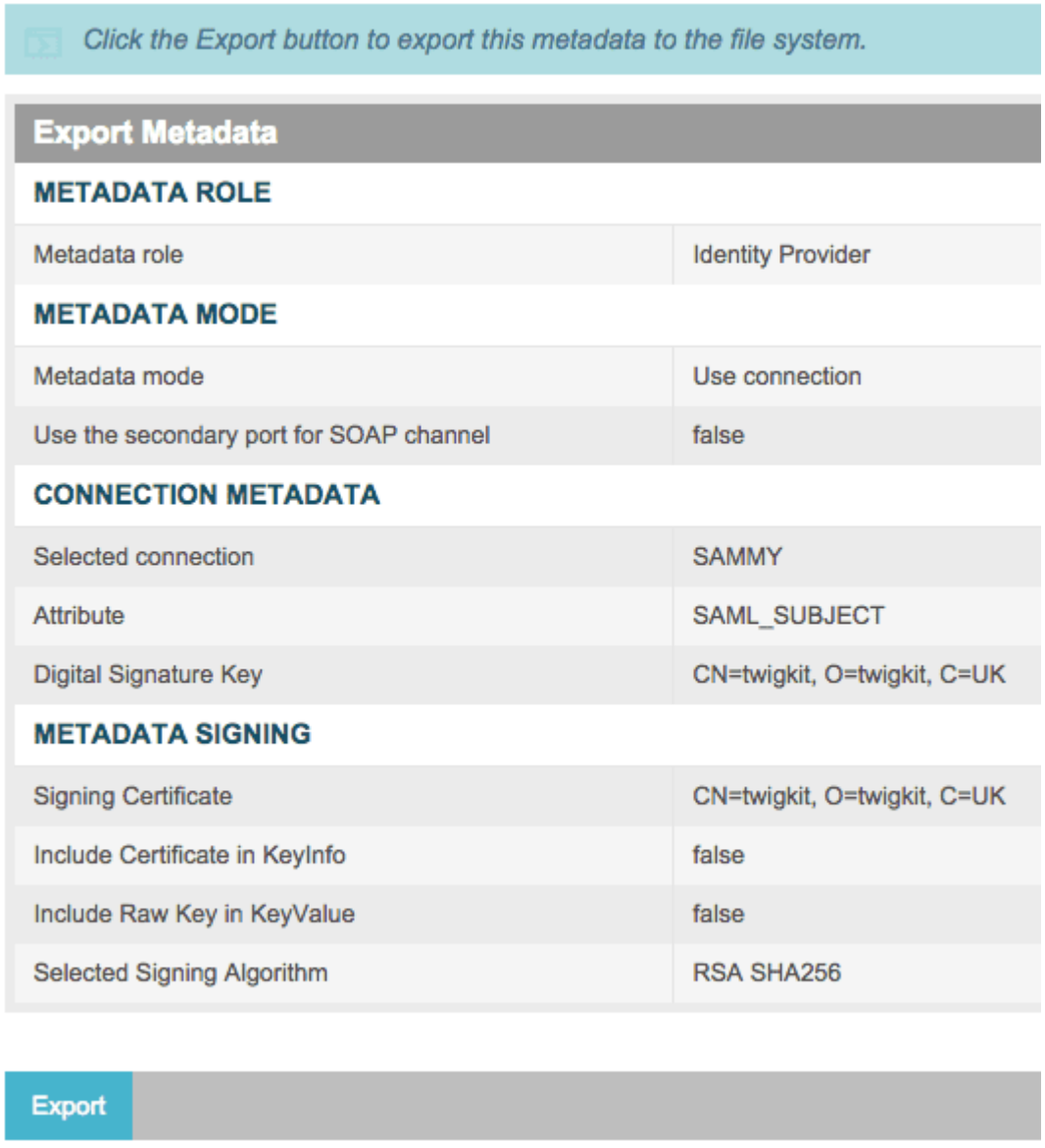
In the PingFederate admin UI, choose **Administrative functions > Metadata Export**.











5. Add the IdP security certificates

Depending on the security certificates used, different configurations are required. Often the IdP metadata XML file supplies the correct certificate - in which case, you do not need any extra configuration.

However in many cases you must import the public key for the certificate used by the IdP (for SSL encryption and to sign SAML messages) to the keystore used by the application. By default, this keystore is configured to be located at `src/main/resources/samlKeystore.jks`.

In the case of SP initiated SSO, you will also must upload the keys used by the SP to sign SAML requests to the IdP system's trusted certificate store.

6. Ensure the IdP is configured for integration with the SP (the Appkit application)

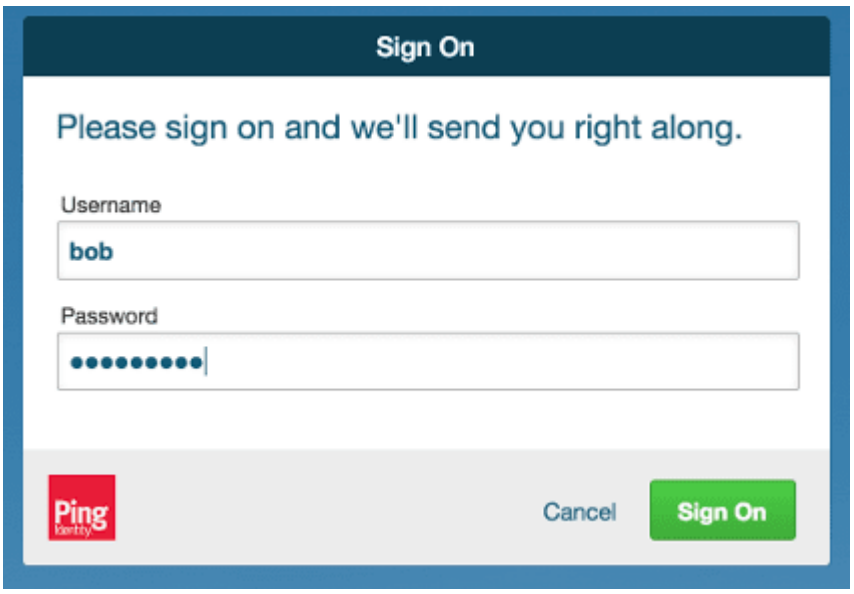
In some integration scenarios, the IdP will have already set these properties and might require you to configure the Appkit application accordingly. However, the application's SSO consumer and SLO service URLs should be defined by the SP and provided to the IdP for them to configure within the IdP system.

It is possible to export the SP metadata as XML from a running application, however, because IdP implementations can vary, we recommend that this is configured in the IdP system manually using these settings:

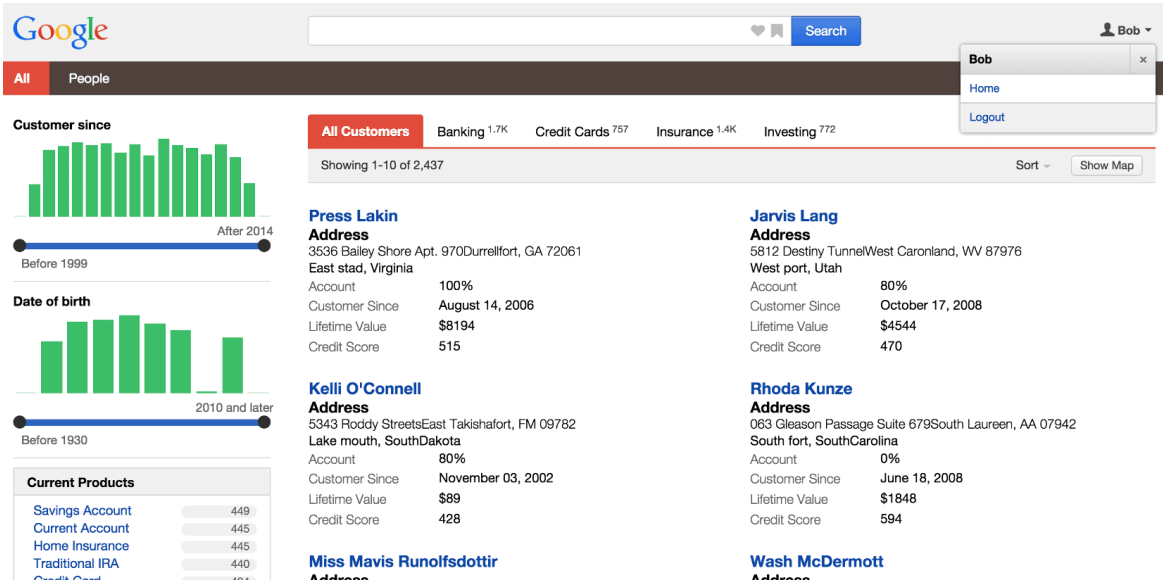
SP (partner) entity ID: As configured in `saml.conf` . Protocol: `SAML 2.0` Browser SSO: `true` IdP initiated SSO & SLO: `true` Assertion consumer service endpoint URL: (POST or Artifact) `http://your-twigkit-host/your-twigkit-app/saml/SSO` SLO service URL: (POST) `http://your-twigkit-host/your-twigkit-app/logout/` Artifact resolver location: `http://your-twigkit-host/your-twigkit-app/saml/SSO/`

Testing the SAML provided SSO

1. Run the Appkit application and navigate to the root of the application context (for example, `http://localhost:8080/`).
   You should be presented with the IdP's login form; see the example below:



2. Enter valid credentials for a user in the IdP's set of users.
   You will be logged into Appkit and should see the main applications search page, including the user's details:



When logout is selected, the user should be logged out using SLO and presented with the IdP's configured 'successful logout' page:

If you navigate to the root of the application again (or any application backed by this IdP) you should then be presented with the login form.

If this authentication lifecycle behaves as shown above, then the SAML authentication is configured and working. Obviously, the IdP screens will vary between providers and many providers let you customize these pages to fit the branding of the organisation.

Removing SAML form-based authentication

If the IdP does not use form based authentication as a fallback mechanism, remove the `defaultFailureUrl` property from the `unauthenticatedRedirectHandler` bean.

```
<beans:property name="defaultFailureUrl" value="${slo-
url:https://localhost:9031/idp/startSSO.ping?PartnerSpId=SAMMY}"/>
```

Scale a Fusion 4.x Cluster

How-to

> Compatible versions: 4.2
>
> Works with: *Fusion Server*

To scale a Fusion cluster, you can add new Fusion nodes, add new dedicated indexing nodes, or move Fusion to new nodes.

Adding a new Fusion node to an existing cluster

Follow these steps to add a new node to an existing Fusion cluster:

> **Note**
>
> If you are running embedded zookeepers from Fusion in an ensemble for your cluster, ensure that the you are running an odd number of zookeepers for your environment after addition of the new node.

1. Stop Fusion on all nodes in the cluster.
   This ensures that there is no data inconsistency between the instances when the new node comes up.
2. Decompress the new copy of Fusion and place it in the desired directory.
3. Configure the `fusion.cors` (`fusion.properties` in Fusion 4.x) file to match your requirements. If you will also run the embedded ZooKeeper, add the new node's IP/hostname and port to the `default.zk.connect` string and copy this change to all other instances in your cluster. Configure the memory and other JVM options for the Fusion modules, then save the file.
4. *If embedded ZooKeepers are used in your cluster and you intend to start ZooKeeper on this node*, then follow the additional steps below. If not, then you are ready to start all nodes in the cluster.
   a. Copy the `$FUSION_HOME/conf/zookeeper/zoo.cfg` file from one of the existing nodes to the new node, overwriting the default file.
   b. Add the entry for the new ZooKeeper to the server list in the `zoo.cfg` file.
   The entry format is `server.x=IP:port:port`. For example, if this is the 5th node, then the new entry in `zoo.cfg` is `server.5=IP:port:port`.
   c. Create a `zookeeper` folder under `$FUSION_HOME/data`.
   d. Create a new `myid` file in `$FUSION_HOME/data/zookeeper`.
   The contents of this file must be an integer equal to the number of the new ZooKeeper node in the ensemble. For example, if the new node will be the 5th node in your ZooKeeper ensemble, then the `myid` file should contain the value "5".
   e. Copy the `$FUSION_HOME/data/zookeeper/version-2` directory from one of the existing nodes to the new node, overwriting the default directory.
   f. Modify the connect string for the default search cluster:
      i. Start ZooKeeper on all nodes.
      Next, you will need the `zkcli` script, located in `$FUSION_HOME/apps/solr-dist/server/scripts/cloud-scripts`. Use `zkcli.sh` for Unix or `zkcli.bat` for Windows. The examples below use the Unix script.
      ii. Download the default search cluster file:

```
./zkcli.sh -z <zk1>:<port1>,<zk2>:<port2>,... -cmd
getfile <path_to_default_cluster>
<path_to_dump_file>.json
```

The path will differ depending on your Fusion version:

- 2.4.x: `/lucid/search-clusters/default`
- 3.x: `/lwfusion/<fusion_version>/core/search-clusters/default`

For example:

```
./zkcli.sh -z localhost:9983 -cmd getfile
/lwfusion/3.1.2/core/search-clusters/default
default_search_cluster.json
```

iii. In the downloaded JSON file, find the `connectString` key and replace the old IP value with the IP of the new Fusion node.

> **Note**
>
> Be sure to specify the chroot if your cluster is configured to use it.

For example:

```
{ "id" : "default", "connectString" :
"localhost:9983/lwfusion/3.1.2/solr",
"zkClientTimeout" : 30000, "zkConnectTimeout" :
60000, "cloud" : true, "bufferFlushInterval" : 1000,
"bufferSize" : 100, "concurrency" : 10, "authConfig"
: { "authType" : "none" }, "validateCluster" : true
}
```

iv. Upload the modified search cluster file:

```
./zkcli.sh -z <zk1>:<port1>,<zk2>:<port2>,... -cmd
putfile <path_to_default_cluster>
<path_to_dump_file>.json
```

For example:

```
./zkcli.sh -z localhost:9983 -cmd putfile
/lwfusion/3.1.2/core/search-clusters/default
default_search_cluster.json
```

5. Start Fusion on all nodes in the cluster.

Adding an indexing node to a Fusion cluster

If you need more capacity for indexing, you can add nodes dedicated to indexing. To do this, you add a new Fusion node, configure it to only run the Solr service, then allocate replicas of your collections to the new node.

1. Install the Fusion package on the new node.
2. Edit `fusion.cors` (`fusion.properties` in Fusion 4.x) as follows:
   a. Edit `group.default` to include only the Solr service.
   For example, change

```
group.default = zookeeper, solr, api, connectors-rpc,
connectors-classic, admin-ui, proxy, webapps
```

   to

```
group.default = solr
```

   b. Uncomment `default.zk.connect` and point it to the cluster's ZooKeeper instances.
   For example, change

```
# default.zk.connect = localhost:9983
```

   to

```
default.zk.connect =
172.23.1.1:9983,172.23.1.2:9983,172.23.1.3:9983/solr
```

c. Save the file.

3. Start Fusion on the new node:

```
bin/fusion start
```

At this point, the new node is added to the cluster. No indexing takes place on the new node yet.

4. Allocate one or more collection replicas to this node:
   a. Open the Solr UI at `http://<new-node-hostname>:8983/solr/`.
   b. Click **Collections**
   c. Select a collection to replicate on the new indexing node.
   d. Click **Shard: shard1** (or another shard if you have more than one for this collection).
   e. Click **add replica**.
   f. From the **Node** drop-down list, select the new node.
   g. Click **Create Replica**.
   To verify that the collection is being replicated, you can click **Cloud** and view the replicas.

Tip

Consider whether secondary collections should also be replicated. For example, consider adding replicas for the signals and aggregations collections associated with the main collections that you are replicating.

Moving Fusion from one node to another

1. Stop Fusion on all nodes in the cluster.
   This ensures that there is no data inconsistency between the instances when the new node comes up.
2. Compress the Fusion node you wish to move.
3. Copy the compressed file to the destination.
4. Starting with step 2, follow the instructions above for adding a new node.

Scale Spark Aggregations for Fusion 4.x

How-to

> Compatible versions: 4.2
>
> Works with: *Fusion Server*

Consider the process of running a simple aggregation on 130M signals. For an aggregation of this size, it helps to tune your Spark configuration.

Speed up tasks and avoid timeouts

One of the most common issues encountered when running an aggregation job over a large signals data set is task timeout issues in Stage 2 (`foreachPartition`). This is typically due to slowness indexing aggregated jobs back into Solr or due to JavaScript functions.

The solution is to increase the number of partitions of the aggregated RDD (the input to Stage 2). By default, Fusion uses 25 partitions. Here, we increase the number of partitions to 72. Set these configuration properties:

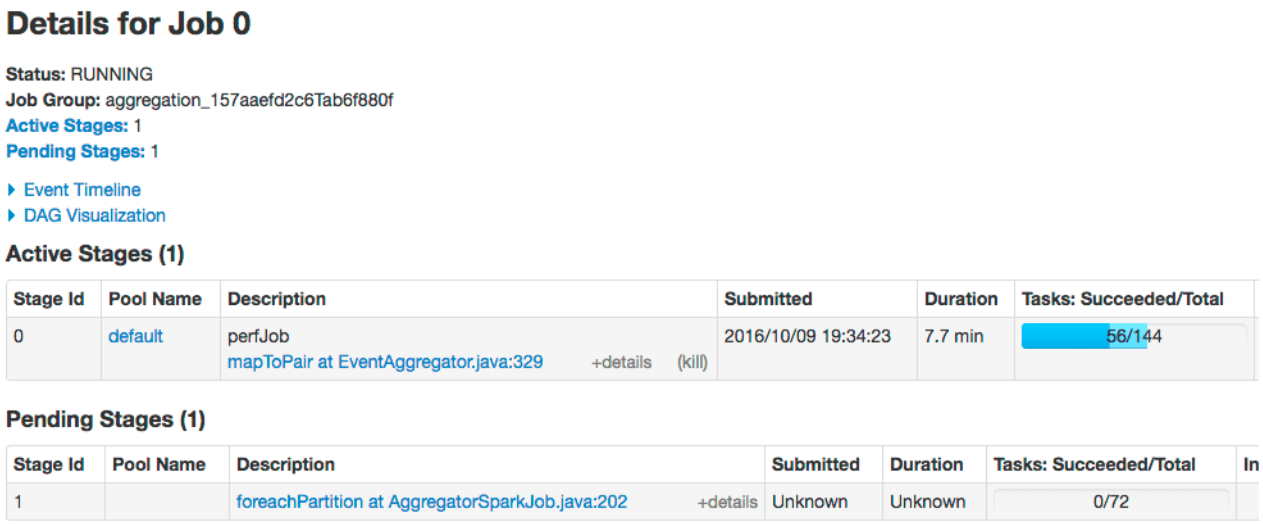
- `spark.default.parallelism` .* Default number of partitions in RDDs returned by transformations like `join` , `reduceByKey` , and `parallelize` when not specified by the user:

```
curl -u USERNAME:PASSWORD -H 'Content-type:application/json' -
X PUT -d '72'
"https://FUSION_HOST:6764/api/configurations/spark.default.par
allelism"
```

- `spark.sql.shuffle.partitions` .* Number of partitions to use when shuffling data for joins or aggregations.

```
curl -u USERNAME:PASSWORD -H 'Content-type:application/json' -
X PUT -d '72'
"https://FUSION_HOST:6764/api/configurations/spark.sql.shuffle
.partitions"
```

After making these changes, the `foreachPartition` stage of the job will use 72 partitions:



Increase rows read per page

You can increase the number of rows read per page (the default is 10000) by passing the rows parameter when starting your aggregation job; for example:

```
curl -u USERNAME:PASSWORD -XPOST
"https://FUSION_HOST:6764/api/aggregator/jobs/perf_signals/perfJob?
rows=20000&sync=false"
```

For example, we were able to read 130M signals from Solr in 18 minutes at ~120K rows/sec using rows=20000 vs. 21 minutes using the default 10000.

Improve job performance

You can increase performance when reading input data from Solr using the `splits_per_shard` read option, which defaults to 4. This configuration setting governs how many Spark tasks can read from Solr concurrently. Increasing this value can improve job performance but also adds load on Solr.

Schedule a datasource job

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

To schedule a datasource job, complete the following:

1. Use the *Index Workbench* to set up a datasource.
2. Click **System > Scheduler**.
3. Select the datasource from the job list.
4. Click **New Schedule**.
5. Select and configure a trigger:
    - **After Another Job Completes**
      Enter the job ID and job result that trigger this one.
    - **Cron String**
      Enter a Quartz cron expression. See the Quartz documentation for details.
    - **Start + Interval**
      Enter a start date/time, an interval, and the interval units.

Schedule a Datasource Job

How-to

> Compatible versions: 4.2, 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion Server*, *Fusion*

1. *Create a datasource*.
2. Navigate to **System** > **Scheduler**.
3. Select the datasource from the job list.
4. Click **New Schedule**.
5. Select and configure a trigger:
    - **After Another Job Completes**
      Enter the job ID and job result that trigger this one.
    - **Cron String**
      Enter a Quartz cron expression. See the Quartz documentation for details.
    - **Start + Interval**
      Enter a start date/time, an interval, and the interval units.

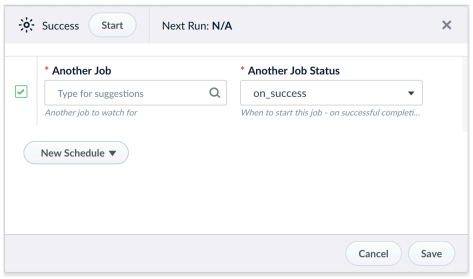
Schedule a job

How-to

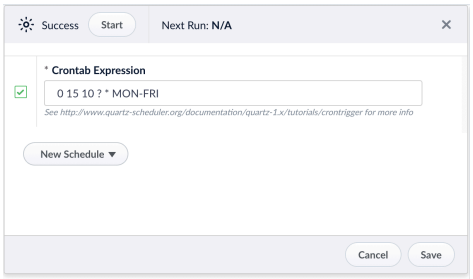
> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

How to schedule a job

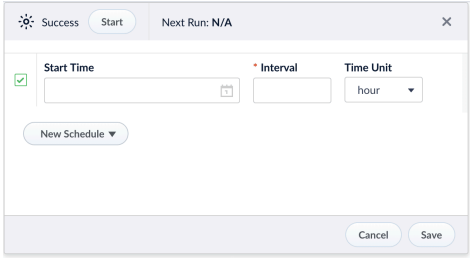
1. In the Managed Fusion workspace, navigate to **Collections > Scheduler**.
2. Select the job from the job list.
3. Click **Run**.
4. Click **New Schedule**.
5. Select a job trigger:
    - **After Another Job Completes**
      Enter the job ID and job result that trigger this one.



    - **Cron String**
      Enter a Quartz cron expression, using UTC time unless `default.timezone` is set to another time zone in `fusion.cors`. See the Quartz documentation for additional details.



    - **Start + Interval**
      Enter a start date/time, an interval, and the interval units. Include seconds in the start time.



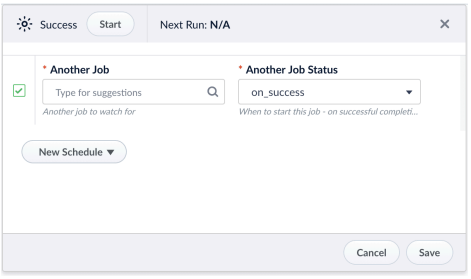
Schedule a Job

How-to

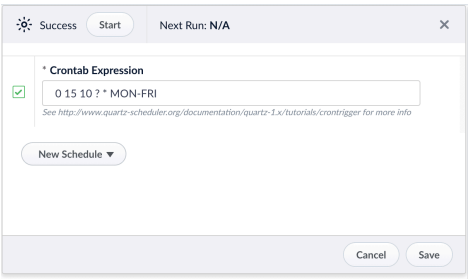
> Compatible versions: 4.2, 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion Server*, *Fusion*

How to schedule a job

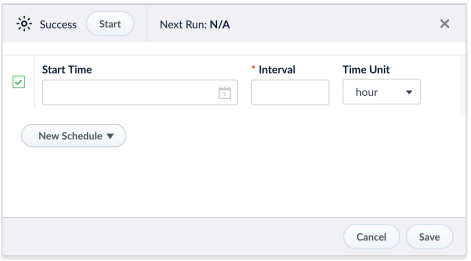
1. In the Fusion workspace, navigate to **Collections > Scheduler**.
2. Select the job from the job list.
3. Click **Run**.
4. Click **New Schedule**.
5. Select a job trigger:
    - **After Another Job Completes**
      Enter the job ID and job result that trigger this one.



    - **Cron String**
      Enter a Quartz cron expression, using UTC time unless `default.timezone` is set to another time zone in `fusion.cors` (`fusion.properties` in Fusion 4.x). See the Quartz documentation for additional details.



    - **Start + Interval**
      Enter a start date/time, an interval, and the interval units. Include seconds in the start time.



Secure Communication with a SolrCloud Cluster

How-to

> Compatible versions: 4.2
>
> Works with: *Fusion Server*

You can configure Fusion and an external SolrCloud cluster so that communication between Fusion and the SolrCloud cluster is secured. Use either Kerberos or basic authentication. You can secure communication for both the default search cluster and for other SolrCloud clusters.

Note: Securing communication between Fusion and a *bundled* default search cluster is not supported.

The required steps differ. These are the high-level steps. Detailed steps follow.

- **Default search cluster.** Define configuration parameters for bootstrapping Fusion, and then bootstrap Fusion.
- **Other SolrCloud clusters.** In the Fusion UI, add the external SolrCloud cluster.

Default Search Cluster

If your default search cluster is in an *external* SolrCloud cluster, then you can secure the cluster with Kerberos or basic authentication, and then configure Fusion to communicate securely with the cluster.

Prerequisite

Prerequisite: Secure the default search cluster. Use either Solr's Basic Authentication Plugin or Kerberos Authentication Plugin .

Don't start Fusion yet. Below, you will define bootstrap properties, and then bootstrap Fusion.

Configure and Bootstrap Fusion

1. Create a `.properties` file for the initial bootstrap of Fusion. Place the file outside of the Fusion installation, for example, in `/tmp`. You will delete the file at the end of this procedure:

```
touch /tmp/fusion-bootstrap.properties
```

2. Edit the `fusion-bootstrap.properties` file to define Fusion initial-bootstrap configuration properties. Example strings are in bold italics. Replace those with your own values.

Caution

Consult with your Kerberos administrator about the correct configuration properties.

- **Kerberos authentication** – Specify the authentication type (`kerberos`), the Kerberos principal, and the Kerberos keytab file:

```
default-search-cluster.auth-type=kerberos default-
search-cluster.auth-principal=fusion@MYORG.ORG default-
search-cluster.auth-keytab=/path-to-file/keytab.kt
```

- Basic authentication: – Specify the authentication type (`basic`), the username of the Solr user to use for authentication, and the password of that user:

```
default-search-cluster.auth-type=basic default-search-
cluster.auth-user=admin default-search-cluster.auth-
password=admin-password
```

The Solr user must be the admin user or a different user with full administrative privileges.

Note

Fusion doesn't support Solr authorization plugins.

3. Edit the fusion.cors file:
   a. Uncomment and change the value of this property to point to an external ZooKeeper:

```
# default.zk.connect = localhost:9983
```

   b. Uncomment and change the value of this property to use an external SolrCloud cluster:

```
# default.solrZk.connect = localhost:2181/solr-zk-
namespace
```

   c. Remove `zookeeper` and `solr` from the `group.default` property:

```
group.default = api, connectors, ui
```

   d. Add a configuration property for the path to the initial-bootstrap properties file:

```
initial-bootstrap-properties-path = /tmp/fusion-
bootstrap.properties
```

4. Change your working directory to the directory that contains the Fusion binaries:

```
$ cd ~/{path_to}https://FUSION_HOST:FUSION_PORT/bin
```

5. Bootstrap Fusion:

```
./fusion start
```

6. After Fusion starts:
    a. Delete the initial-bootstrap properties file:

```
$ rm /tmp/fusion-bootstrap.properties
```

    b. Edit the `fusion.cors` (`fusion.properties` in Fusion 4.x) file to remove the entry for the initial-bootstrap properties file:

```
initial-bootstrap-properties-path = /tmp/fusion-
bootstrap.properties
```

Other SolrCloud Cluster

You can secure an *external* SolrCloud cluster with Kerberos or basic authentication, and then configure Fusion to communicate securely with the cluster.

Prerequisite

Prerequisite: Secure the default search cluster. Use either Solr's *Basic Authentication Plugin* or *Kerberos Authentication Plugin*.

Add the secure SolrCloud cluster in the Fusion UI (Basic Auth)

1. log in to the Fusion UI as the user `admin`.
2. Click **System > Home > System > Solr Clusters > New Solr Cluster**.
3. Click **Advanced**.
4. Specify the required values ID and Connect String. Under Solr Cluster Authentication, check **include**. Choose Authentication Type **basic**, and specify a username and password for authentication.
5. Click **Save new**.

Add the secure SolrCloud cluster in the Fusion UI (Kerberos)

1. log in to the Fusion UI as the user `admin`.
2. Click **System > Home > System > Solr Clusters > New Solr Cluster**.
3. Click **Advanced**.
4. Specify the required values ID and Connect String. Under Solr Cluster Authentication, check **include**. Choose Authentication Type **kerberos**, and specify a Kerberos keytab file and Kerberos principal for authentication.
5. Click **Save new**.

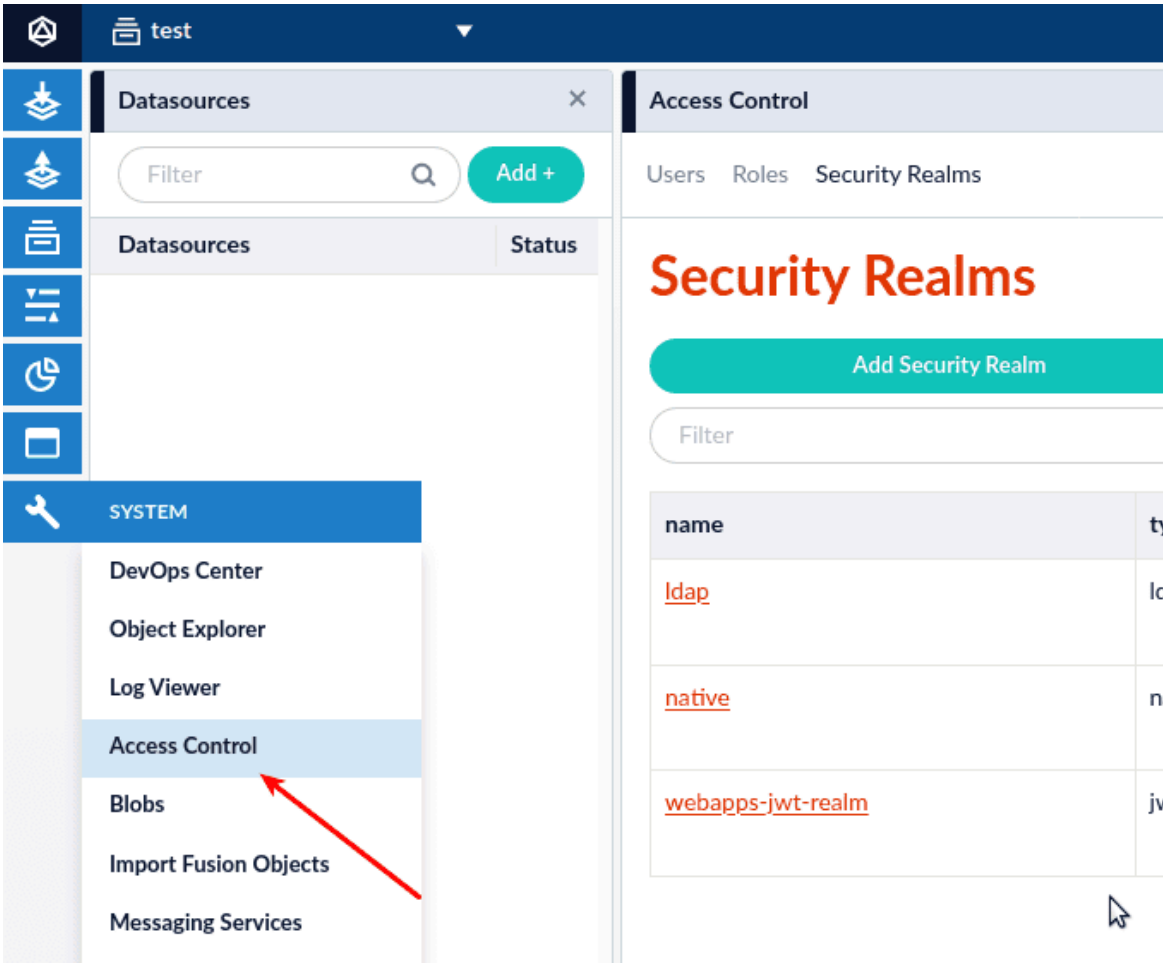
Security LDAP Realm that Connects to Active Directory

How-to

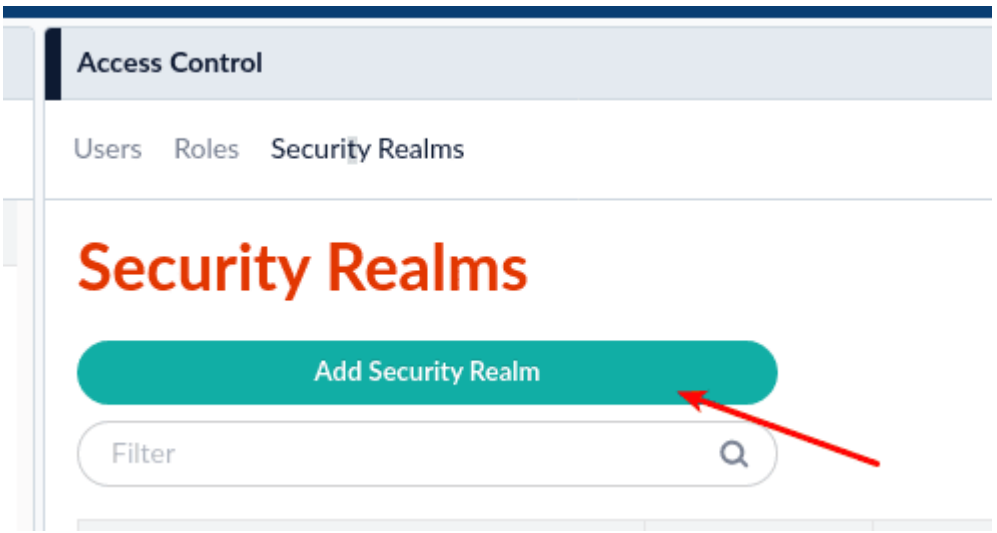
> Compatible versions: 4.2
>
> Works with: *Fusion Server*

Configure Fusion for a Security LDAP Realm that Connects to Active Directory

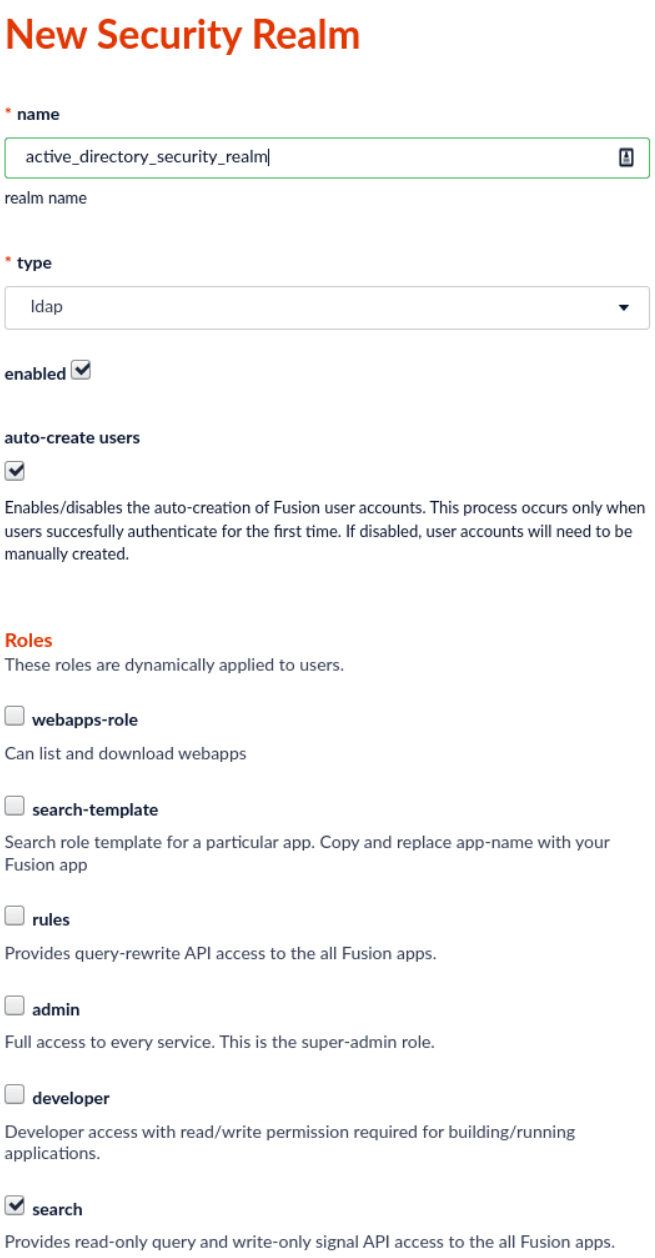
1. Log in as an admin to Fusion and go to **Security Realms**.



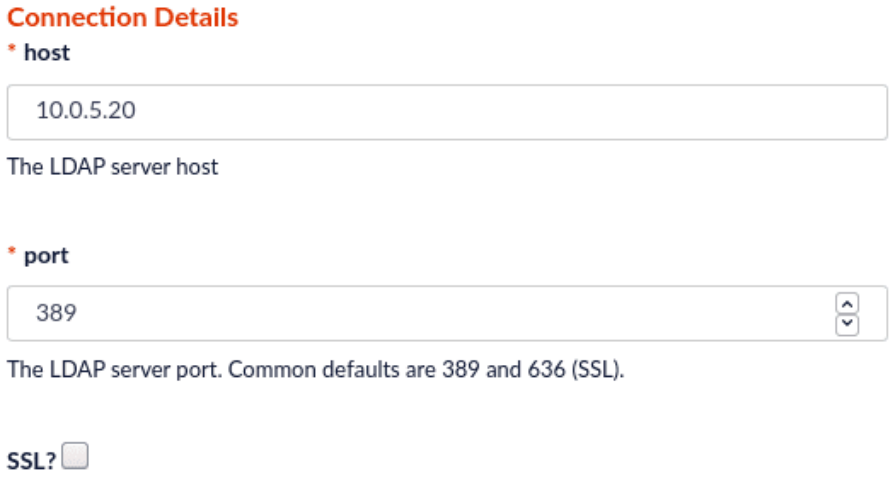
2. Go to the Security Realms tab and click **Add Security Realm**.



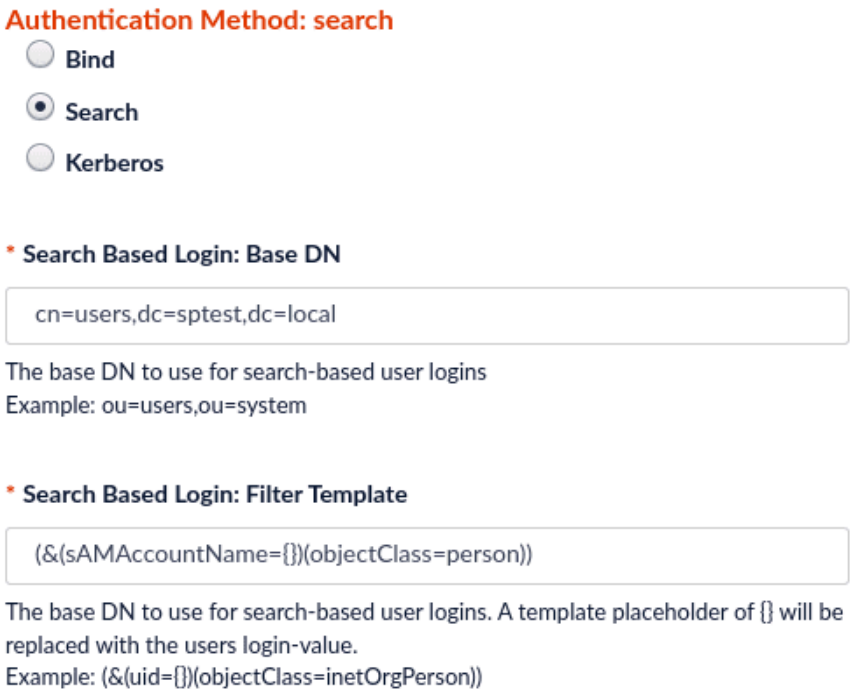
3. Give it a name and choose **type: ldap**.
    - If you want an AD user to be automatically created in Fusion when they are seen for the first time, leave “Auto-create users” checked. Typically, you will leave this checked. Otherwise, you have to preload Fusion with users prior to them logging in.
    - Specify what Roles you want users to have by default when logging in to this realm.



4. Specify your AD server hostname. Choose port 389 for non-SSL, and 636 for SSL.

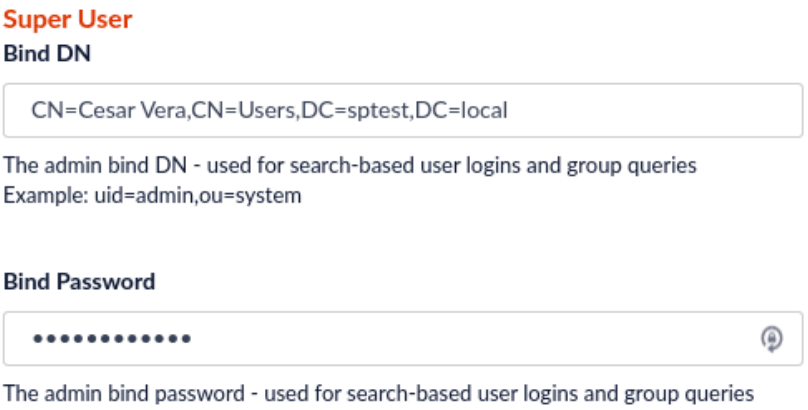


5. For **Authentication Method: search**, choose **Search** from the 3 options.
   a. In the **Search Based Login: Base DN** field, specify the DN of the lowest place in the AD tree where your users will need to be searched for. in my example, it is `cn=users,dc=sptest,dc=local`.
   b. For the **Search Based Login: Filter Template**, use `(&(sAMAccountName={})(objectClass=person))`.

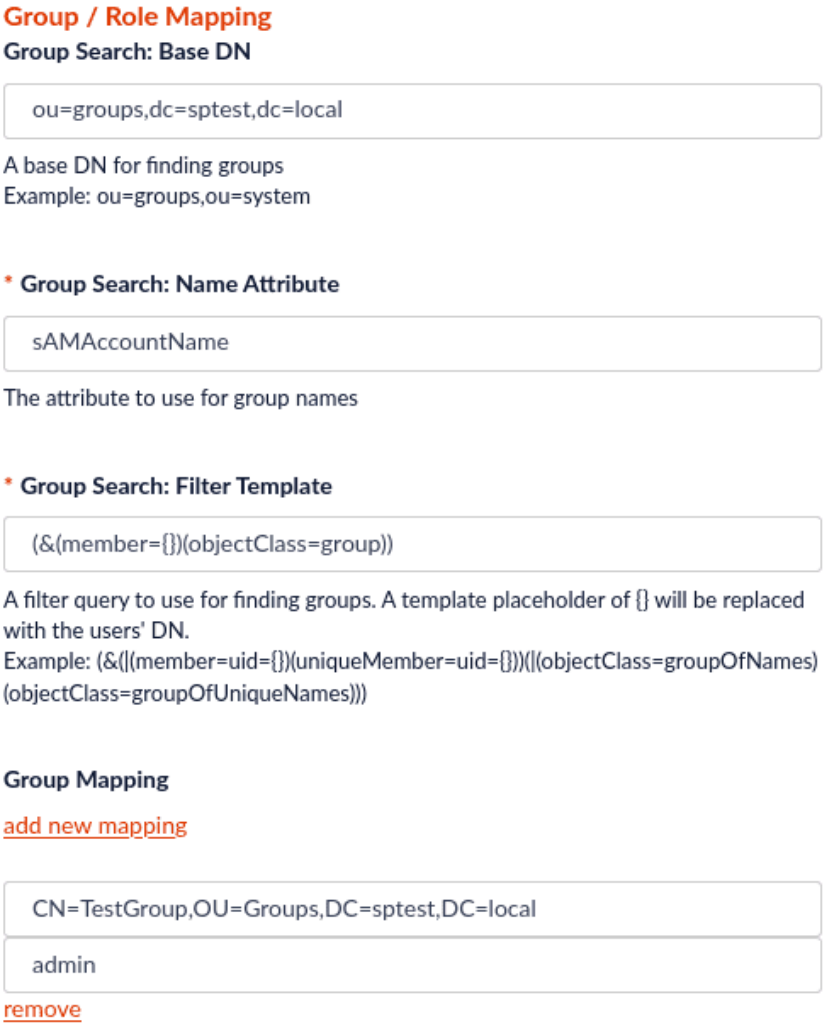


6. For **Super User - Bind DN** enter the DN of a user with access to crawl and enter that user's password.

> **Note**
>
> This says "Super User", but the AD account does not need to be an Administrator. Your user only needs to be able to read group member attributes.



7. Next is the **Group / Role Mapping**.
    a. For **Group Search: Base DN**, similar to what you did for the user DN, enter the lowest level of the AD tree where applicable groups can be found.
    b. For **Group Search: Name Attribute**, enter `sAMAccountName`.
    c. For **Group Search: Filter Template**, enter `(&(member={})(objectClass=group))`.
    d. For **Group Mapping**, enter a mapping of each Fusion role name to the DN of a group you want to be in that role.
    Example:



Creating a Delegation of Control of the bind user

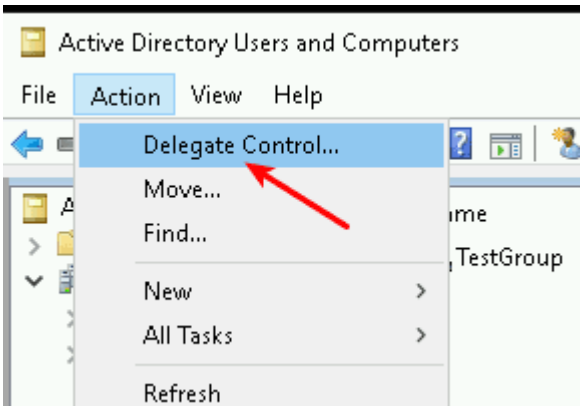
The LDAP Realm needs to be able to get a user's groups in order to do the Group Mapping.

By default, Active Directory allows all Authenticated Users to list group memberships. So chances are, everything will work.

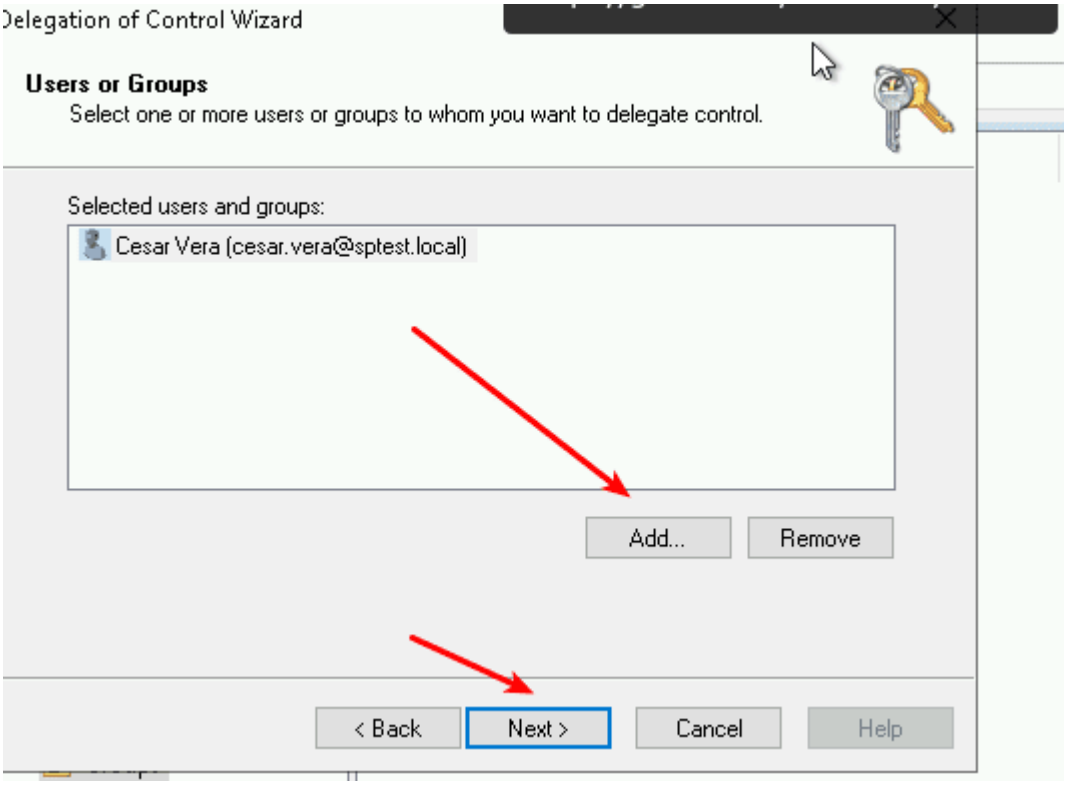
But, if your security is tightened in your environment, the user you choose in **Super User** section may not have access to list group memberships. This will result in your users having no roles because the role lookup will not be able to connect your user's AD groups with their Fusion roles.

In this case, you may need to delegate control of this account to be able to Read the tree.

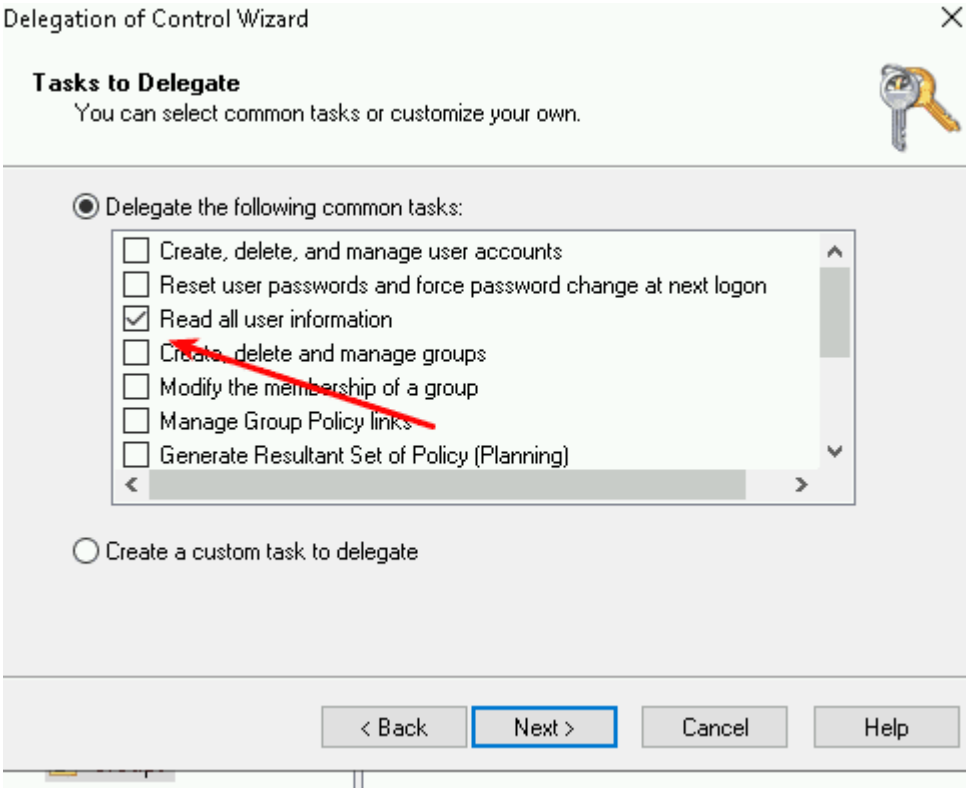
1. Go to Active Directory Users and Computers → Action → Delegate Control.



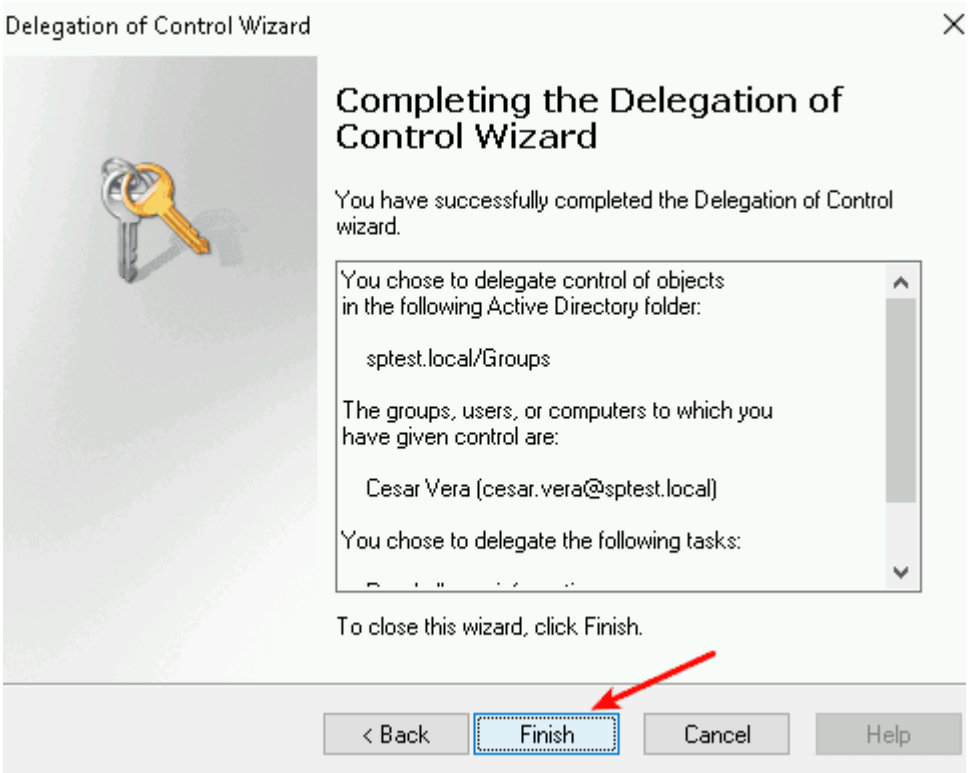
2. Enter the user:



3. Add the **Read all user information** checkbox.


Delegation of Control Wizard
Tasks to Delegate
You can select common tasks or customize your own.
Delegate the following common tasks:
Create, delete, and manage user accounts
Reset user passwords and force password change at next logon
Read all user information
Create, delete and manage groups
Modify the membership of a group
Manage Group Policy links
Generate Resultant Set of Policy (Planning)
Create a custom task to delegate
< Back
Next >
Cancel
Help

4. Finish.


Delegation of Control Wizard
Completing the Delegation of Control Wizard
You have successfully completed the Delegation of Control wizard.
You chose to delegate control of objects in the following Active Directory folder:
sptest.local/Groups
The groups, users, or computers to which you have given control are:
Cesar Vera (cesar.vera@sptest.local)
You chose to delegate the following tasks:
To close this wizard, click Finish.
< Back
Finish
Cancel
Help

See a Smart Answers (QnA) Job's Status and Output

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion*

This topic explains how you can track the running steps of the following jobs:

- Fusion 5.3 and later:
  - *Smart Answers Supervised Training*
  - *Smart Answers Coldstart Training*
  - *Smart Answers Evaluate Pipeline*
- Fusion 5.1 and 5.2:
  - *QnA Supervised Training*
  - *QnA Coldstart Training*
  - *Evaluate QnA Pipeline*

The ML Model service logs provide information on data pre-processing, training steps, evaluations, and model generation for those jobs. Follow the instructions below to access the logs.

1. Find the name of the ML Model service pod:

   ```
   kubectl get pods -n <your-namespace> | grep ml-model-service-ui
   ```

   The pod name looks like `<namespace>-ml-model-service-ui-<hash>-<random>`, as in `f5-ml-model-service-ui-547dd78d6-p9d6q`.
2. Set up port forwarding:

   ```
   kubectl port-forward <namespace>-ml-model-service-ui-<hash>-
   <random> 8001:8001 -n <your-namespace>
   ```

3. In the Fusion UI, navigate to **Collections** > **Jobs**.
4. Select your Smart Answers (QnA) model training job.
5. Click **Job History**.
6. Find the ML Model workflow ID.
7. In the ML Model service UI, locate instances of the ID in the logs.

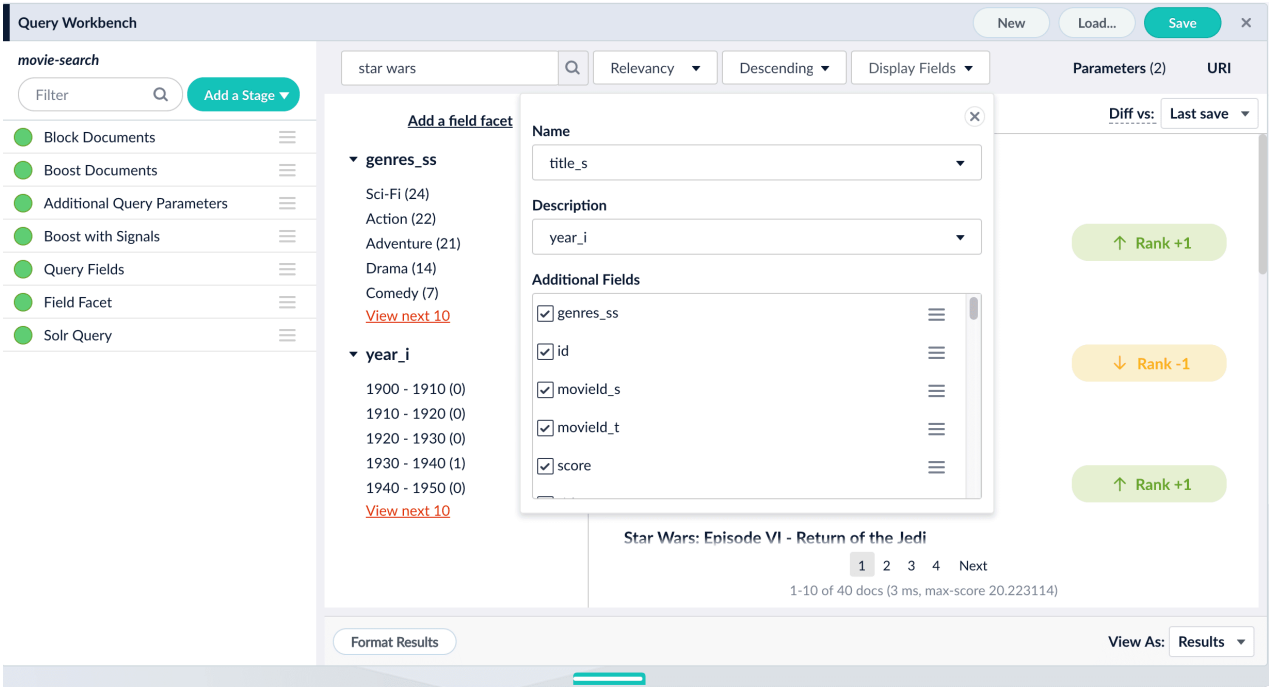
Select Fields to Display in the Query Workbench

How-to

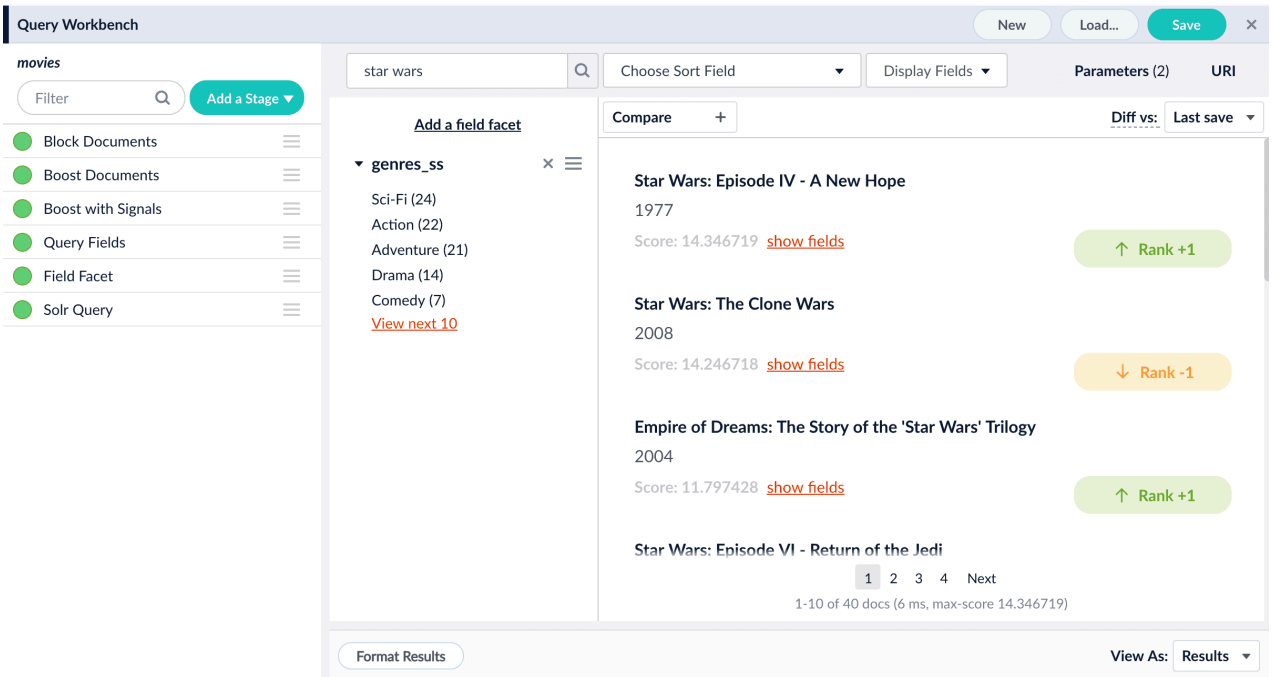
> Compatible versions: 4.2, 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion Server*, *Fusion*

While the *Query Fields stage* configures which fields match against incoming queries, the search application itself determines which fields to display in the search results.

For example, the search results below use the `title_s` field as the document name and the `year_i` field as the document description:



Adding `year_i` as the description removed it from the list of field facets and now we see the year of release for each movie title in the search results:



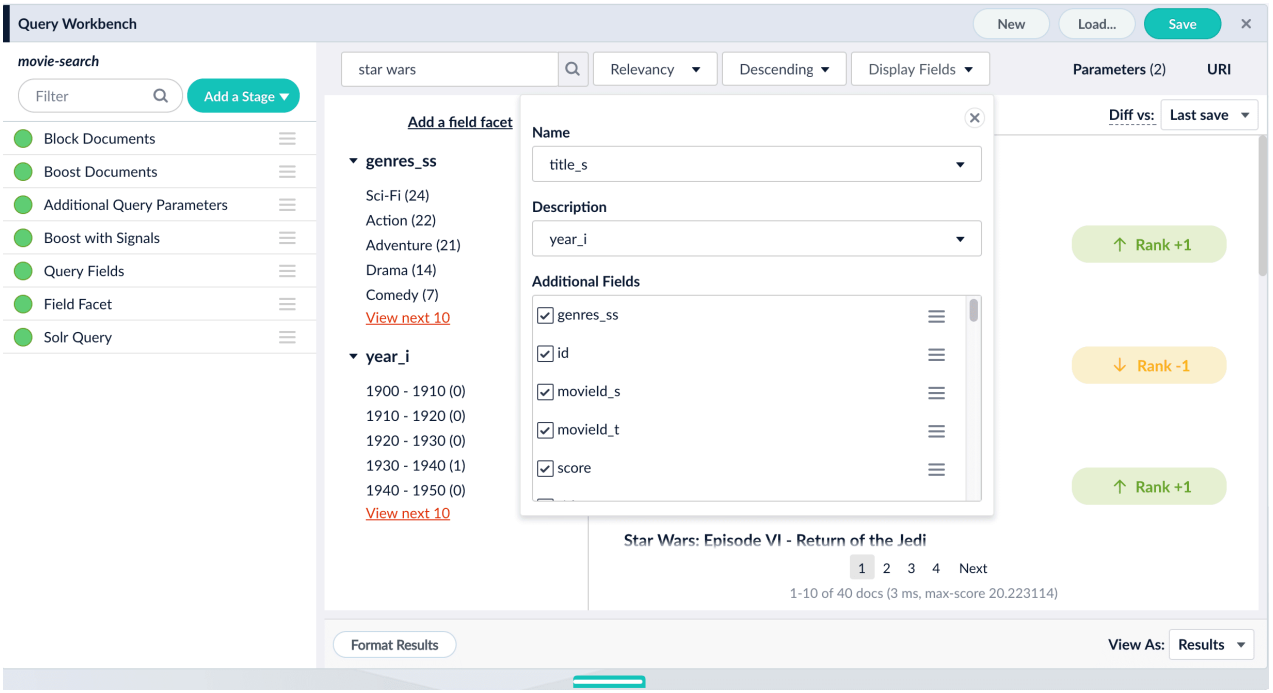
Select Fields to Display in the Query Workbench

How-to

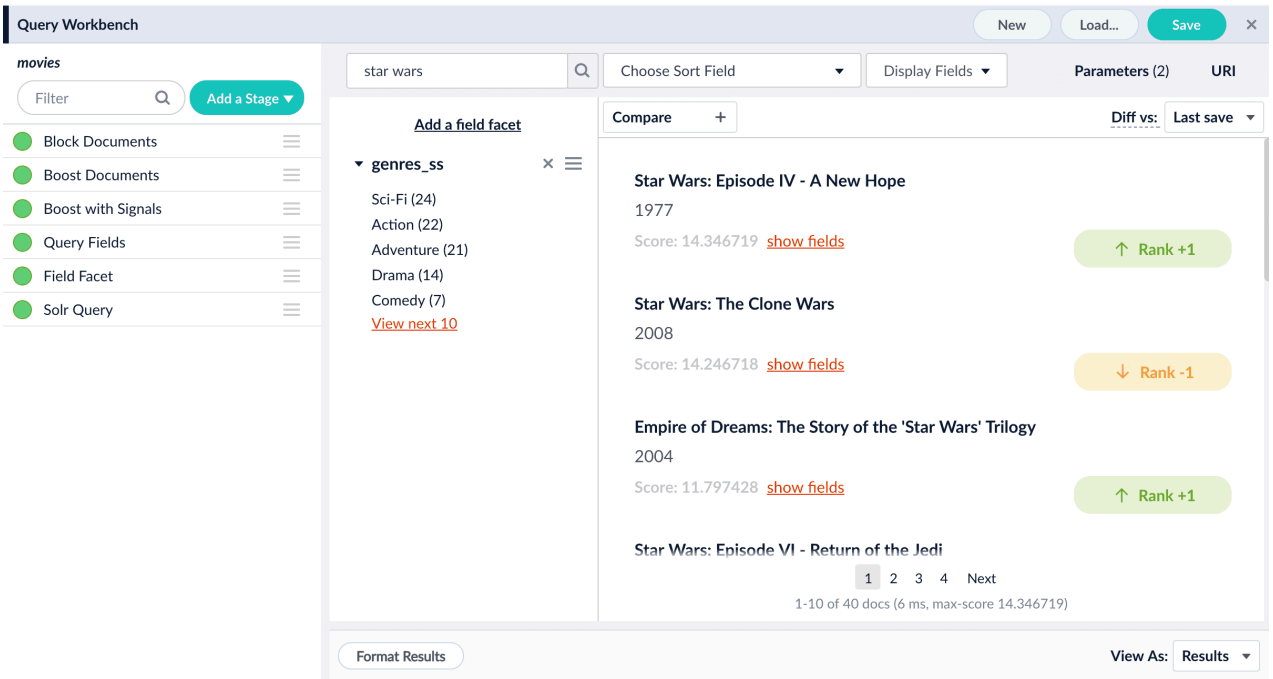
> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

While the *Query Fields stage* configures which fields match against incoming queries, the search application itself determines which fields to display in the search results.

For example, the search results below use the `title_s` field as the document name and the `year_i` field as the document description:



Adding `year_i` as the description removed it from the list of field facets and now we see the year of release for each movie title in the search results:



Self-Contained Deployment

How-to

> Compatible versions: 4.2
>
> Works with: *App Studio*

This deployment type has an embedded Tomcat server and is ready to deploy.

Compiling a self-contained application

To compile the application, run the `app-studio` script with the `dist` target, like this:

Unix:

```
./app-studio dist
```

Windows:

```
app-studio.bat dist
```

You can find the output in the `dist` directory:

```
search-app-project.zip search-app-standalone/ README.md app/ build/
config/ keystore.jks lib search-app-start.bat search-app-stop.bat
search-app-stop.ps1 search-app.sh
```

The `search-app-project.zip` is a snapshot of your project at the time you compiled the application. The `search-app-standalone/` directory is the self-contained application.

Self-Executing Deployment

How-to

> Compatible versions: 4.2
>
> Works with: *App Studio*

In this deployment type, your search interface runs within the Java runtime environment (JRE).

Compiling a .jar file

A *downloaded project* comes with an `app-studio` script that creates a `.jar` file.

1. In a shell window, switch to your project directory:

```
cd /path/to/Project_Directory
```

2. Run the following command:

```
./app-studio package
```

The script creates the following file in the `dist` directory of your project:
`search-app-VERSION.x-y.y.y.jar`

This is your self-executing application file.

Launching a self-executing deployment

```
java -jar search-app-VERSION.x-y.y.y.jar
```

Send Your Initial Data Stream

How-to

> Works with: *Never Null*

Once your data is formatted to conform to the requirements for *product data* and *signals data*, you need to upload it to a secure data bucket and provide access for Lucidworks.

How to get your data into Never Null

1. Confirm that your product data is formatted according to the *product data requirements*.
2. Confirm that your signals data is formatted according to the *signals data requirements*.
3. Upload it to a secure data bucket, such as GCS, S3, or Azure Blob Storage.
4. Provide Lucidworks with access to the data bucket.

Lucidworks will inform you when the initial model training is complete and your Never Null data is ready to *query*.

Plan on periodically *updating your Never Null data* to incorporate your latest product and signals data using a similar method.

Set Up a Pre-Trained Cold Start Model for Smart Answers

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion*

Lucidworks provides these pre-trained cold start models for *Smart Answers*:

- `qna-coldstart-large` -this is a large model trained on variety of corpuses and tasks.
- `qna-coldstart-multilingual` -covers 16 languages. List of supported languages: Arabic, Chinese-simplified, Chinese-traditional, English, French, German, Italian, Japanese, Korean, Dutch, Polish, Portuguese, Spanish, Thai, Turkish, Russian.

When you use these models, you do not need to run the model training job. Instead, you run a job that deploys the model into Fusion. The *Create Seldon Core Model Deployment job* deploys your model as a Docker image in Kubernetes, which you can scale up or down like other Fusion services.

These models are a good basis for a cold start solution if your data does not contain much domain-specific terminology. Otherwise, consider *training a model* using your existing content.

Tip

Dimension size of vectors for both models is **512**. You might need this information when creating collections in Milvus.

1. Deploy a pre-trained cold-start model into Fusion

The pre-trained cold-start models are deployed using a Fusion job called *Create Seldon Core Model Deployment*. This job downloads the selected pre-trained model and installs it in Fusion.

1. Navigate to **Collections** > **Jobs**.
2. Select **Add** > **Create Seldon Core Model Deployment**.
3. Enter a **Job ID**, such as `deploy-qna-coldstart-multilingual` or `deploy-qna-coldstart-large`.
4. Enter the **Model Name**, one of the following:
   - `qna-coldstart-multilingual`
   - `qna-coldstart-large`
5. In the **Docker Repository** field, enter `lucidworks`.
6. In the **Image Name** field, enter one of the following:
   - `qna-coldstart-multilingual:v1.1`
   - `qna-coldstart-large:v1.1`
7. Leave the **Kubernetes Secret Name for Model Repo** field empty.
8. In the **Output Column Names for Model** field, enter one of the following:
   - `qna-coldstart-multilingual:[vector]`
   - `[vector, compressed_vector]`
9. Click **Save**.
10. Click **Run** > **Start** to start the deployment job.

Next steps

1. *Configure The Smart Answers Pipelines*
2. *Evaluate a Smart Answers Query Pipeline*

Set Up a Pre-Trained Cold Start Model for Smart Answers

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

Lucidworks provides these pre-trained cold start models for *Smart Answers*:

- `qna-coldstart-large` -this is a large model trained on variety of corpuses and tasks.
- `qna-coldstart-multilingual` -covers 16 languages. List of supported languages: Arabic, Chinese-simplified, Chinese-traditional, English, French, German, Italian, Japanese, Korean, Dutch, Polish, Portuguese, Spanish, Thai, Turkish, Russian.

When you use these models, you do not need to run the model training job. Instead, you run a job that deploys the model into Managed Fusion. The *Create Seldon Core Model Deployment job* deploys your model as a Docker image in Kubernetes, which you can scale up or down like other Managed Fusion services.

These models are a good basis for a cold start solution if your data does not contain much domain-specific terminology. Otherwise, consider *training a model* using your existing content.

Tip

Dimension size of vectors for both models is **512**. You might need this information when creating collections in Milvus.

1. Deploy a pre-trained cold-start model into Managed Fusion

The pre-trained cold-start models are deployed using a Managed Fusion job called *Create Seldon Core Model Deployment*. This job downloads the selected pre-trained model and installs it in Managed Fusion.

1. Navigate to **Collections** > **Jobs**.
2. Select **Add** > **Create Seldon Core Model Deployment**.
3. Enter a **Job ID**, such as `deploy-qna-coldstart-multilingual` or `deploy-qna-coldstart-large`.
4. Enter the **Model Name**, one of the following:
   - `qna-coldstart-multilingual`
   - `qna-coldstart-large`
5. In the **Docker Repository** field, enter `lucidworks`.
6. In the **Image Name** field, enter one of the following:
   - `qna-coldstart-multilingual:v1.1`
   - `qna-coldstart-large:v1.1`
7. Leave the **Kubernetes Secret Name for Model Repo** field empty.
8. In the **Output Column Names for Model** field, enter one of the following:
   - `qna-coldstart-multilingual:[vector]`
   - `qna-coldstart-large:[vector, compressed_vector]`
9. Click **Save**.
10. Click **Run** > **Start** to start the deployment job.

Next steps

1. *Evaluate a Smart Answers Query Pipeline*

Set Up Active Directory Authentication for Appkit

How-to

> Compatible versions: 4.2, 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *App Studio*

While integration with Active Directory can be achieved using the LDAP setup, Spring Security provides a tailored security provider which makes configuring security using Active Directory easier.

This article describes how to set up authentication against an Active Directory server. It assumes that your Appkit application is managed as a Maven project with all project dependencies described in a Maven POM file.

1. Add the security provider dependency

Authentication against Active Directory requires the Active Directory module.

To enable the Active Directory module

1. Remove any existing security provider dependency from the `pom.xml` under the root of the project.
2. Add this within the `dependencies` tag:

```
<dependency> <groupId>twigkit</groupId>
<artifactId>twigkit.security.provider.active-
directory</artifactId> <version>${project.parent.version}
</version> </dependency>
```

3. To configure Appkit to invoke this module on startup, change the `security.conf` file in `src/main/resources/conf/security/` to contain:

```
type: spring_security
```

2. Configure security filtering in the application

Spring Security operates using a Servlet filter that must be mapped in the `web.xml` file in `src/main/webapp/WEB-INF`:

```
<!-- Spring Security --> <context-param> <param-
name>contextConfigLocation</param-name> <param-value>classpath:spring-
security.xml</param-value> </context-param> <listener> <listener-
class>org.springframework.web.context.ContextLoaderListener</listener-
class> </listener> <filter> <filter-
name>springSecurityFilterChain</filter-name> <filter-
class>org.springframework.web.filter.DelegatingFilterProxy</filter-
class> </filter> <filter-mapping> <filter-
name>springSecurityFilterChain</filter-name> <url-pattern>/*</url-
pattern> </filter-mapping> <!-- Spring Security Ends -->
```

Added this as the first filter in the chain (at the top of the `web.xml` file).

3. Point the application to the security backend

No Spring Security XML file is required. Just add this in a file named `active-directory.conf` in `src/main/resources/conf/security`:

```
ldap-server-url: ldap://sharepoint-dc-mtyy7623.cloudapp.net:389/
domain: sharepoint-dev.twigkit.com
```

Configure this to point to the Active Directory server implementing the LDAP protocol and the domain under which users are located.

THE LDAP PROTOCOL

Note the use of the LDAP protocol in the url - `ldap://`. In some cases, SSL encryption is used with the 'LDAPS' protocol. This requires the certificate is trusted by the authentication client (the JVM running Appkit). There are several ways to configure the keystore for the JVM to trust the certificate from the LDAP server. The most common way to do this is to import the certificate into the JVM's default keystore.

Authentication manager configuration

Unlike the configuration for the LDAP module, the Active Directory specific configuration is minimal. A user can then login using either their username or the fully qualified principal (for example, `bloggsj@my-domain.com`). By default roles are assigned using the `memberOf` attribute values of the user entry.

4 Verify the configuration

You can now verify the authentication in an Appkit application using the *widget:login-form* on a login page which is typically located in `src/main/webapp/login.jsp` in your application source tree.

Related articles

- *Set Up Lucidworks Fusion Authentication*
- *Set Up HTTP Headers Authentication*

- *Set Up Kerberos Authentication*
- *Set Up LDAP Authentication*
- *Set Up SiteMinder Authentication*
- *Set Up Spring Authentication*
- *Set Up Tomcat Container Authentication*
- *Set Up Windows Authentication*

Set Up Active Directory Federation Services (ADFS) Authentication for App Studio or Appkit

How-to

> Compatible versions: 4.2, 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *App Studio*

The Active Directory Federation Services (ADFS) security provider is available in Appkit.

Note

OAuth based authentication against ADFS requires **ADFS 3** (or newer) which is available from Windows Server 2012 R2 onwards.

To authenticate against ADFS, perform the steps in this article.

1 Add the security provider dependency

To add ADFS as a security provider in the Oauth Security module:

1. Remove any existing security provider dependency from the `pom.xml` under the root of the project.
2. Add a security provider dependency for ADFS to the `dependencies` tag in `pom.xml`:

```
<dependency> <groupId>twigkit</groupId>
<artifactId>twigkit.security.provider.oauth.adfs3</artifactId>
<version>${project.parent.version}</version> </dependency>
```

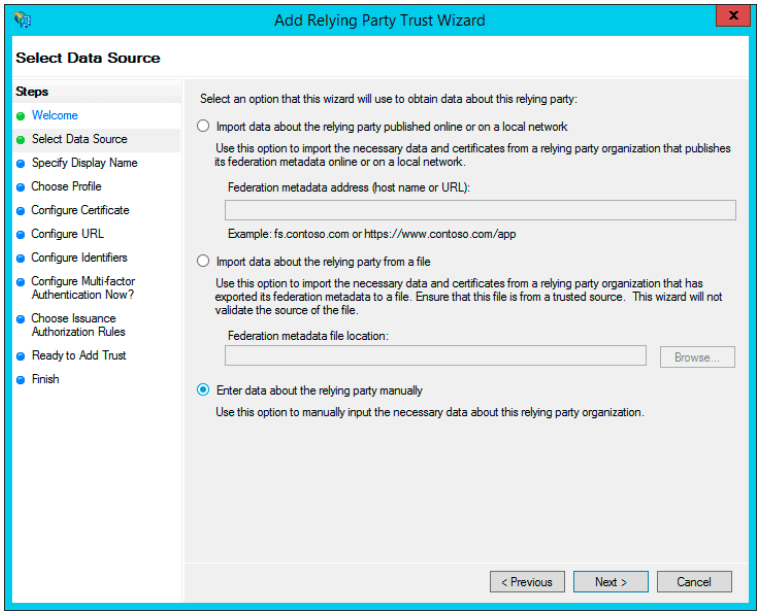
3. Configure Appkit to invoke the Oauth Security module on startup. Change the `security.conf` file in `src/main/resources/conf/security/` to contain:

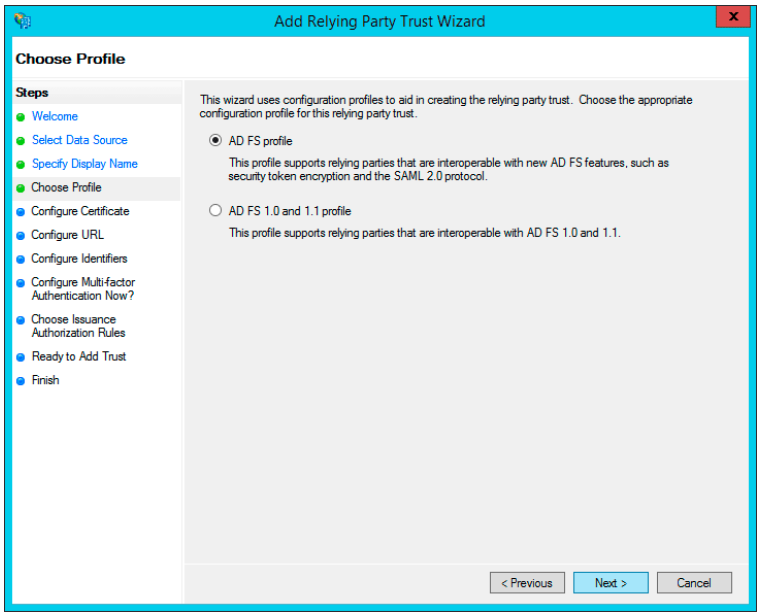
```
type: oauth
```

Note

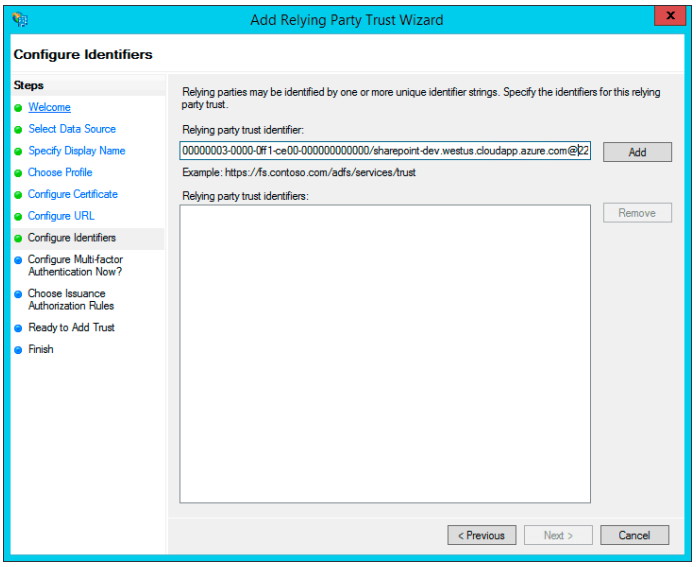
You must remove any existing `spring-security.xml` file because this module encapsulates all Spring configuration automatically.

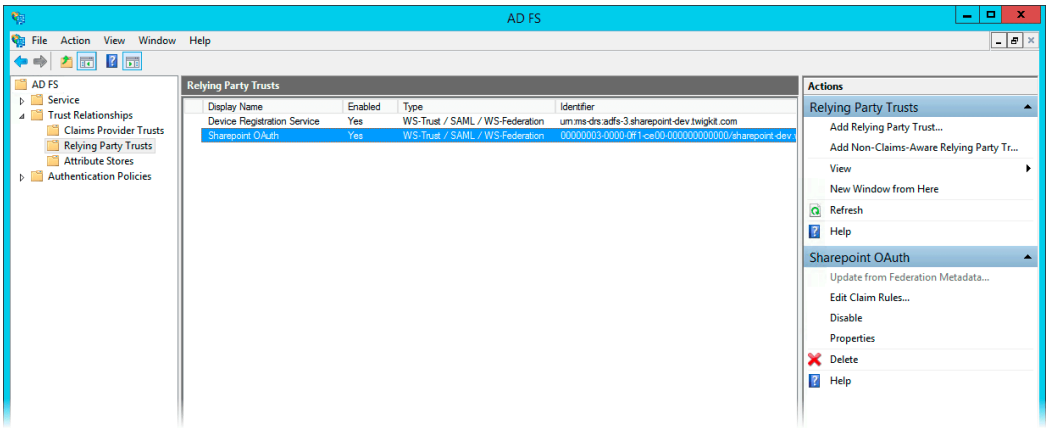
2 Set up the application (relying party trust) in ADFS



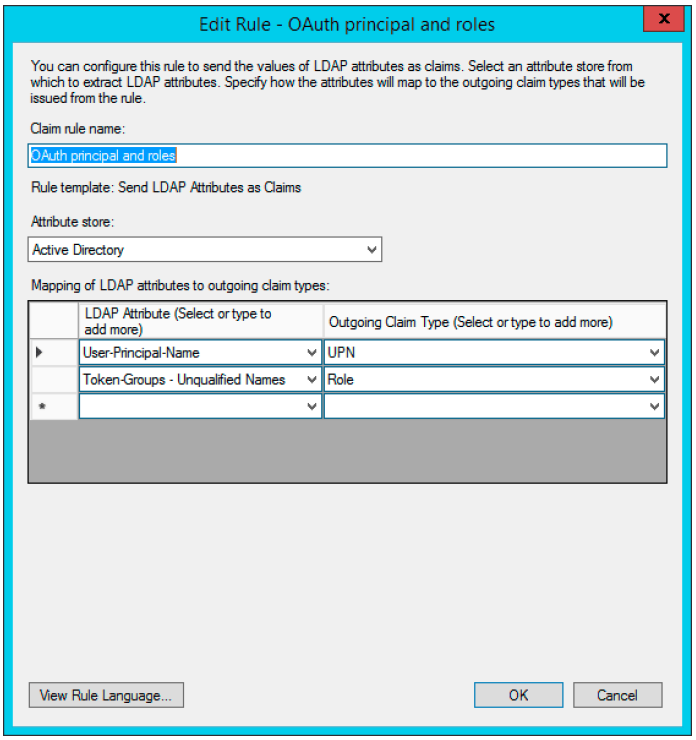


1. Add a new relying party in ADFS management. The identifier is usually set to the URL of the Appkit application itself.





2. Edit the Claim Rules to pass through the credentials as shown in the screenshot. It is important the configuration is as shown in the screenshots here. The User Principal Name is passed through as the 'UPN' claim and the Token-Groups Unqualified Name is passed through as the 'Role' claim.



3. Open a PowerShell session to create the OAuth ADFS 'client' for the Appkit application:

```
Add-ADFSClient -Name "Appkit OAuth" -ClientId "1234567890-ABCDEF" -RedirectUri="http://localhost:8080/oauthLogin"
```

The `ClientId` should be a GUID and the `RedirectUri` must point to the Appkit application URL, with `/oauthLogin` appended. Here, `localhost` is used for testing an Appkit application running on the local development machine.

3 Configure the OAuth module for the application setup in ADFS

Create a new configuration file in `conf/security/oauth.conf`.

`oauth.conf`

```
client-id: 168f3ee4-63fc-4723-a61a-6473f6cb515d adfs-url: https://your-adfs-server/adfs resource: http://localhost:8080 high-trust: false
```

Here, you must change the `client-id`, `adfs-url` and `resource` parameters for the environment in question.

The `client-id` is the ID that was set up against ADFS using the `Add-ADFSClient` PowerShell command.

The `adfs-url` is the URL of the ADFS server with the `/adfs` context appended.

The `resource` is the relying party trust identifier set up in ADFS management.

The `high-trust' parameter is only required when integrating the ADFS and SharePoint modules. This article is only concerned with authentication against ADFS.

3 Add the Spring filter to the web.xml

Add this to the `web.xml` file of the project:

```
<!-- Spring Security --> <context-param> <param-
name>contextConfigLocation</param-name> <param-value>classpath:spring-
security.xml</param-value> </context-param> <listener> <listener-
class>org.springframework.web.context.ContextLoaderListener</listener-
class> </listener> <filter> <filter-
name>springSecurityFilterChain</filter-name> <filter-
class>org.springframework.web.filter.DelegatingFilterProxy</filter-
class> </filter> <filter-mapping> <filter-
name>springSecurityFilterChain</filter-name> <url-pattern>/*</url-
pattern> </filter-mapping> <listener> <listener-class>
org.springframework.web.context.request.RequestContextListener
</listener-class> </listener> <!-- Spring Security Ends -->
```

Inclusion of a 'RequestContextListener' is not required in a standard Appkit-plus-Spring Security application.

4 Test the authentication

If a user is not logged via ADFS prior to visiting the application they will be redirected to the ADFS OAuth login page. The user experience is a typical login page.

After the user logs in, the user will be returned to the application as an authenticated user.

The Appkit user's details will also be populated with any basic information available from the decoded OAuth token such as roles and the user principal name.

Related articles

- *Set Up Facebook Authentication*
- *Set Up Google Authentication*
- *Set Up Office 365 Authentication*

Set up App-Only Auth with Key for SharePoint Online

How-to

> Compatible versions: 4.2
>
> Works with: *Fusion Connectors*, *Fusion Server*, *Fusion*

These steps are based on Microsoft's Granting Access via Azure AD App-Only.

Create the .cer and .pfx file

1. Connect a Windows system (any Windows machine is fine).
2. Save this PowerShell script as **Create-SelfSignedCertificate.ps1**.

```
#Requires -RunAsAdministrator <# .SYNOPSIS Creates a Self Signed
Certificate for use in server to server authentication .DESCRIPTION
.EXAMPLE PS C:\> .\Create-SelfSignedCertificate.ps1 -CommonName
"MyCert" -StartDate 2015-11-21 -EndDate 2017-11-21 This will create a
new self signed certificate with the common name "CN=MyCert". During
creation you will be asked to provide a password to protect the
private key. .EXAMPLE PS C:\> .\Create-SelfSignedCertificate.ps1 -
CommonName "MyCert" -StartDate 2015-11-21 -EndDate 2017-11-21 -
Password (ConvertTo-SecureString -String "MyPassword" -AsPlainText -
Force) This will create a new self signed certificate with the common
name "CN=MyCert". The password as specified in the Password parameter
will be used to protect the private key .EXAMPLE PS C:\> .\Create-
SelfSignedCertificate.ps1 -CommonName "MyCert" -StartDate 2015-11-21 -
EndDate 2017-11-21 -Force This will create a new self signed
certificate with the common name "CN=MyCert". During creation you will
be asked to provide a password to protect the private key. If there is
already a certificate with the common name you specified, it will be
removed first. #> Param( [Parameter(Mandatory=$true)]
[string]$CommonName, [Parameter(Mandatory=$true)]
[DateTime]$StartDate, [Parameter(Mandatory=$true)] [DateTime]$EndDate,
[Parameter(Mandatory=$false, HelpMessage="Will overwrite existing
certificates")] [Switch]$Force, [Parameter(Mandatory=$false)]
[SecureString]$Password ) # DO NOT MODIFY BELOW function
CreateSelfSignedCertificate(){ #Remove and existing certificates with
the same common name from personal and root stores #Need to be very
wary of this as could break something
if($CommonName.ToLower().StartsWith("cn=")) { # Remove CN from common
name $CommonName = $CommonName.Substring(3) } $certs = Get-ChildItem -
Path Cert:\LocalMachine\my | Where-Object{$_.Subject -eq
"CN=$CommonName"} if($certs -ne $null -and $certs.Length -gt 0) {
if($Force) { foreach($c in $certs) { remove-item $c.PSPath } } else {
Write-Host -ForegroundColor Red "One or more certificates with the
same common name (CN=$CommonName) are already located in the local
```

```
certificate store. Use -Force to remove them"; return $false } } $name
= new-object -com "X509Enrollment.CX500DistinguishedName.1"
$name.Encode("CN=$CommonName", 0) $key = new-object -com
"X509Enrollment.CX509PrivateKey.1" $key.ProviderName = "Microsoft RSA
SChannel Cryptographic Provider" $key.KeySpec = 1 $key.Length = 2048
$key.SecurityDescriptor = "D:PAI(A;;0xd01f01ff;;;SY)
(A;;0xd01f01ff;;;BA)(A;;0x80120089;;;NS)" $key.MachineContext = 1
$key.ExportPolicy = 1 # This is required to allow the private key to
be exported $key.Create() $serverauthoid = new-object -com
"X509Enrollment.CObjectId.1"
$serverauthoid.InitializeFromValue("1.3.6.1.5.5.7.3.1") # Server
Authentication $ekuoids = new-object -com
"X509Enrollment.CObjectIds.1" $ekuoids.add($serverauthoid) $ekuext =
new-object -com "X509Enrollment.CX509ExtensionEnhancedKeyUsage.1"
$ekuext.InitializeEncode($ekuoids) $cert = new-object -com
"X509Enrollment.CX509CertificateRequestCertificate.1"
$cert.InitializeFromPrivateKey(2, $key, "") $cert.Subject = $name
$cert.Issuer = $cert.Subject $cert.NotBefore = $StartDate
$cert.NotAfter = $EndDate $cert.X509Extensions.Add($ekuext)
$cert.Encode() $enrollment = new-object -com
"X509Enrollment.CX509Enrollment.1"
$enrollment.InitializeFromRequest($cert) $certdata =
$enrollment.CreateRequest(0) $enrollment.InstallResponse(2, $certdata,
0, "") return $true } function ExportPFXFile() {
if($CommonName.ToLower().StartsWith("cn=")) { # Remove CN from common
name $CommonName = $CommonName.Substring(3) } if($Password -eq $null)
{ $Password = Read-Host -Prompt "Enter Password to protect private
key" -AsSecureString } $cert = Get-ChildItem -Path
Cert:\LocalMachine\my | where-object{$_.Subject -eq "CN=$CommonName"}
Export-PfxCertificate -Cert $cert -Password $Password -FilePath
"$($CommonName).pfx" Export-Certificate -Cert $cert -Type CERT -
FilePath "$CommonName.cer" } function RemoveCertsFromStore() { # Once
the certificates have been been exported we can safely remove them
from the store if($CommonName.ToLower().StartsWith("cn=")) { # Remove
```

```
CN from common name $CommonName = $CommonName.Substring(3) } $certs =
Get-ChildItem -Path Cert:\LocalMachine\my | Where-Object{$_.Subject -
eq "CN=$CommonName"} foreach($c in $certs) { remove-item $c.PSPath } }
if(CreateSelfSignedCertificate) { ExportPFXFile RemoveCertsFromStore
}
```

3. Open a Command prompt or PowerShell: Run as Administrator
4. Run the below script to create the .pfx and .cer files needed for the next steps, where:
    a. `StartDate` : Current Date
    b. `EndDate` : Current Date + 2 years
    c. `CommonName` : Name given to the .cer and .pfx files, such as `YourFileName`

```
\Create-SelfSignedCertificate.ps1 -CommonName "YourFileName" -
StartDate YYYY-MM-DD -EndDate YYYY-MM-DD
```

At running the command, a password will be requested to protect the private key. **YourFileName.pfx** and **YourFileName.cer** files will be created.

Register a new App

- Go to the Azure Portal , select the **Azure Active Directory** section, and choose the option **App registrations**.
- Register a new application: **App Registrations** → **New Registration**
  https://portal.azure.com/#blade/Microsoft_AAD_IAM/ActiveDirectoryMenuBlade/RegisteredApps

Important

Once the application has been created, copy the **Application (client) ID** as you'll need to configure the datasource.

Grant permission to App

- From **API permissions** in the left menu bar, click on the **Add a permission** button. Choose the permissions that you will grant to this application.
    - Choose **Sharepoint** permissions, and select the permissions needed (**Application** → **Sites.FullControl.All**)
- Click on the **Grant admin consent for OrganizationName** button and confirm the action by clicking on the **Yes** button that appears at the top.
- From **Certificates & Secrets** in the left menu, upload the Certificate file (YourFIleName.cer) created earlier:

Obtain the base64 code from the .pfx file

It can be obtained in two ways

1. From Command Line (Linux)
    - Convert the .pfx to .pc12

```
keytool –importkeystore –srckeystore YourFileName.pfx –
srcstoretype pkcs12 –destkeystore yourcert.p12 –
deststoretype pkcs12
```

*Convert the .pc12 to base64

```
base64 –i yourcert.p12 > base64.txt
```

    - Open the .txt file and copy the value generated, you will need it to configure the datasource config

2. From PowerShell
    - From PowerShell, run the following commands:

```
$fileContentBytes = get–content '.\YourFileName.pfx' –
Encoding Byte
[System.Convert]::ToBase64String($fileContentBytes) |
Out–File 'pfx–encoded–bytes.txt'
```

    - Open the .txt file and copy the value generated, you will need it to configure the datasource config

Renew the Certificate

When the certificate expires, the crawl will fail with the following error:

```
Client assertion contains an invalid signature. [Reason – The key used is expired 1AuthenticationException: {"error_description":"AADSTS700027: Client assertion contains an invalid signature. [Reason – The key used is expired., Thumbprint of key used by client:"...]}
```

The .cer and .pfx need to be renewed. Follow these steps:

- Execute the script from '' to get a new .cer and .pfx files
- Go to the Azure Portal , select the **Azure Active Directory** section, choose the option **App registrations**, and open the Application from where the certificate expired.
    - From **Certificates & Secrets'in** the left menu bar, remove the older certificate, and upload the new one
    - Obtain the base64 from the renewed .pfx file

Set up App-only Authentication With Private Key into Sharepoint v2 plugin

- Client ID, and Base64 from the .pfx file, and password for the private key, must be set into **Azure AD Client ID**, **AZURE AD PKCS12 BASE64 KEYSTORE**, and **AZURE AD PKCS12 KEYSTORE PASSWORD** properties respectively
- Set the appropriate tenant into **Azure AD Tenant** property, for example, `lucidworksfusion.onmicrosoft.com`

Set Up Appkit Cookie Cracker Authentication

How-to

> Compatible versions: 4.2
>
> Works with: *App Studio*

The Appkit ‘Cookie Cracker’ authentication provider allows authentication against an endpoint which implements the Appkit cookie cracking endpoint protocol. This allows single sign on via a cookie value set by a 'third party' service prior to entering the Appkit application.

1 Add the security provider dependency

Authentication against the cookie cracker requires the cookie cracker module. To enable this module, first remove any existing security provider dependency from the `pom.xml` file under the root of the project, and then add this within the `dependencies` tag:

```
<dependency> <groupId>twigkit</groupId>
<artifactId>twigkit.security.provider.cookie-cracker</artifactId>
<version>${project.parent.version}</version> </dependency>
```

Note

Because this module packages all Spring configuration, any `spring-security.xml` configuration file should be removed for simplicity.

2 Configure the cookie cracking endpoint

Configure these parameters in `conf/security/cookie_cracking.conf` to specify the endpoint URL, and cookies to pass to the cracking endpoint:

```
url: http://localhost:8080/twigkit/services/gsa/auth cookies: twigkit-
token domain: localhost
```

- `url` is the URL of the cookie cracking endpoint that acts as an authentication provider.
- `cookies` specifies those cookies to pass through to the cookie cracking endpoint for inspection.
- `domain` specifies a domain to set on the cookies listed (optional except when the cookie cracking endpoint exists on a separate domain to the Appkit application, in which case it is required).

Related articles

- *Set Up Active Directory Authentication*
- *Set Up Lucidworks Fusion Authentication*
- *Set Up HTTP Headers Authentication*
- *Set Up Kerberos Authentication*
- *Set Up LDAP Authentication*
- *Set Up SiteMinder Authentication*
- *Set Up Spring Authentication*
- *Set Up Tomcat Container Authentication*
- *Set Up Windows Authentication*

Set up Dynamic Pricing

How-to

> Compatible versions: 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

Dynamic Pricing lets you sort, facet, boost, and query multiple pricing lists and product inventories efficiently.

Extensive product catalogs and pricing lists result in large datasets that can overwhelm Solr. Dynamic Pricing solves this big data issue by storing versioned price lists outside of Solr. These files are then processed into a binary format easily ingested by Solr.

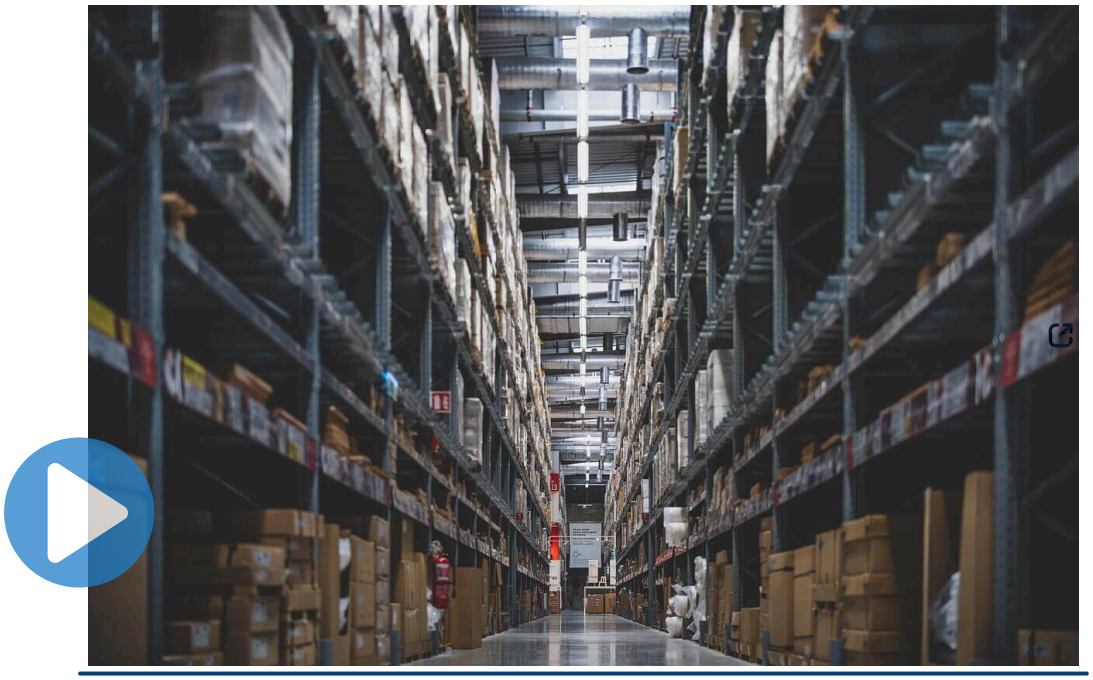
Note

Contact your Lucidworks representative to enable this feature.

LucidAcademy

Lucidworks offers free training to help you get started.

The **Course** for **Dynamic Pricing** focuses on how Dynamic Pricing maximizes custom pricing strategies:



Visit the LucidAcademy to see the full training catalog.

Format raw data

To use Dynamic Pricing, you need to prepare and format your files. Store the raw external files in the following directory structure:

```
EXTERNAL_FILES_ROOT/COLLECTION/N_TOP_LEVEL_SUBDIRS/BASENAME/SEQUENCE_ID/FILE
```

Format	Description	Example
EXTERNAL_FILES_ROOT	The root directory for the external files.	`root`
COLLECTION	The Solr collection the external files use.	`bucket1`
N_TOP_LEVEL_SUBDIRS	Subdirectories to hold files. How many subdirectories, what they are named and which files they hold does not affect the output of the file processing, but it does impact replication performance. Replication of raw files is applied at the top level subdirectories. Lucidworks recommends that you have at least eight top level directories to maximize the benefits of parallelization.	`SubDir`
BASENAME	A directory for each file. The directory name must be the same as the filename without the extension (basename). For example if the file is named `example.txt` the directory must be named `example`.	`example`
SEQUENCE_ID	The sequenceId directories hold different versions of the external files. SequenceIds must be positive whole numbers, parsable to a Java Long class. New versions of the file must be placed in a higher sequenceId directory.	`1661540544007`
FILE	Each sequencedD directory holds one external file. Files that are being written must have a temporary extension other than `.txt` and be renamed to a `.txt` extension after being written. If Solr dynamic fields are used, the file name must end with a "_" postfix that maps directly to a Solr dynamic field.	`example.txt`

Sample full file path: `$root/bucket1/example/1661540544007/example_ef.txt`

Text file format

The format of the text file is a whitespace-separated list of fields that appears as:

```
id1 float1 id2 float2
```

or

```
id1 routeKey1 float1 id2 routeKey1 float2
```

The `id` must map to a unique `documentId` in a Solr collection. The `float` value must parse to a Java float. The `routeKey` must be provided if documents are routed to shards with a specific route key. Each file is split into separate partitions for each shard. Do not include whitespaces in `ids` and `route keys` because this causes file format errors.

Optional blank lines and comments starting with `#` are allowed.

Incremental updates

Dynamic Pricing supports incremental updates of external files. Store the incremental update files in the root of the BASENAME directory:

`EXTERNAL_FILES_ROOT/COLLECTION/N_TOP_LEVEL_SUBDIRS/BASENAME`

Structure the file name of the incremental files as `<BASENAME>-<SEQUENCE_ID>.inc`.

The extension of the incremental files is `.inc`. When incremental files are being written, use a temporary extension other than `.inc` and rename them to a `.inc` extension after being written.

Sample path of an incremental file: `$root/bucket1/foo_ef/foo_ef-1666023034.inc`

Incremental updates apply to the full external file in the latest sequenceId directory. The existing external file does not change. Instead, Dynamic Pricing writes a new full external file to a new sequenceId directory with the merged incremental updates. The new sequenceId directory is taken from the highest sequenceId of the incremental files. Once the new external file is written, the incremental files are deleted.

The format of the records in the incremental files must be the same as the records in the *external files*.

Set Up Email Alerts

How-to

> Compatible versions: 4.2, 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *App Studio*

Email alerts can be configured statically in `conf` files to be sent at regular intervals throughout the lifecycle of an application. Use these steps to get up and running with the Alerting module:

1. Add this dependency to your `pom.xml` file:

```
<dependency> <groupId>twigkit</groupId>
<artifactId>twigkit.alert.runtime</artifactId>
<version>${project.parent.version}</version> </dependency>
```

2. Set up `conf/alert/email/smtp.conf`; see below for an example configuration:

```
# User to log into Appkit twigkit-user: foo twigkit-password:
bar # User to send emails as from: foobar@twigkit.com password:
foobar # Override to address for all config-based alerts to:
reporting@some-alerting-domain.com # SMTP details
mail.smtp.host: smtp.gmail.com mail.smtp.port: 587
mail.smtp.auth: true mail.smtp.starttls.enable: true
```

3. Set up `conf/alert/email/alerts/alerts.conf`; see below for an example configuration:

```
email-template: http://localhost:8080/email/
```

4. Add the alert `.conf` files to `conf/alert/email/alerts/my-alert-name-here.conf`; see below for an example:

```
to: foobar@twigkit.com subject: New documents on brewing query:
?q=homebrew cron-expression: 30 * * * * ?
```

   Here, the `cron-expression` defines how often to send the alert as described in this Quartz tutorial.

5. Create a JSP, in this case configured as above, on a standard Appkit application would be in `src/main/webapp/WEB-INF/pages/email.jsp` and the content would include:

```
<%@ page contentType="text/html" pageEncoding="UTF-8" %> <%@
include file="/WEB-INF/tags/client/taglibs.jspf" %>
<search:localization var="resources" bundleName="resources" />
<search:twigkit> <search:platform conf="platforms.solr" />
<search:query var="query" parameters="*" /> <search:response
var="response" platform="${platform}" query="${query}" />
</search:twigkit> <!DOCTYPE html> <html
xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
<h1>Automated alert for query ${query.value}</h1>
<search:resultList response="${response}"> <search:result>
<search:field name="title" /> <search:field name="description"
/> </search:result> </search:resultList> </html>
```

Related articles

- *Set Up Persisting Alerts*
- *Set Up Static Alerts*
- *Set Up User-specified Alerts*

Set Up Facebook Authentication

How-to

> Compatible versions: 4.2, 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *App Studio*

To authenticate against Facebook, perform the steps in this article.

1 Add the security provider dependency

To add Facebook as a security provider in the Oauth Security module:

1. Remove any existing security provider dependency from the `pom.xml` under the root of the project.
2. Add a security provider dependency for Facebook to the `dependencies` tag in `pom.xml`:

```
<dependency> <groupId>twigkit</groupId>
<artifactId>twigkit.security.provider.oauth.facebook</artifactId
> <version>${project.parent.version}</version> </dependency>
```

3. Configure Appkit to invoke the Oauth Security module on startup. Change the `security.conf` file in `src/main/resources/conf/security/` to contain:

```
type: oauth
```

> **Note**
>
> You must remove any existing `spring-security.xml` file because this module encapsulates all Spring configuration automatically.

2 Configure the OAuth module for the application setup in Facebook

Add the relevant configuration in a file in `conf/security/oauth.conf`. For example:

```
client-id: 1669796236622670 client-secret:
c064ea0f5e5c13a2a6c5f6366f7426cb scope: public_profile,email
```

Including the user profile scope lets Appkit automatically pull the user profile information into their Appkit user details.

This guide assumes the `client-id` and `client-secret` settings have already been generated in the 'Facebook for Developers' admin console and provided to you as an application developer. For more information, see the Facebook documentation on OAuth.

Note

The application's 'Site URL' must be configured in the console under 'settings' as `http://your-application-url/oauthLogin`.

3 Add the Spring filter to the web.xml

Add this to the `web.xml` file of the project:

```
<!-- Spring Security --> <context-param> <param-
name>contextConfigLocation</param-name> <param-value>classpath:spring-
security.xml</param-value> </context-param> <listener> <listener-
class>org.springframework.web.context.ContextLoaderListener</listener-
class> </listener> <filter> <filter-
name>springSecurityFilterChain</filter-name> <filter-
class>org.springframework.web.filter.DelegatingFilterProxy</filter-
class> </filter> <filter-mapping> <filter-
name>springSecurityFilterChain</filter-name> <url-pattern>/*</url-
pattern> </filter-mapping> <listener> <listener-class>
org.springframework.web.context.request.RequestContextListener
</listener-class> </listener> <!-- Spring Security Ends -->
```

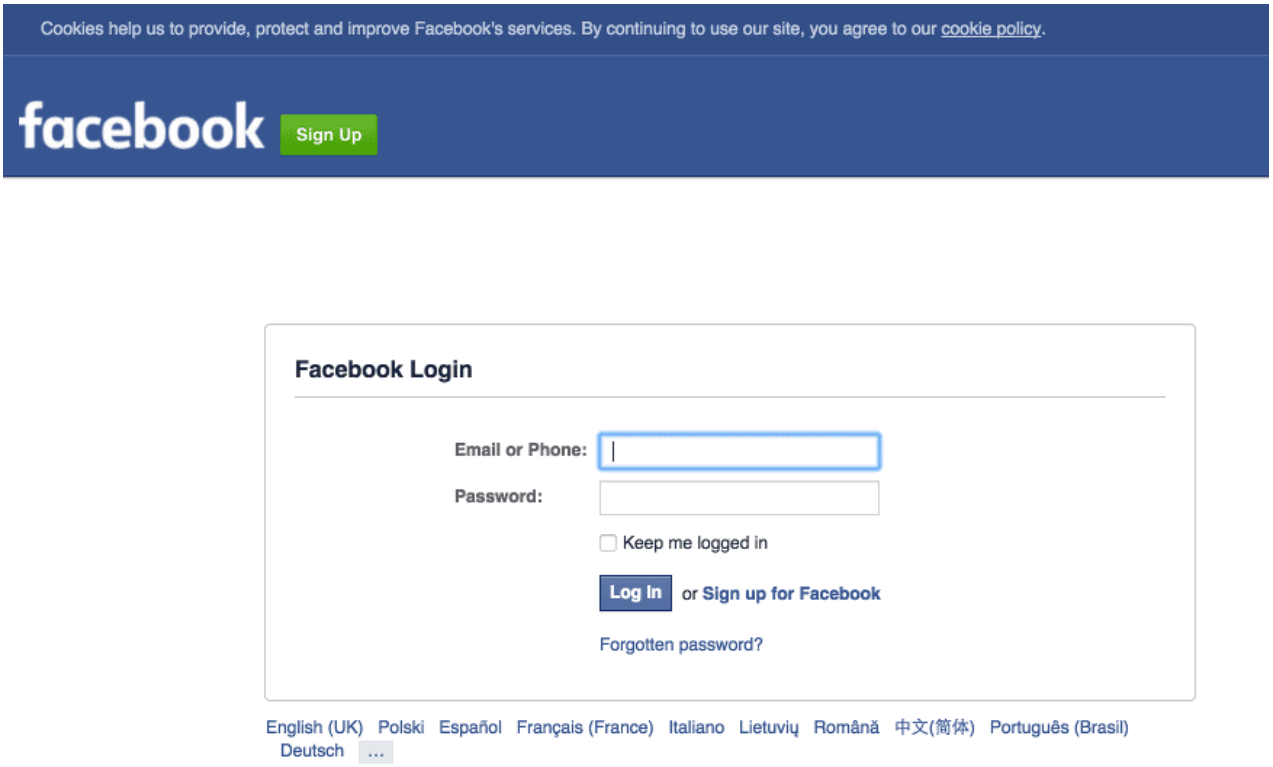
Inclusion of a 'RequestContextListener' is not required in a standard Appkit-plus-Spring Security application.

4 Test the authentication

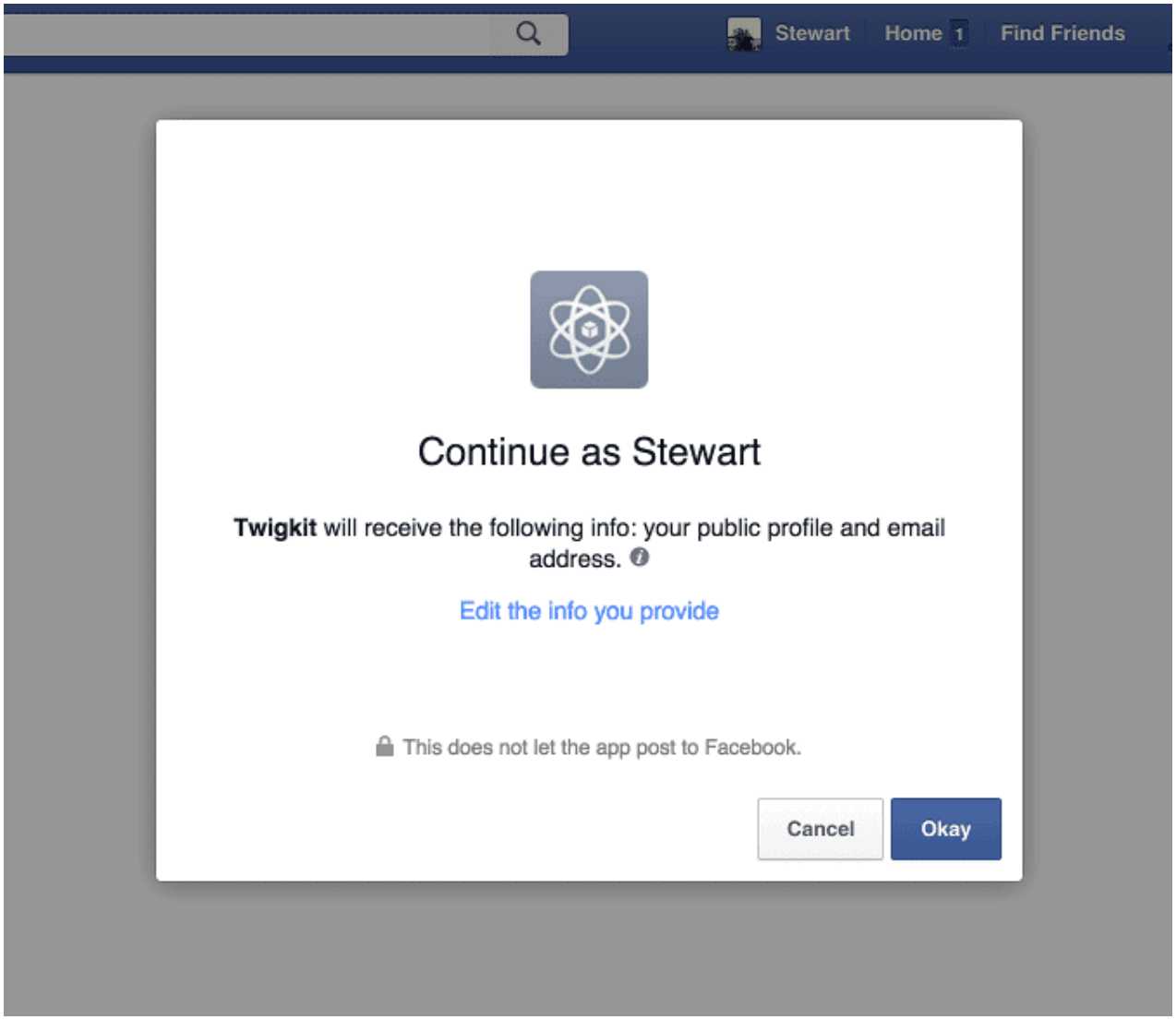
If a user is not logged into a Facebook account prior to visiting the application they will be redirected to the Facebook login page. They will then be asked if they want to approve the application to access the services defined in the scopes parameter. If the user profile scope is included the Appkit user's details will also be populated with any basic information available from the decoded OAuth token such as first name, last name and email address.

Using OAuth can create a seamless experience where, after the token handshake and approval is complete, the user will always be logged into the application as long as their session with Facebook is active.

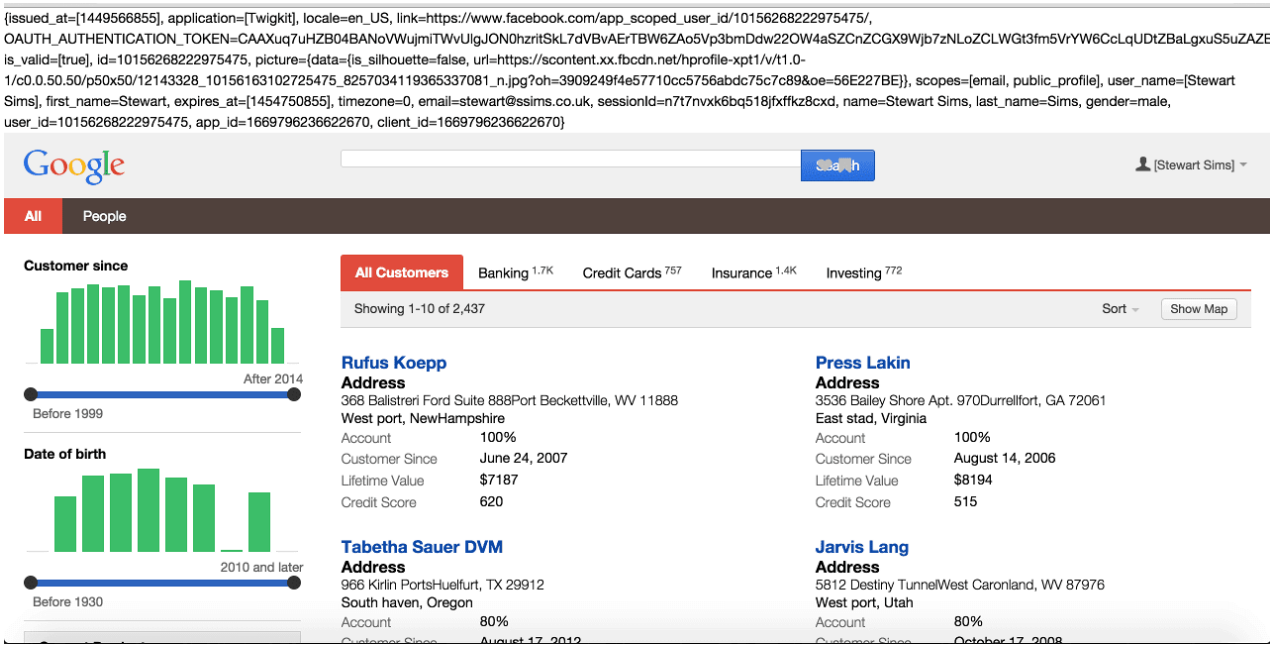
User logs in:



User approves application:



The user is signed into the application with an OAuth token, and will be signed in automatically from now on unless the token expires or the user logs out of Facebook entirely.



Related articles

- *Set Up Active Directory Federation Services (ADFS) Authentication*
- *Set Up Google Authentication*
- *Set Up Office 365 Authentication*

Set Up Google Authentication

How-to

> Compatible versions: 4.2, 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *App Studio*

To authenticate against Google services, perform the steps in this article.

1 Add the security provider dependency

To add Google as a security provider in the Oauth Security module:

1. Remove any existing security provider dependency from the `pom.xml` under the root of the project.
2. Add a security provider dependency for Google to the `dependencies` tag in `pom.xml` :

```
<dependency> <groupId>twigkit</groupId>
<artifactId>twigkit.security.provider.oauth.google</artifactId>
<version>${project.parent.version}</version> </dependency>
```

3. Configure Appkit to invoke this module on startup. Change the `security.conf` file in `src/main/resources/conf/security/` to contain:

```
type: oauth
```

4. Remove any existing `spring-security.xml` file, because this module encapsulates all Spring configuration automatically.

2 Configure the OAuth module for application setup in Google Developer Console

Add the relevant configuration in a file in `conf/security/oauth.conf` :

```
client-id: 419084461435-
2370lvl0rpcr8b8lu0ljkv3l2ib2ahfc.apps.googleusercontent.com client-
secret: 5p-Ph7AcvnFBfMQSSUNv-umQ scope:
https://www.googleapis.com/auth/userinfo.profile
```

Including the user profile scope lets Appkit automatically pull the user profile information into their Appkit user details.

This guide assumes the `client-id` and `client-secret` settings have already been generated in the Google Developer Console and provided to you as an application developer. For more information about this, see the Google documentation on OAuth .

Note

The authorized redirect URI must be configured in the Google Developer Console as `http://your-application-url/oauthLogin` .

3 Add the Spring filter to the web.xml

Add this to the `web.xml` of the project:

```
<!-- Spring Security --> <context-param> <param-
name>contextConfigLocation</param-name> <param-value>classpath:spring-
security.xml</param-value> </context-param> <listener> <listener-
class>org.springframework.web.context.ContextLoaderListener</listener-
class> </listener> <filter> <filter-
name>springSecurityFilterChain</filter-name> <filter-
class>org.springframework.web.filter.DelegatingFilterProxy</filter-
class> </filter> <filter-mapping> <filter-
name>springSecurityFilterChain</filter-name> <url-pattern>/*</url-
pattern> </filter-mapping> <listener> <listener-class>
org.springframework.web.context.request.RequestContextListener
</listener-class> </listener> <!-- Spring Security Ends -->
```

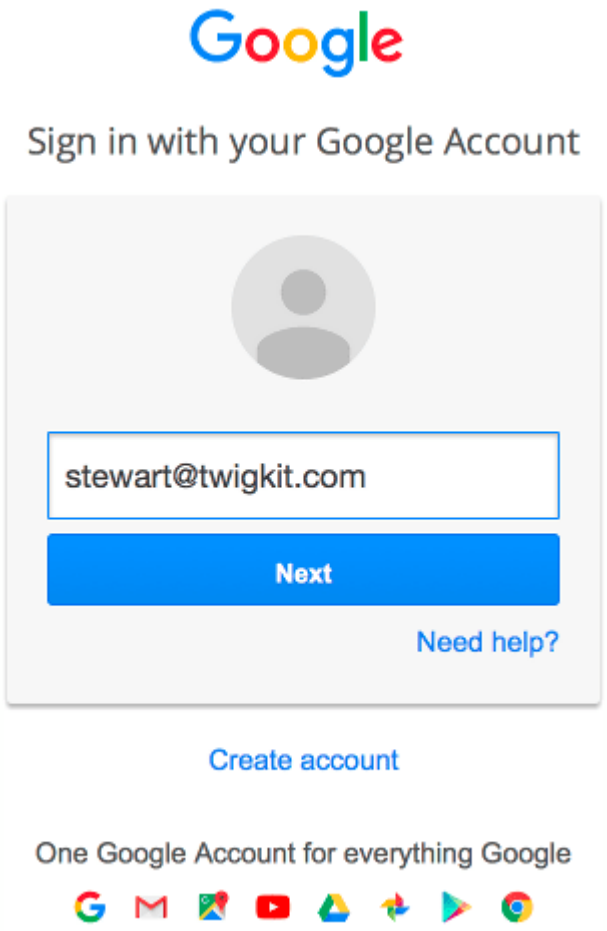
Inclusion of a 'RequestContextListener' is not required in a standard Appkit-plus-Spring Security application.

4 Test the authentication

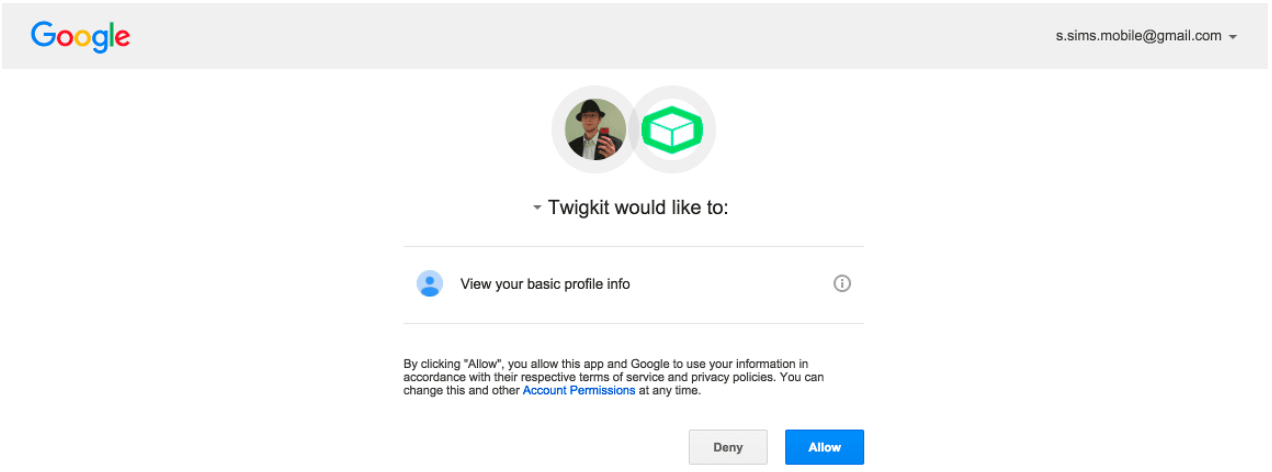
If a user is not logged into a Google account prior to visiting the application they will be redirected to the Google login page. They will then be asked if they want to approve the application to access the services defined in the scopes parameter. If the user profile scope is included the Appkit user's details will also be populated with any basic information available from the decoded OAuth token such as first name, last name and email address.

Using OAuth can create a seamless experience where, after the token handshake and approval is complete, the user will always be logged into the application as long as their session with Google is active.

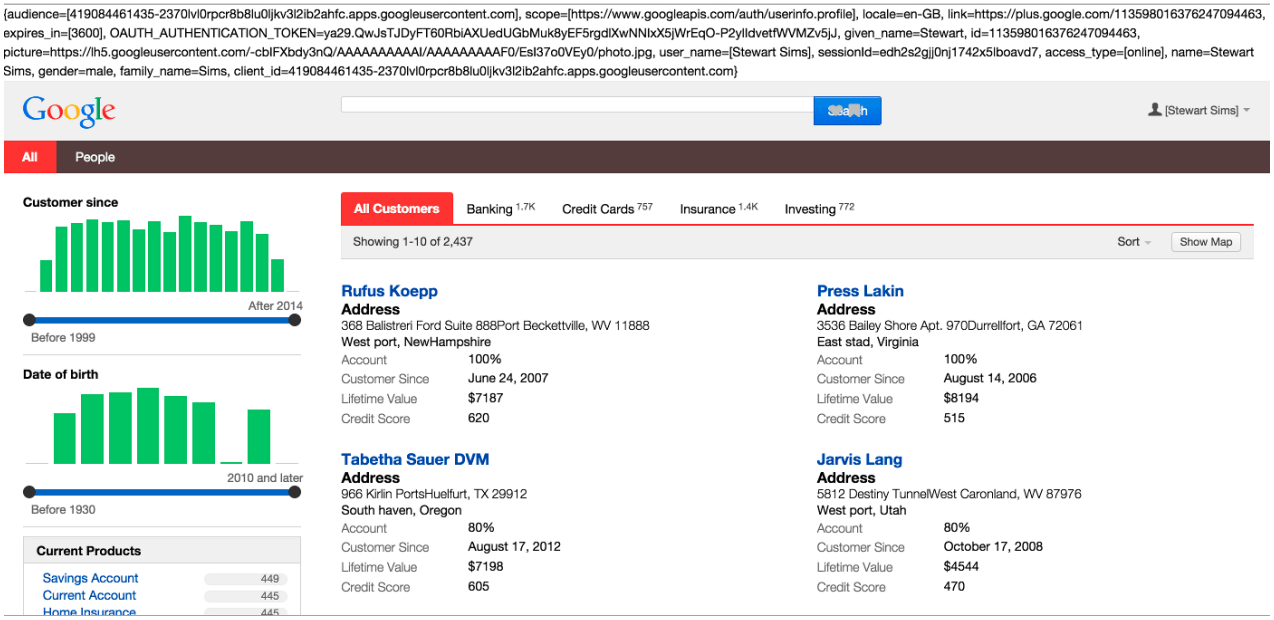
User logs in:



User approves application:



The user is signed into the application with an OAuth token, and will be signed in automatically from now on unless the token expires or the user logs out of Google entirely.



Related articles

- *Set Up Active Directory Federation Services (ADFS) Authentication*
- *Set Up Facebook Authentication*
- *Set Up Office 365 Authentication*

Set Up HTTP Headers Authentication

How-to

> Compatible versions: 4.2, 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *App Studio*

Appkit supports authentication via generic HTTP headers passed through from a component behind a secure firewall sitting in front of the Appkit application. If there is such an authentication component in the environment where you are using Appkit, you can configure the HTTP headers that the application will use to pick up the user's details.

Adding header security

1. Add this dependency to the `pom.xml` file:

```
<dependency> <groupId>twigkit</groupId>
<artifactId>twigkit.security.provider.http-header</artifactId>
<version>${project.parent.version}</version> </dependency>
```

2. Modify `conf/security/security.conf` to contain this:

```
type: header
```

Configuring the headers used

Create a new configuration file in `conf/security/header/header.conf`.

A typical configuration might look like:

```
user-id: remote_user roles: x-group
```

Here, the user's ID will be extracted from the value of the `remote_user` header and roles from the `x-group` header.

All possible properties are listed here:

user-id: name of header that contains the ID of the user

roles: name of the header that contains the user's roles - multiple values will be picked up from the header if specified by the comma separated header value convention

domain: name of the header that contains the domain for the user

detail-prefix: prefix of any headers containing values which should be added to the user's details (for example, if `X-USER-` a header with the name `X-USER-DEPARTMENT` and its value will be added as an entry to the user details map)

Related articles

- *Set Up Active Directory Authentication*
- *Set Up Lucidworks Fusion Authentication*
- *Set Up Kerberos Authentication*

- *Set Up LDAP Authentication*
- *Set Up SiteMinder Authentication*
- *Set Up Spring Authentication*
- *Set Up Tomcat Container Authentication*
- *Set Up Windows Authentication*

Set Up Kerberos Authentication

How-to

> Compatible versions: 4.2, 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *App Studio*

The Kerberos security provider for Appkit allows authentication using Kerberos. With this provider configured, an application deployed within a Kerberos 'realm' can securely authenticate a user against a Kerberos 'Key Distribution Center'. This module is primarily intended to simplify the process of authenticating a user silently in a Windows environment by leveraging the built in support for integrated Windows authentication in Internet Explorer. However it should be possible to use this module to authenticate the user in any Kerberos compliant environment.

This guide focuses on how to set up Kerberos authentication within an Appkit application in a Windows environment. It assumes Kerberos is enabled and already working on a domain controller.

1 Add the security provider dependency

Authentication against Kerberos requires the Kerberos provider module. To enable this module first remove any existing security provider dependency from the `pom.xml` under the root of the project and add this within the `dependencies` tag:

```
<dependency> <groupId>twigkit</groupId>
<artifactId>twigkit.security.provider.kerberos</artifactId>
<version>${project.parent.version}</version> </dependency>
```

Then to configure Appkit to invoke this module on startup change the `security.conf` file in `src/main/resources/conf/security/` to contain:

```
type: kerberos
```

2 Configure security filtering in the application

This module uses Spring Security which operates using a Servlet filter which must be mapped in the `web.xml` file in `src/main/webapp/WEB-INF` :

```
<!-- Spring Security --> <context-param> <param-
name>contextConfigLocation</param-name> <param-value>classpath:spring-
security.xml</param-value> </context-param> <listener> <listener-
class>org.springframework.web.context.ContextLoaderListener</listener-
class> </listener> <filter> <filter-
name>springSecurityFilterChain</filter-name> <filter-
class>org.springframework.web.filter.DelegatingFilterProxy</filter-
class> </filter> <filter-mapping> <filter-
name>springSecurityFilterChain</filter-name> <url-pattern>/*</url-
pattern> </filter-mapping> <!-- Spring Security Ends -->
```

This should be added as the first filter in the chain (at the top of the web.xml file). Remove any existing spring-security.xml file in the project because this configuration is pre-packaged by the Kerberos module.

3 Configure the application for the Kerberos environment

To use Kerberos with an application, a 'keytab' file is required. This identifies the application (service) and the host on which it is running in a special encrypted host key. You must configure a service principal and obtain a keytab file (usually done on the domain controller using Windows powershell) prior to configuring the Appkit application. After you have the service principal name and the keytab file configuration is easy. Add a file named `kerberos.conf` to `src/main/resources/conf/security/` with these parameters:

```
service-principal: HTTP/sharepoint-sp@sharepoint-dev.twigkit.com
keytab-location: classpath:sharepoint-dev.keytab debug: true #
optional (defaults to '/WEB-INF/pages/login.jsp') login-page: /WEB-
INF/pages/login.jsp
```

Set the service principal and keytab location accordingly. The debug parameter provides useful logging output from the Kerberos protocol interactions, and is recommended to be set to true for development. The login page is optional and provides a fallback page. If this is a standard Appkit login.jsp file, then the fallback form will be used to attempt login using form-based authentication (see *Fallback strategy* for more information). Alternatively this can direct to an 'unauthorized' error or debug page.

4 Test the authentication

Login to an account within the Kerberos 'realm' and navigate to the application in IE. The user should be silently authenticated.

Further information on using the Kerberos provider

Fallback strategy

This module includes a default fallback strategy in the event Kerberos authentication fails. Reasons for failure can include:

- Incorrect configuration of the application
- An invalid keytab file
- A browser that does not support Kerberos via the SPNEGO/Negotiate protocol
- Unauthorized access
- Access from outside the Kerberos 'realm'
- The environment is not configured correctly for Kerberos (for example, some other authentication mechanism 'preferred')

The fallback strategy is such that if Kerberos authentication fails a form is presented to attempt form based authentication. The page used is configurable as described above, and if forms based authentication is not supported it is recommended to point to a 'failure' or error page. In some scenarios it makes sense to have this fallback, depending on the nature of the security and environment in which the application is deployed. It should be noted that while a browser such as Internet Explorer supports NTLM, and often will attempt to fall back on this, this is not part of Kerberos and, due to restrictions in current Java runtimes, this is not supported.

General troubleshooting

First, check all of the above reasons the authentication can fail or 'fall back' from Kerberos. Use the `debug` configuration parameter and check the logs for information that could help to identify the cause. If it is a Windows environment it is recommended to check with the systems administrators that everything works correctly on the domain controller (check the Kerberos key distribution service, permissions of the service principal account, etc). You can view the headers passed from the browser in the debug information in the logs, these might provide an indication of why authentication failed (if Kerberos is used).

If you find or suspect that NTLM is used, ensure you are trying to access the site using the host name configured in the SPN (service principal name). For example, if the SPN is `HTTP/sharepoint-sp@sharepoint-dev.twigkit.com`, you must use the URL for the host machine `sharepoint-sp`: either `http://sharepoint-sp/` or `http://sharepoint-sp.sharepoint-dev.twigkit.com/`. Using 'localhost' or another alias will not work.

Authentication works but the user is prompted for a username and password

Usually in a Windows environment Kerberos allows silent authentication (single sign on via the account logged into the machine). In some cases the user is presented with a dialog (not an HTML form) the first time they visit the application or if the session expires, but when they enter their credentials it works fine. This can be a simple case that the user's Windows credentials are not correctly configured or IE has 'integrated windows authentication' disabled. Make sure this is enabled in IE and that the credentials are saved correctly in the user account settings under the 'Control Panel'.

How does this differ from Integrated Windows Authentication?

Previously in Appkit, the only way to do silent 'integrated' authentication with Windows was using the Tomcat ISAPI connector, as described in our article about *Integrated Windows Authentication*.

The Kerberos Security module simplifies the process for Kerberos enabled environments and keeps all of the configuration within the application.

Related articles

- *Set Up Active Directory Authentication*
- *Set Up Lucidworks Fusion Authentication*
- *Set Up HTTP Headers Authentication*
- *Set Up LDAP Authentication*
- *Set Up SiteMinder Authentication*
- *Set Up Spring Authentication*
- *Set Up Tomcat Container Authentication*
- *Set Up Windows Authentication*

Set Up LDAP Authentication

How-to

> Compatible versions: 4.2, 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *App Studio*

Appkit allows for authentication against a server using LDAP (Lightweight Directory Access Protocol). It assumes that your Appkit application is managed as a Maven project with all project dependencies described in a Maven POM file.

1 Add the security provider dependency

Authentication against LDAP requires the LDAP module. To enable this module first remove any existing security provider dependency from the `pom.xml` under the root of the project and add this within the `dependencies` tag:

```
<dependency> <groupId>twigkit</groupId>
<artifactId>twigkit.security.provider.ldap</artifactId>
<version>${project.parent.version}</version> </dependency>
```

Then to configure Appkit to invoke this module on startup change the `security.conf` file in `src/main/resources/conf/security/` to contain:

```
type: spring_security
```

2 Configure security filtering in the application

Spring Security operates using a Servlet filter that must be mapped in the `web.xml` file in `src/main/webapp/WEB-INF`:

```
<!-- Spring Security --> <context-param> <param-
name>contextConfigLocation</param-name> <param-value>classpath:spring-
security.xml</param-value> </context-param> <listener> <listener-
class>org.springframework.web.context.ContextLoaderListener</listener-
class> </listener> <filter> <filter-
name>springSecurityFilterChain</filter-name> <filter-
class>org.springframework.web.filter.DelegatingFilterProxy</filter-
class> </filter> <filter-mapping> <filter-
name>springSecurityFilterChain</filter-name> <url-pattern>/*</url-
pattern> </filter-mapping> <!-- Spring Security Ends -->
```

Add this Servlet filter as the first filter in the chain (at the top of the `web.xml` file).

3 Point the application to the security backend

No Spring Security XML file is required. Add this in a file named `ldap.conf` in `src/main/resources/conf/security`:

```
ldap-server-url: ldap://sharepoint-dc-mtyy7623.cloudapp.net:389/
manager-dn: sharepoint-dev@sharepoint-dev.twigkit.com manager-
password: abcdef123* user-search-base: CN=Users,dc=sharepoint-
dev,dc=twigkit,dc=com user-search-filter: (uid={0}) group-role-
attribute: cn group-search-base: CN=Groups,DC=sharepoint-
dev,DC=twigkit,DC=com group-search-filter: (member={0}) role-prefix:
ROLE_
```

The first three settings configure the LDAP server's location and manager credentials. The manager is the account used by Spring Security to access the LDAP server. The DN for this user should be available from the LDAP server or directory containing the user.

The remaining settings reference the LDAP server configuration to be used and provide the attributes required to determine the username and roles.

- `user-search-base` is the DN used under which users are retrieved
- `user-search-filter` is the property used to determine a match against the username specified by the user
- `group-role-attribute` is the property from which the name of the roles will be populated
- `group-search-base` is the DN used under which groups are retrieved
- `group-search-filter` is the property used to determine the group membership of a user
- `role-prefix` is a prefix applied to the roles reported by Spring Security

The roles picked up from LDAP can later be used for security trimming and personalization.

THE LDAP SERVER PROTOCOL

Note the use of the LDAP protocol in the URL - `ldap://`. In some cases SSL encryption is used with the 'LDAPS' protocol. This requires the certificate is trusted by the authentication client (the JVM running Appkit). There are several ways to configure the keystore for the JVM to trust the certificate from the LDAP server. The most common way to do this is to import the certificate into the JVM's default keystore.

4 Verify the configuration

You can now verify the authentication in an Appkit application using the *widget:login-form* on a login page which is typically located in `src/main/webapp/login.jsp` in your application source tree.

Related articles

- *Set Up Active Directory Authentication*
- *Set Up Lucidworks Fusion Authentication*
- *Set Up HTTP Headers Authentication*
- *Set Up Kerberos Authentication*
- *Set Up SiteMinder Authentication*
- *Set Up Spring Authentication*
- *Set Up Tomcat Container Authentication*
- *Set Up Windows Authentication*

Set Up Lucidworks Fusion Authentication

How-to

> Compatible versions: 4.2, 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *App Studio*

The Fusion security provider for Appkit allows authentication against Fusion using the Fusion Sessions API. With this provider configured, an application deployed within a Fusion defined 'realm' such as `native` can securely authenticate a user against Fusion.

These sections describe how to set up Fusion authentication using the Sessions API within an Appkit application.

1. Add the security provider dependency

Authentication against Fusion using the Sessions API requires Appkit's Fusion security provider module. To enable this module, first remove any existing security provider dependency from the `pom.xml` under the root of the project and insert this within the `dependencies` tag:

```
<dependency> <groupId>twigkit</groupId>
<artifactId>twigkit.security.provider.fusion</artifactId>
<version>${project.parent.version}</version> </dependency>
```

2. Update security.conf and add fusion.conf

To invoke this module when Appkit starts up, add this to `security.conf` in `src/main/resources/conf/security/`:

```
type: spring_security
```

Then configure the session host, which security realm to use and the session timeout in `fusion.conf` in `src/main/resources/conf/security/fusion`. For example:

```
session-host: http://localhost:8764 session-timeout: 30 realm-name:
native
```

The session-timeout parameter should be a time in minutes and be equal to your Fusion session timeout. By default, this value is 10. For further information, refer to the Fusion Session Api documentation.

3. Configure security filtering in the application

Spring Security operates using a Servlet filter which must be mapped in the web.xml file in `src/main/webapp/WEB-INF`:

```
<!-- Spring Security --> <context-param> <param-
name>contextConfigLocation</param-name> <param-value>classpath:spring-
security.xml</param-value> </context-param> <listener> <listener-
class>org.springframework.web.context.ContextLoaderListener</listener-
class> </listener> <filter> <filter-
name>springSecurityFilterChain</filter-name> <filter-
class>org.springframework.web.filter.DelegatingFilterProxy</filter-
class> </filter> <filter-mapping> <filter-
name>springSecurityFilterChain</filter-name> <url-pattern>/*</url-
pattern> </filter-mapping> <!-- Spring Security Ends -->
```

This should be added as the first filter in the chain (at the top of the web.xml file).

4. Instruct the application to use the Fusion security provider

For integration with security providers supported by Spring Security, configuration is managed in the *spring-security.xml* file in `src/main/resources`.

Leave the initial section of this file dealing with HTTP URL patterns as configured. Remove any existing authentication manager configuration below that initial section, and add this:

```
<!-- Fusion authentication provider --> <beans:bean
id="fusionAuthenticationProvider"
class="twigkit.security.fusion.FusionAuthenticationProvider"/> <!--
Authentication manager configuration --> <authentication-manager>
<authentication-provider ref="fusionAuthenticationProvider"/>
</authentication-manager>
```

Now when a user logs into Appkit, if they have been successfully authenticated against Fusion, they will receive a session cookie that can be re-used to send subsequent requests. Access to Fusion in this way can extend for up to 10 minutes if there is no activity and up to a maximum of 8-hours if a request is received within a 10-minute interval.

Related articles

- *Set Up Active Directory Authentication*
- *Set Up HTTP Headers Authentication*
- *Set Up Kerberos Authentication*
- *Set Up LDAP Authentication*
- *Set Up SiteMinder Authentication*
- *Set Up Spring Authentication*
- *Set Up Tomcat Container Authentication*

- *Set Up Windows Authentication*

Set Up NT Lan Manager Authentication for Confluence Connector

How-to

> Compatible versions: 4.2
>
> Works with: *Fusion Connectors*, *Fusion Server*, *Fusion*

This topic describes how to configure a Confluence site and authenticate with NT Lan Manager (NTLM) to use the Fusion connector.

Configure Active Directory for Confluence

Add a new directory with the following settings:

- **Name.** Directory name.
- **Directory Type. Microsoft Active Directory**.
- **Hostname.** Hostname of server running Lightweight Directory Access Protocol (LDAP).
- **Port.** Port number.
- **Username.** LDAP user login.
- **Password.** LDAP user password.
- **Base DN.** Distinguished Name (DN) of the LDAP object that is the root node from which to search for users and groups.
- **Additional User DN.** DN prepended to the base DN to limit user search scope.
- **Additional Group DN.** DN prepended to the base DN to limit group search scope.
- **Permissions > Read/Write**

Create authenticating account

1. Access the **Server Manager** in the Active Directory.
2. Select **Roles > Active Directory Domain Services**.
3. Select **Active Directory Users and Computers** to expand the node.
4. Expand the directory and right-click **Computers** to create the new account.
5. Select the **Member of** tab.
6. Select **Domain Computers**.
7. Select the **General** tab and enter values in each field.

> **Note**
>
> The **Computer name** field is required.

Configure delegation for the authenticating account

1. Access the authenticating account and select the **Delegation** tab.
2. Select **Trust this computer for delegation to specified services only.**
3. In the **Trust** field, select **Use any authentication protocol**.
4. Select **Add**.
5. In the Add Services window, select **Users or Computers**.
6. Select the server running the netlogon service from the results list and select **OK**.
7. In the **Service Type** column, select **netlogon** and select **OK**.

> **Note**
>
> The Delegation tab displays the netlogon service available for the account.

8. Save the following script to the Active Directory server:

```
curl -v -L --ntlm -H 'User-Agent: Mozilla/5.0 (compatible;
LucidWorks-Anda/4.0)' --negotiate -u 'administrator:FroFro123#'
'http://192.168.1.82:8090/download/attachments/65601/044451.html
?version=1&modificationDate=1558624541596&api=v2&download=true'
```

9. Execute the command with the hostname and password to set:
   `SetComputerPassword.vbs Confluence$@WIN-424E42TCKBB FroFro123#`
   The following is a sample result:

```
' Copyright (c) 2018, IOPLEX Software ' ' All rights reserved. '
' Redistribution and use in source and binary forms, with or
without ' modification, are permitted provided that the
following conditions ' are met: ' ' * Redistributions of source
code must retain the above copyright ' notice, this list of
conditions and the following disclaimer. ' ' * Redistributions
in binary form must reproduce the above copyright ' notice, this
list of conditions and the following disclaimer in the '
documentation and/or other materials provided with the
distribution. ' ' * Neither the name of IOPLEX Software nor the
names of its ' contributors may be used to endorse or promote
products derived from ' this software without specific prior
written permission. ' ' THIS SOFTWARE IS PROVIDED BY THE
COPYRIGHT HOLDERS AND CONTRIBUTORS "AS ' IS" AND ANY EXPRESS OR
IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, ' THE IMPLIED
WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR '
PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL COPYRIGHT OWNER OR
CONTRIBUTORS ' BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
SPECIAL, EXEMPLARY, ' OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
NOT LIMITED TO, PROCUREMENT OF ' SUBSTITUTE GOODS OR SERVICES;
LOSS OF USE, DATA, OR PROFITS; OR BUSINESS ' INTERRUPTION)
HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN '
CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR
OTHERWISE) ' ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE,
EVEN IF ADVISED OF ' THE POSSIBILITY OF SUCH DAMAGE. Option
Explicit Dim strPrinc, names, objComputer If
WScript.arguments.count <> 2 Then WScript.Echo "Usage:
SetComputerPassword.vbs <ComputerPrincipalName> <Password>"
WScript.Quit End If strPrinc = WScript.arguments.item(0) names =
Split(strPrinc,"@") If Ubound(names) <> 1 Or
InStrRev(names(0),"$") <> Len(names(0)) Then WScript.Echo
"Error: The Computer principal name must be in principal form
such as with a $ and @ signs (such as jespa1$@busicorp.local)."
```

```
WScript.Quit End If Set objComputer = GetObject("WinNT://" &
names(1) & "/" & names(0)) objComputer.GetInfo
objComputer.SetPassword WScript.arguments.item(1)
objComputer.SetInfo WScript.Echo "The password was set
successfully." WScript.Quit
```

See *Confluence Connector Configuration Reference* for more information.

Install and configure EasySSO

1. Access **General Configuration > Find New Apps**.
2. Search for **NTLM** and select the **EasySSO Admin** app to install it.
3. In the **jespa Licensing** section, select the latest `jespa.zip` file and download the file.
4. Install the file and buy a license.
5. Enter values in the following fields to configure the app:
    - **Domain.** Fully-qualified domain name (FQDN) of your domain.
    - **Account.** Active directory authentication account.
    - **Password.** Authentication account password.
6. Select **Save**.
7. Select **Test Connection** to verify NTLM authentication with the account logs in to Confluence.

Set Up Office 365 Authentication

How-to

> Compatible versions: 4.2, 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *App Studio*

To authenticate against Office 365, perform the steps in this article.

1 Add the security provider dependency

To add Office 365 as a security provider in the Oauth Security module:

1. Remove any existing security provider dependency from the `pom.xml` under the root of the project.
2. Add a security provider dependency for Office 365 to the `dependencies` tag in `pom.xml` :

```
<dependency> <groupId>twigkit</groupId>
<artifactId>twigkit.security.provider.oauth.office365</artifactI
d> <version>${project.parent.version}</version> </dependency>
```

3. Configure Appkit to invoke the Oauth Security module on startup. Change the `security.conf` file in `src/main/resources/conf/security/` to contain:

```
type: oauth
```

Note

You must remove any existing `spring-security.xml` file because this module encapsulates all Spring configuration automatically.

2 Configure the OAuth module for the application setup in Azure AD

Add the relevant configuration in a file in `conf/security/oauth.conf` :

```
azure-ad-tenant-id: 746a834e-ba89-4191-926d-f2c220b79a4a resource:
https://your-organisation.sharepoint.com/ client-id: 08de5b32-569a-
4387-ab94-08b7b6fc1ed8 client-secret:
x7vRc8VJ9CHjSM5XXyHpCSRaD5JH/6VsB8P5ZGz4B8M=
```

This guide assumes these settings have already been set up in the Azure AD instance in Microsoft's Azure portal and provided to you as an application developer.

Note

The reply URL must be configured in Azure AD as `http://your-application-url/oauthLogin`.

3 Add the Spring filter to the web.xml file

Add this to the `web.xml` file of the project:

```
<!-- Spring Security --> <context-param> <param-
name>contextConfigLocation</param-name> <param-value>classpath:spring-
security.xml</param-value> </context-param> <listener> <listener-
class>org.springframework.web.context.ContextLoaderListener</listener-
class> </listener> <filter> <filter-
name>springSecurityFilterChain</filter-name> <filter-
class>org.springframework.web.filter.DelegatingFilterProxy</filter-
class> </filter> <filter-mapping> <filter-
name>springSecurityFilterChain</filter-name> <url-pattern>/*</url-
pattern> </filter-mapping> <listener> <listener-class>
org.springframework.web.context.request.RequestContextListener
</listener-class> </listener> <!-- Spring Security Ends -->
```

Inclusion of a 'RequestContextListener' is not required in a standard Appkit-plus-Spring Security application.

4 Test the authentication

If a user is not logged into Office 365 prior to visiting the application, then they will be redirected to the Microsoft Online login page.

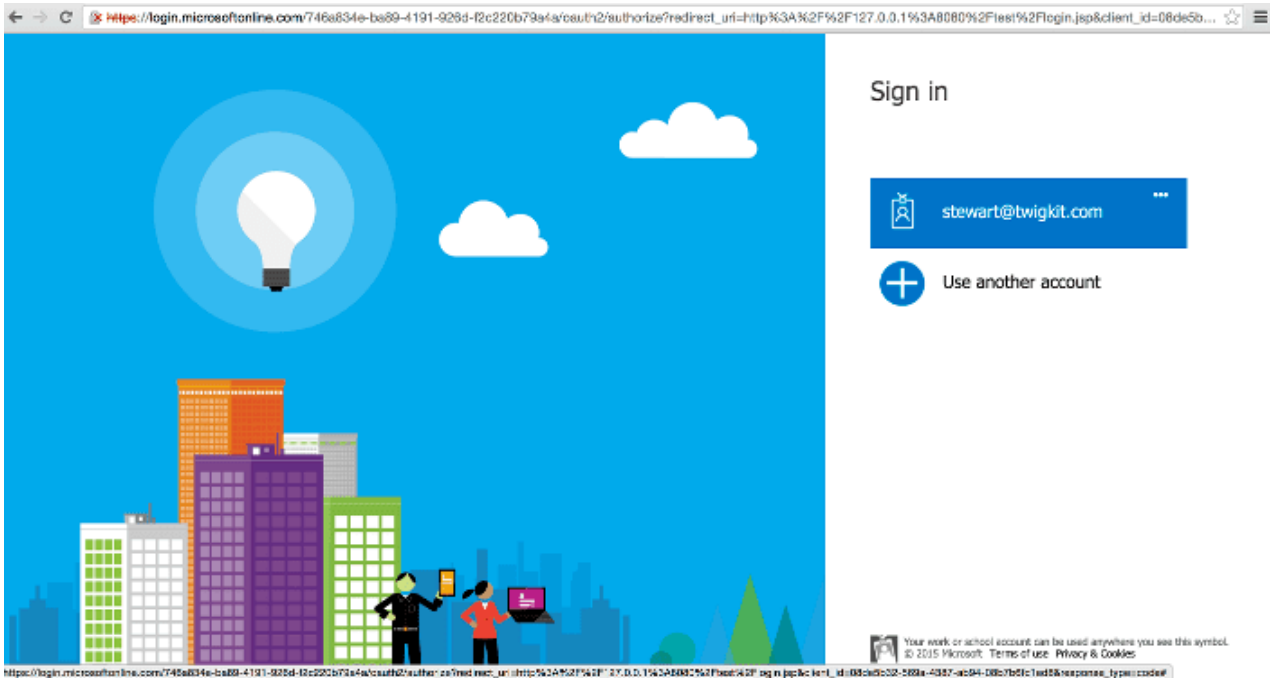
If the Azure administrator has set app approval explicitly, after the user is logged in, the user will be returned to the application as an authenticated user.

The Appkit user's details will be populated with any basic information available from the decoded OAuth token such as first name, last name and email address.

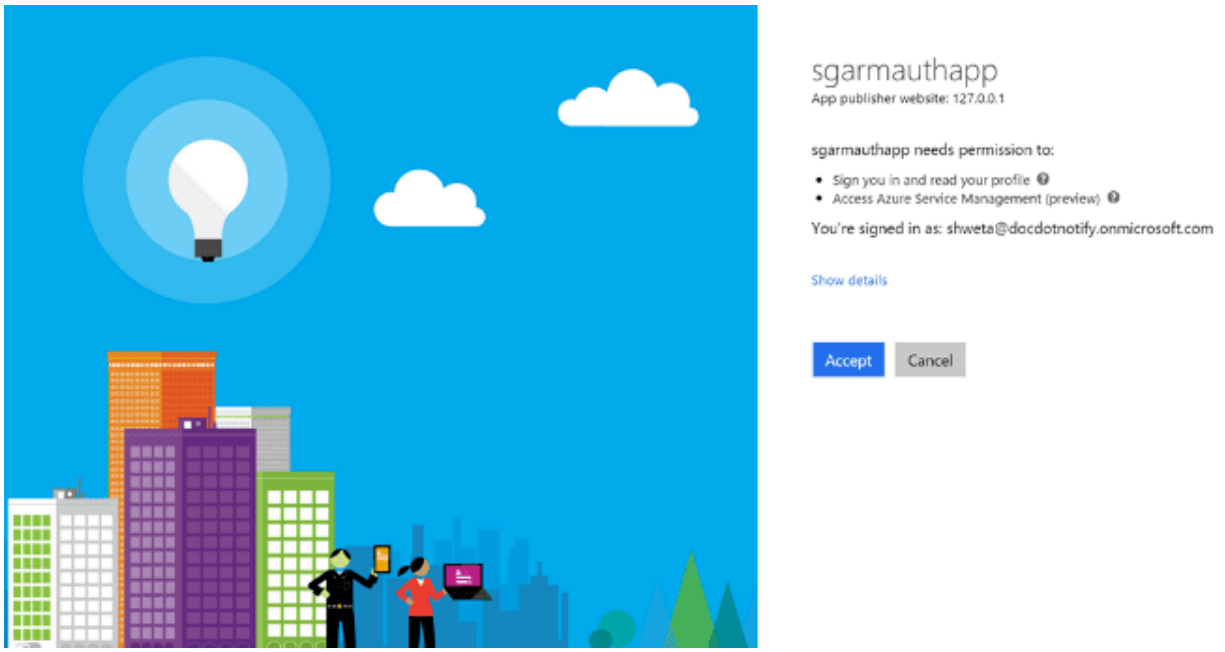
If approval is not set by the Azure AD administrator, then there will be an intermediary step after the user logs into Microsoft Online requesting their explicit approval for the application to access their credentials.

Using OAuth can create a seamless experience where, after the token handshake and approval is complete, the user will always be logged into the application as long as their session with Microsoft Online/Office 365 is active.

User logs in:

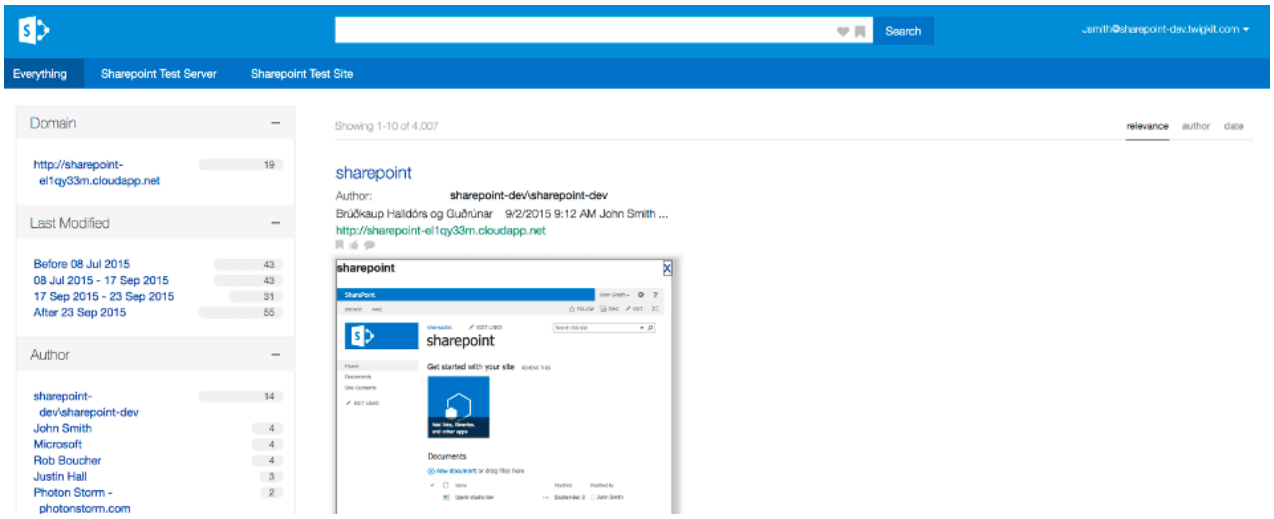


User approves application:



(This is optional, depending on the Azure AD configuration.)

The user is signed into the application with an OAuth token, and will be signed in automatically from now on unless the token expires or the user logs out of Office 365 entirely.



Related articles

- *Set Up Active Directory Federation Services (ADFS) Authentication*
- *Set Up Facebook Authentication*
- *Set Up Google Authentication*

Set Up Persisting Alerts

How-to

> Compatible versions: 4.2, 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *App Studio*

By default, the alerting module produces "in-memory" alerts, meaning that any alerts registered will only be sent for the duration of the application lifecycle.

For static configuration based alerts this is not an issue as they are registered again on application startup. However for alerts configured by a user with the 'saved query alert form' all alerts will be lost when the application is shut down or redeployed. Therefore on a production system, it makes sense to configure persistence of alerts using a database. This is simple to configure with these steps.

1: Configure your alerting module database

1. Add this dependency to your `pom.xml` :

```
<dependency> <groupId>twigkit</groupId>
<artifactId>twigkit.social.runtime</artifactId>
<version>${project.parent.version}</version> </dependency>
<dependency> <groupId>twigkit</groupId>
<artifactId>twigkit.alert.db.derby.memory</artifactId>
<version>${project.parent.version}</version> </dependency>
```

2. Remove any existing reference to `twigkit.social` modules. Currently the in-memory database and MySQL are supported (`twigkit.alert.db.mysql`).

2: Add alerting module ORM mapping to your persistence configuration

Assuming you have configured your social datastore in a `persistence.xml` file, add this entry right after the existing `mapping-file` XML element:

```
<mapping-file>META-INF/alert/orm.xml</mapping-file>
```

3: Tell the Quartz scheduler how to connect to the database

Add a `quartz.properties` file to the root of your class path (typically, in `src/main/resources` when building a Maven project form source) that contains the database configuration for the Quartz scheduler.

This is a *sample* `quartz.properties` *file* with placeholder values for the database-specific parameters you must fill in. For example:

```
org.quartz.scheduler.instanceName = AlertScheduler
org.quartz.threadPool.threadCount = 4 org.quartz.jobStore.class =
org.quartz.impl.jdbcjobstore.JobStoreTX
org.quartz.jobStore.driverDelegateClass =
org.quartz.impl.jdbcjobstore.StdJDBCDelegate
org.quartz.jobStore.dataSource = AlertDataSource
org.quartz.dataSource.AlertDataSource.driver =
${javax.persistence.jdbc.driver}
org.quartz.dataSource.AlertDataSource.URL =
${javax.persistence.jdbc.url}
org.quartz.dataSource.AlertDataSource.user =
${javax.persistence.jdbc.user}
org.quartz.dataSource.AlertDataSource.password =
${javax.persistence.jdbc.password}
org.quartz.dataSource.AlertDataSource.maxConnections = 30
org.quartz.jobStore.tablePrefix = qrtz_
```

Replace the variables in curly braces on the right with their respective values from your `persistence.xml` file.

Related articles

- *Set Up Email Alerts*
- *Set Up Static Alerts*
- *Set Up User-specified Alerts*

Set up semantic vector search

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

Note

These instructions are for Milvus and do not work with Solr SVS.

These are the suggested steps to get started with semantic vector search (SVS).

1. Craft training data.
   The training data fed to SVS models creates connections between consumer queries and products from a catalog. Crafting the training data is the biggest influence in terms of the performance and quality a SVS model can deliver.
2. Diagnose Zero Search Result themes.
   Identify the common themes across top zero result queries to craft the right kinds of training data appropriate for the kinds of problem themes.
   This step involves looking at a report of the top zero search result queries and grouping them into categories such as:
   - In Stock/Out of Stock
   - Product Not Carried
   - Foreign Vocabulary
   - Misspelling
3. Create Zero Search Result training pairs.
   A Zero Search Result Query (ZRQ) is the most obvious indicator that a search engine is struggling to retrieve relevant results, so create training data pairs geared toward learning what consumers were looking for when submitting such terms. Zero Search Result (ZSR) training pairs are created by looking at what a consumer did after getting to a zero results page. More specifically, look at what products a consumer adds to cart following a zero results page. For a ZSR query, a subset will try to search another way, either through using different search terms or by browsing category pages. This is referred to as "Learning from the Persistent Shopper." In the end, pairs of data are created which look like `ZSR Query, Next Item Added-to-Cart`.
4. Create Abandoned Search training pairs.
   Another indicator for troublesome queries or poor relevance are Abandoned Searches (AS). An AS is defined when a consumer submits a query and receives search results but does not click on any of those results. Look at what a consumer adds to their cart following an AS. The pairs of this type follow the format of `AS query, Next Item Added-to-Cart`.
5. Create Converted Search training pairs.
   The two prior types of training data outline a circumventive consumer path to find relevant results. These circumventive paths are essential to enhance the coverage of the SVS models in terms of edge cases and understanding foreign vernacular to the product catalog. The final type of training data, Converted Search (CS), outlines the 'happy path' a consumer takes. A happy path occurs when a consumer submits a search, finds exactly what they are looking for in the search results, then adds that item to their cart.
   This type of training data is essential in enhancing the precision and accuracy of the SVS model by providing positively converting pairs where there is a direct correlation between the query submitted and the search results returned. This is also the simplest type of training data to create, as it is just an aggregation of already collected `Search Query, Item Added-to-Cart` pairs.
6. Apply a filter.
   The last step of the training data crafting process is to apply a filter to the aggregated training data pairs collection to remove noisy data pairs for ZRQ and AS types. A noisy data pair is defined by a pair in which the outlined search query has no connection to the next item added to cart. Noisy signals can manifest in several different ways, such as shopping from a list of items and not finding a product. Instead of being persistent and using different methods to find that or a similar product, a shopper might move on to the next, unrelated, item on their list. To remove noisy pairs from the training data collection, look at the pairs with low aggregation counts and selectively remove them.
7. Train the SVS models.
   After validating that the training data pairs were created appropriately, utilize this training data to build SVS deep learning models using supervised learning methodology. This means the pre-trained models are used as a base for understanding relationships between items in the product catalog, then the crafted training data is used to refine the baseline relationships specific to a business and its consumers. To maximize the effectiveness of these SVS models, conduct model trials (or iterations) to find the best configuration and training data combination.
8. Deploy model.

Upon completion of a Supervised Training Job, a model is deployed using Seldon-Core, an open source technology used to deploy and manage models within a Kubernetes cluster. This Seldon-Core model deployment is used both in the index and query pipeline procedures.

9. Index product catalog.
   Part of the implementation of SVS includes encoding the product catalog into a vector space by creating a vector for each product within the catalog and storing them in an appropriate collection. Milvus is an open source vector search similarity engine and is used to store the encoded vectors from the product catalog and to perform vector similarity searches in query pipelines.
10. Integrate with an index pipeline.
    To populate a Milvus collection with vectors from the catalog, add **Encode into Milvus** stage to the index pipeline. This **Encode into Milvus** stage vectorizes each product within the catalog and stores it inside a Milvus collection as the products are being indexed into a Solr collection. Each vector in a Milvus collection has a corresponding `milvus_id` stored as part of the product record in the Solr collection.
11. Set up a job in *Create collections in Milvus*.
    Give the job and collection a name and fill in the **Dimension** parameter. The **Dimension** parameter is always 2x the value input for the **RNN Function Units List** parameter in the Supervised Training job. For example, if the RNN Function Units List parameter is 128, then the Dimension parameter is 256.
12. Refresh the trained model.
    The trained model also needs a periodic refresh to account for changes in consumer behavior, such as seasonal changes. To refresh the SVS models, a new model is trained using the most recent consumer training data and the catalog is reindexed using the freshly created model and queries are switched to also be encoded using the new model.
13. Integrate with a query pipeline.
    Prepare the SVS solution for load testing and production by integrating vector search with query pipelines. There are three main components to this integration:
    - Ensure that the Vector Search component of the pipeline is only triggered when zero results are returned by the main query pipeline.
    - Ensure that the results returned by vector search maintain the functionality of results from the traditional pipeline, such as sorting and faceting.
    - Ensure that searches involving the vector search components are tagged appropriately in the generated signals for analytics purposes.

14. Conduct load testing.
    Test load based on needs, for example against a 20% null result rate at peak QPS plus comfort room (20% of 1200 QPS + comfort room = ~300 QPS for vector search). If the initial vector search configuration does not perform well against such benchmarks, consider configuration changes:
    - Optimization of resource allocation
    - Turning off Solr spell check within the vector search pipeline. Removing spell check is the main configuration change needed to drive the query response times down to acceptable levels.

Set Up Semantic Vector Search

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion*

Note

These instructions are for Milvus and do not work with Solr SVS. In Fusion 5.9.5 and above, see *Configure Neural Hybrid Search* to set up semantic vector search.

These are the suggested steps to get started with Semantic Vector Search (SVS).

1. Craft training data.
   The training data fed to SVS models creates connections between consumer queries and products from a catalog. Crafting the training data is the biggest influence in terms of the performance and quality a SVS model can deliver.
2. Diagnose Zero Search Result themes.
   Before create training data pairs, first identify the common themes across top zero result queries to craft the right kinds of training data appropriate for the kinds of problem themes.
   This step involves looking at a report of the top zero search result queries and grouping them into categories such as:
   - In Stock/Out of Stock
   - Product Not Carried
   - Foreign Vocabulary
   - Misspelling
3. Create Zero Search Result training pairs.
   A Zero Search Result Query (ZRQ) is the most obvious indicator that a search engine is struggling to retrieve relevant results, so create training data pairs geared toward learning what consumers were looking for when submitting such terms. Zero Search Result (ZSR) training pairs are created by looking at what a consumer did after getting to a zero results page. More specifically, look at what products a consumer adds to cart following a zero results page. For a ZSR query, a subset will try to search another way, either through using different search terms or by browsing category pages. This is referred to as "Learning from the Persistent Shopper." In the end, pairs of data are created which look like `ZSR Query, Next Item Added-to-Cart`.
4. Create Abandoned Search training pairs.
   Another indicator for troublesome queries or poor relevance are Abandoned Searches (AS). An AS is defined when a consumer submits a query and receives search results but does not click on any of those results. Look at what a consumer adds to their cart following an AS. The pairs of this type follow the format of `AS query, Next Item Added-to-Cart`.
5. Create Converted Search training pairs.
   The two prior types of training data outline a circumventive consumer path to find relevant results. These circumventive paths are essential to enhance the coverage of the SVS models in terms of edge cases and understanding foreign vernacular to the product catalog. The final type of training data, Converted Search (CS), outlines the 'happy path' a consumer takes. A happy path occurs when a consumer submits a search, finds exactly what they are looking for in the search results, then adds that item to their cart.
   This type of training data is essential in enhancing the precision and accuracy of the SVS model by providing positively converting pairs where there is a direct correlation between the query submitted and the search results returned. This is also the simplest type of training data to create, as it is just an aggregation of already collected `Search Query, Item Added-to-Cart` pairs.
6. Apply a filter.
   The last step of the training data crafting process is to apply a filter to the aggregated training data pairs collection to remove noisy data pairs for ZRQ and AS types. A noisy data pair is defined by a pair in which the outlined search query has no connection to the next item added to cart. Noisy signals can manifest in several different ways, such as shopping from a list of items and not finding a product. Instead of being persistent and using different methods to find that or a similar product, a shopper might move on to the next, unrelated, item on their list. To remove noisy pairs from the training data collection, look at the pairs with low aggregation counts and selectively remove them.
7. Train the SVS models.
   After validating that the training data pairs were created appropriately, utilize this training data to build SVS deep learning models using supervised learning methodology. This means the pre-trained models are used as a base for understanding relationships between items in the product catalog, then the crafted training data is used to refine the baseline relationships specific to a business and its consumers. To maximize the effectiveness of these SVS models, conduct model trials (or iterations) to find the best configuration and training data combination.

8. Deploy model.
   Upon completion of a Supervised Training Job, a model is deployed using Seldon-Core, an open source technology used to deploy and manage models within a Kubernetes cluster. This Seldon-Core model deployment is used both in the index and query pipeline procedures.
9. Index product catalog.
   Part of the implementation of SVS includes encoding the product catalog into a vector space by creating a vector for each product within the catalog and storing them in an appropriate collection. Milvus is an open source vector search similarity engine and is used to store the encoded vectors from the product catalog and to perform vector similarity searches in query pipelines.
10. Integrate with an index pipeline.
    To populate a Milvus collection with vectors from the catalog, add **Encode into Milvus** stage to the index pipeline. This **Encode into Milvus** stage vectorizes each product within the catalog and stores it inside a Milvus collection as the products are being indexed into a Solr collection. Each vector in a Milvus collection has a corresponding `milvus_id` stored as part of the product record in the Solr collection.
11. Set up a **Create Collections in Milvus** job.
    Give the job and collection a name and fill in the **Dimension** parameter. The **Dimension** parameter is always 2x the value input for the **RNN Function Units List** parameter in the Supervised Training job. For example, if the RNN Function Units List parameter is 128, then the Dimension parameter is 256.
12. Refresh the trained model.
    The trained model also needs a periodic refresh to account for changes in consumer behavior, such as seasonal changes. To refresh the SVS models, a new model is trained using the most recent consumer training data and the catalog is reindexed using the freshly created model and queries are switched to also be encoded using the new model.
13. Integrate with a query pipeline.
    Prepare the SVS solution for load testing and production by integrating vector search with query pipelines. There are three main components to this integration:
    - Ensure that the Vector Search component of the pipeline is only triggered when zero results are returned by the main query pipeline.
    - Ensure that the results returned by vector search maintain the functionality of results from the traditional pipeline, such as sorting and faceting.
    - Ensure that searches involving the vector search components are tagged appropriately in the generated signals for analytics purposes.

14. Conduct load testing.
    Test load based on needs, for example against a 20% null result rate at peak QPS plus comfort room (20% of 1200 QPS + comfort room = ~300 QPS for vector search). If the initial vector search configuration does not perform well against such benchmarks, consider configuration changes:
    - Optimization of resource allocation
    - Turning off Solr spell check within the vector search pipeline. Removing spell check is the main configuration change needed to drive the query response times down to acceptable levels.

Set Up SiteMinder Authentication

How-to

> Compatible versions: 4.2, 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *App Studio*

Appkit can integrate with a SiteMinder single-sign-on authentication provider. SiteMinder is typically installed in a web server placed in front of the application server that runs Appkit.

To enable SiteMinder authentication:

1. Add this module to your POM (`pom.xml` file in the root of the application):

```
<dependency> <groupId>twigkit</groupId>
<artifactId>twigkit.security.provider.siteminder</artifactId>
<version>${project.parent.version}</version> </dependency>
```

2. Update your `security.conf` (found in `src/main/resources/conf/security/`) to use SiteMinder:

```
type: siteminder
```

Related articles

- *Set Up Active Directory Authentication*
- *Set Up Lucidworks Fusion Authentication*
- *Set Up HTTP Headers Authentication*
- *Set Up Kerberos Authentication*
- *Set Up LDAP Authentication*
- *Set Up Spring Authentication*
- *Set Up Tomcat Container Authentication*
- *Set Up Windows Authentication*

Set Up Spring Authentication

How-to

> Compatible versions: 4.2, 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *App Studio*

The Spring Security authentication provider for Appkit interfaces with both Spring Security 3 and 4. Projects that adopt Appkit version 3 must switch to using Spring Security 4.

Authorized Spring principals become fully-qualified Appkit user profiles, along with any roles or groups that the Spring module provides. Spring Security integrates with a number of authentication protocols, including LDAP, Active Directory, Basic Authentication, Kerberos, JAAS, and others. For a full list, see here.

Below you will find some common configurations of the `spring-security.xml` file for integrating with different security mechanisms for authentication and authorization.

Note

The references given are for Spring Security 3. For information on how these configurations would change with Spring Security 4, refer to the Release Notes for Appkit 3.

The first portion of the configuration file contains the required access levels for URL patterns used in Appkit and in most cases stays the same for all configurations.

Hard-coded users

```
<?xml version="1.0" encoding="UTF-8"?> <beans:beans
xmlns="http://www.springframework.org/schema/security"
xmlns:beans="http://www.springframework.org/schema/beans"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.springframework.org/schema/beans
          http://www.springframework.org/schema/beans/spring-beans-
3.0.xsd
          http://www.springframework.org/schema/security
          http://www.springframework.org/schema/security/spring-
security-3.1.xsd"> <http pattern="/styles/**" security="none"/> <http
pattern="/assets/**" security="none"/> <http pattern="/javascript/**"
security="none"/> <http pattern="/wro/**" security="none"/> <http use-
expressions="true" disable-url-rewriting="true"> <intercept-url
pattern="/login/" access="isAnonymous()"/> <intercept-url
pattern="/twigkit/resources/**" access="isAnonymous() or
isAuthenticated()"/> <intercept-url
pattern="/twigkit/services/gsa/auth/**" access="isAnonymous() or
isAuthenticated()"/> <intercept-url pattern="/mock-response/search"
access="isAnonymous() or isAuthenticated()"/> <intercept-url
pattern="/**" access="isAuthenticated()"/> <form-login login-
page="/login/" authentication-failure-url="/login/?access=denied"/>
<http-basic/> <logout logout-url="/logout/" logout-success-url="/"/>
</http> <!-- Simple Configuration Example --> <authentication-manager>
<authentication-provider> <user-service> <user name="admin"
password="admin" authorities="USER, ADMIN"/> <user name="user"
password="user" authorities="USER"/> </user-service> </authentication-
provider> </authentication-manager> </beans:beans>
```

Here, you can see that two users are configured - 'user' and 'admin'. Both have been assigned the role named 'USER' and the admin user also has the 'ADMIN' role.

LDAP

```
<?xml version="1.0" encoding="UTF-8"?> <beans:beans
xmlns="http://www.springframework.org/schema/security"
xmlns:beans="http://www.springframework.org/schema/beans"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.springframework.org/schema/beans
          http://www.springframework.org/schema/beans/spring-beans-
3.0.xsd
          http://www.springframework.org/schema/security
          http://www.springframework.org/schema/security/spring-
security-3.1.xsd"> <http pattern="/local-data/**" security="none"/>
<http pattern="/styles/**" security="none"/> <http
pattern="/assets/**" security="none"/> <http pattern="/javascript/**"
security="none"/> <http pattern="/wro/**" security="none"/> <http use-
expressions="true" disable-url-rewriting="true"> <intercept-url
pattern="/login/" access="isAnonymous()"/> <intercept-url
pattern="/twigkit/resources/**" access="isAnonymous() or
isAuthenticated()"/> <intercept-url
pattern="/twigkit/services/gsa/auth/**" access="isAnonymous() or
isAuthenticated()"/> <intercept-url pattern="/mock-response/search"
access="isAnonymous() or isAuthenticated()"/> <intercept-url
pattern="/**" access="isAuthenticated()"/> <form-login login-
page="/login/" authentication-failure-url="/login/?access=denied"/>
<http-basic/> <logout logout-url="/logout/" logout-success-url="/"/>
</http> <!-- Active Directory Configuration --> <ldap-server
id="ldapServer" url="ldap://my-ldap.my-domain.com:389/" manager-
dn="gsaindexing@my-domain.com" manager-password="ABC1234567890*" />
<authentication-manager> <ldap-authentication-provider server-
ref="ldapServer" user-search-base="OU=[Root],DC=uk,DC=corp,DC=my-
domain,DC=com" user-search-filter="sAMAccountName={0}" group-role-
attribute="cn" group-search-base="OU=[Root],DC=uk,DC=corp,DC=my-
domain,DC=com" group-search-filter="(member={0})" role-prefix="ROLE_"
/> </authentication-manager> </beans:beans>
```

Note the use of the LDAP protocol. In some cases, SSL encryption is used with the 'LDAPS' protocol. This requires the certificate to be trusted by the authentication client (the JVM running Appkit). There are several ways to configure the keystore for the JVM to trust the certificate from the LDAP server. The most common way to do this is to import the certificate into the JVM's default keystore .

Spring 'remember me' feature

Spring Security comes with a mechanism for allowing a user's details to be remembered beyond the current browser session, avoiding the user having to provide credentials on a login page every time. This integrates with the 'remember' attributes on the *widget:login-form* tag.

There are two methods to allow the user's session to be persistent using either a browser cookie or a database backed persistence method. The first is most commonly used, and to implement this all that is required is to add the remember-me line to the `spring-security.xml` HTTP configuration:

```
... <http use-expressions="true" disable-url-rewriting="true">
<intercept-url pattern="/login/" access="isAnonymous()"/> <intercept-
url pattern="/twigkit/resources/**" access="isAnonymous() or
isAuthenticated()"/> <intercept-url
pattern="/twigkit/services/gsa/auth/**" access="isAnonymous() or
isAuthenticated()"/> <intercept-url pattern="/mock-response/search"
access="isAnonymous() or isAuthenticated()"/> <intercept-url
pattern="/**" access="isAuthenticated()"/> <form-login login-
page="/login/" authentication-failure-url="/login/?access=denied"/>
<http-basic/> <logout logout-url="/logout/" logout-success-url="/"/>
<remember-me key="KEY-U5ED-F0R-C00KIE-ENCRYPT1ON"/> </http> ...
```

Re-directing users to the original requested URL post-login

Spring Security provides an authentication strategy that allows a requested URL to be saved before requiring the user to log in; once credentials have been authenticated the user is re-directed to the original URL.

This strategy is implemented as part of the `<form-login …>` tag defined via the `authentication-success-handler-ref`. For example:

```
<form-login login-page="/login/" authentication-failure-url="/login/?
access=denied" authentication-success-handler-
ref="login_success_handler"/>
```

The `login_success_handler` is a reference to the Spring Security class that implements the described strategy. This class is defined via this bean that needs to be added to the `spring-security.xml` as follows:

```
<http..> ... </http> <beans:bean id="login_success_handler"
class="org.springframework.security.web.authentication.SavedRequestAwa
reAuthenticationSuccessHandler"/>
```

Related articles

- *Set Up Active Directory Authentication*
- *Set Up Lucidworks Fusion Authentication*
- *Set Up HTTP Headers Authentication*
- *Set Up Kerberos Authentication*
- *Set Up LDAP Authentication*
- *Set Up SiteMinder Authentication*
- *Set Up Tomcat Container Authentication*
- *Set Up Windows Authentication*

Set Up Static Alerts

How-to

> Compatible versions: 4.2, 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *App Studio*

Email alerts can be configured statically in `conf` files to be sent at regular intervals throughout the lifecycle of an application. Use these steps to get up and running with the Alerting module:

1. Add this dependency to your `pom.xml` file:

```
<dependency> <groupId>twigkit</groupId>
<artifactId>twigkit.alert.runtime</artifactId>
<version>${project.parent.version}</version> </dependency>
```

2. Set up `conf/alert/email/smtp.conf`; see below for an example configuration:

```
# User to log into Appkit twigkit-user: foo twigkit-password:
bar # User to send emails as from: foobar@twigkit.com password:
foobar # Override to address for all config-based alerts to:
reporting@some-alerting-domain.com # SMTP details
mail.smtp.host: smtp.gmail.com mail.smtp.port: 587
mail.smtp.auth: true mail.smtp.starttls.enable: true
```

3. Set up `conf/alert/email/alerts/alerts.conf`; see below for an example configuration:

```
email-template: http://localhost:8080/email/
```

4. Add the alert `.conf` files to `conf/alert/email/alerts/my-alert-name-here.conf`; see below for an example:

```
to: foobar@twigkit.com subject: New documents on brewing query:
?q=homebrew cron-expression: 30 * * * * ?
```

   Here, the `cron-expression` defines how often to send the alert as described in this Quartz tutorial.

5. Create a JSP, in this case configured as above, on a standard Appkit application would be in `src/main/webapp/WEB-INF/pages/email.jsp` and the content would include:

```
<%@ page contentType="text/html" pageEncoding="UTF-8" %> <%@
include file="/WEB-INF/tags/client/taglibs.jspf" %>
<search:localization var="resources" bundleName="resources" />
<search:twigkit> <search:platform conf="platforms.solr" />
<search:query var="query" parameters="*" /> <search:response
var="response" platform="${platform}" query="${query}" />
</search:twigkit> <!DOCTYPE html> <html
xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
<h1>Automated alert for query ${query.value}</h1>
<search:result-list response="response"> <search:result>
<search:field name="title" /> <search:field name="description"
/> </search:result> </search:result-list> </html>
```

Related articles

- *Set Up Persisting Alerts*
- *Set Up Email Alerts*
- *Set Up User-specified Alerts*

Set Up Tomcat Container Authentication

How-to

> Compatible versions: 4.2, 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *App Studio*

Apache Tomcat is an open source Java servlet container. Tomcat implements Java enterprise specs, including the Java Servlet, JavaServer Pages JSP, and WebSockets APIs.

To have Tomcat manage authentication for your application:

1. Add this module to your POM (`pom.xml` file in the root of the application):

```
<dependency> <groupId>twigkit</groupId>
<artifactId>twigkit.security.provider.tomcat7</artifactId>
<version>${project.parent.version}</version> </dependency>
```

2. Set the requisite type in `security.conf` (found in `src/main/resources/conf/security/`):

```
type: tomcat7
```

This picks up the user and all roles from the configured Tomcat authentication mechanism. This is often used in Windows environments. More information on this type of authentication can be found in the Tomcat documentation.

Related articles

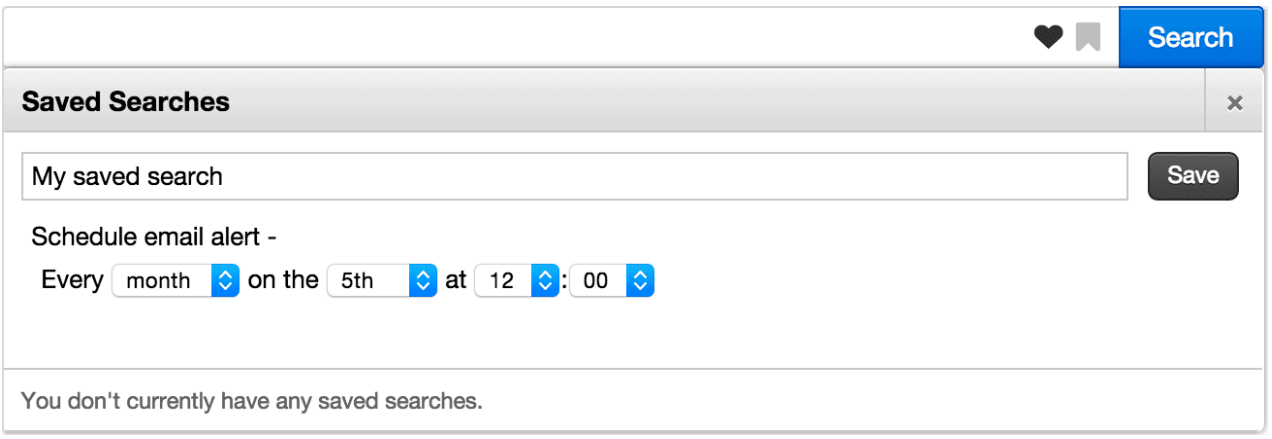
- *Set Up Active Directory Authentication*
- *Set Up Lucidworks Fusion Authentication*
- *Set Up HTTP Headers Authentication*
- *Set Up Kerberos Authentication*
- *Set Up LDAP Authentication*
- *Set Up SiteMinder Authentication*
- *Set Up Spring Authentication*
- *Set Up Windows Authentication*

Set Up User-specified Alerts

How-to

> Compatible versions: 4.2, 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *App Studio*

You can set up Appkit to provide user-specified email alerts in an app.



Setup

To set up user-specified alerts:

1. Add this dependency to your `pom.xml`. Remove any previous reference to `twigkit.social` modules.

```
<dependency> <groupId>twigkit</groupId>
<artifactId>twigkit.social.runtime</artifactId>
<version>${project.parent.version}</version> </dependency>
<dependency> <groupId>twigkit</groupId>
<artifactId>twigkit.alert.db.derby.memory</artifactId>
<version>${project.parent.version}</version> </dependency>
```

2. Add a configuration file in `conf/alert/email/smtp.conf`; see below for an example configuration:

```
# User to log into Appkit (service account) twigkit-user: admin
twigkit-password: password # Email account to use to send alert
emails from: foobar@twigkit.com password: foobar # SMTP details
mail.smtp.host: smtp.gmail.com mail.smtp.port: 587
mail.smtp.auth: true mail.smtp.starttls.enable: true
```

3. Add another configuration file for the email template in `conf/alert/email/alerts/alerts.conf`; see below for an example configuration:

```
email-template: http://localhost:8080/email/ subject-prefix: New
report for saved query
```

4. Add the saved query alert scheduling form to the search header (header.tag):

```
<social:searchHeader query="${query}" action="${action}"
logoImage="/assets/logo-google.png" logoHeight="42"
logoImageAlt="Appkit for GSA"> <jsp:attribute
name="searchFormBody"> <instant:result-list
platformConf="platforms.gsa"> <instant:result> <a href="
{{result.fields.url.val[0]}}"> <instant:field fieldName="title"
/> </a> </instant:result> </instant:result-list>
</jsp:attribute> <jsp:attribute name="searchControls"> <%--
Bookmarks --%> <widget:popover id="bookmark-list"
title="Bookmarks" linkText="" align="right">
<social:bookmarkList emptyText="You do not currently have any
bookmarks." user="${social:getCurrentUserProfile()}" date-
format="dd MMM yyyy" /> </widget:popover> <%-- Saved Searches --
%> <widget:popover id="saved-searches" title="Saved Searches"
linkText="" align="right"> <alert:savedQueryAlertForm
query="${query}" /> <alert:savedQueryAlertList
user="${social:getCurrentUserProfile()}" action="${action}"
date-format="dd MMM yyyy" emptyText="You do not currently have
any saved searches." /> </widget:popover> </jsp:attribute>
</social:searchHeader>
```

5. Create the email template JSP, in **`src/main/webapp/WEB-INF/pages/email.jsp`**. For example:

```
<%@ page contentType="text/html" pageEncoding="UTF-8" %> <%@
include file="/WEB-INF/tags/client/taglibs.jspf" %>
<search:twigkit> <search:platform var="delegatePlatform"
conf="platforms.elasticsearch" /> <alert:platform
delegatePlatform="${delegatePlatform}" /> <alert:query
dateField="updated" /> <alert:response var="response"
platform="${platform}" query="${query}"> </alert::response>
</search:twigkit> <!DOCTYPE html> <html
xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
<h1>Automated alert for query ${query.value}</h1>
<search:result-list response="response"> <search:result>
<search:field name="title" /> <search:field name="summary" />
</search:result> </search:result-list> </html>
```

Note the following:

- The 'alert seen' functionality is reliant on a field in the data specifying the date the document was last updated, in this case named `updated`
- The 'to address' of the email alert will either be chosen by the user via the form (default) or can be set by using the `toAddress` attribute on the saved query alert form tag.
- Appkit includes full user impersonation, so any security trimming is applied automatically.

Saved Query Alert Form tag attributes

`query (twigkit.model.Query)`
The Query object to be registered for alerting.

`defaultTitle (java.lang.String)`
Default title of a saved query (if no query supplied).

`scheduleAlertText (java.lang.String)`
Text to display above schedule component.

`toAddress (java.lang.String)`
The email address to which the alert should be sent. If not supplied (left empty), the user will be presented with a field to specify this (default behavior).

`toAddressPlaceholderText (java.lang.String)`
Placeholder text for 'to' email address field which is displayed if toAddress attribute is blank

`toAddressLabelText (java.lang.String)`
Label text for 'to' email address field which is displayed if toAddress attribute is blank

`deliveryFrequencyLabelText (java.lang.String)`
Text for label above scheduling component

Related articles

- *Set Up Persisting Alerts*
- *Set Up Email Alerts*

- *Set Up Static Alerts*

Set Up User-specified Email Alerts

How-to

> Compatible versions: 4.2
>
> Works with: *App Studio*

You can set up Appkit to provide user-specified email alerts in an app.

To set up user-specified alerts:

1. Add this dependency to your **`pom.xml`**. Remove any previous reference to **`twigkit.social`** modules.

```
<dependency> <groupId>twigkit</groupId>
<artifactId>twigkit.social.runtime</artifactId>
<version>${project.parent.version}</version> </dependency>
<dependency> <groupId>twigkit</groupId>
<artifactId>twigkit.alert.db.derby.memory</artifactId>
<version>${project.parent.version}</version> </dependency>
```

2. Add a configuration file in **`conf/alert/email/smtp.conf`**; see below for an example configuration:

```
# User to log into Appkit (service account) twigkit-user: admin
twigkit-password: password # Email account to use to send alert
emails from: foobar@twigkit.com password: foobar # SMTP details
mail.smtp.host: smtp.gmail.com mail.smtp.port: 587
mail.smtp.auth: true mail.smtp.starttls.enable: true
```

3. Add another configuration file for the email template in **`conf/alert/email/alerts/alerts.conf`**; see below for an example configuration:

```
email-template: http://localhost:8080/email/ subject-prefix: New
report for saved query
```

4. Add the saved query alert scheduling form to the search header (header.tag):

```
<social:searchHeader query="${query}" action="${action}"
logoImage="/assets/logo-google.png" logoHeight="42"
logoImageAlt="Appkit for GSA"> <jsp:attribute
name="searchFormBody"> <instant:resultList
platformConf="platforms.gsa"> <instant:result> <a href="
{{result.fields.url.val[0]}}"> <instant:field fieldName="title"
/> </a> </instant:result> </instant:resultList> </jsp:attribute>
<jsp:attribute name="searchControls"> <\%-- Bookmarks --%>
<widget:popover id="bookmark-list" title="Bookmarks" linkText=""
align="right"> <social:bookmarkList emptyText="You do not
currently have any bookmarks."
user="${social:getCurrentUserProfile()}" format="dd MMM yyyy" />
</widget:popover> <\%-- Saved Searches --%> <widget:popover
id="saved-searches" title="Saved Searches" linkText=""
align="right"> <alert:savedQueryAlertForm query="${query}" />
<alert:savedQueryAlertList
user="${social:getCurrentUserProfile()}" action="${action}"
format="dd MMM yyyy" emptyText="You do not currently have any
saved searches." /> </widget:popover> </jsp:attribute>
</social:searchHeader>
```

5. Create the email template JSP, in **`src/main/webapp/WEB-INF/pages/email.jsp`**. For example:

```
<%@ page contentType="text/html" pageEncoding="UTF-8" %> <%@
include file="/WEB-INF/tags/client/taglibs.jspf" %>
<search:twigkit> <search:platform var="delegatePlatform"
conf="platforms.elasticsearch" /> <alert:platform
delegatePlatform="${delegatePlatform}" /> <alert:query
dateField="updated" /> <alert:response var="response"
platform="${platform}" query="${query}"> </alert::response>
</search:twigkit> <!DOCTYPE html> <html
xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
<h1>Automated alert for query ${query.value}</h1>
<search:result-list response="response"> <search:result>
<search:field name="title" /> <search:field name="summary" />
</search:result> </search:result-list> </html>
```

Note the following:

- The 'alert seen' functionality is reliant on a field in the data specifying the date the document was last updated, in this case named `updated`
- The 'to address' of the email alert will either be chosen by the user via the form (default) or can be set by using the `toAddress` attribute on the saved query alert form tag.
- Appkit includes full user impersonation, so any security trimming is applied automatically.

Set Up Windows Authentication

How-to

> Compatible versions: 4.2, 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *App Studio*

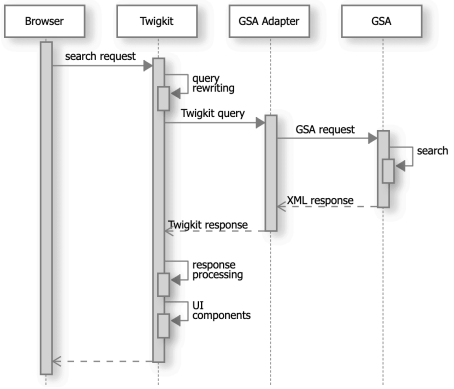
Appkit can be used with Integrated Windows Authentication for single sign on (SSO) in your search application using Microsoft Internet Information Services.

This article includes all the necessary steps to configure the IIS component on the Windows server. These instructions assume that your Appkit application is managed as a Maven project with all project dependencies described in a Maven POM file.

Note

All screen shots and references to IIS configuration below were obtained from IIS 8.0 on a Windows Server 2012 VM. These might differ if using a different version of Windows or IIS.

Interaction diagram



1 Prerequisites

1.1 Make sure IIS is installed on your Windows server

Install IIS on your Windows server.

1.2 Install the Oracle JDK on the Windows server

Install the Oracle Java SE Development Kit.

1.3 Install Tomcat

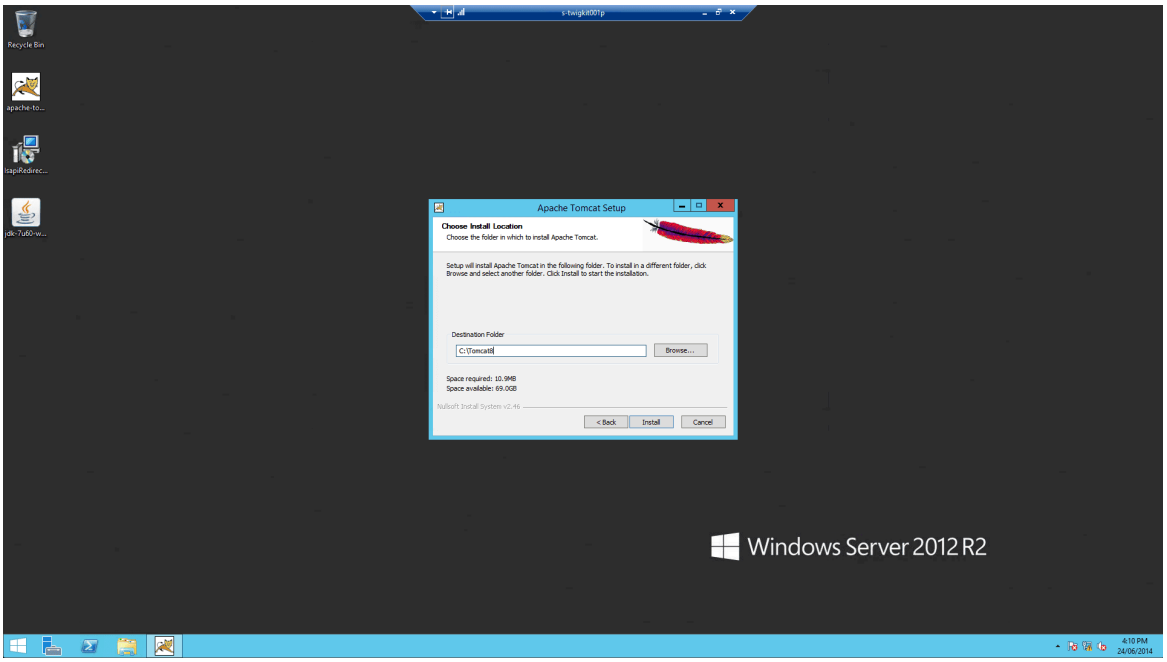
Tomcat versions 7 and 8 are both supported. In this, we assume you are using Tomcat 8.0.21.

To install Tomcat

1. Download the Tomcat installer (Windows Service Installer `.exe`) and follow the installation instructions. After installation is complete, do not start the Tomcat service just yet.

   Note

   Install Tomcat at a **whitespace-free path** (for example, `D:\tomcat8`). This path will be referred to as `CATALINA_HOME` hereafter. All the screenshots that follow assume you installed Tomcat in `D:\Tomcat8`.

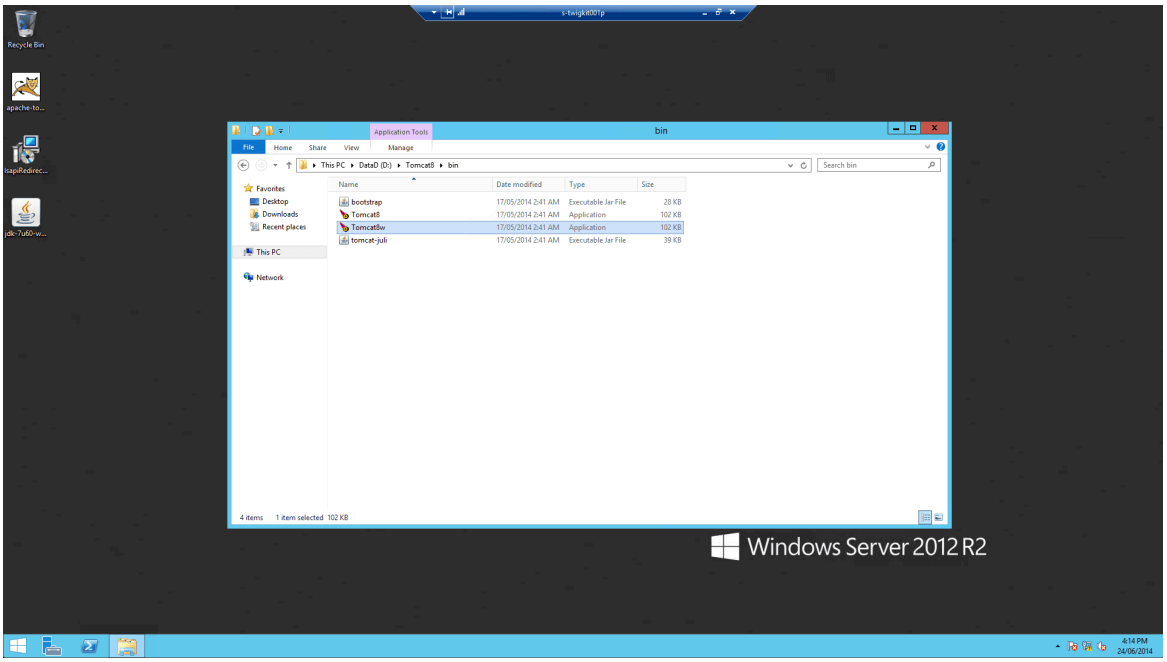


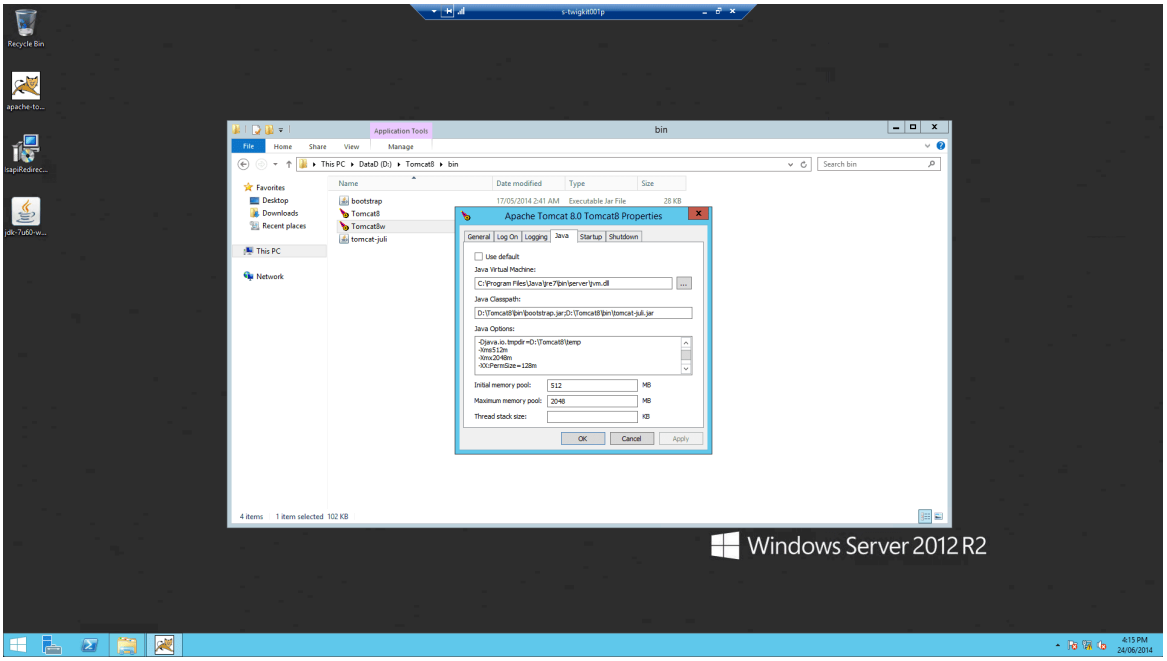
2. After the installation is complete, change Tomcat memory settings as follows:
   a. Run `%CATALINA_HOME%\bin\Tomcat8w.exe`
   b. On the Java tab, set Java Options to include:

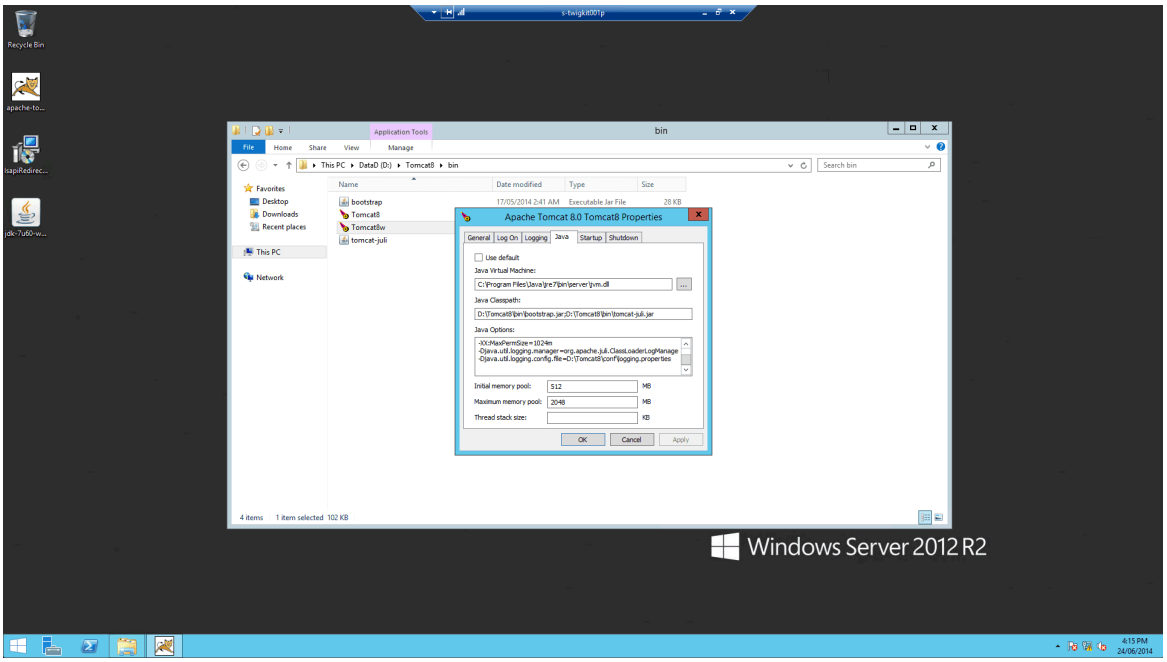
```
-Xms512m -Xmx2048m -XX:PermSize=128m -
XX:MaxPermSize=1024m
```

3. Repeat these settings in the input boxes below:

```
Initial memory pool: 512 Maximum memory pool: 2048
```


Recycle Bin
apache-to...
isapiRedirec...
jdk-7u60-w...
bin
Application Tools
File
Home
Share
View
Manage
This PC
DataD (D:)
Tomcat8
bin
Search bin
Favorites
Desktop
Downloads
Recent places
This PC
Network
Name
Date modified
Type
Size
bootstrap
17/05/2014 2:41 AM
Executable Jar File
28 KB
Tomcat8
17/05/2014 2:41 AM
Application
102 KB
Tomcat8w
17/05/2014 2:41 AM
Application
102 KB
tomcat-juli
17/05/2014 2:41 AM
Executable Jar File
39 KB
4 items
1 item selected 102 KB
Windows Server 2012 R2
4:14 PM
24/06/2014


Recycle Bin
apache-to...
isapiRedirec...
jdk-7u60-w...
bin
Application Tools
File
Home
Share
View
Manage
This PC
DataD (D:)
Tomcat8
bin
Search bin
Favorites
Desktop
Downloads
Recent places
This PC
Network
Name
Date modified
Type
Size
bootstrap
17/05/2014 2:41 AM
Executable Jar File
28 KB
Tomcat8
Tomcat8w
tomcat-juli
Apache Tomcat 8.0 Tomcat8 Properties
General
Log On
Logging
Java
Startup
Shutdown
Use default
Java Virtual Machine:
C:\Program Files\Java\jre7\bin\server\jvm.dll
Java Classpath:
D:\Tomcat8\bin\bootstrap.jar;D:\Tomcat8\bin\tomcat-juli.jar
Java Options:
-Djava.io.tmpdir=D:\Tomcat8\temp
-Xms512m
-Xmx2048m
-XX:PermSize=128m
Initial memory pool:
512
MB
Maximum memory pool:
2048
MB
Thread stack size:
KB
OK
Cancel
Apply
4 items
1 item selected 102 KB
Windows Server 2012 R2
4:15 PM
24/06/2014



1.4 Verify the Tomcat installation

Start up Tomcat via Windows Services and go to `http://localhost:8080`. You should be presented with the default Tomcat manager application.

2 Configure and deploy the Appkit application

2.1 Configure the security provider

Authentication against IIS requires the "generic" Appkit security provider module. This is a thin layer which integrates with the standard Java servlet API to pick up the user's credentials for authentication. To enable this module first remove any existing security provider dependency from the `pom.xml` under the root of the project and add this within the `dependencies` tag:

```
<dependency> <groupId>twigkit</groupId>
<artifactId>twigkit.security.core</artifactId>
<version>${project.parent.version}</version> </dependency>
```

Then to configure Appkit to invoke this module on startup change the `security.conf` file in `src/main/resources/conf/security/` to contain:

```
type: generic
```

Make sure to remove any existing security provider configuration if there is one including the 'Spring Security' XML file and Spring related entries in the `web.xml` file if Spring was previously being used in the application.

2.2 Deploy the application to Tomcat

After these steps are completed, package your Appkit web application as a WAR file and deploy into Tomcat. Hereafter we assume you package the application as `myapp.war` so that the application can be accessed as `http://localhost:8080/myapp/`.

3 Configure the Tomcat-ISAPI connector

The Tomcat ISAPI connector is a DLL file that you can download from the Apache Tomcat website.

Below, we assume that you have downloaded the ZIP file relevant to your OS type and extracted `isapi_redirect.dll` from the archive.

3.1 Configure Tomcat for ISAPI

1. Add isapi_redirect.dll to `%CATALINA_HOME%\bin`.
2. Create `%CATALINA_HOME%\conf\workers.properties.minimal` with:

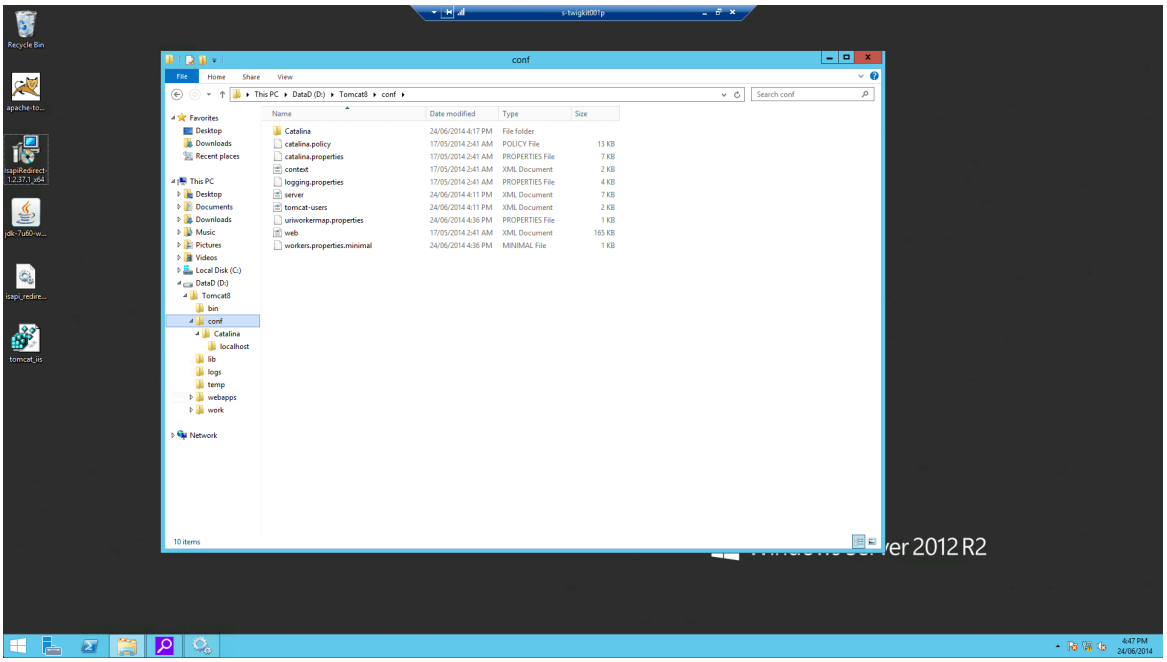
```
worker.list=worker1 worker.worker1.host=127.0.0.1
worker.worker1.port=8009 worker.worker1.type=ajp13
```

3. Create `%CATALINA_HOME%\conf\uriworkermap.properties` with:

```
/*=worker1
```

4. (**Important**) In `%CATALINA_HOME%\conf\server.xml`, you must put `tomcatAuthentication="false"` on the AJP connector:

```
<Ajp13Connector port="8009" tomcatAuthentication="false" />
```



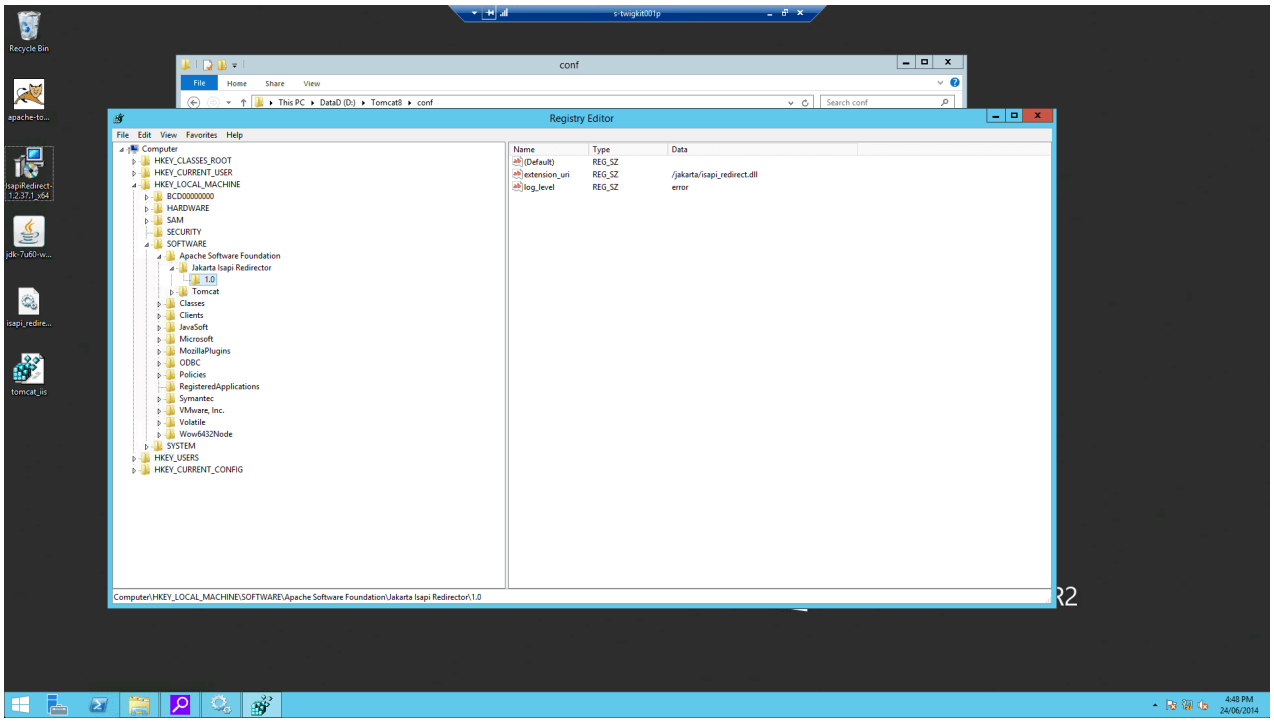
Note

From now on, we assume you installed Tomcat in `D:\Tomcat8` as noted above.

3.2 Configure Windows registry settings

This key needs to be modified in the registry with the specified properties:

```
HKEY_LOCAL_MACHINE\SOFTWARE\Apache Software Foundation\Jakarta Isapi Redirector\1.0
```

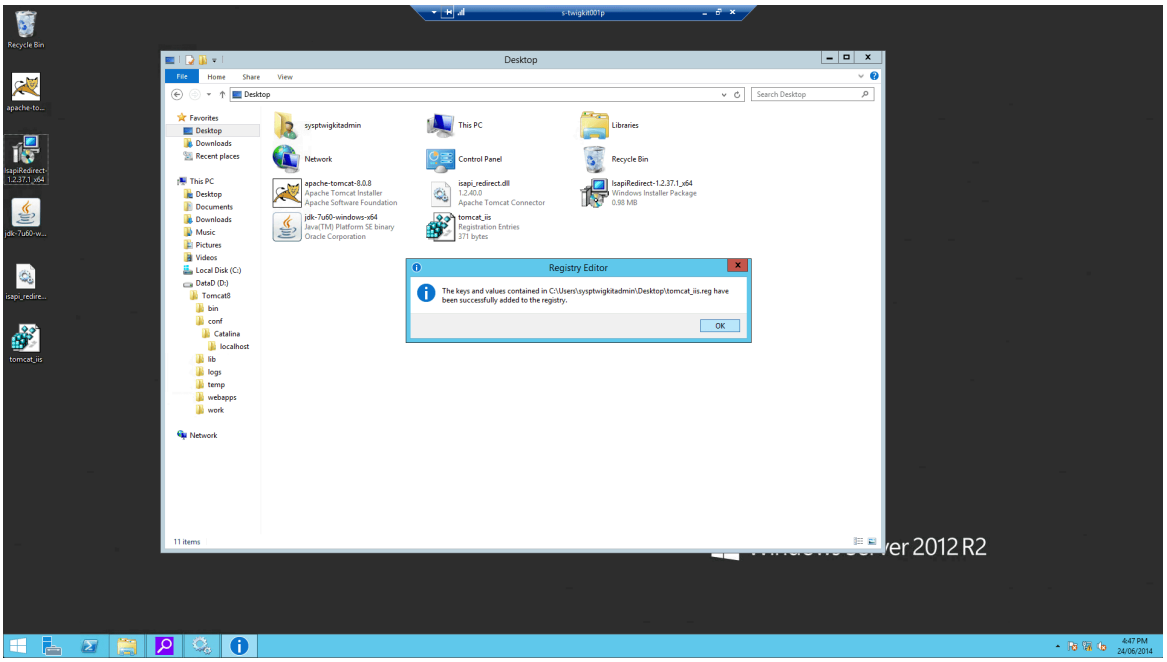


1. Add a string value with the name `extension_uri` and a value of `/jakarta/isapi_redirect.dll`
2. Add a string value with the name `log_file` and a value pointing to where you want your log file to be (for example, `D:\Tomcat8\logs\isapi.log`).
3. Add a string value with the name `log_level` and a value for your log level (can be `debug`, `info`, `error` or `emerg`).
4. Add a string value with the name `worker_file` and a value which is the full path to your `workers.properties` file (for example, `D:\Tomcat8\conf\workers.properties.minimal`)
5. Add a string value with the name `worker_mount_file` and a value which is the full path to your `uriworkermap.properties` file (for example, `D:\Tomcat8\conf\uriworkermap.properties`)

To save trouble, you can create a file named `tomcat_iis_settings.reg` that includes this (replacing `D:\Tomcat8` with your own `CATALINA_HOME`):

```
Windows Registry Editor Version 5.00
[HKEY_LOCAL_MACHINE\SOFTWARE\Apache Software Foundation\Jakarta
Isapi Redirector\1.0] @=""
"extension_uri"="/jakarta/isapi_redirect.dll"
"log_file"="D:\\Tomcat8\\logs\\isapi_redirect.log"
"log_level"="error"
"worker_file"="D:\\Tomcat8\\conf\\workers.properties.minimal"
"worker_mount_file"="D:\\Tomcat8\\conf\\uriworkermap.properties
"
```

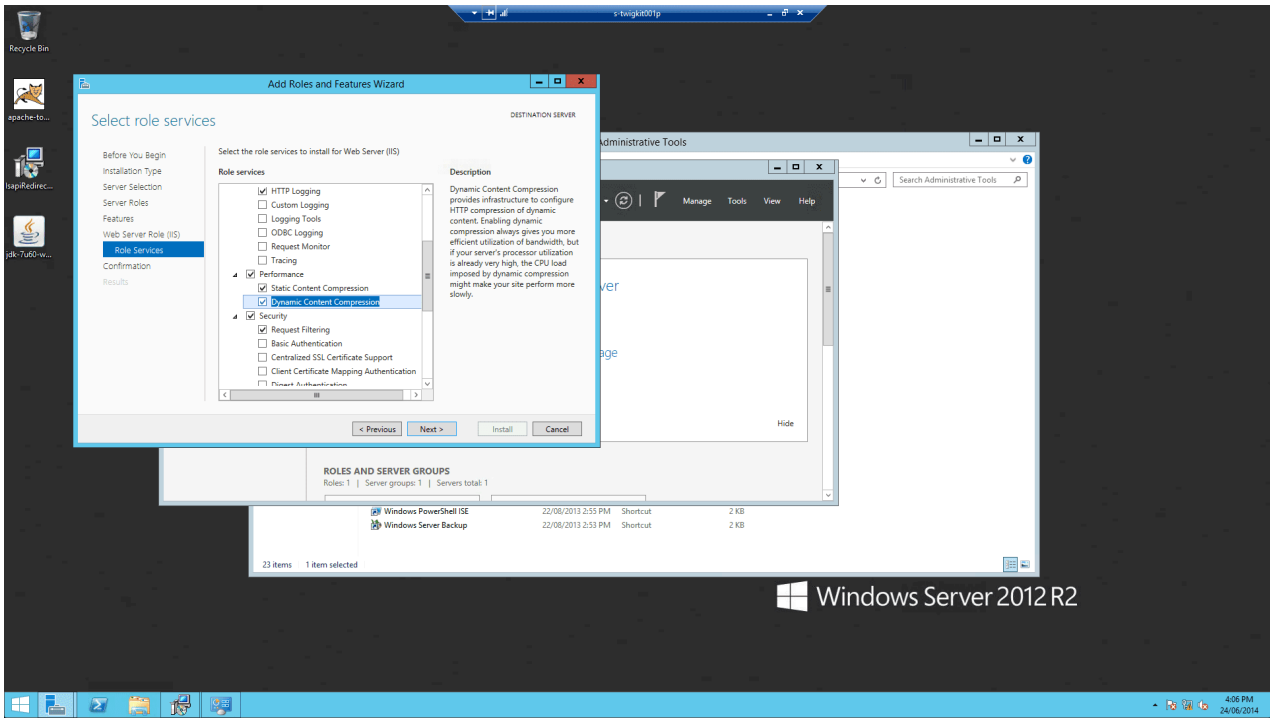
6. On the Windows server, double-click the `tomcat_iis_settings.reg` file to import the settings.



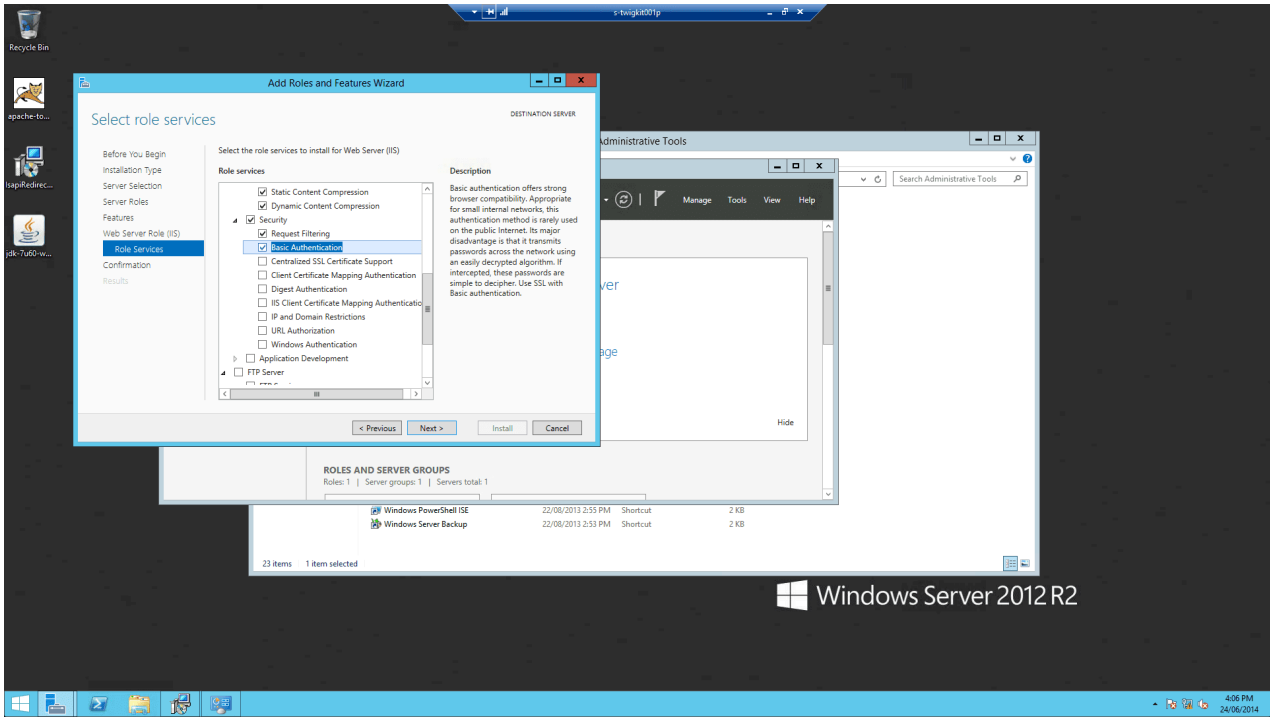
4 Configure the IIS

This assumes that IIS is already installed on the Windows server.

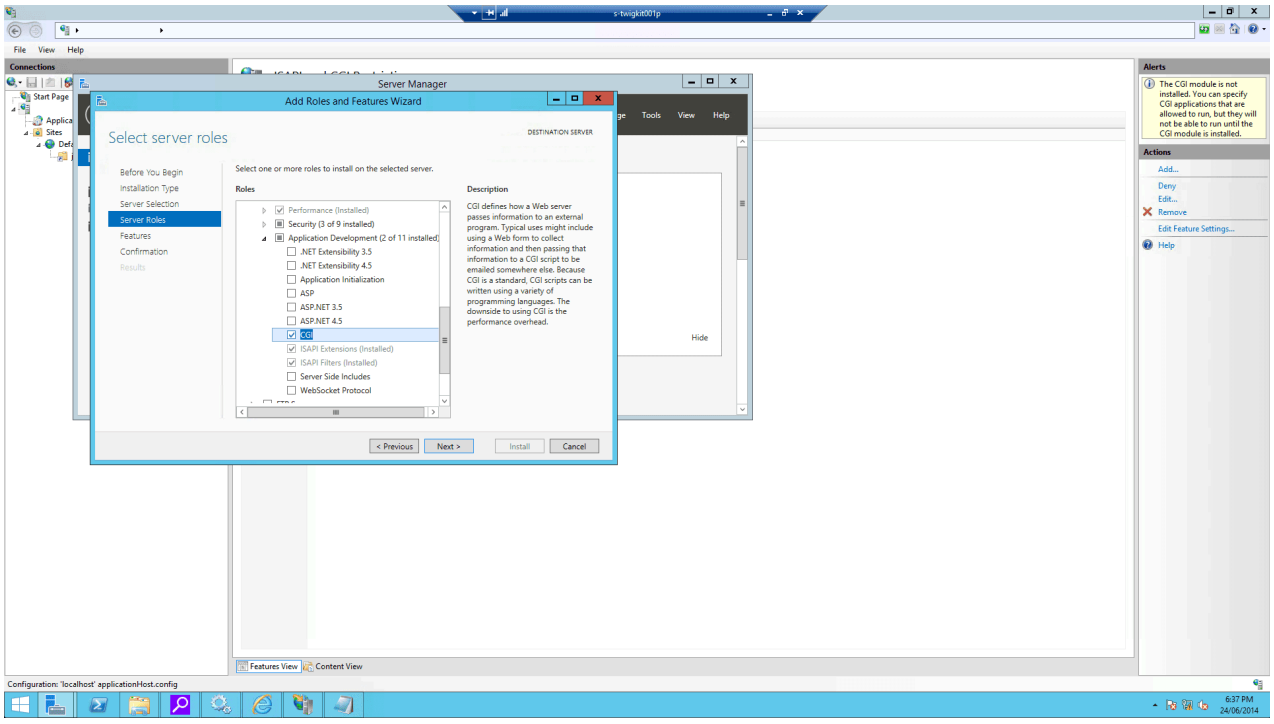
4.1 Enable ISAPI Filters and ISAPI Extensions



4.2 Enable Windows authentication

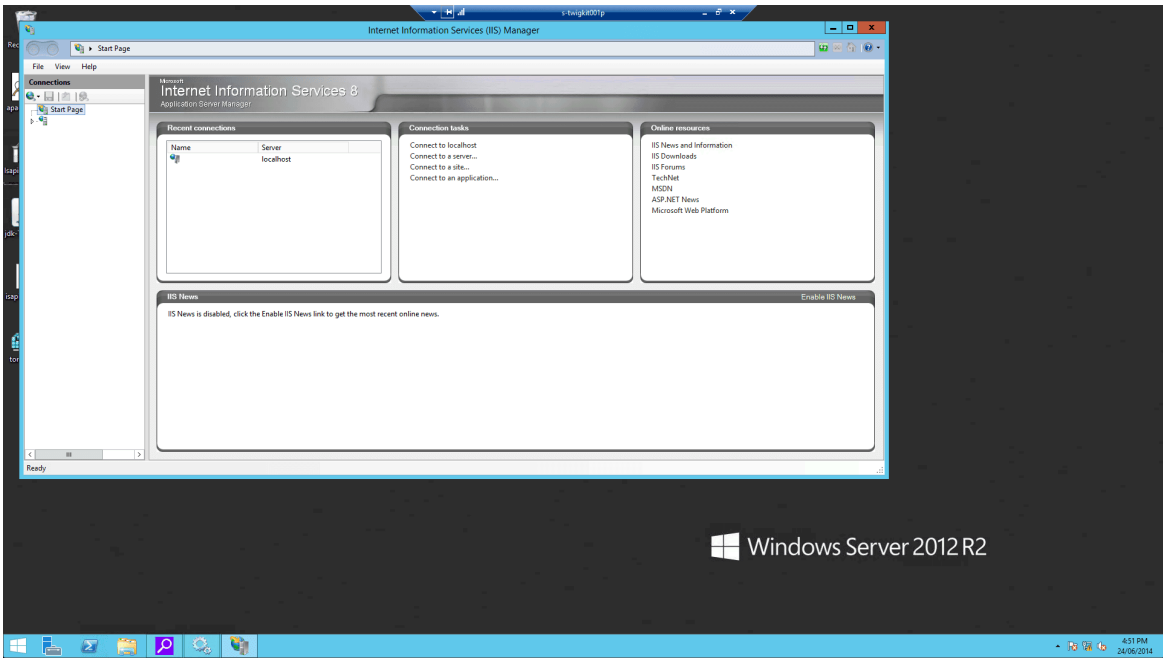


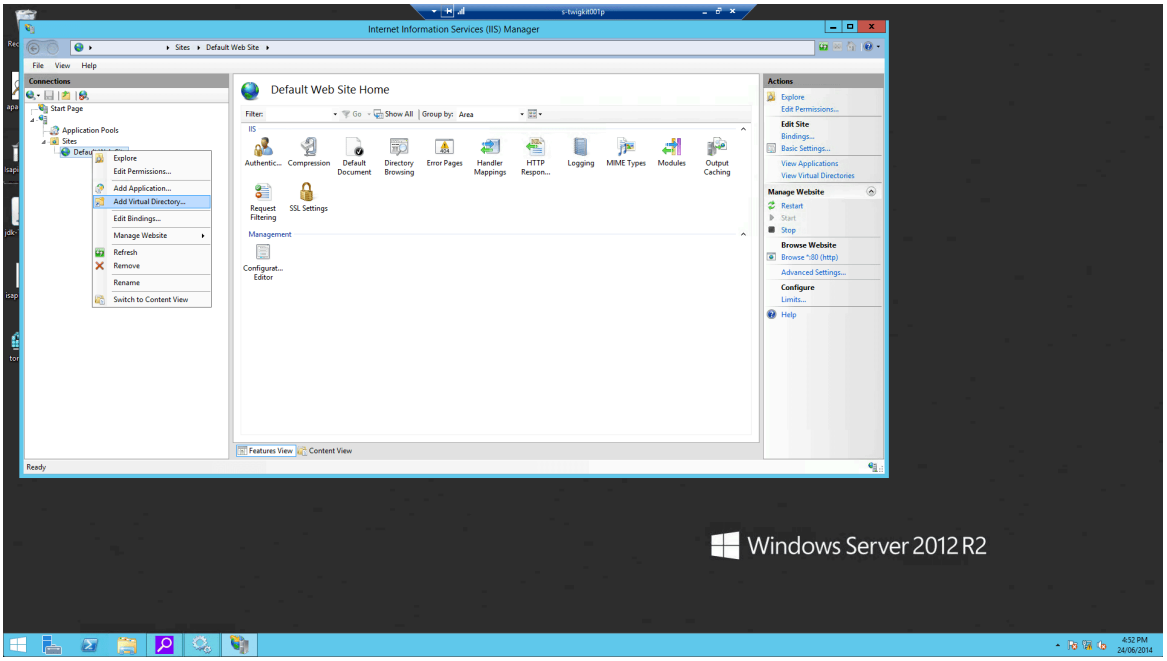
4.3 Enable CGI filtering

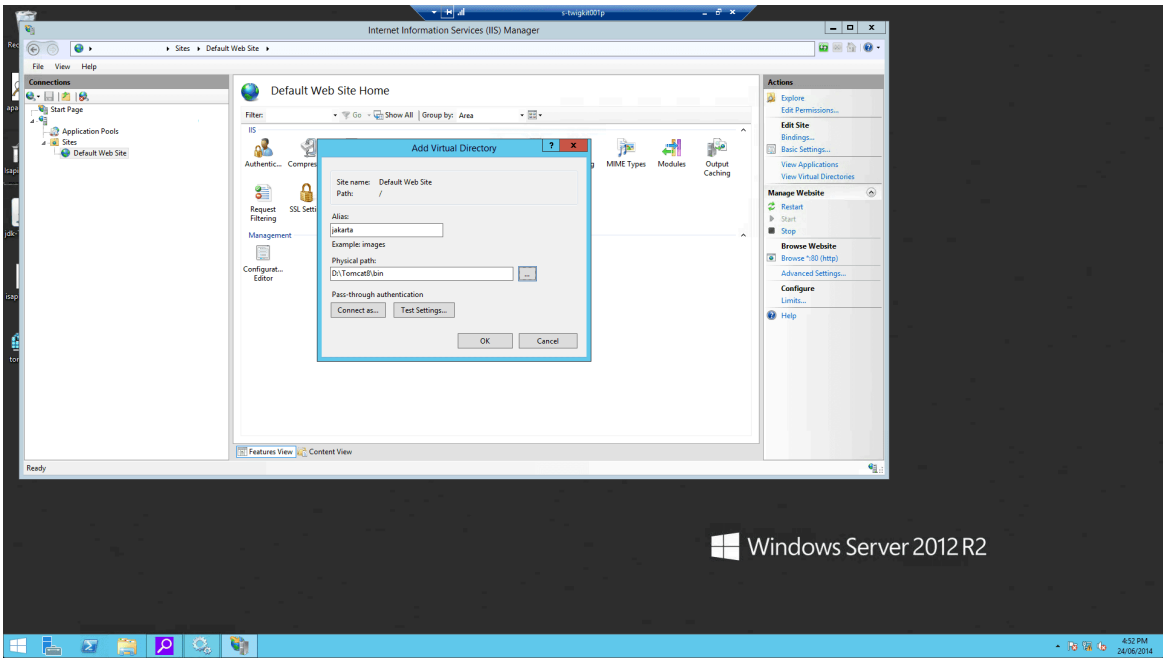


4.4 Configure the virtual directory and website on IIS

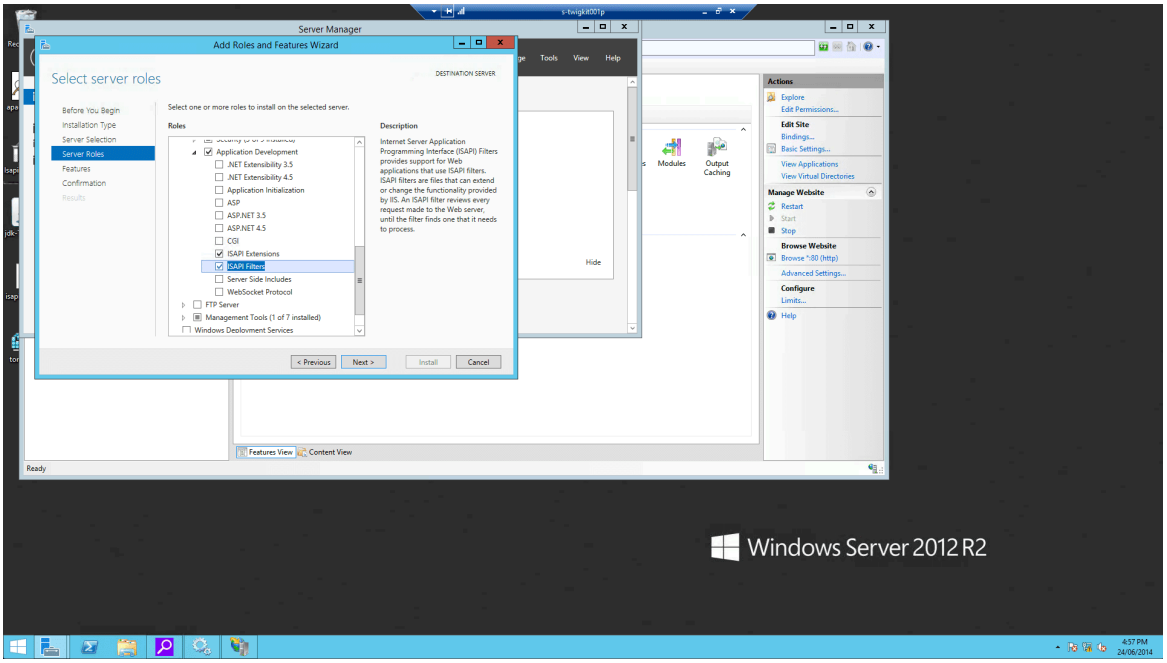
1. Using the IIS management console, add a new virtual directory to your IIS/PWS web site. The name of the virtual directory must be `jakarta`. Its physical path should be the directory where you placed `isapi_redirect.dll` (in our example, it is `D:\Tomcat8\bin`). While creating this new virtual directory assign it with execute access.

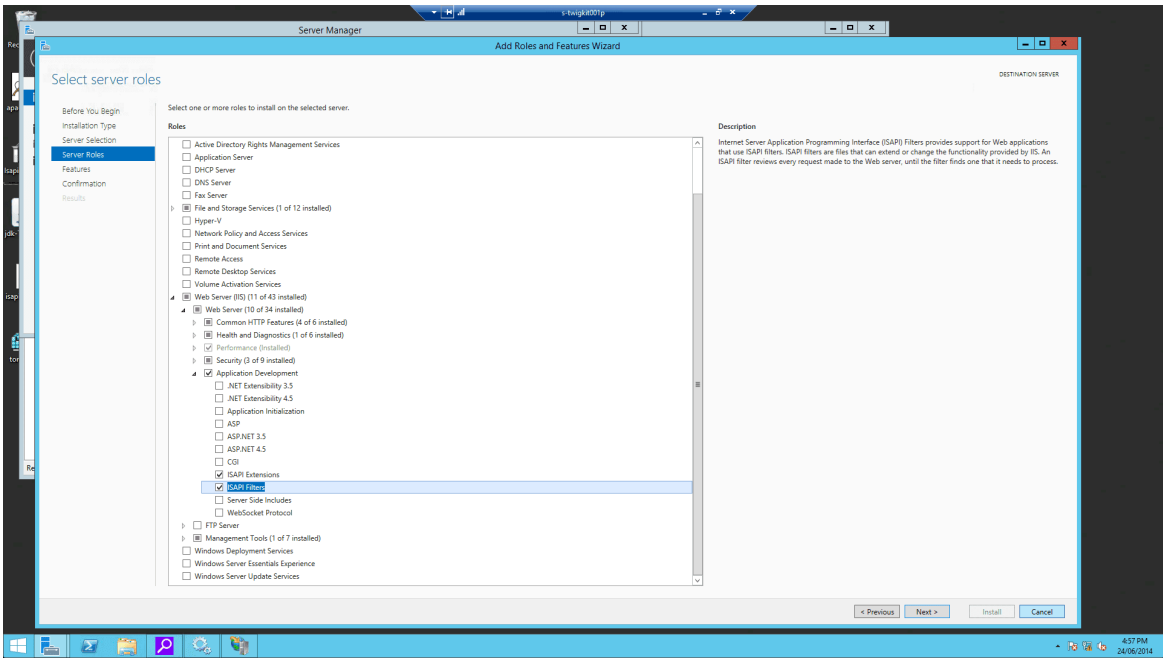

Internet Information Services (IIS) Manager
Start Page
File View Help
Connections
Start Page
Internet Information Services 8
Application Server Manager
Recent connections
Name
Server
localhost
Connection tasks
Connect to localhost
Connect to a server...
Connect to a site...
Connect to an application...
Online resources
IIS News and Information
IIS Downloads
IIS Forums
TechNet
MSDN
ASP.NET News
Microsoft Web Platform
IIS News
Enable IIS News
IIS News is disabled, click the Enable IIS News link to get the most recent online news.
Ready
Windows Server 2012 R2

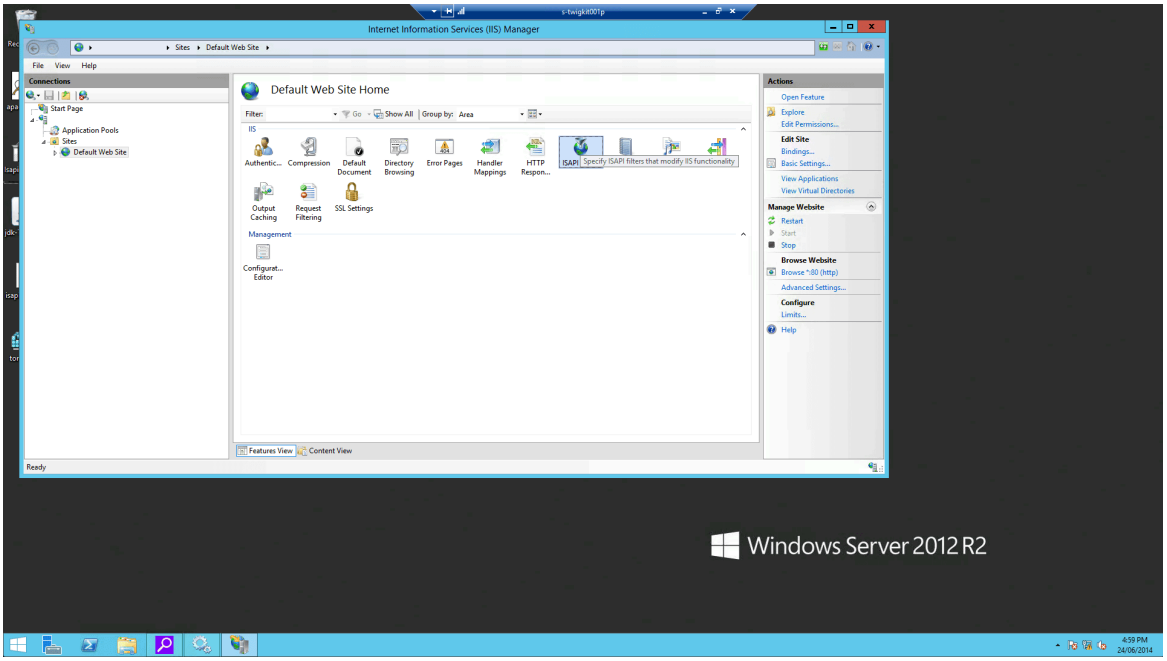

Internet Information Services (IIS) Manager
Sites ▸ Default Web Site ▸
File View Help
Connections
Start Page
Application Pools
Sites
Default Web Site
Explore
Edit Permissions...
Add Application...
Add Virtual Directory...
Edit Bindings...
Manage Website
Refresh
Remove
Rename
Switch to Content View
Default Web Site Home
Filter:
Show All
Group by: Area
IIS
Authentic...
Compression
Default Document
Directory Browsing
Error Pages
Handler Mappings
HTTP Respon...
Logging
MIME Types
Modules
Output Caching
Request Filtering
SSL Settings
Management
Configurat... Editor
Features View
Content View
Actions
Explore
Edit Permissions...
Edit Site
Bindings...
Basic Settings...
View Applications
View Virtual Directories
Manage Website
Restart
Start
Stop
Browse Website
Browse *:80 (http)
Advanced Settings...
Configure
Limits...
Help
Ready
Windows Server 2012 R2

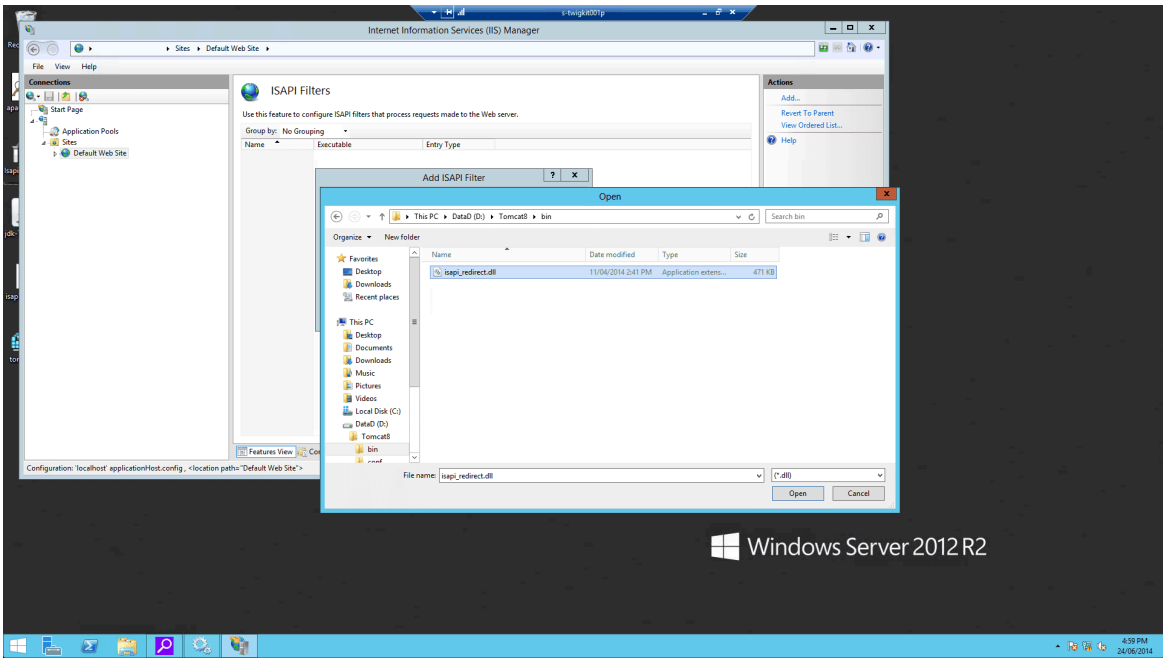


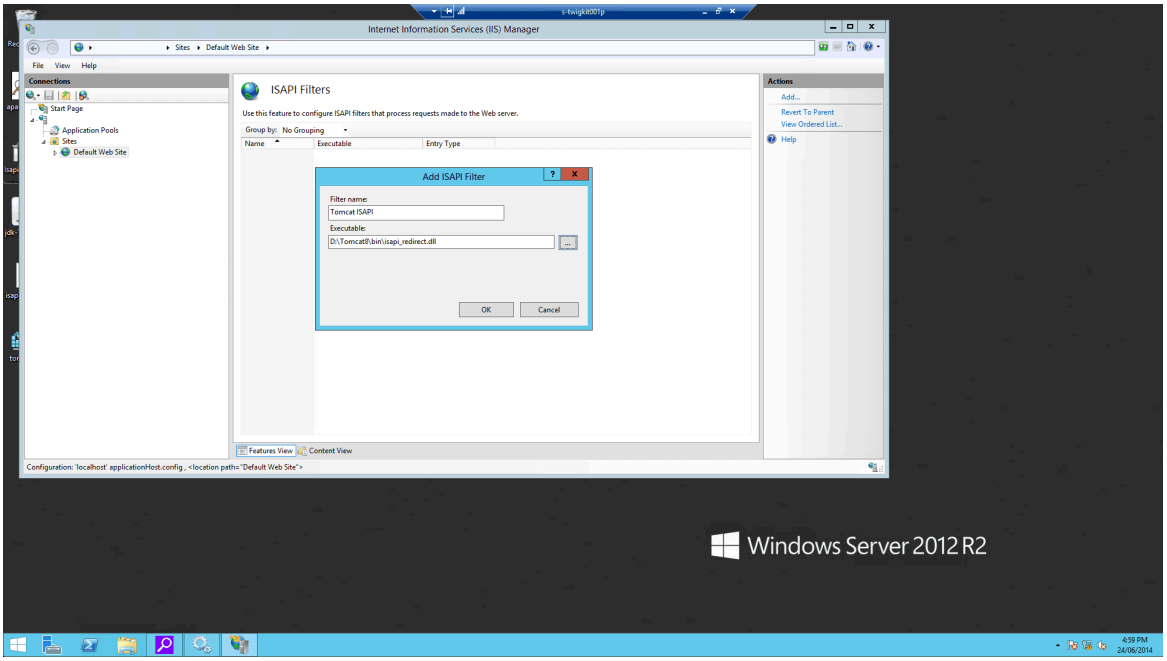
2. Using the IIS management console, add `isapi_redirect.dll` as a filter in your IIS/PWS web site. The name of the filter should reflect its task (here we use the name `jakarta`). Its executable must be `D:\Tomcat8\bin\isapi_redirect.dll`.



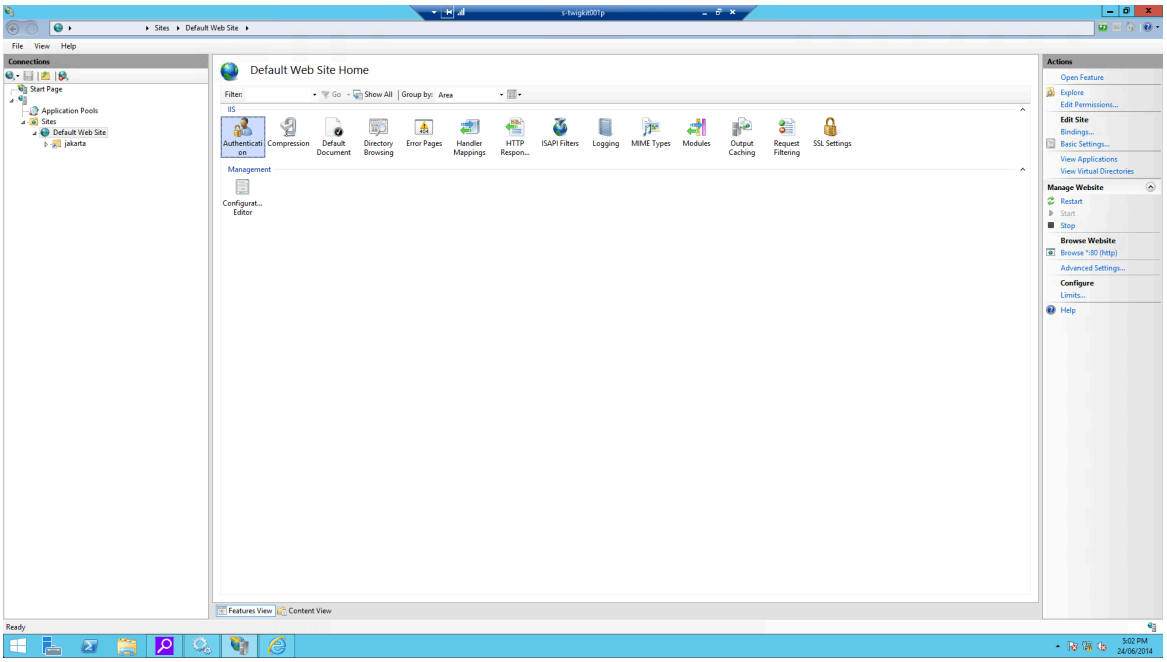

Server Manager
Add Roles and Features Wizard
Select server roles
DESTINATION SERVER
Before You Begin
Installation Type
Server Selection
Server Roles
Features
Confirmation
Results
Select one or more roles to install on the selected server.
Roles
Active Directory Rights Management Services
Application Server
DHCP Server
DNS Server
Fax Server
File and Storage Services (1 of 12 installed)
Hyper-V
Network Policy and Access Services
Print and Document Services
Remote Access
Remote Desktop Services
Volume Activation Services
Web Server (IIS) (11 of 43 installed)
Web Server (10 of 34 installed)
Common HTTP Features (4 of 6 installed)
Health and Diagnostics (1 of 6 installed)
Performance (Installed)
Security (3 of 9 installed)
Application Development
.NET Extensibility 3.5
.NET Extensibility 4.5
Application Initialization
ASP
ASP.NET 3.5
ASP.NET 4.5
CGI
ISAPI Extensions
ISAPI Filters
Server Side Includes
WebSocket Protocol
FTP Server
Management Tools (1 of 7 installed)
Windows Deployment Services
Windows Server Essentials Experience
Windows Server Update Services
Description
Internet Server Application Programming Interface (ISAPI) Filters provides support for Web applications that use ISAPI filters. ISAPI filters are files that can extend or change the functionality provided by IIS. An ISAPI filter reviews every request made to the Web server, until the filter finds one that it needs to process.
< Previous
Next >
Install
Cancel

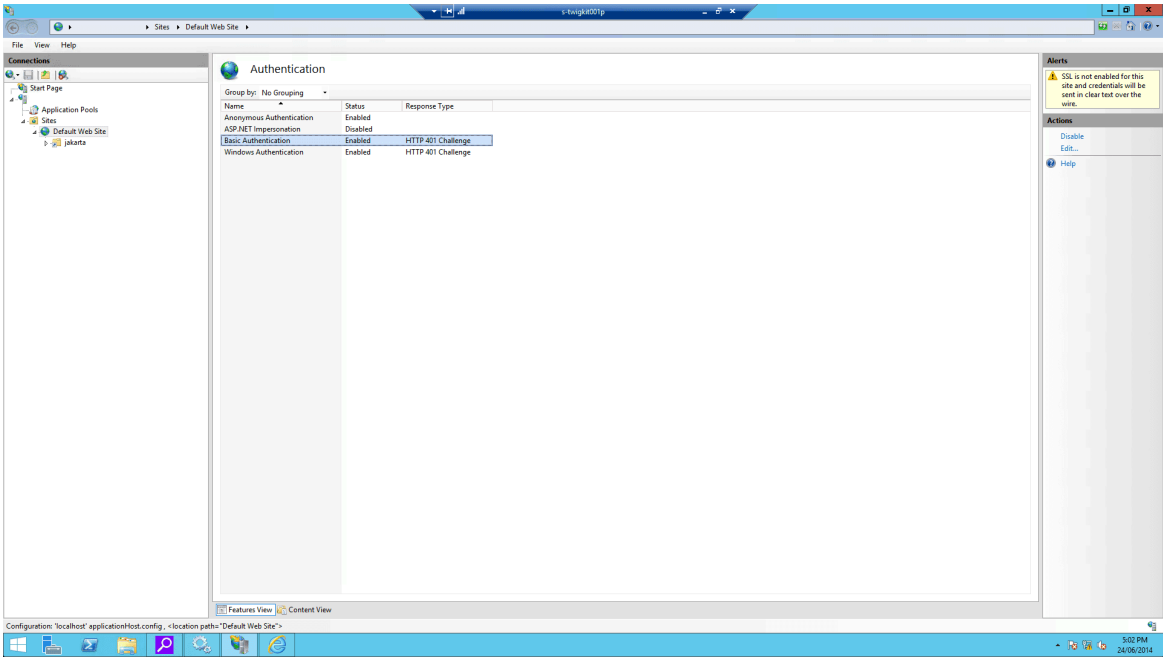

Internet Information Services (IIS) Manager
Default Web Site Home
Windows Server 2012 R2

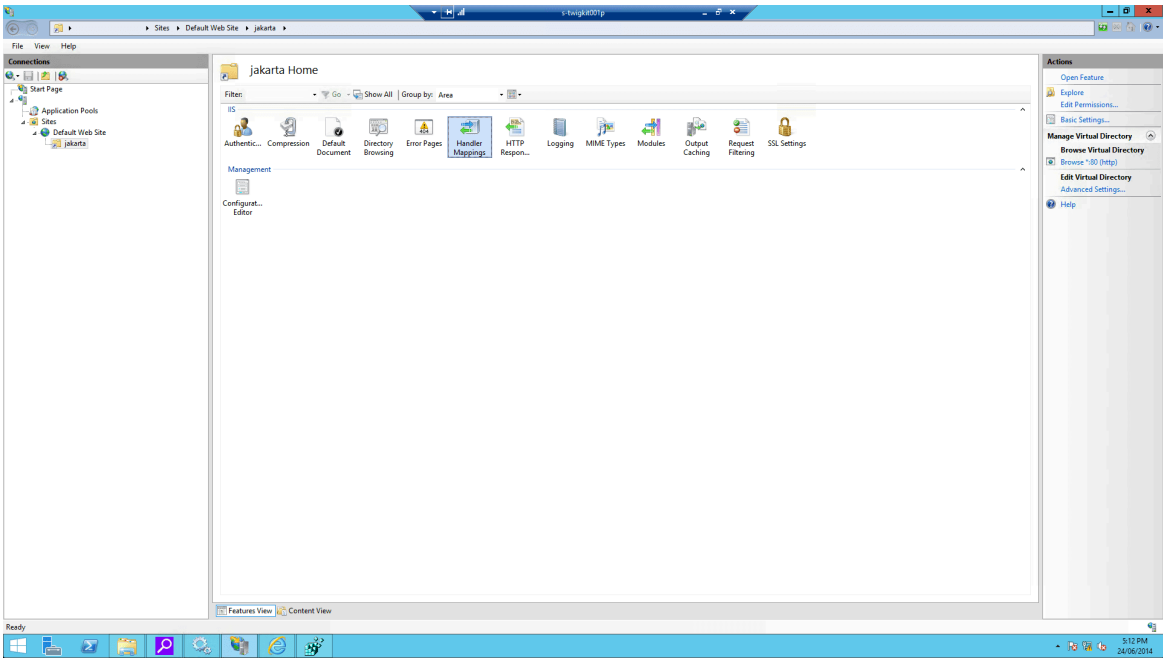


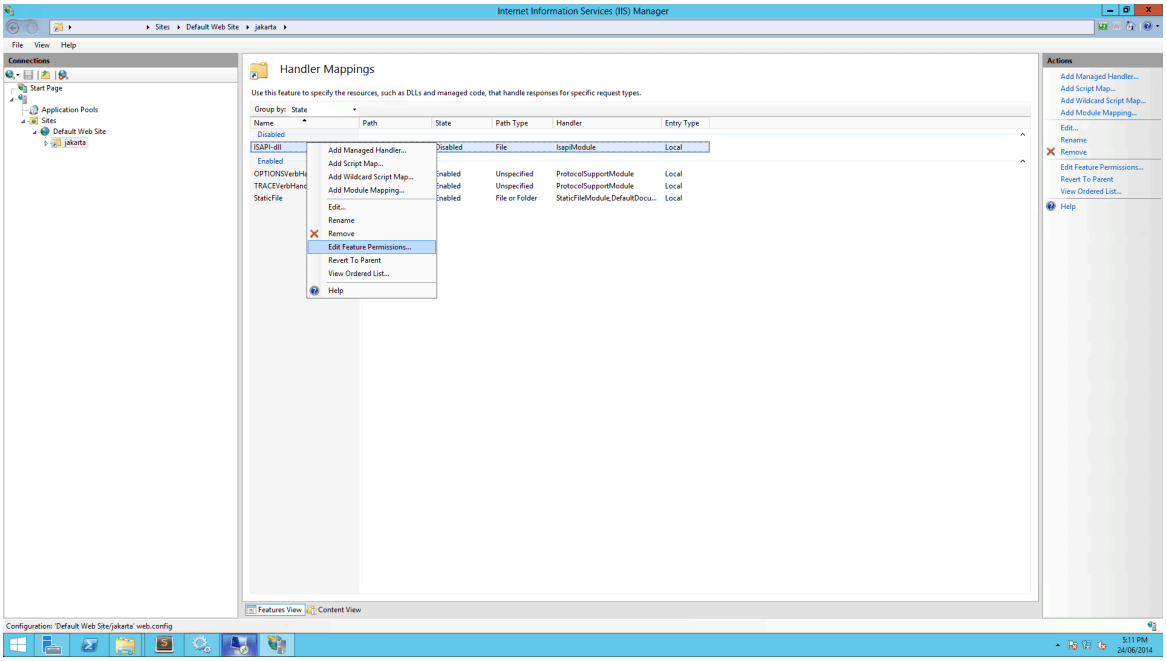


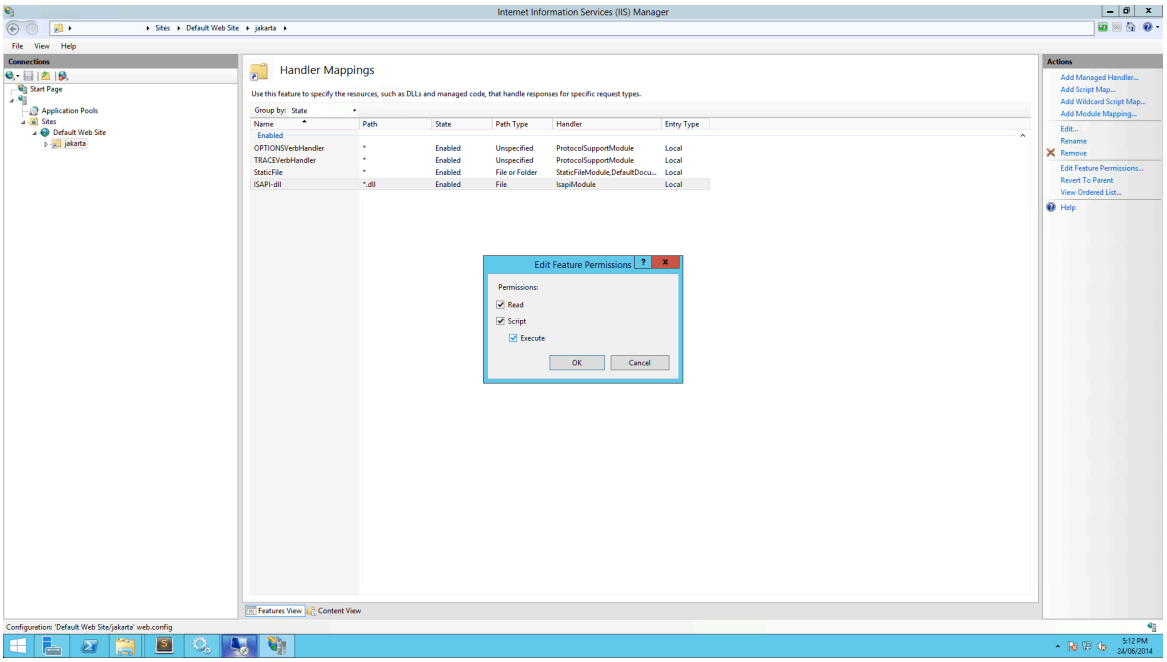
3. Enable Windows authentication for your IIS website:


Default Web Site Home
Authentication
Compression
Default Document
Directory Browsing
Error Pages
Handler Mappings
ISAPI Filters
Logging
MIME Types
Modules
Output Caching
Request Filtering
SSL Settings
Configurat... Editor
Features View
Content View
Ready

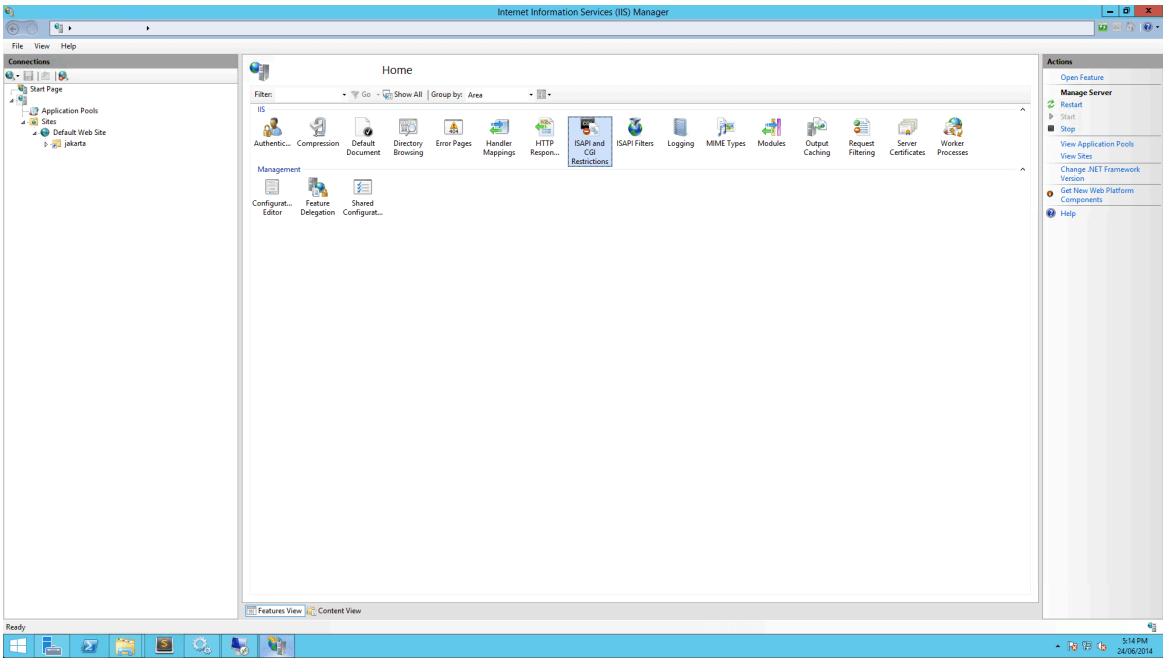

Authentication
Anonymous Authentication
ASP.NET Impersonation
Basic Authentication
Windows Authentication
Enabled
Disabled
HTTP 401 Challenge
Features View
Content View

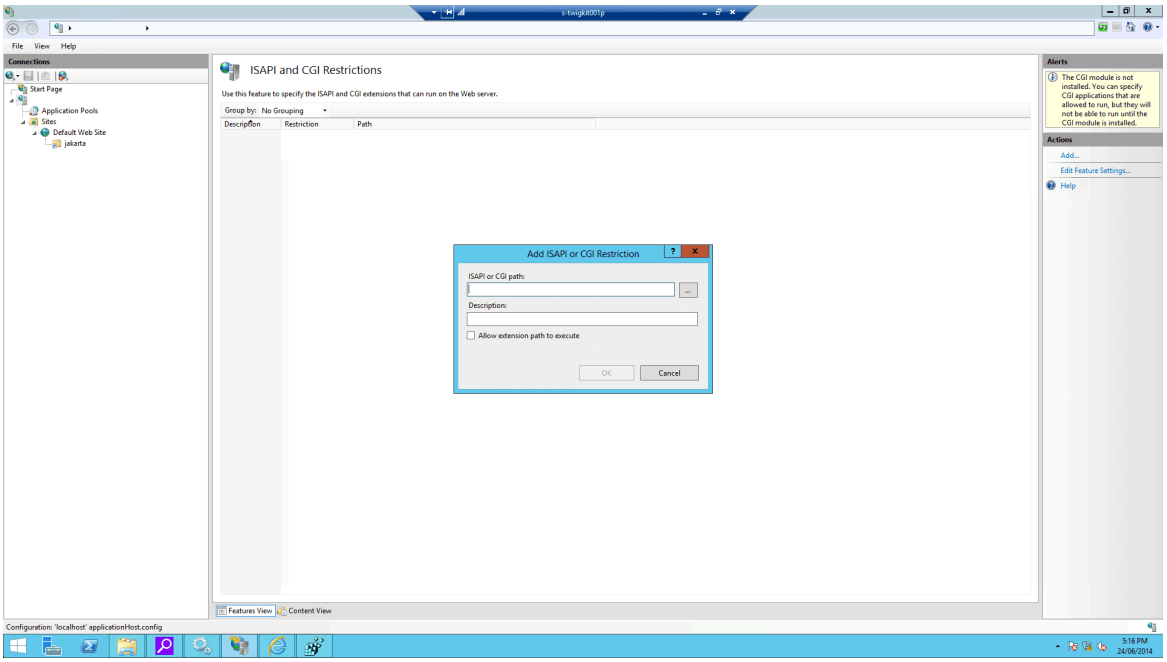

jakarta Home
Handler Mappings

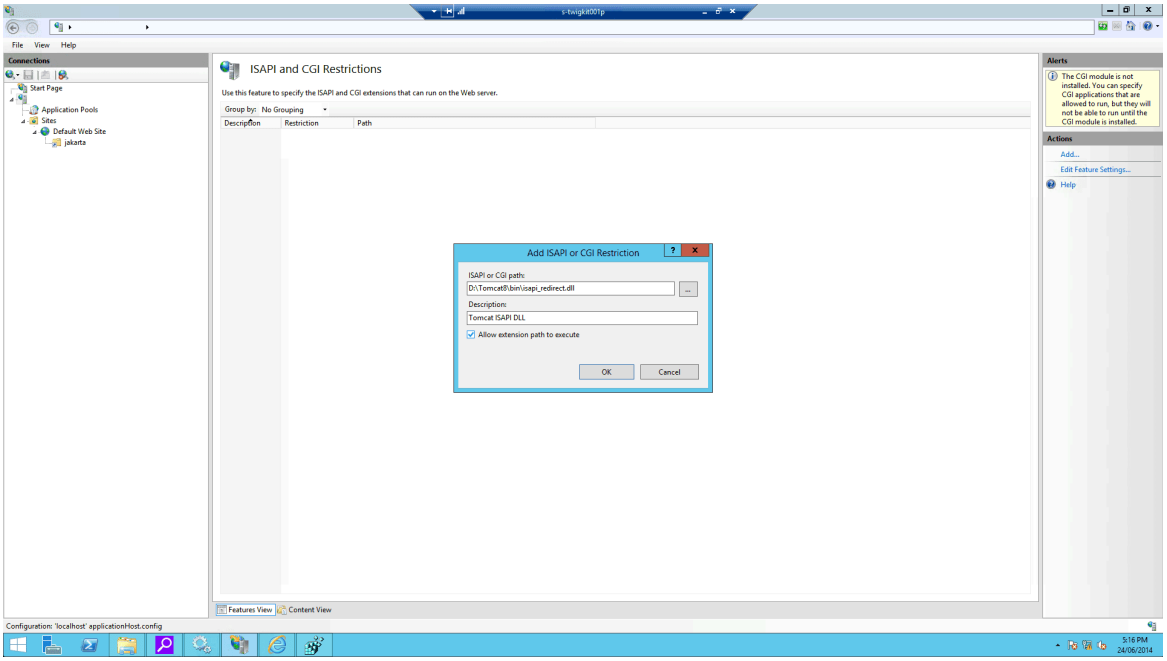

Internet Information Services (IIS) Manager
Handler Mappings

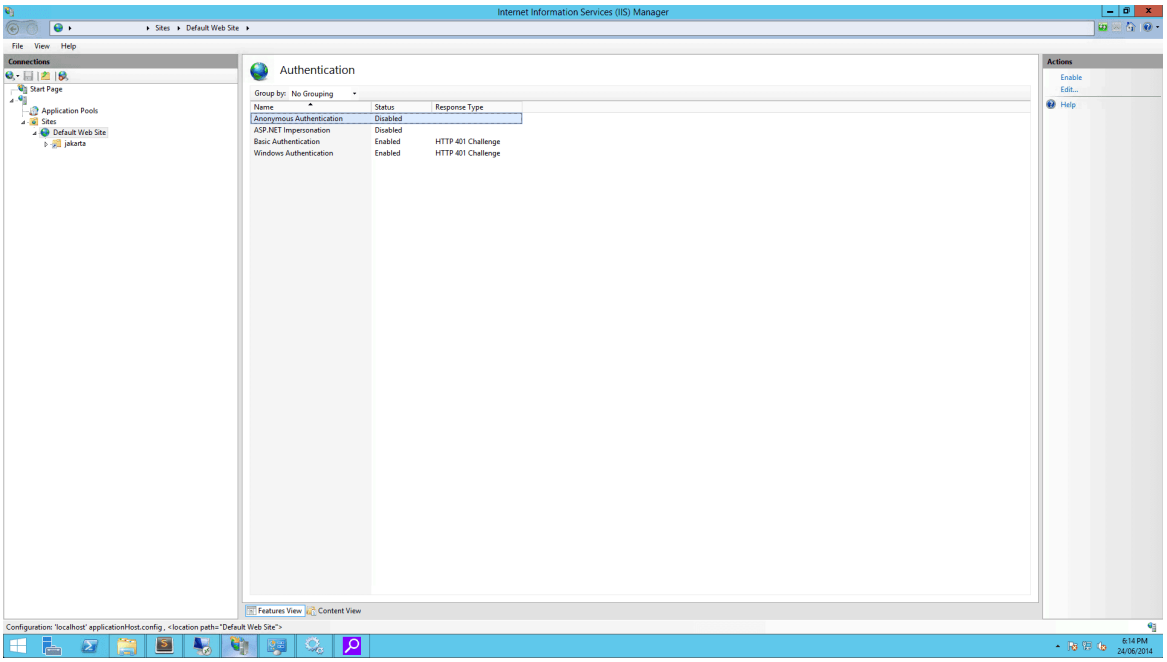


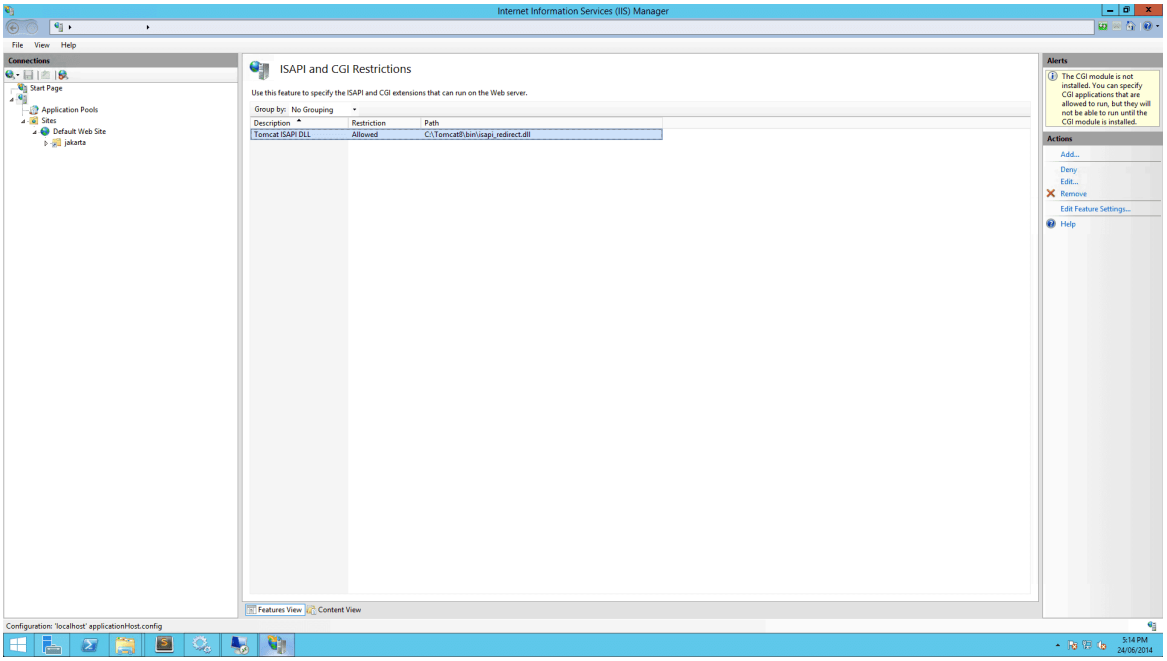
4. On IIS Admin, select the server, and click ISAPI and CGI Restrictions. Then open it and Add... with the correct `isapi_redirect.dll` path:




ISAPI and CGI Restrictions
Use this feature to specify the ISAPI and CGI extensions that can run on the Web server.
Add ISAPI or CGI Restriction
ISAPI or CGI path:
Description:
Allow extension path to execute
OK
Cancel


ISAPI and CGI Restrictions
Use this feature to specify the ISAPI and CGI extensions that can run on the Web server.
Add ISAPI or CGI Restriction
ISAPI or CGI path:
D:\Tomcat9\bin\isapi_redirect.dll
Description:
Tomcat ISAPI DLL
Allow extension path to execute
OK
Cancel





5. Restart IIS and Tomcat.

5 Verify the installation

If all goes well, you should be able to visit `http://localhost/myapp/`, which should present the authenticated Windows user to the Appkit application.

Note

At this point, Tomcat would still be accessible on port 8080, so if you visit `http://localhost:8080/myapp/`, you should see the same interface, except that your user is not authenticated.

6 Troubleshooting

Users receive the error 'the request entity is too large' in the browser

This is usually due to an issue with the Tomcat-ISAPI communication and the settings of the AJP connector in Tomcat must be modified to increase the maximum packet size:

1. In the `workers.properties` file referenced by the tomcat-connection definition, set the packet size to the maximum:

```
worker.<worker name>.max_packet_size=65536
```

2. Set `packetSize` in the AJP Connector definition:

```
<Connector port="8009" protocol="AJP/1.3" redirectPort="8443"
packetSize="65536" tomcatAuthentication="false" />
```

Related articles

- *Set Up Active Directory Authentication*
- *Set Up Lucidworks Fusion Authentication*
- *Set Up HTTP Headers Authentication*
- *Set Up Kerberos Authentication*
- *Set Up LDAP Authentication*
- *Set Up SiteMinder Authentication*
- *Set Up Spring Authentication*
- *Set Up Tomcat Container Authentication*

Setting Up a Mock Platform Response

How-to

> Compatible versions: 4.2, 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *App Studio*

In some cases it can be difficult to access a search platform. In these situations, it can be useful to work against a static response in the form of a file containing a typical search response (usually as XML).

This is not a recommended method for developing a fully featured search application from start to finish, because it is impossible to emulate the dynamic nature of a live platform. However this is useful in a number of situations including:

- Where networking or security considerations prevent access to the platform
- Debugging issues in an application where it is necessary to recreate a specific type of response
- Initial development phases where access to a live platform with realistic data is unavailable
- Deterministic test conditions (this could be for unit tests as well as any other form of manual or automated testing)

How to set up a mock platform

A few simple steps are required to run an application against a 'mock' response:

1. Create a mock response file by saving the XML from a query issued directly against the platform, or request one from someone who can access a platform instance. This should represent the type of response you want to mock. See this example of a Solr XML response:

<?xml version="1.0" encoding="UTF-8"?> <response> <lst name="responseHeader"> <int name="status">0</int> <int name="QTime">24</int> <lst name="params"> <str name="spellcheck">true</str> <str name="facet">true</str> <str name="indent">true</str> <str name="facet.limit">10</str> <str name="wt">xmk</str> <str name="rows">5</str> <str name="fl">*,score</str> <str name="start">0</str> <arr name="q"> <str>*:*</str> </arr> <str name="q.op">OR</str> <str name="facet.field">month</str> <str name="qt">/prodover</str> <str name="fq">category:Clothing</str> </lst> </lst> <result name="response" numFound="21111" start="0" maxScore="1.0"> <doc> <str name="category">Clothing</str> <str name="p_image_path_1_s">/s/c/sc1402sep26pictga10015.jpg</str> <int name="p_id_i">356980</int> <date name="p_date">2014-10-07T00:00:00Z</date> <str name="p_name">Generic T-Shirt</str> <float name="p_price">552.63</float> <arr name="colors"> <str>Black</str> <str>White</str> <str>Green</str> </arr> <long name="_version_">1486360119259168768</long> <float name="score">1.0</float> </doc> <doc> <str name="category">Clothing</str> <str name="p_image_path_1_s">/s/c/sc1402sep26pictga10015.jpg</str> <int name="p_id_i">356980</int> <date name="p_date">2014-10-07T00:00:00Z</date> <str name="p_name">Generic T-Shirt</str> <float name="p_price">552.63</float> <arr name="colors"> <str>Black</str> <str>White</str> <str>Green</str> </arr> <long name="_version_">1486360119259168768</long> <float name="score">1.0</float> </doc> <doc> <str name="category">Clothing</str> <str name="p_image_path_1_s">/s/c/sc1402sep26pictga10015.jpg</str> <int name="p_id_i">356980</int> <date name="p_date">2014-10-07T00:00:00Z</date> <str name="p_name">Generic T-Shirt</str> <float name="p_price">552.63</float> <arr name="colors"> <str>Black</str> <str>White</str> <str>Green</str> </arr> <long

```
name="_version_">1486360119259168768</long> <float
name="score">1.0</float> </doc> <doc> <str
name="category">Clothing</str> <str
name="p_image_path_1_s">/s/c/sc1402sep26pictga10015.jpg</str>
<int name="p_id_i">356980</int> <date name="p_date">2014-10-
07T00:00:00Z</date> <str name="p_name">Generic T-Shirt</str>
<float name="p_price">552.63</float> <arr name="colors">
<str>Black</str> <str>White</str> <str>Green</str> </arr> <long
name="_version_">1486360119259168768</long> <float
name="score">1.0</float> </doc> <doc> <str
name="category">Clothing</str> <str
name="p_image_path_1_s">/s/c/sc1402sep26pictga10015.jpg</str>
<int name="p_id_i">356980</int> <date name="p_date">2014-10-
07T00:00:00Z</date> <str name="p_name">Generic T-Shirt</str>
<float name="p_price">552.63</float> <arr name="colors">
<str>Black</str> <str>White</str> <str>Green</str> </arr> <long
name="_version_">1486360119259168768</long> <float
name="score">1.0</float> </doc> </result> <lst
name="facet_counts"> <lst name="facet_queries"/> <lst
name="facet_fields"> <lst name="month"> <int
name="November">9140</int> <int name="October">7936</int> <int
name="September">7215</int> <int name="August">7088</int> <int
name="July">6878</int> <int name="June">5238</int> <int
name="May">4351</int> <int name="April">4207</int> <int
name="March">3593</int> <int name="February">3002</int> </lst>
</lst> <lst name="facet_dates"/> <lst name="facet_ranges"/> <lst
name="facet_intervals"/> </lst> </response>
```

2. Copy the XML file into your project to:

```
src/main/webapp/WEB-INF/pages/mock-response.xml
```

3. Modify the *URL rules configuration* to add a rule that emulates the path at which the search platform interface exists. For example:

```
<rule> <from>^/login/</from> <to last="true">/login.jsp</to>
</rule> <rule> <from>^/login</from> <to
last="true">/login.jsp</to> </rule> <rule match-type="wildcard">
<name>Ignore files on URL path.</name> <from>/**.*</from> <to
last="true">/$0</to> </rule> <rule> <from>^/mock-
response/</from> <to last="true">/WEB-INF/pages/mock-
response.xml</to> </rule> <rule match-type="wildcard">
<name>HTML 5 URL Rewrite</name> <from>/**</from> <to
last="true">/index.jsp</to> </rule>
```

4. Configure the platform host to point to the relevant URL:

```
name: twigkit.search.solr.Solr result-id-field: url host:
http://localhost:8080/mock-response defaultQuery: test
```

The application is now configured to use this response XML file instead of a live platform whenever a query is issued. Obviously this will mean the same result set is retrieved no matter how the query is modified through the user interface. It is possible to mock different result sets by adding more complex URL rules to intercept the different parameters modified by interacting with Appkit UI components, however this is generally not recommended.

Share Objects Between Fusion Apps

How-to

> Compatible versions: 4.2, 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion Server*, *Fusion*

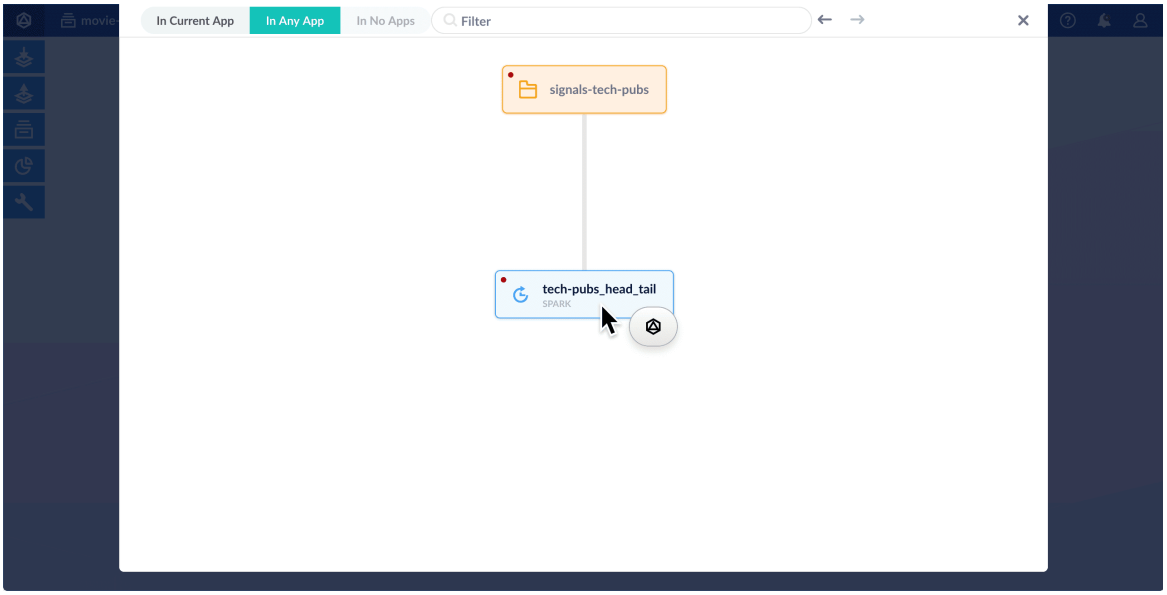
Sharing objects between apps lets you reuse parts of apps and synchronize apps without duplicating objects. For example, after you have developed a query pipeline that is well-suited to your data and search application, you can reuse it in another app with a different set of datasources.

Important

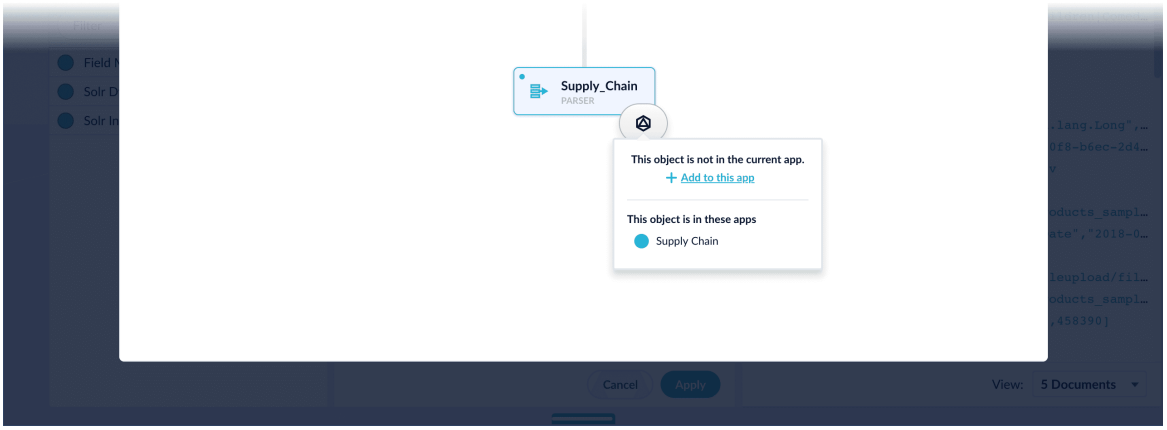
When a shared object is modified in one app, the changes affect all other apps that share the object.

To share an object between apps:

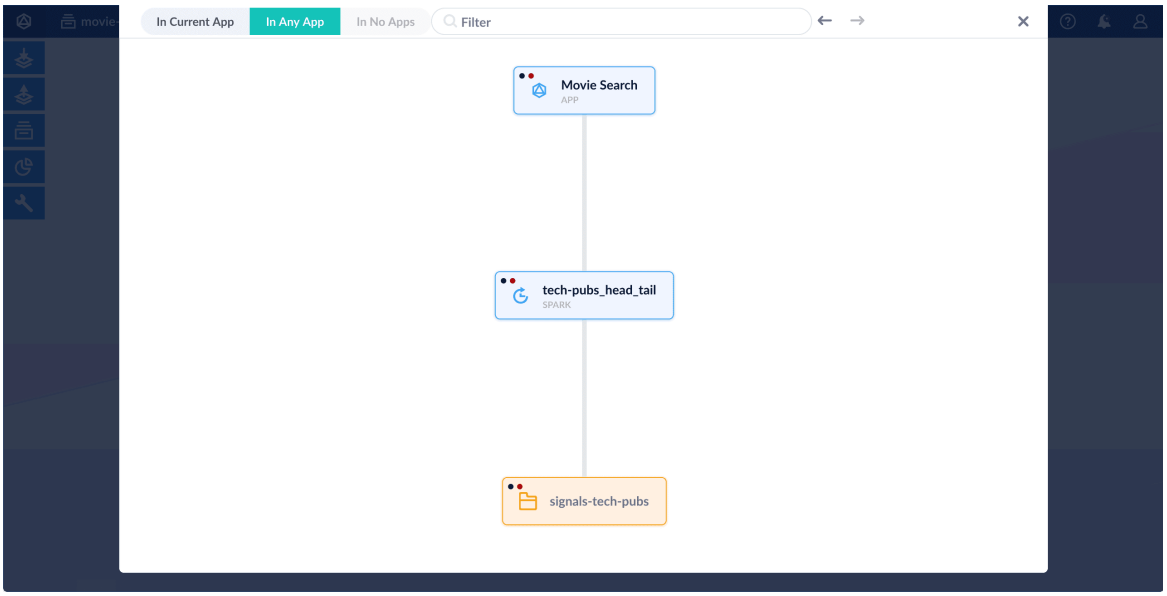
1. Navigate to the app that does not yet include the object.
2. Navigate to **System** > **Object Explorer** or click **Find Objects**.
3. Click **In Any App**.
4. Select the object you want to include in this app.
5. Hover over the object to reveal the app icon.



6. Click the app icon.
7. Select **Add to this app**.



Object Explorer now displays a link between the object and this app:

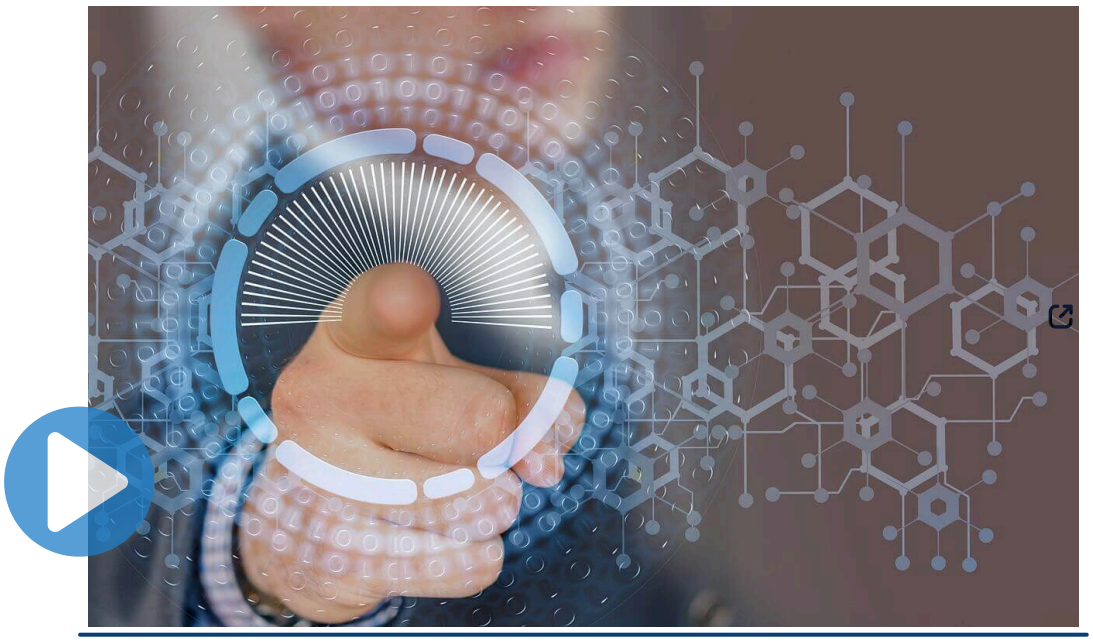


Now you will see the shared object in the Fusion workspace for this app.

LucidAcademy

Lucidworks offers free training to help you get started.

The **Course** for **Fusion Applications and Collections** focuses on the features of Fusion apps and collections and best practices when working with them:



Visit the LucidAcademy to see the full training catalog.

Share Objects Between Managed Fusion Apps

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

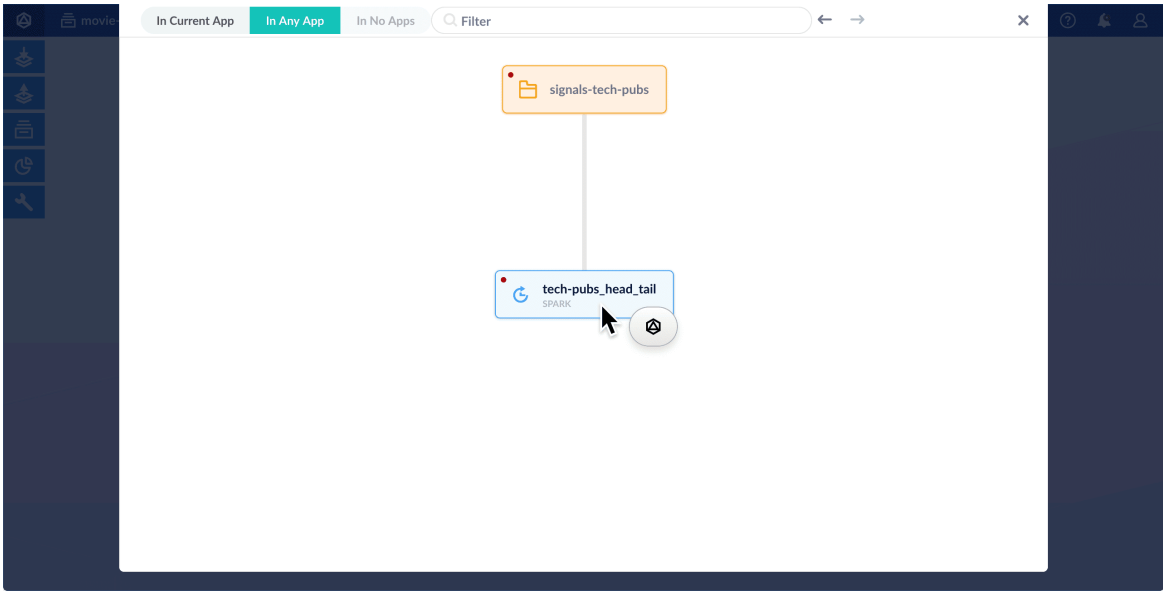
Sharing objects between apps lets you reuse parts of apps and synchronize apps without duplicating objects. For example, after you have developed a query pipeline that is well-suited to your data and search application, you can reuse it in another app with a different set of datasources.

Important

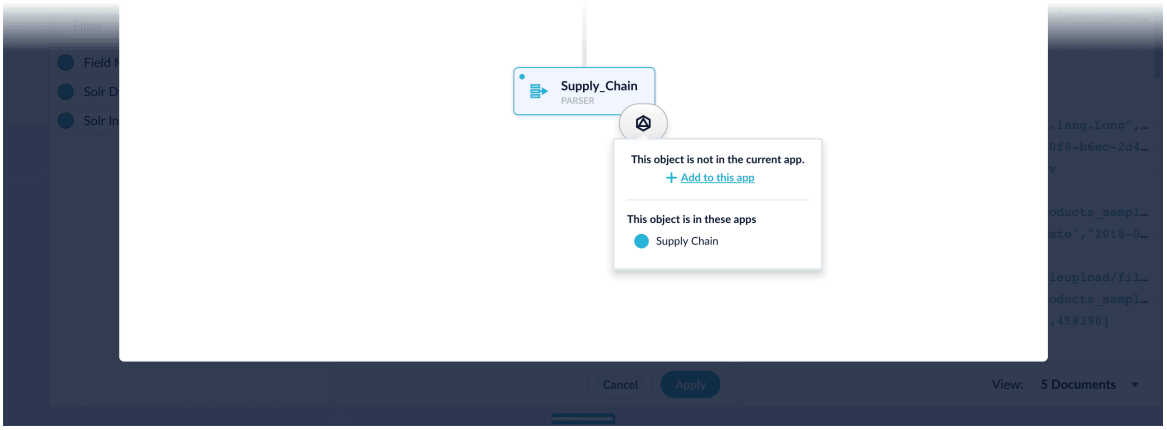
When a shared object is modified in one app, the changes affect all other apps that share the object.

To share an object between apps:

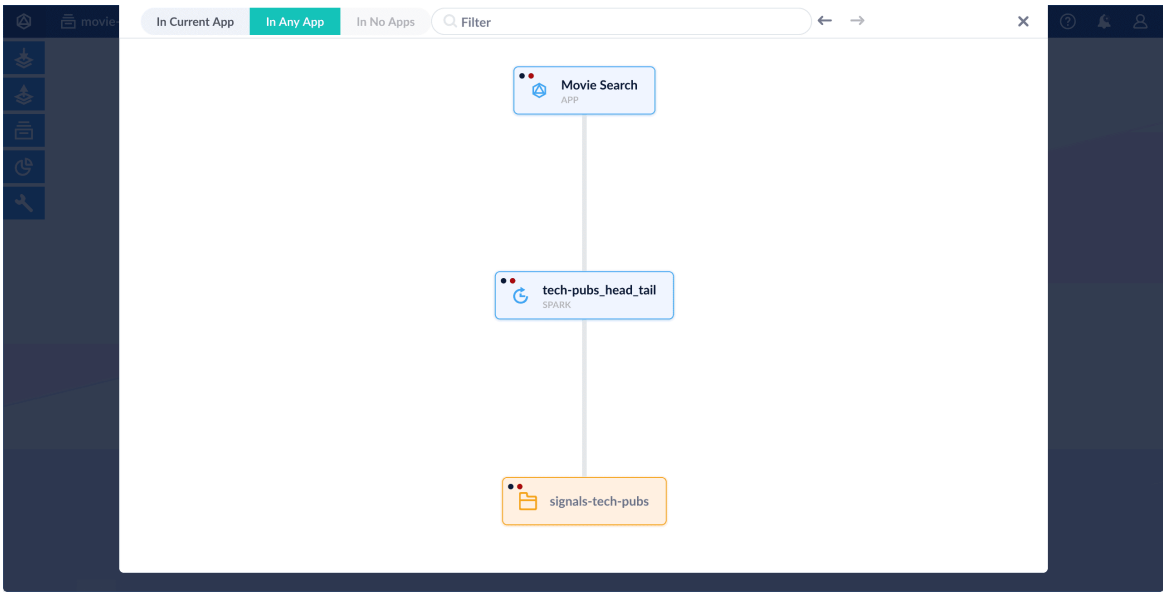
1. Navigate to the app that does not yet include the object.
2. Navigate to **System** > **Object Explorer** or click **Find Objects**.
3. Click **In Any App**.
4. Select the object you want to include in this app.
5. Hover over the object to reveal the app icon.



6. Click the app icon.
7. Select **Add to this app**.



Object Explorer now displays a link between the object and this app:

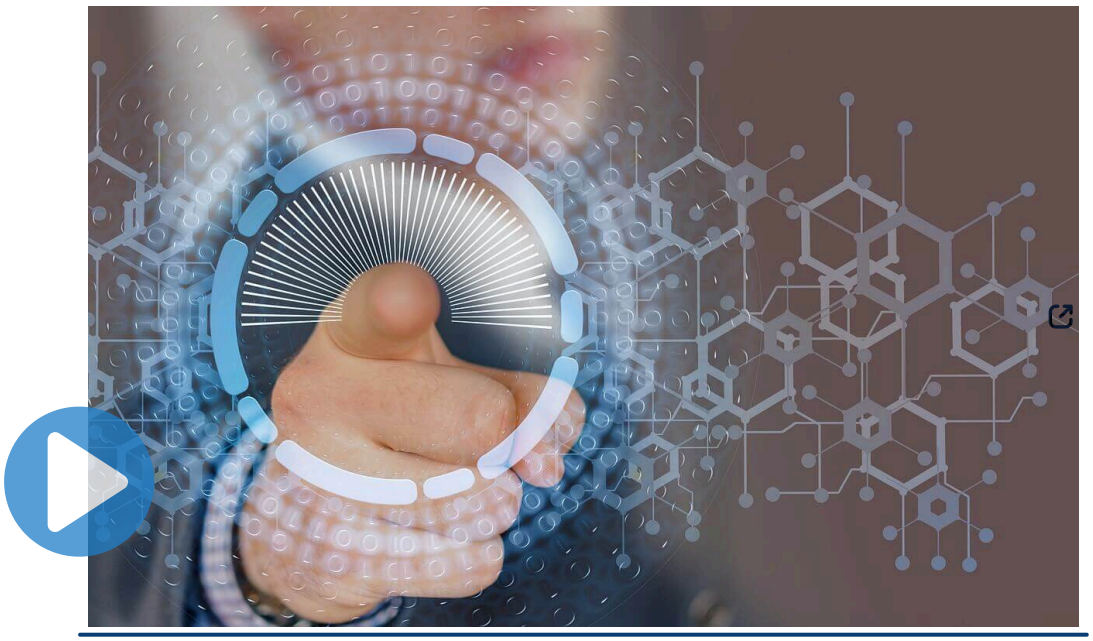


Now you will see the shared object in the Fusion workspace for this app.

LucidAcademy

Lucidworks offers free training to help you get started.

The **Course** for **Fusion Applications and Collections** focuses on the features of Fusion apps and collections and best practices when working with them:



Visit the LucidAcademy to see the full training catalog.

Shut Down App Studio

How-to

> Compatible versions: 4.2
>
> Works with: *App Studio*

Version 4.1 and later

App Studio has two scripts for startup and shutdown in the root directory of an App Studio project (`app-studio`):

- `app-studio.sh` . A Bash script for Linux and macOS
- `app-studio.bat` . A batch file for Windows

You also use these scripts to *build an app from source*.

Obtain usage information about the scripts

For usage information, enter:

- Linux and macOS: `./app-studio.sh -help`
- Windows: `app-studio.bat -help`

For command-specific usage information, enter:

- Linux and macOS: `./app-studio *command* -help`
- Windows: `app-studio.bat *command* -help`

Below, optional parameters are indicated by square brackets, which you do not type.

Shutdown commands

- Linux and macOS: `./app-studio stop [-p port] [-V]`
- Windows: `app-studio.bat stop [-p port] [-V]`

Parameters

Bold italic text is a placeholder for a value. For example, for `-m *memory*` you might use `-m 4g` .

Parameter	Description
`-p *port*`	Specify the port on which the App Studio web server will listen. The default port is 8080.
`-V`	Output verbose messages from the script.

Stopping multiple App Studio instances

You can run multiple instances of App Studio on a single host by specifying different ports for all instances on startup, at the command line, using the `-p` parameter.

When running multiple instances, also use the `-p` flag to indicate which instance of App Studio to shut down, for example:

```
./app-studio.sh stop -p 9000
```

Version 4.0

Stop the server
`./app-studio.sh stop` Optionally: * `-s` or `--stop-port` + The port to use when issuing the `stop` command.
app-studio-standalone-stop.bat

What is next

- See *Create a New Web App* to learn how to create your first search app.

Smart Answers Cold Start Part 2

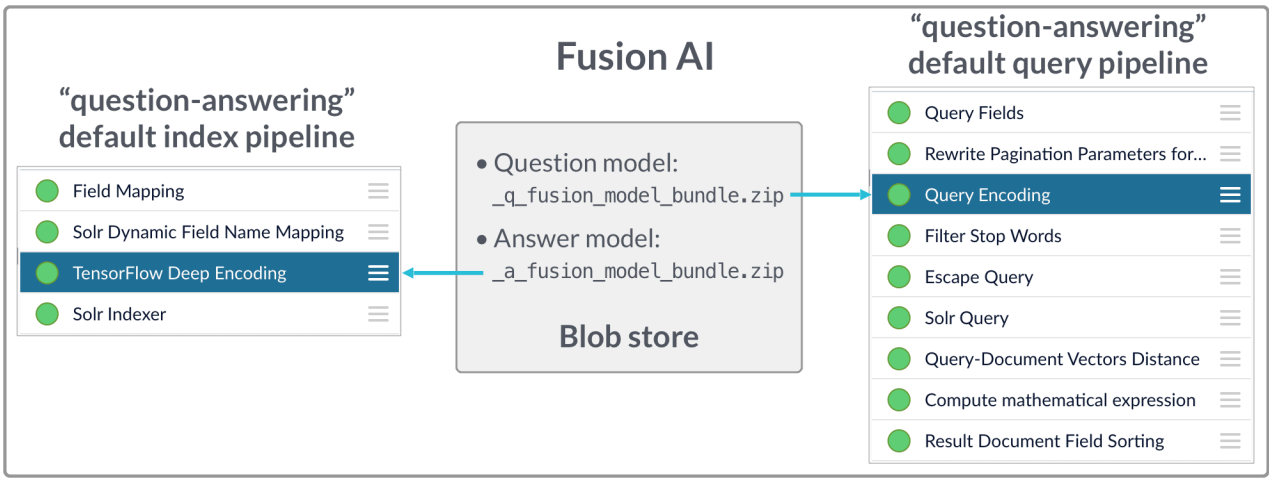
Model Deployment (Fusion 4.2 only)

How-to

> Compatible versions: 4.2
>
> Works with: *Fusion AI*

The Smart Answers deployment procedure for the FAQ solution and the cold start solution is the same:

1. *Upload the question and answer models* to the blob store.
2. *Configure the* `question-answering` *index pipeline* so that it uses the answer model.
3. *Configure the* `question-answering` *query pipeline* so that it uses the question model.



See also *Best Practices*.

How to upload the models to the blob store

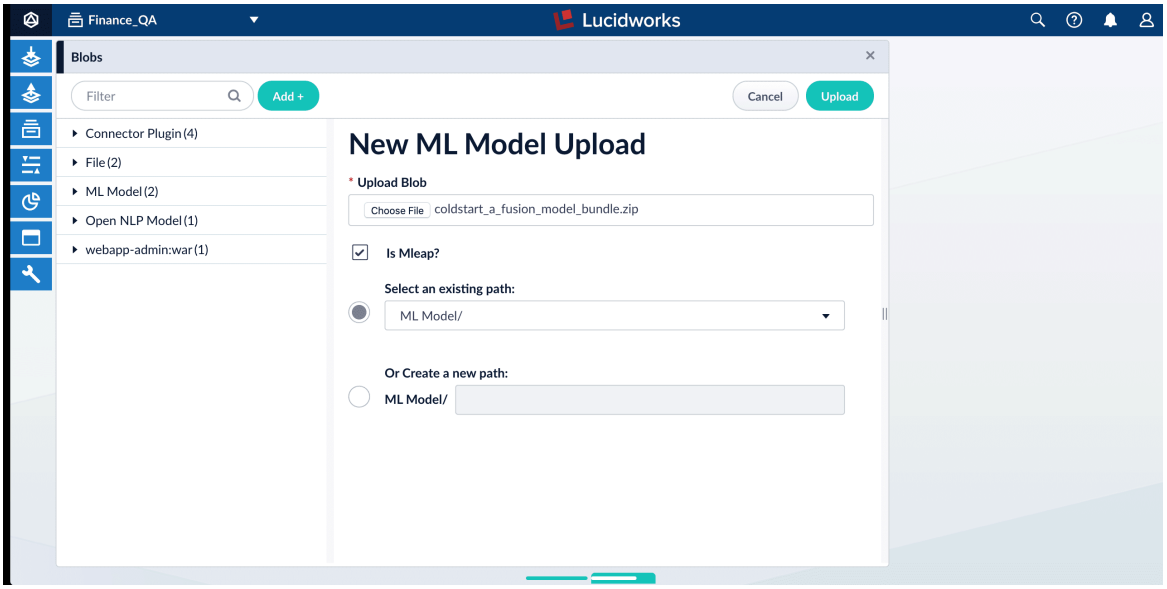
Two model files are generated by the *model training* Docker job:

- `x_a_fusion_model_bundle.zip` - The answer model.
- `x_q_fusion_model_bundle.zip` - The question model.

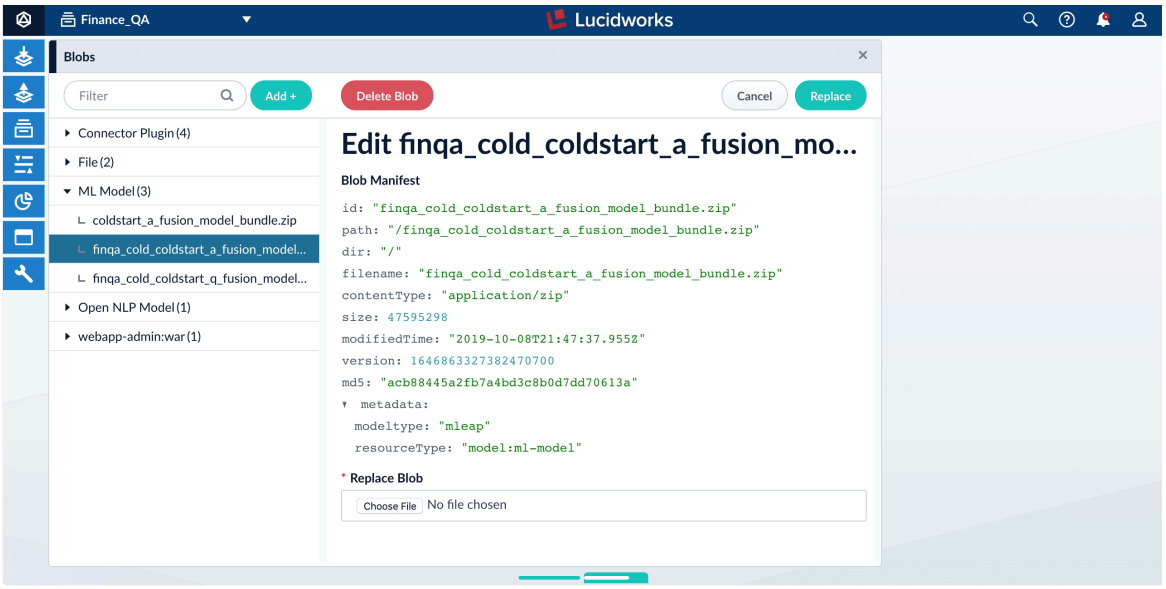
Both files must be deployed to the Fusion *blob store*.

How to upload the deep-learning models to the blob store

1. In the Fusion UI, navigate to **System** > **Blobs**.
2. Click **Add** > **ML Model**.
3. Click **Choose File** and select one of the model files.
4. Make sure the **Is Mleap?** checkbox is selected.



5. Click **Upload**.



6. Repeat steps 2-5 for the other model file.

Fusion configuration overview

- Fusion index pipeline
  The index pipeline uses the trained model in `x_a_fusion_model_bundle.zip` to generate dense vectors for the documents to be indexed.
- Fusion query pipeline
  The query pipeline uses the trained model in `x_q_fusion_model_bundle.zip` to generate dense vectors for incoming questions on the fly, then compare those with the indexed dense vectors for answers to find answers, or with the indexed dense vectors for historical questions to find similar questions.

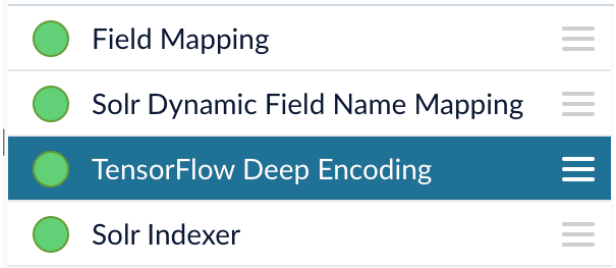
Another option is to have two separate index stages, one for questions and another for answers. Then, at query time, two query stages compute query-to-question distance and query-to-answer distance. Both scores are ensembled into a final similarity score. The two options are illustrated in *Pipeline setup examples*.

How to configure the default pipelines

If you have an AI license, then the following default index and query pipelines are included in any newly-created Fusion app:

Default index pipelines		Default query pipelines	
`question-answering`	For encoding one field.	`question-answering`	Calculates vectors distances between an encoded query and one document vector field. Should be used together with `question-answering` index pipeline.
`question-answering-dual-fields`	For encoding two fields (question and answer pairs, for example).	`question-answering-dual-fields`	Calculates vectors distances between an encoded query and two document vector fields. After that, scores are ensembled. Should be used together with the `question-answering-dual-fields` index pipeline.
See *Configure the index pipeline* below.		See *Configure the query pipeline* below.	

Configure the index pipeline



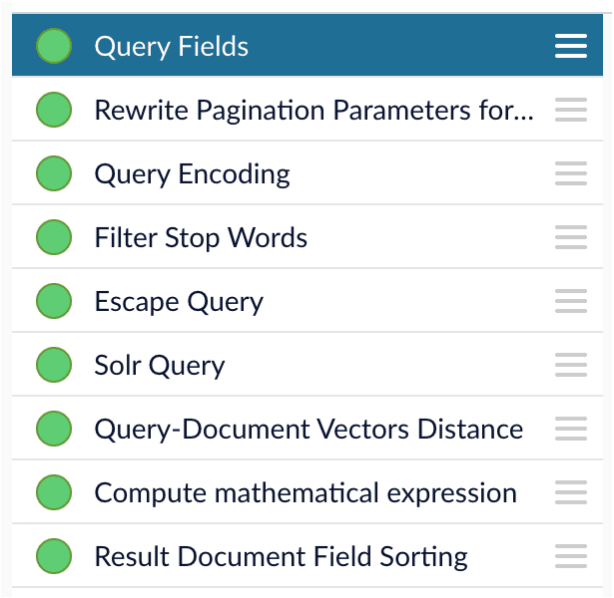
1. Open the Index Workbench.
2. Load or create your datasource using the default question-answering index pipeline.
3. In the *TensorFlow Deep Encoding stage*, change the value of **TensorFlow Deep Learning Encoder Model ID** to the model ID of the `x_a_fusion_model_bundle.zip` model that was uploaded to the blob store.

   Tip

   The `a` (answer) model allows you to encode longer text.

4. Change **Document Feature Field** to the document field name to be processed and encoded into dense vectors.
5. Save the datasource.
6. Index your data.

Configure the query pipeline



1. Open the Query Workbench.
2. Load or create your datasource using one of the default question-answering query pipelines.
3. In the *Query Fields stage*, update **Return Fields** to return additional fields that should be displayed with each answer, such as fields corresponding to title, text, or ID.
   It is recommended that you remove the asterisk (`*`) field and specify each individual field you want to return, as returning too many fields will affect runtime performance.
   Do not remove `compressed_document_vector_s` , `document_clusters_ss` , and `score` as these fields are necessary for later stages
4. In the *TensorFlow Deep Encoding stage*, change **TensorFlow Deep Learning Encoder Model ID** value to the model ID of the `x_q_fusion_model_bundle.zip` model that was uploaded to the blob store.

   Tip

   The (`q`) (question) model is slightly more efficient for short natural-language questions.

5. Save the query pipeline.

Pipeline Setup Examples

- *Example 1: Index and retrieve question and answer separately*
- *Example 2: Index and retrieve question and answer together*

Example 1: Index and retrieve question and answer separately

Based on your search Web page design, you can put best-matched questions and answers in separate sections, or if you only want to retrieve answers and serve to chatbot app, please index them separately in different documents.

For example, in the picture below, we construct the input file for the index pipeline such that the text part of the question/answer is stored in `text_t` , and we add an additional field `type_s` whose value is "question" or "answer" to separate the two types.

In the TensorFlow Deep Encoding stage, we specify **Document Feature Field** as `text_t` so that `compressed_document_vector_s` is generated based on this field.

At search time, we can apply a filter query on the `type_s` field to return either a question or an answer.

You can achieve a similar result by using the default `question-answering` index and query pipelines.

Example 2: Index and retrieve question and answer together.

If you prefer to show question and answer together in one document (that is, treat the question as the title and the answer as the description), you can index them together in the same document. It's similar to the `question-answering-dual-fields` index and query pipelines default setup.

For example, in the picture below, we added two TensorFlow Deep Encoding stages and named them Answers Encoding and Questions Encoding respectively. In the Questions Encoding stage, we specify **Document Feature Field** to be question_t, and changed the default values for **Vector Field**, **Clusters Field** and **Distances Field** to question_vector_ds, question_clusters_ss and question_distances_ds respectively. In the Answers Encoding stage, we specify **Document Feature Field** to be answer_t, and changed the default values for **Vector Field**, **Clusters Field** and **Distances Field** to answer_vector_ds, answer_clusters_ss and answer_distances_ds respectively. (Detailed information of the above field setup please refer to the "Appendix C: Detailed Pipeline Setup" section.)

Since we have two dense vectors generated in the index (compressed_question_vector_s and compressed_answer_vector_s), at query time, we need to compute query to question distance and query to answer distance. This can be setup as the picture shown below. We added two **Vectors distance per Query/Document** stages and named them QQ Distance and QA Distance respectively. In the QQ Distance stage, we changed the default values for **Document Vector Field** and **Document Vectors Distance Field** to compressed_question_vector_s and qq_distance respectively. In the QA Distance stage, we changed the default values for **Document Vector Field** and **Document Vectors Distance Field** to compressed_answer_vector_s and qa_distance respectively. (Detailed information of the above field setup please refer to the "Appendix C: Detailed Pipeline Setup" section.)

Now we have two distances (query to question distance and query to answer distance), we can ensemble them together with Solr score to get a final ranking score. This is recommended especially when you have limited FAQ dataset and want to utilize both question and answer information. This ensemble can be done in the **Compute mathematical expression** stage as shown below.

Query pipeline evaluation

Stages and configurations of the query pipeline will impact ranking results. For example, we may need to make decisions about whether to use clustering option in the *TensorFlow Deep Encoding stage*, what should be the weights of each ranking score in the **Compute mathematical expression** stage, whether adding other Fusion stages (such as text tagger, ML stage) would help. Comparing two different query pipelines with different models and setups may be needed as well.

The **Query Pipeline evaluation** job in the training docker image can help evaluate the rankings of Fusion returned results from different pipeline setups, additionally help search for the best set of weights in the ensemble score.

In order to evaluate the rankings, we need to know the ground truth answers for some testing questions. For example, user can provide around 200 testing questions, and for each question, we need to know which indexed answers are the correct answers in Fusion, so that we can compute positions of correct answers in the current ranking. User needs to provide a file in CSV format which contains the testing questions (**Testing query field** parameter in the job) and correct answer text or id (**Ground truth field** parameter in the job), put this file in the mapped input folder where we put other training data files, then specify the file name in the **Evaluation file name** parameter. NOTE: if there are multiple matching answers for a question, please put question and answer pairs in different rows. In Fusion, there should be a returned field which contains values that match the answer text or id, please specify this field in the **Answer or id field in Fusion** parameter. NOTE: if use answer text, please make sure the formatting of answer text in the evaluation file is the same as the answer text in Fusion so that we can find matches. Several Fusion access parameters are also needed in order to grab results from Fusion in batch, such as login username, password, host ip, app name, collection and pipeline name.

The job will provide a variety of metrics (controlled by **Metrics list *parameter) at different positions (controlled by *Metrics@k list** parameter) in the logs for the chosen final ranking score (specified in **Ranking score** parameter). For example, if choose **Ranking score** as "ensemble_score", the program will rank results by the ensemble score returned in **Compute mathematical expression** stage. If choose **Ranking score** as "score" (default field name for Solr score in Fusion), then the ranking evaluation will be based only on Solr score. In addition to the metrics in logs, a CSV results evaluation file will be generated. It will provide correct answer positions for each testing question as well as top returned results for each field specified in **Return fields** parameter. User can specify the file name for this results file in the **Ranking results file name** parameter, then download the file from `http://<external hostname>:5550/evaluation`.

Another function of this job is to help choose weights for different ranking scores such as Solr score, query to question distance, query to answer distance in **Compute mathematical expression** stage. If interested in performing this weights selection, please choose **Whether perform weights selection *parameter to true, list the set of score names in *List of ranking scores for ensemble** parameter. Since we can use different scaling methods for Solr score in the stage, please choose which Solr scale function you used in the stage in **Solr scale function** parameter. **Target metric to use for weight selection** parameter allows to specify metric that should be optimized during weights selection, for example `recall@3`. Metric values at different positions for different weights combinations will be shown in the log, sorted descendingly based on metric specified above. NOTE: Weights

selection can take a while to run for big evaluation datasets, thus if only interested in comparing pipelines, please turn this function off by specifying **Whether perform weights selection** parameter to false.

There are a few additional advanced parameters that might be useful but not required to provide. **Additional query parameters** allows to provide extra query parameters like `rowsFromSolrToRerank` in a dictionary format. **Sampling proportion** and **Sampling seed** provides a possibility to run evaluation job only on a sample of data.

Tip: Please make sure the Ranking score, answer text or id, and list of ranking scores for ensemble (if perform weights selection) are returned by Fusion by checking the fields returned in the Query Workbench. User can setup the returned fields in Return Fields section of Query Fields stage.

Smart Answers Instance Setup (Fusion 4.2 only)

How-to

> Compatible versions: 4.2
>
> Works with: *Fusion AI*

Set up a CPU AWS/Azure instance:

1. SSH into the instance.
2. Install Docker .
3. Start the Docker service:

```
$ sudo service docker start
```

4. Add the current user to the “docker” group so you can execute Docker commands without using `sudo`:

```
$ sudo usermod –aG docker ${USER}
```

5. Log out and log back in again to pick up the new Docker group permissions.
   You can do this by closing your current SSH terminal window and reconnecting to your instance in a new one. Your new SSH session will have the appropriate Docker group permissions.

Set up a GPU AWS instance:

1. SSH into the instance.
2. Install the official NVIDIA driver package:

```
$ sudo apt-key adv --fetch-keys
http://developer.download.nvidia.com/compute/cuda/repos/ubuntu16
04/x86_64/7fa2af80.pub $ sudo sh -c 'echo "deb
http://developer.download.nvidia.com/compute/cuda/repos/ubuntu16
04/x86_64 /" > /etc/apt/sources.list.d/cuda.list' $ sudo apt-get
update *&&* sudo apt-get install -y --no-install-recommends
cuda-drivers
```

Set up a GPU Azure VM:

1. Download and install the CUDA drivers from the NVIDIA website. For example, for Ubuntu 16.04 LTS:

```
$ CUDA_REPO_PKG=cuda-repo-ubuntu1604_10.0.130-1_amd64.deb $ wget
-O /tmp/${CUDA_REPO_PKG}
http://developer.download.nvidia.com/compute/cuda/repos/ubuntu16
04/x86_64/${CUDA_REPO_PKG} $ sudo dpkg -i /tmp/${CUDA_REPO_PKG}
$ sudo apt-key adv --fetch-keys
https://developer.download.nvidia.com/compute/cuda/repos/ubuntu1
604/x86_64/7fa2af80.pub $ rm -f /tmp/${CUDA_REPO_PKG} $ sudo
apt-get update $ sudo apt-get install cuda-drivers
```

1. Download and install Docker:

```
sudo apt install docker.io
```

Solr Highlighting

How-to

> Compatible versions: 4.2
>
> Works with: *App Studio*

App Studio is able to leverage the Solr Highlighting component to visually distinguish any hits in a search result. There are a number of parameters that can be included in any Solr queries where highlighting should take place. These parameters can be added either in an App Studio `<search:query>` tag, or in a Fusion query pipeline.

App Studio also allows for custom styling of any highlighted hits, which is covered at the end of this page.

How to Configure Solr Highlighting in App Studio

1. Open any HTML page that uses a `<search:query>` tag where highlighting should take place.
2. Modify the `<search:query>` tag to include the required `hl.fl` parameter (a complete list of optional parameters can be found here):

Note

If not using App Studio Enterprise, an additional `hl` parameter must be included and set to `true` for highlighting to work. App Studio Enterprise sets this parameter to true by default in `../src/main/resources/conf/platforms/fusion/fusion.conf` (via `highlighting: true`).

```
<search:query var="query" parameters="*" results-per-page="12">
<query:custom name="hl.fl" value="description_t"></query:custom>
</search:query>
```

Parameter Name	Parameter Value	Description
hl.fl	description_t	comma separated list of fields where highlighting should occur
hl (only required if not using App Studio Enterprise)	true	Boolean flag for whether or not highlighting should occur

3. Refresh the search UI in your browser and any hits will appear highlighted. NOTE: you must have the field(s) specified in the `hl.fl` parameter above displayed using a `<search:field>` on the HTML page.
4. See the Styling Highlighted Hits section below for steps to add custom styling to search hits.

How to Configure Solr Highlighting in a Fusion Query Pipeline

1. In the Query Workbench, add or modify an existing Additional Query Parameters stage.
2. Add the following required parameters (a full list of optional parameters can be found here):

Note

If not using App Studio Enterprise, an additional `hl` parameter must be included and set to `true` for highlighting to work. App Studio Enterprise sets this parameter to true by default in `../src/main/resources/conf/platforms/fusion/fusion.conf` (via `highlighting: true`).

Parameter Name	Parameter Value	Description
hl.fl	description_t	comma separated list of fields where highlighting should occur
hl (only required if not using App Studio Enterprise)	true	Boolean flag for whether or not highlighting should occur

3. Save the query pipeline, and return to your search application.
4. Run a new search that utilizes the query pipeline modified in step 2 and any hits will appear highlighted. NOTE: you must have the field(s) specified in the `hl.fl` parameter above displayed using a `<search:field>` on the HTML page.
5. See the Styling Highlighted Hits section below for steps to add custom styling to search hits.

Styling Highlighted Hits

1. Add the following CSS in order to add custom styling to Solr highlighting:
    a. If using App Studio Enterprise, this should go in `../src/main/webapp/styles/includes/custom.less`
    b. Any custom CSS can be used in place of the example attributes below.

```
.result { em { background-color: yellow; font-weight:
bold; font-style: normal; } }
```

Spark Configuration for Fusion 4.x

How-to

> Compatible versions: 4.2
>
> Works with: *Fusion Server*

Spark has a number of configuration properties. In this section, we will cover some of the key settings you will need to use Fusion's Spark integration.

For the full set of Fusion's spark-related configuration properties, see the *Spark Jobs API*.

Spark master/worker resource allocation

Note

If you co-locate Spark workers and Solr nodes on the same server, then be sure to reserve some CPU for Solr to avoid a compute intensive Spark job from starving Solr of CPU resources.

Number of cores allocated

To change the CPU usage per worker, you need to use the *Fusion configuration API* to update this setting, as in the following example.

```
curl -u USERNAME:PASSWORD -H 'Content-type:application/json' -X PUT -d
'6' \
http://localhost:8764/api/configurations/fusion.spark.worker.cores
```

You can also over-allocate cores to a spark-worker, which usually is recommended for hyper-threaded cores by setting the property `spark-worker.envVars` to `SPARK_WORKER_CORES=<number of cores>` in the `fusion.cors` (`fusion.properties` in Fusion 4.x) file on all nodes hosting a spark-worker. For example, a r4.2xlarge instance in EC2 has 8 CPU cores, but the following configuration will improve utilization and performance:

```
spark-
worker.envVars=SPARK_WORKER_CORES=16,SPARK_SCALA_VERSION=2.11,SPARK_PU
BLIC_DNS=${default.address},SPARK_LOCAL_IP=${default.address}
```

You can obtain the IP address that the Spark master web UI binds to with this API command:

```
curl http://<FUSION_HOST>/api/spark/master
```

Tip

We encourage you to set the `default.address` property in `fusion.cors` (`fusion.properties` in Fusion 4.x) to ensure that all Spark processes have a consistent address to bind to.

After making this change to your Spark worker nodes, you must restart the spark-worker process on each.

On Unix:

```
./spark-worker restart
```

Give this command from the `bin` directory below the Fusion home directory, for example, `/opt/fusion/latest.x`.

On Windows:

```
spark-worker.cmd restart
```

Give this command from the `bin` directory below the Fusion home directory, for example, `C:\lucidworks\fusion\latest.x`.

Memory allocation

The amount of memory allocated to each worker process is controlled by Fusion property `fusion.spark.worker.memory` which specifies the total amount of memory available for all executors spun up by that Spark Worker process. This is the quantity seen in the memory column against a worker entry in the Workers table.

The JVM memory setting (`-Xmx`) for the spark-worker process configured in the `fusion.cors` (`fusion.properties` in Fusion 4.x) file controls how much memory the spark-worker needs to manage executors (and not how much memory should be made available to your job(s)). When modifying the `-Xmx` value, use `curl` as follows:

```
curl -u USERNAME:PASSWORD -H 'Content-type:application/json' -X PUT -d
'8g' \
http://localhost:8764/api/configurations/fusion.spark.worker.memory
```

Tip

Typically, 512m to 1g is sufficient for the spark-worker JVM process.

The Spark worker process manages executors for multiple jobs running concurrently. For certain types of aggregation jobs you can also configure the per executor memory, but this can impact how many jobs you can run concurrently in your cluster. Unless explicitly specified using the parameter `spark.executor.memory`, Fusion calculates the amount of memory that can be allocated to the executor

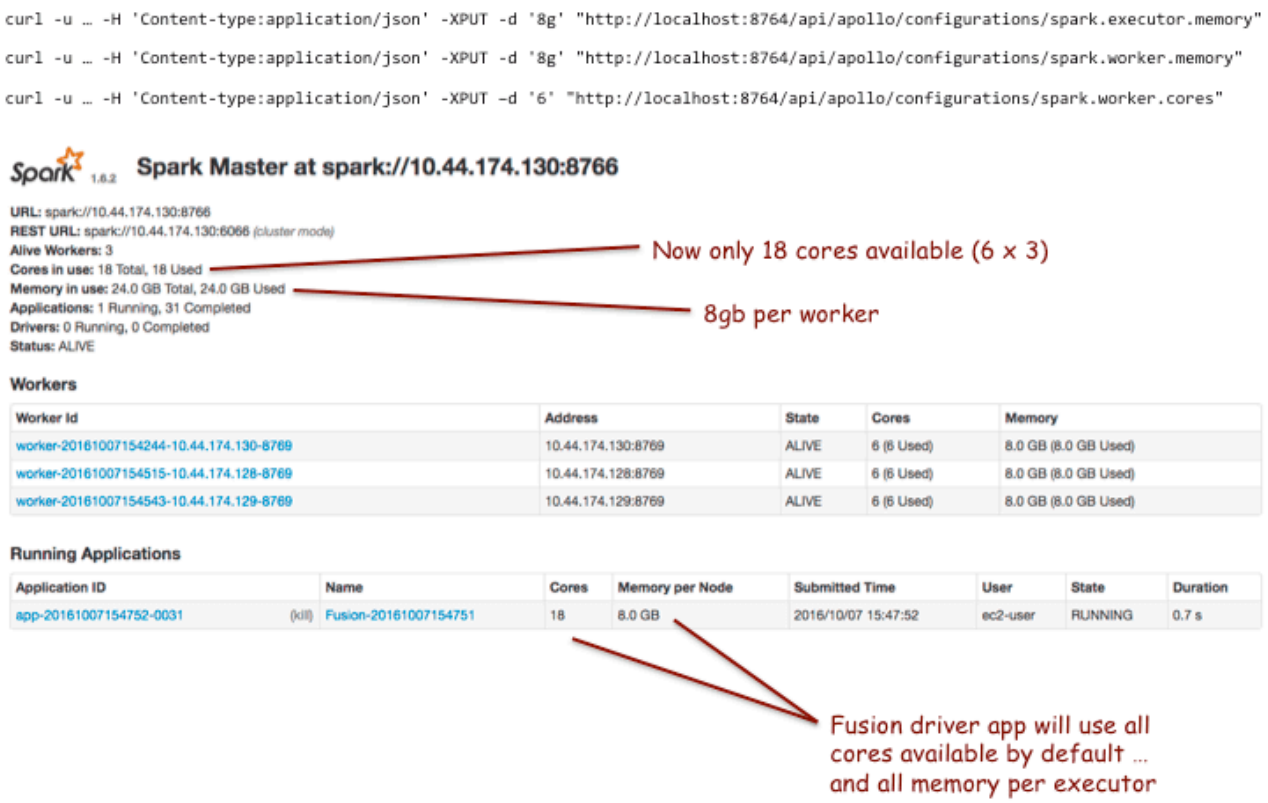
Aggregation Spark jobs always get half the memory of the amount assigned to the workers. This is controlled by the `fusion.spark.executor.memory.fraction` property, which is set to `0.5` by default.

For example, Spark workers have 4 Gb of memory by default and the executors for aggregator Spark jobs are assigned 2 Gb for each executor.

To let Fusion aggregation jobs use more of the memory of the workers, increase `fusion.spark.executor.memory.fraction` property to `1`. Use this property instead of the Spark executor memory property.

```
curl -u USERNAME:PASSWORD -H 'Content-type:application/json' -X PUT -d
'1' \
http://localhost:8764/api/configurations/fusion.spark.executor.memory.
fraction
```

After making these changes and restarting the workers, when we run a Fusion job, we get the following:



Cores per driver allocation

The configuration property `fusion.spark.cores.fraction` lets you limit the number of cores used by the Fusion driver applications (default and scripted). For example, in the screenshot above, we see 18 total CPUs available.

We set the cores fraction property to 0.5 via the following command:

```
curl -u USERNAME:PASSWORD -H 'Content-type:application/json' -X PUT -d
'0.5' \
http://localhost:8764/api/configurations/fusion.spark.cores.fraction
```

This cuts the number of available cores in half, as shown in the following screenshot:

Running Applications

Application ID	Name	Cores	Memory per Node	Submitted Time	User	State	Duration
app-20161007160422-0035 (kill)	Fusion-20161007160421	9	8.0 GB	2016/10/07 16:04:22	ec2-user	RUNNING	1 s

Ports used by Spark in Fusion

This table lists the default port numbers used by Spark processes in Fusion.

Port number	Process
4040	SparkContext web UI
7337	Shuffle port for Apache Spark worker
8767	Spark master web UI
8770	Spark worker web UI
8766	Spark master listening port
8769	Spark worker listening port
8772 (`spark.driver.port`)	Spark driver listening port
8788 (`spark.blockManager.port`)	Spark BlockManager port

If a port is not available, Spark uses the next available port by adding `1` to the assigned port number. For example, if 4040 is not available, Spark uses 4041 (if available, or 4042, and so forth).

Ensure that the ports in the above table are accessible, as well as a range of up to 16 subsequent ports. For example, open ports 8772 through 8787, and 8788 through 8804, because a single node can have more than one Spark driver and Spark BlockManager.

Spark-related directories and files in Fusion

The following directories and files are for Spark components and logs in Fusion.

Spark components

These directories and files are for Spark components:

Path (relative to Fusion home)	Notes
`bin/spark-master`	Script to manage (`start` , `stop` , `status` , etc.) the Spark Master service in Fusion
`bin/spark-worker`	Script to manage (`start` , `stop` , `status` , etc.) the Spark Worker service in Fusion
`bin/sql`	Script to manage (`start` , `stop` , `status` , etc.) the SQL service in Fusion
`bin/spark-shell`	Wrapper script to launch the interactive Spark shell with the Spark Master URL and shaded JAR
`apps/spark-dist`	Apache Spark distribution; contains all JAR files needed to run Spark in Fusion
`apps/spark/hadoop`	Hadoop home directory used by Spark jobs running in Fusion
`apps/spark/driver/lib`	Add custom JAR files to this directory to include in all Spark jobs
`apps/spark/lib`	JAR files used to construct the classpath for the `spark-worker` , `spark-master` , and `sql` services in Fusion
`var/spark-master`	Working directory for the `spark-master` service
`var/spark-worker`	Working directory for the `spark-worker` service; keep an eye on the disk usage under this directory as temporary application data for running Spark jobs is saved here
`var/spark-workDir-*`	Temporary work directories are created in when an application is running. They are removed after the driver is shut down or closed.
`var/sql`	Working directory for the SQL service
`var/api/work/spark-shaded-*.jar`	The shaded JAR built by the API service; contains all classes needed to run Fusion Spark jobs. If one of the jars in the Fusion API has changed, then a new shaded jar is created with an updated name.

Spark logs

These directories and files are for configuring and storing Spark logs:

Path (relative to Fusion home)	Notes
Log configuration	
`conf/spark-master-log4j2.xml`	Log configuration file for the `spark-master` service
`conf/spark-worker-log4j2.xml`	Log configuration file for the `spark-worker` service
`conf/spark-driver-log4j2.xml`	Log configuration file for the Spark Driver application launched by Fusion; this file controls the log settings for most Spark jobs run by Fusion
`conf/spark-driver-scripted-log4j.xml` *(Fusion 4.1+ only.)*	Log configuration file for custom script jobs and Parallel Bulk Loader (PBL) based jobs
`conf/spark-driver-launcher-log4j2.xml`	Log configuration file for jobs built using the Spark Job Workbench
`conf/spark-executor-log4j2.xml`	Log configuration file for Spark executors; log messages are sent to STDOUT and can be viewed from the Spark UI
`conf/sql-log4j2.xml`	Log configuration file for the Fusion SQL service
Logs	
`var/log/spark-master/*`	Logs for the `spark-master` service
`var/log/spark-worker/*`	Logs for the `spark-worker` service
`var/log/sql/*`	Logs for the `sql` service
`var/log/api/spark-driver-default.log`	Main log file for built-in Fusion Spark jobs
`var/log/api/spark-driver-scripted.log`	Main log file for custom script jobs
`var/log/api/spark-driver-launcher.log`	Main log file for custom jobs built using the Spark Job Workbench

Connection configurations for an SSL-enabled Solr cluster

You will need to set these Java system properties used by SolrJ:

- `javax.net.ssl.trustStore`
- `javax.net.ssl.trustStorePassword`
- `javax.net.ssl.trustStoreType`

For the following Spark configuration properties:

- `spark.executor.extraJavaOptions`

- **fusion.spark.driver.jvmArgs**
- **spark.driver.extraJavaOptions**

```
> curl -H 'Content-type:application/json' -X PUT \ -d '-
Djavax.net.ssl.trustStore=/opt/app/jobs/ssl/solrtrust.jks -
Djavax.net.ssl.trustStorePassword=changeit -
Djavax.net.ssl.trustStoreType=jks' \
"http://localhost:8764/api/configurations/spark.executor.extraJavaOpti
ons" > curl -H 'Content-type:application/json' -X PUT \ -d '-
Djavax.net.ssl.trustStore=/opt/app/jobs/ssl/solrtrust.jks -
Djavax.net.ssl.trustStorePassword=changeit -
Djavax.net.ssl.trustStoreType=jks' \
"http://localhost:8764/api/configurations/fusion.spark.driver.jvmArgs"
> curl -H 'Content-type:application/json' -X PUT \ -d '-
Djavax.net.ssl.trustStore=/opt/app/jobs/ssl/solrtrust.jks -
Djavax.net.ssl.trustStorePassword=changeit -
Djavax.net.ssl.trustStoreType=jks' \
"http://localhost:8764/api/configurations/spark.driver.extraJavaOption
s"
```

Spark Getting Started for Fusion 4.x

How-to

> Compatible versions: 4.2
>
> Works with: *Fusion Server*

The public GitHub repository Fusion Spark Bootcamp contains examples and labs for learning how to use Fusion's Spark features.

In this section, you will walk through some basic concepts of using Spark in Fusion. For more exposure, you should work through the labs in the Fusion Spark Bootcamp.

Starting the Spark Master and Spark Worker services

The Fusion run script `/opt/fusion/latest.x/bin/fusion` (on Unix) or `C:\lucidworks\fusion\latest.x\bin\fusion.cmd` (on Windows) does not start the `spark-master` and `spark-worker` processes. This reduces the number of Java processes needed to run Fusion and therefore reduces memory and CPU consumption.

Jobs that depend on Spark, for example, aggregations, will still execute in what Spark calls local mode. When in local mode, Spark executes tasks in-process in the driver application JVM. Local mode is intended for jobs that consume/produce small datasets.

One caveat about using local mode is that a persistent Spark UI is not available. But you can access the driver/job application UI at port `:4040` while the local SparkContext is running.

To scale Spark in Fusion to support larger data sets and to speed up processing, you should start the `spark-master` and `spark-worker` services.

On Unix:

```
./spark-master start ./spark-worker start
```

On Windows:

```
spark-master.cmd start spark-worker.cmd start
```

Give these commands from the `bin` directory below the Fusion home directory, for example, `/opt/fusion/latest.x` (on Unix) or `C:\lucidworks\fusion\latest.x` (on Windows).

Tip

To have the `spark-master` and `spark-worker` processes start and stop with `bin/fusion start` and `bin/fusion stop` (on Unix) or `bin\fusion.cmd start` and `bin\fusion.cmd stop` (on Windows), add them to the `group.default` definition in `fusion.cors` (`fusion.properties` in Fusion 4.x). For example:

In Fusion 4.1+

```
group.default = zookeeper, solr, api, connectors-
classic, connectors-rpc, proxy, webapps, admin-ui, log-
shipper, spark-master, spark-worker
```

In Fusion 4.0.x

```
group.default = zookeeper, solr, api, connectors-rpc,
connectors-classic, admin-ui, proxy, webapps, spark-
master, spark-worker
```

Viewing the Spark Master

After starting the master and worker services, direct your browser to `http://localhost:8767` to view the Spark master web UI, which should resemble this:

If you do not see the master UI and at least one worker in the ALIVE state, check these logs.

On Unix:

```
/opt/fusion/latest.x/var/log/spark-master/spark-master.log
/opt/fusion/latest.x/var/log/spark-worker/spark-worker.log
```

On Windows:

```
C:\lucidworks\fusion\latest.x\var\log\spark-master\spark-master.log
C:\lucidworks\fusion\latest.x\var\log\spark-worker\spark-worker.log
```

Use this Fusion API request to get the status of the Spark master:

```
curl http://localhost:8764/api/spark/master/status
```

This request should return a response of the form:

```
[ { "url" : "spark://192.168.1.9:8766", "status" : "ALIVE", "workers"
: [ { "id" : "worker-20161005175058-192.168.1.9-8769", "host" :
"192.168.1.9", "port" : 8769, "webuiaddress" :
"http://192.168.1.9:8770", "cores" : 8, "coresused" : 0, "coresfree" :
8, "memoryused" : 0, "memoryfree" : 2048, "state" : "ALIVE",
"lastheartbeat" : 1475711489460 } ], ...
```

If you have multiple Spark masters running in a Fusion cluster, each will be shown in the status but only one will be ALIVE; the other masters will be in STANDBY mode.

Tip

If you are operating a multi-node Spark cluster, we recommend running multiple Spark master processes to achieve high-availability. If the active one fails, the standby will take over.

Running a job in the Spark shell

After you have started the Spark master and Spark worker, run the Fusion Spark shell.

On Unix:

```
./spark-shell
```

On Windows:

```
spark-shell.cmd
```

Give these commands from the **`bin`** directory below the Fusion home directory, for example, **`/opt/fusion/latest.x`** (on Unix) or **`C:\lucidworks\fusion\latest.x`** (on Windows).

The shell can take a few minutes to load the first time because the script needs to download the shaded Fusion JAR file from the API service.

If ports are locked down between Fusion nodes, specify the Spark driver and BlockManager ports, for example:

On Unix:

```
./spark-shell --conf spark.driver.port=8772 --conf
spark.blockManager.port=8788
```

On Windows:

```
spark-shell.cmd --conf spark.driver.port=8772 --conf
spark.blockManager.port=8788
```

When the Spark shell is initialized, you will see the prompt:

```
scala>
```

Type **`:paste`** to activate paste mode in the shell and paste in the following Scala code:

```
val readFromSolrOpts = Map( "collection" -> "system_logs", "fields" ->
"host_s,level_s,type_s,message_txt,thread_s,timestamp_tdt", "query" ->
"level_s:[* TO *]" ) val logsDF =
spark.read.format("solr").options(readFromSolrOpts).load
logsDF.registerTempTable("fusion_logs") var sqlDF = spark.sql("""" |
SELECT COUNT(*) as num_values, level_s as level | FROM fusion_logs |
GROUP BY level_s | ORDER BY num_values desc | LIMIT 10""".stripMargin)
sqlDF.show(10,false)
```

Press CTRL+D to execute the script. Your results should resemble these results:

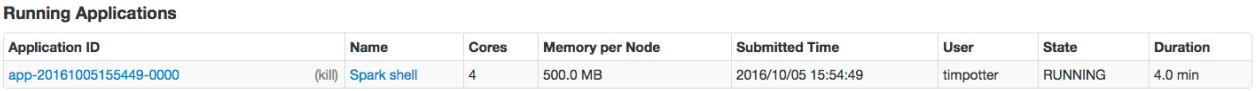
```
scala> :paste // Entering paste mode (ctrl-D to finish) val
readFromSolrOpts = Map( "collection" -> "system_logs", "fields" ->
"host_s,level_s,type_s,message_txt,thread_s,timestamp_tdt", "query" ->
"level_s:[* TO *]" ) val logsDF =
spark.read.format("solr").options(readFromSolrOpts).load
logsDF.registerTempTable("fusion_logs") var sqlDF = spark.sql("""" |
SELECT COUNT(*) as num_values, level_s as level | FROM fusion_logs |
GROUP BY level_s | ORDER BY num_values desc | LIMIT 10""".stripMargin)
sqlDF.show(10,false) // Exiting paste mode, now interpreting. warning:
there was one deprecation warning; re-run with -deprecation for
details +----------+-----+ |num_values|level| +----------+-----+ |3960
|INFO | |257 |WARN | +----------+-----+ readFromSolrOpts:
scala.collection.immutable.Map[String,String] = Map(collection ->
system_logs, fields ->
host_s,level_s,type_s,message_txt,thread_s,timestamp_tdt, query ->
level_s:[* TO *]) logsDF: org.apache.spark.sql.DataFrame = [host_s:
string, level_s: string ... 4 more fields] sqlDF:
org.apache.spark.sql.DataFrame = [num_values: bigint, level: string]
```

Do not worry about WARN log messages when running this script. They are benign messages from Spark SQL

Congratulations, you just ran your first Fusion Spark job that reads data from Solr and performs a simple aggregation!

The Spark master web UI

The Spark master web UI lets you dig into the details of the Spark job. In your browser (`http://localhost:8767`), there should be a job named "Spark shell" under running applications (the application ID will be different than the following screenshot):



Click the application ID, and then click the **Application Detail UI** link. You will see this information about the completed job:

Notice the tabs at the top of the UI that let you dig into details about the running application. Take a moment to explore the UI. It can answer these questions about your application:

- How many tasks were needed to execute this job?
- Which JARs were added to the classpath for this job? (Look under the Environment tab.)
- How many executor processes were used to run this job? Why? (Look at the Spark configuration properties under the Environment tab.)
- How many rows were read from Solr for this job? (Look under the SQL tab.)

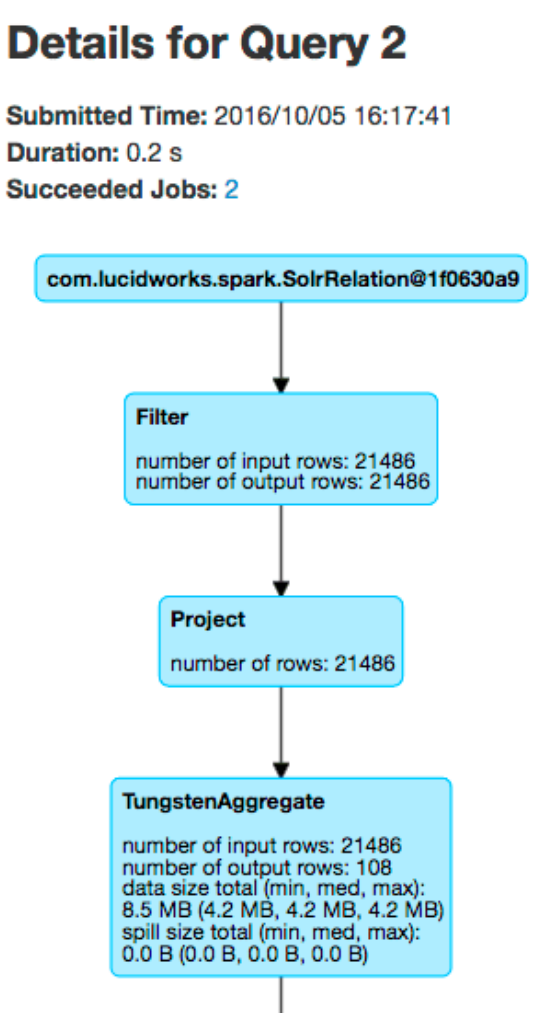
For the above run, the answers are:

- 205 tasks were needed to execute this job.
- The Environment tab shows that one of the JAR files is named `spark-shaded-*.jar` and was "Added By User".
- It took 2 executor processes to run this job. Each executor has 2 CPUs allocated to it and the `bin/spark-shell` script asked for 4 total CPUs for the shell application.
- This particular job read about 21K rows from Solr, but this number will differ based on how long Fusion has been running.

The key take-away is that you can see how Spark interacts with Solr using the UI.

Spark job tuning

Returning to the first question, why were 202 tasks needed to execute this job?



The reason is that SparkSQL defaults to using 200 partitions when performing distributed group by operations; see the property `spark.sql.shuffle.partitions`.

Because our data set is so small, let us adjust Spark so that it only uses 4 tasks. In the Spark shell, execute the following Scala:

```
spark.conf.set("spark.sql.shuffle.partitions", "4")
```

You just need to re-execute the final query and `show` command:

```
val readFromSolrOpts = Map( "collection" -> "logs", "fields" ->
"host_s,port_s,level_s,message_t,thread_s,timestamp_tdt" ) val logsDF
= spark.read.format("solr").options(readFromSolrOpts).load
logsDF.registerTempTable("fusion_logs") var sqlDF = spark.sql("""" |
SELECT COUNT(*) as num_values, level_s as level | FROM fusion_logs |
GROUP BY level_s | ORDER BY num_values desc | LIMIT 10""".stripMargin)
sqlDF.show(10,false)
```

Now if you look at the Job UI, you will see a new job that executed with only 6 executors! You have just had your first experience with tuning Spark jobs.

Spark Operations

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion*

These topics provide how-tos for Spark operations:

- *Configure Spark Job Resource Allocation*
- *Configure Spark Credentials in The Kubernetes Cluster*
- *Get Logs for A Spark Job*
- *Clean Up Spark Driver Pods*
- *Install The Spark History Server*
- *Configure The Spark History Server*
- *Access The Spark History Server*

Node Selectors

You can control which nodes Spark executors are scheduled on using a Spark configuration property for a job:

```
spark.kubernetes.node.selector.<LABEL>=<LABEL_VALUE>
```

Use the `LABEL` specified for the node, and the name of the node as the `LABEL_VALUE`. For example, if a node is labeled with `fusion_node_type=spark_only`, schedule Spark executor pods to run on that node using:

```
spark.kubernetes.node.selector.fusion_node_type=spark_only
```

Tip

In Fusion 5.5, Spark version 2.4.x does not support tolerations for Spark pods. As a result, Spark pods can't be scheduled on any nodes with taints in Fusion 5.5.

Cluster mode

Fusion 5 ships with Spark and operates in "cluster mode" on top of Kubernetes. In cluster mode, each Spark driver runs in a separate pod, and resources can be managed per job. Each executor also runs in its own pod.

Spark config defaults

The table below shows the default configurations for Spark. These settings are configured in the job-launcher config map, accessible using `kubectl get configmaps <release-name>-job-launcher`. Some of these settings are also configurable via Helm.

Spark Resource Configurations

Spark Configuration	Default value	Helm Variable
`spark.driver.memory`	3g	
`spark.executor.instances`	2	`executorInstances`
`spark.executor.memory`	3g	
`spark.executor.cores`	6	
`spark.kubernetes.executor.request.cores`	3	
`spark.sql.caseSensitive`	true	

Spark Kubernetes Configurations

Spark Configuration	Default value	Helm Variable
`spark.kubernetes.container.image.pullPolicy`	Always	`image.imagePullPolicy`
`spark.kubernetes.container.image.pullSecrets`		`image.imagePullSecrets`
`spark.kubernetes.authenticate.driver.serviceAccountName`	<name>-job-launcher-spark	
`spark.kubernetes.driver.container.image`	fusion-dev-docker.ci-artifactory.lucidworks.com	`image.repository`
`spark.kubernetes.executor.container.image`	fusion-dev-docker.ci-artifactory.lucidworks.com	`image.repository`

Spark Troubleshooting

How-to

> Compatible versions: 4.2
>
> Works with: *Fusion Server*

This article contains tips and techniques for troubleshooting Spark.

Begin troubleshooting process

1. First determine if the job is a Spark job. Spark jobs display in the Fusion UI Jobs panel and start with `spark:`. Additionally, a Spark job has a job ID attributed as **SPARK JOB ID** .
   View a comprehensive list of spark jobs:
   - Fusion 4.x:
     - *Spark Jobs*
     - *Spark Jobs API*

2. Next, check whether Spark services are enabled or if it is a local Spark instance instantiated to run Spark-related jobs.
   a. Go to `fusion.properties` and look for `group.default`. The line should have `spark-master` and `spark-worker` in the list, for example `group.default = zookeeper, solr, api, connectors-classic, connectors-rpc, proxy, webapps, admin-ui, spark-master, spark-worker, log-shipper` .
   b. Connect to `spark-client` via shell script by navigating to **fusion_home/bin/** directory and attempt to start via `./spark-shell`. It should successfully connect with a message, for example "**Launching Spark Shell with Fusion Spark Master: local**". If **spark-shell** fails to connect at all, copy the error message and pass it to Lucidworks support.

3. Try connecting to Apache Spark's admin URL. It should be accessible via `host:8764`. Check if scheduled jobs are complete or running.
4. Re-confirm the status of Spark services by querying the API endpoint at `http://localhost:8764/api/spark/info`. The API should return `mode` and `masterUrl`. If `mode` and `masterUrl` are **local**, then Spark services are not enabled explicitly or they are in a failure state. If Spark services are enabled then you will see `mode` as **STANDALONE**.
5. If Spark services were enabled but the API endpoint returned `mode` as **LOCAL** then there is an issue with starting Spark services.
   a. Restart Fusion with its Spark services as a first option.
   b. Check driver default logs via API endpoint and increase the **numbers of rows** param as required, for example`http://<FUSION_HOST>/api/spark/log/driver/default?rows=100` OR `http://localhost:8764/api/spark/log/driver/default?rows=100` .
   c. If you do not see an error stack trace in detail via the API endpoint, check the server tail Spark driver default logs by navigating to **fusion_home/var/log/api/** and do `tail -F spark-driver-default.log`, or copy the complete log files under `/fusion_home/var/log/api/` (for example, **spark-driver-default.log**, **spark-driver-scripted.log**, **api.log**, **spark-driver-script-stdout.log**) and share with Lucidworks to troubleshoot the actual issue.
      i. Logs required to troubleshoot Spark jobs failure are responses to below endpoints
         A. `http://localhost:8764/api/spark/master/config`
         B. `http://localhost:8764/api/spark/worker/config`
         C. `http://localhost:8764/api/spark/master/status`
         D. **https://host:8764/api/spark/info**
         E. **https://host:8764/api/spark/configurations**
         F. **http://192.168.29.185:8764/api/apollo/configurations**

Known issues and solutions

1. Standard Fusion setup without Spark services enabled expects the Spark jobs to work in local mode, however they are failing. When Spark services are not configured to start, a local Spark instance is instantiated to run Spark-related jobs. This Spark instance could possibly have issues.
   Steps
   a. Run the curl command:

```
curl -u USERNAME:PASSWORD -X POST -H "Content-
type:application/json"
http://localhost:8764/api/apollo/configurations/fusion.spa
rk.driver.jar.exclusions -d
".*io.grpc.*,.*org.apache.spark.*,.*org.spark-
project.*,.*org.apache.hadoop.*,.*org.apache.derby.*,.*spa
rk-assembly.*,.*spark-network.*,.*spark-
examples.*,.*\\/hadoop-.*,.*\\/tachyon.*,.*\\/datanucleus.
*,.*\\/scala-library.*,.*\\/solr-commons-csv.*,.*\\/spark-
csv.*,.*\\/hive-jdbc-shaded.*,.*\\/sis-
metadata.*,.*\\/bcprov.*,.*spire.*,.*com.chuusai.*,.*shape
less.*" curl -u USERNAME:PASSWORD -X POST -H "Content-
type: application/json" -d "true"
http://host:8764/api/apollo/configurations/spark.sql.caseS
ensitive
```

b. Remove any shaded jars on file system.

```
find . -name "spark-shaded*jar" -exec rm {} \;
```

c. Restart Fusion, specifically API and spark-worker services (if running).

2. Spark job (script or aggregation) is not getting all the resources available on the workers. By default, each application is only configured to get 0.5 of available memory on the cluster and 0.8 of available cores. Refer below links on how to make changes.
   a. *Number of instances and cores allocated*
   b. *Memory allocation*

Other troubleshooting steps

Log API endpoints for Spark jobs

Log endpoints are useful for debugging Spark jobs on multiple nodes. In a distributed environment, the log endpoints parse the last N log lines from different Spark log files on multiple nodes and output the responses from all nodes as `text/plain` (which renders nicely in browsers) sorted by the timestamp.

The REST API Reference documents *log endpoints for Spark jobs*. The URIs for the endpoints contain `/api/spark/log`.

The most useful log API endpoint is the `spark/log/job/` endpoint, which goes through all Fusion REST API and Spark logs, filters the logs by the `jobId` (using MDC, the mapped diagnostic context), and merges the output

from different files.

For example, to obtain log content for the job ***jobId*** :

```
curl –u USERNAME:PASSWORD
"https://FUSION_HOST:6764/api/spark/log/job/jobId"
```

Note

Log endpoints will only output data from log files on nodes on which the API service is running.

SPECIFIC ISSUES

These are some specific issues you might encounter.

Job hung in waiting status

Check the logs for a message that looks like:

```
2016–10–07T11:51:44,800 – WARN [Timer–0:Logging$class@70] – Initial
job has not accepted any resources; check your cluster UI to ensure
that workers are registered and have sufficient resources
```

If you see this, then it means your job has requested more CPU or memory than is available. For instance, if you ask for 4g but there is only 2g available, then the job will just hang in WAITING status.

Lost executor due to heartbeat timeout

If you see errors like the following:

```
2016-10-09T19:56:51,174 - WARN [dispatcher-event-loop-
5:Logging$class@70] - Removing executor 1 with no recent heartbeats:
160532 ms exceeds timeout 120000 ms 2016-10-09T19:56:51,175 - ERROR
[dispatcher-event-loop-5:Logging$class@74] - Lost executor 1 on ip-10-
44-188-82.ec2.internal: Executor heartbeat timed out after 160532 ms
2016-10-09T19:56:51,178 - WARN [dispatcher-event-loop-
5:Logging$class@70] - Lost task 22.0 in stage 1.0 (TID 166, ip-10-44-
188-82.ec2.internal): ExecutorLostFailure (executor 1 exited caused by
one of the running tasks) Reason: Executor heartbeat timed out after
160532 ms
```

This is most likely due to an OOM in the executor JVM (preventing it from maintaining the heartbeat with the application driver). However, we have seen cases where tasks fail, but the job still completes, so you will need to wait it out to see if the job recovers.

Another situation when this can occur is when a shuffle size (incoming data for a particular task) exceeds 2GB. This is hard to predict in advance because it depends on job parallelism and the number of records produced by earlier stages. The solution is to re-submit the job with increased job parallelism.

Spark Master will not start on EC2

See aws-instances-and-java-net-unknownhostexception for a solution.

For custom Spark troubleshooting, see *Troubleshoot Custom Spark Jobs*.

Split Couchbase Documents

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion Connectors*, *Fusion Server*, *Fusion*

The Couchbase V1 connector uses the Cross-Datacenter Replication (XDCR) feature of Couchbase to retrieve data stored in Couchbase continuously in real-time.

Because Couchbase has a flexible data model, documents may have a nested JSON structure. It is possible to split nested documents with a `splitpath` property, which uses an XPath-style path to the element to split on. These paths do not accept wildcards.

Tip

These paths do not accept wildcards.

For example, if you have a document that looks like this:

```
{ "first": "John", "last": "Doe", "grade": 8, "exams": [ { "subject":
"Maths", "test" : "term1", "marks":90}, { "subject": "Biology", "test"
: "term1", "marks":86} ] }
```

If we want to split this document on the `exams` element and create two documents, each with a different subject, we would define `"splitpath":"/exams"` in our datasource definition.

Tip

When using the Fusion UI to configure the datasource, enter the path *without* quotes.

The output from retrieving the document should look like this:

```
{ "first": "John", "last": "Doe", "grade": 8, "exams": [ { "subject":
"Maths", "test" : "term1", "marks":90 } ] }, { "first": "John",
"last": "Doe", "grade": 8, "exams": [ { "subject": "Biology", "test" :
"term1", "marks":86 } ] }
```

Split Couchbase Documents

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

The Couchbase V1 connector uses the Cross-Datacenter Replication (XDCR) feature of Couchbase to retrieve data stored in Couchbase continuously in real-time.

Because Couchbase has a flexible data model, documents may have a nested JSON structure. It is possible to split nested documents with a `splitpath` property, which uses an XPath-style path to the element to split on. These paths do not accept wildcards.

> **Tip**
>
> These paths do not accept wildcards.

For example, if you have a document that looks like this:

```
{ "first": "John", "last": "Doe", "grade": 8, "exams": [ { "subject":
"Maths", "test" : "term1", "marks":90}, { "subject": "Biology", "test"
: "term1", "marks":86} ] }
```

If we want to split this document on the `exams` element and create two documents, each with a different subject, we would define `"splitpath":"/exams"` in our datasource definition.

> **Tip**
>
> When using the Managed Fusion UI to configure the datasource, enter the path *without* quotes.

The output from retrieving the document should look like this:

```
{ "first": "John", "last": "Doe", "grade": 8, "exams": [ { "subject":
"Maths", "test" : "term1", "marks":90 } ] }, { "first": "John",
"last": "Doe", "grade": 8, "exams": [ { "subject": "Biology", "test" :
"term1", "marks":86 } ] }
```

SSL Security (Unix)

How-to

> Compatible versions: 4.2
>
> Works with: *Fusion Server*

Fusion's UI (which is accessed through the Fusion Proxy service) can run over SSL for secure communication with any HTTP client, using the Java Secure Socket Extension (JSSE) framework. You configure Fusion for SSL by configuring Fusion's Proxy and UI services.

Note

Fusion 4.0.2 or later is required to enable SSL security. Version 4.0.2 fixed a bug in the authentication proxy that incorrectly redirected some requests when SSL is enabled.

Required software

Configuring Fusion for SSL requires the following software:

- **Java Development Kit.** To store certificates, you can use the Java keytool Key and Certificate Management utility which is a part of the JDK. The JDK is also *a requirement for Fusion Server*.
- **OpenSSL**. You might need the `openssl` command line tool:
  - If you have the certificate chain and private key as separate files, then you can use the `openssl` command line tool to create a PKCS #12 file.
  - If you have an intermediate CA certificate, then you can use it and `openssl` to generate the certificate chain and private key files.

1. Load an SSL certificate into a Fusion keystore

The SSL protocol is based on public-key cryptography where encryption keys come in public key/private key pairs. An SSL certificate is used to verify the authenticity of a particular server. It contains the web site name, contact email address, company information and the public key used to encrypt the communication which is shared with the entities that communicate with the owner of the public/private key pair.

The server has a locally-protected private key that is accessible via a JSEE keystore.

The keystore maintains both the server certificate and the private key, so that when a server authenticates itself to the client, it uses the private key from its keystore for the initial SSL handshake.

Load the certificate into a Fusion keystore. Perform the tasks in the appropriate section:

- ***Self-signed certificate.*** If Fusion is behind a firewall, you can use a self-signed certificate for SSL communication with other hosts in your internal network. Create a keystore for the Fusion Proxy service and load the keystore with the self-signed PKCS #12 certificate.
- ***Certificate signed by a certificate authority.*** In a production environment, SSL certificates typically originate with certificate signing requests (CSRs) and are signed by a trusted third-party Certificate Authority (CA). Create a keystore for the Fusion Proxy service and load the keystore with the PKCS #12 certificate from a CA.

Alternative 1: Self-signed certificate

Note

If you are using a CSR-originated certificate from a trusted certificate authority, proceed to *Alternative 2: CA-signed certificate*.

If Fusion is behind a firewall, you can use a self-signed certificate for SSL communication with other hosts in your internal network. Create a keystore for the Fusion Proxy service and load the keystore with the self-signed PKCS #12 certificate.

To store certificates, you can use the Java keytool Key and Certificate Management utility which is a part of the JDK.

How to create a keystore and load a self-signed certificate

1. Set environment variables:

```
export JAVA_HOME=JavaHomeDirectory export
FUSION_HOME=FusionHomeDirectory
```

For example, in Fusion 4.1:

```
export JAVA_HOME=/usr/java/jdk1.8.0_171 export
FUSION_HOME=/opt/lucidworks/fusion/4.1.2
```

In Fusion 4.0.x:

```
export JAVA_HOME=/usr/java/jdk1.8.0_171 export
FUSION_HOME=/opt/lucidworks/fusion/4.0.2
```

2. Create the Fusion Proxy service keystore, generate the key pair and self-signed certificate, and load them into the keystore:

```
"$JAVA_HOME/bin/keytool" -genkeypair -keystore
"$FUSION_HOME/apps/jetty/proxy/etc/keystore" -dname
"CN=CommonName, OU=Unknown, O=Unknown, L=Unknown, ST=Unknown,
C=Unknown" -keypass KeyPassword -storepass KeystorePassword -
keyalg RSA -alias selfsigned -deststoretype pkcs12 -ext
SAN=dns:ServerFqdn,ip:ServerIpAddress
```

Important

You must include the qualified domain name and/or the IP address of the Fusion server in the `-ext SAN` part of the command. Failure to do so results in SSL validation errors.

Example command:

```
"$JAVA_HOME/bin/keytool" -genkeypair -keystore
"$FUSION_HOME/apps/jetty/proxy/etc/keystore" -dname
"CN=search.mycorp, OU=Unknown, O=Unknown, L=Unknown, ST=Unknown,
C=Unknown" -keypass 59Winter.Is.Long45 -storepass
46I.Prefer.Vanilla24 -keyalg RSA -alias selfsigned -
deststoretype pkcs12 -ext
SAN=dns:search.mycorp,ip:192.168.1.40,dns:localhost
```

The resulting certificate enables validated SSL transport to these hosts:

```
https://search.mycorp:ProxyPort https://192.168.1.40:ProxyPort
https://localhost:ProxyPort
```

Where `ProxyPort` is the Fusion Proxy port.

Alternative 2: CA-signed certificate

Note

If Fusion is behind a firewall and you are using a self-signed certificate, skip this section. Perform the tasks in *Alternative 1: Self-signed certificate*.

In a production environment, SSL certificates typically originate with certificate signing requests (CSRs) and are signed by a trusted third-party Certificate Authority (CA).

The steps here assume that you are the person who will be obtaining the SSL certificate chain and private key files. If you are not that person, contact your system administrator.

PRELIMINARY STEPS

1. Obtain a domain from a domain registrar.
2. Change the A record of your domain to the public IP address of your web server instance.

GENERATE SSL CERTIFICATE FILES

Use an SSL certificate provider to generate the certificate chain and private key files, or a PKCS #12 certificate, from a trusted CA:

- **Certificate chain and private key files.** In this case, you will need to convert these files into a single certificate file in PKCS #12 format.
- **A PKCS #12 certificate.** This must contain both the certificate chain and private key. In this case, no conversion is necessary.

How to obtain certificate files

1. In most cases, you will need to temporarily open ports 80 and 443 in your firewall configuration. The SSL certificate provider must be able to make successful HTTP and HTTPS requests to your server through the Domain Name System (DNS).
2. Use an SSL certificate provider to generate the certificate chain (`fullchain.pem`) and private key (`privkey.pem`) files, or the PKCS #12 certificate, from a trusted CA. Steps will vary based on the certificate provider. Contact your certificate provider for details.
3. Close ports 80 and 443 in your firewall configuration.
4. Change the A record of your domain to the public domain-name address of your web server instance.

Proceed to the next sections as follows:

- If you have certificate chain and private key files, perform the steps in *Convert the certificate chain and private key files to a PKCS #12 certificate* and *Import the PKCS #12 certificate into the Fusion Proxy service keystore*.
- If you have a PKCS #12 certificate, perform the steps in *Import the PKCS #12 certificate into the Fusion Proxy service keystore*.

CONVERT THE CERTIFICATE CHAIN AND PRIVATE KEY FILES TO A PKCS #12 CERTIFICATE

Note

If you have a PKCS #12 certificate, skip this section and proceed to the section *Import the PKCS #12 certificate into the Fusion Proxy service keystore*.

How to create a PKCS #12 certificate

With the certificate chain and private key as separate files, use the `openssl` command line tool in OpenSSL to create a PKCS #12 certificate.

```
openssl pkcs12 -export -out /path/to/keystore.p12 -in
/path/to/fullchain.pem -inkey /path/to/privkey.pem
```

Important

Do not enter a blank password.

CREATE THE FUSION PROXY SERVICE KEYSTORE AND IMPORT THE PKCS #12 CERTIFICATE

Use the Java keytool Key and Certificate Management utility to create a keystore for the Fusion Proxy service (`$FUSION_HOME/apps/jetty/proxy/etc/keystore`) and import the PKCS #12 certificate file. Fusion uses this certificate to perform SSL.

1. To create the keystore and import the PKCS #12 certificate:
2. Use the keytool `import` command to create a JSSE keystore.

```
keytool -importkeystore -srckeystore /path/to/keystore.p12 -
srcstoretype PKCS12 -destkeystore
"$FUSION_HOME/apps/jetty/proxy/etc/keystore" -deststoretype
PKCS12
```

3. (Optional) If desired, delete the PKCS #12 certificate file that resides outside of the Fusion Proxy service keystore (the one you created from the certificate chain and private key files, or obtained from a trusted CA.

```
rm /path/to/keystore.p12
```

2. Enable HTTPS in the Fusion Proxy service

Before beginning these steps, *load an SSL certificate into a Fusion keystore*.

How to enable HTTPS in the Fusion Proxy service:

1. (*Only for Fusion Server 4.0.x and 4.1.0*) Prevent the `start.jar` program from downloading a default keystore file, which is not needed. Edit `$FUSION_HOME/apps/jetty/home/modules/ssl.mod`. Comment out the indicated line using `#`. Change:

```
[files]
https://raw.githubusercontent.com/eclipse/jetty.project/master/j
etty-server/src/test/config/etc/keystore?
id=${jetty.tag.version}|etc/keystore
```

To:

```
[files] #
https://raw.githubusercontent.com/eclipse/jetty.project/master/j
etty-server/src/test/config/etc/keystore?
id=${jetty.tag.version}|etc/keystore
```

2. Set environment variables:

```
export JAVA_HOME=JavaHomeDirectory export
FUSION_HOME=FusionHomeDirectory
```

For example, in Fusion 4.1:

```
export JAVA_HOME=/usr/java/jdk1.8.0_171 export
FUSION_HOME=/opt/lucidworks/fusion/4.1.2
```

In Fusion 4.0:

```
export JAVA_HOME=/usr/java/jdk1.8.0_171 export
FUSION_HOME=/opt/lucidworks/fusion/4.0.2
```

3. Add HTTPS protocol support to the Jetty TLS (SSL) connector:

```
cd "$FUSION_HOME/apps/jetty/proxy/" java -jar
"$FUSION_HOME/apps/jetty/home/start.jar" --add-to-start=https
```

Example output:

```
INFO: ssl initialized (transitively) in ${jetty.base}/start.ini
INFO: https initialized in ${jetty.base}/start.ini INFO: Base
directory was modified
```

4. Get the obfuscated version of your keystore password:

```
java -cp "$FUSION_HOME/apps/jetty/home/lib/*"
org.eclipse.jetty.util.security.Password PASSWORD
```

Replace **PASSWORD** with the password you used for the keystore. If the password contains special characters, URL encode them .
Example output:

```
2018-05-15 12:32:48.988:INFO::main: Logging initialized @133ms
password345XYZ OBF:1vn21ugu1saj1v9i1v941sar1ugw1vo0
MD5:b91cd1a54781790beaa2baf741fa6789
```

5. Edit the file **$FUSION_HOME/apps/jetty/proxy/start.ini**:
   a. Include obfuscated passwords by adding these properties to the end of the file:

```
* `jetty.sslContext.keyStorePassword` *
`jetty.sslContext.keyManagerPassword` *
`jetty.sslContext.trustStorePassword`
```

   b. Use the OBF-encrypted password from step 4 (including the **OBF:** string) as the value for all three of the properties.
   For example:

```
## Keystore password
jetty.sslContext.keyStorePassword=OBF:2uha1vgt1jg01a4b1a4j
1jda1vg11ugg ... ## KeyManager password
jetty.sslContext.keyManagerPassword=OBF:2uha1vgt1jg01a4b1a
4j1jda1vg11ugg ## Truststore password
jetty.sslContext.trustStorePassword=OBF:2uha1vgt1jg01a4b1a
4j1jda1vg11ugg
```

   c. Set the local SSL port by adding the **jetty.ssl.port** property to the end of the file, and providing the port number. For example:

```
## Connector port to listen on jetty.ssl.port=8443
```

d. Save the file `$FUSION_HOME/apps/jetty/proxy/start.ini`.

3. Restart Fusion and test access through HTTPS

1. Restart all Fusion services:

```
./bin/fusion restart
```

HTTPS should now be enabled in the Fusion Proxy service.

2. Sign in to the Fusion UI. Specify the HTTPS URL scheme and SSL port, for example, `https://search.mycorp:8443`.

4. Disable HTTP access to the Fusion Proxy service

Disable HTTP access. You have a choice. Perform the tasks in the appropriate section:

- ***Disable HTTP access on the firewall or load balancer*** *- This is the preferred approach.*
- ***Disable listening for HTTP requests in the Fusion Proxy service***

Alternative 1: Disable HTTP access on the firewall or load balancer

Disable HTTP access to the Fusion Proxy service on the firewall or load balancer:

1. Disallow all requests for port 8764 from the outside world. Only `localhost` should be able to communicate with Fusion on the non-SSL port 8764. Block all other requestors.
2. If you are using a firewall or load balancer in front of Fusion, use it to redirect all HTTP requests to use HTTPS instead. For example, Apache would redirect all incoming HTTP traffic to HTTPS.

Alternative 2: Disable listening for HTTP requests in the Fusion Proxy service

Important

Ideally, you should *disable HTTP access using the firewall or load balancer*. Follow the steps in this section *only* if disabling HTTP access on the firewall or load balancer is not feasible.

You can only use this alternative if your SSL certificate covers a hostname that can be accessed from the local host. For example, if your certificate only covers `https://fusion.com`, then your local machine must be able to access Fusion from that exact host. If necessary, change the `hosts` file so that this can work.

How to disable HTTP

1. Edit `/opt/lucidworks/fusion/latest.x/apps/jetty/proxy/start.d/http.ini`.
   a. Change this line:

```
--module=http
```

To:

```
#--module=http
```

b. Save the file.

2. Edit the Fusion configuration file, `/opt/lucidworks/fusion/latest.x/conf/fusion.cors` (`fusion.properties` in Fusion 4.x).
   a. Ensure that the Agent JVM uses the Fusion Proxy service's keystore by adding this to the end of the file:

```
agent.jvmOptions=-
Djavax.net.ssl.trustStore="${FUSION_HOME}/apps/jetty/proxy
/etc/keystore" -Djavax.net.ssl.trustStorePassword=PASSWORD
-
Djavax.net.ssl.keyStore="${FUSION_HOME}/apps/jetty/proxy/e
tc/keystore" -Djavax.net.ssl.keyStorePassword=PASSWORD
```

   Replace `PASSWORD` with your Fusion keystore password.
   b. Uncomment the `default.address` and change it to the hostname of the server that is validated by your SSL certificate.
   If the hostname saved in `default.address` is not validated by your SSL certificate, then the Fusion Proxy service will not start, because the agent's liveness detector will not be able to access the HTTPS port to determine whether Fusion is running.

Important

If you self-signed the certificate, then the `default.address` must match the hostname you specified while signing the certificate. Failure to do this will result in the Fusion Proxy service not starting after you have disabled HTTP.

For example, if your SSL certificate's validated hostname is `search.mycorp`, then change:

```
#default.address = 127.0.0.1
```

To:

```
default.address = search.mycorp
```

c. Change the `proxy.port` to the SSL port you chose.
d. Uncomment `proxy.ssl` and change its value to `true`. Change:

```
# proxy.ssl=false
```

To:

```
proxy.ssl=true
```

Stopping and starting Fusion is needed for these changes to take effect. But there is a bit more to do concerning the Fusion Agent before restarting Fusion.

5. Ensure that the Fusion Agent can do heartbeats against the Proxy service

Important

The heartbeat service is deprecated in Fusion 4.2+. This section does not apply to Fusion 4.2+.

By default, the Fusion Agent makes HTTP requests to the Fusion Proxy service to ask whether the Proxy service is running (heartbeat checks). So, when you disable HTTP access to the Proxy service, the Fusion Agent needs an alternative way to do the heartbeat checks.

1. Add an entry to the `/etc/hosts` file for the Fusion node so that the DNS routes to the correct IP address:

```
192.168.1.6 search.mycorp
```

Or using a local IP address:

```
127.0.0.1 search.mycorp
```

2. Add `proxy.address` and specify the hostname of the server on which the Proxy service runs, for example:

```
proxy.address=mydomain
```

3. Stop and start all Fusion services (do not just perform a restart):

```
cd "$FUSION_HOME/bin" ./fusion stop ./fusion start
```

HTTPS should now be disabled in the Fusion Proxy service.

4. Verify that you cannot sign in to the Fusion UI using the HTTP URL scheme and non-SSL port, for example, `http://search.mycorp:8764`.

References and tutorials

- Transport Layer Security (Wikipedia)
- Public Key Certificate (Wikipedia)
- OpenSSL Cookbook (free ebook)
- OpenSSL Command Line Utilities (OpenSSL wiki)
- Java Tutorials: Generating and Verifying Certificates
- IBM developerWorks: What is the JSSE all about?

SSL Security (Windows)

How-to

> Compatible versions: 4.2
>
> Works with: *Fusion Server*

Fusion's UI (which is accessed through the Fusion Proxy service) can run over SSL for secure communication with any HTTP client, using the Java Secure Socket Extension (JSSE) framework. You configure Fusion for SSL by configuring Fusion's Proxy and UI services.

Note

Fusion 4.0.2 or later is required to enable SSL security. Version 4.0.2 fixed a bug in the authentication proxy that incorrectly redirected some requests when SSL is enabled.

Required software

Configuring Fusion for SSL requires the following software:

- **Java Development Kit.** To store certificates, you can use the Java keytool Key and Certificate Management utility which is a part of the JDK. The JDK is also *a requirement for Fusion Server*.
- **OpenSSL**. You might need the `openssl` command line tool:
  - If you have the certificate chain and private key as separate files, then you can use the `openssl` command line tool to create a PKCS #12 file.
  - If you have an intermediate CA certificate, then you can use it and `openssl` to generate the certificate chain and private key files.

Overview of procedure

How to configure Fusion for SSL:

1. *Load an SSL certificate into a Fusion keystore.*
2. *Enable SSL in the Fusion Proxy service.*
3. *Restart Fusion and test access through HTTPS.*
4. *Disable HTTP access to the Fusion Proxy service.*
5. *Ensure that the Fusion Agent can do heartbeats against the Proxy service.*

Note

The heartbeat service is deprecated in Fusion 4.2+.

1. Load an SSL certificate into a Fusion keystore

The SSL protocol is based on public-key cryptography where encryption keys come in public key/private key pairs. An SSL certificate is used to verify the authenticity of a particular server. It contains the web site name, contact email address, company information and the public key used to encrypt the communication which is shared with the entities that communicate with the owner of the public/private key pair.

The server has a locally-protected private key that is accessible via a JSEE keystore.

The keystore maintains both the server certificate and the private key, so that when a server authenticates itself to the client, it uses the private key from its keystore for the initial SSL handshake.

Load the certificate into a Fusion keystore. Perform the tasks in the appropriate section:

- ***Self-signed certificate.*** If Fusion is behind a firewall, you can use a self-signed certificate for SSL communication with other hosts in your internal network. Create a keystore for the Fusion Proxy service and load the keystore with the self-signed PKCS #12 certificate.
- ***Certificate signed by a certificate authority.*** In a production environment, SSL certificates typically originate with certificate signing requests (CSRs) and are signed by a trusted third-party Certificate Authority (CA). Create a keystore for the Fusion Proxy service and load the keystore with the PKCS #12 certificate from a CA.

Alternative 1: Self-signed certificate

Note

If you are using a CSR-originated certificate from a trusted certificate authority, proceed to *Alternative 2: CA-signed certificate*.

If Fusion is behind a firewall, you can use a self-signed certificate for SSL communication with other hosts in your internal network. Create a keystore for the Fusion Proxy service and load the keystore with the self-signed PKCS #12 certificate.

To store certificates, you can use the Java keytool Key and Certificate Management utility which is a part of the JDK.

How to create a keystore and load a self-signed certificate

1. Set environment variables:

```
set JAVA_HOME=JavaHomeDirectory set
FUSION_HOME=FusionHomeDirectory
```

2. Create the Fusion Proxy service keystore, generate the key pair and self-signed certificate, and load them into the keystore:

```
"%JAVA_HOME%\bin\keytool.exe" -genkeypair -keystore
"%FUSION_HOME%\apps\jetty\proxy\etc\keystore" -dname
"CN=CommonName, OU=Unknown, O=Unknown, L=Unknown, ST=Unknown,
C=Unknown" -keypass KeyPassword -storepass KeystorePassword -
keyalg RSA -alias selfsigned -deststoretype pkcs12 -ext
SAN=dns:ServerFqdn,ip:ServerIpAddress
```

Important

You must include the qualified domain name and/or the IP address of the Fusion server in the `-ext SAN` part of the command. Failure to do so results in SSL validation errors.

Example command:

```
"%JAVA_HOME%\bin\keytool.exe" -genkeypair -keystore
"%FUSION_HOME%\apps\jetty\proxy\etc\keystore" -dname
"CN=CommonName, OU=Unknown, O=Unknown, L=Unknown, ST=Unknown,
C=Unknown" -keypass 59Winter.Is.Long45 -storepass
46I.Prefer.Vanilla24 -keyalg RSA -alias selfsigned -
deststoretype pkcs12 -ext
SAN=dns:search.mycorp,ip:192.168.1.40,dns:localhost
```

The resulting certificate enables validated SSL transport to these hosts:

```
https://search.mycorp:ProxyPort https://192.168.1.40:ProxyPort
https://localhost:ProxyPort
```

Where `ProxyPort` is the Fusion Proxy port.

Alternative 2: CA-signed certificate

Note

If Fusion is behind a firewall and you are using a self-signed certificate, skip this section. Perform the tasks in *Alternative 1: Self-signed certificate*.

In a production environment, SSL certificates typically originate with certificate signing requests (CSRs) and are signed by a trusted third-party Certificate Authority (CA).

The steps here assume that you are the person who will be obtaining the SSL certificate chain and private key files. If you are not that person, contact your system administrator.

PRELIMINARY STEPS

1. Obtain a domain from a domain registrar.
2. Change the A record of your domain to the public IP address of your web server instance.

GENERATE SSL CERTIFICATE FILES

Use an SSL certificate provider to generate the certificate chain and private key files, or a PKCS #12 certificate, from a trusted CA:

- **Certificate chain and private key files.** In this case, you will need to convert these files into a single certificate file in PKCS #12 format.
- **A PKCS #12 certificate that contains both the certificate chain and private key.** In this case, no conversion is necessary.

How to obtain certificate files

1. In most cases, you will need to temporarily open ports 80 and 443 in your firewall configuration. The SSL certificate provider must be able to make successful HTTP and HTTPS requests to your server through the Domain Name System (DNS).
2. Use an SSL certificate provider to generate the certificate chain (`fullchain.pem`) and private key (`privkey.pem`) files, or the PKCS #12 certificate, from a trusted CA. Steps will vary based on the certificate provider. Contact your certificate provider for details.
3. Close ports 80 and 443 in your firewall configuration.
4. Change the A record of your domain to the public domain-name address of your web server instance.

Proceed to the next sections as follows:

- If you have certificate chain and private key files, perform the steps in *Convert the certificate chain and private key files to a PKCS #12 certificate* and *Import the PKCS #12 certificate into the Fusion Proxy service keystore*.
- If you have a PKCS #12 certificate, perform the steps in *Import the PKCS #12 certificate into the Fusion Proxy service keystore*.

CONVERT THE CERTIFICATE CHAIN AND PRIVATE KEY FILES TO A PKCS #12 CERTIFICATE

Note

If you have a PKCS #12 certificate, skip this section and proceed to the section *Import the PKCS #12 certificate into the Fusion Proxy service keystore*.

How to create a PKCS #12 certificate

With the certificate chain and private key as separate files, use the `openssl` command line tool in OpenSSL to create a PKCS #12 certificate.

```
openssl pkcs12 -export -out \path\to\keystore.p12 -in
\path\to\fullchain.pem -inkey \path\to\privkey.pem
```

Important

Do not enter a blank password.

CREATE THE FUSION PROXY SERVICE KEYSTORE AND IMPORT THE PKCS #12 CERTIFICATE

Use the Java keytool Key and Certificate Management utility to create a keystore for the Fusion Proxy service (`%FUSION_HOME%\apps\jetty\proxy\etc\keystore`) and import the PKCS #12 certificate file. Fusion uses this certificate to perform SSL.

Note

If you are starting with a certificate file in PFX format (file extension `.pfx`), that format is now identical to PKCS #12 format.

1. To create the keystore and import the PKCS #12 certificate:

```
"%JAVA_HOME%\bin\keytool.exe" –genkeypair –alias mykeystore –
keyalg RSA –keysize 2048 –keystore keystore.jks
"%JAVA_HOME%\bin\keytool.exe" –importkeystore –srckeystore
mycertificate.p12 –srcstoretype PKCS12 –destkeystore
keystore.jks
```

2. Use the keytool **import** command to create a JSSE keystore.

```
keytool –importkeystore –srckeystore \path\to\keystore.p12 –
srcstoretype PKCS12 –destkeystore
"%FUSION_HOME%\apps\jetty\proxy\etc\keystore" –deststoretype
PKCS12
```

3. (Optional) If desired, delete the PKCS #12 certificate file that resides outside of the Fusion Proxy service keystore (the one you created from the certificate chain and private key files, or obtained from a trusted CA.

```
del \path\to\keystore.p12
```

2. Enable HTTPS in the Fusion Proxy service

Before beginning these steps, *load an SSL certificate into a Fusion keystore*.

How to enable HTTPS in the Fusion Proxy service:

1. Set environment variables:

```
set JAVA_HOME=JavaHomeDirectory set
FUSION_HOME=FusionHomeDirectory
```

2. Add HTTPS protocol support to the Jetty TLS (SSL) connector:

```
cd "%FUSION_HOME%\apps\jetty\proxy\"
java -jar "%FUSION_HOME%\apps\jetty\home\start.jar" --add-to-start=https
```

Example output:

```
INFO: ssl initialized (transitively) in ${jetty.base}/start.ini
INFO: https initialized in ${jetty.base}/start.ini INFO: Base directory was modified
```

3. Get the obfuscated version of your keystore password:

```
java -cp "$FUSION_HOME/apps/jetty/home/lib/*"
org.eclipse.jetty.util.security.Password PASSWORD
```

Replace `PASSWORD` with the password you used for the keystore. If the password contains special characters, URL encode them .
Example output:

```
2018-05-15 12:32:48.988:INFO::main: Logging initialized @133ms
password345XYZ OBF:1vn21ugu1saj1v9i1v941sar1ugw1vo0
MD5:b91cd1a54781790beaa2baf741fa6789
```

4. Edit the file `%FUSION_HOME%\apps\jetty\proxy\start.ini` :
   a. Include obfuscated passwords by adding these properties to the end of the file:
      - `jetty.sslContext.keyStorePassword`
      - `jetty.sslContext.keyManagerPassword`
      - `jetty.sslContext.trustStorePassword`

   b. Use the OBF-encrypted password from step 4 (including the `OBF:` string) as the value for all three of the properties.
   For example:

```
## Keystore password
jetty.sslContext.keyStorePassword=OBF:2uha1vgt1jg01a4b1a4j
1jda1vg11ugg ... ## KeyManager password
jetty.sslContext.keyManagerPassword=OBF:2uha1vgt1jg01a4b1a
4j1jda1vg11ugg ## Truststore password
jetty.sslContext.trustStorePassword=OBF:2uha1vgt1jg01a4b1a
4j1jda1vg11ugg
```

c. Set the local SSL port by adding the `jetty.ssl.port` property to the end of the file, and providing the port number. For example:

```
## Connector port to listen on jetty.ssl.port=8443
```

d. Save the file `%FUSION_HOME%\apps\jetty\proxy\start.ini`.

3. Restart Fusion and test access through HTTPS

1. Restart all Fusion services:

```
bin\fusion.cmd restart
```

   HTTPS should now be enabled in the Fusion Proxy service.
2. Sign in to the Fusion UI. Specify the HTTPS URL scheme and SSL port, for example, `https://search.mycorp:8443`.

4. Disable HTTP access to the Fusion Proxy service

Disable HTTP access. You have a choice. Perform the tasks in the appropriate section:

- ***Disable HTTP access on the firewall or load balancer*** *- This is the preferred approach.*
- ***Disable listening for HTTP requests in the Fusion Proxy service***

Alternative 1: Disable HTTP access on the firewall or load balancer

Disable HTTP access to the Fusion Proxy service on the firewall or load balancer:

1. Disallow all requests for port 8764 from the outside world. Only `localhost` should be able to communicate with Fusion on the non-SSL port 8764. Block all other requestors.

2. If you are using a firewall or load balancer in front of Fusion, use it to redirect all HTTP requests to use HTTPS instead. For example, Apache would redirect all incoming HTTP traffic to HTTPS.

Alternative 2: Disable listening for HTTP requests in the Fusion Proxy service

Important

Ideally, you should *disable HTTP access using the firewall or load balancer*. Follow the steps in this section *only* if disabling HTTP access on the firewall or load balancer is not feasible.

You can only use this alternative if your SSL certificate covers a hostname that can be accessed from the local host. For example, if your certificate only covers `https://fusion.com`, then your local machine must be able to access Fusion from that exact host. If necessary, change the `hosts` file so that this can work.

How to disable HTTP

1. Edit `\lucidworks\fusion\latest.x\apps\jetty\proxy\start.d\http.ini`.
   a. Change this line:

```
--module=http
```

   To:

```
#--module=http
```

   b. Save the file.
2. Edit the Fusion configuration file, `\lucidworks\fusion\latest.x\conf\fusion.cors` (`fusion.properties` in Fusion 4.x).
   a. Ensure that the Agent JVM uses the Fusion Proxy service's keystore by adding this to the end of the file:

```
agent.jvmOptions=-Djavax.net.ssl.trustStore="%
{FUSION_HOME}%\\apps\\jetty\\proxy\\etc\\keystore" -
Djavax.net.ssl.trustStorePassword=PASSWORD -
Djavax.net.ssl.keyStore="%
{FUSION_HOME}%\\apps\\jetty\\proxy\\etc\\keystore" -
Djavax.net.ssl.keyStorePassword=PASSWORD
```

   Replace `PASSWORD` with your Fusion keystore password.

b. Uncomment the `default.address` and change it to the hostname of the server that is validated by your SSL certificate.
If the hostname saved in `default.address` is not validated by your SSL certificate, then the Fusion Proxy service will not start, because the agent's liveness detector will not be able to access the HTTPS port to determine whether Fusion is running.

Important

If you self-signed the certificate, then the `default.address` must match the hostname you specified while signing the certificate. Failure to do this will result in the Fusion Proxy service not starting after you have disabled HTTP.

For example, if your SSL certificate's validated hostname is `search.mycorp`, then change:

```
#default.address = 127.0.0.1
```

To:

```
default.address = search.mycorp
```

c. Change the `proxy.port` to the SSL port you chose. For example, change:

```
proxy.port = 8764
```

To:

```
proxy.port = 8443
```

d. Uncomment `proxy.ssl` and change its value to `true`. Change:

```
# proxy.ssl=false
```

To:

```
proxy.ssl=true
```

Stop and start Fusion for these changes to take effect.

References and tutorials

- Transport Layer Security (Wikipedia)
- Public Key Certificate (Wikipedia)
- OpenSSL Cookbook (free ebook)
- OpenSSL Command Line Utilities (OpenSSL wiki)
- Java Tutorials: Generating and Verifying Certificates
- IBM developerWorks: What is the JSSE all about?

Start App Studio

How-to

> Compatible versions: 4.2
>
> Works with: *App Studio*

Version 4.1 and later

App Studio has two scripts for startup and shutdown in the root directory of an App Studio project (`app-studio`):

- `app-studio.sh` . A Bash script for Linux and macOS
- `app-studio.bat` . A batch file for Windows

You also use these scripts to *build an app from source*.

Obtain usage information about the scripts

For usage information, enter:

- Linux and macOS: `./app-studio.sh -help`
- Windows: `app-studio.bat -help`

For command-specific usage information, enter:

- Linux and macOS: `./app-studio command -help`
- Windows: `app-studio.bat command -help`

Below, optional parameters are indicated by square brackets, which you do not type.

Startup commands

- Linux and macOS: `./app-studio start [--production] [-f] [-p port] [-m memory] [-t timeout] [-V]`
- Windows: `app-studio.bat start [-f] [-p port] [-m memory] [-t timeout] [-V]`

Parameters

Bold italic text is a placeholder for a value. For example, for `-m memory` you might use `-m 4g` .

Parameter	Description
`--production`	Start the app in production mode, which disables the *configuration wizard* and the *code editor*.
`-f`	Start App Studio in the foreground. By default, App Studio starts in the background.
`-p port`	Specify the port on which the App Studio web server will listen. The default port is 8080.
`-m memory`	Set the minimum (-Xms) and maximum (-Xmx) heap size for the JVM. For example, `-m 4g` results in: `-Xms4g -Xmx4g` . The default heap size is `512m` .
`-t timeout`	Set the startup timeout in seconds.
`-V`	Output verbose messages from the script.

Starting multiple App Studio instances

You can run multiple instances of App Studio on a single host by specifying different ports for all instances. Do this on startup, at the command line, using the `-p` parameter, for example:

```
./app-studio.sh start -p 9000
```

Version 4.0

Before starting App Studio, verify that Fusion Server is running and that it has one or more collections of indexed data.

Run App Studio at the command line from the `app-studio` directory, like this:

	Unix	Windows
Start the server	`./app-studio.sh start` Options: • `--production` Run the server in production mode. • `--extraJvmArguments` Space character-delimited additional JVM arguments; multiple arguments must be quoted. Example: `"-Xms2G -Xmx5G"` • `-p` or `--port` The port on which to run the server. • `-s` or `--stop-port` The port to use when issuing the `stop` command; it must be re-specified when running `stop`.	`app-studio-start.bat`
Stop the server	`./app-studio.sh stop` Optionally: • `-s` or `--stop-port` The port to use when issuing the `stop` command.	`app-studio-standalone-stop.bat`
Restart the server	`./app-studio.sh restart`	`app-studio-stop.bat` `app-studio-start.bat`

Starting in production mode

App Studio's *standalone mode* can be used for small production deployments, by starting in production mode.

Production mode disables the *configuration wizard* and the *code editor*.

To use standalone mode in a production environment, start App Studio with the `--production` flag:

```
./app-studio-standalone.sh start --production
```

What is next

- See *Create a New Web App* to learn how to create your first search app.

Start or Stop Fusion

How-to

> Compatible versions: 4.2
>
> Works with: *Fusion Server*

This topic explains how to start and stop Fusion Server and its services using the scripts in the `bin` directory below the Fusion home directory:

- `/opt/fusion/latest.x/bin` (Unix)
- `C:\lucidworks\fusion\latest.x\bin` (Windows)

Command summary

You can control all Fusion services at once under the management of the Fusion agent, or you can control services individually.

How to control all services using the Fusion agent:

- **Unix.** `/opt/fusion/latest.x/bin/fusion <command>`
- **Windows.** `C:\lucidworks\fusion\latest.x\bin\fusion.cmd <command>`

How to control individual services:

- **Unix.** `/opt/fusion/latest.x/bin/<servicename> <command>`
  For example: `/opt/fusion/latest.x/bin/proxy restart`
- **Windows.** `C:\lucidworks\fusion\latest.x\bin\<servicename>.cmd <command>`
  For example: `C:\lucidworks\fusion\latest.x\bin\proxy.cmd restart`

Tip

When starting services individually, start Zookeeper first.

The commands below can be issued to the `fusion` / `fusion.cmd` script to issue the command to all services in the correct sequence, or they can be issued to an individual service.

`start`	Start one or all Fusion services.
`status`	Display the status of one or all Fusion services.
`restart`	Restart one or all Fusion services.
`stop`	Stop one or all Fusion services.
`run`	Start one or all Fusion services in the foreground.
`run-in-shell` (Unix only) *(Fusion 4.2+ only.)*	Start an individual service using Bash's `exec` function, which allows the service to assume the shell process's PID. See *Run Fusion in shell mode* below.

Define groups of services

The `fusion.cors` (`fusion.properties` in Fusion 4.x) file includes the property definition `group.default`. This property defines the Fusion services to start and stop by default (if no property is named in the start or stop command).

The default list of services out-of-the-box is also the *minimum* set of services, with the exception of the `log-shipper` service, which you can remove if you do not use it.

Here is the `group.default` definition in `fusion.cors` (`fusion.properties` in Fusion 4.x):

In Fusion 4.1+.

```
group.default = zookeeper, solr, api, connectors-classic, connectors-rpc, proxy, webapps, admin-ui, log-shipper
```

With the exception of the `log-shipper` service, these are all required services. Even if only using RPC connectors, the `connectors-classic` service is required. The `log-shipper` service is required to use the Log Shipper.

In Fusion 4.0.x.

```
group.default = zookeeper, solr, api, connectors-rpc, connectors-classic, admin-ui, proxy, webapps
```

Even if only using RPC connectors, the `connectors-classic` service is required.

How to modify the default list of services

In Fusion 4.1+.

Edit the `group.default` property, for example, to include Spark and SQL related services:

```
group.default = zookeeper, solr, api, connectors-classic, connectors-rpc, proxy, webapps, admin-ui, log-shipper, spark-master, spark-worker, sql
```

In Fusion 4.0.x.

Edit the `group.default` property, for example, to include Spark related services:

```
group.default = zookeeper, solr, api, connectors-classic, connectors-rpc, proxy, webapps, admin-ui, spark-master, spark-worker
```

How to define other lists of services (Unix)

You can define other lists of services by defining other `group` properties.

In Fusion 4.1+.

For example, define this group property to start and stop services for Spark and SQL together:

```
group.spark-only = spark-master, spark-worker, sql
```

In Fusion 4.0.x.

For example, define this group property to start and stop services for Spark together:

```
group.spark-only = spark-master, spark-worker
```

Define this group property to start and stop services for classic and RPC connectors together:

```
group.connectors = connectors-classic, connectors-rpc
```

Unix

Start and stop Fusion on Unix.

Start Fusion

All Fusion start scripts must be executed by a user who has permissions to read and write to the directories where Fusion is installed. These scripts do not need to be run as root (or sudo), nor should they be. Use a suitable user, or create a new one, and then ensure that it owns the directory where Fusion resides, (for example, `C:\lucidworks`).

Give the commands that follow from the directory `fusion/latest.x/bin`.

START REQUIRED SERVICES

Start the required services that are defined in the `group.default` property.

How to start all required services

`./fusion start`

Tip

This is equivalent to `./fusion start default`. You can omit the group name `default`.

START A GROUP OF SERVICES

You can start a group of services together. Reference the property in `fusion.cors` (`fusion.properties` in Fusion 4.x) that *defines the group*.

Examples of when this is useful are:

- **Spark and SQL.** The `spark-master`, `spark-worker`, and `sql` services are interdependent and should be started and stopped together.

```
./fusion start spark-master spark-worker sql
```

- **Classic and RPC connectors.** RPC connectors require both the `connectors-classic` and `connectors-rpc` services to be running.

```
./fusion start connectors-classic connectors-rpc
```

START SERVICES INDIVIDUALLY

You can start services individually.

How to start services individually

- Fusion UI service: `./admin-ui start`
- API services: `./api start`
- Classic Connectors services: `./connectors-classic start`
- RPC Connectors services: `./connectors-rpc start`
- Log shipper service (Fusion 4.1+ only): `./log-shipper start`
- Proxy: `./proxy start`
- Solr: `./solr start`
- Spark Master: `./spark-master start`
- Spark Worker: `./spark-worker start`
- SQL service: `./sql start`
- Web Apps: `./webapps start`
- ZooKeeper: `./zookeeper start`

For information about default ports, see *Default Ports*.

RUN FUSION IN THE FOREGROUND

To run Fusion or any of its services in the foreground, use the `run` command-line argument in place of `start`.

RUN FUSION IN SHELL MODE

Important

This section applies to Fusion 4.2+ only.

To start any of Fusion's services using Bash's `exec` function, which allows the service to assume the shell process's PID, use the `run-in-shell` command-line argument in place of `start` or `run`. The `run-in-shell` argument can only be used to start one service at a time.

Examples

```
./fusion run-in-shell zookeeper
```

or

```
./zookeeper run-in-shell
```

Shell mode is particularly useful in containerized environments, which generally assume that only one process runs per container and that process is "process 0", that is, the initial process invoked within the container, not a separate spawned process.

Stop Fusion

How to stop Fusion services

To stop Fusion or any of its services, use the command above with the `stop` command-line argument in place of `start`, for example:

`./solr stop`

Using systemd to manage processes

On Red Hat Enterprise Linux, CentOS 7 and newer, and Ubuntu 15.04 LTS and newer, we support using the operating system-provided `systemd` for process management.

LAUNCHING FUSION AT SYSTEM START

You can configure `systemd` to launch Fusion when your system starts.

How to launch Fusion at system start:

1. Change your working directory to Fusion's `systemd` directory, for example:

   ```
   cd /opt/fusion/latest.x/init/systemd
   ```

2. Edit `fusion.service` to provide correct values for the `FUSION_HOME` and `JAVA_HOME` environment variables.

3. Stop Fusion if it is already running:

```
/opt/fusion/latest.x/bin/fusion stop
```

4. Create the `systemd` management file, which launches Fusion under `systemd` management:

```
sudo bash install.sh
```

STARTING AND STOPPING FUSION

You can use the `systemctl` command to start and stop Fusion:

```
sudo systemctl start fusion sudo systemctl stop fusion
```

Log files for Fusion services are found in directories under `https://FUSION_HOST:FUSION_PORT/var/log`.

Using Ubuntu Upstart to manage processes

Under Ubuntu 12.04 LTS through Ubuntu 14.10, we support using Upstart for process management. This requires Fusion to be installed in the `/opt/lucidworks/` directory.

To configure upstart, run the following commands:

```
$ cd /opt/lucidworks/fusion/latest/init/upstart $ sudo bash
install.sh
```

If this complains with `no JAVA_HOME set`, replace `sudo` with `sudo -E`. Then you can use the `service` command to control the server:

```
$ sudo service fusion-solr start $ sudo service fusion-api start $
sudo service fusion-connectors start $ sudo service fusion-ui start
```

and similarly use `stop` and `status`.

Log files for Fusion services are found in directories under `https://FUSION_HOST:FUSION_PORT/var/log`.

Windows

Start and stop Fusion on Windows.

Start Fusion

All Fusion start scripts must be executed by a user who has permissions to read and write to the directories where Fusion is installed. Ensure that the user owns the directory where Fusion resides (for example, `C:\lucidworks`).

Give the commands that follow from the directory `fusion\latest.x\bin`.

START REQUIRED SERVICES

How to start all required Fusion services as Java processes

```
fusion.cmd start
```

How to start all required Fusion services as Windows services

```
start-services.cmd
```

START SERVICES INDIVIDUALLY

How to start specific services as Java processes

- UI service:
  `admin-ui.cmd start`
- API services:
  `api.cmd start`
- Classic Connectors services:
  `connectors-classic.cmd start`
- RPC Connectors services:
  `connectors-rpc.cmd start`
- Log shipper service (Fusion 4.1+ only):
  `log-shipper.cmd start`
- Proxy:
  `proxy.cmd start`
- Solr:
  `solr.cmd start`
- Spark Master:
  `spark-master.cmd start`
- Spark Worker:
  `spark-worker.cmd start`
- SQL service:
  `sql.cmd start`
- Web Apps:
  `webapps.cmd start`
- ZooKeeper:
  `zookeeper.cmd start`

For information about default ports, see *Default Ports*.

Run Fusion in the foreground

To run Fusion or any of its services in the foreground, use the `run` command-line argument in place of `start`, for example:

```
connectors.cmd run
```

Stop Fusion

How to stop all Fusion services

- `fusion.cmd stop` (Stop all Fusion services, if they are running as Java processes)
- `stop-services.cmd` (Stop all Fusion services, if they are running as Windows services)

How to stop a specific service running as a Java process

To stop a specific Fusion service that is running as a Java process, use the command above with the `stop` command-line argument in place of `start`, for example:

```
connectors.cmd stop
```

Run Fusion with a service account

This example assumes the following:

Field	Value
Account	FUSION_SVC
Domain	qe
Installation directory	C:\fusion\<version>
Server	EC2AMAZ-79FD9JL

1. As an administrator, create the service account, and install it to the server you want to use for Fusion:

```
C:\Users\Administrator>New-ADServiceAccount -Name "FUSION_SVC" -
RestrictToSingleComputer C:\Users\Administrator>Add-
ADComputerServiceAccount -Identity EC2AMAZ-79FD9JL -
ServiceAccount "FUSION_SVC" C:\Users\Administrator>Install-
ADServiceAccount -Identity "FUSION_SVC"
C:\Users\Administrator>Test-ADServiceAccount "FUSION_SVC"
C:\Users\Administrator>Get-ADServiceAccount "FUSION_SVC"
```

2. Run `install-services.cmd` as a local administrator:

```
C:\Users\Administrator> C:\fusion\4.2.2\bin\install-services.cmd
ECHO is off. Thank you for choosing
================================================== "  _
_     _                    _         " " | |          (_)   | |
| |        " " | |_   _  ___ _  __| |_     _____  _ __| | _____
" " | | | | |/ __| |/ _` \ \ /\ / / _ \| '__| |/ / __| " " | |
|_| | (__| | (_| |\ V  V / (_) | |  |   <\__ \ " "
|_|\__._|\___|_|\__._| \_/\_/ \___/|_|  |_|\_\___/ "
================================================== You will
now be prompted for the username and password of the Windows
account that will launch Fusion. IMPORTANT NOTE 1: When prompted
'Set Account rights to allow log on as a service', enter 'Y'
IMPORTANT NOTE 2: You must enter the username in domain\username
format. .... Starting winsw (https://github.com/kohsuke/winsw)
service wrapper utility ... 2019-06-12 17:37:21,737 INFO -
Starting ServiceWrapper in the CLI mode Username: EC2AMAZ-
79FD9JL\Administrator Password: ********** Set Account rights
to allow log on as a service (y/n)?: n 2019-06-12 17:38:02,970
INFO - Completed. Exit code is 0
```

3. Grant the service account full access to Fusion:

```
C:\Users\Administrator>$path = "C:\fusion" $Acl = Get-Acl
"$path" $permission = "qe\FUSION_SVC$", "FullControl",
"ContainerInherit,ObjectInherit", "None", "Allow" $accessRule =
New-Object System.Security.AccessControl.FileSystemAccessRule
$permission $Acl.SetAccessRule($accessRule) Set-Acl "$path" $Acl
Get-Acl "$path" | fl
```

4. Modify the **Lucidworks Fusion** service to use the service account.
    a. Open **Administrative Tools > Services** on your Windows server.
    b. Select the **Lucidworks Fusion** service.
    c. Open the **Properties > Log On** dialog.
    d. Change the service user account to the `FUSION_SVC` user account.

> **Note**
>
> Only the account name is required. The password is managed by Windows.

Access Fusion after startup

After Fusion services have started, you can open the Fusion UI in a web browser at `http://localhost:8764/` (replace `localhost` with your server name or IP address if necessary).

The first time you access Fusion, you must set the password for the user `admin` and agree to the Fusion Licensing Agreement (which contains terms of service). This is followed by an optional registration step. After this, Fusion displays the Fusion launcher (the page from which you can open apps).

If you cannot access the system, see the *Troubleshooting topic*. *Checking System State* shows how to inspect Fusion services.

Start or Stop the Fusion 5x SQL Service

How-to

> Compatible versions: 5.5
>
> Works with: *Fusion*

Start the Fusion 5.x SQL service

The Fusion 5.x SQL service is an optional service that must be started manually.

When you install a new namespace, the `setup_f5_gke.sh` script generates a "values" yaml that contains Fusion values for the **sql-service > enabled** and **sql-service > replicaCount** fields. If necessary, modify the file:

1. Enter **true** in the **sql-service > enabled** field.
2. Enter the appropriate value in the **sql-service > replicaCount** field.
3. Save the `<provider>_<cluster>_<namespace>_fusion_values.yaml` file and run the generated shell script `<provider>_<cluster>_<namespace>_upgrade_fusion.sh` to apply the changes and start the Fusion SQL service.

Note

For more information, see the section about how to *Create a custom values YAML file*.

The following is an example snippet of the fields to start the Fusion 5.x SQL service:

```
sql-service: enabled: true logstashEnabled: true nodeSelector:
cloud.google.com/gke-nodepool: default-pool replicaCount: 1 service:
thrift: type: "ClusterIP"
```

Stop the Fusion SQL service

The Fusion 5.x SQL service is an optional service that must be stopped manually.

Edit the `fusion-values.yaml` file:

1. Enter **false** in the **sql-service > enabled** field.
2. Enter **0** in the **sql-service > replicaCount** field.
3. Save the `<provider>_<cluster>_<namespace>_fusion_values.yaml` file and run the generated shell script `<provider>_<cluster>_<namespace>_upgrade_fusion.sh` to apply the changes and stop the Fusion SQL service.

The following is an example snippet of the fields to stop the Fusion 5.x SQL service:

```
sql-service: enabled: false logstashEnabled: false nodeSelector:
cloud.google.com/gke-nodepool: default-pool replicaCount: 0 service:
thrift: type: "ClusterIP"
```

Start or Stop the Fusion SQL Service

How-to

> Compatible versions: 4.2
>
> Works with: *Fusion Server*

Starting the Fusion SQL service

The Fusion SQL service is an optional service that must be started manually.

Give these commands from the `bin` directory below the Fusion home directory, for example, `/opt/fusion/latest.x` (on Unix) or `C:\lucidworks\fusion\latest.x` (on Windows).

On Unix:

When starting the Fusion SQL service, the best practice is to also start the Spark master and Spark worker services:

```
./fusion start spark-master spark-worker sql
```

On Windows:

When starting the Fusion SQL service, the best practice is to also start the Spark master and Spark worker services:

```
spark-master.cmd start spark-worker.cmd start sql.cmd start
```

Stopping the Fusion SQL service

The Fusion SQL service is an optional service that must be stopped manually.

Give these commands from the `bin` directory below the Fusion home directory, for example, `/opt/fusion/latest.x` (on Unix) or `C:\lucidworks\fusion\latest.x` (on Windows).

On Unix:

When stopping the Fusion SQL service, the best practice is to also stop the Spark master and Spark worker services:

```
./fusion stop sql spark-worker spark-master
```

On Windows:

When starting the Fusion SQL service, the best practice is to also start the Spark master and Spark worker services:

```
sql.cmd stop spark-worker.cmd stop spark-master.cmd stop
```

Updating the group.default definition

If you plan to run the Fusion SQL service for production, we recommend updating the `group.default` definition in the file `conf/fusion.cors` (`fusion.properties` in Fusion 4.x) (on Unix) or `conf\fusion.cors`

(`fusion.properties` in Fusion 4.x) (on Windows) to include the `spark-master`, `spark-worker`, and `sql` services:

```
group.default = zookeeper, solr, api, connectors-classic, connectors-rpc, proxy, webapps, admin-ui, log-shipper, spark-master, spark-worker, sql
```

Verifying that the Fusion SQL service started

Verify the Fusion SQL service application started. Give these commands from the `bin` directory below the Fusion home directory.

On Unix:

```
./sql status
```

On Windows:

```
sql.cmd status
```

Alternatively, check the Spark UI, for example:

Running Applications

Application ID	Name	Cores	Memory per Node	Submitted Time	User	State
app-20161007115910-0007 (kill)	FusionSQLEngine-20161007175908	4	2.0 GB	2016/10/07 11:59:10	timpotter	RUNNING

Start the Basic Connector Client

How-to

> Compatible versions: 4.2, 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion Connectors*, *Fusion Server*, *Fusion*

To start a connector client, on the remote node (for example, the datasource), do the following:

1. Copy the connector uberjar from Fusion Server onto the remote node. The connector uberjar is at the following location:

```
FUSION_HOME/apps/connectors/connectors-rpc/client/connector-plugin-client-{fusionVersion}-uberjar.jar
```

2. On the remote node, run:

```
java -jar path/to/uberjar/connector-plugin-client-{fusionVersion}-uberjar.jar path/to/connector/file.zip
```

Step-by-step Query Workbench workflow

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

1. Use the *Index Workbench* to set up datasource(s).
2. Run a crawl on the data.
3. Open the Query Workbench.
4. Select a query pipeline to use or create a new, blank pipeline.

   Note

   The Default pipeline is created automatically and can be used as-is or with modification.

5. Modify the pipeline stages and set the order in which the queries will run.
6. Examine the live results in the Search panel.
7. Modify any pertinent stage settings to fine tune your results.
8. Click **Save** to commit the new settings to the pipeline

After a custom Query Pipeline configuration is saved, it becomes available throughout Managed Fusion and can be re-used and modified.

Step-by-step Query Workbench Workflow

How-to

> Compatible versions: 4.2, 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion Server*, *Fusion*

1. Use the *Index Workbench* to set up datasource(s).
2. Run a crawl on the data.
3. Open the Query Workbench.
4. Select a query pipeline to use or create a new, blank pipeline (the Default pipeline is created automatically and can be used as-is or with modification).
5. Modify the pipeline stages and set the order in which the queries will run.
6. Examine the live results in the Search panel.
7. Modify any pertinent stage settings to fine tune your results.
8. Click **Save** to commit the new settings to the pipeline

After a custom Query Pipeline configuration is saved, it becomes available throughout Fusion and can be re-used and modified.

Subscribing to Events

How-to

> Compatible versions: 4.2, 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *App Studio*

For communication between separate system components, Appkit uses an *event bus* architecture. The event bus handles local events, within the same virtual machine. The event model is quite flexible and allows for any POJO to be transmitted on the event bus. Our current implemention uses Google Guava as the event bus provider.

Sending event notifications

To post an event notification onto the event bus, you must obtain a copy of the global event bus and submit an instance of the class that describes the event. In this example, we assume that a Java class named `MyEvent` captures the event details.

```
import com.google.common.eventbus.EventBus; import
com.google.inject.Inject; @Inject EventBus eventBus; ...
eventBus.post(new MyEvent());
```

Subscribing to events

To subscribe for an event of a particular type, you create a new subscriber class (or use an anonymous class) that has a single-argument method annotated with the `@Subscribe` annotation.

```
import eventBus.register.Subscribe; public class MyEventSubscriber {
@Subscribe public void newEvent(MyEvent event) { // implement your
business logic here } }
```

Next, you must register the subscriber on the event bus, so that events of the given type are routed correctly. This is typically done in your application module or controller.

```
import com.google.common.eventbus.EventBus; import
com.google.inject.Inject; @Inject EventBus eventBus; ...
eventBus.register(new MyEventSubscriber());
```

Standard event types

This types of event are built into the AppKit core framework. You can register your own subscriber to receive notifications of each. See details in our JavaDocs on which attributes are available in each case.

- `twigkit.security.event.LoginEvent` : User logs into the system.
- `twigkit.event.ResponseEvent` : A search response was retrieved from a data provider.

In addition, this events are available if activity tracking is enabled:

- `twigkit.event.QueryEvent` : A search query was formulated and submitted to a data provider.
- `twigkit.event.ClickEvent` : User clicked a link on a page, possibly linked to a search query.

Support for distributed groups claims with the OpenID Connect realm in Azure

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

When a user is assigned to more than 200 groups, Microsoft Azure sends the link to the Microsoft Graph API in its token response instead of listing the user's groups. Managed Fusion must make a custom call to Azure services in order to download the user's groups. See OpenID Connect Core for more information.

To enable this feature in the OpenID Connect realm, add an additional scope to the configuration in Managed Fusion and then add app permissions in Azure.

Add the scope in Managed Fusion

1. Sign in to Managed Fusion and click your application.
2. Click **System > Access Control > Security Realms**.
3. Click the existing OpenID Connect realm.
   You can also create a new OpenID Connect Realm with **Add Security Realm**.
4. In the Scope section, add `https://graph.windows.net/Directory.Read.All`.
5. Click **Save**.

Add app permissions in Azure

1. Navigate to **App registrations > Your App Name**.
2. In View API permissions, click **+ Add a permission**.
3. In the list of Microsoft API options, click **Microsoft Graph**.
4. Click **Application permissions**.
5. Select all permissions in the following drop-downs:
   a. **AccessReview**
   b. **AgreementAcceptance**
   c. **Agreement**
   d. **AppCatalog**

Example configuration

Tip

All configuration settings can be found in the OpenID Connect metadata document, which is located in **App registrations > Your App Name > Endpoints** in the Azure overview. See Fetch the OpenID Connect metadata document for more information.

```
{ "realmType": "oidc", "name": "azure-realm", "enabled": true,
"config": { "autoCreateUsers": true, "code": { "clientSecret":
"XXXXXXX", "redirectUri":
"https://EXAMPLE_COMPANY.b.lucidworks.cloud:6764/",
"authorizationUri": "https://login.microsoftonline.com/XXXXXX-XXXXXX-
XXXXX-XXXXXX/oauth2/v2.0/authorize", "tokenUri":
"https://login.microsoftonline.com/XXXXXX-XXXXXX-XXXXX-
XXXXXX/oauth2/v2.0/token" }, "clientId": "XXXXXX-XXXXXX-XXXXX-XXXXXX",
"jwkSetUri": "https://login.microsoftonline.com/XXXXXX-XXXXXX-XXXXX-
XXXXXX/discovery/v2.0/keys", "userIdAttribute": "preferred_username",
"scope": [ "openid", "email", "profile",
"https://graph.windows.net/Directory.Read.All" ] } }
```

Support for Distributed Groups Claims with the OpenID Connect Realm in Azure

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion Connectors*, *Fusion*

When a user is assigned to more than 200 groups, Microsoft Azure sends the link to the Microsoft Graph API in its token response instead of listing the user's groups. Fusion must make a custom call to Azure services in order to download the user's groups. See OpenID Connect Core for more information.

You can enable this feature in the OpenID Connect realm by adding an additional scope to the configuration in Fusion and adding app permissions in Azure.

Add the scope in Fusion

1. Navigate to **System** > **Access Control** > **Security Realms**, then click the existing OpenID Connect realm. You can also create a new OpenID Connect Realm with **Add Security Realm**.
2. In the **Scope** section, add **https://graph.windows.net/Directory.Read.All**.
3. Click **Save**.

Add app permissions in Azure

1. Navigate to **App registrations** > **Your App Name** >, then scroll down to **View API permissions**
2. Click **+ Add a permission**, then choose **Microsoft Graph** from the list of Microsoft API options.
3. Click **Application permissions**.
4. Select all permissions in the following drop-downs:
    a. **AccessReview**
    b. **AgreementAcceptance**
    c. **Agreement**
    d. **AppCatalog**

Example configuration

Tip

All configuration settings can be found in the **OpenID Connect metadata document**, which is located in **App registrations** > **Your App Name** > **Endpoints** in the Azure overview. See Fetch the OpenID Connect metadata document for more information.

```
{ "realmType": "oidc", "name": "azure-realm", "enabled": true,
"config": { "autoCreateUsers": true, "code": { "clientSecret":
"XXXXXXX", "redirectUri": "http://url-to-fusion-gateway:6764/",
"authorizationUri": "https://login.microsoftonline.com/XXXXXX-XXXXXX-
XXXXX-XXXXXX/oauth2/v2.0/authorize", "tokenUri":
"https://login.microsoftonline.com/XXXXXX-XXXXXX-XXXXX-
XXXXXX/oauth2/v2.0/token" }, "clientId": "XXXXXX-XXXXXX-XXXXX-XXXXXX",
"jwkSetUri": "https://login.microsoftonline.com/XXXXXX-XXXXXX-XXXXX-
XXXXXX/discovery/v2.0/keys", "userIdAttribute": "preferred_username",
"scope": [ "openid", "email", "profile",
"https://graph.windows.net/Directory.Read.All" ] } }
```

Tracking Fusion Signals

How-to

> Compatible versions: 4.2, 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *App Studio*

Appkit provides support for the *Fusion REST Signals API*, allowing user clicks and the creation of annotations (bookmarks, topics etc) to be recorded and tracked.

Incorporate signals

This section describes how to incorporate Fusion Signals activity tracking into your Appkit application.

1. Add the Fusion Signals dependency

Signals tracking in Fusion requires the Fusion Signals module. To enable this module, insert these within the `dependencies` tag of your `pom.xml`:

```
<dependency> <groupId>twigkit</groupId>
<artifactId>twigkit.activity.tracker</artifactId>
<version>${project.parent.version}</version> </dependency>
<dependency> <groupId>twigkit</groupId>
<artifactId>twigkit.message.service.fusion.producer</artifactId>
<version>${project.parent.version}</version> </dependency>
```

2. Add fusion.conf to activity tracking resources

To access this module when Appkit starts up, create a `fusion.conf` file in `resources/conf/message/service/fusion.conf`. In that file, configure the `query-profile`. This is the REST-API endpoint that will ingest signal data into a signals collection. For example:

```
query-profile: my-query-profile signals-index-pipeline:
_signals_ingest commit: true async: true
```

Fusion Signals config

Here, `image_catalog` is the name of a primary collection that will be used to generate an auxiliary collection consisting of activity tracking data.

When accessing Fusion using a service account, the credentials will be pulled from the configuration given in the services/api/fusion.conf file, which should contain these parameters to enable basic authentication:

```
impersonate: true userName: joebloggs password: Enc(<encoded
password>)
```

For more information about how authentication is used when accessing the Fusion Signals endpoint, see the *Authentication* section below.

Several optional parameters are also provided as shown. The first, `signals-index-pipeline`, can be used to define an index pipeline that will to be used to convert the raw JSON signal data into a set of Solr documents. As stated in the *Fusion Signals API documentation*, if no pipeline is defined, then the preconfigured `_signals_ingest` pipeline will be used. Both the remaining two parameters, `commit` and `async`, are booleans. If `commit` is set to true, a Solr commit will occur at the end of indexing allowing a persistent record of the activity to be kept. If `async` is set to true, signals will be indexed in asynchronous mode. In this mode, an autoCommit is issued with each signal and failures are not reported. The default is false.

Creating a collection for signals tracking

Using the *Fusion UI*, when a primary data collection is created, this also creates a collection for the signals. By default, the name of this signals collection is by `<primary collection>_signals`, where `<primary collection>` should be substituted with the name of the primary data collection. After data has been indexed and ingested into the primary collection, the Appkit Fusion Signals module can be used to populate the signals collection.

Tracking user clicks and annotations

Currently, Appkit will send information on user clicks that trigger URL requests as well as when a user creates an annotation, for example, a bookmark or comment.

A signal event of type `click` will include, amongst other things, information about the URL that was clicked as well as the number of times that URL was clicked.

A signal event of type `annotation` will include, amongst other things, information about the annotation target, the collection, and the creator.

For a full list of properties that are stored, refer to the signals collection associated with your application.

Authentication

Service account impersonation requires credentials for the service account and these are stored in the Fusion platform configuration. These credentials will be pulled from the configuration given in the services/api/fusion.conf file.

If the credentials for a service account are not provided or cannot be found, a check will be made to see if a Fusion session cookie is available. Typically, this will be present when the user is known to the Fusion server and the Session API has been set up accordingly.

Related articles

- *Logging Activity*

Train a Smart Answers cold start model

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

Important

The Smart Answers Cold Start Training job is deprecated in Fusion 5.12.

The *cold start solution for Smart Answers* begins with training a model using your existing content. To do this, you run the *Smart Answers Coldstart Training* job. This job uses variety of word embeddings, including custom via Word2Vec training, to learn about the vocabulary that you want to search against.

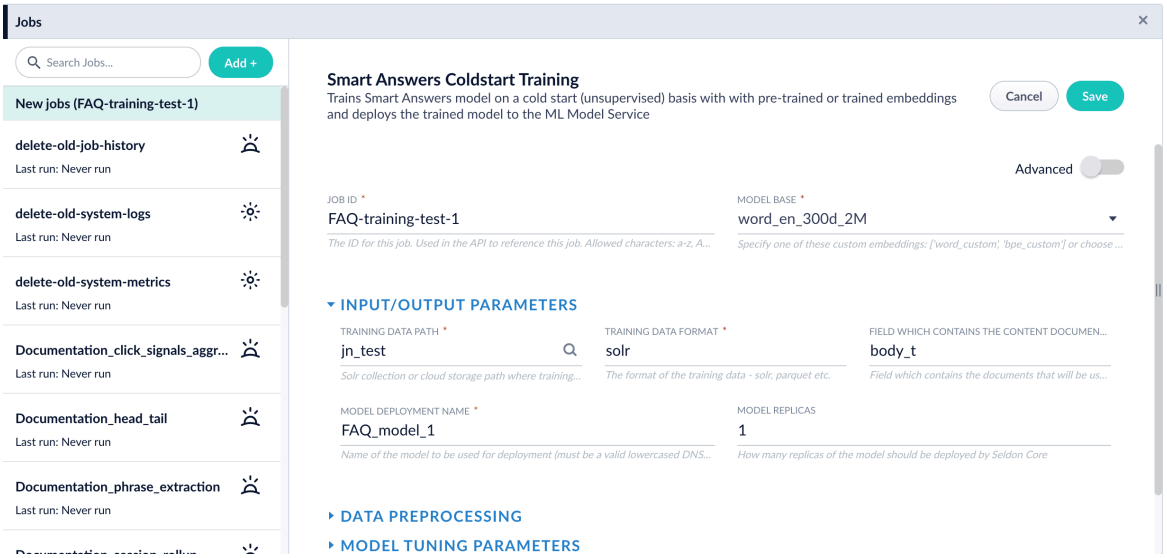
Tip

Smart Answers comes with two pre-trained cold-start models. If your data does not have many domain-specific words, then consider *using a pre-trained model*.

During a cold start, we suggest capturing user feedback such as document clicks, likes, and downloads on the website. After accumulating feedback data and at least 3,000 query/response pairs, the feedback can be used to *train a model using the Supervised method*.

Configure the training job

1. In Fusion, navigate to **Collections** > **Jobs**.
2. Select **Add** > **Smart Answer Coldstart Training**.
3. In the **Training Collection** field, specify the collection that contains the content that can be used to answer questions.
4. Enter the name of the **Field which contains the content documents**.
5. Enter a **Model Deployment Name**.
   The new machine learning model is saved in the blob store with this name. You will reference it later when you configure your pipelines.
6. Configure the **Model base**.
   There are several pre-trained word and BPE embeddings for different languages, as well as a few pre-trained BERT models.
   If you want to train custom embeddings, please select `word_custom` or `bpe_custom`. This trains Word2vec on the data and fields specified in **Training collection** and **Field which contains the content documents**. It might be useful in cases when your content includes unusual or domain-specific vocabulary.
   When you use the pre-trained embeddings, the *log* shows the percentage of processed vocabulary words. If this value is high, then try using custom embeddings.
   During the training job analyzes the content data to select weights for each of the words. The result model performs the weighted average of word embeddings to obtain final single dense vector for the content.
7. Click **Save**.



Note

If using solr as the training data source ensure that the source collection contains the `random_*` dynamic field defined in its `managed-schema.xml`. This field is required for sampling the data. If it is not present, add the following entry to the `managed-schema.xml` alongside other dynamic fields `<dynamicField name="random_*" type="random"/>` and <fieldType class="solr.RandomSortField" indexed="true" name="random"/> alongside other field types.

8. Click **Run** > **Start**.

After training is finished the model is deployed into the cluster and can be used in index and query pipelines.

Next steps

1. *Configure The Smart Answers Pipelines*
2. *Evaluate a Smart Answers Query Pipeline*

Train a Smart Answers Cold Start Model

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion*

Important

The Smart Answers Cold Start Training job is deprecated in Fusion 5.12.

The *cold start solution for Smart Answers* begins with training a model using your existing content. To do this, you run the *Smart Answers Coldstart Training* job (*QnA Coldstart Training* job in Fusion 5.1 and 5.2). This job uses variety of word embeddings, including custom via Word2Vec training, to learn about the vocabulary that you want to search against.

Tip

Smart Answers comes with two pre-trained cold-start models. If your data does not have many domain-specific words, then consider *using a pre-trained model*.

During a cold start, we suggest capturing user feedback such as document clicks, likes, and downloads on the Web site (*App Studio* can help you get started). After accumulating feedback data and at least 3,000 query/response pairs, the feedback can be used to *train a model using the Supervised method*.

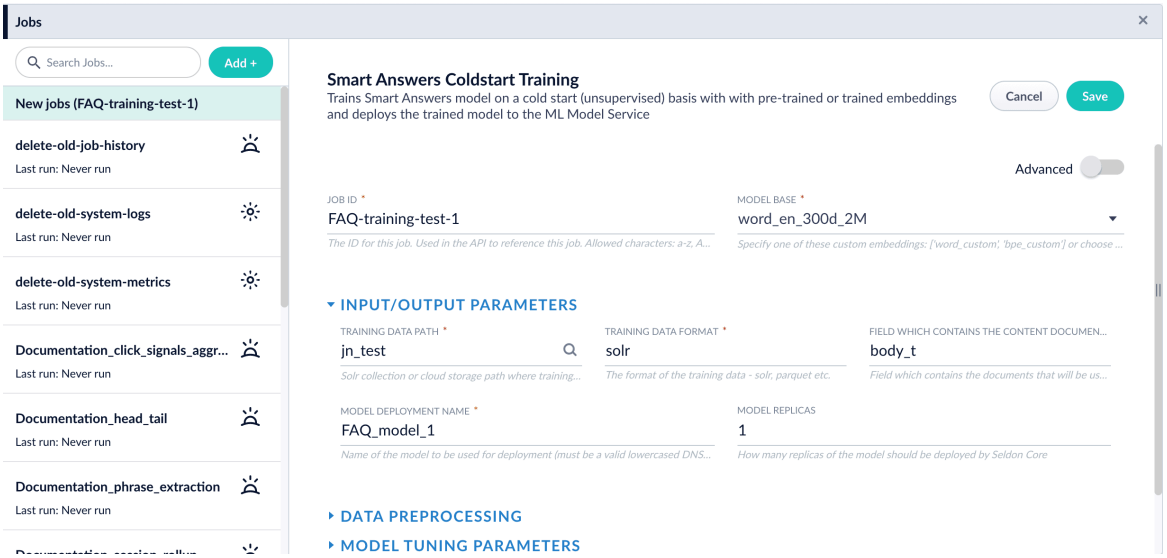
Configure the training job

1. In Fusion, navigate to **Collections** > **Jobs**.
2. Select **Add** > **Smart Answer Coldstart Training**.
3. In the **Training Collection** field, specify the collection that contains the content that can be used to answer questions.

   Note

   In Fusion 5.3 and later, you can also configure this job to read from or write to cloud storage. See *Configure An Argo-Based Job to Access GCS* and *Configure An Argo-Based Job to Access S3*.

4. Enter the name of the **Field which contains the content documents**.
5. Enter a **Model Deployment Name**.
   The new machine learning model will be saved in the blob store with this name. You will reference it later when you configure your pipelines.
6. **Fusion 5.3 and later:** Configure the **Model base**.
   There are several pre-trained word and BPE embeddings for different languages, as well as a few pre-trained BERT models.
   If you want to train custom embeddings, please select `word_custom` or `bpe_custom`. This trains Word2vec on the data and fields specified in **Training collection** and **Field which contains the content documents**. It might be useful in cases when your content includes unusual or domain-specific vocabulary.
   When you use the pre-trained embeddings, the *log* shows the percentage of processed vocabulary words. If this value is high, then try using custom embeddings.
   During the training job analyzes the content data to select weights for each of the words. The result model performs the weighted average of word embeddings to obtain final single dense vector for the content.
7. Click **Save**.



Note

If using solr as the training data source ensure that the source collection contains the `random_*` dynamic field defined in its `managed-schema.xml`. This field is required for sampling the data. If it is not present, add the following entry to the `managed-schema.xml` alongside other dynamic fields `<dynamicField name="random_*" type="random"/>` and <fieldType class="solr.RandomSortField" indexed="true" name="random"/> alongside other field types.

8. Click **Run** > **Start**.

After training is finished the model is deployed into the cluster and can be used in index and query pipelines.

Next steps

1. *Configure The Smart Answers Pipelines*
2. *Evaluate a Smart Answers Query Pipeline*

Train a Smart Answers Supervised Model

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion*

The *Supervised solution for Smart Answers* begins with training a model using your existing data and the *Smart Answers Supervised Training job* (*QnA Supervised Training job* in 5.1 and 5.2), as explained in this topic. The job includes an auto-tune feature that you can use instead of manually tuning the configuration.

See also *Advanced Model Training Configuration for Smart Answers*.

Training job requirements

Storage

150GB plus 2.5 times the total input data size.

Processor and memory

The memory requirements depend on whether you choose GPU or CPU processing:

GPU	CPU
• one core • 11GB RAM	• 32 cores • 32GB RAM

Tip

If your training data contains more than 1 million entries, use GPU.

1. Prepare the input data

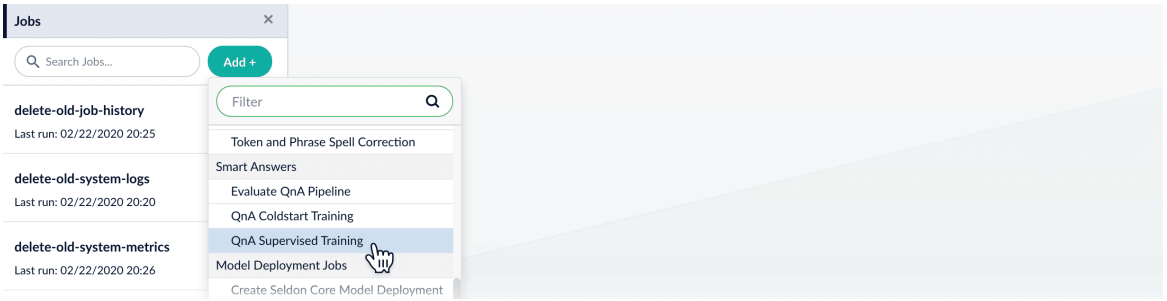
1. Format your input data as question/answer pairs, that is, a query and its corresponding response in each row.
   You can do this in any format that Fusion supports.
   If there are multiple possible answers for a unique question, then repeat the questions and put the pair into different rows to make sure each row has one question and one answer, as in the example JSON below:

```
[{"question":"How to transfer personal auto lease to business auto lease?","answer":"I would approach the lender that you are getting the lease from..."} {"question":"How to transfer personal auto lease to business auto lease?","answer":"See what the contract says about transfers or subleases..."}]
```

2. Index the input data in Fusion.
   If you wish to have the training data in Fusion, index it into a separate collection for training data such as `model_training_input`. Otherwise you can use it directly from the cloud storage.

2. Configure the training job

1. In Fusion, navigate to **Collections** > **Jobs**.
2. Select **Add** > **Smart Answers Supervised Training**:



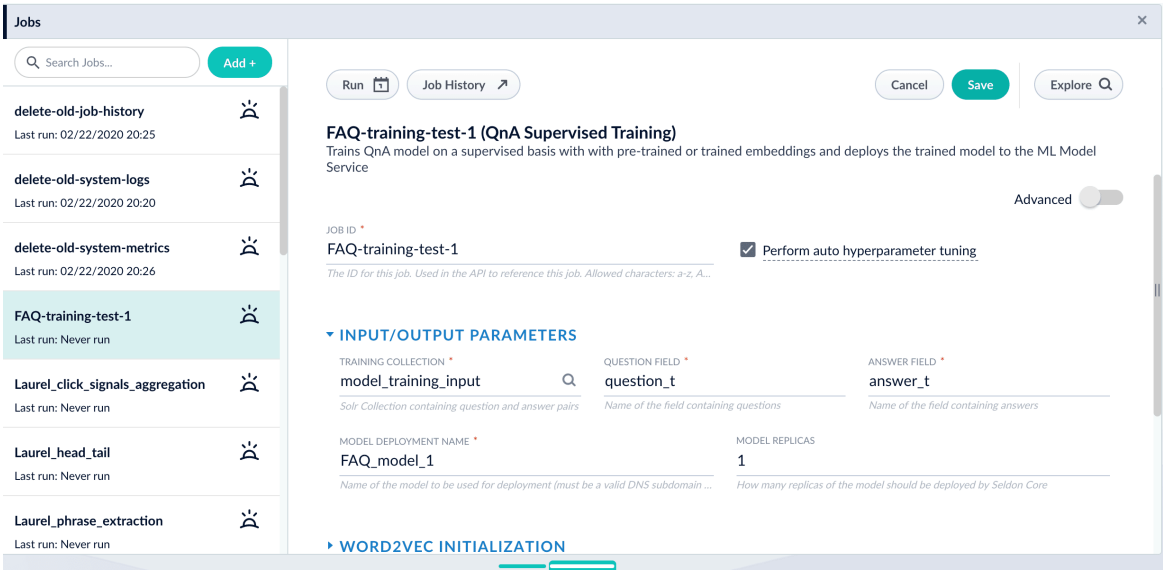
3. In the **Training Collection** field, specify the input data collection that you created when you *prepared the input data*.

> **Note**
>
> In Fusion 5.3 and later, you can also configure this job to read from or write to cloud storage. See *Configure An Argo-Based Job to Access GCS* and *Configure An Argo-Based Job to Access S3*.

4. Enter the names of the **Question Field** and the **Answer Field** in the training collection.
5. Enter a **Model Deployment Name**.
   The new machine learning model will be saved in the blob store with this name. You will reference it later when you configure your pipelines.
6. *Fusion 5.3 and later:* Configure the **Model base**.
   There are several pre-trained word and BPE embeddings for different languages, as well as a few pre-trained BERT models.
   If you want to train custom embeddings, select `word_custom` or `bpe_custom`. This trains Word2vec on the provided data and specified fields. It might be useful in cases when your content includes unusual or domain-specific vocabulary.
   If you have content in addition to the query/response pairs that can be used to train the model, then specify it in the **Texts Data Path**.
   When you use the pre-trained embeddings, the *log* shows the percentage of processed vocabulary words. If this value is high, then try using custom embeddings.
   The job trains a few (configurable) RNN layers on top of word embeddings or fine-tunes a BERT model on the provided training data. The result model uses an attention mechanism to average word embeddings to obtain the final single dense vector for the content.

> **Tip**
>
> Dimension size of vectors for Transformer-based models is 768. For RNN-based models it is 2 times the number units of the last layer. To find the dimension size: download the model, expand the zip, open the log and search for `Encoder output dim size:` line. You might need this information when creating collections in Milvus.

7. *Optional:* Check **Perform auto hyperparameter tuning** to use auto-tune.
   Although training module tries to select the most optimal default parameters based on the training data statistics, auto-tune can extend it by automatically finding even better training configuration through hyper-parameter search. Although this is a resource-intensive operation, it can be useful to identify the best possible RNN-based configuration. Transformer-based models like BERT are not used during auto hyperparameter tuning as they usually perform better yet they are much more expensive on both training and inference time.
8. Click **Save**.



Note

If using solr as the training data source ensure that the source collection contains the `random_*` dynamic field defined in its `managed-schema.xml`. This field is required for sampling the data. If it is not present, add the following entry to the `managed-schema.xml` alongside other dynamic fields `<dynamicField name="random_*" type="random"/>` and <fieldType class="solr.RandomSortField" indexed="true" name="random"/> alongside other field types.

9. Click **Run** > **Start**.

After training is finished the model is deployed into the cluster and can be used in index and query pipelines.

Next steps

1. *See A Smart Answers Supervised Job's Status and Output*
2. *Configure The Smart Answers Pipelines*
3. *Evaluate a Smart Answers Query Pipeline*

Train a Smart Answers Supervised Model

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

The *Supervised solution for Smart Answers* begins with training a model using your existing data and the *Smart Answers Supervised Training job*, as explained in this topic. The job includes an auto-tune feature that you can use instead of manually tuning the configuration.

Training job requirements

Storage

150GB plus 2.5 times the total input data size.

Processor and memory

The memory requirements depend on whether you choose GPU or CPU processing:

GPU	CPU
• one core • 11GB RAM	• 32 cores • 32GB RAM

Tip

If your training data contains more than 1 million entries, use GPU.

1. Prepare the input data

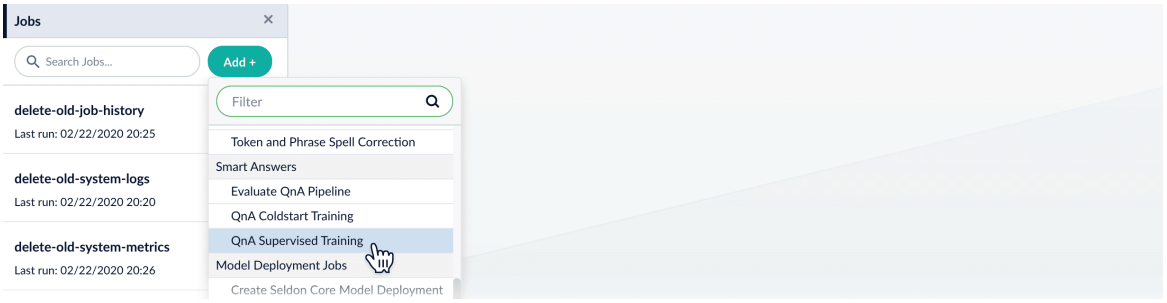
1. Format your input data as question/answer pairs, that is, a query and its corresponding response in each row.
   You can do this in any format that Managed Fusion supports.
   If there are multiple possible answers for a unique question, then repeat the questions and put the pair into different rows to make sure each row has one question and one answer, as in the example JSON below:

```
[{"question":"How to transfer personal auto lease to business auto lease?","answer":"I would approach the lender that you are getting the lease from..."} {"question":"How to transfer personal auto lease to business auto lease?","answer":"See what the contract says about transfers or subleases..."}]
```

2. Index the input data in Managed Fusion.
   If you wish to have the training data in Managed Fusion, index it into a separate collection for training data such as `model_training_input`. Otherwise you can use it directly from the cloud storage.

2. Configure the training job

1. In Managed Fusion, navigate to **Collections** > **Jobs**.
2. Select **Add** > **Smart Answers Supervised Training**:



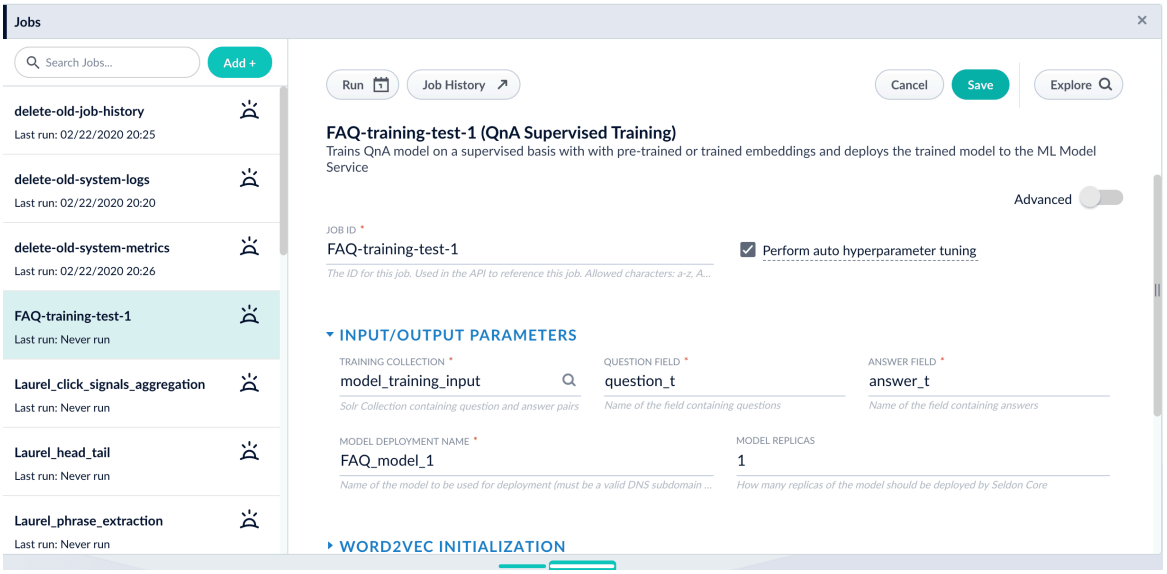
3. In the **Training Collection** field, specify the input data collection that you created when you *prepared the input data*.

> **Note**
>
> You can also configure this job to read from or write to cloud storage. See *Configure An Argo-Based Job to Access GCS* and *Configure An Argo-Based Job to Access S3*.

4. Enter the names of the **Question Field** and the **Answer Field** in the training collection.
5. Enter a **Model Deployment Name**.
   The new machine learning model will be saved in the blob store with this name. You will reference it later when you configure your pipelines.
6. Configure the **Model base**.
   There are several pre-trained word and BPE embeddings for different languages, as well as a few pre-trained BERT models.
   If you want to train custom embeddings, select `word_custom` or `bpe_custom`. This trains Word2vec on the provided data and specified fields. It might be useful in cases when your content includes unusual or domain-specific vocabulary.
   If you have content in addition to the query/response pairs that can be used to train the model, then specify it in the **Texts Data Path**.
   When you use the pre-trained embeddings, the *log* shows the percentage of processed vocabulary words. If this value is high, then try using custom embeddings.
   The job trains a few (configurable) RNN layers on top of word embeddings or fine-tunes a BERT model on the provided training data. The result model uses an attention mechanism to average word embeddings to obtain the final single dense vector for the content.

> **Tip**
>
> Dimension size of vectors for Transformer-based models is 768. For RNN-based models it is 2 times the number units of the last layer. To find the dimension size: download the model, expand the zip, open the log and search for `Encoder output dim size:` line. You might need this information when creating collections in Milvus.

7. *Optional:* Check **Perform auto hyperparameter tuning** to use auto-tune.
   Although training module tries to select the most optimal default parameters based on the training data statistics, auto-tune can extend it by automatically finding even better training configuration through hyper-parameter search. Although this is a resource-intensive operation, it can be useful to identify the best possible RNN-based configuration. Transformer-based models like BERT are not used during auto hyperparameter tuning as they usually perform better yet they are much more expensive on both training and inference time.
8. Click **Save**.



Note

If using solr as the training data source ensure that the source collection contains the `random_*` dynamic field defined in its `managed-schema.xml`. This field is required for sampling the data. If it is not present, add the following entry to the `managed-schema.xml` alongside other dynamic fields `<dynamicField name="random_*" type="random"/>` and <fieldType class="solr.RandomSortField" indexed="true" name="random"/> alongside other field types.

9. Click **Run** > **Start**.

After training is finished the model is deployed into the cluster and can be used in index and query pipelines.

Next steps

1. *See A Smart Answers Supervised Job's Status and Output*
2. *Configure The Smart Answers Pipelines*
3. *Evaluate a Smart Answers Query Pipeline*

Troubleshoot a JDBC Datasource

How-to

> Compatible versions: 4.2, 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion Connectors*, *Fusion Server*, *Fusion*

The JDBC connector fetches documents from a relational database via SQL queries. Under the hood, this connector implements the Solr DataImportHandler (DIH) plugin.

When using the JDBC connector, it is recommended that you work closely with your database administrator to formulate efficient and robust queries.

One source of possible problems is the driver being used. In some cases, indexing may fail due to problems with the driver, in particular older versions of Oracle's JDBC driver. If you have checked that your connection information is correct and your database is allowing the connection, you may want to research if there are any known bugs with the driver you are using.

With Oracle databases, note that column names not enclosed in double-quotes are converted to upper-case, but Solr field names are case sensitive. If your column-to-field mapping is not happening properly, check your SQL statement for any lower-case names not enclosed in double-quotes.

Dates can also be problematic. Solr has a different date format than many relational databases. If you want date and time fields to be indexed properly, you may need to convert database dates into the proper format using date/string convert functions. In Oracle this is the `TO_CHAR` function; in Microsoft SQL, this is the `DATEPART` function.

In MySQL databases, dates are allowed to be 0-strings, such as `0000-00-00`, which is not acceptable to JDBC. If you have legacy date data you may need to add the query parameter `zeroDateTimeBehavior=convertToNull` to your JDBC request string, as in `jdbc:mysql://localhost/myDatabase?zeroDateTimeBehavior=convertToNull`. This will convert the zero-string dates to null values that can be added to the index.

Finally, database timeouts are another problematic area. There are several possible solutions to this, from increasing the timeout in the JDBC request (with `netTimeoutForStreamingResults`), altering the SQL statement to page the results, or dumping the records to CSV and indexing them with another connector.

Troubleshoot a JDBC Datasource

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

The JDBC connector fetches documents from a relational database via SQL queries. Under the hood, this connector implements the Solr DataImportHandler (DIH) plugin.

When using the JDBC connector, it is recommended that you work closely with your database administrator to formulate efficient and robust queries.

One source of possible problems is the driver being used. In some cases, indexing may fail due to problems with the driver, in particular older versions of Oracle's JDBC driver. If you have checked that your connection information is correct and your database is allowing the connection, you may want to research if there are any known bugs with the driver you are using.

With Oracle databases, note that column names not enclosed in double-quotes are converted to upper-case, but Solr field names are case sensitive. If your column-to-field mapping is not happening properly, check your SQL statement for any lower-case names not enclosed in double-quotes.

Dates can also be problematic. Solr has a different date format than many relational databases. If you want date and time fields to be indexed properly, you may need to convert database dates into the proper format using date/string convert functions. In Oracle this is the `TO_CHAR` function; in Microsoft SQL, this is the `DATEPART` function.

In MySQL databases, dates are allowed to be 0-strings, such as `0000-00-00`, which is not acceptable to JDBC. If you have legacy date data you may need to add the query parameter `zeroDateTimeBehavior=convertToNull` to your JDBC request string, as in `jdbc:mysql://localhost/myDatabase?zeroDateTimeBehavior=convertToNull`. This will convert the zero-string dates to null values that can be added to the index.

Finally, database timeouts are another problematic area. There are several possible solutions to this, from increasing the timeout in the JDBC request (with `netTimeoutForStreamingResults`), altering the SQL statement to page the results, or dumping the records to CSV and indexing them with another connector.

Troubleshoot an Alfresco datasource

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

The Alfresco Connector is a crawler for the Alfresco server, which adheres to the Content Management Interoperability Services (CMIS) standard.

If you are having issues with your *Alfresco datasource*, try the troubleshooting steps below:

1. Check the CMIS connection URL

Use the following command to validate the CMIS connection URL is correct. Replace the username, hostname, and port with your own:

```
curl –v –u <username> "https://<hostname>:<port>/alfresco/api/–default–/public/cmis/versions/1.1/atom"
```

The expected response should be a 200 HTTP status response and an XML file describing the CMIS 1.1 AtomPub Service Document. As a first step, the connector retrieves and loads the AtomPub service document and the information there will be useful for future requests.

2. Validate URI templates

The CMIS 1.1 AtomPub service document describes the format of the URLs. For example:

```
<cmisra:uritemplate> <cmisra:template>https://<hostname>:
<port>/alfresco/api/–default–/public/cmis/versions/1.1/atom/id? id=
{id}&amp;filter={filter}&amp;includeAllowableActions=
{includeAllowableActions} &amp;includeACL=
{includeACL}&amp;includePolicyIds={includePolicyIds}
&amp;includeRelationships={includeRelationships} &amp;renditionFilter=
{renditionFilter} </cmisra:template>
<cmisra:type>objectbyid</cmisra:type>
<cmisra:mediatype>application/atom+xml;type=entry</cmisra:mediatype>
</cmisra:uritemplate>
```

Those template URIs will be used to build the requests the connector will require to retrieve data from Alfresco server.

To test if the URI templates are correct, run any of the URI templates on that document, replace the necessary parameters and IDs. For example, to retrieve the root folder using the path parameter: **path=/**

```
curl –v –u admin "https://<hostname>:<port>/alfresco/api/–
default–/public/cmis/versions/1.1/atom/path?
path=/&amp;filter=&amp;includeAllowableActions=&amp;includeACL=&amp;in
cludePolicyIds=&amp;includeRelationships=&amp;renditionFilter="
```

The expected response should be a 200 HTTP status response and an XML file describing the root folder. It's possible to use other URI template queries to validate they are correct.

3. Check Alfresco opencmis properties

If a request using the URI template returns a non OK HTTP status, probably the Alfresco opencmis properties are incorrect. To check that:

- Go to the Alfresco admin console:

```
http://<your–host–name>:<alfresco–
port>/alfresco/service/enterprise/admin
```

- Under the **Support Tools** section, go to the **JMX Settings**
- Click the **Export** button, under the **Export System Settings** section.
- That downloads a file that is listing all the current Alfresco server properties.
- Look for the property: **opencmis.server.value**. The URI templates will be built using the URL set in that property.
    - If the URL set in that property is incorrect change it.
    - It would be necessary to change other properties too:

```
opencmis.context.override=true
opencmis.context.value=
opencmis.servletpath.override=true
opencmis.servletpath.value=
opencmis.server.override=true
opencmis.server.value=https://<hostname>:<port>/alfresco/api
```

- Change those properties in the **alfresco-global.properties** file.

It's important to configure that value with a correct URL. Otherwise the requests will fail and documents can not be indexed.

Configurations taken from a related question in the Alfresco forum.

Troubleshoot an Alfresco Datasource

How-to

> Compatible versions: 4.2, 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion Connectors*, *Fusion Server*, *Fusion*

The Alfresco Connector is a crawler for the Alfresco server, which adheres to the Content Management Interoperability Services (CMIS) standard.

If you are having issues with your *Alfresco datasource*, try the troubleshooting steps below:

1. Check the CMIS connection URL

Use the following command to validate the CMIS connection URL is correct(replace the username, hostname and port):

```
curl –v –u <username> "https://<hostname>:<port>/alfresco/api/–default–/public/cmis/versions/1.1/atom"
```

The expected response should be a 200 HTTP status response and an XML file describing the CMIS 1.1 AtomPub Service Document. As a first step, the connector will retrieve and load the AtomPub service document and the information there will be useful for future requests.

2. Validate URI templates

The CMIS 1.1 AtomPub service document describes the format of the URLs. For example:

```
<cmisra:uritemplate> <cmisra:template>https://<hostname>:
<port>/alfresco/api/–default–/public/cmis/versions/1.1/atom/id? id=
{id}&amp;filter={filter}&amp;includeAllowableActions=
{includeAllowableActions} &amp;includeACL=
{includeACL}&amp;includePolicyIds={includePolicyIds}
&amp;includeRelationships={includeRelationships} &amp;renditionFilter=
{renditionFilter} </cmisra:template>
<cmisra:type>objectbyid</cmisra:type>
<cmisra:mediatype>application/atom+xml;type=entry</cmisra:mediatype>
</cmisra:uritemplate>
```

Those template URIs will be used to build the requests the connector will require to retrieve data from Alfresco server.

To test if the URI templates are correct, run any of the URI templates on that document, replace the necessary parameters and IDs. For example, to retrieve the root folder using the path parameter: **path=/**

```
curl –v –u admin "https://<hostname>:<port>/alfresco/api/–
default–/public/cmis/versions/1.1/atom/path?
path=/&amp;filter=&amp;includeAllowableActions=&amp;includeACL=&amp;in
cludePolicyIds=&amp;includeRelationships=&amp;renditionFilter="
```

The expected response should be a 200 HTTP status response and an XML file describing the root folder. It's possible to use other URI template queries to validate they are correct.

3. Check Alfresco opencmis properties

If a request using the URI template returns a non OK HTTP status, probably the Alfresco opencmis properties are incorrect. To check that:

- Go to the Alfresco admin console:

```
http://<your–host–name>:<alfresco–
port>/alfresco/service/enterprise/admin
```

- Under the **Support Tools** section, go to the **JMX Settings**
- Click the **Export** button, under the **Export System Settings** section.
- That will download a file that is listing all the current Alfresco server properties.
- Look for the property: **opencmis.server.value**. The URI templates will be built using the URL set in that property.
    - If the URL set in that property is incorrect change it.
    - It would be necessary to change other properties too:

```
opencmis.context.override=true
opencmis.context.value=
opencmis.servletpath.override=true
opencmis.servletpath.value=
opencmis.server.override=true
opencmis.server.value=https://<hostname>:<port>/alfresco/api
```

- Change those properties in the **alfresco-global.properties** file.

It's important to configure that value with a correct URL, otherwise the requests will fail and documents can not be indexed.

Configurations taken from a related question in the Alfresco forum.

Troubleshoot App Studio Development

How-to

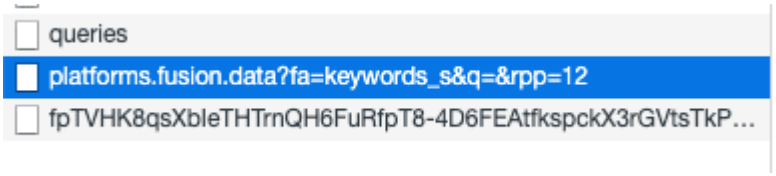
> Compatible versions: 4.2
>
> Works with: *App Studio*

This topic provides troubleshooting tips for issues that can arise when developing web apps using App Studio.

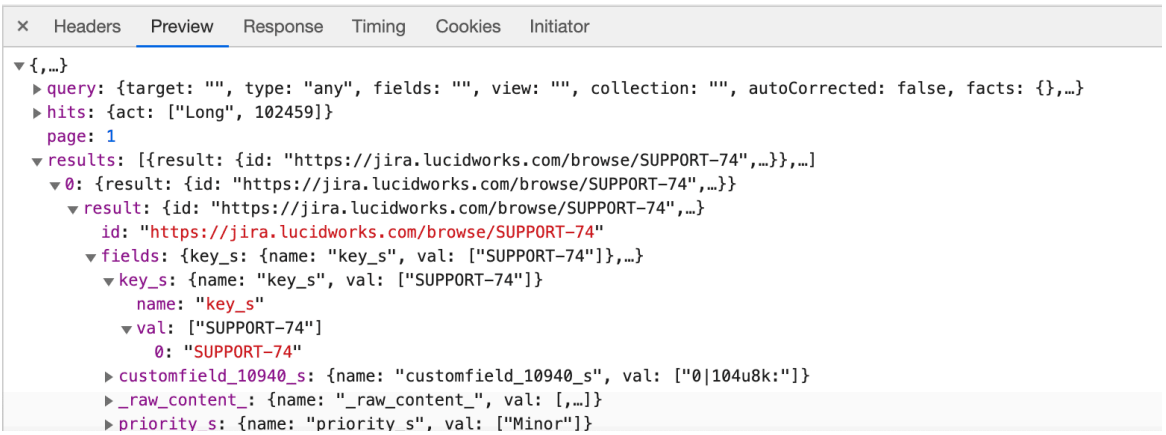
Data is not appearing on the page as expected

Check the response data

1. Open your browser's developer tools.
2. Click the **Network** tab.
3. Look for an XHR request similar to the one shown below (starting with `platforms.fusion…`) and select it.



   Here you can navigate through the values being returned from Fusion.
4. To determine which response is which (as you may have multiple) look at the platform that populates the response, which also populates the element on your page.
   You can then track down the value you are looking for. So if looking at results, a result list iterates through the results, so we should look inside a result. In each result, there's a set of fields and associated values:



   Note that other processing may happen in between receiving the response and displaying it on the page (response processors, modifications in the HTML, security trimming, and so on).

Get query from webapps/servlet logs and fire it against Fusion

If the data in your response object is not as you expect, you can get the query from the App Studio logs to fire directly against the Fusion endpoint, without App Studio in the middle.

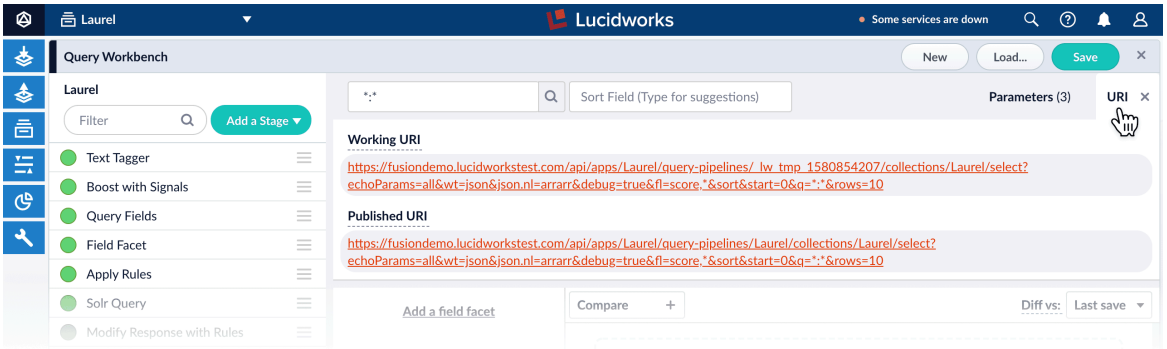
1. Make sure your logger is configured to print out debug logging for the `twigkit` class in `logback.xml` in your environment profile (or if not using a profile, in the main resources directory):

```
<logger name="twigkit" level="DEBUG"/>
```

2. In `fusion/version/var/log/webapps/`, sort by date `(ls -l)` and view the latest one with the suffix of `stderrout`.
   Here you will see some entries that should follow the format of the following:

```
08:57:41.058 DEBUG twigkit.search.solr.SolrPlatform - Searching
Solr with
start=0&rows=20&hl=true&q=match&facet=true&facet.field=
{!ex%3Dgenre_ss}genre_ss&facet.field=year_i&fl=*,score&fq=
{!tag%3Dgenre_ss}genre_ss:
(Animation)&username=admin&session=37045658-9052-4ba4-b1d2-
0221be5e2d02&spellcheck=true&spellcheck.collate=true&q.op=OR
took 200 ms
```

3. To determine which entry is the correct one, check the query that populates the response you're interested in, and match the parameters you're setting in the tag as well as any Solr parameters being applied to the query via interaction on the page (keyword = `q`, facet filters = `fq`, and so on).
4. Go to the Query Workbench in Fusion and select the Published URI in the URI tab:



5. Remove every parameter given in this URL after `select?`.
6. Append the parameters from the webapps logs.
7. Put the resulting URL into your browser.
   The result will be the response of the query fired by App Studio, but unaltered by App Studio.

Page is totally blank/elements look broken or strange

Check the browser console

1. Open your browser's developer tools.
2. Click the **Console** tab.
   Here you can check for any issues related to client-side components.
3. Look for the following error:

```
Uncaught Error: [$injector:modulerr] Failed to instantiate
module myApp due to: Error: [$injector:modulerr] Failed to
instantiate module lightning due to: Error: [$injector:nomod]
Module 'lightning' is not available! You either misspelled the
module name or forgot to load it. If registering a module ensure
that you specify the dependencies as the second argument.
```

- If you see the error above, you should wipe your **`node_modules`** directory and reinstall the modules before building:
    1. **`rm -rf node_modules package-lock.json`**
    2. **`npm cache clear -f`**
    3. **`npm install`**
- If you do not see this error, then typically there is a problem in your Javascript code.

Troubleshoot Chrome Binary for Web Connector

How-to

> Compatible versions: 4.2
>
> Works with: *Fusion Connectors*, *Fusion Server*, *Fusion*

`chromeBinaryPath` *values*

If Google Chrome was installed via package installation manager, specify the Chrome binary in the `chromeBinaryPath`. Example locations are `/usr/bin/google-chrome` or `/path/to/customlocation/of/google-chrome`.

For RHEL builds, run:

```
wget https://dl.google.com/linux/direct/google-chrome-
stable_current_x86_64.rpm yum -y install redhat-lsb libXScrnSaver yum
-y localinstall google-chrome-stable_current_x86_64.rpm
```

Then use `which google-chrome` to find where it installs and use that in the `chromeBinaryPath`.

Troubleshooting Chrome binary errors

Steps to verify:

- Make sure Chrome is installed properly on the system and `google-chrome --version` returns successfully.
- If a custom Chrome binary path is set in the datasource configuration, make sure is it valid.
- If bundled Chromium is used, where is it located? The Chrome install script puts it inside `var/connectors/connectors-classic/chrome-linux`, but Fusion looks for it inside `var/connectors/connectors-classic/chrome-linux64`.
- Ensure proper permissions are set for the Fusion user.
- If running Fusion locally, try unchecking the Headless Browser option. This allows Chrome to open to the page = the crawler is indexing and lets you know that Chrome is being reached by Fusion.

Troubleshoot Custom Spark Jobs

How-to

> Compatible versions: 4.2, 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion Server*, *Fusion*

To troubleshoot problems with a custom job, start by looking for errors in the script-job driver log, `/opt/fusion/latest.x/var/log/api/spark-driver-launcher.log` (on Unix) or `C:\lucidworks\var\fusion\latest.x\var\log\api\spark-driver-launcher.log` (on Windows).

For general Spark job troubleshooting steps, see *Spark troubleshooting*.

Troubleshoot failed datasource jobs

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion Connectors*, *Fusion*, *Managed Fusion*

When indexing large files, or large quantities of files, you may encounter issues such as datasource jobs failing or documents not making it into Fusion.

Overview

When data flows into Fusion, it passes through a Kafka topic first. When the number of documents being created by a connector is large, or when the connector is pulling data into the Kafka topic faster than it can be indexed, the topic fills up and the datasource job fails. For example, if your connector is inputting a large CSV file where every row is imported as a separate Solr document, the indexing processing can time out before the document is fully ingested.

Identify the cause

If you experience failed datasource jobs or notice your connector isn't grabbing all the documents it should, check the logs for the Kafka pod. Look for a message containing the phrases `resetting offset` and `is out of range`, which indicate data has been dropped.

```
2024-05-28T11:49:40.812Z - INFO [pool-140-thread-
3:org.apache.kafka.clients.consumer.internals.Fetcher@1413] -
[Consumer clientId=example_Products-irdcsn, groupId=index-pipeline--
example_Products--fusion.connectors.datasource-products_S3_Load] Fetch
position FetchPosition{offset=6963199, offsetEpoch=Optional[0],
currentLeader=LeaderAndEpoch{leader=Optional[fusion5-kafka-0.fusion5-
kafka-headless.fusion5.svc.cluster.local:9092 (id: 0 rack: null)],
epoch=0}} is out of range for partition fusion.connectors.datasource-
products_S3_Load-2, resetting offset
```

Adjust indexing settings

If you determine that your datasource job is failing due to an issue in Kafka, there are a few options to try.

Adjust retention parameters

One solution is to increase the Kafka data retention parameters to allow for larger documents. You can configure these settings in your `values.yaml` file in the Helm chart.

1. The default value for `kafka.logRetentionBytes` is `1073741824` bytes (1 GB).
   Try increasing this value to `2147483648` bytes (2 GB) or `3221225472` (3 GB), or larger depending on the size of your documents.

   Note

   In Fusion 5.9.5, the default value is increased to 5 GB.

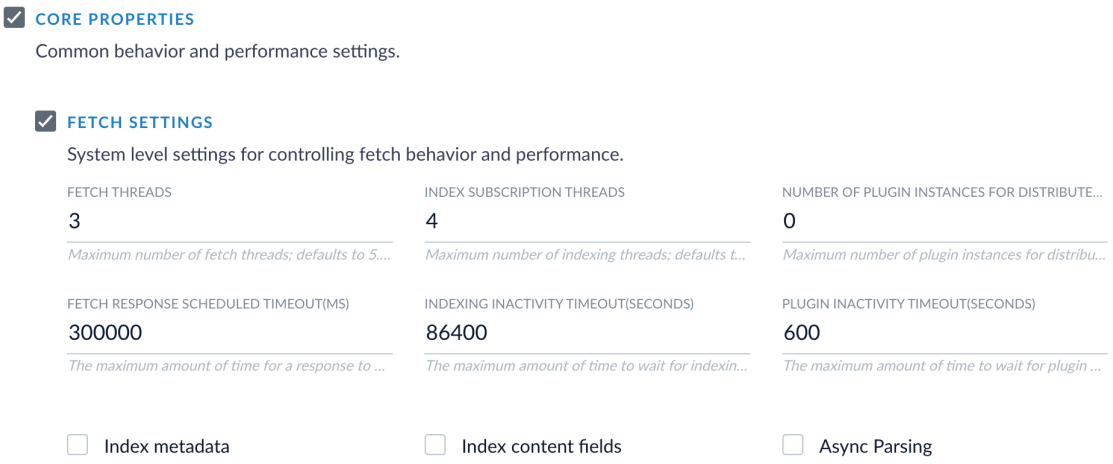
   You can also set this to `-1` to remove the size limit. If you do this, be sure to set an appropriate limit for `logRetentionHours` instead.
2. The default value for `kafka.logRetentionHours` is `168` (7 days).
   If you increase `kafka.logRetentionBytes` by a significant amount (for example, 20 GB), you might need to decrease this setting to prevent running out of disk space. However, because older log entries are deleted when either limit is reached, you should set it high enough to ensure the data remains available until it's no longer needed.

3. In Fusion, go to **Indexing > Datasources** and *create a new datasource* to trigger a new Kafka topic that incorporates these settings.

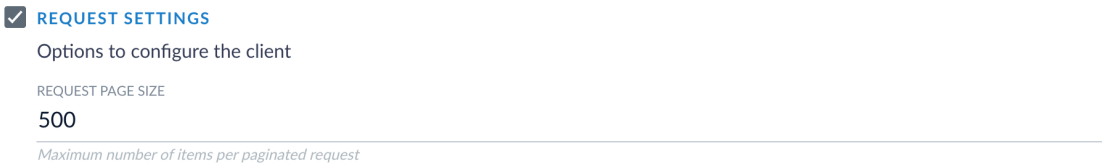
Adjust fetch settings

Another option is to decrease the values for number of fetch threads and request page size in your datasource settings.

1. In Fusion, go to **Indexing > Datasources** and click your datasource.
2. Click the **Advanced** slider to show more settings.
3. Reduce the number of **Fetch Threads**.



4. Reduce the **Request Page Size**.



Note

This setting might not be available in every connector.

Troubleshoot Fusion 4.x Upgrades

How-to

> Compatible versions: 4.2
>
> Works with: *Fusion Server*

Beginning with Fusion 3.1, upgrading Fusion uses a migrator that migrates objects and data from your current version of Fusion to a new version of Fusion or Fusion Server.

The Fusion migrator *log file* will help you troubleshoot difficulties while upgrading. If the migrator fails to complete successfully, we explain how to *correct problems and repeat an upgrade*.

Known issues, and workarounds if available, are listed at the bottom of this topic.

Log file

The log file for the migrator is in `/opt/lucidworks/fusion/x.y.z/var/upgrade/tmp/migrator.log` (on Unix) or `C:\lucidworks\var\fusion\x.y.z\upgrade\tmp\migrator.log` (on Windows). The ***x.y.z*** directory is for the Fusion version that you are migrating *from*.

Completing an upgrade after an error

If the migrator script produces an error, you will need to correct the problem that the script encountered, and then finish the upgrade as explained here.

1. Fix the error

If an upgrade step fails:

1. Examine the *log file* and determine what went wrong.
2. Fix what went wrong.

2. Repeat the step that failed

After you have fixed the source of the error, repeat the step that failed.

Note

You can repeat the step that failed as many times as needed to correct errors that occur. If the step continues to fail, *restart the upgrade from the beginning*.

3. Complete the upgrade

Complete the upgrade.

If a *different* step fails, fix that error, repeat that step, and then perform the remainder of the upgrade procedure.

If the upgrade steps produce no additional errors, then you are finished.

Restarting an upgrade

If an upgrade step continues to fail, then restart the upgrade process from the beginning:

1. Examine the *log file* and determine what went wrong.
2. Remove the directory for the new version of Fusion.
3. Reinstall the new version of Fusion, but do not start it.
4. Fix what went wrong.
5. Perform the entire upgrade procedure from the beginning.

Known issues

Following are known issues with the Fusion migrator.

Upgrades affected	Issue	Workaround (if available)
3.1.x to 4.0.y, 4.1.y, and 4.2.y	After migration, Query Workbench settings are reset for collections other than `default`	For collections other than `default`, print or write down all Query Workbench settings prior to upgrading. After the upgrade, redo the Query Workbench configurations by hand for collections other than `default`.
3.1.x to 4.0.y, 4.1.y, and 4.2.y	Users cannot log in after migration of a SAML realm	This is being investigated. No workaround is available at this time.
3.1.x to 4.0.y, 4.1.y, and 4.2.y	After migration, the query pipeline `default` (created during migration for the app `default`) is associated with collections in the Query Workbench, instead of the query pipelines that *were* associated with the collections before migration (typically `COLLECTION_NAME-default`).	In Query Workbench, for each collection with which the `default` query pipeline is associated, load the correct query pipeline, and then click **Save**.
3.1.x to 4.0.y, 4.1.y, and 4.2.y	Migration of index profiles and query profiles modifies the profiles. For more information, see *Migration of index profiles and query profiles*.	After migration, you might need to adjust the pipeline references of index profiles and query profiles by hand, and/or create new index profiles and query profiles.

<table>
<tr><th>Upgrades affected</th><th>Issue</th><th>Workaround (if available)</th></tr>
<tr><td>3.1.x to 4.2.x</td><td>Migrated JDBC blobs are’t linked to apps.

Note

Apps function fine without these links. You only need to create links if you want the JDBC blobs to be visible under the "In Current App" tab in Object Explorer.</td><td>After migration, link JDBC blobs to apps manually, using either the Fusion UI or the Fusion API.

Fusion UI: (repeat for all apps, and all JDBC blobs in each app) In Object Explorer, locate a JDBC blob under "In Any App" or under "In No Apps". Hover over the object, click the App - icon, and then click Add to this app.

Fusion API:

1. Find the blob IDs of JDBC blobs:

<pre>curl –u USERNAME:PASSWORD "http://localhost:8764/api/blobs?resourceType=driver:jdbc"</pre>
2. (Repeat for all JDBC blobs) To link a JDBC blob to all apps, use this API command:

<pre>curl –u USERNAME:PASSWORD –H "Content-Type:application/json" –X PUT –d '{ "subject" : "group:_shared", "object" : "blob:JDBC_blob_ID", "linkType" : "hasPart", "originator" : "migration" }' "http://localhost:8764/api/links"</pre></td></tr>
</table>

Upgrades affected	Issue	Workaround (if available)
All upgrades *except for* 3.1.x to 3.1.y	The `managed-schema` data in the Solr configuration cannot exceed 1 MB in size. This is an example of a failure message: `2018-05-07T19:31:00,056 - ERROR [main:ZKImportExportComponent@276] - Error writing data to ZK org.apache.zookeeper.KeeperException$ConnectionLossException: KeeperErrorCode = ConnectionLoss for /lwfusion/3.1.5/solr/configs/rh/managed-schema`	Remove unneeded elements from the schema, and then restart the upgrade from the beginning.
3.1.x and 4.0.x to 4.1.y and 4.2.y	Migrated signal-aggregation jobs retain their schedules (by design). To avoid job timeouts, the default schedule for signal-aggregation jobs was increased from every two minutes to every 15 minutes in Fusion 4.1.0.	After migration, consider manually changing the schedule of signal-aggregation jobs to 15 minutes.

Troubleshoot Performance Issues

How-to

> Compatible versions: 4.2
>
> Works with: *Fusion Server*, *Fusion*

Troubleshooting

High CPU load

Heavy CPU load across all nodes in the cluster might indicate the need for more nodes. More typically, however, a single node or service is being overloaded.

- Use `top` or another monitoring utility to find out which processes are using high CPU.
- All Fusion services, including Solr and Zookeeper, run as Java processes. Most services indicate the name of the service via the `-DserviceName` command-line argument.

Common reasons for busy services:

- Connectors-classic: document parsing and ingestion
- Connectors-classic: document-heavy index and/or query pipeline use
- API: document-heavy index and/or query pipeline use
- Solr: use across multiple nodes holding the same collection
- Solr: use across multiple nodes holding shard leaders
- Heavy query load involving facets, stats, and/or sorting
- Heavy document writing to a given collection

What to do:

- Find out which services or nodes are being overloaded.
- Find out if there is known activity to explain the excessive CPU load.
- If no explanation is found, Contact Lucidworks Support .

Low memory

Low memory problems are typically the result of two issues: low system memory and a lack of heap space for an individual service.

LACK OF FREE MEMORY

You can detect a lack of free memory on a server using the `free -h` command. On servers running the Solr service, large amounts of free memory is ideal because this memory is used for filesystem caching. Take note of the amount of free memory and the disk cache in comparison to the overall index size. Ideally, the disk cache should be able to hold all of the index, although this is not as important when using newer SSD technologies.

There are several ways to increase the available memory on a server:

- Change the command-line options configured in `conf/fusion.cors` to reduce the heap sizes of individual services. However, this might result in a *lack of heap space*, as described below.
- Run fewer services on a node and/or reallocate services that need more memory to nodes with extra capacity.
- Add nodes to the cluster.
- Add memory to the nodes.

LACK OF HEAP SPACE

Finding processes that have exited with `OutOfMemoryError` in the logs, or finding a dump file called `java_pidXXX.hprof` in the log directory, indicates that a service failed due to lack of heap space.

Heap space is configured on a per-service basis in the `conf/fusion.cors` file via the `-Xmx` and `-Xms` command-line parameters. Avoid allocating heap sizes that are known to be larger than needed for a service, because these can lead to long GC pauses.

Two common ways to detect long GC pauses include:

- Examine the `gc_*.log` files in the log directories. For deep analysis, upload files to http://gceasy.io .
- Using `top`, look for periods when all cores are busy followed by a spike in one core (with most or all other cores dropping to near zero). This is pattern is typically found when one service is busy and is encountering long GC pauses.

Note

The need for GC analysis varies from application to application.

Troubleshoot Security Trimming Issues

How-to

> Compatible versions: 4.2
>
> Works with: *Fusion*, *Fusion Server*

This topic describes how to troubleshoot issues with the *Security Trimming* query pipeline stage.

One of the most common issues that occurs when working with Fusion is that users do not see the search results expected from security trimming for one reason or another. This issue can show itself in two ways:

- Users see documents that they **should not** see.
- Users **do not see** documents that they **should** see.

An Explanation of the Security Trimming Stage

The Security Trimming stage starts with a user ID. This ID can be a Windows user principal name, a Windows logon ID, an email address, an LDAP user ID, or any other type of identification that represents a search user.

Next, the stage sends your ID to all of the datasources in your application and returns a solr filter query that will trim the data for the specific user.

For example: You give Alfresco the user ID `admin` and the Security Trimming stage will return a filter query such as:

+({!terms f=acls_ss}ADMIN*cmis_read,GROUP_Engineering,GROUP_SustainingEngineering*cmis_read,guest__cmis_read) ---

Since Alfresco documents store the users/groups who have permission to view the document in a special solr field called `acls_ss`, this filter will only return a document if one of the values in the filter matches the `acls_ss` on the document.

Troubleshooting

If you do not receive the expected results from the Security Trimming stage, use the following steps to troubleshoot:

1. Add `&debug=true` to your query so that you get the debug output that will contain the filters that were used when querying.
2. Obtain the filter query that was used for your query. This will contain the groups/user IDs that were matched against the `acl` field for your datasource's documents in order trim your results.
3. Obtain a subset of the documents that were or were not supposed to be returned in your search results, and save the `acl` field for those results. For example, `acls_ss` from the previous section.
4. Compare the `acl` values that were in the filter query to the `acl` values that are on the documents. Search results are only shown when one or more of the `acl` values from the filter match the `acl` values of the documents.
    - If the `acl` values on the document do not match what you expect from your datasource. For example, an Alfreso document gives permission to group XYZ, but that group does not appear in the `acls_ss` field:
        - Make sure the datasource is up to date. It may have a stale index and need a fresh crawl.
        - If the `acls_ss` is still incorrect, open a ticket with Lucidworks Support for further assistance.
    - If the ACL values in the filter query seem inaccurate. For example, you see groups you should not see or are missing groups you should see:
        - Go into your source system and check that the users are actually in the groups that you are expecting them to belong to.
        - If you are sure that the correct groups are not being returned for a user, open a ticket with Lucidworks Support for further assistance.

Troubleshoot SharePoint Permissions

How-to

> Compatible versions: 4.2, 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion Connectors*, *Fusion Server*, *Fusion*

The SharePoint connector retrieves content and metadata from an on-premises SharePoint repository.

When the connector is configured using a SharePoint username without sufficient privileges, the Fusion connectors log file `https://FUSION_HOST:FUSION_PORT/var/log/connectors/connectors.log` contains an error like the following:

```
crawler.common.sharepoint.exception.SharePointException: Server was
unable to process request. ---> Attempted to perform an unauthorized
operation. at
crawler.common.sharepoint.service.BaseService.analyzeResponse(BaseServ
ice.java:194) ~[classes/:?] at
crawler.common.sharepoint.service.SiteDataService.getContentBySiteOrLi
st(SiteDataService.java:169) ~[classes/:?] at
com.lucidworks.permissions.Main.test1(Main.java:50) [classes/:?] at
com.lucidworks.permissions.Main.main(Main.java:32) [classes/:?]
```

This user's permissions may be sufficient to connect via SOAP and read the documents, but not sufficient to get the ACLs and other associated metadata. This may result in complete lack of access to documents, or access to unauthorized documents. Confirm that the configured SharePoint user has the required privileges.

Troubleshoot SQL Queries

How-to

> Compatible versions: 4.2
>
> Works with: *Fusion Server*

If you encounter an issue with a SQL query, the first place to look for more information about the issue is the `var/log/sql/sql.log` file. If you need more verbose log information, change the level to DEBUG for the following loggers in the file `conf/sql-log4j2.xml` (on Unix) or `conf\sql-log4j2.xml` (on Windows):

```
<logger name="com.lucidworks.spark" level="DEBUG"/> <logger
name="com.lucidworks.spark.sql" level="DEBUG"/>
```

After making changes, you must restart the Fusion SQL service. Give these commands from the `bin` directory below the Fusion home directory, for example, `/opt/fusion/latest.x` (on Unix) or `C:\lucidworks\fusion\latest.x` (on Windows).

On Unix:

```
./sql restart
```

On Windows:

```
sql.cmd restart
```

Troubleshoot the DevOps Center

How-to

> Compatible versions: 4.2, 5.5
>
> Works with: *Fusion Server*, *Fusion*

The DevOps Center is enabled by Fusion's default configuration. If you find that the DevOps Center is not correctly populated with data, check the following:

- Check for browser extensions that block HTML5 autoplay.
- Check that the Fusion `log-shipper` service is enabled, as it is by default:

```
fusion/latest.x/bin/log-shipper status
```

If it is not enabled, make sure it is included in the list of default services in `fusion.cors` (`fusion.properties` in Fusion 4.x), as in this example:

```
group.default = zookeeper, solr, api, connectors-classic,
connectors-rpc, proxy, webapps, admin-ui, sql, log-shipper
```

See *Configure Fusion logging* for additional details.

Tip

If you are running some services on separate nodes, make sure those nodes are also running Fusion's `agent` and `log-shipper`.

- In `fusion.cors` (`fusion.properties` in Fusion 4.x), check that `collectMetrics` is "true" and `collectMetricsIntervalSecs` is set to a reasonable interval:

```
# capture host and services metrics # and ship them to
system_monitor Solr collection default.collectMetrics = true
default.collectMetricsIntervalSecs = 30
```

Troubleshoot When Installing Fusion 4.x

How-to

> Compatible versions: 4.2
>
> Works with: *Fusion Server*

This topic explains how to troubleshoot difficulties that occur when installing or upgrading Fusion.

Fusion run script failures

Common problems that cause Fusion run scripts to fail:

- Wrong Java version.
- Spaces in Windows install path.
- Users have insufficient privileges for the installation directory.
- Java `bin` directory not in the `PATH` environment variable.
- Some Fusion services may already be running, or registered as running.
- Roaming IP address; try uncommenting this line in `https://FUSION_HOST:FUSION_PORT/conf/fusion.cors` (`fusion.properties` in Fusion 4.x):

```
default.address = 127.0.0.1
```

Check the Java version

Fusion runs on JDK 1.8 . See *System Requirements*.

Fusion scripts use the environment variable `JAVA_HOME`. To check the setting of this variable, log in to the account used to run Fusion, and check that this variable is set to the proper value. On a linux, Mac, or other Unix system, use the following command:

```
echo $JAVA_HOME
```

On Windows, the command is:

```
echo %JAVA_HOME%
```

Fusion scripts execute both the `java` and `javac` commands. To check the Java version invoked by these commands, run the following commands from a shell or terminal window:

```
java –version javac –version
```

Clear browser cache

If a previous version of Fusion was accessed in the browser with the same URL as that of the newly installed version of Fusion, then there may be old pages and/or cookies in the browser cache. A hard page refresh will clear old pages from the browser cache. If clearing the page cache does not solve this problem, clear session cookies as well.

Stop/Clean up/Start

If the script `https://FUSION_HOST:FUSION_PORT/bin/fusion start` completes without reporting an error, but the Fusion UI displays a message that it cannot find Collections or Datasources, this may be due to Fusion services not being able to communicate properly (via ZooKeeper). This can happen with developer deployments running on a laptop if the network connection changes or is interrupted, especially when using the embedded ZooKeeper instance that is bundled with Fusion.

In this situation, you should stop Fusion, inspect the system processes and if necessary, manually terminate running processes and cleanup `.pid` files to bring the system back to a clean state, then start Fusion once again.

Although the Fusion run script `bin/fusion` provides a restart option, the restart option assumes a correctly functioning system and cannot always recover from system failure.

To stop Fusion:

Run the script `https://FUSION_HOST:FUSION_PORT/bin/fusion` with the argument `stop`:

```
$ cd {path_to}https://FUSION_HOST:FUSION_PORT $ ./bin/fusion stop
Successfully stopped ui (process ID 41524) Successfully stopped
connectors (process ID 41328) Successfully stopped api (process ID
41159) Successfully stopped solr (process ID 41153) Successfully
stopped zookeeper (process ID 41151)
```

After stopping Fusion, you should make sure that no Fusion services are running. When the Fusion scripts start a Fusion service, they record the process id in a `.pid` file in the directory `https://FUSION_HOST:FUSION_PORT/var`. For a Fusion instance that is up and running, we see the following set of `.pid` files:

```
> find {path_to}https://FUSION_HOST:FUSION_PORT/var –name "*.pid" –
print fusion/var/api/api.pid fusion/var/connectors/connectors.pid
fusion/var/solr/solr.pid fusion/var/spark–master/spark–master.pid
fusion/var/spark–worker/spark–worker.pid fusion/var/ui/ui.pid
fusion/var/zookeeper/zookeeper.pid
```

The above output shows the set of `.pid` created by a single Fusion instance running with embedded ZooKeeper and Solr.

But if no Fusion services are running, there should not be any `.pid` files. In the case that all services have been stopped, but there are still some `.pid` files found, these files should be deleted before starting Fusion.

Inspect the log files

If none of the above help, inspect the Fusion log files in directory `https://FUSION_HOST:FUSION_PORT/var/log`.

If you experience unexpected termination when running Fusion, first look in the *log files* for clues.

One setting you can look into is in **$FUSION_HOME/conf/fusion.cors**: `default.supervision.pollingFailureCountThreshold`.

By default, `pollingFailureCountThreshold` is set to `1`, so the Agent restarts all services the second time it fails to reach a service. Try a modest increase, for example set `pollingFailureCountThreshold` to `3`.

Log file names that start with "oom" indicate out-of-memory problems. You might need to increase the amount of memory allotted to that service. The amount of memory allotted to each kind of Fusion service is controlled by environment variables that are set in the `fusion.cors` (`fusion.properties` in Fusion 4.x) file.

Troubleshoot a Windows Install

Check common Windows service install script mistakes:

- Is the account trying run the install script a poweruser/administrator of the server?
- Is the DOMAIN\USERNAME correctly specified? Is the Domain correct?
- Is `java` installed on the `%PATH%`? To use a different Java, specify it in `bin/windows-service-wrapper.xml`.
- Are there any obvious issues in `var/log/windows-service-wrapper.log`?
- Does Fusion start from the normal `bin\fusion start`?
- Are there any other errors in the normal logs?

Increase memory

One other thing that can happen if you have not changed any of the default settings is for the services to run out of memory under heavy load, causing the program to crash.

To find out if this happened, you can check for the presence of any files matching the pattern **oom_killer-**.log* in the log directory for the service that is being restarted, for example, **$FUSION_HOME/var/log/connectors-classic** if that is the one being restarted.

If this is your issue, the first step is to increase the memory of the affected component by modifying `conf/fusion.cors` (`fusion.properties` in Fusion 4.x). Go to the `jvmOptions` for the service in question and change the value of the `-Xmx` flag. By default you will see something like:

```
connectors-classic.jvmOptions = -Xmx1g -Xss256k -
Dcom.lucidworks.connectors.pipelines.embedded=false
```

`-Xmx1g` means this service will fail if it needs more than 1 gigabyte of memory to operate. Increase this memory, for example set the flag to `Xmx4g` for an additional 3 gigabytes. 1g can be very low for some workloads.

More Help and Support

- Lucidworks Fusion Knowledge Base
- Contact Lucidworks

Troubleshooting Apache Pulsar Issues

How-to

> Compatible versions: 5.5
>
> Works with: *Fusion*

Note

See *Apache Pulsar* for general information and frequently asked questions.

Troubleshooting Apache Pulsar in Fusion

Clear Pulsar data

Use this procedure to clear a `PersistentVolumeClaim` (PVC) in Pulsar bookkeeper data and re-establish services.

Note

The *Purge Pulsar data completely* section explains how to proceed if the procedure to correct an *Invalid cookie exception* is not successful.

1. Execute the following command to obtain the number of existing replicas of statefulsets (STS) for both Pulsar Bookkeeper and Broker:

```
BOOKKEEPER_REPLICAS=$(kubectl get sts -l
"app=pulsar,component=bookkeeper" -o jsonpath="
{.items[0].spec.replicas}" ; echo) BROKER_REPLICAS=$(kubectl get
sts -l "app=pulsar,component=broker" -o jsonpath="
{.items[0].spec.replicas}" ; echo)
```

2. Execute the following command to scale down the STS:

```
kubectl get sts -l "app=pulsar" --no-headers | awk '{print $1}'
| xargs kubectl scale --replicas=0 sts
```

3. Execute the following command to delete the Pulsar PVCs:

```
kubectl get pvc -l "app=pulsar" --no-headers | awk '{print $1}'
| xargs kubectl delete pvc
```

Note

When this command is run in Google Kubernetes Engine (GKE), the PersistentVolumes(PVs) associated with the PVCs are also deleted. If you are working in a non-GKE environment, run the command to determine if the PVs associated with the PVCs are deleted. If the PVs are not deleted, you *must* delete them before proceeding. If the PVs are not deleted, the PVs and their data are reused, which causes unpredictable behavior when the system is restarted.

4. Sign in to Zookeeper and execute the following command:

```
ZOOKEEPER_NODE=$(kubectl get pods -l "app=zookeeper" --no-
headers | awk '{print $1}' | head -n1) kubectl exec -it
$ZOOKEEPER_NODE /bin/bash
```

5. Execute the following commands in Zookeeper to run the `zzkCli.sh` script and delete Pulsar information:

```
cd bin zkCli.sh deleteall /pulsar quit exit
```

6. Execute the following commands to scale the `pulsar-bookkeeper` replica count to its previous numbers:

```
kubectl get sts -l "app=pulsar,component=bookkeeper" --no-
headers | awk '{print $1}' | xargs kubectl scale --
replicas=$BOOKKEEPER_REPLICAS sts
```

7. When at least one `pulsar-bookkeeper` created from the previous command is successfully running, execute the following command to scale the `pulsar-broker` replica count to its previous numbers:

```
kubectl get sts -l "app=pulsar,component=broker" --no-headers |
awk '{print $1}' | xargs kubectl scale --
replicas=$BROKER_REPLICAS sts
```

8. When at least one `pulsar-broker` created from the previous command is successfully running, sign in to that `pulsar-broker` instance and execute the following commands:

```
BROKER_NODE=$(kubectl get pods -l "app=pulsar,component=broker"
--no-headers | grep '1/1' | grep Running | head -n1 | awk
'{print $1}') kubectl exec -it $BROKER_NODE /bin/bash cd bin
```

9. While still signed into the `pulsar-broker` pod, execute the following commands to create a:
   - Tenant with the same name as the current Kubernetes namespace
   - Namespace inside the new tenant named `_logs`
   - Topic inside the new namespace named `system_logs`

   Note

   In the commands, replace $(NAMESPACE) with the name of your current Kubernetes namespace.

   Examples

   ```
   ./pulsar-admin tenants create $(NAMESPACE) ./pulsar-
   admin namespaces create $(NAMESPACE)/_logs ./pulsar-
   admin topics create
   persistent://$(NAMESPACE)/_logs/system_logs exit
   ```

   For example, if the namespace is `dev`, the command is:

   ```
   ./pulsar-admin tenants create dev ./pulsar-admin
   namespaces create dev/_logs ./pulsar-admin topics create
   persistent://dev/_logs/system_logs exit
   ```

10. Execute the following command to restart all applicable deployments:

```
kubectl get deployments --no-headers | awk '{print $1}' |
kubectl rollout restart deployment
```

Invalid cookie exception

The `org.apache.bookkeeper.bookie.BookieException$InvalidCookieException` error is typically generated when corrupted or lost data exists on the persistent volumes mounted on bookkeeper nodes.

1. Verify the `managedLedgerDefaultWriteQuorum` field contains an adequate number of bookkeepers for a write quorum. The default value is **2**. If necessary, increase the value.
2. Execute the following command to add more bookkeeper nodes to scale up the `pulsar-bookkeeper` STS:

```
kubectl scale -n example example-pulsar-bookkeeper --replicas 4
```

Note

If the STS cannot scale up, one of the pods in sequence may be failing to start. Temporarily disable the readiness probe and increase the value of `initialDelaySeconds` on the liveness probe. This allows time so the failing probe does not prevent the next pod creation.

3. When a sufficient number of **healthy** bookkeeper nodes exist for a quorum, execute the following command to obtain the failing bookkeeper cookie IDs from Zookeeper:

```
kubectl exec -n example example-zookeeper-0 -- bin/zkCli.sh ls
/pulsar/ledgers/cookies
```

The command returns a list of all bookkeeper IDs registered with the cluster. For example:

```
$ kubectl exec -n example example-zookeeper-0 -- bin/zkCli.sh ls
/pulsar/ledgers/cookies [example-pulsar-bookkeeper-0.example-
pulsar-bookkeeper.example.svc.cluster.local:3181, example-
pulsar-bookkeeper-1.example-pulsar-
bookkeeper.example.svc.cluster.local:3181, example-pulsar-
bookkeeper-2.example-pulsar-
bookkeeper.example.svc.cluster.local:3181]
```

4. Execute the following command to decommission each of the corrupted bookkeeper nodes:

```
decommissionbookie
```

For example, if the **`example-pulsar-bookkeeper-1`** node is corrupt, execute the following command:

```
kubectl exec -n example example-pulsar-bookkeeper-0 --
/pulsar/bin/bookkeeper shell decommissionbookie -bookieid
example-pulsar-bookkeeper-1.example-pulsar-
bookkeeper.example.svc.cluster.local:3181
```

Note

The command *must* finish before executing the command on another corrupt node.

5. Restore the readiness probe and liveness probe to their original state if they were removed or modified earlier in this procedure.
6. Execute the following command to delete the each of the failing pods that were decommissioned. For example:

```
kubectl delete pod -n example example-pulsar-bookkeeper-1
```

Important

You must wait until each of the deleted pods are replaced and the new pods are in a "healthy" state.

7. If necessary, set the value of the `pulsar-bookkeeper` STS to its original size.

Note

If any additional nodes are removed, follow the steps of this procedure to decommission and delete them.

Purge Pulsar data completely

If the procedure defined in *Invalid cookie exception* cannot be completed, and there are **no concerns about losing existing Pulsar metadata**, use this procedure to purge Pulsar Zookeeper data.

Warning

Downtime is required to purge Pulsar metadata completely.

1. Execute the following commands to scale down the `pulsar-broker` and `pulsar-bookkeeper` STS to zero replicas:

```
kubectl scale -n example sts/example-pulsar-bookkeeper --
replicas=0 kubectl scale -n example sts/example-pulsar-broker --
replicas=0
```

Important

You must wait until all of the pods are deleted before proceeding.

2. Execute the following command to list all of the PVCs for `pulsar-bookkeeper` pods:

```
kubectl get pvc -n example -o name | grep pulsar-bookkeeper
```

3. Verify the correct bookkeeper PVCs are listed for your cluster/namespace.
4. Execute the following command to delete each of the PVCs owned by previously-existing `pulsar-bookkeeper` pods:

```
kubectl get pvc -n example -o name | grep pulsar-bookkeeper |
xargs -I {} kubectl delete -n example {}
```

Note

Repeat this command for each PVC to delete them one by one.

5. Execute the following command to delete all nodes under the `/pulsar` path in Zookeeper:

```
kubectl exec -n example example-zookeeper-0 -- bin/zkCli.sh
deleteall /pulsar
```

6. Execute the following commands to scale the `pulsar-broker` and `pulsar-bookkeeper` STS to the appropriate number of replicas:

```
kubectl scale -n example sts/example-pulsar-bookkeeper --
replicas=3 kubectl scale -n example sts/example-pulsar-broker --
replicas=2
```

Important

You must wait until all of the pods are scheduled and ready before proceeding.

7. Restart all remaining Fusion services pods *except* `zookeeper`, `solr`, `pulsar-broker`, and `pulsar-bookkeeper`. Based on the Fusion deployment customization for your system, there may be other pods that should *not* be restarted.

Important

The best practice recommendation is to restart services individually, one at a time. And Fusion deployment specifics must be taken into account before deleting pods. If you determine you want to delete **all** pods in a namespace, execute the following command: `kubectl get pods -n example -o name | grep -v "zookeeper\|solr\|pulsar" | xargs -I {} kubectl delete -n example {}`.

OutOfDirectMemoryError

The bookkeeper pod can exit due to an `OutOfDirectMemoryError`, which is typically generated in clusters with high Pulsar throughput and/or slow disk input/output (I/O). For example:

```
[bookie-io-1-1] ERROR org.apache.bookkeeper.proto.BookieServer -
Unable to allocate memory, exiting bookie
io.netty.util.internal.OutOfDirectMemoryError: failed to allocate
16777216 byte(s) of direct memory (used: 2919235615, max: 2936012800)
```

Bookkeeper uses direct memory to cache data before it is written to disk.

To allocate more direct memory based on the *expected* highest network throughput relative to the disk I/O:

1. Access the `bookkeeper.configData.PULSAR_MEM` helm parameter.
2. Add `-XX:MaxDirectMemorySize` with the appropriate value.
   For example, `-XX:MaxDirectMemorySize=4G` allocates 4 gigabytes of direct memory.

Troubleshooting Appkit App Builds and Deployments

How-to

> Compatible versions: 4.2, 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *App Studio*

Troubleshoot building and deploying Appkit apps.

Increasing memory allocation to Jetty when running with Maven

If you are using the Social Module with Appkit, you might have to increase the amount of memory available to the application server. When starting Appkit up with Maven set this environment variable:

```
export MAVEN_OPTS='-Xms128M -Xmx256M -XX:PermSize=128M -
XX:MaxPermSize=256M'
```

Paths with encoded forward slashes not working in Tomcat

By default, specific versions of Tomcat do not allow encoded forward slashes (%2F) in a URL. To work around this problem set this System property:

```
-Dorg.apache.tomcat.util.buf.UDecoder.ALLOW_ENCODED_SLASH=true
```

This can be done either via catalina.sh or catalina.bat or on Windows via the services properties window, which can be accessed by running `$CATALINA_HOME/bin/tomcat7w.exe` (if using Tomcat 7, change as appropriate for your exact version).

Problem with accented and other UTF-8 characters

Appkit is completely stateless so all information for a Query is contained in the HTTP request. Because Tomcat by default is not configured to handle special characters (such as UTF-8) in URLs, Tomcat might be incorrectly interpreting the HTTP GET request. To correctly handle URLs, open the Tomcat server.xml file `[tomcat installation folder]/conf/server.xml`, find the <Connector> element, and add the URIEncoding attribute to it:

```
<Connector port="8080" protocol="HTTP/1.1" connectionTimeout="20000"
redirectPort="8443" URIEncoding="UTF-8" />
```

For more information, see the Character Encoding FAQ on the Tomcat Wiki.

If you are running Tomcat behind an HTTP server via AJP (Apache or IIS, for example), then you also must add the URIEncoding attribute to the AJP connector found in the Tomcat server.xml file:

```
<Connector port="8009" protocol="AJP/1.3" redirectPort="8443"
URIEncoding="UTF-8" />
```

Error writing to Java Preference Store

In specific cases on Linux, there is an issue with the application server being able to write temporary and other system files. In that case, start up the container with this flag:

```
-Djava.util.prefs.userRoot=/some/writable/directory
```

IIS authentication with Tomcat

When using IIS as an authentication layer in front of Tomcat, the AJP connector must be told not to perform authentication in Tomcat:

```
<Connector port="8009" protocol="AJP/1.3" redirectPort="8443"
tomcatAuthentication="false" />
```

Long Tomcat startup times

There is a known issue with long Tomcat startup times. To avoid this, add this entry to the JVM params, applied in `/usr/share/tomcat7/bin/catalina.sh`:

```
JAVA_OPTS=(all your usual stuff plus:)-
Djava.security.egd=file:/dev/./urandom
```

JVM proxy settings

You can configure your JVM to use an HTTP proxy for all outgoing network traffic by setting these system properties:

```
-Dhttp.proxyHost=${server} -Dhttp.proxyPort=${port} -
Dhttp.proxyUser=${username} -Dhttp.proxyPassword=${password}
```

When using Tomcat, you can configure proxy settings either via catalina.sh or catalina.bat or on Windows via the services properties window, which can be accessed by running `$CATALINA_HOME/bin/tomcat7w.exe` (if using Tomcat 7, change as appropriate for your exact version).

Installing Missing Image Codecs on a Windows server

The standard SUN JDK only ships with a limited number of image codecs, which are needed for manipulating images in Appkit (for example, when resizing to generate a preview). In particular, the standard JDK lacks support for TIFF

images.

To obtain the missing codecs, we recommend installing the Java Advanced Imaging Image I/O Tools, which are available free of charge from:

http://download.java.net/media/jai-imageio/builds/release/1.1/INSTALL-jai_imageio.html#Windows

Down the page, you should download the JDK version of the toolkit and copy it to each delivery server that will be running Appkit:

http://download.java.net/media/jai-imageio/builds/release/1.1/jai_imageio-1_1-lib-windows-i586-jdk.exe

Follow the installation instructions for the toolkit.

Note

For Windows users, the installation must be performed with a 32-bit JDK.*

After it has been installed, you must restart Tomcat to get the new codecs into the Java runtime. To verify which codecs are installed on the server, look for this output in Appkit.log when Tomcat starts again:

```
2013-05-23 09:16:08,146 [main] INFO twigkit.media.MediaModule -
Supported image formats: 2013-05-23 09:16:08,146 [main] INFO
twigkit.media.MediaModule - - BMP 2013-05-23 09:16:08,146 [main] INFO
twigkit.media.MediaModule - - bmp 2013-05-23 09:16:08,146 [main] INFO
twigkit.media.MediaModule - - jpg 2013-05-23 09:16:08,146 [main] INFO
twigkit.media.MediaModule - - JPG 2013-05-23 09:16:08,146 [main] INFO
twigkit.media.MediaModule - - jpeg 2013-05-23 09:16:08,146 [main] INFO
twigkit.media.MediaModule - - wbmp 2013-05-23 09:16:08,146 [main] INFO
twigkit.media.MediaModule - - png 2013-05-23 09:16:08,146 [main] INFO
twigkit.media.MediaModule - - PNG 2013-05-23 09:16:08,146 [main] INFO
twigkit.media.MediaModule - - JPEG 2013-05-23 09:16:08,146 [main] INFO
twigkit.media.MediaModule - - WBMP 2013-05-23 09:16:08,146 [main] INFO
twigkit.media.MediaModule - - TIFF ...
```

Missing web resources

Under some circumstances web resources such as fonts (.ttf/.woff etc.) cannot be retrieved resulting in missing icons in the UI. In many cases this is caused by a bug in the webJAR locator library affecting some Operating Systems. Installing Tomcat on a path containing blanks causes this bug to manifest itself. The fix is simply to move Tomcat to a blank-free location (for example, on Windows in `C:\Tomcat\`).

Tune a Couchbase datasource

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

The Couchbase V2 connector uses the Couchbase Java client to retrieve data stored in Couchbase.

Because the Couchbase connector retrieves data continuously, two properties are available to control the frequency of commits to Solr, which makes the documents available for user queries.

The properties define the maximum number of documents to queue for a commit (set to 50,000 by default) and the maximum amount of time to wait between commits (set to 120 seconds, or 2 minutes).

Documents will be committed when one of those thresholds is reached first, meaning that if 2 minutes have passed and there are only 20,000 documents, a commit will occur. Similarly, if only 1 minute has passed and there are 50,000 documents in the queue, a commit will occur. These properties can be adjusted for your own requirements if needed.

Tune a Couchbase Datasource

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion Connectors*, *Fusion Server*, *Fusion*

The Couchbase V1 connector uses the Cross-Datacenter Replication (XDCR) feature of Couchbase to retrieve data stored in Couchbase continuously in real-time.

Because the Couchbase connector retrieves data continuously, two properties are available to control the frequency of commits to Solr, which makes the documents available for user queries. The properties define the maximum number of documents to queue for a commit (set to 50,000 by default) and the maximum amount of time to wait between commits (set to 120 seconds, or 2 minutes). Documents will be committed when one of those thresholds is reached first, meaning that if 2 minutes have passed and there are only 20,000 documents, a commit will occur. Similarly, if only 1 minute has passed and there are 50,000 documents in the queue, a commit will occur. These properties can be adjusted for your own requirements if needed.

Update Your Data

How-to

> Works with: *Never Null*

Once your *initial data upload and analysis* are complete, you need to keep your Never Null up to date by periodically sending incremental updates. Like the initial data upload, you can upload incremental updates to a secure data bucket and provide access for Lucidworks.

How to update your Never Null data

1. Confirm that your product data is formatted according to the *product data requirements*.
2. Confirm that your signals data is formatted according to the *signals data requirements*.
3. Upload it to a secure data bucket, such as GCS or S3.
4. Provide Lucidworks with access to the data bucket.

Upgrade App Studio

How-to

> Compatible versions: 4.2
>
> Works with: *App Studio*

When you upgrade Fusion Server, that also upgrades App Studio to its latest version, and its embedded Appkit to the Appkit version in the App Studio release. The App Studio version might differ from the Fusion Server version. Not all Fusion Server upgrades have a new version of App Studio.

Note

An App Studio release might not contain the latest version of Appkit. Check the *App Studio Release Notes* and *Appkit Release Notes* to confirm the versions. To upgrade a new app to a more recent version of Appkit than the version that an App Studio release contains, create the app and then *Upgrade Appkit in existing apps*.

The App Studio upgrade does not upgrade the version of Appkit used in existing apps. For that procedure, see *Upgrade Appkit in existing apps*.

Upgrade Appkit in Existing Apps

How-to

> Compatible versions: 4.2
>
> Works with: *App Studio*

When you create an Appkit webapp, it uses a specific version of Appkit. Appkit releases occur independently of App Studio releases.

You can upgrade web apps to use the latest version of Appkit.

How to upgrade

1. Stop the app.
2. In the `package.json` file at the root of your project, update the `appkit-ui` dependency to the latest Appkit version, for example:

```
"appkit-ui": "http://appkit.lucidworks.com/repo/4.2.0/appkit-ui.tar.gz",
```

3. In the `pom.xml` file at the root of your project, update the value of the `parent.version` property to the latest Appkit version, for example:

```
<parent> <groupId>twigkit</groupId>
<artifactId>twigkit.app.js</artifactId> <version>4.2.0</version>
</parent>
```

4. *Upgrade the Appkit Social module*, if required.
5. Perform any additional upgrade steps that are required for a specific Appkit upgrade. See the *Appkit Release Notes* for more information.
6. Start the app.

Upgrade Fusion Server 3.1.x to 4.2.y

How-to

> Compatible versions: 4.2
>
> Works with: *Fusion Server*

Introduction

This article describes how to perform the following upgrade:

- From version: Fusion `3.1.x`
- To version: Fusion `4.2.y`

Important

Only specific version-to-version upgrade sequences are supported. Some upgrades require multiple steps.

For more information, see *Fusion 4.x*.

For Fusion 3.1 and later releases, a migrator is available for upgrading Fusion.

During the upgrade process, the migrator uses a properties file. After downloading and installing the migrator, the properties files is in the `/opt/lucidworks/fusion/3.1.x/var/upgrade` directory (on Unix or MacOS) or the `C:\lucidworks\fusion\3.1.x\var\upgrade\` directory (on Windows). The file names reference the versions you are upgrading from and to. For example:

- To upgrade `3.1.5` to `4.2.5`, the migrator uses the `3.1.x-4.2.x.properties` file.

Migration entails down time and multiple starts and stops of Fusion services. Plan accordingly, especially in terms of disabling external load balancers or monitors that might react adversely to the starts and stops.

Download the latest migrator immediately before upgrading. This helps ensure that the upgrade goes smoothly.

Important

The newer Fusion instance must be newly untarred and *never started*.

About the upgrade

This section describes how connectors, object migrations, and signals are migrated during an upgrade.

Connectors

In Fusion 3.1.0 and later, only a subset of *connectors* are included by default.

The migrator detects which connectors were used in the older version of Fusion, and installs them automatically in Fusion `4.2.y`. This step requires an Internet connection. If no connection is available, then download the connectors from *Fusion 4.x Connector Downloads* and *install them as bootstrap plugins*.

If a connector to be upgraded was not available during the upgrade, then a message in `/opt/lucidworks/fusion/3.1.x/var/upgrade/tmp/migrator.log` (on Unix) or `C:\lucidworks\fusion\3.1.x\var\upgrade\tmp\migrator.log` (on Windows) indicates this.

Only datasources for connectors that are supported in the new Fusion version are upgraded. Datasources for custom connectors are not upgraded.

IF NO INTERNET CONNECTION IS AVAILABLE

If no Internet connection is available during an upgrade, the migrator cannot automatically download the connectors it needs. Using the Fusion UI or API later to install the connectors also might not be an option.

In this case, download the connectors from *Fusion 4.x Connector Downloads* for all existing connectors and place them in `apps/connectors/bootstrap-plugins` for the new deployment (on all Fusion nodes). Do so at the time indicated in the procedures that follow.

ADDING CONNECTORS DURING AN UPGRADE

You can *add* connectors during an upgrade (that is, add connectors that are not in the old deployment).

Download the connectors from *Fusion 4.x Connector Downloads* and place them in `apps/connectors/bootstrap-plugins` for the new version (on all Fusion nodes).

Object migrations and transformations

The migrator upgrades all Fusion 3.1 object types:

- Collections
- Index pipelines
- Query pipelines
- Search cluster configurations
- Schedules
- Aggregations
- Datasources
- Dashboards
- Parsing configurations
- Object groups
- Links
- Tasks
- Jobs
- Spark objects

The migrator adds these Fusion Server object types:

- Apps
- Appkit apps
- Index profiles
- Query profiles
- Blobs

Important

In Fusion Server 4.0 and later, most objects exist in the context of apps. When you upgrade from Fusion 3.1.x to Fusion Server 4.2.y, the migrator creates the app `default` and places object in that app or links them to it, as needed. Some objects are not linked to apps. You can explore objects in *Object Explorer*.

Noteworthy transformations

While migrating objects, the migrator makes these transformations:

- It adds the new `log-shipper` service to `fusion.properties`.
- It removes obsolete log-cleanup tasks.
- It converts the format of job history records.

Migration of index profiles and query profiles

In Fusion Server 4.0, index profiles and query profiles are objects. They have capabilities that exceed those of index profiles and query profiles in prior releases.

- **Prior to Fusion 4.0.** A single index profile could reference multiple index pipelines. A single query profile could reference multiple query pipelines.
- **In Fusion 4.0 and later.** A single index profile can reference a single index pipeline. A single query profile can reference a single query pipeline.

During migration from 3.1.x to 4.0.y, the migrator upgrades index profiles and query profiles as follows:

- **Index profiles.**
  - If a 3.1.x index profile contains a reference to the index pipeline `default`, then the migrated profile retains that single reference.
  - If a 3.1.x index profile does not contain a reference to the index pipeline `default`, then the migrated profile references an index pipeline that has the same name as the index

profile.

- **Query profiles.**
  - If a 3.1.x query profile contains a reference to the query pipeline `default`, then the migrated profile retains that single reference.
  - If a 3.1.x query profile does not contain a reference to the query pipeline `default`, then the migrated profile references a query pipeline that has the same name as the query profile.

After migration, you might need to adjust the pipeline references of index profiles and query profiles by hand, and/or create new index profiles and query profiles.

Access control migration

The migrator upgrades all access control configurations:

- **Security realms.** Security realms do not require adjustments after migration.
- **Roles.** System-created roles do not require adjustments after migration. User-created roles probably require adjustments after migration.
- **Users.** System-created users do not require adjustments after migration. User-created users probably require adjustments after migration.

Migrations that should not require adjustments

This section describes fully automatic parts of migration. You should not have to adjust these items after migration.

The migrator performs these migration tasks:

- It migrates encrypted passwords, so users can log in with the same credentials they used on the older version of Fusion or Fusion Server.
- It adds the user `webapps-system-account`. Fusion Server uses this account. You do not need to do anything with it.
- It adds these permissions to the role `developer`:

```
GET:/license GET,POST,PUT:/appkit GET,POST,PUT:/apps/**
```

- Updates these permissions for the role `search`:

```
POST:/apps/*/signals/** GET:/query/** GET:/apps/*/query/**
POST:/signals/**
```

- Adds the role `webapps-role` with these permissions:

```
GET,HEAD:/webapps/** GET,HEAD:/license
```

Migrations that might require adjustments

This section describes parts of migration for which you might need to make adjustments after migration.

Important

Following migration, we recommend that you review the API and UI permissions for roles, and the roles and API permissions for users.

- If you have created your own developer roles, add these permissions to the roles:

```
GET:/license GET,POST,PUT:/appkit GET,POST,PUT:/apps/**
```

- If you have created your own search roles, add these permissions to the roles:

```
POST:/apps/*/signals/** GET:/query/** GET:/apps/*/query/**
POST:/signals/**
```

Signals and Spark jobs

Fusion Server 4.2.***y*** indexes search log events into the `COLLECTION_NAME_signals` collections as `response` signals, instead of indexing them into the `COLLECTION_NAME_logs` collections.

Roughly 90 new fields used by App Insights are added to existing `COLLECTION_NAME_signals` collections.

You are encouraged to adopt the new signal fields, but you can continue using the old dynamic field names, such as `user_id_s`, until Fusion 5.0. If you adopt the new signals schema, then you must update any Spark jobs that rely on the old field names.

Fusion Server 4.2.***y*** replaces the `_signals_ingest` index pipeline with a new version that works with the new signals schema.

If you made changes to the `_signals_ingest` pipeline, then you will need to manually add those changes to the new configuration after migration.

The migrator preserves legacy-style aggregation jobs. Optionally, you can manually convert these to SQL-based aggregation jobs.

Fusion Server 4.2.***y*** adds a new session rollup job for each collection with signals enabled. The session rollup job creates session signals that contain aggregated information about user activity in a session.

Fusion Server 4.2.***y*** adds a new head/tail analysis job for each collection with signals enabled. The head/tail analysis job uses signals to compute interesting metrics for head-and-tail queries.

Review known issues

Before upgrading, *review the known issues* to see whether any of them apply to the circumstances of your upgrade. Some known issues might require actions *before* upgrading.

That article also contains instructions regarding what to do *if an upgrade step fails*.

Upgrade on Unix

Use this procedure to upgrade Fusion on a single Unix node or on multiple Unix nodes.

Perform the steps in this procedure on the indicated nodes *on which Fusion is running* ("Fusion nodes"). To perform an upgrade, Fusion nodes must have at least these services running:

- API service (`api`)
- Fusion UI service (`ui`)

Important

For every step on multiple nodes, ensure that the step completes *on all Fusion nodes* before going to the next step. There is the notion of a "main node" during the migration process. This node will be used for certain centralized migration activities that do not need to be done on every node, such as downloading connectors that are then uploaded to blob storage that is shared by all, etc. Just pick one of your Fusion nodes to be the "main node"; there is no special requirement as to which one you pick.

Install the Java JDK

Before upgrading to Fusion Server 4.2 or later, you must install the Java JDK on all nodes that will run Fusion Server. Prior versions of Fusion Server could use either the JRE or the JDK.

For more information, see *Java requirements*.

Download and install the newer version of Fusion

Perform these tasks *on all Fusion nodes*:

1. Select the Fusion release to which you are upgrading from *Fusion Server 4.x File Download Links*.
2. Extract the newer version of Fusion:

   ```
   cd /opt/lucidworks mv ~/Downloads/fusion-4.2.y.tar.gz ./ tar -xf fusion-4.2.y.tar.gz
   ```

   For example, if Fusion is currently installed in `/opt/lucidworks/fusion/3.1.x` , then change your working directory to `/opt/lucidworks/` and extract the file there. *do not run the new version of Fusion yet.*
3. Ensure that the *new* version of Fusion has a valid permanent Fusion license before proceeding with the upgrade. Place a valid `license.properties` file in the `/opt/lucidworks/fusion/4.2.y/conf` directory.
4. (*If there are custom* `jar` *files*) If your deployment has custom `jar` files, add them to the new Fusion deployment.
5. (*If you are performing an upgrade without Internet access*) Without Internet access, the migrator cannot download new versions of connectors automatically. Download the *new versions* of connector zip files *for your current connectors* from *Fusion 4.x Connector Downloads* and place them in `apps/connectors/bootstrap-plugins` for the new deployment.
6. (*If you are adding* ***new*** *connectors*) If you want your new deployment to use connectors that *are not* in the current deployment, you can add them now. Download the connector zip files from *Fusion 4.x Connector Downloads* and place them in `apps/connectors/bootstrap-plugins` for the new deployment.
7. Verify that there is sufficient disk space for a second copy of the Solr index directory, `fusion/3.1.x/data/solr` . If there is not sufficient disc space, free up space before proceeding.

Download and install the Fusion migrator

Perform these tasks *on all Fusion nodes*:

1. Download the latest migrator zip file for Unix . (Do this now, even if you have downloaded the migrator before, to ensure that you have the latest version.)
2. Create `FUSION_OLD` and `FUSION_NEW` environment variables that point to the old and new Fusion installation directories respectively (using the full path).

```
export FUSION_OLD="/opt/lucidworks/fusion/3.1.x" export
FUSION_NEW="/opt/lucidworks/fusion/4.2.y"
```

   For example, when upgrading from Fusion 3.1.5 to 4.2.6:

```
export FUSION_OLD="/opt/lucidworks/fusion/3.1.5" export
FUSION_NEW="/opt/lucidworks/fusion/4.2.6"
```

3. Create this directory:

```
mkdir $FUSION_OLD/var/upgrade
```

4. Install the migrator:

```
tar -zxv -C $FUSION_OLD/var/upgrade --strip-components=1 -f
fusion-migrator.tar.gz
```

Run the migrator

Perform these tasks on the indicated nodes:

1. (*On all Fusion nodes*) Start all Fusion services for the old version of Fusion:

```
$FUSION_OLD/bin/fusion start
```

2. (*Only on the main Fusion node*) Run the migrator to export the configuration data from the old version of Fusion:

```
java -jar $FUSION_OLD/var/upgrade/migrator.jar --export
```

This message indicates that the step finished successfully:

```
Old Fusion configuration export (--export) finished
successfully.
```

3. (*On all Fusion nodes*) Stop the old versions of Fusion services and Solr; but not ZooKeeper:

```
$FUSION_OLD/bin/ui stop $FUSION_OLD/bin/connectors stop
$FUSION_OLD/bin/api stop $FUSION_OLD/bin/solr stop
```

Tip

You can see what is running with `$FUSION_OLD/bin/fusion status`.

4. (*Only on secondary Fusion nodes*) Prepare secondary nodes:

```
java -jar $FUSION_OLD/var/upgrade/migrator.jar --prepare-
secondary
```

This message indicates that the step finished successfully:

```
Prepare secondary nodes (--prepare-secondary) finished
successfully.
```

5. (*On all Fusion nodes*) Stop ZooKeeper for the old version of Fusion (*unless you are using an external ZooKeeper instance, in which case you can ignore this step*):

```
$FUSION_OLD/bin/zookeeper stop
```

6. (*Only on the main Fusion node*) Transform configuration data on the main Fusion node:

```
java -jar $FUSION_OLD/var/upgrade/migrator.jar --main-transform
```

Note

Depending on the size of your Solr index, this step can take a long time (for example, multiple tens of minutes).

This message indicates that the step finished successfully:

```
Fusion data transformations on main node (--main-transform)
finished successfully.
```

7. (*On all Fusion nodes*) Start ZooKeeper for the new version of Fusion (*unless you are using an external ZooKeeper instance, in which case you can ignore this step*):

```
$FUSION_NEW/bin/zookeeper start
```

8. (*Only on the main Fusion node*) Import the first part of configuration data into the new version of Fusion:

```
java -jar $FUSION_OLD/var/upgrade/migrator.jar --zookeeper-
import
```

This message indicates that the step finished successfully:

```
New Fusion Zookeeper import (--zookeeper-import) finished
successfully.
```

9. (*If upgrading from Fusion 3.1.0 through 3.1.3; On all Fusion nodes*) Start Solr for the new Fusion version:

```
$FUSION_NEW/bin/solr start
```

10. (*If upgrading from Fusion 3.1.0 through 3.1.3; Only on the main Fusion node*) Run a script to remove all old plugins from the blob store. Replace `solr-address` and `solr-port` as appropriate (as shown in the example):

```
java -cp "$FUSION_OLD/var/upgrade/jython-standalone-
2.7.1.jar:$FUSION_OLD/var/upgrade/migrator.jar"
org.python.util.jython
"$FUSION_OLD/var/upgrade/transformations/manual_delete_old_plugi
n_blobs.py" --solr-address solr-address --solr-port solr-port
```

For example:

```
java -cp "$FUSION_OLD/var/upgrade/jython-standalone-
2.7.1.jar:$FUSION_OLD/var/upgrade/migrator.jar"
org.python.util.jython
"$FUSION_OLD/var/upgrade/transformations/manual_delete_old_plugi
n_blobs.py" --solr-address localhost --solr-port 8983
```

This message indicates that plugins were deleted successfully:

```
Deleted old plugin blobs from solr <?xml version="1.0"
encoding="UTF-8"?> <response> <lst name="responseHeader"> <int
name="status">0</int> <int name="QTime">246</int> </lst>
</response> Old connector plugin blobs were deleted
successfully.
```

11. (*On all Fusion nodes*) Start all Fusion services for the new version of Fusion:

```
$FUSION_NEW/bin/fusion start
```

12. (*Only on the main Fusion node*) Import the second part of configuration data into the new version of Fusion:

```
java -jar $FUSION_OLD/var/upgrade/migrator.jar --fusion-import
```

This message indicates that the step finished successfully:

```
New Fusion object import (--fusion-import) finished
successfully.
```

Tip

After migration, you can find details about the results in the `fusion/3.1.x/var/upgrade/tmp` directory. If the migration produces unexpected results, the files in this directory are helpful for troubleshooting.

Validate the new version of Fusion

How to validate the new version of Fusion

1. (*Only on the main Fusion node*) Restart the new version of Fusion (all services defined in `fusion.properties`):

```
$FUSION_NEW/bin/fusion restart
```

2. Log into the Fusion UI (your `admin` password is the same as for the old installation), and confirm the release number of the new version of Fusion:
    a. Clear your browser's cache.
    Otherwise, you might inadvertently access a cached version of the old Fusion UI and see inconsistent behavior.
    b. In a browser, open the Fusion UI at `http://localhost:8764/` (replace `localhost` with your server name or IP address if necessary).
    c. Log in.
    d. Navigate to **Admin** > **About Fusion**.
    The **About Fusion** panel should display the newer Fusion release number.

3. Ensure that all connectors were installed automatically during the upgrade.
    - For Fusion 4.x from the Fusion launcher, click the tile for a migrated app. Click **System** > **Blobs**. If any connectors are missing from the list, click **Add** > **Connector Plugin** and install them manually.
    - For Fusion 3.x from the Fusion launcher, click **Devops** > **Home** > **Blobs**. If any connectors are missing from the list, click **Add** > **Connector Plugin** and install them manually.

4. Ensure that all customizations you made in the former version of Fusion are present in the new one.
5. If you migrated from version 3.1.x to version 4.2.0 or later, *confirm associations between blobs and apps* and make adjustments if necessary.
6. When you are satisfied with the migration and you have backed up the `fusion/3.1.x/` directory, you can `rm -fr fusion/3.1.x/` to remove the older version of Fusion (*on all Fusion nodes*).

Add support for business rules to existing apps

Fusion AI 4.2 introduces functionality for using business rules, that is, manually created formulas for rewriting queries and responses.

You can add support for business rules to apps that were created in versions of Fusion AI prior to version 4.2. To do so, perform the steps in this section.

The `add-rule-objects-xyz.zip` file (where `xyz` is a version number) specifies the objects to add to an app. It is supplied in the Fusion migrator zip file at the top level. After installing the migrator, the location is `$FUSION_OLD/var/upgrade/import-files`.

Note

Adding support for business rules has a costs. Additional collections and objects are created. Only add support for business rules to apps in which you plan to use them.

You have a choice. You can update each app using the *Fusion UI* or the *Fusion API*.

FUSION UI

For each app in which you plan to use business rules, import the objects in the `add-rule-objects-xyz.zip` file into the app.

How to import business rule objects

1. In the Fusion launcher, click the app into which you want to import objects.

The Fusion workspace appears.

2. In the upper left, click System > **Import Fusion Objects**.

   The Import Fusion Objects window opens.
3. For the data file, select `add-rule-objects-xyz.zip` from your local filesystem. The location in the extracted migrator files is `$FUSION_OLD/var/upgrade/import-files`.
4. Click **Import**.
5. Edit the `Application ID` parameter value to use the app name. If the app name contains spaces, replace those with underscore characters. For example, `Lucene Revolution` would become `Lucene_Revolution`.
6. Click **Import**.
7. If there are conflicts, Fusion prompts you to specify an import policy. Click **Merge** to skip all conflicting objects and import only the non-conflicting objects.
   Fusion confirms that the import was successful.
8. Click **Close** to close the Import Fusion Objects window.

FUSION API

For each app in which you plan to use business rules, import the objects in the `add-rule-objects-xyz.zip` file into the app.

How to import business rule objects

1. Create an `app-name.txt` file with the following content:

   ```
   {"app.id" : "_name-of-migrated-app_"}
   ```

   For example, for the app `Lucene Revolution`:

   ```
   {"app.id" : "Lucene_Revolution"}
   ```

   Here, we assume that you create the files in your home directory, for which the `$HOME` environment variable is defined.
2. Import the business rule objects:

   ```
   curl -u USERNAME:PASSWORD -H "Content-Type:multipart/form-data"
   -X POST -F 'variableValues=path-to-txt-file/app-name.txt' -F
   "importData=path-to-zip-file/add-rule-objects-xyz.zip"
   "http://<FUSION_HOST>/api/objects/import?importPolicy=merge"
   ```

   For example:

```
curl -u admin:Very-SecurePW8 -H "Content-Type:multipart/form-
data" -X POST -F 'variableValues=$HOME/Lucene_Revolution.txt' -F
"importData=$FUSION_OLD/var/upgrade/import-files/add-rule-
objects-001.zip" "http://<FUSION_HOST>/api/objects/import?
importPolicy=merge"
```

Upgrade on Windows

Use this procedure to upgrade Fusion on a single Windows node or multiple Windows nodes.

Perform the steps in this procedure on the indicated nodes *on which Fusion is running* ("Fusion nodes"). To perform an upgrade, Fusion nodes must have at least these services running:

- API service (`api`)
- Fusion UI service (`ui`)

Important

If you are upgrading Fusion on multiple nodes, then, for every step on multiple nodes, ensure that the step completes *on all Fusion nodes* before going to the next step. There is the notion of a "main node" during the migration process. This node will be used for certain centralized migration activities that do not need to be done on every node, such as downloading connectors that are then uploaded to blob storage that is shared by all, etc. Just pick one of your Fusion nodes to be the "main node"; there is no special requirement as to which one you pick.

Install the Java JDK

Before upgrading to Fusion Server 4.2 or later, you must install the Java JDK on all nodes that will run Fusion Server. Prior versions of Fusion Server could use either the JRE or the JDK.

For more information, see *Java requirements*.

Download and install the newer version of Fusion

Perform these tasks *on all Fusion nodes*:

1. Select the Fusion release to which you are upgrading from *Fusion Server 4.x File Download Links*.
2. Move the `fusion-4.2.y.zip` file to the directory that contains the `fusion\` directory. For example, if Fusion is installed in `C:\lucidworks\fusion\3.1.x`, then move the file to `C:\lucidworks`.
3. Unzip the `fusion-4.2.y.zip` file. *do not run the new version of Fusion yet.*
4. For Fusion 4.x, ensure that the *new* version of Fusion has a valid permanent Fusion license before proceeding with the upgrade. Place a valid `license.properties` file in the `C:\lucidworks\fusion\4.2.y\conf` directory.
5. (*If there are custom* `jar` *files*) If your deployment has custom `jar` files, add them to the new Fusion deployment.
6. (*If you are performing an upgrade without Internet access*) Without Internet access, the migrator cannot download new versions of connectors automatically. Download the *new versions* of connector zip files *for your current connectors* from *Fusion 4.x Connector Downloads* and place them in `apps\connectors\bootstrap-plugins` for the new deployment.
7. (*If you are adding* ***new*** *connectors*) If you want your new deployment to use connectors that *are not* in the current deployment, you can add them now. Download the connector zip files from *Fusion 4.x Connector*

Downloads and place them in `apps\connectors\bootstrap-plugins` for the new deployment.

8. Verify that there is sufficient disk space for a second copy of the Solr index directory, `fusion\3.1.x\data\solr`. If there is not sufficient disc space, free up space before proceeding.

Download and install the Fusion migrator

Perform these tasks *on all Fusion nodes*:

1. Download the latest migrator zip file for Windows . (Do this now, even if you have downloaded the migrator before, to ensure that you have the latest version.)
2. Open a Command Prompt window and create `FUSION_OLD` and `FUSION_NEW` environment variables that point to the old and new Fusion installation directories respectively. For example:

```
set FUSION_OLD=C:\lucidworks\fusion\3.1.5 set
FUSION_NEW=C:\lucidworks\fusion\4.2.6
```

3. Create a `fusion\3.1.x\var\upgrade` directory.
4. Unzip the migrator zip file, and move the contents of the extracted folder to `fusion\3.1.x\var\upgrade`.

Run the migrator

Perform these tasks on the indicated nodes:

1. (*On all Fusion nodes*) Start all Fusion services for the old version of Fusion:

```
%FUSION_OLD%\bin\fusion.cmd start
```

2. (*Only on the main Fusion node*) Run the migrator to export the configuration data from the old version of Fusion:

```
java -jar "%FUSION_OLD%\var\upgrade\migrator.jar" --export
```

This message indicates that the step finished successfully:

```
Old Fusion configuration export (--export) finished
successfully.
```

3. (*On all Fusion nodes*) Stop the old versions of Fusion services and Solr; but not ZooKeeper:

```
%FUSION_OLD%\bin\ui.cmd stop %FUSION_OLD%\bin\connectors.cmd
stop %FUSION_OLD%\bin\api.cmd stop %FUSION_OLD%\bin\solr.cmd
stop
```

Tip

You can see what is running with `%FUSION_OLD%\bin\fusion status`.

4. (*Only on secondary Fusion nodes*) Prepare secondary nodes:

```
java -jar "%FUSION_OLD%\var\upgrade\migrator.jar" --prepare-
secondary
```

This message indicates that the step finished successfully:

```
Prepare secondary nodes (--prepare-secondary) finished
successfully.
```

5. (*On all Fusion nodes*) Stop ZooKeeper for the old version of Fusion (*unless you are using an external ZooKeeper instance, in which case you can ignore this step*):

```
%FUSION_OLD%\bin\zookeeper.cmd stop
```

6. (*Only on the main Fusion node*) Transform configuration data on the main Fusion node:

```
java -jar "%FUSION_OLD%\var\upgrade\migrator.jar" --main-
transform
```

Note

Depending on the size of your Solr index, this step can take a long time (for example, multiple tens of minutes).

This message indicates that the step finished successfully:

```
Fusion data transformations on main node (--main-transform)
finished successfully.
```

7. (*On all Fusion nodes*) Start ZooKeeper for the new version of Fusion (*unless you are only using an external ZooKeeper instance, in which case you can ignore this step*):

```
%FUSION_NEW%\bin\zookeeper.cmd start
```

8. (*Only on the main Fusion node*) Import the first part of configuration data into the new version of Fusion:

```
java -jar "%FUSION_OLD%\var\upgrade\migrator.jar" --zookeeper-
import
```

This message indicates that the step finished successfully:

```
New Fusion Zookeeper import (--zookeeper-import) finished
successfully.
```

9. (*If upgrading from Fusion 3.1.0 through 3.1.3; On all Fusion nodes*) Start Solr for the new Fusion version:

```
%FUSION_NEW%\bin\solr.cmd start
```

10. (*If upgrading from Fusion 3.1.0 through 3.1.3; Only on the main Fusion node*) Run a script to remove all old plugins from the blob store. Replace `solr-address` and `solr-port` as appropriate (as shown in

the example):

```
java -cp "%FUSION_OLD%\var\upgrade\jython-standalone-
2.7.1.jar;%FUSION_OLD%\var\upgrade\migrator.jar"
org.python.util.jython
"%FUSION_OLD%\var\upgrade\transformations\manual_delete_old_plug
in_blobs.py" --solr-address solr-address --solr-port solr-port
```

For example:

```
java -cp "%FUSION_OLD%\var\upgrade\jython-standalone-
2.7.1.jar;%FUSION_OLD%\var\upgrade\migrator.jar"
org.python.util.jython
"%FUSION_OLD%\var\upgrade\transformations\manual_delete_old_plug
in_blobs.py" --solr-address localhost --solr-port 8983
```

This message indicates that plugins were deleted successfully:

```
Deleted old plugin blobs from solr <?xml version="1.0"
encoding="UTF-8"?> <response> <lst name="responseHeader"> <int
name="status">0</int> <int name="QTime">246</int> </lst>
</response> Old connector plugin blobs were deleted
successfully.
```

11. (*On all Fusion nodes*) Start all Fusion services for the new version of Fusion:

```
%FUSION_NEW%\bin\fusion.cmd start
```

12. (*Only on the main Fusion node*) Import the second part of configuration data into the new version of Fusion:

```
java -jar "%FUSION_OLD%\var\upgrade\migrator.jar" --fusion-import
```

This message indicates that the step finished successfully:

```
New Fusion object import (--fusion-import) finished successfully.
```

Tip

After migration, you can find details about the results in the `fusion\3.1.x\var\upgrade\tmp` directory. If the migration produces unexpected results, the files in this directory are helpful for troubleshooting.

Validate the new version of Fusion

How to validate the new version of Fusion

1. (*On all Fusion nodes*) Restart all Fusion services for the new version of Fusion:

```
%FUSION_NEW%\bin\fusion.cmd restart
```

2. Log into the Fusion UI (your `admin` password is the same as for the old installation), and confirm the release number of the new version of Fusion:
   a. Clear your browser's cache.
   Otherwise, you might inadvertently access a cached version of the old Fusion UI and see inconsistent behavior.
   b. In a browser, open the Fusion UI at `http://localhost:8764/` (replace `localhost` with your server name or IP address if necessary).
   c. Log in.
   d. Navigate to **Admin** > **About Fusion**.
   The **About Fusion** panel should display the newer Fusion release number.

3. Ensure that all connectors were installed automatically during the upgrade.
   a. For Fusion 4.x from the Fusion launcher, click the tile for a migrated app. Click **System** > **Blobs**. If any connectors are missing from the list, click **Add** > **Connector Plugin** and install them manually.

b. For Fusion 3.x from the Fusion launcher, click **Devops** > **Home** > **Blobs**. If any connectors are missing from the list, click **Add** > **Connector Plugin** and install them manually.

4. Ensure that all customizations you made in the former version of Fusion are present in the new one.
5. If you migrated from version 3.1.x to version 4.2.0 or later, *confirm associations between blobs and apps* and make adjustments if necessary.
6. When you are satisfied with the migration and you have backed up the `fusion\3.1.x` directory, you can remove the older version of Fusion by removing that directory (*on all Fusion nodes*).

Add support for business rules to existing apps

Fusion AI 4.2 introduces functionality for using business rules, that is, manually created formulas for rewriting queries and responses.

You can add support for business rules to apps that were created in versions of Fusion AI prior to version 4.2. To do so, perform the steps in this section.

The `add-rule-objects-xyz.zip` file (where `xyz` is a version number) specifies the objects to add to an app. It is supplied in the Fusion migrator zip file at the top level. After installing the migrator, the location is `%FUSION_OLD%\var\upgrade\import-files\`.

You have a choice. You can update each app using the *Fusion UI* or the *Fusion API*.

FUSION UI

For each app in which you plan to use business rules, import the objects in the `add-rule-objects-xyz.zip` file into the app.

How to import business rule objects

1. In the Fusion launcher, click the app into which you want to import objects.
   The Fusion workspace appears.
2. In the upper left, click System > **Import Fusion Objects**.

   The Import Fusion Objects window opens.
3. For the data file, select `add-rule-objects-xyz.zip` from your local filesystem. The location in the extracted migrator files is `%FUSION_OLD%\var\upgrade\import-files\`.
4. Click **Import**.
5. Edit the `Application ID` parameter value to use the app name. If the app name contains spaces, replace those with underscore characters. For example, `Lucene Revolution` would become `Lucene_Revolution`.
6. Click **Import**.
7. If there are conflicts, Fusion prompts you to specify an import policy. Click **Merge** to skip all conflicting objects and import only the non-conflicting objects.
   Fusion confirms that the import was successful.
8. Click **Close** to close the Import Fusion Objects window.

FUSION API

For each app in which you plan to use business rules, import the objects in the `add-rule-objects-xyz.zip` file into the app.

How to import business rule objects

1. Create an `app-name.txt` file with the following content:

```
{"app.id" : "_name-of-migrated-app_"}
```

For example, for the app **`Lucene Revolution`**:

```
{"app.id" : "Lucene_Revolution"}
```

Here, we assume that you create the files in your home directory, for which the **`%HOMEPATH%`** environment variable is defined.

2. Import the business rule objects:

```
curl -u USERNAME:PASSWORD -H "Content-Type:multipart/form-data"
-X POST -F 'variableValues=path-to-txt-file\app-name.txt' -F
"importData=path-to-zip-file\add-rule-objects-xyz.zip"
"http://<FUSION_HOST>/api/objects/import?importPolicy=merge"
```

For example:

```
curl -u admin:Very-SecurePW8 -H "Content-Type:multipart/form-
data" -X POST -F
'variableValues=%HOMEPATH%\Lucene_Revolution.txt' -F
"importData=%FUSION_OLD%\import-files\add-rule-objects-001.zip"
"http://<FUSION_HOST>/api/objects/import?importPolicy=merge"
```

Upgrade Fusion Server 4.0.x to 4.2.y

How-to

> Compatible versions: 4.2
>
> Works with: *Fusion Server*

Introduction

This article describes how to perform the following upgrade:

- From version: Fusion `4.0.x`
- To version: Fusion `4.2.y`

Important

Only specific version-to-version upgrade sequences are supported. Some upgrades require multiple steps.

For more information, see *Fusion 4.x*.

For Fusion 3.1 and later releases, a migrator is available for upgrading Fusion.

During the upgrade process, the migrator uses a properties file. After downloading and installing the migrator, the properties files is in the `/opt/lucidworks/fusion/4.0.x/var/upgrade` directory (on Unix or MacOS) or the `C:\lucidworks\fusion\4.0.x\var\upgrade\` directory (on Windows). The file names reference the versions you are upgrading from and to. For example:

- To upgrade `4.0.2` to `4.2.5`, the migrator uses the `4.0.x-4.2.x.properties` file.

Migration entails down time and multiple starts and stops of Fusion services. Plan accordingly, especially in terms of disabling external load balancers or monitors that might react adversely to the starts and stops.

Download the latest migrator immediately before upgrading. This helps ensure that the upgrade goes smoothly.

Important

The newer Fusion instance must be newly untarred and *never started*.

About the upgrade

This section describes how connectors, object migrations, and signals are migrated during an upgrade.

Connectors

In Fusion 3.1.0 and later, only a subset of *connectors* are included by default.

The migrator detects which connectors were used in the older version of Fusion, and installs them automatically in Fusion `4.2.y`. This step requires an Internet connection. If no connection is available, then download the connectors from *Fusion 4.x Connector Downloads* and *install them as bootstrap plugins*.

If a connector to be upgraded was not available during the upgrade, then a message in `/opt/lucidworks/fusion/4.0.x/var/upgrade/tmp/migrator.log` (on Unix) or `C:\lucidworks\fusion\4.0.x\var\upgrade\tmp\migrator.log` (on Windows) indicates this.

Only datasources for connectors that are supported in the new Fusion version are upgraded. Datasources for custom connectors are not upgraded.

IF NO INTERNET CONNECTION IS AVAILABLE

If no Internet connection is available during an upgrade, the migrator cannot automatically download the connectors it needs. Using the Fusion UI or API later to install the connectors also might not be an option.

In this case, download the connectors from *Fusion 4.x Connector Downloads* for all existing connectors and place them in `apps/connectors/bootstrap-plugins` for the new deployment (on all Fusion nodes). Do so at the time indicated in the procedures that follow.

ADDING CONNECTORS DURING AN UPGRADE

You can *add* connectors during an upgrade (that is, add connectors that are not in the old deployment).

Download the connectors from *Fusion 4.x Connector Downloads* and place them in `apps/connectors/bootstrap-plugins` for the new version (on all Fusion nodes).

Object migrations and transformations

The migrator automatically migrates these Fusion 4.0 object types, transforming them as needed:

- Collections
- Index pipelines
- Query pipelines
- Search cluster configurations
- Datasources
- Parsing configurations
- Object groups
- Links
- Tasks
- Jobs
- Spark objects
- Apps
- Appkit apps
- Index profiles
- Query profiles
- Blobs

Important

In Fusion Server 4.0 and later, most objects exist in the context of apps. When you upgrade from Fusion Server 4.0.x to 4.2.y, the migrator upgrades app objects, all objects in or linked to objects in apps, and objects that are not linked to apps. You can explore the objects in *Object Explorer*.

Noteworthy transformations

While migrating objects, the migrator makes these transformations:

- It adds the new `log-shipper` service to `fusion.properties`.
- It removes obsolete log-cleanup tasks.
- It converts the format of job history records.

Access control migration

The migrator upgrades all access control configurations:

- Security realms
- Roles
- Users

None of these should require adjustments after migration.

Association of blobs with apps

The association of blobs with apps becomes stronger in Fusion Server 4.1.1. Newly created blobs are only linked to the app to which they are uploaded, and are only shared with other apps when the blobs are directly linked to those apps (added to the apps in *Object Explorer* or linked using the *Links API*).

When migrating from pre-4.1.1 versions to 4.1.1 or later, the resulting apps might not contain all of the blobs that they contained prior to migration.

Note

It is important to stress that this change *does not break the apps*.

What *might* change is the blobs that are linked to an app. This can be important if you export and re-import apps.

Recommendation: After migration from a pre-4.1.1 version to version 4.1.1 or later, we recommend that you use either Object Explorer or the Links API to manually link all blobs to the apps in which they are used.

Review known issues

Before upgrading, *review the known issues* to see whether any of them apply to the circumstances of your upgrade. Some known issues might require actions *before* upgrading.

That article also contains instructions regarding what to do *if an upgrade step fails*.

Upgrade on Unix

Use this procedure to upgrade Fusion on a single Unix node or on multiple Unix nodes.

Perform the steps in this procedure on the indicated nodes *on which Fusion is running* ("Fusion nodes"). To perform an upgrade, Fusion nodes must have at least these services running:

- API service (`api`)
- Proxy service (`proxy`)

Important

For every step on multiple nodes, ensure that the step completes *on all Fusion nodes* before going to the next step. There is the notion of a "main node" during the migration process. This node will be used for certain centralized migration activities that do not need to be done on every node, such as downloading connectors that are then uploaded to blob storage that is shared by all, etc. Just pick one of your Fusion nodes to be the "main node"; there is no special requirement as to which one you pick.

Ensure that your current version of Fusion has a valid license

Ensure that your *current* version of Fusion has a valid permanent Fusion license before proceeding with the upgrade. Place a valid `license.properties` file in the `/opt/lucidworks/fusion/4.0.x/conf` directory.

Install the Java JDK

Before upgrading to Fusion Server 4.2 or later, you must install the Java JDK on all nodes that will run Fusion Server. Prior versions of Fusion Server could use either the JRE or the JDK.

For more information, see *Java requirements*.

Download and install the newer version of Fusion

Perform these tasks *on all Fusion nodes*:

1. Select the Fusion release to which you are upgrading from *Fusion Server 4.x File Download Links*.
2. Extract the newer version of Fusion:

```
cd /opt/lucidworks mv ~/Downloads/fusion-4.2.y.tar.gz ./ tar -xf
fusion-4.2.y.tar.gz
```

For example, if Fusion is currently installed in `/opt/lucidworks/fusion/4.0.x`, then change your working directory to `/opt/lucidworks/` and extract the file there. *do not run the new version of Fusion yet.*

3. Ensure that the *new* version of Fusion has a valid permanent Fusion license before proceeding with the upgrade. Place a valid `license.properties` file in the `/opt/lucidworks/fusion/4.2.y/conf` directory.
4. (*If there are custom* `jar` *files*) If your deployment has custom `jar` files, add them to the new Fusion deployment.
5. (*If you are performing an upgrade without Internet access*) Without Internet access, the migrator cannot download new versions of connectors automatically. Download the *new versions* of connector zip files *for your current connectors* from *Fusion 4.x Connector Downloads* and place them in `apps/connectors/bootstrap-plugins` for the new deployment.
6. (*If you are adding* ***new*** *connectors*) If you want your new deployment to use connectors that *are not* in the current deployment, you can add them now. Download the connector zip files from *Fusion 4.x Connector Downloads* and place them in `apps/connectors/bootstrap-plugins` for the new deployment.
7. Verify that there is sufficient disk space for a second copy of the Solr index directory, `fusion/4.0.x/data/solr`. If there is not sufficient disc space, free up space before proceeding.

Download and install the Fusion migrator

Perform these tasks *on all Fusion nodes*:

1. Download the latest migrator zip file for Unix. (Do this now, even if you have downloaded the migrator before, to ensure that you have the latest version.)
2. Create `FUSION_OLD` and `FUSION_NEW` environment variables that point to the old and new Fusion installation directories respectively (using the full path).

```
export FUSION_OLD="/opt/lucidworks/fusion/4.0.x" export
FUSION_NEW="/opt/lucidworks/fusion/4.2.y"
```

For example, when upgrading from Fusion 4.0.2 to 4.2.6:

```
export FUSION_OLD="/opt/lucidworks/fusion/4.0.2" export
FUSION_NEW="/opt/lucidworks/fusion/4.2.6"
```

3. Create this directory:

```
mkdir $FUSION_OLD/var/upgrade
```

4. Install the migrator:

```
tar -zxv -C $FUSION_OLD/var/upgrade --strip-components=1 -f
fusion-migrator.tar.gz
```

Run the migrator

Perform these tasks on the indicated nodes:

1. (*On all Fusion nodes*) Start all Fusion services for the old version of Fusion:

```
$FUSION_OLD/bin/fusion start
```

2. (*Only on the main Fusion node*) Run the migrator to export the configuration data from the old version of Fusion:

```
java -jar $FUSION_OLD/var/upgrade/migrator.jar --export
```

This message indicates that the step finished successfully:

```
Old Fusion configuration export (--export) finished
successfully.
```

3. (*On all Fusion nodes*) Stop the old versions of Fusion services and Solr; but not ZooKeeper:

```
$FUSION_OLD/bin/webapps stop $FUSION_OLD/bin/admin-ui stop
$FUSION_OLD/bin/proxy stop $FUSION_OLD/bin/connectors-classic
stop $FUSION_OLD/bin/connectors-rpc stop $FUSION_OLD/bin/api
stop $FUSION_OLD/bin/solr stop
```

If Spark and SQL services are running, also stop those:

```
$FUSION_OLD/bin/spark-master stop $FUSION_OLD/bin/spark-worker
stop $FUSION_OLD/bin/sql stop
```

Tip

You can see what is running with **`$FUSION_OLD/bin/fusion status`**.

4. (*Only on secondary Fusion nodes*) Prepare secondary nodes:

```
java -jar $FUSION_OLD/var/upgrade/migrator.jar --prepare-
secondary
```

This message indicates that the step finished successfully:

```
Prepare secondary nodes (--prepare-secondary) finished
successfully.
```

5. (*On all Fusion nodes*) Stop ZooKeeper for the old version of Fusion (*unless you are using an external ZooKeeper instance, in which case you can ignore this step*):

```
$FUSION_OLD/bin/zookeeper stop
```

6. (*Only on the main Fusion node*) Transform configuration data on the main Fusion node:

```
java -jar $FUSION_OLD/var/upgrade/migrator.jar --main-transform
```

Note

Depending on the size of your Solr index, this step can take a long time (for example, multiple tens of minutes).

This message indicates that the step finished successfully:

```
Fusion data transformations on main node (--main-transform)
finished successfully.
```

7. (*On all Fusion nodes*) Start ZooKeeper for the new version of Fusion (*unless you are using an external ZooKeeper instance, in which case you can ignore this step*):

```
$FUSION_NEW/bin/zookeeper start
```

8. (*Only on the main Fusion node*) Import the first part of configuration data into the new version of Fusion:

```
java -jar $FUSION_OLD/var/upgrade/migrator.jar --zookeeper-
import
```

This message indicates that the step finished successfully:

```
New Fusion Zookeeper import (--zookeeper-import) finished
successfully.
```

9. (*On all Fusion nodes*) Start all Fusion services for the new version of Fusion:

```
$FUSION_NEW/bin/fusion start
```

10. (*Only on the main Fusion node*) Import the second part of configuration data into the new version of Fusion:

```
java -jar $FUSION_OLD/var/upgrade/migrator.jar --fusion-import
```

This message indicates that the step finished successfully:

```
New Fusion object import (--fusion-import) finished
successfully.
```

Tip

After migration, you can find details about the results in the `fusion/4.0.x/var/upgrade/tmp` directory. If the migration produces unexpected results, the files in this directory are helpful for troubleshooting.

Validate the new version of Fusion

How to validate the new version of Fusion

1. (*Only on the main Fusion node*) Restart the new version of Fusion (all services defined in `fusion.properties`):

```
$FUSION_NEW/bin/fusion restart
```

2. Log into the Fusion UI (your `admin` password is the same as for the old installation), and confirm the release number of the new version of Fusion:
   a. Clear your browser's cache.
   Otherwise, you might inadvertently access a cached version of the old Fusion UI and see inconsistent behavior.
   b. In a browser, open the Fusion UI at `http://localhost:8764/` (replace `localhost` with your server name or IP address if necessary).
   c. Log in.
   d. Navigate to **Admin** > **About Fusion**.
   The **About Fusion** panel should display the newer Fusion release number.

3. Ensure that all connectors were installed automatically during the upgrade.
   - For Fusion 4.x from the Fusion launcher, click the tile for a migrated app. Click **System** > **Blobs**. If any connectors are missing from the list, click **Add** > **Connector Plugin** and install them manually.

- For Fusion 3.x from the Fusion launcher, click **Devops** > **Home** > **Blobs**. If any connectors are missing from the list, click **Add** > **Connector Plugin** and install them manually.

4. Ensure that all customizations you made in the former version of Fusion are present in the new one.
5. If you migrated from version 4.0.x to version 4.2.0 or later, *confirm associations between blobs and apps* and make adjustments if necessary.
6. When you are satisfied with the migration and you have backed up the `fusion/4.0.x/` directory, you can `rm -fr fusion/4.0.x/` to remove the older version of Fusion (*on all Fusion nodes*).

Add support for business rules to existing apps

Fusion AI 4.2 introduces functionality for using business rules, that is, manually created formulas for rewriting queries and responses.

You can add support for business rules to apps that were created in versions of Fusion AI prior to version 4.2. To do so, perform the steps in this section.

The `add-rule-objects-xyz.zip` file (where `xyz` is a version number) specifies the objects to add to an app. It is supplied in the Fusion migrator zip file at the top level. After installing the migrator, the location is `$FUSION_OLD/var/upgrade/import-files`.

Note

Adding support for business rules has a costs. Additional collections and objects are created. Only add support for business rules to apps in which you plan to use them.

You have a choice. You can update each app using the *Fusion UI* or the *Fusion API*.

FUSION UI

For each app in which you plan to use business rules, import the objects in the `add-rule-objects-xyz.zip` file into the app.

How to import business rule objects

1. In the Fusion launcher, click the app into which you want to import objects.
   The Fusion workspace appears.
2. In the upper left, click System > **Import Fusion Objects**.
   The Import Fusion Objects window opens.
3. For the data file, select `add-rule-objects-xyz.zip` from your local filesystem. The location in the extracted migrator files is `$FUSION_OLD/var/upgrade/import-files`.
4. Click **Import**.
5. Edit the `Application ID` parameter value to use the app name. If the app name contains spaces, replace those with underscore characters. For example, `Lucene Revolution` would become `Lucene_Revolution`.
6. Click **Import**.
7. If there are conflicts, Fusion prompts you to specify an import policy. Click **Merge** to skip all conflicting objects and import only the non-conflicting objects.
   Fusion confirms that the import was successful.
8. Click **Close** to close the Import Fusion Objects window.

FUSION API

For each app in which you plan to use business rules, import the objects in the `add-rule-objects-xyz.zip` file into the app.

How to import business rule objects

1. Create an `app-name.txt` file with the following content:

```
{"app.id" : "_name-of-migrated-app_"}
```

   For example, for the app `Lucene Revolution` :

```
{"app.id" : "Lucene_Revolution"}
```

   Here, we assume that you create the files in your home directory, for which the `$HOME` environment variable is defined.
2. Import the business rule objects:

```
curl -u USERNAME:PASSWORD -H "Content-Type:multipart/form-data"
-X POST -F 'variableValues=path-to-txt-file/app-name.txt' -F
"importData=path-to-zip-file/add-rule-objects-xyz.zip"
"http://<FUSION_HOST>/api/objects/import?importPolicy=merge"
```

   For example:

```
curl -u admin:Very-SecurePW8 -H "Content-Type:multipart/form-
data" -X POST -F 'variableValues=$HOME/Lucene_Revolution.txt' -F
"importData=$FUSION_OLD/var/upgrade/import-files/add-rule-
objects-001.zip" "http://<FUSION_HOST>/api/objects/import?
importPolicy=merge"
```

Upgrade on Windows

Use this procedure to upgrade Fusion on a single Windows node or multiple Windows nodes.

Perform the steps in this procedure on the indicated nodes *on which Fusion is running* ("Fusion nodes"). To perform an upgrade, Fusion nodes must have at least these services running:

- API service (`api`)

- Proxy service (`proxy`)

Important

If you are upgrading Fusion on multiple nodes, then, for every step on multiple nodes, ensure that the step completes *on all Fusion nodes* before going to the next step. There is the notion of a "main node" during the migration process. This node will be used for certain centralized migration activities that do not need to be done on every node, such as downloading connectors that are then uploaded to blob storage that is shared by all, etc. Just pick one of your Fusion nodes to be the "main node"; there is no special requirement as to which one you pick.

Ensure that your current version of Fusion has a valid license

Ensure that your current version of Fusion has a valid permanent Fusion license before proceeding with the upgrade. Place a valid `license.properties` file in the `C:\lucidworks\fusion\4.0.x\conf` directory.

Install the Java JDK

Before upgrading to Fusion Server 4.2 or later, you must install the Java JDK on all nodes that will run Fusion Server. Prior versions of Fusion Server could use either the JRE or the JDK.

For more information, see *Java requirements*.

Download and install the newer version of Fusion

Perform these tasks *on all Fusion nodes*:

1. Select the Fusion release to which you are upgrading from *Fusion Server 4.x File Download Links*.
2. Move the `fusion-4.2.y.zip` file to the directory that contains the `fusion\` directory. For example, if Fusion is installed in `C:\lucidworks\fusion\4.0.x`, then move the file to `C:\lucidworks`.
3. Unzip the `fusion-4.2.y.zip` file. *do not run the new version of Fusion yet.*
4. For Fusion 4.x, ensure that the *new* version of Fusion has a valid permanent Fusion license before proceeding with the upgrade. Place a valid `license.properties` file in the `C:\lucidworks\fusion\4.2.y\conf` directory.
5. (*If there are custom* `jar` *files*) If your deployment has custom `jar` files, add them to the new Fusion deployment.
6. (*If you are performing an upgrade without Internet access*) Without Internet access, the migrator cannot download new versions of connectors automatically. Download the *new versions* of connector zip files *for your current connectors* from *Fusion 4.x Connector Downloads* and place them in `apps\connectors\bootstrap-plugins` for the new deployment.
7. (*If you are adding* ***new*** *connectors*) If you want your new deployment to use connectors that *are not* in the current deployment, you can add them now. Download the connector zip files from *Fusion 4.x Connector Downloads* and place them in `apps\connectors\bootstrap-plugins` for the new deployment.
8. Verify that there is sufficient disk space for a second copy of the Solr index directory, `fusion\4.0.x\data\solr`. If there is not sufficient disc space, free up space before proceeding.

Download and install the Fusion migrator

Perform these tasks *on all Fusion nodes*:

1. Download the latest migrator zip file for Windows. (Do this now, even if you have downloaded the migrator before, to ensure that you have the latest version.)
2. Open a Command Prompt window and create `FUSION_OLD` and `FUSION_NEW` environment variables that point to the old and new Fusion installation directories respectively. For example:

```
set FUSION_OLD=C:\lucidworks\fusion\4.0.2 set
FUSION_NEW=C:\lucidworks\fusion\4.2.6
```

3. Create a `fusion\4.0.x\var\upgrade` directory.
4. Unzip the migrator zip file, and move the contents of the extracted folder to `fusion\4.0.x\var\upgrade`.

Run the migrator

Perform these tasks on the indicated nodes:

1. (*On all Fusion nodes*) Start all Fusion services for the old version of Fusion:

```
%FUSION_OLD%\bin\fusion.cmd start
```

2. (*Only on the main Fusion node*) Run the migrator to export the configuration data from the old version of Fusion:

```
java -jar "%FUSION_OLD%\var\upgrade\migrator.jar" --export
```

This message indicates that the step finished successfully:

```
Old Fusion configuration export (--export) finished
successfully.
```

3. (*On all Fusion nodes*) Stop the old versions of Fusion services and Solr; but not ZooKeeper:

```
%FUSION_OLD%\bin\webapps.cmd stop %FUSION_OLD%\bin\admin-ui.cmd
stop %FUSION_OLD%\bin\proxy.cmd stop
%FUSION_OLD%\bin\connectors-classic.cmd stop
%FUSION_OLD%\bin\connectors-rpc.cmd stop
%FUSION_OLD%\bin\api.cmd stop %FUSION_OLD%\bin\solr.cmd stop
```

If Spark and SQL services are running, also stop those:

```
%FUSION_OLD%\bin\spark-master.cmd stop %FUSION_OLD%\bin\spark-
worker.cmd stop %FUSION_OLD%\bin\sql.cmd stop
```

Tip

You can see what is running with `%FUSION_OLD%\bin\fusion status`.

4. (*Only on secondary Fusion nodes*) Prepare secondary nodes:

```
java -jar "%FUSION_OLD%\var\upgrade\migrator.jar" --prepare-
secondary
```

This message indicates that the step finished successfully:

```
Prepare secondary nodes (--prepare-secondary) finished
successfully.
```

5. (*On all Fusion nodes*) Stop ZooKeeper for the old version of Fusion (*unless you are using an external ZooKeeper instance, in which case you can ignore this step*):

```
%FUSION_OLD%\bin\zookeeper.cmd stop
```

6. (*Only on the main Fusion node*) Transform configuration data on the main Fusion node:

```
java -jar "%FUSION_OLD%\var\upgrade\migrator.jar" --main-transform
```

Note

Depending on the size of your Solr index, this step can take a long time (for example, multiple tens of minutes).

This message indicates that the step finished successfully:

```
Fusion data transformations on main node (--main-transform) finished successfully.
```

7. (*On all Fusion nodes*) Start ZooKeeper for the new version of Fusion (*unless you are only using an external ZooKeeper instance, in which case you can ignore this step*):

```
%FUSION_NEW%\bin\zookeeper.cmd start
```

8. (*Only on the main Fusion node*) Import the first part of configuration data into the new version of Fusion:

```
java -jar "%FUSION_OLD%\var\upgrade\migrator.jar" --zookeeper-import
```

This message indicates that the step finished successfully:

```
New Fusion Zookeeper import (--zookeeper-import) finished
successfully.
```

9. (*On all Fusion nodes*) Start Solr for the new Fusion version:

```
%FUSION_NEW%\bin\solr.cmd start
```

10. (*Only on the main Fusion node*) Run a script to remove all old plugins from the blob store. Replace ***solr-address*** and ***solr-port*** as appropriate (as shown in the example):

```
java -cp "%FUSION_OLD%\var\upgrade\jython-standalone-
2.7.1.jar;%FUSION_OLD%\var\upgrade\migrator.jar"
org.python.util.jython
"%FUSION_OLD%\var\upgrade\transformations\manual_delete_old_plug
in_blobs.py" --solr-address solr-address --solr-port solr-port
```

For example:

```
java -cp "%FUSION_OLD%\var\upgrade\jython-standalone-
2.7.1.jar;%FUSION_OLD%\var\upgrade\migrator.jar"
org.python.util.jython
"%FUSION_OLD%\var\upgrade\transformations\manual_delete_old_plug
in_blobs.py" --solr-address localhost --solr-port 8983
```

This message indicates that plugins were deleted successfully:

```
Deleted old plugin blobs from solr <?xml version="1.0"
encoding="UTF-8"?> <response> <lst name="responseHeader"> <int
name="status">0</int> <int name="QTime">246</int> </lst>
</response> Old connector plugin blobs were deleted
successfully.
```

11. (*On all Fusion nodes*) Start all Fusion services for the new version of Fusion:

```
%FUSION_NEW%\bin\fusion.cmd start
```

12. (*Only on the main Fusion node*) Import the second part of configuration data into the new version of Fusion:

```
java -jar "%FUSION_OLD%\var\upgrade\migrator.jar" --fusion-
import
```

This message indicates that the step finished successfully:

```
New Fusion object import (--fusion-import) finished
successfully.
```

Tip

After migration, you can find details about the results in the `fusion\4.0.x\var\upgrade\tmp` directory. If the migration produces unexpected results, the files in this directory are helpful for troubleshooting.

Validate the new version of Fusion

How to validate the new version of Fusion

1. (*On all Fusion nodes*) Restart all Fusion services for the new version of Fusion:

```
%FUSION_NEW%\bin\fusion.cmd restart
```

2. Log into the Fusion UI (your **admin** password is the same as for the old installation), and confirm the release number of the new version of Fusion:
    a. Clear your browser's cache.
    Otherwise, you might inadvertently access a cached version of the old Fusion UI and see inconsistent behavior.
    b. In a browser, open the Fusion UI at **http://localhost:8764/** (replace **localhost** with your server name or IP address if necessary).
    c. Log in.
    d. Navigate to **Admin** > **About Fusion**.
    The **About Fusion** panel should display the newer Fusion release number.

3. Ensure that all connectors were installed automatically during the upgrade.
    a. For Fusion 4.x from the Fusion launcher, click the tile for a migrated app. Click **System** > **Blobs**. If any connectors are missing from the list, click **Add** > **Connector Plugin** and install them manually.

    b. For Fusion 3.x from the Fusion launcher, click **Devops** > **Home** > **Blobs**. If any connectors are missing from the list, click **Add** > **Connector Plugin** and install them manually.

4. Ensure that all customizations you made in the former version of Fusion are present in the new one.
5. If you migrated from version 4.0.x to version 4.2.0 or later, *confirm associations between blobs and apps* and make adjustments if necessary.
6. When you are satisfied with the migration and you have backed up the **fusion\4.0.*x*** directory, you can remove the older version of Fusion by removing that directory (*on all Fusion nodes*).

Add support for business rules to existing apps

Fusion AI 4.2 introduces functionality for using business rules, that is, manually created formulas for rewriting queries and responses.

You can add support for business rules to apps that were created in versions of Fusion AI prior to version 4.2. To do so, perform the steps in this section.

The **add-rule-objects-*xyz*.zip** file (where ***xyz*** is a version number) specifies the objects to add to an app. It is supplied in the Fusion migrator zip file at the top level. After installing the migrator, the location is **%FUSION_OLD%\var\upgrade\import-files**.

You have a choice. You can update each app using the *Fusion UI* or the *Fusion API*.

FUSION UI

For each app in which you plan to use business rules, import the objects in the **add-rule-objects-*xyz*.zip** file into the app.

How to import business rule objects

1. In the Fusion launcher, click the app into which you want to import objects.
The Fusion workspace appears.

2. In the upper left, click System > **Import Fusion Objects**.

The Import Fusion Objects window opens.

3. For the data file, select **add-rule-objects-*xyz*.zip** from your local filesystem. The location in the extracted migrator files is **%FUSION_OLD%\var\upgrade\import-files**.
4. Click **Import**.

5. Edit the **Application ID** parameter value to use the app name. If the app name contains spaces, replace those with underscore characters. For example, **Lucene Revolution** would become **Lucene_Revolution**.
6. Click **Import**.
7. If there are conflicts, Fusion prompts you to specify an import policy. Click **Merge** to skip all conflicting objects and import only the non-conflicting objects.
   Fusion confirms that the import was successful.
8. Click **Close** to close the Import Fusion Objects window.

FUSION API

For each app in which you plan to use business rules, import the objects in the **add-rule-objects-*xyz*.zip** file into the app.

How to import business rule objects

1. Create an **app-name.txt** file with the following content:

```
{"app.id" : "_name-of-migrated-app_"}
```

   For example, for the app **Lucene Revolution**:

```
{"app.id" : "Lucene_Revolution"}
```

   Here, we assume that you create the files in your home directory, for which the **%HOMEPATH%** environment variable is defined.
2. Import the business rule objects:

```
curl -u USERNAME:PASSWORD -H "Content-Type:multipart/form-data"
-X POST -F 'variableValues=path-to-txt-file\app-name.txt' -F
"importData=path-to-zip-file\add-rule-objects-xyz.zip"
"http://<FUSION_HOST>/api/objects/import?importPolicy=merge"
```

   For example:

```
curl -u admin:Very-SecurePW8 -H "Content-Type:multipart/form-data" -X POST -F 'variableValues=%HOMEPATH%\Lucene_Revolution.txt' -F "importData=%FUSION_OLD%\import-files\add-rule-objects-001.zip" "http://<FUSION_HOST>/api/objects/import?importPolicy=merge"
```

Upgrade Fusion Server 4.1.x to 4.2.y

How-to

> Compatible versions: 4.2
>
> Works with: *Fusion Server*

Introduction

This article describes how to perform the following upgrade:

- From version: Fusion `4.1.x`
- To version: Fusion `4.2.y`

Important

Only specific version-to-version upgrade sequences are supported. Some upgrades require multiple steps.

For more information, see *Fusion 4.x*.

For Fusion 3.1 and later releases, a migrator is available for upgrading Fusion.

During the upgrade process, the migrator uses a properties file. After downloading and installing the migrator, the properties files is in the `/opt/lucidworks/fusion/4.1.x/var/upgrade` directory (on Unix or MacOS) or the `C:\lucidworks\fusion\4.1.x\var\upgrade\` directory (on Windows). The file names reference the versions you are upgrading from and to. For example:

- To upgrade `4.1.3` to `4.2.5`, the migrator uses the `4.1.x-4.2.x.properties` file.

Migration entails down time and multiple starts and stops of Fusion services. Plan accordingly, especially in terms of disabling external load balancers or monitors that might react adversely to the starts and stops.

Download the latest migrator immediately before upgrading. This helps ensure that the upgrade goes smoothly.

Important

The newer Fusion instance must be newly untarred and *never started*.

About the upgrade

This section describes how connectors, object migrations, and signals are migrated during an upgrade.

Connectors

In Fusion 3.1.0 and later, only a subset of *connectors* are included by default.

The migrator detects which connectors were used in the older version of Fusion, and installs them automatically in Fusion `4.2.y`. This step requires an Internet connection. If no connection is available, then download the connectors from *Fusion 4.x Connector Downloads* and *install them as bootstrap plugins*.

If a connector to be upgraded was not available during the upgrade, then a message in `/opt/lucidworks/fusion/4.1.x/var/upgrade/tmp/migrator.log` (on Unix) or `C:\lucidworks\fusion\4.1.x\var\upgrade\tmp\migrator.log` (on Windows) indicates this.

Only datasources for connectors that are supported in the new Fusion version are upgraded. Datasources for custom connectors are not upgraded.

IF NO INTERNET CONNECTION IS AVAILABLE

If no Internet connection is available during an upgrade, the migrator cannot automatically download the connectors it needs. Using the Fusion UI or API later to install the connectors also might not be an option.

In this case, download the connectors from *Fusion 4.x Connector Downloads* for all existing connectors and place them in `apps/connectors/bootstrap-plugins` for the new deployment (on all Fusion nodes). Do so at the time indicated in the procedures that follow.

ADDING CONNECTORS DURING AN UPGRADE

You can *add* connectors during an upgrade (that is, add connectors that are not in the old deployment).

Download the connectors from *Fusion 4.x Connector Downloads* and place them in `apps/connectors/bootstrap-plugins` for the new version (on all Fusion nodes).

Object migrations and transformations

The migrator automatically migrates these Fusion 4.1 object types, transforming them as needed:

- Collections
- Index pipelines
- Query pipelines
- Search cluster configurations
- Datasources
- Parsing configurations
- Object groups
- Links
- Tasks
- Jobs
- Spark objects
- Apps
- Appkit apps
- Index profiles
- Query profiles
- Blobs

Important

In Fusion Server 4.0 and later, most objects exist in the context of apps. When you upgrade from Fusion Server 4.1.x to 4.2.y, the migrator upgrades app objects, all objects in or linked to objects in apps, and objects that are not linked to apps. You can explore the objects in *Object Explorer*.

Access control migration

The migrator upgrades all access control configurations:

- Security realms
- Roles
- Users

None of these should require adjustments after migration.

Association of blobs with apps

The association of blobs with apps becomes stronger in Fusion Server 4.1.1. Newly created blobs are only linked to the app to which they are uploaded, and are only shared with other apps when the blobs are directly linked to those apps (added to the apps in *Object Explorer* or linked using the *Links API*).

When migrating from pre-4.1.1 versions to 4.1.1 or later, the resulting apps might not contain all of the blobs that they contained prior to migration.

Note

It is important to stress that this change *does not break the apps*.

What *might* change is the blobs that are linked to an app. This can be important if you export and re-import apps.

Recommendation: After migration from a pre-4.1.1 version to version 4.1.1 or later, we recommend that you use either Object Explorer or the Links API to manually link all blobs to the apps in which they are used.

Review known issues

Before upgrading, *review the known issues* to see whether any of them apply to the circumstances of your upgrade. Some known issues might require actions *before* upgrading.

That article also contains instructions regarding what to do *if an upgrade step fails*.

Upgrade on Unix

Use this procedure to upgrade Fusion on a single Unix node or on multiple Unix nodes.

Perform the steps in this procedure on the indicated nodes *on which Fusion is running* ("Fusion nodes"). To perform an upgrade, Fusion nodes must have at least these services running:

- API service (`api`)
- Proxy service (`proxy`)

Important

For every step on multiple nodes, ensure that the step completes *on all Fusion nodes* before going to the next step. There is the notion of a "main node" during the migration process. This node will be used for certain centralized migration activities that do not need to be done on every node, such as downloading connectors that are then uploaded to blob storage that is shared by all, etc. Just pick one of your Fusion nodes to be the "main node"; there is no special requirement as to which one you pick.

Ensure that your current version of Fusion has a valid license

Ensure that your *current* version of Fusion has a valid permanent Fusion license before proceeding with the upgrade. Place a valid `license.properties` file in the `/opt/lucidworks/fusion/4.1.x/conf` directory.

Install the Java JDK

Before upgrading to Fusion Server 4.2 or later, you must install the Java JDK on all nodes that will run Fusion Server. Prior versions of Fusion Server could use either the JRE or the JDK.

For more information, see *Java requirements*.

Download and install the newer version of Fusion

Perform these tasks *on all Fusion nodes*:

1. Select the Fusion release to which you are upgrading from *Fusion Server 4.x File Download Links*.
2. Extract the newer version of Fusion:

```
cd /opt/lucidworks mv ~/Downloads/fusion-4.2.y.tar.gz ./ tar -xf fusion-4.2.y.tar.gz
```

   For example, if Fusion is currently installed in `/opt/lucidworks/fusion/4.1.x` , then change your working directory to `/opt/lucidworks/` and extract the file there. *do not run the new version of Fusion yet.*
3. Ensure that the *new* version of Fusion has a valid permanent Fusion license before proceeding with the upgrade. Place a valid `license.properties` file in the `/opt/lucidworks/fusion/4.2.y/conf` directory.
4. (*If there are custom* `jar` *files*) If your deployment has custom `jar` files, add them to the new Fusion deployment.
5. (*If you are performing an upgrade without Internet access*) Without Internet access, the migrator cannot download new versions of connectors automatically. Download the *new versions* of connector zip files *for*

your current connectors from *Fusion 4.x Connector Downloads* and place them in `apps/connectors/bootstrap-plugins` for the new deployment.

6. (*If you are adding **new** connectors*) If you want your new deployment to use connectors that *are not* in the current deployment, you can add them now. Download the connector zip files from *Fusion 4.x Connector Downloads* and place them in `apps/connectors/bootstrap-plugins` for the new deployment.
7. Verify that there is sufficient disk space for a second copy of the Solr index directory, `fusion/4.1.x/data/solr`. If there is not sufficient disc space, free up space before proceeding.

Download and install the Fusion migrator

Perform these tasks *on all Fusion nodes*:

1. Download the latest migrator zip file for Unix. (Do this now, even if you have downloaded the migrator before, to ensure that you have the latest version.)
2. Create `FUSION_OLD` and `FUSION_NEW` environment variables that point to the old and new Fusion installation directories respectively (using the full path).

```
export FUSION_OLD="/opt/lucidworks/fusion/4.1.x" export
FUSION_NEW="/opt/lucidworks/fusion/4.2.y"
```

For example, when upgrading from Fusion 4.1.3 to 4.2.6:

```
export FUSION_OLD="/opt/lucidworks/fusion/4.1.3" export
FUSION_NEW="/opt/lucidworks/fusion/4.2.6"
```

3. Create this directory:

```
mkdir $FUSION_OLD/var/upgrade
```

4. Install the migrator:

```
tar -zxv -C $FUSION_OLD/var/upgrade --strip-components=1 -f
fusion-migrator.tar.gz
```

Run the migrator

Perform these tasks on the indicated nodes:

1. (*On all Fusion nodes*) Start all Fusion services for the old version of Fusion:

```
$FUSION_OLD/bin/fusion start
```

2. (*Only on the main Fusion node*) Run the migrator to export the configuration data from the old version of Fusion:

```
java -jar $FUSION_OLD/var/upgrade/migrator.jar --export
```

This message indicates that the step finished successfully:

```
Old Fusion configuration export (--export) finished
successfully.
```

3. (*On all Fusion nodes*) Stop the old versions of Fusion services and Solr; but not ZooKeeper:

```
$FUSION_OLD/bin/log-shipper stop $FUSION_OLD/bin/admin-ui stop
$FUSION_OLD/bin/webapps stop $FUSION_OLD/bin/proxy stop
$FUSION_OLD/bin/connectors-rpc stop $FUSION_OLD/bin/connectors-
classic stop $FUSION_OLD/bin/api stop $FUSION_OLD/bin/solr stop
```

If Spark and SQL services are running, also stop those:

```
$FUSION_OLD/bin/spark-master stop $FUSION_OLD/bin/spark-worker
stop $FUSION_OLD/bin/sql stop
```

Tip

You can see what is running with **`$FUSION_OLD/bin/fusion status`**.

4. (*Only on secondary Fusion nodes*) Prepare secondary nodes:

```
java -jar $FUSION_OLD/var/upgrade/migrator.jar --prepare-secondary
```

This message indicates that the step finished successfully:

```
Prepare secondary nodes (--prepare-secondary) finished successfully.
```

5. (*On all Fusion nodes*) Stop ZooKeeper for the old version of Fusion (*unless you are using an external ZooKeeper instance, in which case you can ignore this step*):

```
$FUSION_OLD/bin/zookeeper stop
```

6. (*Only on the main Fusion node*) Transform configuration data on the main Fusion node:

```
java -jar $FUSION_OLD/var/upgrade/migrator.jar --main-transform
```

Note

Depending on the size of your Solr index, this step can take a long time (for example, multiple tens of minutes).

This message indicates that the step finished successfully:

```
Fusion data transformations on main node (--main-transform) finished successfully.
```

7. (*On all Fusion nodes*) Start ZooKeeper for the new version of Fusion (*unless you are using an external ZooKeeper instance, in which case you can ignore this step*):

```
$FUSION_NEW/bin/zookeeper start
```

8. (*Only on the main Fusion node*) Import the first part of configuration data into the new version of Fusion:

```
java -jar $FUSION_OLD/var/upgrade/migrator.jar --zookeeper-import
```

This message indicates that the step finished successfully:

```
New Fusion Zookeeper import (--zookeeper-import) finished successfully.
```

9. (*On all Fusion nodes*) Start all Fusion services for the new version of Fusion:

```
$FUSION_NEW/bin/fusion start
```

10. (*Only on the main Fusion node*) Import the second part of configuration data into the new version of Fusion:

```
java -jar $FUSION_OLD/var/upgrade/migrator.jar --fusion-import
```

This message indicates that the step finished successfully:

```
New Fusion object import (--fusion-import) finished successfully.
```

Tip

After migration, you can find details about the results in the `fusion/4.1.x/var/upgrade/tmp` directory. If the migration produces unexpected results, the files in this directory are helpful for troubleshooting.

Validate the new version of Fusion

How to validate the new version of Fusion

1. (*Only on the main Fusion node*) Restart the new version of Fusion (all services defined in `fusion.properties`):

   ```
   $FUSION_NEW/bin/fusion restart
   ```

2. Log into the Fusion UI (your `admin` password is the same as for the old installation), and confirm the release number of the new version of Fusion:
   a. Clear your browser's cache.
   Otherwise, you might inadvertently access a cached version of the old Fusion UI and see inconsistent behavior.
   b. In a browser, open the Fusion UI at `http://localhost:8764/` (replace `localhost` with your server name or IP address if necessary).
   c. Log in.
   d. Navigate to **Admin** > **About Fusion**.
   The **About Fusion** panel should display the newer Fusion release number.

3. Ensure that all connectors were installed automatically during the upgrade.
   - For Fusion 4.x from the Fusion launcher, click the tile for a migrated app. Click **System** > **Blobs**. If any connectors are missing from the list, click **Add** > **Connector Plugin** and install them manually.
   - For Fusion 3.x from the Fusion launcher, click **Devops** > **Home** > **Blobs**. If any connectors are missing from the list, click **Add** > **Connector Plugin** and install them manually.

4. Ensure that all customizations you made in the former version of Fusion are present in the new one.
5. If you migrated from version 4.1.0 to version 4.2.0 or later, *confirm associations between blobs and apps* and make adjustments if necessary.
6. When you are satisfied with the migration and you have backed up the `fusion/4.1.x/` directory, you can `rm -fr fusion/4.1.x/` to remove the older version of Fusion (*on all Fusion nodes*).

Add support for business rules to existing apps

Fusion AI 4.2 introduces functionality for using business rules, that is, manually created formulas for rewriting queries and responses.

You can add support for business rules to apps that were created in versions of Fusion AI prior to version 4.2. To do so, perform the steps in this section.

The `add-rule-objects-xyz.zip` file (where `xyz` is a version number) specifies the objects to add to an app. It is supplied in the Fusion migrator zip file at the top level. After installing the migrator, the location is `$FUSION_OLD/var/upgrade/import-files`.

Note

Adding support for business rules has a costs. Additional collections and objects are created.
Only add support for business rules to apps in which you plan to use them.

You have a choice. You can update each app using the *Fusion UI* or the *Fusion API*.

FUSION UI

For each app in which you plan to use business rules, import the objects in the `add-rule-objects-xyz.zip` file into the app.

How to import business rule objects

1. In the Fusion launcher, click the app into which you want to import objects.
   The Fusion workspace appears.
2. In the upper left, click System > **Import Fusion Objects**.

   The Import Fusion Objects window opens.
3. For the data file, select `add-rule-objects-xyz.zip` from your local filesystem. The location in the extracted migrator files is `$FUSION_OLD/var/upgrade/import-files`.
4. Click **Import**.
5. Edit the `Application ID` parameter value to use the app name. If the app name contains spaces, replace those with underscore characters. For example, `Lucene Revolution` would become `Lucene_Revolution`.
6. Click **Import**.
7. If there are conflicts, Fusion prompts you to specify an import policy. Click **Merge** to skip all conflicting objects and import only the non-conflicting objects.
   Fusion confirms that the import was successful.
8. Click **Close** to close the Import Fusion Objects window.

FUSION API

For each app in which you plan to use business rules, import the objects in the `add-rule-objects-xyz.zip` file into the app.

How to import business rule objects

1. Create an `app-name.txt` file with the following content:

   ```
   {"app.id" : "_name-of-migrated-app_"}
   ```

   For example, for the app `Lucene Revolution`:

   ```
   {"app.id" : "Lucene_Revolution"}
   ```

   Here, we assume that you create the files in your home directory, for which the `$HOME` environment variable is defined.
2. Import the business rule objects:

```
curl -u USERNAME:PASSWORD -H "Content-Type:multipart/form-data"
-X POST -F 'variableValues=path-to-txt-file/app-name.txt' -F
"importData=path-to-zip-file/add-rule-objects-xyz.zip"
"http://<FUSION_HOST>/api/objects/import?importPolicy=merge"
```

For example:

```
curl -u admin:Very-SecurePW8 -H "Content-Type:multipart/form-
data" -X POST -F 'variableValues=$HOME/Lucene_Revolution.txt' -F
"importData=$FUSION_OLD/var/upgrade/import-files/add-rule-
objects-001.zip" "http://<FUSION_HOST>/api/objects/import?
importPolicy=merge"
```

Upgrade on Windows

Use this procedure to upgrade Fusion on a single Windows node or multiple Windows nodes.

Perform the steps in this procedure on the indicated nodes *on which Fusion is running* ("Fusion nodes"). To perform an upgrade, Fusion nodes must have at least these services running:

- API service (`api`)
- Proxy service (`proxy`)

Important

If you are upgrading Fusion on multiple nodes, then, for every step on multiple nodes, ensure that the step completes *on all Fusion nodes* before going to the next step. There is the notion of a "main node" during the migration process. This node will be used for certain centralized migration activities that do not need to be done on every node, such as downloading connectors that are then uploaded to blob storage that is shared by all, etc. Just pick one of your Fusion nodes to be the "main node"; there is no special requirement as to which one you pick.

Ensure that your current version of Fusion has a valid license

Ensure that your current version of Fusion has a valid permanent Fusion license before proceeding with the upgrade. Place a valid `license.properties` file in the `C:\lucidworks\fusion\4.1.x\conf` directory.

Install the Java JDK

Before upgrading to Fusion Server 4.2 or later, you must install the Java JDK on all nodes that will run Fusion Server. Prior versions of Fusion Server could use either the JRE or the JDK.

For more information, see *Java requirements*.

Download and install the newer version of Fusion

Perform these tasks *on all Fusion nodes*:

1. Select the Fusion release to which you are upgrading from *Fusion Server 4.x File Download Links*.
2. Move the `fusion-4.2.y.zip` file to the directory that contains the `fusion\` directory. For example, if Fusion is installed in `C:\lucidworks\fusion\4.1.x`, then move the file to `C:\lucidworks`.
3. Unzip the `fusion-4.2.y.zip` file. *do not run the new version of Fusion yet.*
4. For Fusion 4.x, ensure that the *new* version of Fusion has a valid permanent Fusion license before proceeding with the upgrade. Place a valid `license.properties` file in the `C:\lucidworks\fusion\4.2.y\conf` directory.
5. (*If there are custom* `jar` *files*) If your deployment has custom `jar` files, add them to the new Fusion deployment.
6. (*If you are performing an upgrade without Internet access*) Without Internet access, the migrator cannot download new versions of connectors automatically. Download the *new versions* of connector zip files *for your current connectors* from *Fusion 4.x Connector Downloads* and place them in `apps\connectors\bootstrap-plugins` for the new deployment.
7. (*If you are adding* ***new*** *connectors*) If you want your new deployment to use connectors that *are not* in the current deployment, you can add them now. Download the connector zip files from *Fusion 4.x Connector Downloads* and place them in `apps\connectors\bootstrap-plugins` for the new deployment.
8. Verify that there is sufficient disk space for a second copy of the Solr index directory, `fusion\4.1.x\data\solr`. If there is not sufficient disc space, free up space before proceeding.

Download and install the Fusion migrator

Perform these tasks *on all Fusion nodes*:

1. Download the latest migrator zip file for Windows. (Do this now, even if you have downloaded the migrator before, to ensure that you have the latest version.)
2. Open a Command Prompt window and create `FUSION_OLD` and `FUSION_NEW` environment variables that point to the old and new Fusion installation directories respectively. For example:

```
set FUSION_OLD=C:\lucidworks\fusion\4.1.3 set
FUSION_NEW=C:\lucidworks\fusion\4.2.6
```

3. Create a `fusion\4.1.x\var\upgrade` directory.
4. Unzip the migrator zip file, and move the contents of the extracted folder to `fusion\4.1.x\var\upgrade`.

Run the migrator

Perform these tasks on the indicated nodes:

1. (*On all Fusion nodes*) Start all Fusion services for the old version of Fusion:

```
%FUSION_OLD%\bin\fusion.cmd start
```

2. (*Only on the main Fusion node*) Run the migrator to export the configuration data from the old version of Fusion:

```
java -jar "%FUSION_OLD%\var\upgrade\migrator.jar" --export
```

This message indicates that the step finished successfully:

```
Old Fusion configuration export (--export) finished
successfully.
```

3. (*On all Fusion nodes*) Stop the old versions of Fusion services and Solr; but not ZooKeeper:

```
%FUSION_OLD%\bin\log-shipper.cmd stop %FUSION_OLD%\bin\admin-
ui.cmd stop %FUSION_OLD%\bin\webapps.cmd stop
%FUSION_OLD%\bin\proxy.cmd stop %FUSION_OLD%\bin\connectors-
rpc.cmd stop %FUSION_OLD%\bin\connectors-classic.cmd stop
%FUSION_OLD%\bin\api.cmd stop %FUSION_OLD%\bin\solr.cmd stop
```

If Spark and SQL services are running, also stop those:

```
%FUSION_OLD%\bin\spark-master.cmd stop %FUSION_OLD%\bin\spark-
worker.cmd stop %FUSION_OLD%\bin\sql.cmd stop
```

Tip

You can see what is running with `%FUSION_OLD%\bin\fusion status`.

4. (*Only on secondary Fusion nodes*) Prepare secondary nodes:

```
java -jar "%FUSION_OLD%\var\upgrade\migrator.jar" --prepare-
secondary
```

This message indicates that the step finished successfully:

```
Prepare secondary nodes (--prepare-secondary) finished
successfully.
```

5. (*On all Fusion nodes*) Stop ZooKeeper for the old version of Fusion (*unless you are using an external ZooKeeper instance, in which case you can ignore this step*):

```
%FUSION_OLD%\bin\zookeeper.cmd stop
```

6. (*Only on the main Fusion node*) Transform configuration data on the main Fusion node:

```
java -jar "%FUSION_OLD%\var\upgrade\migrator.jar" --main-
transform
```

Note

Depending on the size of your Solr index, this step can take a long time (for example, multiple tens of minutes).

This message indicates that the step finished successfully:

```
Fusion data transformations on main node (--main-transform)
finished successfully.
```

7. (*On all Fusion nodes*) Start ZooKeeper for the new version of Fusion (*unless you are only using an external ZooKeeper instance, in which case you can ignore this step*):

```
%FUSION_NEW%\bin\zookeeper.cmd start
```

8. (*Only on the main Fusion node*) Import the first part of configuration data into the new version of Fusion:

```
java -jar "%FUSION_OLD%\var\upgrade\migrator.jar" --zookeeper-import
```

This message indicates that the step finished successfully:

```
New Fusion Zookeeper import (--zookeeper-import) finished successfully.
```

9. (*On all Fusion nodes*) Start Solr for the new Fusion version:

```
%FUSION_NEW%\bin\solr.cmd start
```

10. (*Only on the main Fusion node*) Run a script to remove all old plugins from the blob store. Replace ***solr-address*** and ***solr-port*** as appropriate (as shown in the example):

```
java -cp "%FUSION_OLD%\var\upgrade\jython-standalone-2.7.1.jar;%FUSION_OLD%\var\upgrade\migrator.jar" org.python.util.jython "%FUSION_OLD%\var\upgrade\transformations\manual_delete_old_plugin_blobs.py" --solr-address solr-address --solr-port solr-port
```

For example:

```
java -cp "%FUSION_OLD%\var\upgrade\jython-standalone-
2.7.1.jar;%FUSION_OLD%\var\upgrade\migrator.jar"
org.python.util.jython
"%FUSION_OLD%\var\upgrade\transformations\manual_delete_old_plug
in_blobs.py" --solr-address localhost --solr-port 8983
```

This message indicates that plugins were deleted successfully:

```
Deleted old plugin blobs from solr <?xml version="1.0"
encoding="UTF-8"?> <response> <lst name="responseHeader"> <int
name="status">0</int> <int name="QTime">246</int> </lst>
</response> Old connector plugin blobs were deleted
successfully.
```

11. (*On all Fusion nodes*) Start all Fusion services for the new version of Fusion:

```
%FUSION_NEW%\bin\fusion.cmd start
```

12. (*Only on the main Fusion node*) Import the second part of configuration data into the new version of Fusion:

```
java -jar "%FUSION_OLD%\var\upgrade\migrator.jar" --fusion-
import
```

This message indicates that the step finished successfully:

```
New Fusion object import (--fusion-import) finished
successfully.
```

Tip

After migration, you can find details about the results in the `fusion\4.1.x\var\upgrade\tmp` directory. If the migration produces unexpected results, the files in this directory are helpful for troubleshooting.

Validate the new version of Fusion

How to validate the new version of Fusion

1. (*On all Fusion nodes*) Restart all Fusion services for the new version of Fusion:

```
%FUSION_NEW%\bin\fusion.cmd restart
```

2. Log into the Fusion UI (your `admin` password is the same as for the old installation), and confirm the release number of the new version of Fusion:
   a. Clear your browser's cache.
   Otherwise, you might inadvertently access a cached version of the old Fusion UI and see inconsistent behavior.
   b. In a browser, open the Fusion UI at `http://localhost:8764/` (replace `localhost` with your server name or IP address if necessary).
   c. Log in.
   d. Navigate to **Admin** > **About Fusion**.
   The **About Fusion** panel should display the newer Fusion release number.

3. Ensure that all connectors were installed automatically during the upgrade.
   a. For Fusion 4.x from the Fusion launcher, click the tile for a migrated app. Click **System** > **Blobs**. If any connectors are missing from the list, click **Add** > **Connector Plugin** and install them manually.
   b. For Fusion 3.x from the Fusion launcher, click **Devops** > **Home** > **Blobs**. If any connectors are missing from the list, click **Add** > **Connector Plugin** and install them manually.

4. Ensure that all customizations you made in the former version of Fusion are present in the new one.
5. If you migrated from version 4.1.0 to version 4.2.0 or later, *confirm associations between blobs and apps* and make adjustments if necessary.
6. When you are satisfied with the migration and you have backed up the `fusion\4.1.x` directory, you can remove the older version of Fusion by removing that directory (*on all Fusion nodes*).

Add support for business rules to existing apps

Fusion AI 4.2 introduces functionality for using business rules, that is, manually created formulas for rewriting queries and responses.

You can add support for business rules to apps that were created in versions of Fusion AI prior to version 4.2. To do so, perform the steps in this section.

The `add-rule-objects-xyz.zip` file (where `xyz` is a version number) specifies the objects to add to an app. It is supplied in the Fusion migrator zip file at the top level. After installing the migrator, the location is `%FUSION_OLD%\var\upgrade\import-files\`.

You have a choice. You can update each app using the *Fusion UI* or the *Fusion API*.

FUSION UI

For each app in which you plan to use business rules, import the objects in the `add-rule-objects-xyz.zip` file into the app.

How to import business rule objects

1. In the Fusion launcher, click the app into which you want to import objects.
   The Fusion workspace appears.
2. In the upper left, click System > **Import Fusion Objects**.

   The Import Fusion Objects window opens.
3. For the data file, select `add-rule-objects-xyz.zip` from your local filesystem. The location in the extracted migrator files is `%FUSION_OLD%\var\upgrade\import-files\`.
4. Click **Import**.
5. Edit the `Application ID` parameter value to use the app name. If the app name contains spaces, replace those with underscore characters. For example, `Lucene Revolution` would become `Lucene_Revolution`.
6. Click **Import**.
7. If there are conflicts, Fusion prompts you to specify an import policy. Click **Merge** to skip all conflicting objects and import only the non-conflicting objects.
   Fusion confirms that the import was successful.
8. Click **Close** to close the Import Fusion Objects window.

FUSION API

For each app in which you plan to use business rules, import the objects in the `add-rule-objects-xyz.zip` file into the app.

How to import business rule objects

1. Create an `app-name.txt` file with the following content:

   ```
   {"app.id" : "_name-of-migrated-app_"}
   ```

   For example, for the app `Lucene Revolution`:

   ```
   {"app.id" : "Lucene_Revolution"}
   ```

   Here, we assume that you create the files in your home directory, for which the `%HOMEPATH%` environment variable is defined.
2. Import the business rule objects:

```
curl -u USERNAME:PASSWORD -H "Content-Type:multipart/form-data"
-X POST -F 'variableValues=path-to-txt-file\app-name.txt' -F
"importData=path-to-zip-file\add-rule-objects-xyz.zip"
"http://<FUSION_HOST>/api/objects/import?importPolicy=merge"
```

For example:

```
curl -u admin:Very-SecurePW8 -H "Content-Type:multipart/form-
data" -X POST -F
'variableValues=%HOMEPATH%\Lucene_Revolution.txt' -F
"importData=%FUSION_OLD%\import-files\add-rule-objects-001.zip"
"http://<FUSION_HOST>/api/objects/import?importPolicy=merge"
```

Upgrade Fusion Server 4.2.6 to Latest SP

How-to

> Compatible versions: 4.2
>
> Works with: *Fusion Server*

This article describes how to perform the following upgrade:

- From version: 4.2.6
- To version: Latest SP

Unlike other Fusion upgrades, 4.2.6 SPs are a patched upgrade that can be applied directly to Fusion 4.2.6. If you are not already using Fusion 4.2.6, *complete the upgrade to Fusion 4.2.6* before installing the latest Fusion 4.2.6 SP.

You can view the application files to download at *Fusion Server 4.x File Download Links*.

Important

This upgrade is available to qualified customers whose Lucidworks plans include extended support.

Consult *Upgrade to Fusion 4.x* for information on upgrading to a different version of Fusion.

If you require a fresh installation of a Fusion 4.2.6 SP, consult *Install Fusion 4.x*.

Preliminary upgrade steps (optional)

If you are upgrading to apply the latest Log4j updates to your existing deployment, please follow the proceeding steps.

1. Log in to the Fusion Admin UI with an administrator account.
2. Select **System** → **blobs**
3. Open **webapp-admin-war** → **admin-webapps**
4. Delete the `admin.war`, `insights.war`, `app-studio.war` **WAR** by selecting each entry and clicking **Delete Blob**.

Tip

For more information about blobs, see *Blob Storage*.

Upgrade an existing Fusion installation (Linux, Windows)

Linux

To upgrade an existing Fusion installation on Linux:

1. Place the `fusion-4.2.6-service-pack-patch.jar` and `fusion-4.2.6-sp3.tar.gz` on each server hosting Fusion services.

   Note

   The script does not update external Solr and Zookeeper installations.

2. Shut down all Fusion services on each server that will receive the patched upgrade.
3. Run the `fusion-4.2.6-service-pack-patch.jar` jar (`java -jar fusion-4.2.6-service-pack-patch.jar`) and follow the prompts.
4. Start the Fusion services.

Windows

To upgrade an existing Fusion installation on Windows:

1. Place the `fusion-4.2.6-service-pack-patch.jar` and `fusion-4.2.6-sp3.zip` on each server hosting Fusion services.

> **Note**
>
> The script does not update external Solr and Zookeeper installations.

2. Shut down all Fusion services on each server that will receive the patched upgrade.
3. Run the `fusion-4.2.6-service-pack-patch.jar` jar (`java -jar fusion-4.2.6-service-pack-patch.jar`) and follow the prompts.
4. Start the Fusion services.

Enable the Tika parser service

Fusion 4.2.6 SPs include *an asynchronous Tika parser service*. This service is not automatically enabled upon upgrading. To enable the Tika parser service:

1. Open `fusion/conf/fusion.properties` in a text editor.
2. Add the following Tika entry to the end of the file.

```
# Tika parser service tika-server.port = 8889 # If your tika
server is properly secured behind a firewall, then you can then
set the # -Dtika.server.allowLocalFileOptimization=true to allow
you to use the "local file optimization" which reduces # http
overhead during local tika parses. tika-server.jvmOptions = -
Dtika.MAX_HEAP_MB=1000 -Dtika.NUM_PROCESSES=2 -
Dtika.ALLOW_LOCAL_FILE_OPTIMIZATION=true
```

3. In the `log-shipper` section of the file, add the following in a new line:

```
log-shipper.logs.tika-server = java
```

4. Save the file.
5. Download the tika-server-log4j2.xml file .
6. Add the TikaServer logging configuration to the `fusion/conf` directory.

> **Note**
>
> To automatically start the Tika parser service when starting Fusion, add `tika-server` to the `group.default` line. Otherwise, you can start the Tika parser server with the `$fusion/bin/tika-server` script.

Upgrade Fusion Server 4.2.x to 4.2.y

How-to

> Compatible versions: 4.2
>
> Works with: *Fusion Server*

Introduction

This article describes how to perform the following upgrade:

- From version: Fusion `4.2.x`
- To version: Fusion `4.2.y`

Important

Only specific version-to-version upgrade sequences are supported. Some upgrades require multiple steps.

For more information, see *Fusion 4.x*.

For Fusion 3.1 and later releases, a migrator is available for upgrading Fusion.

During the upgrade process, the migrator uses a properties file. After downloading and installing the migrator, the properties files is in the `/opt/lucidworks/fusion/4.2.x/var/upgrade` directory (on Unix or MacOS) or the `C:\lucidworks\fusion\4.2.x\var\upgrade\` directory (on Windows). The file names reference the versions you are upgrading from and to. For example:

- To upgrade `4.2.3` to `4.2.5`, the migrator uses the `4.2.x-4.2.x.properties` file.

Migration entails down time and multiple starts and stops of Fusion services. Plan accordingly, especially in terms of disabling external load balancers or monitors that might react adversely to the starts and stops.

Download the latest migrator immediately before upgrading. This helps ensure that the upgrade goes smoothly.

Important

The newer Fusion instance must be newly untarred and *never started*.

About the upgrade

This section describes how connectors, object migrations, and signals are migrated during an upgrade.

Connectors

In Fusion 3.1.0 and later, only a subset of *connectors* are included by default.

The migrator detects which connectors were used in the older version of Fusion, and installs them automatically in Fusion `4.2.y`. This step requires an Internet connection. If no connection is available, then download the connectors from *Fusion 4.x Connector Downloads* and *install them as bootstrap plugins*.

If a connector to be upgraded was not available during the upgrade, then a message in `/opt/lucidworks/fusion/4.2.x/var/upgrade/tmp/migrator.log` (on Unix) or `C:\lucidworks\fusion\4.2.x\var\upgrade\tmp\migrator.log` (on Windows) indicates this.

Only datasources for connectors that are supported in the new Fusion version are upgraded. Datasources for custom connectors are not upgraded.

IF NO INTERNET CONNECTION IS AVAILABLE

If no Internet connection is available during an upgrade, the migrator cannot automatically download the connectors it needs. Using the Fusion UI or API later to install the connectors also might not be an option.

In this case, download the connectors from *Fusion 4.x Connector Downloads* for all existing connectors and place them in `apps/connectors/bootstrap-plugins` for the new deployment (on all Fusion nodes). Do so at the time indicated in the procedures that follow.

ADDING CONNECTORS DURING AN UPGRADE

You can *add* connectors during an upgrade (that is, add connectors that are not in the old deployment).

Download the connectors from *Fusion 4.x Connector Downloads* and place them in `apps/connectors/bootstrap-plugins` for the new version (on all Fusion nodes).

Object migrations and transformations

The migrator automatically migrates these Fusion 4.2 object types, transforming them as needed:

- Collections
- Index pipelines
- Query pipelines
- Search cluster configurations
- Datasources
- Parsing configurations
- Object groups
- Links
- Tasks
- Jobs
- Spark objects
- Apps
- Appkit apps
- Index profiles
- Query profiles
- Blobs

Important

In Fusion Server 4.0 and later, most objects exist in the context of apps. When you upgrade from Fusion Server 4.2.x to 4.2.y, the migrator upgrades app objects, all objects in or linked to objects in apps, and objects that are not linked to apps. You can explore the objects in *Object Explorer*.

Access control migration

The migrator upgrades all access control configurations:

- Security realms
- Roles
- Users

None of these should require adjustments after migration.

Review known issues

Before upgrading, *review the known issues* to see whether any of them apply to the circumstances of your upgrade. Some known issues might require actions *before* upgrading.

That article also contains instructions regarding what to do *if an upgrade step fails*.

Upgrade on Unix

Use this procedure to upgrade Fusion on a single Unix node or on multiple Unix nodes.

Perform the steps in this procedure on the indicated nodes *on which Fusion is running* ("Fusion nodes"). To perform an upgrade, Fusion nodes must have at least these services running:

- API service (`api`)
- Proxy service (`proxy`)

Important

For every step on multiple nodes, ensure that the step completes *on all Fusion nodes* before going to the next step. There is the notion of a "main node" during the migration process. This node will be used for certain centralized migration activities that do not need to be done on every node, such as downloading connectors that are then uploaded to blob storage that is shared by all, etc. Just pick one of your Fusion nodes to be the "main node"; there is no special requirement as to which one you pick.

Ensure that your current version of Fusion has a valid license

Ensure that your *current* version of Fusion has a valid permanent Fusion license before proceeding with the upgrade. Place a valid `license.properties` file in the `/opt/lucidworks/fusion/4.2.x/conf` directory.

Download and install the newer version of Fusion

Perform these tasks *on all Fusion nodes*:

1. Select the Fusion release to which you are upgrading from *Fusion Server 4.x File Download Links*.
2. Extract the newer version of Fusion:

   ```
   cd /opt/lucidworks mv ~/Downloads/fusion-4.2.y.tar.gz ./ tar -xf fusion-4.2.y.tar.gz
   ```

   For example, if Fusion is currently installed in `/opt/lucidworks/fusion/4.2.x`, then change your working directory to `/opt/lucidworks/` and extract the file there. *do not run the new version of Fusion yet.*
3. Ensure that the *new* version of Fusion has a valid permanent Fusion license before proceeding with the upgrade. Place a valid `license.properties` file in the `/opt/lucidworks/fusion/4.2.y/conf` directory.
4. (*If there are custom* `jar` *files*) If your deployment has custom `jar` files, add them to the new Fusion deployment.
5. (*If you are performing an upgrade without Internet access*) Without Internet access, the migrator cannot download new versions of connectors automatically. Download the *new versions* of connector zip files *for your current connectors* from *Fusion 4.x Connector Downloads* and place them in `apps/connectors/bootstrap-plugins` for the new deployment.
6. (*If you are adding* ***new*** *connectors*) If you want your new deployment to use connectors that *are not* in the current deployment, you can add them now. Download the connector zip files from *Fusion 4.x Connector Downloads* and place them in `apps/connectors/bootstrap-plugins` for the new deployment.
7. Verify that there is sufficient disk space for a second copy of the Solr index directory, `fusion/4.2.x/data/solr`. If there is not sufficient disk space, free up space before proceeding.

Download and install the Fusion migrator

Perform these tasks *on all Fusion nodes*:

1. Download the latest migrator zip file for Unix. (Do this now, even if you have downloaded the migrator before, to ensure that you have the latest version.)
2. Create `FUSION_OLD` and `FUSION_NEW` environment variables that point to the old and new Fusion installation directories respectively (using the full path).

```
export FUSION_OLD="/opt/lucidworks/fusion/4.2.x" export
FUSION_NEW="/opt/lucidworks/fusion/4.2.y"
```

For example, when upgrading from Fusion 4.2.0 to 4.2.6:

```
export FUSION_OLD="/opt/lucidworks/fusion/4.2.0" export
FUSION_NEW="/opt/lucidworks/fusion/4.2.6"
```

3. Create this directory:

```
mkdir $FUSION_OLD/var/upgrade
```

4. Install the migrator:

```
tar -zxv -C $FUSION_OLD/var/upgrade --strip-components=1 -f
fusion-migrator.tar.gz
```

Run the migrator

Perform these tasks on the indicated nodes:

1. (*On all Fusion nodes*) Start all Fusion services for the old version of Fusion:

```
$FUSION_OLD/bin/fusion start
```

2. (*Only on the main Fusion node*) Run the migrator to export the configuration data from the old version of Fusion:

```
java -jar $FUSION_OLD/var/upgrade/migrator.jar --export
```

This message indicates that the step finished successfully:

```
Old Fusion configuration export (--export) finished
successfully.
```

3. (*On all Fusion nodes*) Stop the old versions of Fusion services and Solr; but not ZooKeeper:

```
$FUSION_OLD/bin/log-shipper stop $FUSION_OLD/bin/sql stop
$FUSION_OLD/bin/admin-ui stop $FUSION_OLD/bin/webapps stop
$FUSION_OLD/bin/proxy stop $FUSION_OLD/bin/connectors-rpc stop
$FUSION_OLD/bin/connectors-classic stop $FUSION_OLD/bin/api stop
$FUSION_OLD/bin/solr stop
```

If Spark services are running, also stop those:

```
$FUSION_OLD/bin/spark-master stop $FUSION_OLD/bin/spark-worker
stop
```

Tip

You can see what is running with **`$FUSION_OLD/bin/fusion status`**.

4. (*Only on secondary Fusion nodes*) Prepare secondary nodes:

```
java -jar $FUSION_OLD/var/upgrade/migrator.jar --prepare-
secondary
```

This message indicates that the step finished successfully:

```
Prepare secondary nodes (--prepare-secondary) finished
successfully.
```

5. (*On all Fusion nodes*) Stop ZooKeeper for the old version of Fusion (*unless you are using an external ZooKeeper instance, in which case you can ignore this step*):

```
$FUSION_OLD/bin/zookeeper stop
```

6. (*Only on the main Fusion node*) Transform configuration data on the main Fusion node:

```
java -jar $FUSION_OLD/var/upgrade/migrator.jar --main-transform
```

Note

Depending on the size of your Solr index, this step can take a long time (for example, multiple tens of minutes).

This message indicates that the step finished successfully:

```
Fusion data transformations on main node (--main-transform)
finished successfully.
```

7. (*On all Fusion nodes*) Start ZooKeeper for the new version of Fusion (*unless you are using an external ZooKeeper instance, in which case you can ignore this step*):

```
$FUSION_NEW/bin/zookeeper start
```

8. (*Only on the main Fusion node*) Import the first part of configuration data into the new version of Fusion:

```
java -jar $FUSION_OLD/var/upgrade/migrator.jar --zookeeper-import
```

This message indicates that the step finished successfully:

```
New Fusion Zookeeper import (--zookeeper-import) finished successfully.
```

9. (*On all Fusion nodes*) Start all Fusion services for the new version of Fusion:

```
$FUSION_NEW/bin/fusion start
```

10. (*Only on the main Fusion node*) Import the second part of configuration data into the new version of Fusion:

```
java -jar $FUSION_OLD/var/upgrade/migrator.jar --fusion-import
```

This message indicates that the step finished successfully:

```
New Fusion object import (--fusion-import) finished successfully.
```

Tip

After migration, you can find details about the results in the `fusion/4.2.x/var/upgrade/tmp` directory. If the migration produces unexpected results, the files in this directory are helpful for troubleshooting.

Validate the new version of Fusion

How to validate the new version of Fusion

1. (*Only on the main Fusion node*) Restart the new version of Fusion (all services defined in `fusion.properties`):

```
$FUSION_NEW/bin/fusion restart
```

2. Log into the Fusion UI (your `admin` password is the same as for the old installation), and confirm the release number of the new version of Fusion:
   a. Clear your browser's cache.
   Otherwise, you might inadvertently access a cached version of the old Fusion UI and see inconsistent behavior.
   b. In a browser, open the Fusion UI at `http://localhost:8764/` (replace `localhost` with your server name or IP address if necessary).
   c. Log in.
   d. Navigate to **Admin** > **About Fusion**.
   The **About Fusion** panel should display the newer Fusion release number.

3. Ensure that all connectors were installed automatically during the upgrade.
   - For Fusion 4.x from the Fusion launcher, click the tile for a migrated app. Click **System** > **Blobs**. If any connectors are missing from the list, click **Add** > **Connector Plugin** and install them manually.
   - For Fusion 3.x from the Fusion launcher, click **Devops** > **Home** > **Blobs**. If any connectors are missing from the list, click **Add** > **Connector Plugin** and install them manually.

4. Ensure that all customizations you made in the former version of Fusion are present in the new one.
5. When you are satisfied with the migration and you have backed up the `fusion/4.2.x/` directory, you can `rm -fr fusion/4.2.x/` to remove the older version of Fusion (*on all Fusion nodes*).

Add support for business rules to existing apps

Fusion AI 4.2 introduces functionality for using business rules, that is, manually created formulas for rewriting queries and responses.

You can add support for business rules to apps that were created in versions of Fusion AI prior to version 4.2. To do so, perform the steps in this section.

The `add-rule-objects-xyz.zip` file (where `xyz` is a version number) specifies the objects to add to an app. It is supplied in the Fusion migrator zip file at the top level. After installing the migrator, the location is `$FUSION_OLD/var/upgrade/import-files`.

Note

Adding support for business rules has a costs. Additional collections and objects are created.
Only add support for business rules to apps in which you plan to use them.

You have a choice. You can update each app using the *Fusion UI* or the *Fusion API*.

FUSION UI

For each app in which you plan to use business rules, import the objects in the `add-rule-objects-xyz.zip` file into the app.

How to import business rule objects

1. In the Fusion launcher, click the app into which you want to import objects.
   The Fusion workspace appears.
2. In the upper left, click System > **Import Fusion Objects**.

   The Import Fusion Objects window opens.
3. For the data file, select `add-rule-objects-xyz.zip` from your local filesystem. The location in the extracted migrator files is `$FUSION_OLD/var/upgrade/import-files`.
4. Click **Import**.
5. Edit the `Application ID` parameter value to use the app name. If the app name contains spaces, replace those with underscore characters. For example, `Lucene Revolution` would become `Lucene_Revolution`.
6. Click **Import**.
7. If there are conflicts, Fusion prompts you to specify an import policy. Click **Merge** to skip all conflicting objects and import only the non-conflicting objects.
   Fusion confirms that the import was successful.
8. Click **Close** to close the Import Fusion Objects window.

FUSION API

For each app in which you plan to use business rules, import the objects in the `add-rule-objects-xyz.zip` file into the app.

How to import business rule objects

1. Create an `app-name.txt` file with the following content:

   ```
   {"app.id" : "_name-of-migrated-app_"}
   ```

   For example, for the app `Lucene Revolution`:

   ```
   {"app.id" : "Lucene_Revolution"}
   ```

   Here, we assume that you create the files in your home directory, for which the `$HOME` environment variable is defined.
2. Import the business rule objects:

   ```
   curl -u USERNAME:PASSWORD -H "Content-Type:multipart/form-data"
   -X POST -F 'variableValues=path-to-txt-file/app-name.txt' -F
   "importData=path-to-zip-file/add-rule-objects-xyz.zip"
   "http://<FUSION_HOST>/api/objects/import?importPolicy=merge"
   ```

For example:

```
curl -u admin:Very-SecurePW8 -H "Content-Type:multipart/form-
data" -X POST -F 'variableValues=$HOME/Lucene_Revolution.txt' -F
"importData=$FUSION_OLD/var/upgrade/import-files/add-rule-
objects-001.zip" "http://<FUSION_HOST>/api/objects/import?
importPolicy=merge"
```

Upgrade on Windows

Use this procedure to upgrade Fusion on a single Windows node or multiple Windows nodes.

Perform the steps in this procedure on the indicated nodes *on which Fusion is running* ("Fusion nodes"). To perform an upgrade, Fusion nodes must have at least these services running:

- API service (`api`)
- Proxy service (`proxy`)

Important

If you are upgrading Fusion on multiple nodes, then, for every step on multiple nodes, ensure that the step completes *on all Fusion nodes* before going to the next step. There is the notion of a "main node" during the migration process. This node will be used for certain centralized migration activities that do not need to be done on every node, such as downloading connectors that are then uploaded to blob storage that is shared by all, etc. Just pick one of your Fusion nodes to be the "main node"; there is no special requirement as to which one you pick.

Ensure that your current version of Fusion has a valid license

Ensure that your current version of Fusion has a valid permanent Fusion license before proceeding with the upgrade. Place a valid `license.properties` file in the `C:\lucidworks\fusion\4.2.x\conf` directory.

Download and install the newer version of Fusion

Perform these tasks *on all Fusion nodes*:

1. Select the Fusion release to which you are upgrading from *Fusion Server 4.x File Download Links*.
2. Move the `fusion-4.2.y.zip` file to the directory that contains the `fusion\` directory. For example, if Fusion is installed in `C:\lucidworks\fusion\4.2.x`, then move the file to `C:\lucidworks`.
3. Unzip the `fusion-4.2.y.zip` file. *do not run the new version of Fusion yet.*
4. For Fusion 4.x, ensure that the *new* version of Fusion has a valid permanent Fusion license before proceeding with the upgrade. Place a valid `license.properties` file in the `C:\lucidworks\fusion\4.2.y\conf` directory.
5. (*If there are custom* `jar` *files*) If your deployment has custom `jar` files, add them to the new Fusion deployment.
6. (*If you are performing an upgrade without Internet access*) Without Internet access, the migrator cannot download new versions of connectors automatically. Download the *new versions* of connector zip files *for your current connectors* from *Fusion 4.x Connector Downloads* and place them in `apps\connectors\bootstrap-plugins` for the new deployment.
7. (*If you are adding* ***new*** *connectors*) If you want your new deployment to use connectors that *are not* in the current deployment, you can add them now. Download the connector zip files from *Fusion 4.x Connector*

Downloads and place them in `apps\connectors\bootstrap-plugins` for the new deployment.

8. Verify that there is sufficient disk space for a second copy of the Solr index directory, `fusion\4.2.x\data\solr`. If there is not sufficient disk space, free up space before proceeding.

Download and install the Fusion migrator

Perform these tasks *on all Fusion nodes*:

1. Download the latest migrator zip file for Windows . (Do this now, even if you have downloaded the migrator before, to ensure that you have the latest version.)
2. Open a Command Prompt window and create `FUSION_OLD` and `FUSION_NEW` environment variables that point to the old and new Fusion installation directories respectively. For example:

```
set FUSION_OLD=C:\lucidworks\fusion\4.2.0 set
FUSION_NEW=C:\lucidworks\fusion\4.2.6
```

3. Create a `fusion\4.2.x\var\upgrade` directory.
4. Unzip the migrator zip file, and move the contents of the extracted folder to `fusion\4.2.x\var\upgrade`.

Run the migrator

Perform these tasks on the indicated nodes:

1. (*On all Fusion nodes*) Start all Fusion services for the old version of Fusion:

```
%FUSION_OLD%\bin\fusion.cmd start
```

2. (*Only on the main Fusion node*) Run the migrator to export the configuration data from the old version of Fusion:

```
java -jar "%FUSION_OLD%\var\upgrade\migrator.jar" --export
```

This message indicates that the step finished successfully:

```
Old Fusion configuration export (--export) finished
successfully.
```

3. (*On all Fusion nodes*) Stop the old versions of Fusion services and Solr; but not ZooKeeper:

```
%FUSION_OLD%\bin\log-shipper.cmd stop %FUSION_OLD%\bin\sql.cmd
stop %FUSION_OLD%\bin\admin-ui.cmd stop
%FUSION_OLD%\bin\webapps.cmd stop %FUSION_OLD%\bin\proxy.cmd
stop %FUSION_OLD%\bin\connectors-rpc.cmd stop
%FUSION_OLD%\bin\connectors-classic.cmd stop
%FUSION_OLD%\bin\api.cmd stop %FUSION_OLD%\bin\solr.cmd stop
```

If Spark and SQL services are running, also stop those:

```
%FUSION_OLD%\bin\spark-master.cmd stop %FUSION_OLD%\bin\spark-
worker.cmd stop
```

Tip

You can see what is running with `%FUSION_OLD%\bin\fusion status`.

4. (*Only on secondary Fusion nodes*) Prepare secondary nodes:

```
java -jar "%FUSION_OLD%\var\upgrade\migrator.jar" --prepare-
secondary
```

This message indicates that the step finished successfully:

```
Prepare secondary nodes (--prepare-secondary) finished
successfully.
```

5. (*On all Fusion nodes*) Stop ZooKeeper for the old version of Fusion (*unless you are using an external ZooKeeper instance, in which case you can ignore this step*):

```
%FUSION_OLD%\bin\zookeeper.cmd stop
```

6. (*Only on the main Fusion node*) Transform configuration data on the main Fusion node:

```
java -jar "%FUSION_OLD%\var\upgrade\migrator.jar" --main-transform
```

Note

Depending on the size of your Solr index, this step can take a long time (for example, multiple tens of minutes).

This message indicates that the step finished successfully:

```
Fusion data transformations on main node (--main-transform) finished successfully.
```

7. (*On all Fusion nodes*) Start ZooKeeper for the new version of Fusion (*unless you are only using an external ZooKeeper instance, in which case you can ignore this step*):

```
%FUSION_NEW%\bin\zookeeper.cmd start
```

8. (*Only on the main Fusion node*) Import the first part of configuration data into the new version of Fusion:

```
java -jar "%FUSION_OLD%\var\upgrade\migrator.jar" --zookeeper-import
```

This message indicates that the step finished successfully:

```
New Fusion Zookeeper import (--zookeeper-import) finished
successfully.
```

9. (*On all Fusion nodes*) Start Solr for the new Fusion version:

```
%FUSION_NEW%\bin\solr.cmd start
```

10. (*Only on the main Fusion node*) Run a script to remove all old plugins from the blob store. Replace ***solr-address*** and ***solr-port*** as appropriate (as shown in the example):

```
java -cp "%FUSION_OLD%\var\upgrade\jython-standalone-
2.7.1.jar;%FUSION_OLD%\var\upgrade\migrator.jar"
org.python.util.jython
"%FUSION_OLD%\var\upgrade\transformations\manual_delete_old_plug
in_blobs.py" --solr-address solr-address --solr-port solr-port
```

For example:

```
java -cp "%FUSION_OLD%\var\upgrade\jython-standalone-
2.7.1.jar;%FUSION_OLD%\var\upgrade\migrator.jar"
org.python.util.jython
"%FUSION_OLD%\var\upgrade\transformations\manual_delete_old_plug
in_blobs.py" --solr-address localhost --solr-port 8983
```

This message indicates that plugins were deleted successfully:

```
Deleted old plugin blobs from solr <?xml version="1.0"
encoding="UTF-8"?> <response> <lst name="responseHeader"> <int
name="status">0</int> <int name="QTime">246</int> </lst>
</response> Old connector plugin blobs were deleted
successfully.
```

11. (*On all Fusion nodes*) Start all Fusion services for the new version of Fusion:

```
%FUSION_NEW%\bin\fusion.cmd start
```

12. (*Only on the main Fusion node*) Import the second part of configuration data into the new version of Fusion:

```
java -jar "%FUSION_OLD%\var\upgrade\migrator.jar" --fusion-
import
```

This message indicates that the step finished successfully:

```
New Fusion object import (--fusion-import) finished
successfully.
```

Tip

After migration, you can find details about the results in the `fusion\4.2.x\var\upgrade\tmp` directory. If the migration produces unexpected results, the files in this directory are helpful for troubleshooting.

Validate the new version of Fusion

How to validate the new version of Fusion

1. (*On all Fusion nodes*) Restart all Fusion services for the new version of Fusion:

```
%FUSION_NEW%\bin\fusion.cmd restart
```

2. Log into the Fusion UI (your **admin** password is the same as for the old installation), and confirm the release number of the new version of Fusion:
    a. Clear your browser's cache.
    Otherwise, you might inadvertently access a cached version of the old Fusion UI and see inconsistent behavior.
    b. In a browser, open the Fusion UI at **http://localhost:8764/** (replace **localhost** with your server name or IP address if necessary).
    c. Log in.
    d. Navigate to **Admin** > **About Fusion**.
    The **About Fusion** panel should display the newer Fusion release number.

3. Ensure that all connectors were installed automatically during the upgrade.
    a. For Fusion 4.x from the Fusion launcher, click the tile for a migrated app. Click **System** > **Blobs**. If any connectors are missing from the list, click **Add** > **Connector Plugin** and install them manually.
    b. For Fusion 3.x from the Fusion launcher, click **Devops** > **Home** > **Blobs**. If any connectors are missing from the list, click **Add** > **Connector Plugin** and install them manually.
4. Ensure that all customizations you made in the former version of Fusion are present in the new one.
5. When you are satisfied with the migration and you have backed up the **fusion\4.2.*x*** directory, you can remove the older version of Fusion by removing that directory (*on all Fusion nodes*).

Add support for business rules to existing apps

Fusion AI 4.2 introduces functionality for using business rules, that is, manually created formulas for rewriting queries and responses.

You can add support for business rules to apps that were created in versions of Fusion AI prior to version 4.2. To do so, perform the steps in this section.

The **add-rule-objects-*xyz*.zip** file (where ***xyz*** is a version number) specifies the objects to add to an app. It is supplied in the Fusion migrator zip file at the top level. After installing the migrator, the location is **%FUSION_OLD%\var\upgrade\import-files**.

You have a choice. You can update each app using the *Fusion UI* or the *Fusion API*.

FUSION UI

For each app in which you plan to use business rules, import the objects in the **add-rule-objects-*xyz*.zip** file into the app.

How to import business rule objects

1. In the Fusion launcher, click the app into which you want to import objects.
The Fusion workspace appears.
2. In the upper left, click System > **Import Fusion Objects**.
The Import Fusion Objects window opens.
3. For the data file, select **add-rule-objects-*xyz*.zip** from your local filesystem. The location in the extracted migrator files is **%FUSION_OLD%\var\upgrade\import-files**.
4. Click **Import**.
5. Edit the **Application ID** parameter value to use the app name. If the app name contains spaces, replace those with underscore characters. For example, **Lucene Revolution** would become

`Lucene_Revolution`.

6. Click **Import**.
7. If there are conflicts, Fusion prompts you to specify an import policy. Click **Merge** to skip all conflicting objects and import only the non-conflicting objects.
   Fusion confirms that the import was successful.
8. Click **Close** to close the Import Fusion Objects window.

FUSION API

For each app in which you plan to use business rules, import the objects in the `add-rule-objects-xyz.zip` file into the app.

How to import business rule objects

1. Create an `app-name.txt` file with the following content:

```
{"app.id" : "_name-of-migrated-app_"}
```

   For example, for the app `Lucene Revolution`:

```
{"app.id" : "Lucene_Revolution"}
```

   Here, we assume that you create the files in your home directory, for which the `%HOMEPATH%` environment variable is defined.
2. Import the business rule objects:

```
curl -u USERNAME:PASSWORD -H "Content-Type:multipart/form-data"
-X POST -F 'variableValues=path-to-txt-file\app-name.txt' -F
"importData=path-to-zip-file\add-rule-objects-xyz.zip"
"http://<FUSION_HOST>/api/objects/import?importPolicy=merge"
```

   For example:

```
curl -u admin:Very-SecurePW8 -H "Content-Type:multipart/form-data" -X POST -F 'variableValues=%HOMEPATH%\Lucene_Revolution.txt' -F "importData=%FUSION_OLD%\import-files\add-rule-objects-001.zip" "http://<FUSION_HOST>/api/objects/import?importPolicy=merge"
```

Upgrade the Appkit Social Module

How-to

> Compatible versions: 4.2
>
> Works with: *App Studio*

Note

These steps are only required if you are currently using the Appkit Social module - this is indicated by the presence of a `<dependency>` block in the `pom.xml` file at the root of your project that has an `<artifactId>` entry that starts with `twigkit.social`. If you are not using the Appkit Social module (i.e. there is no `twigkit.social` entry in your `pom.xml`), then you do not need to follow the steps in this section.

How to upgrade the Appkit Social module

1. Remove the existing `twigkit.social` `<dependency>` block from your `pom.xml` file and replace it with the following:

```
<dependency> <groupId>twigkit</groupId>
<artifactId>twigkit.social.provider.platform</artifactId>
<version>${project.parent.version}</version> </dependency>
```

2. Create a configuration file named `src/main/resources/conf/social/social.conf` with the following contents:

```
# Use conf/platforms/fusion/social.conf for Social module
platform configuration platform: platforms.fusion.social
```

   The `platform` value indicates the name of the platform configuration file to use, which you will create in the next step.

3. Create a configuration file named `src/main/resources/conf/platforms/fusion/social.conf` that contains the following (note that the name of the file matches the `platform` value configured in the previous step):

```
# For every Fusion App, a corresponding
YOUR_FUSION_APP_user_prefs collection is automatically created.
# Appkit uses this collection for storing collaboration user-
data. # Name of the collection to store user data in collection:
YOUR_FUSION_APP_user_prefs # Name of the Query & Index pipelines
to use - _system is the Fusion default # pipeline: _system #
Enable writing new Social entities into the platform readOnly:
false # Prevent users from searching this platform directly
webservice-enabled: false
```

Replace the string **`YOUR_FUSION_APP`** in the **`collection`** parameter with the name of the Fusion app you want to associate your social data with.

Upgrade to Fusion 4.x

How-to

> Compatible versions: 4.2
>
> Works with: *Fusion Server*

When you have a Fusion-based search application running, at some point it might be necessary to upgrade to a later version of Fusion. We provide a migrator tool to simplify the upgrade process.

Tip

See the *release history* to find out what is new, including which versions of Solr, Spark, and ZooKeeper are bundled with each Fusion release.

The migrator transfers over *most* of the objects that make up your search application, all configurations and customizations for your application, and all data in collections in the application.

Note

In some cases, manual steps are required for objects that the migrator cannot handle automatically. We give you instructions and guidance about what might be required. You should also review the log of the upgrade in `/opt/fusion/x.y.z/var/upgrade/tmp/migrator.log` (on Unix) or `C:\lucidworks\var\fusion\x.y.z\upgrade\tmp\migrator.log` (on Windows). The *x.y.z* directory is for the Fusion version that you are migrating *from*.

Key points

Following are some key points about upgrading Fusion:

- **Migration involves down time.** The upgrade process involves multiple starts and stops of Fusion services. Please plan accordingly, especially in terms of disabling external load balancers or monitors that might react adversely to the starts and stops.
- **Current deployment is preserved.** Upgrades preserve the current Fusion deployment, copying information over from the current deployment to the new one. This provides a rapid roll-back option if you encounter problems during the upgrade process.
- **If the upgrade fails.** If an upgrade fails, *there is a procedure for dealing with that*.

Supported upgrade sequences

Important

Only specific version-to-version upgrade sequences are supported. Some upgrades require multiple steps.

These upgrade sequences are supported.

Upgrades to the current version

- **3.1.x to 4.2.y.** From any 3.1.x version to 4.2.6 SP1 (one step, using the migrator)
- **4.0.x to 4.2.y.** From any 4.0.x version to 4.2.6 SP1 (one step, using the migrator)
- **4.1.x to 4.2.y.** From any 4.1.x version to 4.2.6 SP1 (one step, using the migrator)

For links to these procedures, see *Per-version instruction sets*.

Upgrades to prior versions

Using the migrator:

- **3.1.x to 4.0.y.** From 3.1.5 directly to 4.0.2 (one step)
  For more information, see *Upgrade Fusion 3.1.x to 4.0.y*.
- **4.0.x to 4.0.y.** From 4.0.0 or 4.0.1 to 4.0.2 (one step)
  For more information, see *Upgrade Fusion Server 4.0.x to 4.0.y*.

- **3.1.x to 4.1.y.** From any 3.1.x version to 4.1.3 (one step, using the migrator)
  For more information, see *Upgrade Fusion Server 3.1.x to 4.1.y*.
- **4.0.x to 4.1.y.** From 4.0.2 to 4.1.3 (one step, using the migrator)
  For more information, see *Upgrade Fusion Server 4.0.x to 4.1.y*.
- **4.1.x to 4.1.y.** From 4.1.0 to 4.1.3 (one step, using the migrator)
  For more information, see *Upgrade Fusion Server 4.1.x to 4.1.y*.

Example

For example, to upgrade from Fusion 3.0.1 to Fusion Server 4.2.5, you would perform the following upgrades (both of them using the migrator):

1. Upgrade from Fusion 3.0.1 to Fusion 3.1.5
2. Upgrade from Fusion 3.1.5 to Fusion Server 4.2.5

Per-version instruction sets

To upgrade to a later version of Fusion from an existing installation requires transferring over all configurations and data from your existing Fusion installation to the new version.

How to upgrade from Fusion 3.1.x to Fusion Server 4.2.y

Perform the steps in this article:

Upgrade from Fusion Server 3.1.x to 4.2.y - Run a migrator to upgrade from Fusion Server 3.1.x to 4.2.y.

How to upgrade from Fusion 4.0.x to Fusion Server 4.2.y

Perform the steps in this article:

Upgrade from Fusion Server 4.0.x to 4.2.y - Run a migrator to upgrade from Fusion Server 4.0.x to 4.2.y.

How to upgrade from Fusion 4.1.x to Fusion Server 4.2.y

Perform the steps in this article:

Upgrade from Fusion Server 4.1.x to 4.2.y - Run a migrator to upgrade from Fusion Server 4.1.x to 4.2.y.

How to upgrade from Fusion 4.2.x to Fusion Server 4.2.y

Perform the steps in this article:

Upgrade from Fusion Server 4.2.x to 4.2.y - Run a migrator to upgrade from Fusion Server 4.2.x to 4.2.y.

Upload a JDBC Driver to Fusion Server

How-to

> Compatible versions: 4.2, 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion Server*, *Fusion*, *Fusion Connectors*

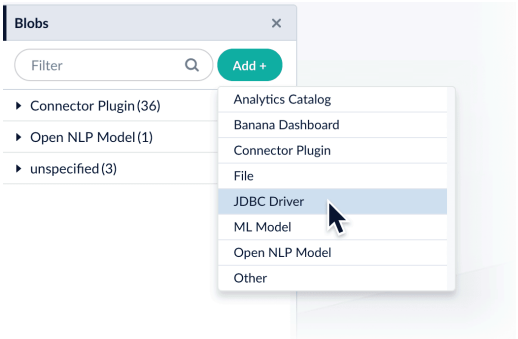
The JDBC connector fetches documents from a relational database via SQL queries. Under the hood, this connector implements the Solr DataImportHandler (DIH) plugin.

Fusion stores JDBC drivers in the blob store. You can upload a driver using the Fusion UI or the Blob Store API.

- The JDBC V1 connector is supported in Fusion releases 5.3 and earlier.
- The JDBC V2 connector is supported in Fusion releases 5.4 and later.

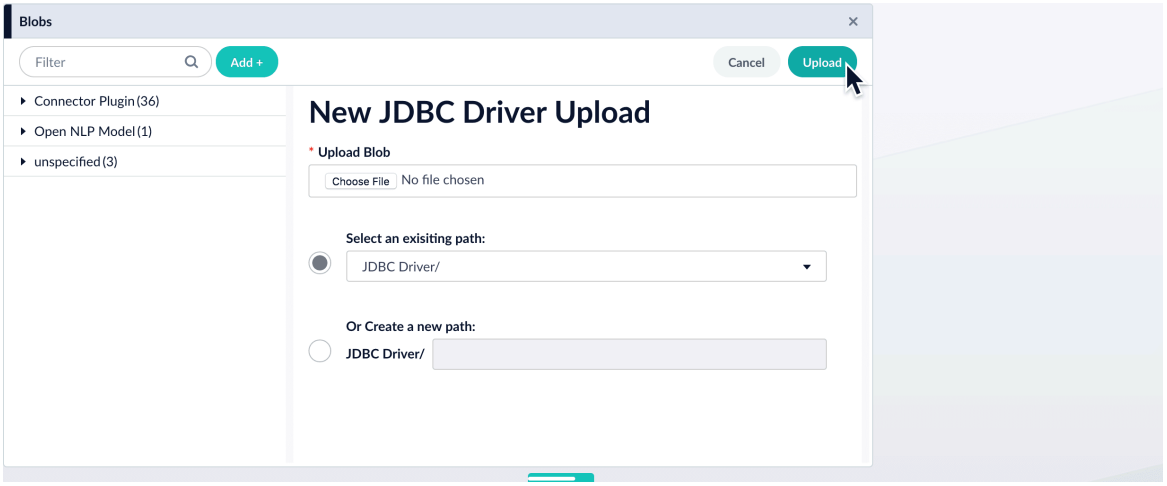
How to upload a JDBC driver using the Fusion UI

1. In the Fusion UI, navigate to **System** > **Blobs**.
2. Click **Add**.
3. Select **JDBC Driver**.

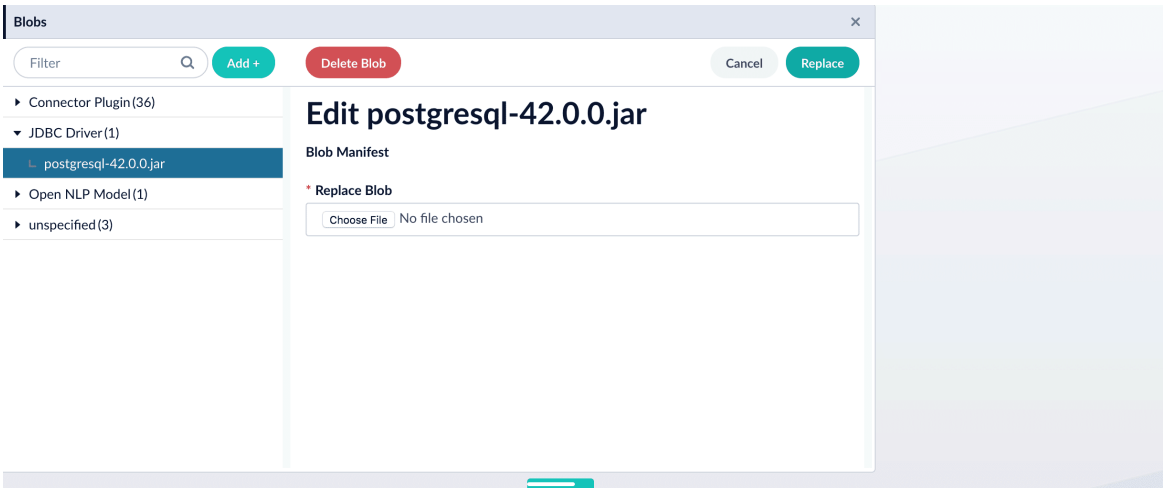


The "New 'JDBC Driver' Upload" panel appears.

4. Click **Choose File** and select the .jar file from your file system.



5. Click **Upload**.
   The new driver's blob manifest appears.



From this screen you can also delete or replace the driver.

How to install a JDBC driver using the API

1. Upload the JAR file to Fusion's blob store using the `/blobs/{id}` *endpoint*.
   Specify an arbitrary blob ID, and a **`resourceType`** value of **`plugin:connector`**, as in this example:

```
curl -u USERNAME:PASSWORD -H "content-type:application/java-
archive" -H "content-length:707261" -X PUT --data-binary
@postgresql-42.0.0.jar http://localhost:8764/api/blobs/mydriver?
resourceType=driver:jdbc
```

Success response:

```
{ "name" : "mydriver", "contentType" : "application/java-
archive", "size" : 707261, "modifiedTime" : "2017-06-
09T19:00:48.919Z", "version" : 0, "md5" :
"c67163ca764bfe632f28229c142131b5", "metadata" : { "subtype" :
"driver:jdbc", "drivers" : "org.postgresql.Driver",
"resourceType" : "driver:jdbc" } }
```

Fusion automatically publishes the event to the cluster, and the listeners perform the driver installation process on each node.

Tip

If the blob ID is identical to an existing one, the old driver will be uninstalled and the new driver will installed in its place. To get the list of existing blob IDs, run: `curl -u USERNAME:PASSWORD https://FUSION_HOST:FUSION_PORT/api/blobs`

2. To verify the uploaded driver, run:

```
curl -u USERNAME:PASSWORD
https://FUSION_HOST:FUSION_PORT/api/blobs/BLOB_ID/manifest
```

Where the `BLOB_ID` is the name specified during upload, such as "mydriver" above. A success response looks like this:

```
{ "name" : "mydriver", "contentType" : "application/java-
archive", "size" : 707261, "modifiedTime" : "2017-06-
09T19:05:17.897Z", "version" : 1569755095787110400, "md5" :
"c67163ca764bfe632f28229c142131b5", "metadata" : { "subtype" :
"driver:jdbc", "drivers" : "org.postgresql.Driver",
"resourceType" : "driver:jdbc" } }
```

How to manually upload a JDBC V1 driver for Fusion 4.x.x releases

For Fusion 4.x.x release levels, it is recommended that you upload the JDBC driver jar files manually. This prevents some issues with dynamically loading the driver causing errors, particularly with Oracle and Microsoft SQL drivers.

The following steps specify how to upload a JDBC driver JAR file to Fusion manually. The example uses the Microsoft SQL Server JDBC driver `mssql-jdbc-6.6.2-jre8.jar` file.

1. Delete the JDBC V1 driver uploaded using the Blob store upload window.
2. Save the JDBC V1 driver JAR file to the `apps/lib` folder. For example: `apps\libs\mssql-jdbc-6.2.2.jre8.jar`
3. Open the `apps/jetty/connectors-classic/webapps/connectors-extra-classpath.txt` file and add the following line: `apps/libs/mssql-jdbc-6.2.2.jre8.jar`
4. Restart connectors-classic.
5. Specify the JDBC driver class name in the **Manually uploaded JDBC Driver Name** field.
6. Enter **com.microsoft.sqlserver.jdbc.SQLServerDriver** in the **JDBC Driver Name** field.

Upload a JDBC Driver to Managed Fusion

How-to

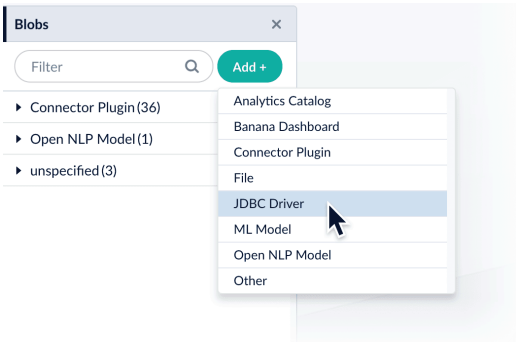
> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

The JDBC connector fetches documents from a relational database via SQL queries. Under the hood, this connector implements the Solr DataImportHandler (DIH) plugin.

Managed Fusion stores JDBC drivers in the blob store. You can upload a driver using the Managed Fusion UI or the Blob Store API. Learn more about the *JDBC V2 connector*.

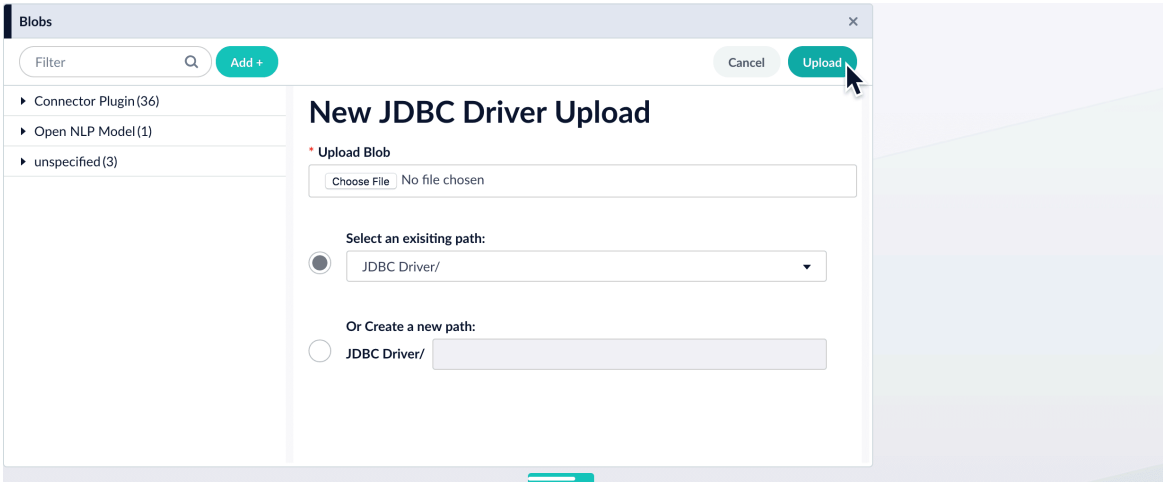
How to upload a JDBC driver using the Fusion UI

1. In the Managed Fusion UI, navigate to **System** > **Blobs**.
2. Click **Add**.
3. Select **JDBC Driver**.

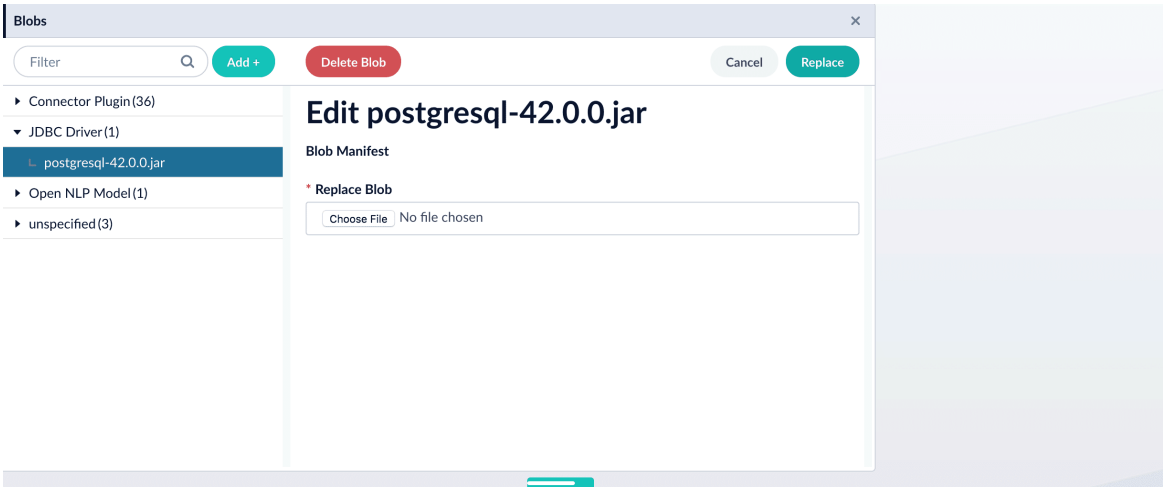


The "New 'JDBC Driver' Upload" panel appears.

4. Click **Choose File** and select the .jar file from your file system.



5. Click **Upload**.
   The new driver's blob manifest appears.



From this screen you can also delete or replace the driver.

How to install a JDBC driver using the API

1. Upload the JAR file to Managed Fusion's blob store using the `/blobs/{id}` *endpoint*. Specify an arbitrary blob ID, and a **`resourceType`** value of **`plugin:connector`**, as in this example:

```
curl -u USERNAME:PASSWORD -H "content-type:application/java-archive" -H "content-length:707261" -X PUT --data-binary @postgresql-42.0.0.jar https://EXAMPLE_COMPANY.b.lucidworks.cloud/api/blobs/mydriver?resourceType=driver:jdbc
```

Success response:

```
{ "name" : "mydriver", "contentType" : "application/java-archive", "size" : 707261, "modifiedTime" : "2017-06-09T19:00:48.919Z", "version" : 0, "md5" : "c67163ca764bfe632f28229c142131b5", "metadata" : { "subtype" : "driver:jdbc", "drivers" : "org.postgresql.Driver", "resourceType" : "driver:jdbc" } }
```

Managed Fusion automatically publishes the event to the cluster, and the listeners perform the driver installation process on each node.

Tip

If the blob ID is identical to an existing one, the old driver will be uninstalled and the new driver will installed in its place. To get the list of existing blob IDs, run: `curl -u USERNAME:PASSWORD https://FUSION_HOST:FUSION_PORT/api/blobs`

2. To verify the uploaded driver, run:

```
curl -u USERNAME:PASSWORD
https://EXAMPLE_COMPANY.b.lucidworks.cloud/api/blobs/BLOB_ID/man
ifest
```

Where the `BLOB_ID` is the name specified during upload, such as "mydriver" above. A success response looks like this:

```
{ "name" : "mydriver", "contentType" : "application/java-
archive", "size" : 707261, "modifiedTime" : "2017-06-
09T19:05:17.897Z", "version" : 1569755095787110400, "md5" :
"c67163ca764bfe632f28229c142131b5", "metadata" : { "subtype" :
"driver:jdbc", "drivers" : "org.postgresql.Driver",
"resourceType" : "driver:jdbc" } }
```

Use a Configured Platform

How-to

> Compatible versions: 4.2, 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *App Studio*

The ConfiguredPlatformProvider is a Java class that lets you gain access to an instance of a platform configured via the 'conf' files specified in an Appkit project. This is useful when writing custom code which you would like to have access to a platform already defined in an Appkit application, for example indexing or storing documents in a platform.

To use this, simply inject the ConfiguredPlatformProvider into the class in which you want to gain access to the platform's programmatic API:

```
@Inject private ConfiguredPlatformProvider myPlatformProvider; You can then access a configured platform using this method public Platform get(String name)
```

For example, if you have a platform configured in `src/main/resources/conf/platforms/gsa/gsa.conf`, in your code you would have:

```
Platform myPlatform = myPlatformProvider.get("platforms.gsa"); myPlatform.search(X) //search the platform myPlatform.store(X) //post a document to the platform
```

Most platforms support these general methods that you might want to use within the custom code that interfaces with the platform:

```
public twigkit.model.Response search(twigkit.model.Query query) throws twigkit.platform.PlatformException; public void store(twigkit.model.Content... contents) throws twigkit.platform.PlatformException; public twigkit.model.Content load(java.lang.String s) throws twigkit.platform.PlatformException; public void delete(twigkit.model.Query query) throws twigkit.platform.PlatformException;
```

A `PlatformOperationNotSupportedException` will be thrown if the method is not supported by the underlying platform. See the Javadoc for the `twigkit.model.Platform` interface for more information.

Deploying your service

Appkit uses Guice to inject dependencies into Java classes. The easiest way to let Guice know about your new Java class (service) so that it can inject the ConfiguredPlatformProvider is to define your class as a managed Appkit service. This is done by adding a configuration file with the fully qualified name of the Java class under:

```
/META-INF/ /services/ twigkit.service.AppkitService <-- this is a
file
```

You can enter a fully qualified Java class per line for each managed service you want to deploy.

An alternative is to use the AppkitApplication class to get an instance of the ConfiguredPlatformProvider:

```
ConfiguredPlatformProvider platformProvider =
TwigKitApplication.getInstance().getInjector().getInstance(ConfiguredP
latformProvider.class);
```

This creates code that is more tightly coupled, so is generally less desirable, but still completely functional.

Use a Private Repository for Docker Images

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion*

This article teaches you how to use a private repository to manage your Docker images. When you install or upgrade Fusion, you can specify the private repository instead of the public repository. To find all Docker images required for your private repository, see *List All Docker Images and Versions*. If you're deploying on-premises, you'll need to *set up on-premises private Docker registries*.

1. *Download* the `values.sh` script.
2. Run the script: `./values.sh`.

Note

If you receive an error stating `bash: ./values.sh: Permission denied`:

1. Make the script executable: `chmod +x values.sh`.
2. Run the script again: `./values.sh`.

The script creates a new file, `<provider>_<cluster>_<namespace>_fusion_repository.yaml`. The contents of the file resembles the following:

```
--- admin-ui: initContainer: image: repository:
altostrat.com/repo image: repository: altostrat.com/repo api-
gateway: keytoolUtils: image: repository: "altostrat.com/repo"
initContainer: image: repository: altostrat.com/repo image:
repository: altostrat.com/repo argo: images: ...
```

3. Open the `<provider>_<cluster>_<namespace>_upgrade_fusion.sh` script for editing.
4. Update the `MY_VALUES` values to refer to the `<provider>_<cluster>_<namespace>_fusion_repository.yaml` file:

```
MY_VALUES="" MY_VALUES="$MY_VALUES --values
<provider>_<cluster>_<namespace>_fusion__repository.yaml"
MY_VALUES="$MY_VALUES --values
<provider>_<cluster>_<namespace>_fusion_values.yaml"
MY_VALUES="$MY_VALUES --values
<provider>_<cluster>_<namespace>_fusion_resources.yaml"
MY_VALUES="$MY_VALUES --values
<provider>_<cluster>_<namespace>_fusion_affinity.yaml"
MY_VALUES="$MY_VALUES --values
<provider>_<cluster>_<namespace>_fusion_replicas.yaml"
```

5. Run the `<provider>_<cluster>_<namespace>_upgrade_fusion.sh` script.

Use a Remote Connector with Pulsar Proxy

How-to

> Compatible versions: 5.5
>
> Works with: *Fusion Connectors*, *Fusion*

To use a remote (V2) connector, configure both Fusion and the connector so on-premise customer content is transported to a Fusion instance running in the cloud. Remote connectors are controlled from Fusion.

Warning

As of Fusion 5.6.0, the **connector-plugin-standalone.jar** file referenced in this article is no longer published. This file is required for the steps in this article.

Note

Pulsar ports are not kept open on Managed Fusion instances.

Required infrastructure and software versions

- Google Cloud Platform
- Java 11
- Fusion 5.3 and later

Install and configuration process overview

An overview of the process to install and use a remote connector is:

- Install Fusion
- Install the remote connector in the network so it can access the content source
- Configure and run the remote connector
- Access the Fusion Datasource tab and add the remote connector as a datasource
- Configure the connector
- Perform all datasource tasks as if the connector was installed directly into Fusion

Security configuration

The security configuration is used to expose IP addresses and ports in Google Cloud Platform (GCP) that are not typically exposed in the production standard setup. In addition, the customer server requires direct outbound TCP ports 80/443 and 6650/6651.

Configure Fusion

1. During the Fusion installation, add or modify the last four lines in this example of the `_fusion_values.yaml` file to enable the **pulsar-proxy** information:

```
pulsar: broker: annotations: prometheus.io/scrape: "true"
prometheus.io/port: "8080" bookkeeper: annotations:
prometheus.io/scrape: "true" prometheus.io/port: "8000"
components: proxy: true image: repository: "lucidworks"
```

2. Run the `_upgrade_fusion` script:

```
./gke_lw-sales-us-west1_poc-remote-f5-poc_upgrade_fusion.sh
```

Note

For more information, see the fusion-cloud-native repository.

3. Execute the `kubectl get svc | grep pulsar-proxy` command to determine the load balancer IP and the required ports. The following example indicates the Fusion load balancer for the pulsar-proxy has an IP address of **34.105.102.171** and uses ports **80** and **6650**. The IP address and ports you generate may vary.

```
$ kubectl get svc | grep pulsar-proxy NAME TYPE CLUSTER-IP
EXTERNAL-IP PORT(S) AGE <namespace>-pulsar-pulsar-proxy
LoadBalancer 10.75.1.81 34.105.102.171
80:32182/TCP,6650:31884/TCP 6m7s
```

Configure the remote (V2) connector

1. Access the Lucidworks Plugin page.
2. Select and download the `connector-plugin-standalone.jar` file applicable to the release.
3. Create a `config.yaml` file and enter the following basic values:
   - **pulsar.service-url** - The value is typically 6650, but is obtained running the `kubectl get svc | grep pulsar-proxy` command.
   - **pulsar.admin-url** - The value is typically 80, but is obtained running the `kubectl get svc | grep pulsar-proxy` command.

     Note

     See *Description of Pulsar proxy properties for other values*.

     The following is an example of the `config.yaml` file:

```
# # The connector process will send messages through
Pulsar. # tenant-name is the namespace of the Fusion
cluster # pulsar: service-url:
pulsar://35.230.38.171:6650 admin-url:
https://35.230.38.171:80 tenant-name: ppt
authenticationEnabled: true # tlsEnabled: false #
tlsTrustCertsFilePath: ca.crt # # The user name/password
to Fusion and the URL where the # proxy: user: test
password: test url: https://fusion.servername.here:6764/
# # The name/location of the connector zip file #
plugin: path: fs.zip type: suffix: remote
```

4. Execute the following command:

```
java -Xms256m -Xmx2048m -jar connector-plugin-standalone.jar
config.yaml
```

The command output is:

```
For help use: 'java -jar connector-plugin-standalone.jar --help'
. ____ _ __ _ _ /\\ / ___'_ __ _ _(_)_ __ __ _ \ \ \ \ ( ( )\___
| '_ | '_| | '_ \/ _` | \ \ \ \ \\/ ___)| |_)| | | | | || (_| |
) ) ) ) ' |____| .__|_| |_|_| |_\__, | / / / /
=========|_|===|___/=/_/_/_/ :: Spring Boot :: [additional
connectivity output]
```

Description of Pulsar proxy properties

Property	Description
`pulsar.service-url`	Pulsar Service URL. For example, when Transport Layer Security (TLS) is: • Disabled, the URL is pulsar://35.247.112.3:6650 • Enabled, the URL is pulsar+ssl://35.230.38.171:6651
`pulsar.admin-url`	Pulsar Admin URL. For example, when Transport Layer Security (TLS) is: • Disabled, the URL is http://35.247.112.3:8080 • Enabled, the URL is https://35.247.112.3:443
`tenant-name`	Pulsar Tenant Name (kube namespace)
`authenticationEnabled`	Pulsar authentication enabled flag
`tlsEnabled`	TLS enabled flag
`tlsTrustCertsFilePath`	Trust certs file path
`proxy.user`	Fusion proxy user
`proxy.password`	Fusion proxy password
`proxy.url`	Fusion proxy url
`plugin.path`	Path of plugin zip file
`plugin.type.suffix`	Plugin type suffix. For example, the: • `lucidworks.testplugin` ID with suffix `remote` is `lucidworks.testplugin.remote` • Name **Test Connector** is **Test Connector (remote)**

The following is an example of the YAML file for the JDBC V2 connector:

```
pulsar: service-url: pulsar://35.230.38.171:6650 admin-url:
http://35.230.38.171:80 tenant-name: poc-f5-instance
authenticationEnabled: true # tlsEnabled: false #
tlsTrustCertsFilePath: ca.crt proxy: user: test password: test url:
http://35.197.110.199:6764/ plugin: path: lucidworks.connector.jdbc-
1.0.0.zip type: suffix: remote
```

Remote connector with Transport Security Layer (TLS) enabled

To use remote connector with TLS enabled, deploy Fusion with TLS and pulsar proxy enabled and then obtain the certificates to connect to pulsar proxy.

Obtain certificates to connect to pulsar proxy

1. Execute the following command to Generate the certificate file from Kubernetes secrets used by the pulsar proxy component:

```
kubectl get secret <namespace>-pulsar-pulsar-proxy-0-tls -o yaml
| grep ca.crt | cut -d':' -f2
LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSUZ2akNDQTZhZ0F3SUJBZ0lV
RjR
2cG5paFIxYk5JV3FQWDdJb09pOXlyR3dBd0RRWUpLb1pJaHZjTkFRU4KQlFBd1p
URUxN
QWtHQTFVRUJoTUNWVk14RXpBUkJnTlZCQWdUQ2tOaGJHbG1iM0p1YVdFeEZqQVVC
Z05WQk==
```

2. Copy the value generated and create a **`ca.cart`** file that includes the value and the following content:

```
-----BEGIN CERTIFICATE-----
LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSUZ2akNDQTZhZ0F3SUJBZ0lV
RjR
2cG5paFIxYk5JV3FQWDdJb09pOXlyR3dBd0RRWUpLb1pJaHZjTkFRU4KQlFBd1p
URUxN
QWtHQTFVRUJoTUNWVk14RXpBUkJnTlZCQWdUQ2tOaGJHbG1iM0p1YVdFeEZqQVVC
Z05WQk== -----END CERTIFICATE-----
```

3. Execute the following YAML configuration to run the remote connector:

```
pulsar: service-url: pulsar+ssl://35.230.38.171:6651 admin-url:
http://35.230.38.171:443 tenant-name: poc-f5-instance
authenticationEnabled: true tlsEnabled: true
tlsTrustCertsFilePath: ca.crt proxy: user: test password: test
url: http://35.197.110.199:6764/ plugin: path:
lucidworks.connector.jdbc-1.0.0.zip type: suffix: remote
```

Use Advanced Linguistics with Babel Street

How-to

> Compatible versions: 4.2, 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion Server*, *Fusion*

The Fusion Advanced Linguistics Package embeds Babel Street's (formerly Basistech) Rosette natural language processing tools for multilingual text analysis. To improve search recall, Rosette Base Linguistics (RBL) handles the unique linguistic phenomena of more than 30 Asian and European languages. Rosette Entity Extractor (REX) identifies named entities such as people, locations, and organizations, allowing you to quickly refine your search, remove noise, and increase search relevance.

Using Named Entities (REX)

REX extracts named entities in multiple languages, including English, Chinese (traditional and simplified), and German. In English, it extracts multiple entity types and subtypes. This includes the following entity types (along with their associated subtypes):

- `PERSON`
- `LOCATION`
- `ORGANIZATION`
- `PRODUCT`
- `TITLE`
- `NATIONALITY`
- `RELIGION`

In this tutorial, we will extract named entities from English news articles.

Create Application

To begin, create a new application called "entities".

Configuration

EDIT SOLR CONFIGURATION

We will begin by adding the Basis library elements to the `solrconfig.xml` file. We will also add a new update processor to perform the entity extraction.

1. Navigate to **System > Solr Config** to edit the `solrconfig.xml` file.
2. Fusion 5.8 and earlier: In the `<lib/>` directive section, add the lines below. Fusion 5.9 and later already contain these lines.

```
<lib dir="/opt/basistech/rex-je/lib" regex=".*\.jar" /> <lib
dir="/opt/basistech/solr/lib/" regex=".*\.jar" />
```

> **Important**
>
> For Fusion 4.x users, the `dir` paths are the local REX installation path.

3. In the `<updateRequestProcessorChain/>` section, add the following lines after the existing processor chains:

```
<updateRequestProcessorChain name="rex"> <processor
class="com.basistech.rosette.solr.EntityExtractorUpdateProcessor
Factory"> <str name="rootDirectory">/opt/basistech/rex-je</str>
<str name="fields">text_eng</str> </processor> <processor
class="solr.LogUpdateProcessorFactory"/> <processor
class="solr.RunUpdateProcessorFactory"/>
</updateRequestProcessorChain>
```

Note the reference to a field called `text_eng`. We will create this field through the Fusion UI in the next step.

4. Save your changes to `solrconfig.xml`.

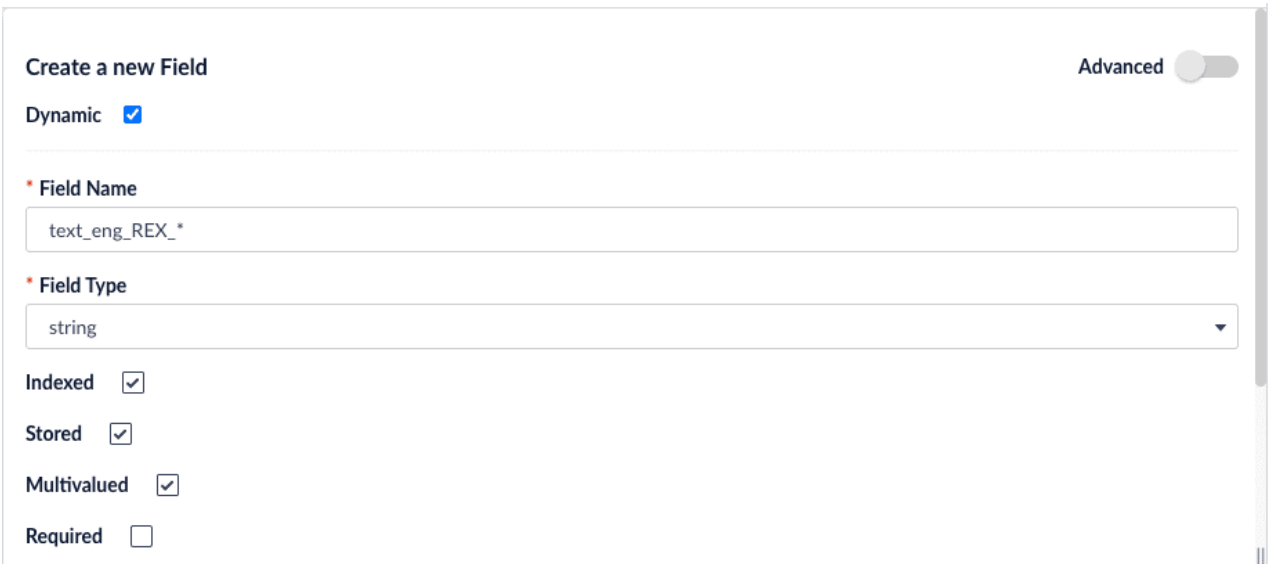
DEFINE FIELDS

The data file we will use, *eng_docs.csv*, contains two fields:

- `title`. An article headline
- `article_text`. The text content of the article

We will index these two static fields and also define a set of dynamic fields to hold the extracted entities. To create new fields, navigate to **Collections > Fields** and click **Add a Field**. Create the following fields:

Field name	Field type	Other options
`title`	`string`	Use default options.
`text_eng`	`text_en`	Use default options.
`text_eng_REX_*`	`string`	Create this field as a dynamic field by clicking the **Dynamic** checkbox. Click the **Multivalued** checkbox. Leave other options as defaults.



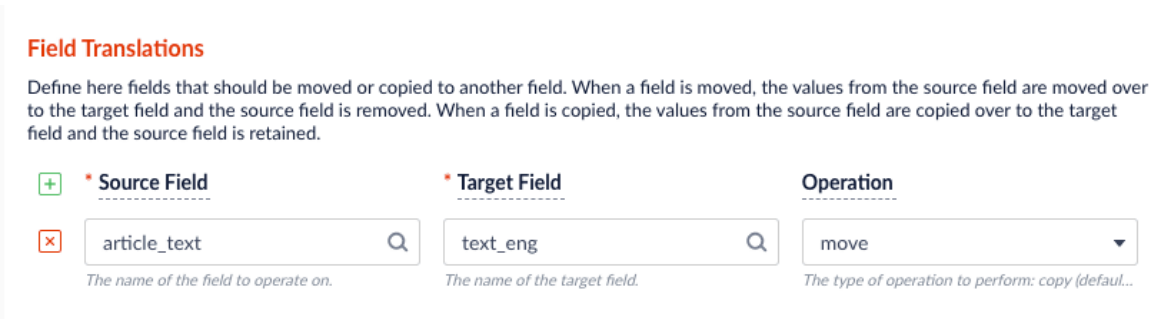
Note

Be sure to save each field after creating it.

Indexing Data

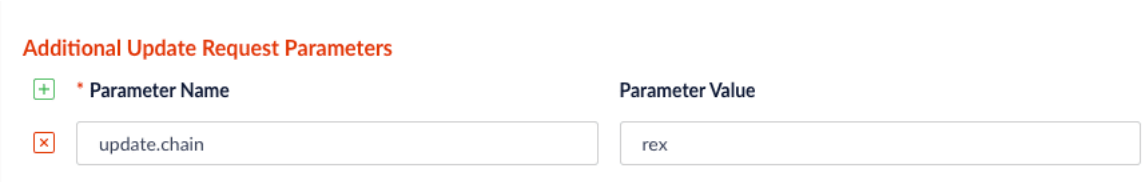
CREATE INDEXING PIPELINE

1. Navigate to **Indexing > Indexing Pipelines**.
2. Click **Add** and create a new pipeline called `test-entities`.
3. Select the **Field Mapping** stage.
4. In the **Field Translations** section, add a new row with source `article_text` and target `text_eng`. Set the **Operation** to `move`.



The CSV file we will upload contains a field title, but since this matches the title field that we created earlier, there is no need to map it. Leave the **Unmapped Fields** section with its default option, `keep`.

5. Select the **Solr Indexer** stage.
6. In the **Additional Update Request Parameters** section, add a new row with parameter name `update.chain` and value `rex`.

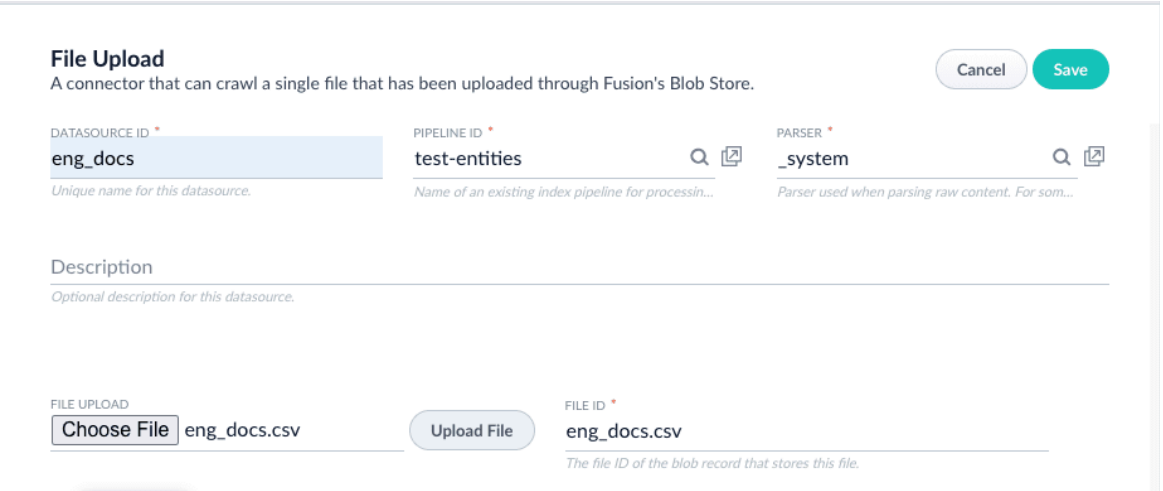


7. Save the new pipeline.

CREATE DATASOURCE

In this step, we will upload and index our documents from the data file.

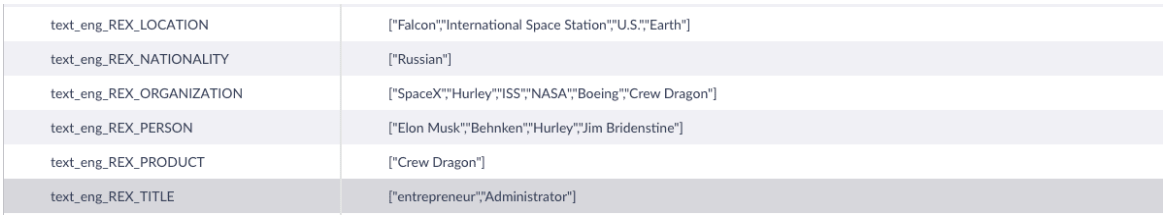
1. Navigate to **Indexing > Datasources**.
2. Click **Add** and select **File Upload V2** from the dropdown menu.
3. Enter `eng_docs` for the **Datasource ID**. Alternatively, use a name you prefer.
4. Select `test-entities` for the **Pipeline ID**.
5. In the **File Upload** field, choose the sample file `eng_docs.csv` and click **Upload File**. The File ID field will be automatically populated. Leave all other values as their defaults.



6. Save the new datasource. The form will refresh, adding a set of buttons at the top.
7. Click **Run** and then **Start**. When the job is finished, you will see "Success" in the popup form.

Querying Data

1. Navigate to **Querying > Query Workbench**. The default query is `*:*`, which should bring up three documents.
2. For the document with title "SpaceX Successfully Launches its First Crewed Spaceflight", select **Show fields**. You will see a number of entities listed under the `text_eng_REX_*` fieldnames.



3. Search on these multivalued fields. For example, set your query to `text_eng_REX_LOCATION:"New York"` to return the article that contains a mention of New York.

Customization (Advanced)

When setting up the Solr configuration, you specified the rootDirectory and fields options in your processor chain. REX provides a number of other configuration options you can set to control how entities are extracted. For example, if you are finding false positives, you can set parameters instructing REX to return only entities above a confidence threshold. The confidence threshold is a value between `0` and `1` and applies to entities extracted by the statistical model. We recommend starting with a low value, around `0.2`. In your `solrconfig.xml` file, add the options `calculateConfidence` and `confidenceThreshold` to your processor chain definition:

```
<updateRequestProcessorChain name="rex"> <processor
class="com.basistech.rosette.solr.EntityExtractorUpdateProcessorFactor
y"> <str name="rootDirectory">/opt/basistech/rex-je</str> <str
name="fields">text_eng</str> <str
name="calculateConfidence">True</str> <str
name="confidenceThreshold">0.2</str> </processor> <processor
class="solr.LogUpdateProcessorFactory"/> <processor
class="solr.RunUpdateProcessorFactory"/>
</updateRequestProcessorChain>
```

Save the changes, re-index your data, and perform the same query on `*:*`. Note that for the SpaceX article, "Falcon", is now correctly omitted from the list of LOCATIONs.

text_eng_REX_LOCATION	["International Space Station","U.S.","Earth"]
text_eng_REX_NATIONALITY	["Russian"]
text_eng_REX_ORGANIZATION	["SpaceX","ISS","NASA","Boeing"]
text_eng_REX_PERSON	["Elon Musk","Behnken","Hurley","Jim Bridenstine"]
text_eng_REX_TITLE	["Administrator"]

If there is a particular entity you want to make sure is extracted or rejected, or if you wish to create a custom entity type, REX also supports gazetteers and regular expressions.

GAZETTEERS

A gazetteer is a UTF-8 text file in which the first line is the entity type. It is followed by the names of entities you wish to extract, separated by newlines, in the language of your documents. Comments can be prefixed by the # symbol. Create a file spacecraft_gaz.txt with the following lines:

```
SPACECRAFT ISS International Space Station Vostok Soyuz Dragon Crew
Dragon Cargo Dragon Voyager Apollo
```

REGULAR EXPRESSIONS

REX uses the Tcl regex format. Create a file, **`zulu_time_regex.xml`**, file with the following lines:

```
<regexps> <regexp lang="eng" type="ZULU_TIME">(?i)\m(?:[01]?\d|2[0-4])
(?:[0-5]\d) (?:UTC|GMT)</regexp> </regexps>
```

This regular expression will extract as entity type `ZULU_TIME` all spans that consist of a 4-digit military time unit followed by the time zone designator UTC or GMT.

EXAMPLE

To instruct REX to use the gazetteer and regex file, edit your `solrconfig.xml` file. The `addGazetteers` option takes four parameters:

- language
- file
- accept (`True`) or reject (`False`)
- case-sensitive (`True` or `False`)

For example, with `<str name="addGazetteers">eng,/path/to/spacecraft_gaz.txt,True,True</str>`:

language	file	accept	case-sensitive
`eng`	`/path/to/spacecraft_gaz.txt`	`True`	`True`

The `addRegularExpressions` option takes two parameters:

- file
- accept (`True`) or reject (`False`)

For example, with `<str name="addRegularExpressions">/path/to/zulu_time_regex.xml,True</str>`:

file	accept
`/path/to/zulu_time_regex.xml`	`True`

The result in the `solrconfig.xml` file:

```
<updateRequestProcessorChain name="rex"> <processor
class="com.basistech.rosette.solr.EntityExtractorUpdateProcessorFactor
y"> <str name="rootDirectory">/opt/basistech/rex-je</str> <str
name="fields">text_eng</str> <str
name="calculateConfidence">True</str> <str
name="confidenceThreshold">0.2</str> <str
name="addGazetteers">eng,/path/to/spacecraft_gaz.txt,True,True</str>
<str
name="addRegularExpressions">/path/to/zulu_time_regex.xml,True</str>
</processor> <processor class="solr.LogUpdateProcessorFactory"/>
<processor class="solr.RunUpdateProcessorFactory"/>
</updateRequestProcessorChain>
```

Now, when you re-index your data and search `*:*`, the SpaceX document will have new entities listed in the `text_eng_REX_SPACECRAFT` and `text_eng_REX_ZULU_TIME` dynamic fields.

Note

Additional Fusion deployment configurations are needed to use the REX gazetteer and regex options.

Using Multilingual Search (RBL)

RBL provides a set of linguistic tools to prepare your data for analysis. Language-specific models provide base forms (lemmas) of words, parts-of-speech tagging, compound components, normalized tokens, stems, and roots.

In this tutorial, we will index and query headlines in English, Chinese, and German to demonstrate the linguistics capabilities of RBL: lemmatization, tokenization, and decompounding.

Create Application

To begin, create a new application called "multilingual".

Configuration

EDIT SOLR CONFIGURATION

We will begin by adding the Basis library elements to the `solrconfig.xml` file.

1. Navigate to **System > Solr Config** to edit the `solrconfig.xml` file.
2. In the `<lib/>` directive section, add the following lines:

```
<lib dir="/opt/basistech/solr/lib/" regex=".*\.jar" />
```

Important

For Fusion 4.x users, the `dir` path is the local REX installation path.

3. Save your changes to `solrconfig.xml`.

EDIT SCHEMA

Add a `fieldType` element for each language to be processed by the application. The `fieldType` element includes two analyzers: one for indexing documents and one for querying documents. Each analyzer contains a tokenizer and a token filter. The `language` attribute is set to the language code, equal to the ISO 639-3 code in most cases. The `rootDirectory` points to the RBL directory.

1. Navigate to **System > Solr Config** to edit the `managed-schema.xml` file.
2. In the **fieldType** section, add the following new field types: `basis_english`, `basis_chinese`, and `basis_german`.

```
<fieldtype class="solr.TextField" name="basis_english">
<analyzer type="index"> <tokenizer
class="com.basistech.rosette.lucene.BaseLinguisticsTokenizerFact
ory" language="eng" rootDirectory="/opt/basistech/rbl-je"/>
<filter
class="com.basistech.rosette.lucene.BaseLinguisticsTokenFilterFa
ctory" language="eng" rootDirectory="/opt/basistech/rbl-je"/>
</analyzer> <analyzer type="query"> <tokenizer
class="com.basistech.rosette.lucene.BaseLinguisticsTokenizerFact
ory" language="eng" query="true"
rootDirectory="/opt/basistech/rbl-je"/> <filter
class="com.basistech.rosette.lucene.BaseLinguisticsTokenFilterFa
ctory" language="eng" query="true"
rootDirectory="/opt/basistech/rbl-je"/> </analyzer> </fieldtype>
<fieldtype class="solr.TextField" name="basis_chinese">
<analyzer type="index"> <tokenizer
class="com.basistech.rosette.lucene.BaseLinguisticsTokenizerFact
ory" language="zho" rootDirectory="/opt/basistech/rbl-je"/>
<filter
class="com.basistech.rosette.lucene.BaseLinguisticsTokenFilterFa
ctory" language="zho" rootDirectory="/opt/basistech/rbl-je"/>
</analyzer> <analyzer type="query"> <tokenizer
class="com.basistech.rosette.lucene.BaseLinguisticsTokenizerFact
ory" language="zho" query="true"
rootDirectory="/opt/basistech/rbl-je"/> <filter
class="com.basistech.rosette.lucene.BaseLinguisticsTokenFilterFa
ctory" language="zho" query="true"
rootDirectory="/opt/basistech/rbl-je"/> </analyzer> </fieldtype>
<fieldtype class="solr.TextField" name="basis_german"> <analyzer
type="index"> <tokenizer
class="com.basistech.rosette.lucene.BaseLinguisticsTokenizerFact
ory" language="deu" rootDirectory="/opt/basistech/rbl-je"/>
<filter
```

```
class="com.basistech.rosette.lucene.BaseLinguisticsTokenFilterFa
ctory" language="deu" rootDirectory="/opt/basistech/rbl-je"/>
</analyzer> <analyzer type="query"> <tokenizer
class="com.basistech.rosette.lucene.BaseLinguisticsTokenizerFact
ory" language="deu" query="true"
rootDirectory="/opt/basistech/rbl-je"/> <filter
class="com.basistech.rosette.lucene.BaseLinguisticsTokenFilterFa
ctory" language="deu" query="true"
rootDirectory="/opt/basistech/rbl-je"/> </analyzer>
</fieldtype>
```

Note

You can incorporate any additional Solr filters you need, such as the Solr lowercase filter. However, filters should be added into the chain *after* the Base Linguistics token filter. If you modify the token stream too significantly before RBL, you degrade its ability to analyze the text.

3. Save your changes to `managed-schema.xml`.

DEFINE FIELDS

The data file we will use, *multilingual_headlines.csv*, contains fields for headlines in three languages: `eng_headline`, `zho_headline`, and `deu_headline`. The analysis chain requires a field definition with a `type` attribute that maps to the `fieldType` you defined in the schema.

To create new fields, navigate to **Collections > Fields** and click **Add a Field**. Create the following fields:

Field name	Field type	Other options
`text_eng`	`basis_english`	Use default options.
`text_zho`	`basis_chinese`	Use default options.
`text_deu`	`basis_german`	Use default options.

Note

Be sure to save each field after creating it.

Indexing Data

CREATE INDEXING PIPELINE

1. Navigate to **Indexing > Indexing Pipelines**.

2. Click **Add** and create a new pipeline called `test-multilingual`.
3. Select the **Field Mapping** stage.
4. In the **Field Translations** section, add three new rows:

Field name	Field name	Operation
`eng_headline`	`text_eng`	`move`
`zho_headline`	`text_zho`	`move`
`deu_headline`	`text_deu`	`move`

5. Save the new pipeline.

CREATE DATASOURCE

In this step, we will upload and index our documents from the data file.

1. Navigate to **Indexing > Datasources**.
2. Click **Add** and select **File Upload** from the dropdown menu.
3. Enter `multilingual_headlines` for the **Datasource ID**. Alternatively, use a name you prefer.
4. Select `test-multilingual` for the **Pipeline ID**.
5. In the **File Upload** field, choose the sample file `multilingual_headlines.csv` and click **Upload File**. The File ID field will be automatically populated. Leave all other values as their defaults.
6. Save the new datasource. The form will refresh, adding a set of buttons at the top.
7. Click **Run** and then **Start**. When the job is finished, you will see "Success" in the popup form.

Querying Data

1. Navigate to **Querying > Query Workbench**. The default query is `*:*`, which should bring up ten documents.
2. Follow the examples in the subsections below to see how Fusion's Advanced Linguistics capabilities can improve your search results.

LEMMATIZATION

A "lemma" is the canonical form of a word, or the version of a word that you find in the dictionary. For example, the lemma of "mice" is "mouse". The words "speaks", "speaking", "spoke", and "spoken" all share the same lemma: "speak".

With RBL, you can perform searches by lemma, thus increasing your search results. This example demonstrates this practice with the words "knife" and "knives" below.

1. For ease of viewing results, select the **Display Fields** dropdown and enter `text_eng` in the **Description** field.
2. Enter the query `text_eng:knife` in the search box.

Two documents are returned. One of the headlines, "Iowa City Man Arrested After Alleged Altercation with a Knife", is an exact match on the query term `knife`. With a standard Solr text field type, this would be the only result returned. However, the special type `basis_english` we configured allows the search engine to recognize "knives" as a form of "knife". Therefore, the article "The Best Ways to Sharpen Kitchen Knives at Home" is also returned.

RBL can significantly reduce your dependence on creating, maintaining, and using large synonym lists.

TOKENIZATION

Tokenization is the process of separating a piece of text into smaller units called "tokens". Tokens can be words, characters, or subwords, depending on how they are defined and analyzed. The RBL tokenizer first determines sentence boundaries, then segments each sentence into individual tokens. The most useful tokens are often words, though they may also be numbers or other characters.

In some languages like Chinese and Japanese, word tokens are not separated by whitespace, and words can consist of one, two, or more characters. For example, the tokens in 我喜歡貓 (I like cats) are 我 (I), 喜歡 (like), and 貓 (cats).

RBL uses statistical models to identify token boundaries, allowing for more accurate search results.

1. For ease of viewing results, select the **Display Fields** dropdown and enter `text_zho` in the **Description** field.
2. Enter the query `text_zho:美國` (United States) in the search box.

The document "美國男子染疫無呼吸道症狀但難以說話行走" ("U.S. Man has no Respiratory Symptoms but has Difficulty Talking and Walking") is returned, as 美國 (United States) is a token in the headline.

With a standard Solr text field type, this headline would be naively tokenized with one character per token. Therefore, a search for `美` (beautiful) would trigger a false positive match, even though it is not a word in this context. However, with the advanced analytics we have configured here, the query `text_zho:美` will correctly return zero results.

COMPOUNDS

RBL can decompose Chinese, Danish, Dutch, German, Hungarian, Japanese, Korean, Norwegian, and Swedish compounds, returning the lemmas of each of the components. The lemmas may differ from their surface form in the compound, such that the concatenation of the components is not the same as the original compound (or its lemma). Components are often connected by elements that are present only in the compound form. RBL allows Solr to index and query on these components, increasing recall of search results.

1. For ease of viewing results, select the **Display Fields** dropdown and enter `text_deu` in the **Description** field.
2. Enter the query `text_deu:Land` in the search box.

The result "Mitarbeiter sitzen in ihren Heimatländern fest" ("Employees are Stuck in Their Home Countries") returned.

This headline contains the compound "Heimatländern" (home countries). A search on `Land` (country) with a standard Solr text field type would not trigger a match. However, because RBL performs decompounding with lemmatization, searching on `Heimat` or `Land` will return a result.

Customization (Advanced)

When setting up the Solr configuration, you specified the language and rootDirectory options in your field type definition. This is sufficient for most use cases. However, RBL does provide more options to control the behavior of the tokenizer and analyzer. For example, the default tokenization does not consider URLs. As a result, `https://lucidworks.com` is tokenized as `https`, `lucidworks`, and `com`.

If you wish to recognize URLs, you can add the option `urls="true"` to the tokenizer in your field type definition:

```
<fieldtype class="solr.TextField" name="basis_english"> <analyzer
type="index"> <tokenizer
class="com.basistech.rosette.lucene.BaseLinguisticsTokenizerFactory"
language="eng" rootDirectory="/opt/basistech/rbl-je" urls="true"/>
<filter
class="com.basistech.rosette.lucene.BaseLinguisticsTokenFilterFactory"
language="eng" rootDirectory="/opt/basistech/rbl-je"/> </analyzer>
<analyzer type="query"> <tokenizer
class="com.basistech.rosette.lucene.BaseLinguisticsTokenizerFactory"
language="eng" query="true" rootDirectory="/opt/basistech/rbl-je"
urls="true"/> <filter
class="com.basistech.rosette.lucene.BaseLinguisticsTokenFilterFactory"
language="eng" query="true" rootDirectory="/opt/basistech/rbl-je"/>
</analyzer> </fieldtype>
```

This will instruct RBL to consider URLs as a single token, for example: **`https://lucidworks.com`**.

To see a list of all options, consult the full RBL documentation.

Use Business Rules in the Rules Editor

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion*

This article describes how to use the *Business Rules* screen in the *Rules Editor*.

LucidAcademy

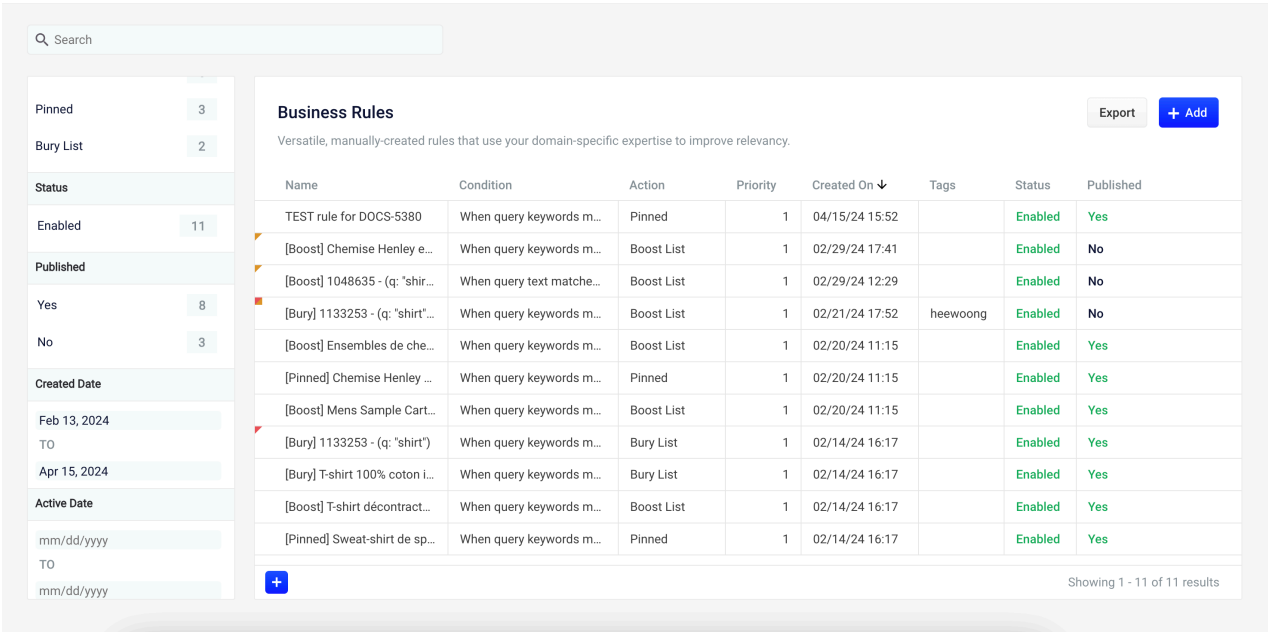
Lucidworks offers free training to help you get started.

The **Course** for **Query Rewriting with Business Rules and Search Rewrites** focuses on how using business rules and search rewrites gives you the opportunity to boost conversions and create better customer experiences by improving your relevancy:

Visit the LucidAcademy to see the full training catalog.

Facet rules

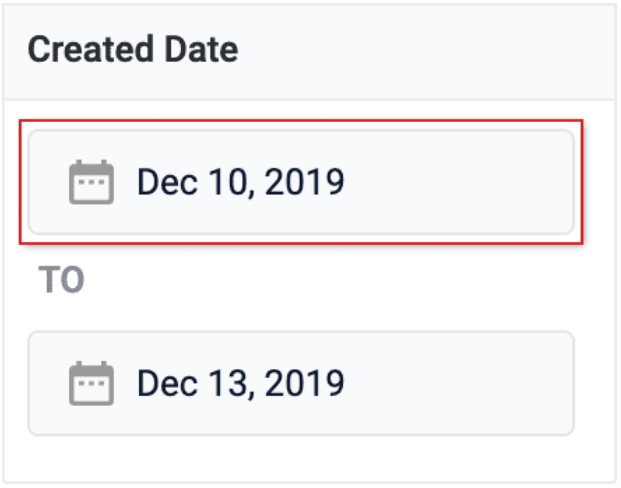
Facet menu



To facet the rules list by **Action**, **Status**, **Published**, or **Tags**, click one of the facet items within the facet categories.

To facet the rules list by **Created Date** or **Expiration Date**:

1. Click the date start value. A calendar will appear.



2. Click a date to select the start of your facet range.
3. Click the bottom date and select the end of your facet range.

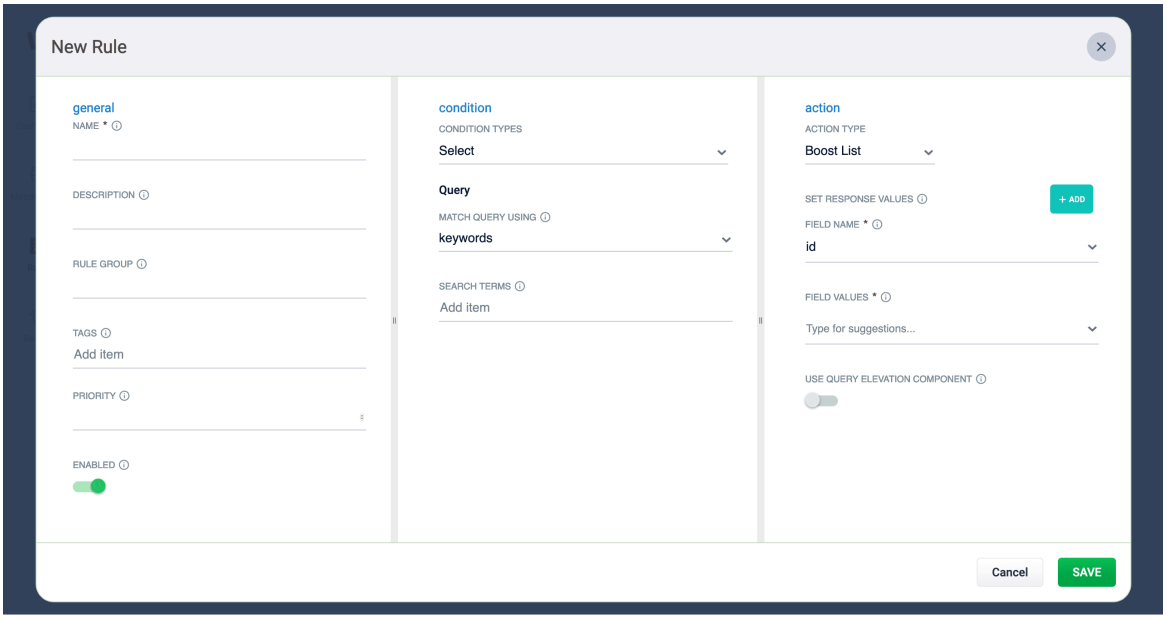
Search

Enter a term in the search bar to quickly facet your business rules.

Rules

Create a rule

1. Click the + Add button in the top-right corner of the card. Alternatively, click the + button in the bottom-left corner of the card. The **New Rule** dialog box will appear.



2. Complete the required fields, at a minimum.

General	Condition	Action
In the **General** column, only the **Name** field is required. Other fields are optional: • **Description** is an arbitrary string you can use to describe this rule. • **Rule Group** can be a user-defined group that you use to organize your rules. • **Tags** are another way to organize your rules. Tags appear as facets in the Business Rules interface, so you can filter the set of visible rules by tag. • **Priority** can be used to determine which rule should apply first if multiple rules are activated for the same request. Priority is an integer value, ranging from 1 to infinity, with the value 1 given lowest priority. If multiple rules have the same value, those rules are applied in a random order. • **Enabled** means that this rule is applied but not necessarily published. Disabling a rule helps ensure that it is not accidentally published.	Conditions are triggers that activate the rule when they match the current date and time, query, or field values. See *Rule conditions* for more information.	A rule can take different types of actions when the specified conditions are met. See *Action types* for more information. You can also create *custom actions*.

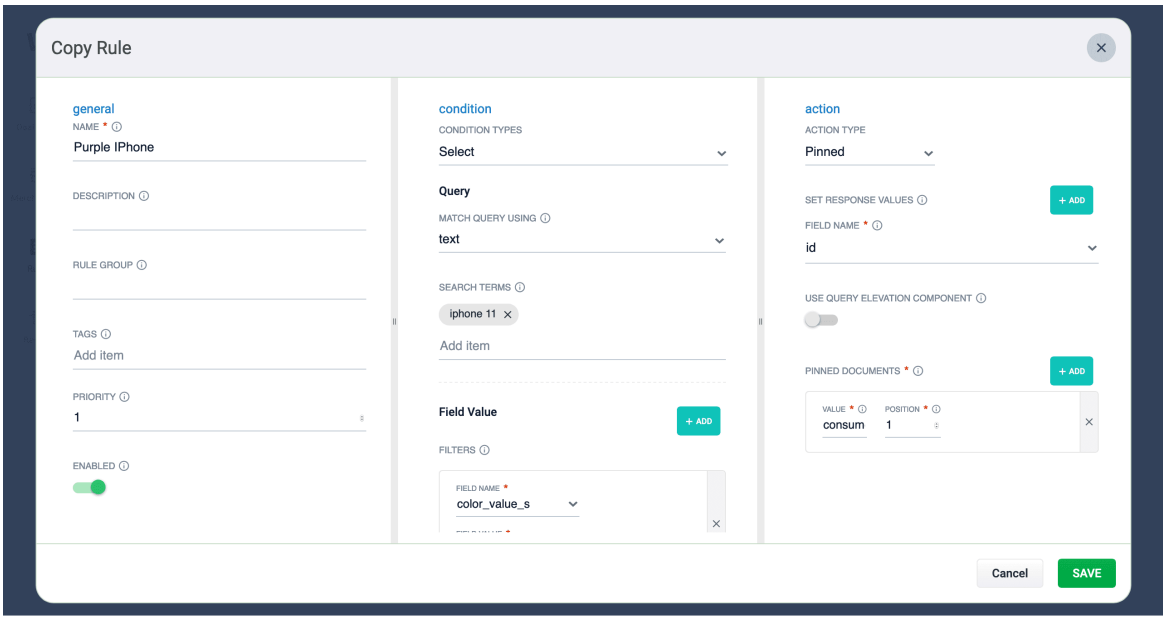
Tip

Hover over the icon for more information.

3. Once complete, click the SAVE button.

Copy a rule

1. Hover over a rule. A row of buttons will appear on the right.
2. Click the button. The **Copy Rule** dialog box will appear.



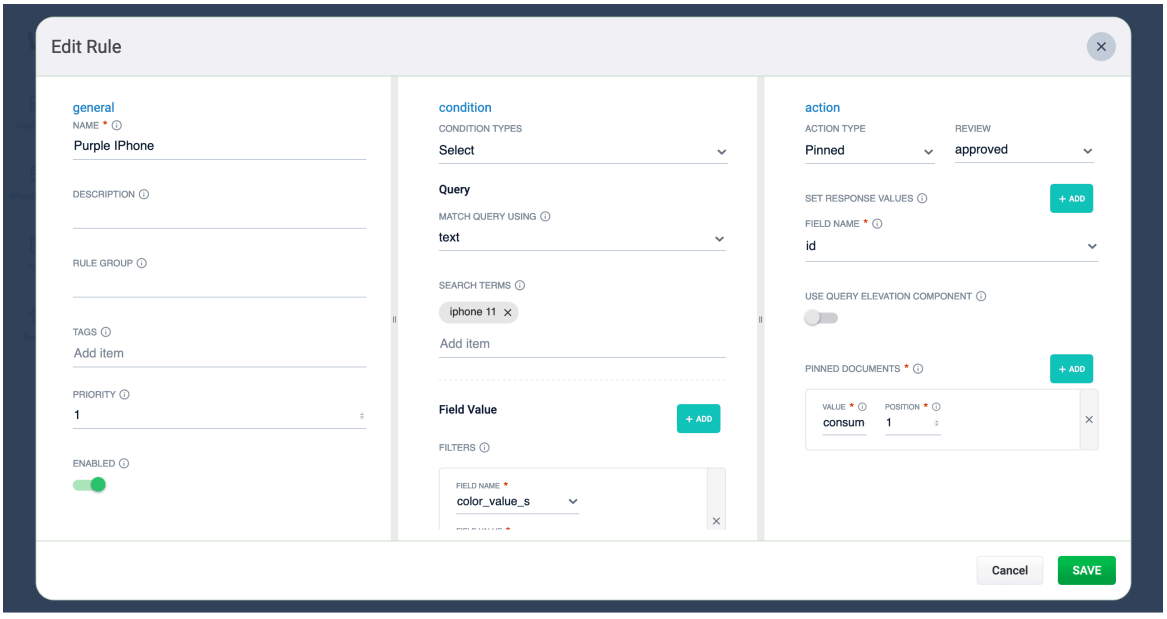
3. At a minimum, change the title of the rule.

4. Once complete, click the **SAVE** button.

Edit a rule

1. Hover over a rule. A row of buttons will appear on the right.

2. Click the button. The **Edit Rule** dialog box will appear.



3. Make your desired changes to the rule.

4. Once complete, click the SAVE button.

Approve/deny rules

1. Hover over a rule. A checkbox will appear on the left.
2. Click the checkbox to select the rule.
3. Repeat this process to select multiple rules. *(optional)*
4. Click the {approve-button} button to approve the selected rules, or click the {deny-button} button to deny the rules.

Enable/disable a rule

1. Hover over a rule. A checkbox will appear on the left.
2. Click the checkbox to select the rule.
3. Repeat this process to select multiple rules. *(optional)*
4. Click the Enable button to enable the selected rules, or click the Disable button to disable the rules.

Publish a rule

1. Hover over a rule. A checkbox will appear on the left.
2. Click the checkbox to select the rule.
3. Repeat this process to select multiple rules. *(optional)*
4. Click the Publish Selected button to publish the selected rules.

Tag a rule

Tags are a powerful tool for organizing and publishing rules according to a keyword, such as `black-friday-sale`.

1. Hover over a rule. A checkbox will appear on the left.
2. Click the checkbox to select the rule.
3. Repeat this process to select multiple rules. *(optional)*
4. Click the Add tags button. A dialog box will appear.
5. Enter a tag. To enter multiple tags, press **Tab** after writing each tag.
6. Click the + Add button to add the tag to the rule.

Note

See *General Configuration Tags* for information about how to use tagnames in the request URL to fire specific rules.

Delete rules

DELETE A SINGLE RULE

1. Hover over a rule. A row of buttons will appear on the right.
2. Click the button. A confirmation box will appear.
3. Click the DELETE button to confirm.

DELETE MULTIPLE RULES

1. Hover over a rule. A checkbox will appear on the left.
2. Click the checkbox to select the rule.
3. Repeat this process to select multiple rules. *(optional)*
4. Click the Delete button to delete the selected rules. A confirmation box will appear.
5. Click the DELETE button to confirm.

Sort rules

1. Hover over column titles. A ↓ icon will appear for some titles to indicate the rules list can be sorted by that column.
2. Click on the title column to sort the rules.

Note

The column that is currently sorting the rules list is indicated by an ↓ or ↑ icon.

Export rules

To export the rules list, simply click the Export button. This generates a `.csv` file containing your rules.

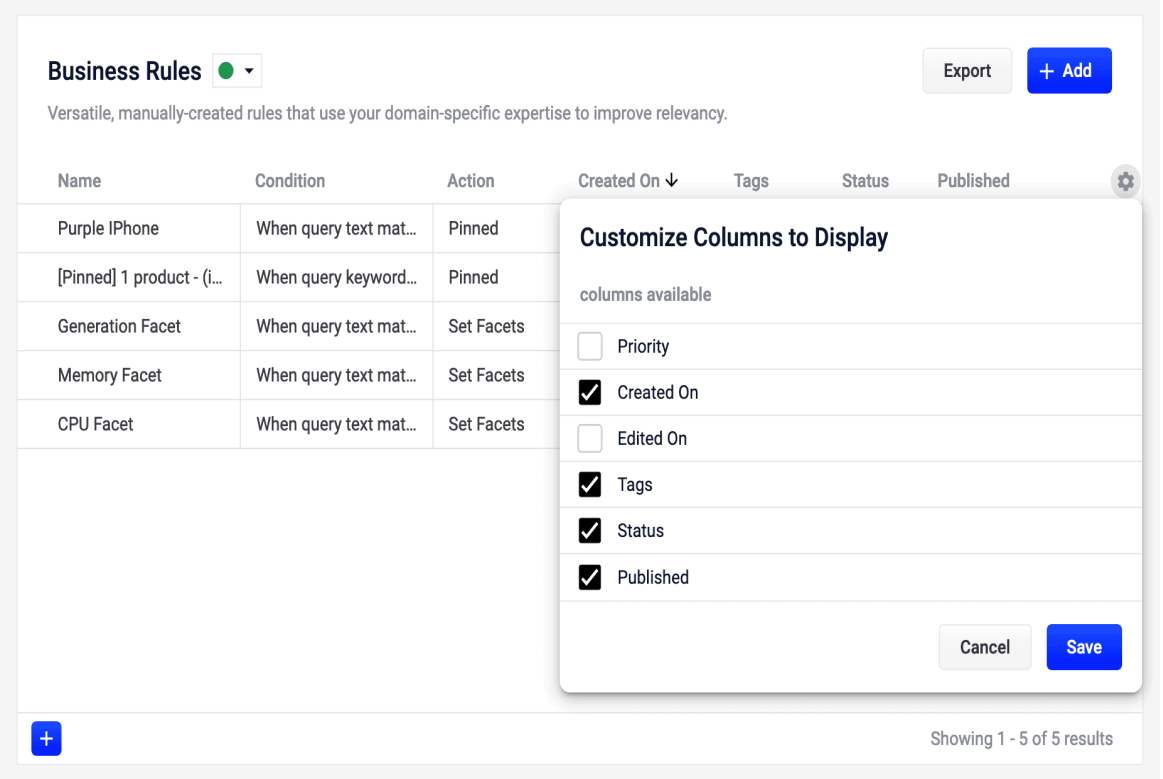
Columns

Customize columns

To customize the columns:

1. Hover over the row of column titles. A ⚙ button will appear on the right.

2. Click the ⚙ button. The **Customize Columns to Display** dialog box will appear.



3. Click the checkboxes to select or deselect the columns you want to display.

4. Click the **Save** button to save your changes.

Use Business Rules in the Rules Editor

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

This article describes how to use the *Business Rules* screen in the *Rules Editor*.

LucidAcademy

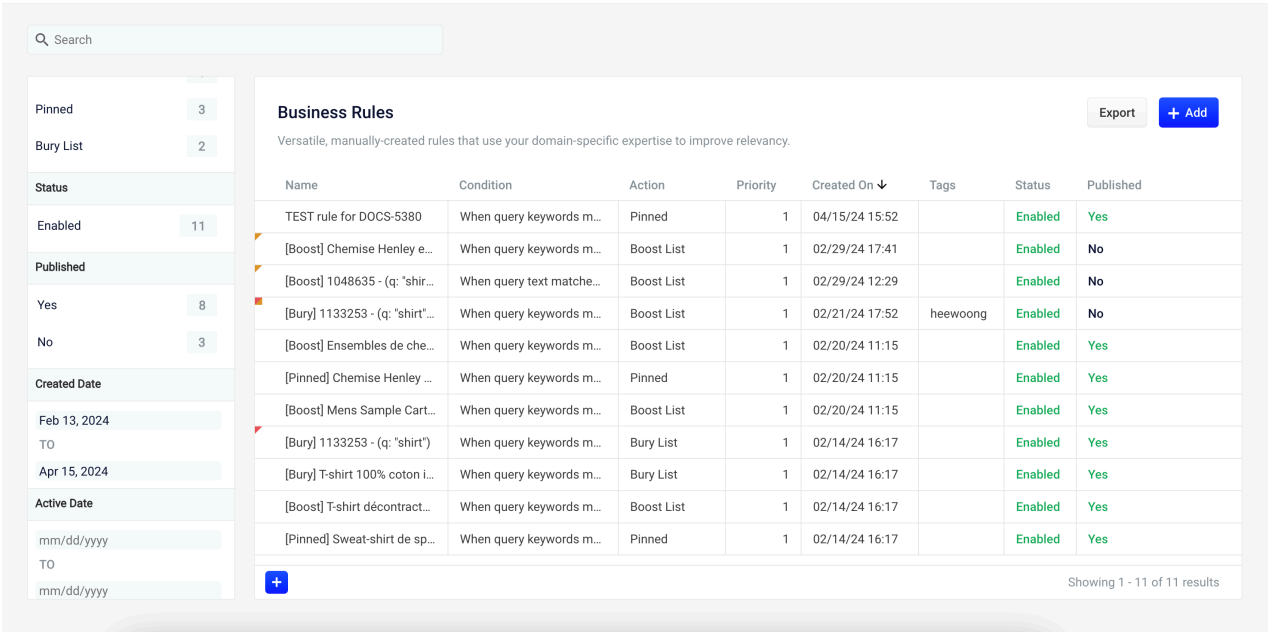
Lucidworks offers free training to help you get started.

The **Course** for **Query Rewriting with Business Rules and Search Rewrites** focuses on how using business rules and search rewrites gives you the opportunity to boost conversions and create better customer experiences by improving your relevancy:

Visit the LucidAcademy to see the full training catalog.

Facet rules

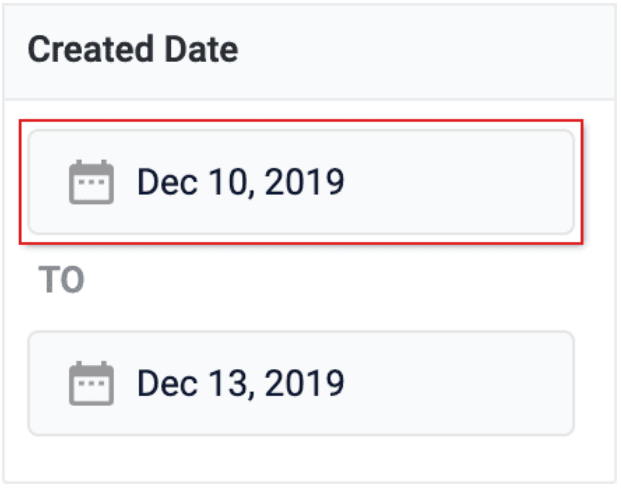
Facet menu



To facet the rules list by **Action**, **Status**, **Published**, or **Tags**, click one of the facet items within the facet categories.

To facet the rules list by **Created Date** or **Expiration Date**:

1. Click the date start value. A calendar will appear.



2. Click a date to select the start of your facet range.
3. Click the bottom date and select the end of your facet range.

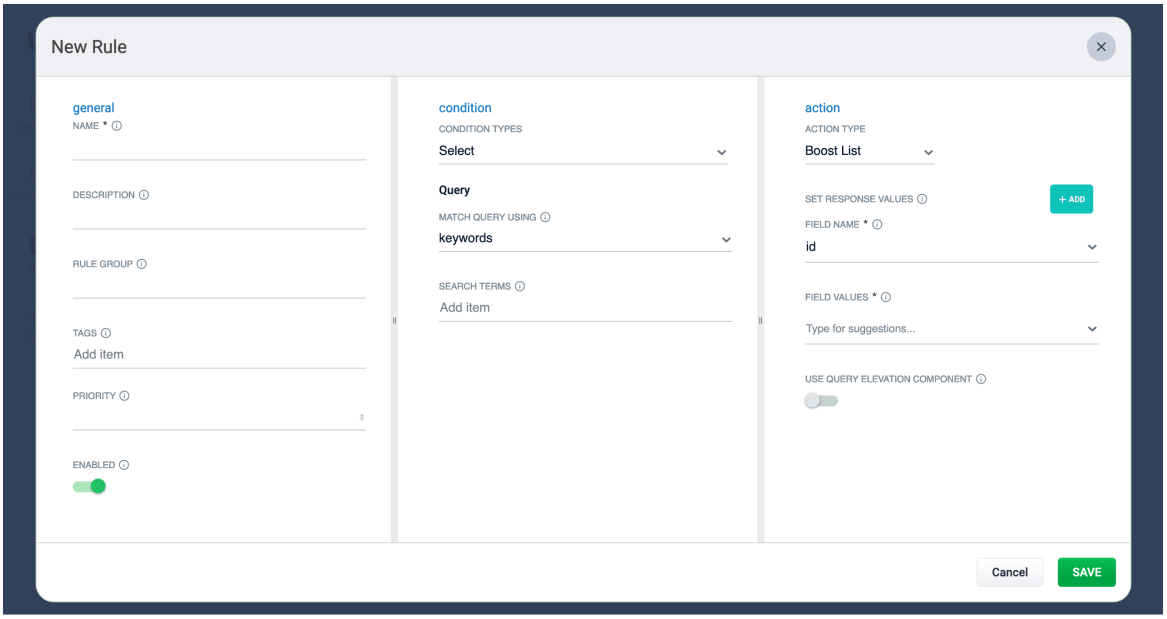
Search

Enter a term in the search bar to quickly facet your business rules.

Rules

Create a rule

1. Click the + Add button in the top-right corner of the card. Alternatively, click the + button in the bottom-left corner of the card. The **New Rule** dialog box appears.



2. Complete the required fields, at a minimum.

General	Condition	Action
In the **General** column, only the **Name** field is required. Other fields are optional: • **Description** is an arbitrary string you can use to describe this rule. • **Rule Group** can be a user-defined group that you use to organize your rules. • **Tags** are another way to organize your rules. Tags appear as facets in the Business Rules interface, so you can filter the set of visible rules by tag. • **Priority** can be used to determine which rule should apply first if multiple rules are activated for the same request. Priority is an integer value, ranging from 1 to infinity, with the value 1 given lowest priority. If multiple rules have the same value, those rules are applied in a random order. • **Enabled** means that this rule is applied but not necessarily published. Disabling a rule helps ensure that it is not accidentally published.	Conditions are triggers that activate the rule when they match the current date and time, query, or field values. See *Rule conditions* for more information.	A rule can take different types of actions when the specified conditions are met. See *Action types* for more information. You can also create *custom actions*.

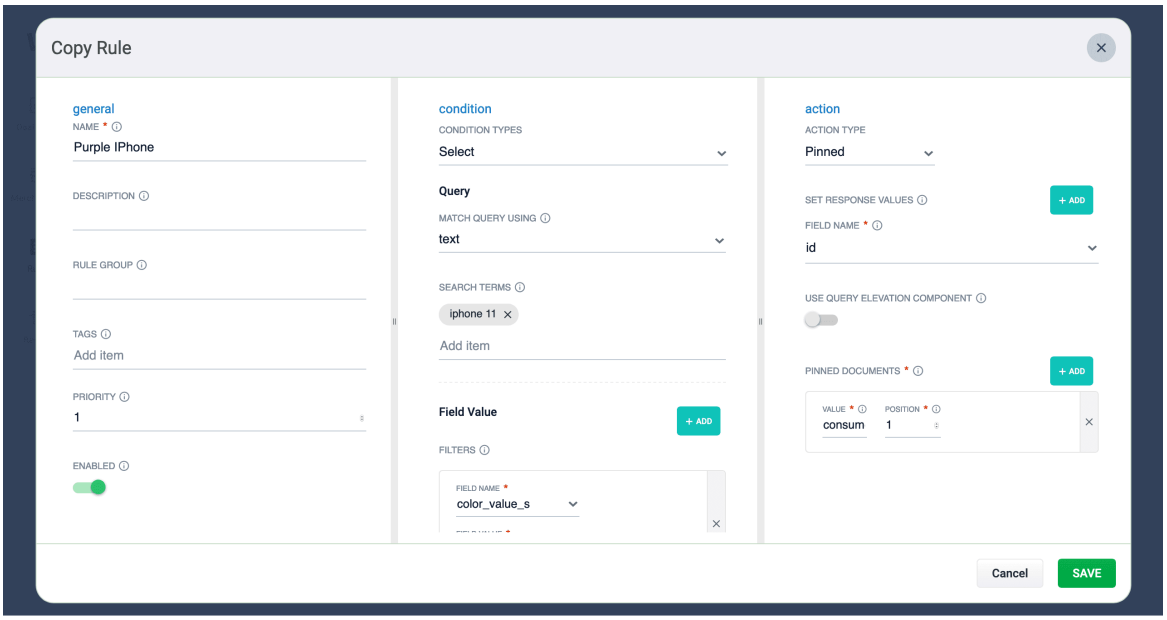
Tip

Hover over the icon for more information.

3. Once complete, click the SAVE button.

Copy a rule

1. Hover over a rule. A row of buttons appears on the right.
2. Click the button. The **Copy Rule** dialog box appears.



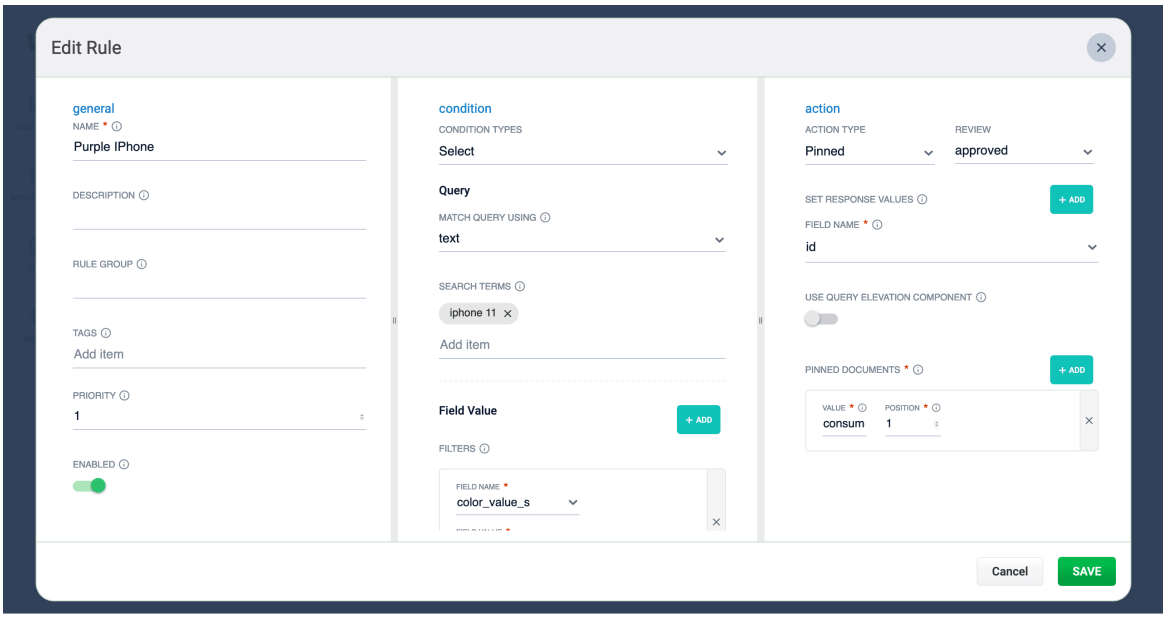
3. At a minimum, change the title of the rule.

4. Once complete, click the **SAVE** button.

Edit a rule

1. Hover over a rule. A row of buttons appears on the right.

2. Click the button. The **Edit Rule** dialog box appears.



3. Make your desired changes to the rule.

4. Once complete, click the SAVE button.

Approve/deny rules

1. Hover over a rule. A checkbox appears on the left.
2. Click the checkbox to select the rule.
3. Repeat this process to select multiple rules. *(optional)*
4. Click the {approve-button} button to approve the selected rules, or click the {deny-button} button to deny the rules.

Enable/disable a rule

1. Hover over a rule. A checkbox appears on the left.
2. Click the checkbox to select the rule.
3. Repeat this process to select multiple rules. *(optional)*
4. Click the Enable button to enable the selected rules, or click the Disable button to disable the rules.

Publish a rule

1. Hover over a rule. A checkbox appears on the left.
2. Click the checkbox to select the rule.
3. Repeat this process to select multiple rules. *(optional)*
4. Click the Publish Selected button to publish the selected rules.

Tag a rule

Tags are a powerful tool for organizing and publishing rules according to a keyword, such as `black-friday-sale`.

1. Hover over a rule. A checkbox appears on the left.
2. Click the checkbox to select the rule.
3. Repeat this process to select multiple rules. *(optional)*
4. Click the Add tags button. A dialog box appears.
5. Enter a tag. To enter multiple tags, press **Tab** after writing each tag.
6. Click the + Add button to add the tag to the rule.

Note

See *General Configuration Tags* for information about how to use tagnames in the request URL to fire specific rules.

Delete rules

DELETE A SINGLE RULE

1. Hover over a rule. A row of buttons appears on the right.
2. Click the button. A confirmation box appears.
3. Click the DELETE button to confirm.

DELETE MULTIPLE RULES

1. Hover over a rule. A checkbox appears on the left.
2. Click the checkbox to select the rule.
3. Repeat this process to select multiple rules. *(optional)*
4. Click the Delete button to delete the selected rules. A confirmation box appears.
5. Click the DELETE button to confirm.

Sort rules

1. Hover over column titles. A ↓ icon appears for some titles to indicate the rules list can be sorted by that column.
2. Click on the title column to sort the rules.

Note

The column that is currently sorting the rules list is indicated by an ↓ or ↑ icon.

Export rules

To export the rules list, simply click the Export button. This generates a `.csv` file containing your rules.

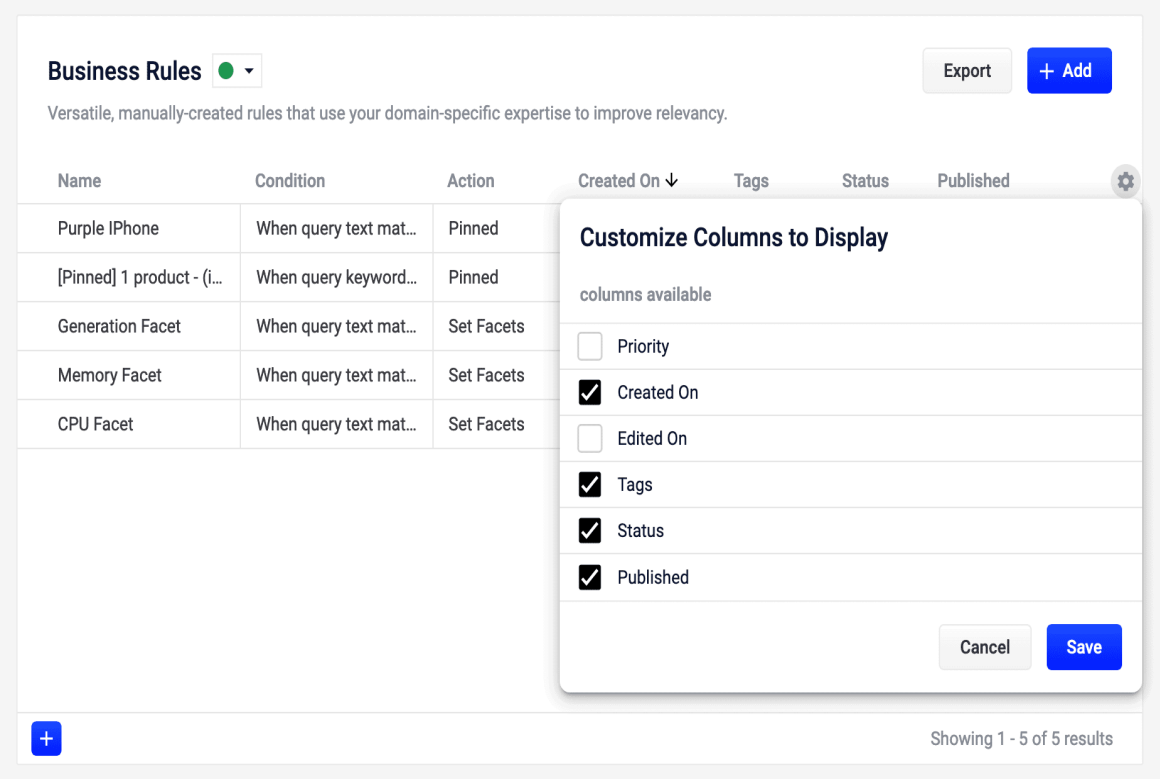
Columns

Customize columns

To customize the columns:

1. Hover over the row of column titles. A ⚙ button appears on the right.

2. Click the ⚙ button. The **Customize Columns to Display** dialog box appears.



3. Click the checkboxes to select or deselect the columns you want to display.

4. Click the **Save** button to save your changes.

Use Dashboards in Fusion

How-to

> Compatible versions: 4.2, 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion Server*, *Fusion*

This topic details the options available to use Fusion dashboards.

Dashboards main page

When you open the Fusion dashboards, your organization's dashboards display, as well as the EXPLORE DASHBOARDS section that provides options to create or display specific types of dashboards.

You can click the icons in the top right corner of the screen to perform the following functions:

Option	Notes
Goto saved default	Open the saved default dashboard.
New	Create a new dashboard. Options are: • Time-series dashboard • Non time-series dashboard
Load	Load a dashboard using one of the following options: • Load dashboard JSON layout from file Click **Local File > Choose File > select file > Open**. Saved dashboards are listed at the bottom of the dialog box. Click a dashboard to open it. Filter dashboards by entering an exact-match string in the Filter box. If there are more that 20 saved dashboards, use the pagination controls to find a dashboard. To view the next set of pages, click the forward arrows. To view the prior set of pages, click the back arrows. Or click a page number to view the dashboards on that page. • Enter the gist URL or number (the last part of the URL) **Enter value > click Get** . A link that opens the dashboard appears below a thin line. The link displays the dashboard name. Click the link to open the dashboard. • Type to filter. Enter dashboard type to filter and display if the dashboard matches.
Save	Save options include: • Locally ◦ Export to File ◦ Set as Browser Default. Click to set the current dashboard as the default. ◦ Clear Browser Default. Click to clear the current dashboard. The default dashboard for the browser becomes the system default dashboard `default.json` (the Fusion Logs dashboard). • Solr ◦ Make Public. If this checkbox is selected, the dashboard is shared with everyone. ◦ Service Logs. Enter a value and click **Save** to save the log.

Option	Notes
Configure Dashboard	Click to configure tabs for General, Rows, Control, and Solr dashboard settings.
Delete a dashboard saved in Solr	1. Click **Load**. 2. Identify the file to delete and click **Delete**. 3. Click **OK** to confirm the deletion.
Delete a dashboard saved in a file	Navigate to the file and delete it.
Delete a dashboard saved as a Github gist	Open the gist URL and click **Delete**.

For more information, see:

- *Fusion 4.x Built-in Dashboards*
- *Fusion 5.x Built-in Dashboards*

Interact with data

The table describes options to interact with data displayed in certain dashboards.

Option	Notes
Collections	Choose the collection from which to obtain data. If a dashboard is configured to let users choose among collections, select the collection from the **Collections** drop-down list at the top right of the dashboard.
Search	Enter the value to search. • To search for a specific value, enter a term or phrase and click **Search**. • To retrieve all data, enter *.* and click **Search**.
Clear search box	Click to clear the search box and return the most general query, as if you entered *.*.
Waiting for data	Indicates the query is waiting for data to be returned. If the indicator spins for an extended time while other panels display data, the panel is not configured correctly.
Select item	Click a table row. In a graphical-visualization panel, click the data item. **Note** Not all panel types let you select data in this way.
Export data	Click to enter the number of rows to export and select the destination file type. For example, CSV, XML, or JSON.
Hide row	Click to hide the row from the display.
Expand row	Click the row to display detailed information.

Interact with specific types of panels

The table describes options for specific types of panels.

Table and text panels

Option	Notes
Show or hide the field list	Click the right or left arrow, respectively, at the top of the panel.
Sort a table column	Click the up arrow at the top of the column to sort in ascending order. Click the down arrow at the top of the column to sort in descending order.
View next or previous page of hits	Click the right or left arrow, respectively, at the bottom of the page of hits.
  View first page	Click to view the first page of hits for that particular panel.
Show record details	Click the table row and then select the display format of Table, JSON, or Raw.
Hide details	Click the up arrow at the top right of the details.

Histogram and Range Facet panels

The table describes options for the Histogram and Range Facet panels.

Option	Notes
Zoom in on a subset of data	Click the area you want to enlarge, then drag across the range of data to zoom in.
  Zoom out	Click to zoom out.

Filtering panels

The table describes filter options for panels.

Option	Notes
Edit filter	Click to edit a panel filter.
Toggle filter	Click to toggle the filter to the opposite setting (on or off).
Remove filter	Click to remove the current filter. **Caution** To display correctly, specific filters must be set for certain types of panels. Fusion also creates filters based on your interactions with data. You can add a filter back, but there is no undo. Before you remove a filter, toggle off the filter and verify the results. If the results are accurate, it is safe to remove it.

Heat Map panels

The table describes the option for heat map panels.

Option	Notes
Transpose rows and columns	Click to reverse the display of rows and columns.

Filter by Fusion request ID

To only display information about log entries with a specific Fusion request ID, you can filter the contents of a logs dashboard by the Fusion request ID.

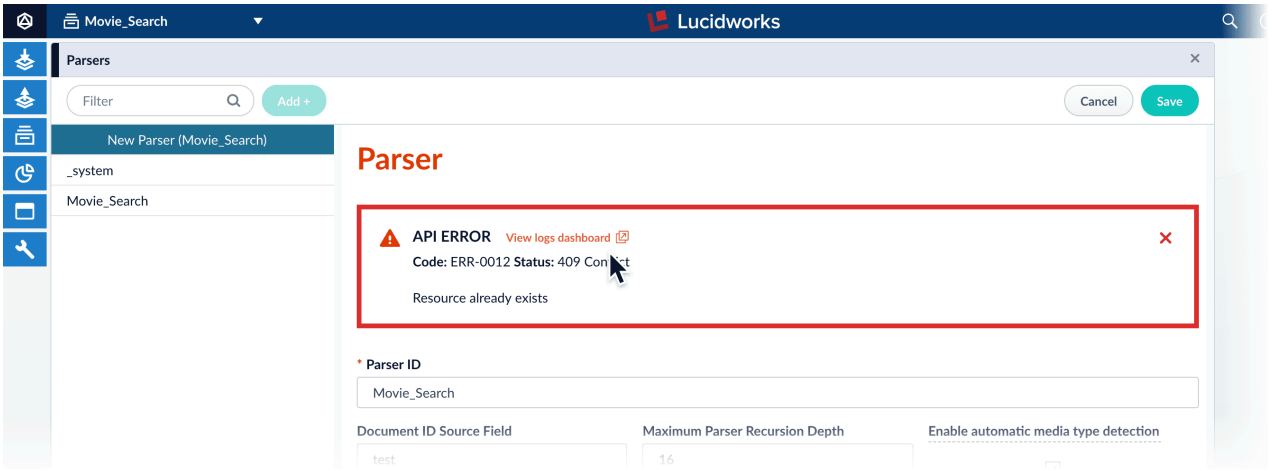
Filter the contents in the following ways:

- *Click a link in the error message or toast*
- *Filter in a logs dashboard - for releases prior to 5.4.0*
- *Append a query filter parameter to the dashboard URL*

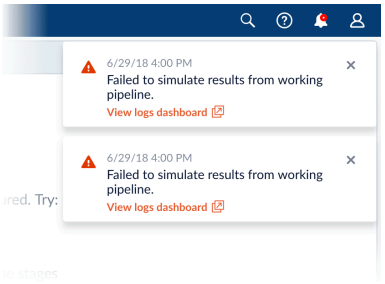
Click a link in the error message or toast

To view the dashboard for a Fusion UI API error or API error toast, click **View logs dashboard**. The system displays the dashboard using a filter for that request ID.

API error with link in Fusion UI:

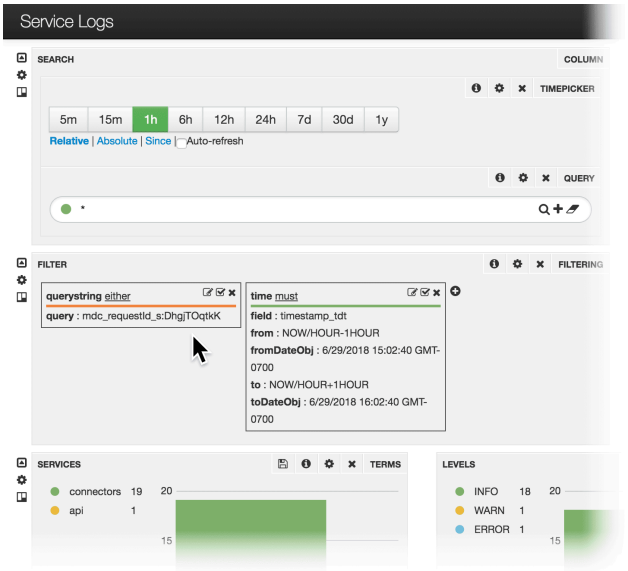


API error toast with link:



Filtered Service Logs dashboard:

Click **View logs dashboard** to open the Service Logs dashboard with filters by the Fusion request ID. For example, the filtering query for the `mdc_requestId_s` field displays the `fusion-request-id` of `BWQSjPPYgB`.



Filter in a logs dashboard

Important

This section only applies to Fusion versions prior to 5.4.0.

You can filter the dashboards for service logs, access logs, and combined logs by Fusion request ID.

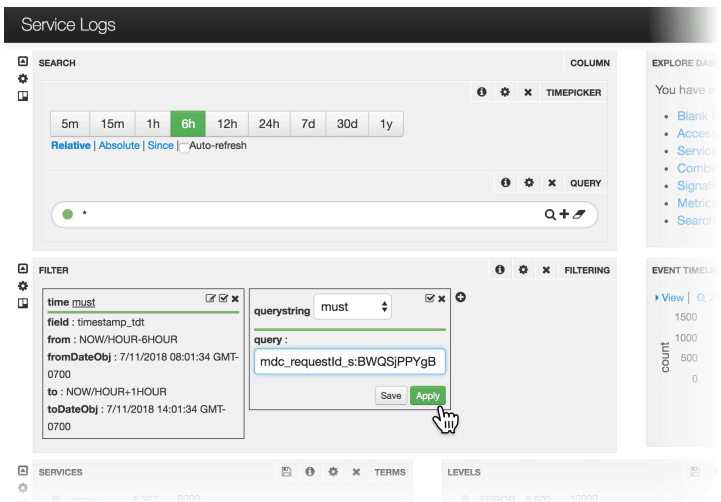
To filter by Fusion request ID, complete the following:

1. Obtain the Fusion request ID returned by a request. For example, notice the Fusion request ID `BWQSjPPYgB` returned by this API command:

```
curl -v -u USERNAME:PASSWORD
http://127.0.0.1:8764/api/apps/inventory * Trying 127.0.0.1... *
TCP_NODELAY set * Connected to 127.0.0.1 (127.0.0.1) port 8764
(#0) * Server auth using Basic with user 'admin' > GET
/api/apps/inventory HTTP/1.1 > Host: 127.0.0.1:8764 >
Authorization: Basic YWRtaW46UGFzc3dvcmQxMjM= > User-Agent:
curl/7.54.0 > Accept: */* > < HTTP/1.1 404 Not Found < Content-
Type: application/json < fusion-request-id: BWQSjPPYgB < Server:
Jetty(9.3.8.v20160314) < Set-Cookie: id=4962c806-f3b5-485f-8fb4-
c20e608b700c;Path=/;HttpOnly < Content-Length: 394 < Connection:
close < { "type" : "RESTError", "httpStatusCode" : 404, "error"
: "ERR-0003", "message" : "A web application error occurred",
"details" : "The Fusion App with id 'inventory' does not exist",
"cause" : null, "requestId" : "BWQSjPPYgB", "httpStatusMessage"
: "Not Found", "localizedMessage" : "The Fusion App with id
'inventory' does not exist", "suppressed" : [ ], "class" : null
* Closing connection 0 }
```

2. Open the logs dashboard.
3. Specify the Fusion request ID in the Filter panel, with a query for the `mdc_requestId_s` field that specifies the value of `fusion-request-id` for which you want to search.

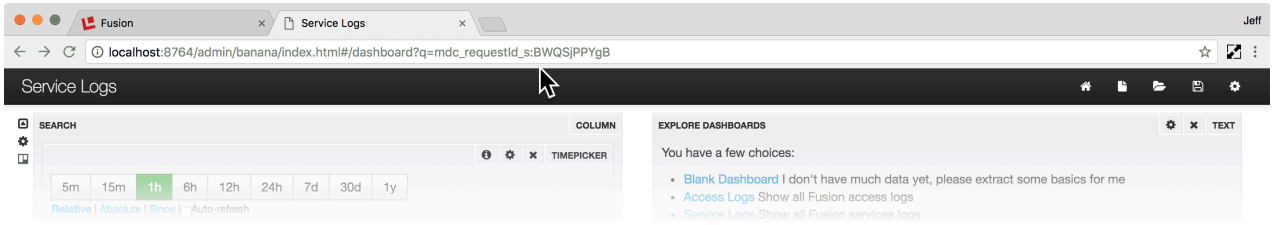
In this example, the `fusion-request-id` is `BWQSjPPYgB`:



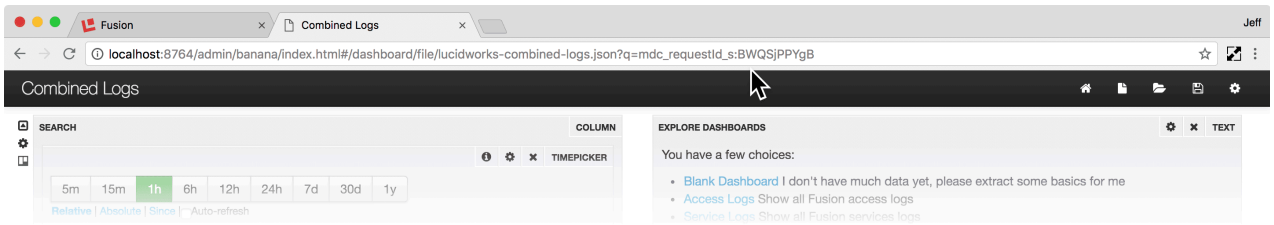
Append a query filter parameter to the dashboard URL

As an alternative to constructing a query filter in the dashboard, you can append a query filter parameter to the dashboard URL. In the query filter, use the field name `mdc_requestID_s`.

Example 1: Default dashboard (Service Logs)



Example 2: Combined Logs dashboard



Open the Service Logs dashboard from a job history

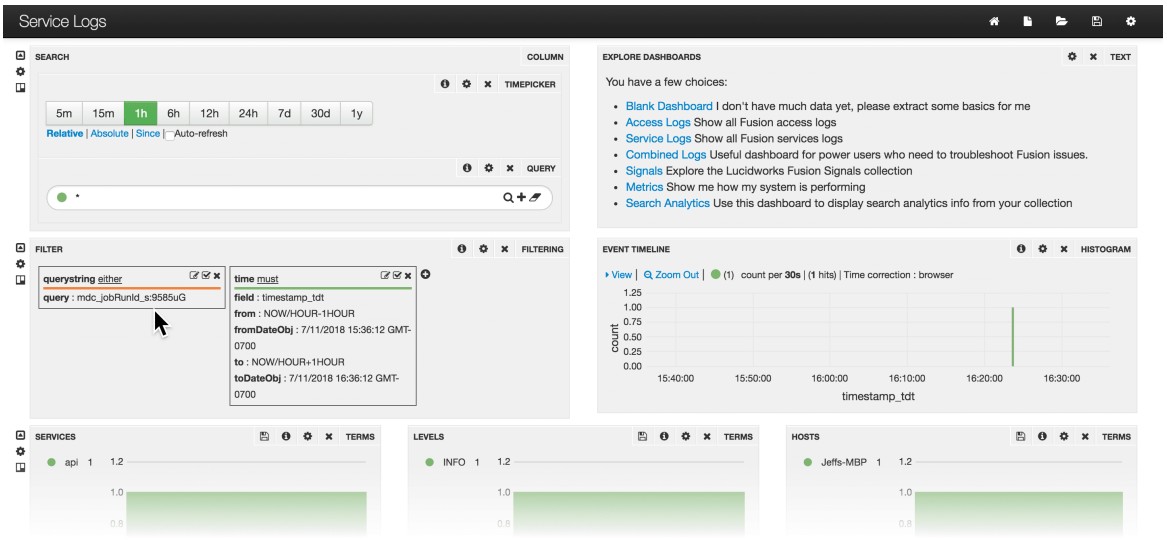
1. Open the job history.

   - For most jobs, click **Collections** > **Jobs**, select a job, and then click **Job History**.

   - For datasource jobs, click **Indexing** > **Datasources**, select a datasource, and then click **Job History**.

     In the job history, you will see a link named `runId`:

2. Click the link to view the Service Logs dashboard filtered by the `runId`:


Service Logs
SEARCH
COLUMN
TIMEPICKER
5m 15m 1h 6h 12h 24h 7d 30d 1y
Relative | Absolute | Since | Auto-refresh
QUERY
*
EXPLORE DASHBOARDS
TEXT
You have a few choices:
Blank Dashboard I don't have much data yet, please extract some basics for me
Access Logs Show all Fusion access logs
Service Logs Show all Fusion services logs
Combined Logs Useful dashboard for power users who need to troubleshoot Fusion issues.
Signals Explore the Lucidworks Fusion Signals collection
Metrics Show me how my system is performing
Search Analytics Use this dashboard to display search analytics info from your collection
FILTER
FILTERING
querystring either
query : mdc_jobRunId_s:9585uG
time must
field : timestamp_tdt
from : NOW/HOUR-1HOUR
fromDateObj : 7/11/2018 15:36:12 GMT-0700
to : NOW/HOUR+1HOUR
toDateObj : 7/11/2018 16:36:12 GMT-0700
EVENT TIMELINE
HISTOGRAM
View | Zoom Out | (1) count per 30s | (1 hits) | Time correction : browser
count
1.25
1.00
0.75
0.50
0.25
0.00
15:40:00
15:50:00
16:00:00
16:10:00
16:20:00
16:30:00
timestamp_tdt
SERVICES
TERMS
api 1
LEVELS
TERMS
INFO 1
HOSTS
TERMS
Jeffs-MBP 1

Use Dashboards in Managed Fusion

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

This topic explainds how to use Managed Fusion dashboards.

Dashboards main page

When you open the Managed Fusion dashboards, your organization's dashboards display, as well as the EXPLORE DASHBOARDS section that provides options to create or display specific types of dashboards.

You can click the icons in the top right corner of the screen to perform the following functions:

Option	Notes
Goto saved default	Open the saved default dashboard.
New	Create a new dashboard. Options are: • Time-series dashboard • Non time-series dashboard
Load	Load a dashboard using one of the following options: • Load dashboard JSON layout from file Click **Local File > Choose File > select file > Open**. Saved dashboards are listed at the bottom of the dialog box. Click a dashboard to open it. Filter dashboards by entering an exact-match string in the Filter box. If there are more that 20 saved dashboards, use the pagination controls to find a dashboard. To view the next set of pages, click the forward arrows. To view the prior set of pages, click the back arrows. Or click a page number to view the dashboards on that page. • Enter the gist URL or number (the last part of the URL) **Enter value > click Get** . A link that opens the dashboard appears below a thin line. The link displays the dashboard name. Click the link to open the dashboard. • Type to filter. Enter dashboard type to filter and display if the dashboard matches.

Option	Notes
Save	Save options include: • Locally ◦ Export to File ◦ Set as Browser Default. Click to set the current dashboard as the default. ◦ Clear Browser Default. Click to clear the current dashboard. The default dashboard for the browser becomes the system default dashboard `default.json` (the Managed Fusion Logs dashboard). • Solr ◦ Make Public. If this checkbox is selected, the dashboard is shared with everyone. ◦ Service Logs. Enter a value and click **Save** to save the log.
Configure Dashboard	Click to configure tabs for General, Rows, Control, and Solr dashboard settings.
Delete a dashboard saved in Solr	1. Click **Load**. 2. Identify the file to delete and click **Delete**. 3. Click **OK** to confirm the deletion.
Delete a dashboard saved in a file	Navigate to the file and delete it.
Delete a dashboard saved as a Github gist	Open the gist URL and click **Delete**.

For more information, see *Managed Fusion Built-in Dashboards*

Interact with data

The table describes options to interact with data displayed in certain dashboards.

Option	Notes
Collections	Choose the collection from which to obtain data. If a dashboard is configured to let users choose among collections, select the collection from the **Collections** drop-down list at the top right of the dashboard.
Search	Enter the value to search. • To search for a specific value, enter a term or phrase and click **Search**. • To retrieve all data, enter *.* and click **Search**.
Clear search box	Click to clear the search box and return the most general query, as if you entered *.*.
Waiting for data	Indicates the query is waiting for data to be returned. If the indicator spins for an extended time while other panels display data, the panel is not configured correctly.
Select item	Click a table row. In a graphical-visualization panel, click the data item. **Note** Not all panel types let you select data in this way.
Export data	Click to enter the number of rows to export and select the destination file type. For example, CSV, XML, or JSON.
Hide row	Click to hide the row from the display.
Expand row	Click the row to display detailed information.

Interact with specific types of panels

The table describes options for specific types of panels.

Table and text panels

Option	Notes
Show or hide the field list	Click the right or left arrow, respectively, at the top of the panel.
Sort a table column	Click the up arrow at the top of the column to sort in ascending order. Click the down arrow at the top of the column to sort in descending order.
View next or previous page of hits	Click the right or left arrow, respectively, at the bottom of the page of hits.
  View first page	Click to view the first page of hits for that particular panel.
Show record details	Click the table row and then select the display format of Table, JSON, or Raw.
Hide details	Click the up arrow at the top right of the details.

Histogram and Range Facet panels

The table describes options for the Histogram and Range Facet panels.

Option	Notes
Zoom in on a subset of data	Click the area you want to enlarge, then drag across the range of data to zoom in.
  Zoom out	Click to zoom out.

Filtering panels

The table describes filter options for panels.

Option	Notes
Edit filter	Click to edit a panel filter.
Toggle filter	Click to toggle the filter to the opposite setting (on or off).
Remove filter	Click to remove the current filter. **Caution** To display correctly, specific filters must be set for certain types of panels. Managed Fusion also creates filters based on your interactions with data. You can add a filter back, but there is no undo. Before you remove a filter, toggle off the filter and verify the results. If the results are accurate, it is safe to remove it.

Heat Map panels

The table describes the option for heat map panels.

Option	Notes
Transpose rows and columns	Click to reverse the display of rows and columns.

Use Experience Optimizer facets

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

Add a facet

You can create rules that add facets by interacting directly with the Facet List in Interactive Mode. You can add facets that are set unconditionally, or facets that are only set when your customers run a particular query. For example, you might want to set a music genre facet when a customer navigates to the music section of your catalog.

Adding a facet automatically creates a new rule if a default facet rule does not already exist. If a default facet rule does exist, adding a facet updates that rule with the new facet.

Enable the facet zone

Facets display in the zone on the left panel of the Experience Optimizer, once enabled. Follow the steps below to turn on the facets zone:

1. Click **Start Task**.

   The button displays.

2. In the left panel, click **+Add** and enter the desired parameters.
3. Click **Done**.

Edit the facet zone widget

You can also add facets that are not dependent on rules being triggered.

1. In the facet zone, click **+Add**.
2. In Facet Field, enter the desired facet parameter.

   Tip

   You can add multiple facet parameters by clicking **+ Add**.

3. Click **Save** to add the facet.

Add additional facets

1. Hold the pointer above or below an existing facet.
2. Click the **+** button and add the new facet.

Move a facet

1. Hold the pointer over a facet.
2. In the menu that displays, click-and-hold the drag bar.
3. Move the facet to the desired location.
4. Release the pointer.

Remove a facet

1. Hold the pointer over a facet.
2. In the menu that displays, click **Remove**.

Block a facet item

1. Hold the pointer over a facet.
2. In the menu that displays, click **Block**.

Bury a facet item

1. Hold the pointer over a facet.

2. In the menu that displays, click **Bury**.

Boost a facet item

1. Hold the pointer over a facet.
2. In the menu that displays, click **Boost**.

Reorder boosted facet items

1. Hold the pointer over the boosted facet item to move.
2. In the menu that displays, click-and-hold the drag bar.
3. Move the facet to the desired location.
4. Release the pointer.

Use Experience Optimizer Facets

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion*

Add a Facet

You can create rules that add facets by interacting directly with the Facet List in Interactive Mode. You can add facets that are set unconditionally, or facets that are only set when your customers run a particular query. For example, you might want to set a music genre facet when a customer navigates to the music section of your catalog.

Adding a facet automatically creates a new rule if a default facet rule does not already exist. If a default facet rule does exist, adding a facet updates that rule with the new facet.

Enable the Facet Zone

In Fusion 5.4 and later, facets appear in the zone on the left panel of the Experience Optimizer, once enabled. Follow the steps below to turn on the facets zone:

1. Click **Start Task**.

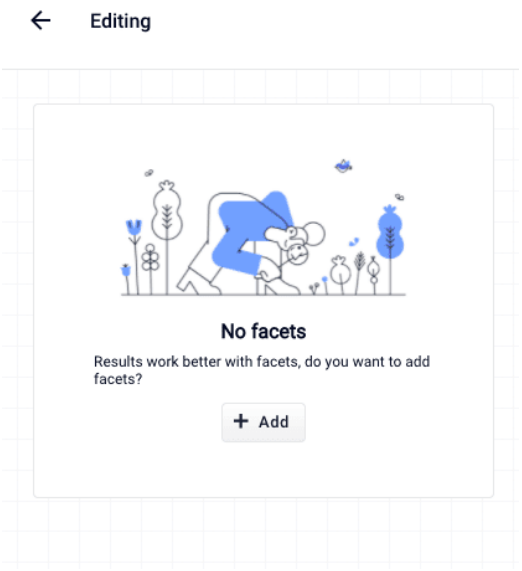
   The button appears.

2. In the left panel, click **+Add** and enter the desired parameters.
3. Click **Done** when you've finished.

> You can also add facets that are not dependent on rules being triggered. To do this, edit the Facet Zone widget (5.4 and later) or the *Facet List widget* (5.3 and earlier).

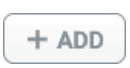
Edit the Facet Zone widget (5.4 and later)

1. In the facet zone, click **+Add**. The **Edit Rule** dialog box appears.



2. Click the **Facet Field** field, and enter the desired facet parameter.

> **Tip**
>
> You can add multiple facet parameters by clicking the **+ Add** button:



1. Click the **Save** button to add your facet:

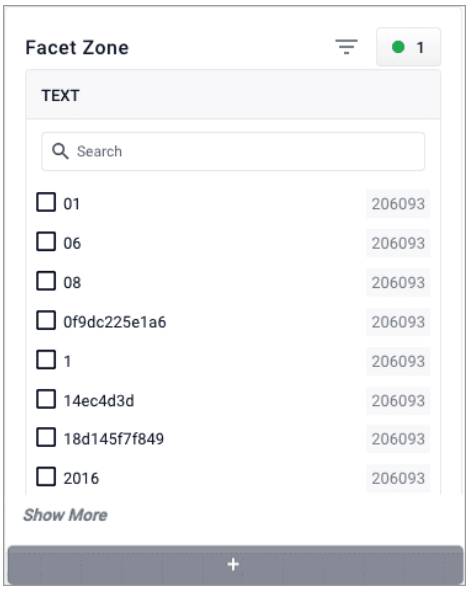
Edit the Facet List widget (5.3 and earlier)

1. Select **Add facet**. The **Edit Rule** dialog box appears.
2. Select the **Facet Field** field, and enter the desired facet parameter.

3. Click the **Save** button to add your facet:

Add Additional Facets

1. Hover your cursor above or below an existing facet. A **+** button appears:



2. Click the **+** button.

Move a Facet

1. Hover your cursor over a facet. A menu overlay appears at the top.
2. Click-and-hold the drag bar:

3. Move the facet to the desired location.
4. Release the click.

Remove a Facet

1. Hover your cursor over a facet. A menu overlay appears at the top.
2. Click the **Remove** button:

Block a Facet Item

1. Hover your cursor over a facet item. A menu overlay appears.
2. Click the **Block** button:

Bury a Facet Item

1. Hover your cursor over a facet item. A menu overlay appears.
2. Click the **Bury** button:

Boost a Facet Item

1. Hover your cursor over a facet item. A menu overlay appears.
2. Click the **Boost** button:

Reorder Boosted Facet Items

1. Hover over a boosted facet item. A drag bar appears.
2. Click and hold the drag bar:

3. Move the facet item to the desired location.
4. Release the click.

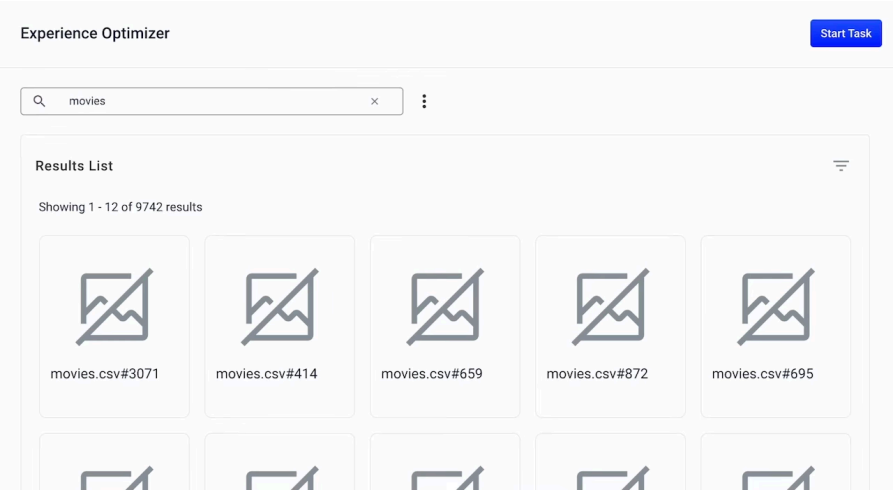
Use Experience Optimizer query rewrites

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

You can create query rewrite rules in Experience Optimizer. This is similar to using the *Rules Editor* to create query rewrite rules, except there are additional options in Experience Optimizer.

1. In Managed Fusion, navigate to **Relevance** > **Rules** > **Optimizer**.
2. Enter a search term or phrase in the search bar.
3. Click **Start Task**.
4. Hover your cursor next to the query. A **+** button appears:
5. Click the **+** button. A list of query rewrites options will appear: *Head/Tail*, *Misspelling*, *Phrase*, *Synonym*, and *Remove Words*.



6. Click on the type of rule you'd like to configure.

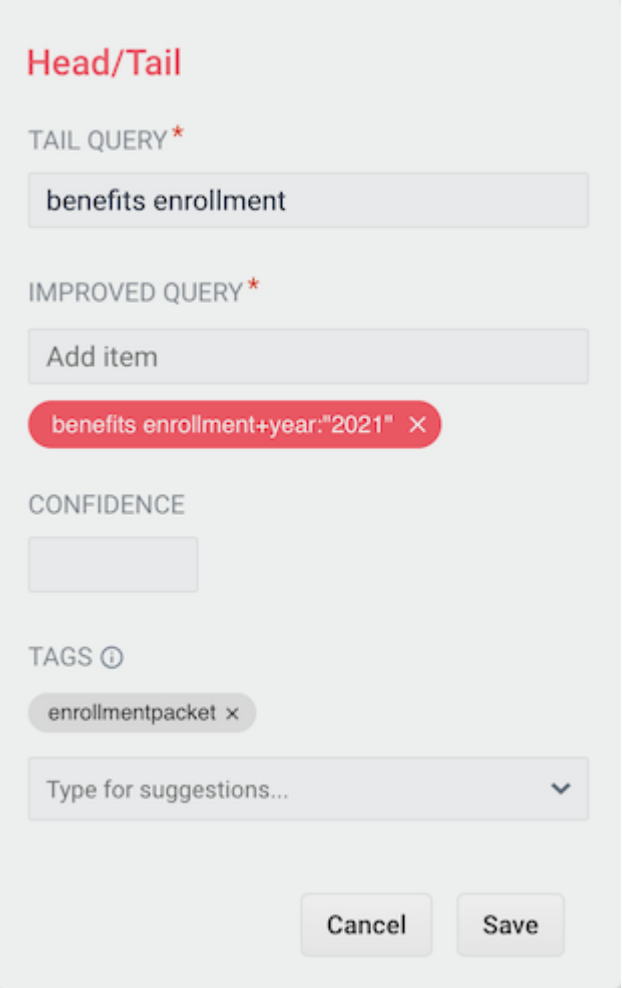
Head/Tail

You can create a Head/Tail rewrite to improve search results using methods other than correcting for misspellings or synonym expansion. When a poorly defined search term is identified, the original term is replaced by an improved search term.

For example, a search for `benefits enrollment` could be improved by using the search term `benefits enrollment+year:"2021"` (in this case making use of the year field in the data).

Most Head/Tail rewrites are typically created automatically via machine learning. However, if desired, custom rewrites can be manually created using the following steps.

1. From the list of query rewrite options, select **Head/Tail**. A form will appear:

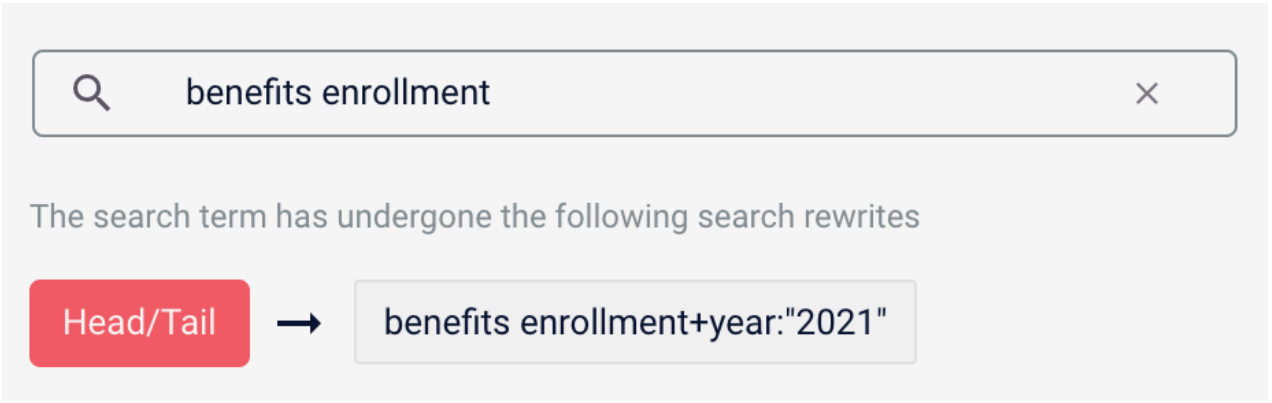


This form contains the following fields:

Parameter	Description	Example Value
Tail Query	The tail query itself.	`benefits enrollment`
Improved Query	The query that will replace the tail query phrase.	`benefits enrollment+year:"2021"`
Tags	Optional metadata tags that can be used to identify and organize rewrites.	`enrollmentpacket`

2. Enter one or more improved search terms in the **Improved Query** field.
3. Click the **Save** button.

The search is re-run using the improved search term, and a Search box information panel drops down to show that the new query rewrite has fired:

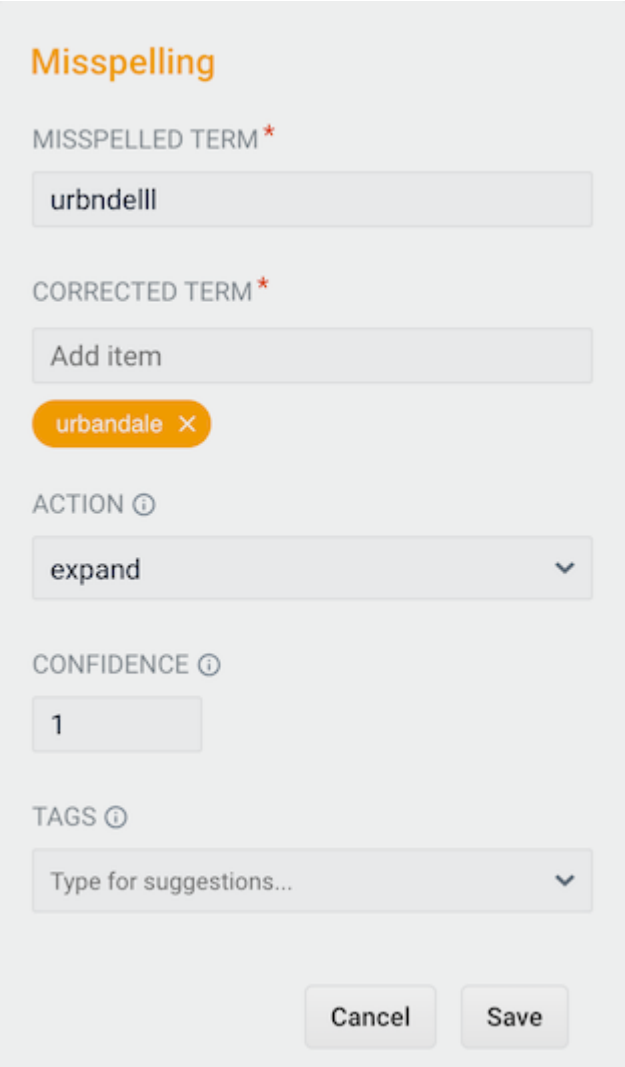


Misspelling

You can create a misspelling query rewrite to detect and correct common spelling mistakes. When a customer enters a search term containing a known misspelling, the incorrect spelling is replaced with the spelling correction.

For example, if your clients frequently misspell or mistype the word `questionnaire` as `questionare`, you can set up a query rewrite to automatically correct it.

1. From the list of query rewrite options, select **Misspelling**. A form will appear:

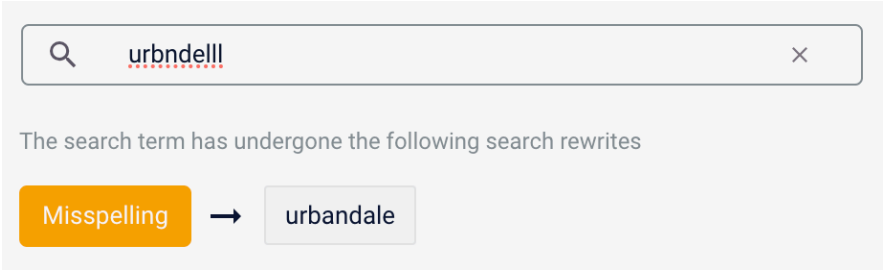


This form contains the following fields:

Parameter	Description	Example Value
Misspelled Term	The phrase itself.	`questionare`
Corrected Term	The term that will replace the misspelled term.	`questionnaire`
Action	Action to perform.	`expand`
Confidence	Confidence score from the phrase job. A confidence level of `1` represents 100% confidence. For rules created automatically via machine learning, the confidence level reflects the output from the machine learning model.	`1`
Tags	Optional metadata tags that can be used to identify and organize rewrites.	`enrollmentpacket`

2. Enter one or more spelling corrections in the **Corrected Term** field.
3. Click the **Save** button.

The search is re-run using the corrected spelling, and a Search box information panel drops down to show that the new query rewrite has fired:

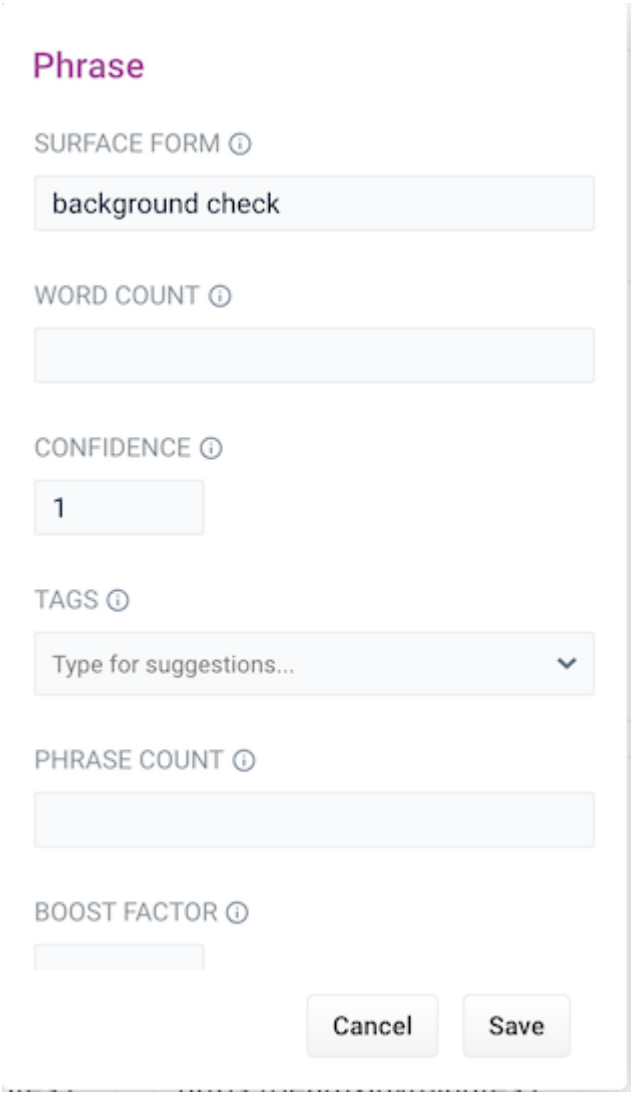


Phrase

You can use query rewriting to identify phrases used in search terms so that products with matching phrases are boosted in the search results. This is helpful when users do not use quotation marks to identify phrases in their search terms.

For example, without phrase detection a search for the words `background check` would show results for both `background` and `check`. With phrase detection, this search would correctly boost results for `"background check"`.

1. From the list of query rewrite options, select **Phrase**. A form will appear:

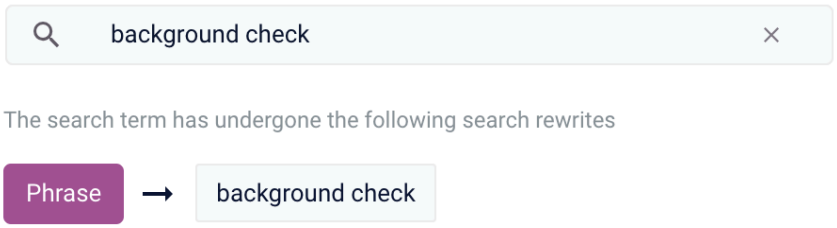


This form contains the following fields:

Parameter	Description	Example Value
Surface Form	The phrase itself.	`background check`
Word Count	Indicates how many words are included in the phrase.	`2`
Confidence	Confidence score from the phrase job. A confidence level of `1` represents 100% confidence. For rules created automatically via machine learning, the confidence level will reflect the output from the machine learning model.	`1`
Tags	Optional metadata tags that can be used to identify and organize rewrites.	`enrollmentpacket`
Phrase Count	Denotes how many times this phrase was found in the source. This value is automatically set via machine learning. It does not need to be set manually.	`5`
Boost Factor	The factor to use to boost this phrase in matching queries.	`2.0`
Slop Factor	Phrase slop, or the distance between the terms of the query while still considering it a phrase match.	`10`

2. Enter the number of words in the phrase in the **Word Count** field.
3. Click the **Save** button.

The search is re-run, with your search words identified as a phrase, and a Search box information panel drops down to show that the new query rewrite has fired:

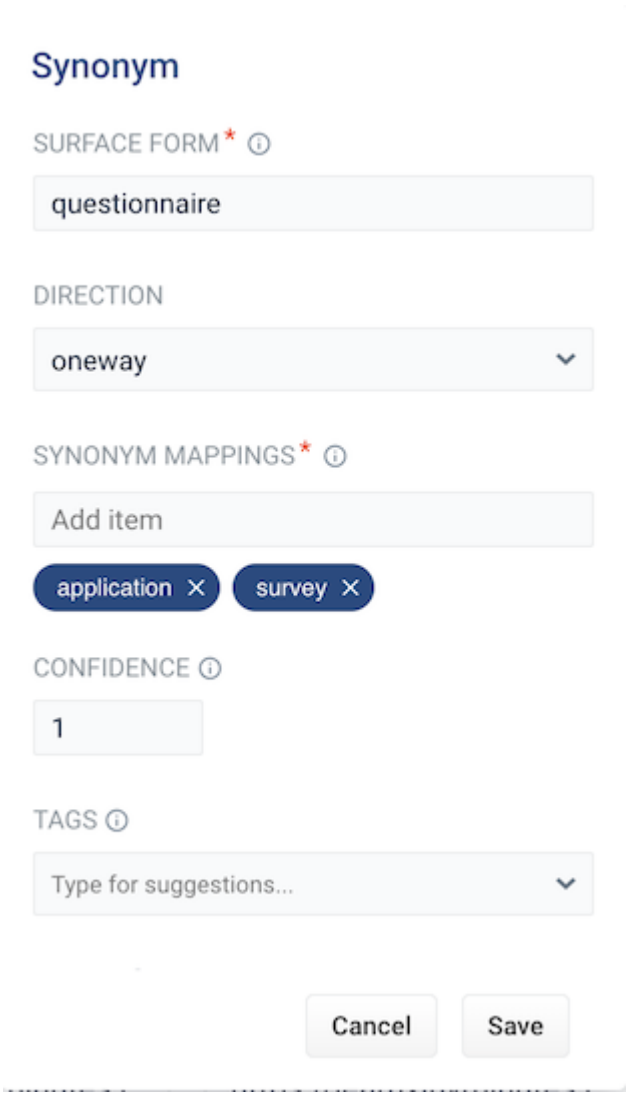


Synonym

You can specify synonyms for a specified search term so that alternative words with the same meaning are automatically used in the search query. When a customer enters a search term with a synonym match, the alternative words are used instead of, or in addition to, the original search term.

For example, the search term `questionnaire` could have the synonyms `application` and `survey`.

1. From the list of query rewrite options, select **Synonym**. A form will appear:

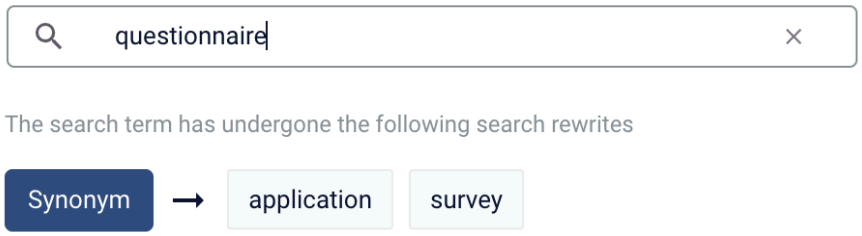


This form contains the following fields:

Parameter	Description	Example Value
Surface Form	The term that has synonyms.	`questionnaire`
Direction	With a **oneway** search, the original search term is replaced by the synonym. In the preceding example, `questionnaire` would be replaced by the alternative words `application` and `survey`. With a **symmetric** search, the search query is expanded to include the original term and the synonyms, resulting in a greater number of potential hits. In the preceding example, this time the query would include `questionnaire`, `application`, and `survey`.	`symmetric`
Synonym Mappings	Synonyms for the surface form.	`application`, `survey`
Confidence	Confidence score from the phrase job. A confidence level of `1` represents 100% confidence. For rules created automatically via machine learning, the confidence level will reflect the output from the machine learning model.	`1`
Tags	Optional metadata tags that can be used to identify and organize rewrites.	`enrollment packet`
Count	How many times this term occurred in the signal data when it was discovered. This value is optional when a rewrite is being defined manually.	`5`

2. Choose whether the direction is **oneway** or **symmetric**.
3. Enter one or more alternative words in the **Synonym Mappings** field.
4. Click the **Save** button.

The search is re-run using the synonyms, and a Search box information panel drops down to show that the new query rewrite has fired:



Remove Words

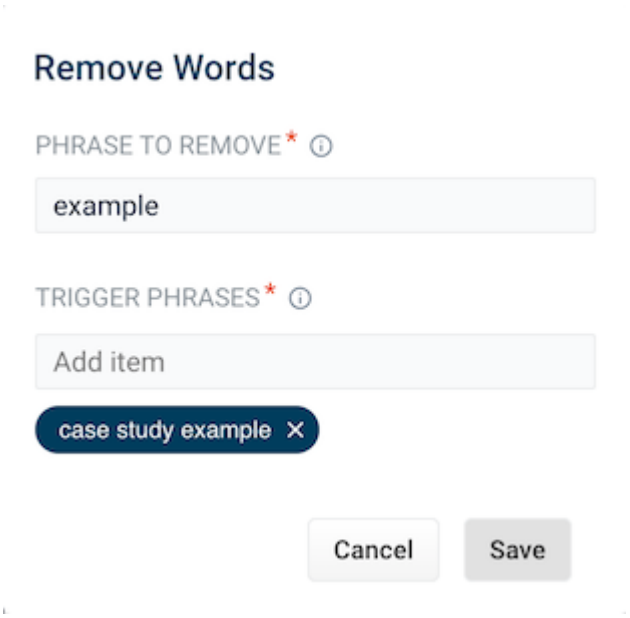
Note

The Remove Words feature is available in Fusion 5.4 and later.

You can create a Remove Words query rewrite to remove words from a query. This query rewrite is helpful when a word in the search query does not add value to the search results.

For example, you can rewrite a search query for `case study examples` to remove `examples` and then display results for `case study`.

1. From the list of query rewrite options, select Remove Words. A form appears:

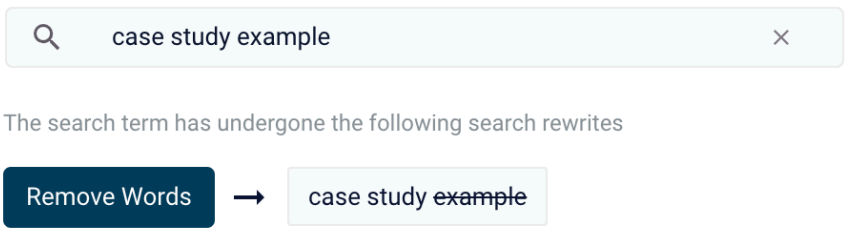


2. This form contains the following fields:

Parameter	Description	Example Value
Phrase to remove	The words to remove from the trigger phrase.	`examples`
Trigger phrases	The query that prompts the removal of the phrase. The trigger phrase is not necessarily a complete query. If the query contains the trigger phrase, then Fusion removes the phrase in the Phrase to Remove field.	`case study examples`

1. Enter a phrase to remove and a trigger phrase. Note that the phrase to remove is auto-populated with the query.
2. Click **Save**.

The search is re-run, and a Search box information panel drops down to show that the new query rewrite has fired:



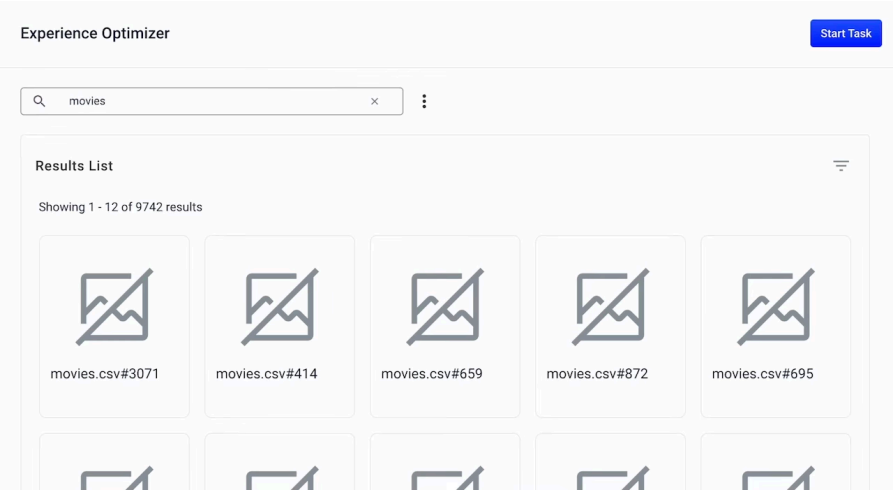
Use Experience Optimizer Query Rewrites

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion*

You can create query rewrite rules in Experience Optimizer. This is similar to using the *Rules Editor* to create query rewrite rules, except there are additional options in Experience Optimizer.

1. In Fusion, navigate to **Relevance** > **Rules** > **Optimizer**.
2. Enter a search term or phrase in the search bar.
3. Click **Start Task**.
4. Hover your cursor next to the query. A **+** button will appear:
5. Click the **+** button. A list of query rewrites options will appear: *Head/Tail*, *Misspelling*, *Phrase*, *Synonym*, and *Remove Words*.

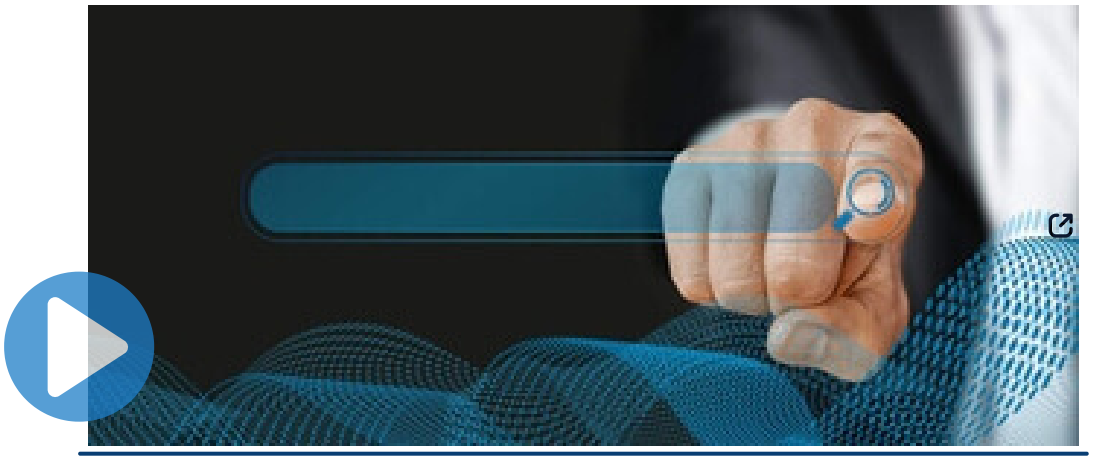


6. Click on the type of rule you'd like to configure.

LucidAcademy

Lucidworks offers free training to help you get started.

The **Course** for **Experience Optimizer: Rewrites Manager** focuses on how to create search rewrites that optimize your user's search results:



Visit the LucidAcademy to see the full training catalog.

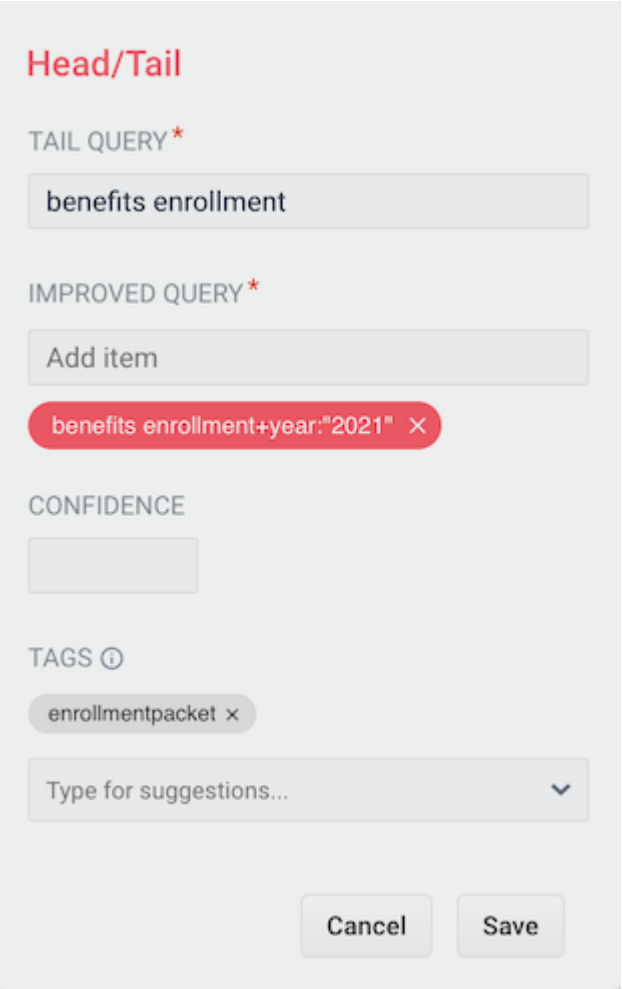
Head/Tail

You can create a Head/Tail rewrite to improve search results using methods other than correcting for misspellings or synonym expansion. When a poorly defined search term is identified, the original term is replaced by an improved search term.

For example, a search for `benefits enrollment` could be improved by using the search term `benefits enrollment+year:"2021"` (in this case making use of the year field in the data).

Most Head/Tail rewrites will typically be created automatically via machine learning. However, if desired, custom rewrites can be manually created using the following steps.

1. From the list of query rewrite options, select **Head/Tail**. A form will appear:

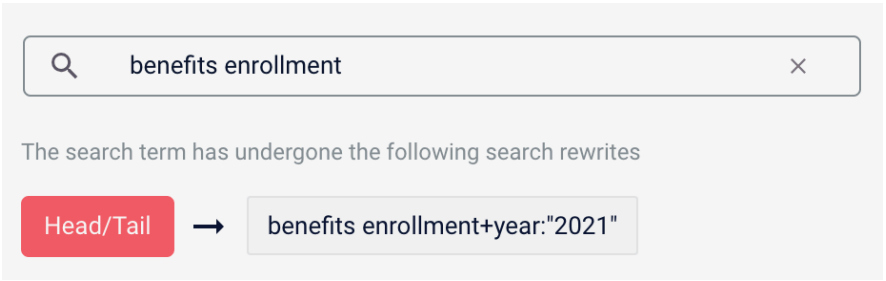


This form contains the following fields:

Parameter	Description	Example Value
Tail Query	The tail query itself.	`benefits enrollment`
Improved Query	The query that will replace the tail query phrase.	`benefits enrollment+year:"2021"`
Tags	Optional metadata tags that can be used to identify and organize rewrites.	`enrollmentpacket`

2. Enter one or more improved search terms in the **Improved Query** field.
3. Click the **Save** button.

The search is re-run using the improved search term, and a Search box information panel drops down to show that the new query rewrite has fired:

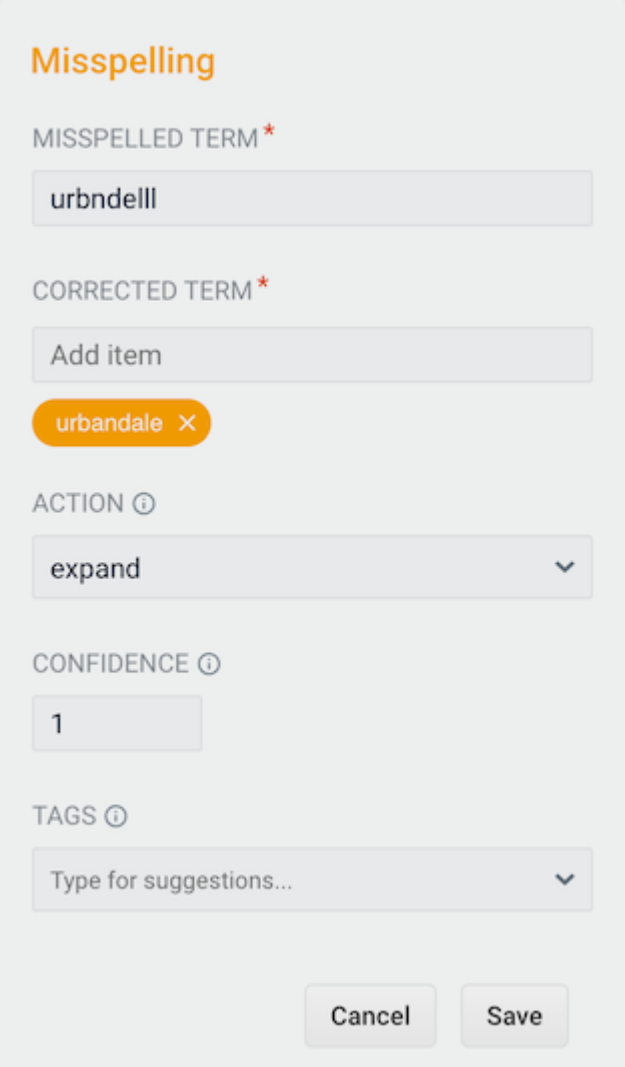


Misspelling

You can create a misspelling query rewrite to detect and correct common spelling mistakes. When a customer enters a search term containing a known misspelling, the incorrect spelling is replaced with the spelling correction.

For example, if your clients frequently misspell or mistype the word `questionnaire` as `questionare`, you can set up a query rewrite to automatically correct it.

1. From the list of query rewrite options, select **Misspelling**. A form will appear:

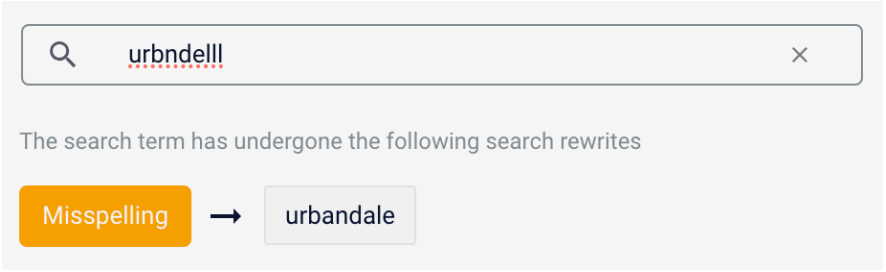


This form contains the following fields:

Parameter	Description	Example Value
Misspelled Term	The phrase itself.	`questionare`
Corrected Term	The term that will replace the misspelled term.	`questionnaire`
Action	Action to perform.	`expand`
Confidence	Confidence score from the phrase job. A confidence level of `1` represents 100% confidence. For rules created automatically via machine learning, the confidence level will reflect the output from the machine learning model.	`1`
Tags	Optional metadata tags that can be used to identify and organize rewrites.	`enrollmentpacket`

2. Enter one or more spelling corrections in the **Corrected Term** field.
3. Click the **Save** button.

The search is re-run using the corrected spelling, and a Search box information panel drops down to show that the new query rewrite has fired:

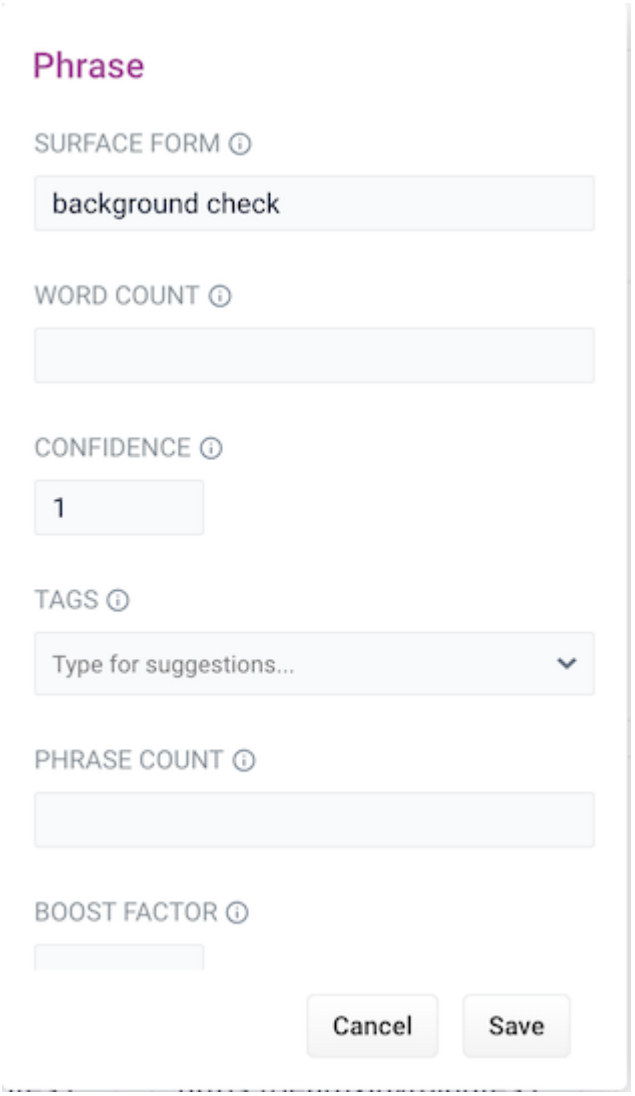


Phrase

You can use query rewriting to identify phrases used in search terms so that products with matching phrases are boosted in the search results. This is helpful when users do not use quotation marks to identify phrases in their search terms.

For example, without phrase detection a search for the words `background check` would show results for both `background` and `check`. With phrase detection, this search would correctly boost results for `"background check"`.

1. From the list of query rewrite options, select **Phrase**. A form will appear:

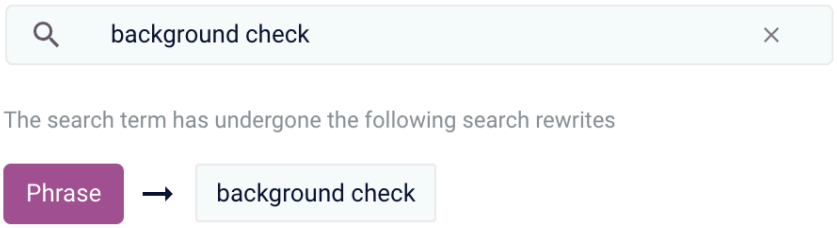


This form contains the following fields:

Parameter	Description	Example Value
Surface Form	The phrase itself.	`background check`
Word Count	Indicates how many words are included in the phrase.	`2`
Confidence	Confidence score from the phrase job. A confidence level of `1` represents 100% confidence. For rules created automatically via machine learning, the confidence level will reflect the output from the machine learning model.	`1`
Tags	Optional metadata tags that can be used to identify and organize rewrites.	`enrollmentpacket`
Phrase Count	Denotes how many times this phrase was found in the source. This value is automatically set via machine learning. It does not need to be set manually.	`5`
Boost Factor	The factor to use to boost this phrase in matching queries.	`2.0`
Slop Factor	Phrase slop, or the distance between the terms of the query while still considering it a phrase match.	`10`

2. Enter the number of words in the phrase in the **Word Count** field.
3. Click the **Save** button.

The search is re-run, with your search words identified as a phrase, and a Search box information panel drops down to show that the new query rewrite has fired:

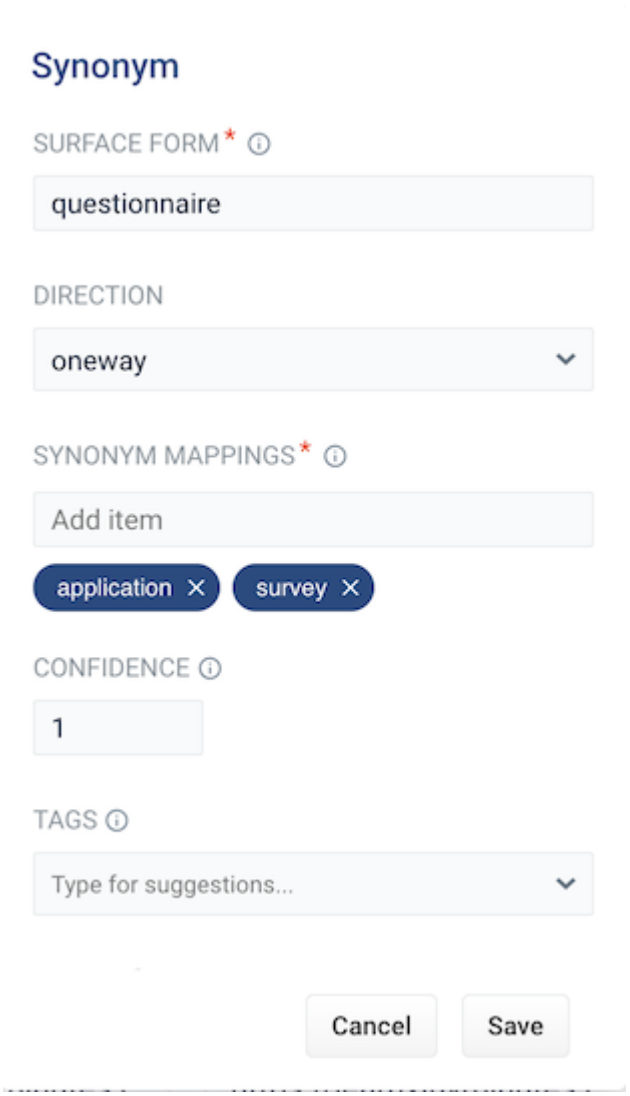


Synonym

You can specify synonyms for a specified search term so that alternative words with the same meaning are automatically used in the search query. When a customer enters a search term with a synonym match, the alternative words are used instead of, or in addition to, the original search term.

For example, the search term `questionnaire` could have the synonyms `application` and `survey`.

1. From the list of query rewrite options, select **Synonym**. A form will appear:

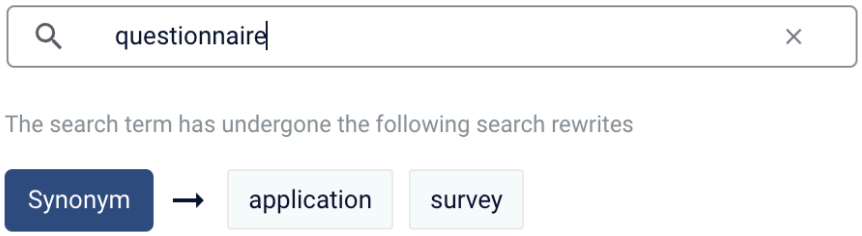


This form contains the following fields:

Parameter	Description	Example Value
Surface Form	The term that has synonyms.	`questionnaire`
Direction	With a **oneway** search, the original search term is replaced by the synonym. In the example above, `questionnaire` would be replaced by the alternative words `application` and `survey`. With a **symmetric** search, the search query is expanded to include the original term and the synonyms, resulting in a greater number of potential hits. In the example above, this time the query would include `questionnaire`, `application`, and `survey`.	`symmetric`
Synonym Mappings	Synonyms for the surface form.	`application`, `survey`
Confidence	Confidence score from the phrase job. A confidence level of `1` represents 100% confidence. For rules created automatically via machine learning, the confidence level will reflect the output from the machine learning model.	`1`
Tags	Optional metadata tags that can be used to identify and organize rewrites.	`enrollment packet`
Count	How many times this term occurred in the signal data when it was discovered. This value is optional when a rewrite is being defined manually.	`5`

2. Choose whether the direction is **oneway** or **symmetric**.
3. Enter one or more alternative words in the **Synonym Mappings** field.
4. Click the **Save** button.

The search is re-run using the synonyms, and a Search box information panel drops down to show that the new query rewrite has fired:



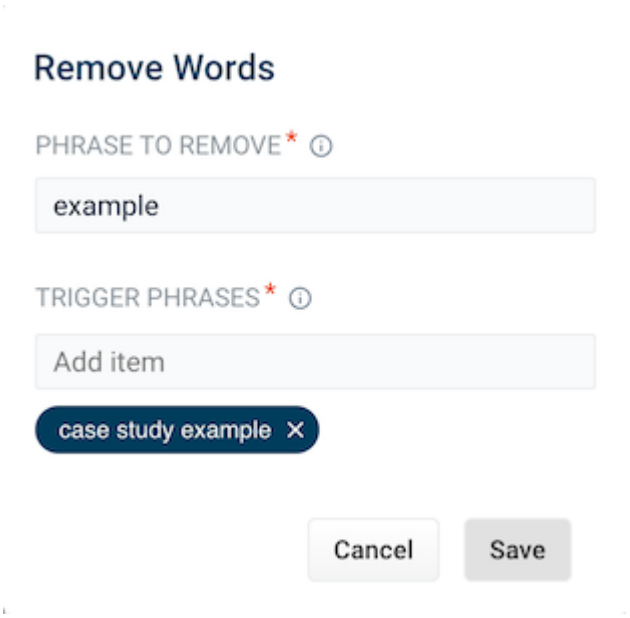
Remove Words

> **Note**
>
> The Remove Words feature is available in Fusion 5.4 and later.

You can create a Remove Words query rewrite to remove words from a query. This query rewrite is helpful when a word in the search query does not add value to the search results.

For example, you can rewrite a search query for `case study examples` to remove `examples` and then display results for `case study`.

1. From the list of query rewrite options, select Remove Words. A form appears:

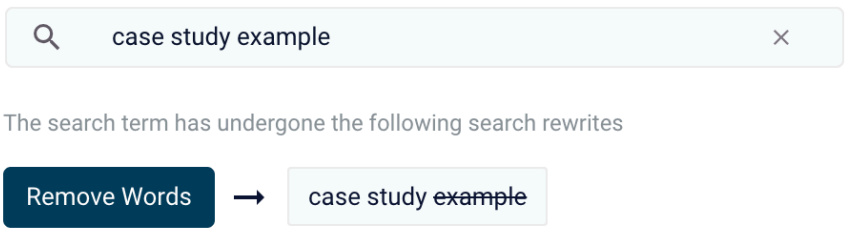


2. This form contains the following fields:

Parameter	Description	Example Value
Phrase to remove	The words to remove from the trigger phrase.	`examples`
Trigger phrases	The query that prompts the removal of the phrase. The trigger phrase is not necessarily a complete query. If the query contains the trigger phrase, then Fusion removes the phrase in the Phrase to Remove field.	`case study examples`

1. Enter a phrase to remove and a trigger phrase. Note that the phrase to remove is auto-populated with the query.
2. Click **Save**.

The search is re-run, and a Search box information panel drops down to show that the new query rewrite has fired:



Use Experience Optimizer Tasks

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion*

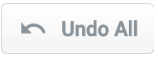
Start a task

1. Click the **Start Task** button:

This enables Interactive Mode. In Interactive Mode, you can interact directly with some of the widgets on the Experience Optimizer board, creating and managing business rules as you simulate your customers' searches. You can:

- Pin, block, boost, and bury documents.
- Set additional facets.
- Create search rewrites from poorly performing search terms.
- See which rules and search rewrites are triggered by a search, then edit or delete them as needed.
- Sort search results by ascending or descending order, or by relevance.
- Check product information.

Undo changes in a task

1. Click the **Undo All** button: 

. A confirmation dialog box will appear.

2. Select **Undo** to confirm.

Exit a task

Exiting a task will

1. Click the **Exit** button:

Publish a task

You can publish any changes you make to rules back to the Fusion data source where they can impact live queries.

1. Click the **Publish** button:

. A confirmation dialog box will appear.

2. Select **Publish** to confirm.

Note

Experience Optimizer counts unpublished changes in the top navigation bar:

Use Experience Optimizer Tasks

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

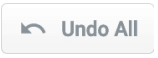
Start a task

1. Click the **Start Task** button:

This enables Interactive Mode. In Interactive Mode, you can interact directly with some of the widgets on the Experience Optimizer board, creating and managing business rules as you simulate your customers' searches. You can:

- Pin, block, boost, and bury documents.
- Set additional facets.
- Create search rewrites from poorly performing search terms.
- See which rules and search rewrites are triggered by a search, then edit or delete them as needed.
- Sort search results by ascending or descending order, or by relevance.
- Check product information.

Undo changes in a task

1. Click the **Undo All** button: 

. A confirmation dialog box will appear.

2. Select **Undo** to confirm.

Exit a task

Exiting a task will

1. Click the **Exit** button:

Publish a task

You can publish any changes you make to rules back to the Fusion data source where they can impact live queries.

1. Click the **Publish** button:

. A confirmation dialog box will appear.

2. Select **Publish** to confirm.

Note

Experience Optimizer counts unpublished changes in the top navigation bar:

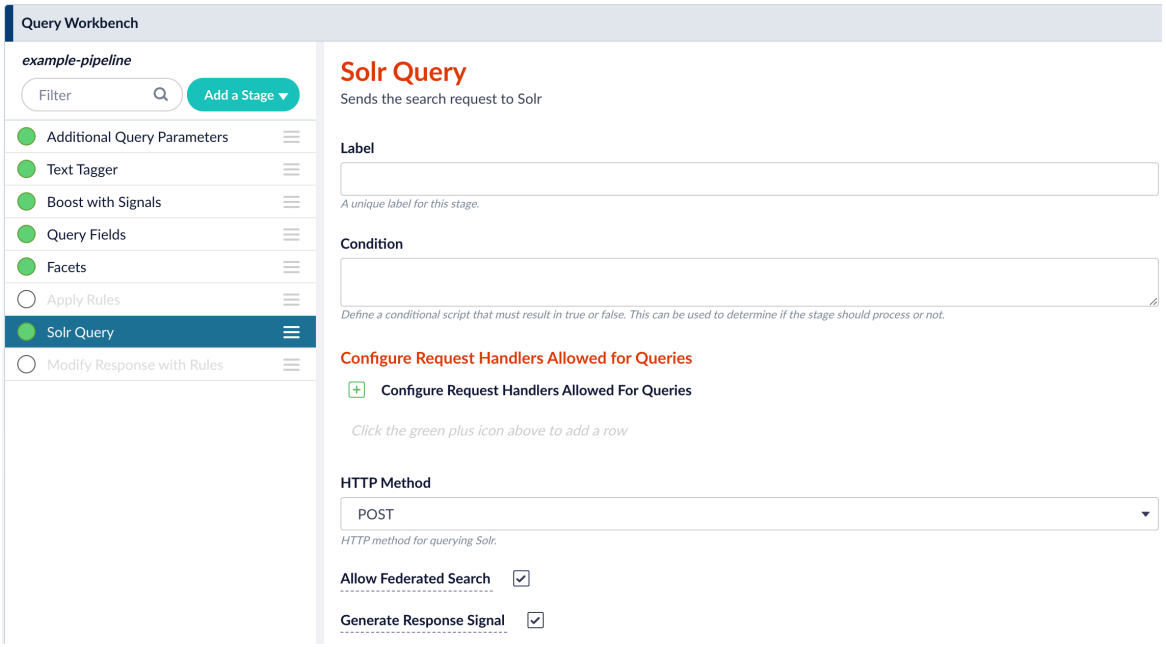
Use Federated Search

How-to

> Compatible versions: 4.2, 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion Server*, *Fusion*

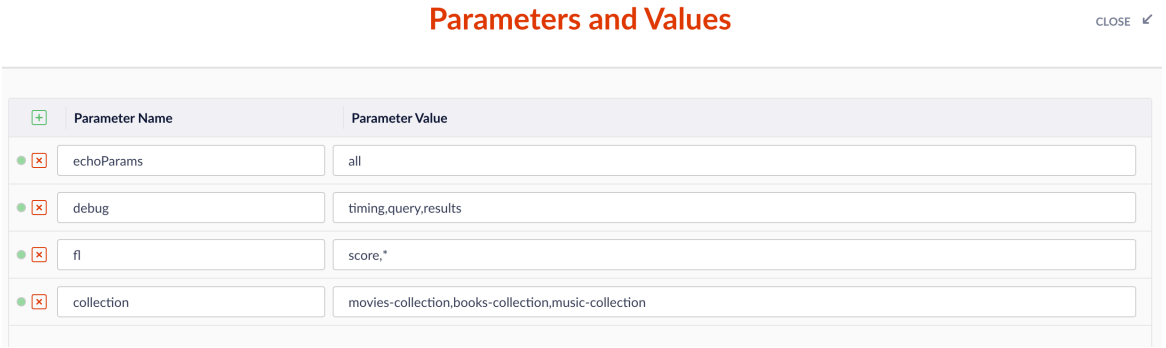
Federated search lets you query across multiple collections in Fusion. This is useful if you keep separate data collections for security or compliance reasons or maintain different collections based on data type.

1. In the Fusion workspace, navigate to **Querying > Query Workbench**.
2. Click **Solr Query** to open the Solr Query panel.
3. Enable **Allow Federated Search** then click **Apply**.



4. In the workbench area, click **Parameters** then click **Edit Parameters**.

5. Click Add ⊞ to add a parameter. For the **Name**, enter `collection` and for the **Parameter Value**, enter a comma-separated list of collections you want to query. For example, `movies-collection,books-collection,music-collection`.



6. Click **Close**.
7. Check that your pipeline is querying documents from all specified collections.

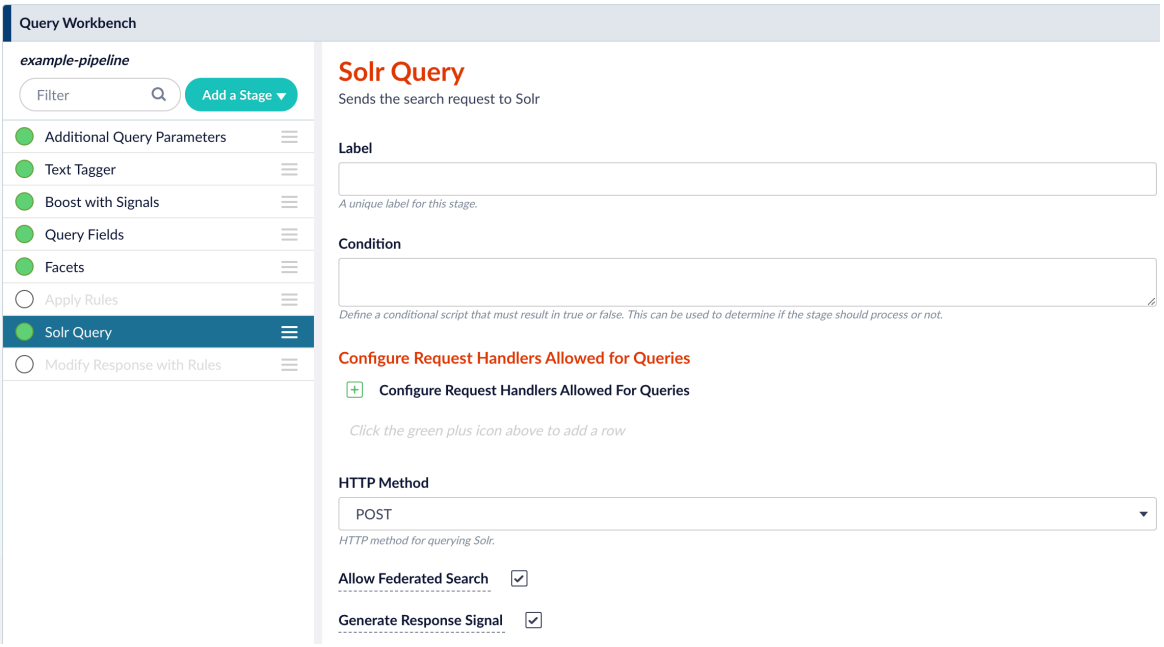
Use Federated Search

How-to

> Compatible versions: 4.2, 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

Federated search lets you query across multiple collections in Fusion. This is useful if you keep separate data collections for security or compliance reasons or maintain different collections based on data type.

1. In the Fusion workspace, navigate to **Querying > Query Workbench**.
2. Click **Solr Query** to open the Solr Query panel.
3. Enable **Allow Federated Search** then click **Apply**.



4. In the workbench area, click **Parameters** then click **Edit Parameters**.

5. Click Add ⊞ to add a parameter. For the **Name**, enter `collection` and for the **Parameter Value**, enter a comma-separated list of collections you want to query. For example, `movies-collection,books-collection,music-collection`.

6. Click **Close**.
7. Check that your pipeline is querying documents from all specified collections.

Use Git with Appkit

How-to

> Compatible versions: 4.2, 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *App Studio*

At Lucidworks, we use GitHub for hosting and Git for version control of our code and customers' code. There are several good guides on the web to using Git detailed below, along with some high level instructions for getting started.

Getting access to the project

To gain access to a project, you must sign up for a free GitHub account. After you are signed up, contact Lucidworks Support and tell us who you are, your GitHub username, and which project you work on, and we can add you to the right project.

Using the Fork-and-Branch Git workflow

When working with multiple developers on a single codebase, we use a Fork-and-Branch model. This lets you create a copy of a repository, make your changes and then submit them back to the original repository for review by the owner before they are merged in.

GitHub Desktop App (Windows and Mac)

If you use the GitHub desktop application (details and install guide) there is a good Github guide on how forking with the GitHub desktop app works.

Command Line (Windows and Mac)

If you use the command-line git tools then the “fork and branch” workflow process is documented well in this blog post:

The article above covers it really well, but at a high level the process is as follows:

1. Fork a GitHub repository.
2. Clone the forked repository to your local system.
3. Add a Git remote for the original repository.
4. Create a feature branch in which to place your changes.
5. Make your changes to the new branch.
6. Commit the changes to the branch.
7. Push the branch to GitHub.
8. Open a pull request from the new branch to the original repo.
9. Keeping Your Fork in Sync.

If you are stuck, search for a specific command in the Pro Git book; it is a useful resource.

Forking a GitHub Repository

Forking a repository is really straightforward:

Make sure you’re logged into GitHub with your account. Find the GitHub repository with which you’d like to work.

Click the Fork button on the upper right-hand side of the repository’s page. That’s it — you now have a copy of the original repository in your GitHub account.

Making a local clone

Even though you have a copy of the original repository in your GitHub account, you must copy it (clone it) locally so you can work on it. Navigate to the forked repository on github.com (this is the copy of the original repository residing in your GitHub account) and look on the right-hand side of the web page. You should see an area that is labeled “HTTPS clone URL”. Simple copy the URL there and use as follows:

```
git clone https://github.com/username/projectname.git
```

Here, **username** is your GitHub username and **projectname** is the specific name of the project (GitHub provides you the URL to copy and paste for ease).

Git will copy down the repository, both contents and commit history, to your system. Git will also add a Git remote named `origin` that points back to the forked repository in your GitHub account.

Adding a Git Remote

Git already added a Git remote named `origin` to the clone of the Git repository on your system, and this will let you push changes back up to the forked repository in your GitHub account using `git commit` (to add commits locally) and `git push`.

To use the "fork and branch" workflow, you'll must add a Git remote pointing back to the original repository (the one you forked on GitHub).

You'd want to add a Git remote that points back to the original repository, like this:

```
git remote add upstream https://github.com/twigkit/projectname.git
```

Of course, you'd want to replace the URL after `git remote add upstream` with the appropriate clone URL for the original project repository.

Keeping Your Fork in Sync

Your forked repository does not stay in sync automatically - you must take care of this yourself - it is good practice to do this before you start work to pull in changes from other developers. To ensure your code has the latest upstream code changes, first make sure you are on your local master branch:

```
git checkout master
```

Then either pull the latest changes in - these commands pull (fetch and merge) the most recent changes from the upstream repository, and pushes them back to your repository:

```
git pull upstream master git push origin master
```

Or explicitly fetch and merge the most recent changes from the upstream repository:

```
git fetch upstream git merge upstream/master git push origin master
```

Working with feature branches

The master branch is the master copy of the code. When working on individual features, it is better to work in branches. Using branches, you can switch between different development tasks easily mid-flow.

CREATE A NEW BRANCH

```
git checkout –b <new branch name>
```

For example, suppose you are working on ISSUE-1234. Then before doing any code changes, make sure you create a new feature branch like so:

```
git checkout –b ISSUE–1234
```

COMMIT CHANGES

1. See what files you have changed and what is queued up to commit:

```
git status
```

2. Add some changes you have made to commit:

```
git add </path/to/filename>
```

3. Make your changes and then commit the changes to the branch:

```
git commit –m 'short descriptive note about my changes'
```

PUSH CHANGES BACK TO YOUR REPOSITORY

When you're ready to commit your changes, push those changes up to the corresponding feature branch on your remote fork. This does not push the changes back to Appkit - just to your forked copy:

```
git push –u origin <new branch name>
```

See https://help.github.com/articles/pushing-to-a-remote/ and https://git-scm.com/book/en/v2/Git-Basics-Working-with-Remotes.

Other useful commands are `git branch` (lists what branches you have created) and `git remote -v` (lists what remote repositories you are linked to).

Sending changes back to the master repository - opening a Pull Request

When your feature branch is ready to have its changes brought into the master branch, you must go to your GitHub fork and issue a new pull request from that feature branch. If the branch does not already exist in GitHub, a pull request will automatically be created when you push the new branch up to your repository - GitHub will prompt you to create a pull request when you view the repo online (I'm assuming you're using your browser and not the GitHub native apps). The maintainers of the original project can use this pull request to pull your changes across to their repository and, if they approve of the changes, merge them into the main repository. See this guide on how to create a pull request in GitHub for more details.

It is good practice to ensure the pull request has a clear description of what it entails, and it should refer back to the issue that is being resolved in either the title or the description.

CODE REVIEW

After you have created the pull request, you should assign it to a colleague for peer review. If any issues are identified during the review, the reviewer can add code comments, and assign the pull request back to you to fix. After the pull request has passed its code review, the reviewer can assign the pull request to a dedicated merge master who can do the final merge.

Milestones and releases

GitHub lets you tag your work and keep track of which tasks should be completed by specific milestones or releases. You might also want to consider following this practice to ensure development work on your codebase is sufficiently prioritized.

Additional resources - great Git resources

The best reference on the web is the git SCM book - https://git-scm.com/book/en/v2 - covers everything you must know about Git - a great reference if you are stuck or if you are getting started, a simplified Git Guide.

Use Hierarchical Facets

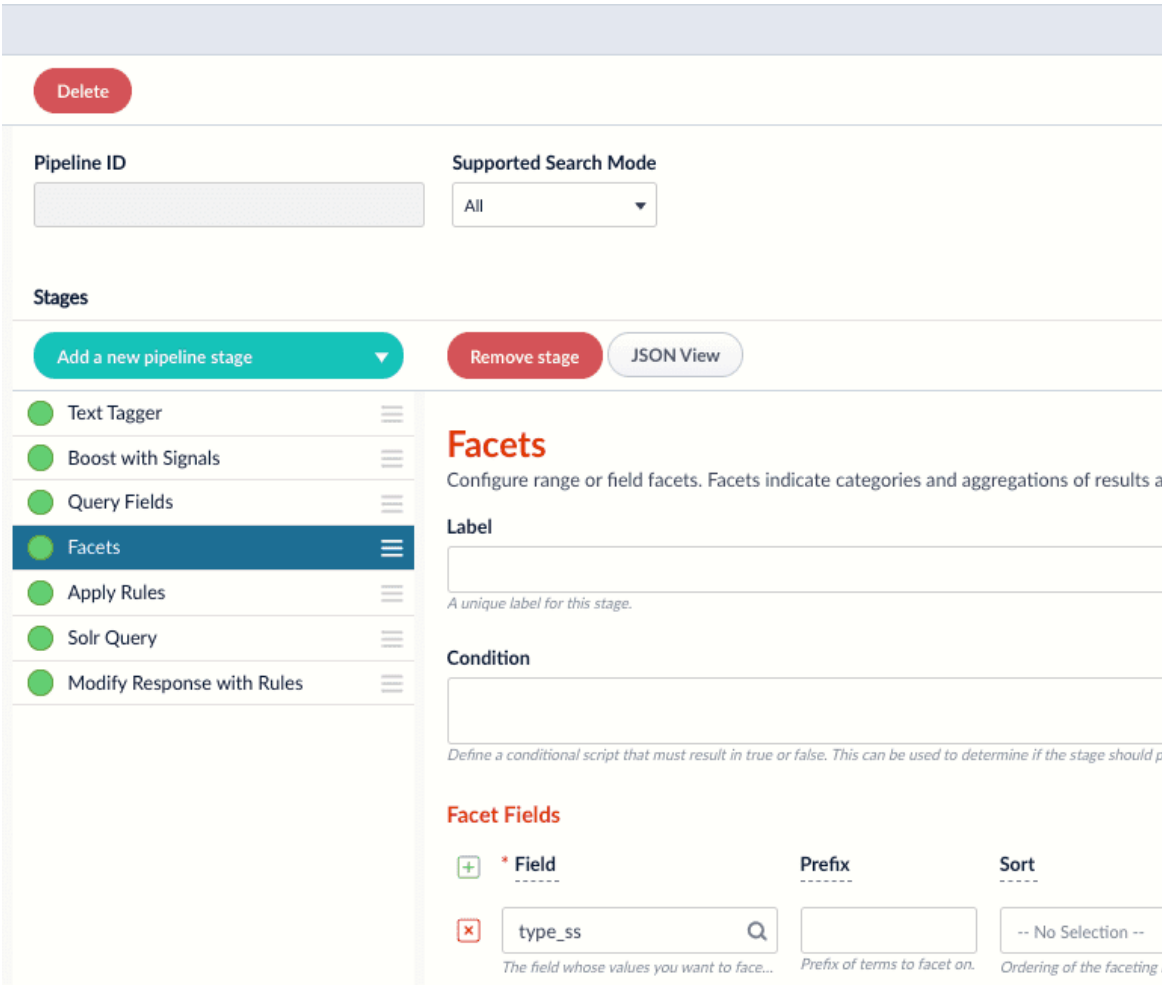
How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

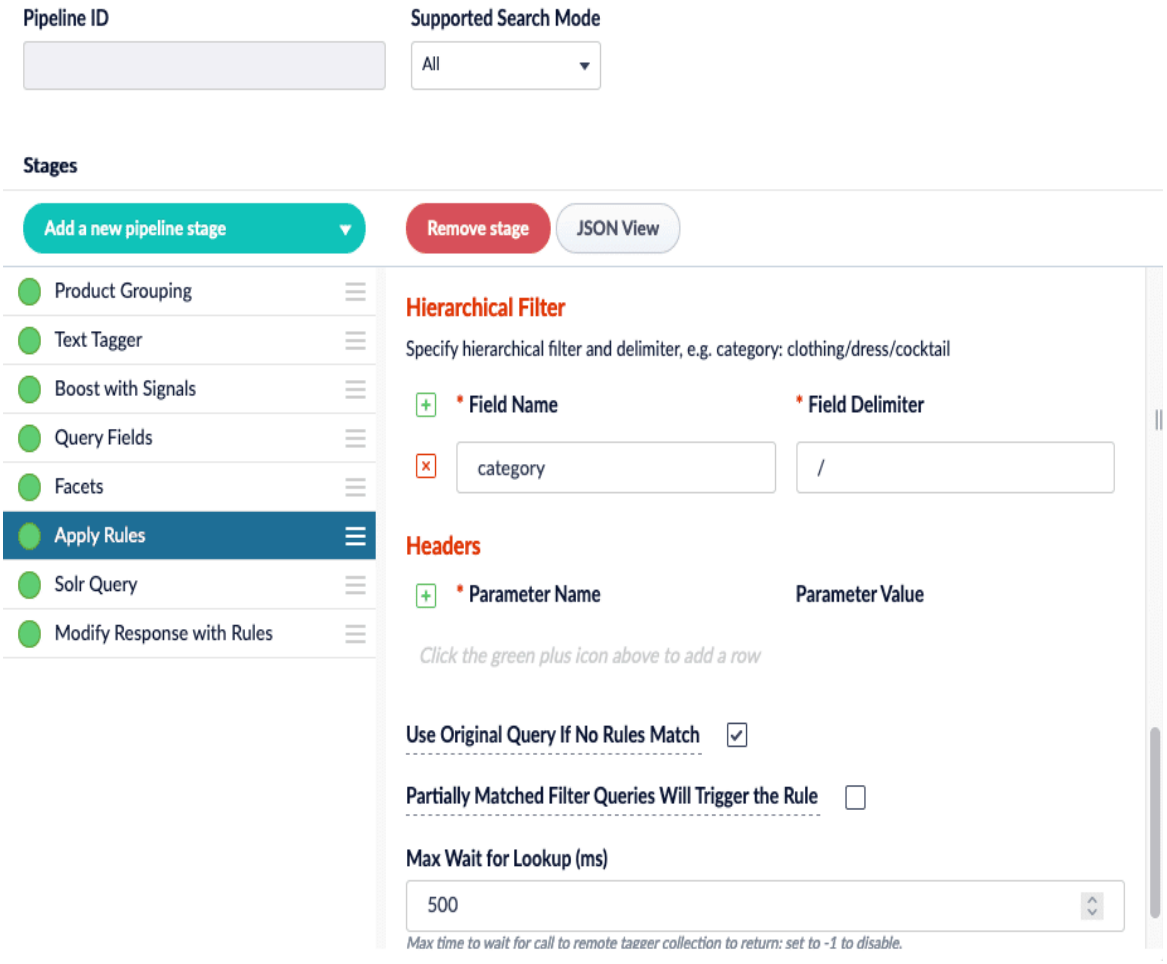
Managed Fusion 5.5.2 and earlier

To use hierarchical facets in Predictive Merchandiser, do the following:

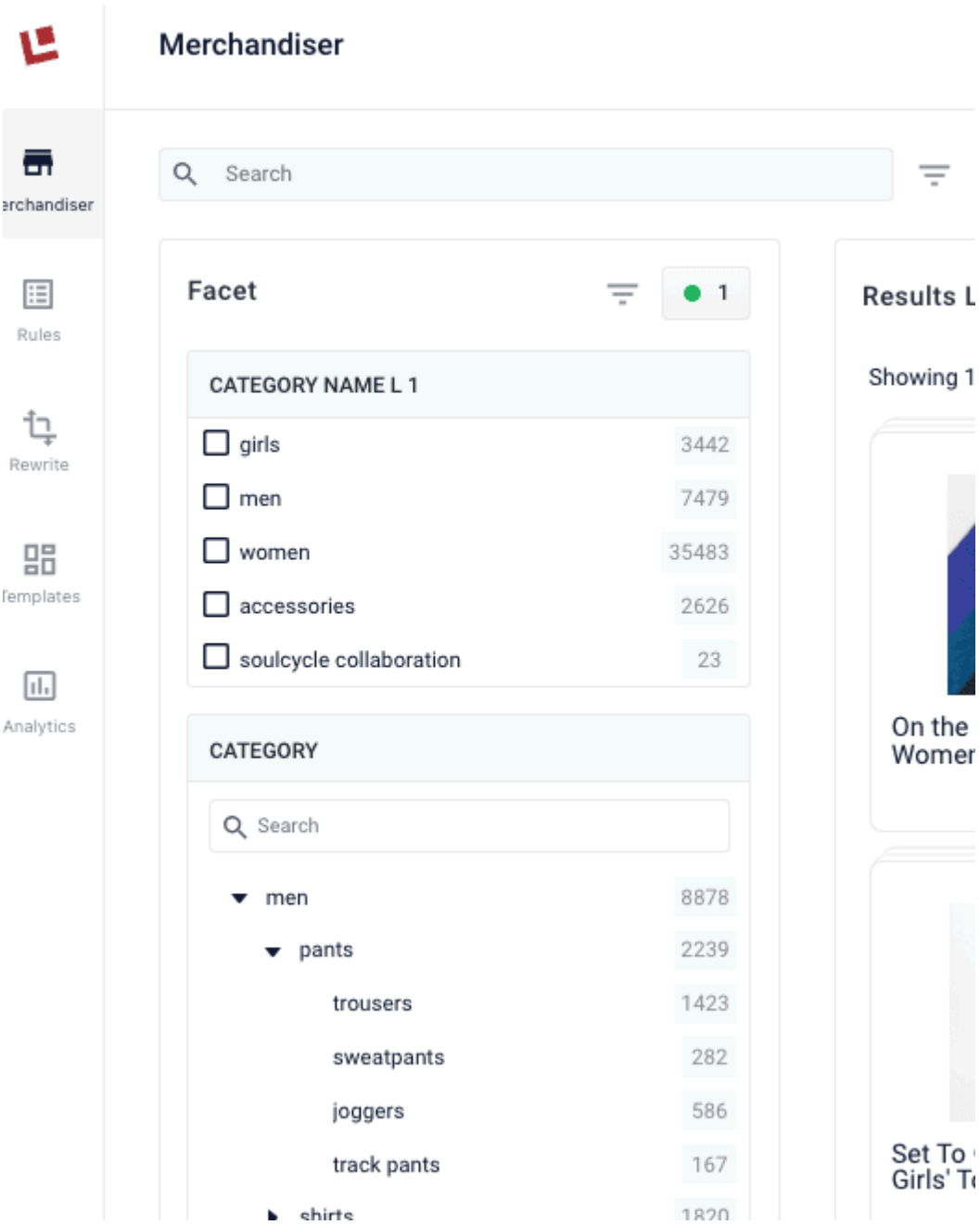
1. Select fields to facet in the *Facets* query pipeline stage.



2. In the *Apply Rules* query pipeline stage, select **Field Name** and **Field Delimiter** to use. This assumes a facet selected in Step 1 can be separated according to a delimiter, for example `/` or `|`.



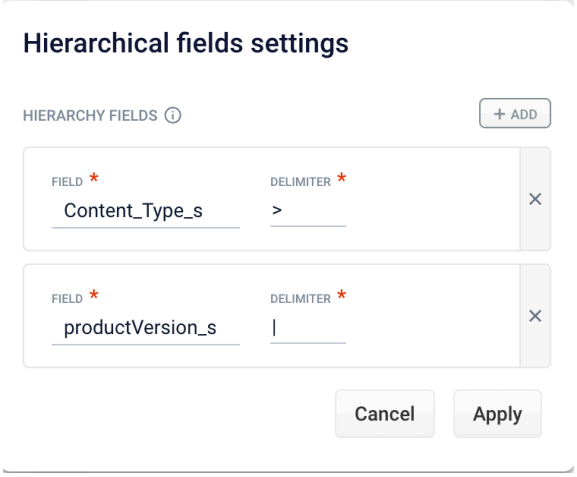
3. Predictive Merchandiser's facet zone can now display child facets in an expandable list.



Managed Fusion 5.6 and later

To configure hierarchical fields settings:

1. Access the application and click **Relevance > Rules > Templates**.
2. Click the three vertical dots to the right of the application name to display the Hierarchical fields settings section.



3. Complete one of the following:
   - To add a new field, click **+ Add**. Enter values in the **Field** and **Delimiter** fields and click **Apply**.

     Note

     Any symbol can be used as a delimiter.

   - To edit an existing field, click **Field** and select the value to edit. Enter the appropriate value in the **Delimiter** field and click **Apply**.
   - To delete a field setting, select the value in **Field**, click **X**, and then click **Apply**.
   - To cancel any changes before you apply them, click **Cancel**.

Use Hierarchical Facets

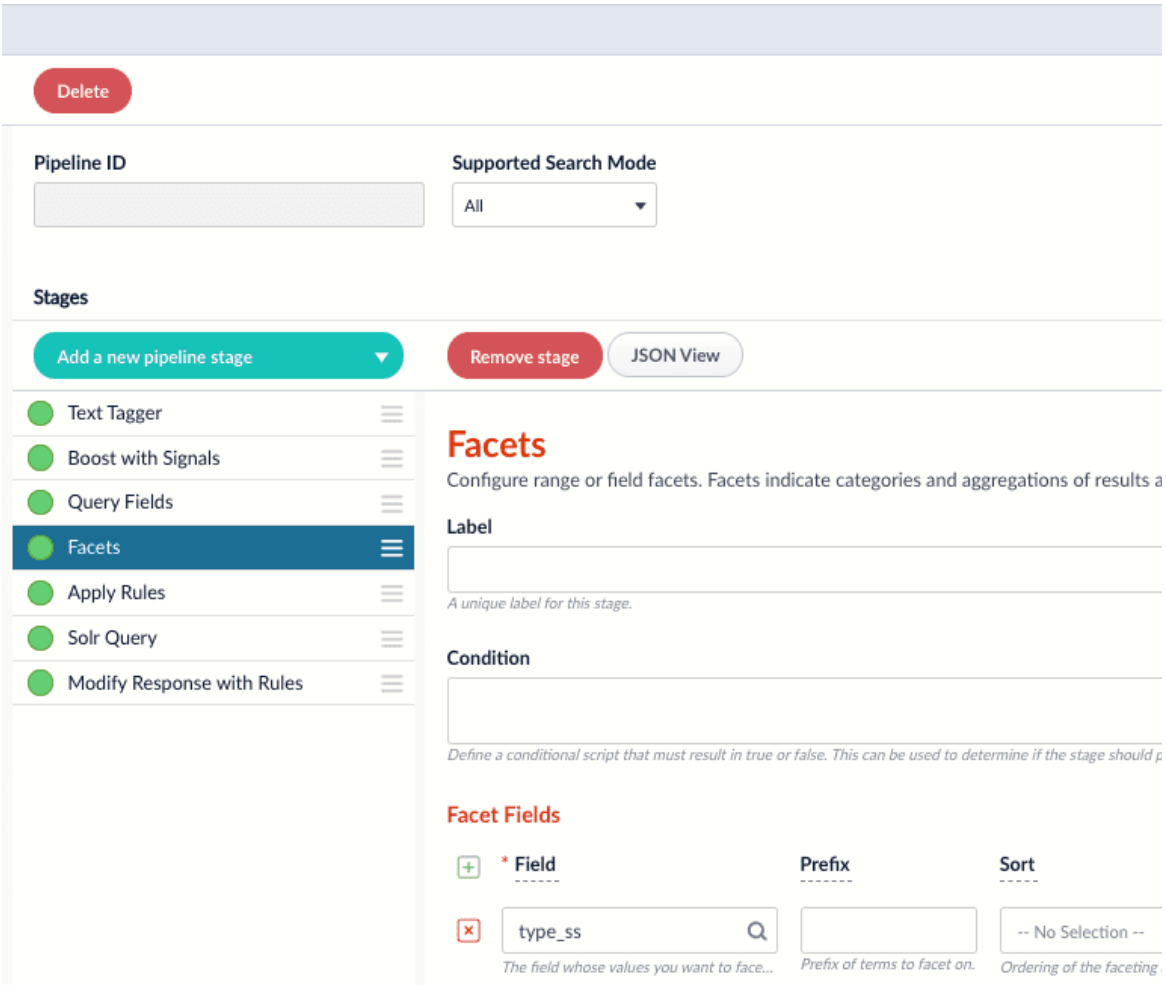
How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion*

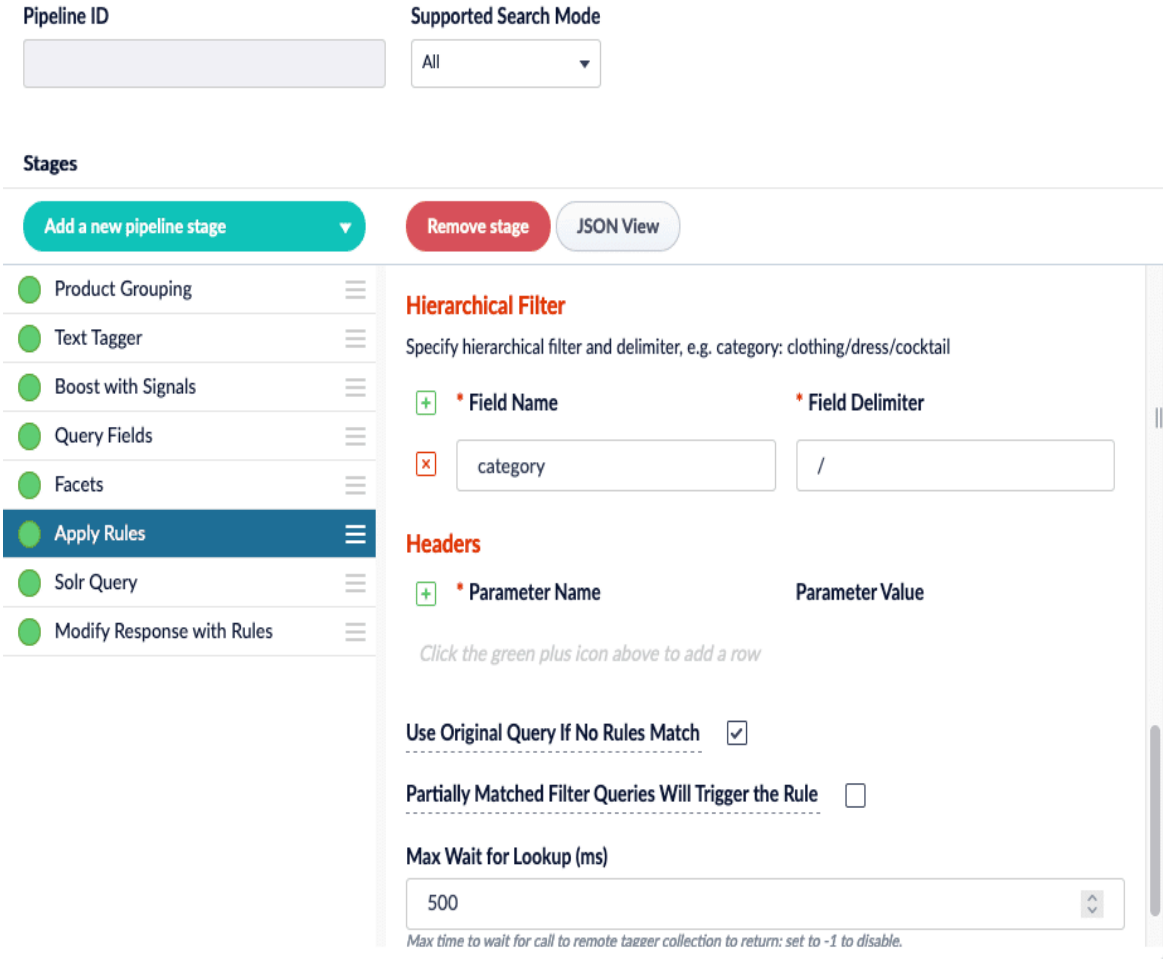
Fusion 5.5.2 and earlier

To use *hierarchical facets in Predictive Merchandiser*, do the following:

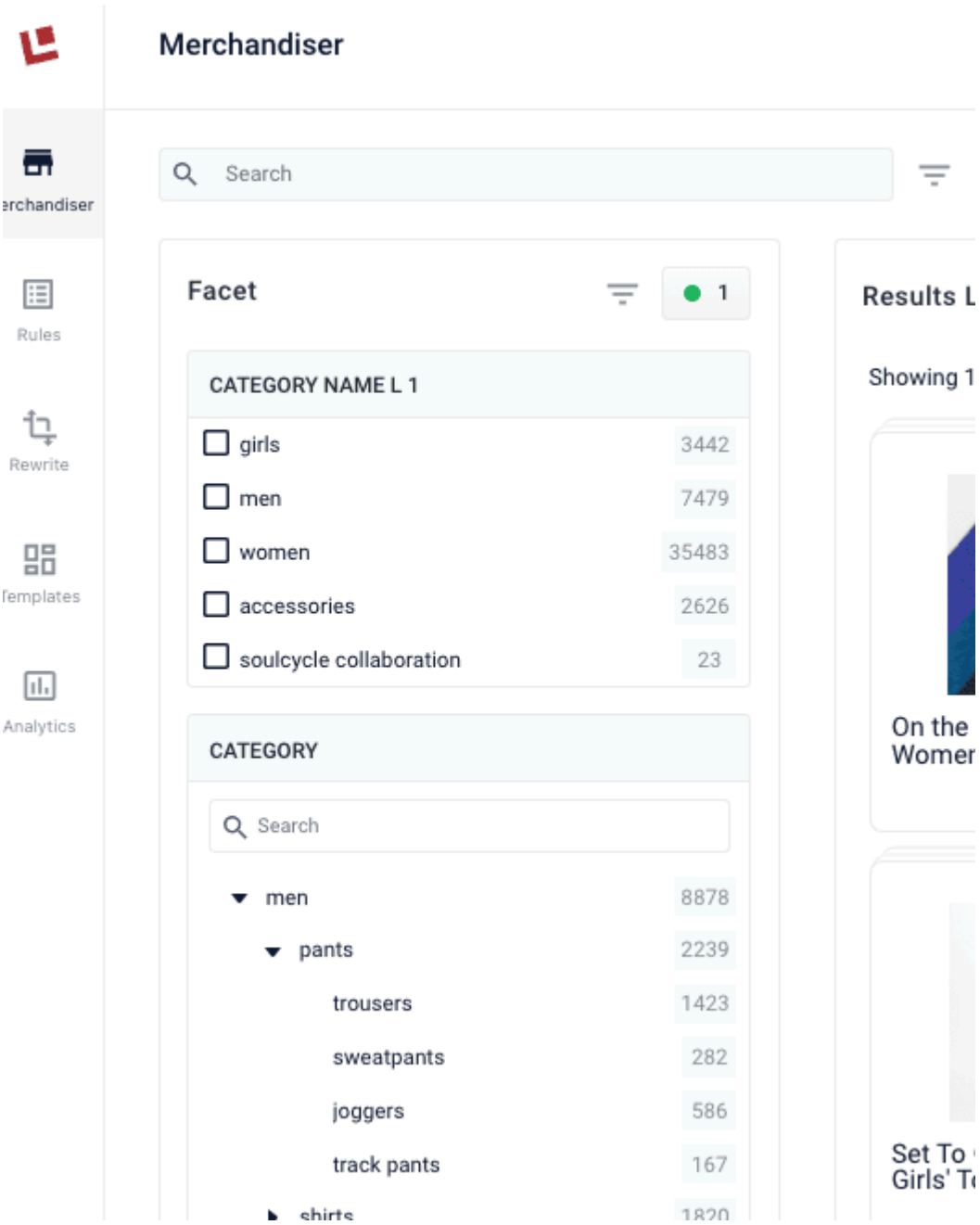
1. Select fields to facet in the *Facets* query pipeline stage.



2. In the *Apply Rules* query pipeline stage, select **Field Name** and **Field Delimiter** to use. This assumes a facet selected in Step 1 can be separated according to a delimiter, for example `/` or `|`.



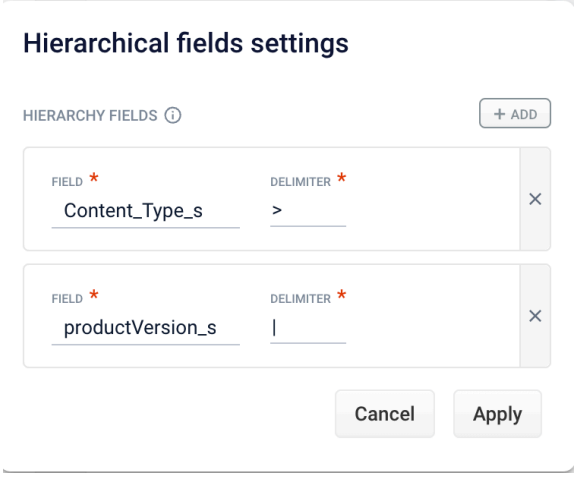
3. Predictive Merchandiser's facet zone can now display child facets in an expandable list.



Fusion 5.6 and later

To configure hierarchical fields settings:

1. Access the application and click **Relevance > Rules > Templates**.
2. Click the three vertical dots to the right of the application name to display the Hierarchical fields settings section.



3. Complete one of the following:
   - To add a new field, click **+ Add**. Enter values in the **Field** and **Delimiter** fields and click **Apply**.

     Note

     Any symbol can be used as a delimiter.

   - To edit an existing field, click **Field** and select the value to edit. Enter the appropriate value in the **Delimiter** field and click **Apply**.
   - To delete a field setting, select the value in **Field**, click **X**, and then click **Apply**.
   - To cancel any changes before you apply them, click **Cancel**.

Use Jupyter with Fusion SQL

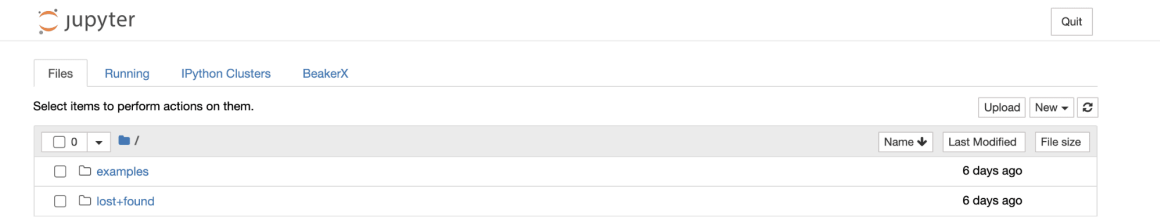
How-to

> Compatible versions: 5.5
>
> Works with: *Fusion*

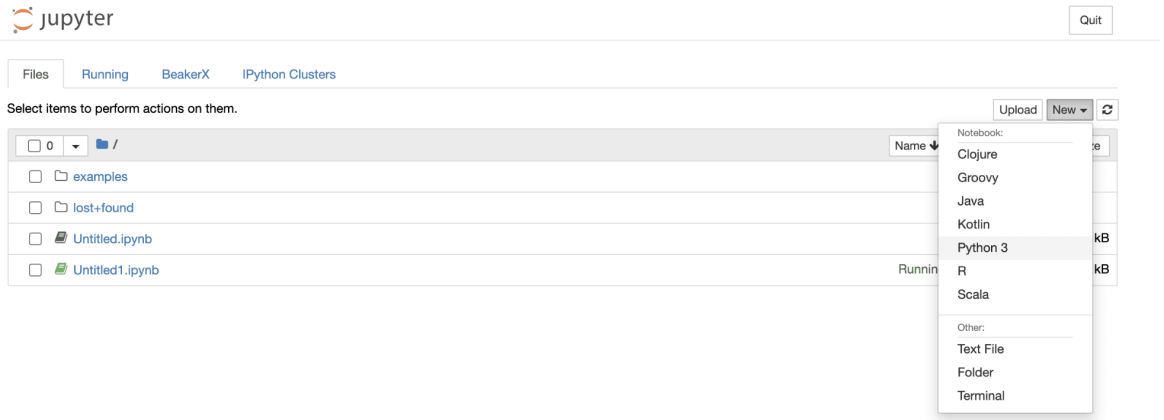
This guide provides basic setup steps for connecting to Fusion SQL in a Jupyter notebook, then shows examples of how to query and visualize result sets and statistical functions from Fusion SQL. The full scope of Fusion SQL query capabilities is covered in *Querying Fusion SQL*.

Basic setup

1. Navigate to `https://FUSION_HOST:6764/jupyter` in your browser.
   The Jupyter home page will load in your browser:



2. Create a new Python 3 notebook:



3. Connect to Fusion SQL through the Fusion proxy by issuing the following commands into the notebook:

```
In [1]: %load_ext fusionsql

In [2]: username="admin"
        hostandport="proxy:6764"

In [4]: import getpass
        password=getpass.getpass()
        ........

In [5]: %fusionsql connect
        Connecting to http://proxy:6764/api
        Auth successful for user 'admin' for host 'http://proxy:6764/api'
```

Showing and describing tables

The `fusionsql` Python extension is used to issue commands and SQL queries to Fusion SQL.

The `show tables` command can be used to list the available tables:

```
In [10]: %fusionsql show tables
Time taken for response '407.309' ms
```

Out[10]:

	tableName	isTemporary
0	car_stuff_v45a1a25a	True
1	claim_pytest_v45a1a25a	True
2	defect_pytest_v45a1a25a	True
3	iris	True
4	logs	True
5	sql_catalog	True
6	system_autocomplete	True
7	system_blobs	True
8	system_history	True
9	system_jobs_history	True
10	system_logs	True
11	testapp	True
12	vehicle_pytest_v45a1a25a	True

The `describe tablename` command can be used list the columns and data types in a table:

```
In [11]: %fusionsql describe iris
Time taken for response '354.545' ms
```

Out[11]:

	col_name	data_type
0	id	string
1	petal_length_d	double
2	petal_width_d	double
3	sepal_length_d	double
4	sepal_width_d	double
5	species_s	string

Querying with SQL

SQL is used to query the data in the tables. All SQL queries return a Pandas dataframe. If the result set is not set to a variable it is displayed in the notebook. The example below shows a SQL query and output:

```
In [12]: %fusionsql select * from iris limit 5
Time taken for response '944.652' ms
```

Out[12]:

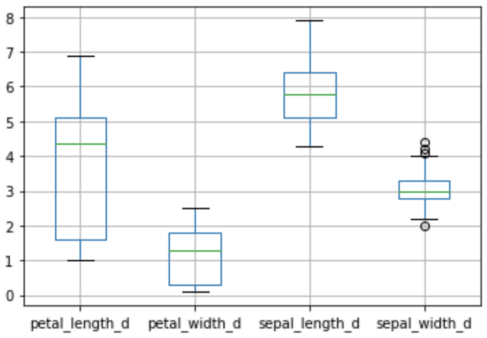
	id	petal_length_d	petal_width_d	sepal_length_d	sepal_width_d	species_s
0	iris_49	1.4	0.2	4.6	3.2	setosa
1	iris_93	4.6	1.4	6.1	3.0	versicolor
2	iris_73	4.0	1.3	6.1	2.8	versicolor
3	iris_22	1.7	0.2	5.4	3.4	setosa
4	iris_138	5.6	2.4	6.3	3.4	virginica

Sampling and visualization

Basic select queries that do not include an order by return a random sample that matches the result set. Fusion SQL has very fast and scalable sampling capabilities. Sampled result sets can be set to variables which are Pandas dataframes. The result sets can be manipulated and visualized through standard dataframe functions.

The example below issues a SQL query and sets it to the variable data, which is a Pandas dataframe. The next statement calls the `boxplot` function directly on the dataframe. The boxplot appears below. Boxplots allow for the quick visual comparison of the distributions within the columns in the dataframe.

```
In [61]: data = %fusionsql select * from iris limit 150
         data.boxplot(column=['petal_length_d', 'petal_width_d', 'sepal_length_d', 'sepal_width_d'])

         Time taken for response '884.996' ms

Out[61]: <matplotlib.axes._subplots.AxesSubplot at 0x7f7ba94b1048>
```



In the example above we're comparing the distributions of the **petal_length_d**, **petal_width_d**, **sepal_length_d** and **sepal_width_d** fields.

This result set also contains a categorical field called **species_s**. We can change the orientation of the dataframe so that we can do a boxplot of the **petal_length_d** field by the **species_s** field. The dataframe has a `pivot` function which can change the orientation of the result set so that the values in the **species_s** field become the column headers of the dataframe. Below is an example of how to use the **pivot** function:

```
In [7]: data = %fusionsql select id, petal_length_d, species_s from iris limit 150
        data.pivot(index='id', columns='species_s', values='petal_length_d')

        Time taken for response '873.18' ms
```

Out[7]:

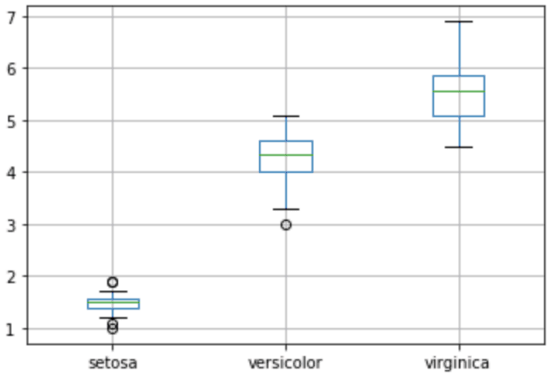
species_s	setosa	versicolor	virginica
id			
iris_10	1.4	NaN	NaN
iris_100	NaN	3.0	NaN
iris_101	NaN	4.1	NaN
iris_102	NaN	NaN	6.0
iris_103	NaN	NaN	5.1
...	...	...	...
iris_95	NaN	3.3	NaN
iris_96	NaN	4.2	NaN
iris_97	NaN	4.2	NaN
iris_98	NaN	4.2	NaN
iris_99	NaN	4.3	NaN

150 rows × 3 columns

Notice that we now have a matrix with the **species_s** field as column headers. Each row in the table only has **petal_length_d** data populated for one of the **species_s** types.

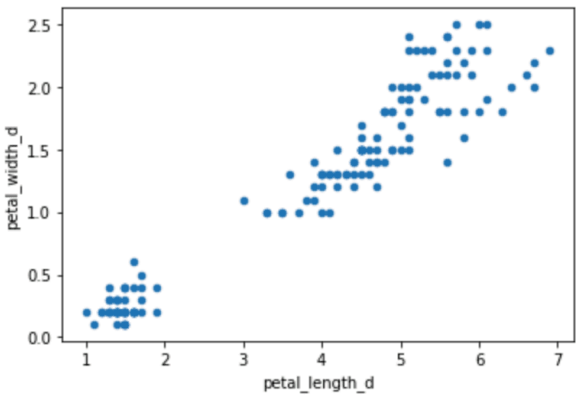
The **boxplot** function conveniently ignores the NaN values to create a boxplot for each **species_s** column.

```
In [17]: data = %fusionsql select id, petal_length_d, species_s from iris limit 150
         piv = data.pivot(index='id', columns='species_s', values='petal_length_d')
         piv.boxplot(column=['setosa', 'versicolor', 'virginica'])

         Time taken for response '874.233' ms

Out[17]: <matplotlib.axes._subplots.AxesSubplot at 0x7f692c75ecf8>
```



Samples can also be used to visualize the relationship between two numeric fields. The example below takes a sample and sets it to a dataframe. The dataframe `plot` function is then used to plot a scatter plot with **`petal_length_d`** on the x-axis and **`petal_width_d`** on the y-axis. From this plot we can quickly see how **`petal_length`** affects **`petal_width`**.

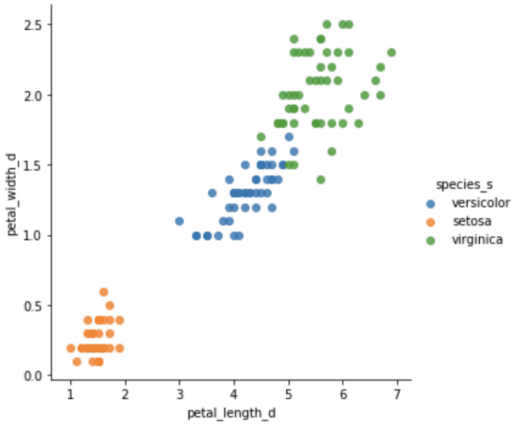
```
In [69]: data = %fusionsql select * from iris limit 150
         data.plot(kind="scatter", x='petal_length_d', y='petal_width_d')

         Time taken for response '350.43' ms

Out[69]: <matplotlib.axes._subplots.AxesSubplot at 0x7f7ba9383080>
```



We can also add the **`species_s`** label to the visualization and color the points by species. This allows us to see the relationship between the two variables by species.

In order to add colored labels we'll use the Seaborn Python package. In the example below the Seaborn package is imported and then the `lmplot` function is used to plot the labeled and colored scatter plot.

```
In [77]: import seaborn as sns

In [82]: data = %fusionsql select petal_length_d, petal_width_d, species_s from iris limit 150
         sns.lmplot( x="petal_length_d", y="petal_width_d", data=data, fit_reg=False, hue='species_s', legend=True)

         Time taken for response '937.793' ms

Out[82]: <seaborn.axisgrid.FacetGrid at 0x7f7ba746a470>
```



Descriptive statistics

Descriptive statistics for a result set can be calculated using standard SQL.

```
In [33]: %fusionsql select count(*) as cnt, avg(response_d) as avg, sum(response_d) sum, min(response_d) as min, max(response_d) as max from logs

         Time taken for response '998.744' ms
```

Out[33]:

	cnt	avg	sum	min	max
0	100000	855.068318	8.550683e+07	672.325726	1034.619513

Aggregations and visualizations

Single and multi-dimensional aggregations can be calculated using `GROUP BY SQL` queries.

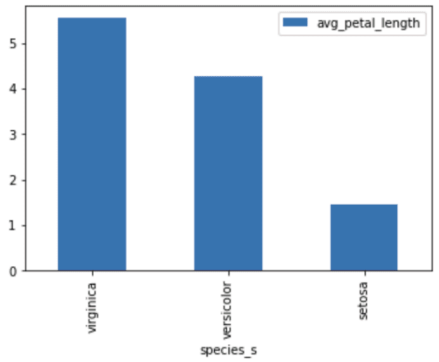
```
In [36]: %fusionsql select species_s, avg(petal_length_d) as avg_petal_length from iris group by species_s

         Time taken for response '103.883' ms
```

Out[36]:

	species_s	avg_petal_length
0	virginica	5.552
1	versicolor	4.260
2	setosa	1.464

The result of aggregations can be set to a Pandas dataframe and visualized:

```
In [37]: data = %fusionsql select species_s, avg(petal_length_d) as avg_petal_length from iris group by species_s
         Time taken for response '818.187' ms

In [38]: data.plot(kind="bar", x="species_s", y="avg_petal_length")
Out[38]: <matplotlib.axes._subplots.AxesSubplot at 0x7f7bb1ada390>
```



Time series and visualization

Fusion SQL has a powerful time series extension implemented through the SQL `date_format` *function*. The **`date_format`** function allows for efficient grouping over datetime fields. The **`date_format`** function also allows for the specification of the output format and the gap.

The example below shows the output of a **`GROUP BY`** query over the **`date_format`** function.

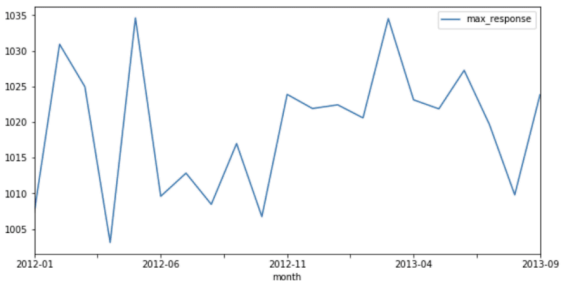
```
In [34]: %fusionsql select date_format(tdate_dt, 'yyyy-MM') as month, max(response_d) as max_response from logs where tdate_dt > '2012-01-01' group by month
         Time taken for response '506.116' ms
```

Out[34]:

	month	max_response
0	2012-01	1007.346532
1	2012-02	1030.914944
2	2012-03	1024.965294
3	2012-04	1003.101608
4	2012-05	1034.619513
5	2012-06	1009.586484
6	2012-07	1012.834901
7	2012-08	1008.454979
8	2012-09	1016.984867
9	2012-10	1006.735295
10	2012-11	1023.884146
11	2012-12	1021.905858
12	2013-01	1022.417567
13	2013-02	1020.586021
14	2013-03	1034.520365
15	2013-04	1023.117141
16	2013-05	1021.855578
17	2013-06	1027.258175
18	2013-07	1019.655087
19	2013-08	1009.789087
20	2013-09	1023.837521

The example below shows the time series result set plotted on a line chart using the dataframe **`plot`** function.

```
In [28]: data = %fusionsql select date_format(tdate_dt, 'yyyy-MM') as month, max(response_d) as max_response from logs where tdate_dt > '2012-01-01' group by month
         data.plot(kind="line", x='month', y='max_response', figsize=(10,5))

         Time taken for response '655.563' ms

Out[28]: <matplotlib.axes._subplots.AxesSubplot at 0x7febec211710>
```



Statistical analysis and visualizations

Fusion SQL includes a library of functions designed for statistical analysis and visualization. The section below demonstrates a subset of these functions within Jupyter notebook.

Histograms

Histograms can be created from a numeric field using the `hist` *function*. The **`hist`** function takes three parameters:

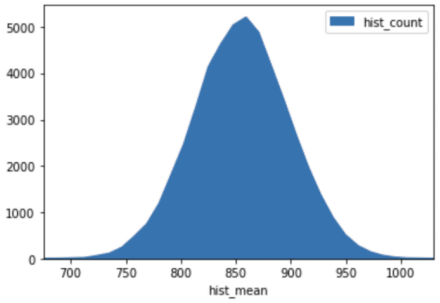
- The numeric field to create the histogram from
- The number of bins in the histogram
- The sample size

The hist function returns the mean of each bin. There are three other fields available for selection when the **`hist`** function is used:

- **`hist_count`** : The number of samples within each bin.
- **`hist_prob`** : The probability that sample will fall within each bin.
- **`hist_cum_prob`** : The cumulative probability for each bin.

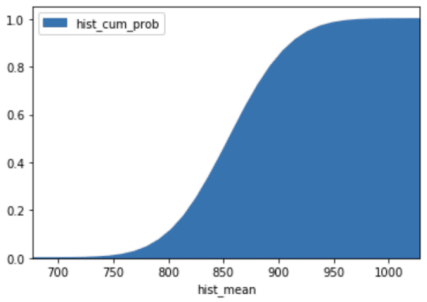
Below is an example of the **`hist`** function visualized in an area chart with the **`hist_mean`** on the x-axis and the **`hist_count`** on the y-axis:

```
In [48]: data = %fusionsql select hist(response_d, 32, 50000) as hist_mean, hist_count, hist_prob, hist_cum_prob from logs
         data.plot(kind='area',x='hist_mean',y='hist_count')

         Time taken for response '906.846' ms

Out[48]: <matplotlib.axes._subplots.AxesSubplot at 0x7f7ba96406d8>
```



Below is an example with **`hist_cum_prob`** plotted on the y-axis:

```
In [47]: data = %fusionsql select hist(response_d, 32, 50000) as hist_mean, hist_count, hist_prob, hist_cum_prob from logs
         data.plot(kind='area',x='hist_mean',y='hist_cum_prob')

         Time taken for response '83.437' ms

Out[47]: <matplotlib.axes._subplots.AxesSubplot at 0x7f7ba96df278>
```



Fitting a gaussian curve to a histogram

The `gaussfit` *function* fits a gaussian curve to a histogram. You would do this to visualize how well the data in a numeric field fits a normal distribution.

The **`gaussfit`** function takes two parameters:

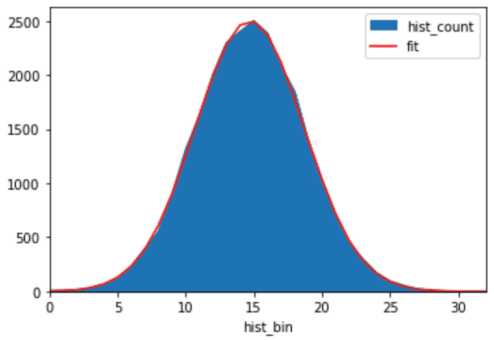
- The numeric field to create the histogram from
- The sample size

The **`gaussfit`** function returns the fitted curve at each bin in the histogram. There are two additional fields that can be selected when the **`gaussfit`** function is used:

- **`hist_bin`** : The bin number for the histogram
- **`hist_count`** : The count of the number of samples for each bin

The example below shows the histogram and fitted curve plotted on the same figure. The example demonstrates how to run the **`gaussfit`** function and how to overlay the fitted curve on top of the histogram.

```
In [50]: import matplotlib.pyplot as plt

In [51]: data = %fusionsql select gaussfit(response_d, 25000) as fit, hist_bin, hist_count from testapp
         fig, ax = plt.subplots(1, 1)
         data.plot(kind='area',x='hist_bin',y='hist_count', ax=ax)
         data.plot(kind='line',x='hist_bin',y='fit', color='red', ax=ax)
         plt.show()

         Time taken for response '991.1' ms
```



Percentiles

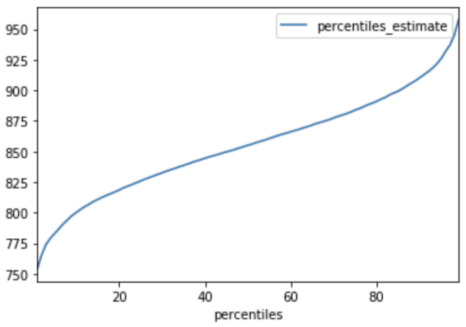
The `percentiles` *function* calculates percentile estimates for a numeric field. The **`percentiles`** function takes two parameters:

- The numeric field to calculate the percentiles from
- The sample size

The `percentiles` function returns the percentile (1-99). The `percentiles_estimate` field returns the estimated percentile value for each percentile.

Below is an example of the `percentiles` function plotted on a line chart:

```
In [54]: data = %fusionsql select percentiles(response_d, 10000) as percentiles, percentiles_estimate from logs
         data.plot(kind='line',x='percentiles',y='percentiles_estimate')

         Time taken for response '993.528' ms

Out[54]: <matplotlib.axes._subplots.AxesSubplot at 0x7f7ba9518be0>
```



Correlation matrices

Correlation matrices can be computed using the `corr_matrix` *function*. The `corr_matrix` function takes two parameters:

- A single quoted string containing a comma delimited list of fields to build the correlation matrix from
- The sample size

The `corr_matrix` function returns the Pearson's correlation coefficient for each combination of the fields. The `matrix_x` and `matrix_y` fields contain the x and y labels of the matrix.

The example below shows the output of the `corr_matrix` function:

```
In [21]: %fusionsql select corr_matrix('petal_width_d, petal_length_d, sepal_length_d, sepal_width_d', 150) as corr, matrix_x, matrix_y from iris

         Time taken for response '43.448' ms

Out[21]:
```

	corr	matrix_x	matrix_y
0	1.00000	petal_width_d	petal_width_d
1	0.96276	petal_length_d	petal_width_d
2	0.81795	sepal_length_d	petal_width_d
3	-0.35654	sepal_width_d	petal_width_d
4	0.96276	petal_width_d	petal_length_d
5	1.00000	petal_length_d	petal_length_d
6	0.87175	sepal_length_d	petal_length_d
7	-0.42052	sepal_width_d	petal_length_d
8	0.81795	petal_width_d	sepal_length_d
9	0.87175	petal_length_d	sepal_length_d
10	1.00000	sepal_length_d	sepal_length_d
11	-0.10937	sepal_width_d	sepal_length_d
12	-0.35654	petal_width_d	sepal_width_d
13	-0.42052	petal_length_d	sepal_width_d
14	-0.10937	sepal_length_d	sepal_width_d
15	1.00000	sepal_width_d	sepal_width_d

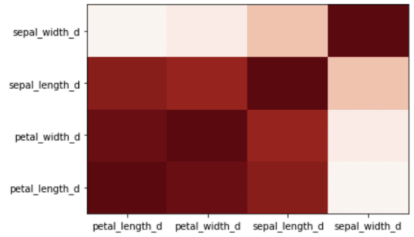
The dataframe created by the `corr_matrix` function can be pivoted into a matrix using the `pivot` function:

```
In [22]: data = %fusionsql select corr_matrix('petal_width_d, petal_length_d, sepal_length_d, sepal_width_d', 150) as corr, matrix_x, matrix_y from iris
         data.pivot(index='matrix_x', columns='matrix_y', values='corr')

         Time taken for response '727.756' ms

Out[22]:
```

matrix_x \ matrix_y	petal_length_d	petal_width_d	sepal_length_d	sepal_width_d
petal_length_d	1.00000	0.96276	0.87175	-0.42052
petal_width_d	0.96276	1.00000	0.81795	-0.35654
sepal_length_d	0.87175	0.81795	1.00000	-0.10937
sepal_width_d	-0.42052	-0.35654	-0.10937	1.00000

Once pivoted, the Matplotlib `pcolor` function can be used to display the correlation matrix as a heat map. Below is an example of how to visualize the **`corr_matrix`** function:

```
In [10]: import numpy as np

In [9]: data = %fusionsql select corr_matrix('petal_width_d, petal_length_d, sepal_length_d, sepal_width_d', 150) as corr, matrix_x, matrix_y from iris
         piv = data.pivot(index='matrix_x', columns='matrix_y', values='corr')
         plt.pcolor(piv, cmap=plt.cm.Reds)
         plt.yticks(np.arange(0.5, len(piv.index), 1), piv.index)
         plt.xticks(np.arange(0.5, len(piv.columns), 1), piv.columns)
         plt.show()

         Time taken for response '643.939' ms
```



Linear regression and residual plots

The `linearfit` *function* can be used to visualize bi-variate linear regression plots and residual plots. The **`linearfit`** function takes two parameters:

- The numeric predictor or independent variable field (x).
- The numeric dependent variable field (y).

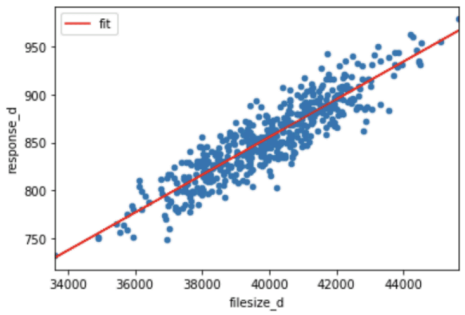
The **`linearfit`** function returns the predicted value for each of the predictor values. There are also three other fields which can be selected in the result set:

- The numeric predictor or independent variable field
- The numeric dependent variable field
- residual : The difference between the actual value and the predicted value. This represents the error for each prediction.

Using predicted values and the residual values the regression and residual plots can be visualized.

The example below shows the regression plot. The regression plot is drawn by first plotting the dependent and independent variables in scatter plot. Then the predicted values are overlaid through the scatter to visualize how well the line fits through the points.

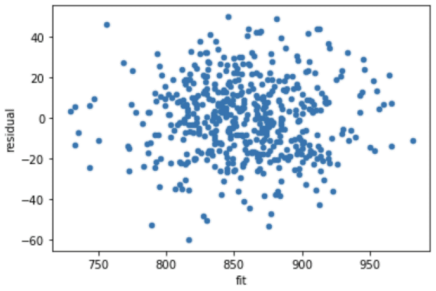
```
In [70]: lfit = %fusionsql select linearfit(filesize_d, response_d) as fit, filesize_d, response_d from logs limit 500
         fig, ax = plt.subplots(1, 1)
         lfit.plot(kind='scatter', x='filesize_d',y='response_d', ax=ax)
         lfit.plot(kind='line', x='filesize_d',y='fit', color='red', ax=ax)
         plt.show()

         Time taken for response '318.941' ms
```



The residual plot is done by plotting the predictions on the x-axis and the residual values on the y-axis. The residual plot visualizes the error across the full range of predictions.

```
In [74]: rdata = %fusionsql select linearfit(filesize_d, response_d) as fit, residual, filesize_d, eresponse_d from logs limit 500
         rdata.plot(kind='scatter',x='fit',y='residual')
         plt.show()
```

```
Time taken for response '399.673' ms
```



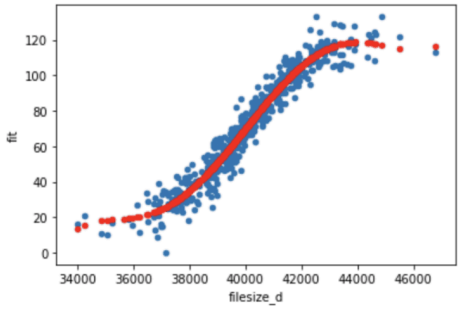
Polynomial non-linear regression

The `polyfit` *function* fits a smooth curve through a bi-variate scatter plot. The **`polyfit`** function works similarly to the **`linearfit`** function except it has one extra parameter which specifies the degree of the polynomial used to fit the curve. The higher the degree, the larger the number of curves. The degree is typically an integer between 1 and 5.

Below is an example of a regression plot using the **`polyfit`** function with a 5 degree polynomial. The only difference in the plotting technique from **`linearfit`** is that the predictions are plotted with a scatter plot. See **`linearfit`** for more details on regression plots.

```
In [71]: pfit = %fusionsql select polyfit(filesize_d, eresponse_d, 5) as fit, filesize_d, eresponse_d from logs limit 500
         fig, ax = plt.subplots(1, 1)
         pfit.plot(kind='scatter',x='filesize_d',y='eresponse_d', ax=ax)
         pfit.plot(kind='scatter',x='filesize_d',y='fit', color='red', ax=ax)
         plt.show()
```

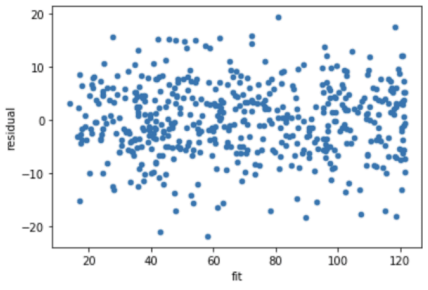
```
Time taken for response '596.786' ms
```



The residual plot is handled in exactly the same manner as with the **`linearfit`** residual plot.

```
In [72]: rdata = %fusionsql select polyfit(filesize_d, eresponse_d, 5) as fit, residual, filesize_d, eresponse_d from logs limit 500
         rdata.plot(kind='scatter',x='fit',y='residual')
         plt.show()
```

```
Time taken for response '537.67' ms
```



External Access

If you wish to access Fusion SQL features from an external Jupyter notebook, you can use this `gist` to create an access object for the service, which can be used in a similar way to the internal notebook:

```
fsql = FusionSQL("https://hostname:port/api", "username", password)
fsql.run_sql_query("select * from sql_catalog")
```

This will return a Pandas dataframe containing the results of the query.

Use Kerberos for JDBC Authentication

How-to

> Compatible versions: 4.2
>
> Works with: *Fusion Server*

Use the following steps to configure the Fusion SQL service to use Kerberos for authentication.

1. Create a service principal and keytab; your Active Directory or Kerberos administrator will know how to do this. At a minimum, enable the AES 128-bit encryption. You can use 256, but you will have to install the JCE extensions.
   This is an example command to create a keytab file for the service account:

```
ktpass /out c:\fusion.service.keytab /princ
fusion/sawsserver@FUSIONSQL.LOCAL /rndpass /ptype
KRB5_NT_PRINCIPAL /mapUser fusion@FUSIONSQL.LOCAL –mapOp set –
crypto AES128–SHA1
```

2. Copy the keytab file to the Fusion `conf` directory.
3. Update the file `conf/hive–site.xml` (on Unix) or `conf\hive–site.xml` (on Windows) to use Kerberos authentication and the correct principal and keytab file installed in step 2.
   On Unix:

```
<property> <name>hive.server2.authentication</name>
<value>Kerberos</value> </property> <property>
<name>hive.server2.authentication.Kerberos.principal</name>
<value>fusion/sawsserver@FUSIONSQL.LOCAL</value> </property>
<property>
<name>hive.server2.authentication.Kerberos.keytab</name>
<value>./conf/fusion.service.keytab</value> </property>
```

   On Windows:

```
<property> <name>hive.server2.authentication</name>
<value>Kerberos</value> </property> <property>
<name>hive.server2.authentication.Kerberos.principal</name>
<value>fusion/sawsserver@FUSIONSQL.LOCAL</value> </property>
<property>
<name>hive.server2.authentication.Kerberos.keytab</name>
<value>conf\fusion.service.keytab</value> </property>
```

4. Install the file that contains information about your Kerberos realm on the Fusion server.
   On Unix:
   Place the file `krb5.conf` in the `etc` directory.
   On Windows:
   Place the file `krb5.ini` in the `C:\Windows` directory.
5. Update the file `conf/fusion.cors` (`fusion.properties` in Fusion 4.x) (on Unix) or `conf\fusion.cors` (`fusion.properties` in Fusion 4.x) (on Windows) to point to the file `krb5.conf` (on Windows) or `krb5.ini` (on Windows) installed in step 4.
   On Unix:

```
sql.jvmOptions = -Xmx1g -
Djava.security.krb5.conf=/etc/krb5.conf
```

   On Windows:

```
sql.jvmOptions = -Xmx1g -
Djava.security.krb5.conf=C:\Windows\krb5.ini
```

Use metadata tags to refine search results

How-to

> Works with: Connected Search

Web data source metadata tags can be ingested as fields. The examples here show how metadata tags can be used to refine search results.

Facet with metadata tags

Using the browse endpoint, this example has a music lessons website send a request to see all pages with a count of the metadata tags for faceting. The request contains field names for the instrument type and player competency levels. The response includes a count for each of the facet values found.

Use the `facets` request parameter and pass the `page.metatags.METADATA_NAME` to get back the counts of metadata tags for faceting. This example includes `fieldList` to simplify the results to only show the specified metadata tag fields.

Request

```
{ "sessionId": "864782f0-af36-4dee-8430-9e73d6006eaa", "utcOffset":
"-05:00", "fieldList":
["page.metatags.instrument","page.metatags.difficulty"], "facets":
["page.metatags.instrument","page.metatags.difficulty"] }
```

Response

```
{ "queryId": "fd110486-f168-47c0-a419-1518a4840589", "hits": 43,
"docs": [ { "type": "page", "id": "441eb3be-7de6-470a-8141-
e416a15c7db1-6a092bd4-5098-466c-94aa-40bf68294303-
687088076571483713333558199345704397 31", "rank": 1,
"datasourceLabels": [], "fields": { "page.metatags.instrument":
"piano", "page.metatags.difficulty": "beginner" }}, { "type": "page",
"id": "441eb3be-7de6-470a-8141-e416a15c7db1-6a092bd4-5098-466c-94aa-
40bf68294303-687088076571483713333558199345704397 31", "rank": 1,
"datasourceLabels": [], "fields": { "page.metatags.instrument":
"guitar", "page.metatags.difficulty": "advanced" }}, ... ], "facets":
{ "field": { "page.metatags.difficulty": { "missing": 5, "counts": [ {
"name": "beginner", "count": 20 }, { "name": "advanced", "count": 18
}]}, "page.metatags.instrument": { "missing": 96, "counts": [ {
"name": "guitar", "count": 26 }, { "name": "piano", "count": 12 }]}
}}}
```

Filter with metadata tags

When you know the facets, you can use them to apply a filter to display selected results. A reader for a music lessons website only wants to see advanced music lessons. Using the browse endpoint, this example has a music lessons website send a request to see all advanced lessons for piano.

Use the `filters` request parameter and pass the `page.metatags.METADATA_NAME` and values to filter the search results. This example includes `fieldList` to simplify the results to only show the specified metadata tag fields.

Request

```
{ "sessionId": "864782f0-af36-4dee-8430-9e73d6006eaa", "utcOffset":
"-05:00", "fieldList":
["page.metatags.instrument","page.metatags.difficulty"], "facets":
["page.metatags.instrument","page.metatags.difficulty"], "filters": [
{"field": "page.metatags.difficulty", "values": ["advanced"]},
{"field": "page.metatags.instrument", "values": ["piano"]} ], "debug":
true }
```

Response

```
{ "queryId": "{placeholderQueryId}", "hits": 4, "docs": [ { "type":
"page", "id": "{placeholderDocumentId}", "rank": 1,
"datasourceLabels": [], "fields": { "page.metatags.instrument": "
piano", "page.metatags.difficulty": " advanced"} }, { "type": "page",
"id": "{placeholderDocumentId}", "rank": 1, "datasourceLabels": [],
"fields": { "page.metatags.instrument": " piano",
"page.metatags.difficulty": " advanced"} }, { "type": "page", "id": "
{placeholderDocumentId}", "rank": 1, "datasourceLabels": [], "fields":
{ "page.metatags.instrument": " piano", "page.metatags.difficulty": "
advanced"} }, { "type": "page", "id": "{placeholderDocumentId}",
"rank": 1, "datasourceLabels": [], "fields": {
"page.metatags.instrument": " piano", "page.metatags.difficulty": "
advanced"} } ]}
```

Use Misspelling Detection

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

LucidAcademy

Lucidworks offers free training to help you get started.

The **Course** for **Query Analytics** focuses on how Fusion provides query analytics to detect and improve underperforming queries:

Visit the LucidAcademy to see the full training catalog.

Reviewing auto-generated spelling corrections

Spelling corrections that are automatically generated by the *Token and Phrase Spell Correction job* are assigned one of the following status values:

- **Auto**
  There are three values for confidence level:

Value	Confidence	Label
0	low confidence	Pending
0.5	median confidence	Auto
1	high confidence	Auto

Note

No action is required on these results, but you can edit them if you wish.

- **Pending**
  The confidence level is ambiguous, and the result must be reviewed by a user before it can be deployed. It will only be moved from the `_query_rewrite_staging` collection to the `_query_rewrite` collection when its status has changed to "Approved" *and* it has been published.

How to review a pending spelling correction result

1. Navigate to **Relevance** > **Rules** > **Rewrite**.

2. Select **Misspelling** tab. The application displays the **Misspelling Detection** screen.

> **Tip**
>
> Notice the **Status** facet on the left. Click **Pending** to view only the items that need review.

3. Click the icon next to the spelling correction.

4. In the **Status** column, select either "Approved" or "Denied".
   Optionally, you can also edit the spelling correction itself.

Adding new spelling corrections

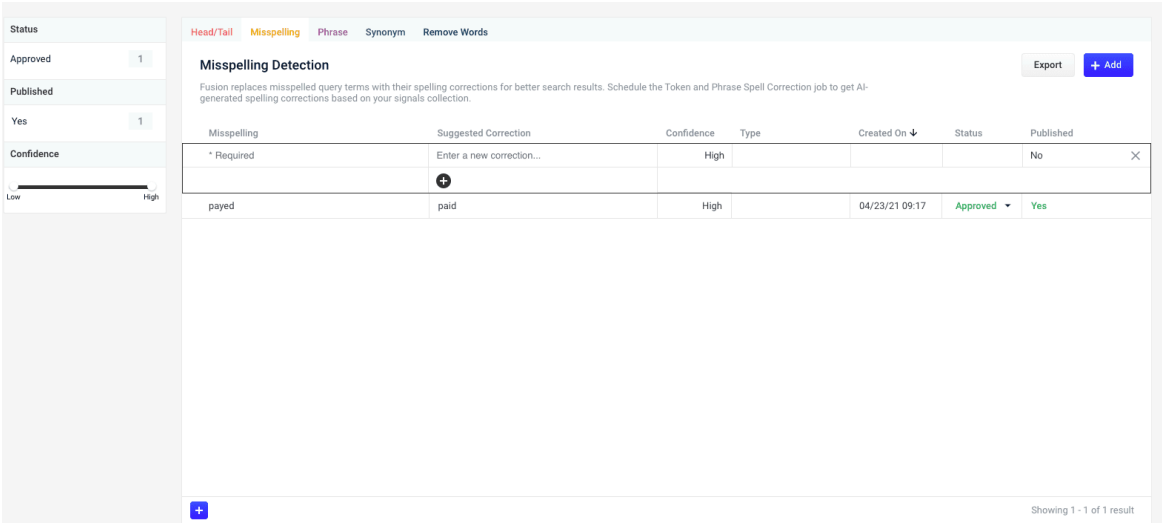
You can manually add spelling corrections in addition to any generated by the Fusion 5.x.x *Token and Phrase Spell Correction job* release.

How to add a spelling correction

1. Navigate to **Relevance** > **Rules** > **Rewrite**.
2. Select **Misspelling** tab. The application displays the **Misspelling Detection** screen.

3. At the bottom of the rules list, click the icon.

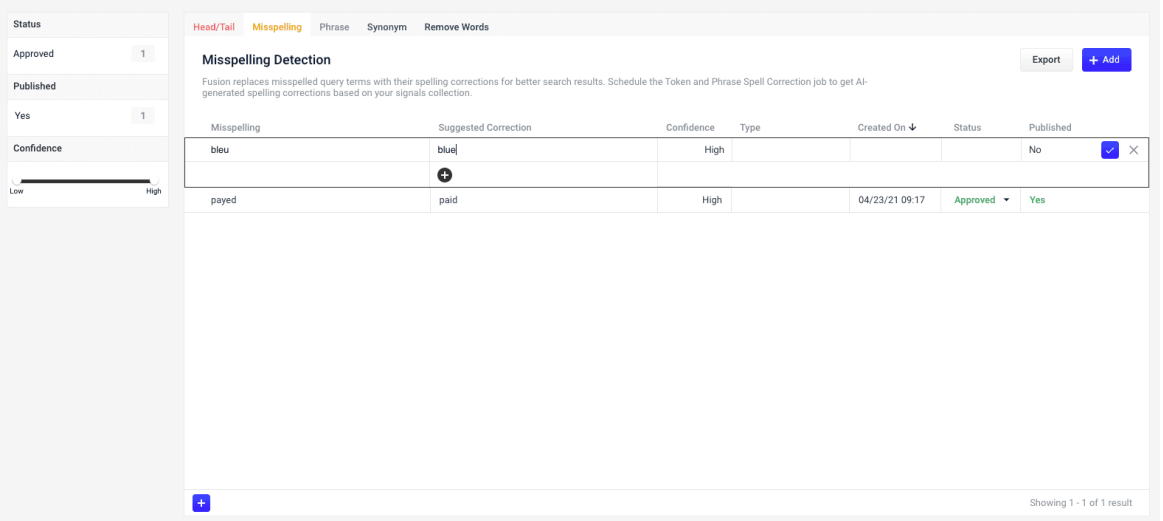
   A new spelling correction appears at the top of the list:



4. Enter the misspelled word or phrase.
5. Enter one or more spelling corrections.

> **Tip**
>
> It is not necessary to set a confidence value.

6. Select the spelling correction's status, depending on whether you want to deploy it the next time you publish your changes ("Approved") or save it for further review ("Pending").
7. Click the check mark to save the new spelling correction:



Publishing your changes

How to publish updated spelling corrections

1. In the **Misspelling Detection** screen, click the **PUBLISH** button.
   Fusion prompts you to confirm that you want to publish your changes.
2. Click **PUBLISH**.

> **Tip**
>
> You can un-publish a query rewrite by changing its status to "denied", then clicking **PUBLISH**.

Tuning the misspelling detection job

The default configuration for the *Token and Phrase Spell Correction job*, respectively, is designed for high accuracy and works well with most signal datasets, depending on the volume and quality of the signals. If you are seeing too few results, or too many inaccurate results, then you can try tuning the job to achieve better results.

To modify job configurations, you must be a Fusion user with one of the following roles or permissions that include access to job configurations:

- *admin*
- *developer*
- *permissions*

Query rewrite jobs post-processing cleanup

To perform more extensive cleanup of query rewrites, complete the procedures in *Query Rewrite Jobs Post-processing Cleanup*.

Use Misspelling Detection

How-to

> Compatible versions: 4.2
>
> Works with: *Fusion AI*, *Fusion Server*

LucidAcademy

Lucidworks offers free training to help you get started.

The **Course** for **Query Analytics** focuses on how Fusion provides query analytics to detect and improve underperforming queries:

Visit the LucidAcademy to see the full training catalog.

Reviewing auto-generated spelling corrections

Spelling corrections that are automatically generated by the *Token and Phrase Spell Correction job* are assigned one of the following status values:

- **Auto**
  There are three values for confidence level:

Value	Confidence	Label
0	low confidence	Pending
0.5	median confidence	Auto
1	high confidence	Auto

Note

No action is required on these results, but you can edit them if you wish.

- **Pending**
  The confidence level is ambiguous, and the result must be reviewed by a user before it can be deployed. It will only be moved from the `_query_rewrite_staging` collection to the `_query_rewrite` collection when its status has changed to "Approved" *and* it has been published.

How to review a pending spelling correction result

1. Navigate to **Relevance** > **Rules**. The application displays the **Query Rewriting** screen.

2. Select **Misspelling Detection**. The application displays the **Misspelling Detection** screen.

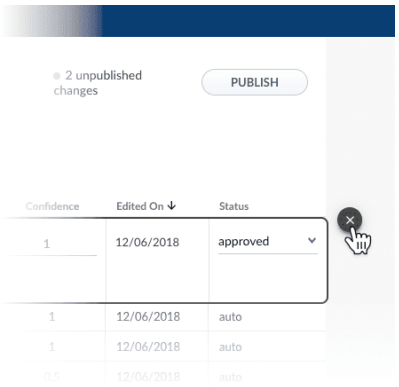
> **Tip**
>
> Notice the **Status** facet on the left. Click **Pending** to view only the items that need review.

3. Click the icon next to the spelling correction.

4. In the **Status** column, select either "Approved" or "Denied".
Optionally, you can also edit the spelling correction itself.

> **Tip**
>
> Although the Confidence field is also editable, changing its value makes no difference.

5. Click the **Close** icon next to the updated spelling correction:



> **Note**
>
> Approving a spelling correction does not automatically deploy it to the `_query_rewrite` collection. When you have finished your review, you must click **Publish** to deploy your changes.

Adding new spelling corrections

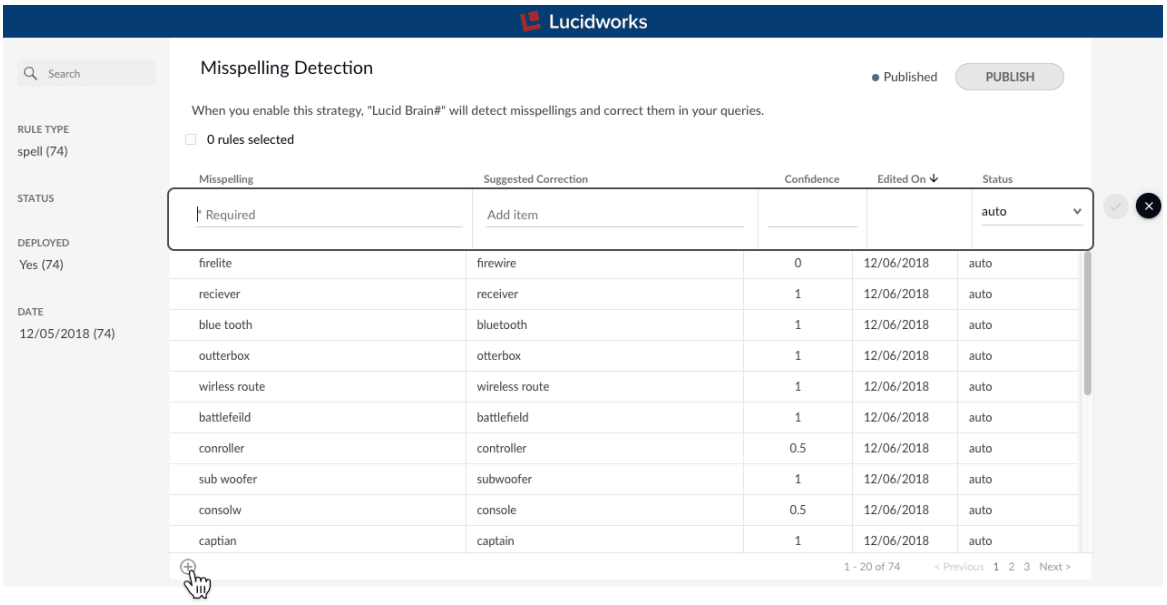
You can manually add spelling corrections in addition to any generated by the *Token and Phrase Spell Correction job*.

How to add a spelling correction

1. Navigate to **Relevance** > **Rules**. The application displays the **Query Rewriting** screen.
2. Select **Misspelling Detection**. The application displays the **Misspelling Detection** screen.

3. At the bottom of the rules list, click the icon.

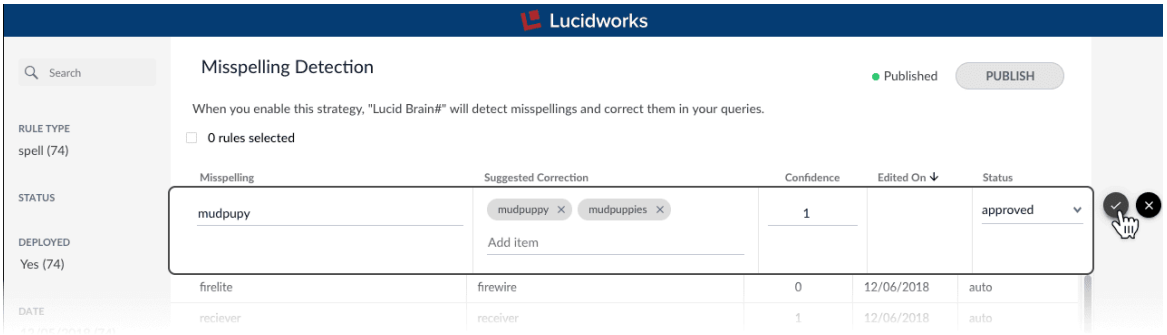
A new spelling correction appears at the top of the list:



4. Enter the misspelled word or phrase.
5. Enter one or more spelling corrections.

> **Tip**
>
> It is not necessary to set a confidence value.

6. Select the spelling correction's status, depending on whether you want to deploy it the next time you publish your changes ("Approved") or save it for further review ("Pending").
7. Click the check mark to save the new spelling correction:



Publishing your changes

How to publish updated spelling corrections

1. In the **Misspelling Detection** screen, click the **PUBLISH** button.
   Fusion prompts you to confirm that you want to publish your changes.
2. Click **PUBLISH**.

> **Tip**
>
> You can un-publish a query rewrite by changing its status to "denied", then clicking **PUBLISH**.

Tuning the misspelling detection job

The default configuration for the *Token and Phrase Spell Correction job* is designed for high accuracy and works well with most signal datasets, depending on the volume and quality of the signals. If you are seeing too few results, or too many inaccurate results, then you can try tuning the job to achieve better results.

To modify job configurations, you must be a Fusion user with one of the following roles or permissions that include access to job configurations:

- *admin*
- *developer*
- *permissions*

Query rewrite jobs post-processing cleanup

To perform more extensive cleanup of query rewrites, complete the procedures in *Query Rewrite Jobs Post-processing Cleanup*.

Use Misspelling Detection

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion*

LucidAcademy

Lucidworks offers free training to help you get started.

The **Course** for **Query Analytics** focuses on how Fusion provides query analytics to detect and improve underperforming queries:

Visit the LucidAcademy to see the full training catalog.

Reviewing auto-generated spelling corrections

Spelling corrections that are automatically generated by the *Token and Phrase Spell Correction job* are assigned one of the following status values:

- **Auto**
  There are three values for confidence level:

Value	Confidence	Label
0	low confidence	Pending
0.5	median confidence	Auto
1	high confidence	Auto

Note

No action is required on these results, but you can edit them if you wish.

- **Pending**
  The confidence level is ambiguous, and the result must be reviewed by a user before it can be deployed. It will only be moved from the `_query_rewrite_staging` collection to the `_query_rewrite` collection when its status has changed to "Approved" *and* it has been published.

How to review a pending spelling correction result

1. Navigate to **Relevance** > **Rules** > **Rewrite**.

2. Select **Misspelling** tab. The application displays the **Misspelling Detection** screen.

> **Tip**
>
> Notice the **Status** facet on the left. Click **Pending** to view only the items that need review.

3. Click the icon next to the spelling correction.

4. In the **Status** column, select either "Approved" or "Denied".
   Optionally, you can also edit the spelling correction itself.

Adding new spelling corrections

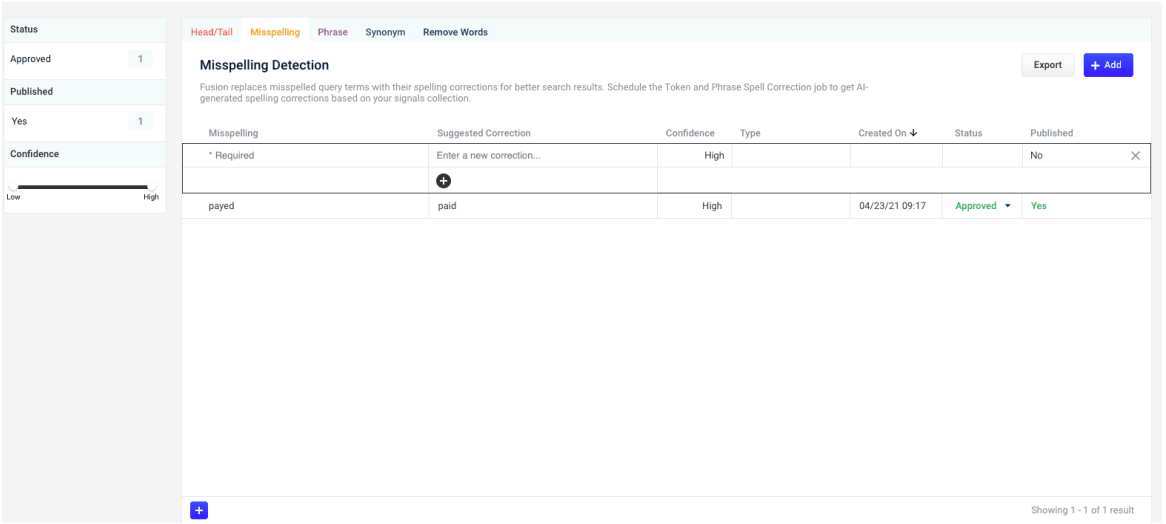
You can manually add spelling corrections in addition to any generated by the Fusion 5.x.x *Token and Phrase Spell Correction job* release.

How to add a spelling correction

1. Navigate to **Relevance** > **Rules** > **Rewrite**.
2. Select **Misspelling** tab. The application displays the **Misspelling Detection** screen.

3. At the bottom of the rules list, click the icon.

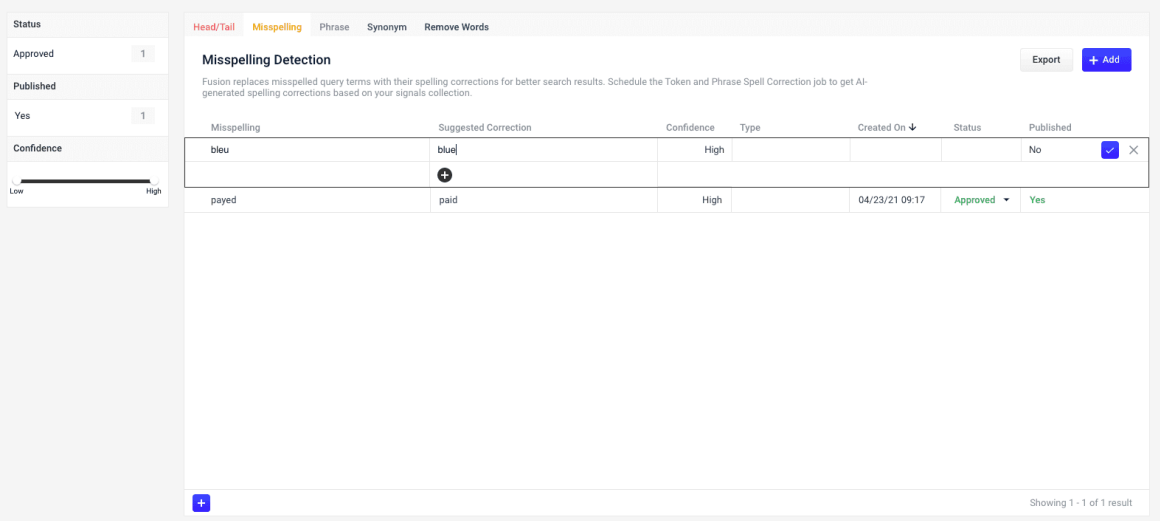
   A new spelling correction appears at the top of the list:



4. Enter the misspelled word or phrase.
5. Enter one or more spelling corrections.

> **Tip**
>
> It is not necessary to set a confidence value.

6. Select the spelling correction's status, depending on whether you want to deploy it the next time you publish your changes ("Approved") or save it for further review ("Pending").
7. Click the check mark to save the new spelling correction:



Publishing your changes

How to publish updated spelling corrections

1. In the **Misspelling Detection** screen, click the **PUBLISH** button.
   Fusion prompts you to confirm that you want to publish your changes.
2. Click **PUBLISH**.

> **Tip**
>
> You can un-publish a query rewrite by changing its status to "denied", then clicking **PUBLISH**.

Tuning the misspelling detection job

The default configuration for the *Token and Phrase Spell Correction job*, respectively, is designed for high accuracy and works well with most signal datasets, depending on the volume and quality of the signals. If you are seeing too few results, or too many inaccurate results, then you can try tuning the job to achieve better results.

To modify job configurations, you must be a Fusion user with one of the following roles or permissions that include access to job configurations:

- *admin*
- *developer*
- *permissions*

Query rewrite jobs post-processing cleanup

To perform more extensive cleanup of query rewrites, complete the procedures in *Query Rewrite Jobs Post-processing Cleanup*.

Use page metadata

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

Access and edit page metadata

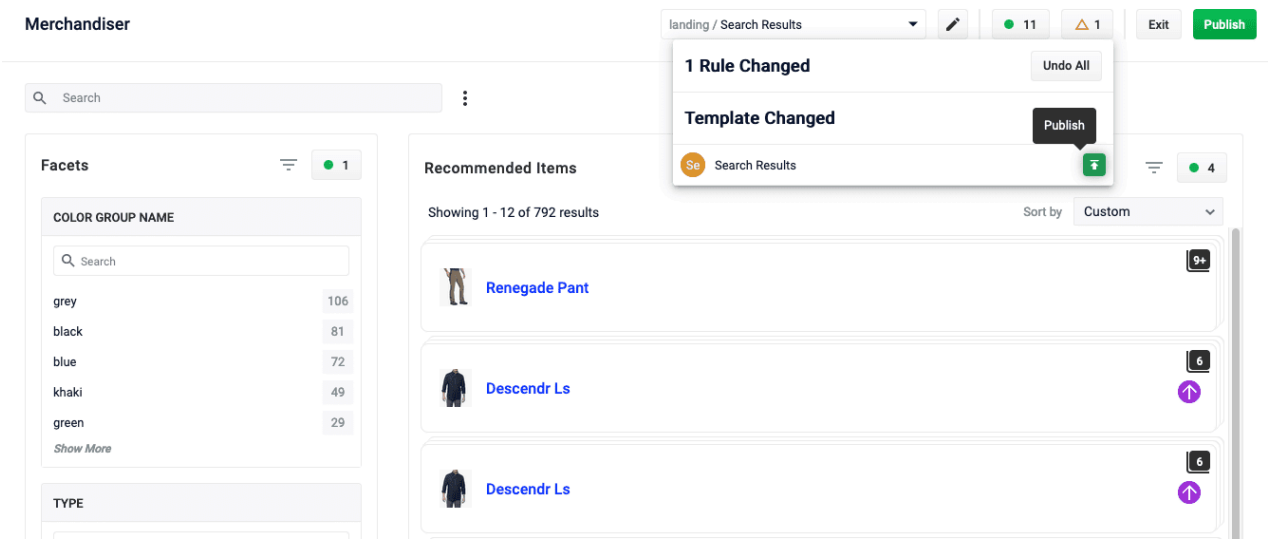
Note

You must be within an active template to use the **More Menu**.

1. Sign into Managed Fusion and click your application.
2. Click **Relevance > Rules > Merchandiser**.
3. Click ⋮ .
4. Click <> .
5. On the Page Metadata screen, edit the information and click **Save**.

Publish a template

After editing your page metadata, your changes will not be applied until you publish the template. The Publish function is available from the Rules toolbar:



1. Click △ 1 .
2. In the Template Changed section, hold the pointer to the right of the page description to display the button.
3. Review the changes and click **Publish** to apply metadata changes to your site.

Clone facet values

1. On the Merchandiser screen, click ⋮ .
2. Click <> .
3. Select one or more of your existing facets and click **Clone values** .

Find a page using URL search

1. On the Merchandiser screen, click ⋮ .
2. Click .
3. In the Domain URL field, select the domain URL for the website you want to search. For example, `/https://www.lucidworks.com` .
4. In the Path URL field, enter the URL you want to search. For example, `/products/managed-fusion` .
5. Click **Search**.

Use Page Metadata

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion*

Use the Page Metadata feature to edit the following fields in a Predictive Merchandiser template:

- `page-title`
- `page-description`
- `page-keywords`
- `image-url`
- `page-url`

Access and Edit Page Metadata

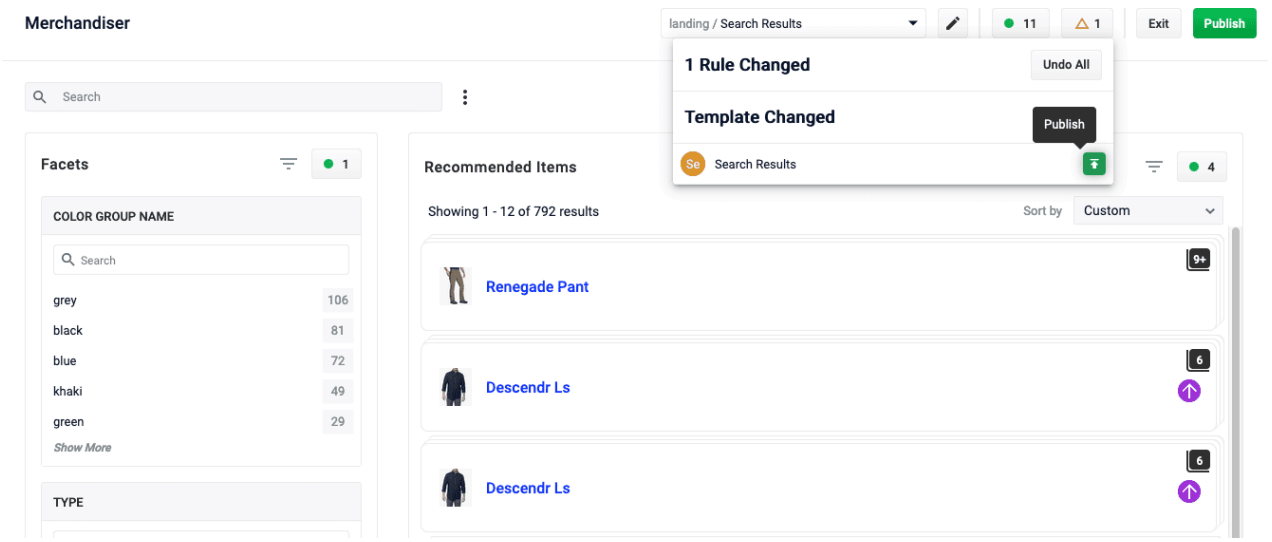
> **Note**
>
> You must be within an active template to use the **More Menu**.

1. Choose your app and navigate to **Rules** > **Merchandiser**.
2. Click the ⋮ .
3. Next, click <> .

   This opens a dialog with predefined fields, based on the current information in place for your page.
4. Make your changes, then click **Save**.

Publish a template

After editing your page metadata, your changes will not be applied until you publish the template. The **Publish** function is available from the **Rules toolbar**:



1. Click △ 1 .

   In the **Template Changed** section, you will see short description of the page that you edited the metadata for.

2. Hover to the right of the page description to see the button.

3. Review the changes and click the **Publish** button at the end of the dialogue to publish the template.
Your metadata changes have now been applied to your site.

Clone Facet Values

1. In the Merchandiser tab, click the .

2. Click .

3. Select one or more of your existing facets and click **Clone values** .

Find a page using URL Search

1. In the Merchandiser tab, click the .

2. Click .

3. In the **Domain URL** field, select the domain URL for the website you wish to search in the dropdown menu. For example, **https://www.lucidworks.com** .
4. Type the path URL you wish to search for in the **Path URL** field. For example, `/products/fusion` .
5. Click **Search**.

Use phrase detection

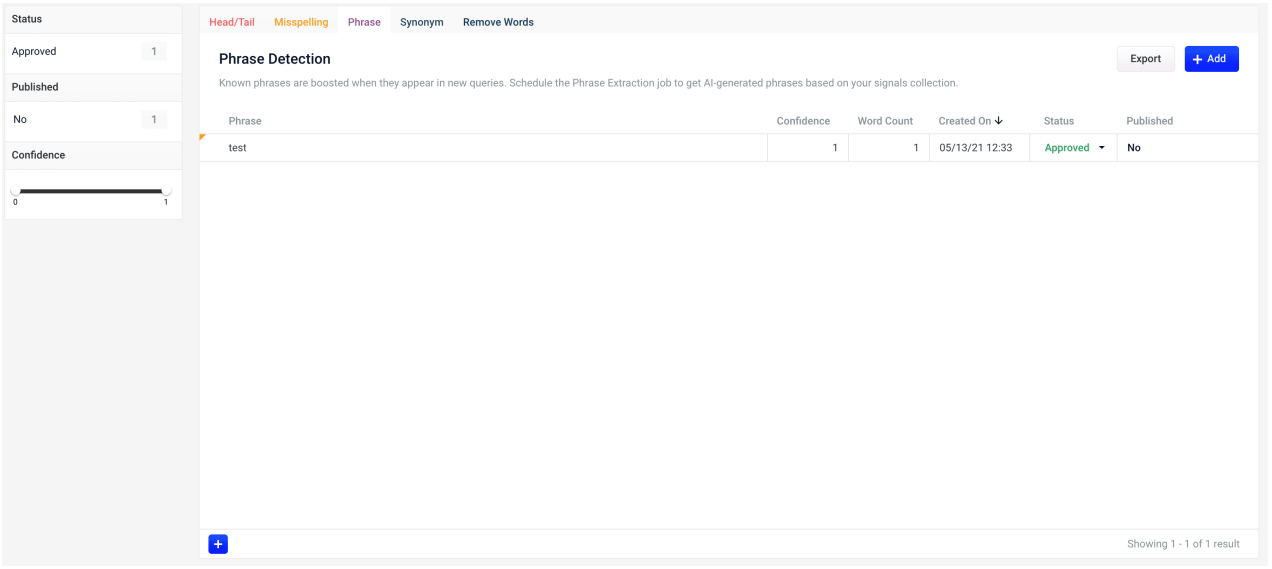
How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

The *Phrase Extraction job* automatically creates phrases based on your AI-generated data. When you navigate to **Relevance** > **Rules** > **Rewrite** and select the **Phrase** tab, the application displays the **Phrase Detection** screen.

Tip

When you manually add new phrases, subsequent job runs use those documents as input for machine learning to improve the job's output. Unlike job-generated documents, manually-added query rewriting documents are never overwritten by new job output.



LucidAcademy

Lucidworks offers free training to help you get started.

The **Course** for **Query Analytics** focuses on how Fusion provides query analytics to detect and improve underperforming queries:

Visit the LucidAcademy to see the full training catalog.

Reviewing auto-generated phrases

Phrases that are automatically generated by the *Phrase Extraction job* are assigned one of the following status values:

- **Auto**
  These results have a confidence level as a threshold to automatically deploy them to the `_query_rewrite` collection. This threshold can be specified in the configuration parameter **Minimum Likelihood Score** (default value 0.1).
  No action is required on these results, but you can edit them if you wish.
- **Pending**
  The confidence level is ambiguous, and the result must be reviewed by a user before it can be deployed. It will only be moved from the `_query_rewrite_staging` collection to the `_query_rewrite` collection when its status has changed to "Approved" *and* it has been published.

How to review a pending phrase result

1. Navigate to **Relevance** > **Rules** > **Rewrite**.
2. Select **Phrase** tab. The application displays the **Phrase Detection** screen.

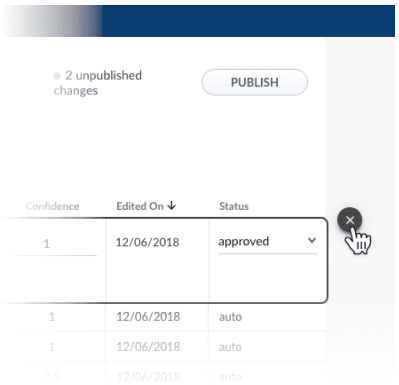
> **Tip**
>
> Notice the **Status** facet on the left. Click **Pending** to view only the items that need review.

3. Click the icon next to the phrase.

4. In the **Status** column, select either "Approved" or "Denied".
   Optionally, you can also edit the phrase itself.

> **Tip**
>
> Although the Confidence field is also editable, changing its value makes no difference.

5. Click the **Close** icon next to the updated phrase:



Note

Approving a phrase does not automatically deploy it to the `_query_rewrite` collection. When you have finished your review, you must click **Publish** to deploy your changes.

Adding new phrases

You can manually add phrases in addition to any generated by the *Phrase Extraction job*.

How to add a phrase

1. Navigate to **Relevance** > **Rules** > **Rewrite**.
2. Select **Phrase** tab. The application displays the **Phrase Detection** screen.
3. At the bottom of the rules list, click the ⊕ icon. A new phrase appears at the top of the list.
4. Enter the phrase.

   Tip

   It is not necessary to set a confidence value.

5. Select the phrase's status, depending on whether you want to deploy it the next time you publish your changes ("Approved") or save it for further review ("Pending").
6. Click the check mark to save the new phrase.

Publishing your changes

How to publish updated phrases

1. In the **Phrase Detection** screen, click the **PUBLISH** button.
   Managed Fusion prompts you to confirm that you want to publish your changes.
2. Click **PUBLISH**.

Tip

You can un-publish a query rewrite by changing its status to "denied", then clicking **PUBLISH**.

Use Phrase Detection

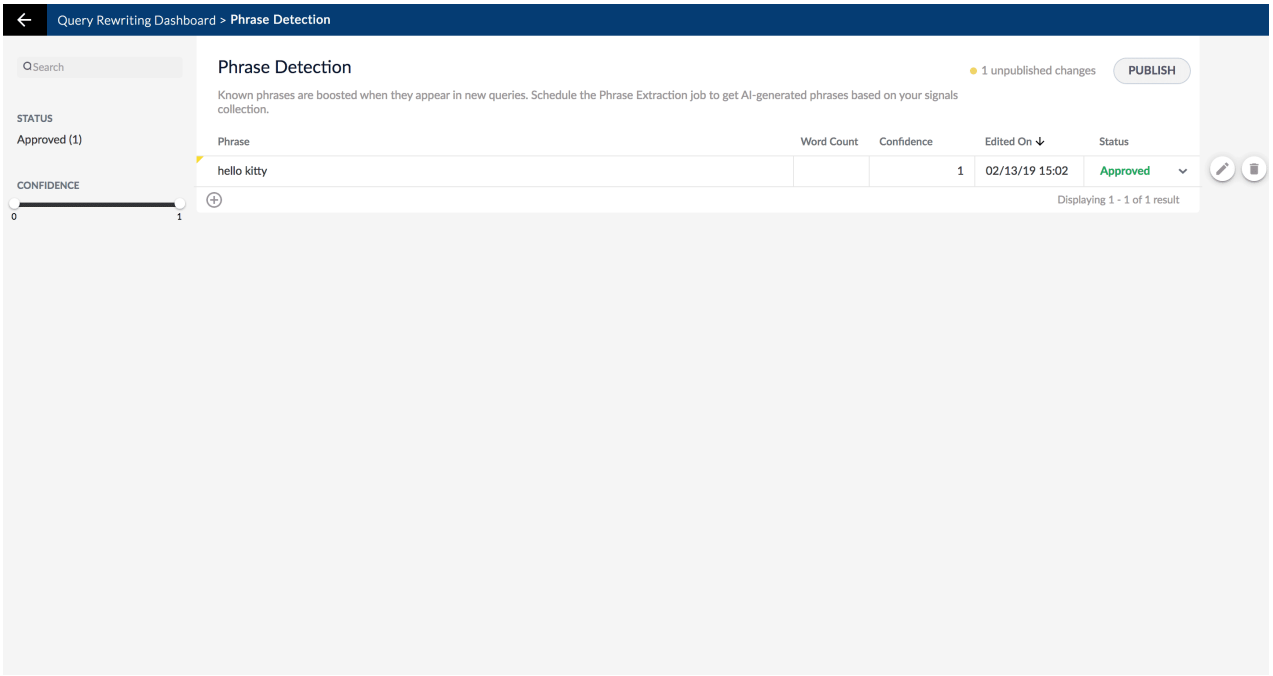
How-to

> Compatible versions: 4.2
>
> Works with: *Fusion AI*, *Fusion Server*

The *Phrase Extraction job* automatically creates phrases based on your AI-generated data. When you navigate to **Relevance** > **Rules**, the application displays the **Query Rewriting** screen. Select **Phrase Detection**. The application displays the **Phrase Detection** screen.

Tip

When you manually add new phrases, subsequent job runs use those documents as input for machine learning to improve the job's output. Unlike job-generated documents, manually-added query rewriting documents are never overwritten by new job output.



LucidAcademy

Lucidworks offers free training to help you get started.

The **Course** for **Query Analytics** focuses on how Fusion provides query analytics to detect and improve underperforming queries:

Visit the LucidAcademy to see the full training catalog.

Reviewing auto-generated phrases

Phrases that are automatically generated by the *Phrase Extraction job* are assigned one of the following status values:

- **Auto**
  These results have a confidence level as a threshold to automatically deploy them to the `_query_rewrite` collection. This threshold can be specified in the configuration parameter **Minimum Likelihood Score** (default value 0.1).
  No action is required on these results, but you can edit them if you wish.
- **Pending**
  The confidence level is ambiguous, and the result must be reviewed by a user before it can be deployed. It will only be moved from the `_query_rewrite_staging` collection to the `_query_rewrite` collection when its status has changed to "Approved" *and* it has been published.

How to review a pending phrase result

1. Navigate to **Relevance** > **Rules**. The application displays the **Query Rewriting** screen.
2. Select **Phrase Detection**. The application displays the **Phrase Detection** screen.

Tip

Notice the **Status** facet on the left. Click **Pending** to view only the items that need review.

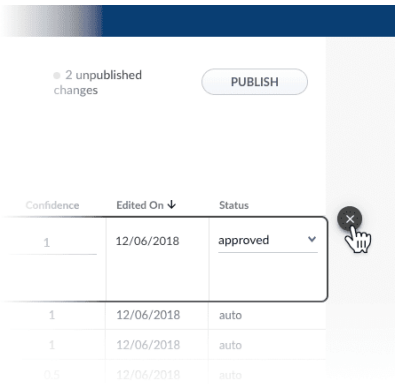
3. Click the icon next to the phrase.

4. In the **Status** column, select either "Approved" or "Denied".
   Optionally, you can also edit the phrase itself.

Tip

Although the Confidence field is also editable, changing its value makes no difference.

5. Click the **Close** icon next to the updated phrase:



Note

Approving a phrase does not automatically deploy it to the `_query_rewrite` collection.
When you have finished your review, you must click **Publish** to deploy your changes.

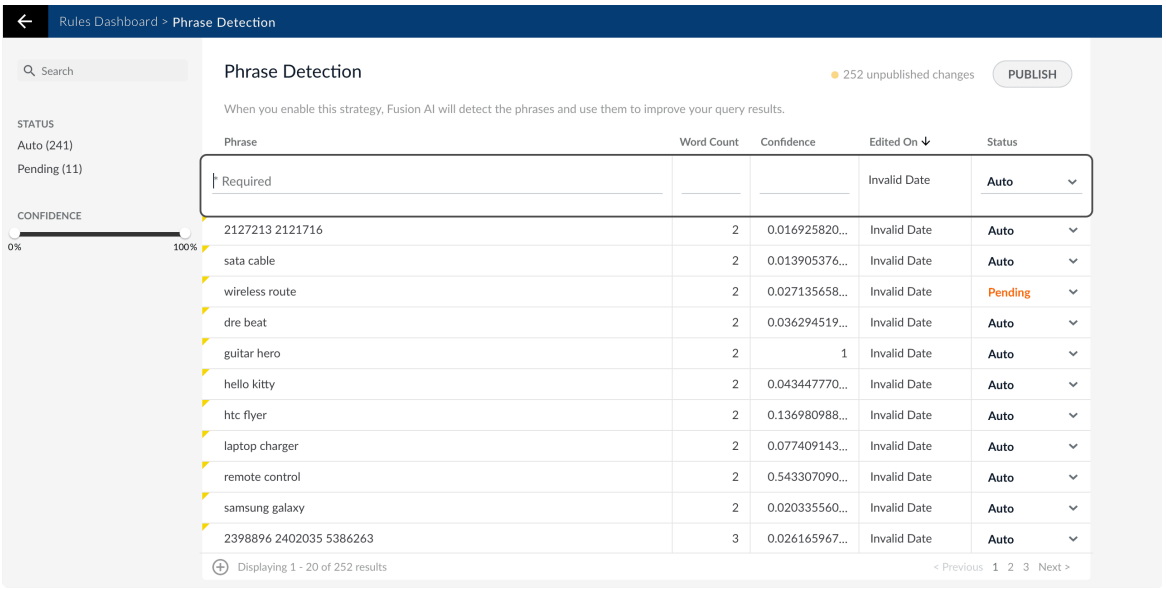
Adding new phrases

You can manually add phrases in addition to any generated by the *Phrase Extraction job*.

How to add a phrase

1. Navigate to **Relevance** > **Rules**. The application displays the **Query Rewriting** screen.
2. Select **Phrase Detection**. The application displays the **Phrase Detection** screen.
3. At the bottom of the rules list, click the ⊕ icon.

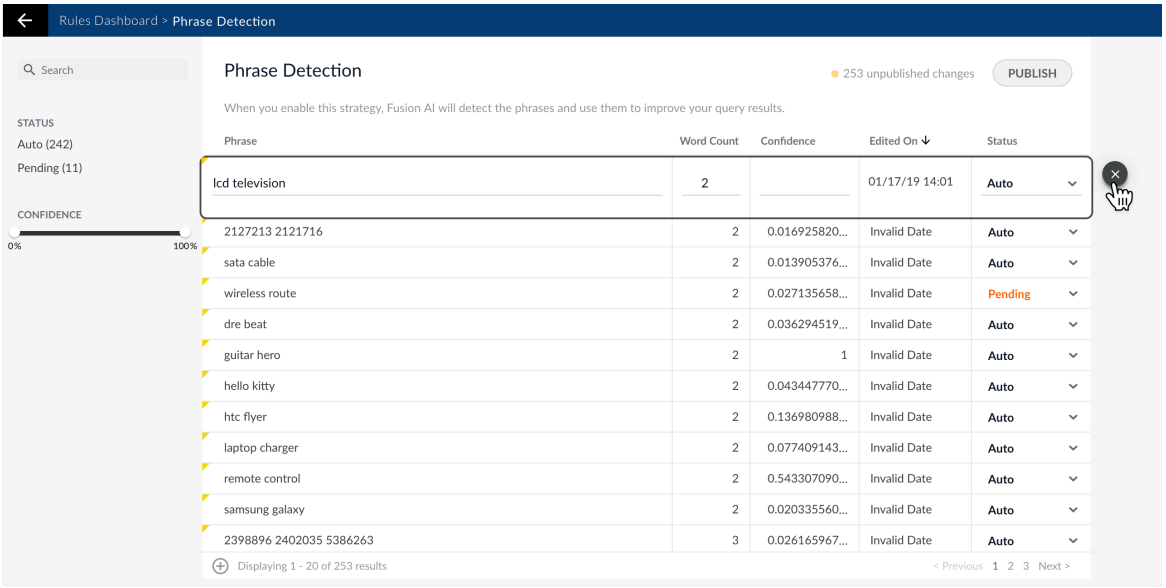
   A new phrase appears at the top of the list:



4. Enter the phrase.

> **Tip**
>
> It is not necessary to set a confidence value.

5. Select the phrase's status, depending on whether you want to deploy it the next time you publish your changes ("Approved") or save it for further review ("Pending").
6. Click the check mark to save the new phrase:



Publishing your changes

How to publish updated phrases

1. In the **Phrase Detection** screen, click the **PUBLISH** button.
   Fusion prompts you to confirm that you want to publish your changes.

2. Click **PUBLISH**.

> **Tip**
>
> You can un-publish a query rewrite by changing its status to "denied", then clicking **PUBLISH**.

Use Phrase Detection

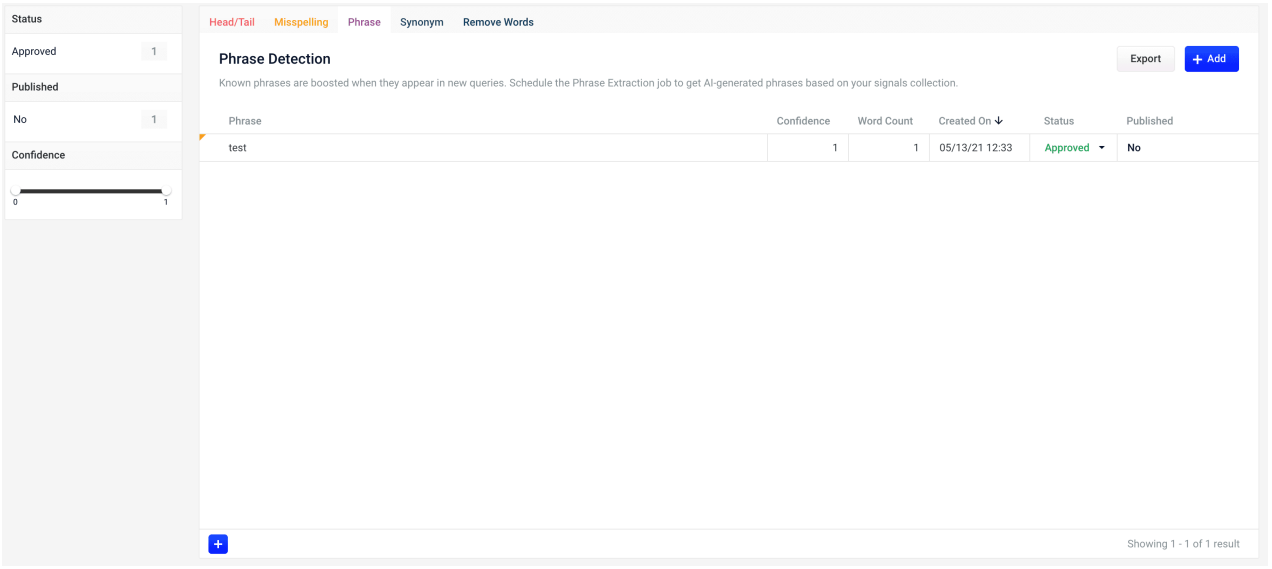
How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion*

The *Phrase Extraction job* automatically creates phrases based on your AI-generated data. When you navigate to **Relevance** > **Rules** > **Rewrite** and select the **Phrase** tab, the application displays the **Phrase Detection** screen.

Tip

When you manually add new phrases, subsequent job runs use those documents as input for machine learning to improve the job's output. Unlike job-generated documents, manually-added query rewriting documents are never overwritten by new job output.



LucidAcademy

Lucidworks offers free training to help you get started.

The **Course** for **Query Analytics** focuses on how Fusion provides query analytics to detect and improve underperforming queries:

Visit the LucidAcademy to see the full training catalog.

Reviewing auto-generated phrases

Phrases that are automatically generated by the *Phrase Extraction job* are assigned one of the following status values:

- **Auto**
  These results have a confidence level as a threshold to automatically deploy them to the `_query_rewrite` collection. This threshold can be specified in the configuration parameter **Minimum Likelihood Score** (default value 0.1).
  No action is required on these results, but you can edit them if you wish.
- **Pending**
  The confidence level is ambiguous, and the result must be reviewed by a user before it can be deployed. It will only be moved from the `_query_rewrite_staging` collection to the `_query_rewrite` collection when its status has changed to "Approved" *and* it has been published.

How to review a pending phrase result

1. Navigate to **Relevance** > **Rules** > **Rewrite**.
2. Select **Phrase** tab. The application displays the **Phrase Detection** screen.

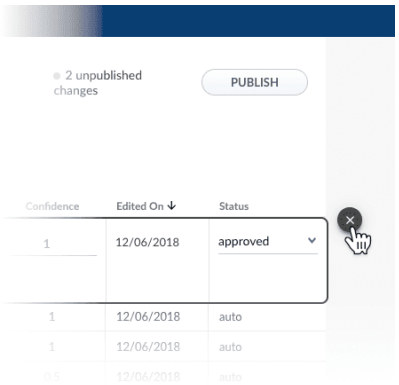
> **Tip**
>
> Notice the **Status** facet on the left. Click **Pending** to view only the items that need review.

3. Click the icon next to the phrase.

4. In the **Status** column, select either "Approved" or "Denied".
   Optionally, you can also edit the phrase itself.

> **Tip**
>
> Although the Confidence field is also editable, changing its value makes no difference.

5. Click the **Close** icon next to the updated phrase:



Note

Approving a phrase does not automatically deploy it to the `_query_rewrite` collection. When you have finished your review, you must click **Publish** to deploy your changes.

Adding new phrases

You can manually add phrases in addition to any generated by the *Phrase Extraction job*.

How to add a phrase

1. Navigate to **Relevance** > **Rules** > **Rewrite**.
2. Select **Phrase** tab. The application displays the **Phrase Detection** screen.
3. At the bottom of the rules list, click the ⊕ icon. A new phrase appears at the top of the list.
4. Enter the phrase.

   Tip

   It is not necessary to set a confidence value.

5. Select the phrase's status, depending on whether you want to deploy it the next time you publish your changes ("Approved") or save it for further review ("Pending").
6. Click the check mark to save the new phrase.

Publishing your changes

How to publish updated phrases

1. In the **Phrase Detection** screen, click the **PUBLISH** button.
   Fusion prompts you to confirm that you want to publish your changes.
2. Click **PUBLISH**.

Tip

You can un-publish a query rewrite by changing its status to "denied", then clicking **PUBLISH**.

Use Predictive Merchandiser Facets

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion*

Add a Facet

You can create rules that add facets by interacting directly with the Facet List in Interactive Mode. You can add facets that are set unconditionally, or facets that are only set when your customers run a particular query. For example, you might want to set a music genre facet when a customer navigates to the music section of your catalog.

Adding a facet automatically creates a new rule if a default facet rule does not already exist. If a default facet rule does exist, adding a facet updates that rule with the new facet.

Enable the Facet Zone (Fusion 5.4 and later)

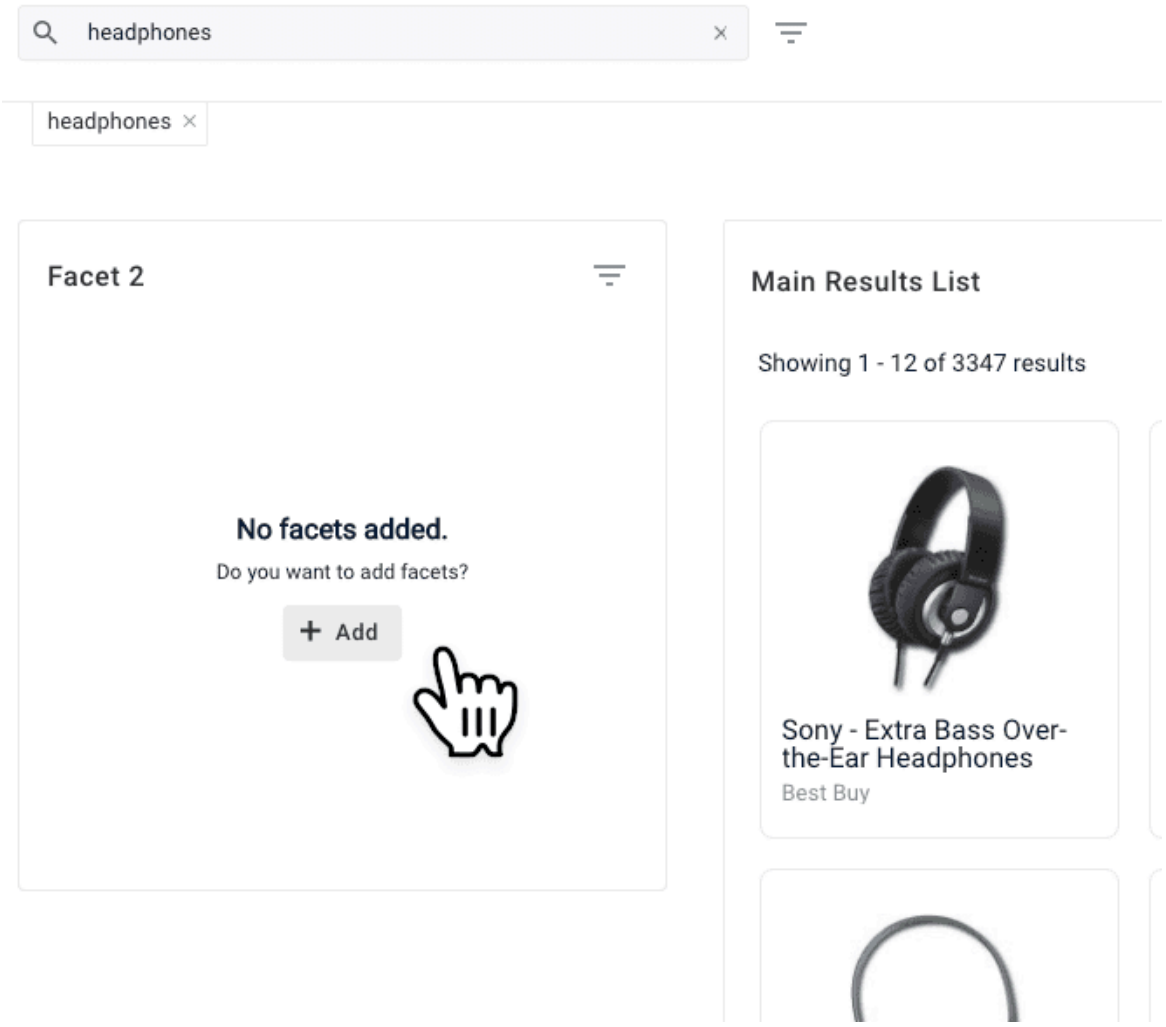
In Fusion 5.4 and later, facets appear in the zone on the left panel of the Predictive Merchandiser, once enabled. Follow the steps below to turn on the facets zone:

1. Click **Start Task**. The button appears.
2. In the left panel, click **+Add** and enter the desired parameters.
3. Click **Done** when you've finished.

> You can also add facets that are not dependent on rules being triggered. To do this, edit the Facet Zone widget (5.4 and later) or the *Facet List widget* (5.3 and earlier).

Edit the Facet Zone widget (5.4 and later)

1. In the facet zone, click **+Add**. The **Edit Rule** dialog box appears.



2. Click the **Facet Field** field, and enter the desired facet parameter.

Tip

You can add multiple facet parameters by clicking the **+ Add** button: + ADD

3. Click the **Save** button to add your facet: Save

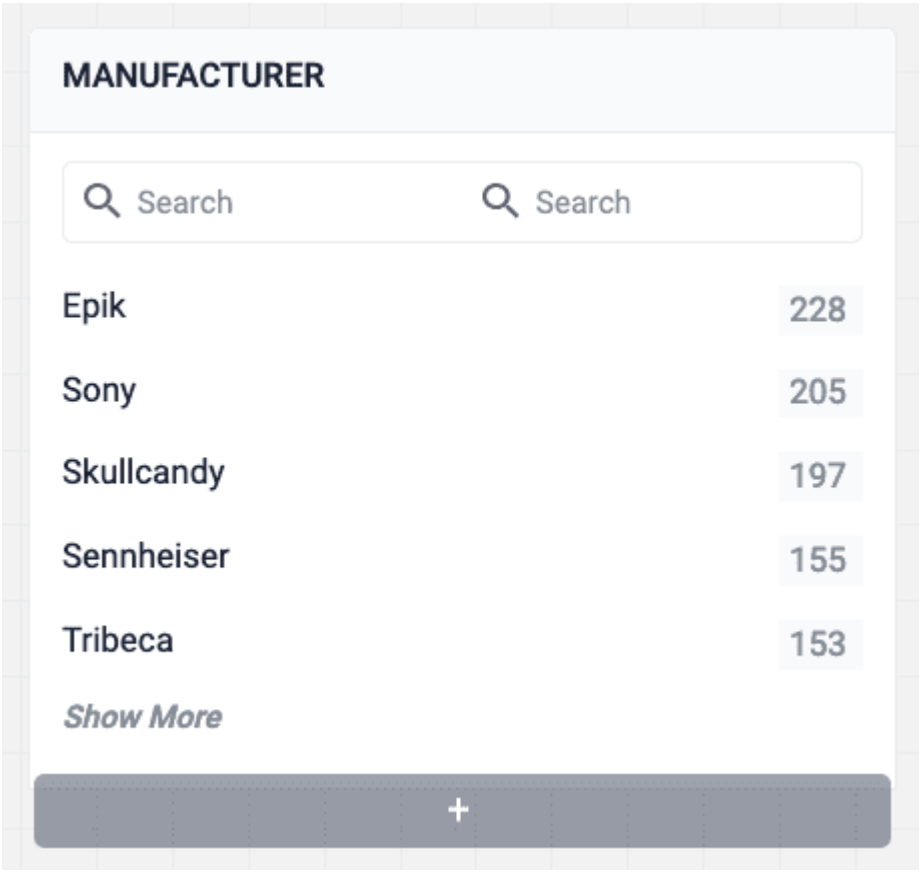
Edit the Facet List widget (5.3 and earlier)

1. Select **Add facet**. The **Edit Rule** dialog box appears.
2. Select the **Facet Field** field, and enter the desired facet parameter.

3. Click the **Save** button to add your facet:

Add Additional Facets

1. Hover your cursor above or below an existing facet. A **+** button appears:



2. Click the **+** button.

Move a Facet

1. Hover your cursor over a facet. A menu overlay appears at the top.
2. Click-and-hold the drag bar:
3. Move the facet to the desired location.
4. Release the click.

Remove a Facet

1. Hover your cursor over a facet. A menu overlay appears at the top.
2. Click the **Remove** button:

Block a Facet Item

1. Hover your cursor over a facet item. A menu overlay appears.
2. Click the **Block** button:

Bury a Facet Item

1. Hover your cursor over a facet item. A menu overlay appears.
2. Click the **Bury** button:

Boost a Facet Item

1. Hover your cursor over a facet item. A menu overlay appears.
2. Click the **Boost** button:

Reorder Boosted Facet Items

1. Hover over a boosted facet item. A drag bar appears.
2. Click and hold the drag bar:
3. Move the facet item to the desired location.
4. Release the click.

Use Predictive Merchandiser Facets

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

Add a Facet

You can create rules that add facets by interacting directly with the Facet List in Interactive Mode. You can add facets that are set unconditionally, or facets that are only set when your customers run a particular query. For example, you might want to set a music genre facet when a customer navigates to the music section of your catalog.

Adding a facet automatically creates a new rule if a default facet rule does not already exist. If a default facet rule does exist, adding a facet updates that rule with the new facet.

Enable the Facet Zone

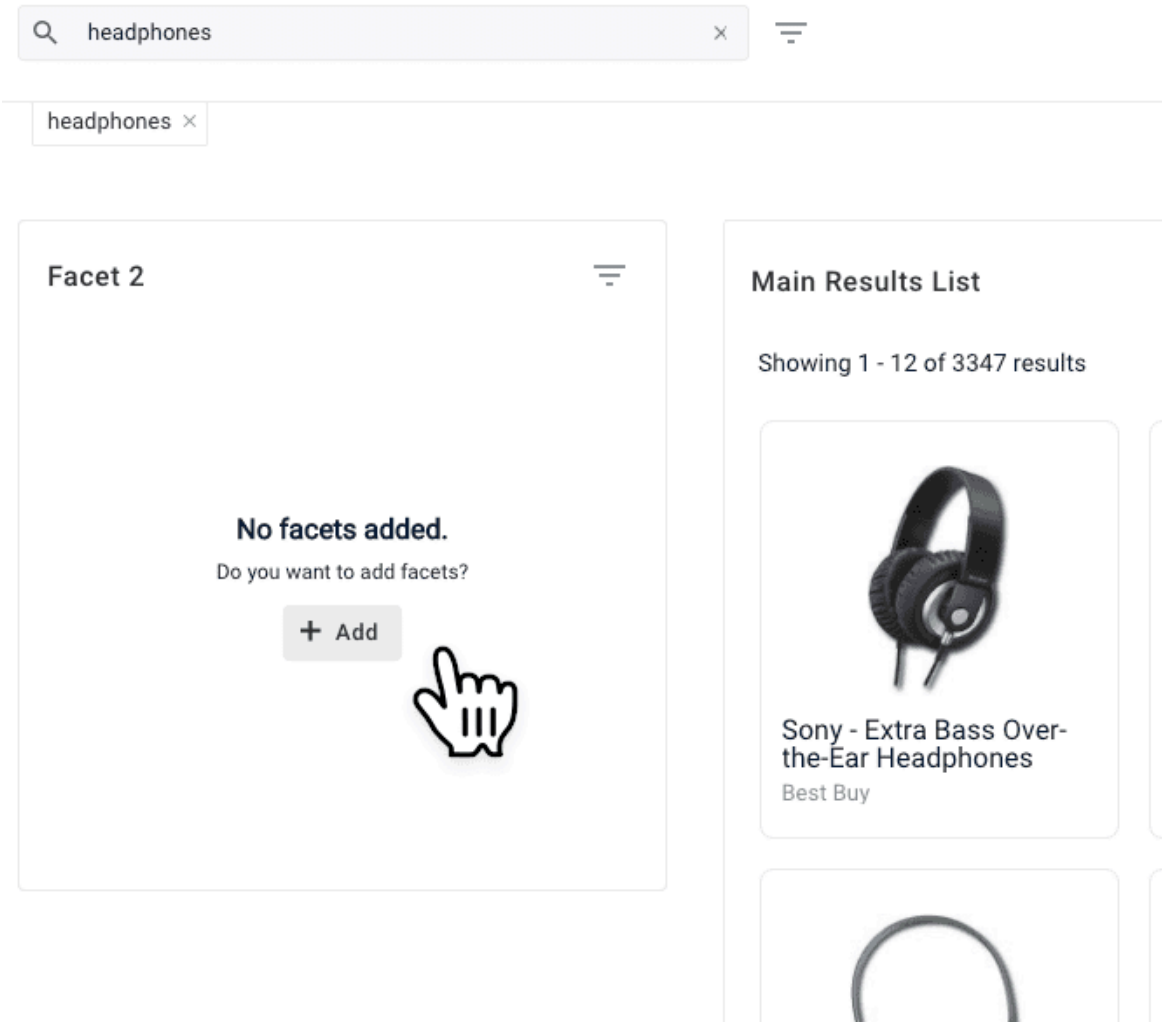
Facets appear in the zone on the left panel of the Predictive Merchandiser, once enabled. Follow the steps below to turn on the facets zone:

1. Click **Start Task**. The button appears.
2. In the left panel, click **+Add** and enter the desired parameters.
3. Click **Done** when you've finished.

> You can also add facets that are not dependent on rules being triggered. To do this, edit the Facet Zone widget.

Edit the Facet Zone widget

1. In the facet zone, click **+Add**. The **Edit Rule** dialog box appears.



2. Click the **Facet Field** field, and enter the desired facet parameter.

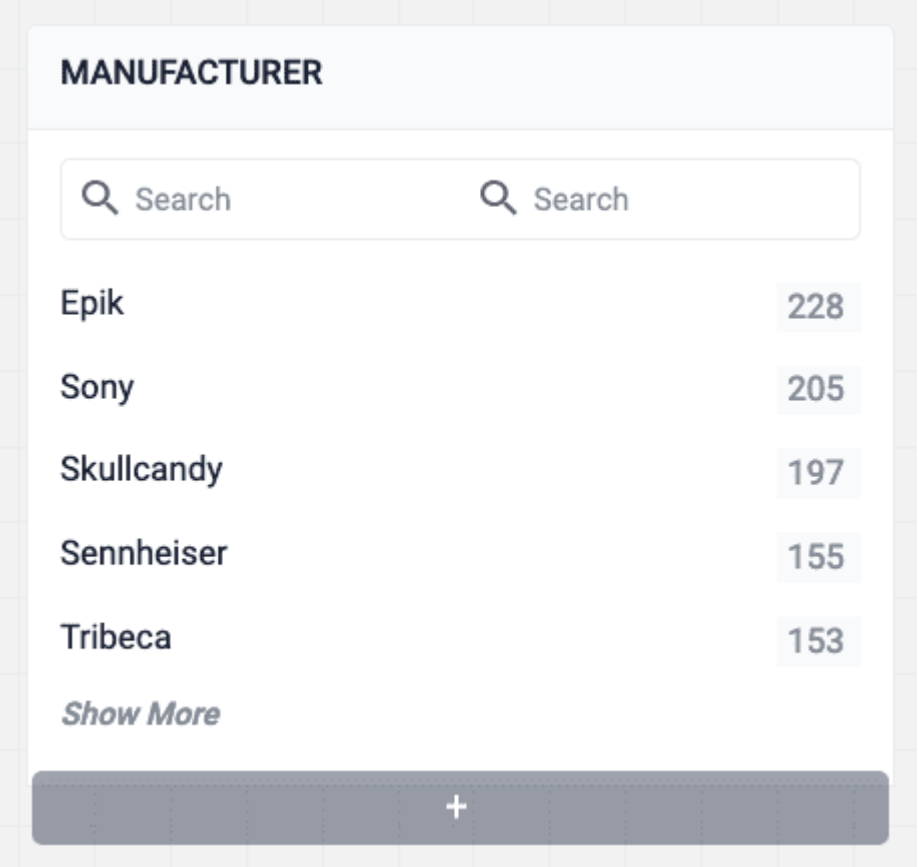
Tip

You can add multiple facet parameters by clicking the **+ Add** button: + ADD

3. Click the **Save** button to add your facet: Save

Add Additional Facets

1. Hover your cursor above or below an existing facet. A **+** button appears:



2. Click the **+** button.

Move a Facet

1. Hover your cursor over a facet. A menu overlay appears at the top.
2. Click-and-hold the drag bar:
3. Move the facet to the desired location.
4. Release the click.

Remove a Facet

1. Hover your cursor over a facet. A menu overlay appears at the top.
2. Click the **Remove** button:

Block a Facet Item

1. Hover your cursor over a facet item. A menu overlay appears.
2. Click the **Block** button:

Bury a Facet Item

1. Hover your cursor over a facet item. A menu overlay appears.
2. Click the **Bury** button:

Boost a Facet Item

1. Hover your cursor over a facet item. A menu overlay appears.
2. Click the **Boost** button:

Reorder Boosted Facet Items

1. Hover over a boosted facet item. A drag bar appears.
2. Click and hold the drag bar:
3. Move the facet item to the desired location.
4. Release the click.

Use Predictive Merchandiser Query Rewrites

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion*

You can create query rewrite rules in Predictive Merchandiser. This is similar to using the *Rules Editor* to create query rewrite rules, except there are additional options in Predictive Merchandiser.

1. In Fusion, navigate to **Relevance** > **Rules** > **Merchandiser**.
2. Enter a search term or phrase in the search bar.
3. Click **Start Task**.
4. Hover your cursor next to the query. A **+** button will appear:
5. Click the **+** button. A list of query rewrites options will appear: *Head/Tail*, *Misspelling*, *Phrase*, *Synonym*, and *Remove Words*.

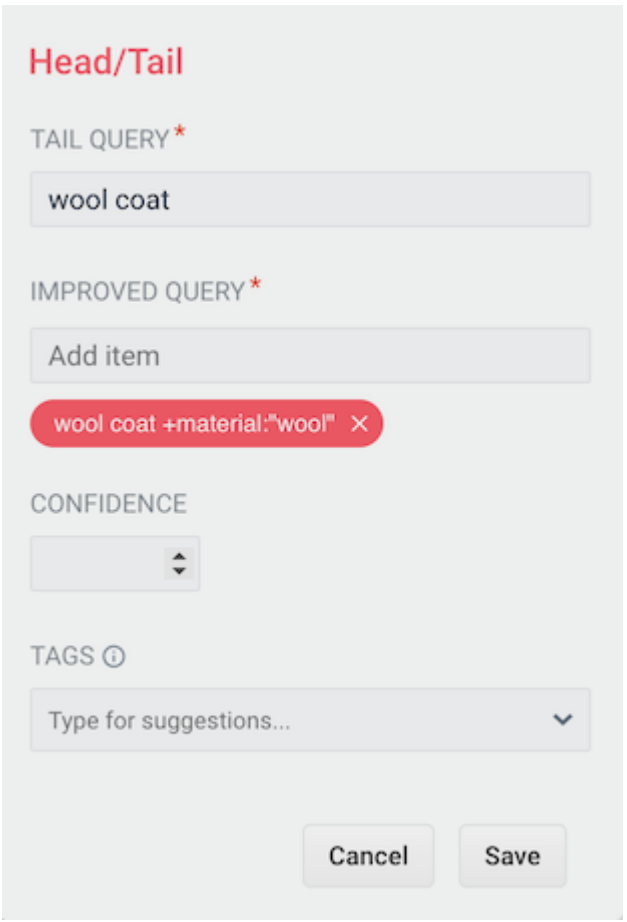
Head/Tail

You can create a Head/Tail rewrite to improve search results using methods other than correcting for misspellings or synonym expansion. When a poorly defined search term is identified, the original term is replaced by an improved search term.

For example, a search for `iphone case yellow` could be improved by using the search term `iphone case yellow +color:"yellow"` (in this case making use of the color field in the data).

Most Head/Tail rewrites are typically created automatically via machine learning. However, if desired, custom rewrites can be manually created using the following steps.

1. From the list of query rewrite options, select **Head/Tail**. A form will appear:

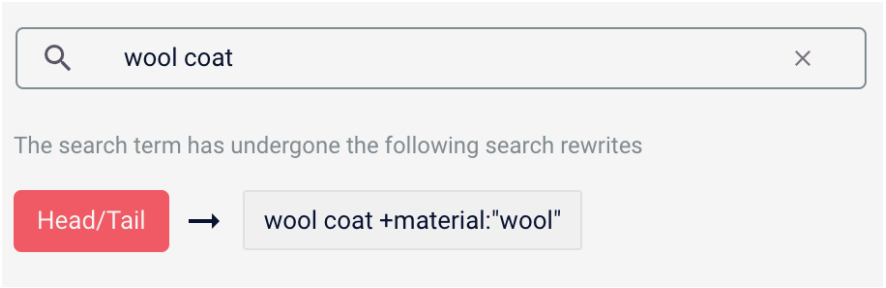


This form contains the following fields:

Parameter	Description	Example Value
Tail Query	The tail query itself.	`iphone case yellow`
Improved Query	The query that will replace the tail query phrase.	`iphone case yellow +color:"yellow"`
Tags	Optional metadata tags that can be used to identify and organize rewrites.	`blackfridaysale`

2. Enter one or more improved search terms in the **Improved Query** field.
3. Click the **Save** button.

The search is re-run using the improved search term, and a Search box information panel drops down to show that the new query rewrite has fired:

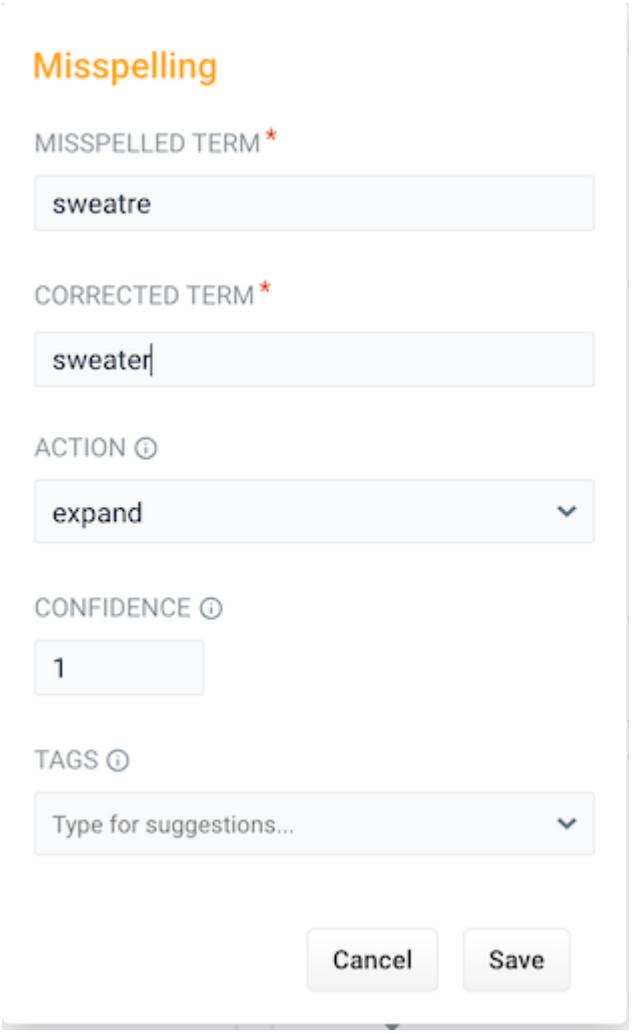


Misspelling

You can create a misspelling query rewrite to detect and correct common spelling mistakes. When a customer enters a search term containing a known misspelling, the incorrect spelling is replaced with the spelling correction.

For example, if your customers frequently misspell or mistype the word `sweater` as `sweatre`, you can set up a query rewrite to automatically correct it.

1. From the list of query rewrite options, select **Misspelling**. A form will appear:

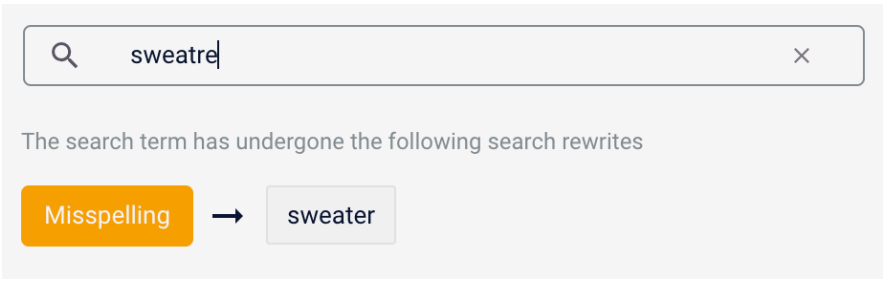


This form contains the following fields:

Parameter	Description	Example Value
Misspelled Term	The phrase itself.	`sweatre`
Corrected Term	The term that will replace the misspelled term.	`sweater`
Action	Action to perform.	
Confidence	Confidence score from the phrase job. A confidence level of `1` represents 100% confidence. For rules created automatically via machine learning, the confidence level will reflect the output from the machine learning model.	`1`
Tags	Optional metadata tags that can be used to identify and organize rewrites.	`blackfriday` `sale`

2. Enter one or more spelling corrections in the **Corrected Term** field.
3. Click the **Save** button.

The search is re-run using the corrected spelling, and a Search box information panel drops down to show that the new query rewrite has fired:

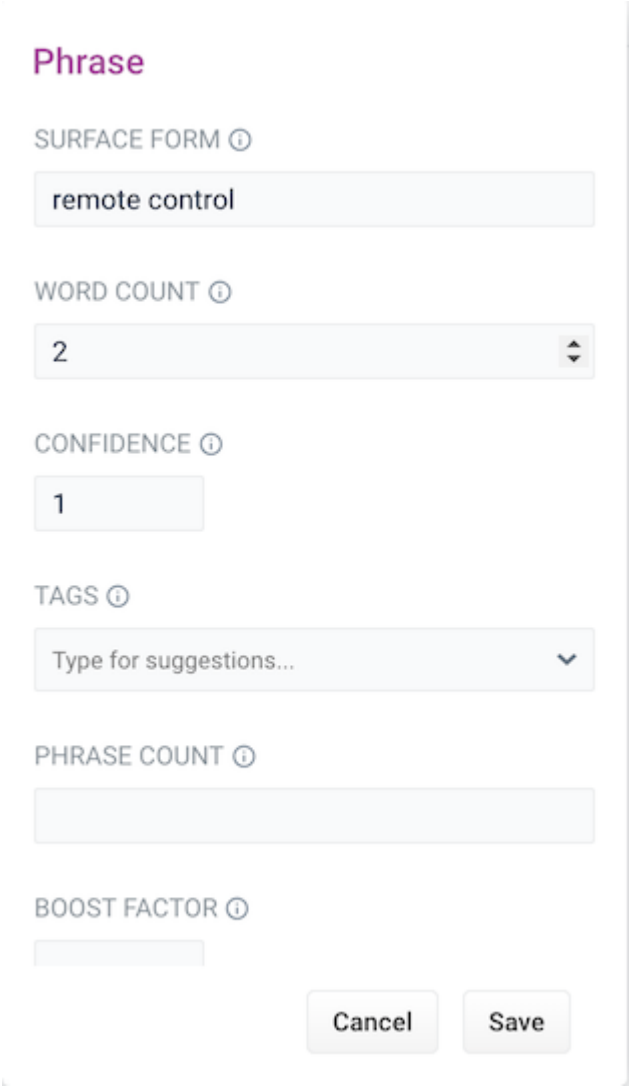


Phrase

You can use query rewriting to identify phrases used in search terms so that products with matching phrases are boosted in the search results. This is helpful when users do not use quotation marks to identify phrases in their search terms.

For example, without phrase detection a search for the words `remote control` would show results for both `remote` and `control`. With phrase detection, this search would correctly boost results for `"remote control"`.

1. From the list of query rewrite options, select **Phrase**. A form will appear:

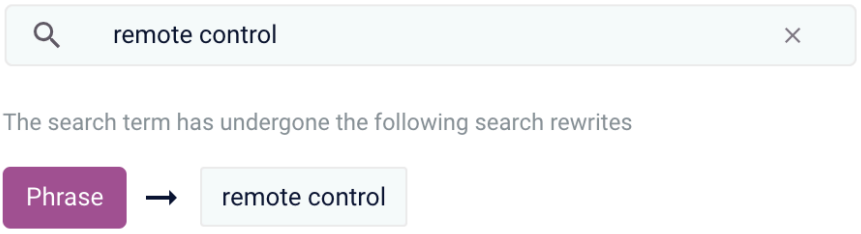


This form contains the following fields:

Parameter	Description	Example Value
Surface Form	The phrase itself.	`remote control`
Word Count	Indicates how many words are included in the phrase.	`2`
Confidence	Confidence score from the phrase job. A confidence level of `1` represents 100% confidence. For rules created automatically via machine learning, the confidence level will reflect the output from the machine learning model.	`1`
Tags	Optional metadata tags that can be used to identify and organize rewrites.	`blackfriday sale`
Phrase Count	Denotes how many times this phrase was found in the source. This value is automatically set via machine learning. It does not need to be set manually.	`5`
Boost Factor	The factor to use to boost this phrase in matching queries.	`2.0`
Slop Factor	Phrase slop, or the distance between the terms of the query while still considering it a phrase match.	`10`

2. Enter the number of words in the phrase in the **Word Count** field.
3. Click the **Save** button.

The search is re-run, with your search words identified as a phrase, and a Search box information panel drops down to show that the new query rewrite has fired:

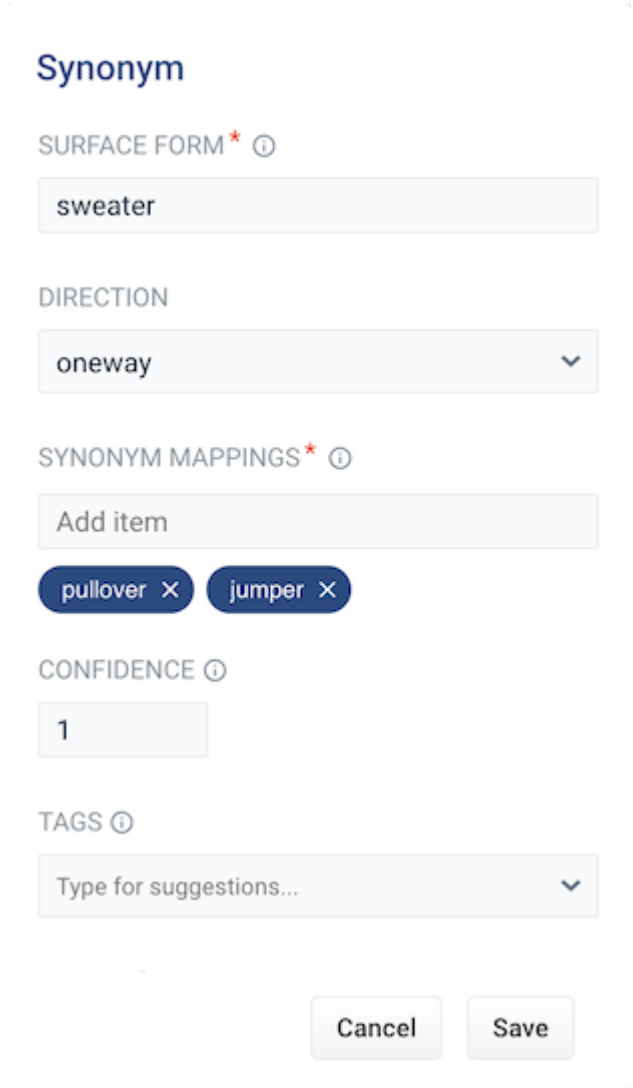


Synonym

You can specify synonyms for a specified search term so that alternative words with the same meaning are automatically used in the search query. When a customer enters a search term with a synonym match, the alternative words are used instead of, or in addition to, the original search term.

For example, the search term `sweater` could have the synonyms `pullover` and `jumper`.

1. From the list of query rewrite options, select **Synonym**. A form will appear:

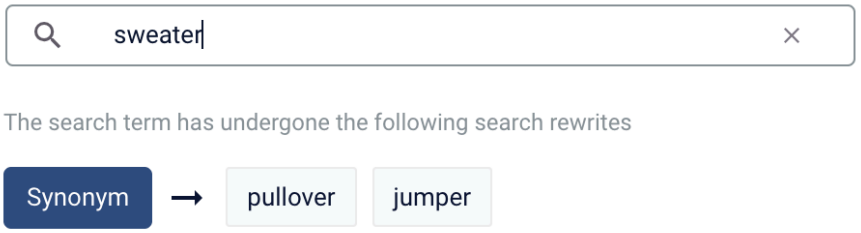


This form contains the following fields:

Parameter	Description	Example Value
Surface Form	The term that has synonyms.	`sweater`
Direction	With a **oneway** search, the original search term is replaced by the synonym. In the example above, `sweater` would be replaced by the alternative words `pullover` and `jumper`. With a **symmetric** search, the search query is expanded to include the original term and the synonyms, resulting in a greater number of potential hits. In the example above, this time the query would include `sweater`, `pullover`, and `jumper`.	`symmetric`
Synonym Mappings	Synonyms for the surface form.	`pullover`, `jumper`
Confidence	Confidence score from the phrase job. A confidence level of `1` represents 100% confidence. For rules created automatically via machine learning, the confidence level will reflect the output from the machine learning model.	`1`
Tags	Optional metadata tags that can be used to identify and organize rewrites.	`blackfridaysale`
Count	How many times this term occurred in the signal data when it was discovered. This value is optional when a rewrite is being defined manually.	`5`

2. Choose whether the direction is **oneway** or **symmetric**.
3. Enter one or more alternative words in the **Synonym Mappings** field.
4. Click the **Save** button.

The search is re-run using the synonyms, and a Search box information panel drops down to show that the new query rewrite has fired:



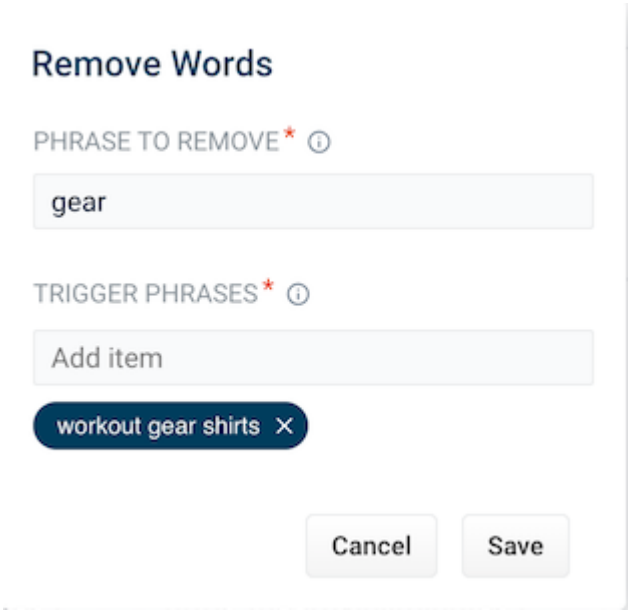
Remove Words

> **Note**
>
> The Remove Words feature is available in Fusion 5.4 and later.

You can create a Remove Words query rewrite to remove words from a query. This query rewrite is helpful when a word in the search query does not add value to the search results.

For example, you can rewrite a search query for `case study examples` to remove `examples` and then display results for `case study`.

1. From the list of query rewrite options, select Remove Words. A form appears:

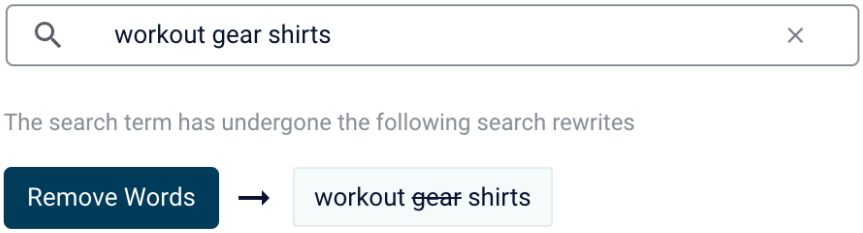


This form contains the following fields:

Parameter	Description	Example Value
Phrase to remove	The words to remove from the trigger phrase.	`examples`
Trigger phrases	The query that prompts the removal of the phrase. The trigger phrase is not necessarily a complete query. If the query contains the trigger phrase, then Fusion removes the phrase in the Phrase to Remove field.	`case study examples`

2. Enter a phrase to remove and a trigger phrase. Note that the phrase to remove is auto-populated with the query.
3. Click **Save**.

The search is re-run, and a Search box information panel drops down to show that the new query rewrite has fired:

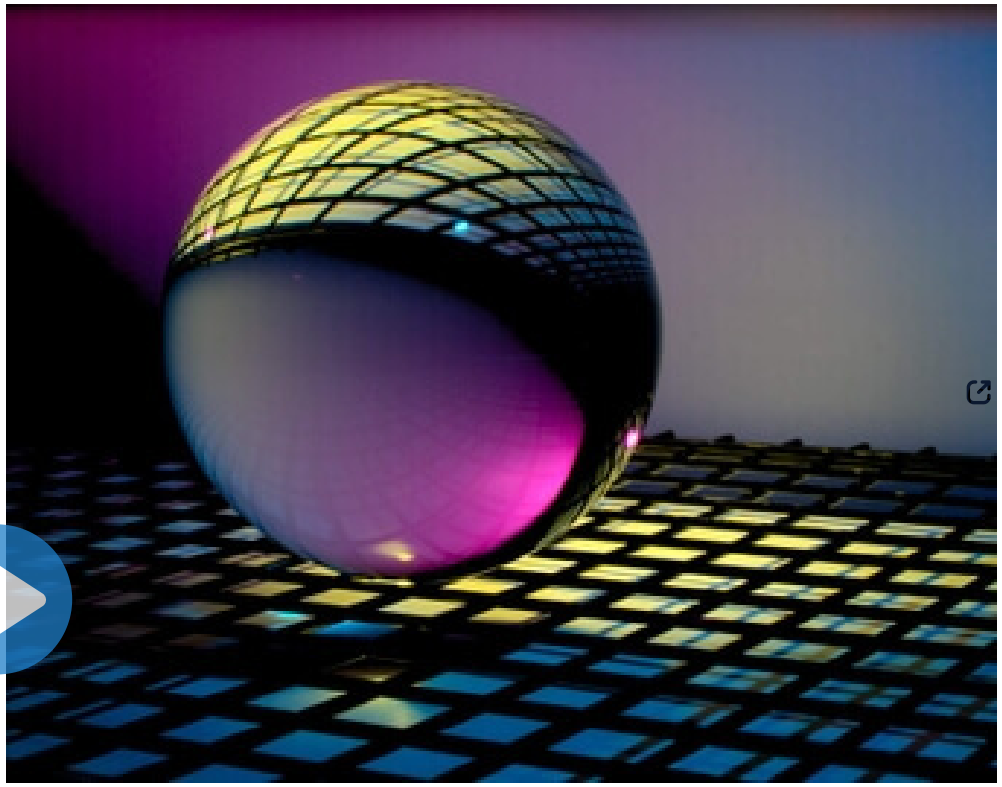


Additional resources

LucidAcademy

Lucidworks offers free training to help you get started.

The **Course** for **Predictive Merchandiser: Rewrites Manager** focuses on using search rewrites to boost conversions and create better customer experiences by improving your relevancy:



Visit the LucidAcademy to see the full training catalog.

Use Predictive Merchandiser Query Rewrites

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

You can create query rewrite rules in Predictive Merchandiser. This is similar to using the *Rules Editor* to create query rewrite rules, except there are additional options in Predictive Merchandiser.

1. In Fusion, navigate to **Relevance** > **Rules** > **Merchandiser**.
2. Enter a search term or phrase in the search bar.
3. Click **Start Task**.
4. Hover your cursor next to the query. A **+** button appears:
5. Click the **+** button. A list of query rewrites options will appear: *Head/Tail*, *Misspelling*, *Phrase*, *Synonym*, and *Remove Words*.

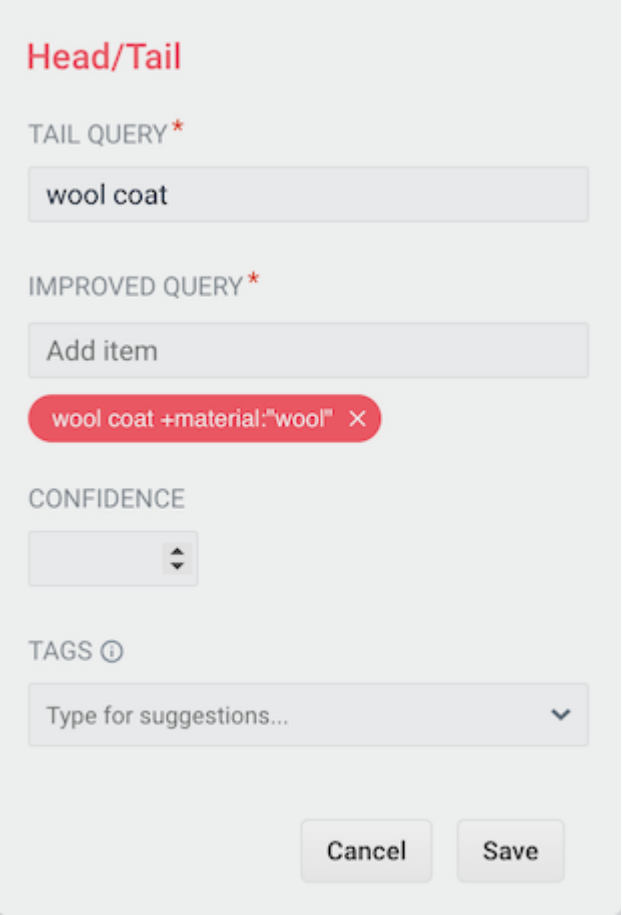
Head/Tail

You can create a Head/Tail rewrite to improve search results using methods other than correcting for misspellings or synonym expansion. When a poorly defined search term is identified, the original term is replaced by an improved search term.

For example, a search for `iphone case yellow` could be improved by using the search term `iphone case yellow +color:"yellow"` (in this case making use of the color field in the data).

Most Head/Tail rewrites are typically created automatically via machine learning. However, if desired, custom rewrites can be manually created using the following steps.

1. From the list of query rewrite options, select **Head/Tail**. A form will appear:

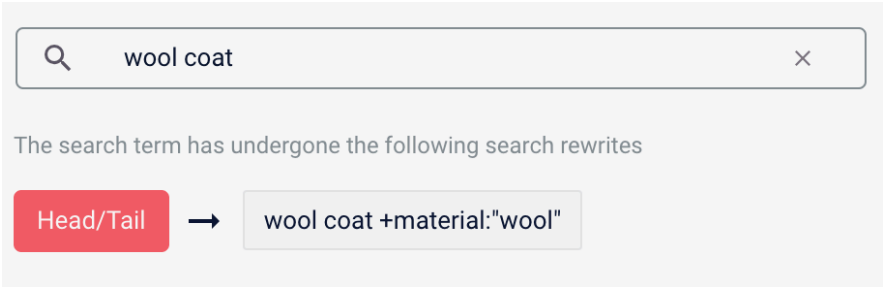


This form contains the following fields:

Parameter	Description	Example Value
Tail Query	The tail query itself.	`iphone case yellow`
Improved Query	The query that will replace the tail query phrase.	`iphone case yellow +color:"yellow"`
Tags	Optional metadata tags that can be used to identify and organize rewrites.	`blackfridaysale`

2. Enter one or more improved search terms in the **Improved Query** field.
3. Click the **Save** button.

The search is re-run using the improved search term, and a Search box information panel drops down to show that the new query rewrite has fired:

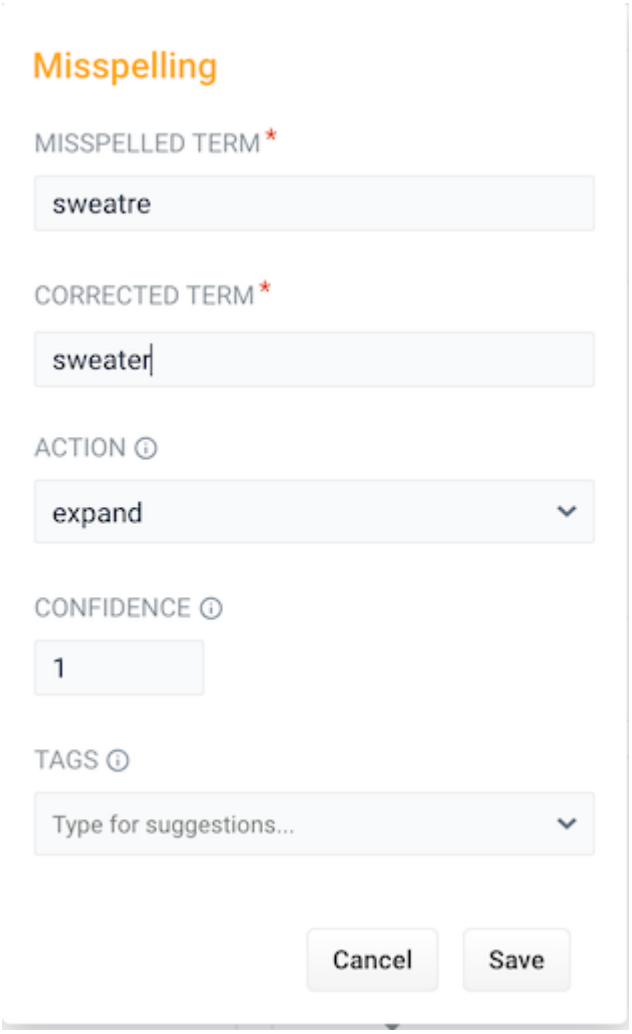


Misspelling

You can create a misspelling query rewrite to detect and correct common spelling mistakes. When a customer enters a search term containing a known misspelling, the incorrect spelling is replaced with the spelling correction.

For example, if your customers frequently misspell or mistype the word `sweater` as `sweatre`, you can set up a query rewrite to automatically correct it.

1. From the list of query rewrite options, select **Misspelling**. A form will appear:

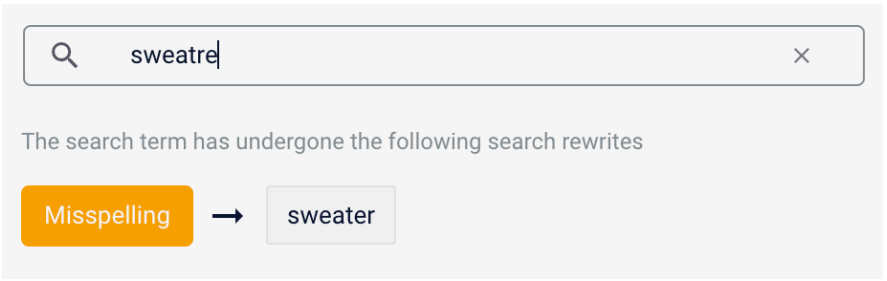


This form contains the following fields:

Parameter	Description	Example Value
Misspelled Term	The phrase itself.	`sweatre`
Corrected Term	The term that will replace the misspelled term.	`sweater`
Action	Action to perform.	
Confidence	Confidence score from the phrase job. A confidence level of `1` represents 100% confidence. For rules created automatically via machine learning, the confidence level will reflect the output from the machine learning model.	`1`
Tags	Optional metadata tags that can be used to identify and organize rewrites.	`blackfriday` `sale`

2. Enter one or more spelling corrections in the **Corrected Term** field.
3. Click the **Save** button.

The search is re-run using the corrected spelling, and a Search box information panel drops down to show that the new query rewrite has fired:

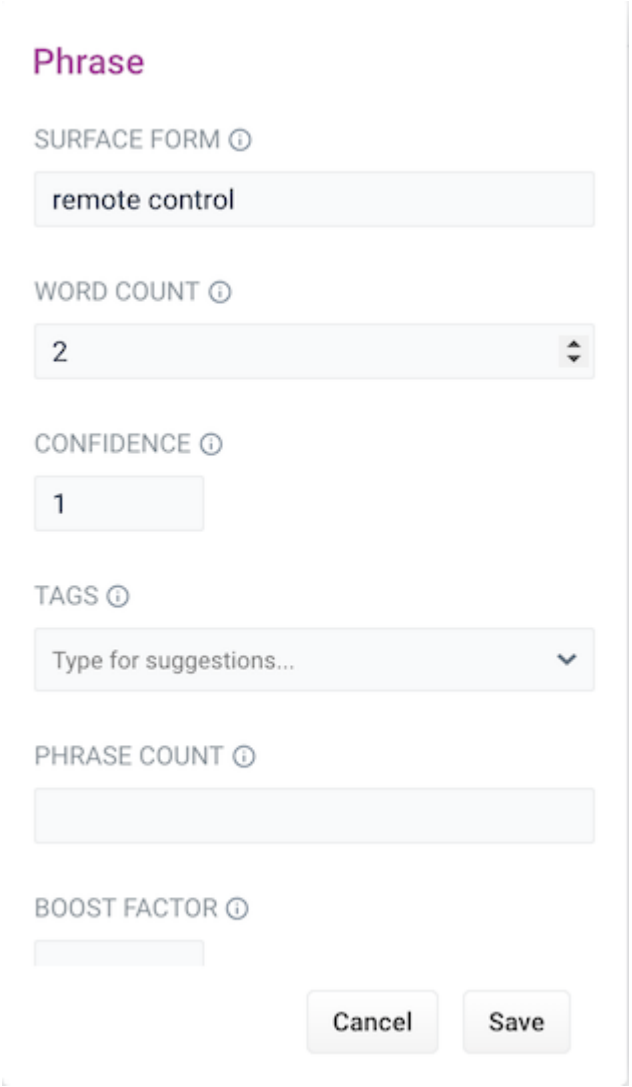


Phrase

You can use query rewriting to identify phrases used in search terms so that products with matching phrases are boosted in the search results. This is helpful when users do not use quotation marks to identify phrases in their search terms.

For example, without phrase detection a search for the words `remote control` would show results for both `remote` and `control`. With phrase detection, this search would correctly boost results for `"remote control"`.

1. From the list of query rewrite options, select **Phrase**. A form will appear:

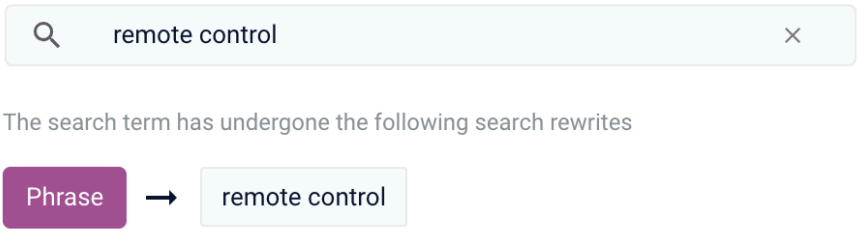


This form contains the following fields:

Parameter	Description	Example Value
Surface Form	The phrase itself.	`remote control`
Word Count	Indicates how many words are included in the phrase.	`2`
Confidence	Confidence score from the phrase job. A confidence level of `1` represents 100% confidence. For rules created automatically via machine learning, the confidence level will reflect the output from the machine learning model.	`1`
Tags	Optional metadata tags that can be used to identify and organize rewrites.	`blackfriday sale`
Phrase Count	Denotes how many times this phrase was found in the source. This value is automatically set via machine learning. It does not need to be set manually.	`5`
Boost Factor	The factor to use to boost this phrase in matching queries.	`2.0`
Slop Factor	Phrase slop, or the distance between the terms of the query while still considering it a phrase match.	`10`

2. Enter the number of words in the phrase in the **Word Count** field.
3. Click the **Save** button.

The search is re-run, with your search words identified as a phrase, and a Search box information panel drops down to show that the new query rewrite has fired:

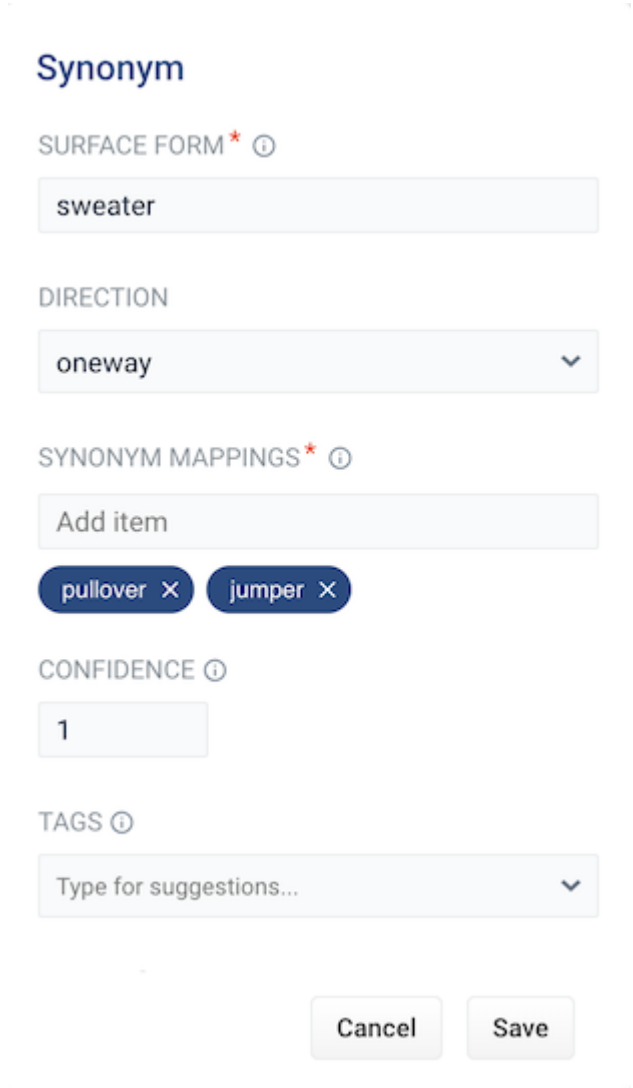


Synonym

You can specify synonyms for a specified search term so that alternative words with the same meaning are automatically used in the search query. When a customer enters a search term with a synonym match, the alternative words are used instead of, or in addition to, the original search term.

For example, the search term `sweater` could have the synonyms `pullover` and `jumper`.

1. From the list of query rewrite options, select **Synonym**. A form will appear:

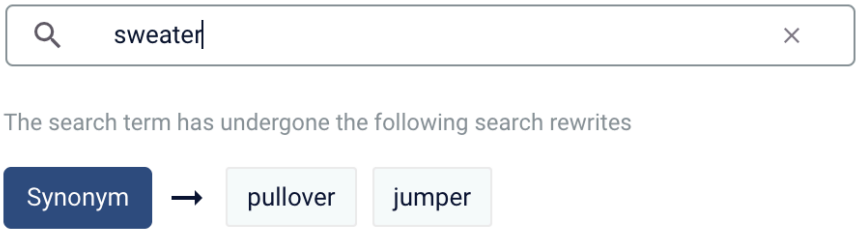


This form contains the following fields:

Parameter	Description	Example Value
Surface Form	The term that has synonyms.	`sweater`
Direction	With a **oneway** search, the original search term is replaced by the synonym. In the example above, `sweater` would be replaced by the alternative words `pullover` and `jumper`. With a **symmetric** search, the search query is expanded to include the original term and the synonyms, resulting in a greater number of potential hits. In the example above, this time the query would include `sweater`, `pullover`, and `jumper`.	`symmetric`
Synonym Mappings	Synonyms for the surface form.	`pullover`, `jumper`
Confidence	Confidence score from the phrase job. A confidence level of `1` represents 100% confidence. For rules created automatically via machine learning, the confidence level will reflect the output from the machine learning model.	`1`
Tags	Optional metadata tags that can be used to identify and organize rewrites.	`blackfridaysale`
Count	How many times this term occurred in the signal data when it was discovered. This value is optional when a rewrite is being defined manually.	`5`

2. Choose whether the direction is **oneway** or **symmetric**.
3. Enter one or more alternative words in the **Synonym Mappings** field.
4. Click the **Save** button.

The search is re-run using the synonyms, and a Search box information panel drops down to show that the new query rewrite has fired:



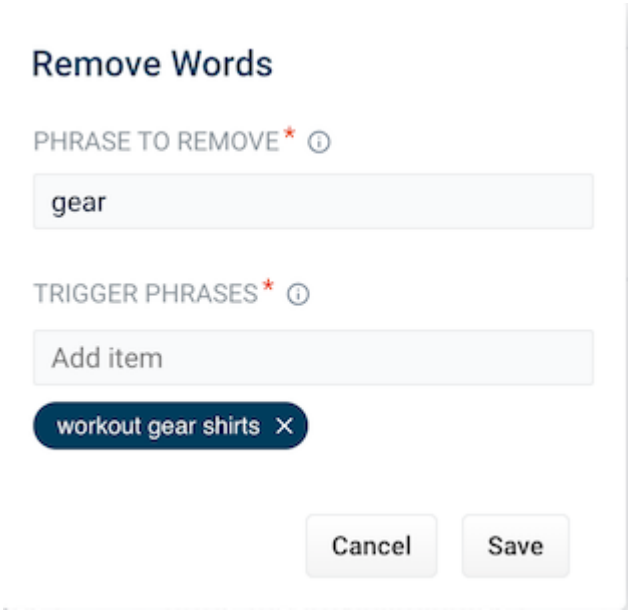
Remove Words

> **Note**
>
> The Remove Words feature is available in Fusion 5.4 and later.

You can create a Remove Words query rewrite to remove words from a query. This query rewrite is helpful when a word in the search query does not add value to the search results.

For example, you can rewrite a search query for `case study examples` to remove `examples` and then display results for `case study`.

1. From the list of query rewrite options, select Remove Words. A form appears:

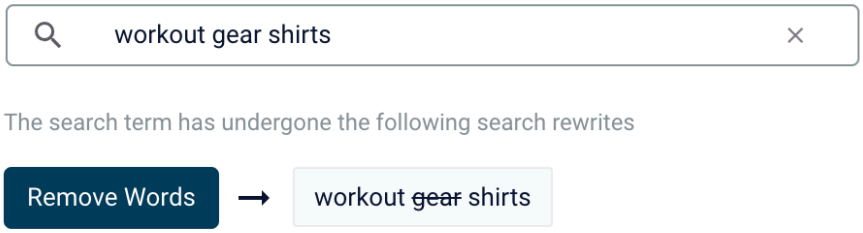


This form contains the following fields:

Parameter	Description	Example Value
Phrase to remove	The words to remove from the trigger phrase.	`examples`
Trigger phrases	The query that prompts the removal of the phrase. The trigger phrase is not necessarily a complete query. If the query contains the trigger phrase, then Fusion removes the phrase in the Phrase to Remove field.	`case study examples`

2. Enter a phrase to remove and a trigger phrase. Note that the phrase to remove is auto-populated with the query.
3. Click **Save**.

The search is re-run, and a Search box information panel drops down to show that the new query rewrite has fired:

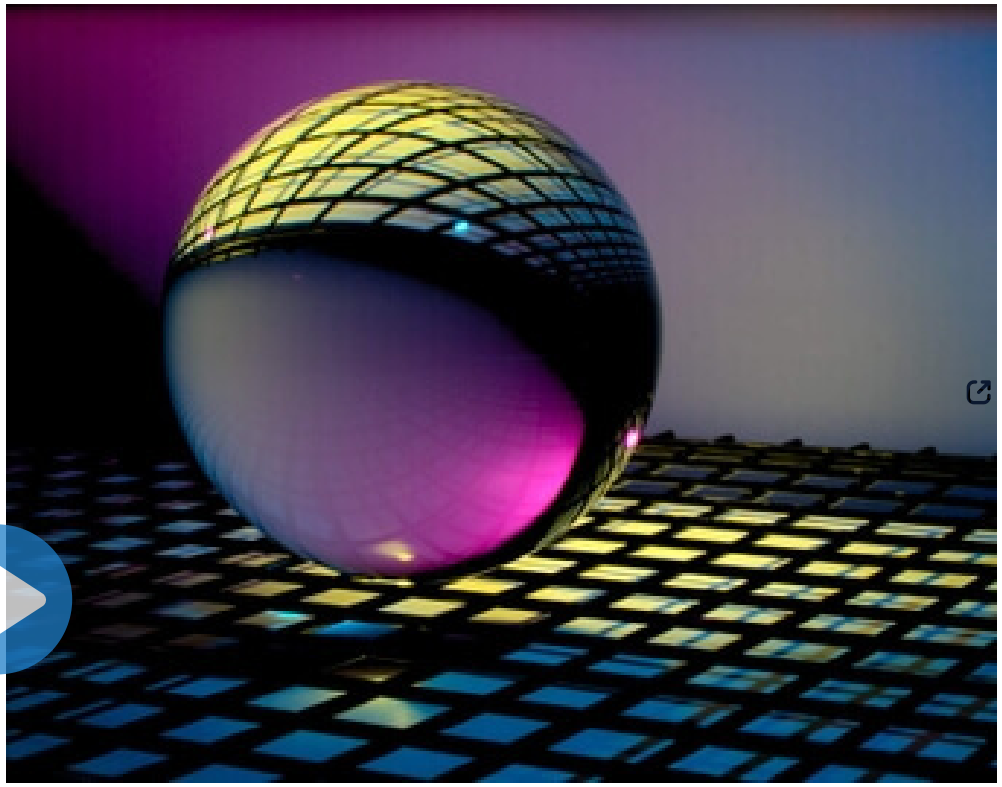


Additional resources

LucidAcademy

Lucidworks offers free training to help you get started.

The **Course** for **Predictive Merchandiser: Rewrites Manager** focuses on using search rewrites to boost conversions and create better customer experiences by improving your relevancy:



Visit the LucidAcademy to see the full training catalog.

Use Predictive Merchandiser Tasks

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion*

Start a task

1. Click the **Start Task** button:

This enables Interactive Mode. In Interactive Mode, you can interact directly with some of the widgets on the Predictive Merchandiser board, creating and managing business rules as you simulate your customers' searches. You can:

- Pin, block, boost and bury products.
- Set additional facets.
- Create search rewrites from poorly performing search terms.
- See which rules and search rewrites are triggered by a search, then edit or delete them as needed.
- Sort search results by ascending or descending order, or by relevance.
- Check product information.

Undo changes in a task

1. Click the **Undo All** button:

. A confirmation dialog box will appear.

2. Select **Undo** to confirm.

Exit a task

Exiting a task will

1. Click the **Exit** button:

Publish a task

You can publish any changes you make to rules back to the Fusion data source where they can impact live queries.

1. Click the **Publish** button:

. A confirmation dialog box will appear.

2. Select **Publish** to confirm.

Note

Predictive Merchandiser counts unpublished changes in the top navigation bar:

Use Predictive Merchandiser Tasks

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

Start a task

1. Click the **Start Task** button:

This enables Interactive Mode. In Interactive Mode, you can interact directly with some of the widgets on the Predictive Merchandiser board, creating and managing business rules as you simulate your customers' searches. You can:

- Pin, block, boost and bury products.
- Set additional facets.
- Create search rewrites from poorly performing search terms.
- See which rules and search rewrites are triggered by a search, then edit or delete them as needed.
- Sort search results by ascending or descending order, or by relevance.
- Check product information.

Undo changes in a task

1. Click the **Undo All** button:

. A confirmation dialog box will appear.

2. Select **Undo** to confirm.

Exit a task

Exiting a task will

1. Click the **Exit** button:

Publish a task

You can publish any changes you make to rules back to the Fusion data source where they can impact live queries.

1. Click the **Publish** button:

. A confirmation dialog box will appear.

2. Select **Publish** to confirm.

Note

Predictive Merchandiser counts unpublished changes in the top navigation bar:

Use Predictive Merchandiser templates

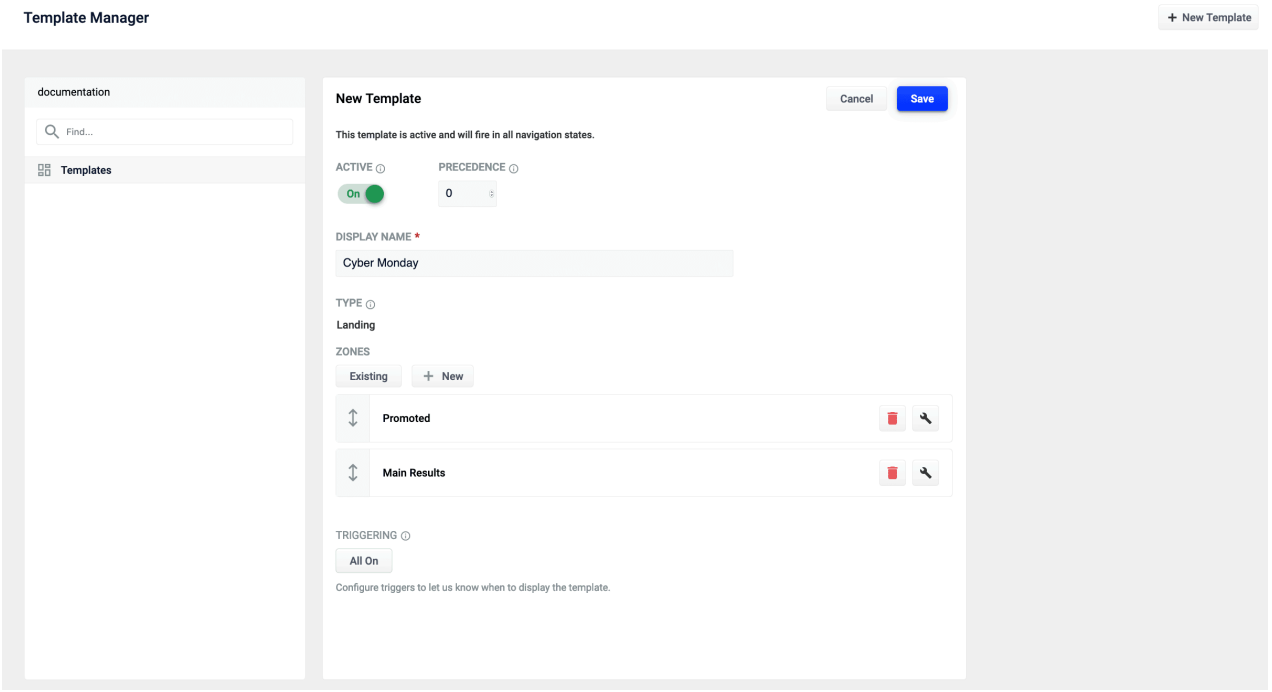
How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

This article teaches you how to use the *Templates* feature in Predictive Merchandiser, including how to:

- *Create a template*
- *Add zones* to the template
- Conditionally *trigger the template*
- *Test* the template in the Predictive Merchandiser screen

In the instructions below, a new template is created with two zones for use during a Cyber Monday sales event. One zone is used to display general search results while the other is used to promote specific products for the sale.



The zones used in this article use different query profiles. To follow the instructions as described, you will need to begin with an app that has at least two query profiles.

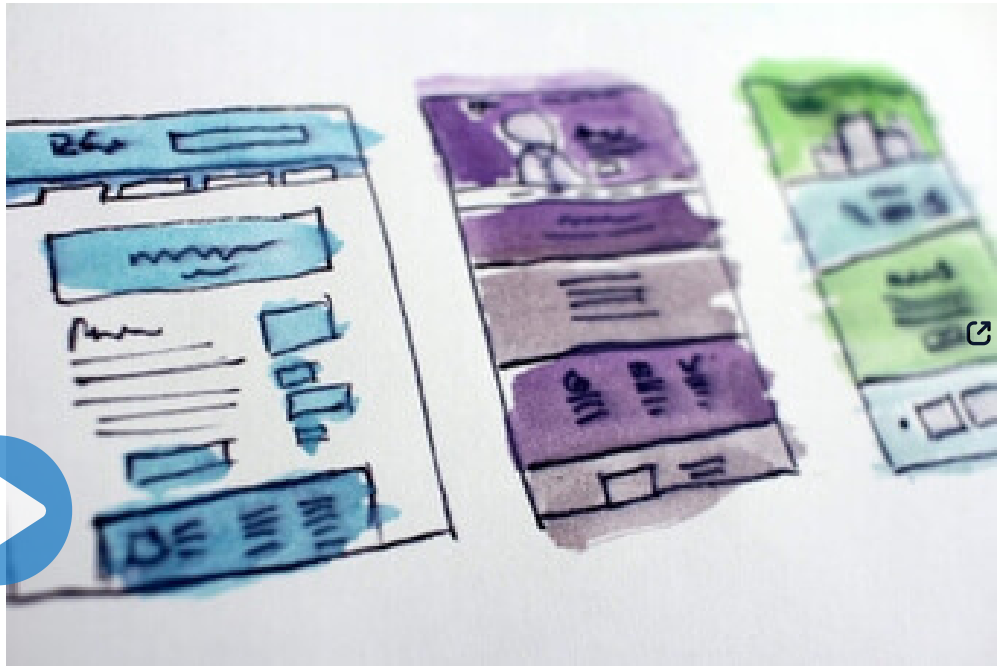
Note

If you need to create a new template type, see *Create a custom Predictive Merchandiser template*.

LucidAcademy

Lucidworks offers free training to help you get started.

The **Course** for **Predictive Merchandiser: Template Manager** focuses on how to use the templates included in Predictive Merchandiser and create your own custom templates:



Visit the LucidAcademy to see the full training catalog.

Create a new template

1. Sign in to Managed Fusion and click your application.
2. Click **Relevance > Rules > Templates > New Template**.
3. In the Template Name field, enter a unique name. In this example, the value `Cyber Monday` is assigned.
4. Select a value in Template Type. This example uses the `landing` template type.
   This value is read-only and for informational purposes only.
5. Click **Save**.

For detailed configuration information, see:

- *Templates UI configuration options*
- *Templating API configuration reference*

Add a zone to a template

Zones enable Managed Fusion users to create rules designed for specific query profiles. Combined with templates, this allows a search application to display results from multiple query profiles in the same view.

In this example, two new zones are created for the Cyber Monday template.

1. Click **Existing** to add an existing zone. In this example, the Main Results List zone is added by selecting the zone and clicking **Apply**.

> **Note**
>
> The Main Results List zone is the default zone. This zone can be edited but cannot be deleted.

2. Click **+New** to add a new zone.
3. In the Display Name field, enter a unique name for the zone. For this example, the value is Promoted.
4. In the Query Profile file, select a value. This example uses a query profile named promotional-results. This query profile is configured with rules that *pin products* to the top of the results.
5. In the Zone Type field, select a value. This example uses main-results-list.
   This value is read-only and for informational purposes only.
6. Set the configuration options for the product image, title, and description per your preference.
7. Click **Save**.

For detailed zone configuration details, see *Zone configuration options*.

Edit an existing zone

To edit a zone that is already added to a template, complete the following:

1. Click **Start Task**.
2. Click the pencil icon to edit the page.
3. Click the wrench icon to edit the zone.

> **Note**
>
> To delete a zone, click the trash can icon.

4. Edit the zone as needed and click **Save**.

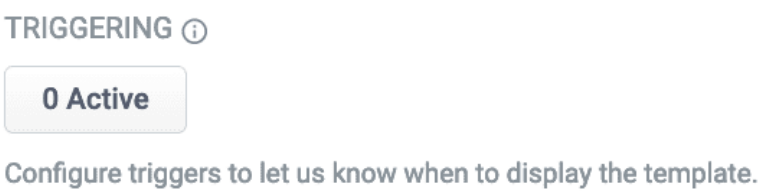
Set triggering options

Triggers describe conditions that must be met for a template to display. Trigger options include a set date range and specific search terms.

For more information, see *Triggering configuration options*.

Date range

1. In the Triggering section, click **0 Active**.



> **Note**
>
> If there are active triggers, this button changes to reflect that number. For example, this button will read **3 Active** when 3 triggers are active.

2. Click the date range selector and select a start and end date. For this example, choose a date range that includes today's date.
3. Click **Save**.

Search terms

1. In the Triggering section, click **0 Active**.
2. Click **Trigger**.
3. In the Search Terms field, enter `cyber monday`.
4. Set the Match Mode field to `phrase`.
5. Click **Add**.
6. Repeat the process to add a second trigger with Search Terms set to `sales event` and Match Mode to `phrase`.
7. Click **Save**.

Test the template

1. Access Predictive Merchandiser.
2. Enter one of the *search terms* you configured in set triggering options. This example used the search terms `cyber monday` and `sales event`.

> **Note**
>
> If today's date does not fall within the template's *date range trigger*, the template will *not* load.

If everything is configured correctly, the Cyber Monday template is loaded. You can now create different rules for both zones, including *pinning* promoted products in the Promoted zone.

Use Predictive Merchandiser Templates

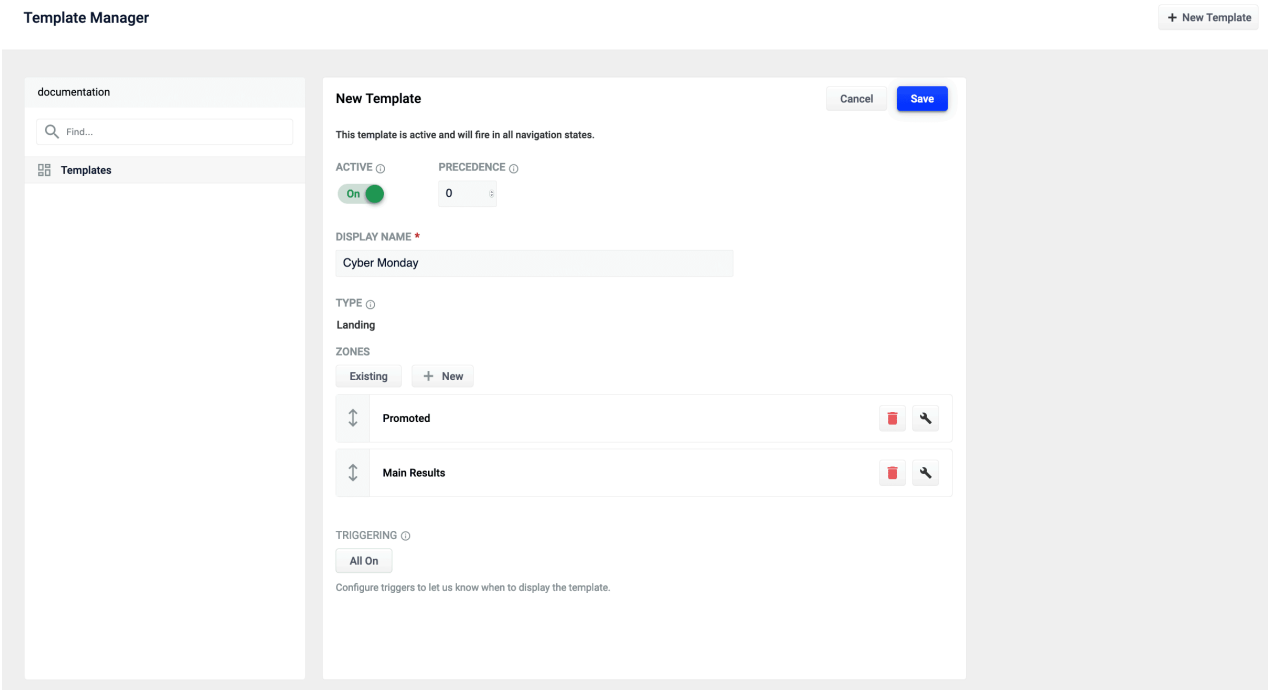
How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion*

This article teaches you how to use the *Templates* feature in Predictive Merchandiser, including how to:

- *Create a template*
- *Add zones* to the template
- Conditionally *trigger the template*
- *Test* the template in the Predictive Merchandiser screen

In the instructions below, a new template is created with two zones for use during a Cyber Monday sales event. One zone is used to display general search results while the other is used to promote specific products for the sale.



Note

The zones used in this article use different query profiles. To follow the instructions as described, you will need to begin with an app that has at least two query profiles.

Tip

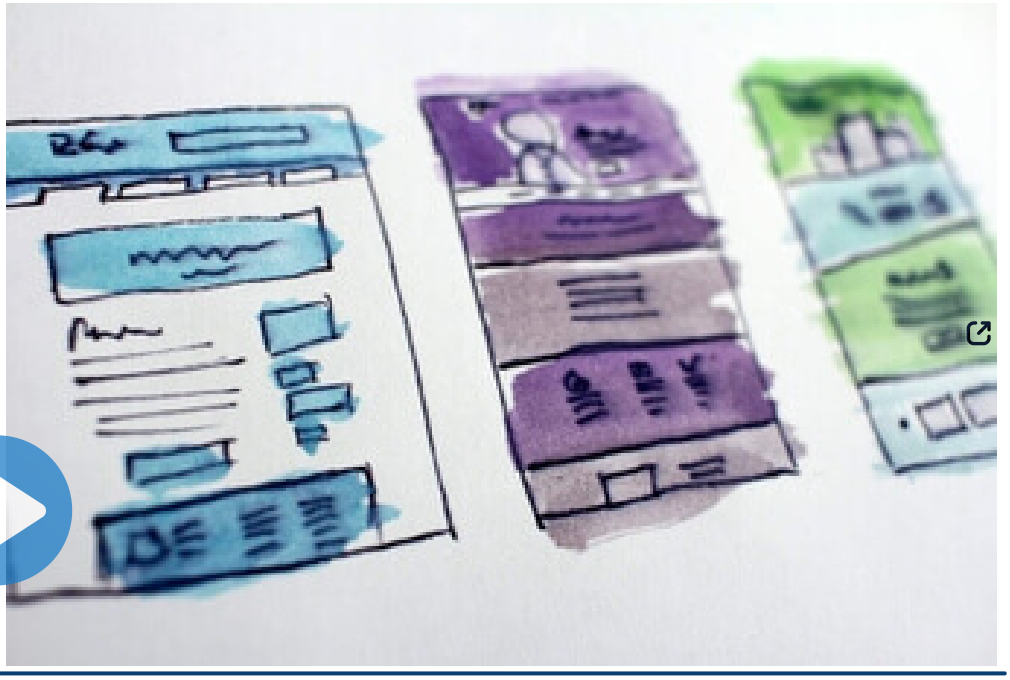
Fusion 5.4+ adds custom template types. To learn how to create one, see *Create a Custom Predictive Merchandiser Template*.

To begin, access Templates in the *Rules Editor* by navigating to **Relevance** > **Rules** > **Templates**.

LucidAcademy

Lucidworks offers free training to help you get started.

The **Course** for **Predictive Merchandiser: Template Manager** focuses on how to use the templates included in Predictive Merchandiser and create your own custom templates:



Visit the LucidAcademy to see the full training catalog.

Create a new template

1. From the Templates screen, click the **New Template** button: + New Template
2. Assign the template a unique name in the **Template Name** field. In this example, the value `Cyber Monday` is assigned.
3. Choose a **Template Type**. This example uses the `landing` template type. This value is read-only and for informational purposes only.
4. Click the **Save** button to save the new template: Save

For full configuration, see:

- *Templates UI Configuration Options*
- *Templating API Configuration Reference*

> **Note**
>
> Images may display other releases if the information between releases does not differ. For example, the Fusion Templating API image reflects version 5.3.5 in both the Fusion 5.2 and 5.3 releases, but reflects 5.4.0 in the Fusion 5.4 release.

Add a zone to a template

Zones enable Fusion users to create rules designed for specific query profiles. Combined with templates, this allows a search application to display results from multiple query profiles in the same view.

In this example, two new zones are created for the **Cyber Monday** template.

1. Add an existing zone to the template by clicking the **Existing** button. In this example, the **Main Results List** zone is added by selecting the zone and clicking the **Apply** button.

1. Create a new zone by clicking the **New** button:

2. Assign the zone a unique name in the **Display Name** field. For this example, the value `Promoted` is assigned.
3. Choose the **Query Profile** for this zone. This example uses a query profile named **promotional-results**. This query profile is configured with rules that *pin products* to the top of the results.
4. Choose the **Zone Type** for this zone. This example uses **main-results-list**.
   This value is read-only and for informational purposes only.
5. Set the configuration options for the product image, title, and description per your preference.

6. Click the **Save** button to save the new zone:

For full zone configuration details, see *Zone Configuration Options*.

Edit an existing zone

You can edit a zone that was already added to a template:

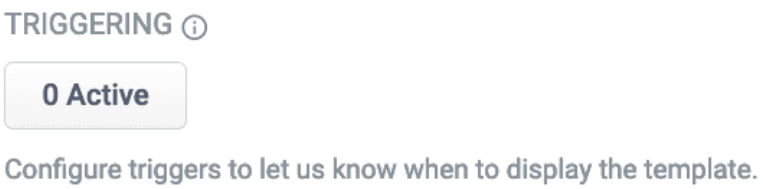
1. Select the **Start Task** button.
2. Select the pencil icon to begin editing the chosen page.
3. Each zone has a trash can icon to delete the zone. Select the wrench icon to edit the zone.
4. In the form that appears, make any necessary edits.
5. Click the **Save** button to save your changes.

Set triggering options

Triggers describe conditions that must be met for a template to display. Trigger options include a set date range and specific search terms. See *Triggering Configuration Options* for additional details.

Date range

1. Click the **0 Active** button under **Triggering**:



> **Note**
>
> If there are active triggers, this button changes to reflect that number. For example, this button will read **3 Active** when 3 triggers are active.

2. Click the date range selector and select a start and end date. For this example, choose a date range that includes today's date.

3. Click the **Save** button to save your changes: Save

Search terms

1. Click the **0 Active** button under **Triggering**.

2. Click the **Trigger** button: + Trigger

3. In the **Search Terms** field, enter `cyber monday`.
4. Set the **Match Mode** field to `phrase`.
5. Click the **Add** button.
6. Repeat the process to add a second trigger with **Search Terms** set to `sales event` and **Match Mode** to `phrase`.

7. Click the **Save** button to save your changes: Save

Test the template

1. Navigate to the **Merchandiser** screen.
2. Enter one of the *search terms* you configured in the **Set triggering options** instructions above. This example used the search terms `cyber monday` and `sales event`.

> **Note**
>
> If today's date does not fall within the template's *date range trigger*, the template will *not* load.

If everything was configured correctly, you should see the **Cyber Monday** template is loaded. You can now create different rules for both zones, including *pinning* promoted products in the **Promoted** zone.

Use query rewrites in the Rules Editor

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

This article describes how to use the Rewrites screen in the *Rules Editor*.

Query rewrite rule types

- **Head/Tail.** Replace underperforming queries with queries that produce higher click-through rates. Query improvements always require review before they are published. Schedule the Head/Tail Analysis job to get AI-generated query improvements based on your signals collection.

  Manually created Head/Tail rewrites must provide an "Improved Query" in a format appropriate to the type of request the rewrite will be applied to. For Legacy requests, this means a valid Solr query fit for use as the `q` parameter. For DSL requests this means a JSON string representing the main DSL query.

  For more information, see:
  - *Underperforming Query Rewriting*
  - *Head/Tail Analysis Jobs*

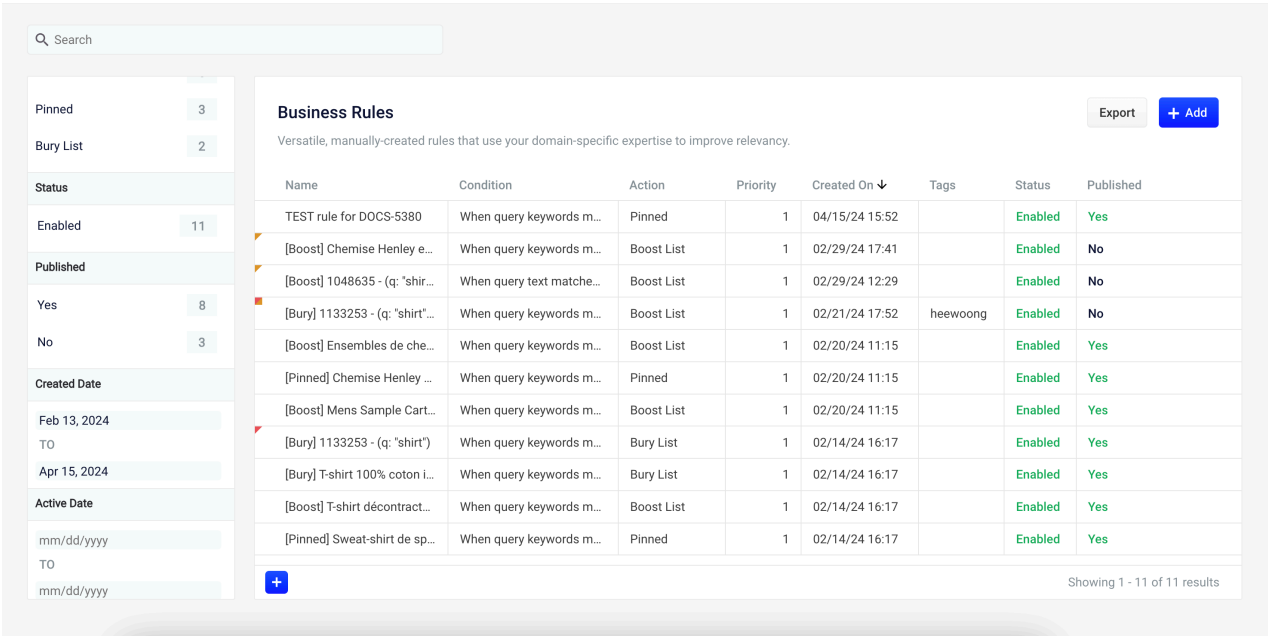
- **Misspelling.** Managed Fusion replaces misspelled query terms with their spelling corrections for better search results. Schedule the Token and Phrase Spell Correction job to get AI-generated spelling corrections based on your signals collection. See *Misspelling Detection*.
- **Phrase.** Known phrases are boosted when they appear in new queries. Schedule the Phrase Extraction job to get AI-generated phrases based on your signals collection.

  For more information, see:
  - *Phrase Detection*
  - *Use Phrase Detection*

- **Synonym.** Synonyms are used to expand queries so that they include all known synonyms of the query terms. To retrieve AI-generated synonyms based on your signals collection, schedule the *Synonym Detection job*. For more information, see *How to use synonym detection*.
- **Remove Words** - Remove particular phrases from queries. Unlike other rewrites, these should be entered manually and are not generated by a job. See *Remove Words*.

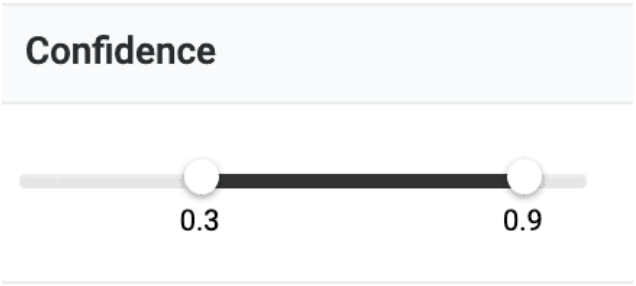
Facets

Facet menu



To facet the rules list by **Status** or **Published**, click one of the facet items within the facet categories.

To facet the rules by **Confidence**, click and drag either slider to set the range of confidence you would like to see:



Note

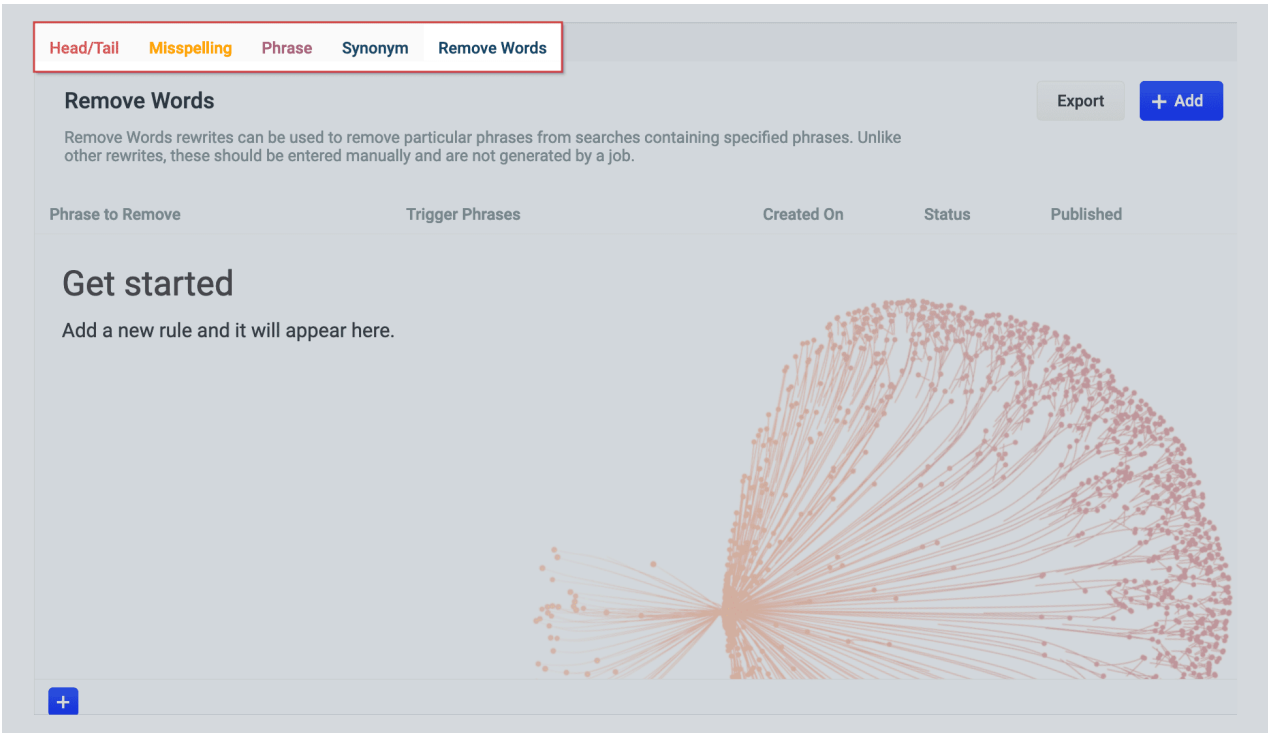
Faceting rules by confidence is not available for Head/Tail rewrites.

Search

Enter a term in the search bar to quickly facet your query rewrite rules.

Rules

Begin by selecting a query rewrite rule types to create.



Create a rule

1. Click **+Add**.
2. In the new table row, complete the required fields, and additional fields as needed.
3. Click the checkmark to save, or close to cancel the changes.

Edit a rule

1. Hold the pointer over the rule to edit.
2. Click the pencil icon.
3. Enter desired changes and click the checkmark to save, or close to cancel the changes.

Approve/deny rules

1. Hold the pointer over a rule.
2. Select the checkbox that displays for that rule.
3. To modify multiple rules, hold the pointer over each rule and select its corresponding checkbox.
4. Click **Approve** or **Deny** as needed.

Publish a rule

1. Hold the pointer over a rule.
2. Select the checkbox that displays for that rule.
3. To publish multiple rules, hold the pointer over each rule and select its corresponding checkbox.
4. Click **Publish Selected** to publish the selected rules.

PUBLISH AN INDIVIDUAL RULE

1. Hold the pointer over a rule.
2. Click the button.
3. To confirm, click **Publish**.

Delete rules

DELETE A SINGLE RULE

1. Hold the pointer over a rule.
2. Click the trash can.
3. To confirm, click **Delete**.

DELETE MULTIPLE RULES

1. Hold the pointer over a rule.
2. Select the checkbox that displays for that rule.
3. To delete multiple rules, hold the pointer over each rule and select its corresponding checkbox.
4. Click **Delete**.
5. To confirm, click **Delete** again.

Sort rules

1. Hold the pointer over column titles. A ↓ icon displays for some titles to indicate the rules list can be sorted by that column.
2. Click the title column to sort the rules.

> **Note**
>
> The column that is currently sorting the rules list is indicated by an ↓ or ↑ icon.

Export rules

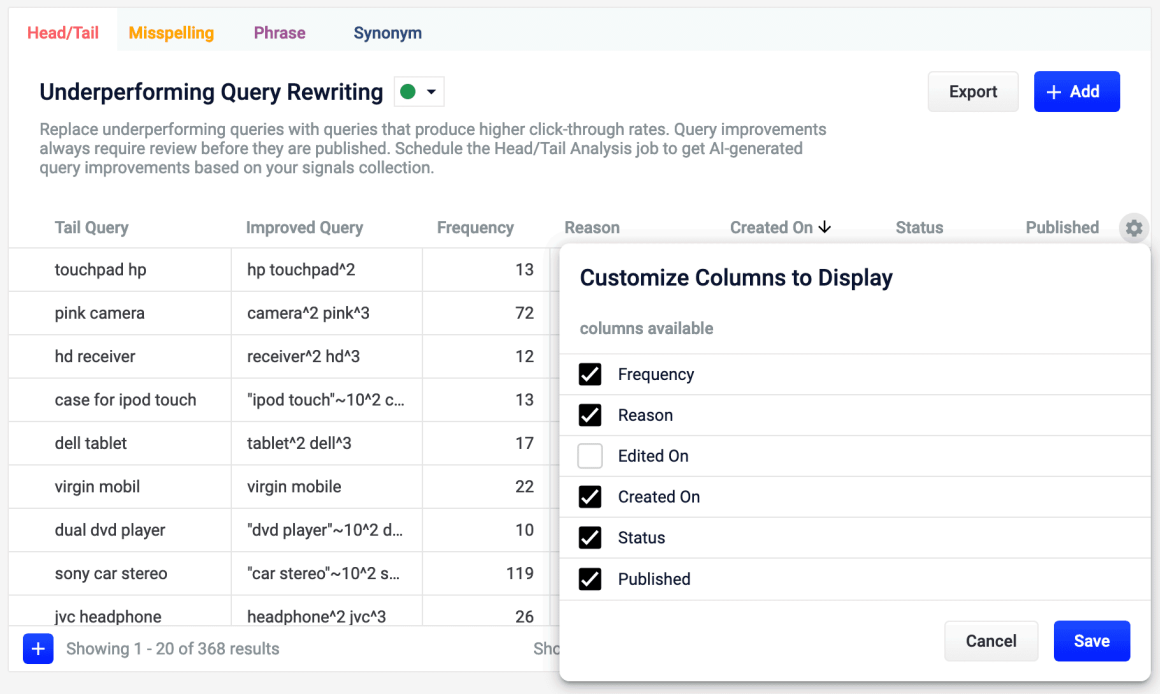
To export the rules list, click **Export**. This generates a `.csv` file that contains your rules.

Columns

Customize columns

To customize columns:

1. Hold the pointer over the row of column titles.
2. Click the ⚙ button.



3. In the Customize Columns to Display screen, select the checkboxes for the columns you want to display. Clear the checkboxes for columns you do not want to display.
4. Click **Save**. To exit without saving your changes, click **Cancel**.

Use Query Rewrites in the Rules Editor

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion*

This article describes how to use the Rewrites screen in the *Rules Editor*.

Query rewrite rule types

- **Head/Tail.** Replace underperforming queries with queries that produce higher click-through rates. Query improvements always require review before they are published. Schedule the Head/Tail Analysis job to get AI-generated query improvements based on your signals collection.

  Manually created Head/Tail rewrites must provide an "Improved Query" in a format appropriate to the type of request the rewrite will be applied to. For Legacy requests, this means a valid Solr query fit for use as the `q` parameter. For DSL requests this means a JSON string representing the main DSL query.

  For more information, see:
  - *Underperforming Query Rewriting*
  - *Head/Tail Analysis Jobs*

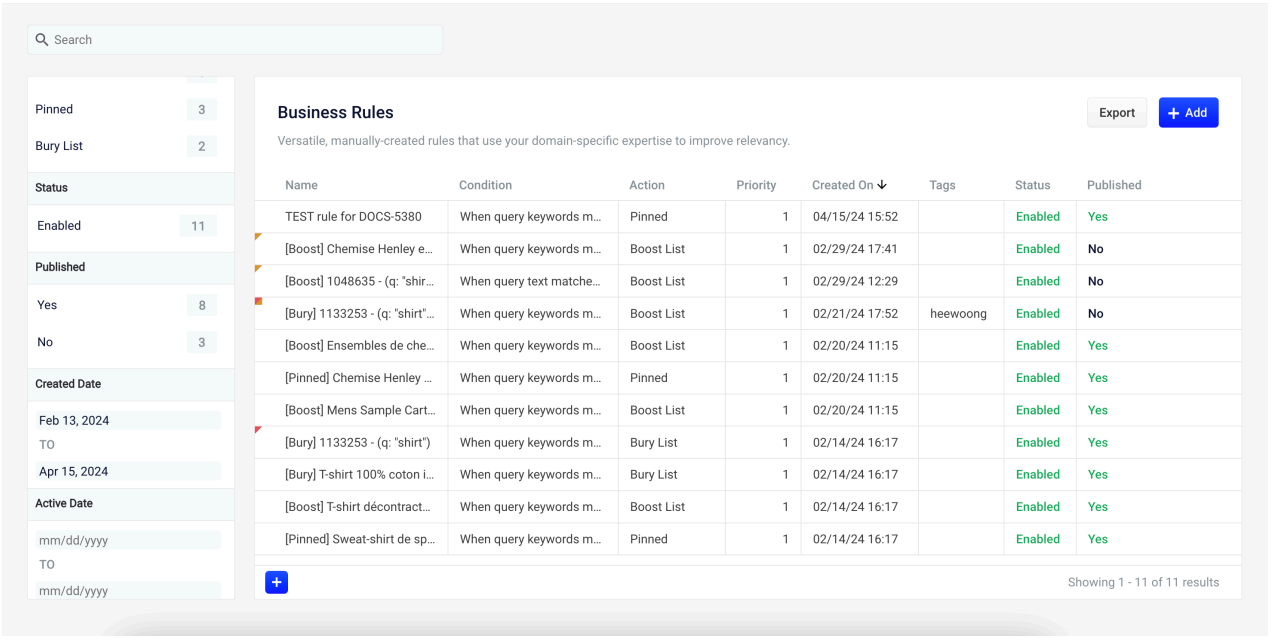
- **Misspelling.** Fusion replaces misspelled query terms with their spelling corrections for better search results. Schedule the Token and Phrase Spell Correction job to get AI-generated spelling corrections based on your signals collection. For more information, see *Misspelling Detection*.
- **Phrase.** Known phrases are boosted when they appear in new queries. Schedule the Phrase Extraction job to get AI-generated phrases based on your signals collection.

  For more information, see:
  - *Phrase Detection*
  - *Use Phrase Detection*

- **Synonym.** Synonyms are used to expand queries so that they include all known synonyms of the query terms. To retrieve AI-generated synonyms based on your signals collection, schedule the *Synonym Detection job*. For more information, see *Use Synonym Detection*.
- **Remove Words** - Remove particular phrases from queries. Unlike other rewrites, these should be entered manually and are not generated by a job. For more information, see *Remove Words*.

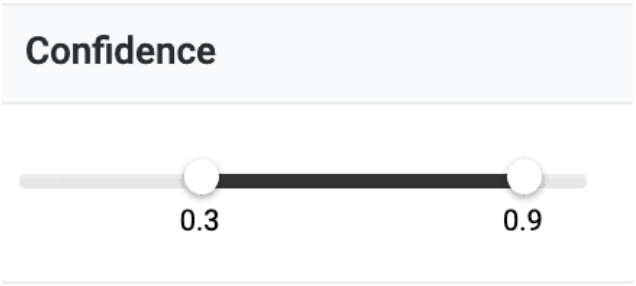
Facets

Facet menu



To facet the rules list by **Status** or **Published**, click one of the facet items within the facet categories.

To facet the rules by **Confidence**, click and drag either slider to set the range of confidence you would like to see:



Note

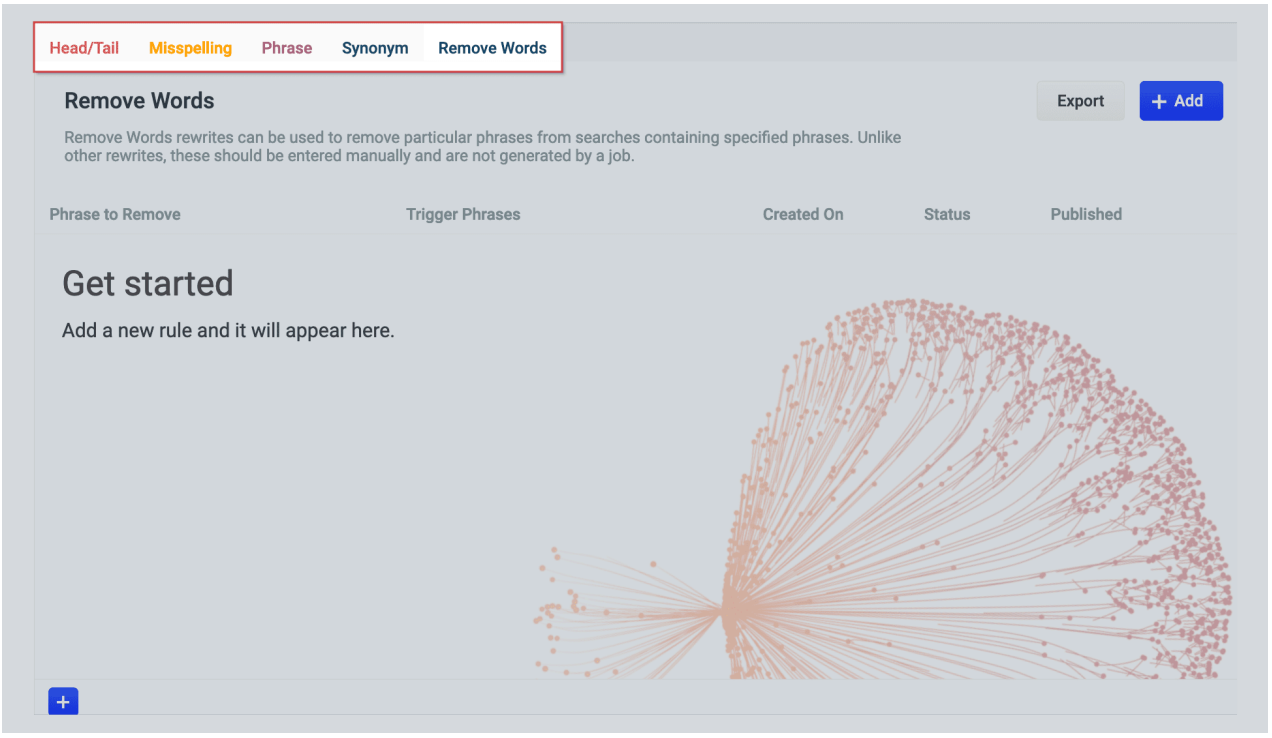
Faceting rules by confidence is not available for Head/Tail rewrites.

Search

Enter a term in the search bar to quickly facet your query rewrite rules.

Rules

Begin by selecting one of the query rewrite rule types you will create.



Create a rule

1. Click **+Add**.
2. In the new table row, complete the required fields, and additional fields as needed.
3. Click the checkmark to save, or close to cancel the changes.

Edit a rule

1. Hold the pointer over the rule to edit.
2. Click the pencil icon.
3. Enter desired changes and click the checkmark to save, or close to cancel the changes.

Approve/deny rules

1. Hold the pointer over a rule.
2. Select the checkbox that displays for that rule.
3. To modify multiple rules, hold the pointer over each rule and select its corresponding checkbox.
4. Click **Approve** or **Deny** as needed.

Publish a rule

1. Hold the pointer over a rule.
2. Select the checkbox that displays for that rule.
3. To publish multiple rules, hold the pointer over each rule and select its corresponding checkbox.
4. Click **Publish Selected** to publish the selected rules.

PUBLISH AN INDIVIDUAL RULE

1. Hold the pointer over a rule.
2. Click the button.
3. To confirm, click **Publish**.

Delete rules

DELETE A SINGLE RULE

1. Hold the pointer over a rule.
2. Click the trash can.
3. To confirm, click **Delete**.

DELETE MULTIPLE RULES

1. Hold the pointer over a rule.
2. Select the checkbox that displays for that rule.
3. To delete multiple rules, hold the pointer over each rule and select its corresponding checkbox.
4. Click **Delete**.
5. To confirm, click **Delete** again.

Sort rules

1. Hold the pointer over column titles. A ↓ icon displays for some titles to indicate the rules list can be sorted by that column.
2. Click the title column to sort the rules.

> **Note**
>
> The column that is currently sorting the rules list is indicated by an ↓ or ↑ icon.

Export rules

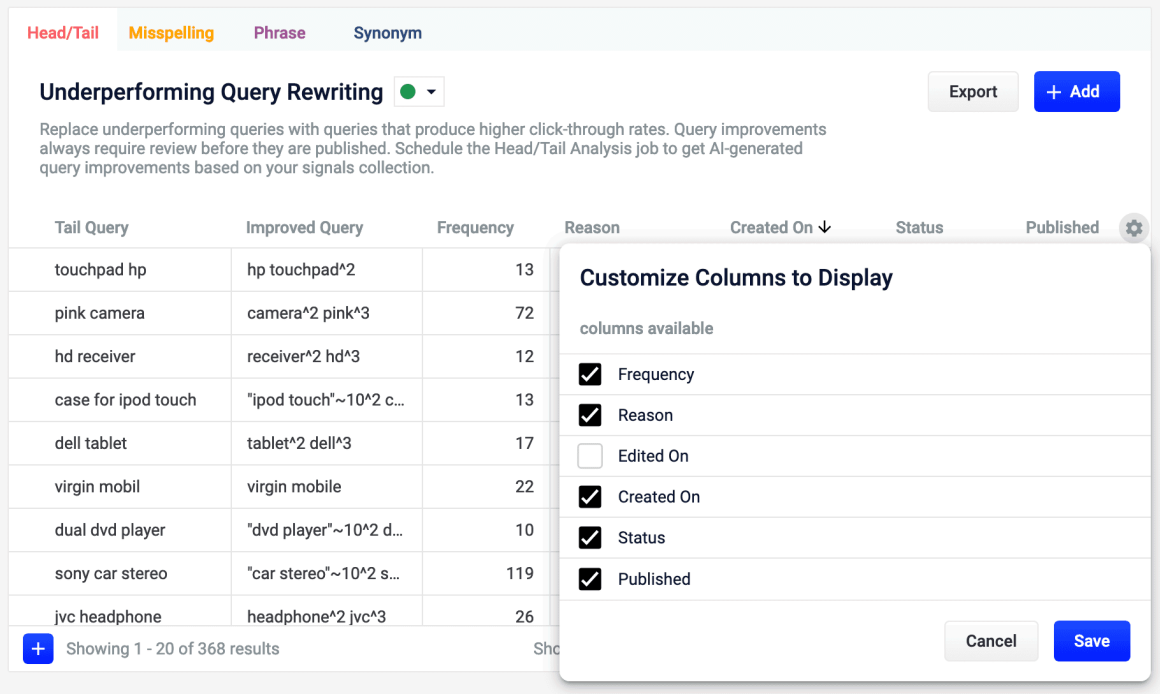
To export the rules list, click **Export**. This generates a `.csv` file that contains your rules.

Columns

Customize columns

To customize columns:

1. Hold the pointer over the row of column titles.
2. Click the ⚙ button.



3. In the Customize Columns to Display screen, select the checkboxes for the columns you want to display. Clear the checkboxes for columns you do not want to display.
4. Click **Save**. To exit without saving your changes, click **Cancel**.

Use Spark Drivers

How-to

> Compatible versions: 4.2
>
> Works with: *Fusion Server*, *Fusion*

Using the Spark default driver

Navigate to **Collections** > **Collections Manager**.

In Fusion 4.2+, select the `system_monitor` collection. For Fusion 4.0.x and 4.1.x, select the `system_metrics` collection.

Then navigate to **Collections** > **Jobs** and select one of the built-in aggregation jobs, such as `session_rollup`. In the diagram above, the spark-master service in Fusion is the Cluster Manager.

You must delete any existing driver applications before launching the job. Even if you have not started any jobs by hand, Fusion's API service might have started one automatically, because Fusion ships with built-in jobs that run in the background which perform rollups of metrics in the `system_monitor` (Fusion 4.2+) or `system_metrics` (Fusion 4.0.x and 4.1.x) collection. Therefore, before you try to launch a job, you should run the following command:

```
curl -X DELETE http://localhost:8764/api/spark/driver
```

Wait a few seconds and use the Spark UI to verify that no Fusion-Spark application (for example, `Fusion-20161005224611`) is running.

In a terminal window or windows, set up a `tail -f` job (on Unix, or the equivalent on Windows) on the `api` and `spark-driver-default` logs:

```
tail -f var/log/api/api.log var/log/api/spark-driver-default.log
```

Give this command from the `bin` directory below the Fusion home directory, for example, `/opt/fusion/latest.x` (on Unix) or `C:\lucidworks\fusion\latest.x` (on Windows).

Now, start any aggregation job from the UI. It does not matter whether or not this job performs any work; the goal of this exercise is to show what happens in Fusion and Spark when you run an aggregation job. You should see activity in both logs related to starting the driver application and running the selected job. The Spark UI will now show a Fusion-Spark app:

Running Applications

Application ID	Name	Cores	Memory per Node	Submitted Time	User	State	Duration
app-20161005164611-0003 (kill)	Fusion-20161005224611	8	2.0 GB	2016/10/05 16:46:11	timpotter	RUNNING	32 s

Use the `ps` command to get more details on this process:

```
ps waux | grep SparkDriver
```

The output should show that the Fusion SparkDriver is a JVM process started by the API service; it is not managed by the Fusion agent. Within a few minutes, the Spark UI will update itself:

URL: spark://192.168.1.9:8766
REST URL: spark://192.168.1.9:6066 *(cluster mode)*
Alive Workers: 1
Cores in use: 8 Total, 0 Used
Memory in use: 2.0 GB Total, 0.0 B Used
Applications: 1 Running, 4 Completed
Drivers: 0 Running, 0 Completed
Status: ALIVE

Workers

Worker Id	Address	State	Cores
worker-20161005115307-192.168.1.9-8769	192.168.1.9:8769	ALIVE	8 (0 Used)

Running Applications

Application ID		Name	Cores	Memory per Node	Submitted Time
app-20161005165054-0004	(kill)	Fusion-20161005225054	0	2.0 GB	2016/10/05 16:50:54

Notice that the application no longer has any cores allocated and that all of the memory available is not being used (0.0B Used of 2.0 GB Total). This is because we launch our driver applications with `spark.dynamicAllocation.enabled=true`. This setting allows the Spark master to reclaim CPU & memory from an application if it is not actively using the resources allocated to it.

Both driver processes (default and scripted) manage a SparkContext. For the default driver, the SparkContext will be shut down after waiting a configurable (`fusion.spark.idleTime`: default 5 mins) idle time. The scripted driver shuts down the SparkContext after every scripted job is run to avoid classpath pollution between jobs.

Using the Spark script-job driver

Fusion supports custom *script jobs*.

Script jobs require a separate driver to isolate the classpath for custom Scala scripts, as well as to isolate the classpath between the jobs, so that classes compiled from scripts do not pollute the classpath for subsequent scripted jobs.

For this reason, the SparkContext that each scripted job uses is immediately shut down after the job is finished and is recreated for new jobs. This adds some startup overhead for scripted jobs.

Refer to the apachelogs lab in the Fusion Spark Bootcamp project for a complete example of a custom script job.

To troubleshoot problems with a script job, start by looking for errors in the script-job driver log `spark-driver-scripted.log` in `/opt/fusion/latest.x/var/log/api/` (on Unix) or `C:\lucidworks\var\fusion\latest.x\var\log\api\` (on Windows).

Spark drivers in a multinode cluster

To find out which node is running the Spark driver which node is running the driver when running a multi-node Fusion deployment which has several nodes running Fusion's API services, you can query the driver status via the following call:

```
curl http://localhost:8764/api/spark/driver/status
```

This returns a status report:

```
{ "/spark-drivers/15797426d56T537184c2" : { "id" :
"15797426d56T537184c2", "hostname" : "192.168.1.9", "port" : 8601,
"scripted" : false } }
```

Use Superset with Fusion SQL

How-to

> Compatible versions: 5.5
>
> Works with: *Fusion*

Apache Superset is a powerful and widely used SQL exploration and visualization tool. This topic explains different ways to use it with Fusion 5.

In Fusion 5.4 and up, Superset is deployed by default as part of your Fusion platform. In earlier versions of Fusion, you must install Superset and the Fusion SQL SQLAlchemy dialect separately.

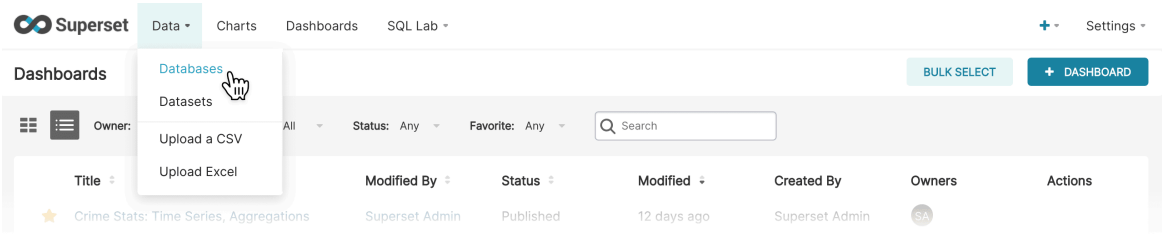
Enable and connect to Superset

Fusion 5.1-5.3	Fusion 5.4 and up
1. *Enable Superset for Fusion 5.1-5.3.* 2. Connect to the Superset UI at `fusion_sql://USERNAME:PASSWORD@<fusion_host>:<ssl_port>/fusion/default`.	1. *Enable Superset for Fusion 5.4 and up.* 2. If you configured the load balancer at the gateway when you enabled Superset, then you need to log in to the Superset gateway using your gateway credentials. 3. Log in to Superset using your Superset credentials.

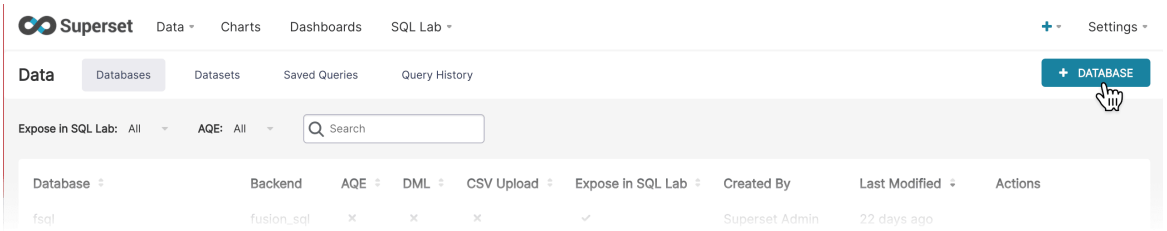
Create a Superset database

After the SQLAlchemy dialect is installed it's time to set up a connection to the database.

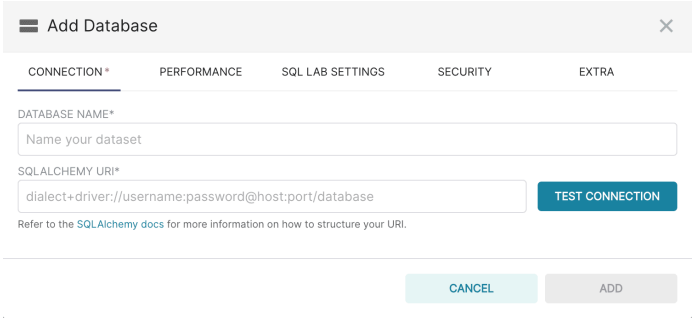
1. In Superset, navigate to **Data** > **Databases**:



2. Click the **+ Database** button in the upper right corner to create a new database:



The **Add Database** screen appears:



3. Add a name for the database.

4. Add the SQLAlchemy URI in the following format:

```
fusion_sql://USERNAME:PASSWORD@<fusiondomain>:
<fusion_ssl_port>/fusion/default
```

Note

The Superset connector uses SSL for connections and queries and will only connect to the Fusion SSL port.

If this command does not work, it is possible there are issues with reaching your Fusion proxy. Enter the following to add the SQLAlchemy URI:

```
fusion_sql://USERNAME:PASSWORD@<fusion_proxy>:
<fusion_proxy_port>/fusion/default
```

Note

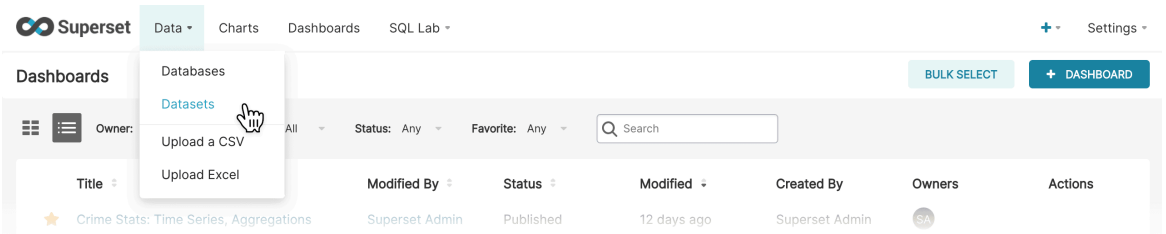
The superset SQL selects are written to the SQL Service Pod log for review and troubleshooting purposes.

5. Click **Test Connection** to make sure the connection is working.
6. Click **Add**.

Create a Superset dataset

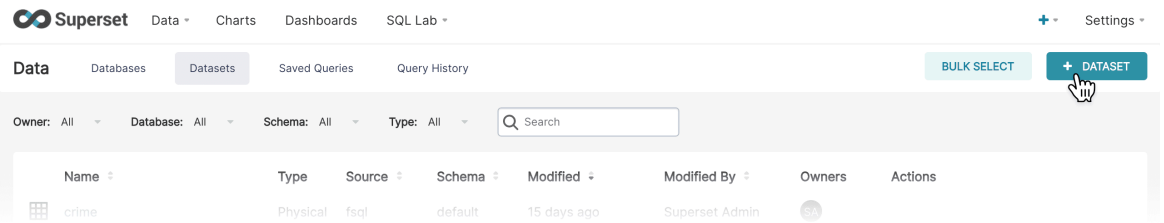
Once a database is created, datasets (also called tables) can be set up.

1. Navigate to **Data** > **Datasets**:

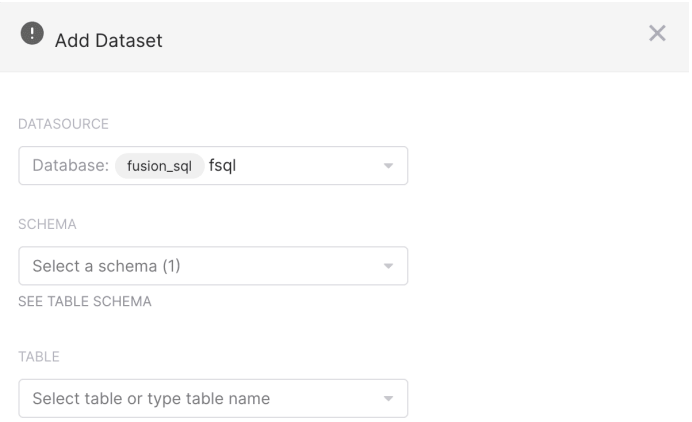


The Datasets screen appears.

2. Click the **+ Dataset** button in the upper right corner:



3. Create a dataset by specifying the **Database**, **Schema**, and **Table Name**. The schema is always “default”.

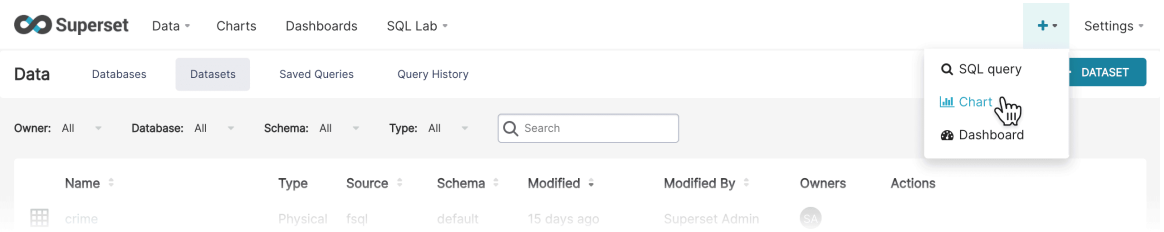


4. Click **Add** to save the dataset.

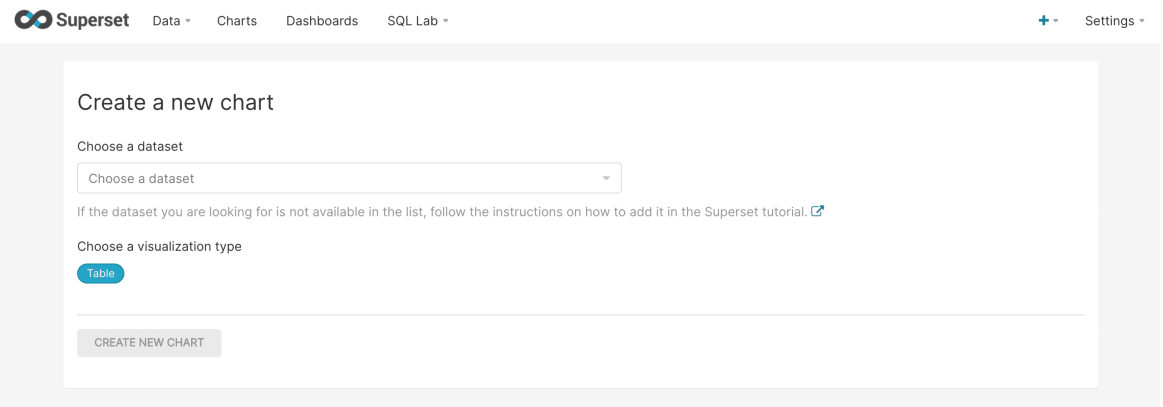
Your first chart: bar chart

In this section we’ll create our first chart. More time will be spent on the process of creating charts in this section than in following sections which will only deal with the specifics of the different kinds of charts.

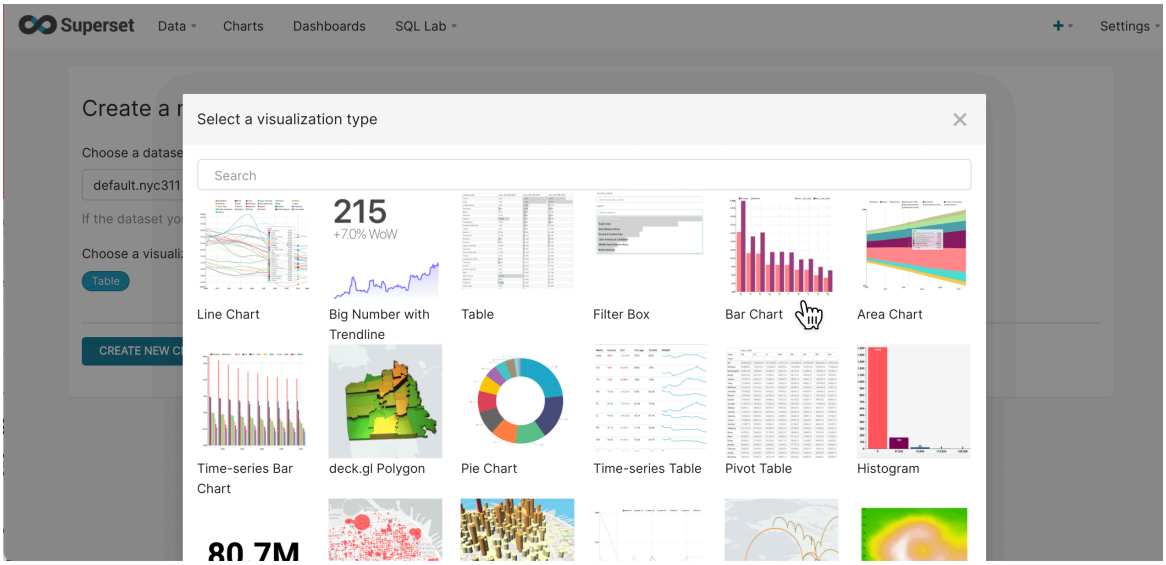
1. Navigate to **+New** > **Chart**:



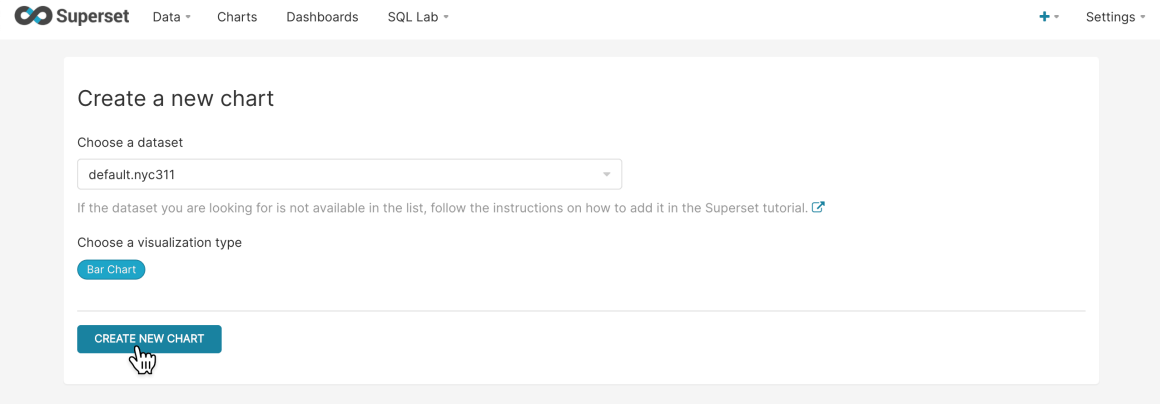
The **Create a new chart** screen appears:



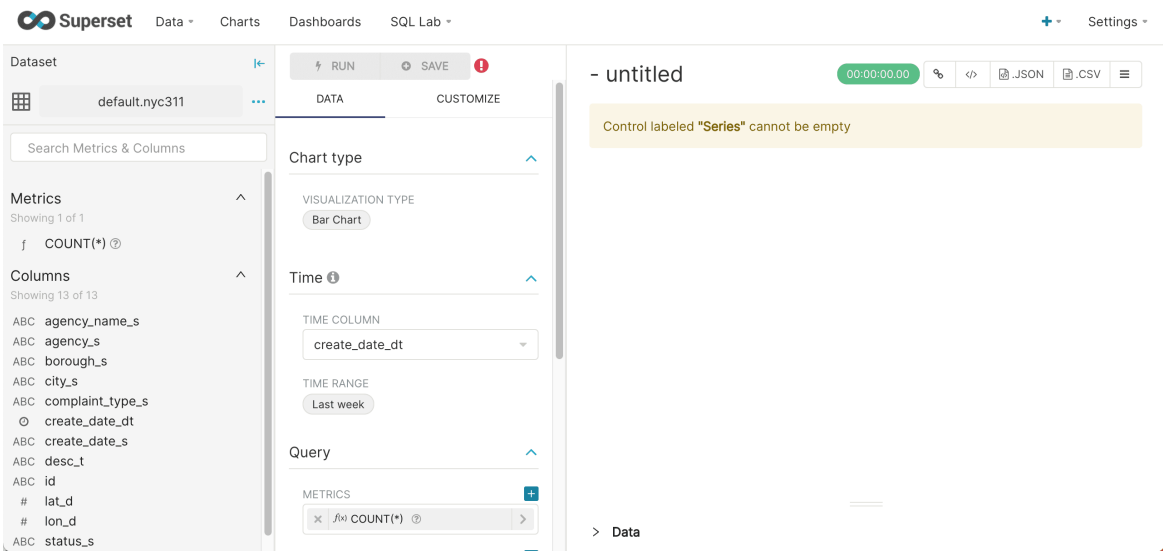
2. Select a datasource (also called a table) from which to create the chart.
3. Click **Table** to change the visualization type.
4. Select the **Bar Chart** visualization type:



5. Click **Create new chart**:

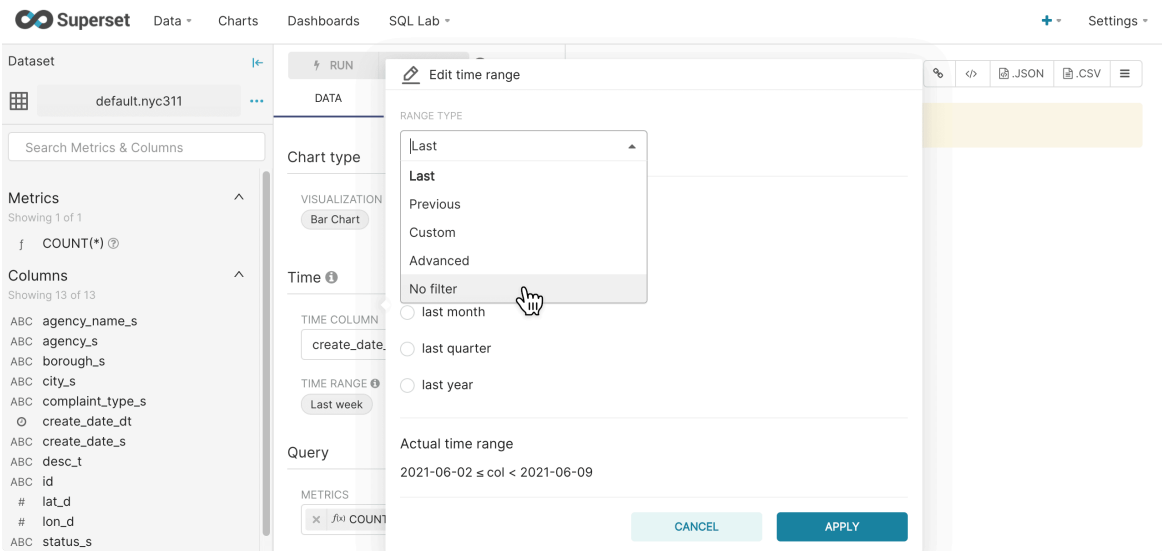


The chart configuration screen appears:

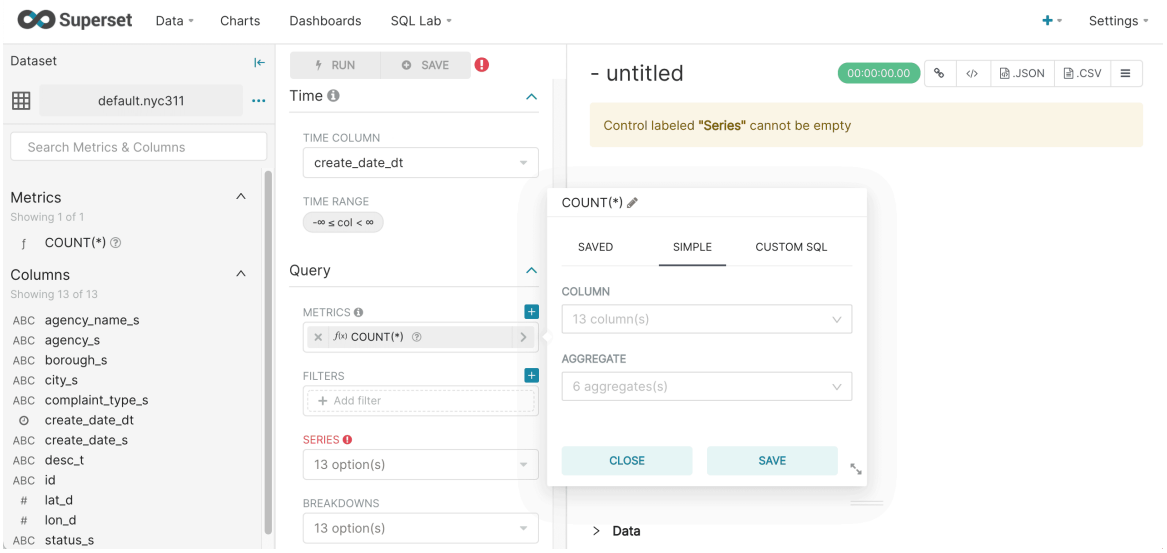


The chart can be configured using the left hand panel and will display on the right.
The first step is to turn off the **Time Range** filter. Time filters, if needed, can be added back in after seeing the initial results.

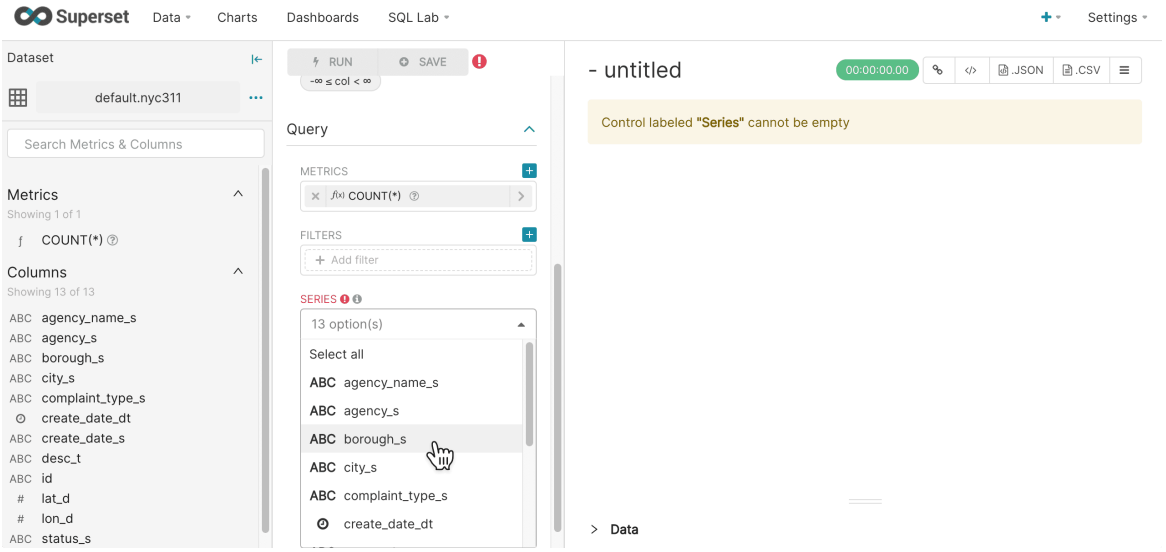
6. To turn off the Time Range filter, click **Last week**, then **Range Type**, and select **No filter**.



7. Click **Apply**.
8. Select a **Metric** from the metrics drop down.
   In this example the `COUNT(*)` metric is chosen. The metric will be the y-axis of the bar chart.

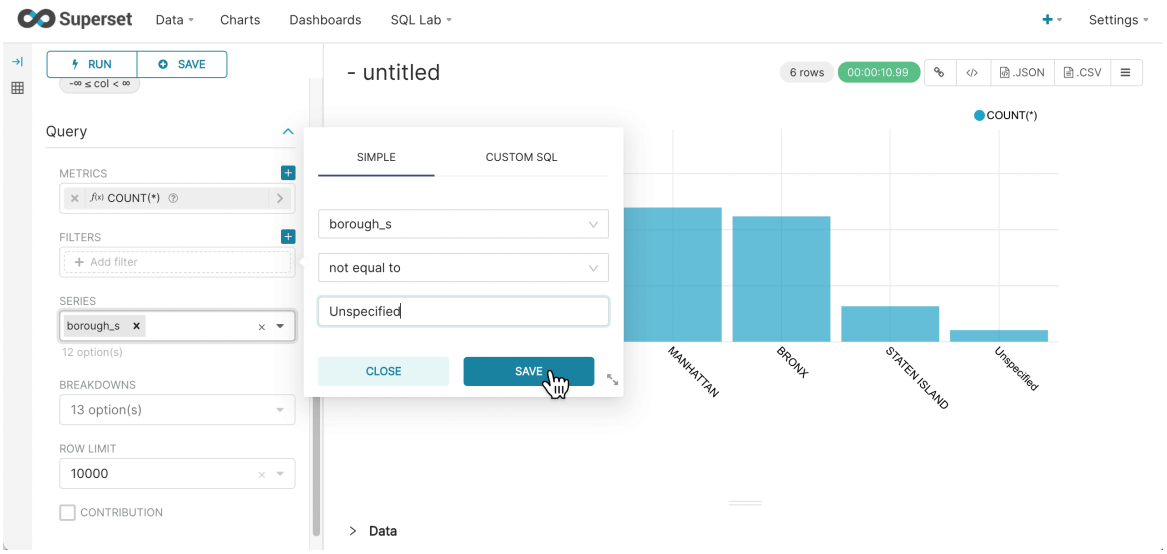


9. Select a **Series**, which will be the x-axis of the bar chart.

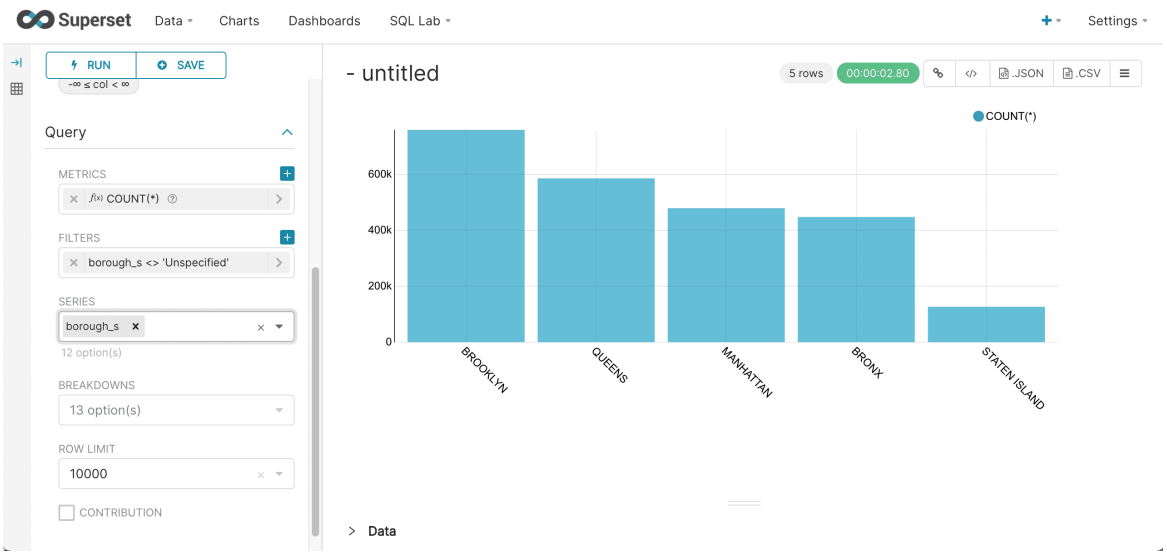


10. Click the **Run Query** button. This will execute the query.
If the query is successful, a bar chart appears in the right panel:

11. Add a filter by clicking on the **Filters** control:

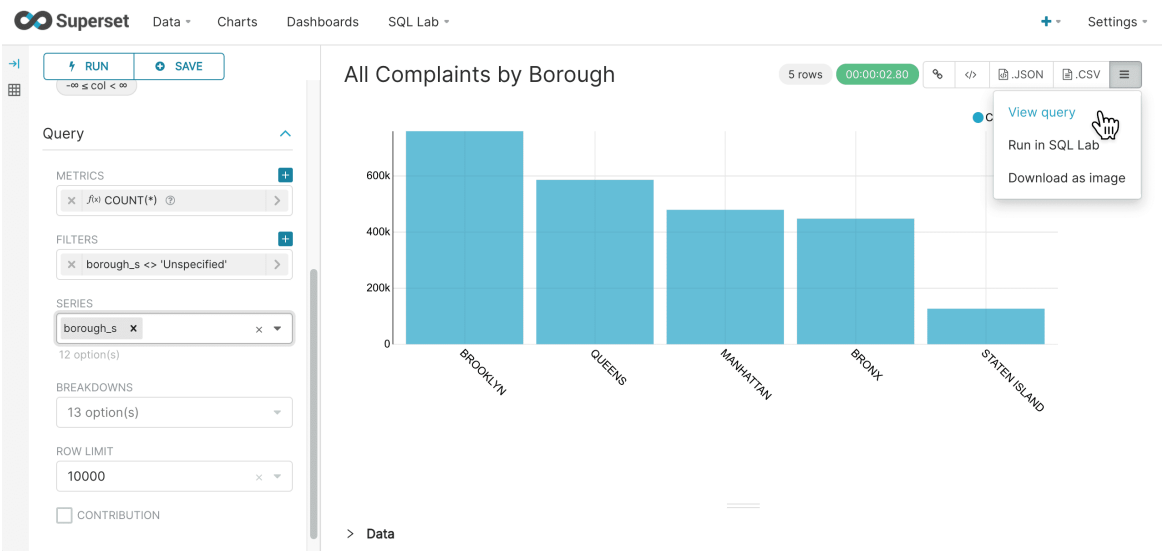


12. Click **Save**.
13. Rerun the query to see the results:

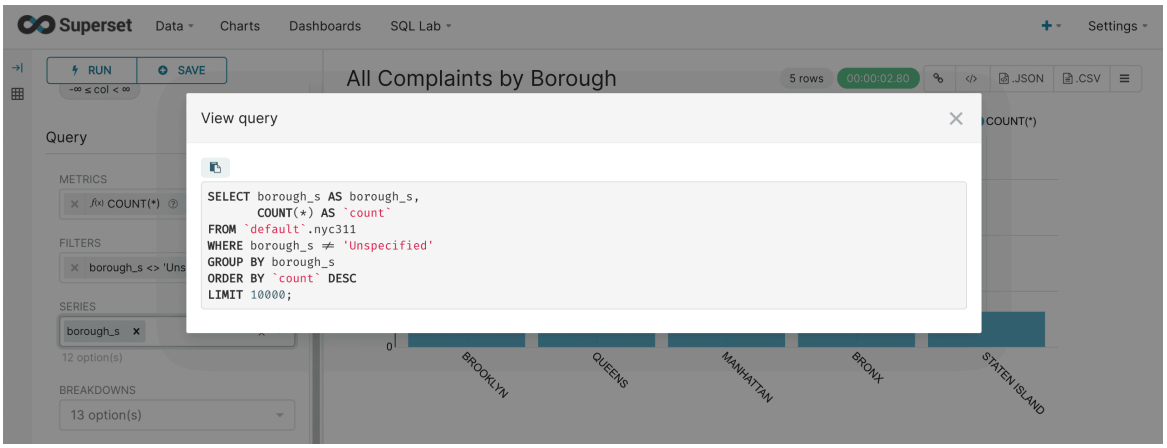


14. Click on the “Untitled” chart title to give the chart a new title:

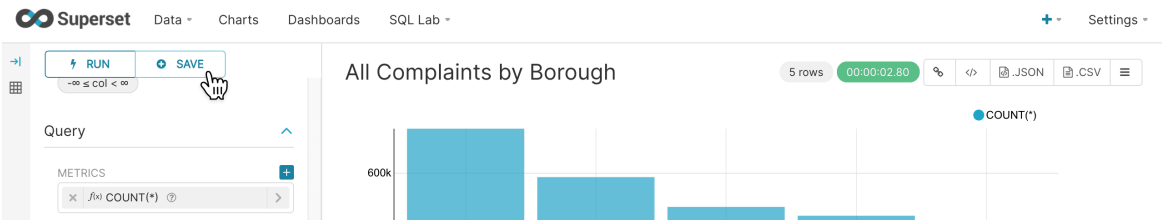
15. Select **View query** in the upper right to see the SQL query that was sent to Fusion SQL:



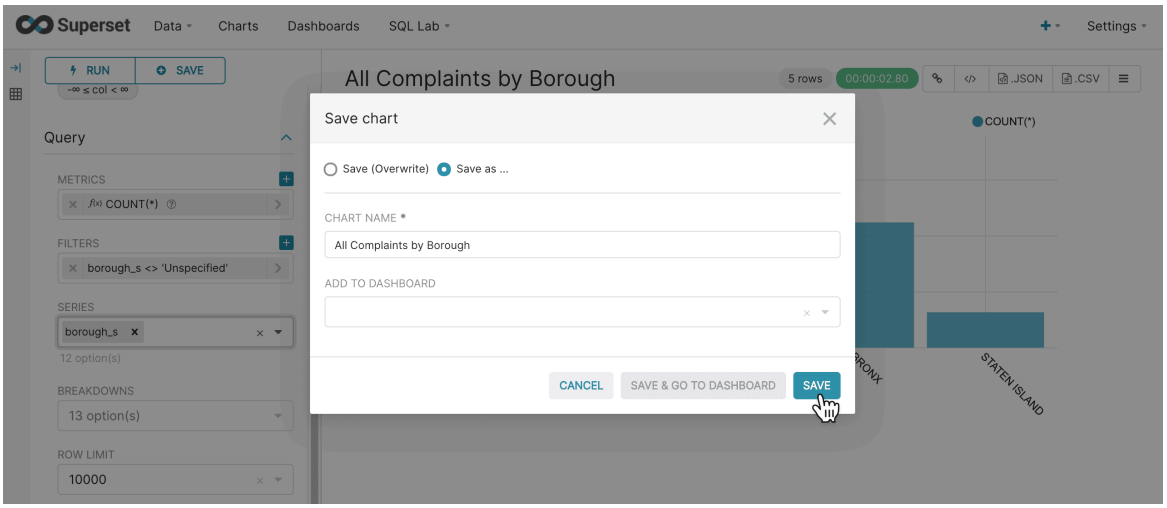
This brings up the query to review:



16. When the chart is complete, click **Save** in the upper left:



17. Enter a name for this chart and click **Save**:

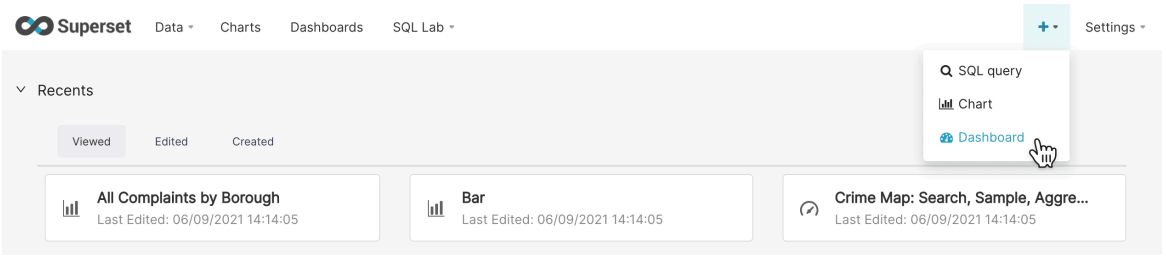


Once the chart is saved it can be added to dashboards later.

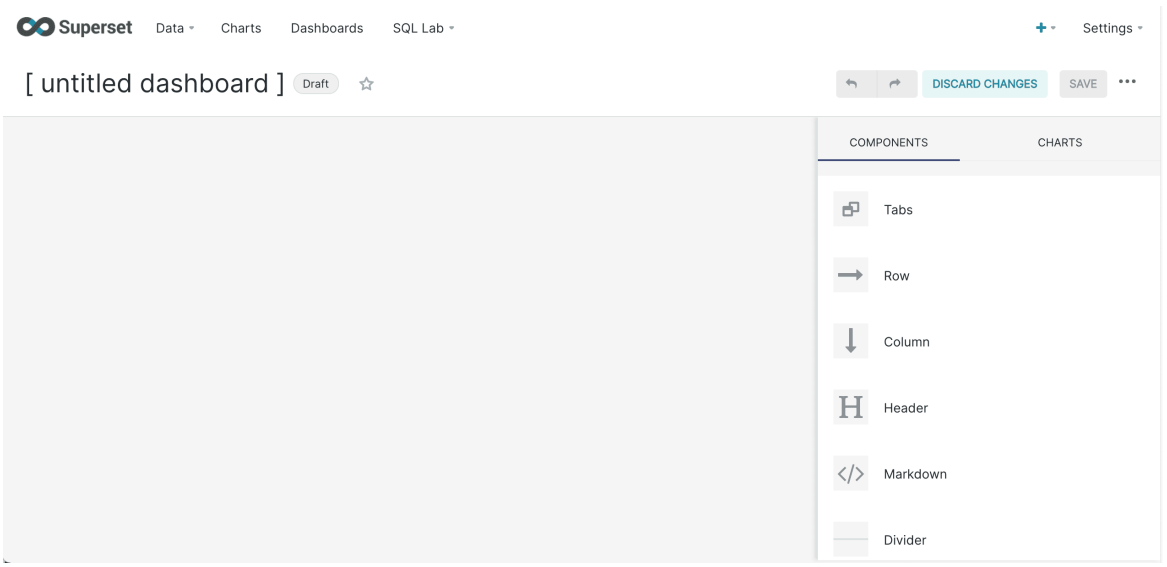
Dashboards

Once we have a chart saved we can add it to a dashboard.

1. Navigate to **+New** > **Dashboard** in the upper right hand corner:

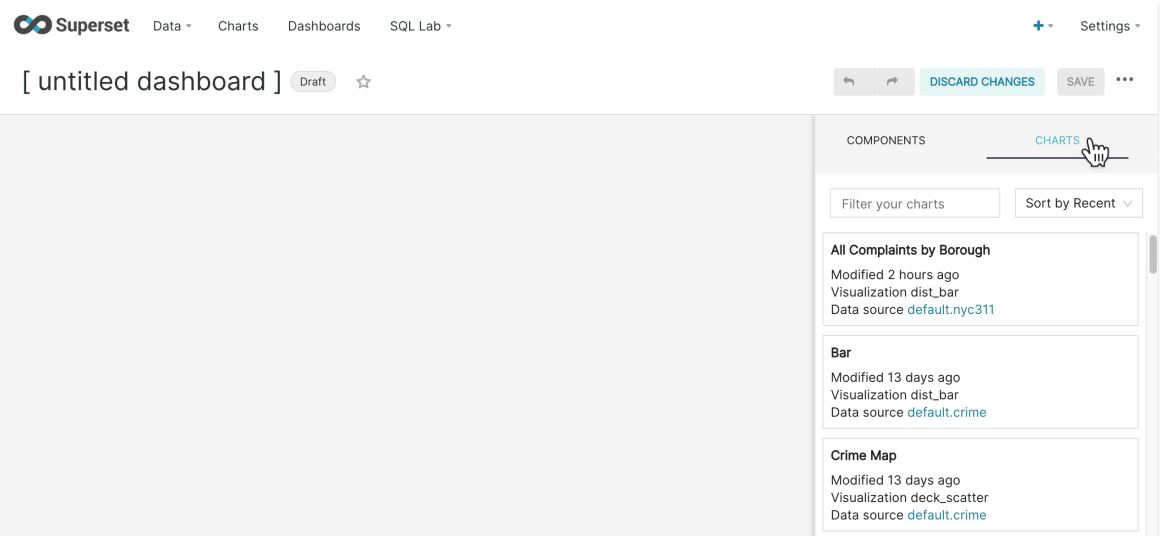


This will take you to the dashboard screen:



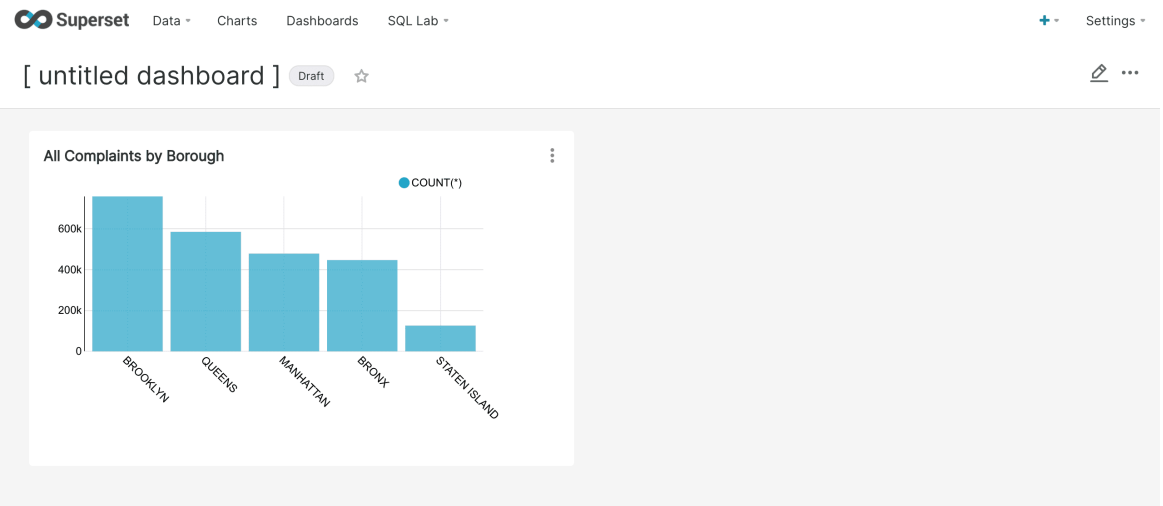
Dashboards are created by dragging components onto the dashboard.

2. Click **Chart** to find your saved charts:



3. Drag your bar chart onto the dashboard and resize it by mousing over the bottom right corner and dragging the chart.
4. When the chart is sized properly, click **Save Changes**.
   You can also title your dashboard by clicking on the title and changing it.

> **Note**
>
> To drag items onto the screen, move them to the top of the component they will be placed in. A blue line will appear on the screen when the item is ready to be dropped. If the blue line is not on the screen, the item will not be dropped.

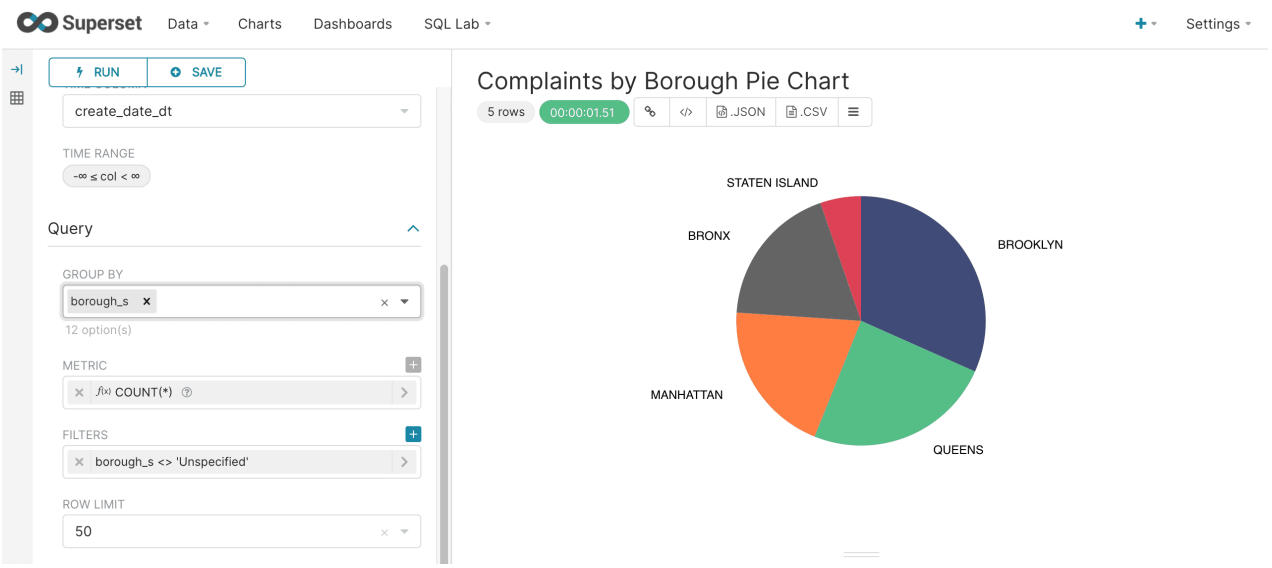


You can continue to add charts to your dashboard as more are created.

More Charts

Pie chart

Now that we have a basic understanding of charts, it's very easy to create a pie chart. In fact, pie charts are very similar to bar charts. Below is an example pie chart:



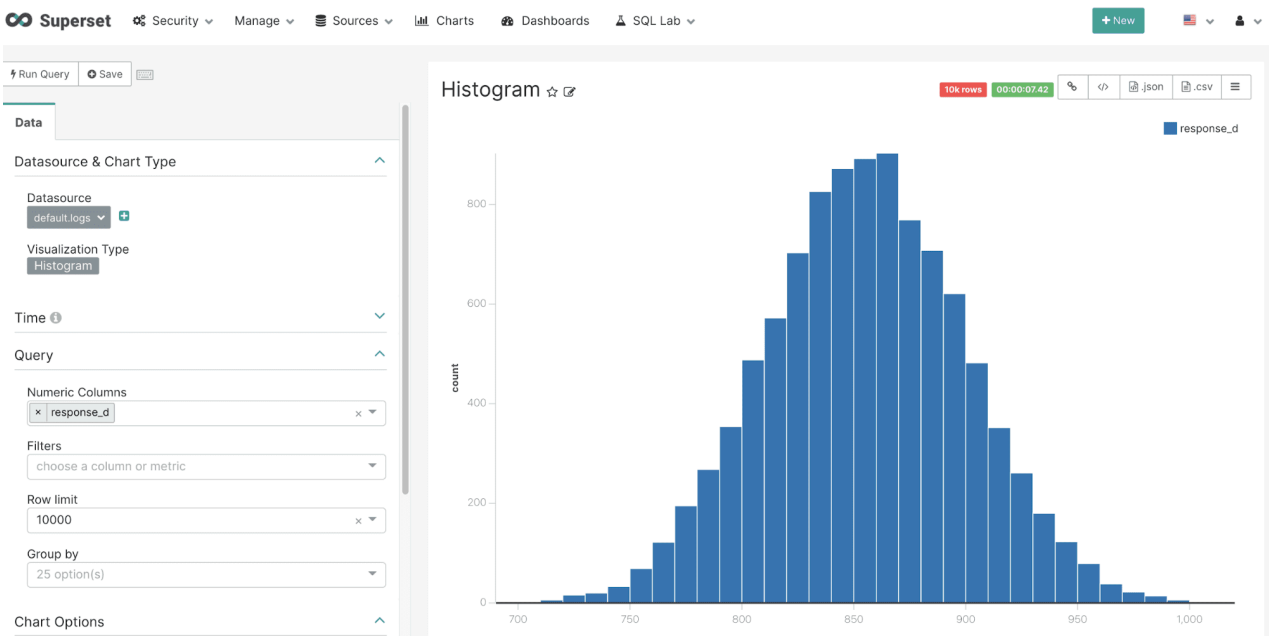
Notice that the **Group By** dropdown box is used instead of the **Series** dropdown in the bar chart. Otherwise the charts are configured the same way.

Histograms

Superset has a Histogram visualization which works perfectly with Fusion SQL's random sampling capability.

The example below shows the simple configuration for a histogram. A numeric field is selected from the Numeric Columns drop down list and the row limit is set to 10000. Filters can be applied as needed. The number of bins can be controlled under **Chart Options**.

Once configured, Superset issues a basic select to Fusion SQL for the numeric field. In Fusion SQL all basic selects without an `ORDER BY` clause return random samples. This random sampling capability allows for histograms that can be used to represent much larger data sets.

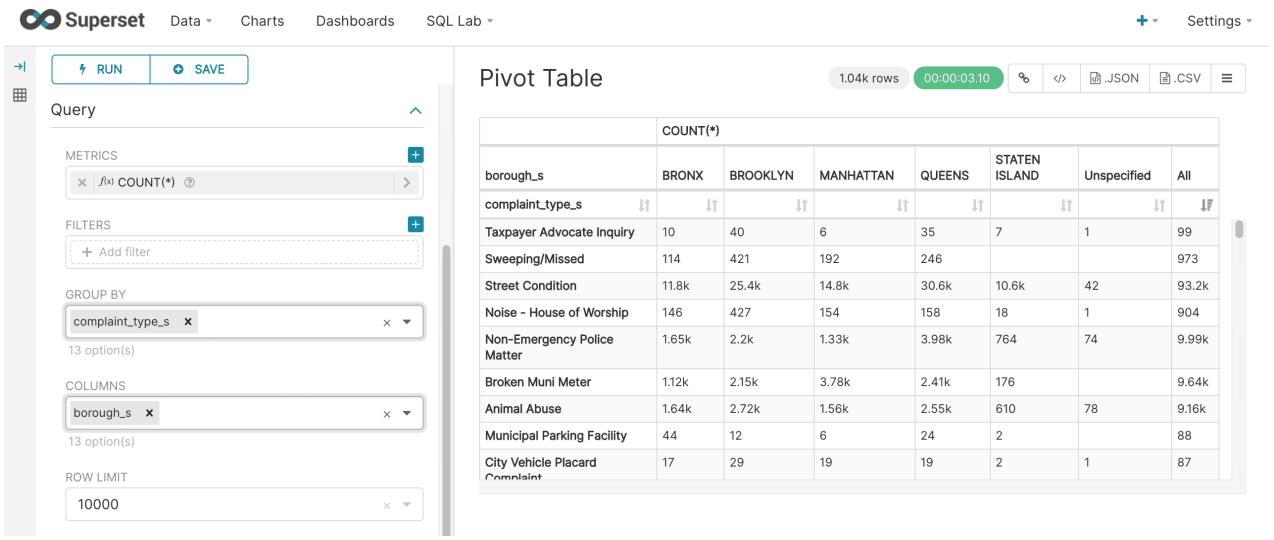


Viewing the query for this chart shows that no `ORDER BY` is used with the query resulting automatically in a random sample:

Pivot table

Superset's Pivot Table visualization can be used to visualize a two dimensional aggregation. In the example below we see that the Pivot Table control panel has both a **Group By** and **Columns** dropdown. In this example the `complaint_type_s` field is selected as the **Group By** field and the `borough_s` field is selected as the **Columns** field. The `COUNT(*)` metric is being calculated across both fields.

This Pivot Table is then displayed with rows showing complaint types and the columns showing the boroughs. The count of complaints for each complaint type and borough populates the cells.



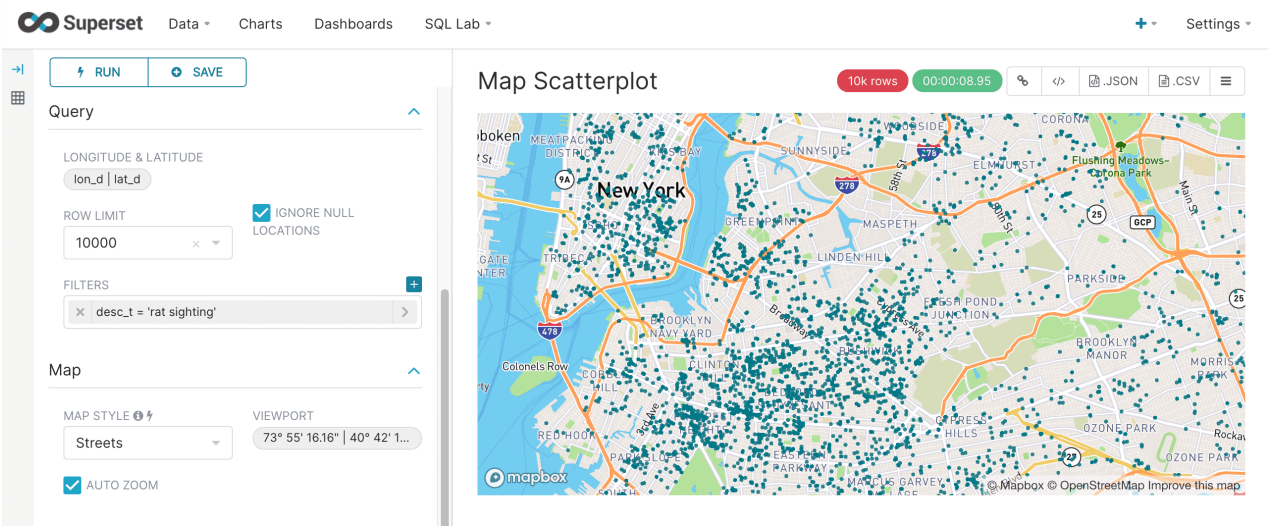
	COUNT(*)						
borough_s	BRONX	BROOKLYN	MANHATTAN	QUEENS	STATEN ISLAND	Unspecified	All
complaint_type_s							
Taxpayer Advocate Inquiry	10	40	6	35	7	1	99
Sweeping/Missed	114	421	192	246			973
Street Condition	11.8k	25.4k	14.8k	30.6k	10.6k	42	93.2k
Noise - House of Worship	146	427	154	158	18	1	904
Non-Emergency Police Matter	1.65k	2.2k	1.33k	3.98k	764	74	9.99k
Broken Muni Meter	1.12k	2.15k	3.78k	2.41k	176		9.64k
Animal Abuse	1.64k	2.72k	1.56k	2.55k	610	78	9.16k
Municipal Parking Facility	44	12	6	24	2		88
City Vehicle Placard Complaint	17	29	19	19	2	1	87

Map scatter plots

Superset has a number of powerful mapping visualizations. The `deck.gl.Scatterplot` is used to display a scatter plot of lat/lon points on a map. This particular visualization works perfectly with Fusion SQL's filtering and random sampling capability.

In the example below the `lat` and `lon` fields in the table are configured and a row limit of 5000 is selected. A filter of "Rat Sighting" in the `desc_t` field is added to select complaints with "Rat Sighting" in the description field. This will return a random sample of 5000 lat, lon points with "Rat Sighting" in the description. The points can then be plotted on the map which can be zoomed and scrolled.

The random sample of points is a useful tool for revealing high density clusters on the map. This works because the probability of returning a point from a high density area is higher than a low density area.



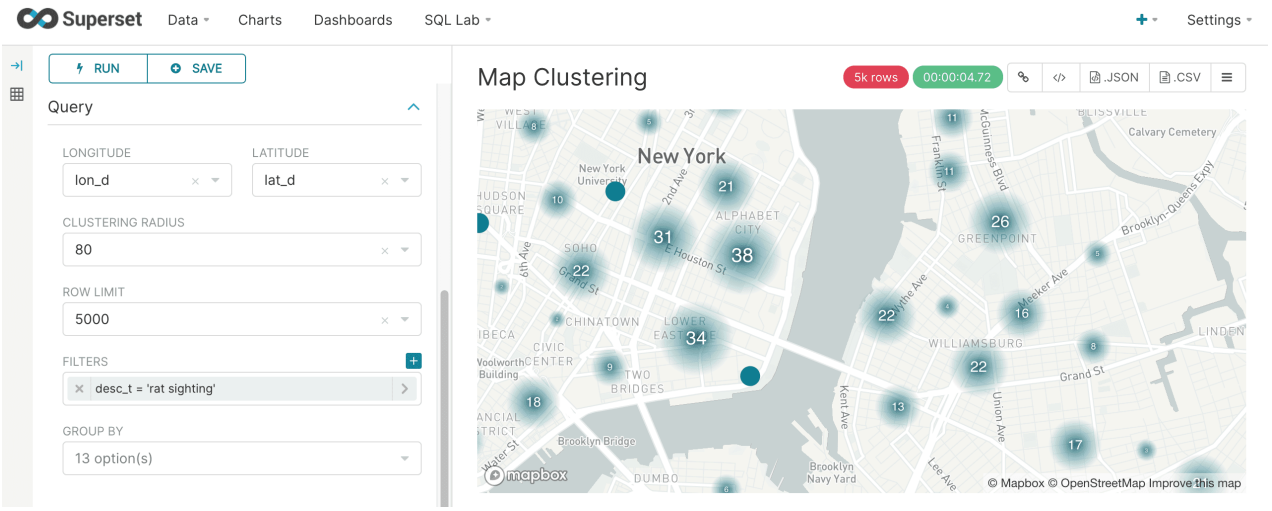
Note

For the map to render correctly, a Map Box key must be available for Superset at run time. In this example the Map Box key was exported to a unix environment variable before starting Superset as follows:

```
export
MAPBOX_API_KEY=pk.eyJ1IjoiZnVzaW9uc3FsIiwiYSI6ImNrOWtudX
pzazAyaW8zbHFwOG50NXVyMGgifQ.ZTyPhJaYQEyqvWY5f-mWbw
```

Map clustering

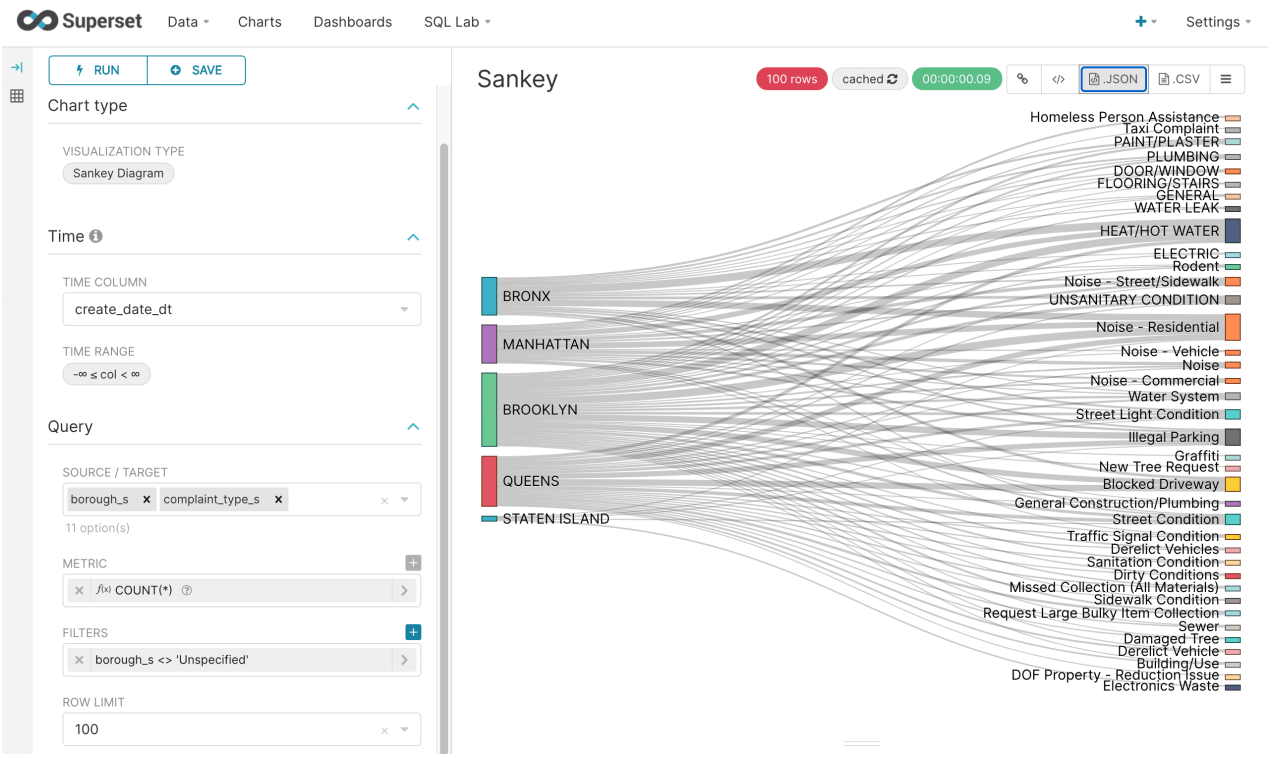
The Superset MapBox visualization is another powerful tool for visualizing geo-spatial data. Instead of a scatter plot the MapBox visualization clusters points on the map. Configuration of the MapBox visualization is very similar to the `deck.gl.Scatterplot` but with an added control for cluster radius. The example below shows the configuration and rendering of the MapBox visualization:



Sankey

The Sankey Diagram is another way to visualize a multi-dimensional aggregation. With a Sankey Diagram the strength of the relationships between dimensions is shown by the width of the pathways between them.

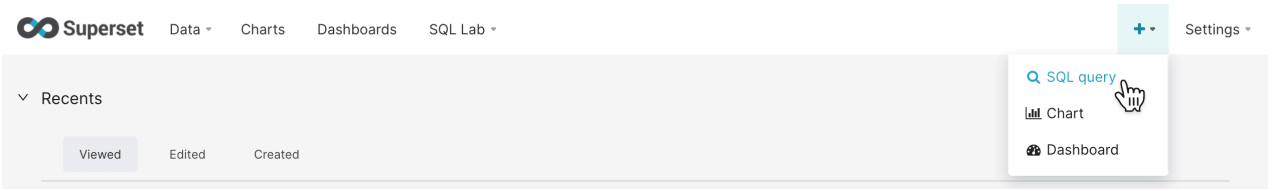
In the example below a Sankey Diagram is configured by having the borough as the source and the complaint type as the target. The `COUNT(*)` metric is used to measure the strength of relationship between borough and complaint type. Under the covers a multi-dimensional SQL aggregation is run on Fusion SQL to create the data for the diagram.



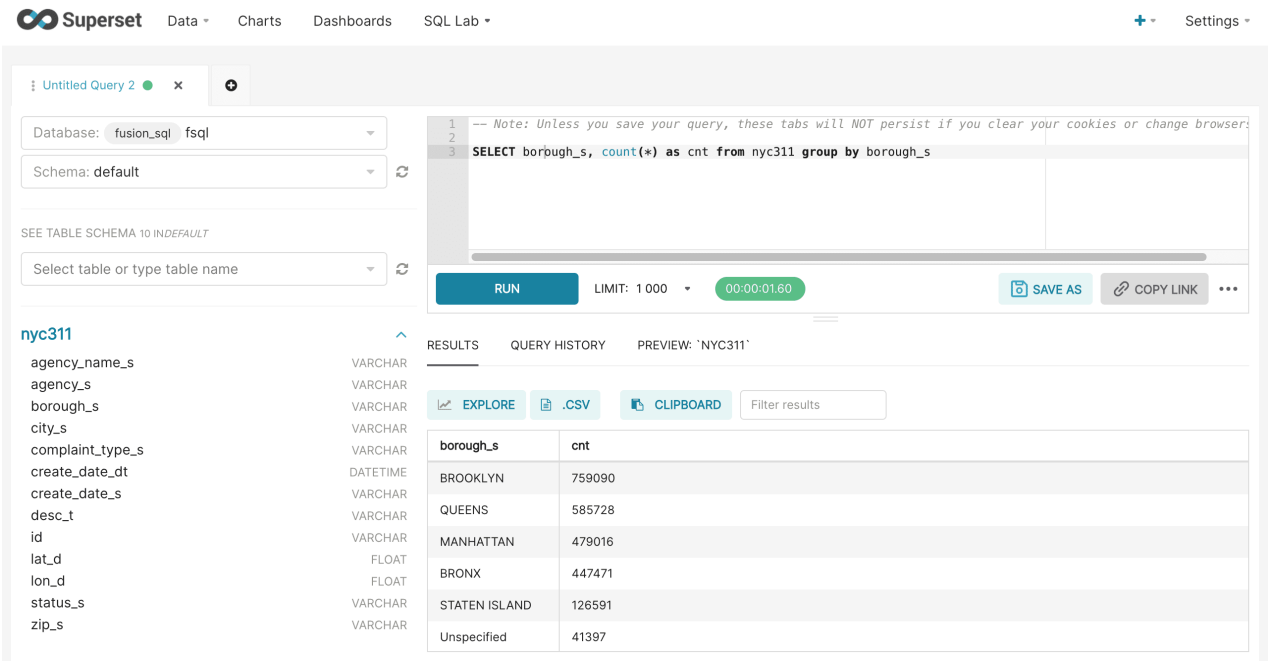
SQL Lab

Up until this point we've been creating visualizations without actually writing any SQL directly. In the next sections we'll explore how to use Superset's SQL Lab to write SQL queries directly and then visualize the results.

The SQL Lab can be accessed by choosing the **SQL Query** option under the **New** button in the upper right corner:



The SQL Lab screen shows the meta-data for the databases, schemas and tables on the left side. In the main section of the screen there is a box for typing in SQL queries. The results display in a table below:



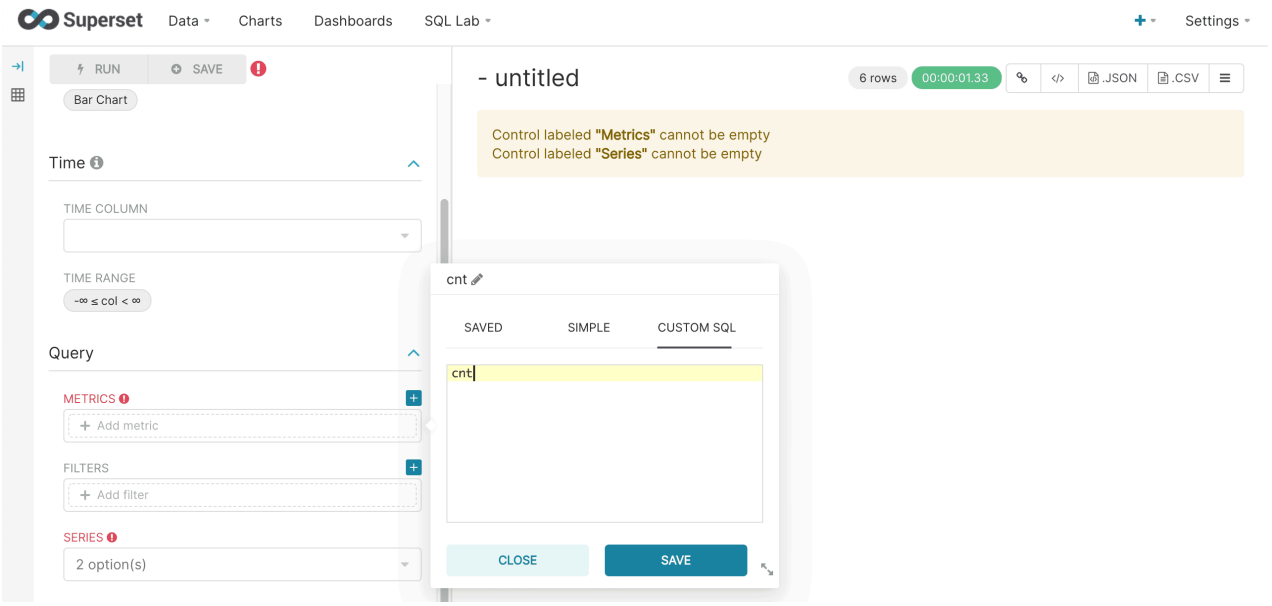
It's important to note that Superset controls the `LIMIT` through the `LIMIT` control in the user interface. This will override any limits added to the query. Otherwise the SQL query will be issued exactly as it's entered.

In this example a SQL aggregation is run and the results are displayed.

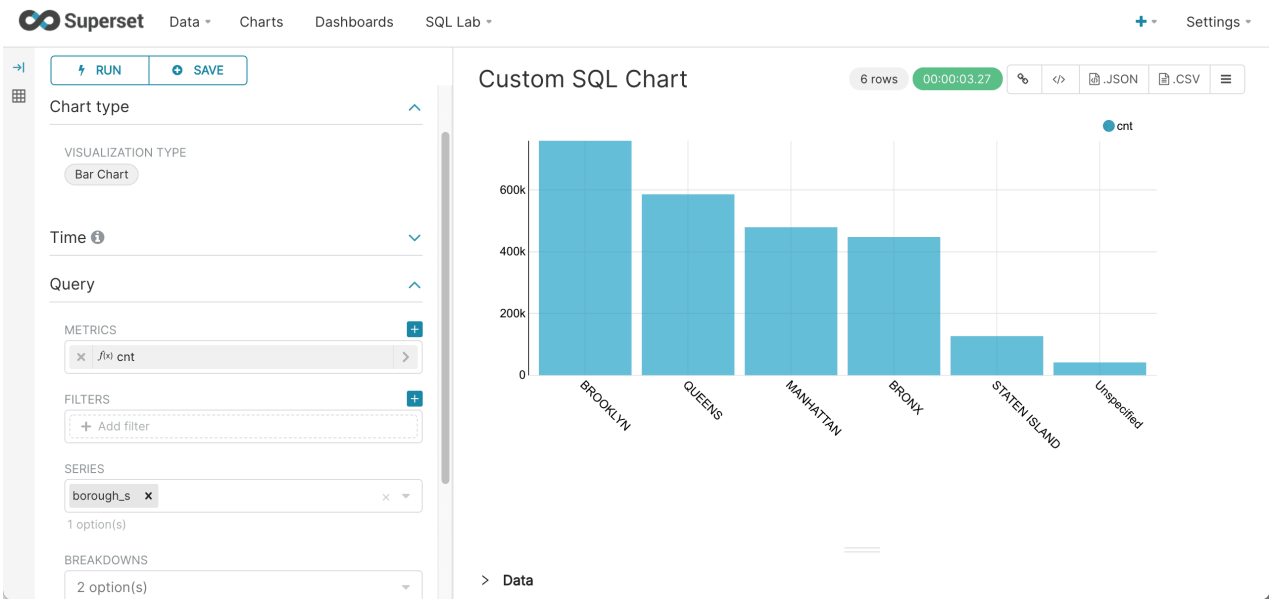
Once the results table is displayed the Explore button can be used to visualize the results. After clicking the Explore button the familiar chart creation screen will load.

The screenshot below shows the first steps in configuring the chart for this custom SQL query. First the **Visualization Type** is changed to "Bar Chart". Then the **Time Range** is set to "No filter".

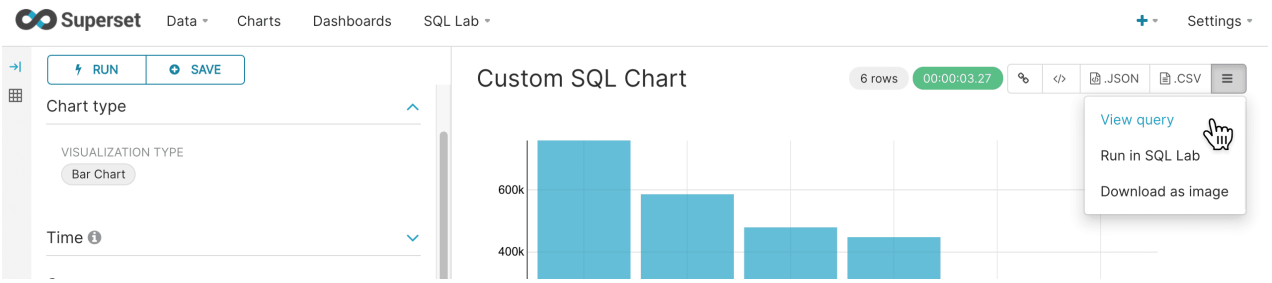
Then the `cnt` field from the result set is selected as the Metric. Remember that in the SQL query the alias for the `COUNT(*)` metric was `cnt`. Superset by default will want to perform an aggregation on top of this, so we'll choose the Custom SQL tab to override this behavior. In the Custom SQL tab simply remove the parentheses from around the `cnt`.



Next we'll choose the group by field, `borough_s`, as the **Series** and run the query. The bar chart for our custom SQL query then renders in the right panel as normal.



We can view the query using the same approach we used for regular charts.



This will show how Superset interacts with the custom query to render the chart.

Time series aggregations

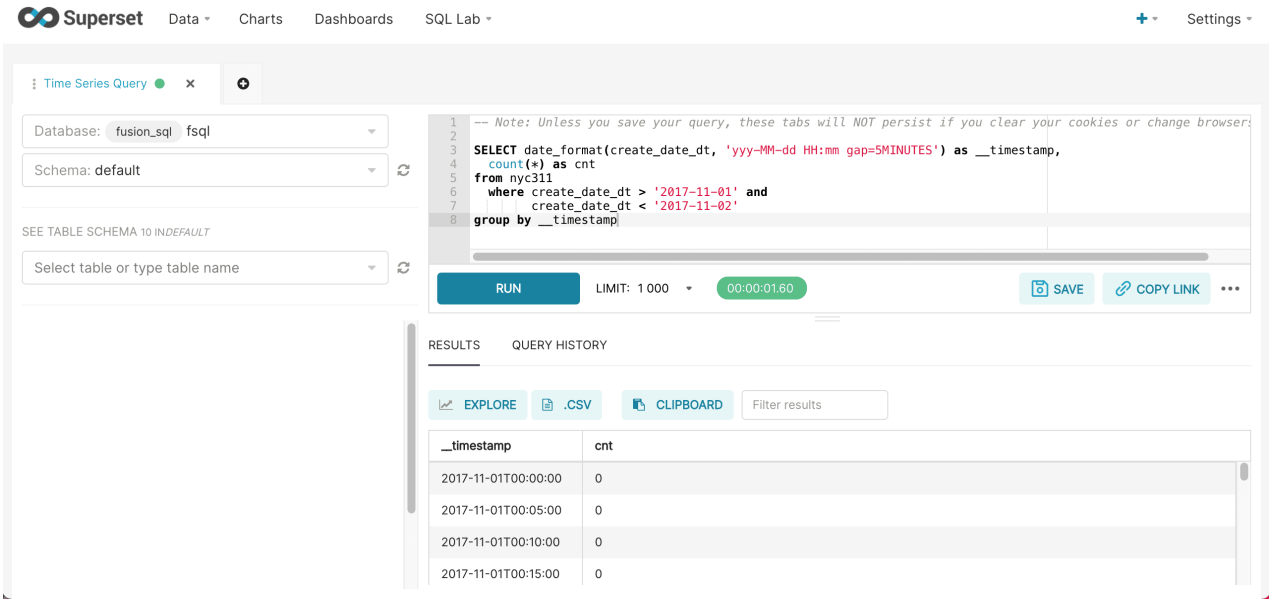
Both Fusion SQL and Superset have powerful time series capabilities. This section describes both the Fusion SQL time series syntax and the Superset visualization.

Basic time series query

Fusion SQL supports high performance and flexible time series aggregation by grouping over the date_format function. The `date_format` function takes two parameters:

- The date/time field
- The format and interval gap. The format is specified using Java SimpleDateFormat templates. The interval gap is specified by using Solr date math syntax.

The example below shows an example of a basic time series query run inside of the Superset SQL Lab:

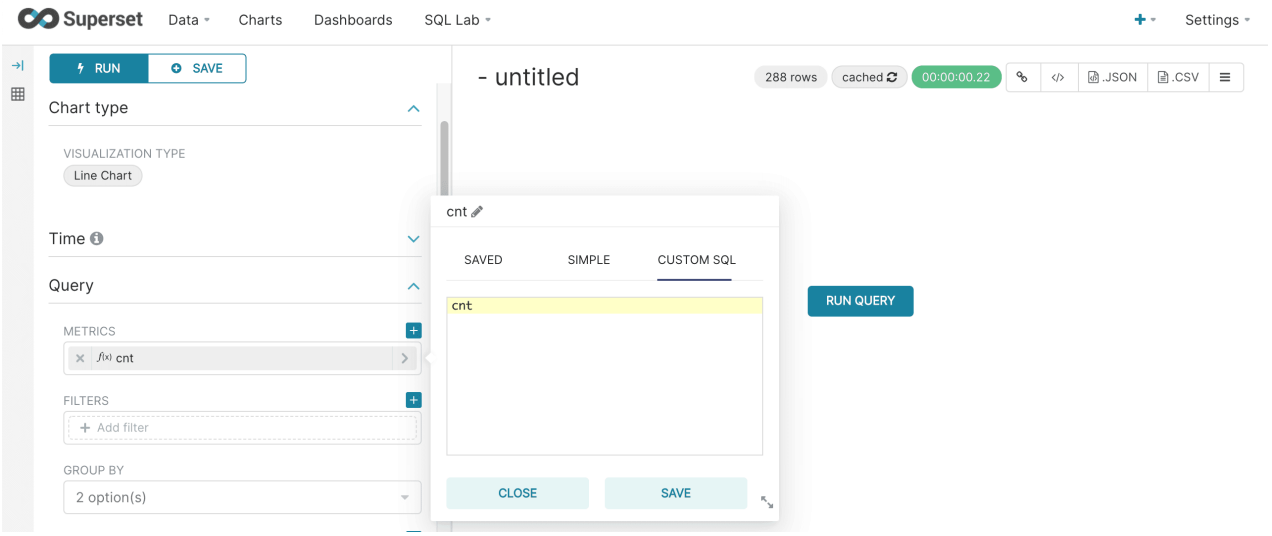


The format `yyyy-MM-dd` specifies that this is a daily time series. The alias `__timestamp` is required for the integration with Superset's time series visualization.

Once the results return we visualize by clicking on the **Explore** icon and selecting a **Line Chart**.

The Line Chart configuration is very basic because the SQL query defines all of the query filters and other details.

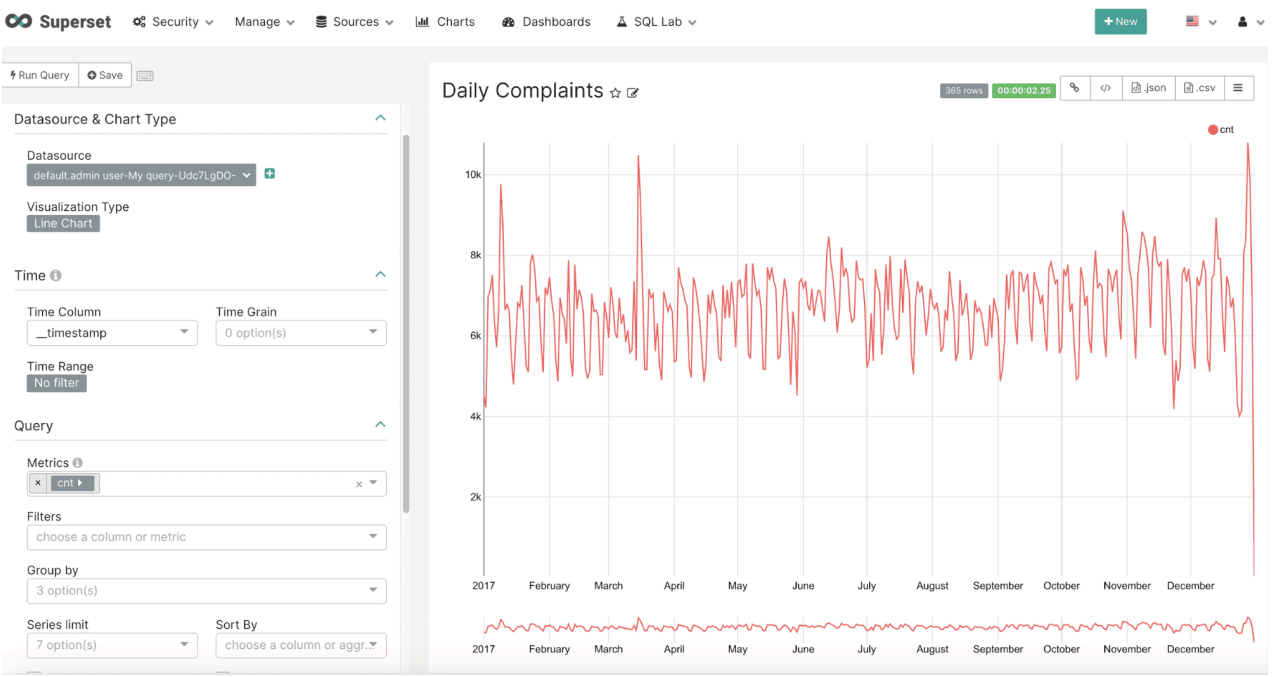
The screen below shows the configuration:



A couple of things to note about the configuration:

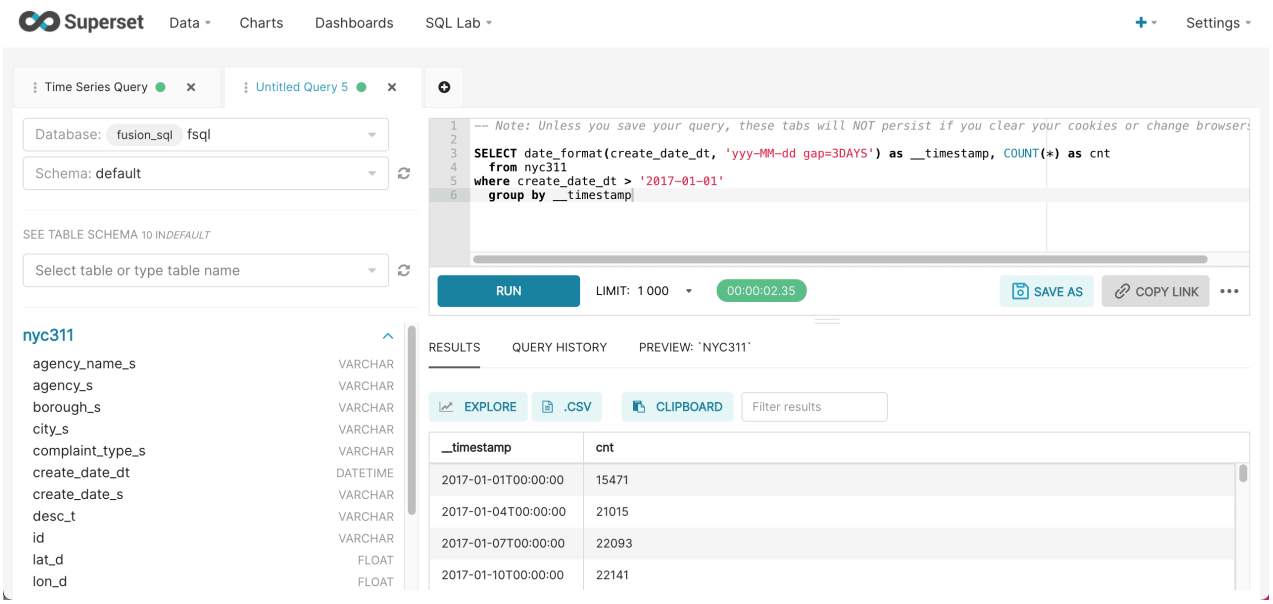
- The Time filter should be turned off. Time filters need to be applied in the SQL query.
- The time column will automatically populate with `__timestamp` field in the result set.
- The metric should be set to the `cnt` field. Use the Custom SQL and remove the parentheses to use the value in the `cnt` field without an aggregation applied to it.

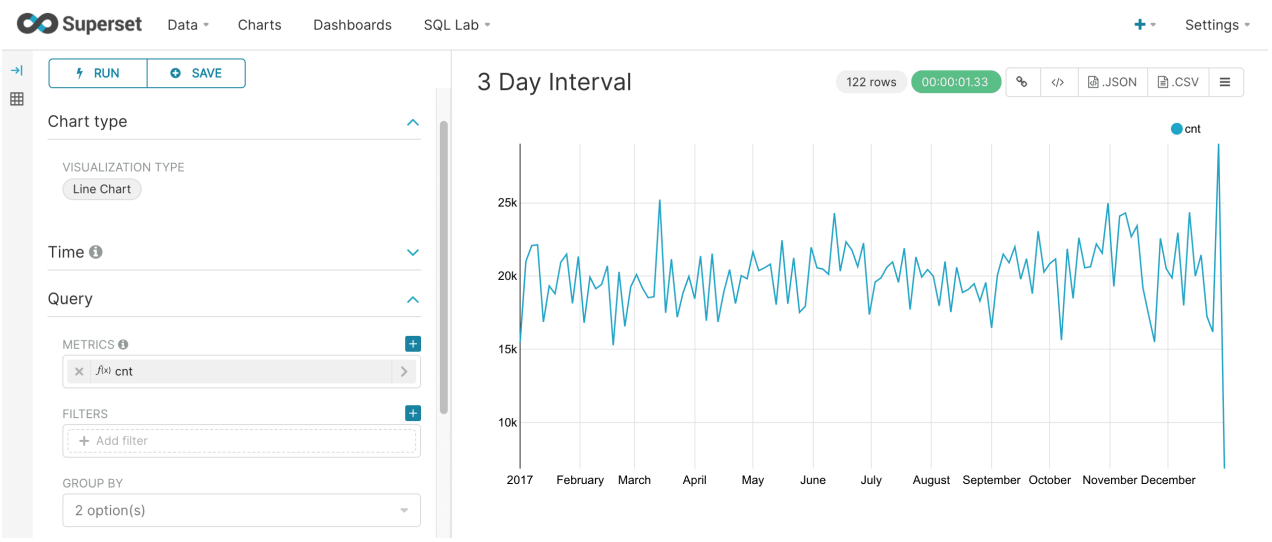
After running the query the chart will render as a time series:



Interval gap

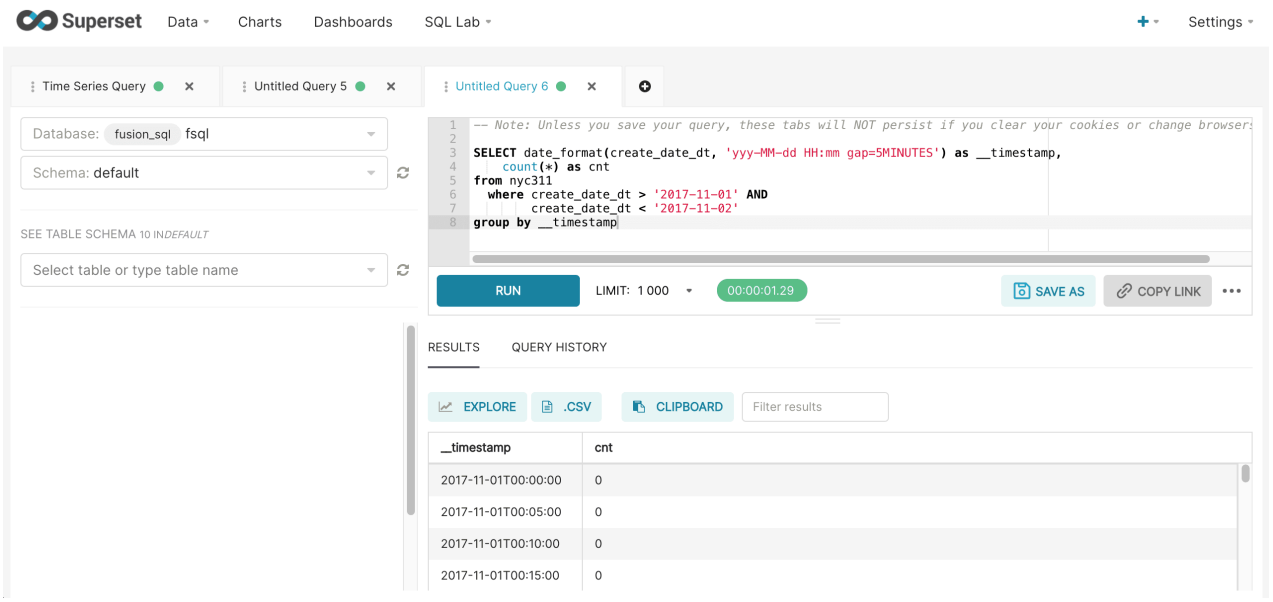
The `gap` parameter can be supplied along with the format to specify the gap interval. The example below specifies a 3 day gap.

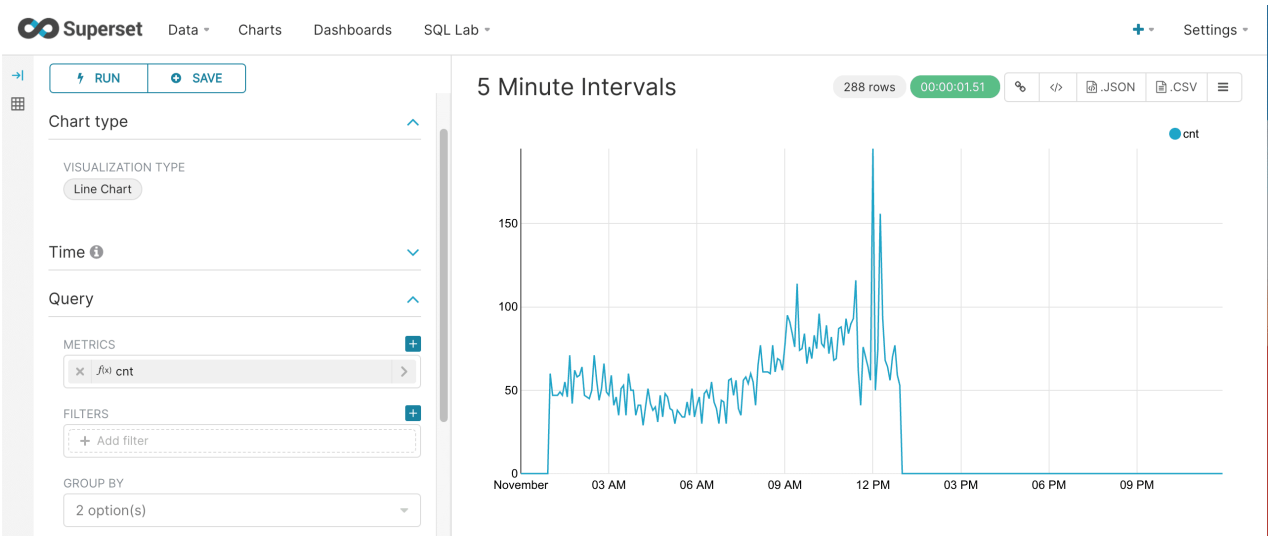




Finer grain queries

Finer time grains can be defined by adding hours, minute or seconds to the time format. The example below shows the format and gap for a 5 minute interval time series.





Fusion SQL statistical functions

Fusion SQL includes a library of statistical and machine learning functions. The section below describes a subset of these functions and how they can be used within Superset.

hist

Fusion SQL supports the `hist` function which pushes the histogram down into server so the full result set doesn't need to be brought over to Superset. This allows for much larger sample sizes to be used for the histogram creation.

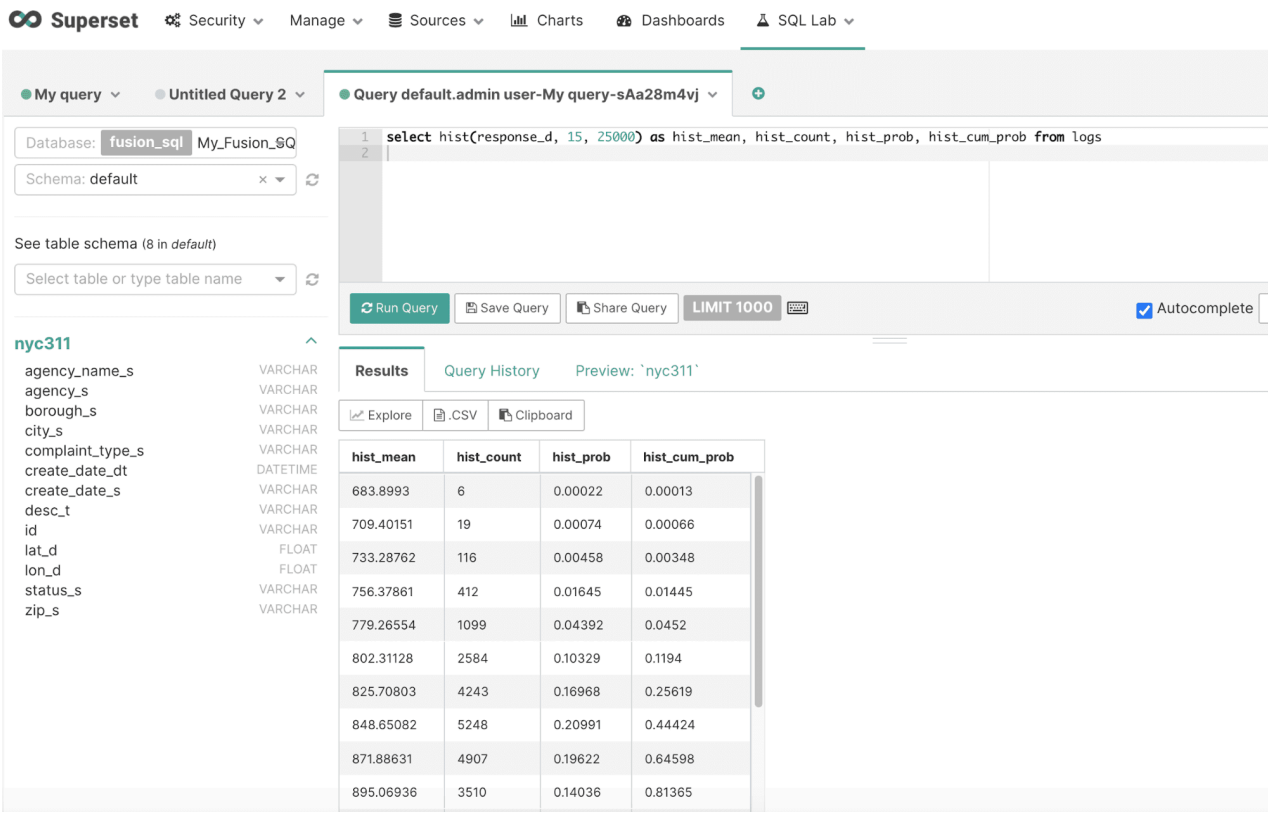
The `hist` function takes three parameters:

- The numeric field to create the histogram from
- The number of bins
- The size of the random sample to create the histogram from

The result will contain a row for each bin in the histogram. The `hist` function returns the mean of each bin in the histogram. There are three other fields that can be selected when the `hist` function is used:

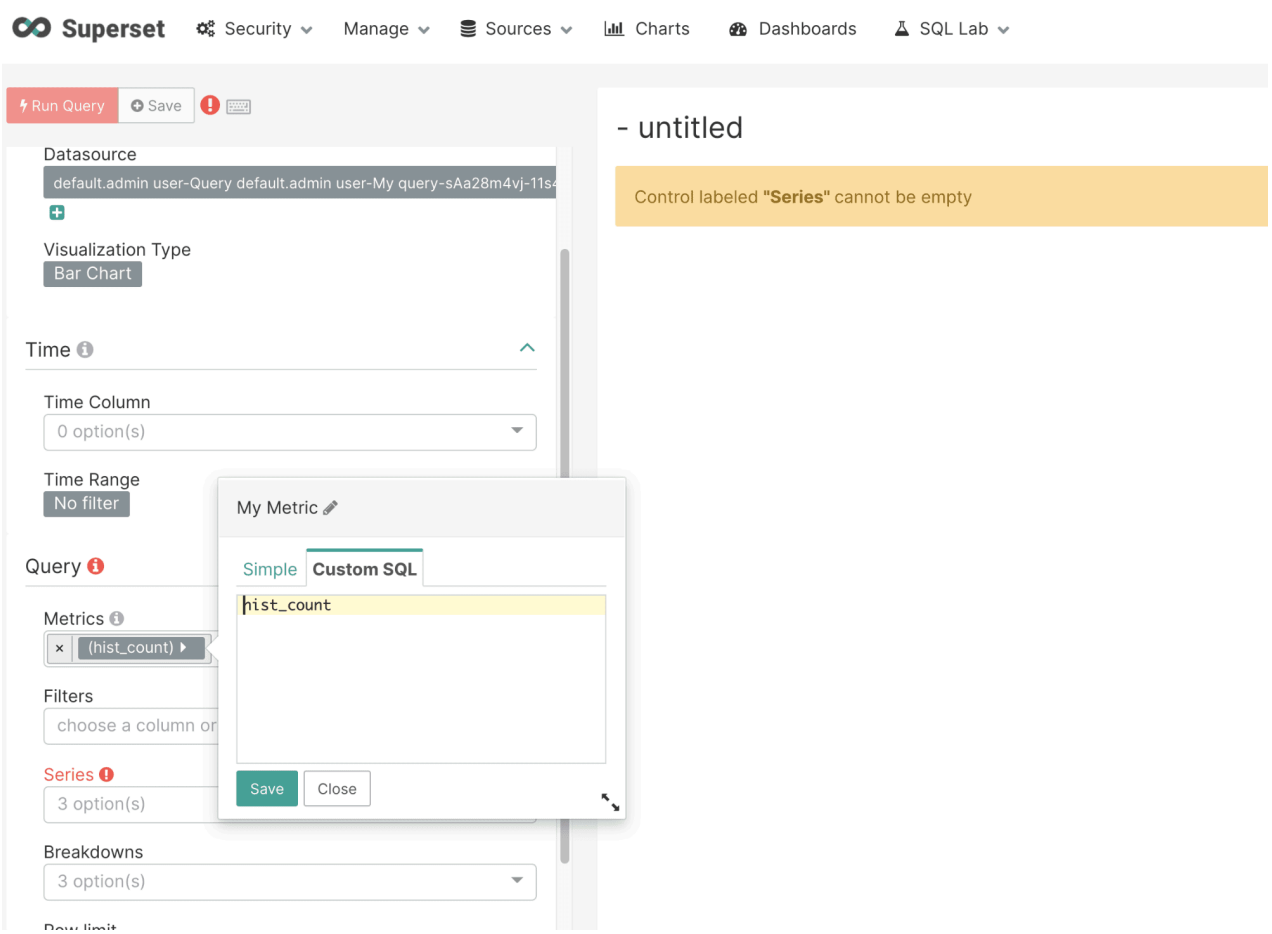
- `hist_count` : The number of samples in the bin
- `hist_prob` : The probability of sample falling within the bin
- `hist_cum_prob` : The cumulative probability of each bin

Below is an example of the `hist` function run in the SQL Lab:

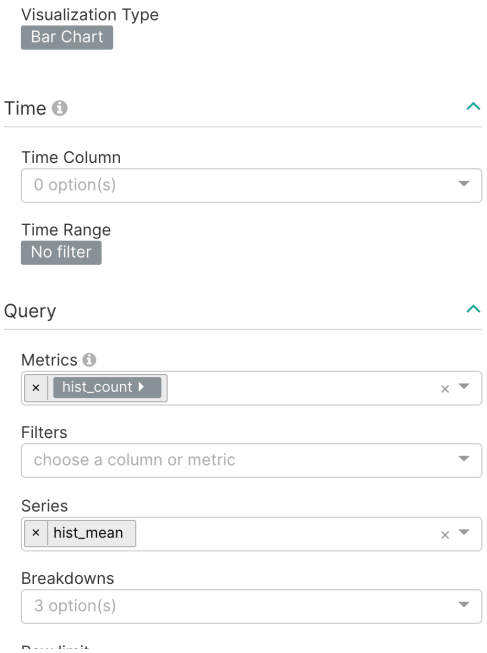


Clicking **Explore** brings up the chart creation screen.

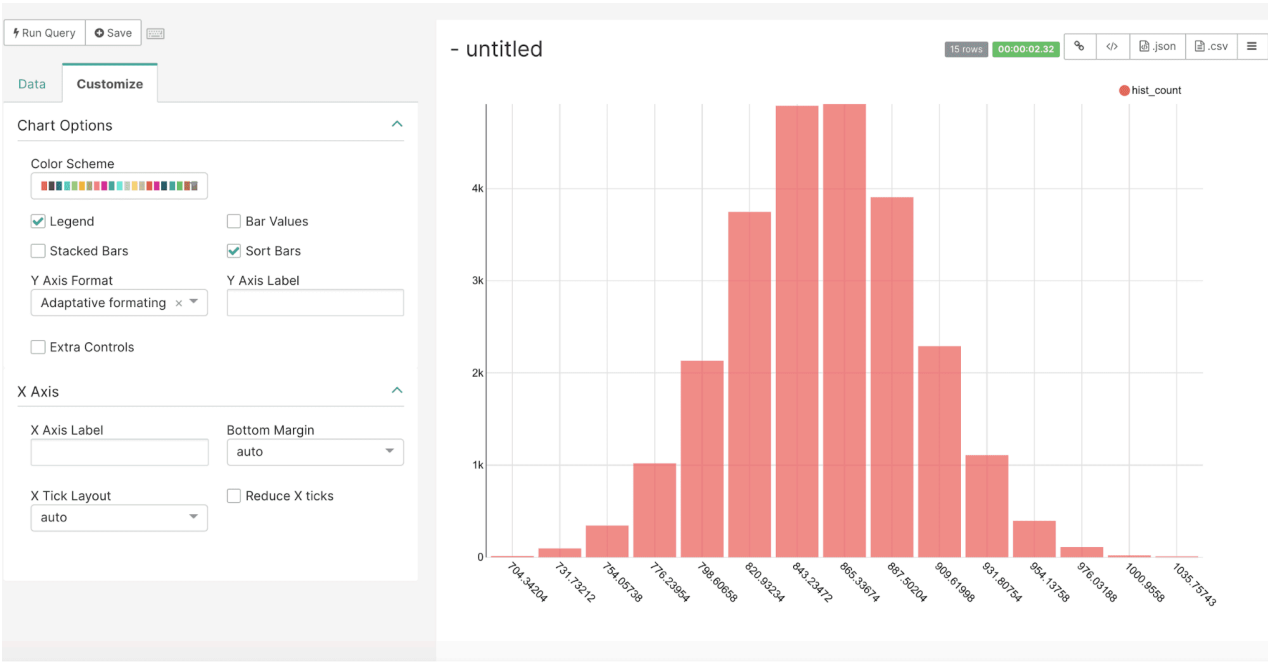
Choose a Bar Chart and select `hist_count` as the metric. This will use the `hist_count` for the y-axis. Be sure to remove the parentheses using the Custom SQL panel shown below:



Next select the `hist_mean` as the Series as shown below:



By default the bar chart will sort the bars in descending order. To maintain the original order of the bar chart go to the **Customize** tab and check the **Sort Bars** box. This will display the histogram bars in the order of the result set as shown below:



geo_cluster

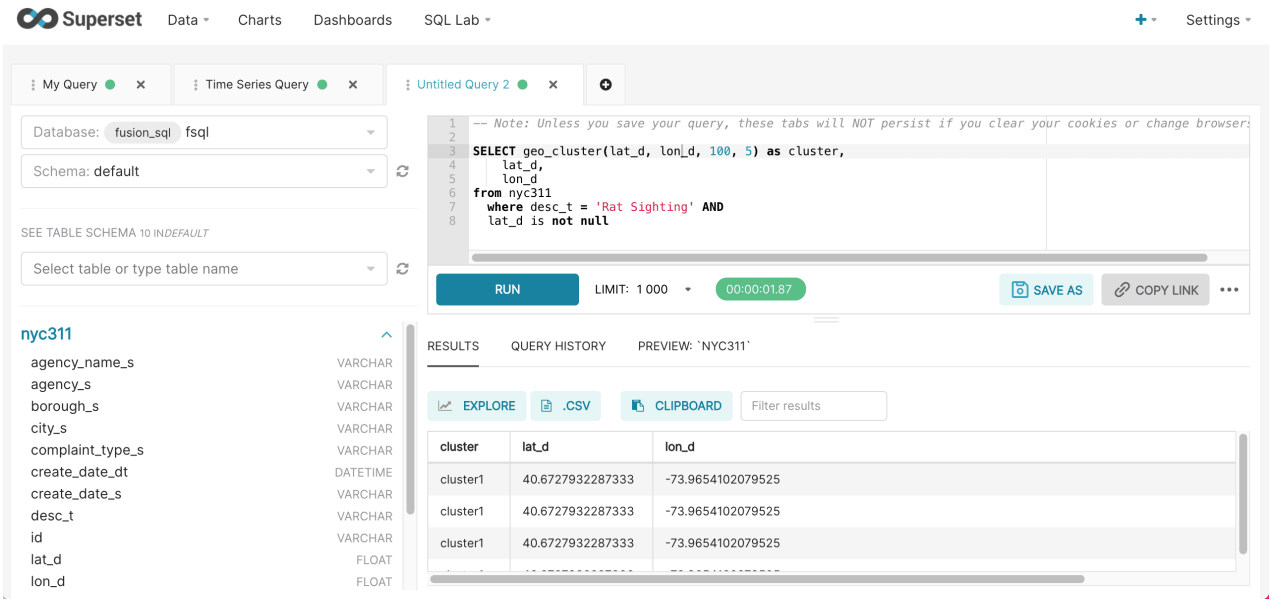
We saw in the section on Mapping that it was very easy to plot lat/lon points on a map with Superset's `deck.gl.Scatterplot` visualization. Fusion SQL can identify the clusters in these scatter plots with the `geo_cluster` function. The `geo_cluster` function performs `geo_spatial` clustering and noise reduction over a set of lat/lon points. Under the covers the `geo_cluster` function uses DBSCAN clustering using the haversine meters distance measure.

The geo-cluster function allows you to specify the criteria that define a cluster. Any points that don't meet the criteria will be filtered from the result set. This filtering provides powerful noise reduction which can help to visualize the clusters on a map.

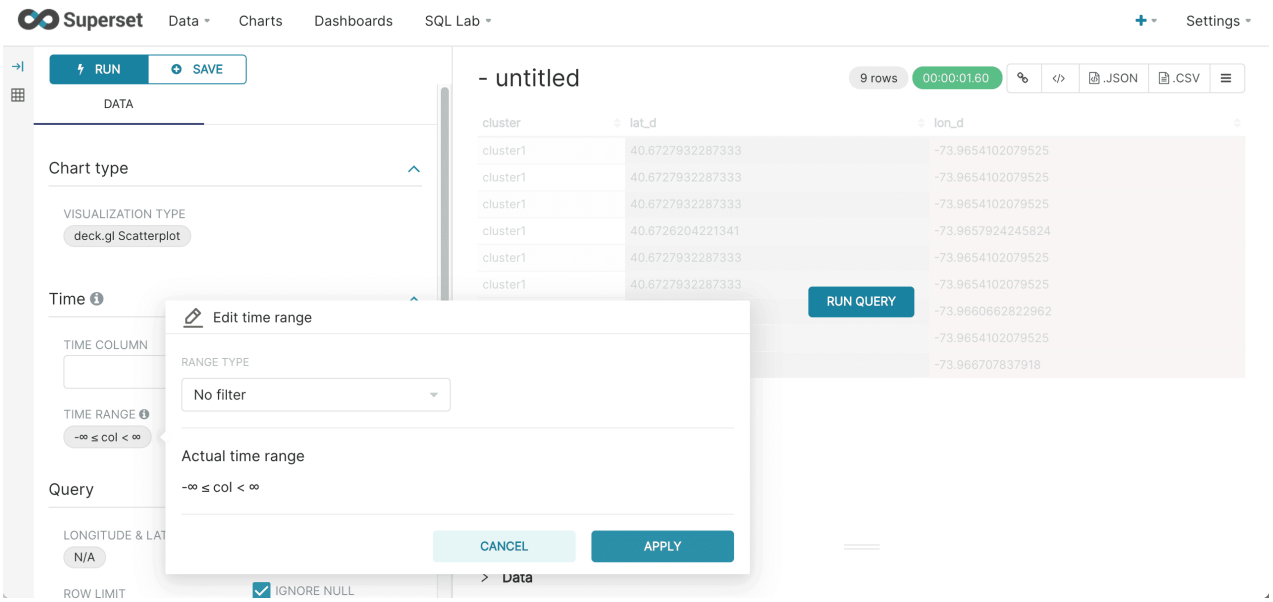
The `geo_cluster` function takes four parameters:

- The latitude column in the index
- The longitude column in the index
- The distance in meters between points for points to be considered a cluster
- The smallest number of points to be considered a cluster

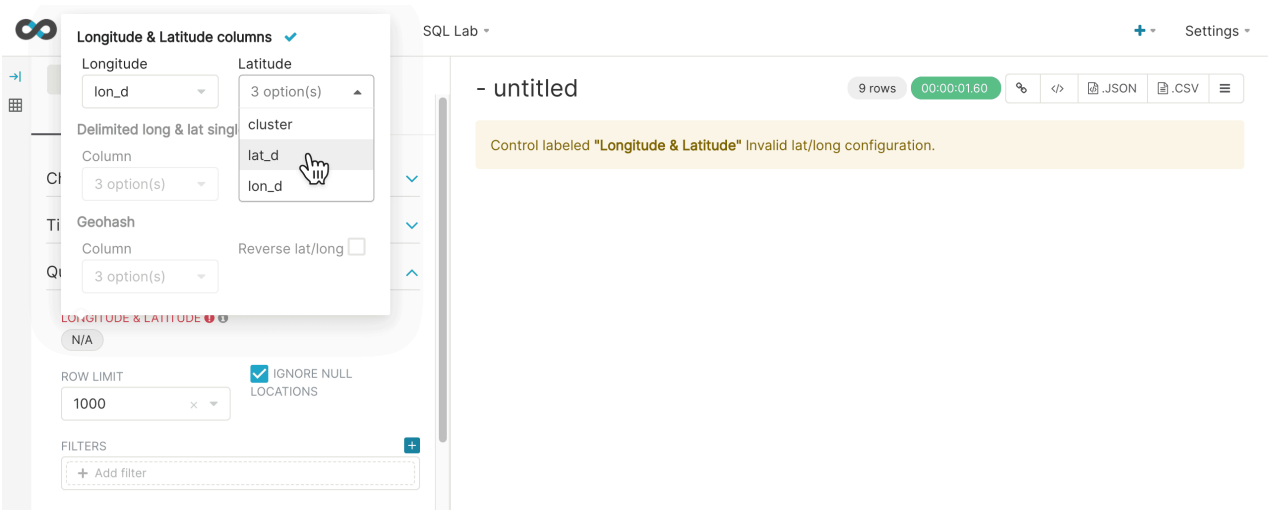
The `geo_cluster` function returns the cluster ID for each point in the result set. The latitude and longitude fields can also be selected with each point. Below is an example of the `geo_cluster` function run inside of the SQL Lab:



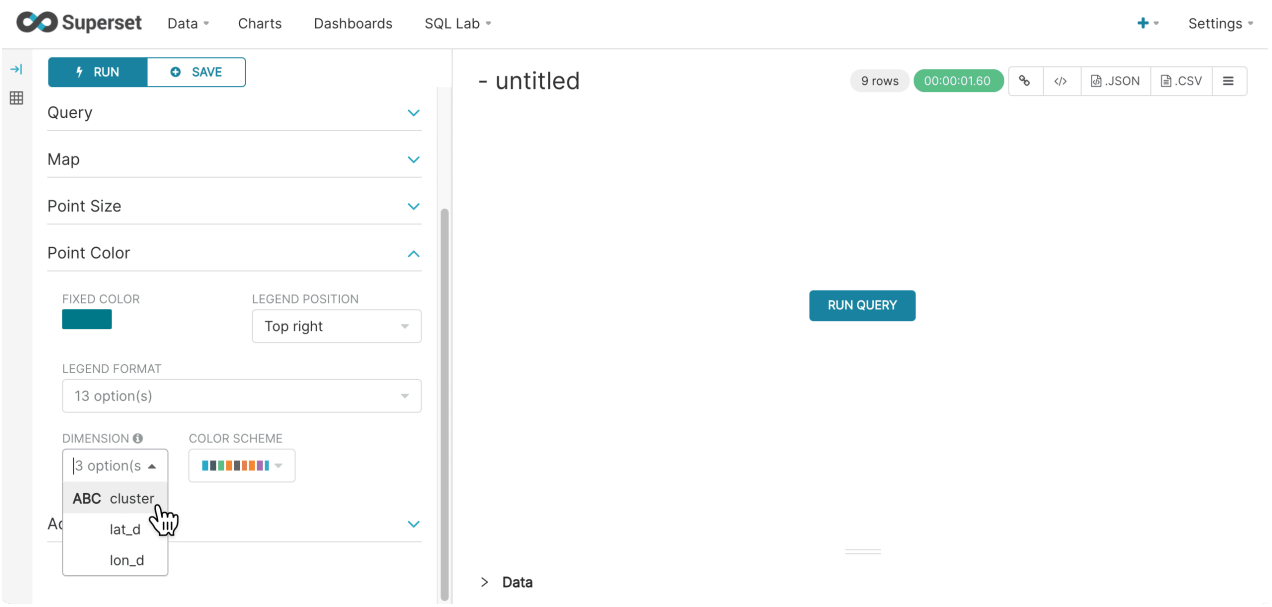
After the results are displayed, click **Explore** to start visualizing the results. The first steps in visualizing is to select the `deck.gl.Scatterplot` visualization and turn off the **Time Range** filter as shown below:



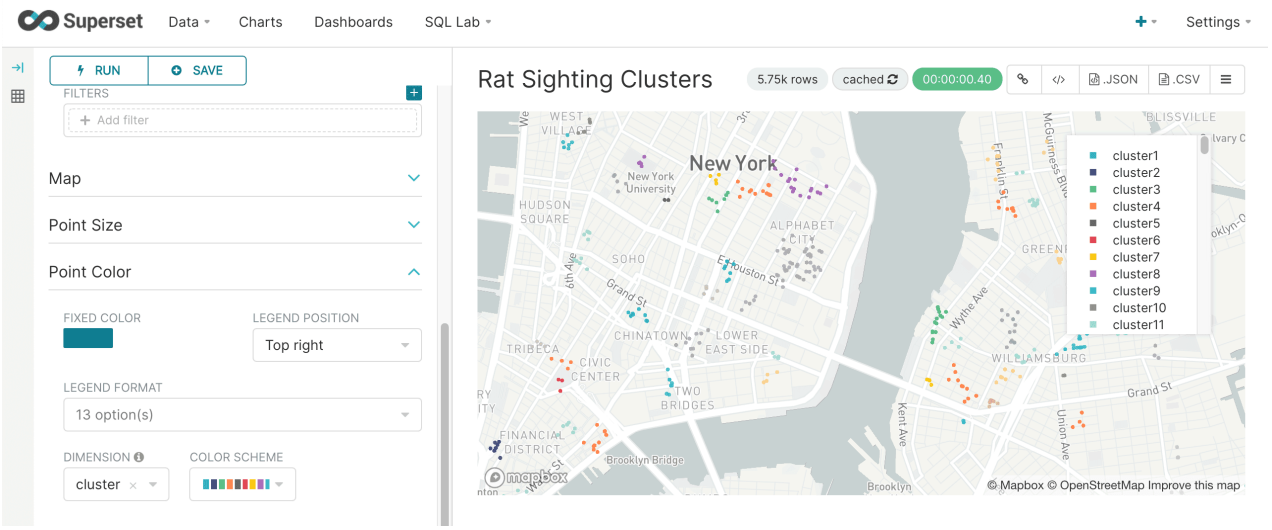
Next you can configure the longitude and latitude columns and choose the row limit. The row limit will control the number of points that are selected for clustering.



Then in the **Point Color** section select the `cluster` column under the **Categorical Color** icon. This will color code the points based on cluster ID.



When you run the query the map will render, displaying only the points that belong to a cluster. The example below shows the color coded clusters on the map:



significant_terms

The `significant_terms` function finds anomaly terms in a text or string field that appear more frequently in a search result set than the entire index.

The `significant_terms` function takes four parameters:

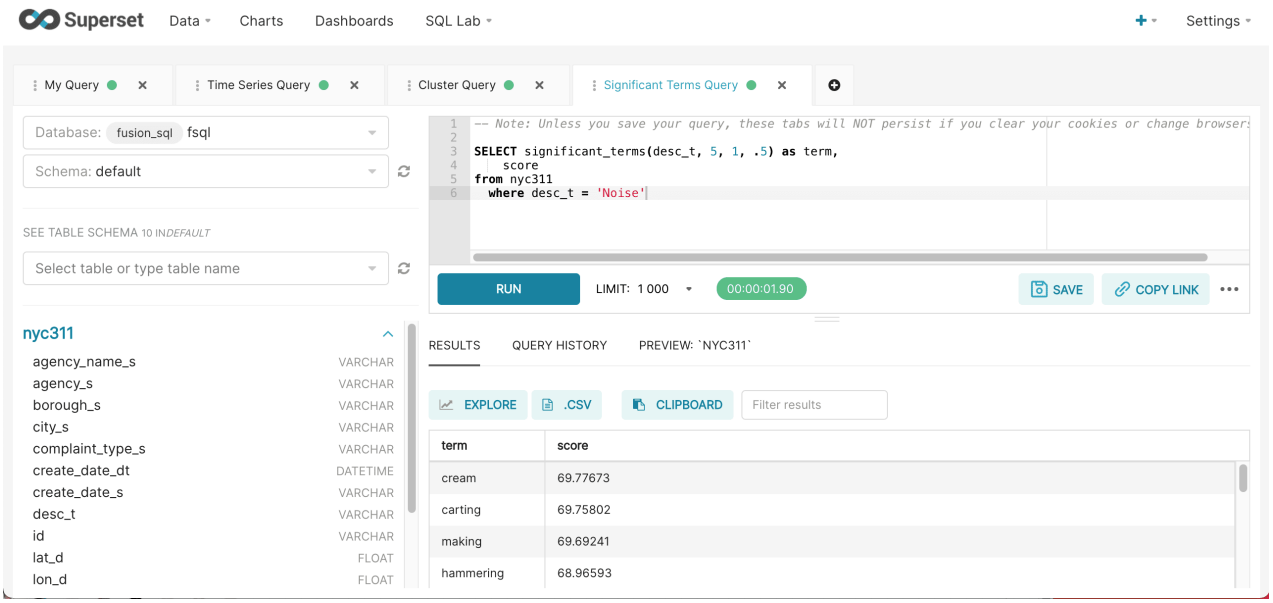
- the string or text field in which to find the terms.
- the minimum term length to be considered a significant term.
- the minimum document frequency for a term to be considered a significant term. If greater than one this value is treated as an absolute number of documents. If the value is a float between 0 and 1 its considered to be a percentage of total documents.
- the maximum document frequency for a term to be considered a significant term. If greater than one this value is treated as an absolute number of documents. If the value is a float between 0 and 1 its considered to be a percentage of total documents.

The result set for the `significant_terms` function contains one row for each significant term. The `significant_terms` function returns the value of the term. There are three additional fields available when the significant_terms function is used:

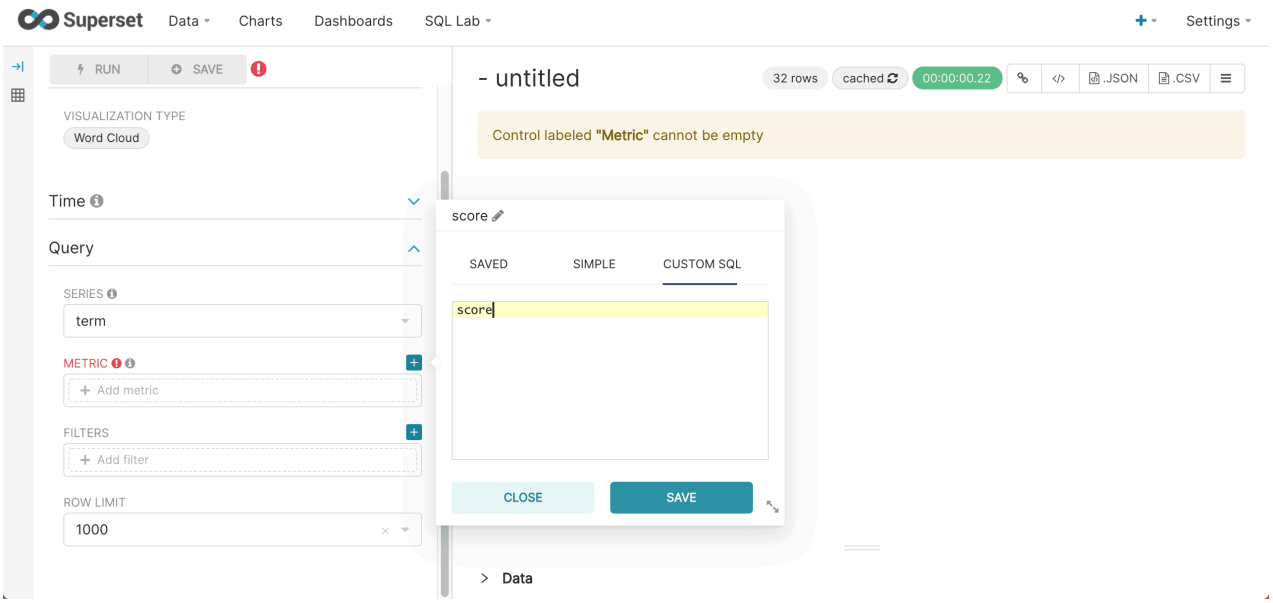
- The foreground field returns the number of documents that contain the term within the result set.
- The background field returns the number of documents that contain the term in the entire index.

- The score field returns the score for the field which is calculated based on the background and foreground counts. Terms are returned in score descending order.

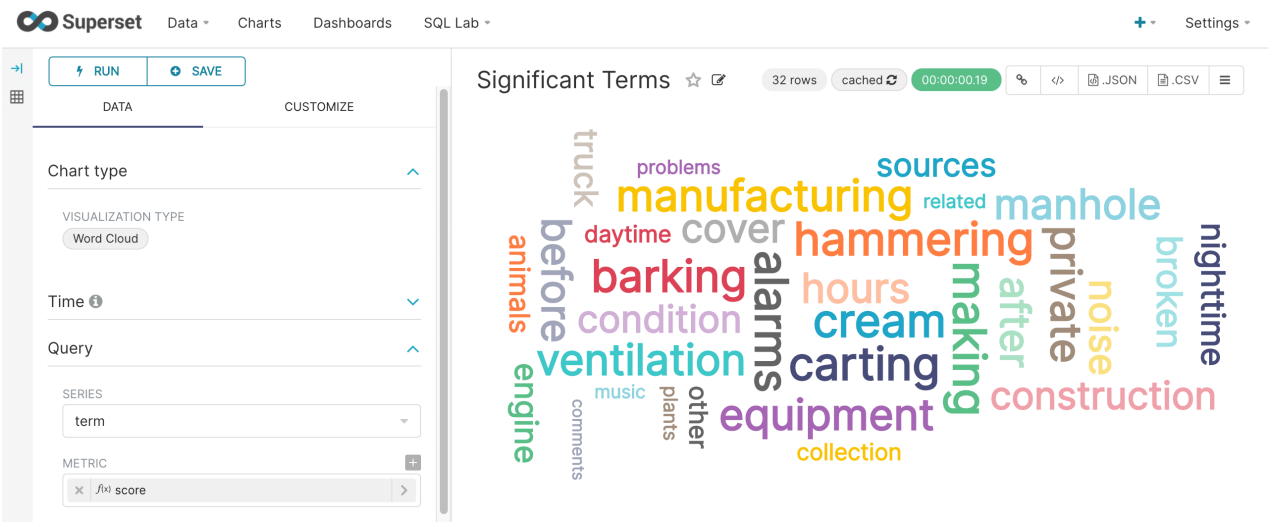
Below is an example of the `significant_terms` query run inside of SQL Lab.



After the results have been displayed they can be visualized by clicking **Explore** and selecting the **Word Cloud** visualization. The word cloud visualization can be configured by selecting `term` in the **Series** drop down box and `score` as the **Metric**. Be sure to choose **Custom SQL** and remove the parentheses from the score so that an aggregation is not applied to the score. This configuration step is shown below:



Once the Word Cloud is configured, run the query to see the word cloud rendered like the screenshot below:



analyze

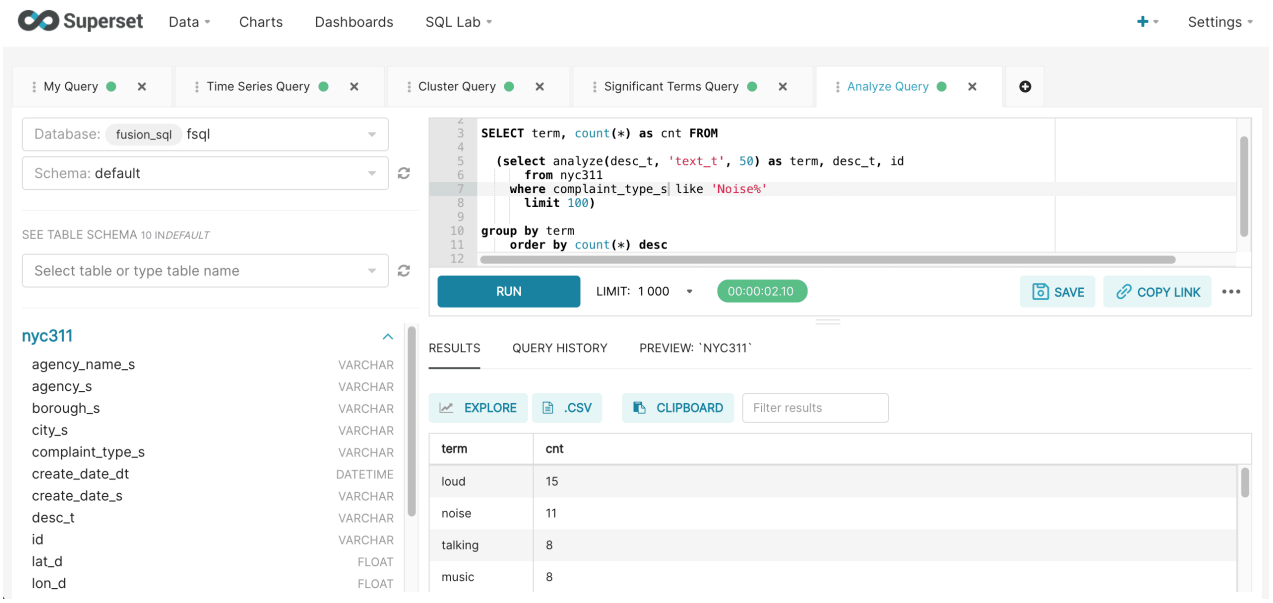
The `analyze` function analyzes a text or string field with a Lucene/Solr analyzer and returns the analyzed tokens. The use of Lucene/Solr analyzers at query time allows for real-time entity extraction by configuring OpenNLP, Stanford NLP or other NLP analyzers.

The `analyze` function takes three parameters:

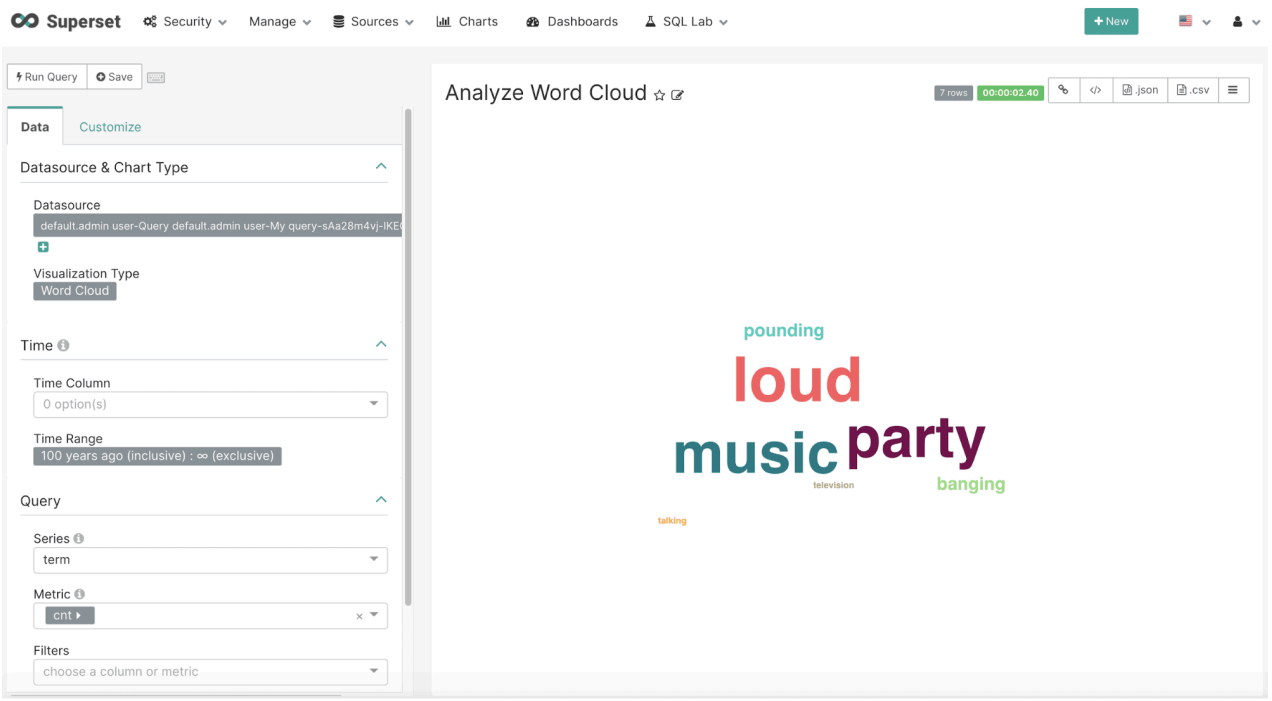
- the text or string field to analyze
- a string literal pointer to the Lucene/Solr analyzer to use. This can be either a field name in the Solr schema with an analyzer defined, or a string that matches a dynamic field pattern.
- the number of documents to analyze

Which documents are analyzed is determined by the WHERE clause. The documents are ordered by score descending to determine the top N documents to analyze.

A SQL aggregation query can be wrapped around the analyzed result set to perform aggregation over the terms. Below is an example of the wrapped aggregation query run inside of SQL Lab:



After the results have been displayed they can be visualized by clicking on the Explore icon and choosing the Word Cloud visualization. The word cloud visualization can be configured by selecting `term` in the **Series** drop down box and `cnt` as the **Metric**. Be sure to choose **Custom SQL** and remove the parentheses from the `cnt` so that an aggregation is not applied to the score. This configuration step is shown below:

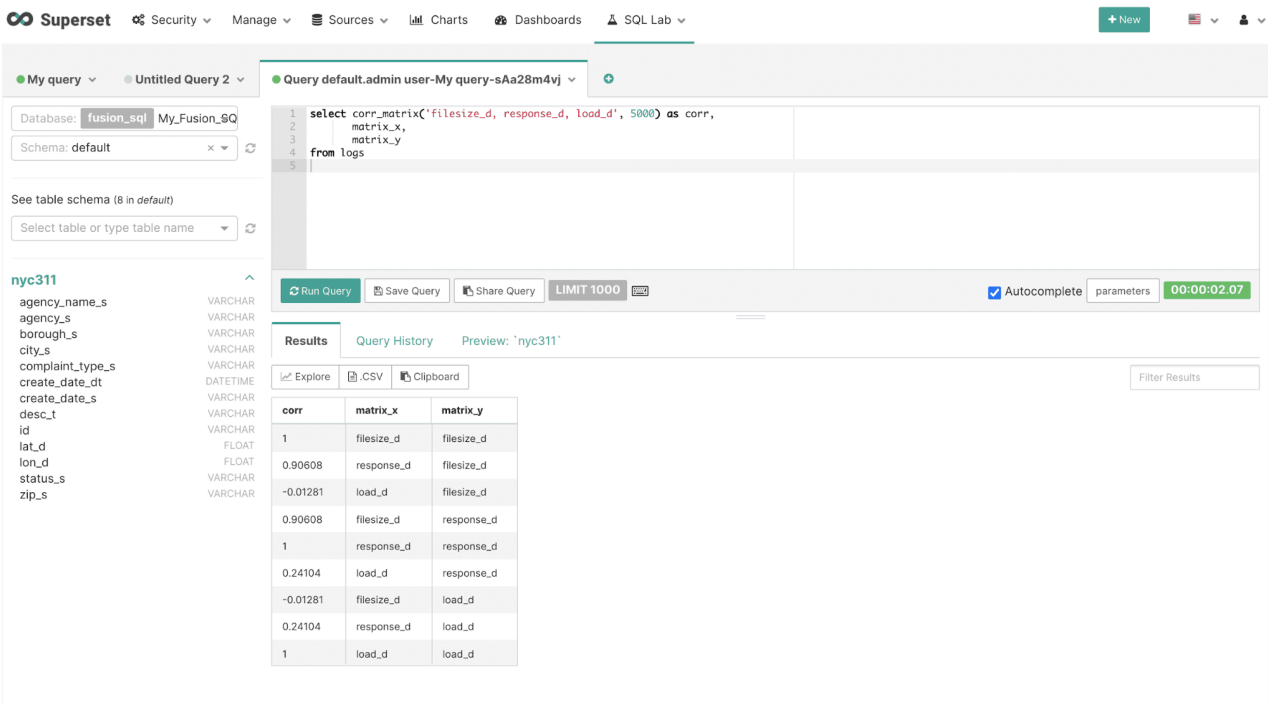


corr_matrix

Correlation matrices can be computed using the `corr_matrix` function. The `corr_matrix` function takes two parameters:

- A string, enclosed in single quotes, containing a comma-separated list of numeric fields for which to calculate the matrix.
- The sample size to compute the correlation matrix from.

The result set for the corr_matrix function contains one row for each two field combination listed in the first parameter. The `corr_matrix` function returns the correlation for the two field combination. There are two additional fields, `matrix_x` and `matrix_y` that contain the values for the field combination for the row.



After the results have been displayed they can be visualized by clicking on the **Explore** icon.

Once on the chart creation screen select the **Heatmap** visualization. The heatmap can be easily configured by selecting the `matrix_x` field in the **X** dropdown and the `matrix_y` field in the **Y** dropdown. In the **Metric**

dropdown select the `corr` field which holds the correlation value for each X/Y combination. Be sure to use the **Custom SQL** tab and remove the parentheses from the `corr` field name so an aggregation is not applied to the `corr` value.

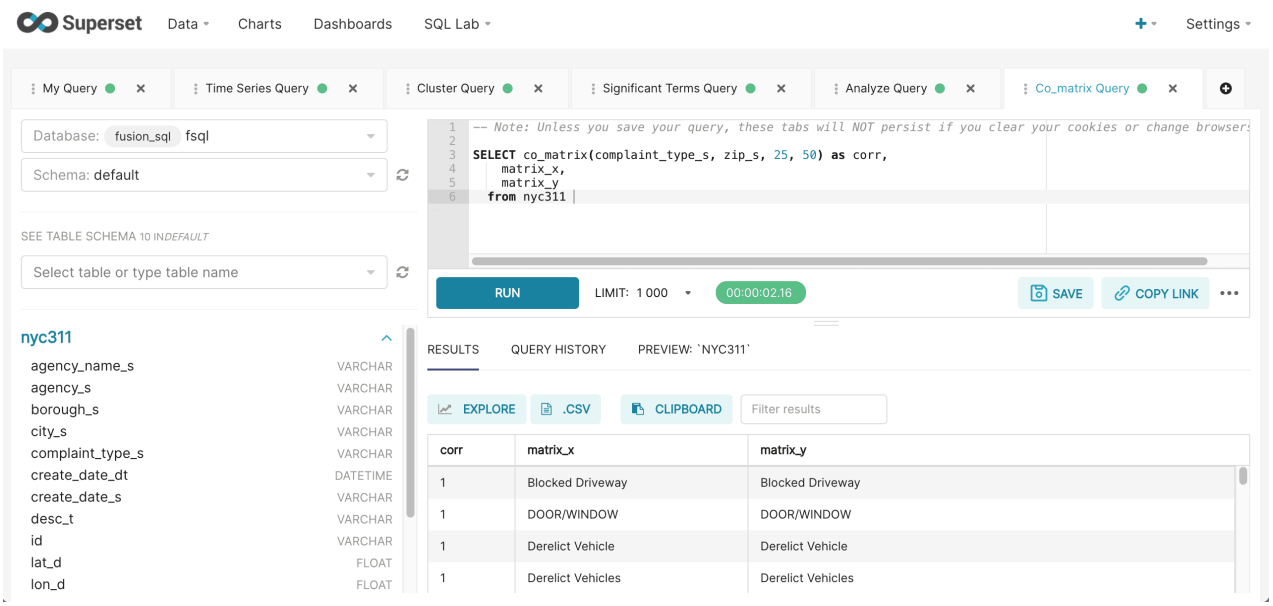
The example below shows the configuration and heatmap visualization for the `corr_matrix` function:

co_matrix

The `co_matrix` function returns a matrix that shows the correlation of values within a categorical field based on their co-occurrence across a different categorical field. For example in a medical database this could be used to correlate diseases by co-occurring symptoms. In the example below the `co_matrix` function is used to correlate complaint types across zip codes in the NYC 311 complaint database. This can be used to better understand how complaint types are correlated across zip codes. The `co_matrix` function has 4 parameters:

- The categorical String field that will be correlated.
- The categorical String field that will be used for co-occurrence.
- The number of categorical variables to correlate.
- The number of categorical variables to calculate co-occurrence from.

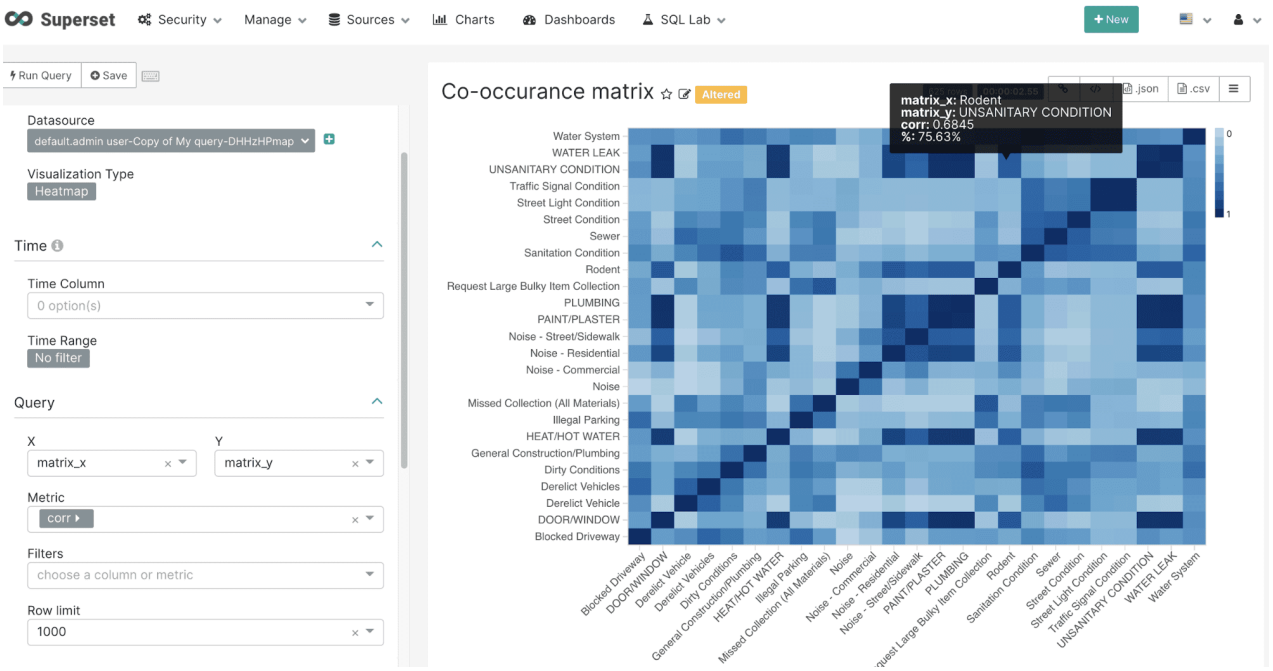
The result set for the `co_matrix` function is a correlation matrix for the first categorical field parameter. The `co_matrix` function returns the correlation for each row. The `matrix_x` and `matrix_y` fields contain the combinations of the top N categorical values.



After the results have been displayed they can be visualized by clicking on the **Explore** icon.

Once on the chart creation screen select the **Heatmap** visualization. The heatmap can be configured by selecting the `matrix_x` field in the X dropdown and the `matrix_y` field in the Y dropdown. In the **Metric** dropdown select `corr` return field which holds the correlation value for each X/Y combination. Be sure to use the Custom SQL tab and remove the parentheses from the `corr` field name so an aggregation is not applied to the `corr` value.

The example below shows the configuration and heatmap visualization for the `co_matrix` function:



Use Superset with Managed Fusion SQL

How-to

> Compatible versions: 5.5
>
> Works with: *Managed Fusion*

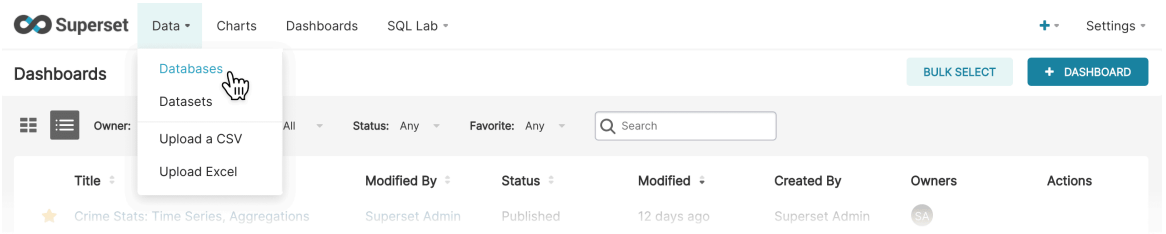
Apache Superset is a powerful and widely used SQL exploration and visualization tool. This topic explains different ways to use it with Managed Fusion.

In Managed Fusion 5.5, Superset is deployed by default as part of your Managed Fusion platform.

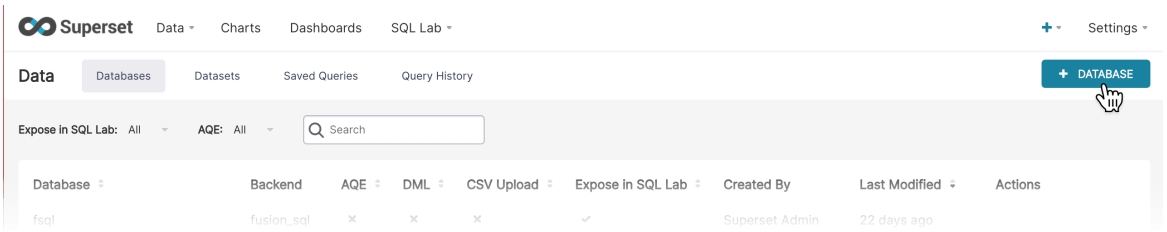
Create a Superset database

After the SQLAlchemy dialect is installed it's time to set up a connection to the database.

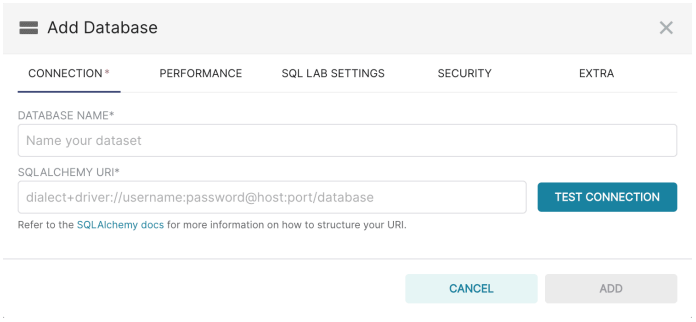
1. In Superset, navigate to **Data** > **Databases**:



2. Click the **+ Database** button in the upper right corner to create a new database:



The **Add Database** screen appears:



3. Add a name for the database.
4. Add the SQLAlchemy URI in the following format:

```
fusion_sql://USERNAME:PASSWORD@<fusiondomain>:
<fusion_ssl_port>/fusion/default
```

Note

The Superset connector uses SSL for connections and queries and will only connect to the Fusion SSL port.

If this command does not work, it is possible there are issues with reaching your Fusion proxy. Enter the following to add the SQLAlchemy URI:

```
fusion_sql://USERNAME:PASSWORD@<fusion_proxy>:
<fusion_proxy_port>/fusion/default
```

Note

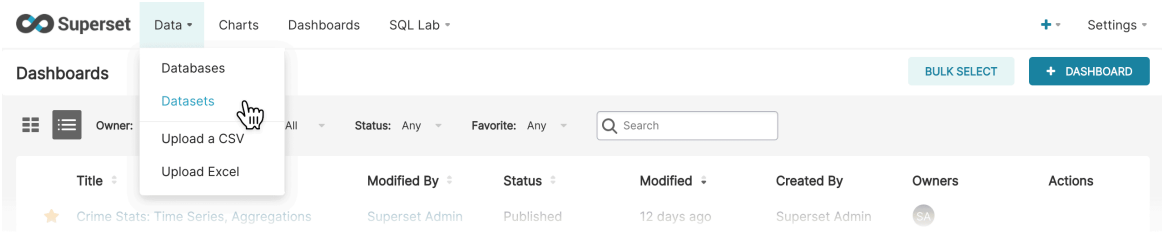
The superset SQL selects are written to the SQL Service Pod log for review and troubleshooting purposes.

5. Click **Test Connection** to make sure the connection is working.
6. Click **Add**.

Create a Superset dataset

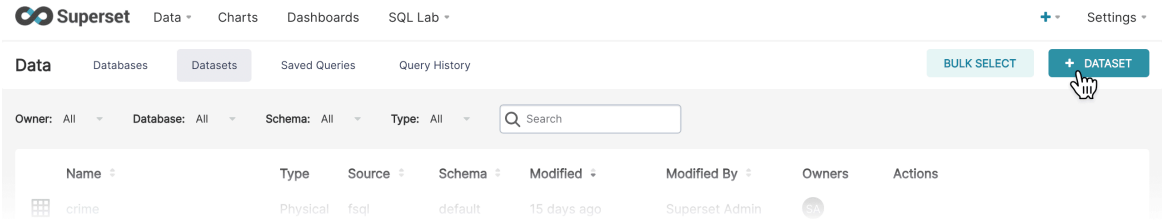
Once a database is created, datasets (also called tables) can be set up.

1. Navigate to **Data** > **Datasets**:

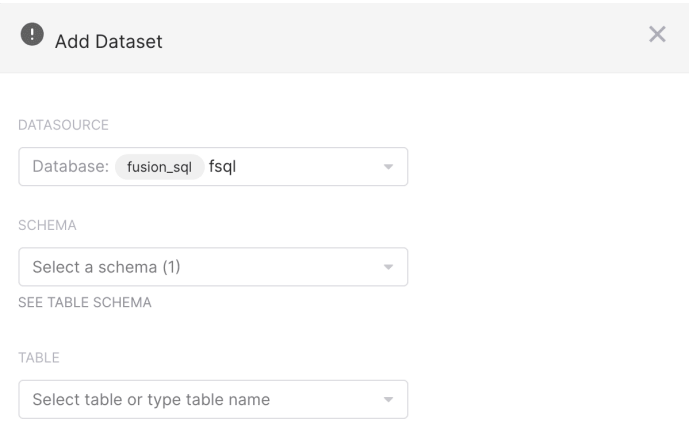


The Datasets screen appears.

2. Click the **+ Dataset** button in the upper right corner:



3. Create a dataset by specifying the **Database**, **Schema**, and **Table Name**. The schema is always “default”.

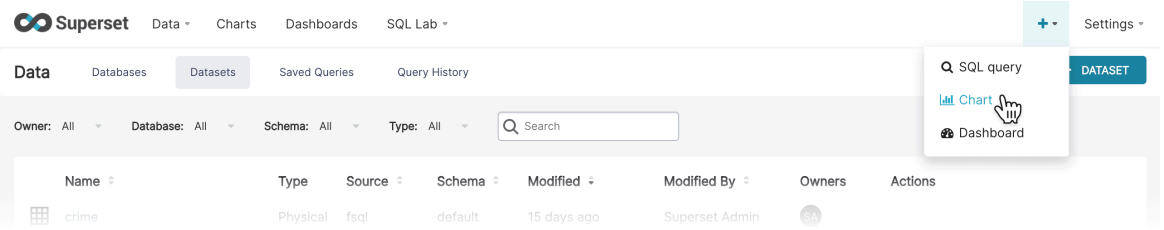


4. Click **Add** to save the dataset.

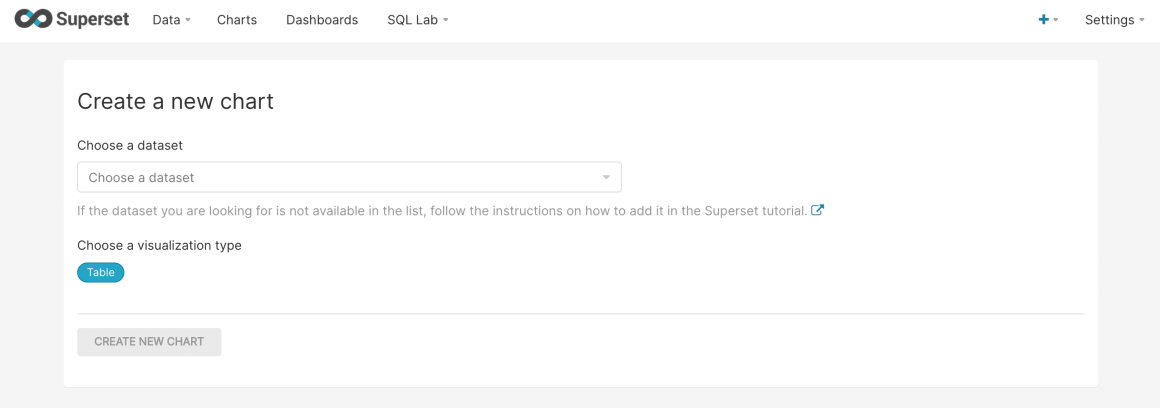
Your first chart: bar chart

In this section we'll create our first chart. More time will be spent on the process of creating charts in this section than in following sections which will only deal with the specifics of the different kinds of charts.

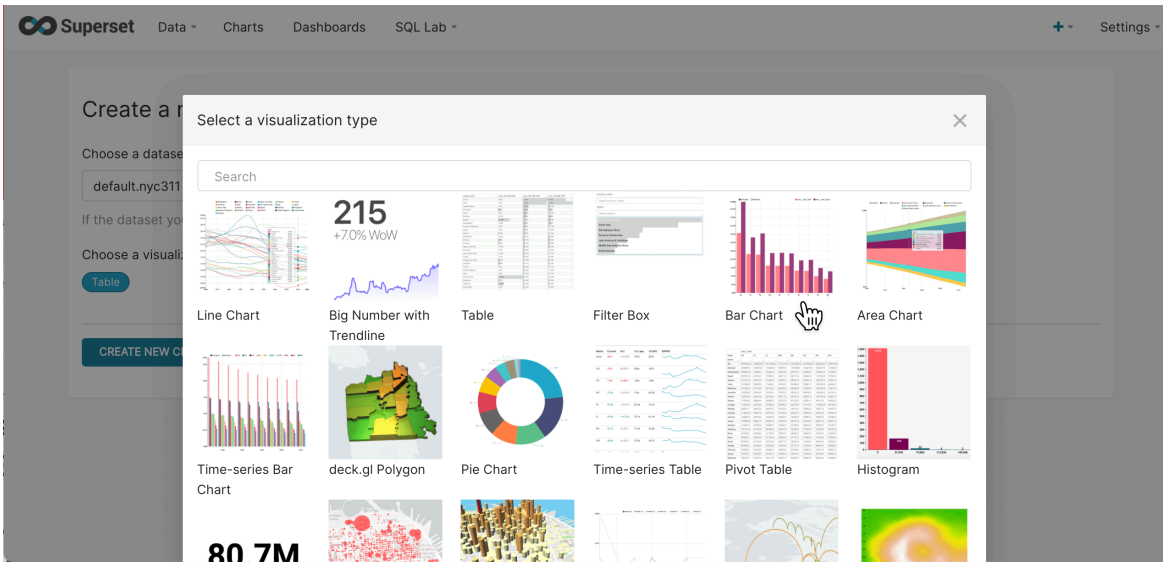
1. Navigate to **+New** > **Chart**:



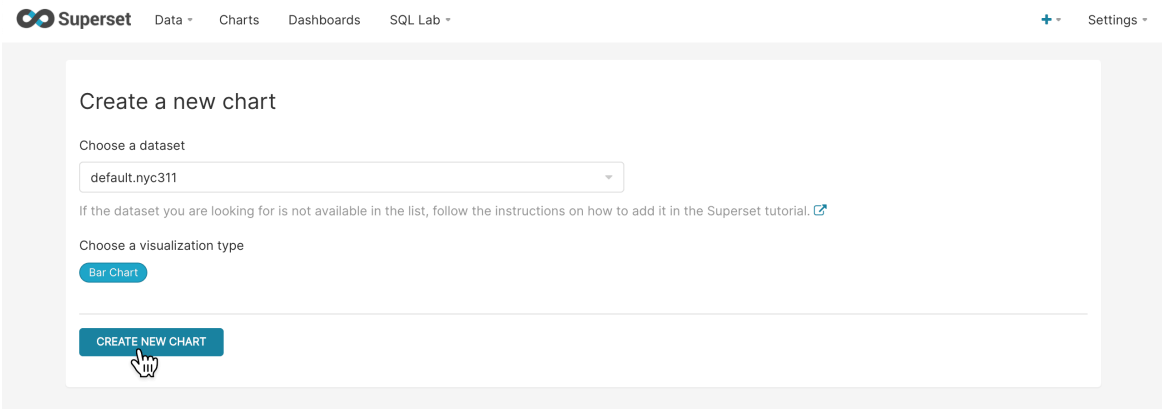
The **Create a new chart** screen appears:



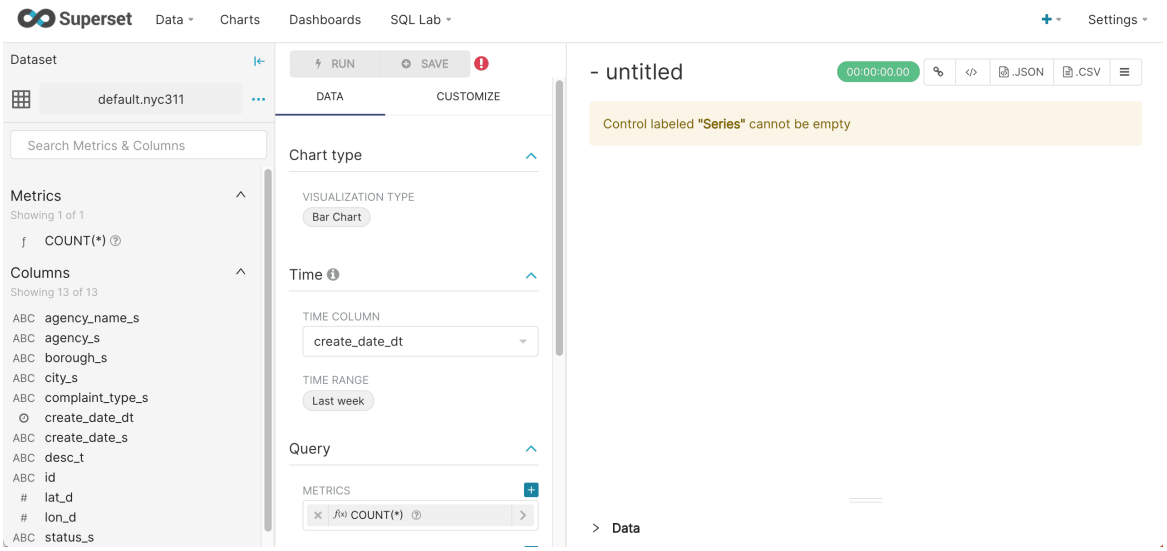
2. Select a datasource (also called a table) from which to create the chart.
3. Click **Table** to change the visualization type.
4. Select the **Bar Chart** visualization type:



5. Click **Create new chart**:



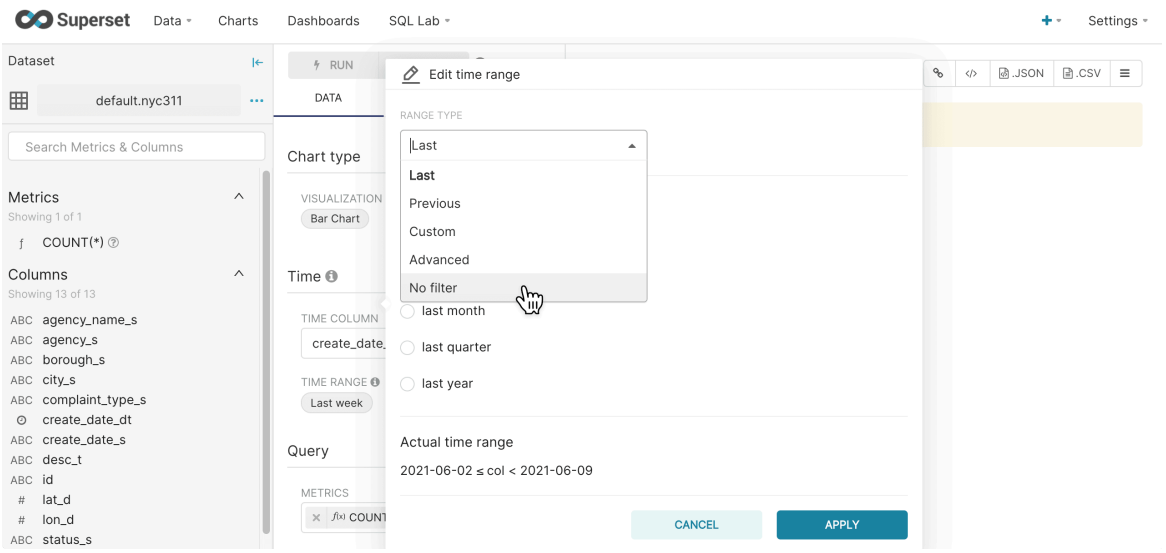
The chart configuration screen appears:



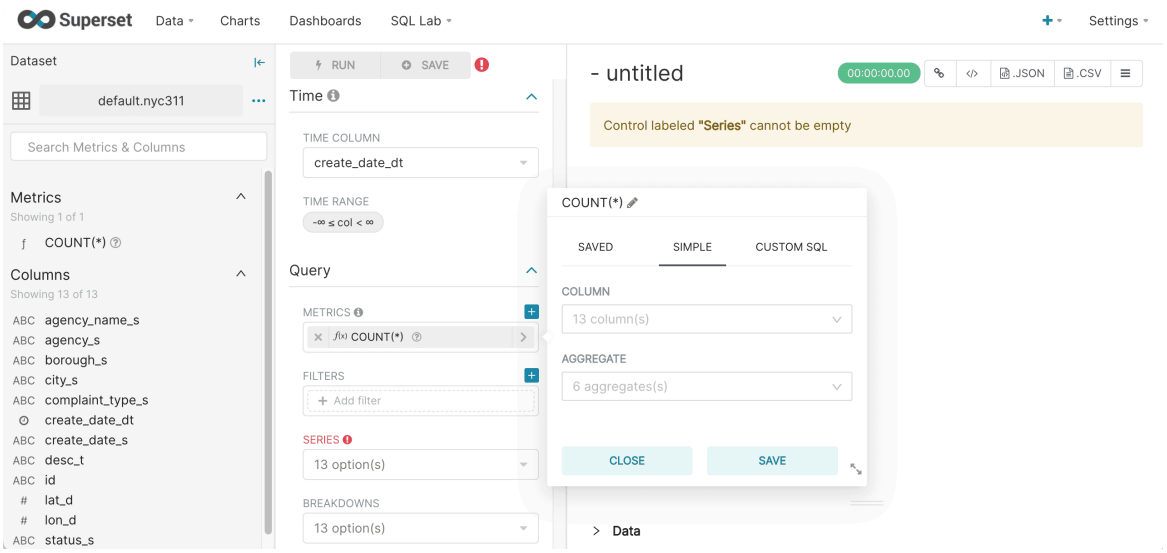
The chart can be configured using the left hand panel and will display on the right.

The first step is to turn off the **Time Range** filter. Time filters, if needed, can be added back in after seeing the initial results.

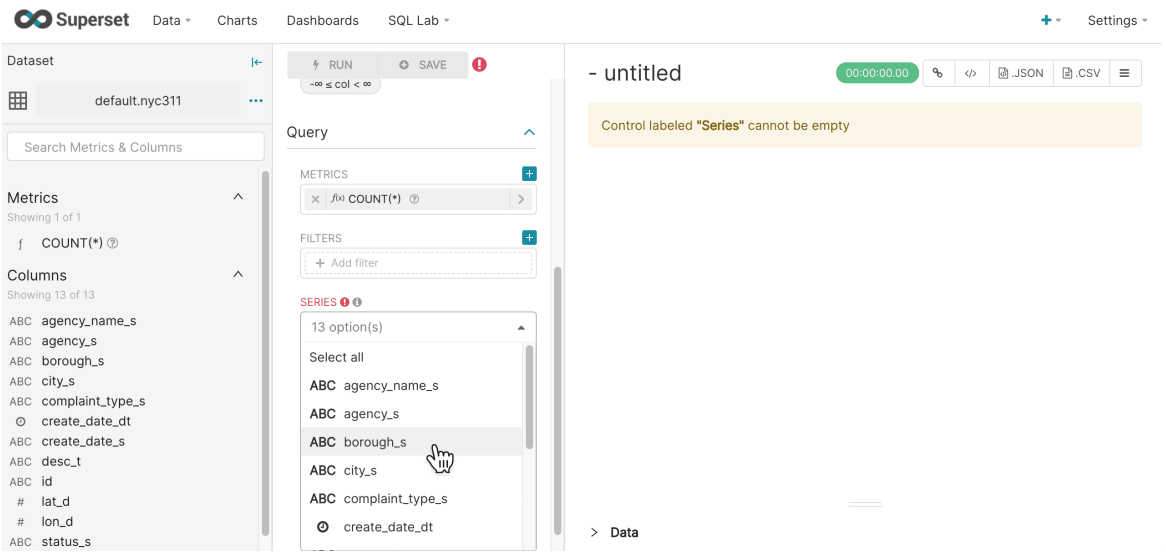
6. To turn off the Time Range filter, click **Last week**, then **Range Type**, and select **No filter**.



7. Click **Apply**.
8. Select a **Metric** from the metrics drop down.
   In this example the `COUNT(*)` metric is chosen. The metric will be the y-axis of the bar chart.

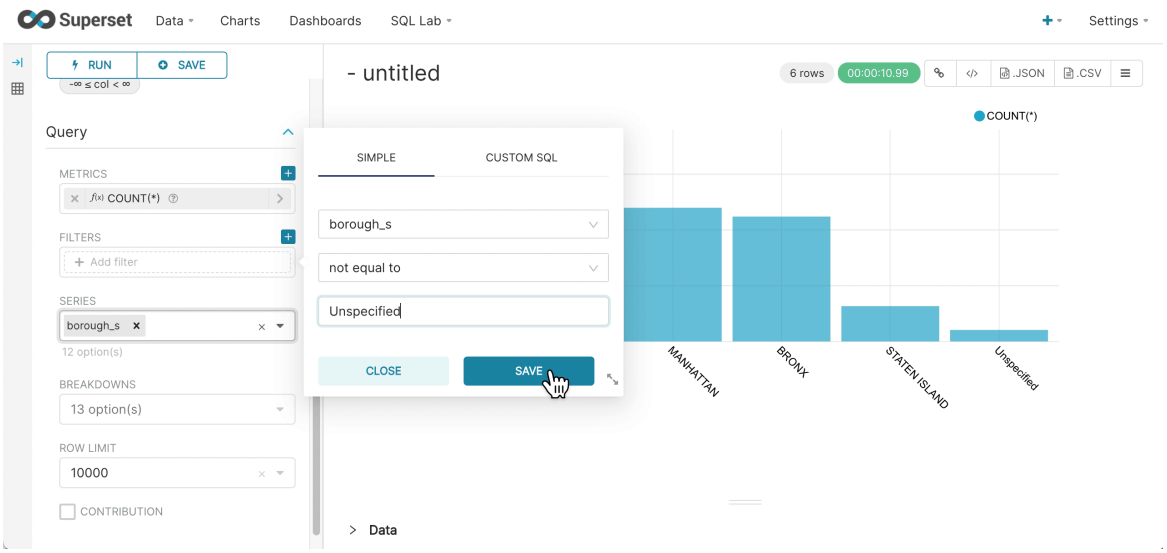


9. Select a **Series**, which will be the x-axis of the bar chart.



10. Click the **Run Query** button. This will execute the query.
If the query is successful, a bar chart appears in the right panel:

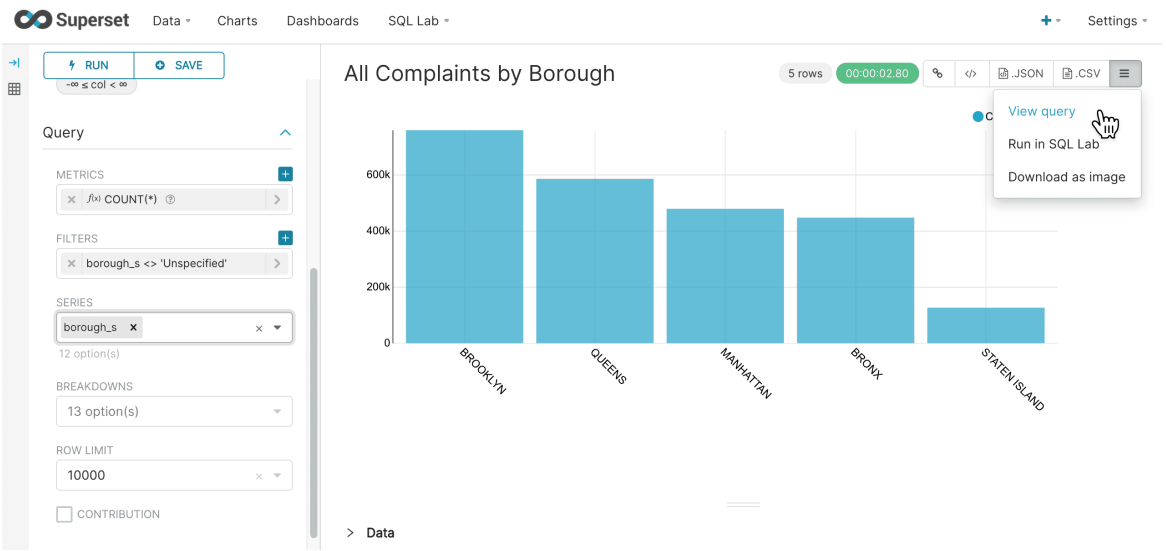
11. Add a filter by clicking on the **Filters** control:



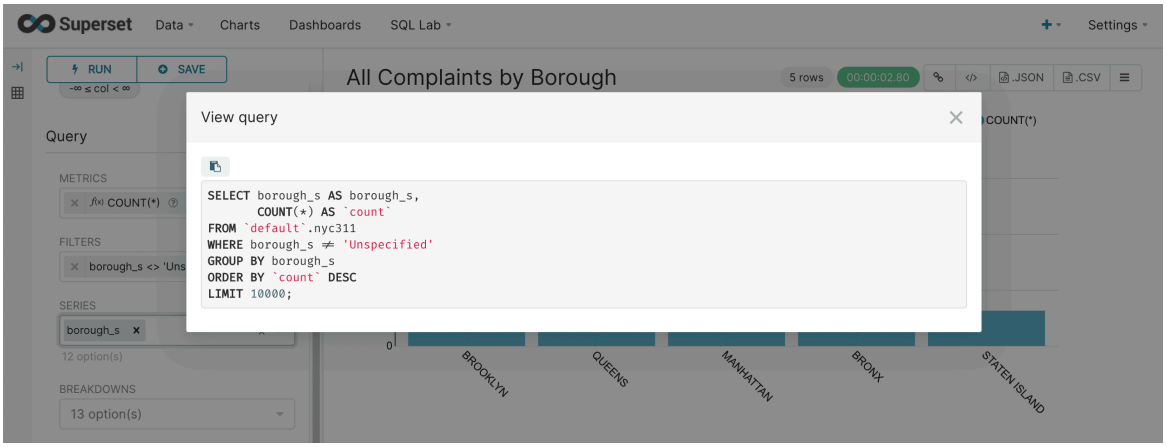
12. Click **Save**.
13. Rerun the query to see the results:

14. Click on the “Untitled” chart title to give the chart a new title:

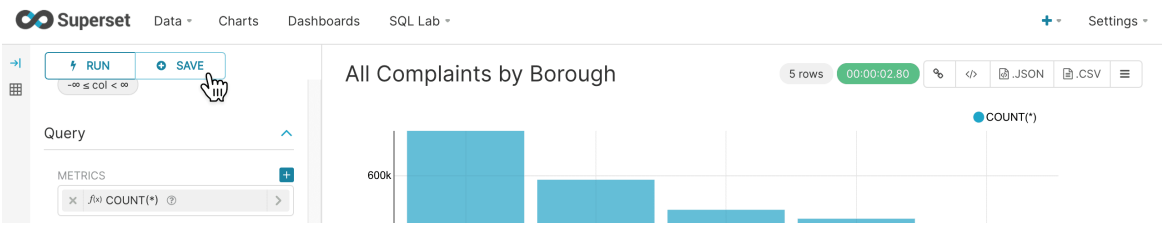
15. Select **View query** in the upper right to see the SQL query that was sent to Fusion SQL:



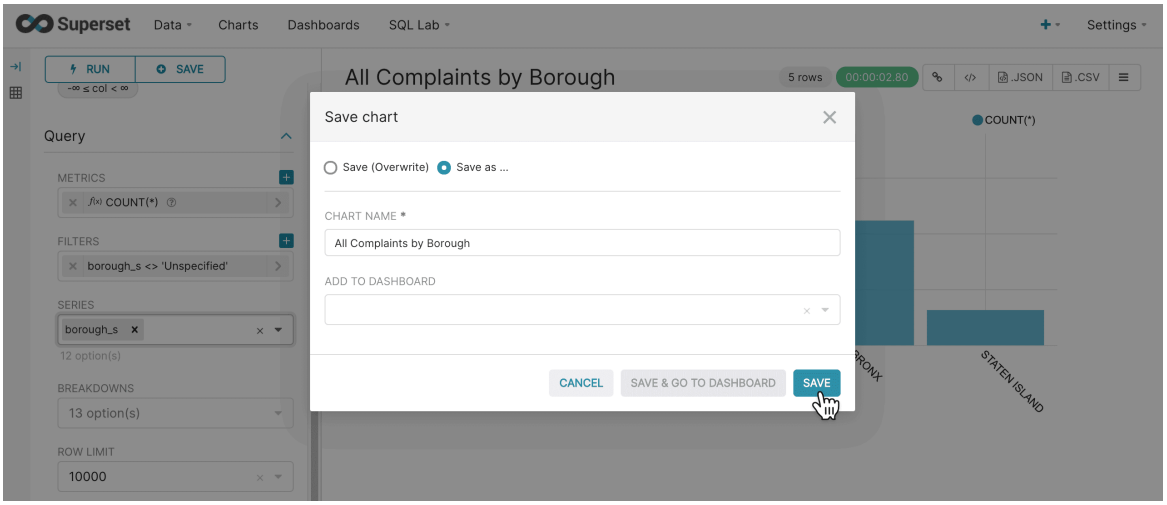
This brings up the query to review:



16. When the chart is complete, click **Save** in the upper left:



17. Enter a name for this chart and click **Save**:

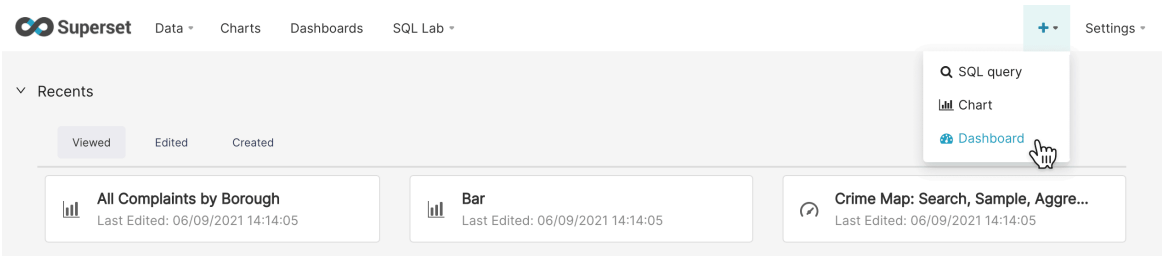


Once the chart is saved it can be added to dashboards later.

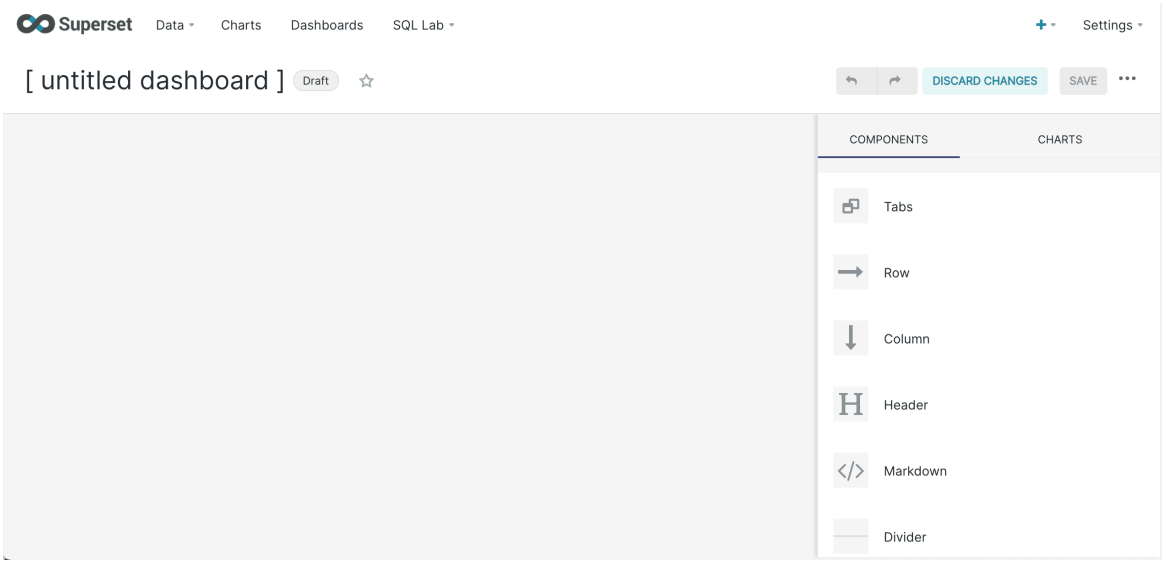
Dashboards

Once we have a chart saved we can add it to a dashboard.

1. Navigate to **+New** > **Dashboard** in the upper right hand corner:

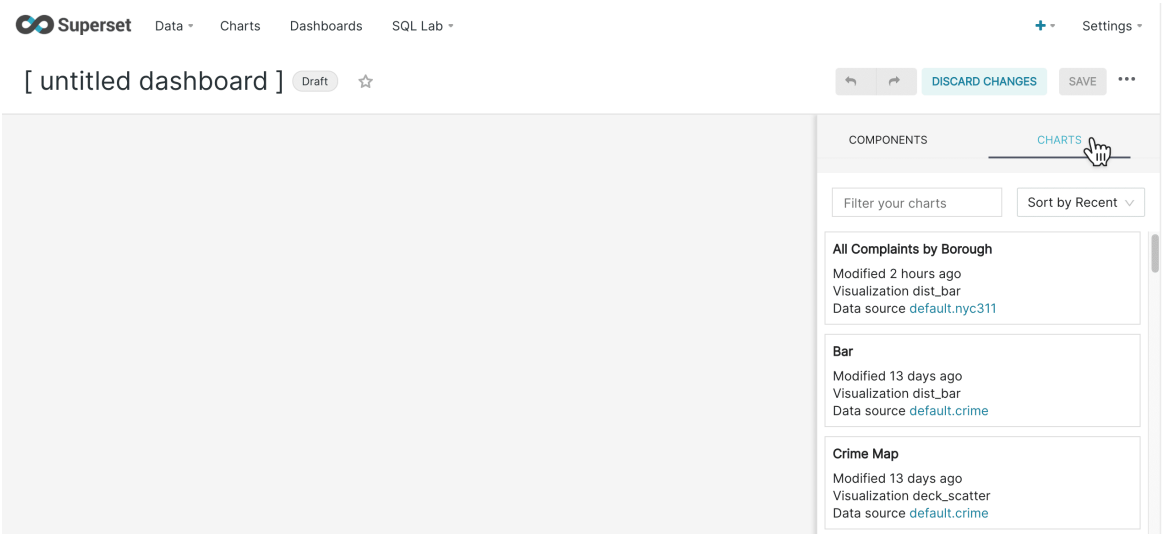


This will take you to the dashboard screen:



Dashboards are created by dragging components onto the dashboard.

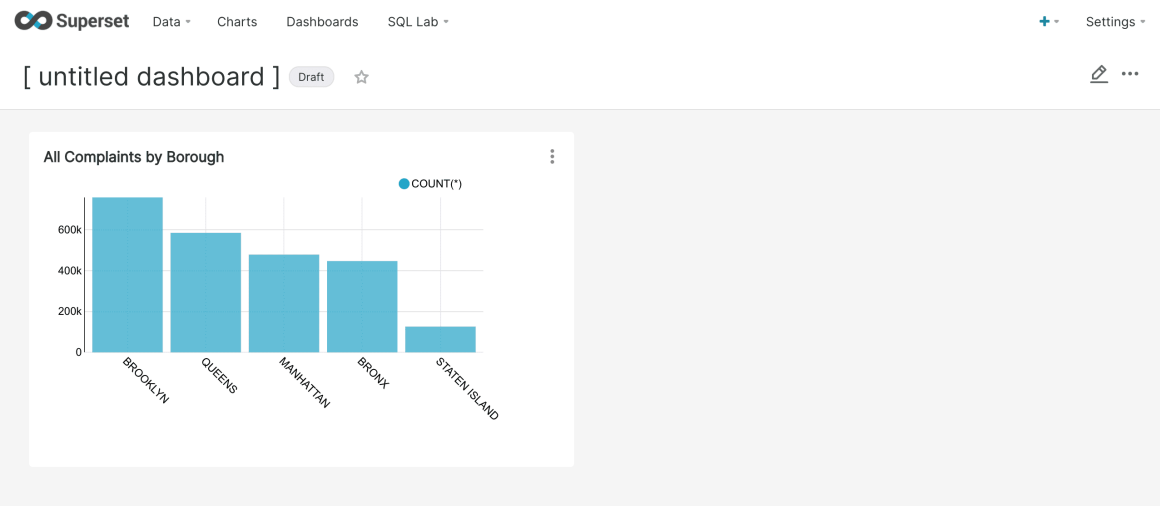
2. Click **Chart** to find your saved charts:



3. Drag your bar chart onto the dashboard and resize it by mousing over the bottom right corner and dragging the chart.
4. When the chart is sized properly, click **Save Changes**.
   You can also title your dashboard by clicking on the title and changing it.

Note

To drag items onto the screen, move them to the top of the component they will be placed in. A blue line will appear on the screen when the item is ready to be dropped. If the blue line is not on the screen, the item will not be dropped.

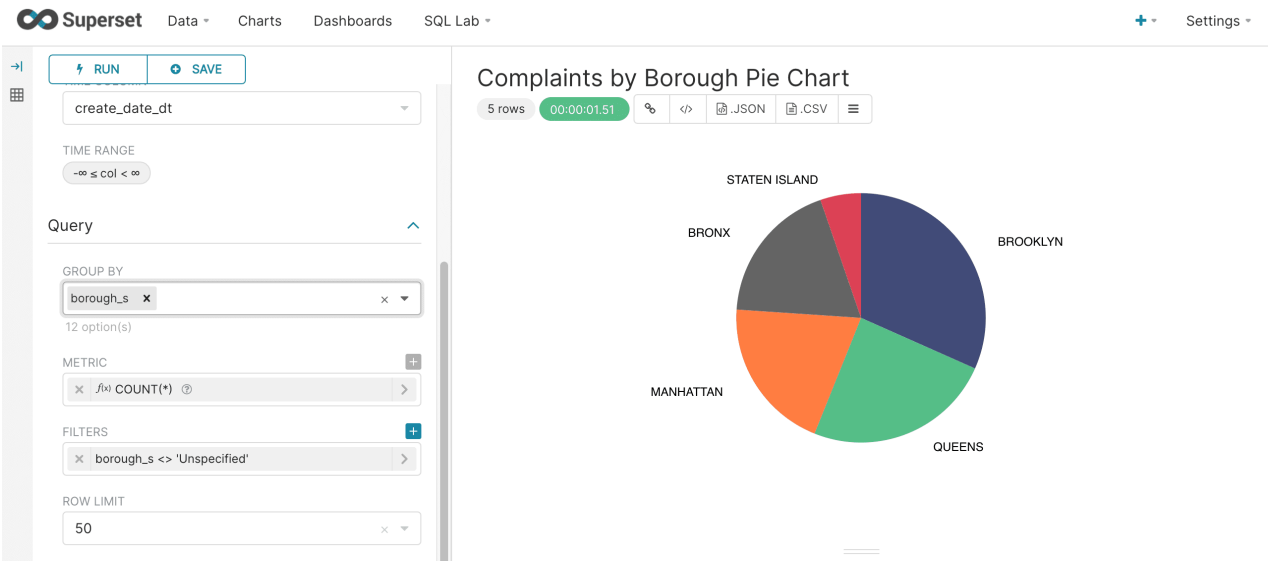


You can continue to add charts to your dashboard as more are created.

More Charts

Pie chart

Now that we have a basic understanding of charts, it's very easy to create a pie chart. In fact, pie charts are very similar to bar charts. Below is an example pie chart:



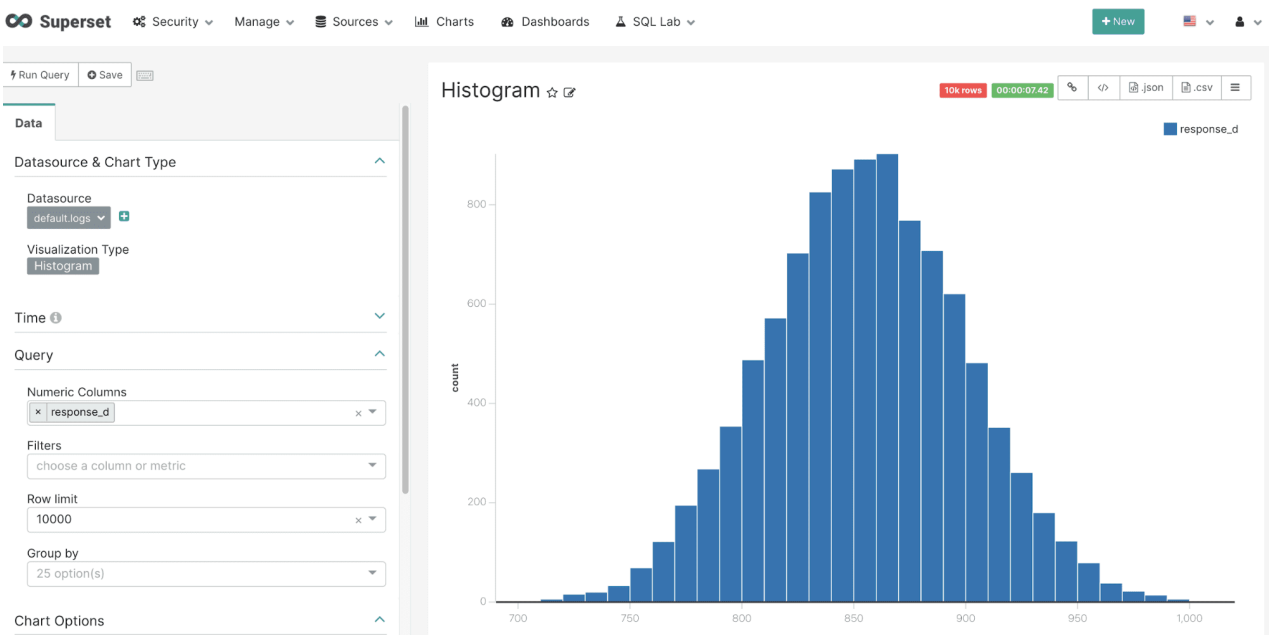
Notice that the **Group By** dropdown box is used instead of the **Series** dropdown in the bar chart. Otherwise the charts are configured the same way.

Histograms

Superset has a Histogram visualization which works perfectly with Fusion SQL's random sampling capability.

The example below shows the simple configuration for a histogram. A numeric field is selected from the Numeric Columns drop down list and the row limit is set to 10000. Filters can be applied as needed. The number of bins can be controlled under **Chart Options**.

Once configured, Superset issues a basic select to Fusion SQL for the numeric field. In Fusion SQL all basic selects without an `ORDER BY` clause return random samples. This random sampling capability allows for histograms that can be used to represent much larger data sets.

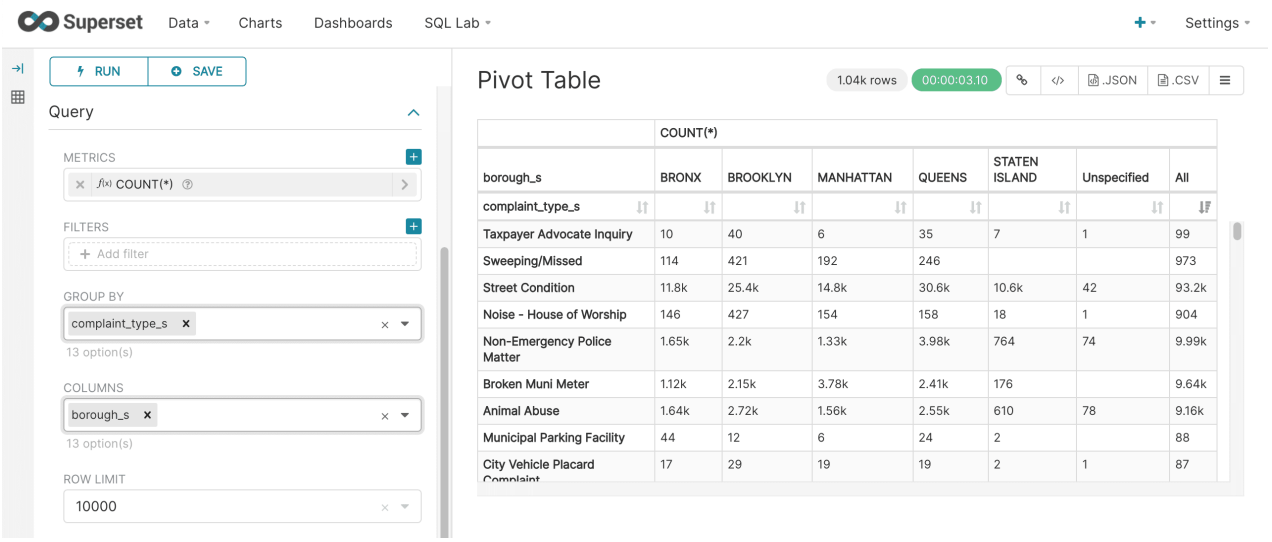


Viewing the query for this chart shows that no `ORDER BY` is used with the query resulting automatically in a random sample:

Pivot table

Superset's Pivot Table visualization can be used to visualize a two dimensional aggregation. In the example below we see that the Pivot Table control panel has both a **Group By** and **Columns** dropdown. In this example the `complaint_type_s` field is selected as the **Group By** field and the `borough_s` field is selected as the **Columns** field. The `COUNT(*)` metric is being calculated across both fields.

This Pivot Table is then displayed with rows showing complaint types and the columns showing the boroughs. The count of complaints for each complaint type and borough populates the cells.



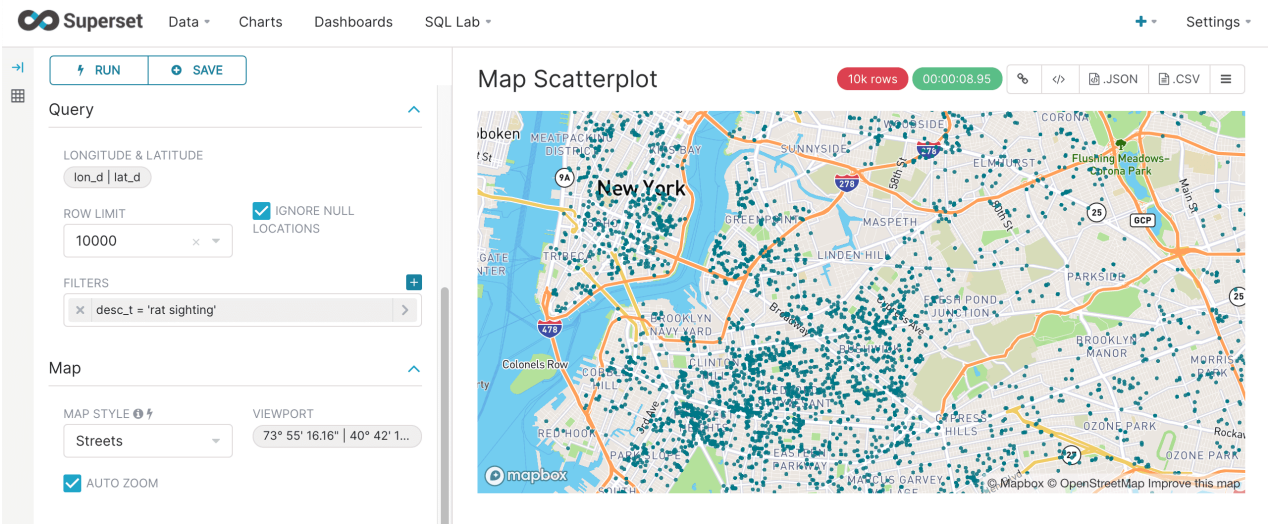
	COUNT(*)						
borough_s	BRONX	BROOKLYN	MANHATTAN	QUEENS	STATEN ISLAND	Unspecified	All
complaint_type_s							
Taxpayer Advocate Inquiry	10	40	6	35	7	1	99
Sweeping/Missed	114	421	192	246			973
Street Condition	11.8k	25.4k	14.8k	30.6k	10.6k	42	93.2k
Noise - House of Worship	146	427	154	158	18	1	904
Non-Emergency Police Matter	1.65k	2.2k	1.33k	3.98k	764	74	9.99k
Broken Muni Meter	1.12k	2.15k	3.78k	2.41k	176		9.64k
Animal Abuse	1.64k	2.72k	1.56k	2.55k	610	78	9.16k
Municipal Parking Facility	44	12	6	24	2		88
City Vehicle Placard Complaint	17	29	19	19	2	1	87

Map scatter plots

Superset has a number of powerful mapping visualizations. The `deck.gl.Scatterplot` is used to display a scatter plot of lat/lon points on a map. This particular visualization works perfectly with Fusion SQL's filtering and random sampling capability.

In the example below the `lat` and `lon` fields in the table are configured and a row limit of 5000 is selected. A filter of "Rat Sighting" in the `desc_t` field is added to select complaints with "Rat Sighting" in the description field. This will return a random sample of 5000 lat, lon points with "Rat Sighting" in the description. The points can then be plotted on the map which can be zoomed and scrolled.

The random sample of points is a useful tool for revealing high density clusters on the map. This works because the probability of returning a point from a high density area is higher than a low density area.



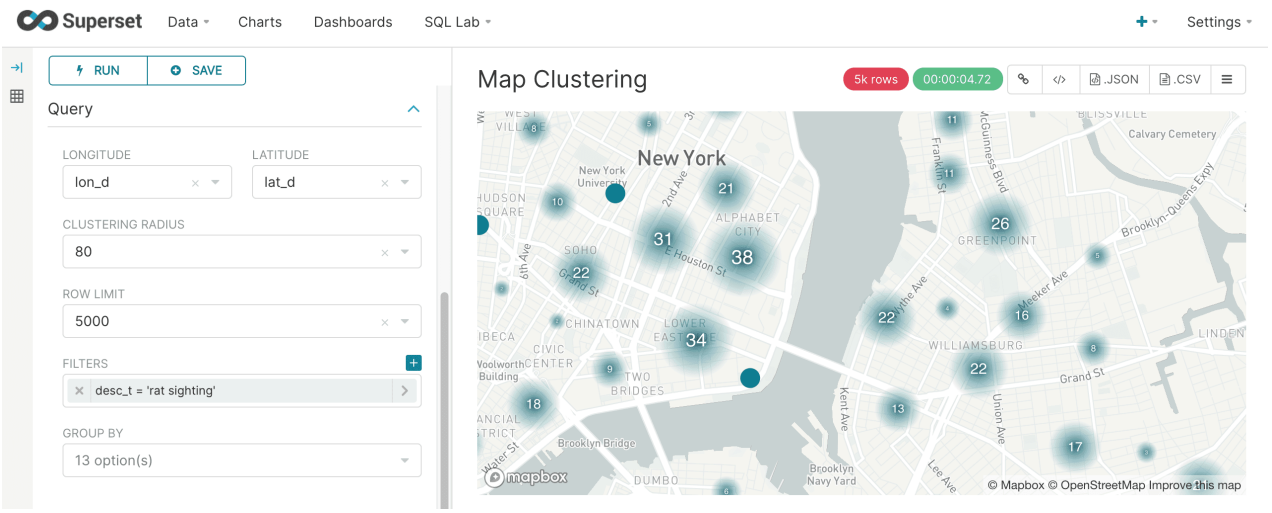
Note

For the map to render correctly, a Map Box key must be available for Superset at run time. In this example the Map Box key was exported to a unix environment variable before starting Superset as follows:

```
export
MAPBOX_API_KEY=pk.eyJ1IjoiZnVzaW9uc3FsIiwiYSI6ImNrOWtudX
pzazAyaW8zbHFwOG50NXVyMGgifQ.ZTyPhJaYQEyqvWY5f-mWbw
```

Map clustering

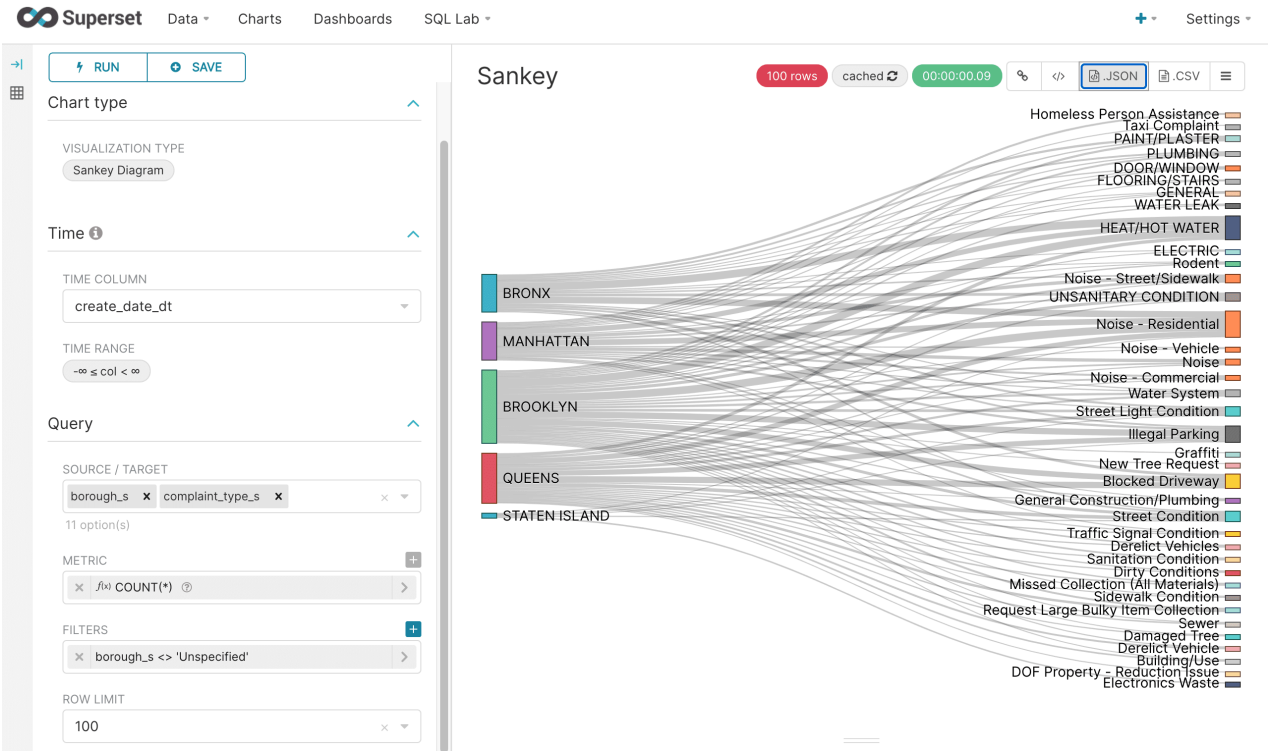
The Superset MapBox visualization is another powerful tool for visualizing geo-spatial data. Instead of a scatter plot the MapBox visualization clusters points on the map. Configuration of the MapBox visualization is very similar to the `deck.gl.Scatterplot` but with an added control for cluster radius. The example below shows the configuration and rendering of the MapBox visualization:



Sankey

The Sankey Diagram is another way to visualize a multi-dimensional aggregation. With a Sankey Diagram the strength of the relationships between dimensions is shown by the width of the pathways between them.

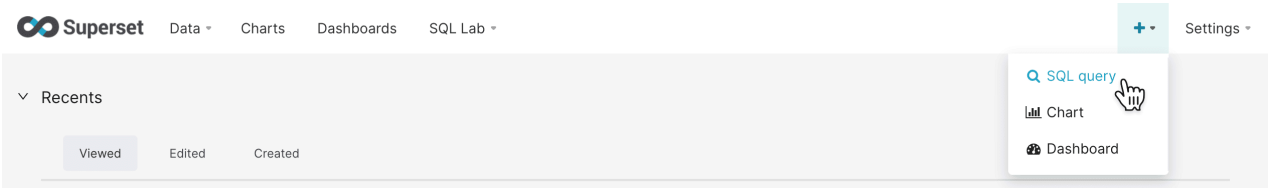
In the example below a Sankey Diagram is configured by having the borough as the source and the complaint type as the target. The `COUNT(*)` metric is used to measure the strength of relationship between borough and complaint type. Under the covers a multi-dimensional SQL aggregation is run on Fusion SQL to create the data for the diagram.



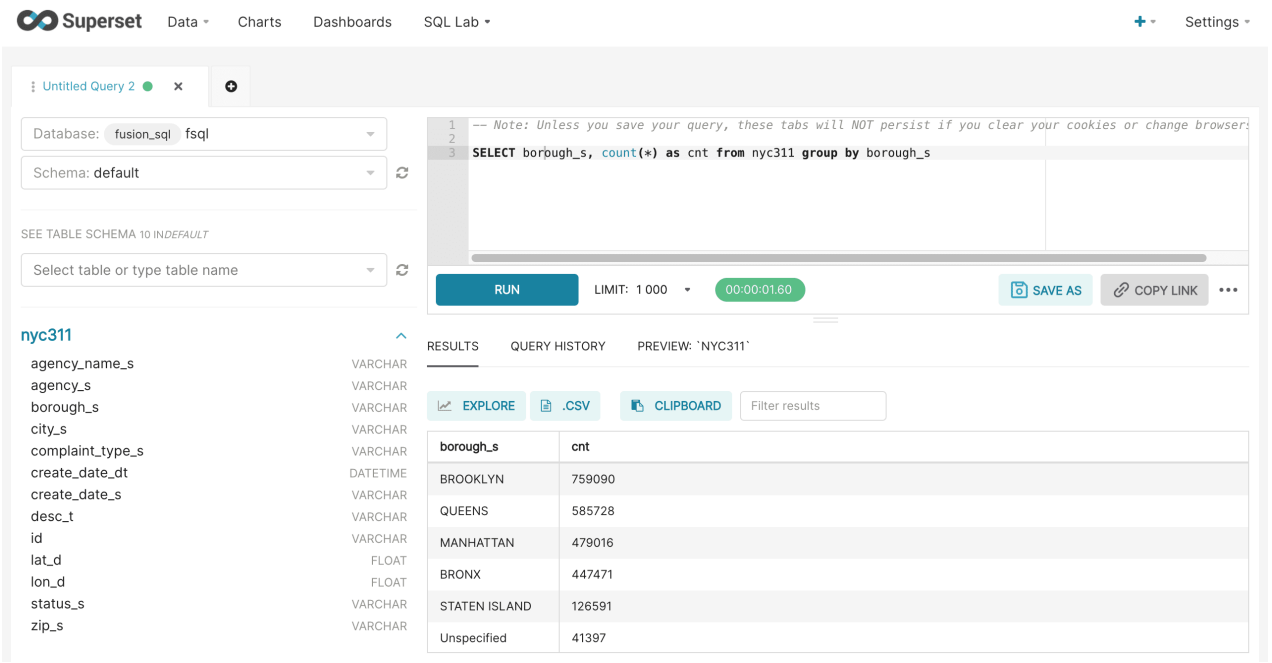
SQL Lab

Up until this point we've been creating visualizations without actually writing any SQL directly. In the next sections we'll explore how to use Superset's SQL Lab to write SQL queries directly and then visualize the results.

The SQL Lab can be accessed by choosing the **SQL Query** option under the **New** button in the upper right corner:



The SQL Lab screen shows the meta-data for the databases, schemas and tables on the left side. In the main section of the screen there is a box for typing in SQL queries. The results display in a table below:



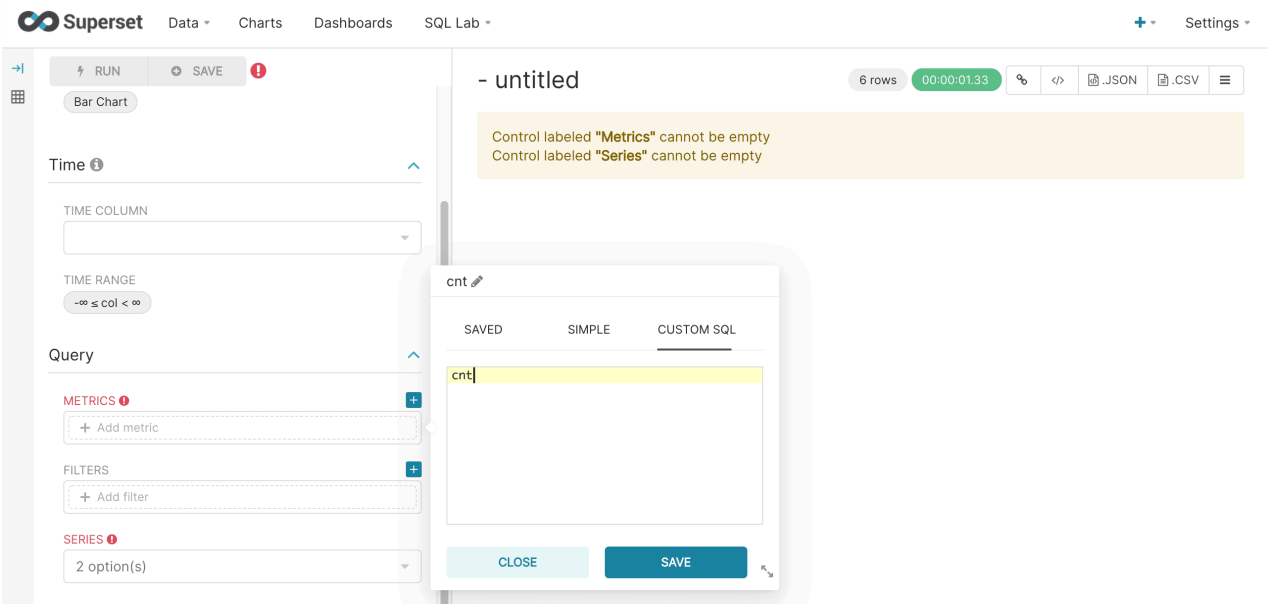
It's important to note that Superset controls the `LIMIT` through the `LIMIT` control in the user interface. This will override any limits added to the query. Otherwise the SQL query will be issued exactly as it's entered.

In this example a SQL aggregation is run and the results are displayed.

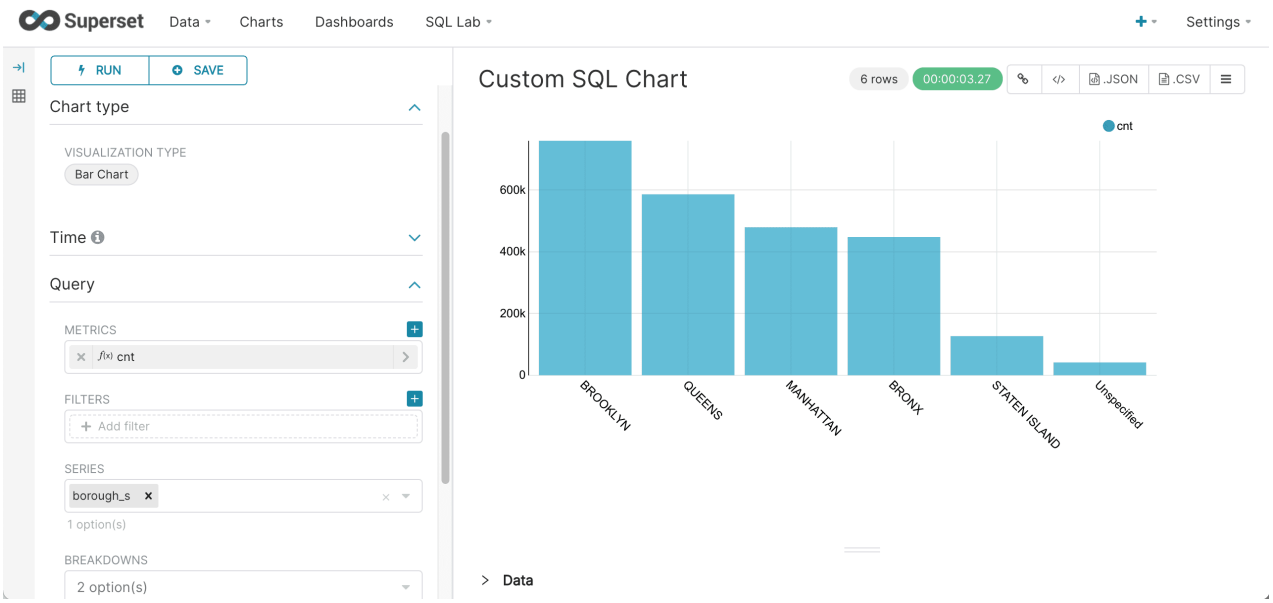
Once the results table is displayed the Explore button can be used to visualize the results. After clicking the Explore button the familiar chart creation screen will load.

The screenshot below shows the first steps in configuring the chart for this custom SQL query. First the **Visualization Type** is changed to "Bar Chart". Then the **Time Range** is set to "No filter".

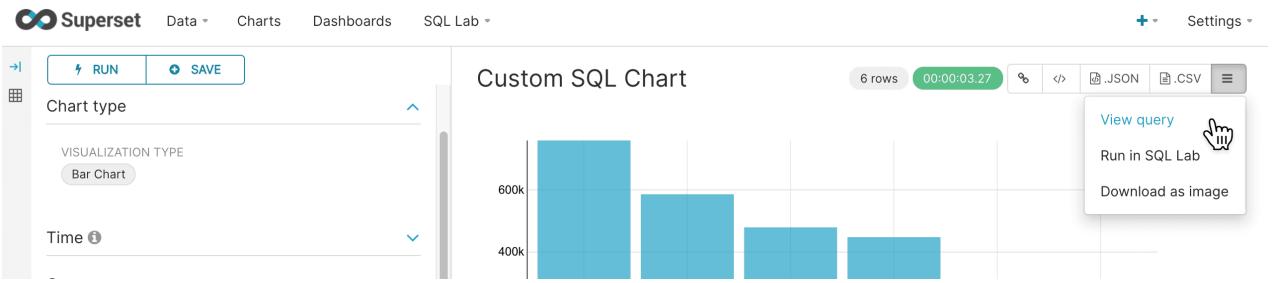
Then the `cnt` field from the result set is selected as the Metric. Remember that in the SQL query the alias for the `COUNT(*)` metric was `cnt`. Superset by default will want to perform an aggregation on top of this, so we'll choose the Custom SQL tab to override this behavior. In the Custom SQL tab simply remove the parentheses from around the `cnt`.



Next we'll choose the group by field, `borough_s`, as the **Series** and run the query. The bar chart for our custom SQL query then renders in the right panel as normal.



We can view the query using the same approach we used for regular charts.



This will show how Superset interacts with the custom query to render the chart.

Time series aggregations

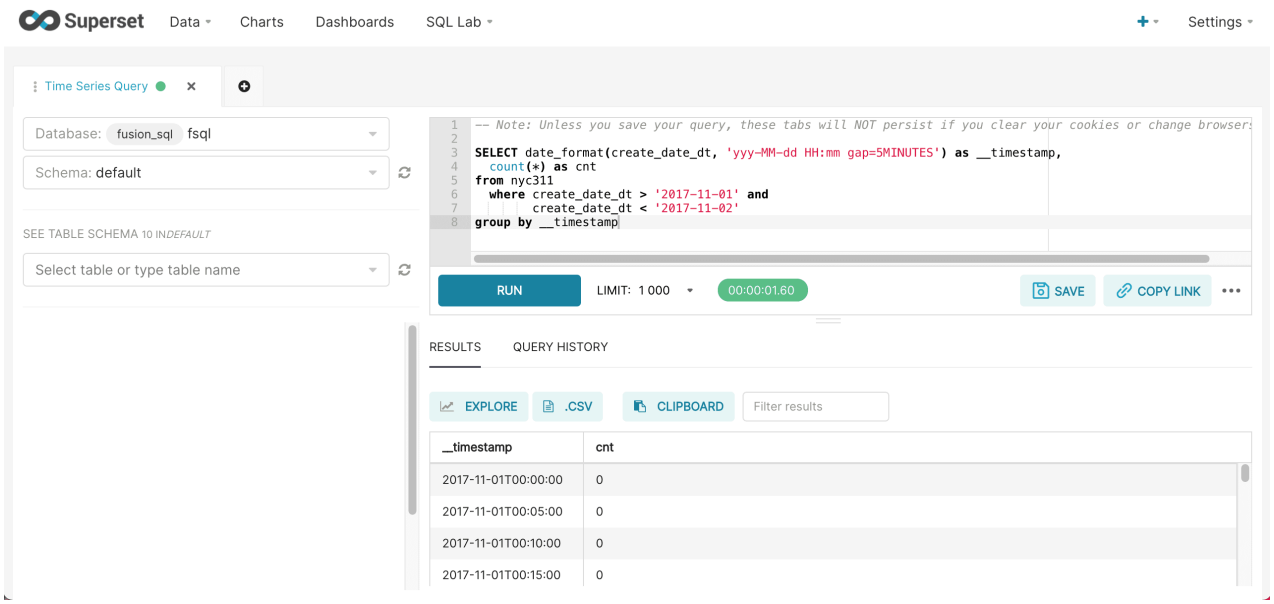
Both Fusion SQL and Superset have powerful time series capabilities. This section describes both the Fusion SQL time series syntax and the Superset visualization.

Basic time series query

Fusion SQL supports high performance and flexible time series aggregation by grouping over the date_format function. The `date_format` function takes two parameters:

- The date/time field
- The format and interval gap. The format is specified using Java SimpleDateFormat templates. The interval gap is specified by using Solr date math syntax.

The example below shows an example of a basic time series query run inside of the Superset SQL Lab:

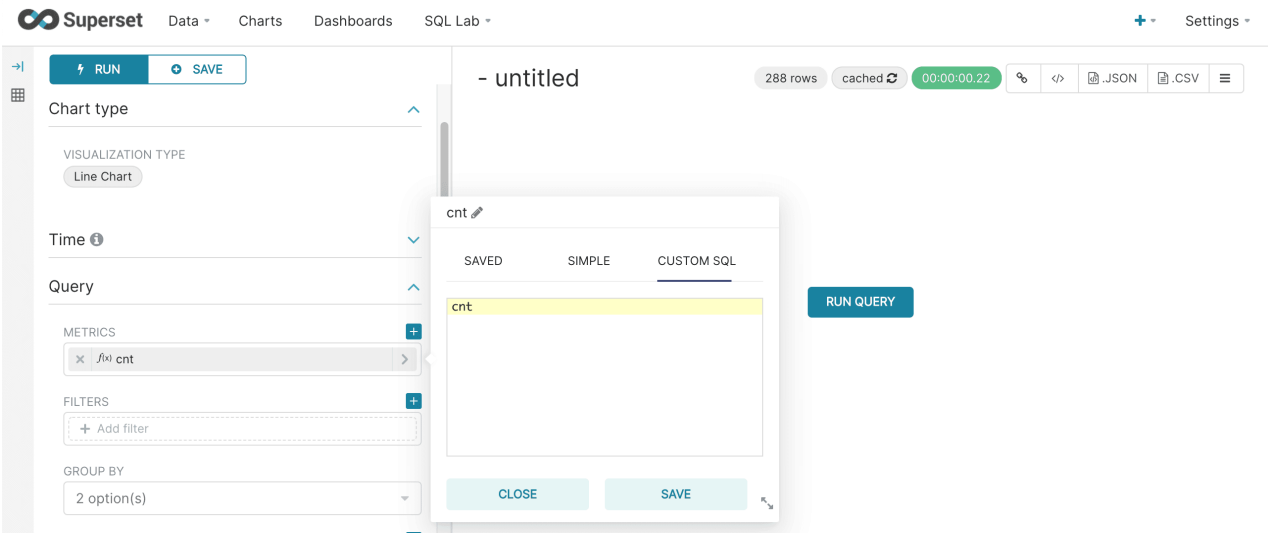


The format `yyyy-MM-dd` specifies that this is a daily time series. The alias `__timestamp` is required for the integration with Superset's time series visualization.

Once the results return we visualize by clicking on the **Explore** icon and selecting a **Line Chart**.

The Line Chart configuration is very basic because the SQL query defines all of the query filters and other details.

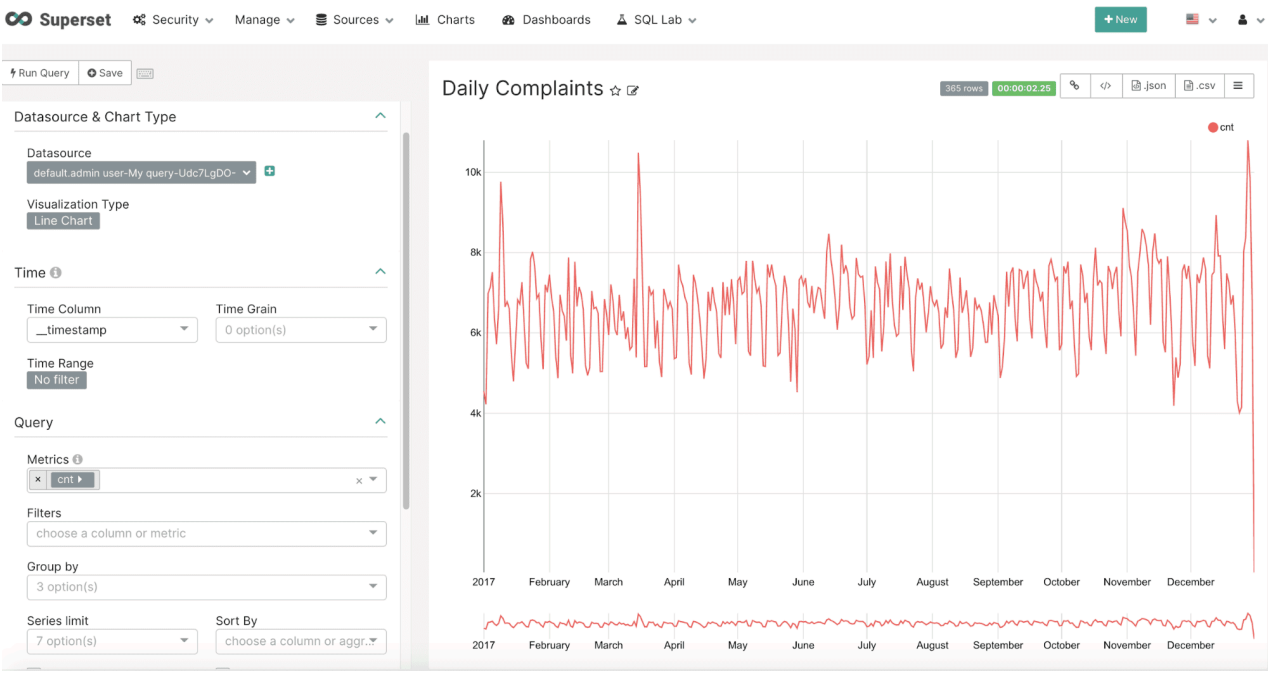
The screen below shows the configuration:



A couple of things to note about the configuration:

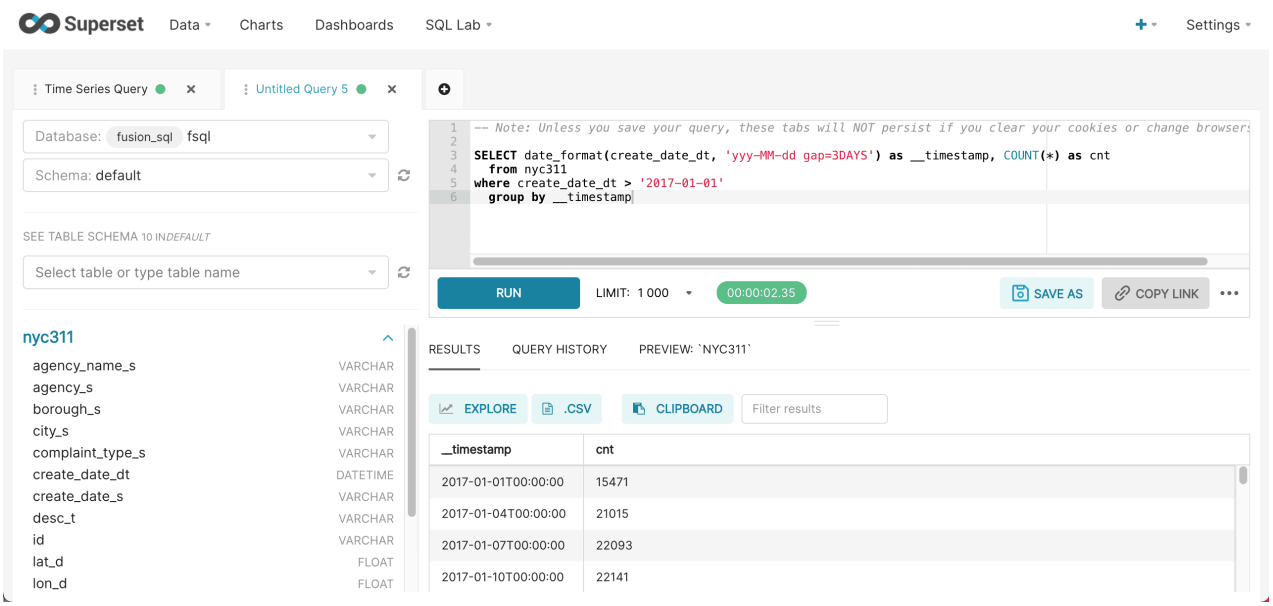
- The Time filter should be turned off. Time filters need to be applied in the SQL query.
- The time column will automatically populate with `__timestamp` field in the result set.
- The metric should be set to the `cnt` field. Use the Custom SQL and remove the parentheses to use the value in the `cnt` field without an aggregation applied to it.

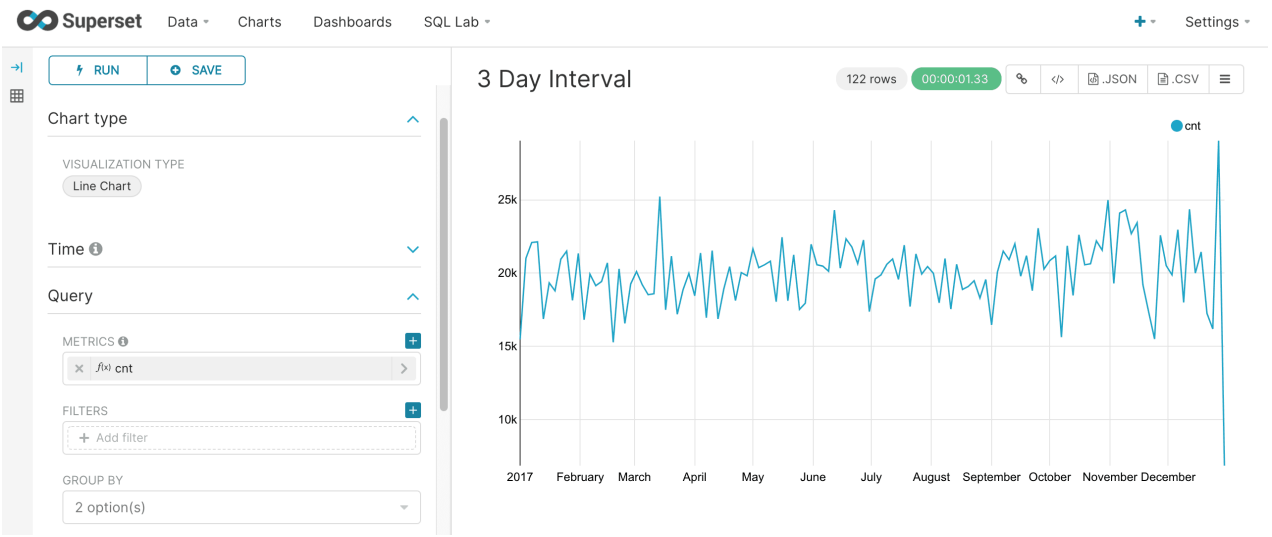
After running the query the chart will render as a time series:



Interval gap

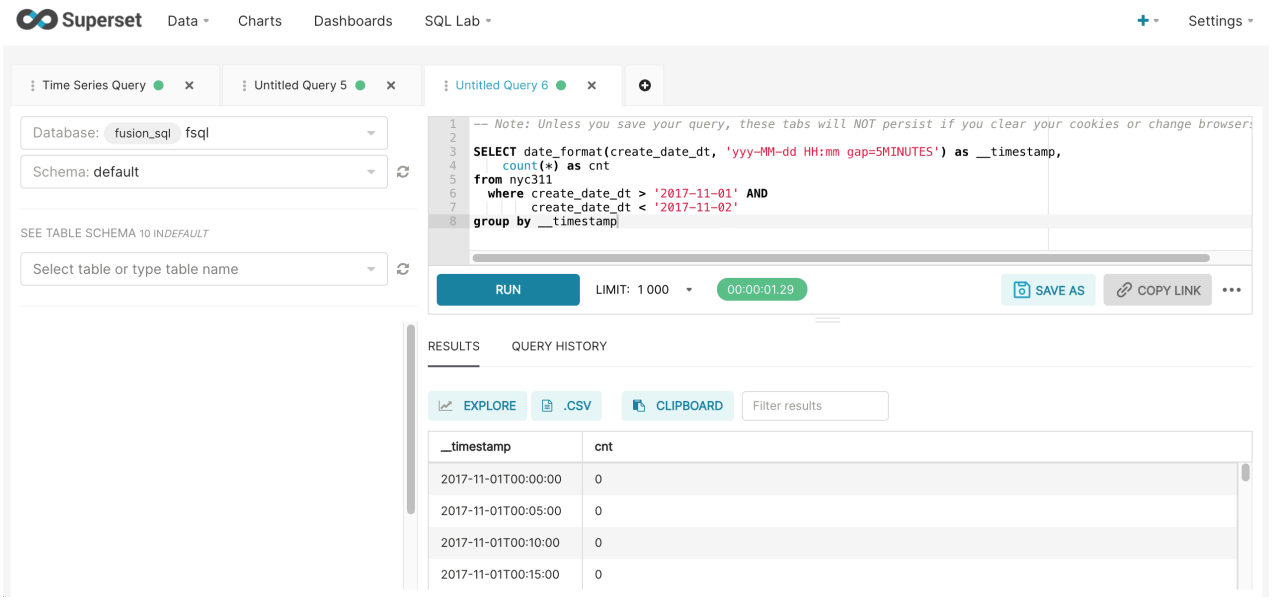
The `gap` parameter can be supplied along with the format to specify the gap interval. The example below specifies a 3 day gap.

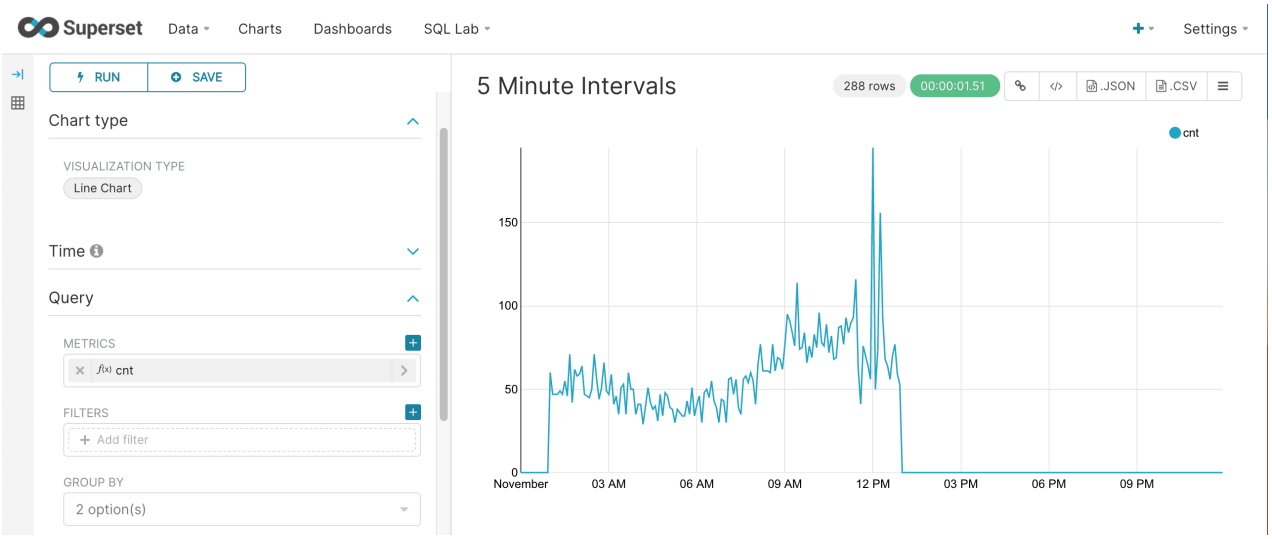




Finer grain queries

Finer time grains can be defined by adding hours, minute or seconds to the time format. The example below shows the format and gap for a 5 minute interval time series.





Fusion SQL statistical functions

Fusion SQL includes a library of statistical and machine learning functions. The section below describes a subset of these functions and how they can be used within Superset.

hist

Fusion SQL supports the `hist` function which pushes the histogram down into server so the full result set doesn't need to be brought over to Superset. This allows for much larger sample sizes to be used for the histogram creation.

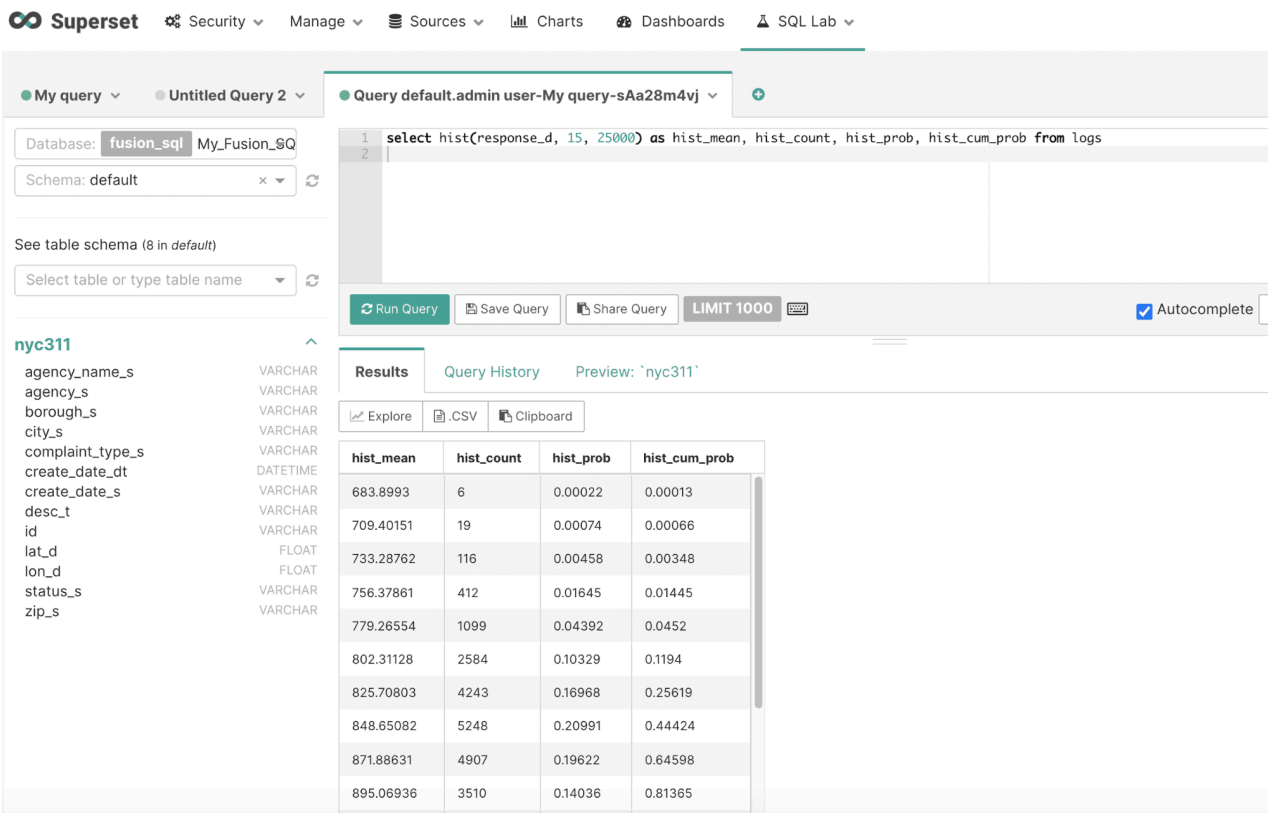
The `hist` function takes three parameters:

- The numeric field to create the histogram from
- The number of bins
- The size of the random sample to create the histogram from

The result will contain a row for each bin in the histogram. The `hist` function returns the mean of each bin in the histogram. There are three other fields that can be selected when the `hist` function is used:

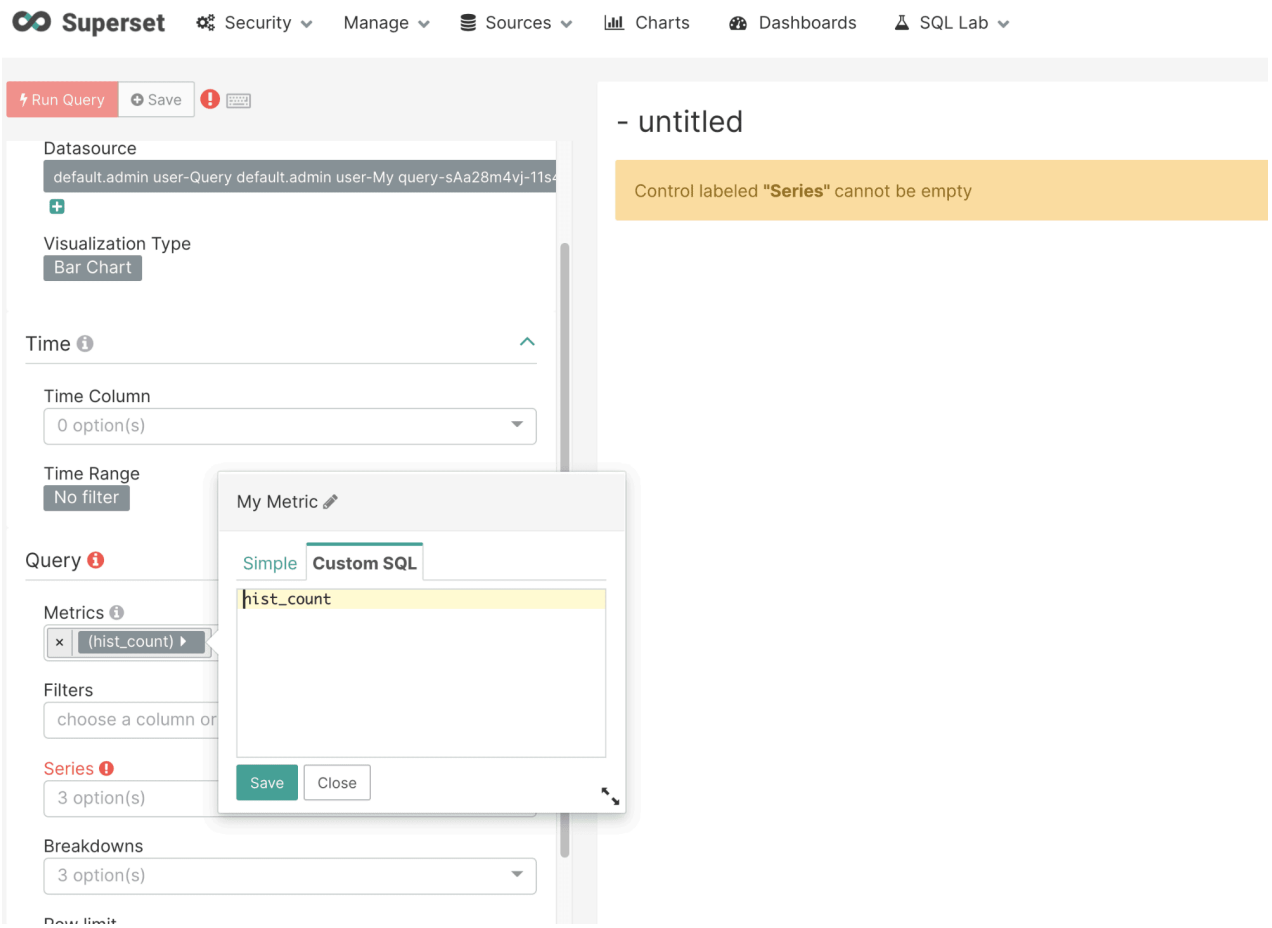
- `hist_count` : The number of samples in the bin
- `hist_prob` : The probability of sample falling within the bin
- `hist_cum_prob` : The cumulative probability of each bin

Below is an example of the `hist` function run in the SQL Lab:

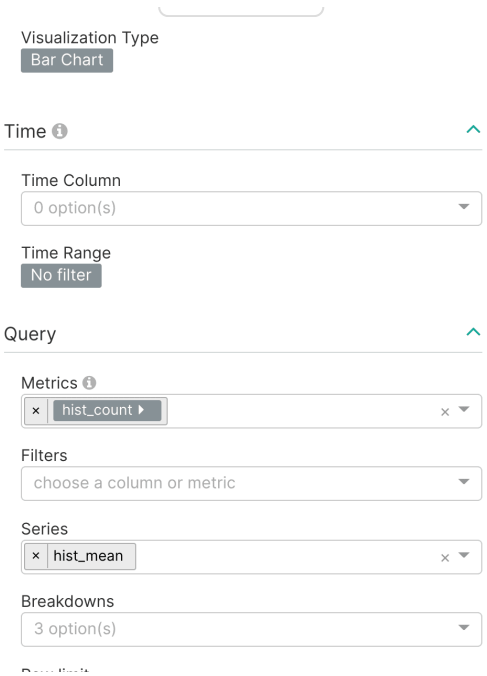


Clicking **Explore** brings up the chart creation screen.

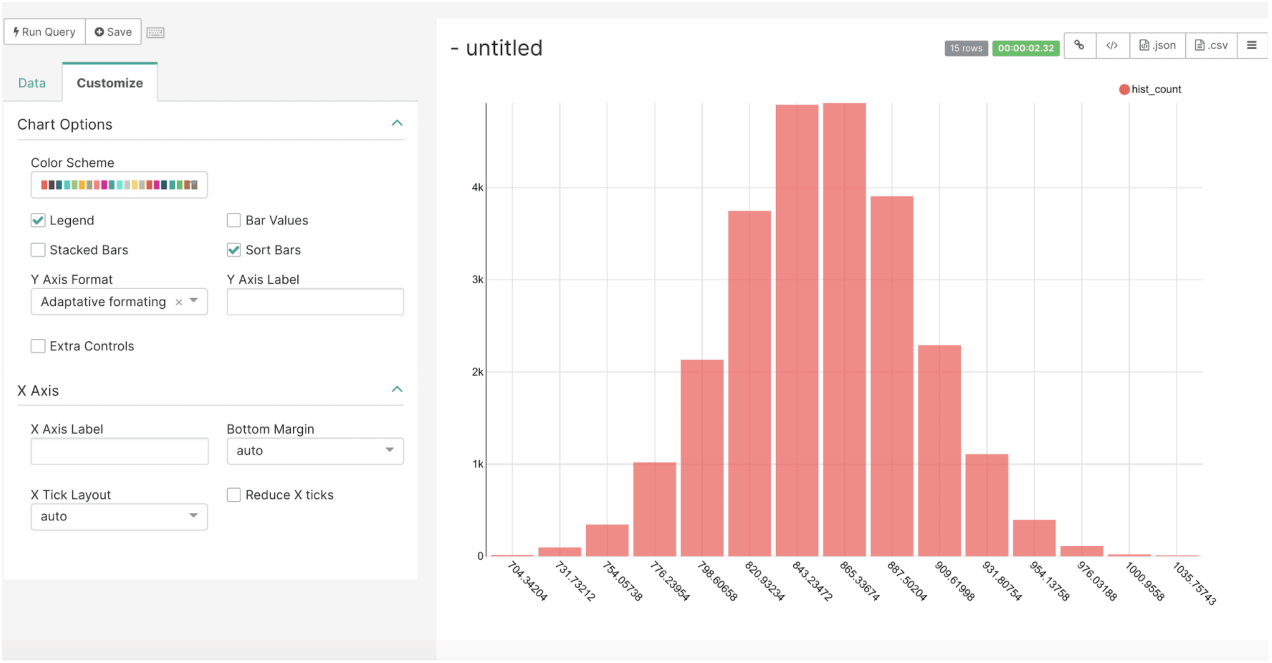
Choose a Bar Chart and select `hist_count` as the metric. This will use the `hist_count` for the y-axis. Be sure to remove the parentheses using the Custom SQL panel shown below:



Next select the `hist_mean` as the Series as shown below:



By default the bar chart will sort the bars in descending order. To maintain the original order of the bar chart go to the **Customize** tab and check the **Sort Bars** box. This will display the histogram bars in the order of the result set as shown below:



geo_cluster

We saw in the section on Mapping that it was very easy to plot lat/lon points on a map with Superset's `deck.gl.Scatterplot` visualization. Fusion SQL can identify the clusters in these scatter plots with the `geo_cluster` function. The `geo_cluster` function performs `geo_spatial` clustering and noise reduction over a set of lat/lon points. Under the covers the `geo_cluster` function uses DBSCAN clustering using the haversine meters distance measure.

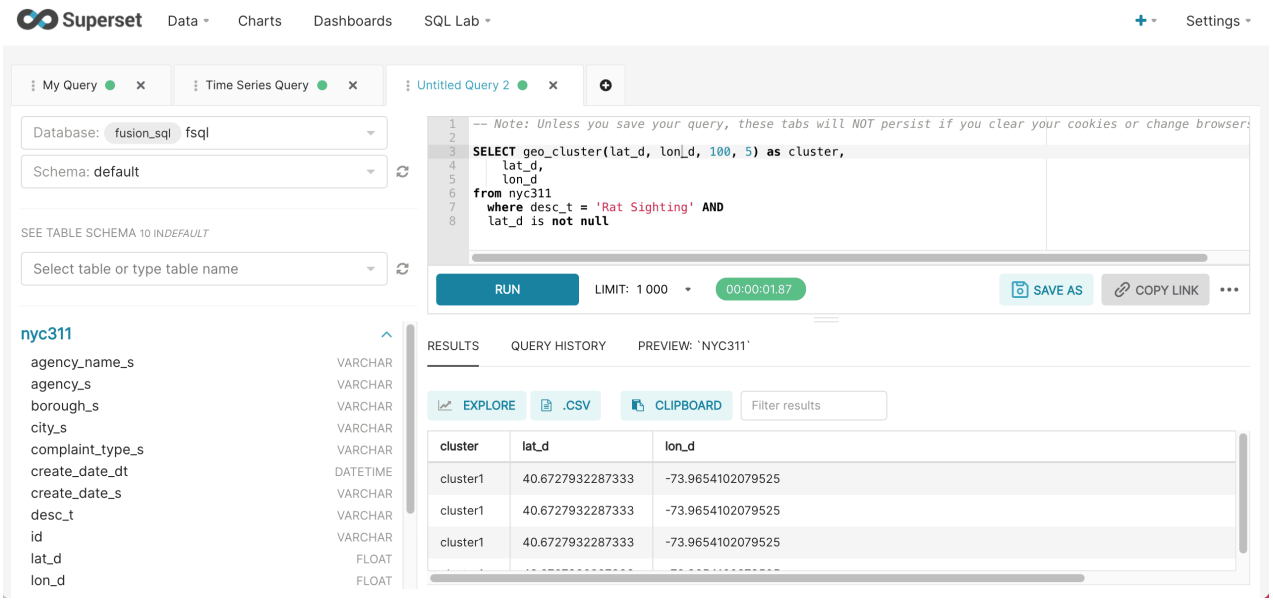
The geo-cluster function allows you to specify the criteria that define a cluster. Any points that don't meet the criteria will be filtered from the result set. This filtering provides powerful noise reduction which can help to visualize the clusters on a map.

The `geo_cluster` function takes four parameters:

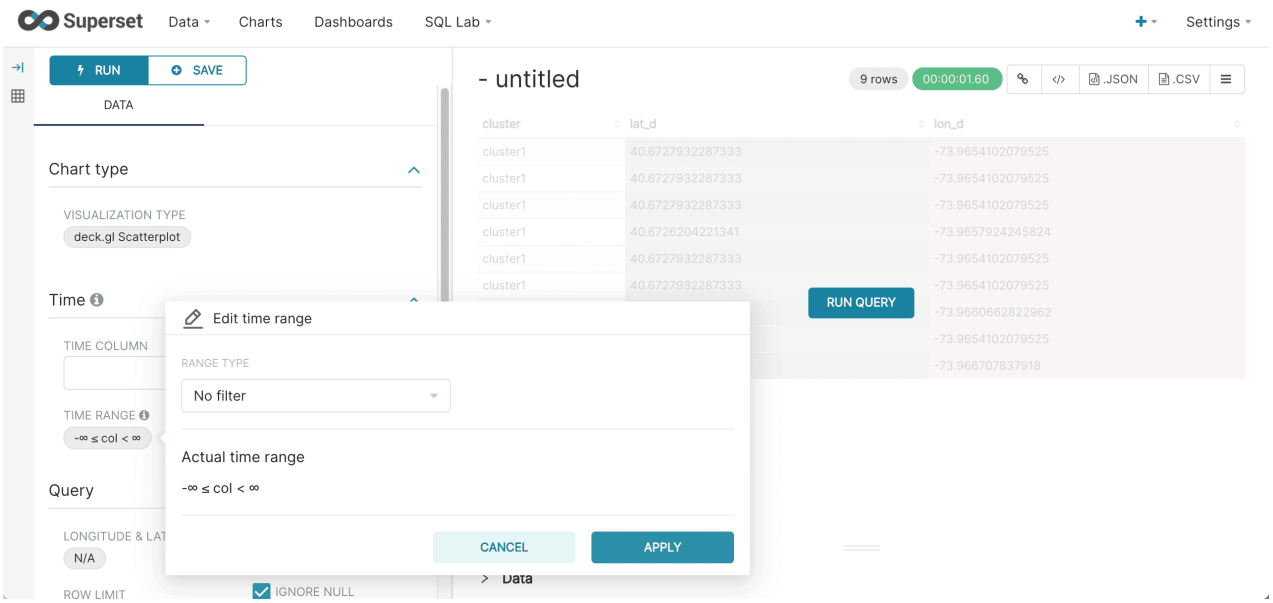
- The latitude column in the index
- The longitude column in the index
- The distance in meters between points for points to be considered a cluster

- The smallest number of points to be considered a cluster

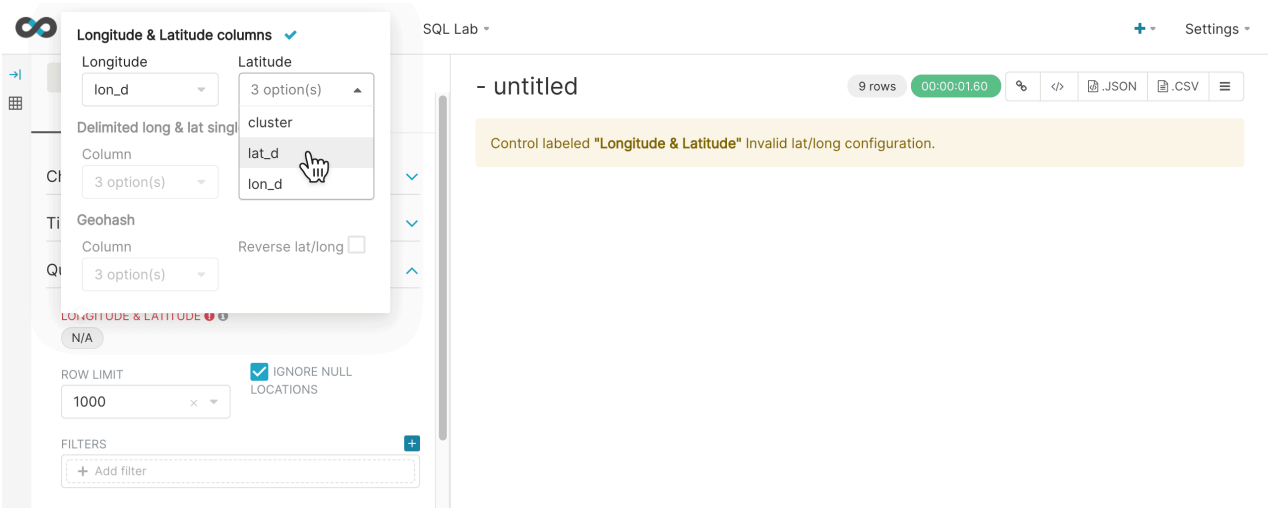
The `geo_cluster` function returns the cluster ID for each point in the result set. The latitude and longitude fields can also be selected with each point. Below is an example of the `geo_cluster` function run inside of the SQL Lab:



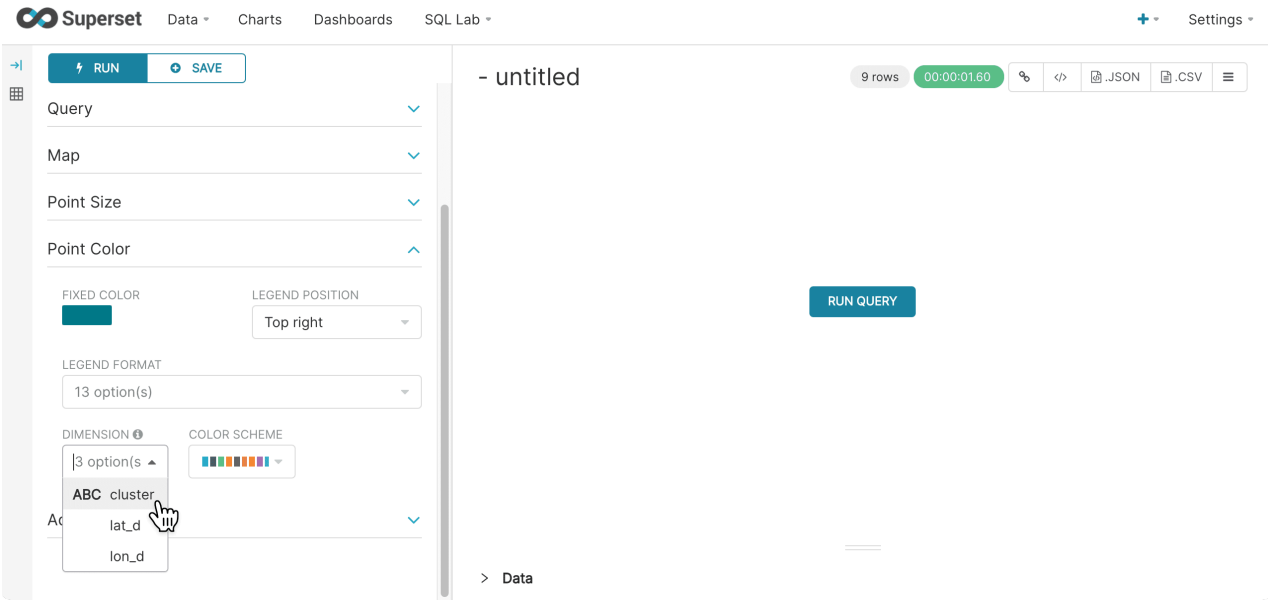
After the results are displayed, click **Explore** to start visualizing the results. The first steps in visualizing is to select the `deck.gl.Scatterplot` visualization and turn off the **Time Range** filter as shown below:



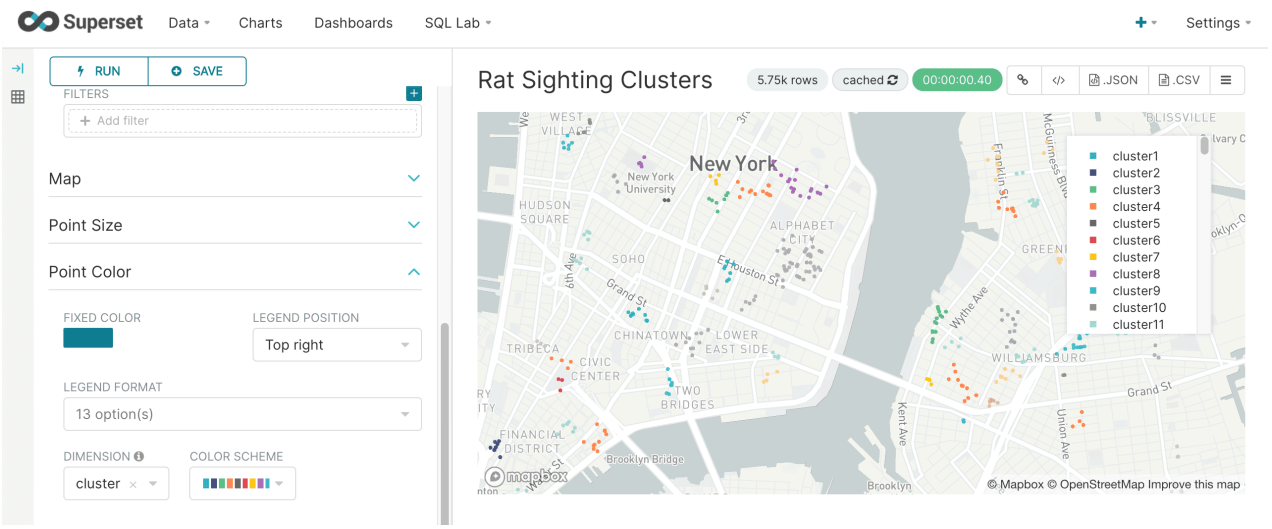
Next you can configure the longitude and latitude columns and choose the row limit. The row limit will control the number of points that are selected for clustering.



Then in the **Point Color** section select the `cluster` column under the **Categorical Color** icon. This will color code the points based on cluster ID.



When you run the query the map will render, displaying only the points that belong to a cluster. The example below shows the color coded clusters on the map:



significant_terms

The `significant_terms` function finds anomaly terms in a text or string field that appear more frequently in a search result set than the entire index.

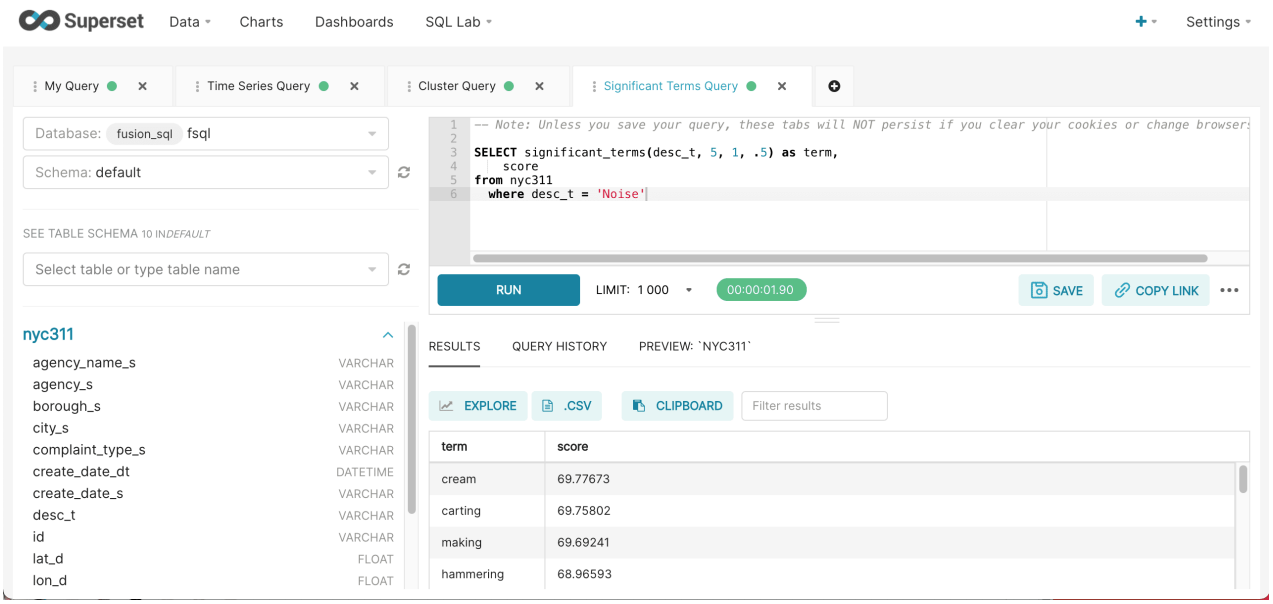
The `significant_terms` function takes four parameters:

- the string or text field in which to find the terms.
- the minimum term length to be considered a significant term.
- the minimum document frequency for a term to be considered a significant term. If greater than one this value is treated as an absolute number of documents. If the value is a float between 0 and 1 its considered to be a percentage of total documents.
- the maximum document frequency for a term to be considered a significant term. If greater than one this value is treated as an absolute number of documents. If the value is a float between 0 and 1 its considered to be a percentage of total documents.

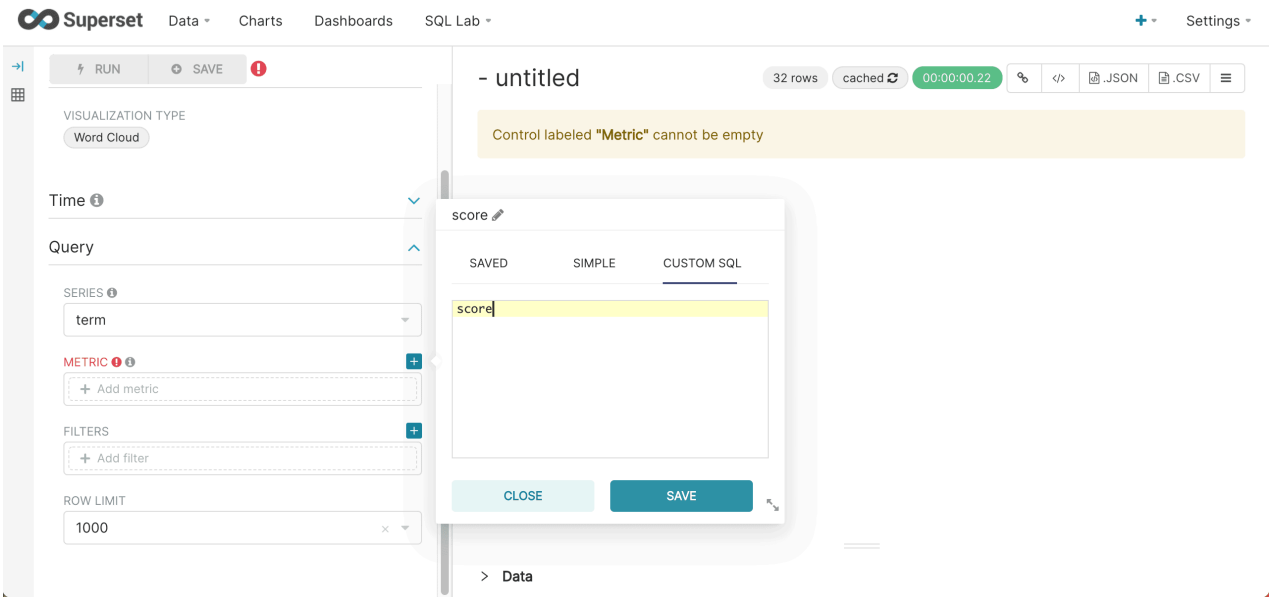
The result set for the `significant_terms` function contains one row for each significant term. The `significant_terms` function returns the value of the term. There are three additional fields available when the significant_terms function is used:

- The foreground field returns the number of documents that contain the term within the result set.
- The background field returns the number of documents that contain the term in the entire index.
- The score field returns the score for the field which is calculated based on the background and foreground counts. Terms are returned in score descending order.

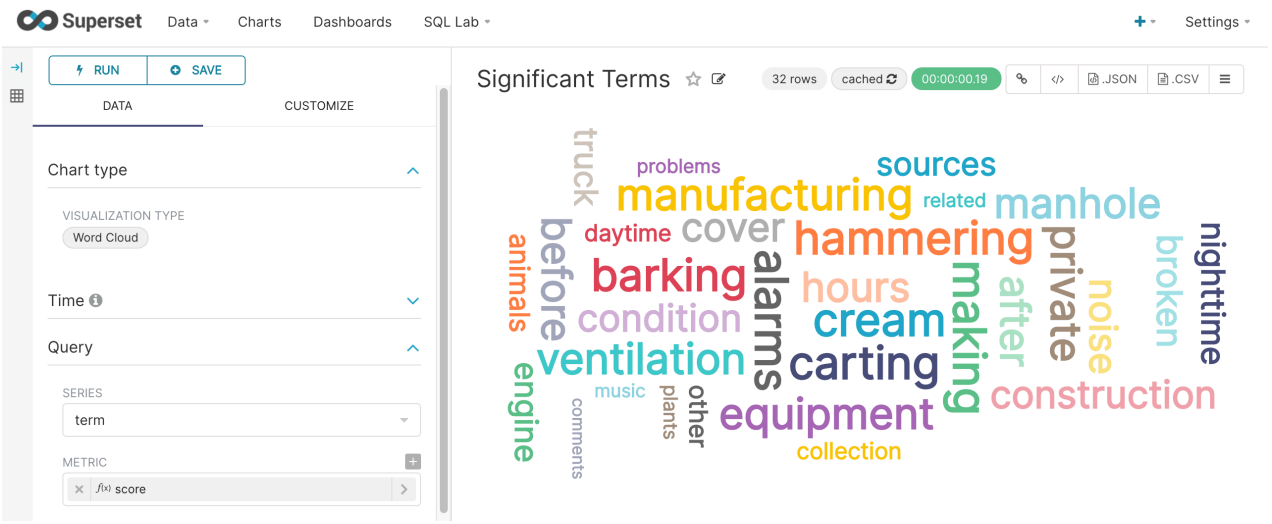
Below is an example of the `significant_terms` query run inside of SQL Lab.



After the results have been displayed they can be visualized by clicking **Explore** and selecting the **Word Cloud** visualization. The word cloud visualization can be configured by selecting `term` in the **Series** drop down box and `score` as the **Metric**. Be sure to choose **Custom SQL** and remove the parentheses from the score so that an aggregation is not applied to the score. This configuration step is shown below:



Once the Word Cloud is configured, run the query to see the word cloud rendered like the screenshot below:



analyze

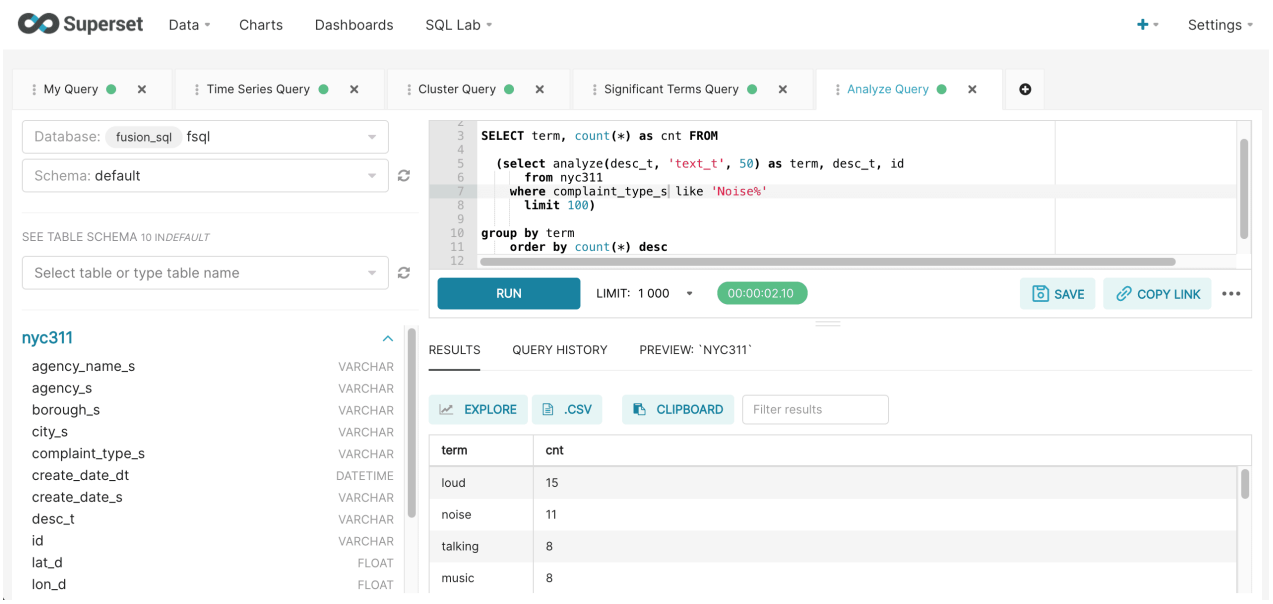
The `analyze` function analyzes a text or string field with a Lucene/Solr analyzer and returns the analyzed tokens. The use of Lucene/Solr analyzers at query time allows for real-time entity extraction by configuring OpenNLP, Stanford NLP or other NLP analyzers.

The `analyze` function takes three parameters:

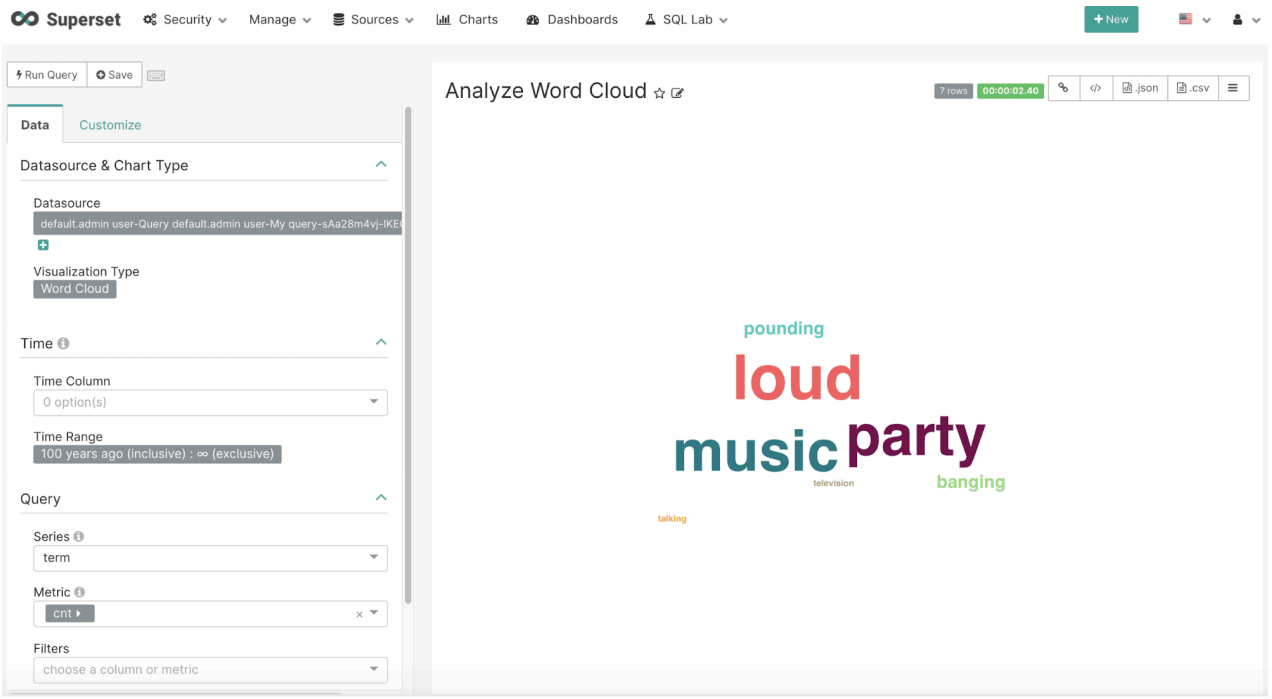
- the text or string field to analyze
- a string literal pointer to the Lucene/Solr analyzer to use. This can be either a field name in the Solr schema with an analyzer defined, or a string that matches a dynamic field pattern.
- the number of documents to analyze

Which documents are analyzed is determined by the WHERE clause. The documents are ordered by score descending to determine the top N documents to analyze.

A SQL aggregation query can be wrapped around the analyzed result set to perform aggregation over the terms. Below is an example of the wrapped aggregation query run inside of SQL Lab:



After the results have been displayed they can be visualized by clicking on the Explore icon and choosing the Word Cloud visualization. The word cloud visualization can be configured by selecting `term` in the **Series** drop down box and `cnt` as the **Metric**. Be sure to choose **Custom SQL** and remove the parentheses from the `cnt` so that an aggregation is not applied to the score. This configuration step is shown below:

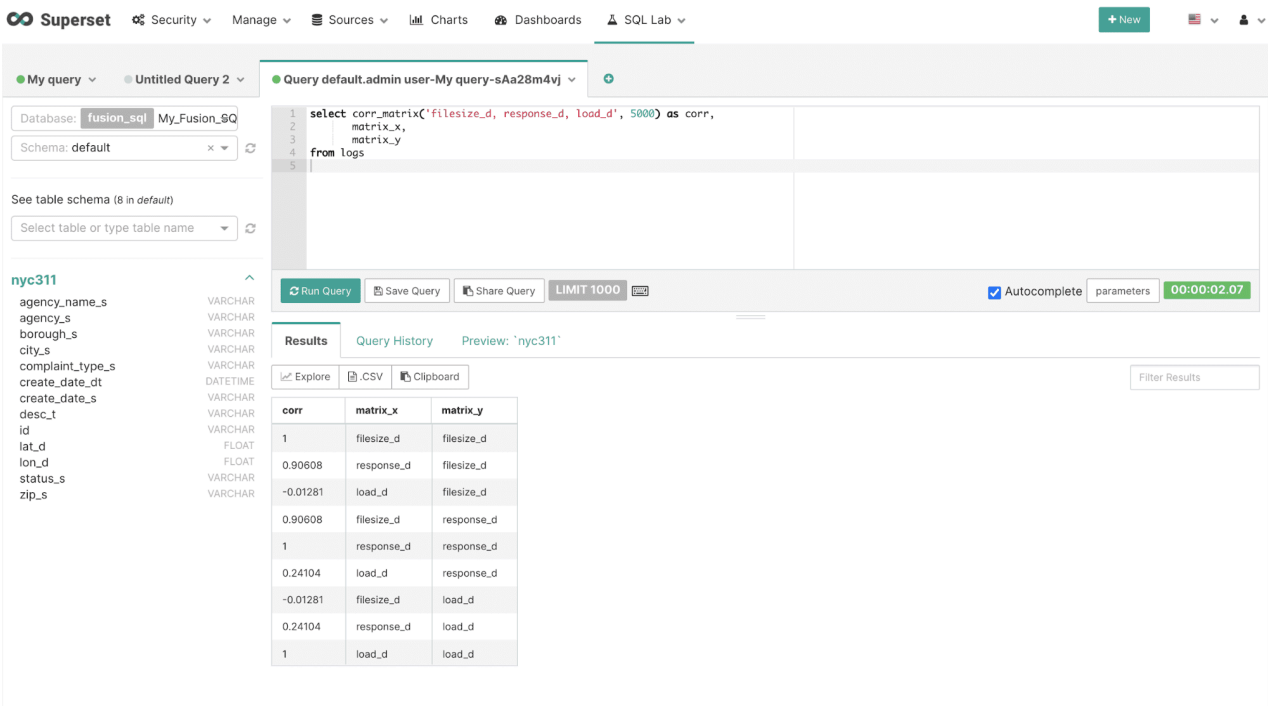


corr_matrix

Correlation matrices can be computed using the `corr_matrix` function. The `corr_matrix` function takes two parameters:

- A string, enclosed in single quotes, containing a comma-separated list of numeric fields for which to calculate the matrix.
- The sample size to compute the correlation matrix from.

The result set for the corr_matrix function contains one row for each two field combination listed in the first parameter. The `corr_matrix` function returns the correlation for the two field combination. There are two additional fields, `matrix_x` and `matrix_y` that contain the values for the field combination for the row.



After the results have been displayed they can be visualized by clicking on the **Explore** icon.

Once on the chart creation screen select the **Heatmap** visualization. The heatmap can be easily configured by selecting the `matrix_x` field in the **X** dropdown and the `matrix_y` field in the **Y** dropdown. In the **Metric** dropdown select the `corr` field which holds the correlation value for each X/Y combination. Be sure to use the **Custom SQL** tab and remove the parentheses from the `corr` field name so an aggregation is not applied to the `corr` value.

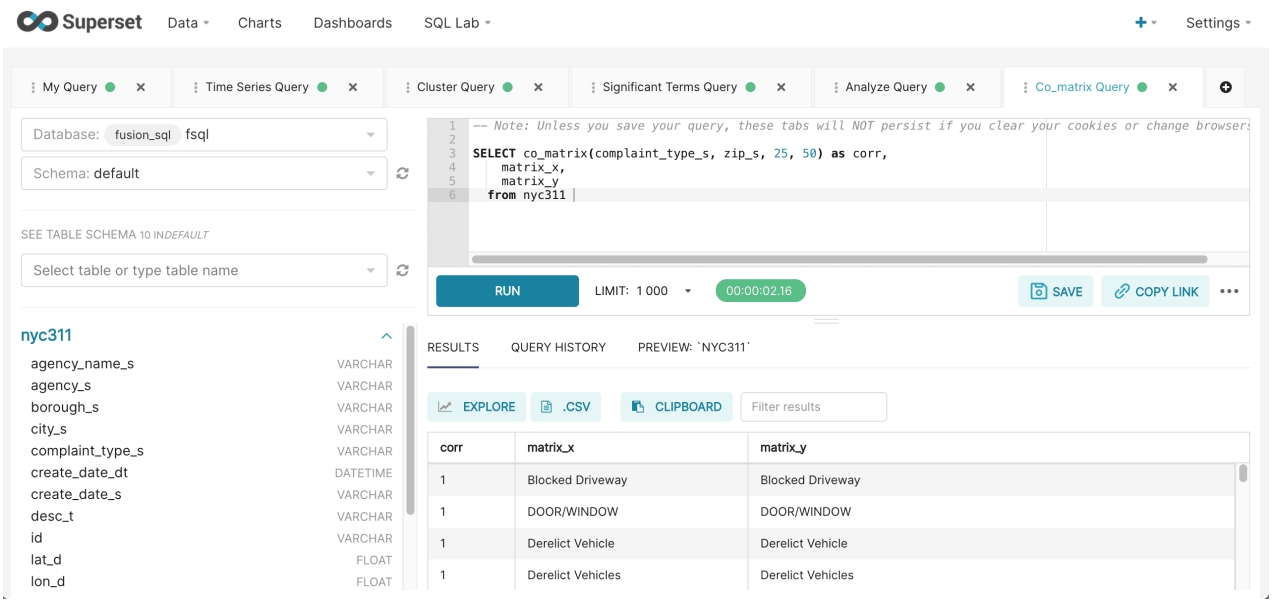
The example below shows the configuration and heatmap visualization for the `corr_matrix` function:

co_matrix

The `co_matrix` function returns a matrix that shows the correlation of values within a categorical field based on their co-occurrence across a different categorical field. For example in a medical database this could be used to correlate diseases by co-occurring symptoms. In the example below the `co_matrix` function is used to correlate complaint types across zip codes in the NYC 311 complaint database. This can be used to better understand how complaint types are correlated across zip codes. The `co_matrix` function has 4 parameters:

- The categorical String field that will be correlated.
- The categorical String field that will be used for co-occurrence.
- The number of categorical variables to correlate.
- The number of categorical variables to calculate co-occurrence from.

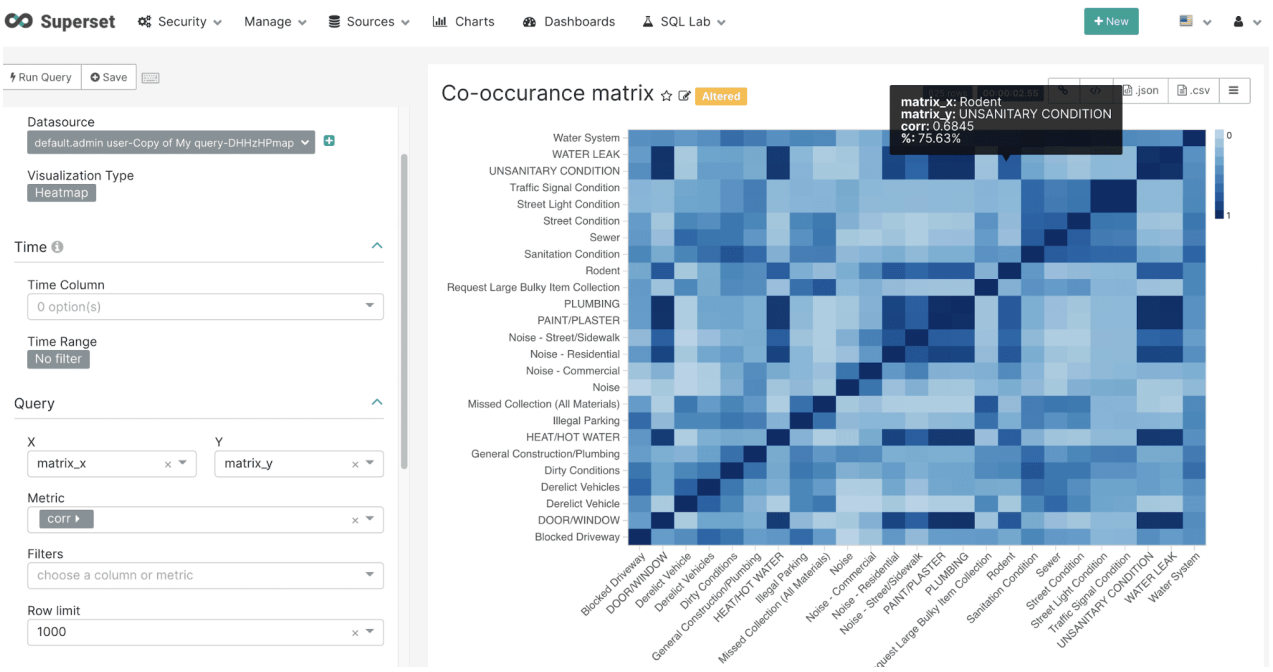
The result set for the `co_matrix` function is a correlation matrix for the first categorical field parameter. The `co_matrix` function returns the correlation for each row. The `matrix_x` and `matrix_y` fields contain the combinations of the top N categorical values.



After the results have been displayed they can be visualized by clicking on the **Explore** icon.

Once on the chart creation screen select the **Heatmap** visualization. The heatmap can be configured by selecting the `matrix_x` field in the X dropdown and the `matrix_y` field in the Y dropdown. In the **Metric** dropdown select `corr` return field which holds the correlation value for each X/Y combination. Be sure to use the Custom SQL tab and remove the parentheses from the `corr` field name so an aggregation is not applied to the `corr` value.

The example below shows the configuration and heatmap visualization for the `co_matrix` function:



Use Synonym Detection

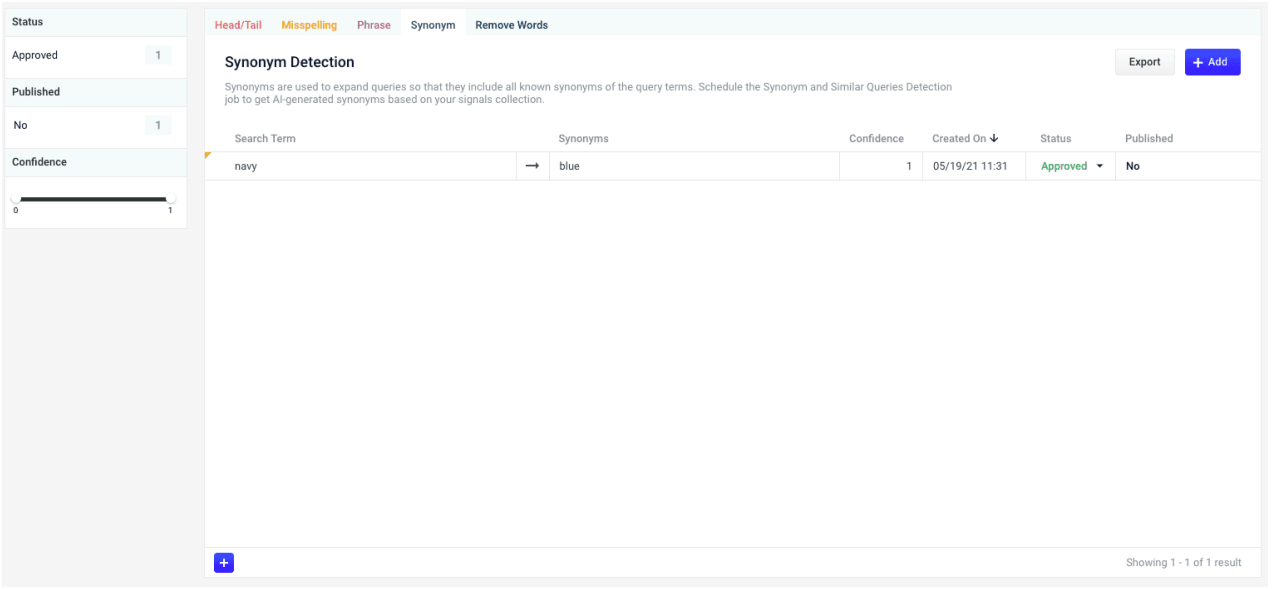
How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion*

Based on the release, synonyms are automatically created based on your AI-generated data. When you navigate to **Relevance** > **Rules** > **Rewrite** and select the **Synonym** tab, the application displays the **Synonym Detection** screen. For more information, see:

Tip

When you manually add new synonym pairs, subsequent job runs use those documents as input for machine learning to improve the job's output. Unlike job-generated documents, manually-added query rewriting documents are never overwritten by new job output.



LucidAcademy

Lucidworks offers free training to help you get started.

The **Course** for **Query Analytics** focuses on how Fusion provides query analytics to detect and improve underperforming queries:

Visit the LucidAcademy to see the full training catalog.

Reviewing auto-generated synonym pairs

Synonyms that are automatically generated by the *Fusion 5.1.x Synonym and Similar Queries Detection job* or *Fusion 5.2.x and later Synonym Detection job* are assigned the following status value:

- **Pending**
  The confidence level is ambiguous, and the result must be reviewed by a user before it can be deployed. It will only be moved from the `_query_rewrite_staging` collection to the `_query_rewrite` collection when its status has changed to "Approved" *and* it has been published.

> **Note**
>
> By default, all results from a synonym job are set to "Pending", since there are usually a limited number of synonyms, and synonym expansion can have high impact on relevancy.

How to review a pending synonym pair result

1. Navigate to **Relevance** > **Rules** > **Rewrite**.
2. Select **Synonym** tab. The application displays the **Synonym Detection** screen.

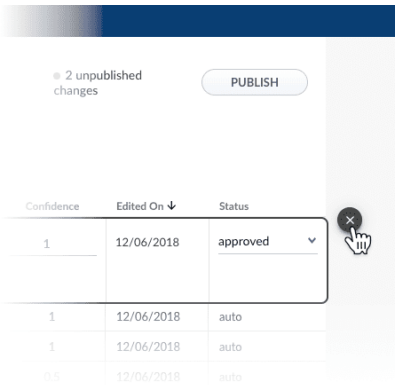
> **Tip**
>
> Notice the **Status** facet on the left. Click **Pending** to view only the items that need review.

3. Click the icon next to the synonym pair.

4. In the **Status** column, select either "Approved" or "Denied".
   Where alternative synonyms were detected, you can click **Suggestions** to view and select them as replacements for the displayed synonym pair.

> **Tip**
>
> Although the Confidence field is also editable, changing its value makes no difference.

5. Click the **Close** icon next to the updated synonym pair:



Note

Approving a synonym pair does not automatically deploy it to the `_query_rewrite` collection. When you have finished your review, you must click **Publish** to deploy your changes.

Adding new synonym pairs

You can manually add synonym pairs in addition to any generated by your Fusion release:

- *Fusion 5.1.x Synonym and Similar Queries Detection job*
- *Fusion 5.2.x and later Synonym Detection job*

How to add a synonym pair

1. Navigate to **Relevance** > **Rules** > **Rewrite**.
2. Select **Synonym** tab. The application displays the **Synonym Detection** screen.
3. At the bottom of the rules list, click the ⊕ icon. A new synonym pair appears at the top of the list.
4. Enter the query term.

   Tip

   It is not necessary to set a confidence value.

5. Select the synonym pair's status, depending on whether you want to deploy it the next time you publish your changes ("Approved") or save it for further review ("Pending").
6. Click the check mark to save the new synonym pair.

Publishing your changes

How to publish updated synonym pairs

1. In the **Synonym Detection** screen, click the **PUBLISH** button.
   Fusion prompts you to confirm that you want to publish your changes.
2. Click **PUBLISH**.

Tip

You can un-publish a query rewrite by changing its status to "denied", then clicking **PUBLISH**.

Query rewrite jobs post-processing cleanup

To perform more extensive cleanup of query rewrites, complete the procedures in *Query Rewrite Jobs Post-processing Cleanup*.

Use Synonym Detection

How-to

> Compatible versions: 4.2
>
> Works with: *Fusion AI*, *Fusion Server*

LucidAcademy

Lucidworks offers free training to help you get started.

The **Course** for **Query Analytics** focuses on how Fusion provides query analytics to detect and improve underperforming queries:

Visit the LucidAcademy to see the full training catalog.

Reviewing auto-generated synonym pairs

Synonyms that are automatically generated by the *Synonym and Similar Queries Detection job* are assigned the following status value:

- **Pending**
  The confidence level is ambiguous, and the result must be reviewed by a user before it can be deployed. It will only be moved from the `_query_rewrite_staging` collection to the `_query_rewrite` collection when its status has changed to "Approved" *and* it has been published.

Note

By default, all results from a synonym job are set to "Pending", since there are usually a limited number of synonyms, and synonym expansion can have high impact on relevancy.

How to review a pending synonym pair result

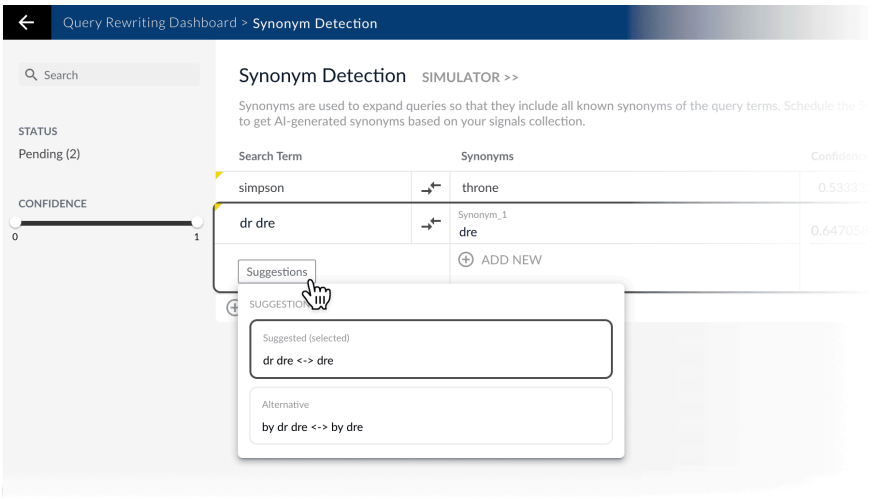
1. Navigate to **Relevance** > **Rules**. The application displays the **Query Rewriting** screen.
2. Select **Synonym Detection**. The application displays the **Synonym Detection** screen.

Tip

Notice the **Status** facet on the left. Click **Pending** to view only the items that need review.

3. Click the icon next to the synonym pair.

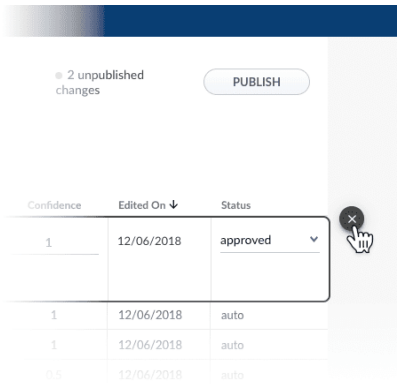
4. In the **Status** column, select either "Approved" or "Denied".
   Optionally, you can also edit the synonym pair itself.
   Where alternative synonyms were detected, you can click **Suggestions** to view and select them as replacements for the displayed synonym pair:



Tip

Although the Confidence field is also editable, changing its value makes no difference.

5. Click the **Close** icon next to the updated synonym pair:



Note

Approving a synonym pair does not automatically deploy it to the `_query_rewrite` collection. When you have finished your review, you must click **Publish** to deploy your changes.

Adding new synonym pairs

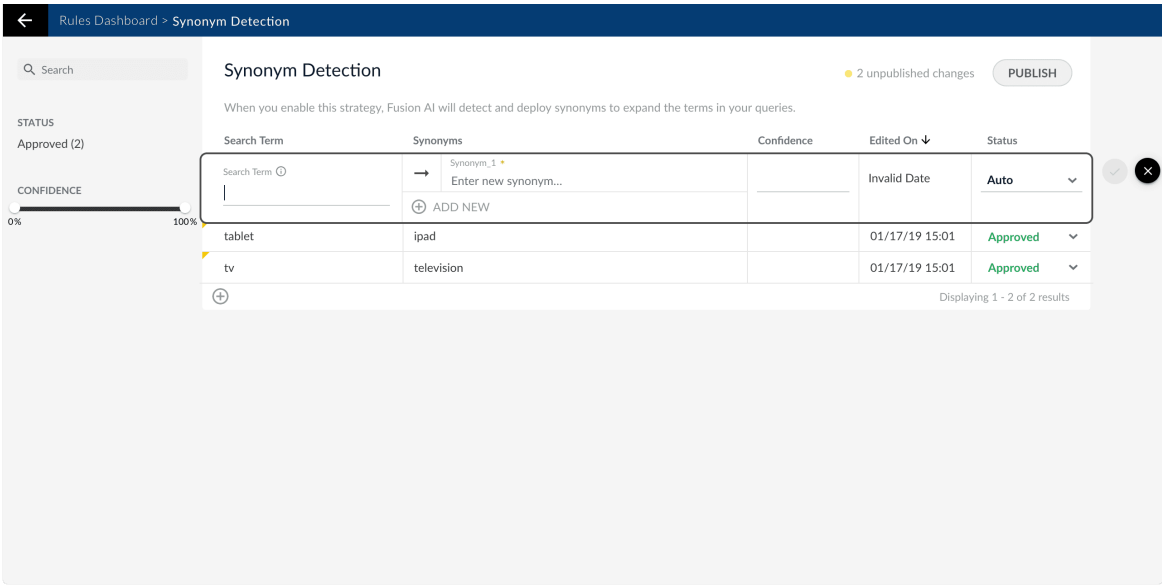
You can manually add synonym pairs in addition to any generated by the *Synonym and Similar Queries Detection job*.

How to add a synonym pair

1. Navigate to **Relevance > Rules**. The application displays the **Query Rewriting** screen.
2. Select **Synonym Detection**. The application displays the **Synonym Detection** screen.

3. At the bottom of the rules list, click the ⊕ icon.

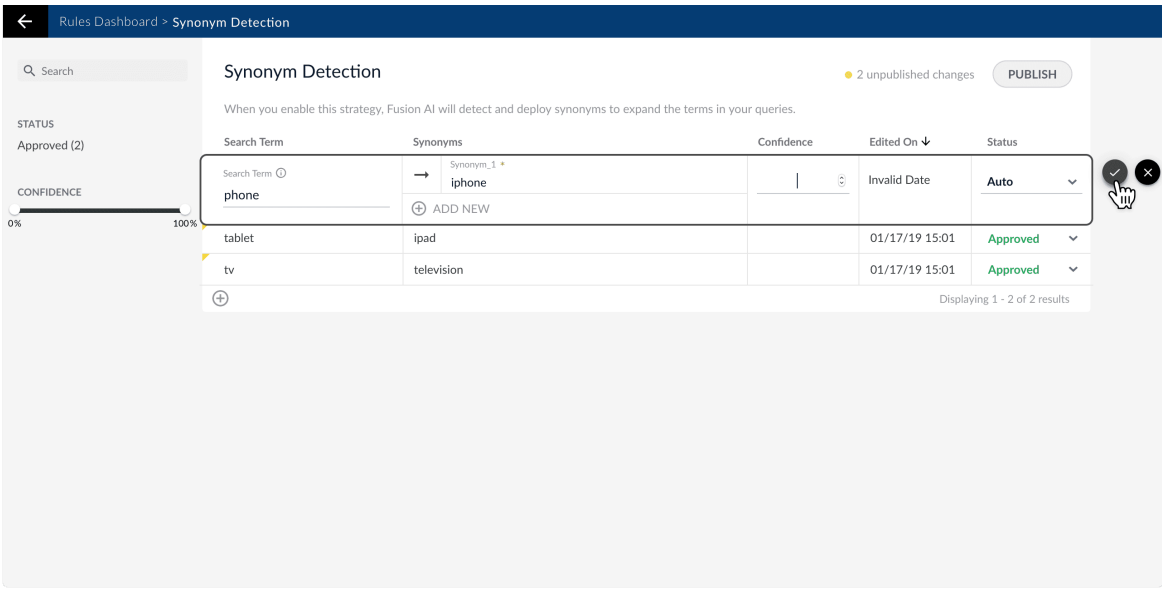
   A new synonym pair appears at the top of the list:



4. Enter the query term.
5. Enter one or more synonym pairs.

> **Tip**
>
> It is not necessary to set a confidence value.

6. Select the synonym pair's status, depending on whether you want to deploy it the next time you publish your changes ("Approved") or save it for further review ("Pending").
7. Click the check mark to save the new synonym pair:



Publishing your changes

How to publish updated synonym pairs

1. In the **Synonym Detection** screen, click the **PUBLISH** button.
   Fusion prompts you to confirm that you want to publish your changes.
2. Click **PUBLISH**.

Tip

You can un-publish a query rewrite by changing its status to "denied", then clicking **PUBLISH**.

Query rewrite jobs post-processing cleanup

To perform more extensive cleanup of query rewrites, complete the procedures in *Query Rewrite Jobs Post-processing Cleanup*.

Use Synonym Detection

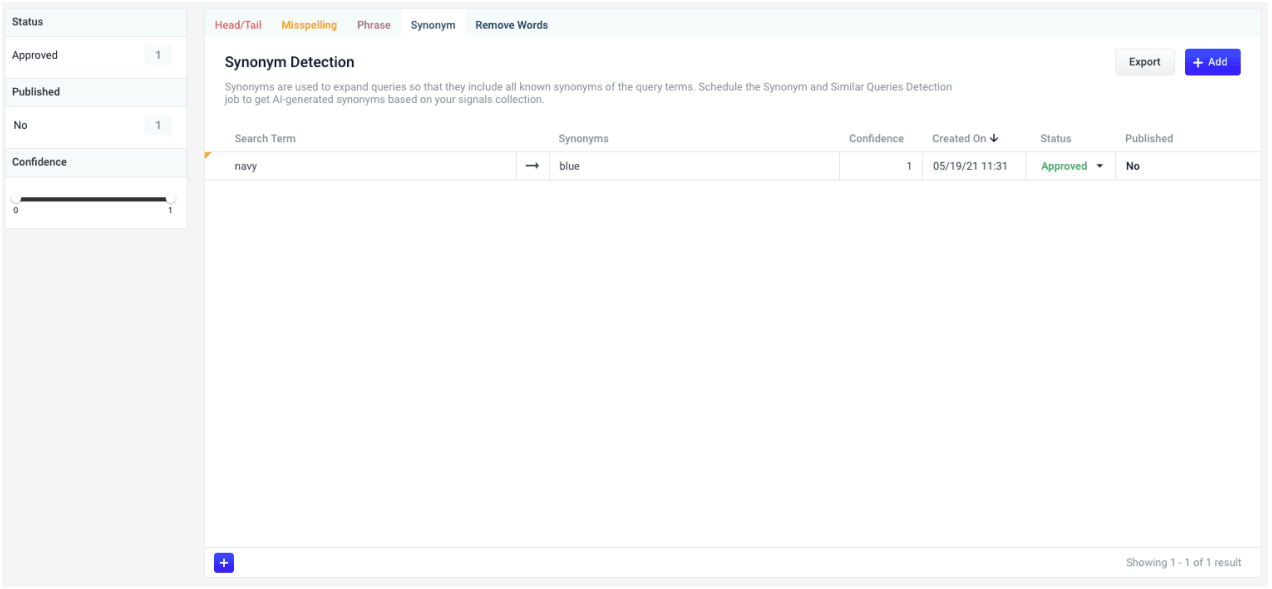
How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

Based on the release, synonyms are automatically created based on your AI-generated data. When you navigate to **Relevance** > **Rules** > **Rewrite** and select the **Synonym** tab, the application displays the **Synonym Detection** screen. For more information, see:

Tip

When you manually add new synonym pairs, subsequent job runs use those documents as input for machine learning to improve the job's output. Unlike job-generated documents, manually-added query rewriting documents are never overwritten by new job output.



LucidAcademy

Lucidworks offers free training to help you get started.

The **Course** for **Query Analytics** focuses on how Fusion provides query analytics to detect and improve underperforming queries:

Visit the LucidAcademy to see the full training catalog.

Reviewing auto-generated synonym pairs

Synonyms that are automatically generated by the synonym jobs are assigned the following status value:

- **Pending**
  The confidence level is ambiguous, and the result must be reviewed by a user before it can be deployed. It will only be moved from the `_query_rewrite_staging` collection to the `_query_rewrite` collection when its status has changed to "Approved" *and* it has been published.

> **Note**
>
> By default, all results from a synonym job are set to "Pending," since there are usually a limited number of synonyms, and synonym expansion can have high impact on relevancy.

How to review a pending synonym pair result

1. Navigate to **Relevance** > **Rules** > **Rewrite**.
2. Select **Synonym** tab. The application displays the **Synonym Detection** screen.

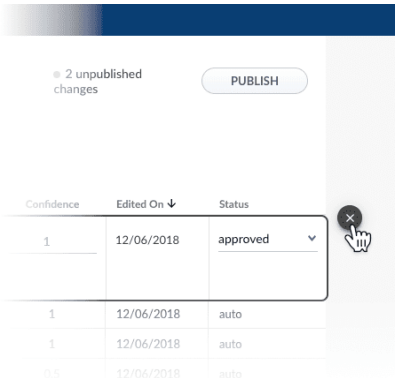
> **Tip**
>
> Notice the **Status** facet on the left. Click **Pending** to view only the items that need review.

3. Click the icon next to the synonym pair.

4. In the **Status** column, select either **Approved** or **Denied**.
   Where alternative synonyms were detected, you can click **Suggestions** to view and select them as replacements for the displayed synonym pair.

> **Tip**
>
> Although the Confidence field is also editable, changing its value makes no difference.

5. Click the **Close** icon next to the updated synonym pair:



Note

Approving a synonym pair does not automatically deploy it to the `_query_rewrite` collection. When you have finished your review, you must click **Publish** to deploy your changes.

Adding new synonym pairs

You can manually add synonym pairs in addition to any generated by Managed Fusion:

- *Synonym Detection job*

How to add a synonym pair

1. Navigate to **Relevance** > **Rules** > **Rewrite**.
2. Select **Synonym** tab. The application displays the **Synonym Detection** screen.
3. At the bottom of the rules list, click the ⊕ icon. A new synonym pair appears at the top of the list.
4. Enter the query term.

   Tip

   It is not necessary to set a confidence value.
5. Select the synonym pair's status, depending on whether you want to deploy it the next time you publish your changes ("Approved") or save it for further review ("Pending").
6. Click the check mark to save the new synonym pair.

Publishing your changes

How to publish updated synonym pairs

1. In the **Synonym Detection** screen, click the **PUBLISH** button.
   Managed Fusion prompts you to confirm that you want to publish your changes.
2. Click **PUBLISH**.

Tip

You can un-publish a query rewrite by changing its status to `denied` then clicking **PUBLISH**.

Query rewrite jobs post-processing cleanup

To perform more extensive cleanup of query rewrites, complete the procedures in *Query Rewrite Jobs Post-processing Cleanup*.

Use System Metrics

How-to

> Compatible versions: 4.2
>
> Works with: *Fusion Server*

Enable metrics indexing in the Fusion UI

1. Navigate to **System** > **System** > **Metrics**.
2. Enable **Record System Metrics Over Time**.

Enable metrics indexing using the REST API

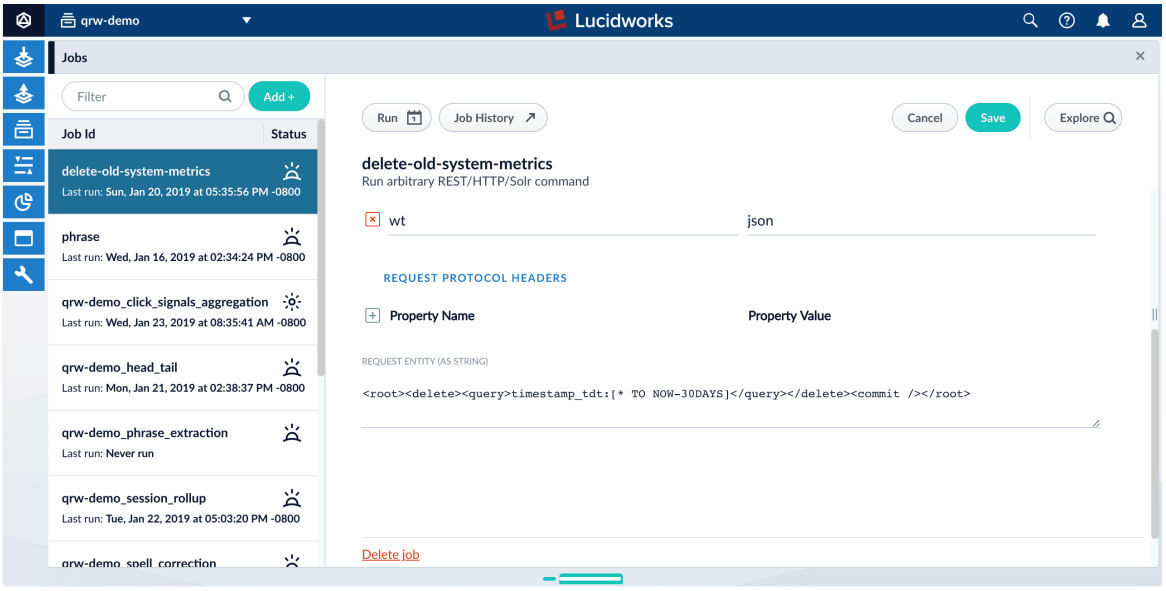
```
curl -u USERNAME:PASSWORD -H 'Content-type:application/json' -X PUT -d
'true'
"http://localhost:8764/api/configurations/com.lucidworks.apollo.metric
s.enabled"
```

There are around 600 different metrics available. In this topic we have highlighted a few that are likely to be the most useful or interesting to you.

The `/system/metrics` endpoint of the *System API* lists all the metrics that the system is currently collecting. Metrics are returned for the current instance only; Fusion instances do not aggregate metrics between nodes.

Adjust the system metrics retention period

1. Navigate to **Collections** > **Jobs**.
2. Select the `delete-old-system-metrics` job.
3. In the job configuration pane, scroll down to **REQUEST ENTITY (AS STRING)**.



4. Change `30DAYS` to the desired period of time to retain system metrics.

Change Metric Collection Frequency

The default frequency to collect metrics is 60 seconds. Since the metrics are stored in a system collection (and a Solr instance), the data can grow to be quite large over time. If you do not need metrics collection to happen as frequently (perhaps during initial implementation), you can change the frequency by modifying the `com.lucidworks.apollo.metrics.poll.seconds` configuration parameter with the *Configurations API*.

For example:

```
curl -u USERNAME:PASSWORD -X PUT -H 'Content-type: application/json' -d '600' http://localhost:8764/api/configurations/com.lucidworks.apollo.metrics.poll.seconds
```

To disable metrics, you could set the **com.lucidworks.apollo.metrics.poll.seconds** parameter to '-1'.

```
curl -u USERNAME:PASSWORD -X PUT -H 'Content-type: application/json' -d '-1' http://localhost:8764/api/configurations/com.lucidworks.apollo.metrics.poll.seconds
```

Use the Active Directory ACL Collection (V1)

How-to

> Works with: *Fusion Connectors*

The Active Directory Connector for ACLs indexes Access Control List (ACL) information into a configured "sidecar" Solr collection, so that it can be used by other connectors.

Note

This article describes how to use an ACL collection generated by the *Active Directory Connector for ACLs* in *V1 platform version*.

During the security trimming stage, the ACL collection is queried using a Solr graph query, which creates a graph of the results using nodes from one document to another.

To use the ACL collection, provide the inputs described below when configuring the *Active Directory Connector for ACLs*:

Input	Description	Example
Start Links	Each LDAP or LDAPS URL that is crawled.	`ldaps://na.lucidworks.com`
LDAP User Principal	The user principal account that crawls LDAP.	`firstname.lastname@na.lucidworks.com`
LDAP User Password	The password of the user that crawls LDAP.	`Password123`
LDAP Search Base	The base DN that performs the crawl.	`DC=na,DC=lucidworks,DC=com`
LDAP User Base (optional)	A list of users that are permitted to access a specific LDAP base.	`OU=Users,DC=na,DC=lucidworks,DC=com`
LDAP Group Base (optional)	A list of groups that are permitted to access a specific LDAP base.	`OU=Groups,DC=na,DC=lucidworks,DC=com`
LDAP User Filter (optional)	A custom attribute filter that finds user records in LDAP.	`(&(objectclass=user)(sAMAccountName=*))`
LDAP Group Filter (optional)	A custom attribute filter that finds group records in LDAP.	`(&(objectclass=group))`
SOLR ACL Collection Name	The name of the ACL collection.	`acl`
Index sAMAccountName Users (Active Directory only)	When active, a document is created in the ACL collection representing a user with an ID of `sAMAccountName`. This allows security trimming on the domain\username version of the username.	`true`
Index userPrincipalName Users (Active Directory only)	When active, a document is created in the ACL collection representing a user with an ID of `userPrincipalName`. This allows security trimming on the `username@fqdn.com` version of the username.	`false`

Use the Analytics screen in the Rules editor

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

This article describes how to use the *Analytics screen* in the *Rules editor*.

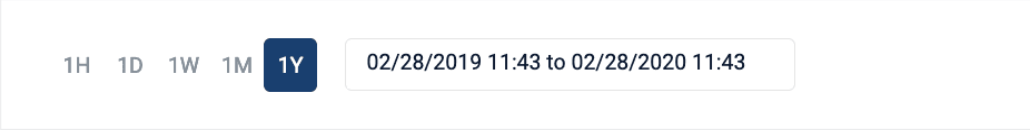
Note

Job configuration affects the data that displays on the dashboard. For example, the `COLLECTION_NAME_session_rollup` *job* aggregates related user activity into a session signal that contains activity count, duration, and keywords (based on user search terms).

UI elements

Data date range

- **Quick Range Selectors.** Click one of the quick range selectors to facet your data.



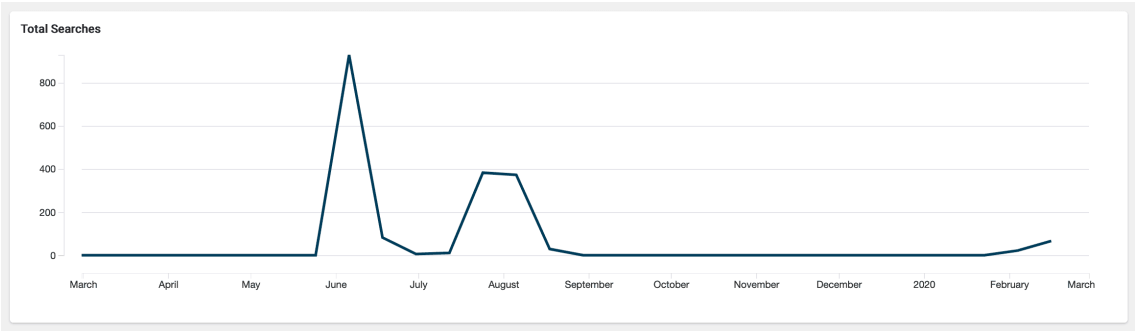
- **Calendar Selector.** Use the calendar selector to specify a start and end date and time facet for your data.

Visitor data

- **Data Cards**

- **Site Visits.** The total number of visits.
- **Total Searches.** The total number of searches performed.
- **Unique Visitors.** The number of unique visitors to your site.
- **Average Session.** The average length of time spent on your site.

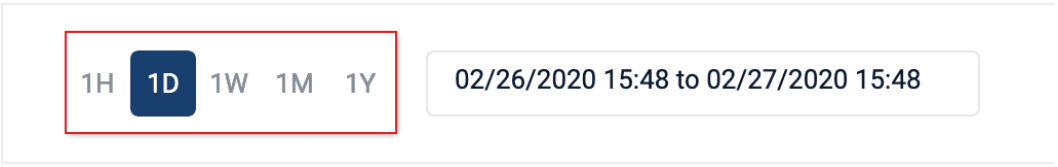
- **Data Graph.** Detailed information for the selected data card.



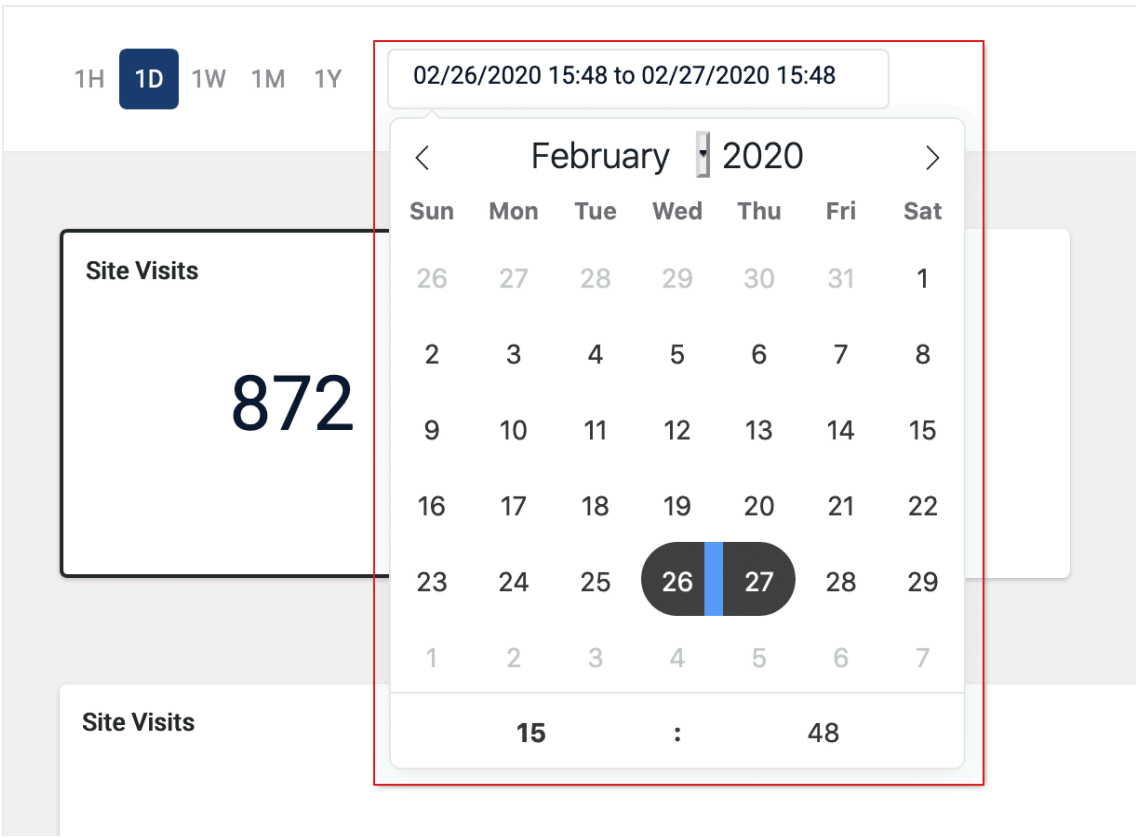
View data by date range

The Rules Editor analytics allows you to view data by a date range. To do this, you can either:

- Click one of the date range presets:



- Click the date range details to display and select your range on the calendar that appears:



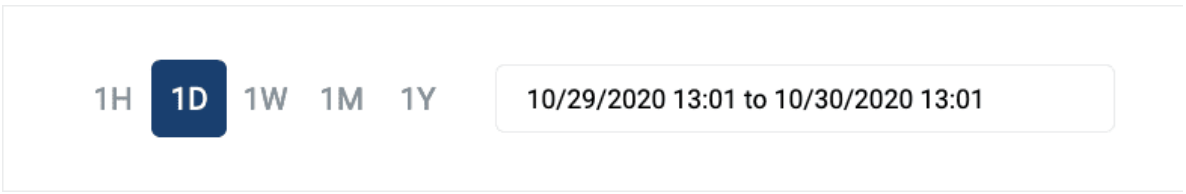
View additional card details

Click one of the analytics cards to view additional details for that card's data.

Configure Analytics cards

1. Sign in to Managed Fusion and click your application.
2. Click **Relevance > Rules > Analytics**.
3. Click the **Edit** pencil icon.
4. In the card you want to edit, click the wrench icon.
5. In the screen that displays, enter your edits.
6. Click **Save**.

If the data is empty, check to make sure the date range includes the data you want to display.



Use the Analytics Screen in the Rules Editor

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion*

This article describes how to use the *Analytics screen* in the *Rules Editor*.

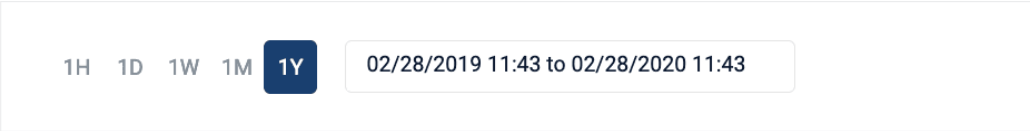
Important

The Analytics screen was renamed from the Dashboard screen in Fusion 5.2.0. All functionality remained the same.

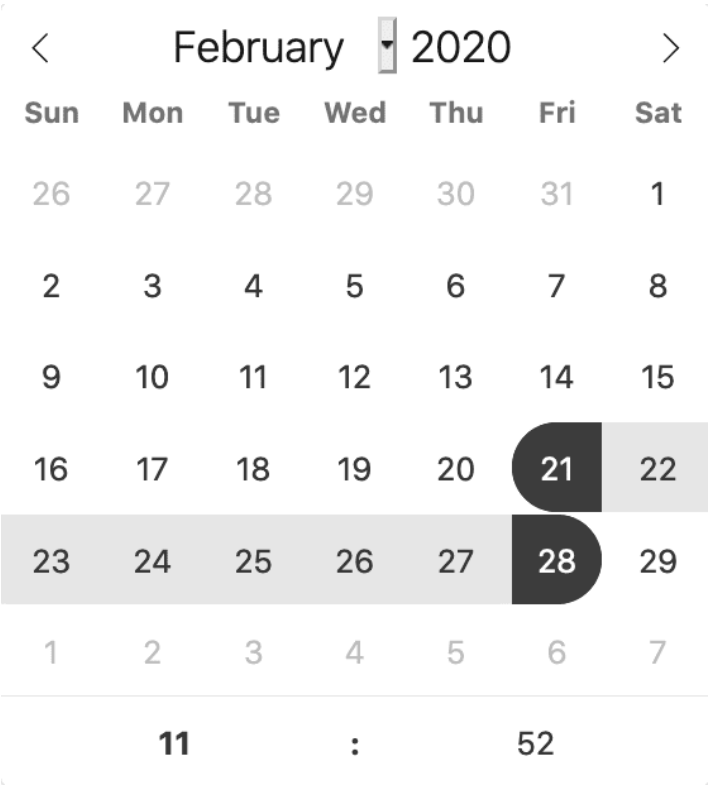
UI elements

Data date range

- **Quick Range Selectors.** Click one of the quick range selectors to facet your data.



- **Calendar Selector.** Use the calendar selector to specify a start and end date and time facet for your data.

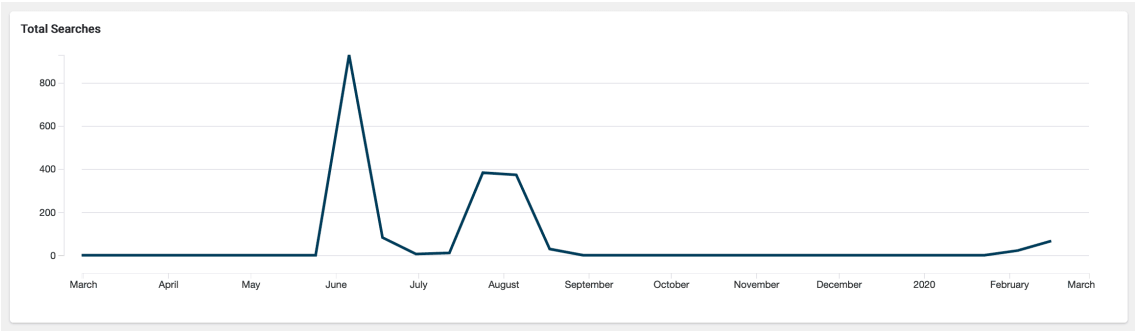


Visitor data

- **Data Cards**

  - **Site Visits.** The total number of visits.
  - **Total Searches.** The total number of searches performed.
  - **Unique Visitors.** The number of unique visitors to your site.
  - **Average Session.** The average length of time spent on your site.

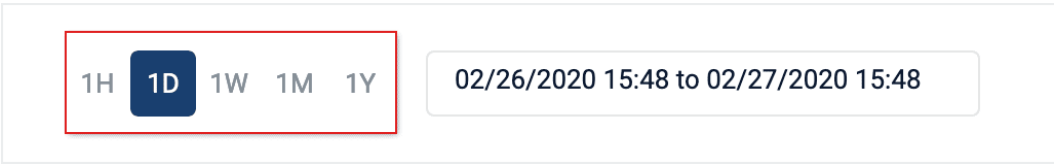
- **Data Graph.** Detailed information for the selected data card.



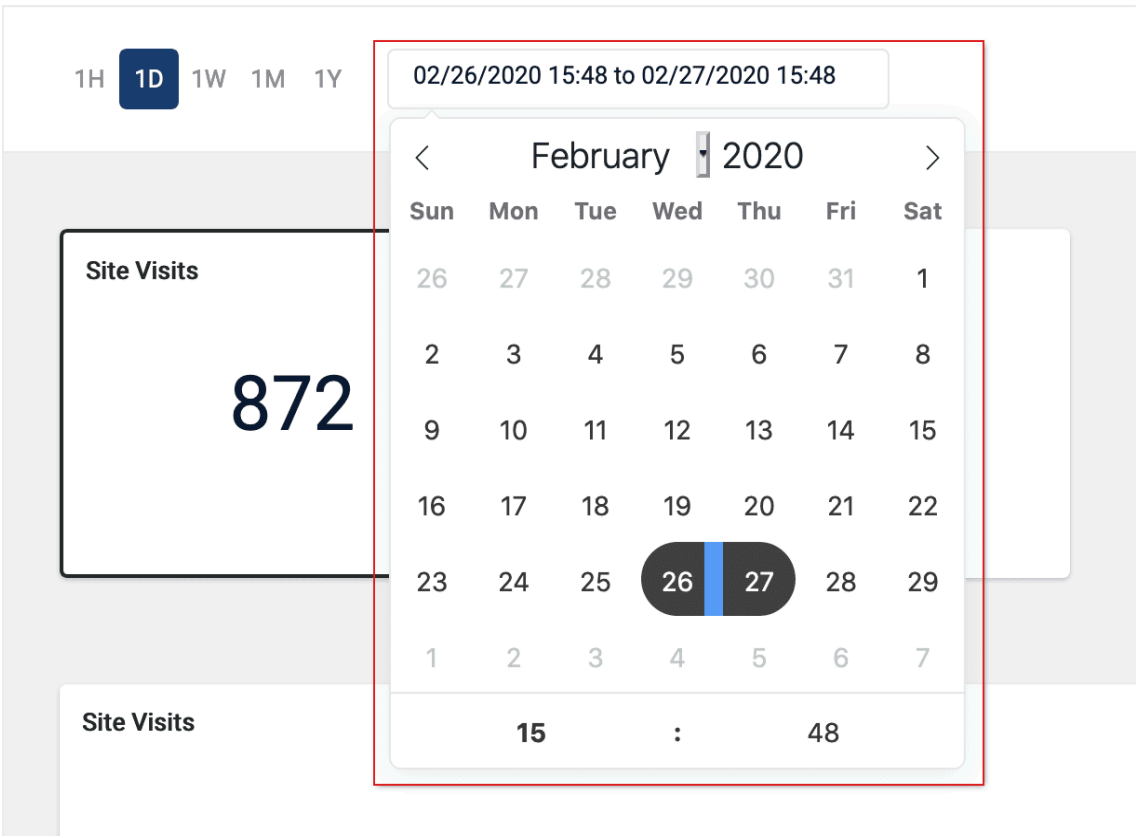
View data by date range

The Rules Editor analytics allows you to view data by a date range. To do this, you can either:

- Click one of the date range presets:



- Click the date range details to display and select your range on the calendar that appears:



View additional card details

Click one of the analytics cards to view additional details for that card's data.

Use the Analytics Screen in the Rules Editor

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion*

This article describes how to use the *Analytics screen* in the *Rules Editor*.

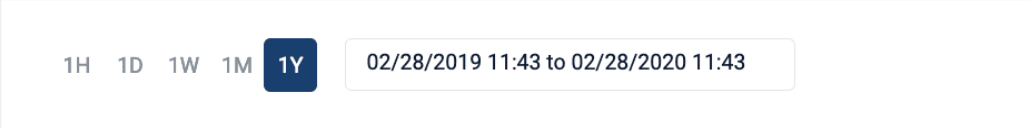
Note

Job configuration affects the data that displays on the dashboard. For example, the `COLLECTION_NAME_session_rollup` *job* aggregates related user activity into a session signal that contains activity count, duration, and keywords (based on user search terms).

UI elements

Data date range

- **Quick Range Selectors.** Click one of the quick range selectors to facet your data.



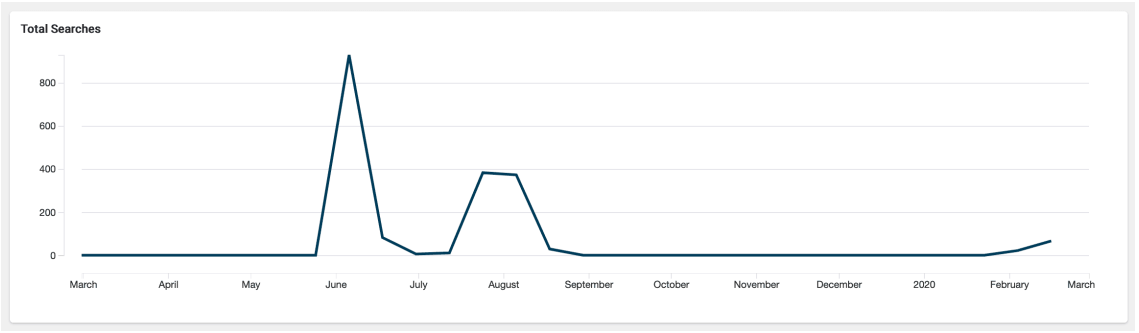
- **Calendar Selector.** Use the calendar selector to specify a start and end date and time facet for your data.

Visitor data

- **Data Cards**

  - **Site Visits.** The total number of visits.
  - **Total Searches.** The total number of searches performed.
  - **Unique Visitors.** The number of unique visitors to your site.
  - **Average Session.** The average length of time spent on your site.

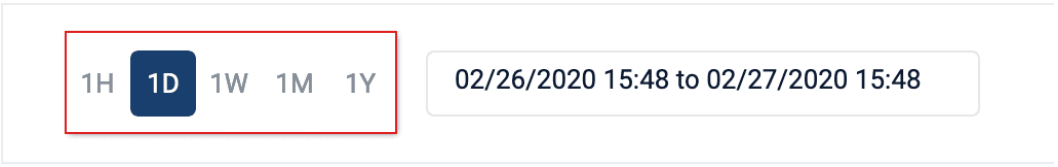
- **Data Graph.** Detailed information for the selected data card.



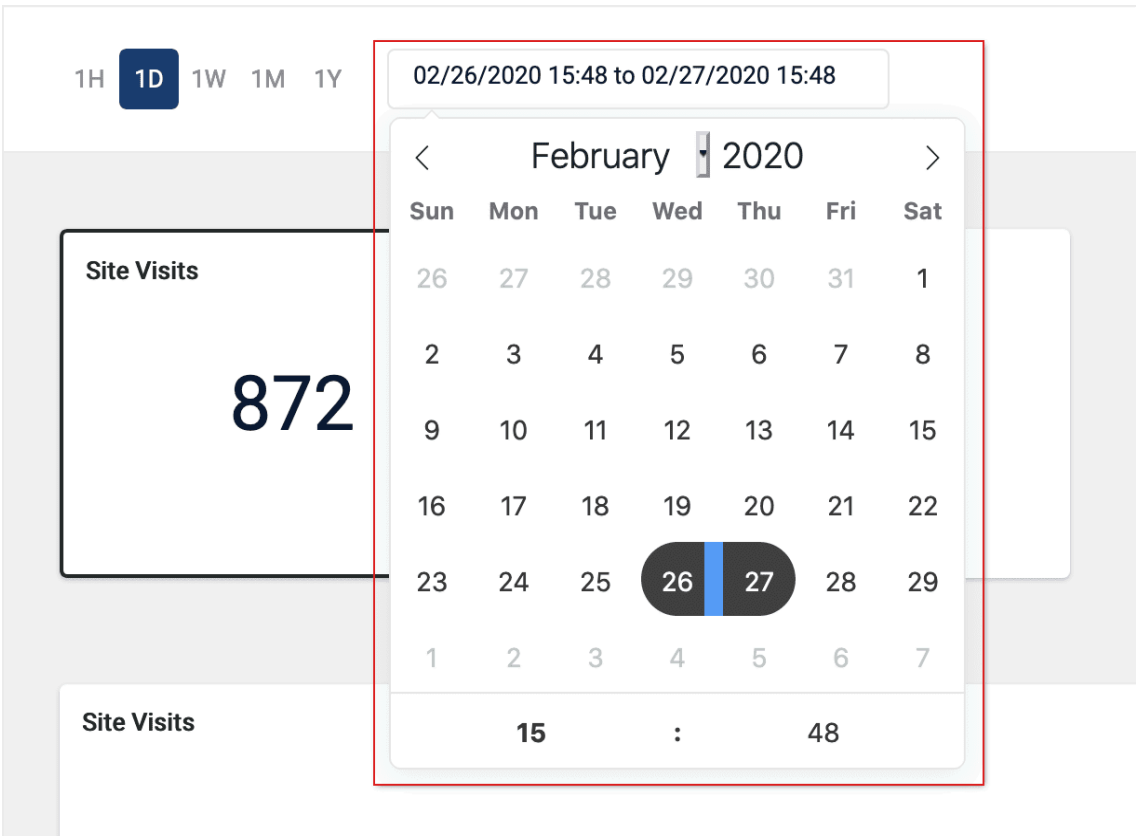
View data by date range

The Rules Editor analytics allows you to view data by a date range. To do this, you can either:

- Click one of the date range presets:



- Click the date range details to display and select your range on the calendar that appears:



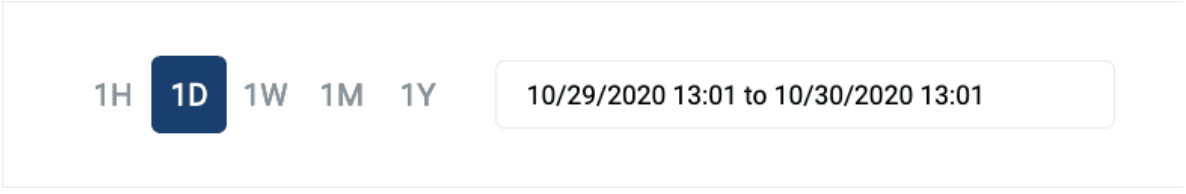
View additional card details

Click one of the analytics cards to view additional details for that card's data.

Configure Analytics cards

1. Hover over one of the cards.
2. Click the **Edit** button that appears:
3. Complete the form that appears.
4. Click the **Save** button.

If the data is empty, check to make sure the date range includes the data you want to display.



Use the Blob Editor

How-to

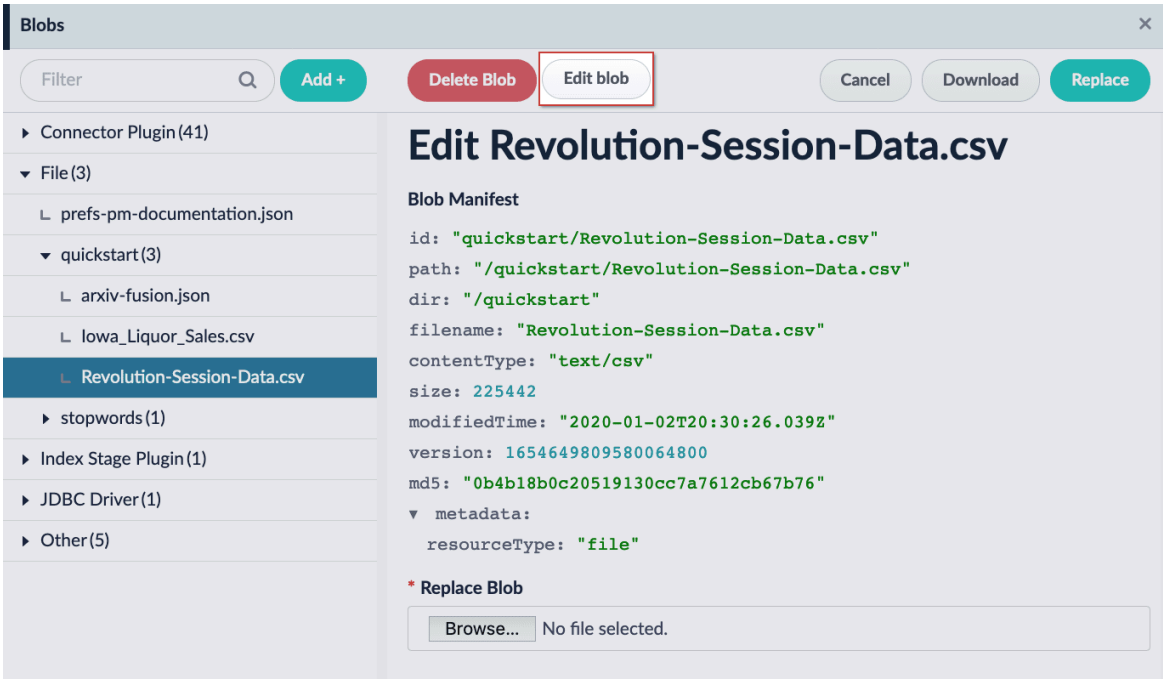
> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion*

The blob editor gives the ability to edit blobs in the Fusion UI in a variety of formats:

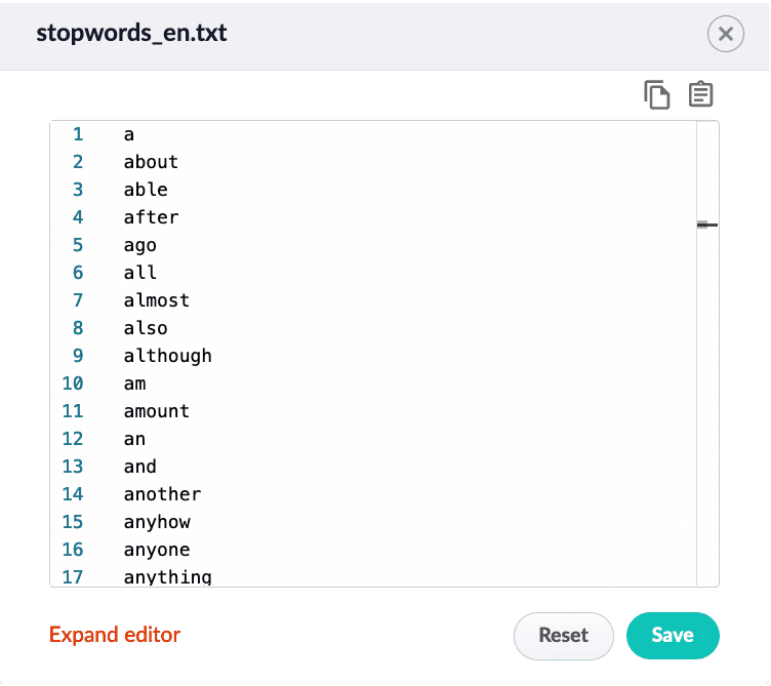
- CSV
- Java
- JavaScript
- Python
- SCALA
- Typescript
- Plain text

This article focuses on editing an existing blob. In this case, the blob is a *stopwords file* that needs to be updated.

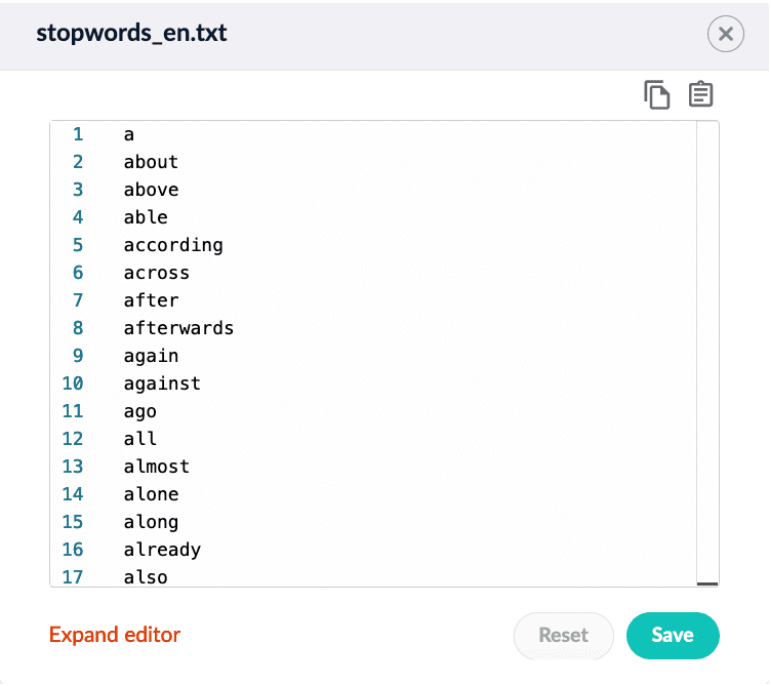
1. Navigate to **System > Blobs** and select a valid blob.
2. Click the **Edit blob** button. This button is only available for valid blob types.



The editor appears:



3. (optional) Click **Expand editor** to view the editor in fullscreen mode.
4. (optional) Click the **Copy** button to copy the blob details. This can be saved somewhere, if desired.
5. Copy the contents of the new stop words file, and click the **Paste** button to replace the contents of the blob. Alternatively, delete the existing contents and manually paste the new contents.



6. Click the **Save** button to save your changes.

Use the Blob Editor

How-to

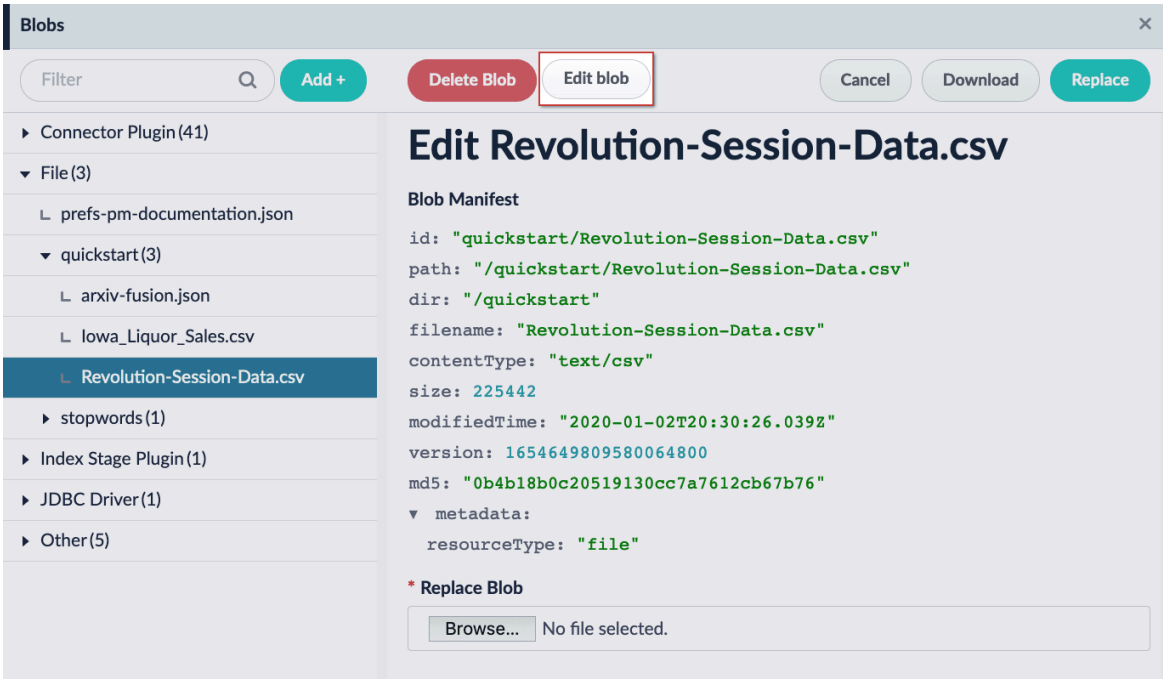
> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

The blob editor gives the ability to edit blobs in the Managed Fusion UI in a variety of formats:

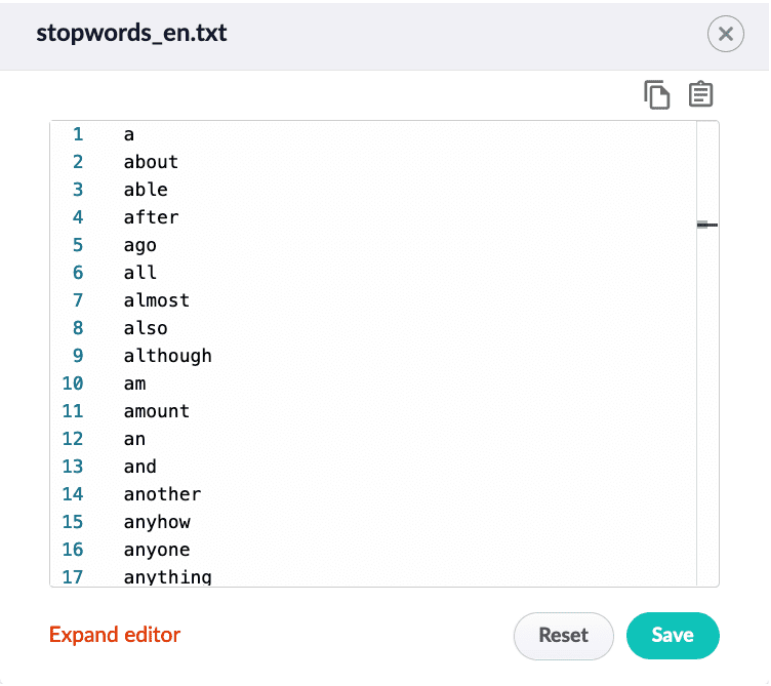
- CSV
- Java
- JavaScript
- Python
- SCALA
- Typescript
- Plain text

This article focuses on editing an existing blob. In this example, the blob is a *stopwords file* that needs to be updated.

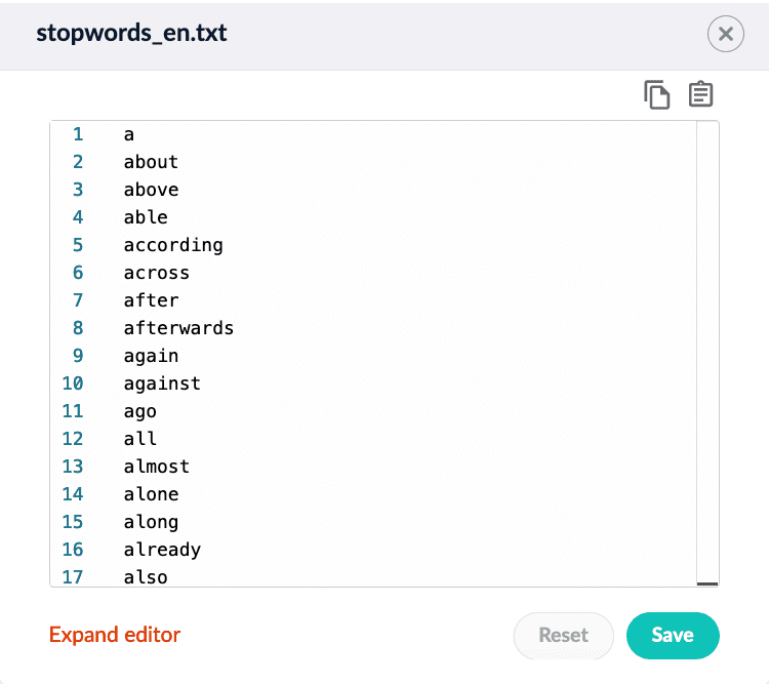
1. Navigate to **System > Blobs** and select a valid blob.
2. Click the **Edit blob** button. This button is only available for valid blob types.



The editor appears:



3. (optional) Click **Expand editor** to view the editor in fullscreen mode.
4. (optional) Click the **Copy** button to copy the blob details. This can be saved somewhere, if desired.
5. Copy the contents of the new stopwords file, and click the **Paste** button to replace the contents of the blob. Alternatively, delete the existing contents and manually paste the new contents.



6. Click the **Save** button to save your changes.

Use the Code Editor

How-to

> Compatible versions: 4.2
>
> Works with: *App Studio*

App Studio's interface includes a built-in code editor that opens in your browser.

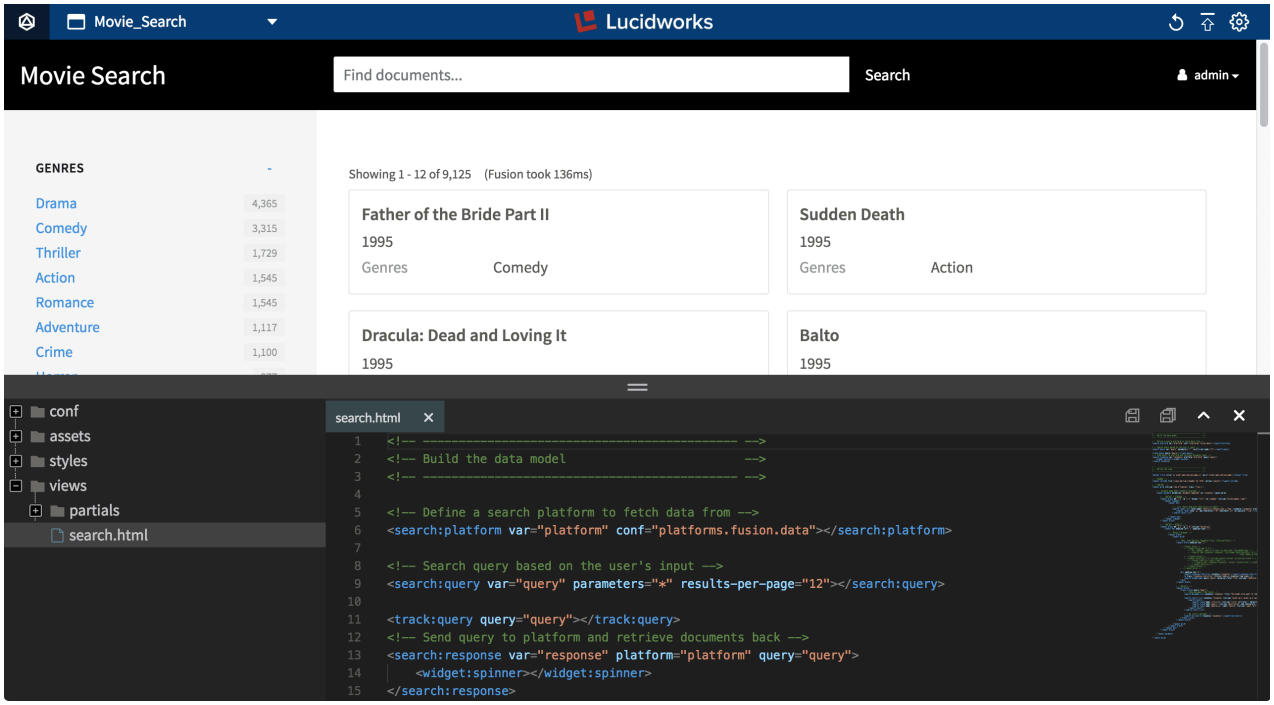
The code editor modifies the file that configures your search interface (`app-studio/app/views/search.html` in version 4.0), using App Studio's *markup tags*. Your search interface reloads automatically when you save your changes in the code editor.

The code editor is also where you can publish your search interface to Fusion Server.

Opening the code editor

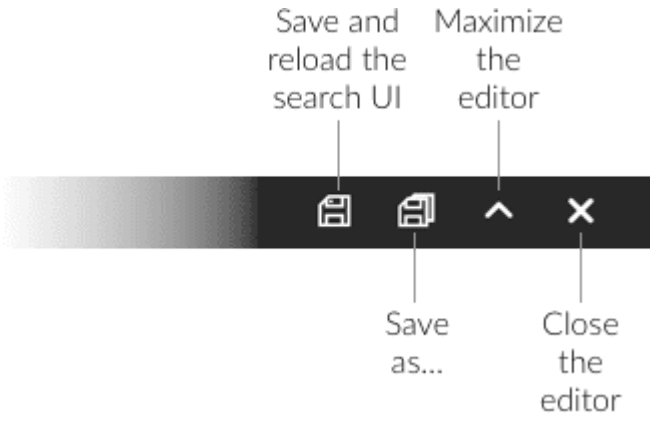
How to open the code editor

- Press the ESC key
  or
- Click the code editor button: </>



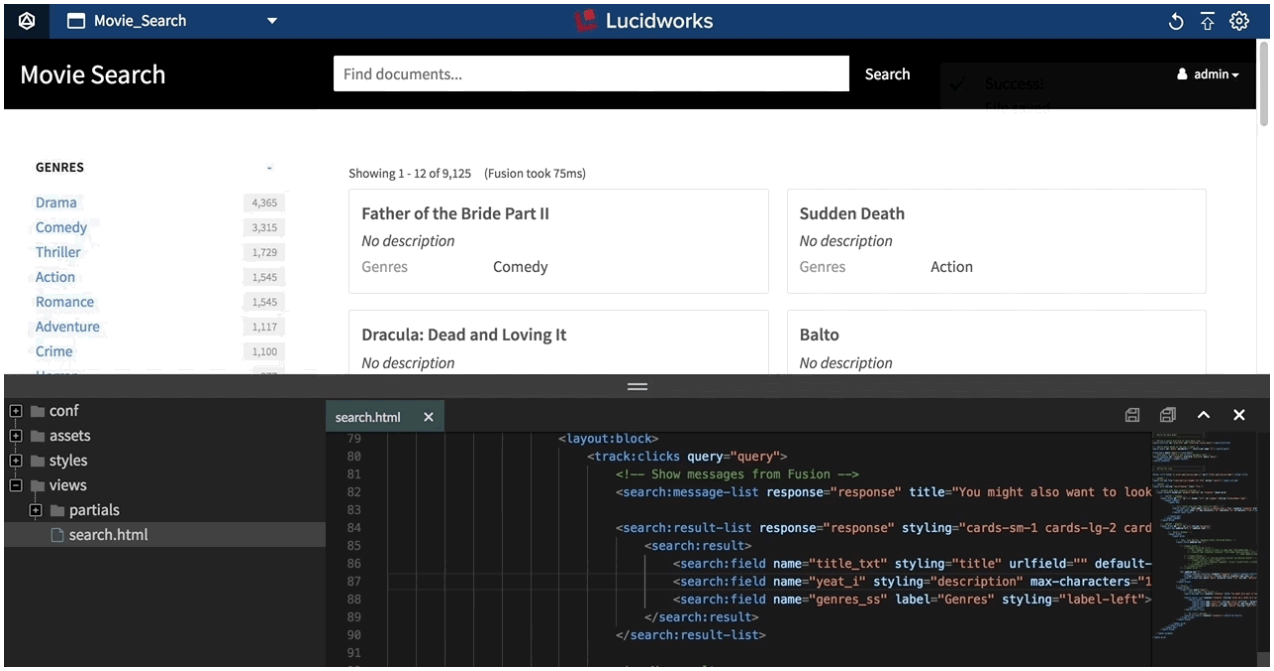
The toolbar

Code editor toolbar



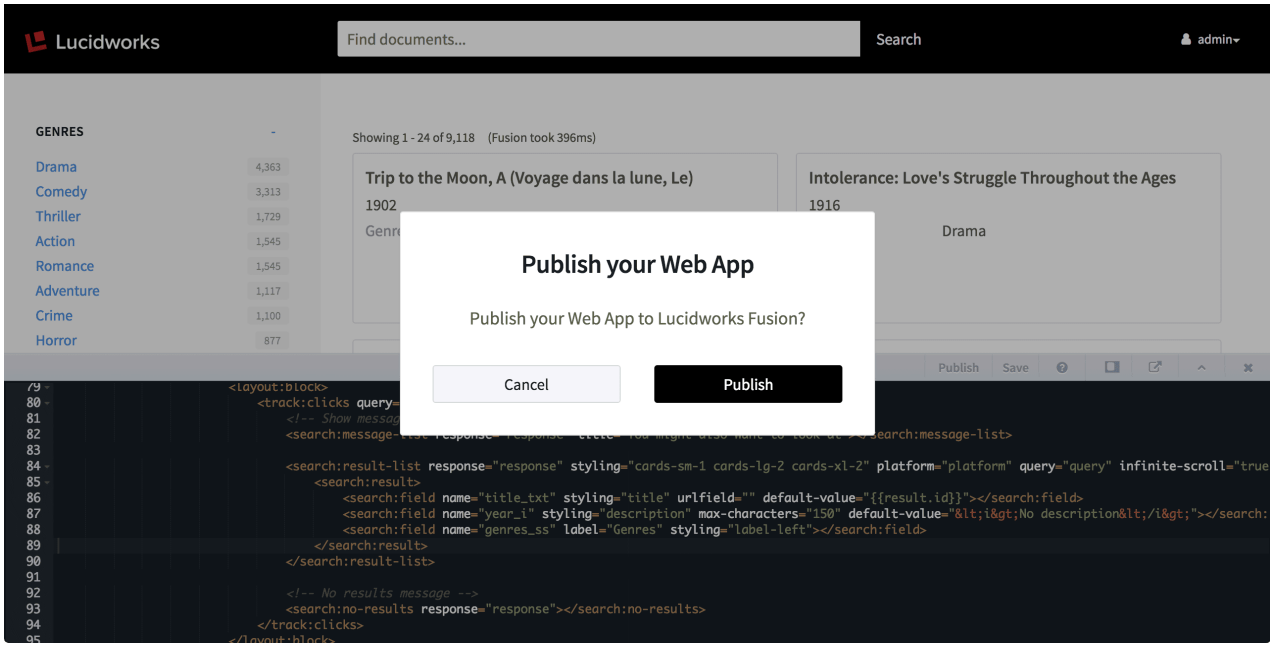
Saving your changes

The **Save** button (or `⌘–S` / `CTRL–S`) saves the file and reloads the page to display your updated interface:



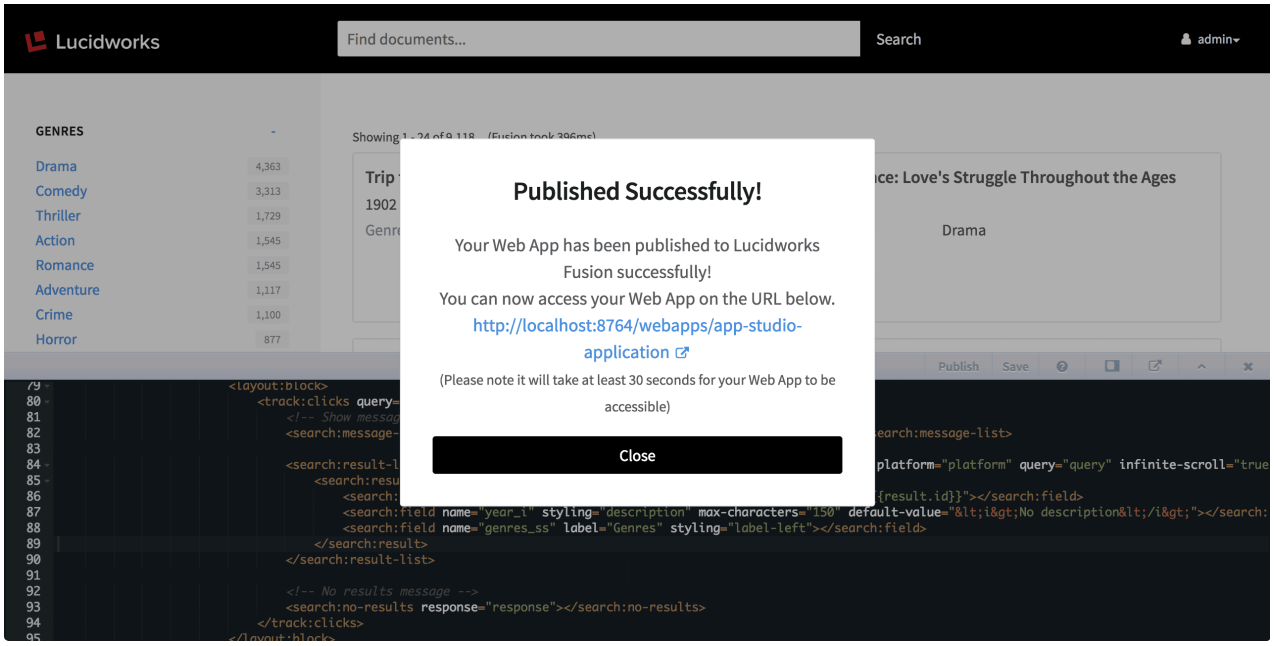
Publishing your search interface from the code editor (version 4.0)

The **Publish** button link:/how-to/631/deploy-a-search-app-in-fusions-webapps-service-4-2 [deploys your search interface to Fusion]. App Studio prompts you to confirm that you want to publish your Web app to Fusion:



Click **Publish** to complete the deployment.

When deployment is complete, App Studio displays the Fusion URL where you can access your search interface:


Lucidworks
Find documents...
Search
admin
GENRES
Drama 4,363
Comedy 3,313
Thriller 1,729
Action 1,545
Romance 1,545
Adventure 1,117
Crime 1,100
Horror 877
Published Successfully!
Your Web App has been published to Lucidworks Fusion successfully!
You can now access your Web App on the URL below.
http://localhost:8764/webapps/app-studio-application
(Please note it will take at least 30 seconds for your Web App to be accessible)
Close
Drama
Publish
Save

Use the Object Explorer

How-to

> Compatible versions: 4.2, 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion Server*, *Fusion*

LucidAcademy

Lucidworks offers free training to help you get started.

The **Course** for **The Object Explorer** focuses on how to view and manage objects such as collections, datasources, index and query pipelines, index and query profiles, and parsers within apps and share them to other applications:

Visit the LucidAcademy to see the full training catalog.

Get started

How to open Object Explorer

On a PC, press Ctrl+K. On a Mac, press Command ⌘+K. Alternatively, click **System > Object Explorer**.

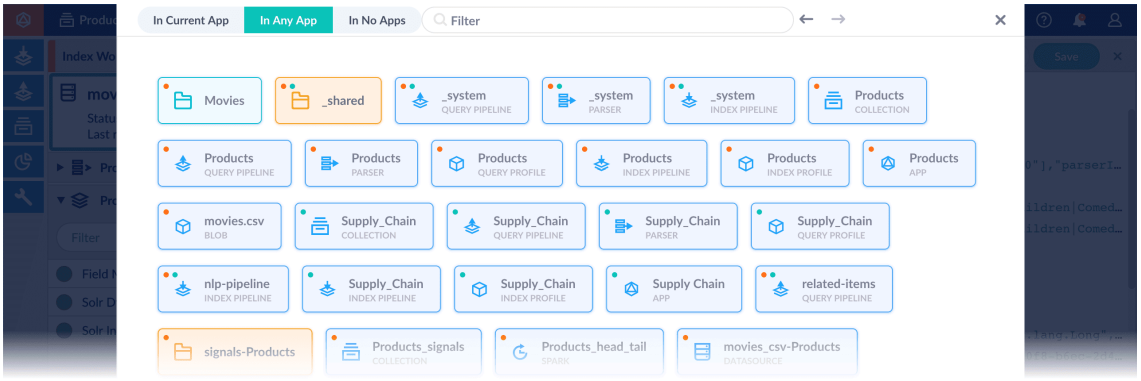
Object Explorer opens and displays either a list of all objects in the current app, or the list of objects that match your most recent search in Object Explorer, whether during this Fusion session or a previous session.

Search and browse

- **Specify the context** To restrict the scope of searching and browsing, click **In Current App**, **In Any App**, or **In No Apps**.
- **Browse all objects in the context** – Specify the context. Object Explorer displays a list of all objects. If you do not see a list of objects, press Enter in the **Filter** box.
- **Search for objects** – Specify the context. Type any part of an object's name or type (the gray, uppercase text below the name, for example, `PARSER`) in the **Filter** box. You can enter `group` to search for groups. Enter the search term in any case. Object Explorer displays a list of matching objects.

- **Display related objects** (with a mouse) – With a list of objects present, click an object to select it. Object Explorer centers the object and displays all related objects.
- **Display related objects** (with Tab keys) – With a list of objects present, tab from object to object. With an object highlighted, press Enter to select it. Object Explorer centers the object and displays all related objects.
- **Move backward and forward through a history of results** – You can move backward and forward through a history of results, that is, the objects you have browsed to, searched for, and clicked on. To the right of the **Filter** box, click the arrow keys.
- **Determine which apps an object is in** – Click **In any App**. You will see dots on the objects that represent the apps that the objects are in. To view a list of the apps for an object, hover over the object, and then

click the App icon.



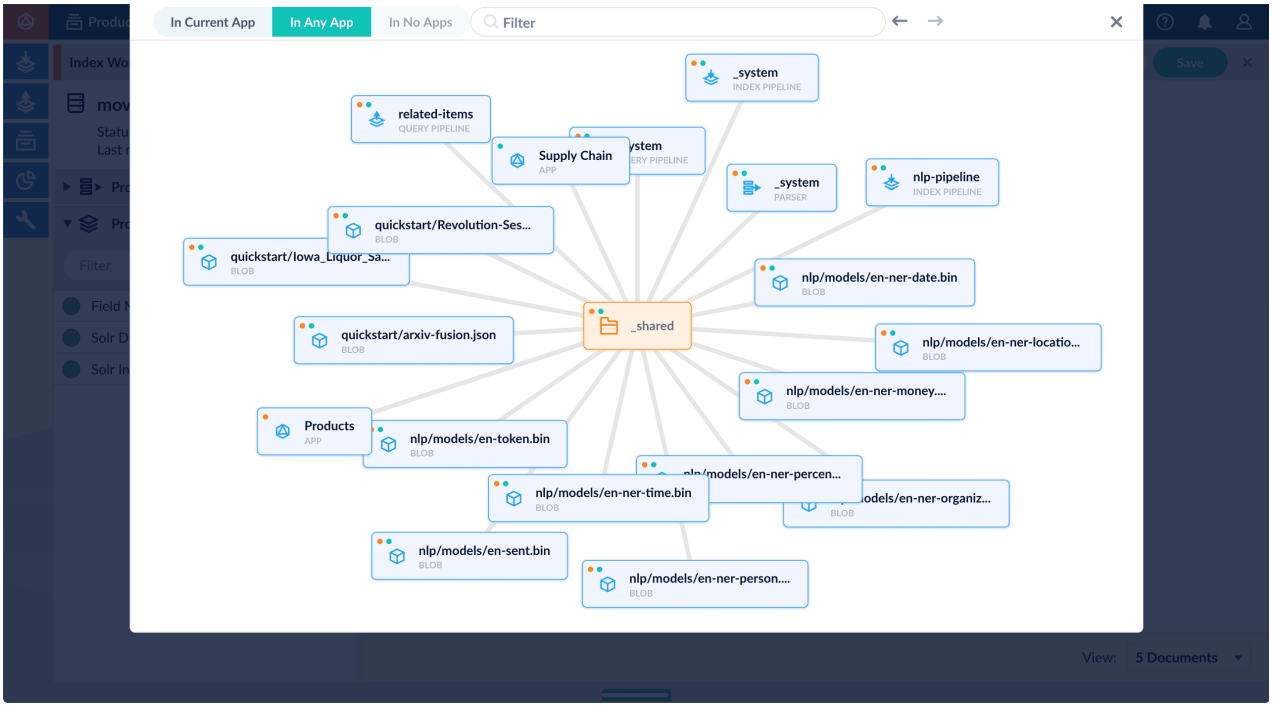
Manage objects

- **Edit an object's name** – For some objects, for example, Group objects, clicking lets you edit the object's name.
- **Manage an object** – For most objects that display the icon when you hover over the object, takes you to the part of the Fusion UI where you can manage the object.
- **Schedule a job or task** – Hover over an object that displays the Schedule icon when you hover.

  Click Schedule . From the scheduling dialog, you can click **Open Schedule** to open the Scheduler.

Group objects

Group objects to easily browse related objects. Fusion groups some objects for you. This is an example of viewing a group, in this case, the `_shared` group:



There are four kinds of groups:

- **System groups** ( ) - Groups created by Fusion to group objects such as signals and items that are shared among apps. You cannot modify the membership of a system group.
- **Recommendations groups** ( ) - Groups related to recommendations. These are a subtype of system groups. You cannot modify the membership of a recommendations group.
- **User-defined groups** ( ) - Groups you create and manage yourself. You can put whatever objects you like in them.
- **Fusion-created groups that are not system groups** ( ) - These look and behave like user-defined groups, but you do not create them. Fusion does. You can put objects in them.

Procedures for managing groups are:

- **View existing groups** – A *group* is a means of tagging objects with a shared, arbitrary identifier.
- **Create a user-defined group** – Hover over an object that you want to add to the group you are creating, and that displays the Add To Group icon when you hover. Click Add To Group . Enter the name of a new group.
  A group name can only contain the characters `A–Z`, `a–z`, `0–9`, `-` (hyphen), and `_` (underscore).

- **Add an object to a group** – Hover over an object that displays the Add To Group icon when you hover. Click Add To Group . Choose an existing group, or enter the name of a new group. You cannot add objects to the group `_shared`. Its members are calculated for you.
- **Remove an object from a group** – Select the group from which you want to remove an object. Object Explorer centers the group. Hover over the object you want to remove, and then click Remove From Group .
- **Delete a user-defined group** – When you remove the last object from a user-defined group, the group remains. To delete it, hover over the group, and then click .

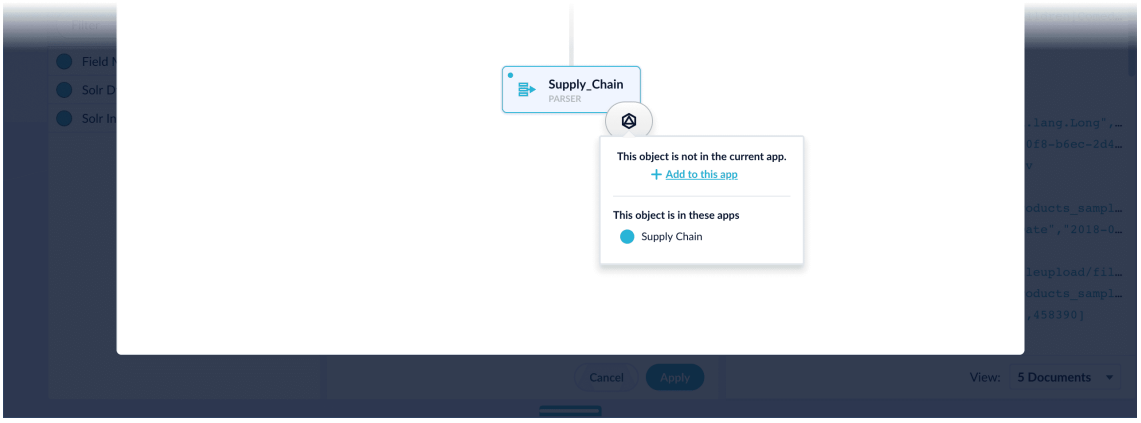
Share objects among apps

Sharing objects among apps lets you configure something once and use it multiple times where needed. When you share an object, the object and any objects that the objects depends on are shared.

Caution

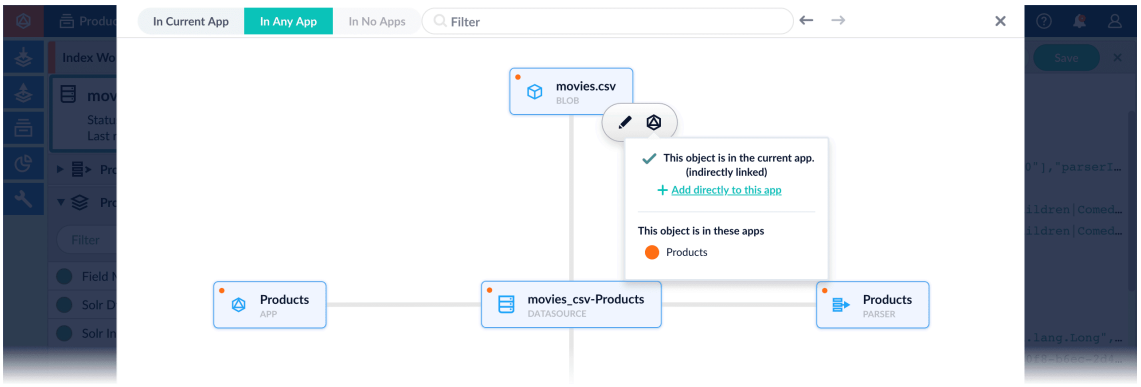
Fusion makes intelligent choices regarding what is shared among apps. Exercise caution when adding and removing objects from apps.

- **Add an object to an app** – While in the Fusion workspace for the app to which you want to add an object, open Object Explorer and click **In Any App**. Search for or browse to the object you want to add. Hover over the object, click the App icon, and then click **Add to this app**.

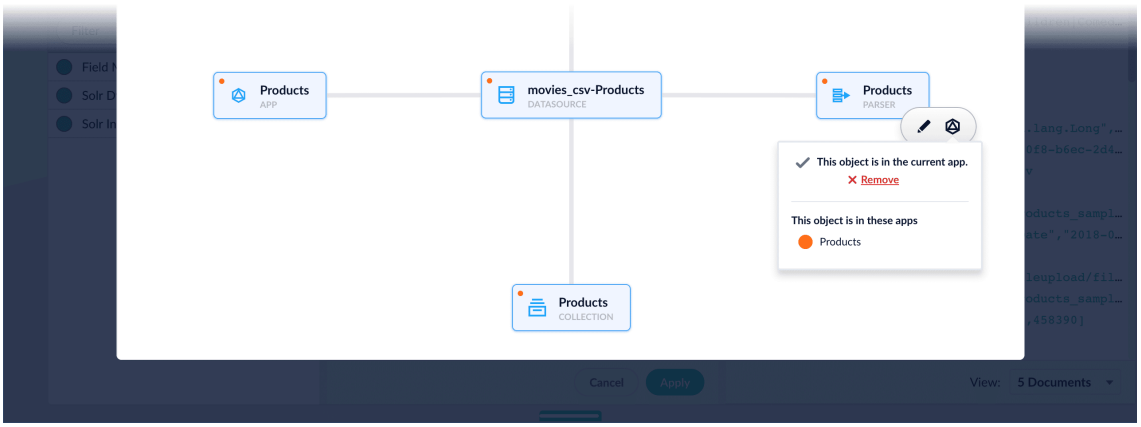


- **Add an object to an app directly** – In cases when an object is linked to an app, but is not linked directly to the app (it is linked via some dependency), you can add the object to an app directly.
  While in the Fusion workspace for the app to which you want to add an object directly, open Object Explorer and click **In Any App**. Search for or browse to the object you want to add. Hover over the object,

click the App icon, and then click **Add to this app directly**.



- **Remove an object from an app** –While in the Fusion workspace for the app from which you want to remove an object, open Object Explorer and click **In Any App**. Search for or browse to the object you want to remove. Hover over the object, click the App icon, and then click **Remove**.



Close Object Explorer

On a PC or Mac, press Esc or click Close . Alternatively on a Mac, you can press Command ⌘+K.

Use the Object Explorer

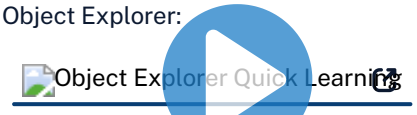
How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

LucidAcademy

Lucidworks offers free training to help you get started.

The **Quick Learning** for **Object Explorer Quick Learning** focuses on a brief overview of the Object Explorer:



Visit the LucidAcademy to see the full training catalog.

LucidAcademy

Lucidworks offers free training to help you get started.

The **Course** for **The Object Explorer** focuses on how to view and manage objects such as collections, datasources, index and query pipelines, index and query profiles, and parsers within apps and share them to other applications:

Visit the LucidAcademy to see the full training catalog.

Get started

How to open Object Explorer

Click **System** > **Object Explorer**.

Object Explorer opens and displays either a list of all objects in the current app, or the list of objects that match your most recent search in Object Explorer, whether during this Managed Fusion session or a previous session.

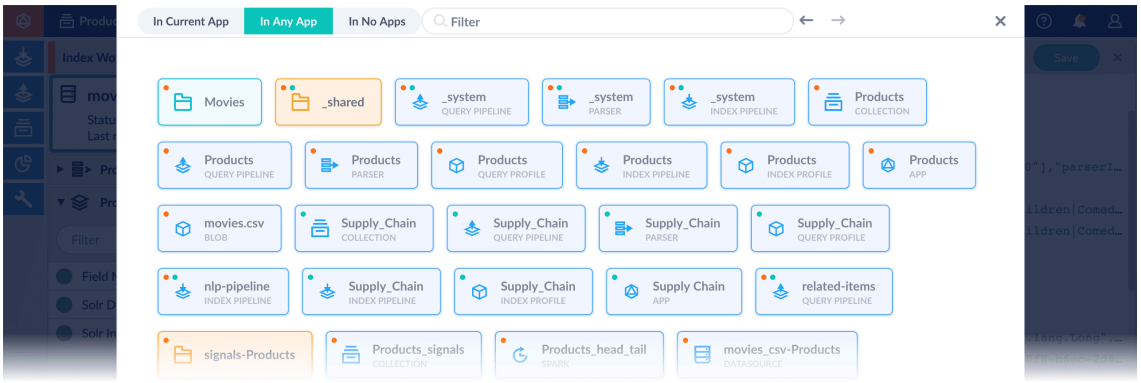
Search and browse

- **Specify the context** To restrict the scope of searching and browsing, click **In Current App**, **In Any App**, or **In No Apps**.
- **Browse all objects in the context** – Specify the context. Object Explorer displays a list of all objects. If you do not see a list of objects, press Enter in the **Filter** box.
- **Search for objects** – Specify the context. Type any part of an object's name or type (the gray, uppercase text below the name, for example, `PARSER`) in the **Filter** box. You can enter `group` to search for

groups. Enter the search term in any case. Object Explorer displays a list of matching objects.

- **Display related objects** (with a mouse) – With a list of objects present, click an object to select it. Object Explorer centers the object and displays all related objects.
- **Display related objects** (with Tab keys) – With a list of objects present, tab from object to object. With an object highlighted, press Enter to select it. Object Explorer centers the object and displays all related objects.
- **Move backward and forward through a history of results** – You can move backward and forward through a history of results, that is, the objects you have browsed to, searched for, and clicked on. To the right of the **Filter** box, click the arrow keys.
- **Determine which apps an object is in** – Click **In any App**. You will see dots on the objects that represent the apps that the objects are in. To view a list of the apps for an object, hover over the object, and then

  click the App icon.



Manage objects

- **Edit an object's name** – For some objects, for example, Group objects, clicking lets you edit the

  object's name.

- **Manage an object** – For most objects that display the icon when you hover over the object,

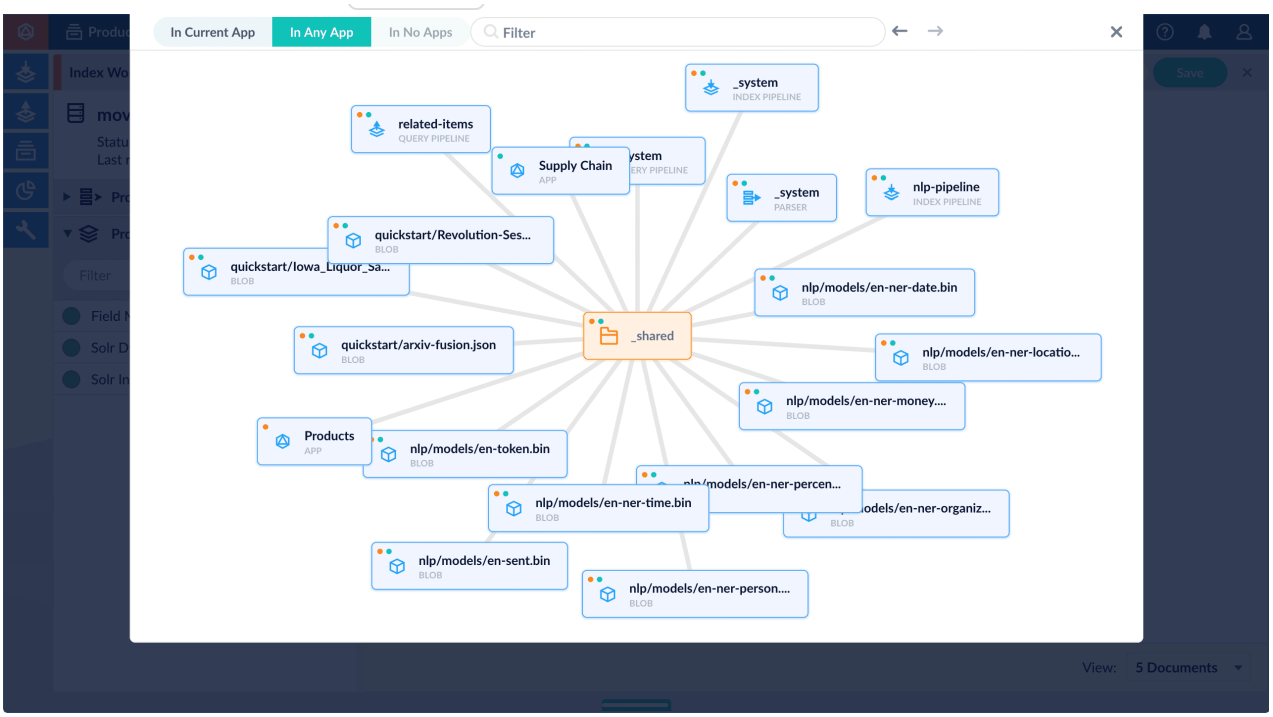
  takes you to the part of the Managed Fusion UI where you can manage the object.

- **Schedule a job or task** – Hover over an object that displays the Schedule icon when you hover.

  Click Schedule . From the scheduling dialog, you can click **Open Schedule** to open the Scheduler.

Group objects

Group objects to easily browse related objects. Managed Fusion groups some objects for you. This is an example of viewing a group, in this case, the `_shared` group:



There are four kinds of groups:

- **System groups** ( ) - Groups created by Managed Fusion to group objects such as signals and items that are shared among apps. You cannot modify the membership of a system group.
- **Recommendations groups** ( ) - Groups related to recommendations. These are a subtype of system groups. You cannot modify the membership of a recommendations group.
- **User-defined groups** ( ) - Groups you create and manage yourself. You can put whatever objects you like in them.
- **Managed Fusion-created groups that are not system groups** ( ) - These look and behave like user-defined groups, but you do not create them. Managed Fusion does. You can put objects in them.

Procedures for managing groups are:

- **View existing groups** – A *group* is a means of tagging objects with a shared, arbitrary identifier.
- **Create a user-defined group** – Hover over an object that you want to add to the group you are creating, and that displays the Add To Group icon when you hover. Click Add To Group . Enter the name of a new group.
  A group name can only contain the characters `A–Z`, `a–z`, `0–9`, `-` (hyphen), and `_` (underscore).

- **Add an object to a group** – Hover over an object that displays the Add To Group icon when you hover. Click Add To Group . Choose an existing group, or enter the name of a new group. You cannot add objects to the group `_shared`. Its members are calculated for you.
- **Remove an object from a group** – Select the group from which you want to remove an object. Object Explorer centers the group. Hover over the object you want to remove, and then click Remove From Group .
- **Delete a user-defined group** – When you remove the last object from a user-defined group, the group remains. To delete it, hover over the group, and then click .

Share objects among apps

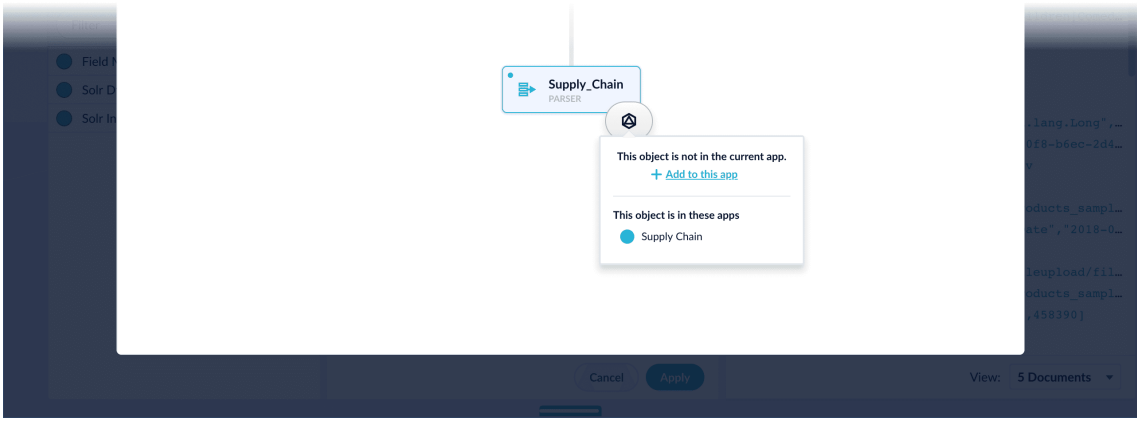
Sharing objects among apps lets you configure something once and use it multiple times where needed. When you share an object, the object and any objects that the objects depends on are shared.

Caution

Managed Fusion makes intelligent choices regarding what is shared among apps. Exercise caution when adding and removing objects from apps.

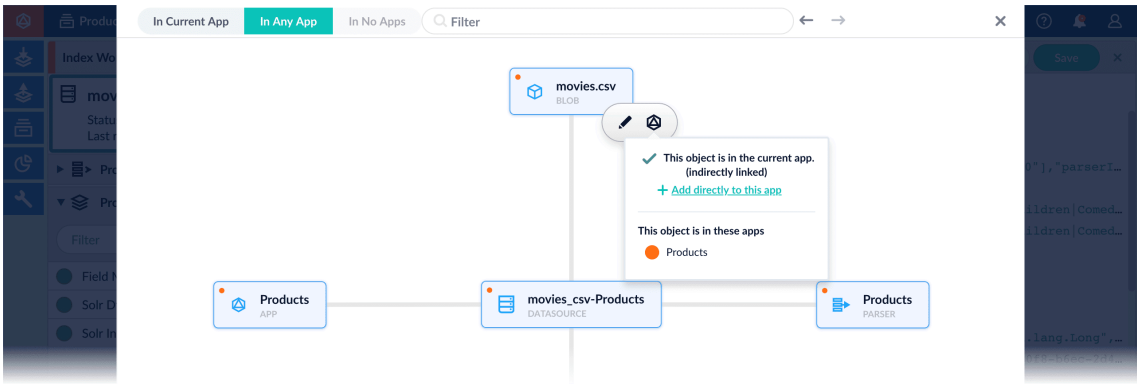
- **Add an object to an app** – While in the Managed Fusion workspace for the app to which you want to add an object, open Object Explorer and click **In Any App**. Search for or browse to the object you want to add.

  Hover over the object, click the App icon, and then click **Add to this app**.

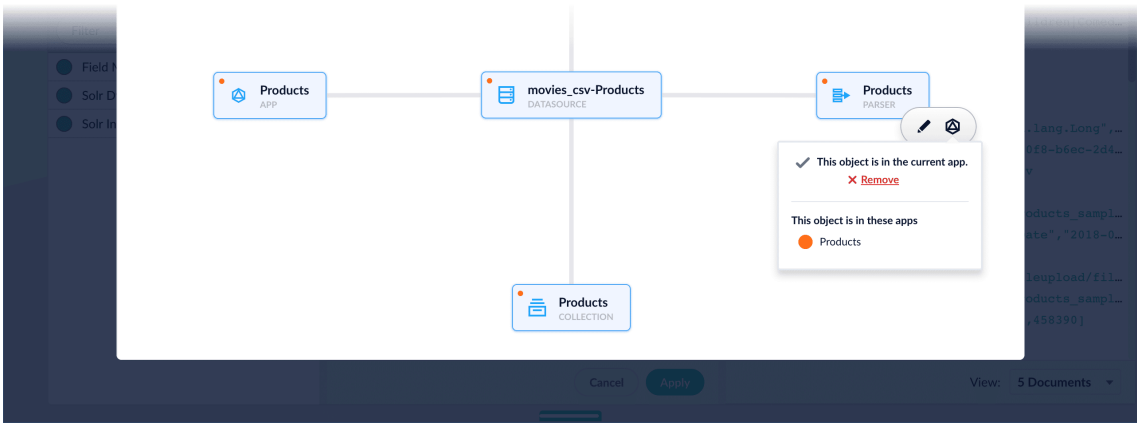


- **Add an object to an app directly** – In cases when an object is linked to an app, but is not linked directly to the app (it is linked via some dependency), you can add the object to an app directly.
  While in the Managed Fusion workspace for the app to which you want to add an object directly, open Object Explorer and click **In Any App**. Search for or browse to the object you want to add. Hover over the

object, click the App icon, and then click **Add to this app directly**.



- **Remove an object from an app** –While in the Managed Fusion workspace for the app from which you want to remove an object, open Object Explorer and click **In Any App**. Search for or browse to the object you want to remove. Hover over the object, click the App icon, and then click **Remove**.



Close Object Explorer

On a PC or Mac, press Esc or click Close .

Use the Pipeline Stage JSON Editor

How-to

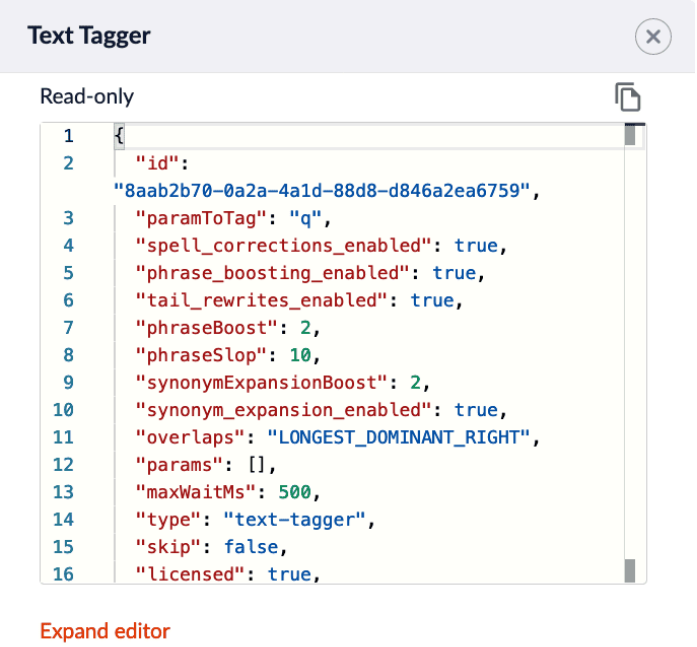
> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion*

This article will focus on copying the JSON configuration for an existing pipeline stage and creating a new stage with the JSON configuration.

Important

The instructions in this article focus on query pipeline stages. However, the steps are the same for index pipeline stages, which are accessed at **Indexing > Index Pipelines > <pipeline-name>**.

1. Navigate to **Querying > Query Pipelines** and select a pipeline.
2. Select an existing pipeline stage to view in the editor.
3. Click the **JSON View** button. The editor will appear:



Note

Existing stages are considered **READ-ONLY**. They will only have a copy button.

4. (optional) Click **Expand editor** to view the editor in fullscreen mode.
5. Click the **Copy** button. The JSON configuration for that stage is copied to your clipboard.
6. Click **Add a new pipeline stage** to create a new stage. Create the same stage type.
7. Click the **JSON View** button.
8. Use the **Paste** button to paste the JSON configuration from your clipboard into the editor. Alternatively, delete the existing JSON configuration and manually paste the new configuration.
   At this time, you can make changes to the JSON configuration inside the editor.
9. Click the **Apply** button to apply your changes to the stage. Although the changes are immediately shown in the UI, the stage is not saved until you click the **Save** button.
10. Save the stage.

Use the Query Elevation Component

How-to

> Compatible versions: 4.2, 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion Server*, *Fusion AI*, *Fusion*

Fusion currently supports the use of the Query Elevation Component (QEC) with boost lists and filter lists. You must configure Solr in order to enable the QEC option.

Note

Although it will not be accessed for the elevation process, Fusion requires an independent `elevate.xml` file for the use of built-in rule actions with QEC due to a known issue in Solr.

1. Create an XML file named `elevate.xml` in the same directory as the `solrconfig.xml` file.
2. Use the following as the contents of the `elevate.xml` file:

```
<?xml version="1.0" encoding="UTF-8" ?> <elevate> </elevate>
```

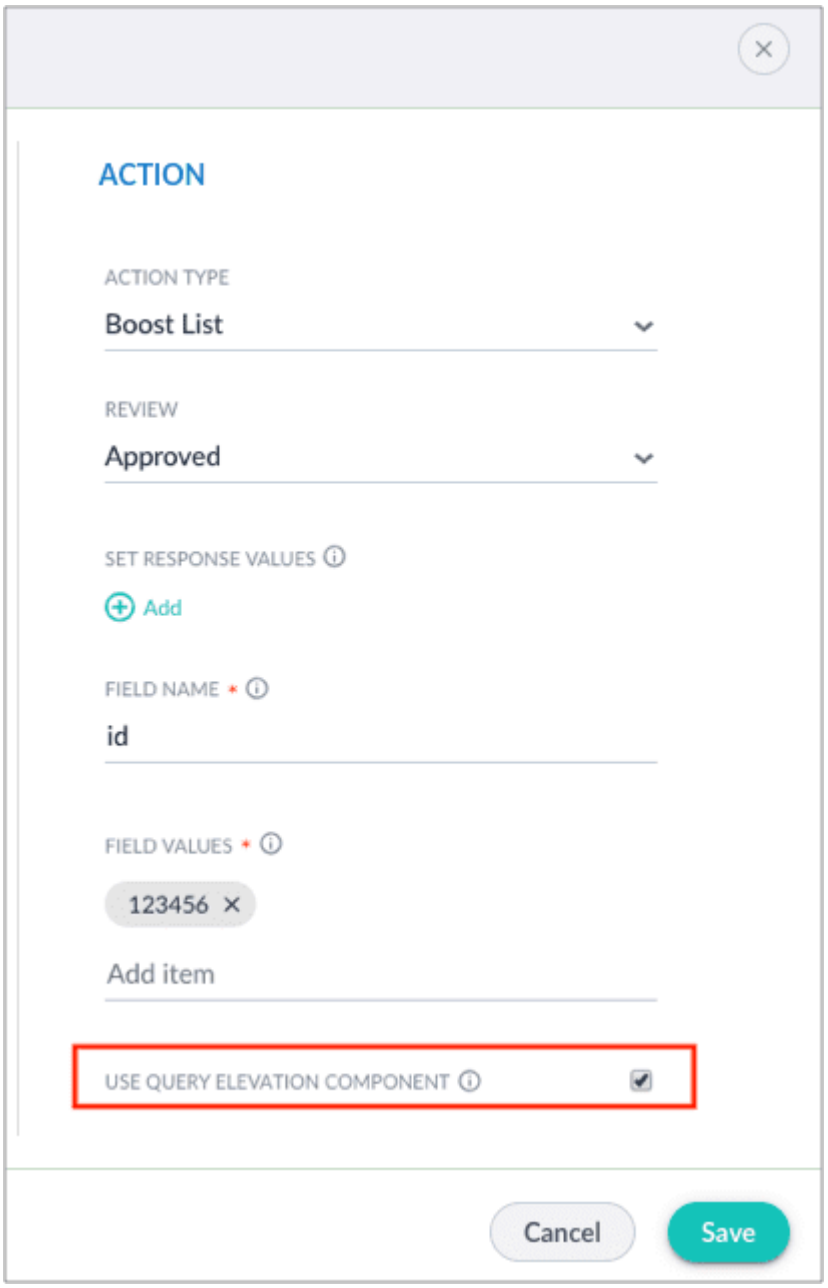
3. In the `solrconfig.xml` file, add the following in the elevator search Component (`searchComponent name="elevator"`). Before adding, check to see if the `searchComponent` already exists within the file.

```
<searchComponent name="elevator"
class="solr.QueryElevationComponent" > <str name="config-
file">elevate.xml</str> </searchComponent>
```

4. Add `elevator` as a 'last component' part of the select handler in the `solrconfig.xml` file. If a `last-components` array is not already defined, create it at the end of the select handler's configuration. If one is already defined, add `<str>elevator</str>` .

```
<requestHandler name="/elevate" class="solr.SearchHandler"> <arr
name="last-components"> <str>elevator</str> </arr>
</requestHandler>
```

5. When you *create a new business rule*, check the **USE QUERY ELEVATION COMPONENT** checkbox to use elevation.



Important

The Query Elevation Component only elevates documents within the *query rewriting rules engine* by the document `id` field. Ensure `id` is entered in the "FIELD NAME" option.

Use the Query Elevation Component

How-to

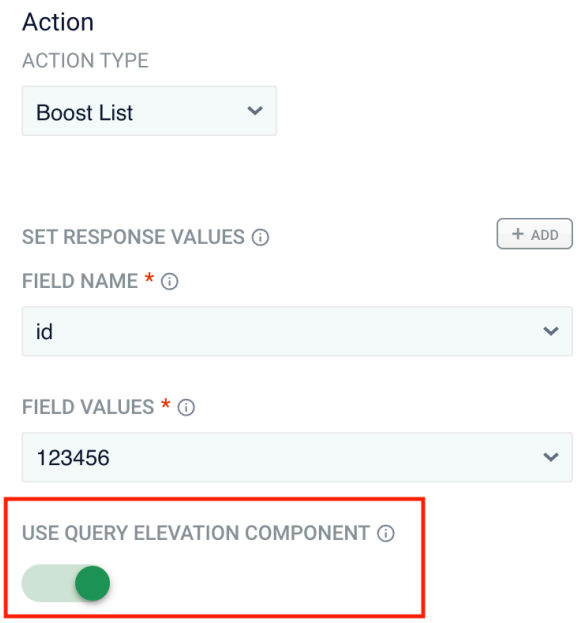
> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

Managed Fusion supports the use of the Query Elevation Component (QEC) with boost lists and filter lists.

Note

Contact your Lucidworks representative to enable this feature.

When you *create a new business rule*, check the **USE QUERY ELEVATION COMPONENT** checkbox to use elevation.



Important

The Query Elevation Component only elevates documents within the *query rewriting rules engine* by the document `id` field. Ensure `id` is entered in the "FIELD NAME" option.

Use the UI to Create Custom Rules

How-to

> Compatible versions: 4.2
>
> Works with: *Fusion AI*

Important

For **My Custom Rules** to show in the UI, you must first create a `custom_rule_type` via the *Query Rewrite API*.

1. Navigate to **Relevance** > **Query Rewriting**.
2. Under **Business Rules**, click **View**.
3. Click the ⊕ icon.

   The **New Rule** window appears.
4. Enter the general parameters and *conditions* for this custom rule.
5. Under **Action**, select **My Custom Rule**.
6. Select the query pipeline to use when this rule is triggered.
7. Enter a name for this custom rule type.
8. Optionally, you can click **Add** under **Return Parameter Policy Override** to override the value of a parameter returned from the specified pipeline:
   1. Enter the name of the parameter.
   2. Enter the desired value.
   3. Select one of the following policies:
      - **Replace** the returned value with the specified value.
      - **Append** the specified value to the returned value.
      - **Remove** the specified parameter from the set of returned parameters.
      - **Default**
   4. Enter a value for **From Custom Rule**.

Use Tika asynchronous parsing

How-to

> Compatible versions: 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

This document describes how to set up your application to use Tika asynchronous parsing.

Unlike synchronous Tika parsing, which uses a parser stage, asynchronous Tika parsing is configured in the datasource and index pipeline. For more information, see *Asynchronous Tika Parsing*.

Important

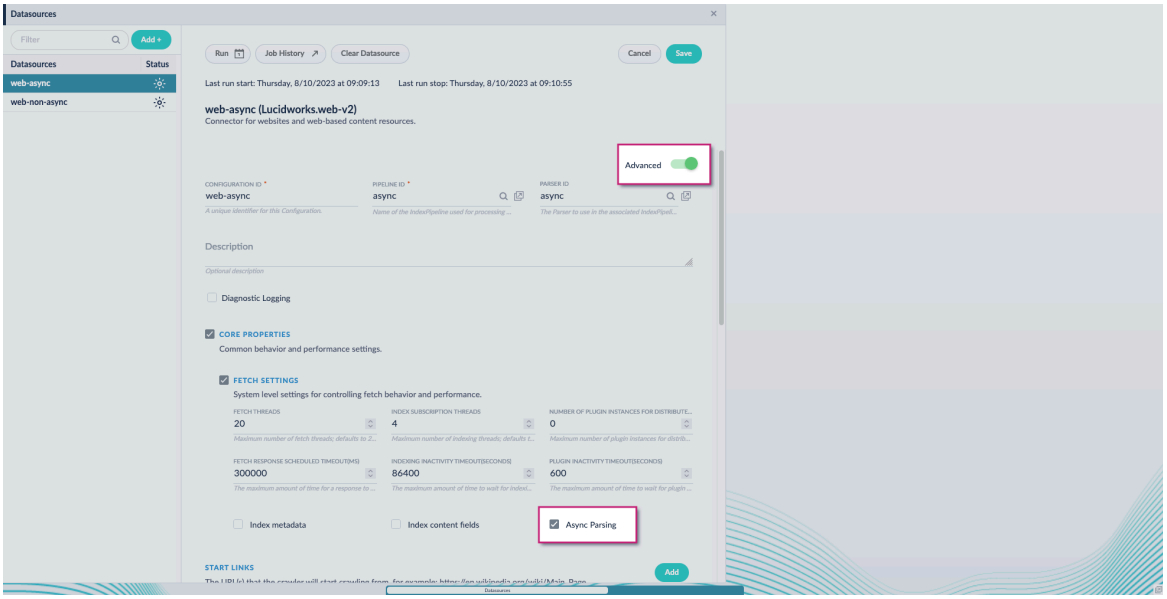
Field names change with asynchronous Tika parsing.

In contrast to synchronous parsing, asynchronous Tika parsing prepends `parser_` to fields added to a document. System fields, which start with `_lw_`, are not prepended with `parser_`.

If you are migrating to asynchronous Tika parsing, and your search application configuration relies on specific field names, update your search application to use the new fields.

Configure the connectors datasource

1. Navigate to your datasource.
2. Enable the **Advanced** view.
3. Enable the **Async Parsing** option.



Important

Managed Fusion 5.9.11 and later uses your parser configuration when using asynchronous parsing.

The asynchronous parsing service performs Tika parsing using Apache Tika Server.

In Managed Fusion 5.8 through 5.9.10, other parsers, such as HTML and JSON, are not supported by the asynchronous parsing service. By enabling asynchronous parsing, the parser configuration linked to your datasource is ignored.

In Managed Fusion 5.9.11 and later, other parsers, such as HTML and JSON, are supported by the asynchronous parsing service. By enabling asynchronous parsing, the parser configuration linked to your datasource is used.

4. Save the datasource configuration.

Configure the parser stage

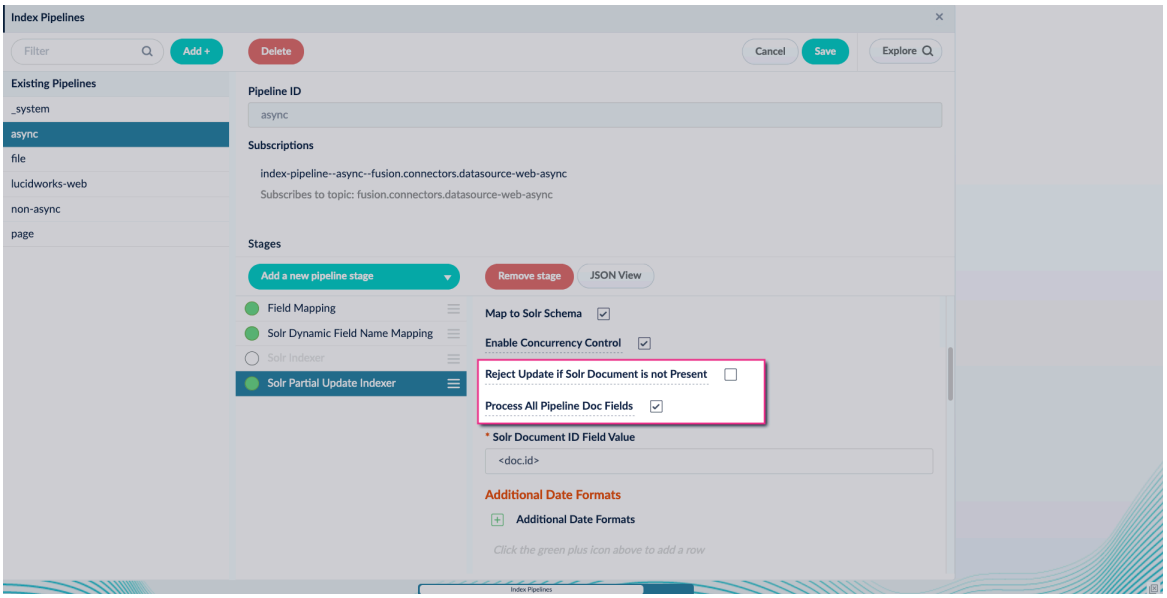
Important

You must do this step in Managed Fusion 5.9.11 and later.

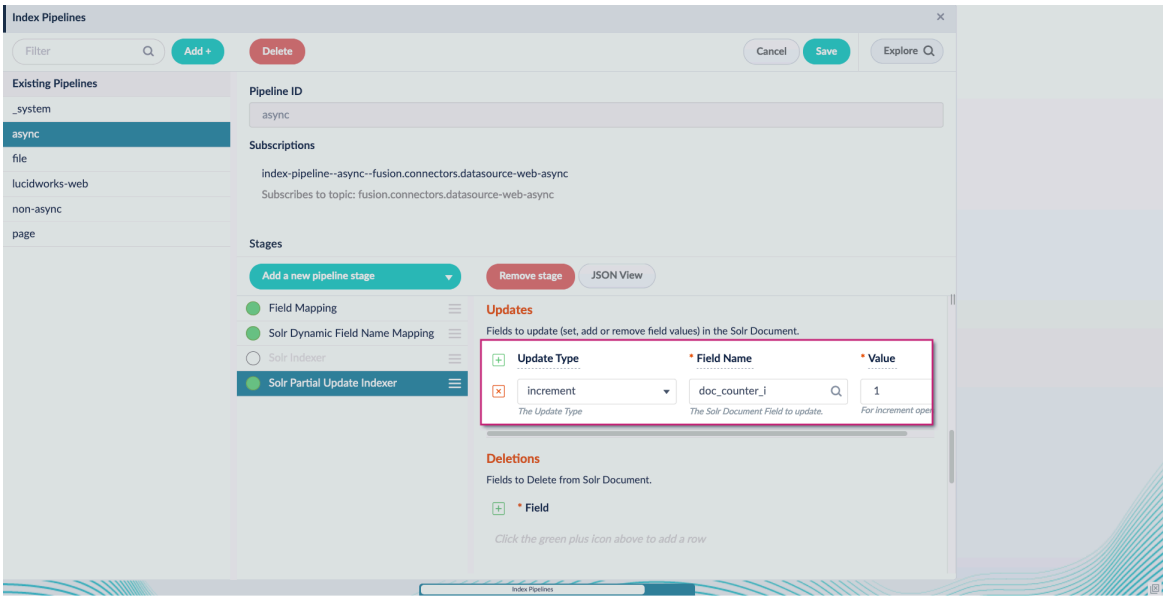
1. Navigate to **Parsers**.
2. Select the parser, or create a new parser.
3. From the **Add a parser stage** menu, select **Apache Tika Container Parser**.
4. (Optional) Enter a label for this stage. This label changes the names from Apache Tika Container Parser to the value you enter in this field.
5. If the Apache Tika Container Parser stage is not already the first stage, drag and drop the stage to the top of the stage list so it is the first stage that runs.

Configure the index pipeline

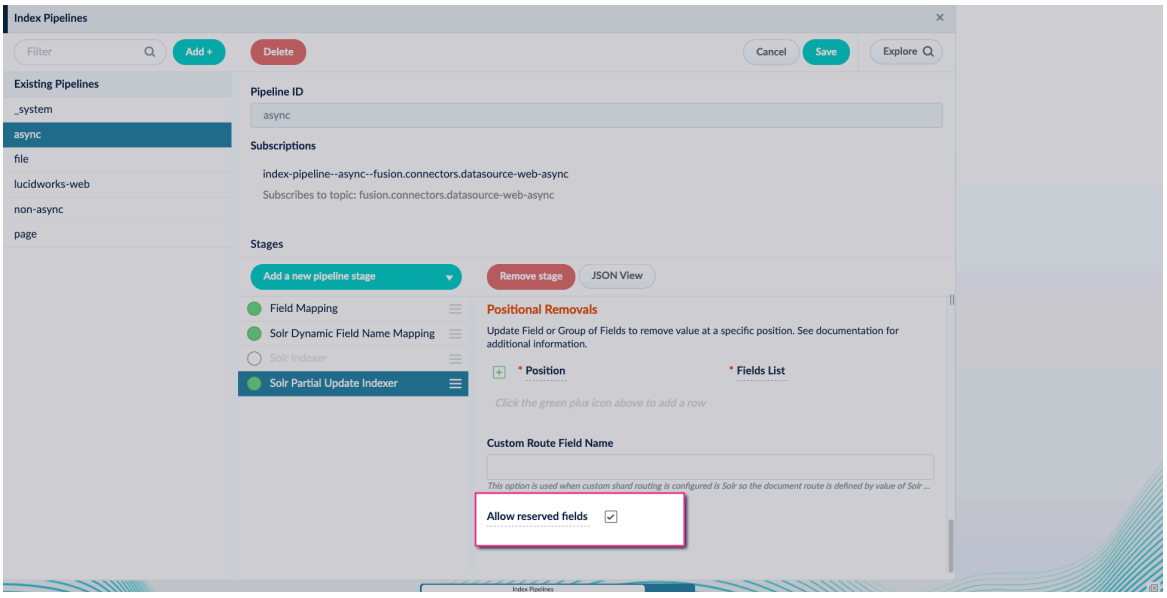
1. Go to the **Index Pipeline** screen.
2. Add the **Solr Partial Update Indexer** stage.
3. Turn off the **Reject Update if Solr Document is not Present** option and turn on the **Process All Pipeline Doc Fields** option:



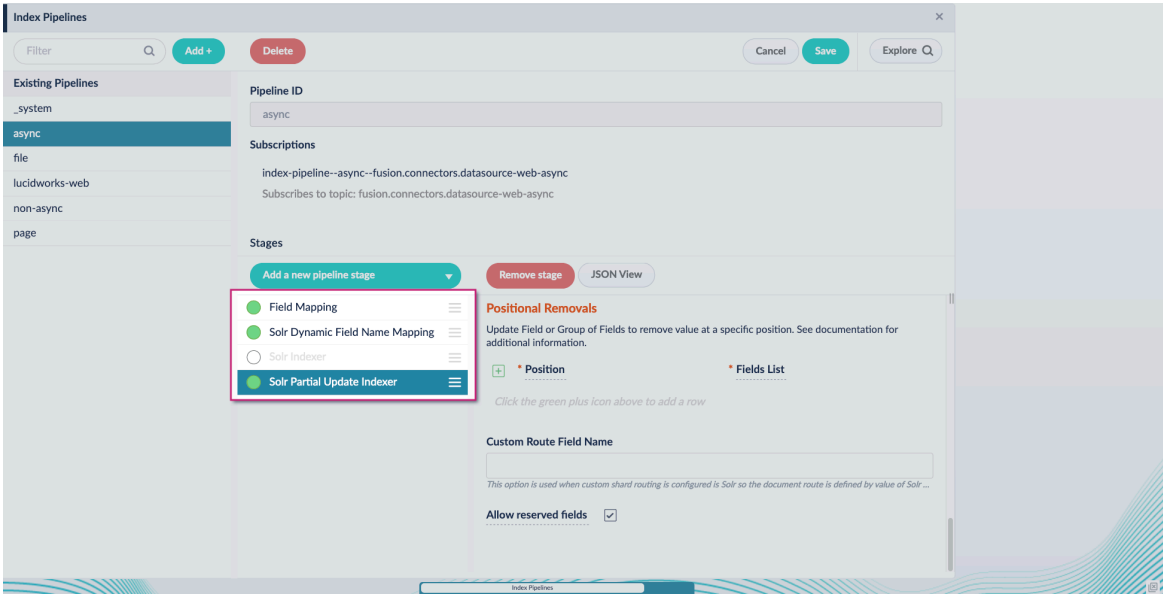
4. Include an extra update field in the stage configuration using any update type and field name. In this example, an incremental field `docs_counter_i` with an increment value of `1` is added:



5. Enable the **Allow reserved fields** option:



6. Click **Save**.
7. Turn off or remove the **Solr Indexer stage**, and move the **Solr Partial Update Indexer stage** to be the last stage in the pipeline.



Asynchronous Tika parsing setup is now complete. Run the datasource indexing job and monitor the results.

Use Tika Asynchronous Parsing

How-to

> Compatible versions: 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion*

This document describes how to set up your application to use Tika asynchronous parsing.

Unlike synchronous Tika parsing, which uses a parser stage, asynchronous Tika parsing is configured in the datasource and index pipeline. For more information, see *Asynchronous Tika Parsing*.

Important

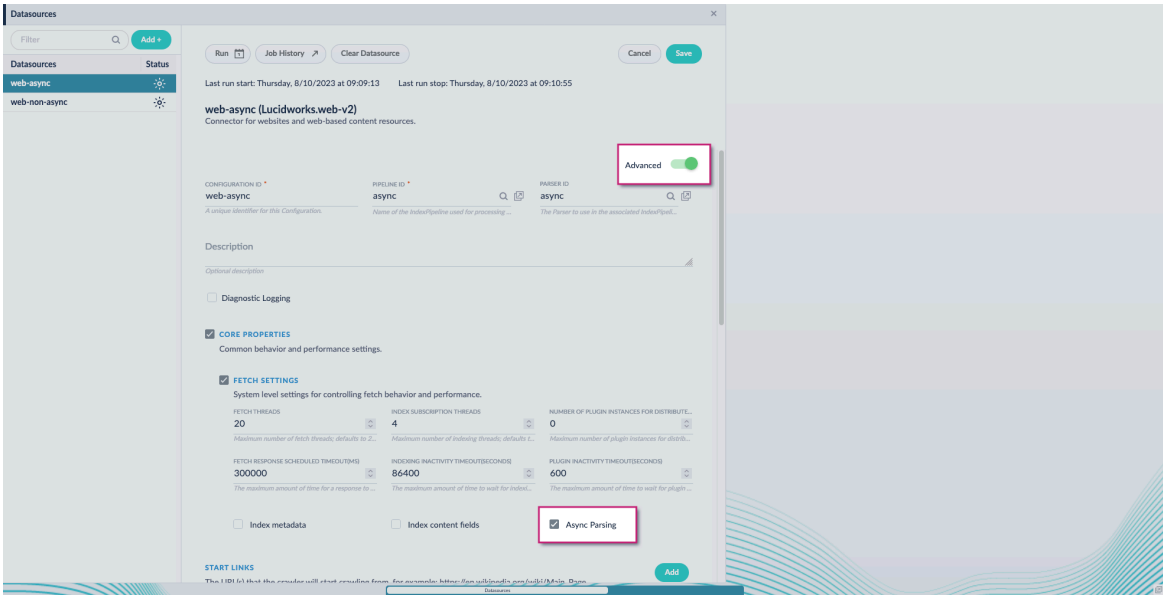
Field names change with asynchronous Tika parsing.

In contrast to synchronous parsing, asynchronous Tika parsing prepends `parser_` to fields added to a document. System fields, which start with `_lw_`, are not prepended with `parser_`.

If you are migrating to asynchronous Tika parsing, and your search application configuration relies on specific field names, update your search application to use the new fields.

Configure the connectors datasource

1. Navigate to your datasource.
2. Enable the **Advanced** view.
3. Enable the **Async Parsing** option.



Important

Fusion 5.9.11 and later uses your parser configuration when using asynchronous parsing.

The asynchronous parsing service performs Tika parsing using Apache Tika Server.

In Fusion 5.8 through 5.9.10, other parsers, such as HTML and JSON, are not supported by the asynchronous parsing service. By enabling asynchronous parsing, the parser configuration linked to your datasource is ignored.

In Fusion 5.9.11 and later, other parsers, such as HTML and JSON, are supported by the asynchronous parsing service. By enabling asynchronous parsing, the parser configuration linked to your datasource is used.

4. Save the datasource configuration.

Configure the parser stage

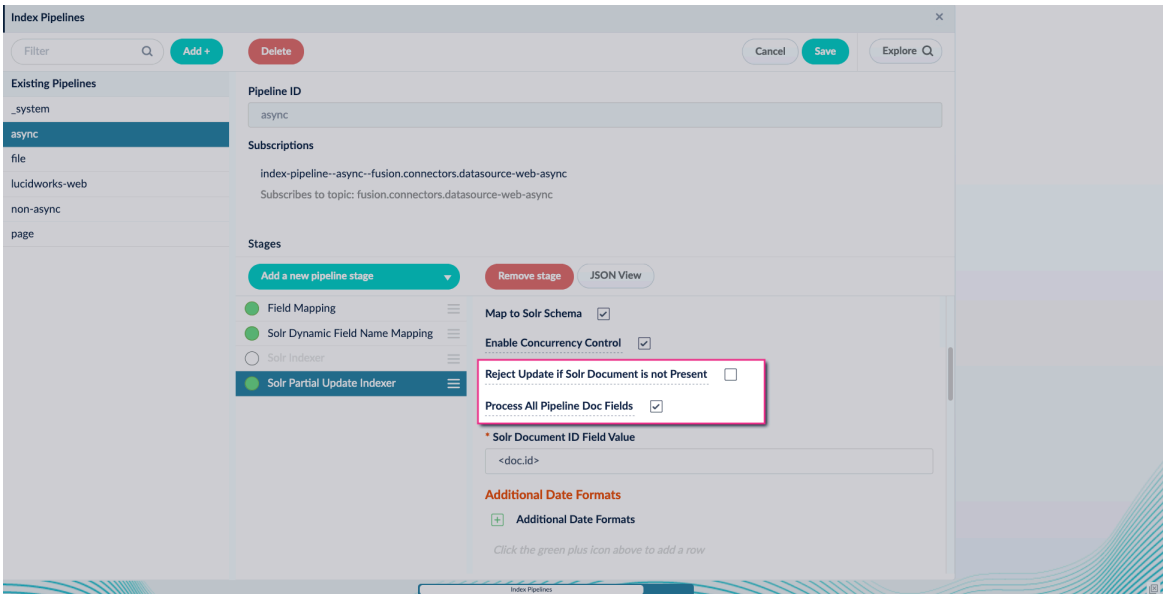
Important

You must do this step in Fusion 5.9.11 and later.

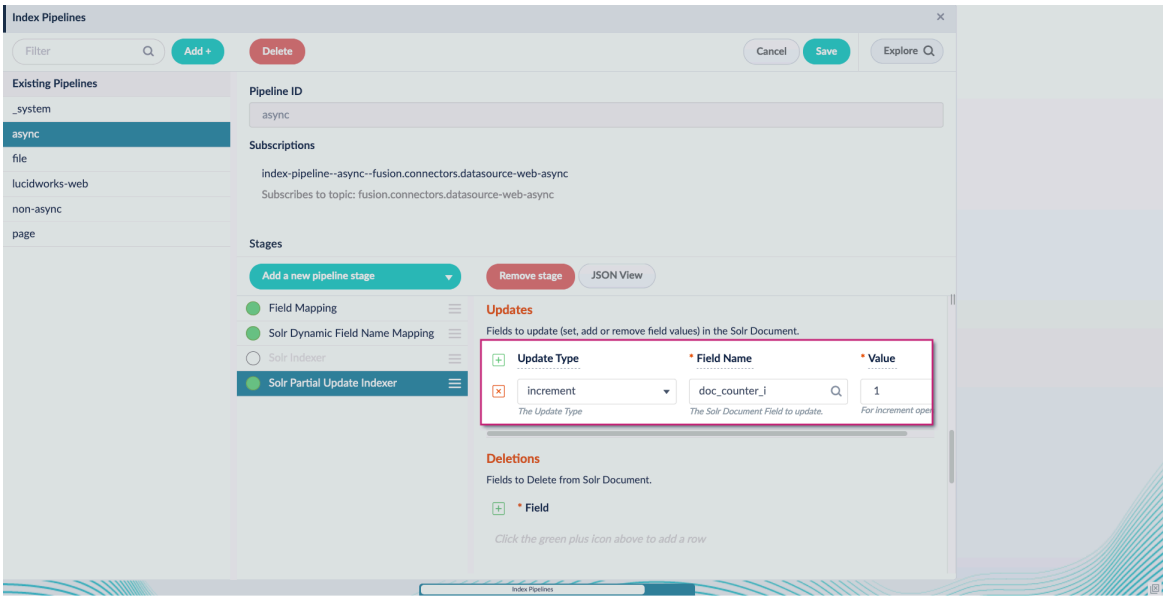
1. Navigate to **Parsers**.
2. Select the parser, or create a new parser.
3. From the **Add a parser stage** menu, select **Apache Tika Container Parser**.
4. (Optional) Enter a label for this stage. This label changes the names from Apache Tika Container Parser to the value you enter in this field.
5. If the Apache Tika Container Parser stage is not already the first stage, drag and drop the stage to the top of the stage list so it is the first stage that runs.

Configure the index pipeline

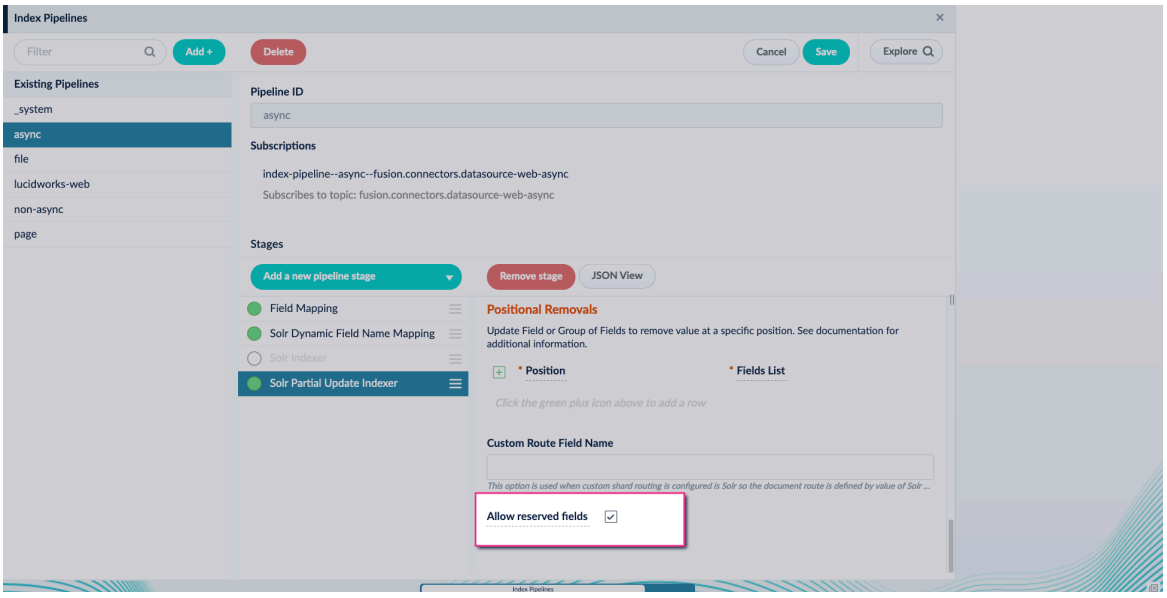
1. Go to the **Index Pipeline** screen.
2. Add the **Solr Partial Update Indexer** stage.
3. Turn off the **Reject Update if Solr Document is not Present** option and turn on the **Process All Pipeline Doc Fields** option:



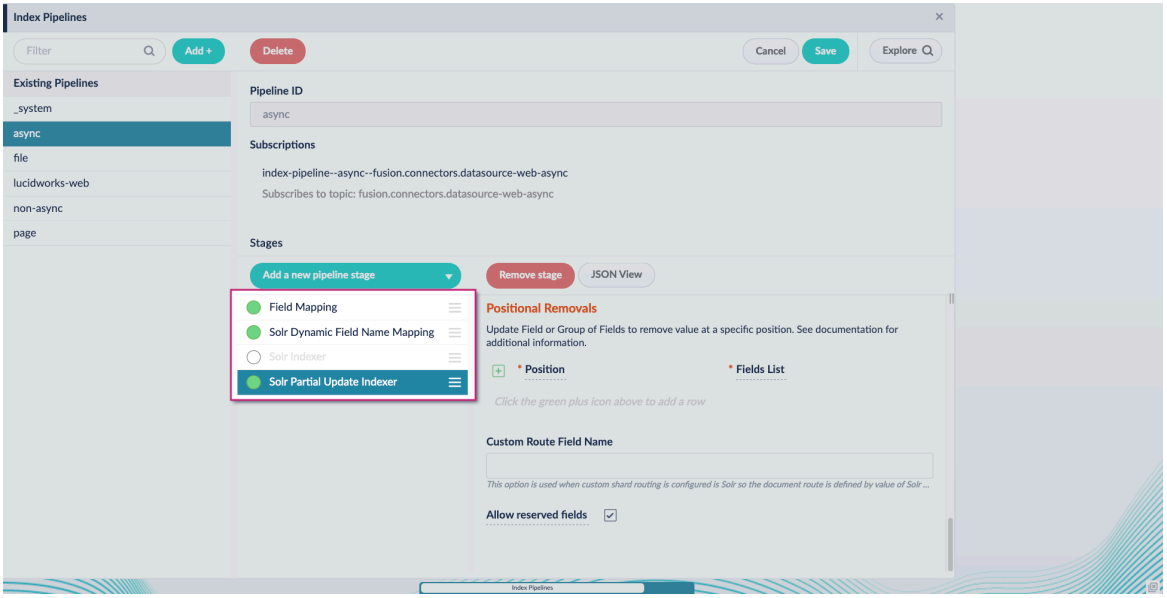
4. Include an extra update field in the stage configuration using any update type and field name. In this example, an incremental field `docs_counter_i` with an increment value of `1` is added:



5. Enable the **Allow reserved fields** option:



6. Click **Save**.
7. Turn off or remove the **Solr Indexer stage**, and move the **Solr Partial Update Indexer stage** to be the last stage in the pipeline.



Asynchronous Tika parsing setup is now complete. Run the datasource indexing job and monitor the results.

Use Virtual Tables with a Common Join Key

How-to

> Compatible versions: 4.2, 5.5
>
> Works with: *Fusion Server*, *Fusion*

With Solr, you can index different document types into the same shard using the composite ID router based on a common route key field. For example, a customer 360 application can index different customer-related document types (contacts, apps, support requests, and so forth) into the same collection, each with a common `customer_id` field. This lets Solr perform optimized joins between the document types using the route key field. This configuration uses Solr's composite ID routing, which ensures that all documents with the same join key field end up in the same shard. See Document Routing.

Providing a compositeIdSpec for the Fusion collection

Before indexing, you need to provide a `compositeIdSpec` for the Fusion collection. For example:

```
curl -u USERNAME:PASSWORD -X POST -H "Content-type:application/json" \
-d '{"id":"customer","solrParams":
{"replicationFactor":1,"numShards":1,"maxShardsPerNode":10},"type":"DA
TA","compositeIdSpec":{"routeKey1Field":"customer_id_s"}}' \
"https://FUSION_HOST:6764/apps/APP_NAME/collections?
defaultFeatures=false"
```

In the example request above, we create a collection named `customer` with the route key field set to `customer_id_s`. When documents are indexed through Fusion, the Solr Index pipeline stage uses the `compositeIdSpec` to create a composite document ID, so documents get routed to the correct shard.

Exposing document types as virtual tables

If you configure your Fusion collection to use a route key field to route different document types to the same shard, then the Fusion SQL service can expose each document type as a virtual table and perform optimized joins between these virtual tables using the route key. To create virtual tables, you simply need to use the Fusion Catalog API on the data asset for the main collection to set the name of the field that determines the document type. For example, if you have a collection named customer that contains different document types (contacts, support tickets, sales contracts, and so forth), then you would set up virtual tables using the following Catalog API update request:

```
curl -XPUT -H "Content-type:application/json"
http://<FUSION_HOST>/api/catalog/fusion/assets/customer -d '{
"projectId" : "fusion", "name" : "customer", "assetType" : "table",
"description" : "Fusion collection customer", "format" : "solr",
"options" : [ "collection -> customer", "exclude_fields ->
_lw_*,*_\\d_coordinate,_raw_content_", "solr.params -> sort=id asc" ],
"cacheOnLoad" : false, "id" : "fusion.customer", "additionalSettings":
{ "virtualTableField":"doc_type_s" } }'
```

In the example above, we set the `virtualTableField` to `doc_type_s`. Fusion sends a facet request to the customer collection to get the unique values of the `doc_type_s` field and creates a data asset for each unique value. Each virtual table is registered in the Fusion SQL service as a table.

Performing optimized joins in SQL

After you have virtual tables configured and documents routed to the same shard using a **`compositeIdSpec`**, you can perform optimized joins in SQL that take advantage of Solr's domain-join facet feature. For example, the following SQL statement results in a JSON facet request to Solr to perform the aggregation:

```
select count(1) num_support_requests, c.industry as industry, a.app_id
as app_id, a.feature_id as feature_id from customer c join support s
on c.customer_id = s.customer_id join apps a on s.customer_id =
a.customer_id where c.region='US-East' AND
s.support_type='Enhancement' AND a.app_type='Search' group by
industry, app_id, feature_id
```

In the example above, we compute the number of feature enhancement requests for Search applications from customers in the US-East region by performing a 3-way join between the **`customer`**, **`support`**, and **`apps`** virtual tables using the **`customer_id`** join key. Behind the scenes, Fusion SQL performs a JSON facet query that exploits all documents with the same **`customer_id`** value being in the same shard. This lets Solr compute the **`count`** for the **`industry`**, **`app_id`**, **`feature_id`** group by key more efficiently than is possible using table scans in Spark.

User Access Request Parameters

How-to

> Compatible versions: 4.2, 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion Server*, *Fusion*

Fusion requests must come from a known user, i.e., a user with a unique user id (UUID). Fusion's ZooKeeper registry tracks all users across all realms. Usernames must be unique within a realm. Fusion creates a globally unique user ID for all users based on the combination of username and realm.

All requests to the Fusion REST API require either a username, password, and security realm name, or the session cookie which contains the unique user ID.

Per-Request Authentication

To pass authentication information with each request, the realmName is specified as a query parameter on the request itself:

```
curl –u joe.smith:password123
"http://www.acme.com:8764/api/collections?realmName=acmeLDAP"
```

The default realmName parameter is "native", so for native authentication, this parameter can be omitted.

Session Cookies

The Fusion UI service endpoint "api/session" can be used to generate a session cookie which contains the unique user id via a POST request whose body consists of a JSON object which contains the username, password information. For users belonging to a realm other than the native realm, the request parameter "realmName" must be specified. The command to generate a session cookie for the admin user with password "password123" is:

```
curl \ –c cookie –i –X POST –H "Content–type:application/json" –d @– \
http://localhost:8764/api/session?realmName=native \ <<EOF {
"username" : "admin" , "password" : "password123" } EOF
```

The curl command takes any number of specialized arguments, followed by the URL of the request endpoint. The arguments used here are:

- `–c` : filename of cookies file. If it exists, cookies are added to it. You can use `–c –` which writes to the terminal window (std out).
- `–i` : include the HTTP-header in the output. Used here to see the cookie returned with the response.
- `–X` : request method, in this case `POST`
- `–H` : request header. The `api/session` endpoint requires `Content–type:application/json`.
- `–d` : Pass POST body as part of the command-line request. To get ready the body from a file, use the syntax `–d @<filename>`. The argument `–d @–` reads the data from stdin.

The header output shows the cookie information:

```
HTTP/1.1 201 Created Set–Cookie: id=996e4adf–bd04–4058–a926–
8ea8ca08c05a;Secure;HttpOnly;Path=/api Content–Length: 0 Server:
Jetty(9.2.11.v20150529)
```

Once the session cookie file has been created, it can be sent along in all subsequent requests to the REST API. For the curl command-line client, the `-b` flag is used to send the contents of the cookie file to the server along with the request.

The following command sends a GET request to the Fusion REST API Collections service to check the status of the `system_logs` collection. The `-b` flag sends in a freshly generated session cookie.

```
> curl -b cookie -i http://localhost:8764/api/collections/system_logs
HTTP/1.1 200 OK Content-Type: application/json; charset=utf-8 Content-
Encoding: gzip Vary: Accept-Encoding, User-Agent Content-Length: 278
Server: Jetty(9.2.11.v20150529) { "id" : "system_logs", "createdAt" :
"2016-03-04T23:29:47.779Z", "searchClusterId" : "default",
"commitWithin" : 10000, "solrParams" : { "name" : "system_logs",
"numShards" : 1, "replicationFactor" : 1 }, "type" : "METRICS",
"metadata" : { } }
```

If the session cookie has expired, the system returns a 401 Unauthorized code:

```
> curl -b cookie -i http://localhost:8764/api/collections/system_logs
HTTP/1.1 401 Unauthorized Content-Type: application/json; charset=utf-
8 Content-Length: 31 Server: Jetty(9.2.11.v20150529) {"code":"session-
idle-timeout"}
```

Using Java Processors

How-to

> Compatible versions: 4.2
>
> Works with: *App Studio*

The App Studio Workflow module provides out of the box processors to manipulate queries and responses from the underlying search engine, as well as an SDK to write your own Java processors.

Implementing processors

You need to make some configuration changes to set up processors on your platforms. There are two ways. The simplest is directly on the platform config file, when the processor requires no arguments. For more complex processors that take input arguments there are additional steps.

Query Processor

Example: Add the user name to the query filtering on the 'username' field. Get the username either from the query payload itself or from the security provider.

src/main/conf/platforms/fusion/data.conf

```
query-profile: my-profile workflow:
twigkit.client.processor.query.AddUser ...
```

src/main/java/twigkit/client/processor/query/AddUser.java

```
package twigkit.client.processor.query; import
com.google.inject.Inject; import org.slf4j.Logger; import
org.slf4j.LoggerFactory; import twigkit.model.Filter; import
twigkit.model.Query; import twigkit.model.Value; import
twigkit.processor.QueryProcessor; import
twigkit.security.SecurityProvider; public class AddUser extends
QueryProcessor { private static final Logger logger =
LoggerFactory.getLogger(AddUser.class); @Inject private
SecurityProvider securityProvider; @Override public Query change(Query
payload) { if (payload != null) { Filter userFilter = new Filter();
userFilter.setField("user"); Value user = new Value(); if
(securityProvider != null) { if (securityProvider.getUser() != null) {
user.setActual(securityProvider.getUser().getName());
userFilter.setValue(user); payload.addFilter(userFilter); } else {
logger.error("No user retrieved from " +
securityProvider.getClass().getName()); } } else if (payload.getUser()
!= null) { user.setActual(payload.getUser().getName());
payload.addFilter(userFilter); } else { logger.error("No
SecurityProvider provided and no user on the query!"); } } return
payload; } }
```

Response processor with configuration

Example: If the response has no results send the query again to a different platform.

src/main/conf/platforms/fusion/fusion.conf

```
requestHandler: /select query-profile: my-profile workflow:
processors.response.fuzzy-search
```

src/main/conf/platforms/fusion/fuzzy.conf

```
requestHandler: /spell
```

src/main/conf/processors/response/fuzzy-search.conf

```
name: twigkit.client.processor.response.FuzzySearch fuzzy-platform:
platforms.fusion.fuzzy
```

src/main/java/twigkit/client/processor/response/FuzzySearch.java

```
package twigkit.client.processor.response; import
com.google.inject.Inject; import org.slf4j.Logger; import
org.slf4j.LoggerFactory; import
twigkit.conf.ConfiguredPlatformProvider; import twigkit.model.Query;
import twigkit.model.Response; import twigkit.model.messages.Message;
import twigkit.platform.Platform; import
twigkit.processor.ResponseProcessor; import java.util.ArrayList;
import java.util.List; public class FuzzySearch extends
ResponseProcessor { private static final Logger logger =
LoggerFactory.getLogger(FuzzySearch.class); @Inject private
ConfiguredPlatformProvider platformProvider; @Override public Response
change(Response response) { String fuzzyPlatformId =
getParameterStringValue("fuzzy-platform"); Platform fuzzyPlatform =
platformProvider.get(fuzzyPlatformId); if
(response.getHits().getActual() > 0){ return response; } else { Query
query = response.getQuery(); Response fuzzyResponse =
fuzzyPlatform.search(query); String queryString =
query.getValue().getActualAsString(); Message fuzzyMessage = new
Message(); fuzzyMessage.setType("fuzzy-results");
fuzzyMessage.setValue("No results found for the term
<em>"+queryString+"</em>. Returning results with similar words.");
List<Message> messages = new ArrayList<Message>();
messages.add(fuzzyMessage); fuzzyResponse.setMessages(messages);
return fuzzyResponse; } } }
```

Using the Appkit Tasks Module

How-to

> Compatible versions: 4.2, 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *App Studio*

The Task module lets you run numerous, time-consuming tasks (atomic units of work) behind the scenes, while continuing to use your application. While your tasks are running, you can monitor their status as well as stop, restart, and cancel them if needed.

Setup

Updating the pom.xml

To add the Task module to a Maven project, add this dependency to your project's `pom.xml` file:

```
<dependency> <groupId>twigkit</groupId>
<artifactId>twigkit.task</artifactId>
<version>${project.parent.version}</version> </dependency>
```

Configuring the task executor

The Task module includes its own task executor that takes charge of managing your tasks. Under the hood, the executor uses a registry that can be configured in order to fine-tune how your tasks will be handled.

To customize the configuration of the task executor, first create a file named `executor.conf` and place this in `src/main/resources/conf/services/task/`. Within that file, these parameters can be set:

`max-tasks-in-memory`: the maximum number of tasks the registry can hold in memory at any one time. The default is 100.

`max-tasks-on-disk`: the maximum number of tasks that can be persisted on disk. The default is 10000000.

`disk-store-location`: the location for storing/persisting tasks. The default is `java.io.tmpdir` -a JVM system property that will change depending on your Operating System.

`time-to-idle`: the amount of time in seconds a task is allowed to be left idle (from last access or modified date) before being removed from the registry. The default is 3600.

`time-to-live`: the amount of time in seconds a task is allowed to exist (from its creation date) before being removed from the registry regardless of how often it is used. The default is effectively forever.

`overflow-to-disk`: whether additional tasks will be stored on disk, if the in-memory registry is full. The default is true.

`persist-to-disk`: whether tasks will be persisted to disk, for example, if the if in-memory registry is full or the application shuts down. The default is true.

As noted, all of the above parameters already contain default values that will be used in case `executor.conf` cannot be found or one or other of the parameters is not defined.

Writing your own tasks

To submit a task to the task executor, the actual actions of the task (what it will do when it starts, stops, and is deleted) must first be defined. A task is written in Java and must follow a prescribed template. This template is as follows:

```
@XmlRootElement public class MyTask extends Task { public MyTask() {
this(null); } public MyTask(User owner) { super(owner); } public
MyTask(User owner, Map<String, String> attributes) { super(owner,
attributes); } @Override public void start() throws TaskException { //
Put here what the task will do when started, for example, download a
file } @Override public void start(Callback<?> callback) throws
TaskException { // This is a work in progress; Do not use. } @Override
public void stop() throws TaskException { // Put here what the task
will do when stopped } @Override public void delete() throws
TaskException { // Put here what the task will do when deleted } }
```

As can be seen from the template, three methods are available that can be used to decide what the task will do when it is started (`start`), when it is stopped (`stop`), and when it is deleted (`delete`) from the registry. These methods are named when particular RESTful endpoints are hit as discussed below.

Configuring your own tasks

After you have written your task (see above), the next step is to create a configuration file for that task. This will let you submit the task to the Task Web Service and define specific properties of the task that will be available to the task when it is started, stopped etc.

As with any other configuration, the task configuration must be stored in `resources/conf` , for example, `resources/conf/tasks/download.conf` . The simplest configuration for a task would be one with just the `name` parameter:

```
name: package1.package2.package3.MyTask
```

Where `name` is the fully-qualified class name of the Java class you have written to define your `Task` .

The configuration can also include a display name, as well as any other attributes that the task might need, for example:

```
display: Download destination: /downloads file-type: pdf zipped: true
```

These additional attributes can be accessed within the task via `getAttributes()` .

Usage

Working with tasks via the Appkit Task RESTful endpoint

The Task web service provides a number of RESTful endpoints that can be used to submit, query, and cancel your tasks. These endpoints are as follows:

`GET /twigkit/api/tasks` : this will get a list of tasks owned by the current user.

`GET /twigkit/api/tasks/{id}` : this will get any information on the task with the given ID, including status and any task-specific attributes.

`GET /twigkit/api/tasks/{id}/status` this will get the status of the task with the given ID.

`POST /twigkit/api/tasks/{task}` : this will submit a task with the given configuration task name to the task executor. The given task name is the configuration name for a task. For example, if the configuration for a particular task was stored in `resources/conf/tasks/download.conf`, the task name passed to the submit endpoint would be `tasks.download`.

`POST /twigkit/api/tasks/{id}/stop` : this will stop the task with the given ID.

`POST /twigkit/api/tasks/{id}/restart` : this will restart the task with the given ID

`DELETE /twigkit/api/tasks/{id}` : this will cancel the task with the given ID. This will call `stop` and `delete` on the task as well as remove the task from the registry.

Task persistence

By default, while an application is running, tasks are periodically flushed to disk. For this purpose, in the `disk-store-location` (see above) a `task-cache.data` file is created to keep track of tasks until the application shuts down when an additional file, `task-cache.index`, is created to persist tasks. Next time the application is run these tasks will be available for continued use.

To disable task persistence, set the configuration parameters `overflow-to-disk` and `persist-to-disk` to `false`.

Task expiration

By default, tasks will not be stored in the registry indefinitely. To customize how long tasks may be stored, two configuration options are available:

- `time-to-live`
  This is the amount of time in seconds a task is allowed to exist before being removed from the registry regardless of how often it is used. The default is effectively forever. This is useful in case resources are limited and it is not clear whether tasks will be used after a finite amount of time.
- `time-to-idle`
  This is the amount of time in seconds a task is allowed to be left idle before being removed from the registry. The default is one hour.

> **Important**
>
> Whatever times are set for these values, they are never reset. This holds true even when the application is not running. If the application is shut down and enough time elapses to allow the task to expire, on restart, the task will no longer be available for use.

To disable task expiration, set `time-to-live` and `time-to-idle` to 0.

DELETING A TASK AFTER EXPIRATION

After a task has expired, it is not only deleted from the registry, but the `delete` method implemented within your task will be named. If there is any final action that you would like your task to perform before it is removed then you can implement that action in the tasks `delete` method.

Task eviction

By default, tasks are stored on disk if the capacity to store tasks in-memory is exceeded. In this case, if the number of tasks being stored exceeds the limit set by the configuration parameter `max-tasks-on-disk` then the least recently used task will be evicted from the registry. The default maximum number of tasks that can be stored on disk is 10000000.

If instead you decide not to store tasks on disk (both `overflow-to-disk` and `persist-to-disk` are `false`) but would like to limit the number of tasks that can be stored in-memory, then the configuration parameter `max-tasks-in-memory` can be set. This will evict the least recently used task once capacity is reached.

DELETING A TASK AFTER EVICTION

After a task has been evicted, it is not only deleted from the registry, but the `delete` method implemented within your task will be named. If there is any final action that you would like your task to perform before it is removed then you can implement that action in the tasks `delete` method.

Validate Your SharePoint Online Crawl Account Permissions Using PowerShell

How-to

> Compatible versions: 4.2
>
> Works with: *Fusion Connectors*, *Fusion Server*

Overview of procedure

How to set up and validate SharePoint Online crawl account permissions:

1. *Authenticate to SharePoint Online.*
2. *Get the FedAuth and rtFa cookie values.*
3. *Add your information to the PowerShell script.*
4. *Add your URL and DigestValue.*

Authenticate to SharePoint Online

Authenticate to SharePoint Online as the service account user.

Get the FedAuth and rtFa cookie values

Open up your Debugging Tools (such as DevTools from Chrome) and copy the `FedAuth` and `rtFa` cookies from session.

Add your information to the PowerShell script

Enter your site and these two cookies into this PowerShell script:

```
# change these 3 variables to match your environment $sharepointUrl =
"https://YourSharePointURL.com" $fedAuthCookieValue =
"EnterYourFedAuthCookieValueHere" $rtFaValue =
"EnterYourRtfaCookieValueHere" $headers = New-Object
"System.Collections.Generic.Dictionary[[String],[String]]"
$headers.Add("Content-Type", "text/xml") $headers.Add("SOAPAction",
"http://schemas.microsoft.com/sharepoint/soap/GetUpdatedFormDigestInfo
rmation") $headers.Add("X-RequestForceAuthentication", "true")
$headers.Add("Cookie", "FedAuth=$fedAuthCookieValue;rtFa=$rtFaValue")
$body = "<?xml version=`"1.0`" encoding=`"utf-8`"?>`n<soap:Envelope
xmlns:xsi=`"http://www.w3.org/2001/XMLSchema-instance`"
xmlns:xsd=`"http://www.w3.org/2001/XMLSchema`"
xmlns:soap=`"http://schemas.xmlsoap.org/soap/envelope/`">`n
<soap:Body>`n <GetUpdatedFormDigestInformation
xmlns=`"http://schemas.microsoft.com/sharepoint/soap/`" />`n
</soap:Body>`n</soap:Envelope>" $response = Invoke-RestMethod
"$sharepointUrl/_vti_bin/sites.asmx" -Method 'POST' -Headers $headers
-Body $body Write-Host "$response"
```

The response will resemble:

```
<?xml version="1.0" encoding="utf-8"?> <soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"> <soap:Body>
<GetUpdatedFormDigestInformationResponse
xmlns="http://schemas.microsoft.com/sharepoint/soap/">
<GetUpdatedFormDigestInformationResult> *
<DigestValue>YourDigestValueWithTimestamp</DigestValue>*
<TimeoutSeconds>1800</TimeoutSeconds>
<WebFullUrl>https://lucidworksfusiondev.sharepoint.com/sites/ndd2</Web
FullUrl> <LibraryVersion>16.0.19527.12083</LibraryVersion>
<SupportedSchemaVersions>14.0.0.0,15.0.0.0</SupportedSchemaVersions>
</GetUpdatedFormDigestInformationResult>
</GetUpdatedFormDigestInformationResponse> </soap:Body>
</soap:Envelope>
```

Add your URL and DigestValue

Copy the `DigestValue` from this response into the PowerShell script below. Also change the SharePoint URL to match your environment.

```
# change these 2 variables to match your environment $sharepointUrl =
"https://YourSharePointURL.com" $digestValue =
"YourDigestValueWithTimestamp" $headers = New-Object
"System.Collections.Generic.Dictionary[[String],[String]]"
$headers.Add("Content-Type", "text/xml") $headers.Add("X-
RequestForceAuthentication", "true") $headers.Add("X-RequestDigest",
"$digestValue") $body = "<Request AddExpandoFieldTypeSuffix=`"true`"
SchemaVersion=`"14.0.0.0`" LibraryVersion=`"16.0.0.0`"
ApplicationName=`".NET Library`"
xmlns=`"http://schemas.microsoft.com/sharepoint/clientquery/2009`">
<Actions><ObjectPath Id=`"2`" ObjectPathId=`"1`"/><ObjectPath Id=`"4`"
ObjectPathId=`"3`"/><Query Id=`"5`" ObjectPathId=`"3`"><Query
SelectAllProperties=`"false`"><Properties><Property Name=`"Webs`"
SelectAll=`"true`"><Query SelectAllProperties=`"false`"><Properties/>
</Query></Property><Property Name=`"Title`" ScalarProperty=`"true`"/>
<Property Name=`"ServerRelativeUrl`" ScalarProperty=`"true`"/>
<Property Name=`"RoleDefinitions`" SelectAll=`"true`"><Query
SelectAllProperties=`"false`"><Properties/></Query></Property>
<Property Name=`"RoleAssignments`" SelectAll=`"true`"><Query
SelectAllProperties=`"false`"><Properties/></Query></Property>
<Property Name=`"HasUniqueRoleAssignments`" ScalarProperty=`"true`"/>
<Property Name=`"Description`" ScalarProperty=`"true`"/><Property
Name=`"Id`" ScalarProperty=`"true`"/><Property
Name=`"LastItemModifiedDate`" ScalarProperty=`"true`"/></Properties>
</Query></Query></Actions><ObjectPaths><StaticProperty Id=`"1`"
TypeId=`"{3747adcd-a3c3-41b9-bfab-4a64dd2f1e0a}`" Name=`"Current`"/>
<Property Id=`"3`" ParentId=`"1`" Name=`"Web`"/></ObjectPaths>
</Request>" $response = Invoke-RestMethod
"$sharepointUrl/_vti_bin/client.svc/ProcessQuery" –Method 'POST' –
Headers $headers –Body $body Write-Host "$response"
```

If permissions are **incorrect**, you will get an error message.

```
[ { "SchemaVersion": "14.0.0.0", "LibraryVersion": "16.0.19527.12083",
"ErrorInfo": { "ErrorMessage": "Access denied. You do not have
permission to perform this action or access this resource.",
"ErrorValue": null, "TraceCorrelationId": "640d309f-b0f7-b000-8100-
49826f3674cb", "ErrorCode": -2147024891, "ErrorTypeName":
"System.UnauthorizedAccessException" }, "TraceCorrelationId":
"640d309f-b0f7-b000-8100-49826f3674cb" } ]
```


If permissions are **correct**, you will get a response that resembles:

[{ "SchemaVersion": "14.0.0.0", "LibraryVersion": "16.0.19527.12083", "ErrorInfo": null, "TraceCorrelationId": "e70c309f-3097-b000-4931-8fe95320fdb0" }, 2, { "IsNull": false }, 4, { "IsNull": false }, 5, { "_ObjectType_": "SP.Web", "_ObjectIdentity_": "e70c309f-3097-b000-4931-8fe95320fdb0|740c6a0b-85e2-48a0-a494-e0f1759d4aa7:site:d714bad3-ef71-4f83-86b6-25405106b719:web:ab8c2f1e-c518-4b2a-a89f-c6404b97a85a", "Webs": { "_ObjectType_": "SP.WebCollection", "_Child_Items_": [] }, "Title": "NddTest", "ServerRelativeUrl": "/sites/ndd2", "RoleDefinitions": { "_ObjectType_": "SP.RoleDefinitionCollection", "_Child_Items_": [{ "_ObjectType_": "SP.RoleDefinition", "_ObjectIdentity_": "e70c309f-3097-b000-4931-8fe95320fdb0|740c6a0b-85e2-48a0-a494-e0f1759d4aa7:site:d714bad3-ef71-4f83-86b6-25405106b719:web:ab8c2f1e-c518-4b2a-a89f-c6404b97a85a:rd:1073741829", "BasePermissions": { "_ObjectType_": "SP.BasePermissions", "High": 2147483647, "Low": 4294967295 }, "Description": "Has full control.", "Hidden": false, "Id": 1073741829, "Name": "Full Control", "Order": 1, "RoleTypeKind": 5 }, { "_ObjectType_": "SP.RoleDefinition", "_ObjectIdentity_": "e70c309f-3097-b000-4931-8fe95320fdb0|740c6a0b-85e2-48a0-a494-e0f1759d4aa7:site:d714bad3-ef71-4f83-86b6-25405106b719:web:ab8c2f1e-c518-4b2a-a89f-c6404b97a85a:rd:1073741828", "BasePermissions": { "_ObjectType_": "SP.BasePermissions", "High": 432, "Low": 1012866047 }, "Description": "Can view, add, update, delete, approve, and customize.", "Hidden": false, "Id": 1073741828, "Name": "Design", "Order": 32, "RoleTypeKind": 4 }, { "_ObjectType_": "SP.RoleDefinition", "_ObjectIdentity_": "e70c309f-3097-b000-4931-8fe95320fdb0|740c6a0b-85e2-48a0-a494-e0f1759d4aa7:site:d714bad3-ef71-4f83-86b6-25405106b719:web:ab8c2f1e-c518-4b2a-a89f-c6404b97a85a:rd:1073741830", "BasePermissions": { "_ObjectType_": "SP.BasePermissions", "High": 432, "Low": 1011030767 }, "Description": "Can add, edit and delete lists; can view, add, update and delete list items and documents.", "Hidden": false, "Id": 1073741830, "Name": "Edit", "Order": 48, "RoleTypeKind": 6 }, { "_ObjectType_": "SP.RoleDefinition", "_ObjectIdentity_": "e70c309f-3097-b000-4931-

8fe95320fdb0|740c6a0b-85e2-48a0-a494-e0f1759d4aa7:site:d714bad3-ef71-4f83-86b6-25405106b719:web:ab8c2f1e-c518-4b2a-a89f-c6404b97a85a:rd:1073741827", "BasePermissions": { "_ObjectType_": "SP.BasePermissions", "High": 432, "Low": 1011028719 }, "Description": "Can view, add, update, and delete list items and documents.", "Hidden": false, "Id": 1073741827, "Name": "Contribute", "Order": 64, "RoleTypeKind": 3 }, { "_ObjectType_": "SP.RoleDefinition", "_ObjectIdentity_": "e70c309f-3097-b000-4931-8fe95320fdb0|740c6a0b-85e2-48a0-a494-e0f1759d4aa7:site:d714bad3-ef71-4f83-86b6-25405106b719:web:ab8c2f1e-c518-4b2a-a89f-c6404b97a85a:rd:1073741826", "BasePermissions": { "_ObjectType_": "SP.BasePermissions", "High": 176, "Low": 138612833 }, "Description": "Can view pages and list items and download documents.", "Hidden": false, "Id": 1073741826, "Name": "Read", "Order": 128, "RoleTypeKind": 2 }, { "_ObjectType_": "SP.RoleDefinition", "_ObjectIdentity_": "e70c309f-3097-b000-4931-8fe95320fdb0|740c6a0b-85e2-48a0-a494-e0f1759d4aa7:site:d714bad3-ef71-4f83-86b6-25405106b719:web:ab8c2f1e-c518-4b2a-a89f-c6404b97a85a:rd:1073741825", "BasePermissions": { "_ObjectType_": "SP.BasePermissions", "High": 48, "Low": 134287360 }, "Description": "Can view specific lists, document libraries, list items, folders, or documents when given permissions.", "Hidden": true, "Id": 1073741825, "Name": "Limited Access", "Order": 160, "RoleTypeKind": 1 }, { "_ObjectType_": "SP.RoleDefinition", "_ObjectIdentity_": "e70c309f-3097-b000-4931-8fe95320fdb0|740c6a0b-85e2-48a0-a494-e0f1759d4aa7:site:d714bad3-ef71-4f83-86b6-25405106b719:web:ab8c2f1e-c518-4b2a-a89f-c6404b97a85a:rd:1073741924", "BasePermissions": { "_ObjectType_": "SP.BasePermissions", "High": 176, "Low": 138612801 }, "Description": "Can view pages, list items, and documents. Document types with server-side file handlers can be viewed in the browser but not downloaded.", "Hidden": false, "Id": 1073741924, "Name": "View Only", "Order": 2147483647, "RoleTypeKind": 0 }, { "_ObjectType_": "SP.RoleDefinition", "_ObjectIdentity_": "e70c309f-3097-b000-4931-8fe95320fdb0|740c6a0b-85e2-48a0-a494-e0f1759d4aa7:site:d714bad3-ef71-4f83-86b6-25405106b719:web:ab8c2f1e-c518-4b2a-a89f-

```
c6404b97a85a:rd:1073741925", "BasePermissions": { "_ObjectType_":
"SP.BasePermissions", "High": 1073741856, "Low": 203624545 },
"Description": "", "Hidden": false, "Id": 1073741925, "Name": "LW
Fusion", "Order": 2147483647, "RoleTypeKind": 0 } ] },
"RoleAssignments": { "_ObjectType_": "SP.RoleAssignmentCollection",
"_Child_Items_": [ { "_ObjectType_": "SP.RoleAssignment",
"_ObjectIdentity_": "e70c309f-3097-b000-4931-8fe95320fdb0|740c6a0b-
85e2-48a0-a494-e0f1759d4aa7:site:d714bad3-ef71-4f83-86b6-
25405106b719:web:ab8c2f1e-c518-4b2a-a89f-c6404b97a85a:ra:3",
"PrincipalId": 3 }, { "_ObjectType_": "SP.RoleAssignment",
"_ObjectIdentity_": "e70c309f-3097-b000-4931-8fe95320fdb0|740c6a0b-
85e2-48a0-a494-e0f1759d4aa7:site:d714bad3-ef71-4f83-86b6-
25405106b719:web:ab8c2f1e-c518-4b2a-a89f-c6404b97a85a:ra:6",
"PrincipalId": 6 }, { "_ObjectType_": "SP.RoleAssignment",
"_ObjectIdentity_": "e70c309f-3097-b000-4931-8fe95320fdb0|740c6a0b-
85e2-48a0-a494-e0f1759d4aa7:site:d714bad3-ef71-4f83-86b6-
25405106b719:web:ab8c2f1e-c518-4b2a-a89f-c6404b97a85a:ra:7",
"PrincipalId": 7 }, { "_ObjectType_": "SP.RoleAssignment",
"_ObjectIdentity_": "e70c309f-3097-b000-4931-8fe95320fdb0|740c6a0b-
85e2-48a0-a494-e0f1759d4aa7:site:d714bad3-ef71-4f83-86b6-
25405106b719:web:ab8c2f1e-c518-4b2a-a89f-c6404b97a85a:ra:8",
"PrincipalId": 8 }, { "_ObjectType_": "SP.RoleAssignment",
"_ObjectIdentity_": "e70c309f-3097-b000-4931-8fe95320fdb0|740c6a0b-
85e2-48a0-a494-e0f1759d4aa7:site:d714bad3-ef71-4f83-86b6-
25405106b719:web:ab8c2f1e-c518-4b2a-a89f-c6404b97a85a:ra:14",
"PrincipalId": 14 } ] }, "HasUniqueRoleAssignments": true,
"Description": "", "Id": "/Guid(ab8c2f1e-c518-4b2a-a89f-
c6404b97a85a)/", "LastItemModifiedDate": "/Date(1580251756000)/" } ]
```

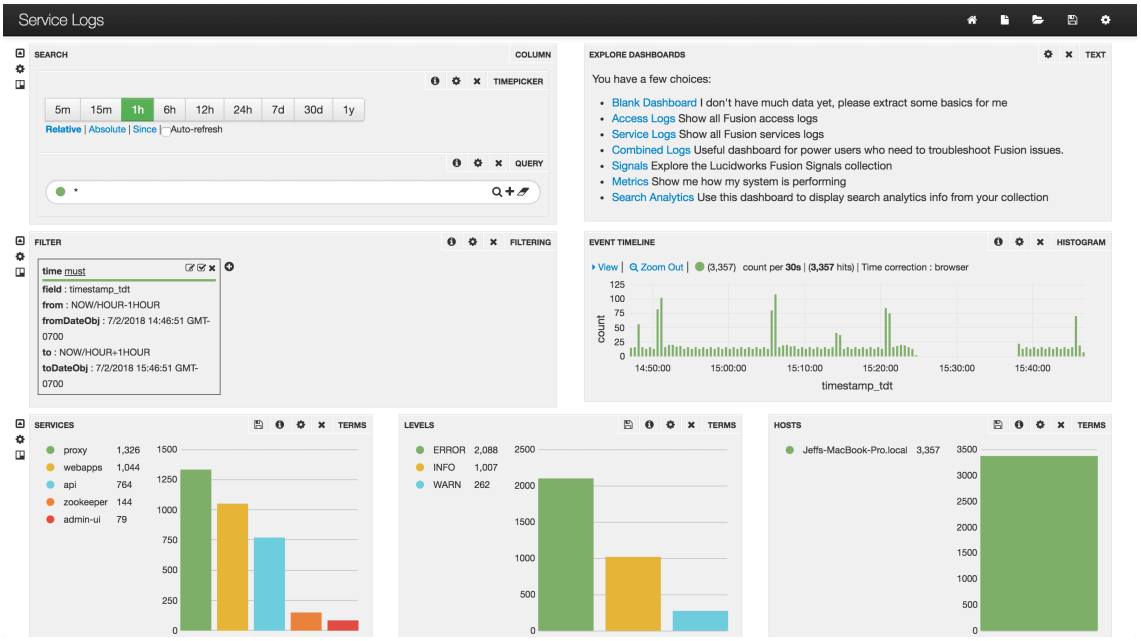
View and Analyze Logs in Fusion 4.1 and 4.2

How-to

> Compatible versions: 4.2
>
> Works with: *Fusion Server*

Fusion has several features that make analysis of log files easier:

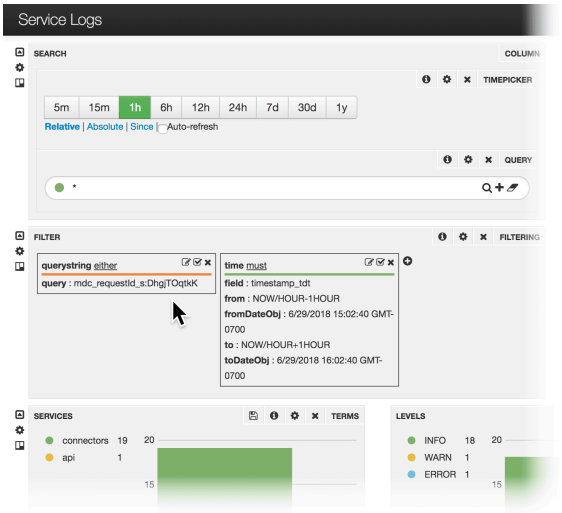
- **View log file dashboards.** The *Service Logs*, *Access Logs*, and *Combined Logs* dashboards provide graphical user interfaces for viewing and analyzing log files.



To open the default dashboard from the Fusion workspace, click Analytics > **Dashboards** or

System > **Log Viewer**.

- **Assign Fusion request IDs.** To make it easier to follow requests through the Fusion system, you can *assign Fusion request IDs*. If you do not, Fusion assigns request IDs automatically.
- **Filter log file dashboards by the Fusion request ID.** In the log file dashboards, you can *filter by Fusion request ID*.



- *Click through from API errors in the Fusion UI* to the Service Logs dashboard filtered by the Fusion request ID of the request that resulted in the error.

View logs in Self-Hosted Fusion 5.6 and later

How-to

> Compatible versions: 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion*

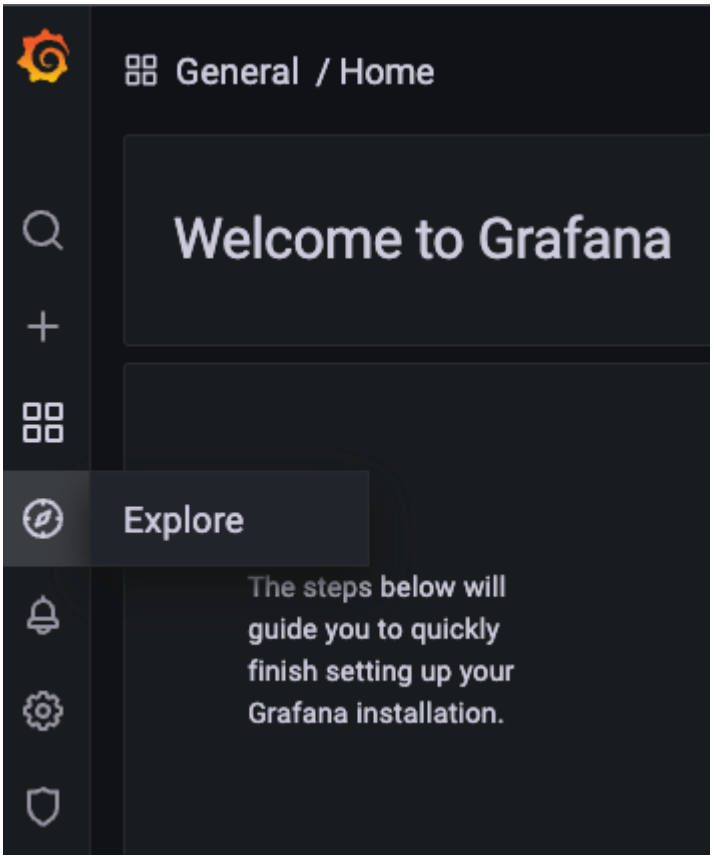
Prerequisites

Before you view logs, complete the procedures in *Configure Grafana, Prometheus, Promtail, and Loki in Fusion*.

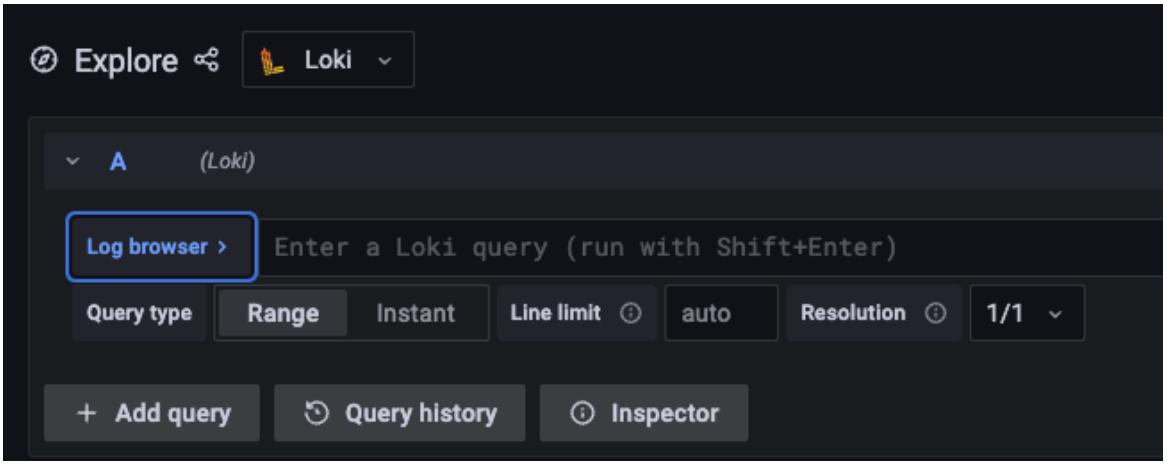
View logs

Use the following steps to view the logs.

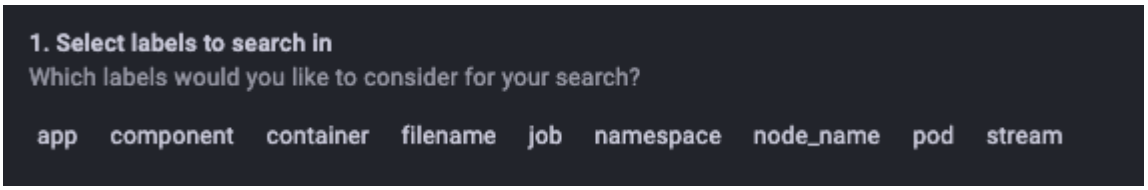
1. Log in to Grafana.
2. Select **Explore** from the icon menu.

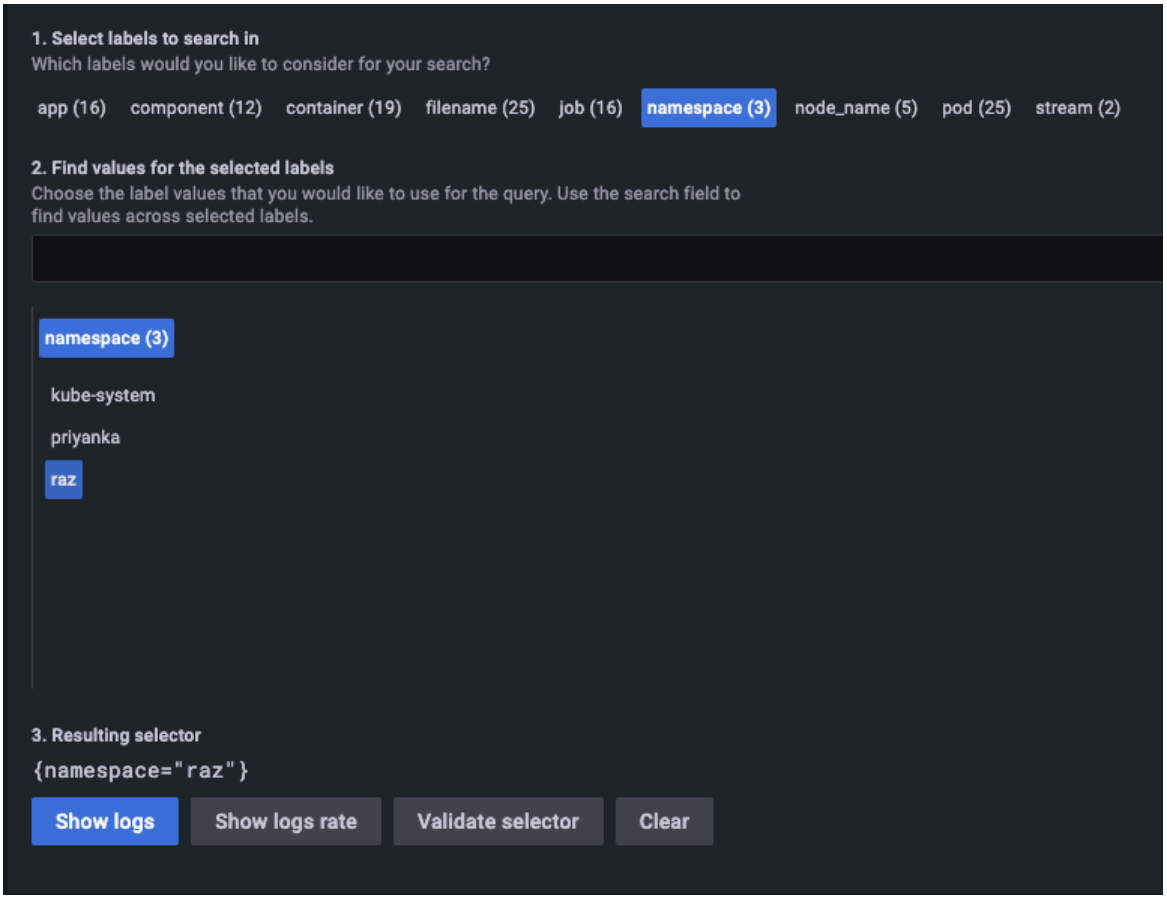


3. Click on **Log Browser** to display the advanced query options.

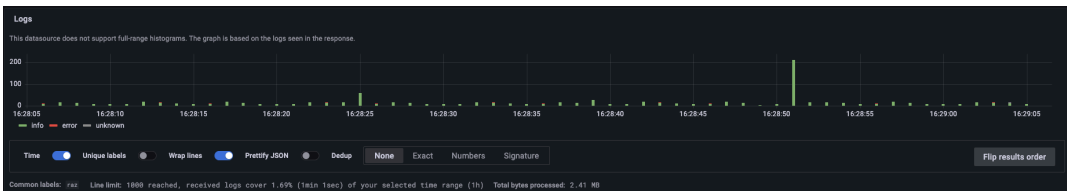


4. Use the **Select labels to search in** options to display the desired filtering criteria.

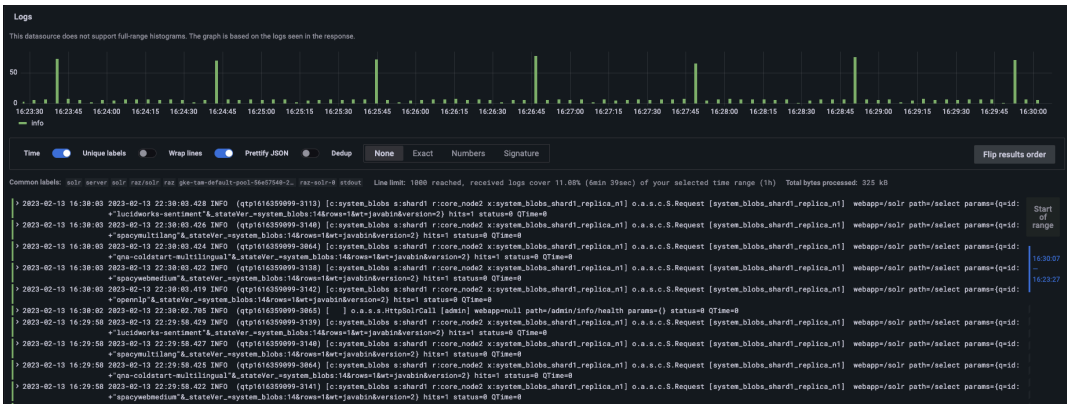




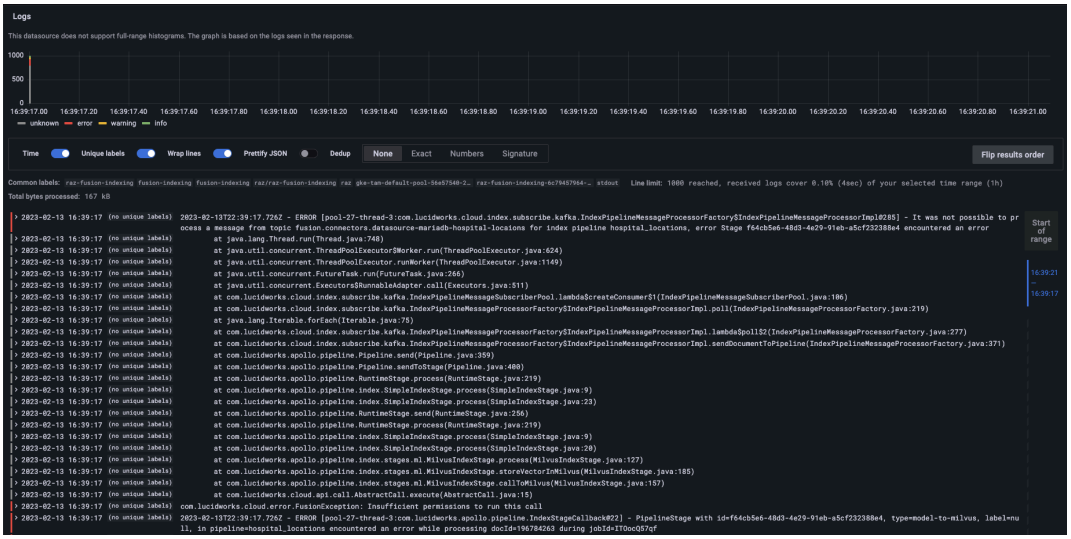
5. Click **Show logs**.
   a. A histogram shows for the timeline.



   b. You also see logs.



c. If there were any errors, the stack trace shows.

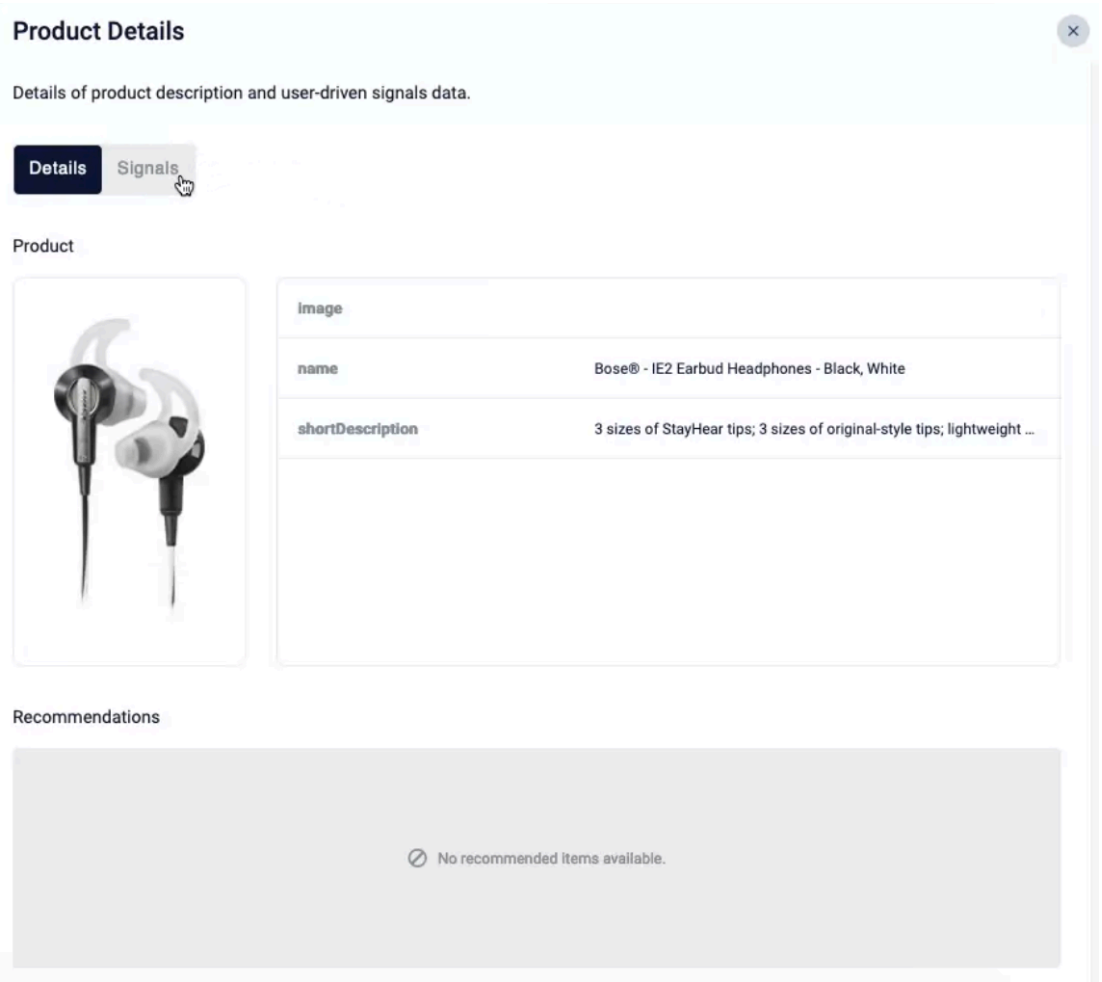


View Product Details in Predictive Merchandiser

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Managed Fusion*

1. Hover your cursor over a product. A menu overlay will appear.
2. Hover your cursor over the **Information** button . The Product Details panel will appear.
3. Use the **Details** and **Signals** buttons to toggle between panes.

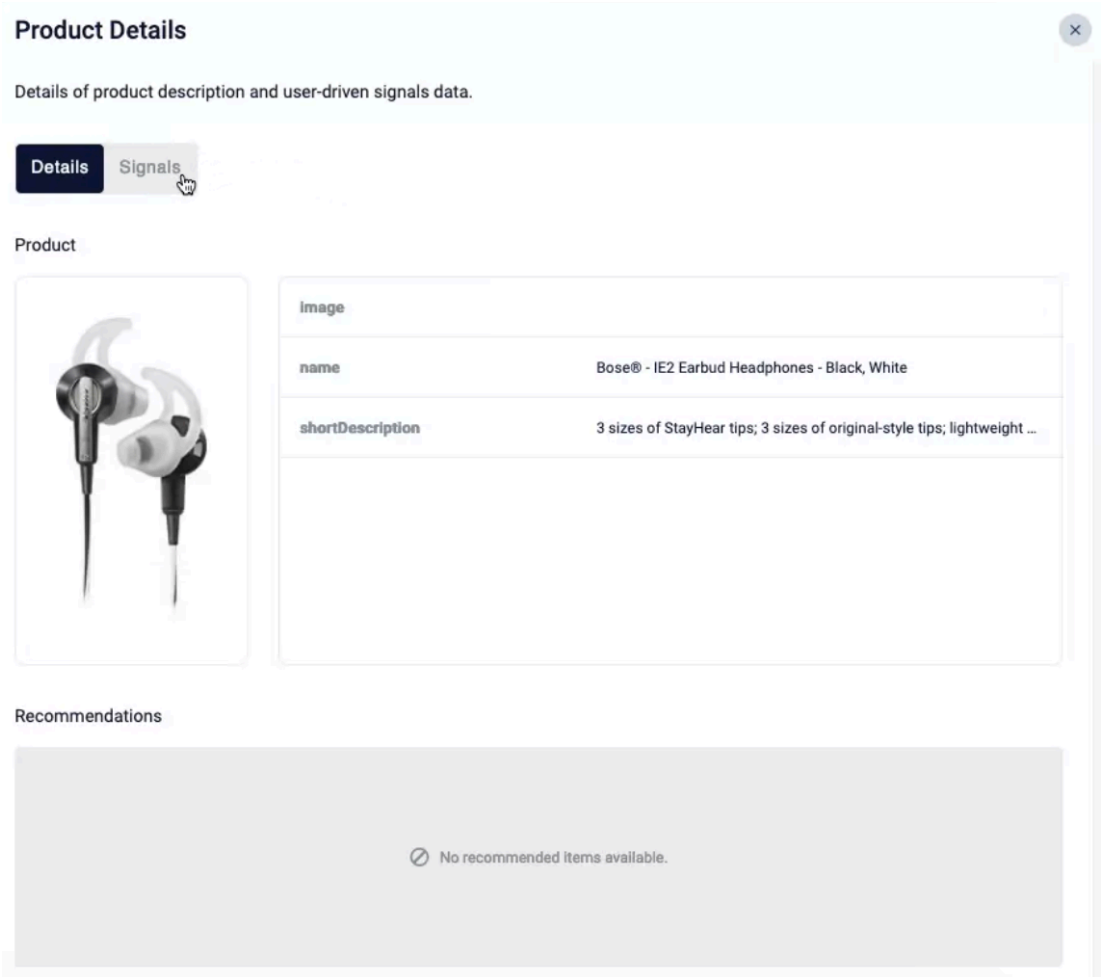


View Product Details in Predictive Merchandiser

How-to

> Compatible versions: 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion*

1. Hover your cursor over a product. A menu overlay will appear.
2. Hover your cursor over the **Information** button . The Product Details panel will appear.
3. Use the **Details** and **Signals** buttons to toggle between panes.



View the SharePoint Export Database File

How-to

> Compatible versions: 4.2, 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion Connectors*, *Fusion Server*, *Fusion*

The SharePoint connector retrieves content and metadata from an on-premises SharePoint repository.

When a crawl is performed with the V1 Optimized connector, a SharePoint export database file is created. This file contains various metadata related to the SharePoint data. It does not store file contents from the SharePoint data.

A Java web viewer, **`sharepoint-exporter.jar`**, is included to browse the export database file. The web viewer is located in the following directory:

```
${FUSION_HOME}/apps/connectors/connectors-classic/plugins/{viewer-directory}/assets/sharepoint-exporter/sharepoint-exporter.jar
```

For example, the:

- SharePoint optimized connector export viewer utility is located at: **`${FUSION_HOME}/apps/connectors/connectors-classic/plugins/lucidworks.sharepoint-optimized/assets/sharepoint-exporter/sharepoint-exporter.jar`**
- SharePoint online optimized connector export viewer utility is located at: **`${FUSION_HOME}/apps/connectors/connectors-classic/plugins/lucidworks.sharepoint-online-optimized/assets/sharepoint-exporter/sharepoint-exporter.jar`**

The web viewer is launched with the following arguments:

- **`-exportDirectoryPath`** - The full path to the export database file.
- **`-port`** - The port which the web viewer server will run on. If unassigned, a random port will be selected.

```
java -cp /opt/fusion/latest/apps/connectors/connectors-classic/plugins/{viewer-directory}/assets/sharepoint-exporter/sharepoint-exporter.jar com.lucidworks.fusion.connector.plugins.sharepoint.exporter.SharepointExportWeb -port 5000 -exportDirectoryPath /opt/fusion/latest/data/connectors/connectors-classic/{export-directory}/example_spo
```

The files are viewed by navigating the directory with any browser.

Sharepoint export viewer

Export Directory:

/home/[redacted]/.fusion/4.1.0/data/connectors/connectors-classic/lucid.sharepoint/5ff51653be64434d82889d8084307a0a

export details

site-collections (1)

sites (2)

lists (6)

list-items (3)

list-item-attachments (1)

role-definitions (1)

role-assignments (2)

Clear

Web V2 connector OAuth access token configuration

How-to

> Compatible versions: 5.9, 5.10, 5.11, 5.12
>
> Works with: *Fusion Connectors*, *Fusion*

The *Web V2 connector* v1.4.0 added the ability to authenticate using OAuth.

Example with PingFederate

This example uses Client Credentials Grant Type and a basic configuration.

1. Create new app in PingIdentity.
2. Edit the app configuration:
    a. Check **Client Credentials** Grant Type.
    b. Set **Token Endpoint Authentication Method** to `Client Secret Post`.
3. Enable the app by clicking on the toggle button to the right of the app name.
4. Go to the **Configuration** tab and click **Get Access Token**. At this point it will probably show an error about scope. Fix it by adding a scope, such as `testScope`.
5. Click the **Get Access Token** button again under the **Configuration** tab. It should generate and display your access token.
6. Go to the **Configuration** tab and click **URLs**. Copy the Token Endpoint URL for further use.
7. You can view a sample of Client Credentials Grant Type access token request in the PingIdentity documentation. In this example the access token request is:

```
$ curl -X POST "https://auth.pingone.com/ENV_ID/as/token" -d
"grant_type=client_credentials&client_id=CLIENT_ID&client_secret
=CLIENT_SECRET&scope=testScope" --header "Content-
Type:application/x-www-form-urlencoded" { "access_token" :
"ACCESS_TOKEN", "token_type" : "Bearer", "expires_in" : 3600,
"scope" : "testScope" }
```

8. In Fusion, go to **CRAWL AUTHENTICATION PROPERTIES** in the datasource configuration and fill in the **ACCESS TOKEN** section by adding the URL and parameters of the request. Replace CLIENT_ID with your own client ID.
    - ACCESS TOKEN REQUEST URL: **https://auth.pingone.com/ENV_ID/as/token**
    - PARAMETER NAME | PARAMETER VALUE: `grant_type` | `client_credentials`
    - PARAMETER NAME | PARAMETER VALUE: `client_id` | CLIENT_ID
    - PARAMETER NAME | PARAMETER VALUE: `scope` | `testScope`

Example with Microsoft Azure AD

1. In Microsoft Entra ID, go to **App Registrations** and register a new app.
2. Go to your app and click **Certificates & secrets** and add a client secret.
3. Copy the client secret **Value** for further use.
4. View the Microsoft documentation for a request to get access token. In this example, we use the client credentials grant type:

```
$ curl -X POST
"https://login.microsoftonline.com/TENANT_ID/oauth2/v2.0/token"
-d
"client_id=CLIENT_ID&scope=https%3A%2F%2Fgraph.microsoft.com%2F.
default&client_secret=CLIENT_SECRET&grant_type=client_credential
s" -H "Content-Type: application/x-www-form-urlencoded"
{"token_type":"Bearer","expires_in":3599,"ext_expires_in":3599,"
access_token":"ACCESS_TOKEN"}
```

5. Copy the endpoint URL and the parameters to the **CRAWL AUTHENTICATION PROPERTIES > ACCESS TOKEN** section in the datasource configuration in Fusion. Replace CLIENT_ID with your own client ID.
    - ACCESS TOKEN REQUEST URL: **`https://login.microsoftonline.com/TENANT_ID/oauth2/v2.0/token`**
    - PARAMETER NAME | PARAMETER VALUE: **`grant_type`** | **`client_credentials`**
    - PARAMETER NAME | PARAMETER VALUE: **`client_id`** | CLIENT_ID
    - PARAMETER NAME | PARAMETER VALUE: **`scope`** | **https://graph.microsoft.com/.default**

Write SQL Aggregations

How-to

> Compatible versions: 4.2, 5.5
>
> Works with: *Fusion Server*, *Fusion*

In this article, we provide guidance to help you make the most of the Fusion SQL aggregation engine.

Project fields into the signals_aggr collection

For legacy reasons, the `COLLECTION_NAME_signals_aggr` collection relies on dynamic field names, such as `doc_id_s` and `query_s` instead of `doc_id` and `query`. Consequently, when you project fields to be written to the `COLLECTION_NAME_signals_aggr` collection, you should use dynamic field suffixes as shown in the SQL snippet below:

```
SELECT SUM(typed_aggr_count_i) AS aggr_count_i, query AS query_s,
query AS query_t, doc_id AS doc_id_s, filters AS filters_s,
SPLIT(filters, ' \\$ ') AS filters_ss, weighted_sum(...) AS weight_d
FROM signal_type_groups GROUP BY query, doc_id, filters
```

You are not required to use this approach, but if you do not use dynamic field suffixes as shown above, you will need to change the boosting stages in Fusion to work with different field names.

Use WITH to organize complex queries

A common pattern in SQL aggregation queries is the use of subqueries to break up the logic into comprehensible units. For more information about the WITH clause, see https://modern-sql.com/feature/with. Let us work through an example to illustrate the key points:

```
1: WITH signal_type_groups AS ( 2: SELECT SUM(count_i) AS
typed_aggr_count_i, 3: doc_id, 4: user_id, 5: type, 6:
time_decay(count_i, timestamp_tdt) AS typed_weight_d 7: FROM
product_signals 8: WHERE type IN ('click','cart','purchase') 9: GROUP
BY user_id, doc_id, type 10: ) SELECT SUM(typed_aggr_count_i) AS
aggr_count_i, 11: doc_id AS doc_id_s, 12: user_id AS user_id_s, 13:
weighted_sum(...) AS weight_d 14: FROM signal_type_groups 15: GROUP BY
doc_id, user_id
```

- At line 1, we declare a statement scoped view named `signal_type_groups` using the WITH keyword.
- Lines 2-9 define the subquery for the `signal_type_groups` view.
- At line 7, we read from the `product_signals` collection in Fusion.
- Line 8 filters the input to only include `click`, `cart`, and `purchase` signals. Behind the scenes, Fusion translates this WHERE IN clause to a Solr filter query, e.g. `fq=type:(click OR cart OR purchase)`. The `signal_type_groups` view produces rows grouped by `user_id`, `doc_id`, and `type` (line 9).
- Starting at line 10, we define a subquery that performs a rollup over the rows in the `signal_type_groups` view by grouping on `doc_id` and `user_id` (line 15). Notice how the WITH statement helps break this complex query up into two units that help make aggregation queries easier to comprehend. You are encouraged to adopt this pattern in your own SQL aggregation queries.

Writing a Custom Authorization

How-to

> Compatible versions: 4.2, 5.5, 5.9, 5.10, 5.11, 5.12
>
> Works with: *App Studio*

In some cases, you might want to apply custom business logic for authorization after a user has been authenticated in Appkit. For example, you might want to load group and role information from an external database or directory, in those cases where the authentication provider does not provide this information.

This is relatively easy to do in Appkit, if you follow these steps.

1. Implement your own authorization filter

MyAuthorizationFilter.java

```
import com.google.inject.Singleton; import
twigkit.model.auth.AnonymousUser; import twigkit.model.auth.Role;
import twigkit.model.auth.User; import
twigkit.security.SecurityContext; import
twigkit.security.filter.AuthorisationFilter; import
javax.servlet.http.HttpServletRequest; import
javax.servlet.http.HttpServletResponse; import java.util.List;
@Singleton public class MyAuthorizationFilter implements
AuthorisationFilter { @Override public void init() { } @Override
public boolean filter(HttpServletRequest servletRequest,
HttpServletResponse servletResponse) throws Exception { if
(SecurityContext.getUser() != null && !(SecurityContext.getUser()
instanceof AnonymousUser)) { User user = SecurityContext.getUser();
List<Role> roles = null; // implement your own business logic here for
(Role role : roles) { user.addRole(role); } } return true; } @Override
public void destroy() { } }
```

2. Bind your authorization filter in Guice

Assuming you already have a Guice application module, add your authorization filter to the authorization bindings like so:

MyAppModule.java

```
import com.google.inject.multibindings.Multibinder; import
twigkit.AbstractTwigKitModule; import
twigkit.security.filter.AuthorisationFilter; public class MyAppModule
extends AbstractTwigKitModule { @Override protected void configure() {
Multibinder<AuthorisationFilter> authorisationBinder =
Multibinder.newSetBinder(binder(), AuthorisationFilter.class);
authorisationBinder.addBinding().to(MyAuthorizationFilter.class); } }
```

If you do not already have a Guice app module, simply create one like above, and add an entry to `src/main/resources/META-INF/services/twigkit.TwigKitModule` (note the capitalization of TwigKitModule) containing the Fully-Qualified Class Name of the module. That is, in `src/main/resources/META-INF/services/twigkit.TwigKitModule` add the entry:

```
your.package.MyAppModule
```