FOM Hochschule für Oekonomie und Management

Hochschulzentrum Köln

Wirtschaftsinformatik

Observability for Micro-Service Architectures

Software Engineering, Salah Zayak, WS 20

Christian Frank (#473088)

November 17, 2020

ChilledCow: lofi hip hop radio - beats to relax/study to

This work is licensed under the Creative Commons Attribution 4.0 International License.
To view a copy of this license, visit http://creativecommons.org/licenses/by/4.0/ or
send a letter to Creative Commons, PO Box 1866, Mountain View, CA 94042, USA.

Table of contents

List of Figures

List of Abbreviations

AWS	Amazon Web Services
CNCF	Cloud Native Computing Foundation
ELT	Extract, Load, Transform
ETL	Extract, Transform, Load
HTTP	Hypertext Transfer Protocol
ITIL	Information Technology Infrastructure Library
ITOM	IT Operations Management
K8s	Kubernetes
REST	Representational State Transfer
SIEM	Security Information and Event Management
stdout	Standard Output (Unix, Linux)

1 Introduction

1.1 Observability

As software architecture transitions more and more from monoliths to a distributed model, based on micro services, the ability to trace and observe the state of an application across multiple instances and components becomes more and more important. Traditional approaches, such as logging to a file, become more difficult if you do not have a single place anymore to look for messages, but potentially hundred different places all over the underlying platform.

In this paper, we want to look at observability for micro-service architectures and start with a general introduction to observability itself.

1.2 Pronouns

In time of a global pandemic, it is especially important to be mindful about hate speech and other forms of offensive communication – this includes taking care about gender inclusivity. "The universal singular they is inclusive of people who identify as male, female or nonbinary."[1] Throughout this text, I will make an attempt to use gender neutral language. To quote Jane Roper on WBUR, “They Is Here to Stay. Get Over It.”[2] Also, should you ever misgender a person, One Medical has some simple advice: Apologize and correct theyself.[3]

1.3 Inclusive Naming

In addition to this, effort is underway to remove offensive language from code, the beginning of a word list[4] is available and will be discussed in more detail within CNCF at the upcoming conferences.

---

[1] *Saguy, Abigail and Williams, Juliet* (Scientific American, 2019) Why we should all use they/them pronouns
[2] *Roper, Jane* (WBUR, 2019): They Is Here to Stay. Get Over It.
[3] See *One Medical* (Instagram, 2020): Navigating Pronouns 101
[4] See *Inclusive Naming* (Inclusive Naming Initiative, 2020): Word replacement list

2 Observability

2.1 Logging & Tracing

Logging, in its most simple form involves writing activity records to stdout. As an example, an application that reads a record from a database could log these activities:

- Database opened
- Record queried and retrieved
- Database closed

This would allow an external observer to watch what the application is doing and, in case of an error, check where the error might have occurred. To further this, the application could increase its verbosity by including trace information, such as the SQL query string in the example above, usually toggled by a flag. In micro-service architectures, this simple form of logging has made a resurgence and is now the standard behavior for containerized applications.

2.2 Monitoring

Monitoring on the other hand usually describes the activity of looking at available metrics, either from the application or the operating system. Metrics could be CPU consumption, packets sent and received, or in our example, the number of records retrieved from the database. Unlike logging, which is an active feature of the application, monitoring generally involves the use of an external tool to query the metrics and display them. We will have a look at some common tools in the next chapter.

2.3 Observability

Observability in Software Engineering seems to be a rather new feature in application design. The term itself is not new, it originated in the world of Engineering.

So, what is Observability? Peter Waterhouse defines it on the New Stack as follows: “Basically, and as the definition states, it’s a measure of how well internal states of a

system can be inferred from knowledge of its external outputs. So, in contrast to monitoring, which is something we actually do, observability (as a noun), is more a property of a system."[5]

As a property of the application itself, it is something that must be included in the application design from the very beginning. Arun Chandrasekaran recommends for Gartner: "Developers are more focused on the functional aspects of these application containers than on the operational requirements of monitoring them […] Developers should instead focus on instrumenting their applications to enable observability."[6]

Instrumentation could include writing meaningful log messages, expose meaningful metrics, and provide status information, for example from an internal state machine.

In the next chapter, we will look at a set of tools to help us with observing micro-service applications.

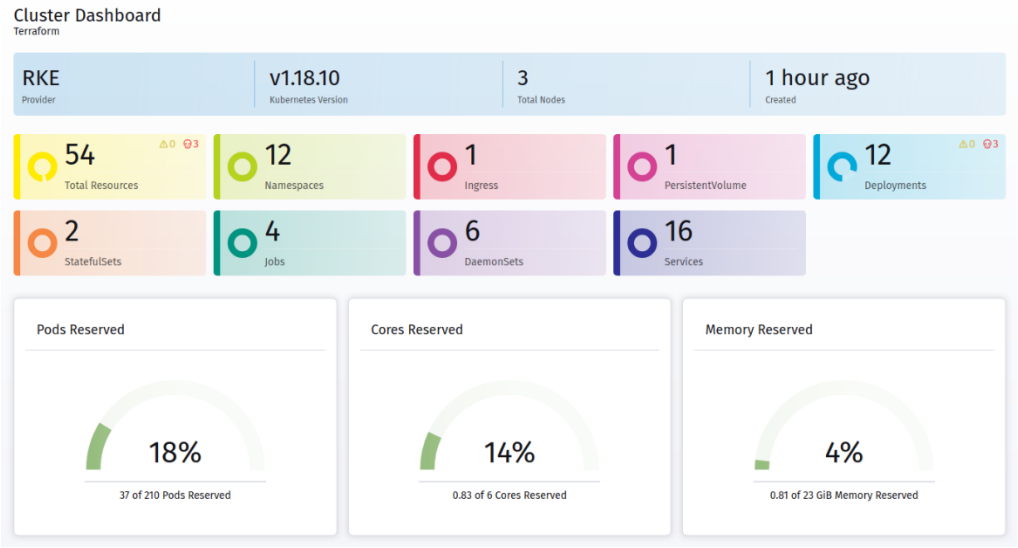
[5] *Waterhouse, Peter* (The New Stack, 2018): Monitoring and Observability
[6] *Chandrasekaran, Arun* (Gartner,2020): Best Practices for Running Containers and Kubernetes in Production

3 Available Tools

3.1 Kubernetes

The most used common platform to run micro-service on is Kubernetes. Kubernetes, or K8s in short, is an orchestrator for various container run-time systems, spread out over several nodes. A collection of nodes under common orchestration is generally referred to as a cluster.

Figure 1 – Kubernetes Cluster



Much like its underlying collection of nodes, Kubernetes itself is also a distributed application. In a Kubernetes cluster we are thus looking at a distributed system orchestrating distributed applications. Furthermore, micro-service applications tend to be stateless and ephemeral, meaning that operational information will vanish after execution. Charlie Fiskeaux summarizes this for the New Stack as follows: “Since observability is the ability to infer the state of a system through knowledge of that system’s outputs, it sure seems like Kubernetes is a system with minimal observability.”[7]

[7] *Fiskeaux, Charlie* (The New Stack, 2020): Why Monitoring Kubernetes Is so Challenging

This is not only an issue for troubleshooting, but also for security – it is quite important to ingest all available information from a Kubernetes cluster into your SIEM environment.[8]

For the rest of this paper, we will focus on Kubernetes as the execution platform and look at the available tools from that perspective.

3.2 Syslog

Syslog is a tool to collect log messages from applications and the underlying operating system itself, either locally or remotely on a central syslog server. Syslog is an integral part of all Unix and Linux systems and described in RFCs 3164 and 5424. Log messages are transported and store as plain text. A typical syslog message for an error might look like this:

```
node_13.68.140.156_01.log: Nov 8 13:03:44 13.68.140.156 az-bff2b1: log:time="2020-11-
08T13:03:41Z" level=info msg="Received error running agent tracker loop. Retrying in 5
seconds." error="rpc error: code = Unavailable desc = last connection error: connection
error: desc = "transport: Error while dialing dial tcp 10.43.187.2:50400: connect:
connection refused""
```

On Kubernetes, platform logs tend to be quite verbose. An hour's worth of logs on a cluster without any active application can easily exceed 200 Megabytes:

```
-rw-r--r-- 1 cfrank ewscom 57718294 Nov 8 16:02 node_13.68.140.156_12.log
-rw-r--r-- 1 cfrank ewscom 43436469 Nov 8 16:01 node_40.114.115.34_12.log
-rw-r--r-- 1 cfrank ewscom 59959679 Nov 8 16:00 node_40.121.50.147_12.log
```

[8] See *Cloudberry Engineering* (Cloudberry Engineering, 2020): A Practical Introduction to Container Security

3.3 Splunk, Icinga, Nagios

Back in the time, when ITIL was still a thing, several ITOM tools were created to cope with the huge amount of log data of traditional IT systems, Splunk[9], Icinga[10] and Nagios[11], to name a few. Goal of these tools was not only to ingest the log data, but also store them in an accessible format and perform initial correlation between the messages, to support automated actions (ETL principle).

3.4 Elasticsearch, FluentD, FluentBit, FireLens

With distributed systems, the amount of log data increased dramatically. At the same time, a new discipline emerged, Data Science. Applied to the management of log data, this means that we would first ingest all log data in an unstructured way and perform the analysis and correlation afterwards (ELT principle). The most used data lake for log data is Elasticsearch[12], and log data ingest can be performed with FluentD[13], FluentBit[14] or FireLens[15].

3.5 Prometheus, Loki, Tempo

Applying the same big data techniques for monitoring data and metrics led to the use of time-series database for storage. The most prominent tool in that category is Prometheus[16], which has become the de-facto standard for monitoring Kubernetes clusters. Recently, the ingestion of log files has been added to the time-series database and a new Tool, Tempo, was created to support correlation.[17]

Efforts are underway to establish an open standard for metrics, based on Prometheus.[18]

---

[9] See *Splunk* (Splunk, 2020): The Data-to-Everything Platform
[10] See *Icinga* (Icinga, 2020): Inspect your Entire Infrastructure
[11] See *Nagios* (Nagios, 2020): The Industry Standard in IT Infrastructure Monitoring
[12] See *Elasticsearch* (Elasticsearch BV, 2020): The heart of the free and open Elastic Stack
[13] See *FluentD* (FluentD Project, 2020): Build Your Unified Logging Layer
[14] See *FluentBit* (Treasure Data, 2020): Cloud Native Log Forwarder
[15] See *FireLens* (Amazon Web Services, 2020): Custom Log Routing
[16] See *Prometheus* (Prometheus, 2020): From metrics to insight
[17] See *Hahn, Silke* (Heise Medien GmbH & Co. KG, 2020): Grafana … neuem Tool Tempo und mit Loki 2.0
[18] See *OpenObservability* (OpenObservability, 2020): Evolving the Prometheus format into a standard

3.6 AWS Dashboards, Thundra

One way of visualizing application state information is with operational dashboards[19]. One example of such a Dashboard on AWS would be the Thundra Application Observability and Security Platform.[20]

3.7 Dynatrace

Another tool targeted at Enterprise IT is Dynatrace, it combines metrics, logs and traces with topology information and AI-based analysis.

3.8 Pixie

A fairly new tool for observability is Pixie, which bills itself as “Instantly troubleshoot your applications on Kubernetes. No instrumentation. Debug with scripts. All inside Kubernetes.”[21] Pixie aims to combine log data with telemetry (monitoring) data.

In addition to all other data inputs, telemetry plays an important role in distributed tracing.[22]

In the following chapter we will focus on Pixie and have a look at a couple of sample telemetry screenshots.

[19] See *Campbell, Matt* (InfoQ, 2020): AWS Publishes Best Practices Guide for Operational Dashboards
[20] See *Thundra* (Amazon Web Services, 2020): Mastering Observability on the Cloud
[21] *Pixie Labs* (Pixie Labs,2020): Instantly troubleshoot your applications on Kubernetes
[22] See *Parker, Austin and Spoonhower, Daniel* (O’Reilly Media, 2020): Distributed Tracing in Practice

4 Using Pixie to observe

4.1 Installing Pixie

Pixie is currently in public Beta, there is not yet any information available on pricing and licensing options.

Pixie runs on Kubernetes; it can be co-located with the application that should be observed and will be installed via Helm:

```
helm repo add pixie https://pixie-helm-charts.storage.googleapis.com

helm install pixie pixie/pixie-chart --set deployKey=xxxxx-xxxxx-xxxxxx
NAME: pixie
LAST DEPLOYED: Sun Nov 8 07:27:31 2020
NAMESPACE: default
STATUS: deployed
REVISION: 1
TEST SUITE: None
```

After installation, these pods should be running:

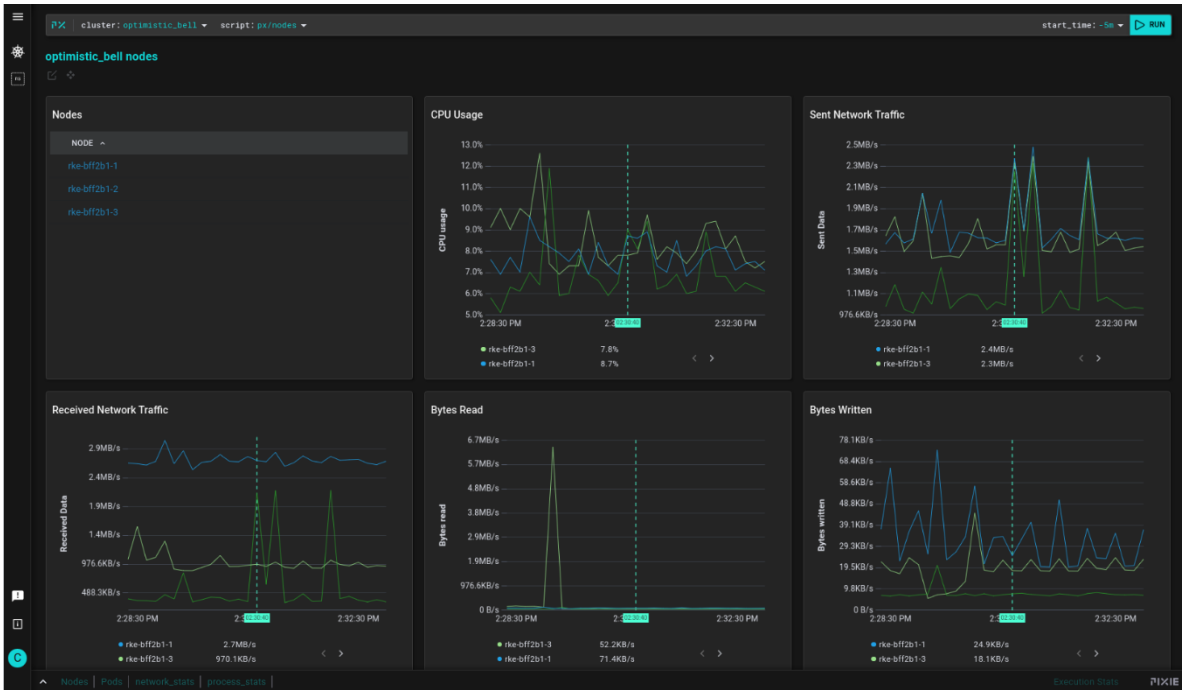
```
NAME                                      READY   STATUS    RESTARTS   AGE
etcd-0                                    1/1     Running   0          38m
kelvin-cb5b59c6b-pxb2q                    1/1     Running   3          54m
nats-operator-94865bc4d-jkw49             1/1     Running   2          54m
pl-nats-1                                 1/1     Running   0          54m
vizier-certmgr-55fbc6b59c-bt7wq           1/1     Running   0          54m
vizier-cloud-connector-7b57dc6d5b-dk2jk   1/1     Running   5          53m
vizier-metadata-6d4fc6cd99-5c2kt          1/1     Running   12         54m
vizier-pem-n99ct                          1/1     Running   4          54m
vizier-pem-vk8nb                          1/1     Running   4          54m
vizier-pem-z7xrp                          1/1     Running   4          54m
vizier-proxy-8cc85b74-vfr2x               1/1     Running   0          54m
vizier-query-broker-86c79b56cf-dhmxm      1/1     Running   6          54m
```

Pixie's vizier will communicate with the Pixie Labs cloud servers, from where the user interface can be accessed.

4.2 Sample cluster telemetry data

After installation, let us first look at some cluster-level telemetry. One important information is the health of the overall platform:

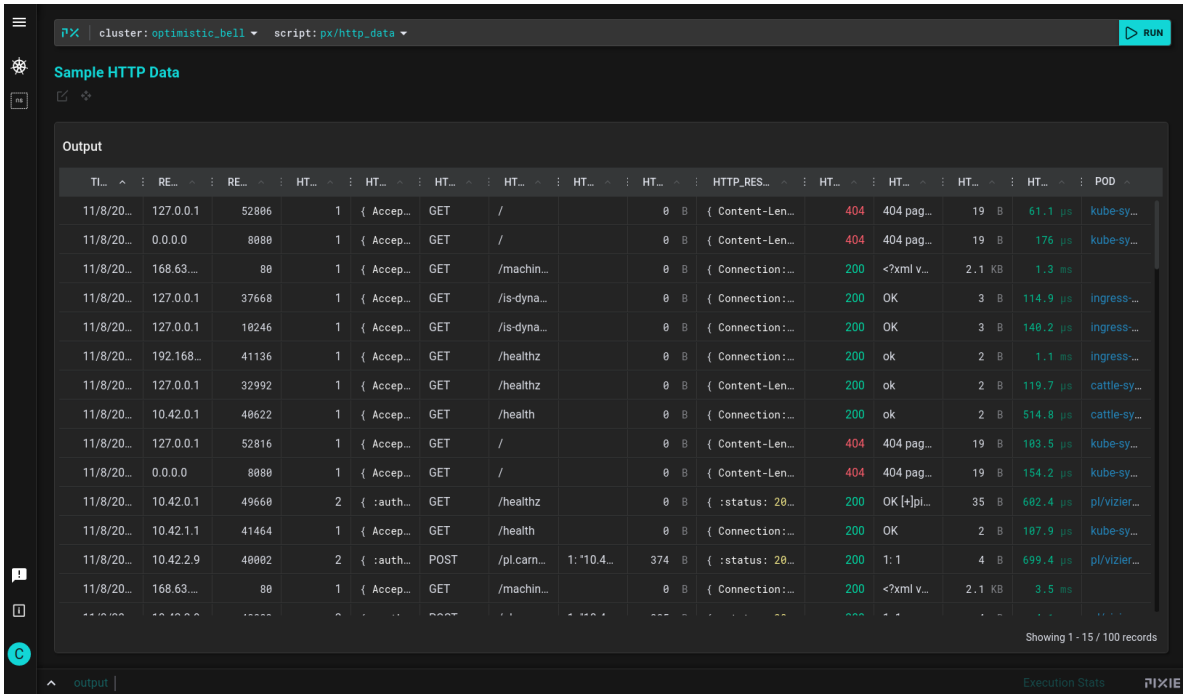
Figure 2 – Pixie Node Stats



The cluster has the standard three-node layout, and we can easily check on the key metrics, CPU usage, disk usage, and network traffic.

For a typical micro-service architecture, most application traffic will be based on REST, so the next important metric to look at is the HTTP data:

Figure 3 – Pixie Http Data

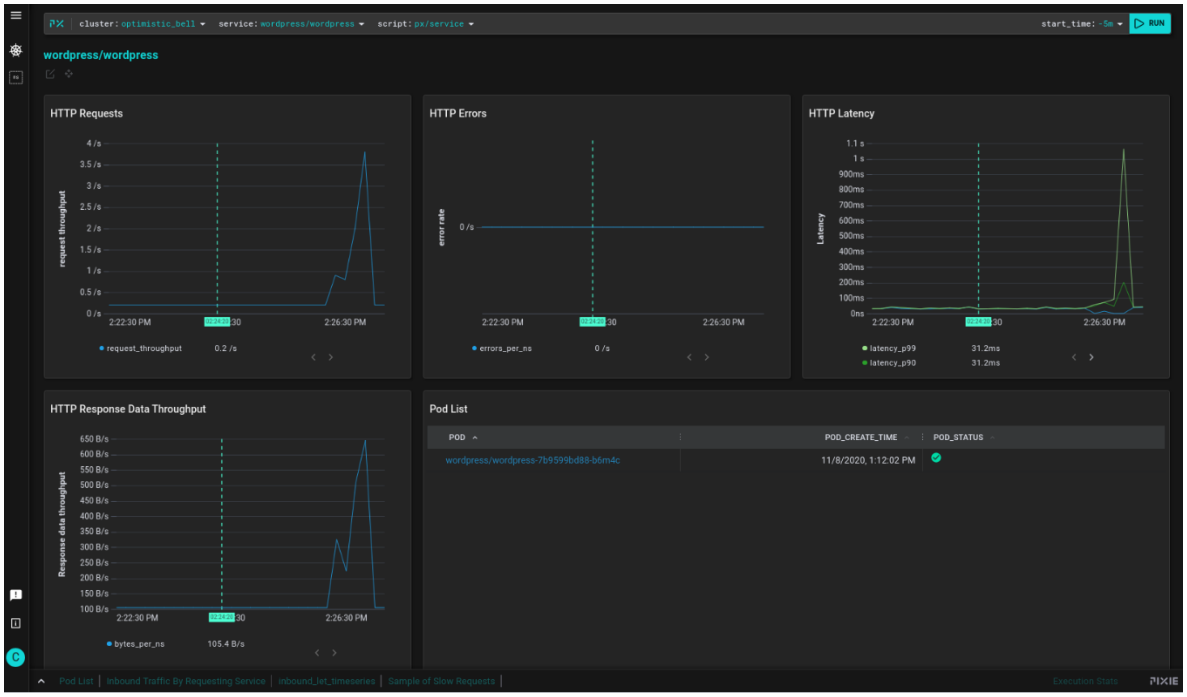


Without any instrumentation, we can easily see platform wide GET and POST operations, together with their results.

4.3 Sample application telemetry data

To look at sample application, we use the “Hello World” of Kubernetes and install a vanilla WordPress instance from its official Helm chart. Again, without any instrumentation, we immediately get a display on the state of our application:

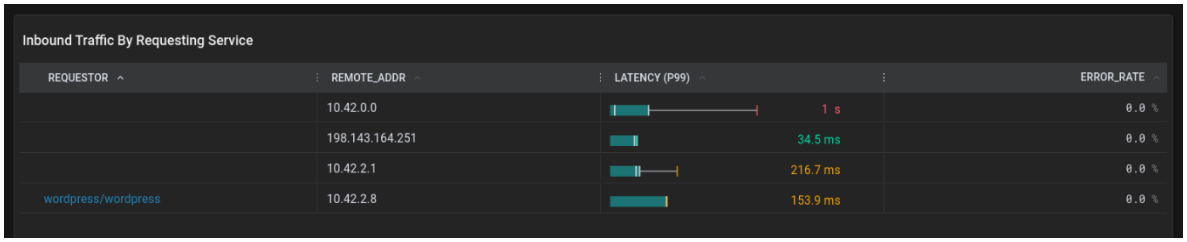
Figure 4 – Wordpress Application Stats



Our application is processing HTTP requests without any error, which in this case, is not entirely unexpected, as it was installed for demonstration purposes only. The key metrics here are Latency and Throughput, good indicators for the state and health of our application.

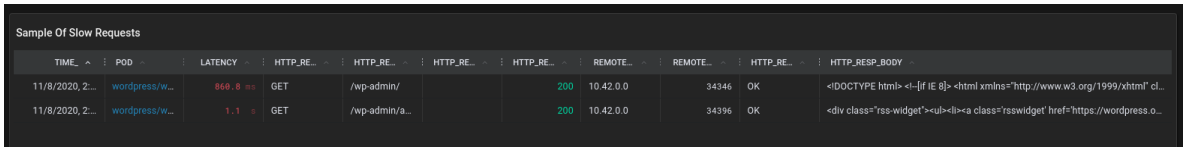
In closing, there are two metrics particularly useful for the state of a distribute application. The first metric is the inbound traffic reaching our application, grouped by source address and latency:

Figure 5 – Wordpress Inbound Traffic



The second metric is showing details about slow requests, which more often than not are an issue for customer support:

Figure 6 – Wordpress Slow Requests



By looking at Request and Response Body, we can help identify the underlying issue by correlating this with the corresponding log or trace messages.

5 Summary

We have seen that is quite important to design your application with observability in mind. Incorporating observability into your application design from the start will greatly benefit operations and troubleshooting in the long run and has been established as good practice in Software Engineering.

If you have the opportunity, install a specialized tool, such as Pixie, to enable operations to troubleshoot your application by having easy access to telemetry and application state information.

We believe that together with the (system) logs, a tool like Pixie can answer almost all questions you might have during troubleshooting.

Happy observing!

References

Campbell, Matt (InfoQ, 2020): AWS Publishes Best Practices Guide for Operational Dashboards <https://www.infoq.com/news/2020/10/aws-dashboards/> (2020-10-31) [Access: 2020-11-14]

Cloudberry Engineering (Cloudberry Engineering, 2020): A Practical Introduction to Container Security <https://cloudberry.engineering/article/practical-introduction-container-security/> (2020-11-01) [Access: 2020-11-14]

Chandrasekaran, Arun (Gartner,2020): Best Practices for Running Containers and Kubernetes in Production <https://www.gartner.com/doc/reprints?id=1-1ZNMNAKC&ct=200811&st=sb> (2020-08-04) [Access: 2020-10-19]

Elasticsearch (Elasticsearch BV, 2020): The heart of the free and open Elastic Stack <https://www.elastic.co/elasticsearch/> (2020) [Access: 2020-11-14]

FireLens (Amazon Web Services, 2020): Custom Log Routing <https://docs.aws.amazon.com/AmazonECS/latest/developerguide/using_firelens.html> (2020) [Access: 2020-11-14]

Fiskeaux, Charlie (The New Stack, 2020): Why Monitoring Kubernetes Is so Challenging and How to Manage It <https://thenewstack.io/why-monitoring-kubernetes-is-so-challenging-and-how-to-manage-it/> (2020-11-06) [Access: 2020-11-14]

FluentBit (Treasure Data, 2020): Cloud Native Log Forwarder <https://fluentbit.io/> (2020) [Access: 2020-11-14]

FluentD (FluentD Project, 2020): Build Your Unified Logging Layer <https://www.fluentd.org/> (2020) [Access: 2020-11-14]

Hahn, Silke (Heise Medien GmbH & Co. KG, 2020): Grafana veranschaulicht Logs mit dem neuem Tool Tempo und mit Loki 2.0 <https://www.heise.de/news/Grafana-veranschaulicht-Logs-mit-dem-neuem-Tool-Tempo-und-mit-Loki-2-0-4939951.html> (2020-10-27) [Access: 2020-11-14]

Icinga (Icinga, 2020): Inspect your Entire Infrastructure <https://icinga.com/> (2020) [Access: 2020-11-14]

Inclusive Naming (Inclusive Naming Initiative, 2020): Word replacement list <https://inclusivenaming.org/word-list/> (2020) [Access: 2020-11-14]

Nagios (Nagios, 2020): The Industry Standard in IT Infrastructure Monitoring <https://www.nagios.com/> (2020) [Access: 2020-11-15]

One Medical (Instagram, 2020): Navigating Pronouns 101 <https://www.instagram.com/p/CBqwZzmhKWQ/?igshid=1lc9nrdjplewq> (2020-06-20) [Access: 2020-11-14]

OpenObservability (OpenObservability, 2020): Evolving the Prometheus exposition format into a standard <https://raw.githubusercontent.com/OpenObservability/OpenMetrics/master/OpenMetrics.md> (2020-11-14) [Access: 2020-11-14]

Parker, Austin and Spoonhower, Daniel (O'Reilly Media, 2020): Distributed Tracing in Practice <https://go.lightstep.com/rs/260-KGM-472/images/distributed-tracing-in-practice-lightstep.pdf> (2020-04-13) [Access: 2020-10-08]

Pixie Labs (Pixie Labs,2020): Instantly troubleshoot your applications on Kubernetes <https://pixielabs.ai/> (2020) [Access: 2020-11-14]

Prometheus (Prometheus, 2020): From metrics to insight <https://prometheus.io/> (2020) [Access: 2020-11-14]

Roper, Jane (WBUR, 2019): They Is Here to Stay. Get Over It. <https://www.wbur.org/cognoscenti/2019/08/13/gender-pronouns-jane-roper> (2019-08-13) [Access: 2020-11-14]

Saguy, Abigail and Williams, Juliet (Scientific American, 2019): Why we should all use they/them pronouns <https://blogs.scientificamerican.com/ voices/why-we-should-all-use-they-them-pronouns/> (2019-04-11) [Access: 2020-05-25]

Splunk (Splunk, 2020): The Data-to-Everything Platform <https://www.splunk.com/> (2020) [Access: 2020-11-14]

Thundra (Amazon Web Services, 2020): Mastering Observability on the Cloud: New Requirements Require a New Approach <https://pages.awscloud.com/GLOBAL-partner-DL-thundra-ebook-2020-learn.html> (2020) [Access: 2020-10-20]

Waterhouse, Peter (The New Stack, 2018): Monitoring and Observability — What is the Difference and Why Does It Matter? <https://thenewstack.io/monitoring-and-observability-whats-the-difference-and-why-does-it-matter/> (2018-04-16) [Access: 2020-11-14]