elastic

David Pilato
Developer | Evangelist
@dadoonet

Ingest Node

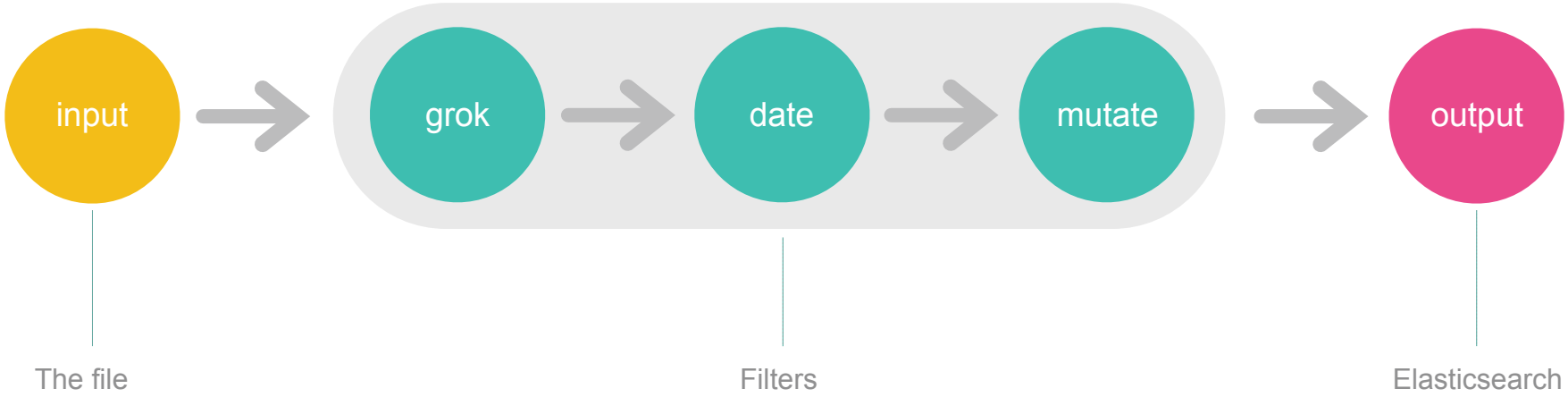
(re)indexing and enriching documents within Elasticsearch

is powered by

Why ingest node?

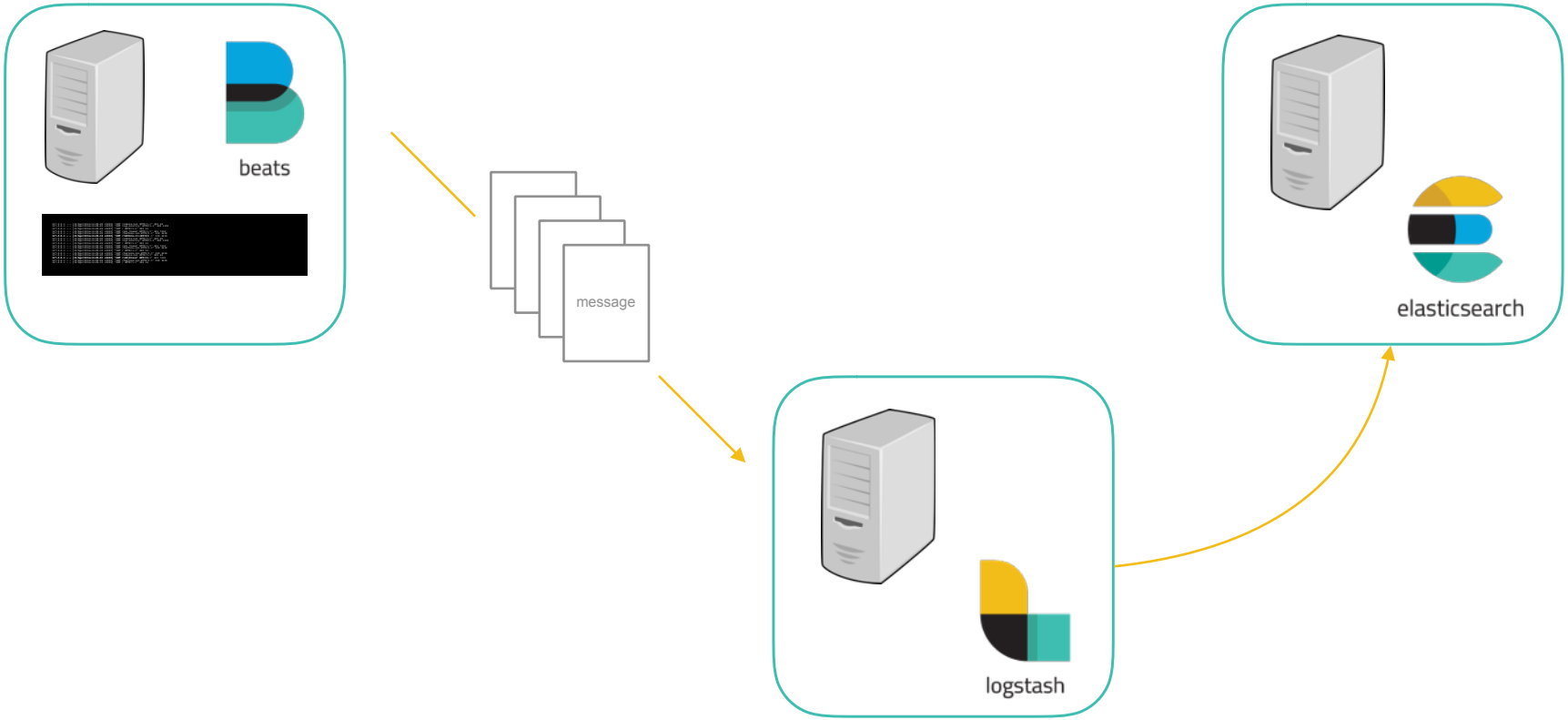
I just want to tail a file.

Logstash: collect, enrich & transport



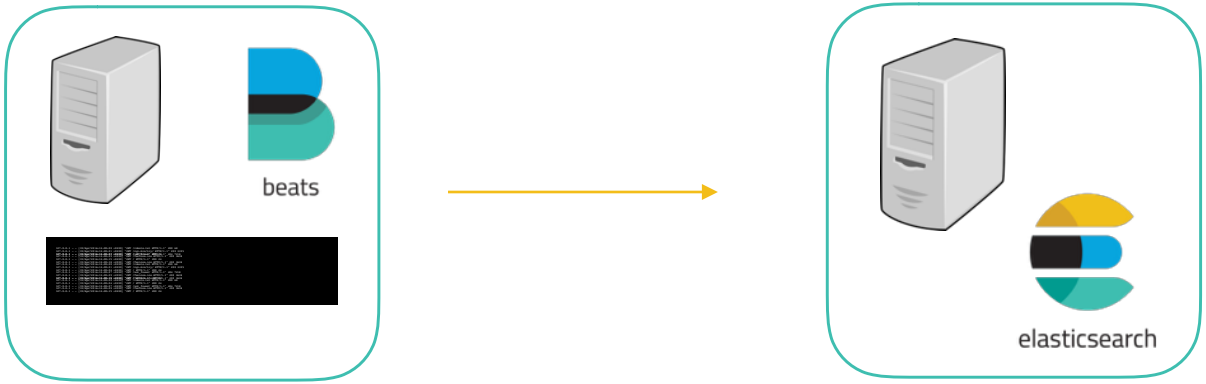
elastic

Logstash common setup



elastic

Ingest node setup



elastic

Filebeat: collect and ship

```
127.0.0.1 - - [19/Apr/2016:12:00:04 +0200] "GET / HTTP/1.1" 200 24
127.0.0.1 - - [19/Apr/2016:12:00:07 +0200] "GET /not_found/ HTTP/1.1" 404 7218
127.0.0.1 - - [19/Apr/2016:12:00:09 +2000] "GET /favicon.ico HTTP/1.1" 200 3638
```


```
{
  "message" : "127.0.0.1 - - [19/Apr/2016:12:00:04 +0200] \"GET / HTTP/1.1\" 200 24"
}
{
  "message" : "127.0.0.1 - - [19/Apr/2016:12:00:07 +0200] \"GET /not_found/ HTTP/1.1\" 404 7218"
}
{
  "message" : "127.0.0.1 - - [19/Apr/2016:12:00:09 +2000] \"GET /favicon.ico HTTP/1.1\" 200 3638"
}
```

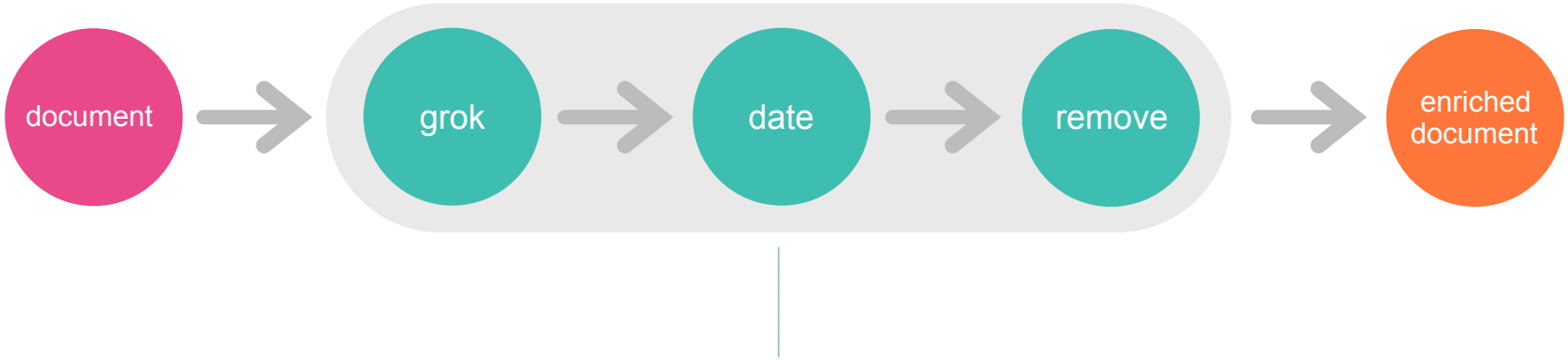
Elasticsearch: enrich and index

```
{
  "message" : "127.0.0.1 - - [19/Apr/2016:12:00:04 +0200] \"GET / HTTP/1.1\" 200 24"
}
```

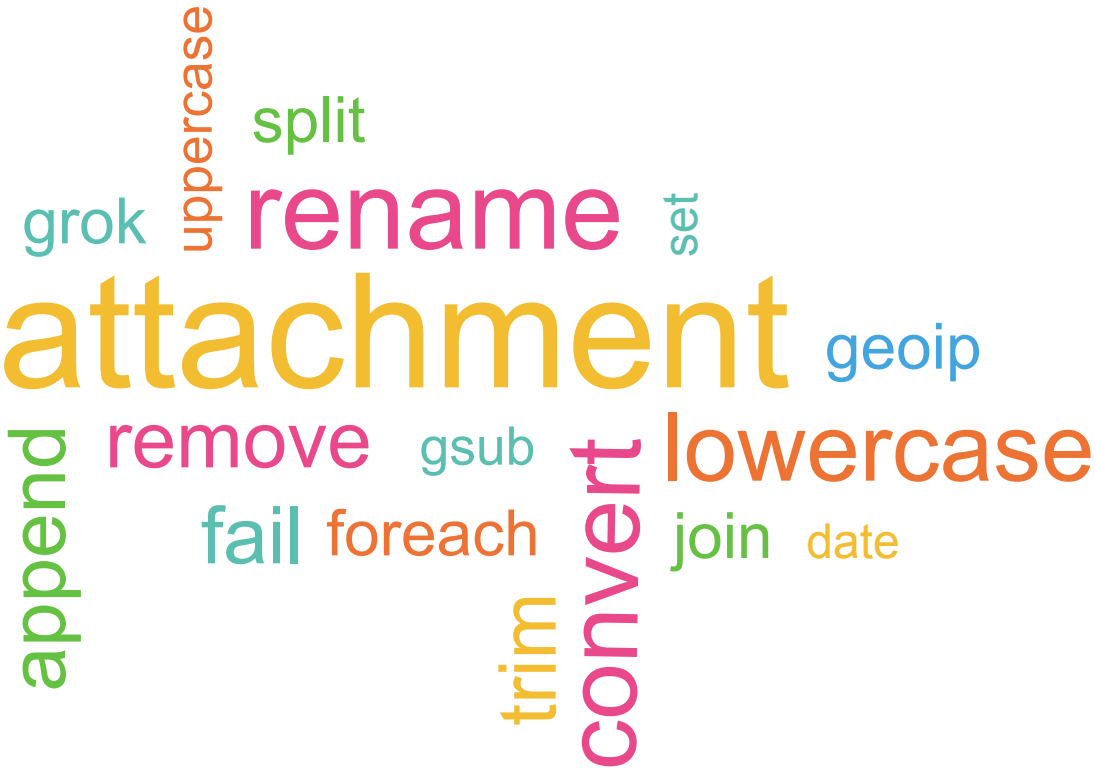
```
{
    "request" : "/",
    "auth" : "-",
    "ident" : "-",
    "verb" : "GET",
    "@timestamp" : "2016-04-19T10:00:04.000Z",
    "response" : "200",
    "bytes" : "24",
    "clientip" : "127.0.0.1",
    "httpversion" : "1.1",
    "rawrequest" : null,
    "timestamp" : "19/Apr/2016:12:00:04 +0200"
}
```

How does ingest node work?

Ingest pipeline



elastic


uppercase
split
grok
rename
set
attachment
geoip
append
remove
gsub
lowercase
fail
foreach
join
date
trim
convert

elastic

Grok processor

Extracts structured fields out of a single text field

```
{
  "grok": {
    "field": "message",
    "patterns": ["%{DATE:date}"]
  }
}
```

Mutate processors

set, remove, rename, convert, gsub, split, join, lowercase, uppercase, trim, append

```
{
  "remove": {
    "field": "message"
  }
}
```

Date processor

Parses a date from a string

```
{
  "date": {
    "field": "timestamp",
    "formats": ["YYYY"]
  }
}
```

Geoip processor

Adds information about the geographical location of IP addresses

```
{
  "geoip": {
    "field": "ip"
  }
}
```

Foreach processor

Do something for every element of an array

```
{
  "foreach": {
    "field" : "values",
    "processor" : {
      "uppercase" : {
        "field" : "_ingest._value"
      }
    }
  }
}
```

Plugins

Introducing new processors is as easy as writing a plugin

```
{
  "your_plugin": {
    ...
  }
}
```

Define a pipeline

```
PUT /_ingest/pipeline/apache-log
{
  "processors" : [
    {
      "grok" : {
        "field": "message",
        "patterns": ["%{COMMONAPACHELOG}"]
      }
    },
    {
      "date" : {
        "field" : "timestamp",
        "formats" : ["dd/MMM/YYYY:HH:mm:ss Z"]
      }
    },
    {
      "remove" : {
        "field" : "message"
      }
    }
  ]
}
```

Pipeline management

Create, Read, Update & Delete

```
PUT /_ingest/pipeline/apache-log
{
  ...
}

GET /_ingest/pipeline/apache-log

GET /_ingest/pipeline/*

DELETE /_ingest/pipeline/apache-log
```

Where can ingest pipelines be used?

Index api

```
PUT /logs/apache/1?pipeline=apache-log
{
    "message" : "..."
}
```

Bulk api

```
PUT /logs/_bulk
{ "index": { "_type": "apache", "_id":
"1", "pipeline": "apache-log" } }\n
{ "message" : "..." }\n
{ "index": {"_type": "mysql", "_id":
"1", "pipeline": "mysql-log" } }\n
{ "message" : "..." }\n
```

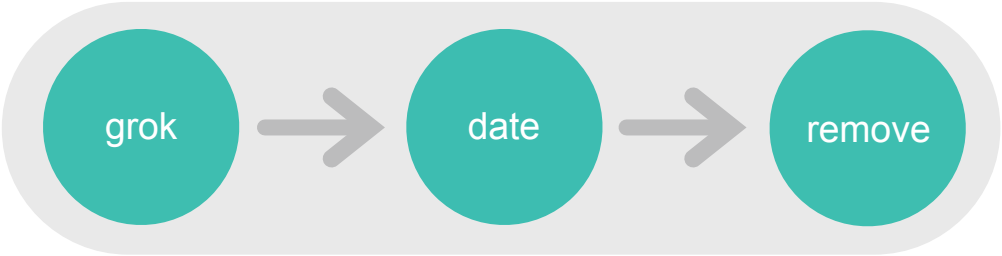
Reindex api

Scroll & bulk indexing made easy

```
POST /_reindex
{
  "source": {
    "index": "logs",
    "type": "apache"
  },
  "dest": {
    "index": "apache-logs",
    "pipeline" : "apache-log"
  }
}
```

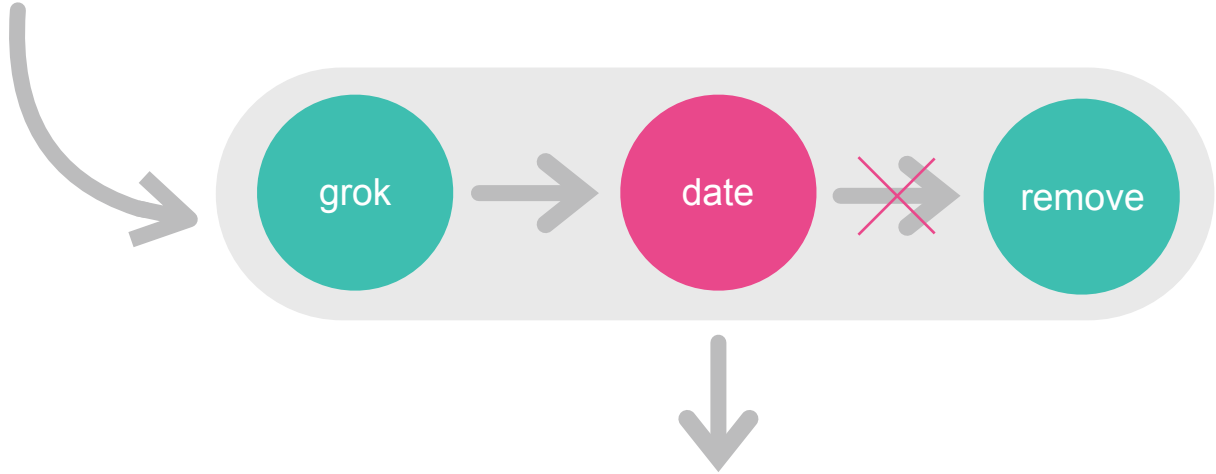
Error handling

```
{
  "message" : "127.0.0.1 - - [19/Apr/2016:12:00:00 +040] \"GET / HTTP/1.1\" 200 24"
}
```



elastic

```
{
  "message" : "127.0.0.1 - - [19/Apr/2016:12:00:00 +040] \"GET / HTTP/1.1\" 200 24"
}
```

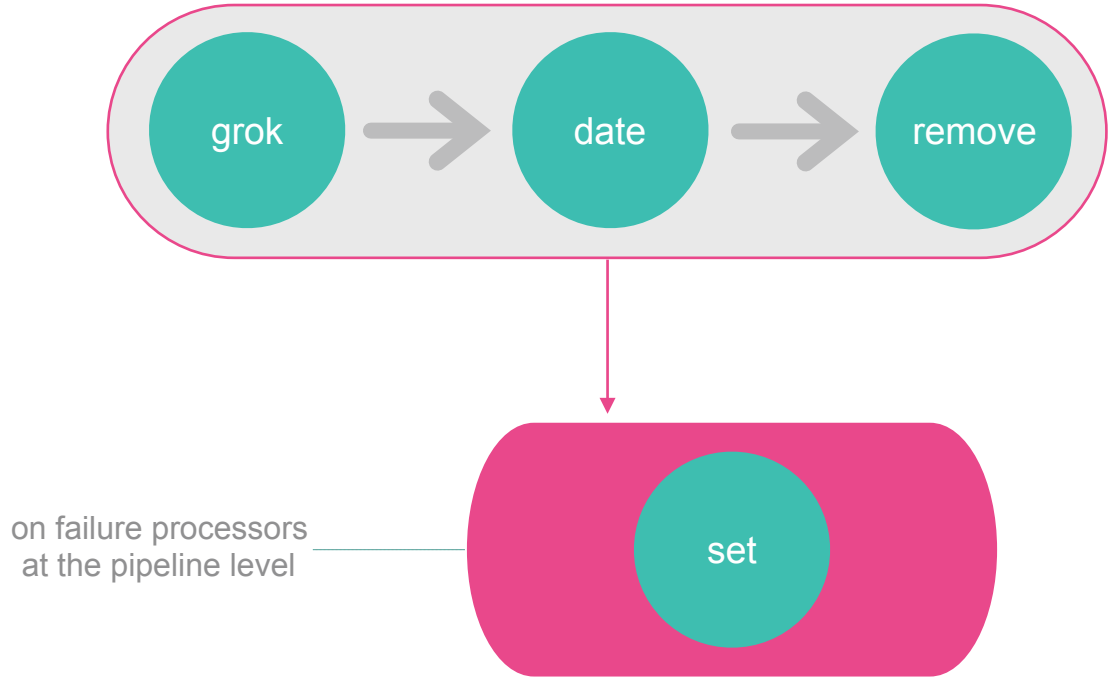


400 Bad Request

unable to parse date [19/Apr/2016:12:00:00 +040]

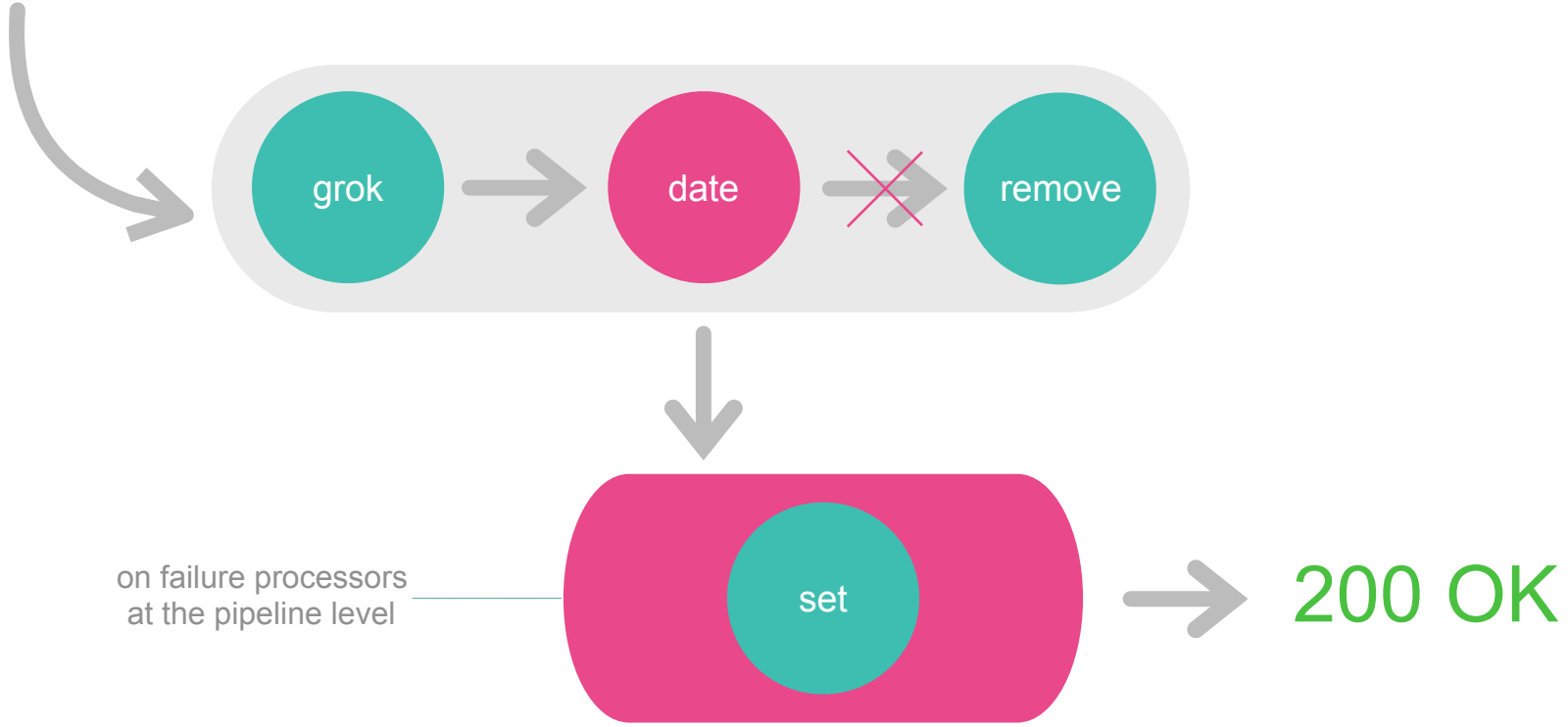
elastic

```
{
  "message" : "127.0.0.1 - - [19/Apr/2016:12:00:00 +040] \"GET / HTTP/1.1\" 200 24"
}
```



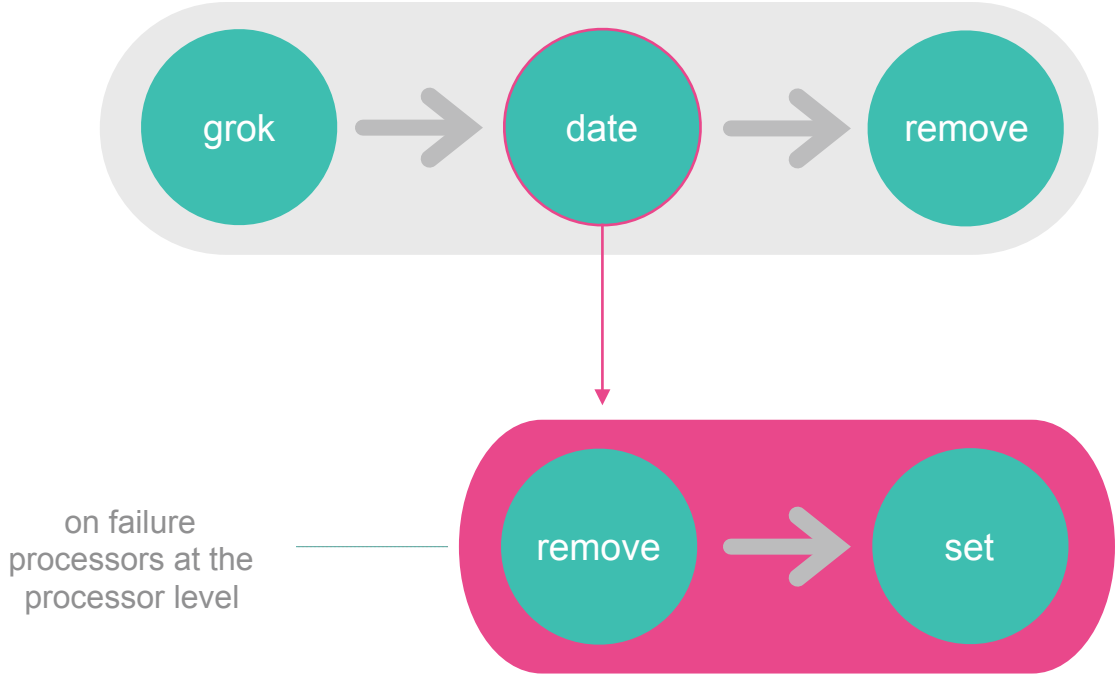
elastic

```
{
  "message" : "127.0.0.1 - - [19/Apr/2016:12:00:00 +040] \"GET / HTTP/1.1\" 200 24"
}
```



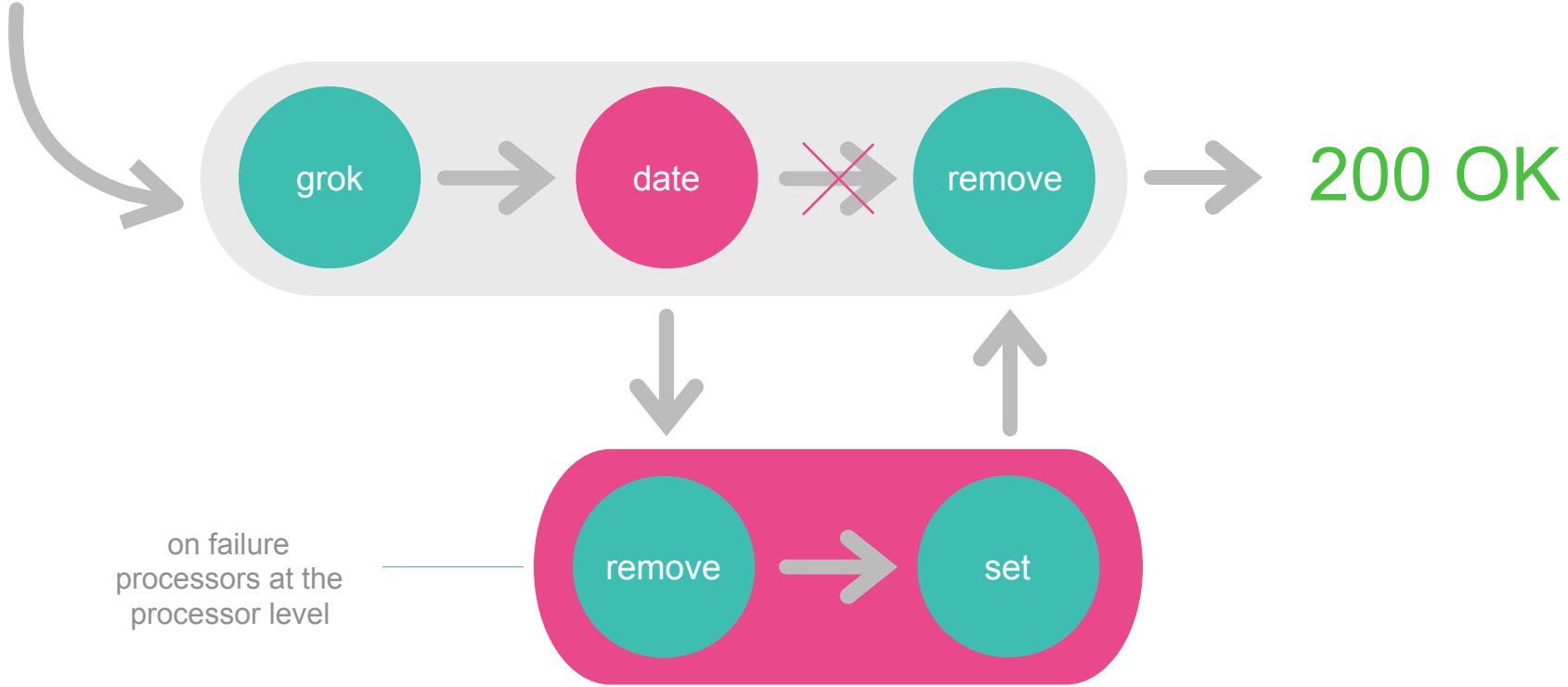
elastic

```
{
  "message" : "127.0.0.1 - - [19/Apr/2016:12:00:00 +040] \"GET / HTTP/1.1\" 200 24"
}
```



elastic

```
{
  "message" : "127.0.0.1 - - [19/Apr/2016:12:00:00 +040] \"GET / HTTP/1.1\" 200 24"
}
```

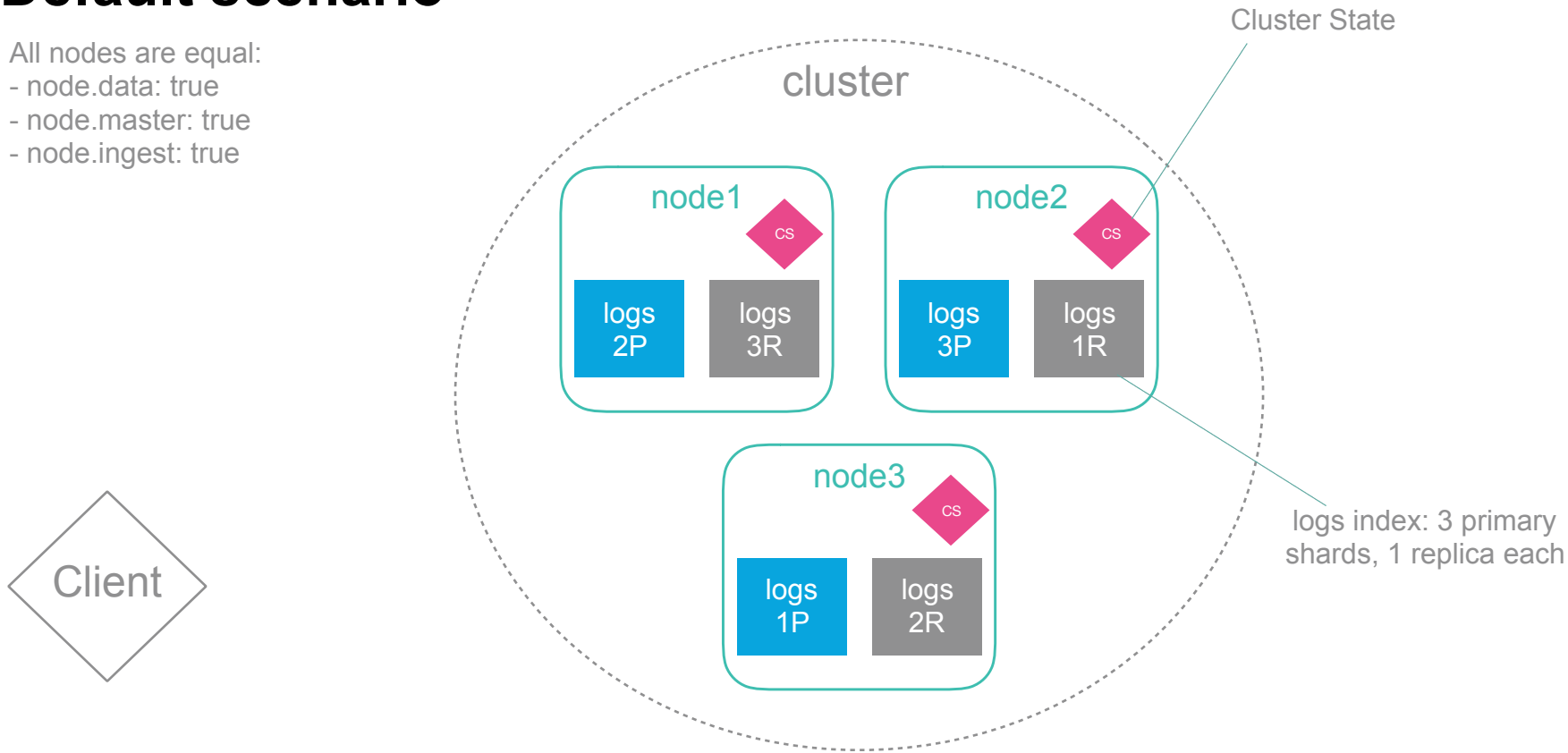


elastic

Ingest node internals

Default scenario

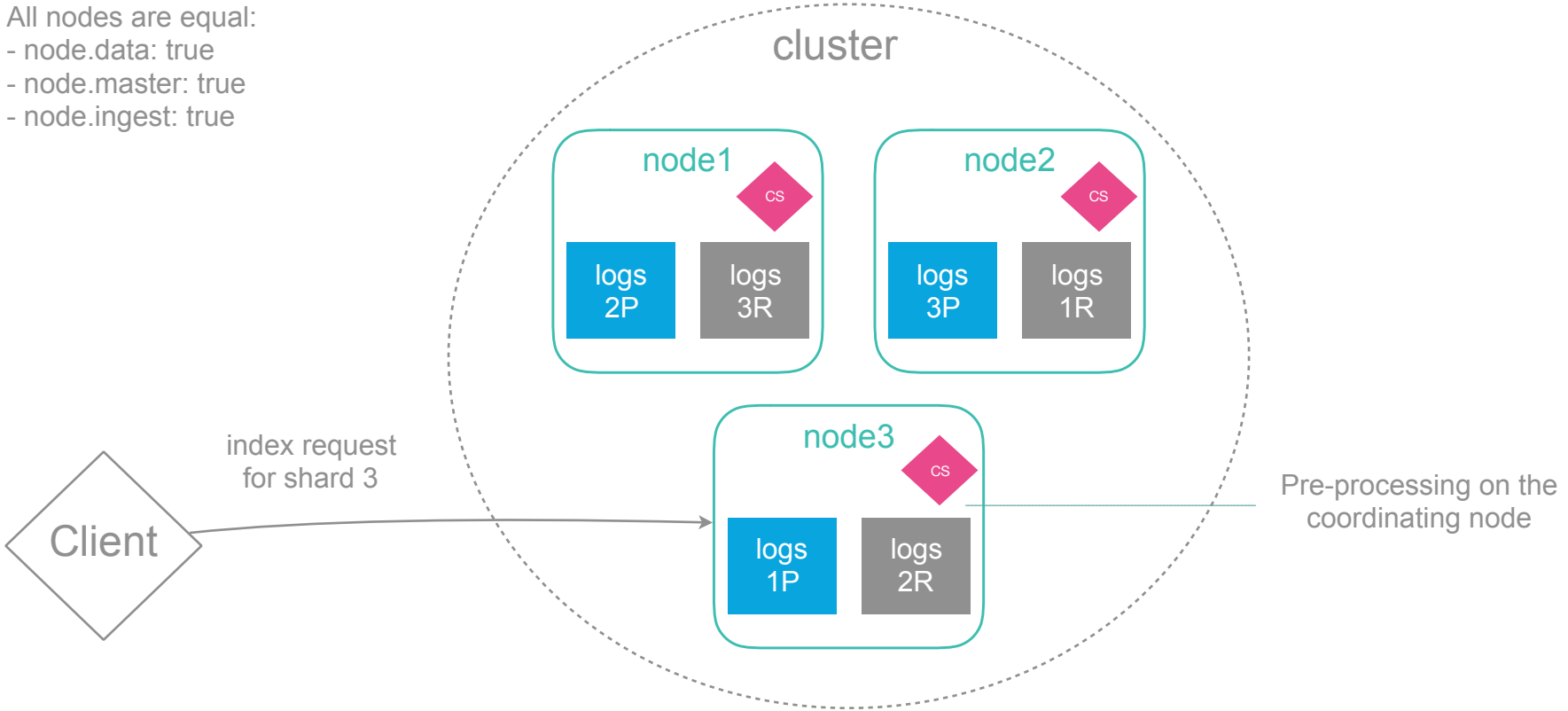
All nodes are equal:
- node.data: true
- node.master: true
- node.ingest: true



elastic

Default scenario

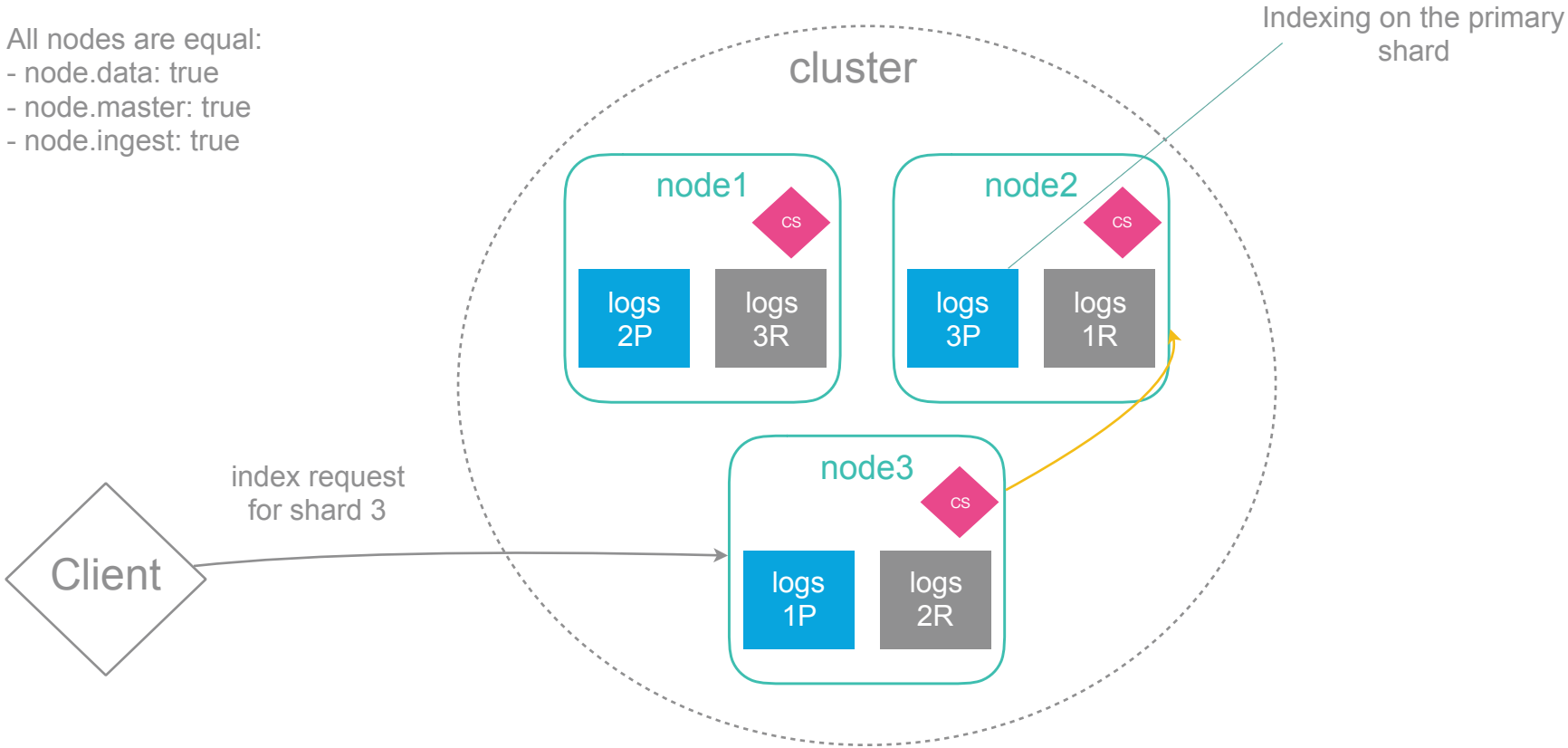
All nodes are equal:
- node.data: true
- node.master: true
- node.ingest: true



elastic

Default scenario

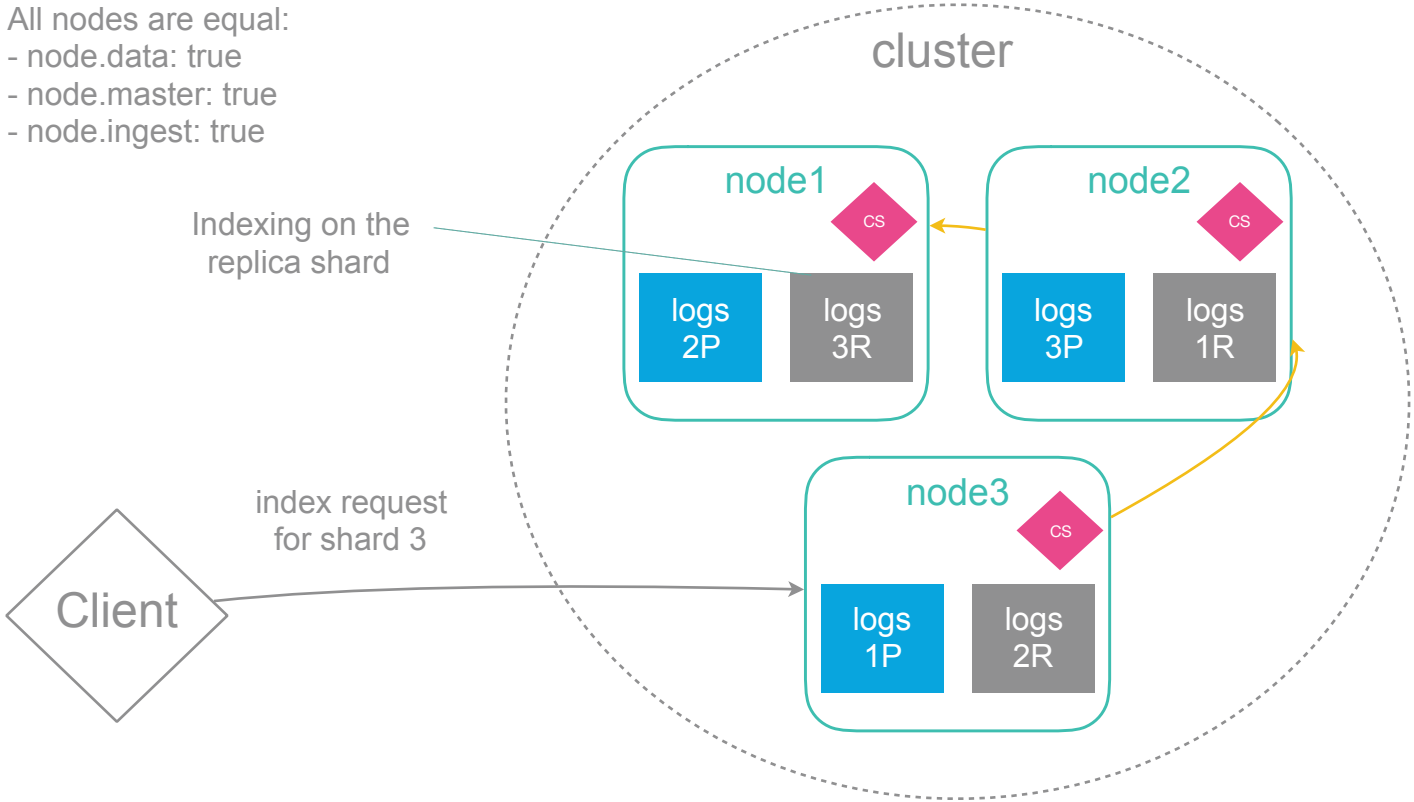
All nodes are equal:
- node.data: true
- node.master: true
- node.ingest: true



elastic

Default scenario

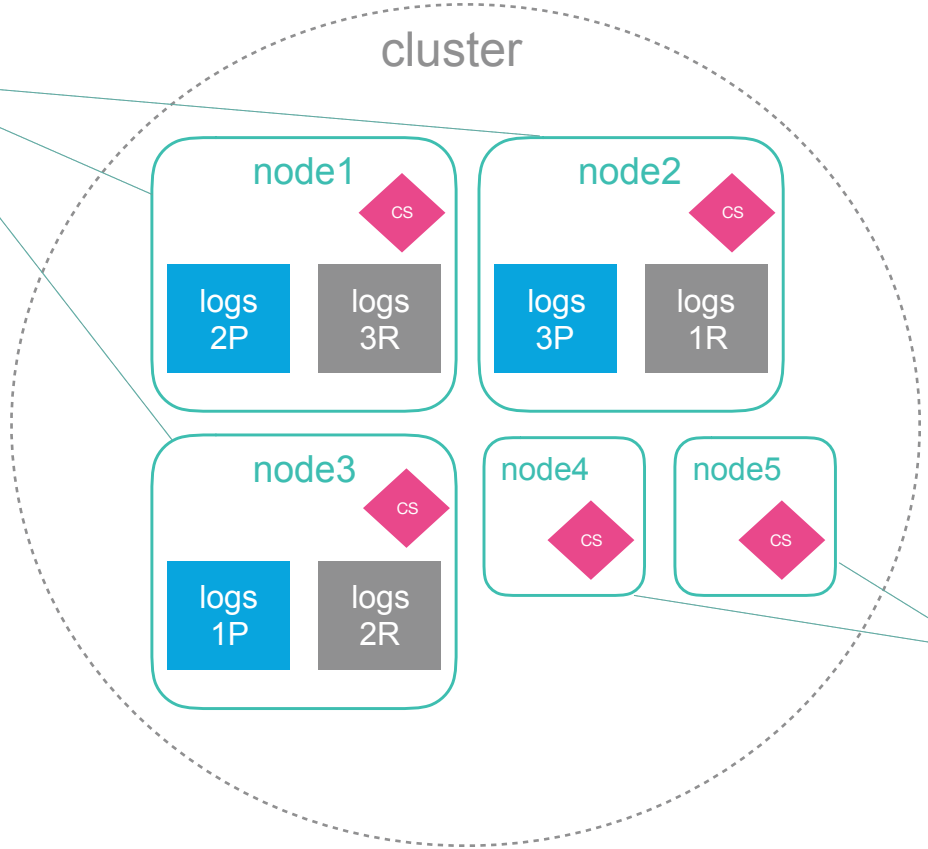
All nodes are equal:
- node.data: true
- node.master: true
- node.ingest: true



elastic

Ingest dedicated nodes

node.data: true
node.master: true
node.ingest: false

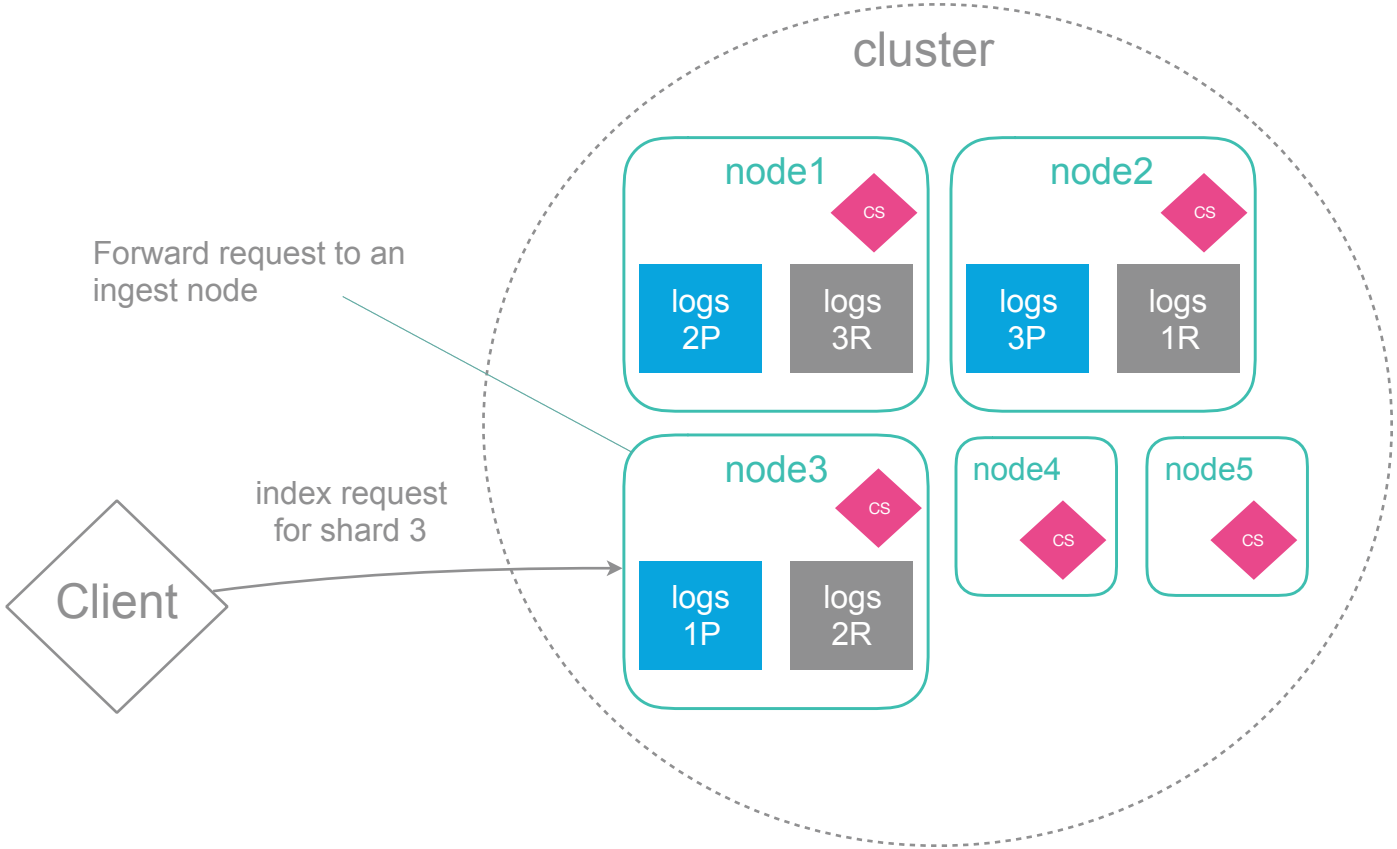


Client

node.data: false
node.master: false
node.ingest: true

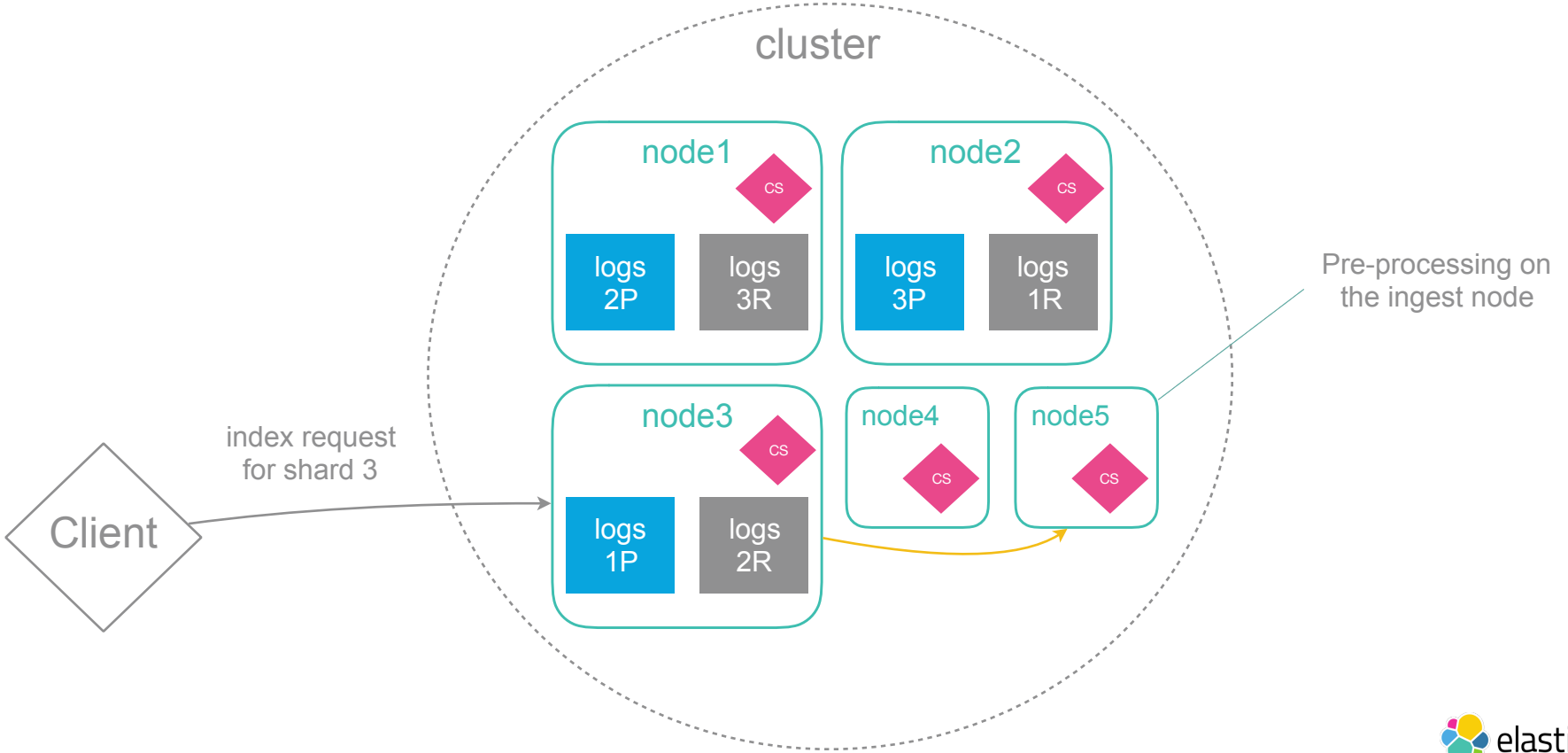
elastic

Ingest dedicated nodes



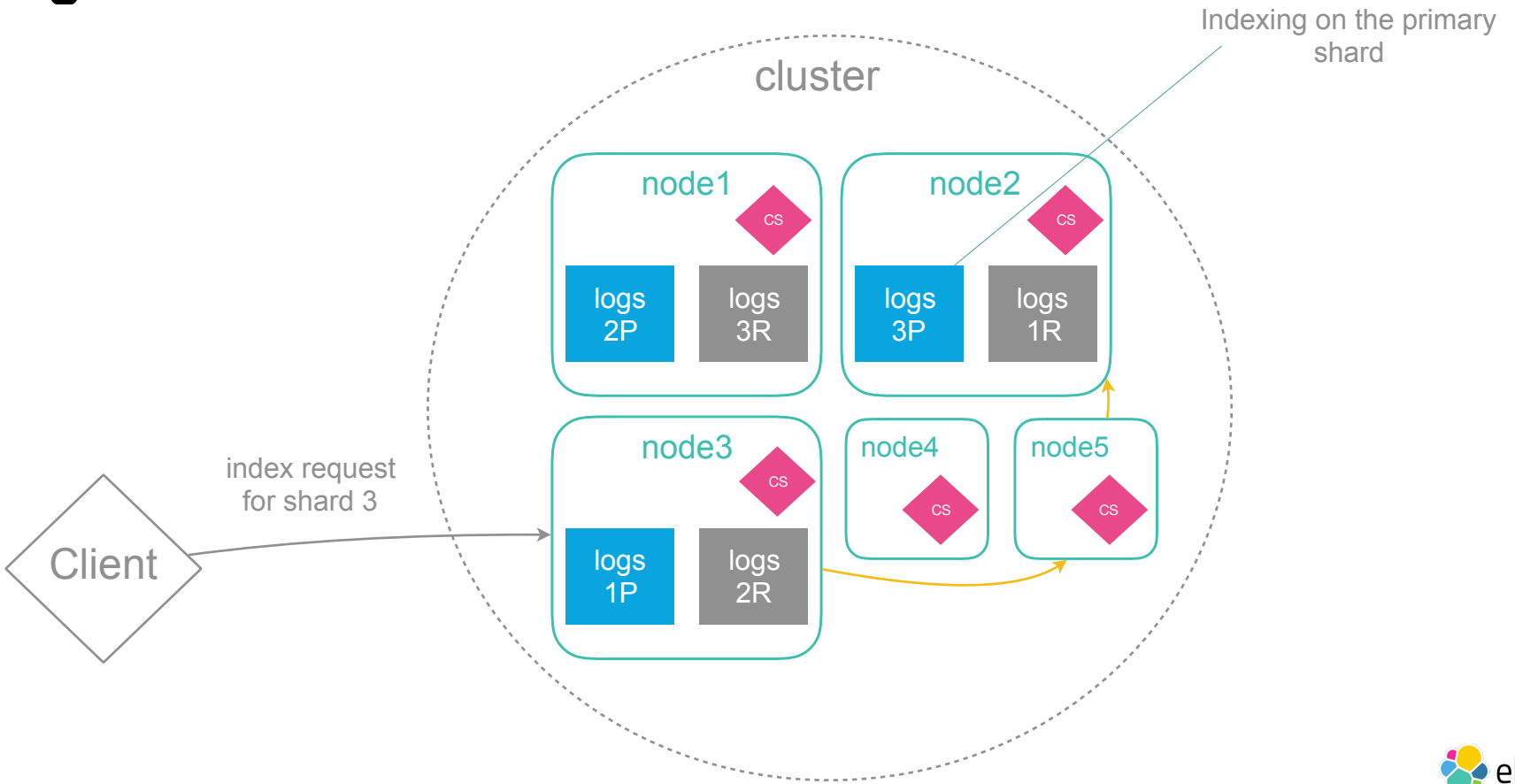
elastic

Ingest dedicated nodes



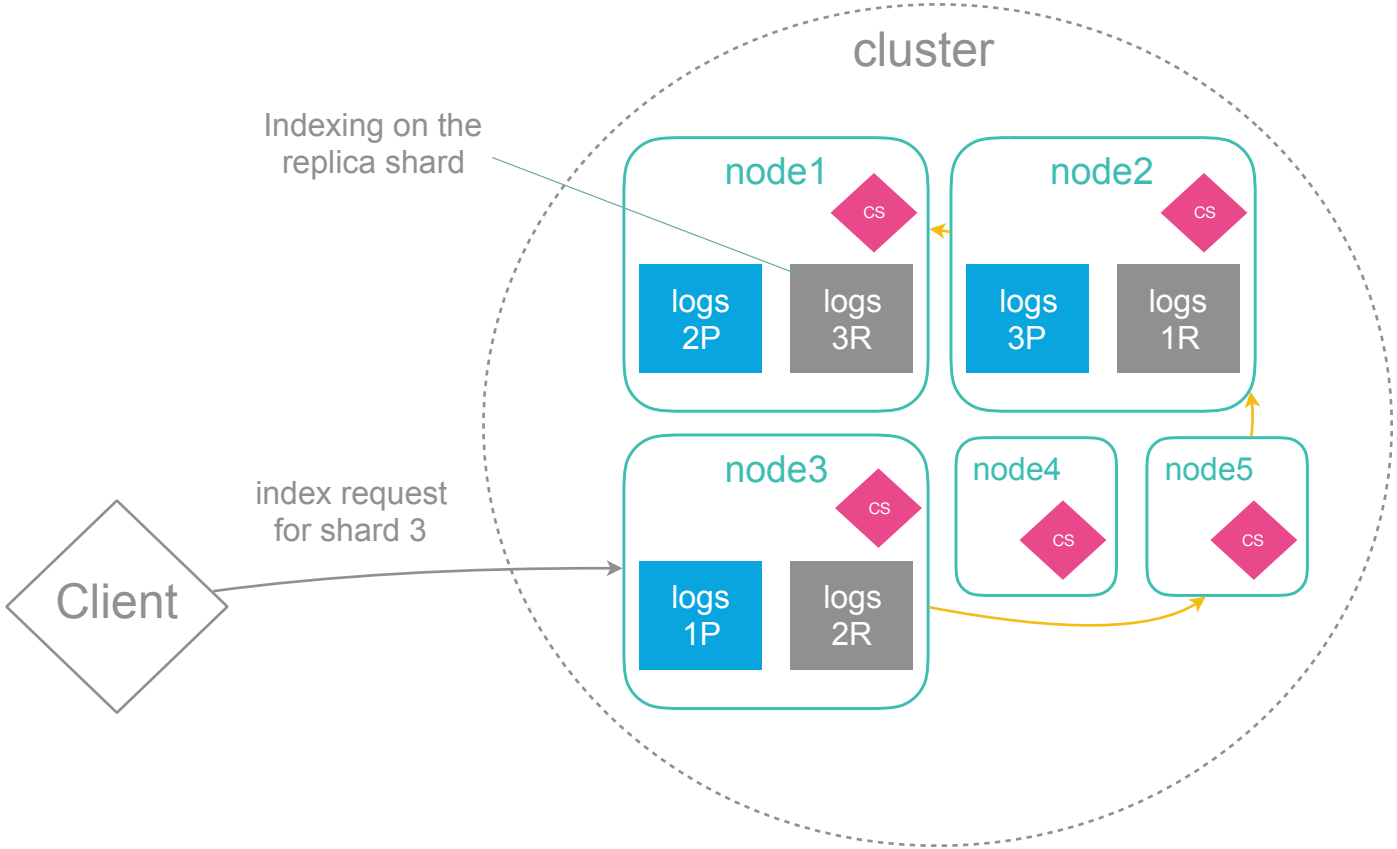
elastic

Ingest dedicated nodes



elastic

Ingest dedicated nodes



elastic

bano-ingest plugin

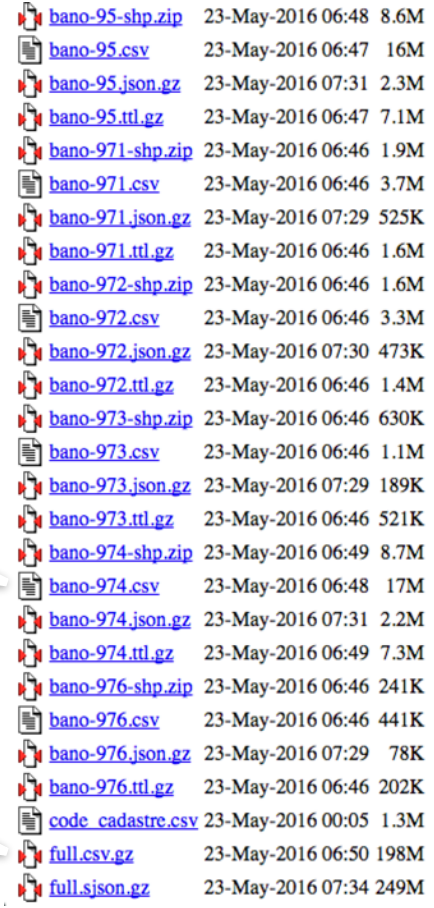
From postal address to geo_point
From geo_point to postal address

What is BANO?

- French Open Data base for postal addresses
- http://openstreetmap.fr/bano
- http://bano.openstreetmap.fr/data/

per department

all addresses



elastic

BANO Format

- bano-976.csv sample (full.csv.gz has same format)

```
976030950H-26,26,RUE DISMA,97660,Bandrélé,CAD,-12.891701,45.202652
976030950H-28,28,RUE DISMA,97660,Bandrélé,CAD,-12.891900,45.202700
976030950H-30,30,RUE DISMA,97660,Bandrélé,CAD,-12.891781,45.202535
976030950H-32,32,RUE DISMA,97660,Bandrélé,CAD,-12.892005,45.202564
976030950H-3,3,RUE DISMA,97660,Bandrélé,CAD,-12.892444,45.202135
976030950H-34,34,RUE DISMA,97660,Bandrélé,CAD,-12.892068,45.202450
976030950H-4,4,RUE DISMA,97660,Bandrélé,CAD,-12.892446,45.202367
976030950H-5,5,RUE DISMA,97660,Bandrélé,CAD,-12.892461,45.202248
976030950H-6,6,RUE DISMA,97660,Bandrélé,CAD,-12.892383,45.202456
976030950H-8,8,RUE DISMA,97660,Bandrélé,CAD,-12.892300,45.202555
976030950H-9,9,RUE DISMA,97660,Bandrélé,CAD,-12.892355,45.202387
976030951J-103,103,RTE NATIONALE 3,97660,Bandrélé,CAD,-12.893639,45.201696
  \_ ID          |    \_ Street Name |          \     \_ Source  \_ Geo point
                 |                   |           \
                 |_ Street Number    |_ Zipcode   \_ City Name
```

Features

- Download, transform and index BANO datasource

```
curl -XPUT 127.0.0.1:9200/_bano/17
curl -XPUT 127.0.0.1:9200/_bano/17,95,29
curl -XPUT 127.0.0.1:9200/_bano/_full
```

- Create a new ingest processor

```
curl -XPUT "localhost:9200/_ingest/pipeline/bano-test?pretty" -d '{
  "description": "my_pipeline",
  "processors": [ {
    "bano": {}
  } ]
}'
```

From structured address (french format)…

```
curl -XPOST "localhost:9200/_ingest/pipeline/bano-test/_simulate?pretty" -d '{
  "docs": [ {
    "_index": "index",
    "_type": "type",
    "_id": "id",
    "_source": {
      "address": {
        "number": "25",
        "street_name": "georges",
        "zipcode": "17440",
        "city": "Aytré"
      }
    }
  } ]
}'
```

From a geo point…

```
curl -XPOST "localhost:9200/_ingest/pipeline/bano-test/_simulate?pretty" -d '{
  "docs": [ {
    "_index": "index",
    "_type": "type",
    "_id": "id",
    "_source": {
      "location": {
        "lat": 46.135283,
        "lon": -1.113750
      }
    }
  } ]
}'
```

Combine with other ingest processors

```
curl -XPUT "localhost:9200/_ingest/pipeline/bano-test-4?pretty" -d '{
  "description": "debug",
  "processors": [ {
      "geoip" : {
        "field" : "ip"
      }
    }, {
    "bano": {
      "location_lat_field": "geoip.location.lat",
      "location_lon_field": "geoip.location.lon"
    }
  } ]
}'
```

From an IP address…

```
curl -XPOST "localhost:9200/_ingest/pipeline/bano-test-4/_simulate?pretty&verbose" -d '{
  "docs": [ {
    "_index": "index",
    "_type": "type",
    "_id": "id",
    "_source": {
      "ip" : "82.229.80.187"
    }
  } ]
}'
```

Demo time!

Writing an ingest plugin

Writing the processor

```
public final class BanoProcessor extends AbstractProcessor {
  private final String cityField;
  public BanoProcessor(String cityField) { this.cityField = cityField; }

  @Override
  public void execute(IngestDocument ingestDocument) {

    // Implement your logic code here
    if (ingestDocument.hasField(cityField)) {
       String city = ingestDocument.getFieldValue(cityField, String.class)

       // Like searching in elasticsearch with a city field
       Location location = banoEsClient.searchByCity(city);

       // Then modify the document as you wish
       Map<String, Object> locationObject = new HashMap<>();
       locationObject.put("lat", location.getLat());
       locationObject.put("lon", location.getLon());
       ingestDocument.setFieldValue("location", locationObject);
    }
  }
}
```

Writing the processor factory

```
public static final class Factory implements Processor.Factory {
  @Override
  public Processor create(Map<String, Processor.Factory> map, String processorTag,
                          Map<String, Object> config) throws Exception {
    // Read the bano processor config
    String cityField = readStringProperty("bano", processorTag, config,
       "city_field",     // We read here the value of "city_field" in config
       "address.city");  // If not set we will read from "address.city" by default

    // Do the same for other fields

    // Create the processor instance
    return new BanoProcessor(cityField, otherFields...);
}
```

Writing an ingest plugin

```
public class IngestBanoPlugin extends Plugin implements IngestPlugin {

  @Override
  public Map<String, Processor.Factory> getProcessors(Processor.Parameters parameters) {
    return Collections.singletonMap("bano", new BanoProcessor.Factory());
  }

}
```

Go get Elasticsearch 5.0.2!

https://www.elastic.co/downloads/elasticsearch

OCT 17TH, 2016 10:55 AM | COMMENTS

Creating an Ingest Plugin for Elasticsearch (Updated for GA)

NOTE: This article is an updated version of Creating an Ingest plugin for elasticsearch

This blog post is part of a series which will teach you:

- How to write a plugin for elasticsearch 5.0 using Maven.
- How to write an ingest plugin for elasticsearch 5.0 (what you are reading now).
- How I wrote the `ingest-bano` plugin which will be hopefully released soonish.

Today, we will focus on writing an Ingest plugin for elasticsearch.

> *Hey! Wait! You wrote `Ingest`? What is that?*

Ingest is a new feature coming in elasticsearch 5.0. It helps you to transform your data on the fly while injecting it into elasticsearch. Read more in elastic blog post.

If you know me and my work before I joined elastic, I have always been in love with data crawling and transformation as I wrote myself some plugins called rivers.

About Me

Developer | Evangelist at elastic and creator of the Elastic French Speakers User Group. Frequent speaker about all things Elastic, in conferences, for User Groups and in companies with BBL talks. In my free time, I enjoy coding and DJ four times per year, just for fun. Living with my family in Cergy, France.

Recent Posts

Elasticsearch Real Integration Tests With Security Enabled (Updated for GA)

Creating Elasticsearch Transport Action (Updated for GA)

Adding a New REST Endpoint to Elasticsearch (Updated for GA)

Elasticsearch Real Integration Tests (Updated for GA)

Creating an Ingest Plugin for Elasticsearch (Updated for GA)

Tweets

elastic

David Pilato
Developer | Evangelist
@dadoonet

Bano ingest plugin

Watch this space: https://github.com/dadoonet
And follow me on Twitter!