
telegraf-kafka Documentation

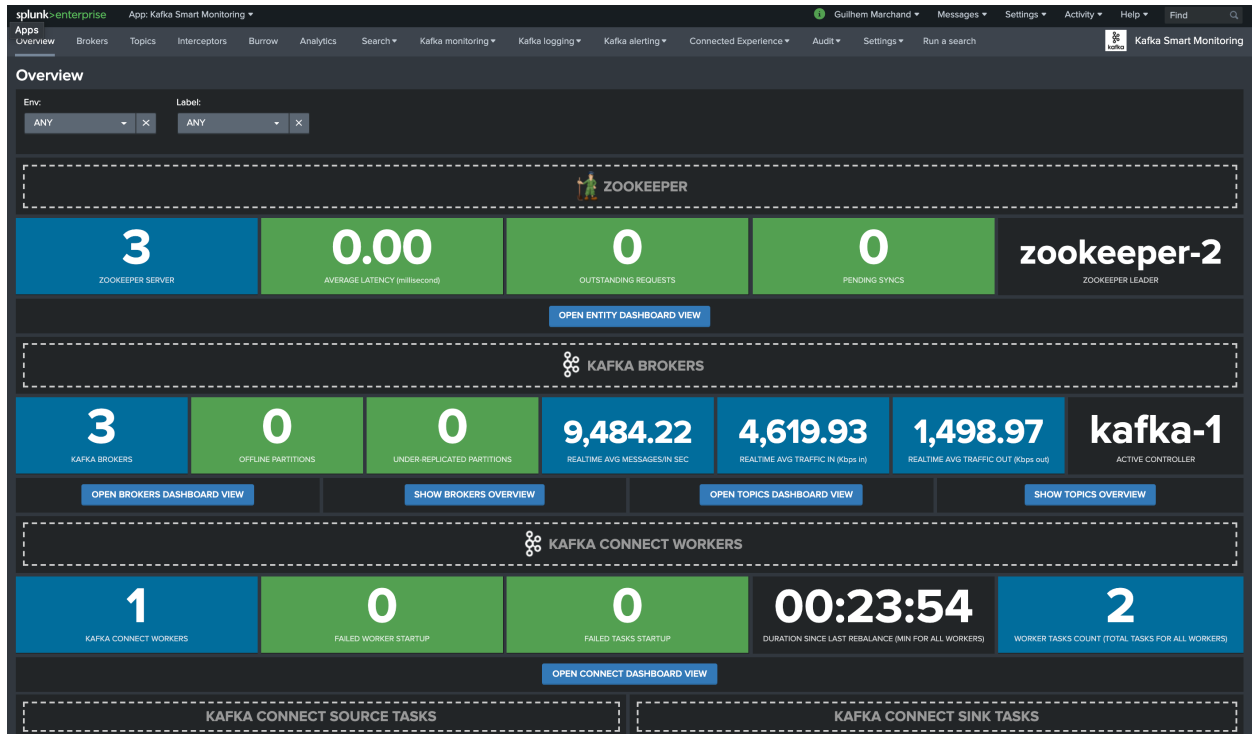
Release 1

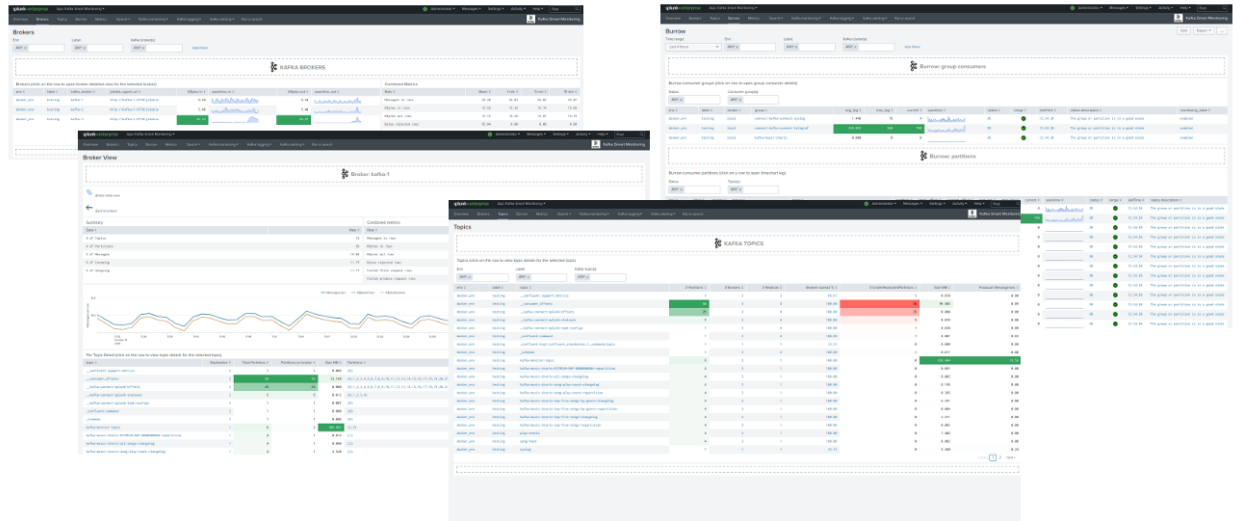
Guilhem Marchand

Nov 08, 2020

1	Overview:	3
1.1	About	3
1.2	Compatibility	5
1.3	Known Issues	5
1.4	Support	5
1.5	Download	6
2	Deployment and configuration:	7
2.1	Deployment & Upgrades	7
2.2	Implementation & data collection	8
2.3	Docker testing templates	45
2.4	Splunk dashboards (health views)	48
2.5	Splunk Connected experience for Cloud Gateway	71
2.6	Kafka infrastructure OOTB alerting	76
3	Troubleshoot:	103
3.1	Troubleshoot & FAQ	103
4	Versioning and build history:	105
4.1	Release notes	105

The Splunk application for Kafka Smart Monitoring with provides performance management, reporting and alerting for Kafka components metrics ingested in the Splunk metric store:





The application provides builtin and native monitoring for Apache Kafka components, as well as the Confluent stack components:

- Zookeeper
- Apache Kafka Brokers
- Apache Kafka Connect
- Confluent schema-registry
- Confluent ksql-server
- Confluent kafka-rest
- Kafka SLA and end to end monitoring with the LinkedIn Kafka monitor
- Confluent Interceptors monitoring for lag monitoring of consumers and producers
- Kafka Consumers lag monitoring with Burrow (Kafka Connect connectors, Kafka Streams...)

Fully multi-tenant compatible, the application can manage different environments, data-centers, etc specially using tags at metrics low level.

It is recommended to read the unified guide for Kafka and Confluent monitoring first:

<https://splunk-guide-for-kafka-monitoring.readthedocs.io>

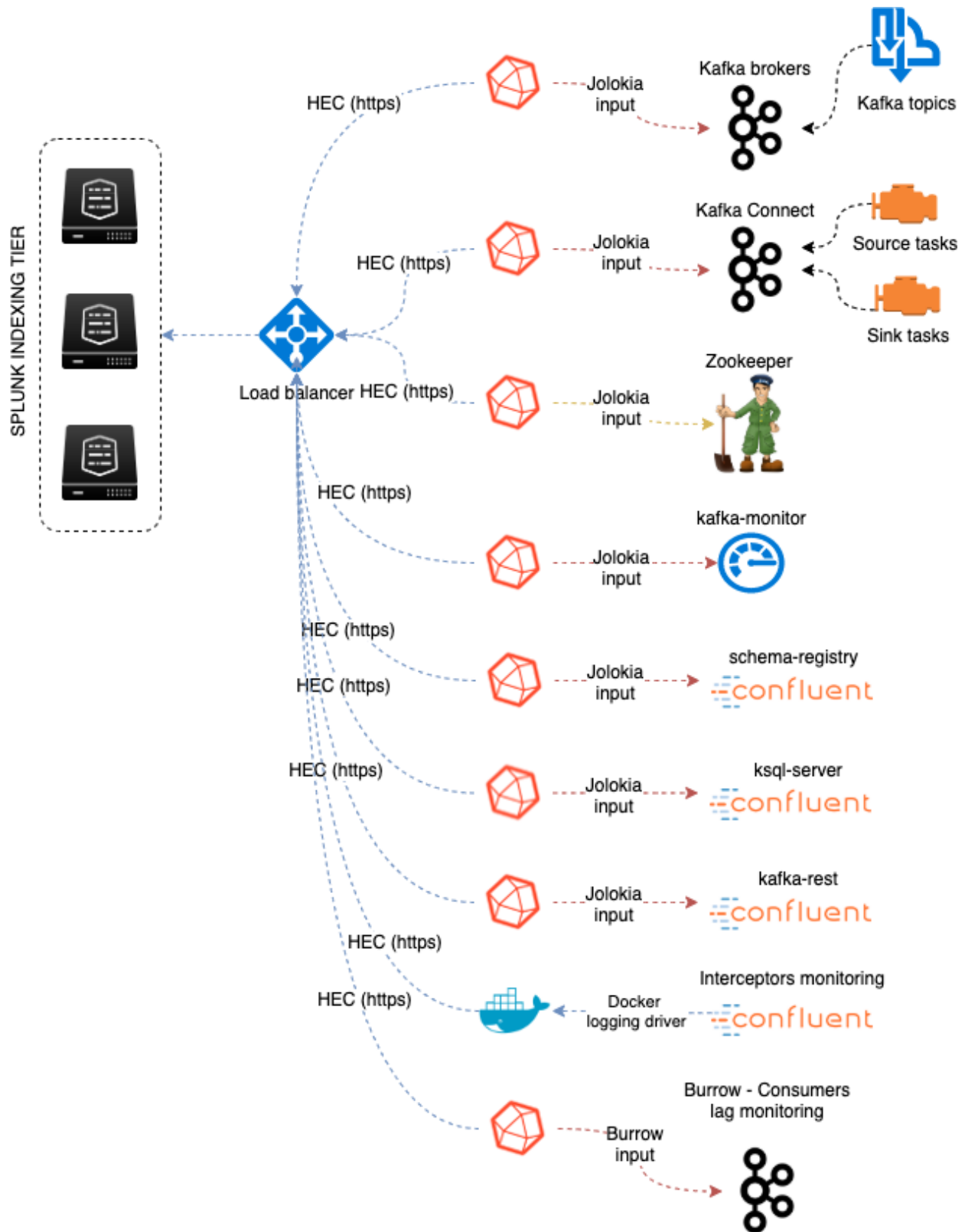
1.1 About

- Author: Guilhem Marchand
- First release published in October 2018
- Purposes:

The Splunk application for Kafka Smart Monitoring leverages the best components to provide a key layer monitoring for your Kafka infrastructure :

- Telegraf from Influxdata (<https://github.com/influxdata/telegraf>)
- Jolokia for the remote JMX collection over http (<https://jolokia.org>)
- Telegraf Jolokia2 input plugin (<https://github.com/influxdata/telegraf/tree/master/plugins/inputs/jolokia2>)
- Telegraf Zookeeper input plugin (<https://github.com/influxdata/telegraf/tree/master/plugins/inputs/zookeeper>)
- Optional: Xinfra Kafka monitor to provide end to end monitoring (<https://github.com/linkedin/kafka-monitor>)
- Optional: Confluent Interceptors Monitoring (<https://docs.confluent.io/current/control-center/installation/clients.html>)
- Optional: Kafka Consumers lag monitoring with Burrow (<https://github.com/linkedin/Burrow>)

An ITSI module is also available: <https://da-itsi-telegraf-kafka.readthedocs.io>



1.2 Compatibility

1.2.1 Splunk compatibility

Splunk core version

- metrics are ingested into the high performance Splunk metric store, Splunk 7.0.x or later is required
 - some queries are built using the latest syntax for metrics, Splunk 7.2.x or later is recommended
-

1.2.2 Telegraf compatibility

Telegraf supports various operating systems and process architectures including any version of Linux and Windows.

For more information:

- <https://portal.influxdata.com/downloads>

1.2.3 Containers compatibility

If you are running Kafka in containers, you are at the right place, all of the components can natively run in docker.

1.2.4 Kafka and Confluent compatibility

Apache Kafka and Confluent compatibility

- Qualification and certification is made against Kafka V2.x and Confluent V6.x, earlier versions might however work with no issues but are not being tested
-

1.2.5 Web Browser compatibility

The application can be used with any of the supported Web Browser by Splunk:

<https://docs.splunk.com/Documentation/Splunk/latest/Installation/Systemrequirements>

1.3 Known Issues

There are no known issues at the moment.

1.4 Support

The Splunk application for Kafka monitoring with Telegraf is community supported.

To get support, use of one the following options:

1.4.1 Splunk community

Open a question in Splunk Community:

- <https://community.splunk.com>

1.4.2 Splunk community Slack

Contact me on Splunk community slack, or even better, ask the community !

- <https://splunk-usergroups.slack.com>

1.4.3 Open a issue in Git

To report an issue, request a feature change or improvement, please open an issue in Github:

- <https://github.com/guilhemmarchand/telegraf-kafka/issues>

1.4.4 Email support

- guilhem.marchand@gmail.com

However, previous options are far better, and will give you all the chances to get a quick support from the community of fellow Splunkers.

1.5 Download

1.5.1 Splunk Application for Kafka monitoring with Telegraf

The Splunk application can be downloaded from:

Splunk base

- <https://splunkbase.splunk.com/app/4268>

GitHub

- <https://github.com/guilhemmarchand/telegraf-kafka>

Deployment and configuration:

2.1 Deployment & Upgrades

2.1.1 Deployment matrix

Splunk roles	required
Search head	yes
Indexer tiers	no

If Splunk search heads are running in Search Head Cluster (SHC), the Splunk application must be deployed by the SHC deployer.

2.1.2 Indexes creation

indexes

- Kafka SDM expects the creation of a metric index, by default `telegraf_kafka` which can be configured by customizing the macro `telegraf_kafka_index`
 - If you use Confluent interceptors, the application expects the creation of a metric index `confluent_interceptor_metrics` which can be configured by customizing the macro `confluent_interceptor_index`
-

2.1.3 Dependencies

The application depends on:

- Horseshoe Meter - Custom Visualization, Splunk Base: <https://splunkbase.splunk.com/app/3166>

2.1.4 Initial deployment

The deployment of the Splunk application for Kafka monitoring with Telegraf is straight forward:

- Using the application manager in Splunk Web (Settings / Manages apps)
- Extracting the content of the tgz archive in the “apps” directory of Splunk
- For SHC configurations (Search Head Cluster), extract the tgz content in the SHC deployer and publish the SHC bundle

2.1.5 Upgrades

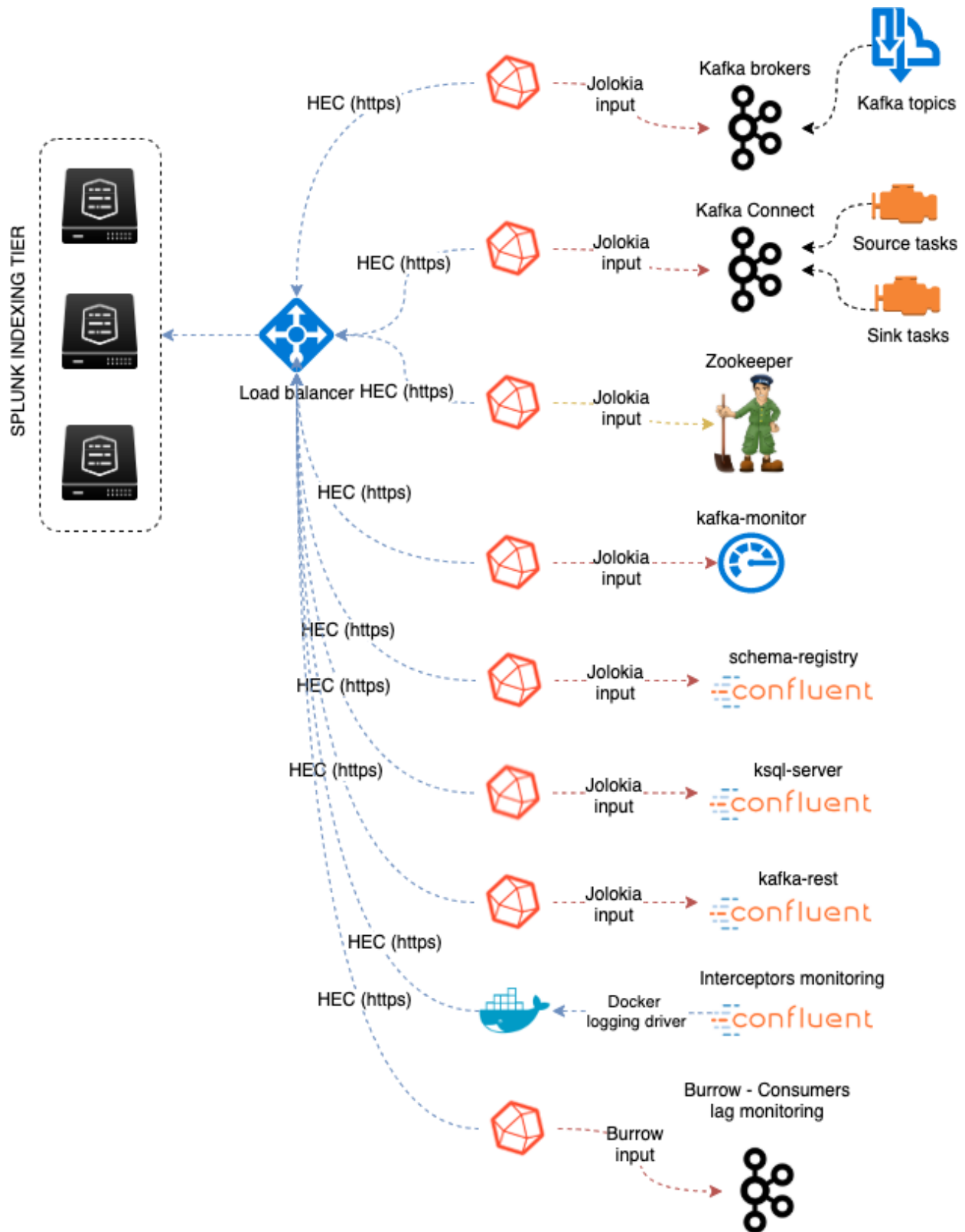
Upgrading the Splunk application is pretty much the same operation than the initial deployment.

2.1.6 Upgrades of the components

Upgrading the different components (Telegraf, Jolokia, etc.) rely on each of the technologies, please consult the deployment main pages.

2.2 Implementation & data collection

Data collection diagram overview:



2.2.1 Splunk configuration

Index definition

indexes

- Kafka SDM expects the creation of a metric index, by default `telegraf_kafka` which can be configured by customizing the macro `telegraf_kafka_index`

indexes.conf example with no Splunk volume::

```
[telegraf_kafka]
coldPath = $SPLUNK_DB/telegraf_kafka/coldddb
datatype = metric
homePath = $SPLUNK_DB/telegraf_kafka/db
thawedPath = $SPLUNK_DB/telegraf_kafka/thaweddb
```

indexes.conf example with Splunk volumes::

```
[telegraf_kafka]
coldPath = volume:cold/telegraf_kafka/coldddb
datatype = metric
homePath = volume:primary/telegraf_kafka/db
thawedPath = $SPLUNK_DB/telegraf_kafka/thaweddb
```

In a Splunk distributed configuration (cluster of indexers), this configuration stands on the cluster master node.

All Splunk searches included in the added refer to the utilisation of a macro called `telegraf_kafka_index` configured in:

- `telegraf-kafka/default/macros.conf`

If you wish to use a different index model, this macro shall be customized to override the default model.

Confluent Interceptors monitoring:

indexes

- If you use Confluent interceptors, the application expects the creation of a metric index `confluent_interceptor_metrics` which can be configured by customizing the macro `confluent_interceptor_index`

indexes.conf example with no Splunk volume::

```
[confluent_interceptor_index]
coldPath = $SPLUNK_DB/confluent_interceptor_index/coldddb
datatype = metric
homePath = $SPLUNK_DB/confluent_interceptor_index/db
thawedPath = $SPLUNK_DB/confluent_interceptor_index/thaweddb
```

indexes.conf example with Splunk volumes::

```
[confluent_interceptor_index]
coldPath = volume:cold/confluent_interceptor_index/coldddb
datatype = metric
homePath = volume:primary/confluent_interceptor_index/db
thawedPath = $SPLUNK_DB/confluent_interceptor_index/thaweddb
```

You can technically use the same index than for telegraf based metrics, or any index of your choice, if so update the macro called **confluent_interceptor_index** configured in:

- telegraf-kafka/default/macros.conf

Role membership

The application creates a builtin Splunk role called “kafka_admin” that provides:

- write permissions to the application name space
 - write permissions to the various KVstore based lookups used for configuration purposes of the application
 - can be used to automatically notify the Kafka administrators if you use Splunk Cloud Gateway and Splunk Mobile Connected Experience
-

We suggest that you configure the Kafka administrators to be member of this role. (by user configuration, role mapping or inheritance)

HEC input ingestion and definition

HTTP Event Collector

- The default recommended way of ingesting the Kafka metrics is using the HTTP Events Collector method which requires the creation of an HEC token
-

inputs.conf example:

```
[http://kafka_monitoring]
disabled = 0
index = telegraf_kafka
indexes = telegraf_kafka
token = 205d43f1-2a31-4e60-a8b3-327eda49944a
```

If you create the HEC input via Splunk Web interface, it is not required to select an explicit value for source and sourcetype.

If you plan to use Confluent Interceptors monitoring, you need to allow the target index too, for instance:

```
[http://kafka_monitoring]
disabled = 0
index = telegraf_kafka
indexes = telegraf_kafka,confluent_interceptor_index
token = 205d43f1-2a31-4e60-a8b3-327eda49944a
```

The HEC input will be ideally relying on a load balancer to provides resiliency and load balancing across your HEC input nodes.

Other ingesting methods

There are other methods possible to ingest the Kafka metrics in Splunk:

- TCP input (graphite format with tags support)

- KAFKA ingestion (Kafka destination from Telegraf in graphite format with tags support, and Splunk connect for Kafka)
- File monitoring with standard Splunk input monitors (file output plugin from Telegraf)

Notes: In the very specific context of monitoring Kafka, it is not a good design to use Kafka as the ingestion method since you will most likely never be able to know when an issue happens on Kafka.

These methods require the deployment of an additional Technology addon: <https://splunkbase.splunk.com/app/4193>

These methods are heavily described here: <https://da-itsi-telegraf-os.readthedocs.io/en/latest/telegraf.html>

These methods should however be considered as a second choice only if sending to HEC is not possible.

2.2.2 Telegraf installation and configuration

Telegraf installation, configuration and start

If you are running Telegraf as a regular process in machine, the standard installation of Telegraf is really straightforward, consult:

- <https://github.com/influxdata/telegraf>

If you have a Splunk Universal Forwarder deployment, you can deploy, run and maintain Telegraf and its configuration through a Splunk application (TA), consult:

- <https://da-itsi-telegraf-os.readthedocs.io/en/latest/telegraf.html#telegraf-deployment-as-splunk-application-deployed-by-splunk->

An example of a ready to use TA application can be found here:

- <https://github.com/guilhemmarchand/TA-telegraf-amd64>

For Splunk customers, this solution has various advantages as you can deploy and maintain using your existing Splunk infrastructure.

Telegraf is extremely container friendly, a container approach is very convenient as you can easily run multiple Telegraf containers to monitor each of the Kafka infrastructure components:

- https://hub.docker.com/r/_/telegraf/

Data collection environment design:

The most scalable and highly available design in term of where placing the Telegraf instances is to deploy Telegraf locally on each server to be monitored (and collect locally the component) or running as a side car container for Kubernetes based environments.

It is possible to collect multiple instances of multiple components via a unique Telegraf instance, however there will be a limit where issues can start, and this design will not provide high availability as the failure of this instance will impact the whole metric collection.

2.2.3 Telegraf configuration generator

The application provides a builtin user interface you can use to generate a telegraf.conf configuration file based on your parameters and for all the components to be monitored:

- Menu Settings / Telegraf Configuration Generator

Configuration helper - generate telegraf sample configuration
Use this configuration helper to create your Telegraf configuration file

Define the value of the env tag: (used as the main identifier for a given environment)
kafka_prod

Define the value of the label tag: (additional environment informational selector)
my_datacenter

Enter the Splunk HEC URL in the form: https://<address of your HEC endpoint>:<port>/services/collector
https://hec.mydomain.com:8088/services/collector

Enter the Splunk HEC token
205d43f1-2a31-4e60-a8b3-327eda49944a

Select the component you want the configuration for:

Component: Kafka Broker X Running on: localhost Port number: 8778

Main Telegraf options:
Interval: 10s Linux OS metrics: Yes

[Generate sample configuration](#)

Telegraf config helper

Generated telegraf.conf:
Use the following content as the definition for your telegraf.conf to monitor and collect the metrics of the component.

```
[global_tags]
# The env tag is used by the application for multi-environments management
env = "kafka_prod"
# The label tag is an additional tag used by the application to identify an environment
label = "my_datacenter"

[agent]
interval = "10s"
flush_interval = "10s"
hostname = "HOSTNAME"

# outputs
[[outputs.http]]
url = "https://hec.mydomain.com:8088/services/collector"
insecure_skip_verify = true
data_format = "splunkmetric"
# Provides time, index, source overrides for the HEC
splunkmetrichec_routing = true
# Additional HTTP headers
[outputs.http.headers]
# Should be set according to the destination/agent for some data format
```

[Download configuration](#)

2.2.4 Telegraf output configuration

Whether you will be running Telegraf in various containers, or installed as a regular software within the different servers composing your Kafka infrastructure, a minimal configuration is required to teach Telegraf how to forward the metrics to your Splunk deployment.

Telegraf is able to send data to Splunk in different ways:

- Splunk HTTP Events Collector (HEC) - Since Telegraf v1.8
- Splunk TCP inputs in Graphite format with tags support and the TA for Telegraf
- Apache Kafka topic in Graphite format with tags support and the TA for Telegraf and Splunk connect for Kafka

Who watches for the watcher?

As you are running a Kafka deployment, it would seem very logical to produce metrics to a Kafka topic and consume these metrics from the topic. However, this is not an ideal monitoring architecture design due to potential lack of visibility in case of an outage or issues.

If you use this same system for monitoring Kafka itself, it is very likely that you will never know when Kafka is broken because the data flow for your monitoring system will be broken as well.

The recommendation is to rely either on Splunk HEC or TCP inputs to forward Telegraf metrics data for the Kafka monitoring.

A minimal configuration for telegraf.conf, running in container or as a regular process in machine and forwarding to HEC:

```
[global_tags]
# the env tag is used by the application for multi-environments management
env = "my_env"
# the label tag is an optional tag used by the application that you can use as
↳ additional label for the services or infrastructure
label = "my_env_label"

[agent]
interval = "10s"
flush_interval = "10s"
hostname = "$HOSTNAME"

# Regular OS monitoring for Linux OS

# Read metrics about cpu usage
[[inputs.cpu]]
  ## Whether to report per-cpu stats or not
  percpu = true
  ## Whether to report total system cpu stats or not
  totalcpu = true
  ## If true, collect raw CPU time metrics.
  collect_cpu_time = false
  ## If true, compute and report the sum of all non-idle CPU states.
  report_active = false

# Read metrics about disk usage by mount point
[[inputs.disk]]

  ## Ignore mount points by filesystem type.
  ignore_fs = ["tmpfs", "devtmpfs", "devfs"]

# Read metrics about disk IO by device
[[inputs.diskio]]

# Get kernel statistics from /proc/stat
[[inputs.kernel]]

# Read metrics about memory usage
[[inputs.mem]]

# Get the number of processes and group them by status
[[inputs.processes]]

# Read metrics about swap memory usage
[[inputs.swap]]

# Read metrics about system load & uptime
[[inputs.system]]

# # Read metrics about network interface usage
[[inputs.net]]

# # Read TCP metrics such as established, time wait and sockets counts.
[[inputs.netstat]]
```

(continues on next page)

(continued from previous page)

```
# # Monitor process cpu and memory usage
[[inputs.procstat]]
    pattern = ".*"

# outputs
[[outputs.http]]
    url = "https://splunk:8088/services/collector"
    insecure_skip_verify = true
    data_format = "splunkmetric"
    ## Provides time, index, source overrides for the HEC
    splunkmetrichec_routing = true
    ## Additional HTTP headers
    [outputs.http.headers]
    # Should be set manually to "application/json" for json data_format
    Content-Type = "application/json"
    Authorization = "Splunk 205d43f1-2a31-4e60-a8b3-327eda49944a"
    X-Splunk-Request-Channel = "205d43f1-2a31-4e60-a8b3-327eda49944a"
```

If for some reasons, you have to use either of the 2 other solutions, please consult:

- <https://da-itsi-telegraf-os.readthedocs.io/en/latest/telegraf.html>

Notes: The configuration above provides out of the box OS monitoring for the hosts, which can be used by the Operating System monitoring application for Splunk:

<https://splunkbase.splunk.com/app/4271/>

2.2.5 Jolokia JVM monitoring



The following Kafka components require Jolokia to be deployed and started, as the modern and efficient interface to JMX that is collected by Telegraf:

- Zookeeper
- Apache Kafka Brokers
- Apache Kafka Connect
- Confluent schema-registry
- Confluent ksql-server
- Confluent kafka-rest

For the complete documentation of Jolokia, see:

- <https://jolokia.org>

Jolokia JVM agent can be started in 2 ways, either as using the `-javaagent` argument during the start of the JVM, or on the fly by attaching Jolokia to the PID of the JVM:

- <https://jolokia.org/reference/html/agents.html#agents-jvm>

2.2.6 Starting Jolokia with the JVM

To start Jolokia agent using the `-javaagent` argument, use such option at the start of the JVM:

```
-javaagent:/opt/jolokia/jolokia.jar=port=8778,host=0.0.0.0
```

Note: This method is the method used in the docker example within this documentation by using the environment variables of the container.

When running on dedicated servers or virtual machines, update the relevant systemd configuration file to start Jolokia automatically:

For Zookeeper

For bare-metals and dedicated VMs:

- Edit: `/lib/systemd/system/confluent-zookeeper.service`
- Add `-javaagent` argument:

```
[Unit]
Description=Apache Kafka - ZooKeeper
Documentation=http://docs.confluent.io/
After=network.target

[Service]
Type=simple
User=cp-kafka
Group=confluent
ExecStart=/usr/bin/zookeeper-server-start /etc/kafka/zookeeper.properties
Environment="KAFKA_OPTS=-javaagent:/opt/jolokia/jolokia.jar=port=8778,host=0.0.0.0"
Environment="LOG_DIR=/var/log/zookeeper"
TimeoutStopSec=180
Restart=no

[Install]
WantedBy=multi-user.target
```

- Reload systemd and restart:

```
sudo systemctl daemon-restart
sudo systemctl restart confluent-zookeeper
```

For container based environments:

Define the following environment variable when starting the containers:

```
KAFKA_OPTS: "-javaagent:/opt/jolokia/jolokia.jar=port=8778,host=0.0.0.0"
```

For Kafka brokers

For bare-metals and dedicated VMs:

- Edit: `/lib/systemd/system/confluent-kafka.service`
- Add `-javaagent` argument:


```
[Unit]
Description=Apache Kafka - broker
Documentation=http://docs.confluent.io/
After=network.target confluent-zookeeper.target

[Service]
Type=simple
User=cp-kafka
Group=confluent
ExecStart=/usr/bin/kafka-server-start /etc/kafka/server.properties
Environment="KAFKA_OPTS=-javaagent:/opt/jolokia/jolokia.jar=port=8778,host=0.0.0.0"
TimeoutStopSec=180
Restart=no

[Install]
WantedBy=multi-user.target
```

- Reload systemd and restart:

```
sudo systemctl daemon-restart
sudo systemctl restart confluent-kafka
```

For container based environments:

Define the following environment variable when starting the containers:

```
KAFKA_OPTS: "-javaagent:/opt/jolokia/jolokia.jar=port=8778,host=0.0.0.0"
```

For Kafka Connect

For bare-metals and dedicated VMs:

- Edit: `/lib/systemd/system/confluent-kafka-connect.service`
- Add `-javaagent` argument:

```
[Unit]
Description=Apache Kafka Connect - distributed
Documentation=http://docs.confluent.io/
After=network.target confluent-kafka.target

[Service]
Type=simple
User=cp-kafka-connect
Group=confluent
ExecStart=/usr/bin/connect-distributed /etc/kafka/connect-distributed.properties
Environment="KAFKA_OPTS=-javaagent:/opt/jolokia/jolokia.jar=port=8778,host=0.0.0.0"
Environment="LOG_DIR=/var/log/connect"
TimeoutStopSec=180
Restart=no

[Install]
WantedBy=multi-user.target
```

- Reload systemd and restart:

```
sudo systemctl daemon-restart
sudo systemctl restart confluent-kafka-connect
```

For container based environments:

Define the following environment variable when starting the containers:

```
KAFKA_OPTS: "-javaagent:/opt/jolokia/jolokia.jar=port=8778,host=0.0.0.0"
```

For Confluent schema-registry

For bare-metals and dedicated VMs:

- Edit: `/lib/systemd/system/confluent-schema-registry.service`
- Add `-javaagent` argument:

```
[Unit]
Description=RESTful Avro schema registry for Apache Kafka
Documentation=http://docs.confluent.io/
After=network.target confluent-kafka.target

[Service]
Type=simple
User=cp-schema-registry
Group=confluent
Environment="LOG_DIR=/var/log/confluent/schema-registry"
Environment="SCHEMA_REGISTRY_OPTS=-javaagent:/opt/jolokia/jolokia.jar=port=8778,
↳host=0.0.0.0"
ExecStart=/usr/bin/schema-registry-start /etc/schema-registry/schema-registry.
↳properties
TimeoutStopSec=180
Restart=no

[Install]
WantedBy=multi-user.target
```

- Reload systemd and restart:

```
sudo systemctl daemon-restart
sudo systemctl restart confluent-schema-registry
```

For container based environments:

Define the following environment variable when starting the containers:

```
SCHEMA_REGISTRY_OPTS: "-javaagent:/opt/jolokia/jolokia.jar=port=8778,host=0.0.0.0"
```

For Confluent ksql-server

For bare-metals and dedicated VMs:

- Edit: `/lib/systemd/system/confluent-ksqldb.service`
- Add `-javaagent` argument:

```

Description=Streaming SQL engine for Apache Kafka
Documentation=http://docs.confluent.io/
After=network.target confluent-kafka.target confluent-schema-registry.target

[Service]
Type=simple
User=cp-ksql
Group=confluent
Environment="LOG_DIR=/var/log/confluent/ksql"
Environment="KSQL_OPTS=-javaagent:/opt/jolokia/jolokia.jar=port=8778,host=0.0.0.0"
ExecStart=/usr/bin/ksql-server-start /etc/ksql/ksql-server.properties
TimeoutStopSec=180
Restart=no

[Install]
WantedBy=multi-user.target

```

- Reload systemd and restart:

```

sudo systemctl daemon-restart
sudo systemctl restart confluent-ksql

```

For container based environments:

Define the following environment variable when starting the containers:

```
KSQL_OPTS: "-javaagent:/opt/jolokia/jolokia.jar=port=8778,host=0.0.0.0"
```

For Confluent kafka-rest

For bare-metals and dedicated VMs:

- Edit: /lib/systemd/system/confluent-kafka-rest.service
- Add `-javaagent` argument:

```

[Unit]
Description=A REST proxy for Apache Kafka
Documentation=http://docs.confluent.io/
After=network.target confluent-kafka.target

[Service]
Type=simple
User=cp-kafka-rest
Group=confluent
Environment="LOG_DIR=/var/log/confluent/kafka-rest"
Environment="KAFKAREST_OPTS=-javaagent:/opt/jolokia/jolokia.jar=port=8778,host=0.0.0.0"
↪

ExecStart=/usr/bin/kafka-rest-start /etc/kafka-rest/kafka-rest.properties
TimeoutStopSec=180
Restart=no

[Install]
WantedBy=multi-user.target

```

- Reload systemd and restart:

```
sudo systemctl daemon-restart
sudo systemctl restart confluent-kafka-rest
```

For container based environments:

Define the following environment variable when starting the containers:

```
KAFKAREST_OPTS: "-javaagent:/opt/jolokia/jolokia.jar=port=8778,host=0.0.0.0"
```

Notes: “KAFKAREST_OPTS” is not a typo, this is the real name of the environment variable for some reason.

2.2.7 Starting Jolokia on the fly

To attach Jolokia agent to an existing JVM, identify its process ID (PID), simplistic example:

```
ps -ef | grep 'kafka.properties' | grep -v grep | awk '{print $1}'
```

Then:

```
java -jar /opt/jolokia/jolokia.jar --host 0.0.0.0 --port 8778 start <PID>
```

Add this operation to any custom init scripts you use to start the Kafka components.

2.2.8 Zookeeper monitoring

Since the v1.1.31, Zookeeper metrics are now collected via JMX and Jolokia rather than the Telegraf Zookeeper plugin.

Collecting with Telegraf

Depending on how you run Kafka and your architecture preferences, you may prefer to collect all the brokers metrics from one Telegraf collector, or installed locally on the Kafka broker machine.

Connecting to multiple remote Jolokia instances:

```
[[inputs.jolokia2_agent]]
  name_prefix = "zk_"
  urls = ["http://zookeeper-1:8778/jolokia", "http://zookeeper-2:8778/jolokia", "http://
↪zookeeper-3:8778/jolokia"]
```

Connecting to the local Jolokia instance:

```
# Zookeeper JVM monitoring
[[inputs.jolokia2_agent]]
  name_prefix = "zk_"
  urls = ["http://$HOSTNAME:8778/jolokia"]
```

Full telegraf.conf example

The following telegraf.conf collects a cluster of 3 Zookeeper nodes:

```

[global_tags]
  # the env tag is used by the application for multi-environments management
  env = "my_env"
  # the label tag is an optional tag used by the application that you can use as
  ↪ additional label for the services or infrastructure
  label = "my_env_label"

[agent]
  interval = "10s"
  flush_interval = "10s"
  hostname = "$HOSTNAME"

# outputs
[[outputs.http]]
  url = "https://splunk:8088/services/collector"
  insecure_skip_verify = true
  data_format = "splunkmetric"
  ## Provides time, index, source overrides for the HEC
  splunkmetrichec_routing = true
  ## Additional HTTP headers
  [outputs.http.headers]
  # Should be set manually to "application/json" for json data_format
  Content-Type = "application/json"
  Authorization = "Splunk 205d43f1-2a31-4e60-a8b3-327eda49944a"
  X-Splunk-Request-Channel = "205d43f1-2a31-4e60-a8b3-327eda49944a"

# Zookeeper JMX collection

[[inputs.jolokia2_agent]]
  name_prefix = "zk_"
  urls = ["http://zookeeper-1:8778/jolokia", "http://zookeeper-2:8778/jolokia", "http://
  ↪ zookeeper-3:8778/jolokia"]

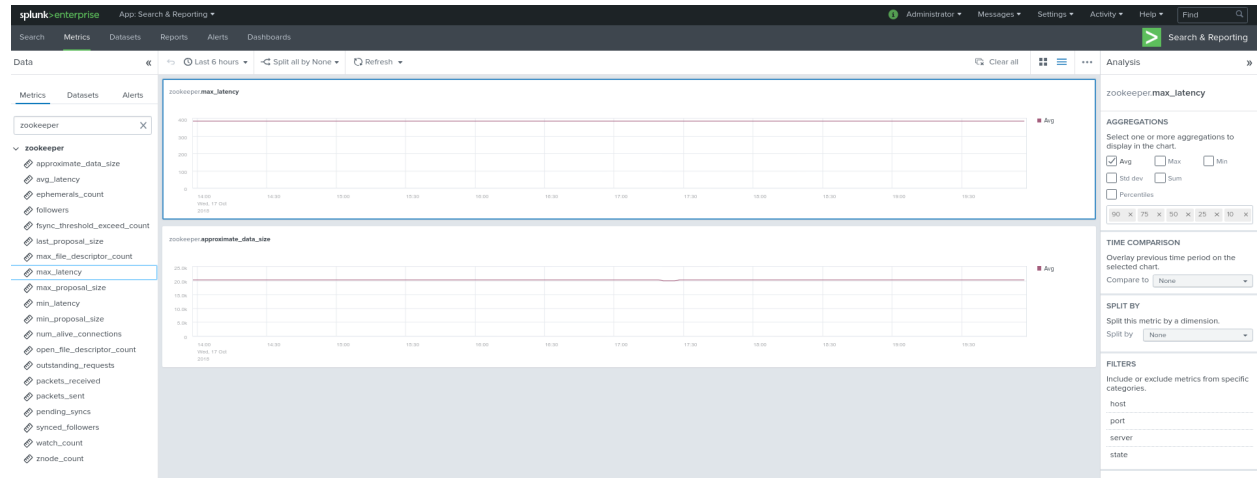
[[inputs.jolokia2_agent.metric]]
  name = "quorum"
  mbean = "org.apache.ZooKeeperService:name0=*"
  tag_keys = ["name0"]

[[inputs.jolokia2_agent.metric]]
  name = "leader"
  mbean = "org.apache.ZooKeeperService:name0=*,name1=*,name2=Leader"
  tag_keys = ["name1"]

[[inputs.jolokia2_agent.metric]]
  name = "follower"
  mbean = "org.apache.ZooKeeperService:name0=*,name1=*,name2=Follower"
  tag_keys = ["name1"]

```

Visualization of metrics within the Splunk metrics workspace application:



Using mcatalog search command to verify data availability:

```
| mcatalog values(metric_name) values(_dims) where index=* metric_name=zk_*
```

2.2.9 Kafka brokers monitoring with Jolokia

Collecting with Telegraf

Depending on how you run Kafka and your architecture preferences, you may prefer to collect all the brokers metrics from one Telegraf collector, or installed locally on the Kafka broker machine.

Connecting to multiple remote Jolokia instances:

```
# Kafka JVM monitoring
[[inputs.jolokia2_agent]]
  name_prefix = "kafka_"
  urls = ["http://kafka-1:8778/jolokia", "http://kafka-2:8778/jolokia", "http://kafka-3:8778/jolokia"]
```

Connecting to the local Jolokia instance:

```
# Kafka JVM monitoring
[[inputs.jolokia2_agent]]
  name_prefix = "kafka_"
  urls = ["http://$HOSTNAME:8778/jolokia"]
```

Full telegraf.conf example

The following telegraf.conf collects a cluster of 3 Kafka brokers:

```
[global_tags]
  # the env tag is used by the application for multi-environments management
  env = "my_env"
  # the label tag is an optional tag used by the application that you can use as
  # additional label for the services or infrastructure
  label = "my_env_label"

[agent]
```

(continues on next page)

(continued from previous page)

```

interval = "10s"
flush_interval = "10s"
hostname = "$HOSTNAME"

# outputs
[[outputs.http]]
  url = "https://splunk:8088/services/collector"
  insecure_skip_verify = true
  data_format = "splunkmetric"
  ## Provides time, index, source overrides for the HEC
  splunkmetrichec_routing = true
  ## Additional HTTP headers
  [outputs.http.headers]
  # Should be set manually to "application/json" for json data_format
  Content-Type = "application/json"
  Authorization = "Splunk 205d43f1-2a31-4e60-a8b3-327eda49944a"
  X-Splunk-Request-Channel = "205d43f1-2a31-4e60-a8b3-327eda49944a"

# Kafka JVM monitoring

[[inputs.jolokia2_agent]]
  name_prefix = "kafka_"
  urls = ["http://kafka-1:18778/jolokia", "http://kafka-2:28778/jolokia", "http://kafka-
↪3:38778/jolokia"]

[[inputs.jolokia2_agent.metric]]
  name = "controller"
  mbean = "kafka.controller:name=*,type=*"
  field_prefix = "$1."

[[inputs.jolokia2_agent.metric]]
  name = "replica_manager"
  mbean = "kafka.server:name=*,type=ReplicaManager"
  field_prefix = "$1."

[[inputs.jolokia2_agent.metric]]
  name = "purgatory"
  mbean = "kafka.server:delayedOperation=*,name=*,
↪type=DelayedOperationPurgatory"
  field_prefix = "$1."
  field_name = "$2"

[[inputs.jolokia2_agent.metric]]
  name = "client"
  mbean = "kafka.server:client-id=*,type=*"
  tag_keys = ["client-id", "type"]

[[inputs.jolokia2_agent.metric]]
  name = "network"
  mbean = "kafka.network:name=*,request=*,type=RequestMetrics"
  field_prefix = "$1."
  tag_keys = ["request"]

[[inputs.jolokia2_agent.metric]]
  name = "network"
  mbean = "kafka.network:name=ResponseQueueSize,type=RequestChannel"
  field_prefix = "ResponseQueueSize"

```

(continues on next page)

(continued from previous page)

```

tag_keys      = ["name"]

[[inputs.jolokia2_agent.metric]]
name          = "network"
mbean         = "kafka.network:name=NetworkProcessorAvgIdlePercent,type=SocketServer"
field_prefix  = "NetworkProcessorAvgIdlePercent"
tag_keys      = ["name"]

[[inputs.jolokia2_agent.metric]]
name          = "topics"
mbean         = "kafka.server:name=*,type=BrokerTopicMetrics"
field_prefix  = "$1."

[[inputs.jolokia2_agent.metric]]
name          = "topic"
mbean         = "kafka.server:name=*,topic=*,type=BrokerTopicMetrics"
field_prefix  = "$1."
tag_keys      = ["topic"]

[[inputs.jolokia2_agent.metric]]
name          = "partition"
mbean         = "kafka.log:name=*,partition=*,topic=*,type=Log"
field_name    = "$1"
tag_keys      = ["topic", "partition"]

[[inputs.jolokia2_agent.metric]]
name          = "log"
mbean         = "kafka.log:name=LogFlushRateAndTimeMs,type=LogFlushStats"
field_name    = "LogFlushRateAndTimeMs"
tag_keys      = ["name"]

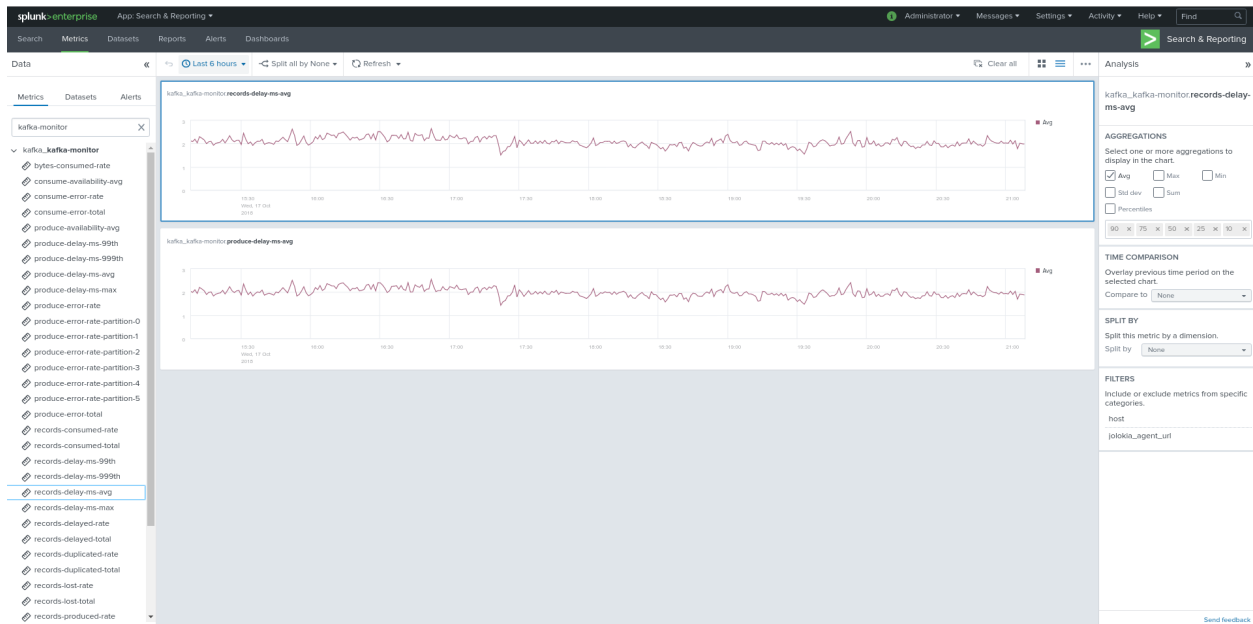
[[inputs.jolokia2_agent.metric]]
name          = "partition"
mbean         = "kafka.cluster:name=UnderReplicated,partition=*,topic=*,type=Partition"
field_name    = "UnderReplicatedPartitions"
tag_keys      = ["topic", "partition"]

[[inputs.jolokia2_agent.metric]]
name          = "request_handlers"
mbean         = "kafka.server:name=RequestHandlerAvgIdlePercent,
↪type=KafkaRequestHandlerPool"
tag_keys      = ["name"]

# JVM garbage collector monitoring
[[inputs.jolokia2_agent.metric]]
name          = "jvm_garbage_collector"
mbean         = "java.lang:name=*,type=GarbageCollector"
paths         = ["CollectionTime", "CollectionCount", "LastGcInfo"]
tag_keys      = ["name"]

```

Visualization of metrics within the Splunk metrics workspace application:



Using mcatalog search command to verify data availability:

```
| mcatalog values(metric_name) values(_dims) where index=* metric_name=kafka_*.*
```

2.2.10 Kafka connect monitoring

Collecting with Telegraf

Connecting to multiple remote Jolokia instances:

```
# Kafka-connect JVM monitoring
[[inputs.jolokia2_agent]]
  name_prefix = "kafka_connect."
  urls = ["http://kafka-connect-1:18779/jolokia","http://kafka-connect-2:28779/jolokia",
↪ "http://kafka-connect-3:38779/jolokia"]
```

Connecting to local Jolokia instance:

```
# Kafka-connect JVM monitoring
[[inputs.jolokia2_agent]]
  name_prefix = "kafka_connect."
  urls = ["http://$HOSTNAME:8778/jolokia"]
```

Full telegraf.conf example

below a full telegraf.conf example:

```
[global_tags]
  # the env tag is used by the application for multi-environments management
  env = "my_env"
  # the label tag is an optional tag used by the application that you can use as ↪
↪ additional label for the services or infrastructure
  label = "my_env_label"
```

(continues on next page)

(continued from previous page)

```

[agent]
  interval = "10s"
  flush_interval = "10s"
  hostname = "$HOSTNAME"

# outputs
[[outputs.http]]
  url = "https://splunk:8088/services/collector"
  insecure_skip_verify = true
  data_format = "splunkmetric"
  ## Provides time, index, source overrides for the HEC
  splunkmetrichec_routing = true
  ## Additional HTTP headers
  [outputs.http.headers]
  # Should be set manually to "application/json" for json data_format
  Content-Type = "application/json"
  Authorization = "Splunk 205d43f1-2a31-4e60-a8b3-327eda49944a"
  X-Splunk-Request-Channel = "205d43f1-2a31-4e60-a8b3-327eda49944a"

# Kafka-connect JVM monitoring

[[inputs.jolokia2_agent]]
  name_prefix = "kafka_connect."
  urls = ["http://kafka-connect-1:18779/jolokia", "http://kafka-connect-2:28779/jolokia", "http://kafka-connect-3:38779/jolokia"]

[[inputs.jolokia2_agent.metric]]
  name = "worker"
  mbean = "kafka.connect:type=connect-worker-metrics"

[[inputs.jolokia2_agent.metric]]
  name = "worker"
  mbean = "kafka.connect:type=connect-worker-rebalance-metrics"

[[inputs.jolokia2_agent.metric]]
  name = "connector-task"
  mbean = "kafka.connect:type=connector-task-metrics,connector=*,task=*"
  tag_keys = ["connector", "task"]

[[inputs.jolokia2_agent.metric]]
  name = "sink-task"
  mbean = "kafka.connect:type=sink-task-metrics,connector=*,task=*"
  tag_keys = ["connector", "task"]

[[inputs.jolokia2_agent.metric]]
  name = "source-task"
  mbean = "kafka.connect:type=source-task-metrics,connector=*,task=*"
  tag_keys = ["connector", "task"]

[[inputs.jolokia2_agent.metric]]
  name = "error-task"
  mbean = "kafka.connect:type=task-error-metrics,connector=*,task=*"
  tag_keys = ["connector", "task"]

# Kafka connect return a status value which is non numerical
# Using the enum processor with the following configuration replaces the string value_
↳by our mapping

```

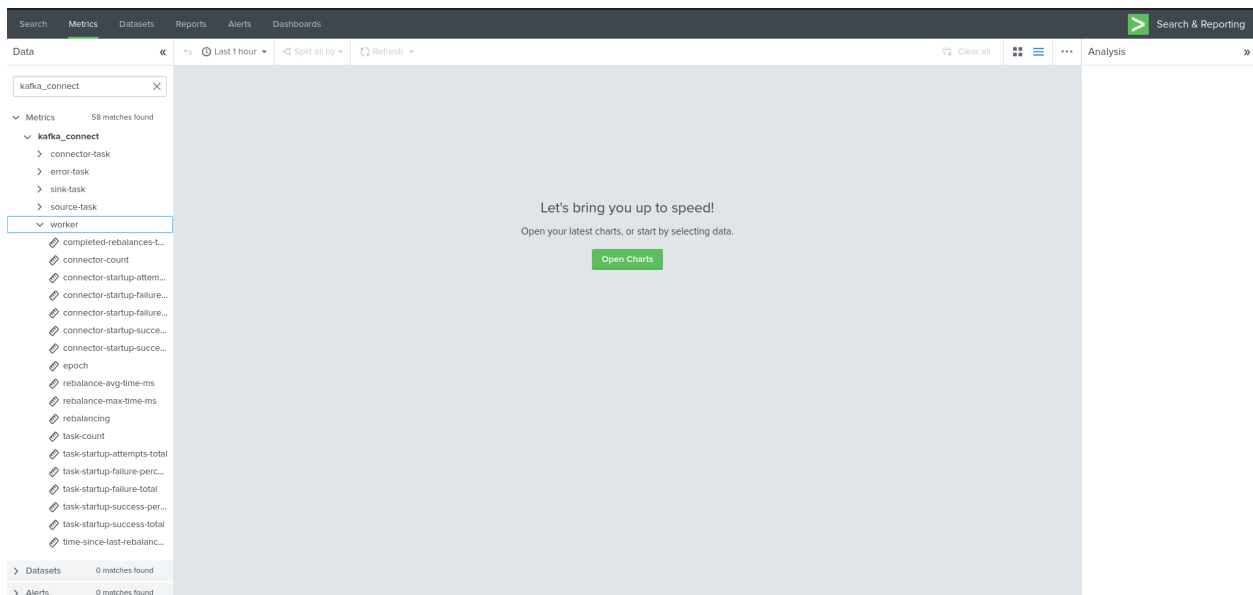
(continues on next page)

(continued from previous page)

```
[processors.enum]
[processors.enum.mapping]
  ## Name of the field to map
  field = "status"

  ## Table of mappings
[processors.enum.mapping.value_mappings]
  paused = 0
  running = 1
  unassigned = 2
  failed = 3
  destroyed = 4
```

Visualization of metrics within the Splunk metrics workspace application:



Using mcatalog search command to verify data availability:

```
| mcatalog values(metric_name) values(_dims) where index=* metric_name=kafka_connect.*
```

2.2.11 Kafka Xinfra monitor - end to end monitoring

Installing and starting the Kafka monitor

LinkedIn provides an extremely powerful open source end to end monitoring solution for Kafka, please consult:

- <https://github.com/linkedin/kafka-monitor>

As a builtin configuration, the kafka-monitor implements a jolokia agent, so collecting the metrics with Telegraf cannot be more easy !

It is very straightforward to run the kafka-monitor in a docker container, first you need to create your own image:

- <https://github.com/linkedin/kafka-monitor/tree/master/docker>

In a nutshell, you would:

```
git clone https://github.com/linkedin/kafka-monitor.git
cd kafka-monitor
./gradlew jar
cd docker
```

Edit the Makefile to match your needs

```
make container
make push
```

Then start your container, example with docker-compose:

```
kafka-monitor:
image: guilhemmarchand/kafka-monitor:2.0.3
hostname: kafka-monitor
volumes:
- ../kafka-monitor:/usr/local/share/kafka-monitor
command: "/opt/kafka-monitor/bin/kafka-monitor-start.sh /usr/local/share/kafka-
↪monitor/kafka-monitor.properties"
```

Once your Kafka monitor is running, you need a Telegraf instance that will be collecting the JMX beans, example:

```
[global_tags]
# the env tag is used by the application for multi-environments management
env = "my_env"
# the label tag is an optional tag used by the application that you can use as
↪additional label for the services or infrastructure
label = "my_env_label"

[agent]
interval = "10s"
flush_interval = "10s"
hostname = "$HOSTNAME"

# outputs
[[outputs.http]]
url = "https://splunk:8088/services/collector"
insecure_skip_verify = true
data_format = "splunkmetric"
## Provides time, index, source overrides for the HEC
splunkmetrichec_routing = true
## Additional HTTP headers
[outputs.http.headers]
# Should be set manually to "application/json" for json data_format
Content-Type = "application/json"
Authorization = "Splunk 205d43f1-2a31-4e60-a8b3-327eda49944a"
X-Splunk-Request-Channel = "205d43f1-2a31-4e60-a8b3-327eda49944a"

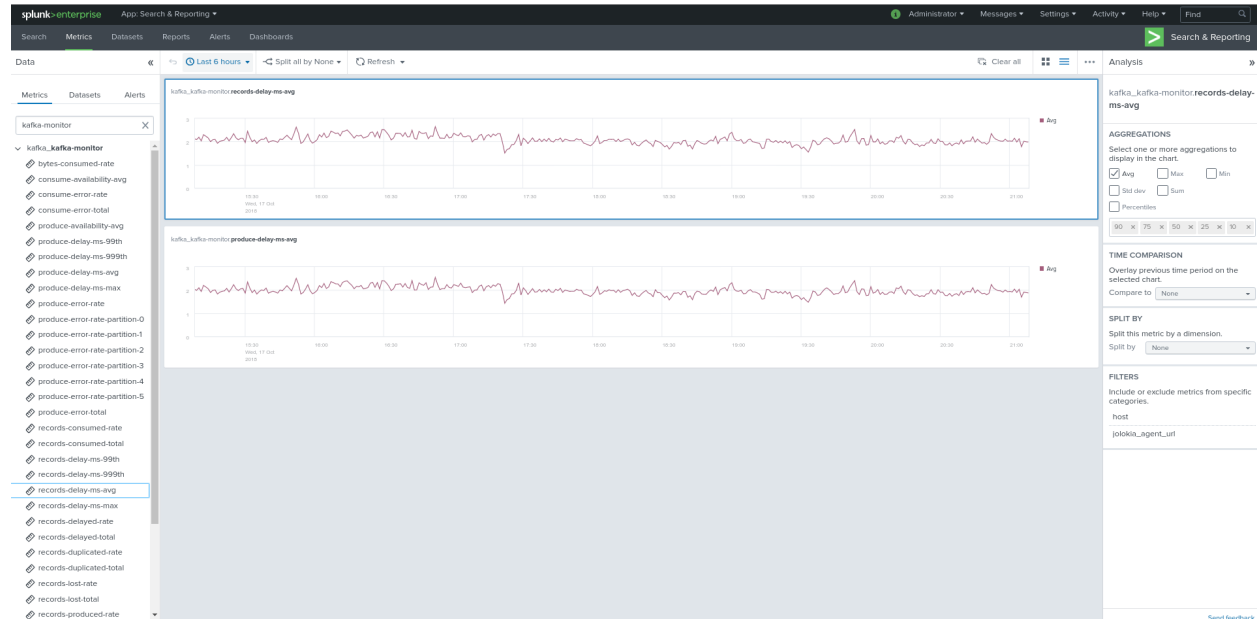
# Kafka JVM monitoring

[[inputs.jolokia2_agent]]
name_prefix = "kafka_"
urls = ["http://kafka-monitor:8778/jolokia"]

[[inputs.jolokia2_agent.metric]]
name = "kafka-monitor"
mbean = "kmf.services:name=*,type=*"

```

Visualization of metrics within the Splunk metrics workspace application:



Using mcatalog search command to verify data availability:

```
| mcatalog values(metric_name) values(_dims) where index=* metric_name=kafka_kafka-
  ↳monitor.*
```

2.2.12 Confluent schema-registry

Collecting with Telegraf

Connecting to multiple remote Jolokia instances:

```
[[inputs.jolokia2_agent]]
  name_prefix = "kafka_schema-registry."
  urls = ["http://schema-registry:18783/jolokia"]
```

Connecting to local Jolokia instance:

```
# Kafka-connect JVM monitoring
[[inputs.jolokia2_agent]]
  name_prefix = "kafka_schema-registry."
  urls = ["http://$HOSTNAME:8778/jolokia"]
```

Full telegraf.conf example

bellow a full telegraf.conf example:

```
[global_tags]
  # the env tag is used by the application for multi-environments management
  env = "my_env"
  # the label tag is an optional tag used by the application that you can use as
  ↳additional label for the services or infrastructure
```

(continues on next page)

(continued from previous page)

```
label = "my_env_label"

[agent]
  interval = "10s"
  flush_interval = "10s"
  hostname = "$HOSTNAME"

# outputs
[[outputs.http]]
  url = "https://splunk:8088/services/collector"
  insecure_skip_verify = true
  data_format = "splunkmetric"
  ## Provides time, index, source overrides for the HEC
  splunkmetrichec_routing = true
  ## Additional HTTP headers
  [outputs.http.headers]
  # Should be set manually to "application/json" for json data_format
  Content-Type = "application/json"
  Authorization = "Splunk 205d43f1-2a31-4e60-a8b3-327eda49944a"
  X-Splunk-Request-Channel = "205d43f1-2a31-4e60-a8b3-327eda49944a"

# schema-registry JVM monitoring

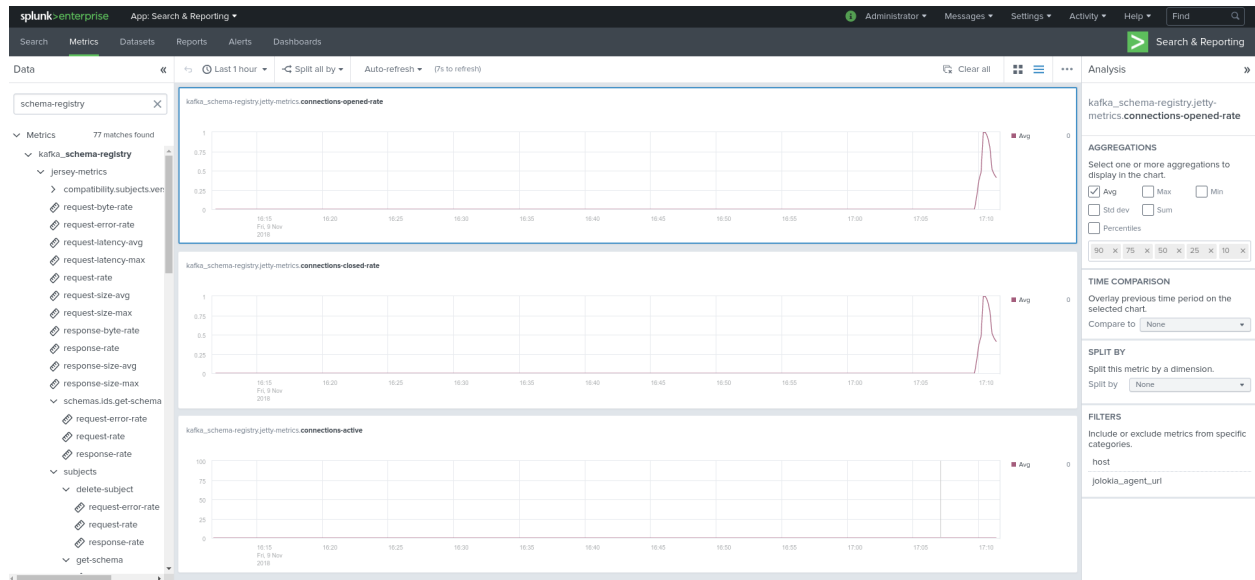
[[inputs.jolokia2_agent]]
  name_prefix = "kafka_schema-registry."
  urls = ["http://schema-registry:18783/jolokia"]

[[inputs.jolokia2_agent.metric]]
  name = "jetty-metrics"
  mbean = "kafka.schema.registry:type=jetty-metrics"
  paths = ["connections-active", "connections-opened-rate", "connections-closed-rate"]

[[inputs.jolokia2_agent.metric]]
  name = "master-slave-role"
  mbean = "kafka.schema.registry:type=master-slave-role"

[[inputs.jolokia2_agent.metric]]
  name = "jersey-metrics"
  mbean = "kafka.schema.registry:type=jersey-metrics"
```

Visualization of metrics within the Splunk metrics workspace application:



Using mcatalog search command to verify data availability:

```
| mcatalog values(metric_name) values(_dims) where index=* metric_name=kafka_schema-registry.*
```

2.2.13 Confluent ksql-server

Collecting with Telegraf

Connecting to multiple remote Jolokia instances:

```
[[inputs.jolokia2_agent]]
  name_prefix = "kafka_"
  urls = ["http://ksql-server-1:18784/jolokia"]
```

Connecting to local Jolokia instance:

```
[[inputs.jolokia2_agent]]
  name_prefix = "kafka_"
  urls = ["http://$HOSTNAME:18784/jolokia"]
```

Full telegraf.conf example

below a full telegraf.conf example:

```
[global_tags]
  # the env tag is used by the application for multi-environments management
  env = "my_env"
  # the label tag is an optional tag used by the application that you can use as
  # additional label for the services or infrastructure
  label = "my_env_label"

[agent]
  interval = "10s"
```

(continues on next page)

(continued from previous page)

```

flush_interval = "10s"
hostname = "$HOSTNAME"

# outputs
[[outputs.http]]
  url = "https://splunk:8088/services/collector"
  insecure_skip_verify = true
  data_format = "splunkmetric"
  ## Provides time, index, source overrides for the HEC
  splunkmetrichec_routing = true
  ## Additional HTTP headers
  [outputs.http.headers]
  # Should be set manually to "application/json" for json data_format
  Content-Type = "application/json"
  Authorization = "Splunk 205d43f1-2a31-4e60-a8b3-327eda49944a"
  X-Splunk-Request-Channel = "205d43f1-2a31-4e60-a8b3-327eda49944a"

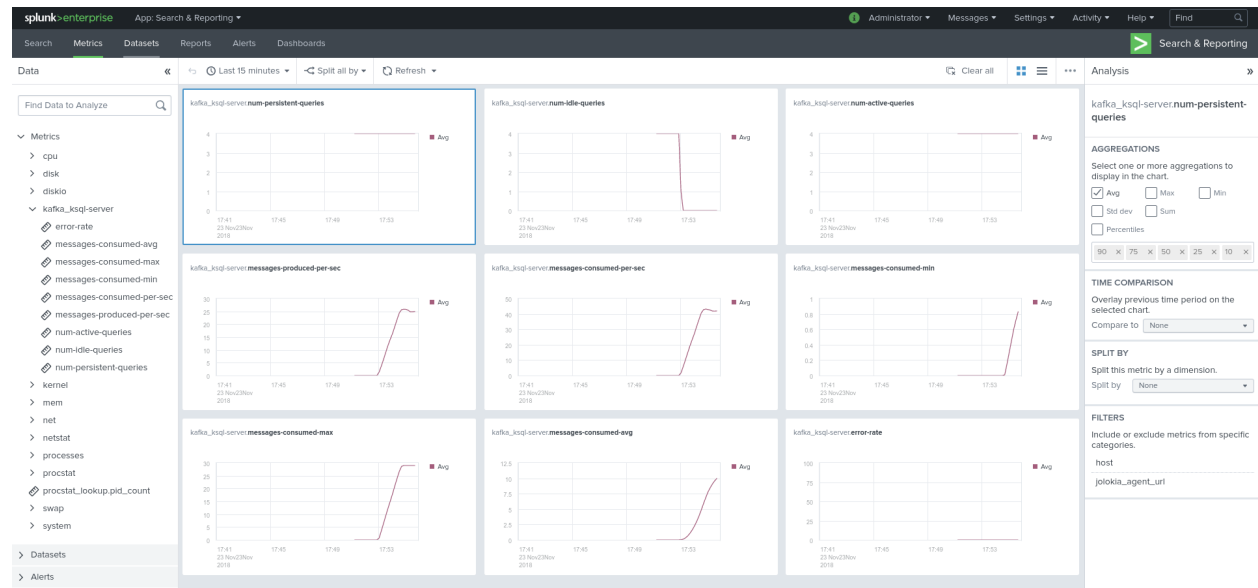
# ksql-server JVM monitoring

[[inputs.jolokia2_agent]]
  name_prefix = "kafka_"
  urls = ["http://ksql-server:18784/jolokia"]

[[inputs.jolokia2_agent.metric]]
  name = "ksql-server"
  mbean = "io.confluent.ksql.metrics:type=*"

```

Visualization of metrics within the Splunk metrics workspace application:



Using mcatalog search command to verify data availability:

```

| mcatalog values(metric_name) values(_dims) where index=* metric_name=kafka_ksql-
server.*

```


2.2.14 Confluent kafka-rest

Collecting with Telegraf

Connecting to multiple remote Jolokia instances:

```
[[inputs.jolokia2_agent]]
  name_prefix = "kafka_kafka-rest."
  urls = ["http://kafka-rest:8778/jolokia"]
```

Connecting to local Jolokia instance:

```
[[inputs.jolokia2_agent]]
  name_prefix = "kafka_kafka-rest."
  urls = ["http://$HOSTNAME:18785/jolokia"]
```

Full telegraf.conf example

bellow a full telegraf.conf example:

```
[global_tags]
  # the env tag is used by the application for multi-environments management
  env = "my_env"
  # the label tag is an optional tag used by the application that you can use as
  ↪ additional label for the services or infrastructure
  label = "my_env_label"

[agent]
  interval = "10s"
  flush_interval = "10s"
  hostname = "$HOSTNAME"

# outputs
[[outputs.http]]
  url = "https://splunk:8088/services/collector"
  insecure_skip_verify = true
  data_format = "splunkmetric"
  ## Provides time, index, source overrides for the HEC
  splunkmetrichec_routing = true
  ## Additional HTTP headers
  [outputs.http.headers]
  # Should be set manually to "application/json" for json data_format
  Content-Type = "application/json"
  Authorization = "Splunk 205d43f1-2a31-4e60-a8b3-327eda49944a"
  X-Splunk-Request-Channel = "205d43f1-2a31-4e60-a8b3-327eda49944a"

# kafka-rest JVM monitoring

[[inputs.jolokia2_agent]]
  name_prefix = "kafka_kafka-rest."
  urls = ["http://kafka-rest:18785/jolokia"]

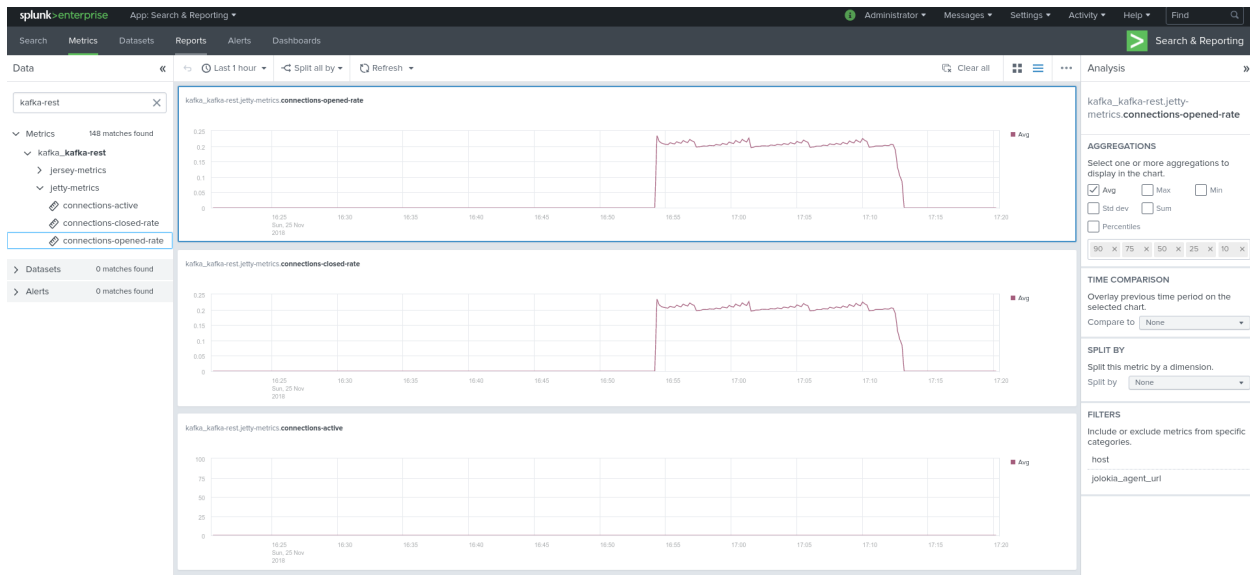
[[inputs.jolokia2_agent.metric]]
  name = "jetty-metrics"
  mbean = "kafka.rest:type=jetty-metrics"
  paths = ["connections-active", "connections-opened-rate", "connections-closed-rate"]
  ↪"]
```

(continues on next page)

(continued from previous page)

```
[[inputs.jolokia2_agent.metric]]
  name      = "jersey-metrics"
  mbean     = "kafka.rest:type=jersey-metrics"
```

Visualization of metrics within the Splunk metrics workspace application:



Using mcatalog search command to verify data availability:

```
| mcatalog values(metric_name) values(_dims) where index=* metric_name=kafka_kafka_
↪ kafka-rest.*
```

2.2.15 Confluent Interceptor Monitoring

Implement Confluent Interceptor integration to Splunk

Confluent Interceptor allows monitoring latency from producers and consumers in any kind of ways and is a very performing and rich way to monitor your Kafka components for Confluent customers:

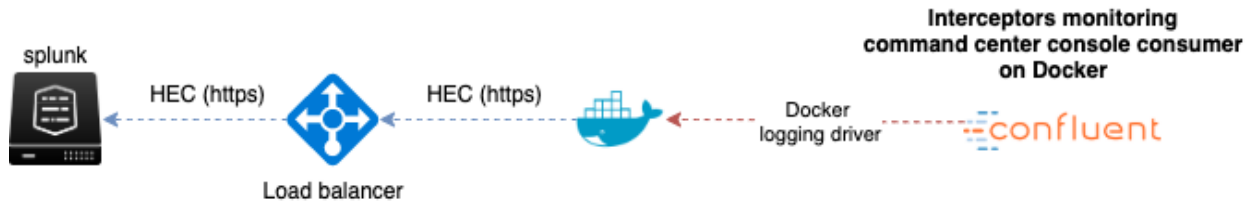
- <https://docs.confluent.io/current/control-center/installation/clients.html>

To collect Confluent Interceptors metrics in Splunk, we use the following method:

- We use a Docker container to run the command center console consumer from the interceptor topic, by default “_confluent-monitoring”
- You cannot consume this topic directly in Splunk without the command center console consumer as it contains binary data that would not be readable without the command center
- Once started, the Docker container consumes the topic and outputs the data in the stdout
- The Docker container uses the Splunk Docker logging driver to forward this data to a Splunk HTTP Event Collector endpoint
- Finally, we use the Splunk logs to metrics capabilities to transform the metric events into metrics stored in the Splunk metric store

For more information about Splunk logs to metrics capabilities, consult:

- <https://docs.splunk.com/Documentation/Splunk/latest/Metrics/L2MOverview>



Make sure you enabled Interceptors in your products as explained in the Confluent documentation, for instance for Kafka Connect you will add the following configuration in your worker properties:

```

producer.interceptor.classes=io.confluent.monitoring.clients.interceptor.
↳MonitoringProducerInterceptor
consumer.interceptor.classes=io.confluent.monitoring.clients.interceptor.
↳MonitoringConsumerInterceptor
  
```

Note: adding this config would require a restart of Kafka Connect to be applied

Once you decided where to run the Docker container, which could be the same machine hosting the command center for example, you will:

Create a new metric (not a event index!) index to store the Confluent interceptor metrics, by default the application expects:

- **confluent_interceptor_metrics**

Create an HEC token dedicated for it, or allow an existing token to forward to the metric index, example:

```

[http://confluent_interceptor_metrics]
disabled = 0
index = confluent_interceptor_metrics
indexes = confluent_interceptor_metrics
token = xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxx
  
```

The following props.conf and transforms.conf configuration need to be deployed to the indexers or intermediate forwarders, these are not need on the search heads:

Define the following sourcetype in a props.conf configuration file in Splunk:

```

[confluent_interceptor]
SHOULD_LINEMERGE=false
LINE_BREAKER=([\r\n]+)
CHARSET=UTF-8
# Add the env, label and host inside the JSON, remove anything before the json start
SEDCMD-add_tags=s/. *?env=([\^s]*)\slabel=([\^s]*)\shost=([\^s]*)\s.*?\/{"env": "\1",
↳ "label": "\2", "host": "\3", /g

# Be strict and performer
TIME_FORMAT=%s%3N
TIME_PREFIX="\timestamp\":"
MAX_TIMESTAMP_LOOKAHEAD=35

# Handle the host Meta
TRANSFORMS-confluent-interceptor-host = confluent_interceptor_host
# Only keep the metrics, send any other events from the container to the null queue
TRANSFORMS-setnull = confluent_interceptor_setnull

# Logs to metrics
  
```

(continues on next page)

(continued from previous page)

```
TRANSFORMS-fieldvalue=confluent_interceptor_fields_extraction
TRANSFORMS-metricslog=confluent_interceptor_eval_pipeline
METRIC-SCHEMA-TRANSFORMS=metric-schema:extract_metrics
```

Define the following transforms in a transforms.conf configuration file in Splunk:

```
[confluent_interceptor_setnull]
REGEX = ^confluentinc/cp-enterprise-control-center
DEST_KEY = queue
FORMAT = nullQueue

[confluent_interceptor_fields_extraction]
FORMAT = $1::$2
REGEX = \"([a-zA-Z0-9_\\.]+)\"::\"?([a-zA-Z0-9_\\.]+)
REPEAT_MATCH = true
SOURCE_KEY = _raw
WRITE_META = true

[confluent_interceptor_host]
DEST_KEY = MetaData:Host
REGEX = \"host\"::\"([^\"]*)\"
FORMAT = host::$1

[confluent_interceptor_eval_pipeline]
INGEST_EVAL = metric_name="confluent_interceptor"

[metric-schema:extract_metrics]
METRIC-SCHEMA-MEASURES-confluent_interceptor=_ALLNUMS_
METRIC-SCHEMA-MEASURES-confluent_interceptor=count,aggregateBytes,aggregateCrc,
↪totalLatency,minLatency,maxLatency,arrivalTime
METRIC-SCHEMA-BLACKLIST-DIMS-confluent_interceptor=host,session,sequence>window,
↪minWindow,maxWindow
```

Define a new Docker container; you can use docker-compose for an easier deployment and maintenance:

- On the machine hosting the Docker container, create a new directory:

```
mkdir /opt/confluent-interceptor
cd /opt/confluent-interceptor
```

- In this directory, copy the command center properties file that you use for command center, at the minimal you need to define the kafka broker and zookeeper connection string:

properties

- We use the properties file to bootstrap the command center console consumer, not an instance of the command center
- You can remove most of the configuration from the properties, what is required is providing the connectivity settings to your Kafka brokers and Zookeeper ensemble
- If you SSL and any mechanism use of authentication, make sure to include the settings accordingly
- Do not update the setting `confluent.controlcenter.data.dir` from your Command center configuration, if the directory cannot be used by the container, the console consumer will not start

control-center.properties

```
##### Server Basics #####

# A comma separated list of Apache Kafka cluster host names (required)
bootstrap.servers=localhost:9092

# A comma separated list of ZooKeeper host names (for ACLs)
zookeeper.connect=localhost:2181
```

Finally, create a new `docker-compose.yml` file as follows, edit the Splunk index, the HEC target and the HEC token to match your deployment:

docker-compose version

- Make sure to download the very last version of docker-compose from <https://docs.docker.com/compose/install>
- If you cannot use a recent version of docker-compose and/or the Docker engine, lower the version on top of the yml file

```
version: '3.8'
services:

  confluent-interceptor:
    image: confluentinc/cp-enterprise-control-center
    restart: always
    hostname: confluent-interceptor
    logging:
      driver: splunk
    options:
      splunk-token: "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxx"
      splunk-url: "https://mysplunk.domain.com:8088"
      splunk-insecureskipverify: "true"
      splunk-verify-connection: "false"
      splunk-index: "confluent_interceptor_metrics"
      splunk-sourcetype: "confluent_interceptor"
      splunk-format: "raw"
      tag: "{{.ImageName}}/{{.Name}}/{{.ID}}"
      env: "env,label,host"
    mem_limit: 600m
    extra_hosts:
      - "kafka-1 kafka-1.acme.com:xxx.xxx.xxx.xxx"
      - "kafka-2 kafka-2.acme.com:xxx.xxx.xxx.xxx"
      - "kafka-3 kafka-3.acme.com:xxx.xxx.xxx.xxx"
    volumes:
      - ../confluent/control-center.properties:/etc/confluent-control-center/control-
        ↪center.properties
    environment:
      env: "docker_env"
      label: "testing"
      host: "confluent-consumer-interceptor"
      command: "/usr/bin/control-center-console-consumer /etc/confluent-control-center/
        ↪control-center.properties --topic _confluent-monitoring"
```

Tip:

- You can include an hosts mapping in the docker-compose file to populate the `/etc/hosts` on the container, depending on your Kafka brokers configuration, it might be required that the container knows how to communicate

with the brokers using their FQDN / host name for instance

- DNS resolution from the container is a potential root cause of failure so it is important you handle this configuration properly

```
version: '3.8'
services:

  confluent-interceptor:
    image: confluentinc/cp-enterprise-control-center
    restart: always
    hostname: confluent-interceptor
    mem_limit: 600m
    extra_hosts:
      - "kafka-1 kafka-1.acme.com:xxx.xxx.xxx.xxx"
      - "kafka-2 kafka-2.acme.com:xxx.xxx.xxx.xxx"
      - "kafka-3 kafka-3.acme.com:xxx.xxx.xxx.xxx"
```

Start the container:

```
docker-compose up -d
```

Shall the system be restarted, or the container be failing, Docker will automatically restart a new container.

After the image has been downloaded, the container automatically starts and metrics start to be forwarded to Splunk:

The screenshot shows the Splunk Search interface. At the top, there's a navigation bar with tabs like Overview, Brokers, Topics, Burrow, Analytics, Search, Kafka monitoring, Kafka logging, Kafka alerting, Connected Experience, Audit, Settings, and Run a search. Below this is a 'New Search' section with a search bar containing the query: `| mcatalog values(metric_name) values(_dims) where index=* metric_name=confluent_interceptor.* by index`. Below the search bar, it shows '10 events (7/70/2020 14:00:00.000 to 18/10/2020 14:46:25.000)' and 'No Event Sampling'. The main table displays search results with columns for index, values(metric_name), and values(_dims). The index column shows 'confluent_interceptor_metrics'. The values(metric_name) column lists various metrics like 'confluent_interceptor.aggregateBytes', 'confluent_interceptor.aggregateCrc', 'confluent_interceptor.arrivalTime', 'confluent_interceptor.count', 'confluent_interceptor.maxLatency', 'confluent_interceptor.minLatency', and 'confluent_interceptor.totalLatency'. The values(_dims) column lists dimensions like 'clientId', 'clientType', 'clusterId', 'group', 'monitoringTopicPartition', 'partition', 'samplePeriod', 'shutdown', 'topic', and 'type'.

index	values(metric_name)	values(_dims)
confluent_interceptor_metrics	confluent_interceptor.aggregateBytes confluent_interceptor.aggregateCrc confluent_interceptor.arrivalTime confluent_interceptor.count confluent_interceptor.maxLatency confluent_interceptor.minLatency confluent_interceptor.totalLatency	clientId clientType clusterId group monitoringTopicPartition partition samplePeriod shutdown topic type

You can use the following search to verify that metrics are being ingested:

```
| mcatalog values(metric_name) values(_dims) where index=* metric_name=confluent_
↪interceptor.* by index
```

You can as well use the msearch command:

```
| msearch index=* filter="metric_name="confluent_interceptor.*"
```

The screenshot shows the 'New Search' interface in Kafka Smart Monitoring. The search query is `metric_name=confluent_interceptor.*`. The results show two events from 18/10/2020. The first event is a heartbeat from a consumer group, and the second event is a metric report for the same group. The interface includes navigation tabs, search filters, and a detailed view of the search results.

Troubleshoot Confluent Interceptor consumer

If you do not receive the metrics in Splunk, there can be different root causes:

- The Docker container started and stopped almost immediately, which is most certainly linked to the properties configuration
- The command-center console consumer cannot access to Kafka due to configuration issues, network connectivity, DNS resolution, etc
- The connectivity between the Docker container and Splunk HTTP Event Collector is not valid

A first verification that can be done easily consists in disabling the Splunk logging driver to review the standard and error output of the container and command-center:

Stop the container if it is running, then edit the configuration, remove, create and start the container:

```
docker-compose stop confluent-interceptor
docker-compose rm -f confluent-interceptor
```

docker-compose.yml

```
version: '3.8'
services:
  confluent-interceptor:
    image: confluentinc/cp-enterprise-control-center
    restart: always
    hostname: confluent-interceptor
    #logging:
    #  driver: splunk
    #  options:
    #    splunk-token: "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxx"
    #    splunk-url: "https://mysplunk.domain.com:8088"
    #    splunk-insecureskipverify: "true"
```

(continues on next page)

(continued from previous page)

```
# splunk-verify-connection: "false"
# splunk-index: "confluent_interceptor_metrics"
# splunk-sourcetype: "confluent_interceptor"
# splunk-format: "raw"
# tag: "{{.ImageName}}/{{.Name}}/{{.ID}}"
# env: "env,label,host"
mem_limit: 600m
volumes:
- ../confluent/control-center.properties:/etc/confluent-control-center/control-
center.properties
environment:
  env: "docker_env"
  label: "testing"
  host: "confluent-consumer-interceptor"
  command: "/usr/bin/control-center-console-consumer /etc/confluent-control-center/
control-center.properties --topic _confluent-monitoring"
```

Then run the container in attached mode: (as opposed to daemon mode with the `-d` option)

```
docker-compose up confluent-interceptor
```

The container will output to stdout, any failure to start the console consumer due to a properties issues would appear clearly:

Press `Ctrl+C` to stop the container

```
Creating template_docker_splunk_localhost_confluent-interceptor_1 ... done
Attaching to template_docker_splunk_localhost_confluent-interceptor_1
confluent-interceptor_1 | OpenJDK 64-Bit Server VM warning: Option
UseConcMarkSweepGC was deprecated in version 9.0 and will likely be removed in a
future release.
confluent-interceptor_1 | SLF4J: Class path contains multiple SLF4J bindings.
confluent-interceptor_1 | SLF4J: Found binding in [jar:file:/usr/share/java/acl/
acl-6.0.0.jar!/org/slf4j/impl/StaticLoggerBinder.class]
confluent-interceptor_1 | SLF4J: Found binding in [jar:file:/usr/share/java/
confluent-control-center/slf4j-log4j12-1.7.30.jar!/org/slf4j/impl/
StaticLoggerBinder.class]
confluent-interceptor_1 | SLF4J: See http://www.slf4j.org/codes.html#multiple_
bindings for an explanation.
confluent-interceptor_1 | SLF4J: Actual binding is of type [org.slf4j.impl.
Log4jLoggerFactory]
confluent-interceptor_1 | WARNING: An illegal reflective access operation has
occurred
confluent-interceptor_1 | WARNING: Illegal reflective access by com.google.
inject.internal.cglib.core.$ReflectUtils$1 (file:/usr/share/java/acl/acl-6.0.0.jar)
to method java.lang.ClassLoader.defineClass(java.lang.String,byte[],int,int,java.
security.ProtectionDomain)
confluent-interceptor_1 | WARNING: Please consider reporting this to the
maintainers of com.google.inject.internal.cglib.core.$ReflectUtils$1
confluent-interceptor_1 | WARNING: Use --illegal-access=warn to enable warnings
of further illegal reflective access operations
confluent-interceptor_1 | WARNING: All illegal access operations will be denied
in a future release
confluent-interceptor_1 | _confluent-monitoring 0 2020-10-18T14:05:57.
771Z null {"clientType":"CONSUMER","clientId":"connector-consumer-sink-
splunk-demo2-0","group":"connect-sink-splunk-demo2","session":"f0538df4-a9bd-458b-
94f6-5d21c94f812d","sequence":"4","window":"0","timestamp":"1603029957771","topic":
"kafka_demo","partition":0,"count":"0","aggregateBytes":"0","aggregateCount":0,
"totalLatency":"0","minLatency":"0","maxLatency":"0","samplePeriod":"15000","type":
"HEARTBEAT","shutdown":false,"minWindow":"-1","maxWindow":"-1",
"monitoringTopicPartition":0,"clusterId":"nBWbr"}
arrivalTime":"0"}
(continues on next page)
```


(continued from previous page)

```

confluent-interceptor_1      | _confluent-monitoring 0      2020-10-18T14:06:00.
↪033Z      null      {"clientType":"CONSUMER","clientId":"connector-consumer-sink-
↪splunk-demo1-0","group":"connect-sink-splunk-demo1","session":"d18293d8-7f25-4b6c-
↪bbfc-07a08efab9af","sequence":"7","window":"0","timestamp":"1603029960033","topic":
↪"kafka_demo","partition":0,"count":"0","aggregateBytes":"0","aggregateCrc":0,
↪"totalLatency":"0","minLatency":"0","maxLatency":"0","samplePeriod":"15000","type":
↪"HEARTBEAT","shutdown":false,"minWindow":"-1","maxWindow":"-1",
↪"monitoringTopicPartition":0,"clusterId":"nBWbrPOaRbyE-2Wp0viUwA","clusterName":"","
↪"arrivalTime":"0"}
confluent-interceptor_1      | _confluent-monitoring 0      2020-10-18T14:06:04.
↪652Z      null      {"clientType":"CONSUMER","clientId":"connector-consumer-sink-
↪splunk-demo3-0","group":"connect-sink-splunk-demo3","session":"f0c0222c-a466-4b60-
↪8497-7cb1d0ebfafc","sequence":"22","window":"0","timestamp":"1603029964652","topic":
↪"kafka_demo_headers","partition":0,"count":"0","aggregateBytes":"0","aggregateCrc
↪":0,"totalLatency":"0","minLatency":"0","maxLatency":"0","samplePeriod":"15000",
↪"type":"HEARTBEAT","shutdown":false,"minWindow":"-1","maxWindow":"-1",
↪"monitoringTopicPartition":0,"clusterId":"nBWbrPOaRbyE-2Wp0viUwA","clusterName":"","
↪"arrivalTime":"0"}
^CGracefully stopping... (press Ctrl+C again to force)
Stopping template_docker_splunk_localhost_confluent-interceptor_1 ... done

```

In the output, raw metrics are:

```

| _confluent-monitoring      0      2020-10-18T14:05:57.771Z      null      {
↪"clientType":"CONSUMER","clientId":"connector-consumer-sink-splunk-demo2-0","group":
↪"connect-sink-splunk-demo2","session":"f0538df4-a9bd-458b-94f6-5d21c94f812d",
↪"sequence":"4","window":"0","timestamp":"1603029957771","topic":"kafka_demo",
↪"partition":0,"count":"0","aggregateBytes":"0","aggregateCrc":0,"totalLatency":"0",
↪"minLatency":"0","maxLatency":"0","samplePeriod":"15000","type":"HEARTBEAT",
↪"shutdown":false,"minWindow":"-1","maxWindow":"-1","monitoringTopicPartition":0,
↪"clusterId":"nBWbrPOaRbyE-2Wp0viUwA","clusterName":"","arrivalTime":"0"}

```

If you can see metrics here, then the command center console consumer is able to bootstrap, access Kafka and Zookeeper, and there are activity in the topic.

Note that if you have no consumers or producers with the Confluent interceptors enabled, there will be no metrics generated here.

Disable command-center startup, keep the container running and exec into the container:

The next troubleshooting steps will allow you to enter the container and manually troubleshoot the startup of command center.

To achieve this, we disable the command replaced by a tail which allows keeping the container ready for operations:

```

version: '3.8'
services:

  confluent-interceptor:
    image: confluentinc/cp-enterprise-control-center
    restart: always
    hostname: confluent-interceptor
    #logging:
    #  driver: splunk
    #  options:
    #    splunk-token: "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxx"
    #    splunk-url: "https://mysplunk.domain.com:8088"
    #    splunk-insecureskipverify: "true"

```

(continues on next page)

(continued from previous page)

```
# splunk-verify-connection: "false"
# splunk-index: "confluent_interceptor_metrics"
# splunk-sourcetype: "confluent_interceptor"
# splunk-format: "raw"
# tag: "{{.ImageName}}/{{.Name}}/{{.ID}}"
# env: "env,label,host"
mem_limit: 600m
volumes:
  - ../confluent/control-center.properties:/etc/confluent-control-center/control-
↪center.properties
environment:
  env: "docker_env"
  label: "testing"
  host: "confluent-consumer-interceptor"
  #command: "/usr/bin/control-center-console-consumer /etc/confluent-control-center/
↪control-center.properties --topic _confluent-monitoring"
command: "tail -f /dev/null"
```

If the container is started, stop the container, then remove and create the container:

```
docker-compose stop confluent-interceptor
docker-compose rm -f confluent-interceptor
```

Start the container in daemon mode:

```
docker-compose up -d confluent-interceptor
```

Exec into the container:

```
docker-compose exec confluent-interceptor /bin/bash
```

Once you are in container, you can review the properties file as it seen by the container, make sure it contains the proper required configuration:

```
cat /etc/confluent-control-center/control-center.properties
```

You can attempt to manually run command-center console consumer and review step by step any failure:

```
/usr/bin/control-center-console-consumer /etc/confluent-control-center/control-center.
↪properties --topic _confluent-monitoring
```

Review carefully any failure.

Tip:

- By default, the exec command will make you enter the container as the relevant user “appuser”
- If you wish to access as root user instead, use the `--user root` argument in the exec command

```
docker-compose exec confluent-interceptor /bin/bash
```

For anymore troubleshooting related to command-center itself, consult:

- <https://docs.confluent.io/current/control-center/installation/troubleshooting.html>

If these steps are fine but you do not receive metrics in Splunk, there might a connectivity issue or misconfiguration on between the Docker container and Splunk, you can force the Docker logging driver to verify the connectivity when starting up:

```
splunk-verify-connection: "true"
```

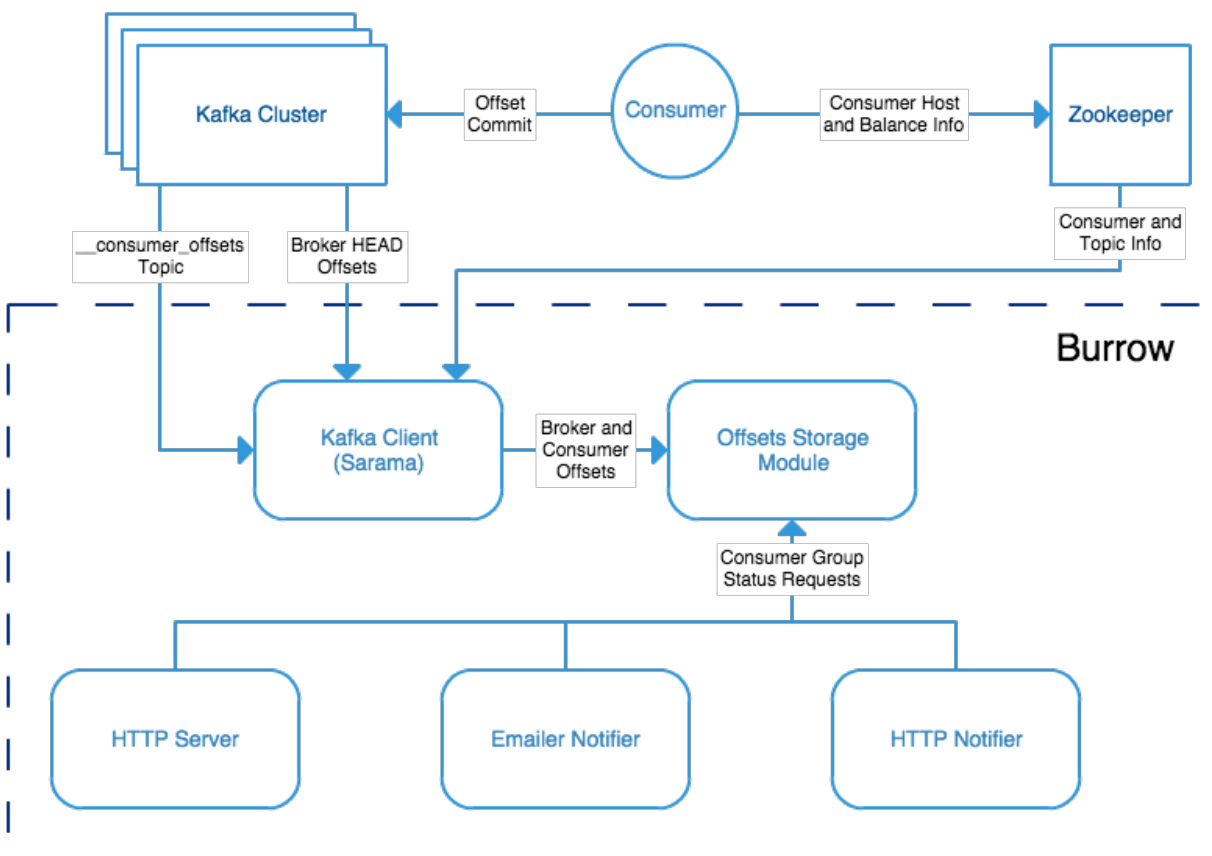
If the connectivity is not working, Docker will refuse to start the container.

2.2.16 Burrow Lag Consumers

As from their authors, Burrow is a monitoring companion for Apache Kafka that provides consumer lag checking as a service without the need for specifying thresholds.

See: <https://github.com/linkedin/Burrow>

Burrow workflow diagram:



Burrow is a very powerful application that monitors all consumers (Kafka Connect connectors, Kafka Streams...) to report an advanced state of the service automatically, and various useful lagging metrics.

Telegraf has a native input for Burrow which polls consumers, topics and partitions lag metrics and statuses over http, use the following telegraf minimal configuration:

See: <https://github.com/influxdata/telegraf/tree/master/plugins/inputs/burrow>

```
[global_tags]
# the env tag is used by the application for multi-environments management
```

(continues on next page)

(continued from previous page)

```

env = "my_env"
# the label tag is an optional tag used by the application that you can use as
↳ additional label for the services or infrastructure
label = "my_env_label"

[agent]
interval = "10s"
flush_interval = "10s"
hostname = "$HOSTNAME"

# outputs
[[outputs.http]]
url = "https://splunk:8088/services/collector"
insecure_skip_verify = true
data_format = "splunkmetric"
  ## Provides time, index, source overrides for the HEC
splunkmetrichec_routing = true
  ## Additional HTTP headers
[outputs.http.headers]
# Should be set manually to "application/json" for json data_format
Content-Type = "application/json"
Authorization = "Splunk 205d43f1-2a31-4e60-a8b3-327eda49944a"
X-Splunk-Request-Channel = "205d43f1-2a31-4e60-a8b3-327eda49944a"

# Burrow

[[inputs.burrow]]
  ## Burrow API endpoints in format "schema://host:port".
  ## Default is "http://localhost:8000".
servers = ["http://dockerhost:9001"]

  ## Override Burrow API prefix.
  ## Useful when Burrow is behind reverse-proxy.
  # api_prefix = "/v3/kafka"

  ## Maximum time to receive response.
  # response_timeout = "5s"

  ## Limit per-server concurrent connections.
  ## Useful in case of large number of topics or consumer groups.
  # concurrent_connections = 20

  ## Filter clusters, default is no filtering.
  ## Values can be specified as glob patterns.
  # clusters_include = []
  # clusters_exclude = []

  ## Filter consumer groups, default is no filtering.
  ## Values can be specified as glob patterns.
  # groups_include = []
  # groups_exclude = []

  ## Filter topics, default is no filtering.
  ## Values can be specified as glob patterns.
  # topics_include = []
  # topics_exclude = []

```

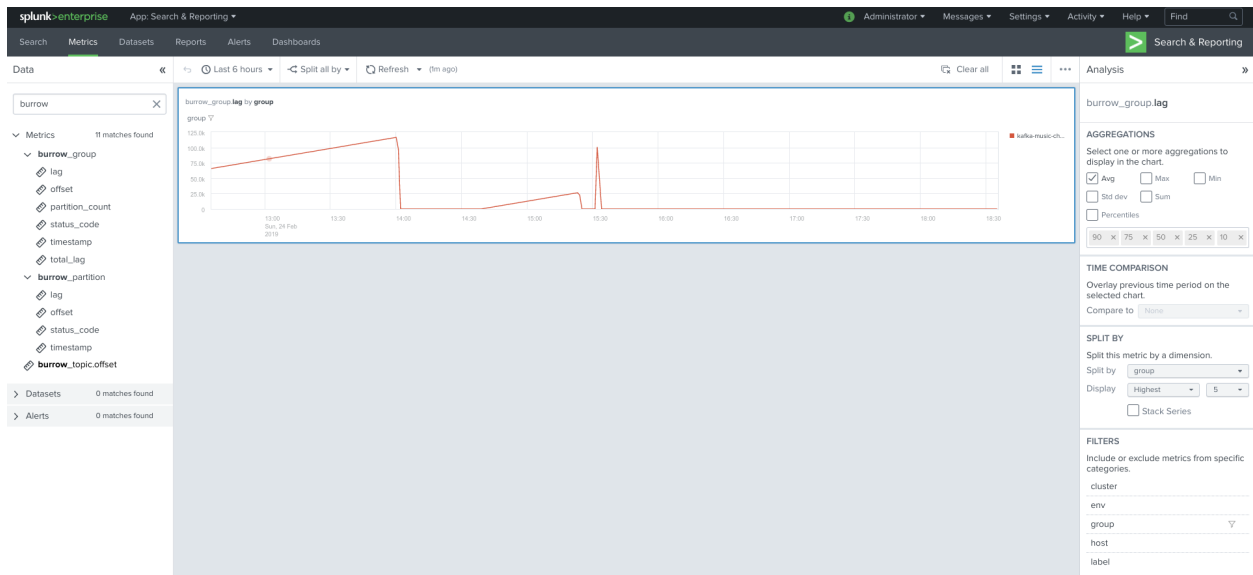
(continues on next page)

(continued from previous page)

```
## Credentials for basic HTTP authentication.
# username = ""
# password = ""

## Optional SSL config
# ssl_ca = "/etc/telegraf/ca.pem"
# ssl_cert = "/etc/telegraf/cert.pem"
# ssl_key = "/etc/telegraf/key.pem"
# insecure_skip_verify = false
```

Visualization of metrics within the Splunk metrics workspace application:



Using mcatalog search command to verify data availability:

```
| mcatalog values(metric_name) values(_dims) where index=* metric_name=burrow_*
```

2.3 Docker testing templates

Docker compose templates are provided in the following repository:

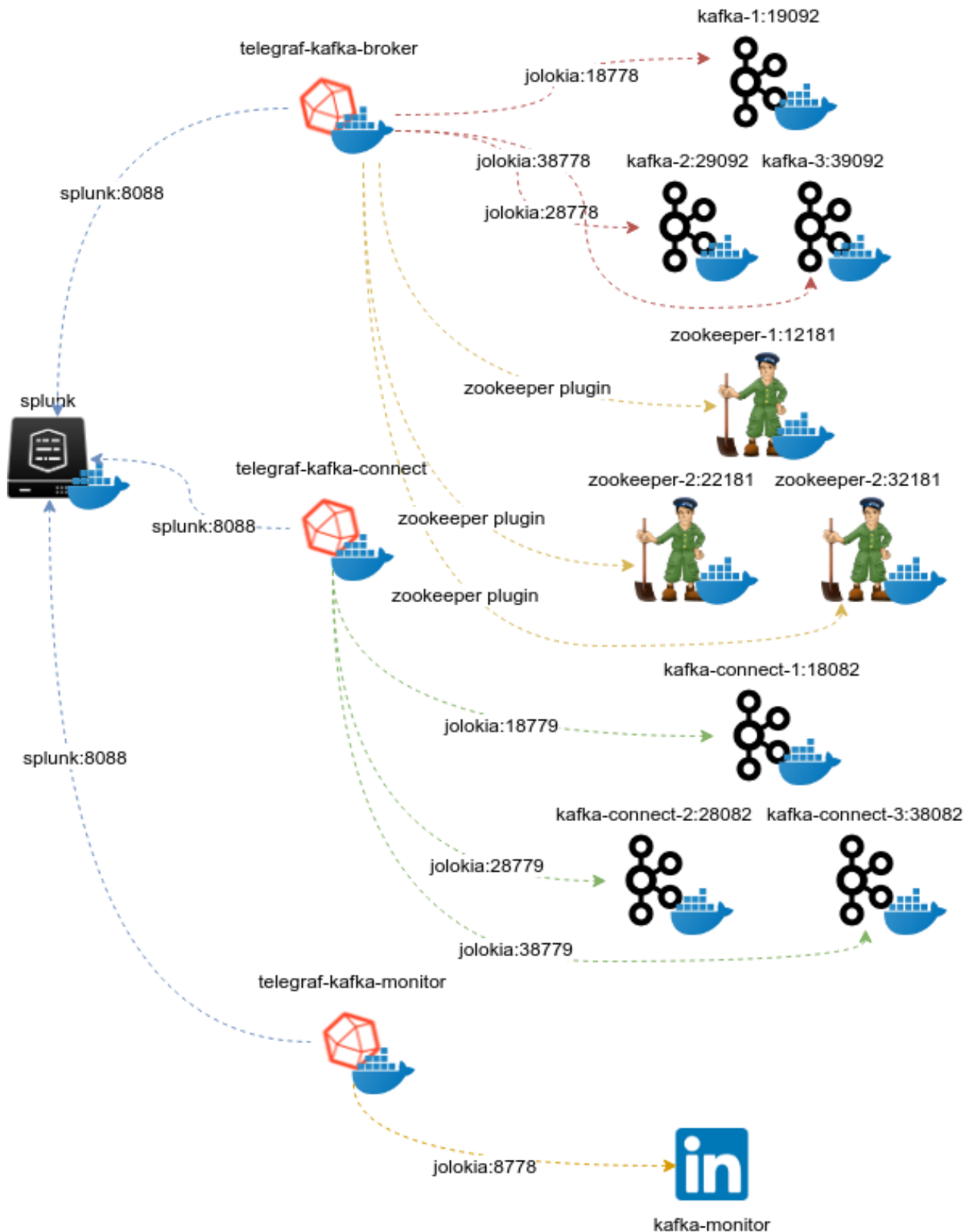
<https://github.com/guilhemmarchand/kafka-docker-splunk>

Using the docker templates allows you to create a full pre-configured Kafka environment with docker, just in 30 seconds.

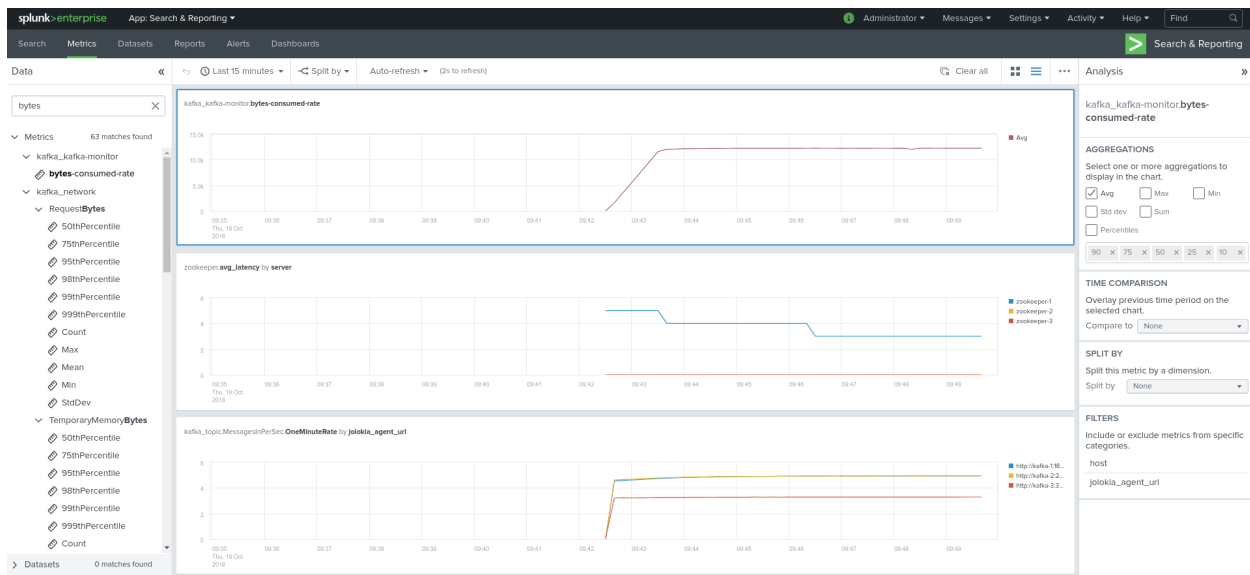
Example:

- Zookeeper cluster (3 nodes)
- Kafka broker cluster (3 nodes)
- Kafka connect cluster (1 node, can be extended up to 3 or more with additional config)
- Confluent schema-registry
- Confluent kafka-rest
- Confluent ksql-server

- Kafka Xinfra SLA monitor container
- Telegraf container polling and sending to your Splunk metric store
- Yahoo Kafka Manager
- Confluent Interceptor console collector
- Kafka Burrow Consumer lag Monitoring



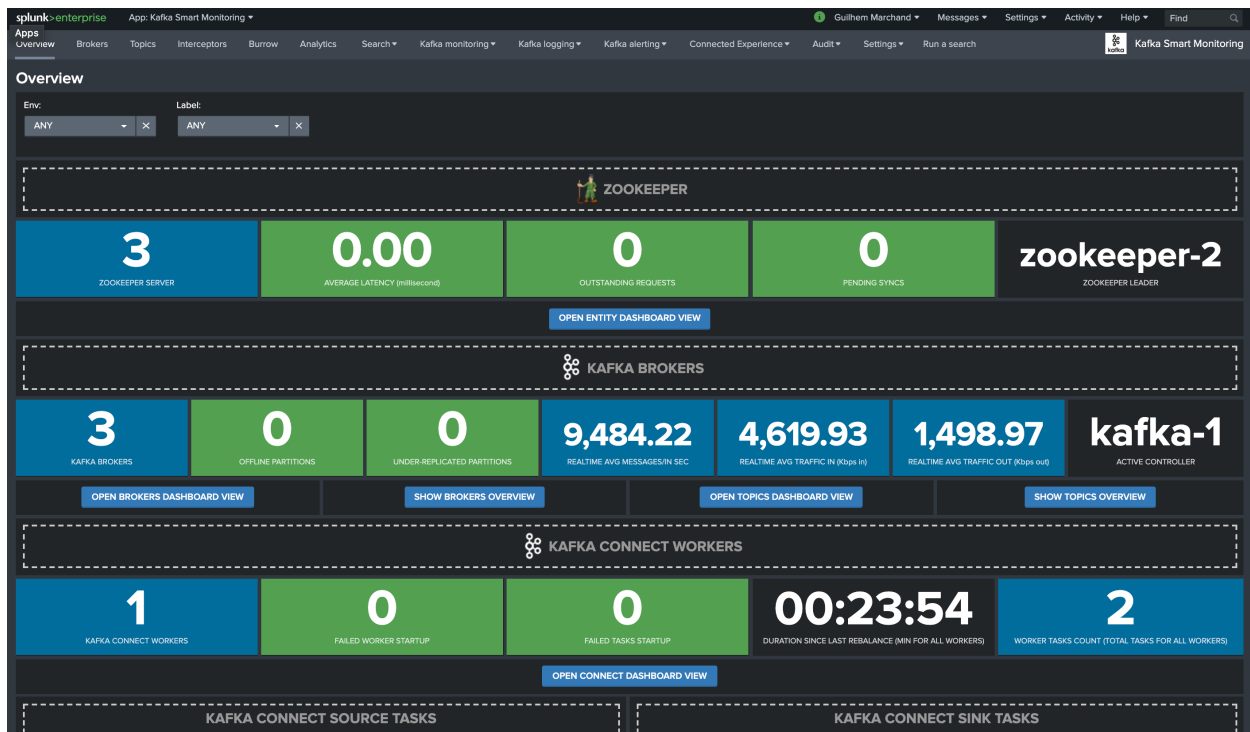
Start the template, have a very short coffee (approx. 30 sec), open Splunk, install the Metrics workspace app and observe the magic happening !



2.4 Splunk dashboards (health views)

2.4.1 Overview (landing page)

The Splunk application home page provides an overview of the Kafka infrastructure:



By default, all Kafka components including Confluent components are shown in the Overview page.

Should a component be not detected, an informational panel replaces the metric related panel.

A configuration user interface is available in the app menu bar:

- Settings / Management of enable components

Use this user interface to enable or disable any of the components depending on your needs:

Kafka Smart Monitoring - Application configuration
Use this interface to manage the main configuration items of the application

Smart Components enablement
Depending on the products in use, you can enable or disable a component, which will automatically configure the component to be visible or hidden from the Overview dashboard

Component	Enabled	Disabled	Error
Zookeeper	3	1	0
Kafka_Broker	3	0	0
Kafka_Connect	3	0	0
Kafka_Burrow	3	0	0
Confluent_Kafka_Rest	3	0	0
Confluent_ksql_server	3	0	0
Confluent_Schema_Registry	3	0	0

Click on any of the row below to open the configuration window for this item.

[Enable all components](#) [Disable all components](#)

label	state	range
Zookeeper	enabled	
Kafka_Broker	enabled	
Kafka_Connect	enabled	
Kafka_Burrow	enabled	
Confluent_Kafka_Rest	enabled	
Confluent_ksql_server	enabled	
Confluent_Schema_Registry	enabled	

2.4.2 Overview Brokers

The Brokers overview page inspired from Yahoo Kafka manager exposes the Kafka brokers and main metrics:

Brokers

Env: Label: Kafka broker(s): [Hide Filters](#)

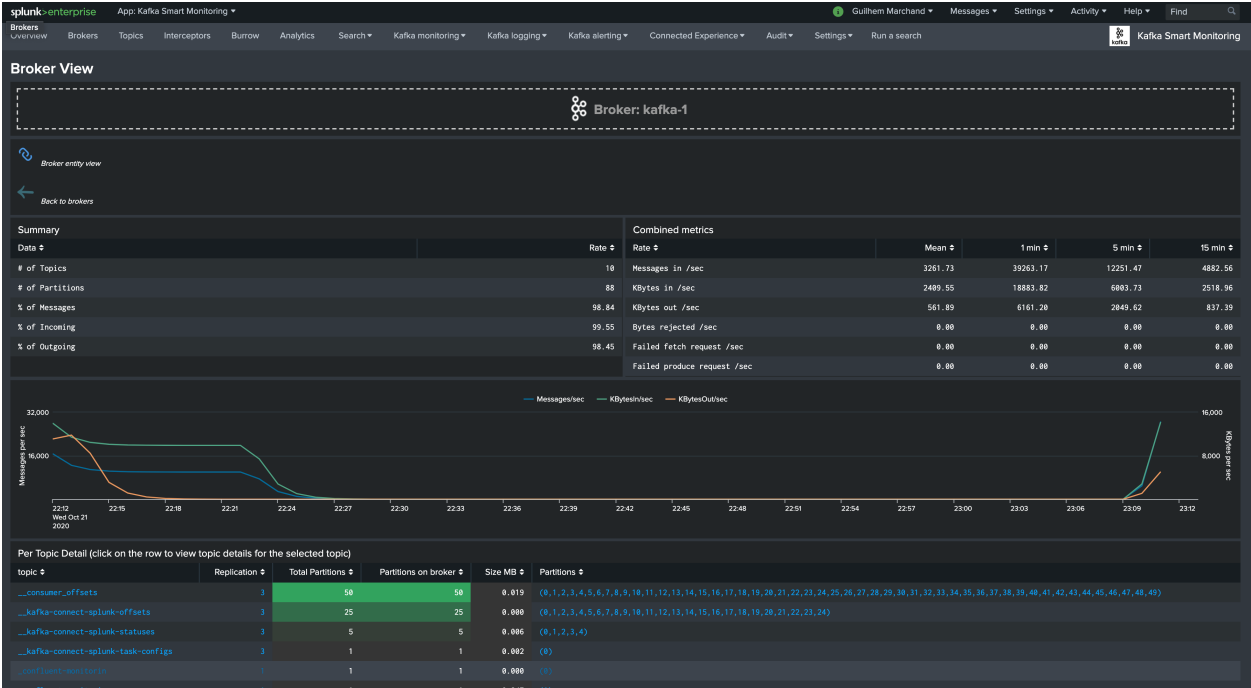
Brokers (click on the row to open broker detailed view for the selected broker)

env	label	kafka_broker	KBytes in	sparkline_in	KBytes out	sparkline_out	offline	under-replicated
docker_env	testing	kafka-1	3253.95		1213.01		0	0
docker_env	testing	kafka-2	5.36		5.36		0	0
docker_env	testing	kafka-3	5.37		5.37		0	0

Combined Metrics

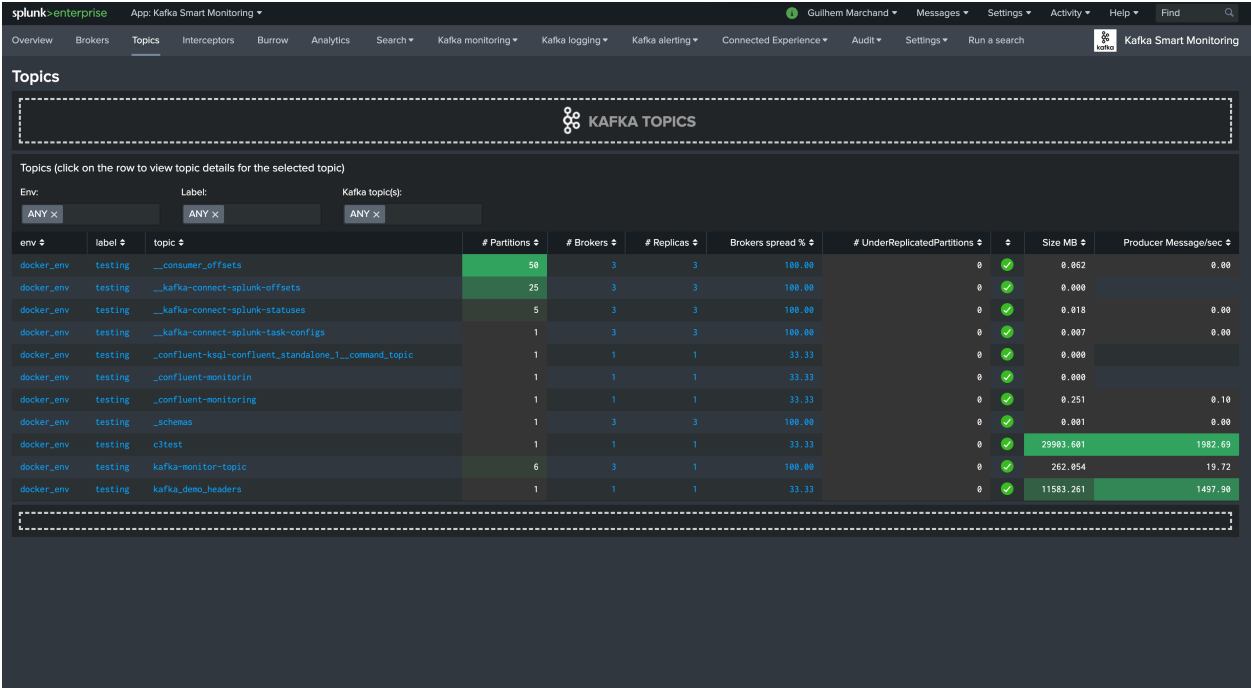
Rate	Mean	1 min	5 min	15 min
Messages in /sec	1029.32	1906.41	432.34	314.32
KBytes in /sec	773.72	1088.23	245.77	286.78
KBytes out /sec	179.17	487.91	92.00	64.46
Bytes rejected /sec	0.00	0.00	0.00	0.00
Failed fetch request /sec	0.00	0.00	0.00	0.00
Failed produce request /sec	0.00	0.00	0.00	0.00

Broker drilldown page:

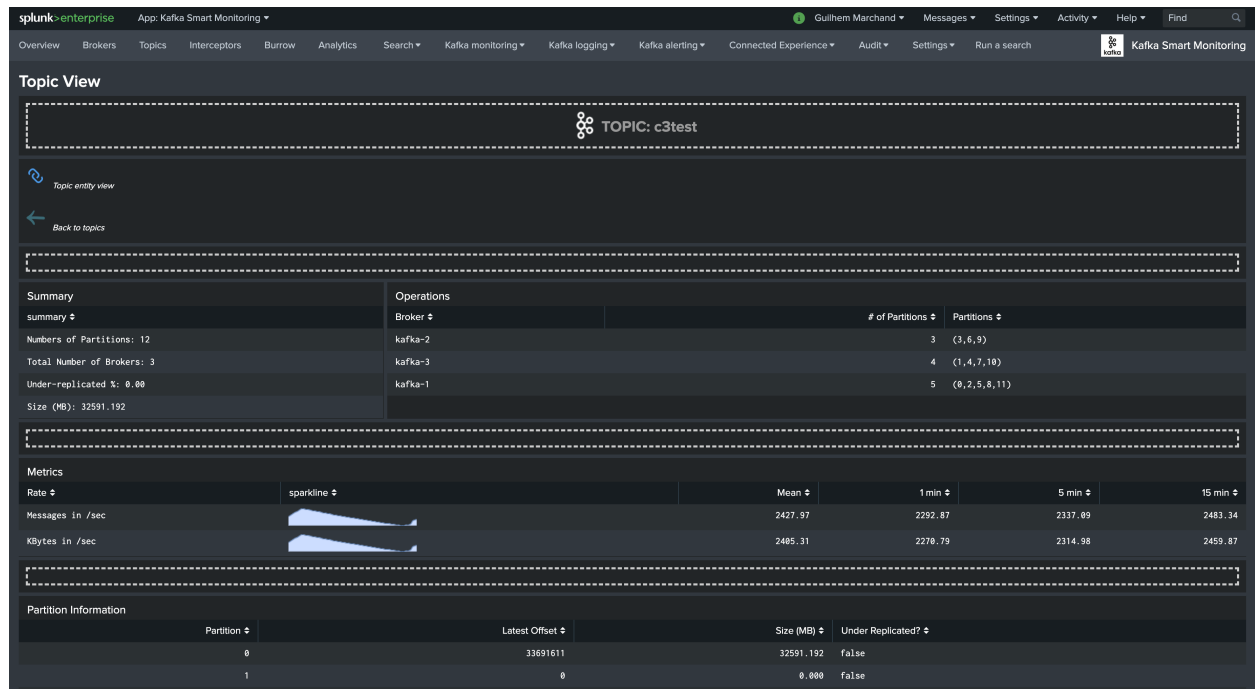


2.4.3 Overview Topics

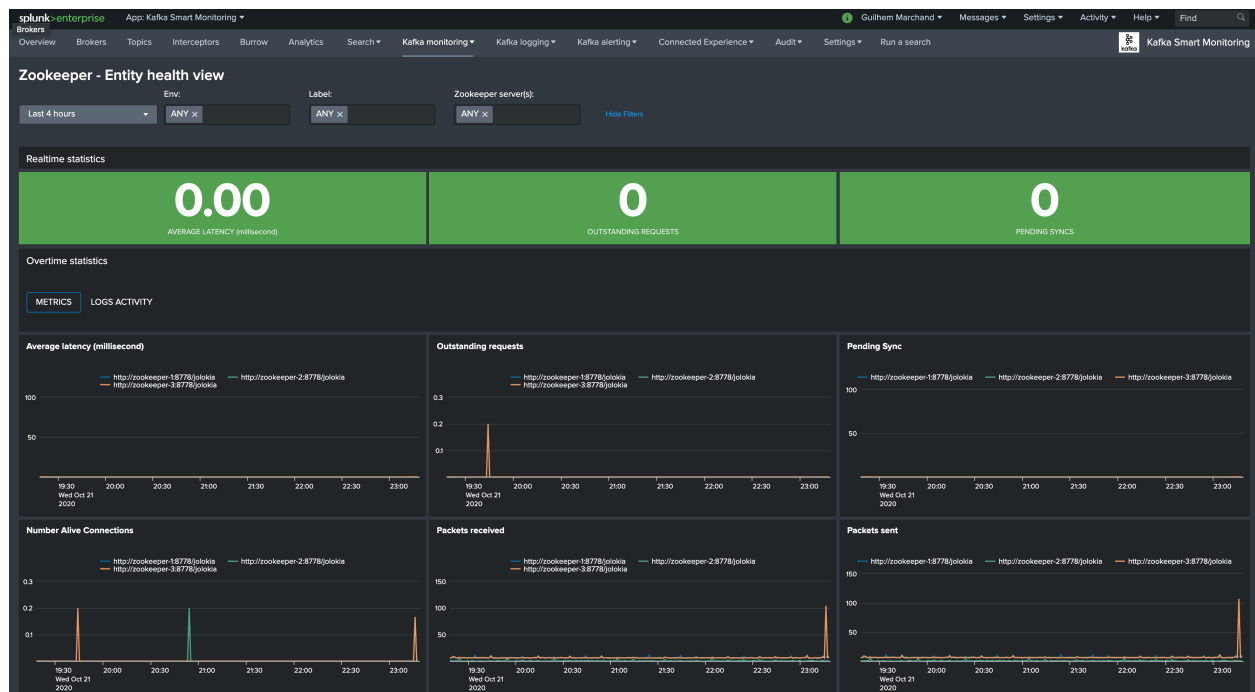
The Topics overview page inspired from Yahoo Kafka manager exposes the Kafka topics and main metrics:

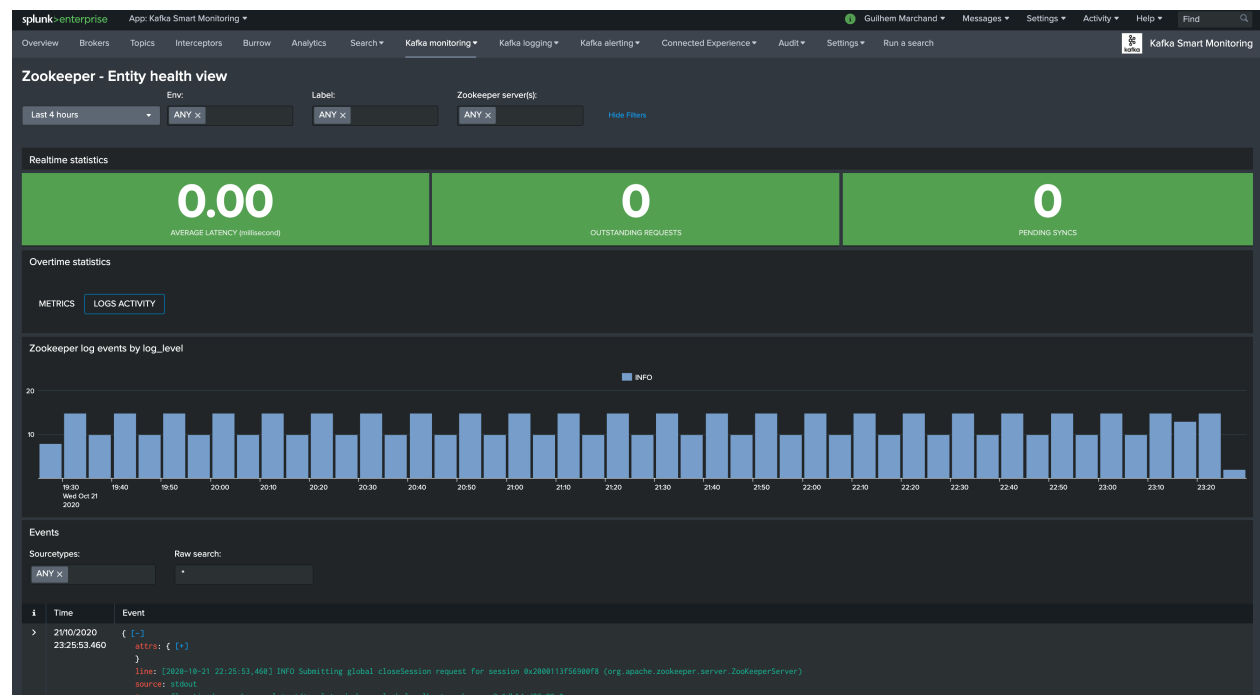


Topic drilldown page:

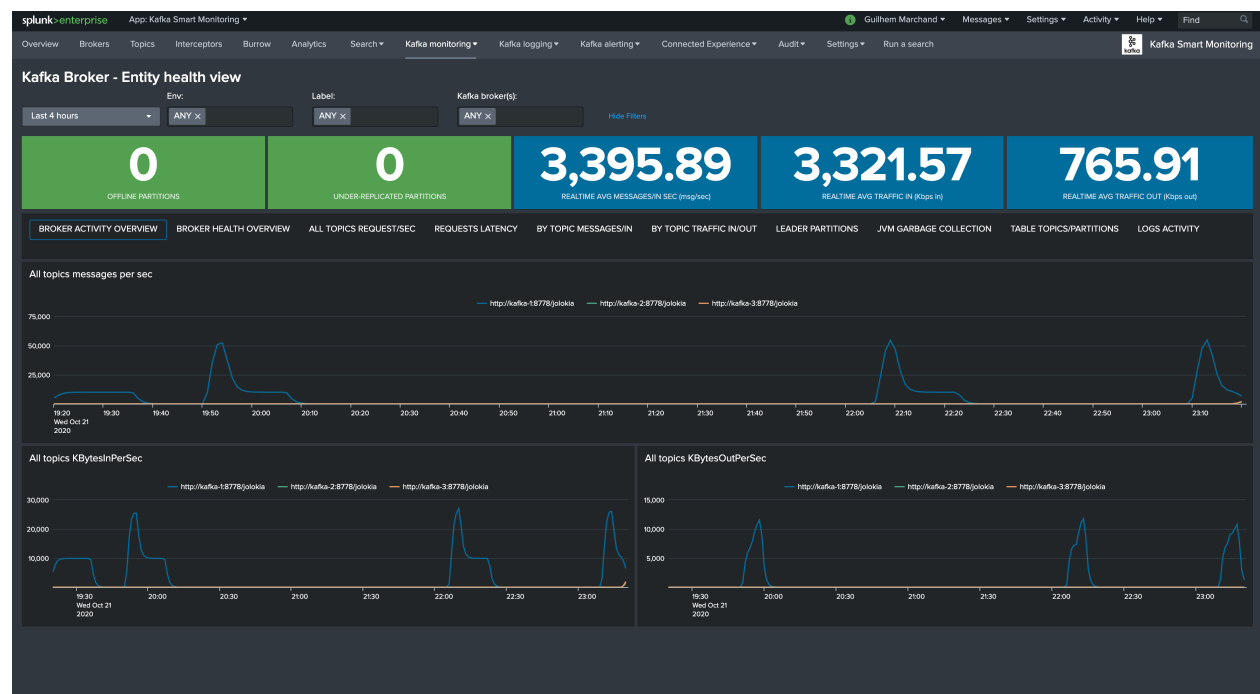


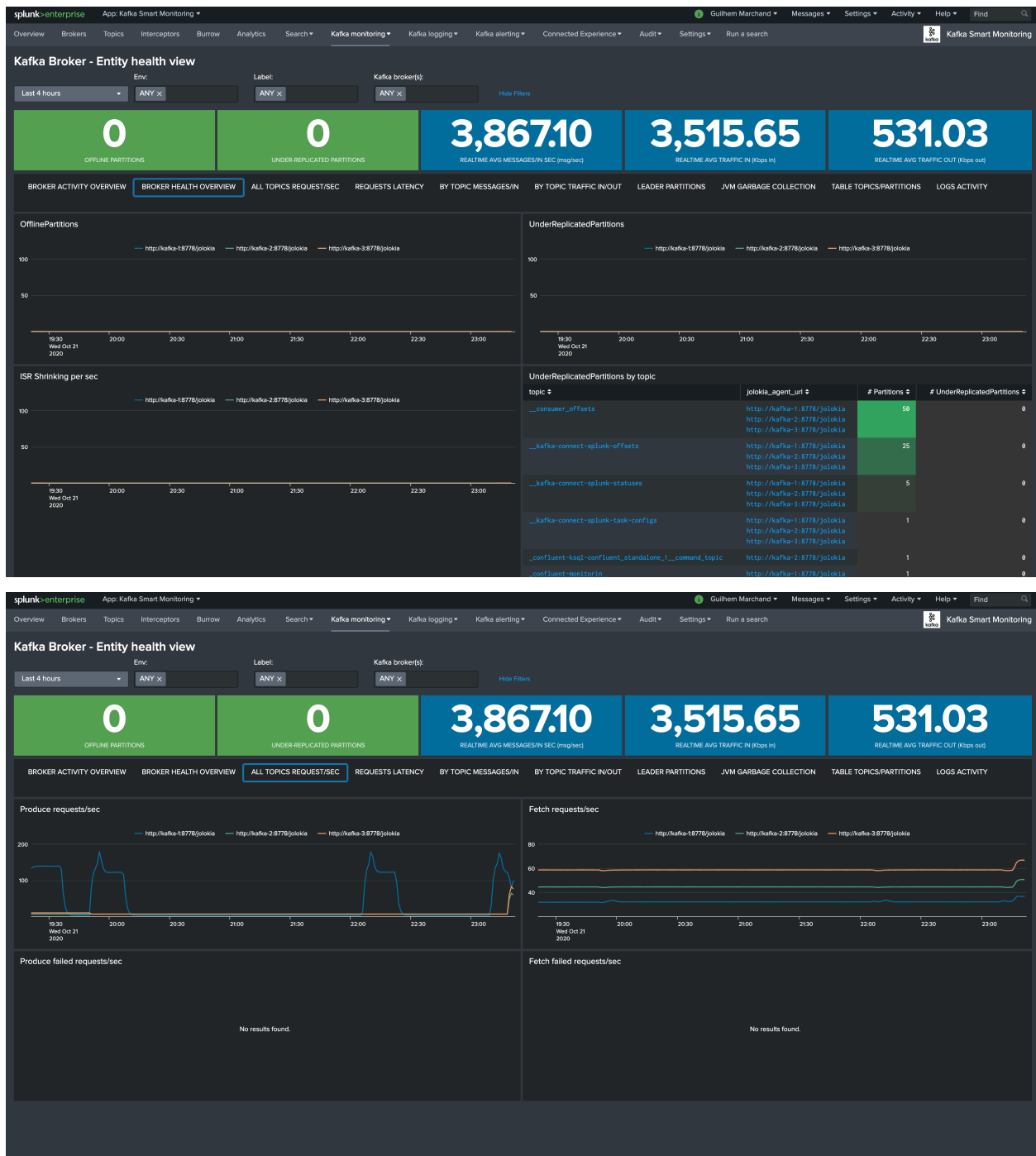
2.4.4 Zookeeper dashboard view

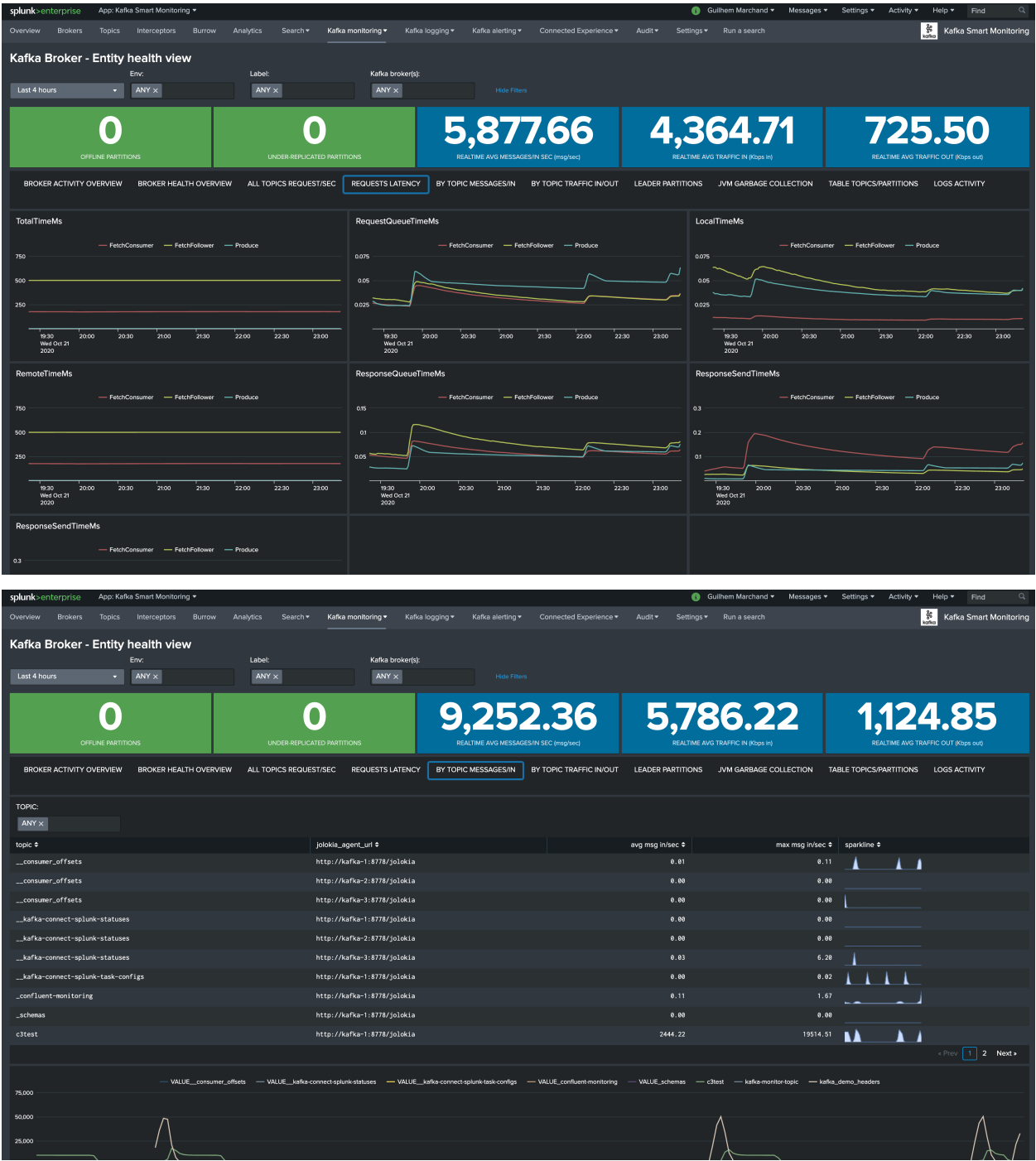


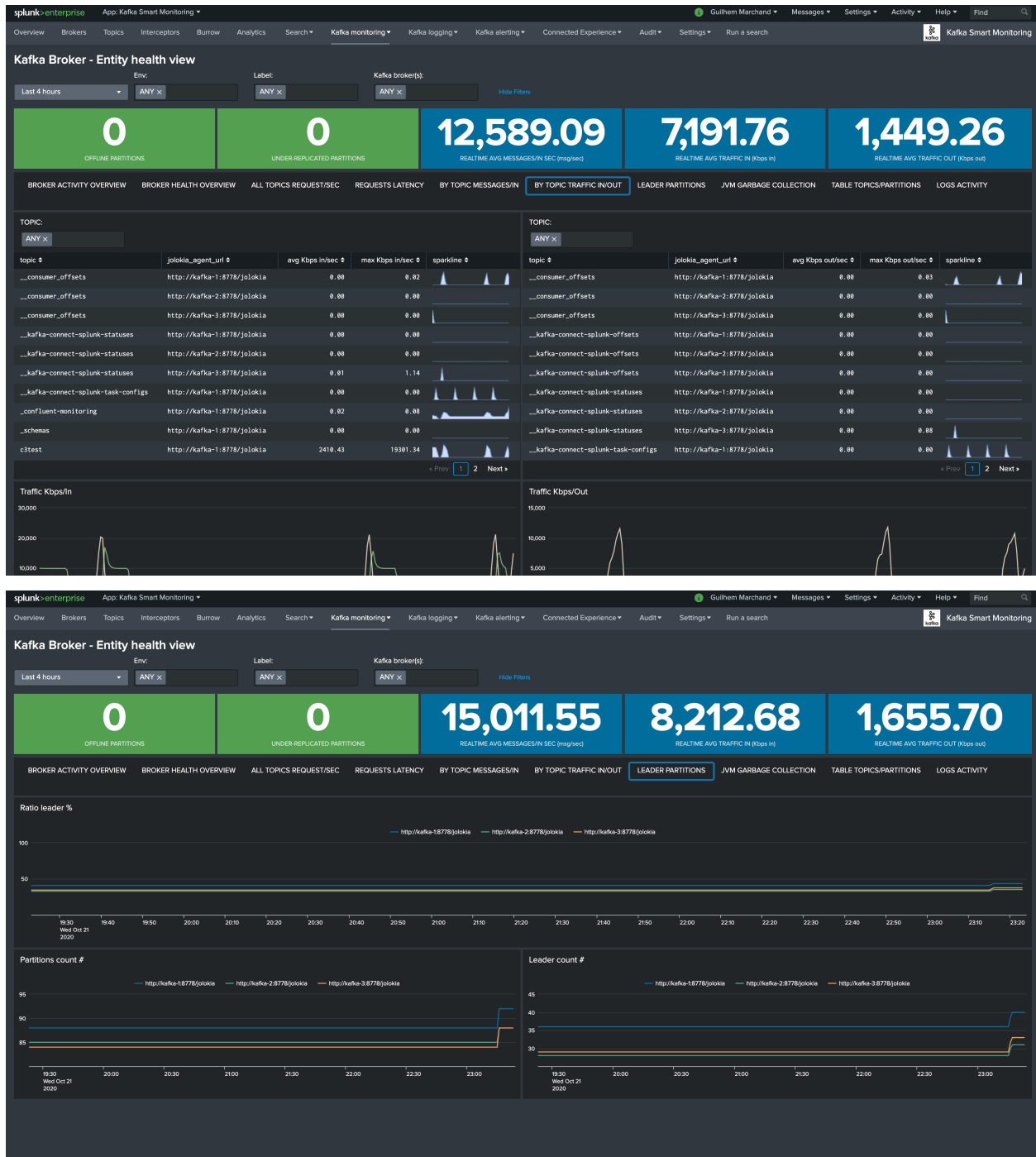


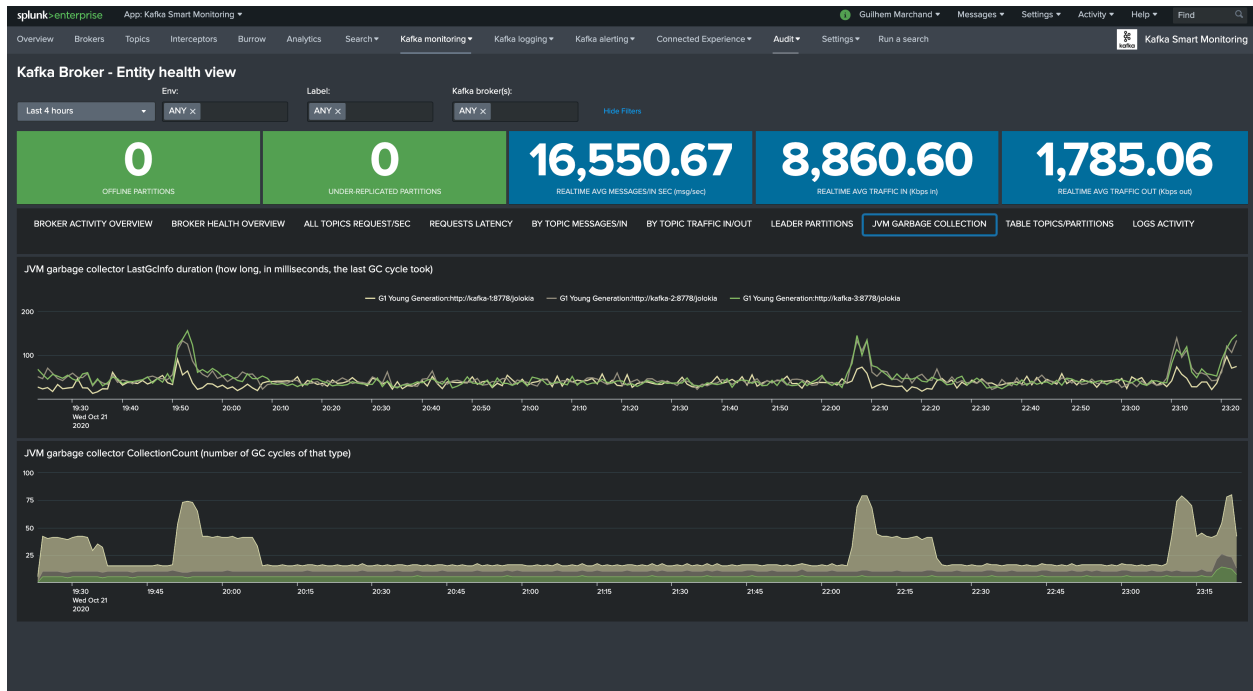
2.4.5 Kafka broker dashboard view











Kafka Broker - Entity health view

Env: Last 4 hours ANY x Label: ANY x Kafka broker(s): ANY x Hide Filters

0
OFFLINE PARTITIONS

0
UNDER REPLICATED PARTITIONS

16,550.67
REALTIME AVG MESSAGES/IN SEC (msg/sec)

8,860.60
REALTIME AVG TRAFFIC IN (Kops in)

1,785.06
REALTIME AVG TRAFFIC OUT (Kops out)

BROKER ACTIVITY OVERVIEW BROKER HEALTH OVERVIEW ALL TOPICS REQUEST/SEC REQUESTS LATENCY BY TOPIC MESSAGES/IN BY TOPIC TRAFFIC IN/OUT LEADER PARTITIONS JVM GARBAGE COLLECTION **TABLE TOPICS/PARTITIONS** LOGS ACTIVITY

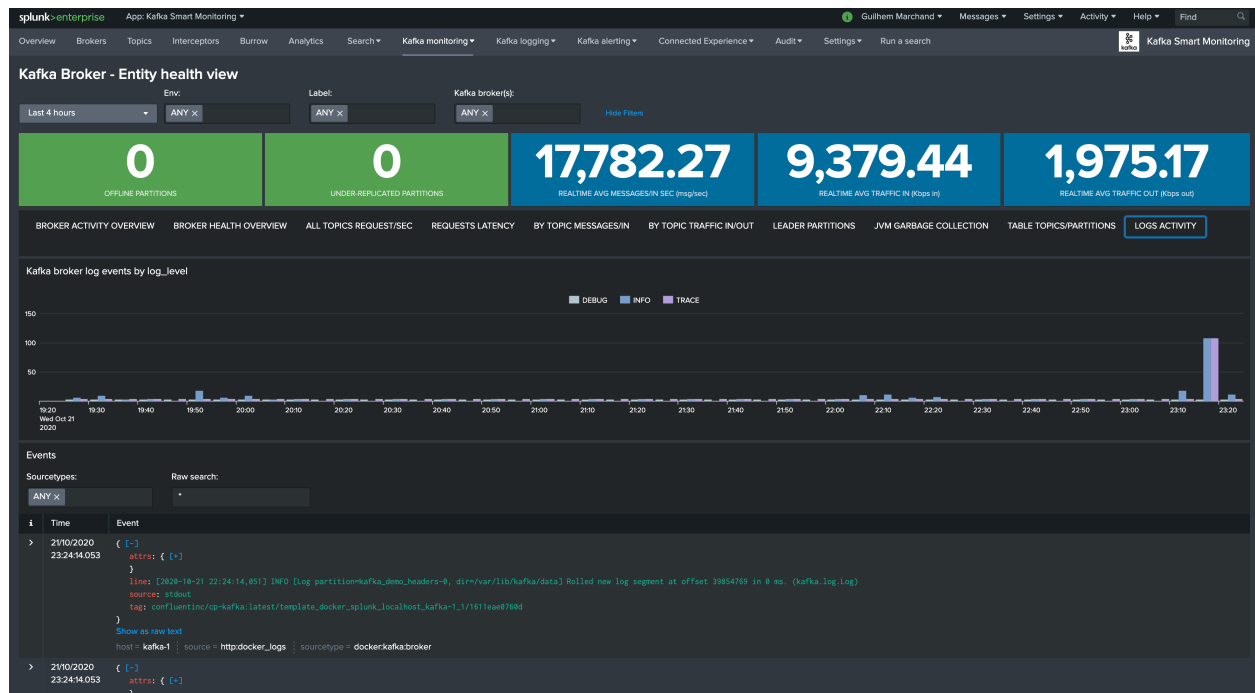
11
TOTAL TOPICS

268
TOTAL PARTITIONS

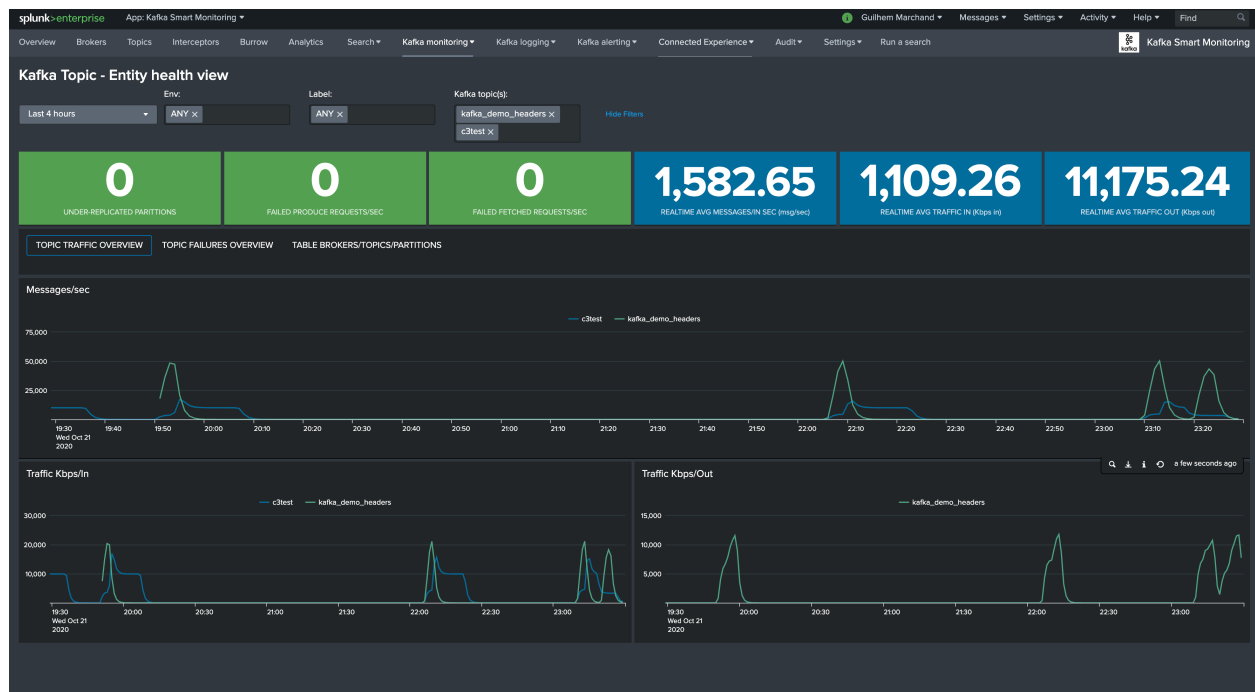
Table Topics/Partitions details

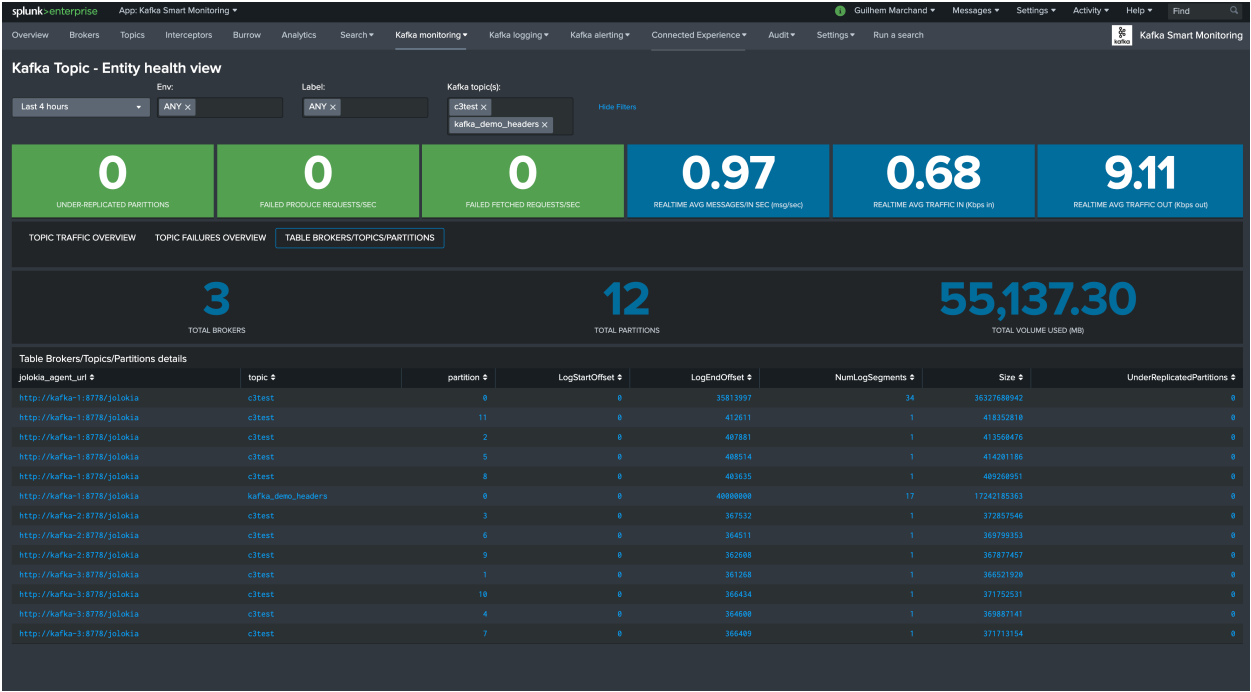
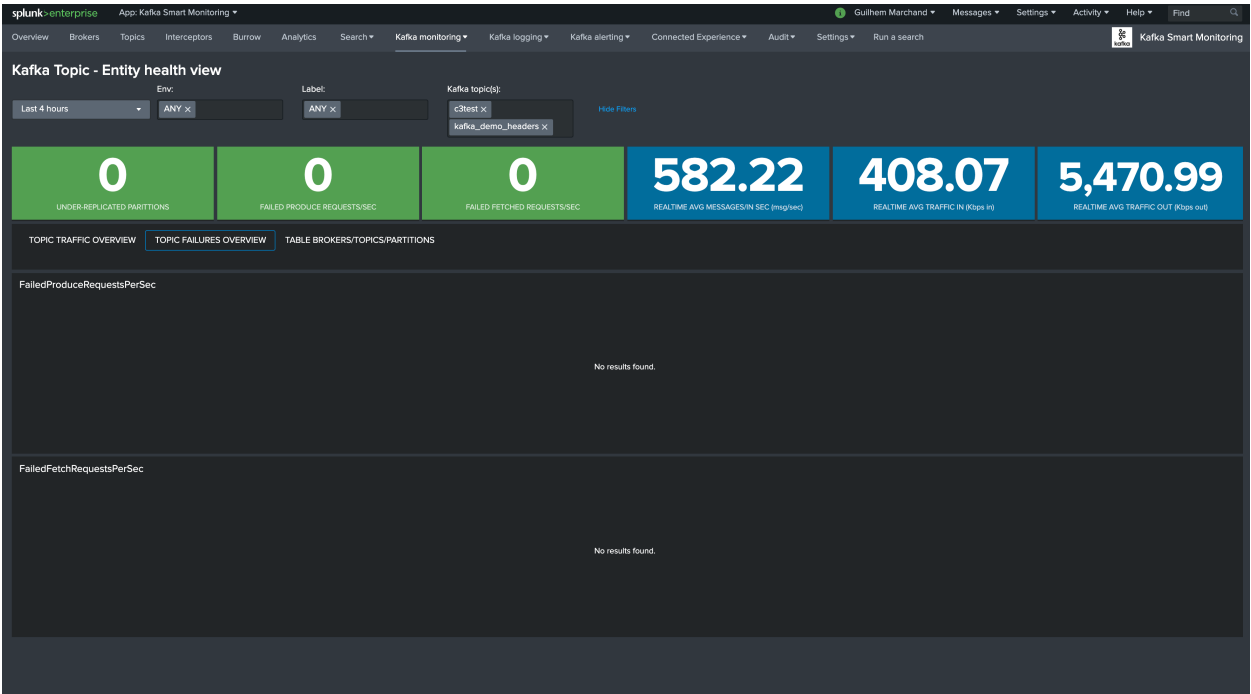
TOPIC: ANY x

jolokia_agent_url #	topic #	partition #	LogStartOffset #	LogEndOffset #	NumLogSegments #	Size #	UnderReplicatedPartitions #	#
http://kafka-1:8778/jolokia	__consumer_offsets	0	0	0	1	0	0	✓
http://kafka-2:8778/jolokia	__consumer_offsets	0	0	0	1	0	0	✓
http://kafka-3:8778/jolokia	__consumer_offsets	0	0	0	1	0	0	✓
http://kafka-1:8778/jolokia	__consumer_offsets	1	0	0	1	0	0	✓
http://kafka-2:8778/jolokia	__consumer_offsets	1	0	0	1	0	0	✓
http://kafka-3:8778/jolokia	__consumer_offsets	1	0	0	1	0	0	✓
http://kafka-1:8778/jolokia	__consumer_offsets	2	0	0	1	0	0	✓
http://kafka-2:8778/jolokia	__consumer_offsets	2	0	0	1	0	0	✓
http://kafka-3:8778/jolokia	__consumer_offsets	2	0	0	1	0	0	✓
http://kafka-1:8778/jolokia	__consumer_offsets	3	0	0	1	0	0	✓
http://kafka-2:8778/jolokia	__consumer_offsets	3	0	0	1	0	0	✓
http://kafka-3:8778/jolokia	__consumer_offsets	3	0	0	1	0	0	✓
http://kafka-1:8778/jolokia	__consumer_offsets	4	0	0	1	0	0	✓
http://kafka-2:8778/jolokia	__consumer_offsets	4	0	0	1	0	0	✓

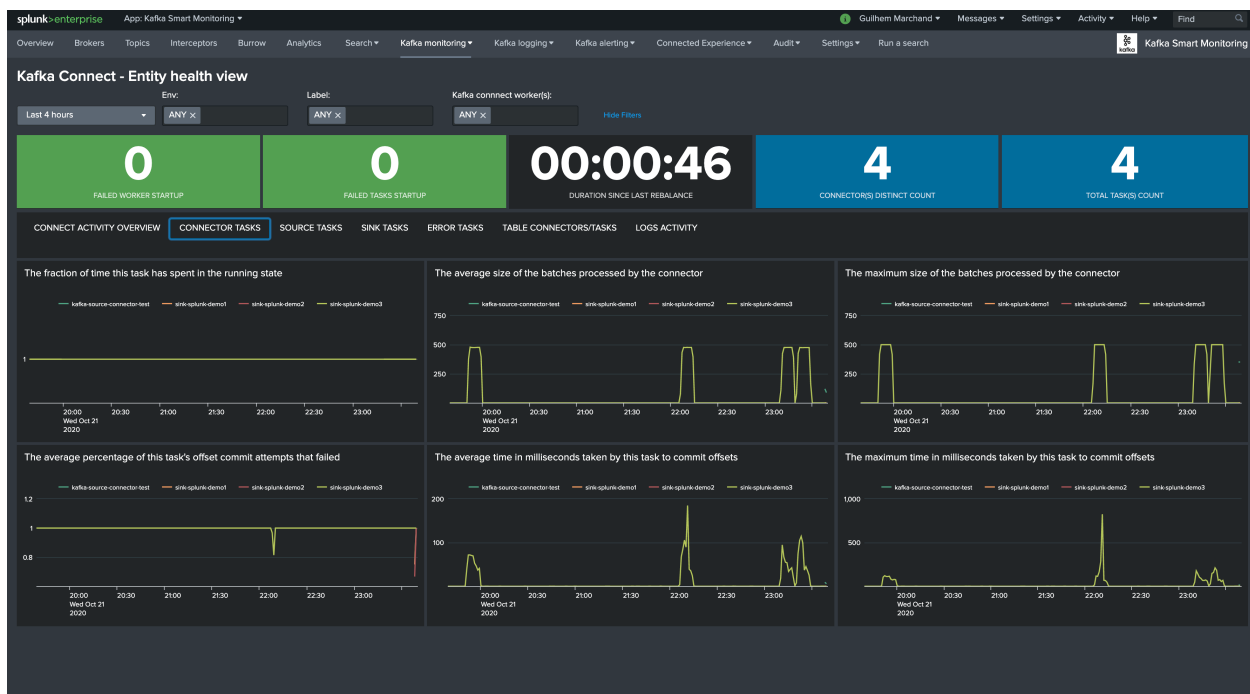
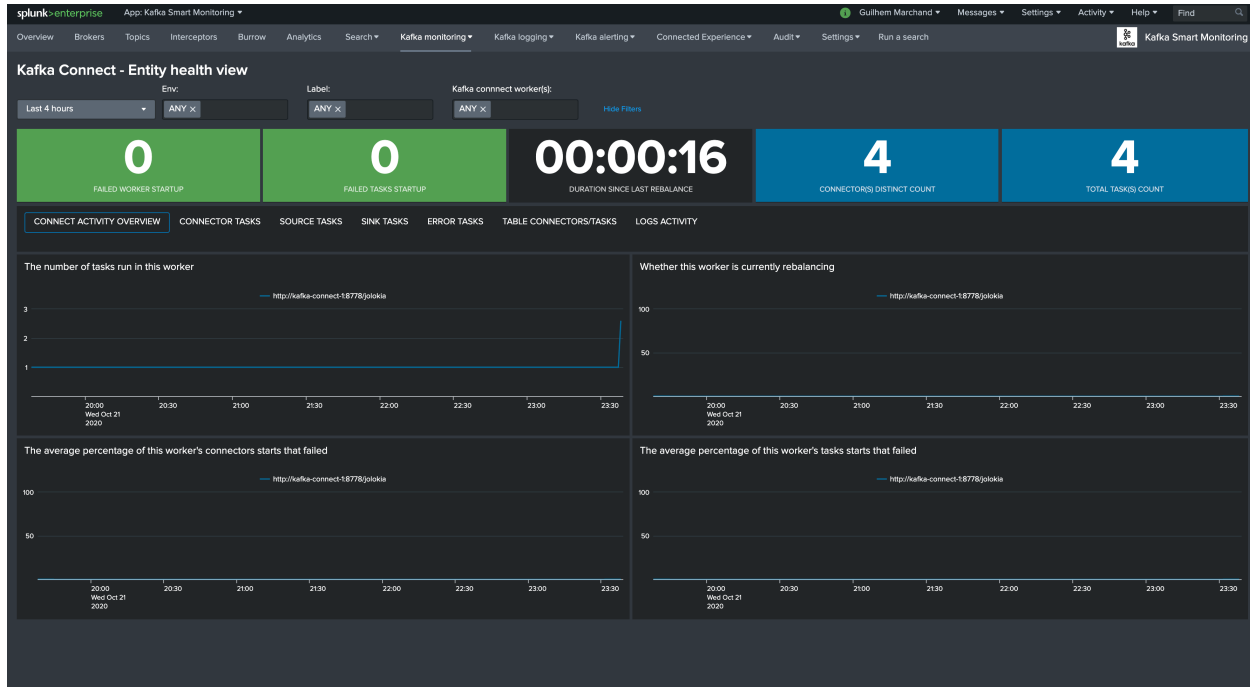


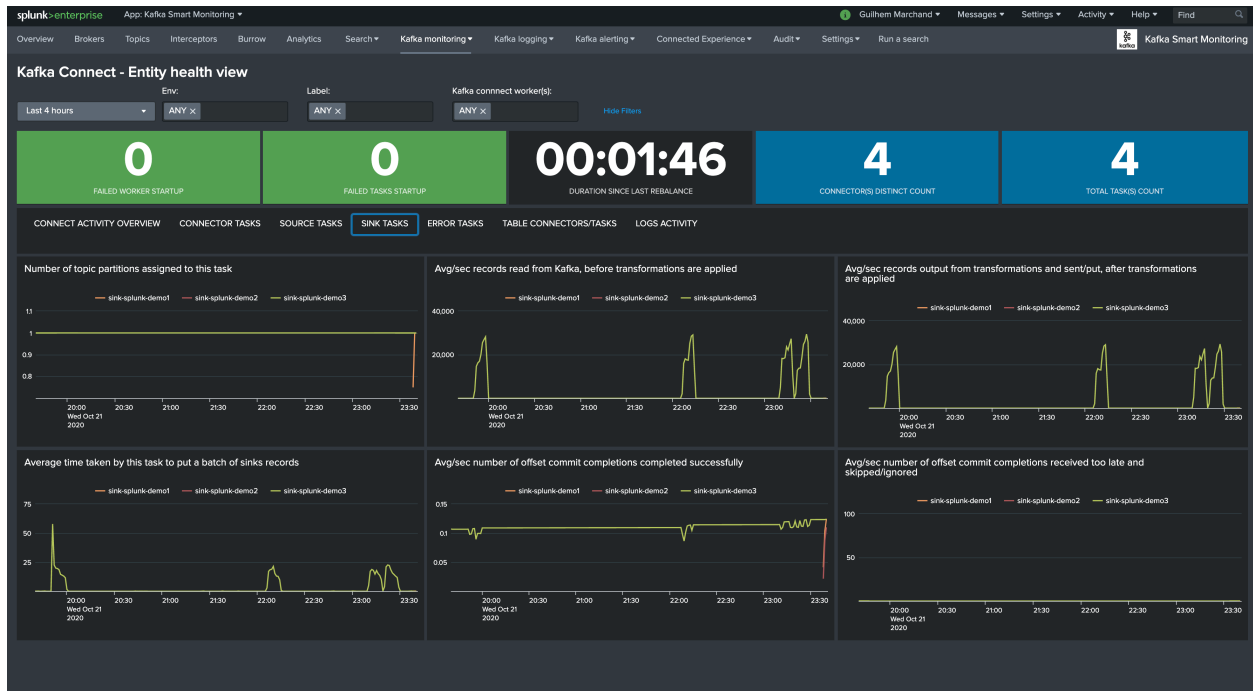
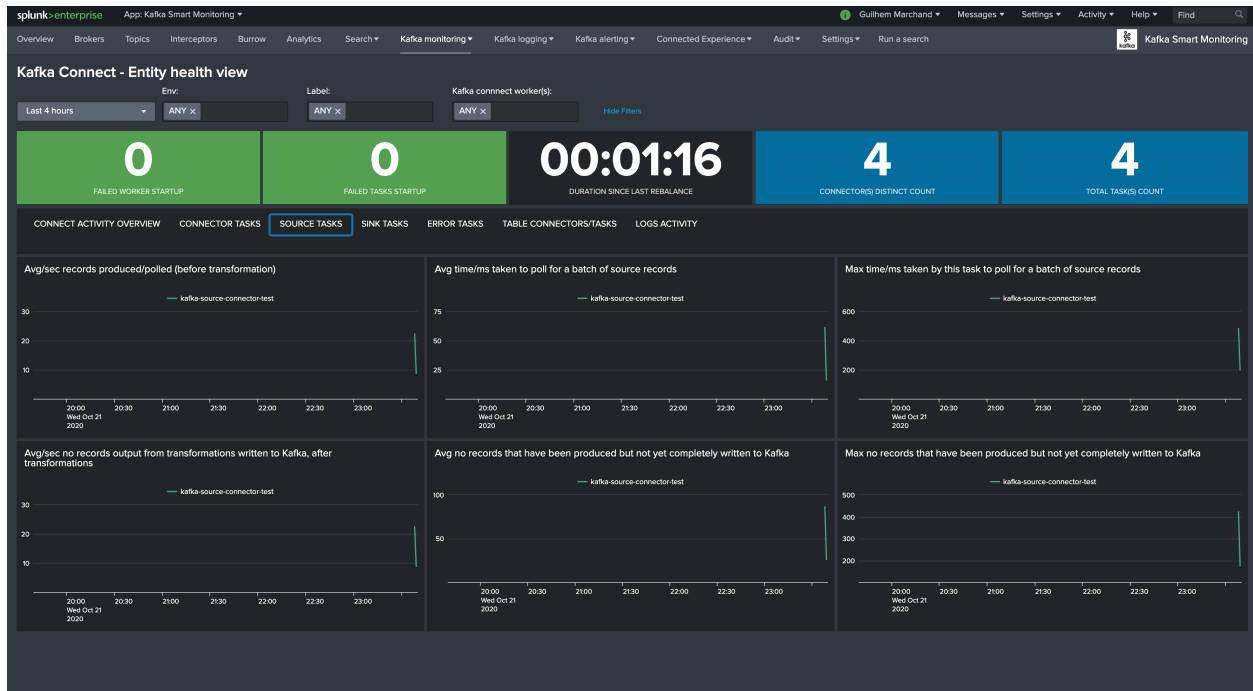
2.4.6 Kafka topic dashboard view

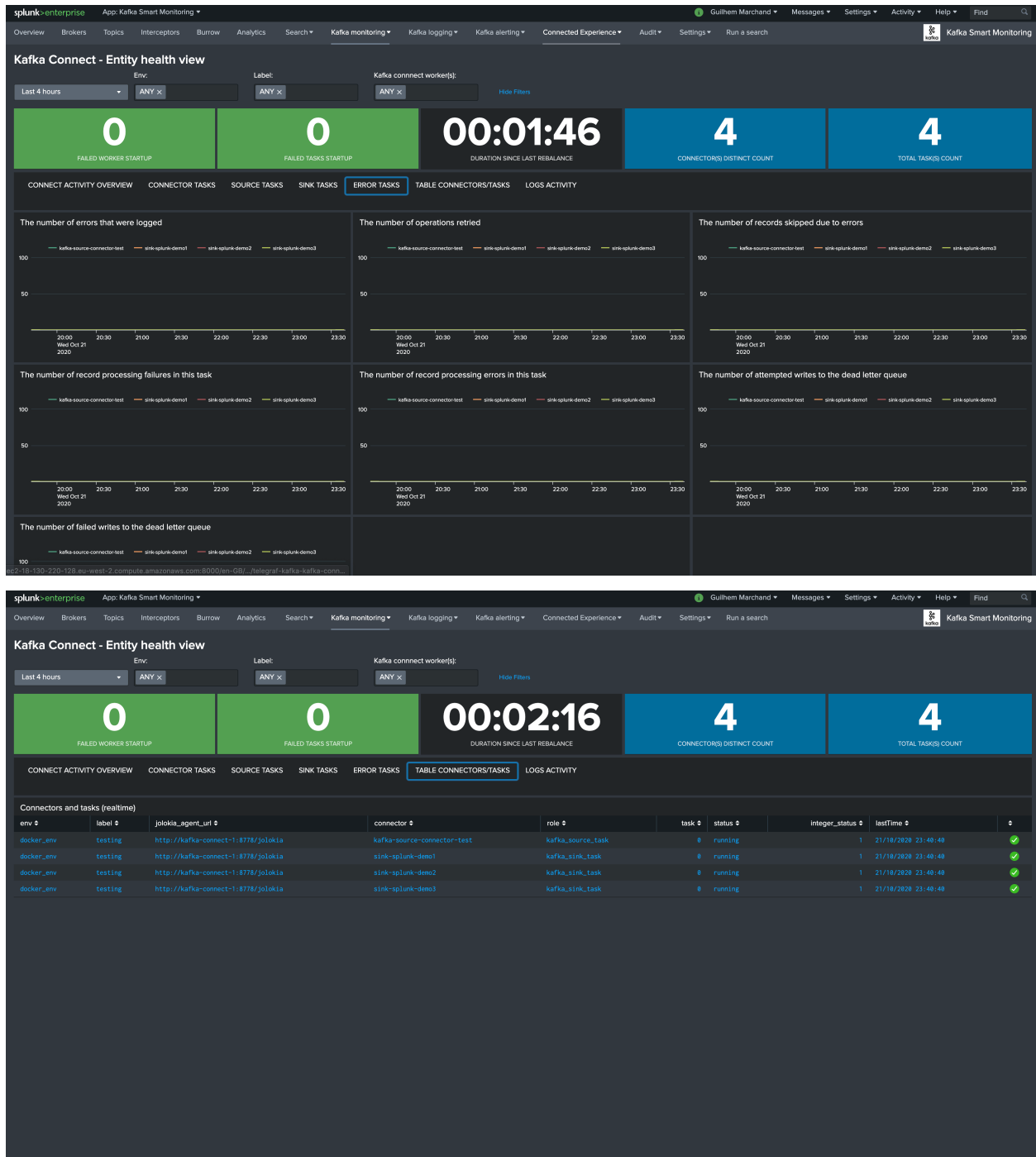


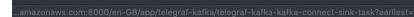


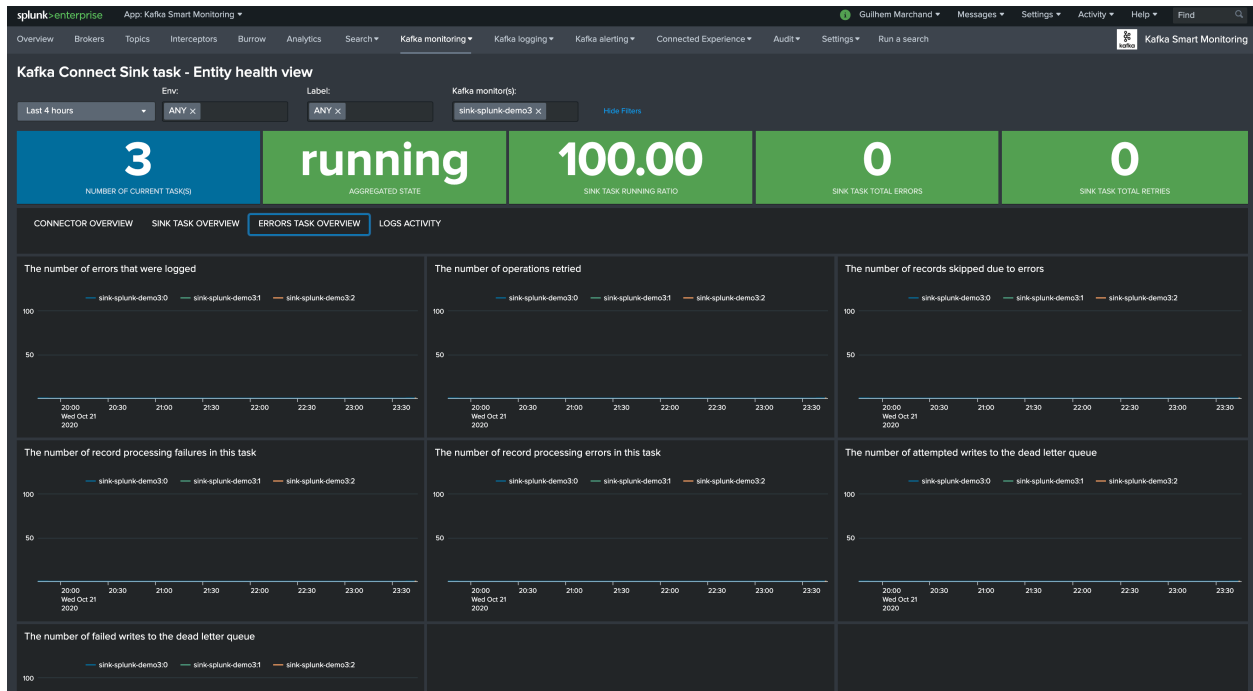
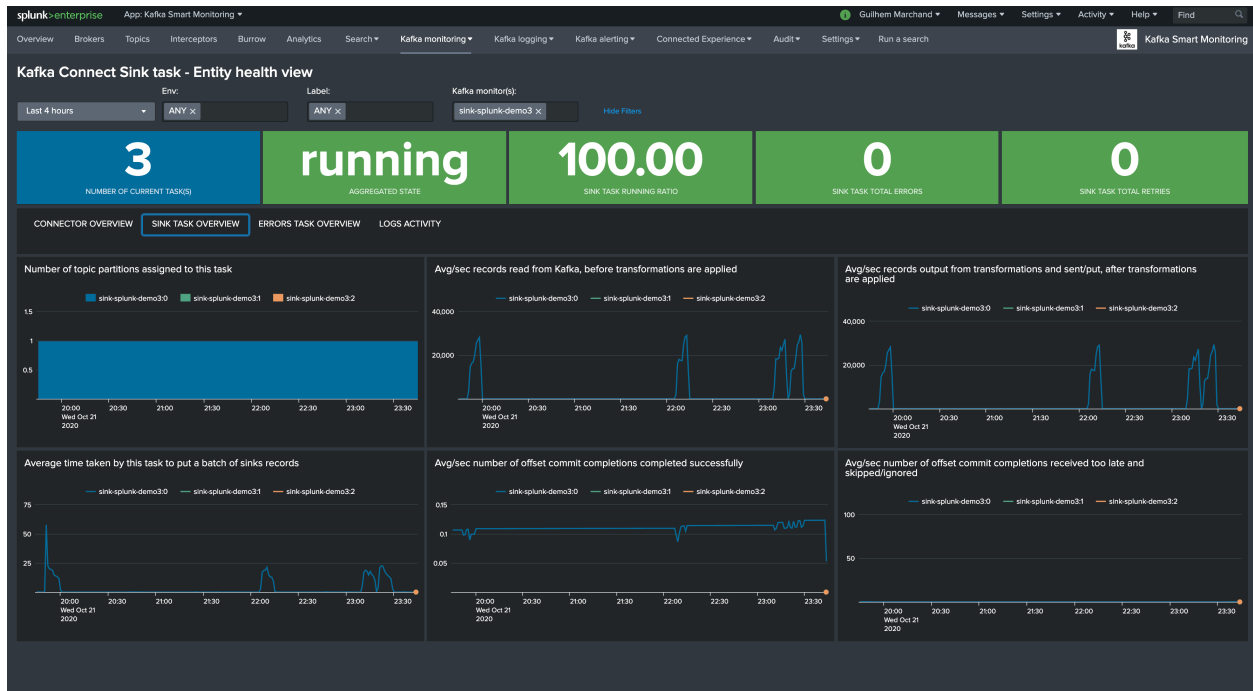
2.4.7 Kafka connect dashboard view

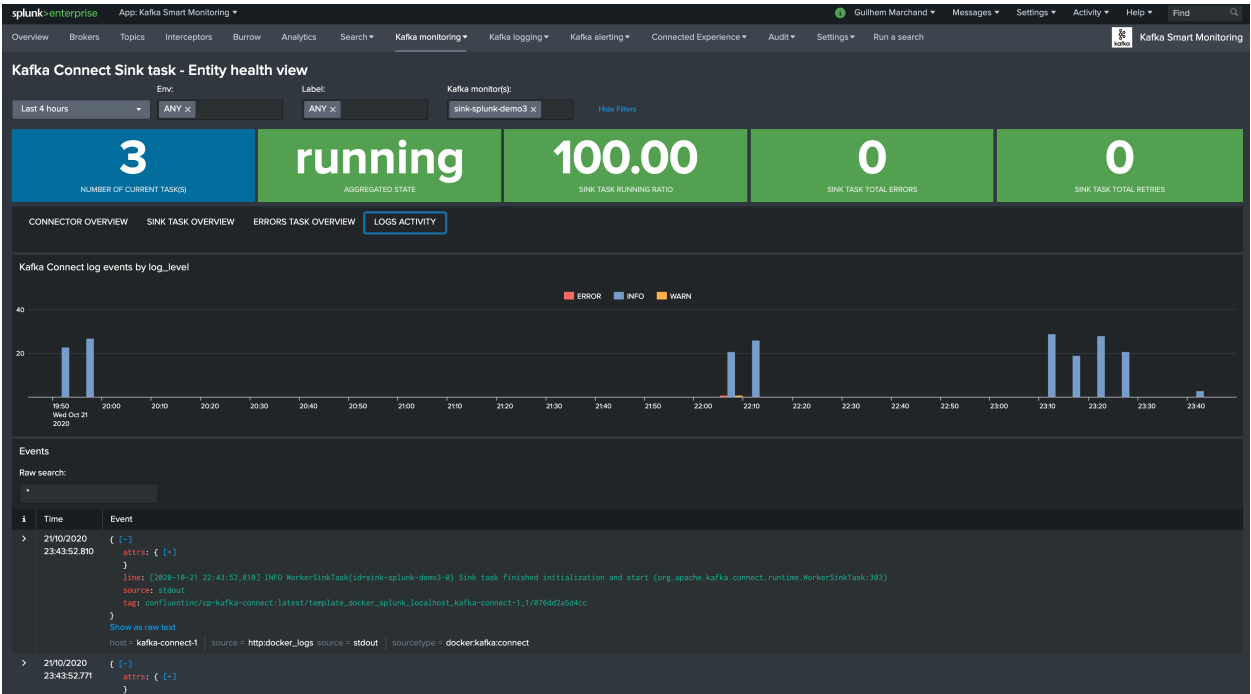




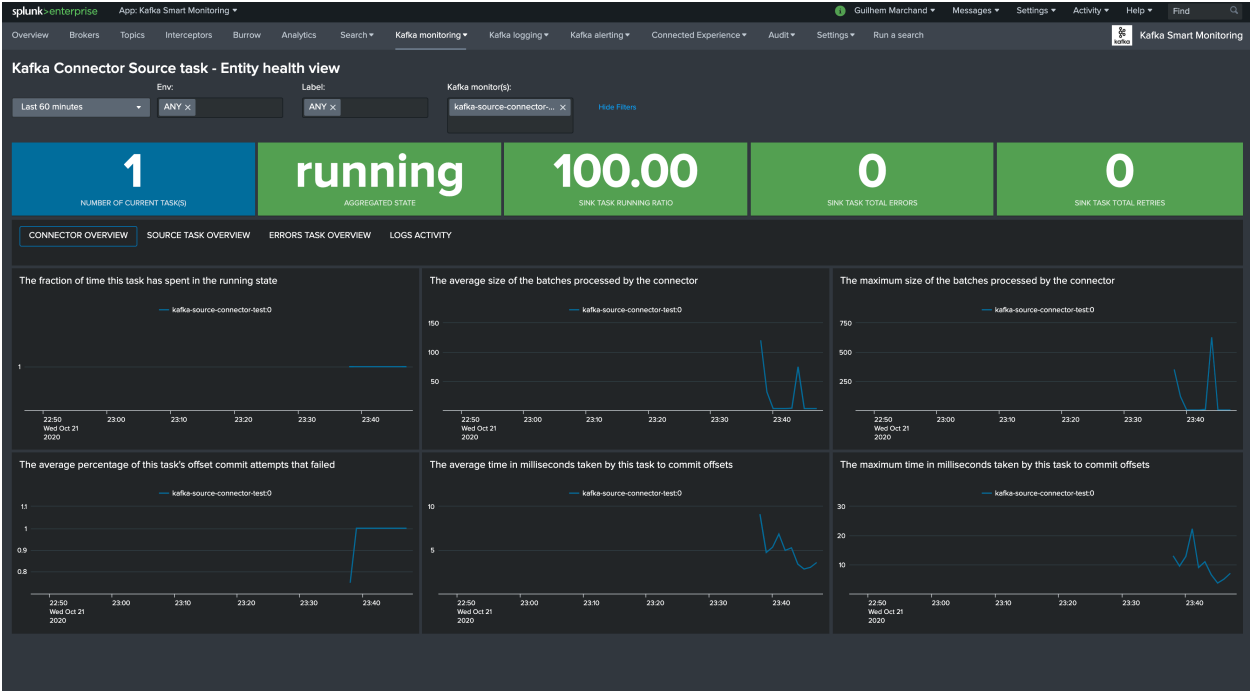


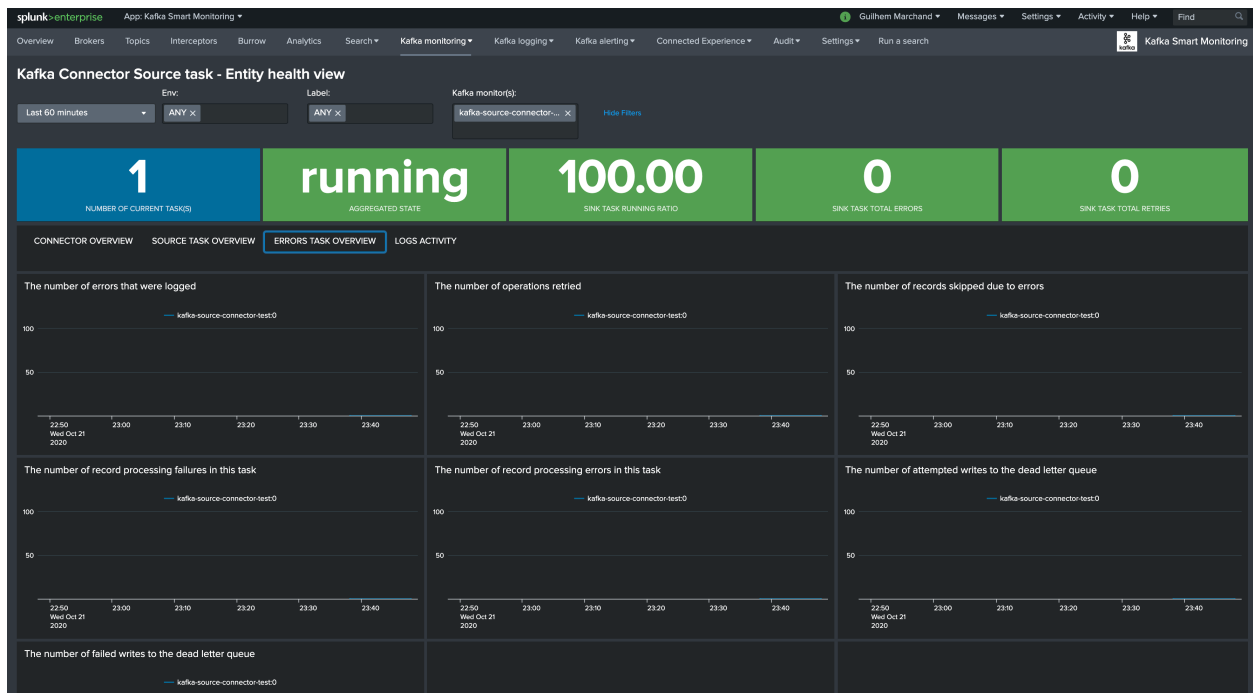
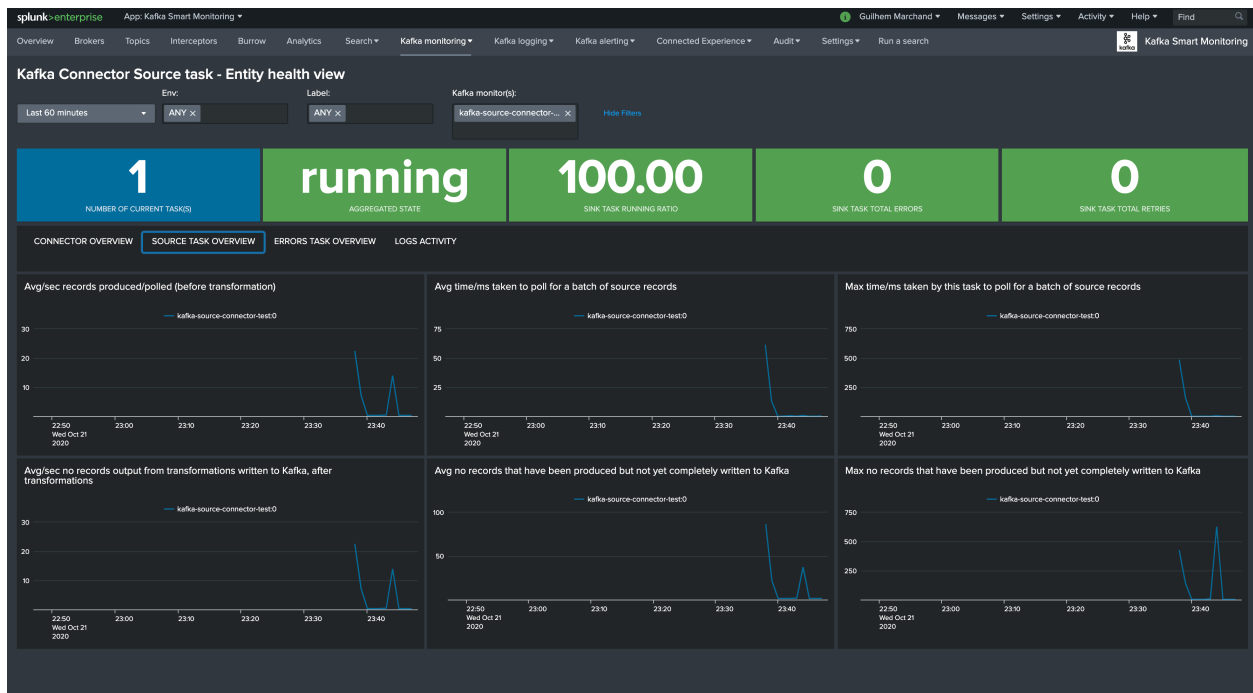


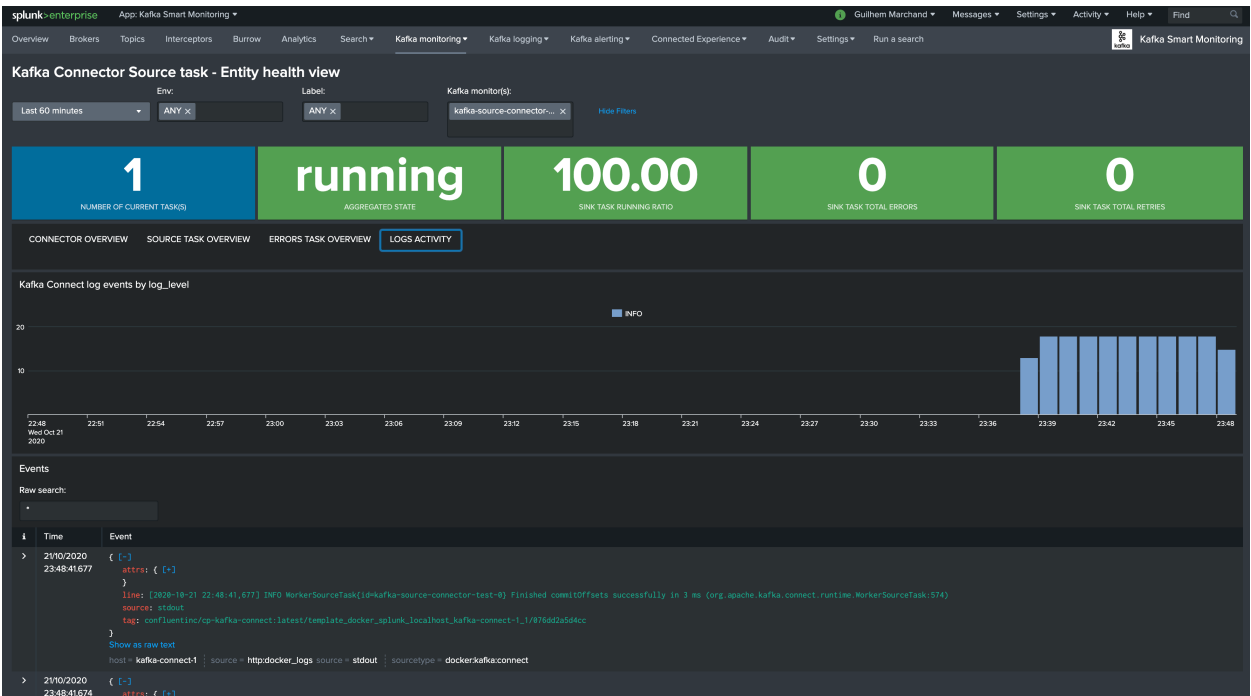




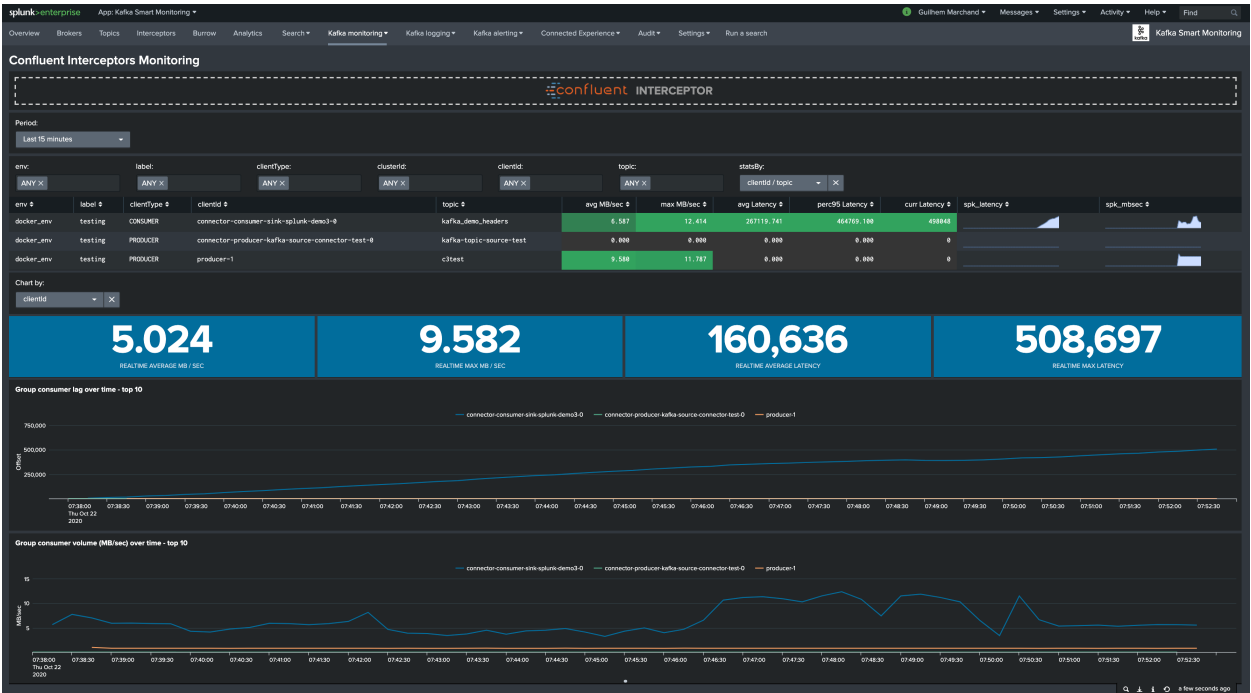
2.4.9 Kafka connect source task dashboard view



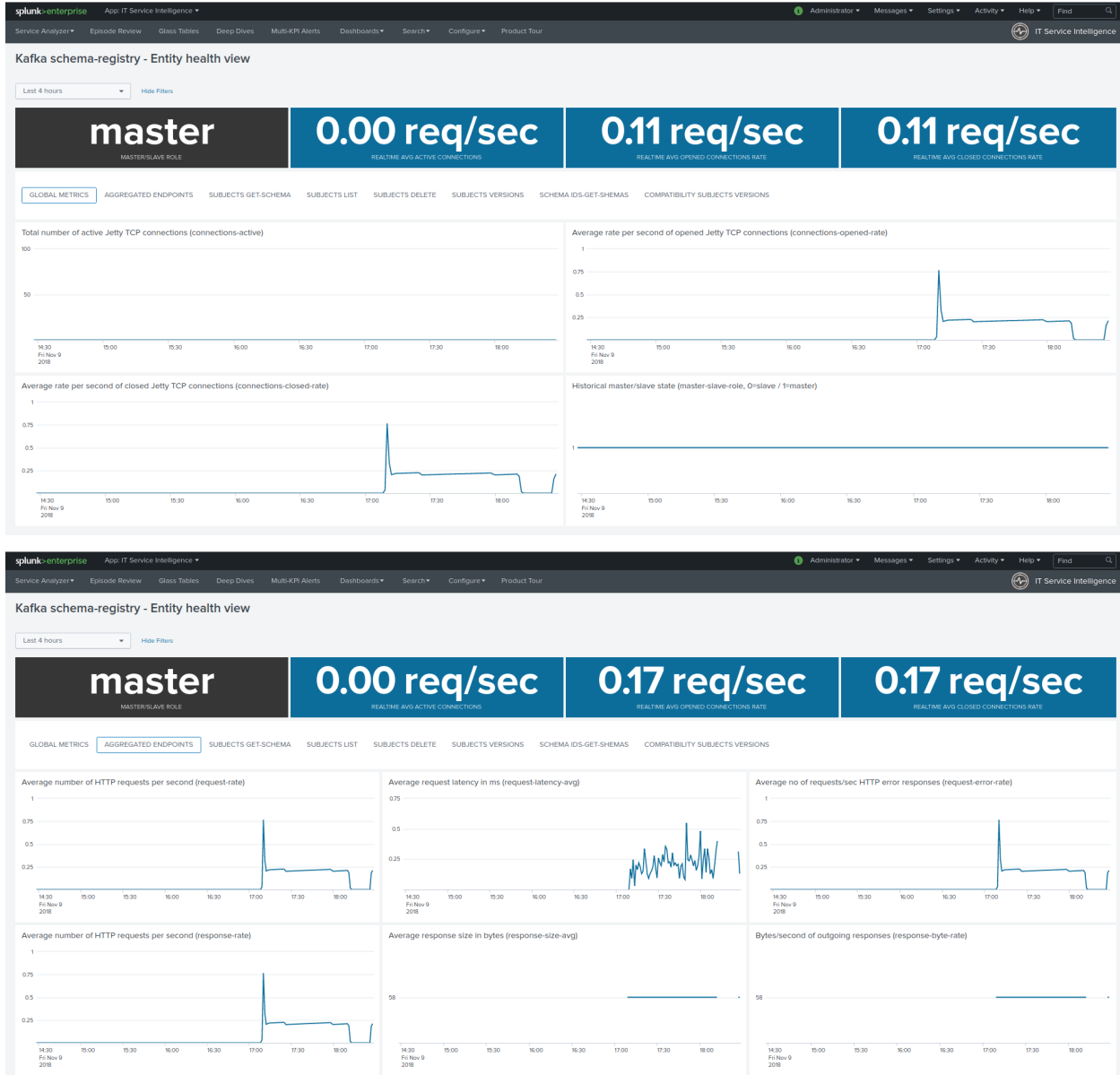




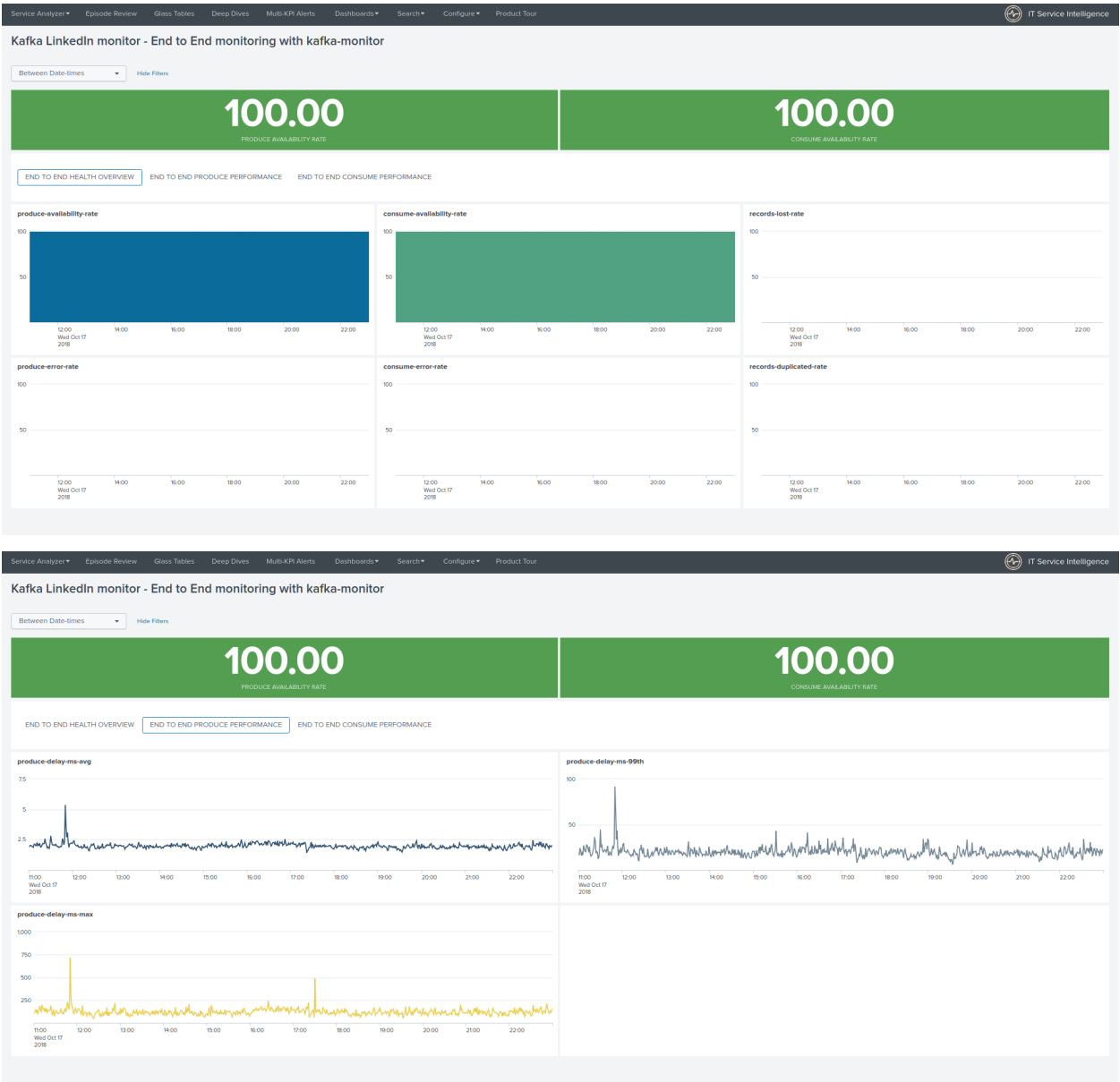
2.4.10 Confluent Interceptors Monitoring

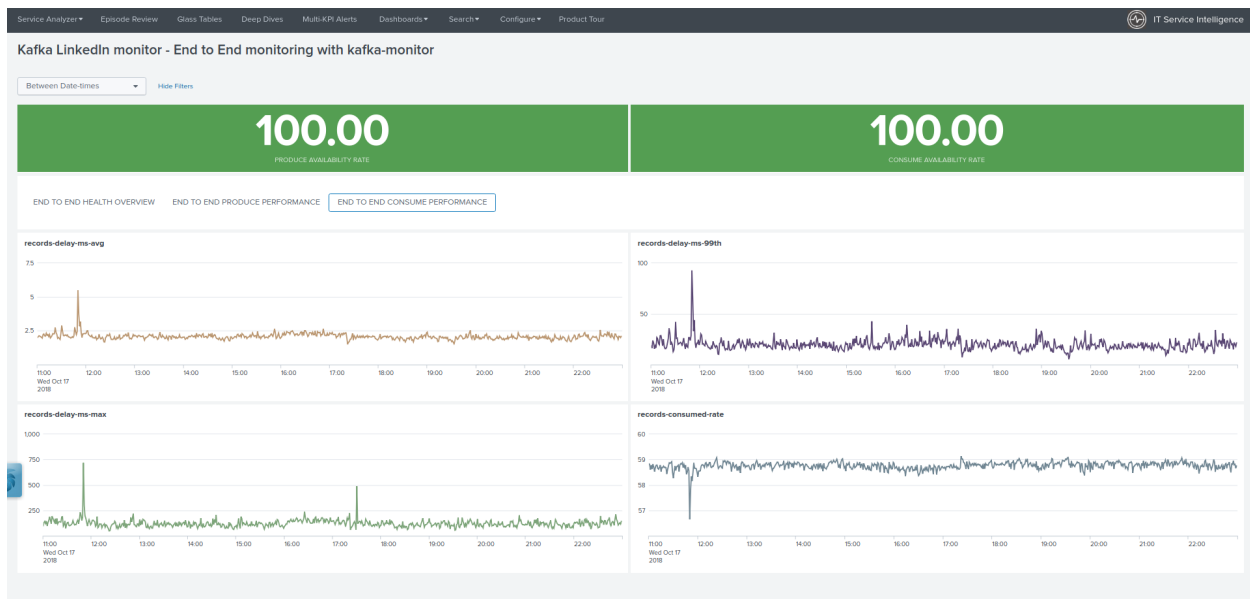


2.4.11 Confluent schema-registry dashboard view

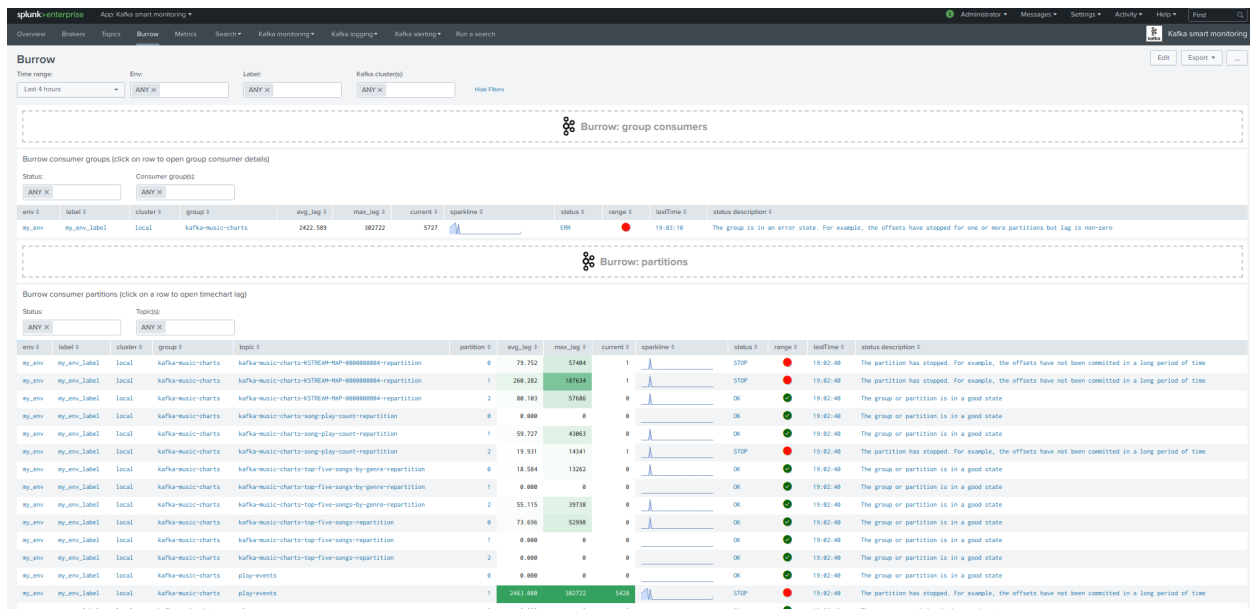


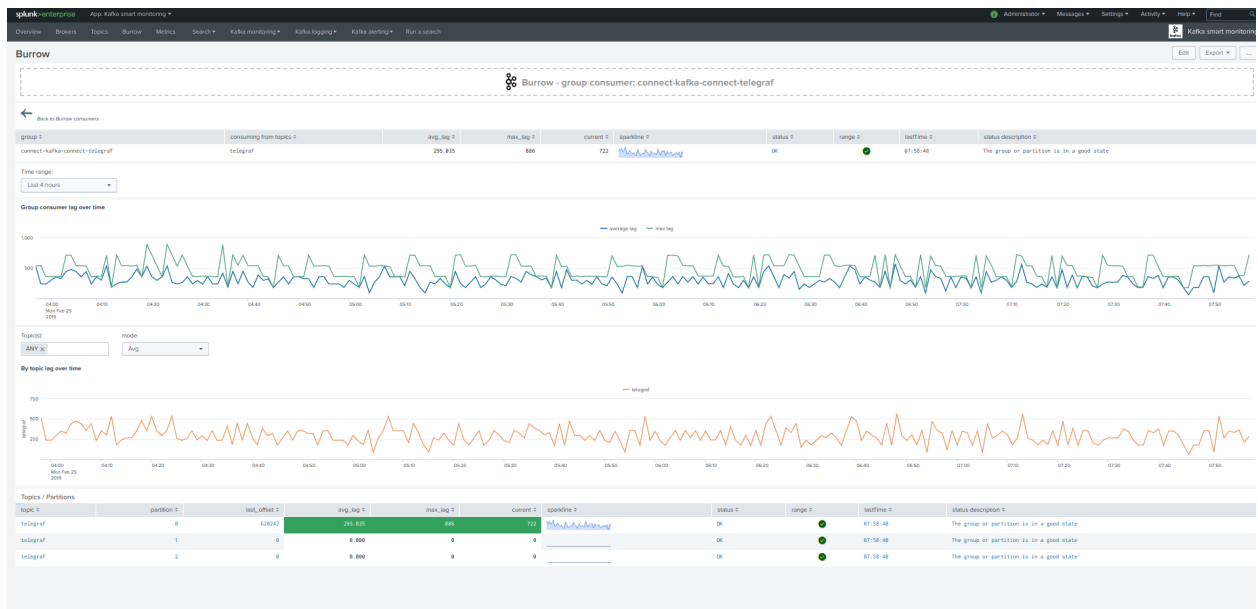
2.4.12 Xinfra Kafka monitor view



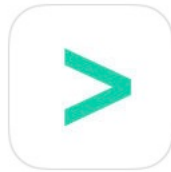


2.4.13 Burrow Kafka Consumers lagging view





2.5 Splunk Connected experience for Cloud Gateway

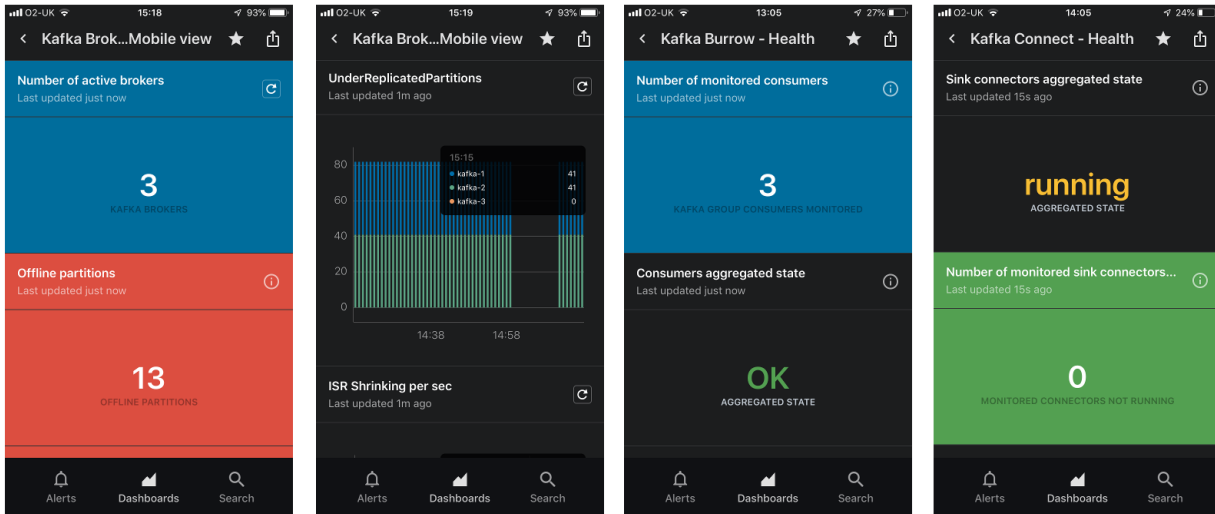


Connected experience dashboards are builtin dashboards optimised for visualization on Splunk Mobile and Splunk TV with Splunk Cloud gateway:

<https://docs.splunk.com/Documentation/Mobile>

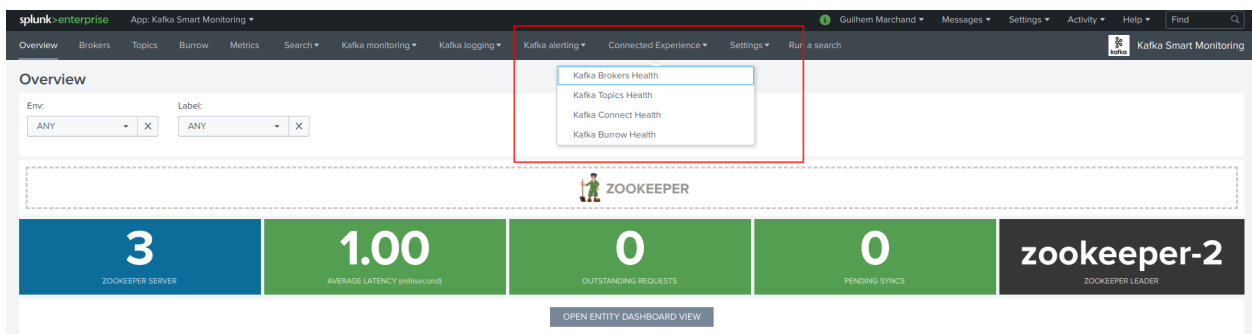
Using Splunk Cloud Gateway and the Connected Experience dashboards, you can easily send data to your mobile Apple TV and device users with compatible and optimized dashboards.

Screenshots from some of the Connected Experience dashboards in Apple Iphone devices:

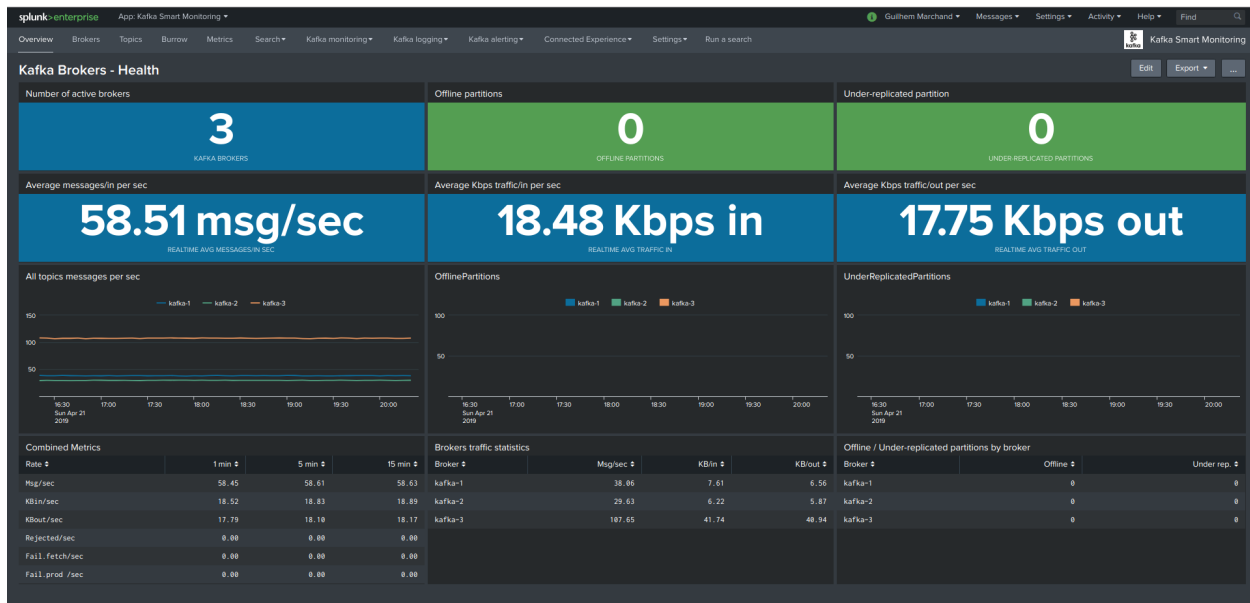


2.5.1 Access the Connected Experience dashboards

The builtin Connected Experience dashboards are available from the “Connected Experience” menu in the Splunk application:



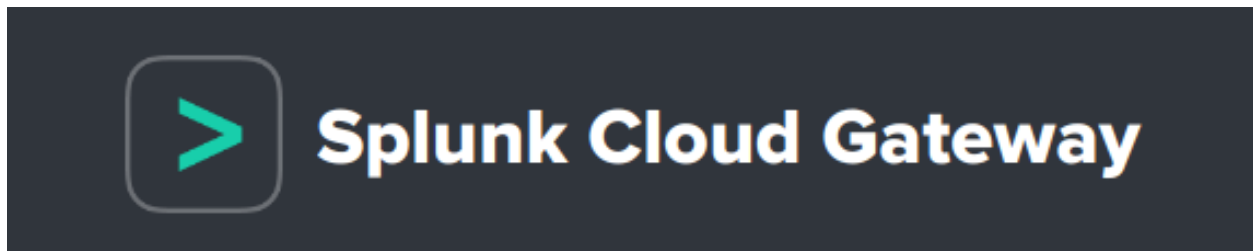
Although these dashboards are specially designed for the Splunk Connected experience, these are entirely compatible with Splunk Web:



For these dashboards to be available from your Apple devices, the builtin permissions share the dashboards to the global level of the Splunk search instance(s). (See metadata/default.meta)

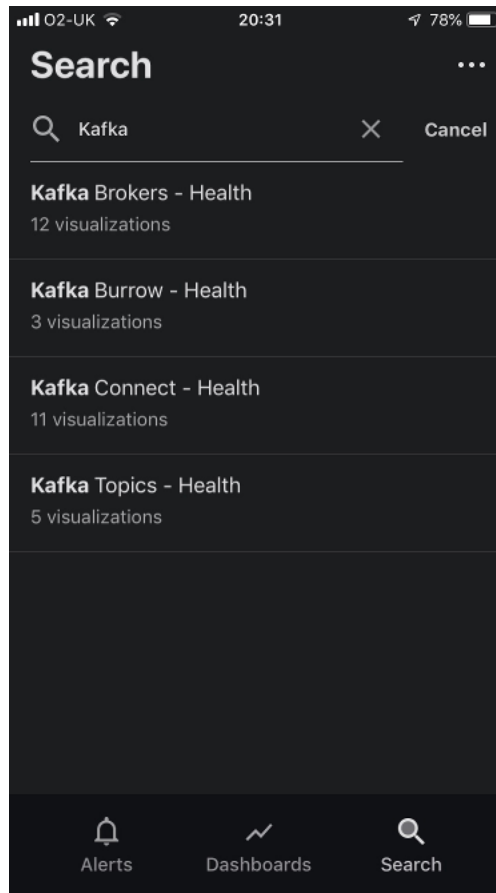
2.5.2 Deploy and configure Splunk Cloud Gateway

Download, install and configure the Splunk Cloud Gateway application in your environment:



<https://splunkbase.splunk.com/app/4250/>

Once you configured Splunk Cloud Gateway and registered a device, you can search for the Kafka Connected Experience dashboards:



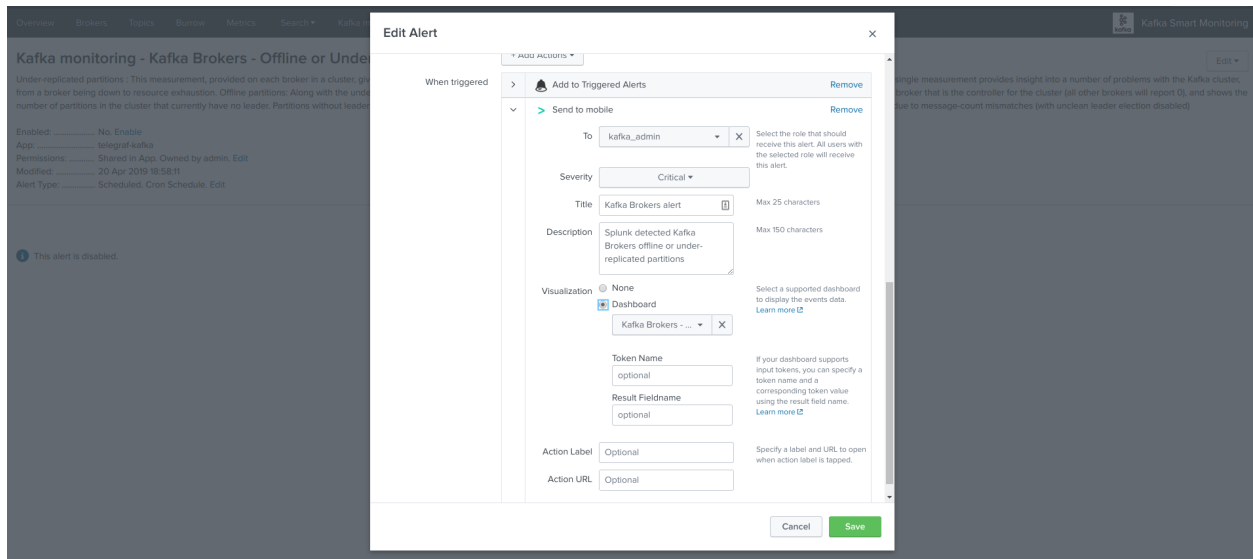
Open any of the dashboards to start your amazing Splunk Connected journey!

2.5.3 Send to mobile alert action

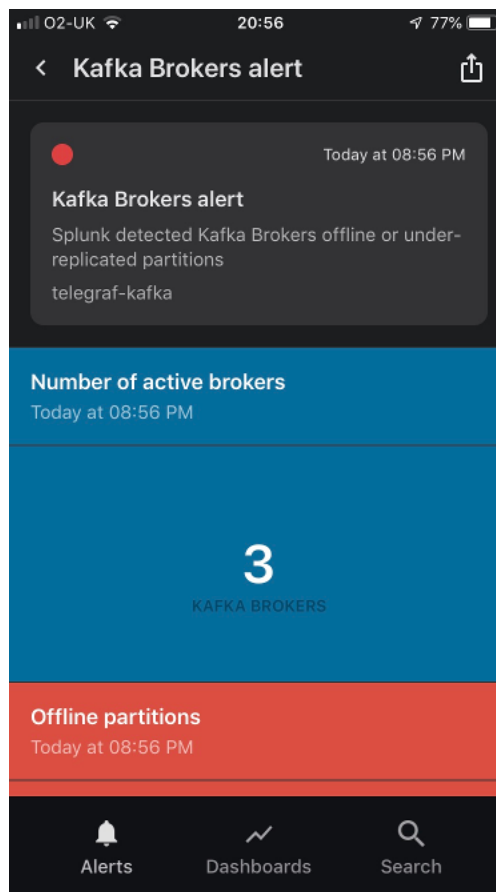
With Splunk Mobile and Splunk Cloud Gateway, you can configure a Kafka alert to send a notification to your apple device, and link with the dashboard of your choice:

<https://docs.splunk.com/Documentation/Alerts/latest/Alerts/SendAlerts>

The application provides a builtin “**kafka_admin**” role that we suggest you use for Kafka alerting, make sure the users that should receive the Kafka alert notifications are members of the role.



Shall an alert trigger, a notification will be sent to the members of the **kafka_admin** role that have a registered device:



2.5.4 Create custom dashboards included in the menu

Because there are limitations in Connected Experience dashboards, you might need to create your own versions if for example you have to manage multiple environments.

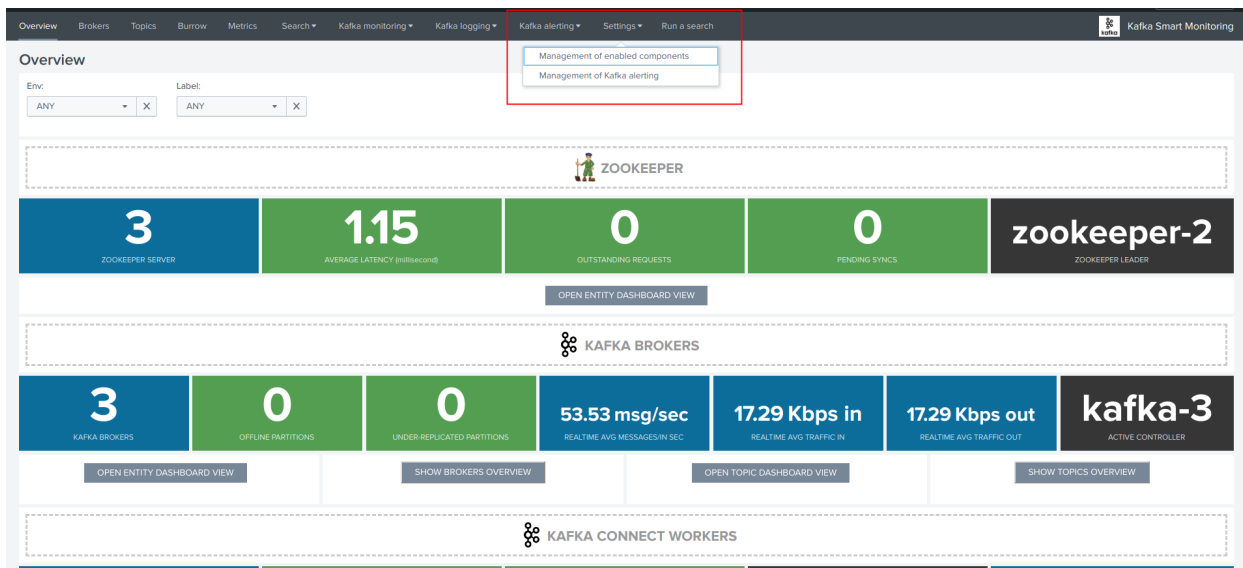
If you need to create custom versions, achieve the following:

- Clone the dashboard you want to customize and make sure the id of the dashboard starts by “cloudgw_”
- Any dashboard starting by “cloudgw_” will automatically be included within the Connected Experience menu
- Modify the dashboard based on your requirements, to focus on a given environment replace any occurrence of `env=* label=*` according to the target
- Make sure to clone the existing permissions of the dashboard, or share the dashboard to all apps to allow it to be visible from the mobile app

2.6 Kafka infrastructure OOTB alerting


The application provides out of the box alerting for all the components of the Kafka and Confluent infrastructure.

Go to app menu / Settings / Management of Kafka alerting:



2.6.1 Management of Kafka alerting (user interface)

The OOTB alerting model relies on several KVstore collections being automatically populated, the user interface “Management of Kafka alerting” allows you to interact easily with different aspects of the monitoring:

Overview Brokers Topics Burrow Metrics Search Kafka monitoring Kafka logging Kafka alerting Settings Run a search  Kafka Smart Monitoring

Kafka Smart Monitoring - Management of embedded alerting framework

Use this interface to manage kafka infrastructure alerting for Kafka

DISABLED

MAINTENANCE MODE STATUS

Enable maintenance mode

Disable maintenance mode

ALERT CONFIGURATION SUMMARY STALE METRICS MONITORING PER COMPONENT STALE METRICS MONITORING PER NUMBER OF NODES KAFKA TOPICS MONITORING KAFKA CONNECT TASKS MONITORING KAFKA CONSUMER GROUP MONITORING

Kafka infrastructure monitoring help - Summary configuration of embedded alerts

Embedded alerts: click on a table row to access object contextual actions

[Refresh this table](#)

title	cron_schedule	schedule_window	alertsuppress.fields	alertsuppress.period	disabled	next_scheduled_time	range
All Kafka components - active node numbers - stale metrics life test	*/* * * * *	0	env, label, role	4h	0	2019-04-14 11:30:00 UTC	🟢
Kafka monitoring - Burrow - group consumers state monitoring	*/* * * * *	0	env, label, cluster, group	4h	0	2019-04-14 11:30:00 UTC	🟢
Kafka monitoring - Confluent kafka-rest - stale metrics life test	*/* * * * *	0	env, label, name	4h	0	2019-04-14 11:30:00 UTC	🟢
Kafka monitoring - Confluent ksqi-server - stale metrics life test	*/* * * * *	0	env, label, name	4h	0	2019-04-14 11:30:00 UTC	🟢
Kafka monitoring - Confluent schema-registry - stale metrics life test	*/* * * * *	0	env, label, name	4h	0	2019-04-14 11:30:00 UTC	🟢
Kafka monitoring - Kafka Brokers - Abnormal number of Active Controllers (2 minutes grace period)	*/* * * * *	0	env, label, kafka_broker	4h	0	2019-04-14 11:30:00 UTC	🟢
Kafka monitoring - Kafka Brokers - Failed producer or consumer was detected	*/* * * * *	0	env, label, kafka_broker, metric_name	4h	0	2019-04-14 11:30:00 UTC	🟢
Kafka monitoring - Kafka Brokers - ISR Shrinking detection	*/* * * * *	0	env, label, kafka_broker	4h	0	2019-04-14 11:30:00 UTC	🟢
Kafka monitoring - Kafka Brokers - Offline or Under-replicated partitions	*/* * * * *	0	env, label, kafka_broker, metric_name	4h	0	2019-04-14 11:30:00 UTC	🟢
Kafka monitoring - Kafka Brokers - stale metrics life test	*/* * * * *	0	env, label, name	4h	0	2019-04-14 11:30:00 UTC	🟢
Kafka monitoring - Kafka Connect - connector or task startup failure detected	*/* * * * *	0	env, label, connector	4h	0	2019-04-14 11:30:00 UTC	🟢
Kafka monitoring - Kafka Connect - stale metrics life test	*/* * * * *	0	env, label, name	4h	0	2019-04-14 11:30:00 UTC	🟢
Kafka monitoring - Kafka Connect - tasks status monitoring	*/* * * * *	0	env, label, connector	4h	0	2019-04-14 11:30:00 UTC	🟢
Kafka monitoring - Kafka Topics - Under-replicated partitions detected on topic	*/* * * * *	0	env, label, topic	4h	0	2019-04-14 11:30:00 UTC	🟢
Kafka monitoring - Kafka Topics - errors detected on a topic	*/* * * * *	0	env, label, topic	4h	0	2019-04-14 11:30:00 UTC	🟢
Kafka monitoring - LinkedIn Kafka Monitor - stale metrics life test	*/* * * * *	0	env, label, name	4h	0	2019-04-14 11:30:00 UTC	🟢
Kafka monitoring - Zookeeper - stale metrics life test	*/* * * * *	0	env, label, name	4h	0	2019-04-14 11:30:00 UTC	🟢

KVstore collections and lookup definitions

The alerting framework relies on several KVstore collections and associated lookup definitions:

Purpose	KVstore collection	Lookup definition
Monitoring per component entity	kv_telegraf_kafka_inventory	kafka_infra_inventory
Monitoring per nodes number	kv_kafka_infra_nodes_inventory	kafka_infra_nodes_inventory
Monitoring of Kafka topics	kv_telegraf_kafka_topics_monitoring	kafka_topics_monitoring
Monitoring per component entity	kv_telegraf_kafka_connect_tasks_monitoring	kafka_connect_tasks_monitoring
Monitoring per Burrow consumers	kv_kafka_burrow_consumers_monitoring	kafka_burrow_consumers_monitoring
Maintenance mode management	kv_telegraf_kafka_alerting_maintenance	kafka_alerting_maintenance

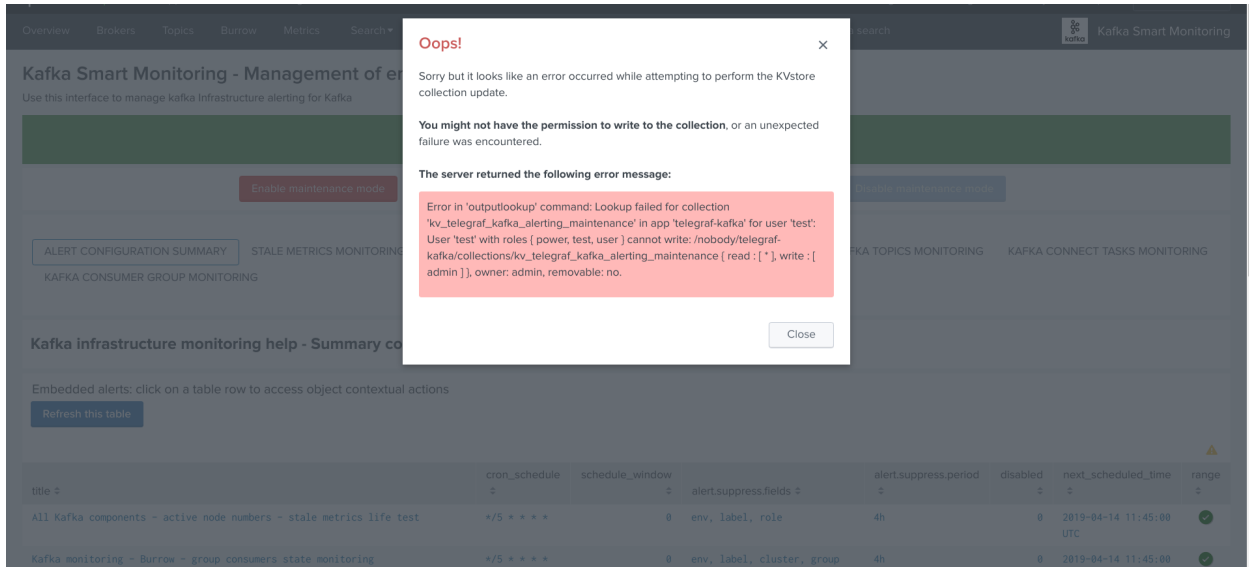
Permissions and authorizations

Managing the alerting framework and its objects require KVstore collection and lookup definition write permissions.

You can rely on the builtin role **kafka_admin** and configure your Kafka administrators to be member of the role.

The role provides the level of permissions required to administrate the KVstore collections.

Shall an unauthorized user attempt to perform an operation, or access to an object that is no readable, the following type of error window will be showed:



Maintenance mode

All alerts are by default driven by the status of the maintenance mode stored in a KVstore collection.

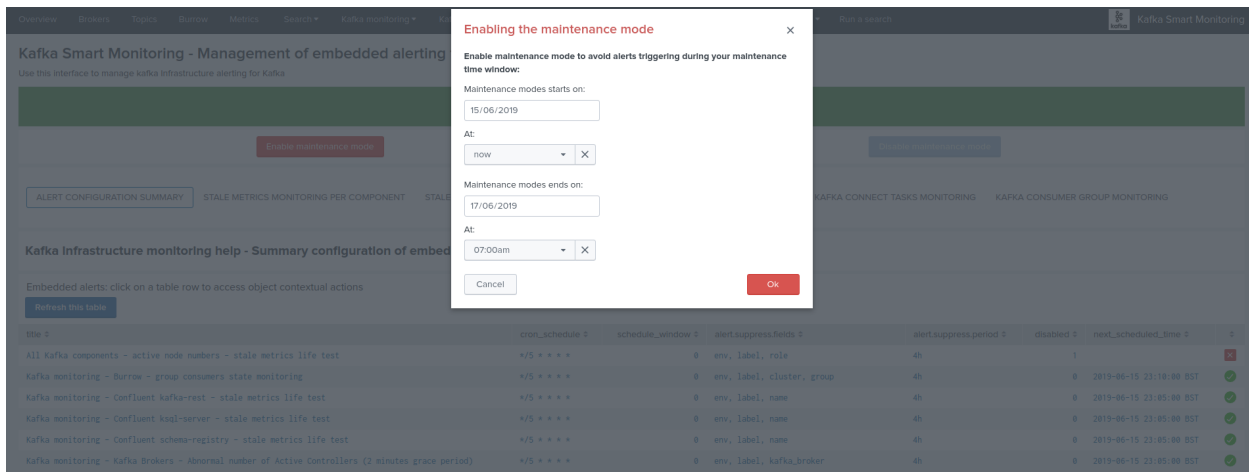
Shall the maintenance be enabled by an administrator, Splunk will continue to run the schedule alerts but none of them will be able to trigger during the maintenance time window.

When the end of maintenance time window is reached, its state will be automatically disabled and alerts will be able to trigger again.

A maintenance time window can start immediately, or be automatically automatically scheduled according to your selection.

Enabling the maintenance mode

- Click on the enable maintenance mode button:



- Within the modal configuration window, enter the date and hours of the end of the maintenance time window:

The screenshot shows the 'Kafka Smart Monitoring - Management of embedded alerting framework' page. At the top, a red banner indicates 'ENABLED MAINTENANCE MODE STATUS' with a countdown timer 'Ends on: 17 Jun 07:00'. Below this, there are buttons for 'Enable maintenance mode' and 'Disable maintenance mode'. A navigation bar includes links like 'ALERT CONFIGURATION SUMMARY', 'STALE METRICS MONITORING PER COMPONENT', etc. A section titled 'Kafka infrastructure monitoring help - Summary configuration of embedded alerts' contains a table of alert configurations.

title	cron_schedule	schedule_window	alert.suppress.fields	alert.suppress.period	disabled	next_scheduled_time	
All Kafka components - active node numbers - stale metrics life test	*/* * * *	0	env, label, role	4h	1		
Kafka monitoring - Burrow - group consumers state monitoring	*/* * * *	0	env, label, cluster, group	4h	0	2019-06-15 23:10:00 BST	✓
Kafka monitoring - Confluent kafka-rest - stale metrics life test	*/* * * *	0	env, label, name	4h	0	2019-06-15 23:05:00 BST	✓
Kafka monitoring - Confluent ksqd-server - stale metrics life test	*/* * * *	0	env, label, name	4h	0	2019-06-15 23:05:00 BST	✓

- When the date and hours of the maintenance time window are reached, the scheduled report “Verify Kafka alerting maintenance status” will automatically disable the maintenance mode.
- If a start date time different than the current time is selected (default), this action will automatically schedule the maintenance time window.

Disabling the maintenance mode

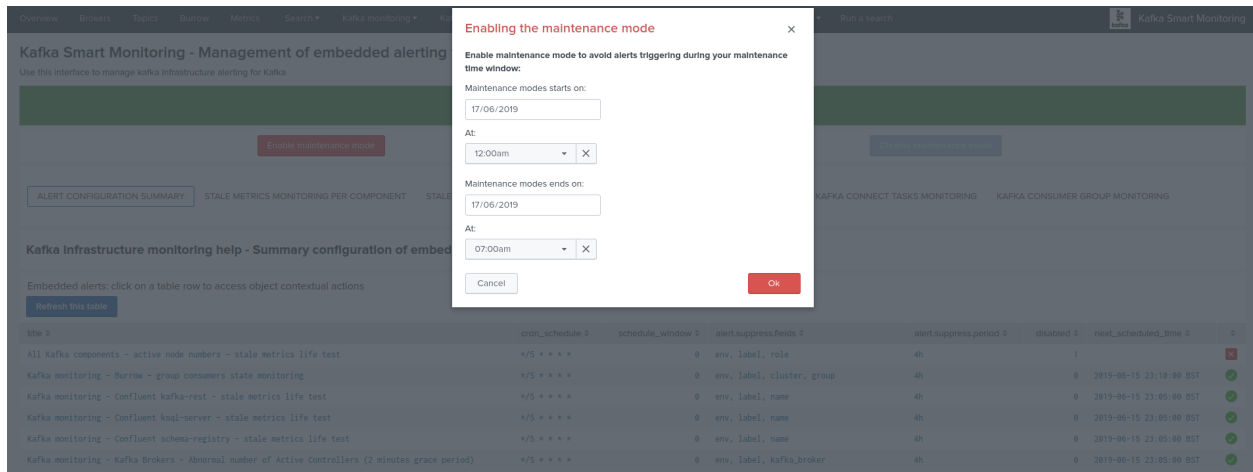
During any time of the maintenance time window, an administrator can decide to disable the maintenance mode:

The screenshot shows the same interface as before, but with a modal dialog box open. The dialog is titled 'Disabling the maintenance mode' and asks 'Do you want to confirm disabling the maintenance mode?'. It includes a warning: 'Once the maintenance mode will be disabled, all activated alerts will be able to trigger.' There are 'Cancel' and 'Ok' buttons. The background interface is dimmed.

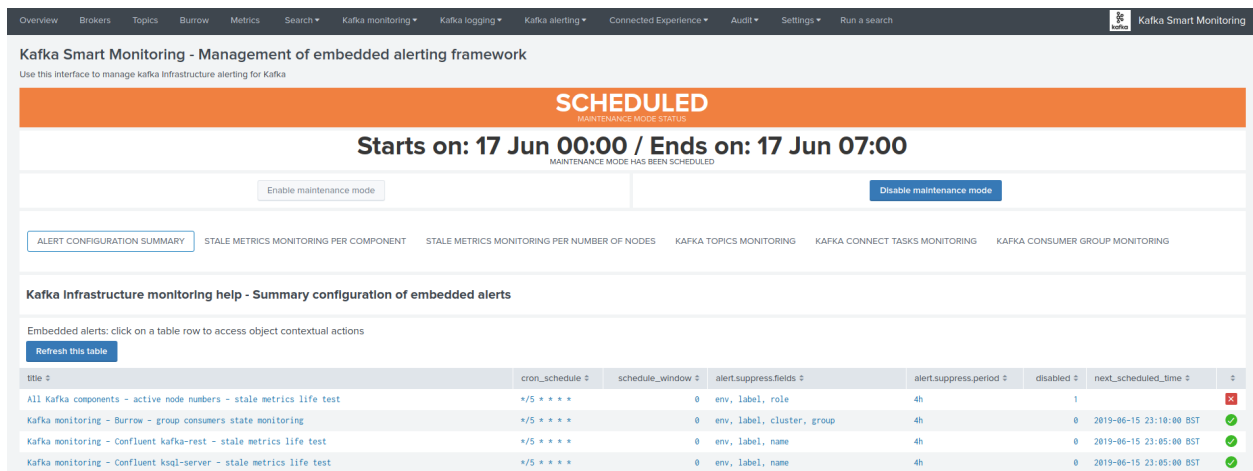
Scheduling a maintenance window

You can configure the maintenance mode to be automatically enabled between a specific date time that you enter in the UI.

- When the end time is reached, the maintenance mode will automatically be disabled, and the alerting will return to normal operations.



- When a maintenance mode window has been scheduled, the UI shows a specific message with the starts / ends on dates:



The collection KVstore endpoint can be programmatically managed, as such it is easily possible to reproduce this behaviour from an external system.

(<https://docs.splunk.com/Documentation/Splunk/latest/RESTREF/RESTkvstore>)

Monitoring state default definition

When new objects are automatically discovered such as Kafka components or topics, these objects are added to the different KVstore collection with a default enabled maintenance mode.

The default maintenance mode that is applied on a per type of object basis can be customised via the associated macros definitions:

Purpose	Macro definition
Type of component (nodes number monitoring)	zookeeper_default_monitoring_state
Zookeeper nodes	zookeeper_default_monitoring_state
Kafka Brokers	kafka_broker_default_monitoring_state
Kafka Topics	kafka_topics_default_monitoring_state
Kafka Connect workers	kafka_connect_default_monitoring_state
Kafka Connect connectors	kafka_connect_tasks_default_monitoring_state
Kafka Burrow group consumers	kafka_burrow_consumers_default_monitoring_state
Confluent Schema registry	schema_registry_default_monitoring_state
Confluent ksql-server	ksql_server_default_monitoring_state
Confluent kafka-rest	kafka_rest_default_monitoring_state
LinkedIn kafka-monitor	kafka_monitor_default_monitoring_state

The default macro definition does the following statement:

```
eval monitoring_state="enabled"
```

A typical customisation can be to disable by default the monitoring state for non Production environments:

```
eval monitoring_state=if(match(env, "(?i)PROD"), "enabled", "disabled")
```

Such that if a new object is discovered for a development environment, this will not be monitored unless a manual update is performed via the user configuration interface.

Administrating collection entries

Each type of component can be administrated in a dedicated tab within the user management interface.

When objects have been discovered, the administrator can eventually search for an object, and click on the object definition, which opens the modal interaction window:

Kafka infrastructure monitoring help - Stale metric monitoring per component

The KVsstore collection defines the state of alerting for a given entity, through the following items:

- monitoring_state**: only enabled entities will trigger when alerts conditions are met, such as a component being down or unreachable.
- grace_period**: a minimal grace period in seconds that will be applied before the alert triggers. (for metrics availability test only)

Use the update collection button to immediately run the components discovery report, or use the reset button to flush and reset the state of the collection.

Update the collection now

Reset the collection now

13

13

0

TOTAL COMPONENTS DISCOVERED

TOTAL COMPONENTS MONITORED

NOT MONITORED COMPONENTS

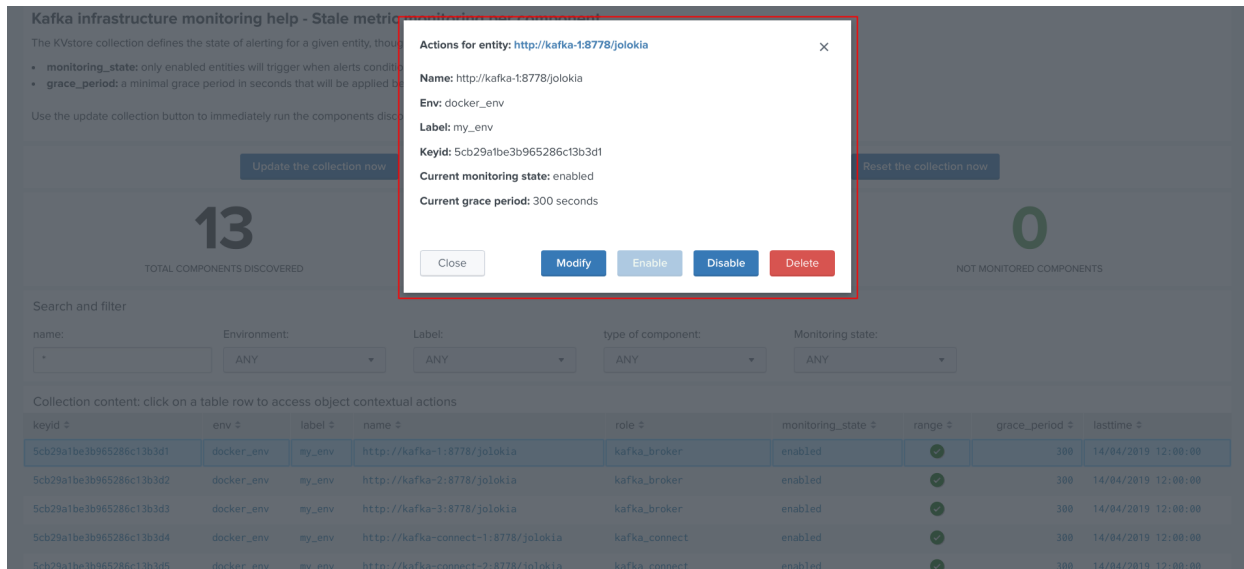
Search and filter

name: Environment: Label: type of component: Monitoring state:

Collection content: click on a table row to access object contextual actions

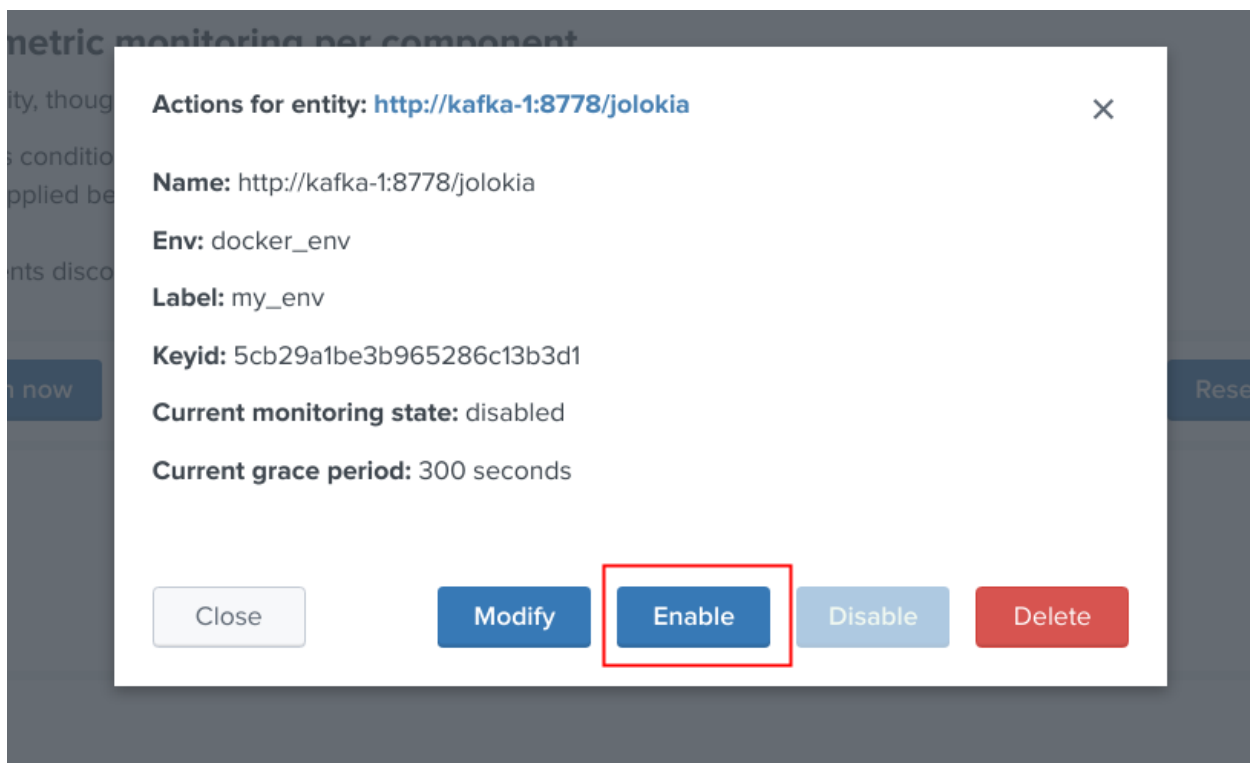
keyid ↕	env ↕	label ↕	name ↕	role ↕	monitoring_state ↕	range ↕	grace_period ↕	lasttime ↕
5cb29a1be3b965286c13b3d1	docker_env	my_env	http://kafka-1:8778/jolokia	kafka_broker	enabled	✓	300	14/04/2019 12:00:00
5cb29a1be3b965286c13b3d2	docker_env	my_env	http://kafka-2:8778/jolokia	kafka_broker	enabled	✓	300	14/04/2019 12:00:00
5cb29a1be3b965286c13b3d3	docker_env	my_env	http://kafka-3:8778/jolokia	kafka_broker	enabled	✓	300	14/04/2019 12:00:00
5cb29a1be3b965286c13b3d4	docker_env	my_env	http://kafka-connect-1:8778/jolokia	kafka_connect	enabled	✓	300	14/04/2019 12:00:00
5cb29a1be3b965286c13b3d5	docker_env	my_env	http://kafka-connect-2:8778/jolokia	kafka_connect	enabled	✓	300	14/04/2019 12:00:00

The modal interaction window provides information about this object, and different action buttons depending on this type of object:

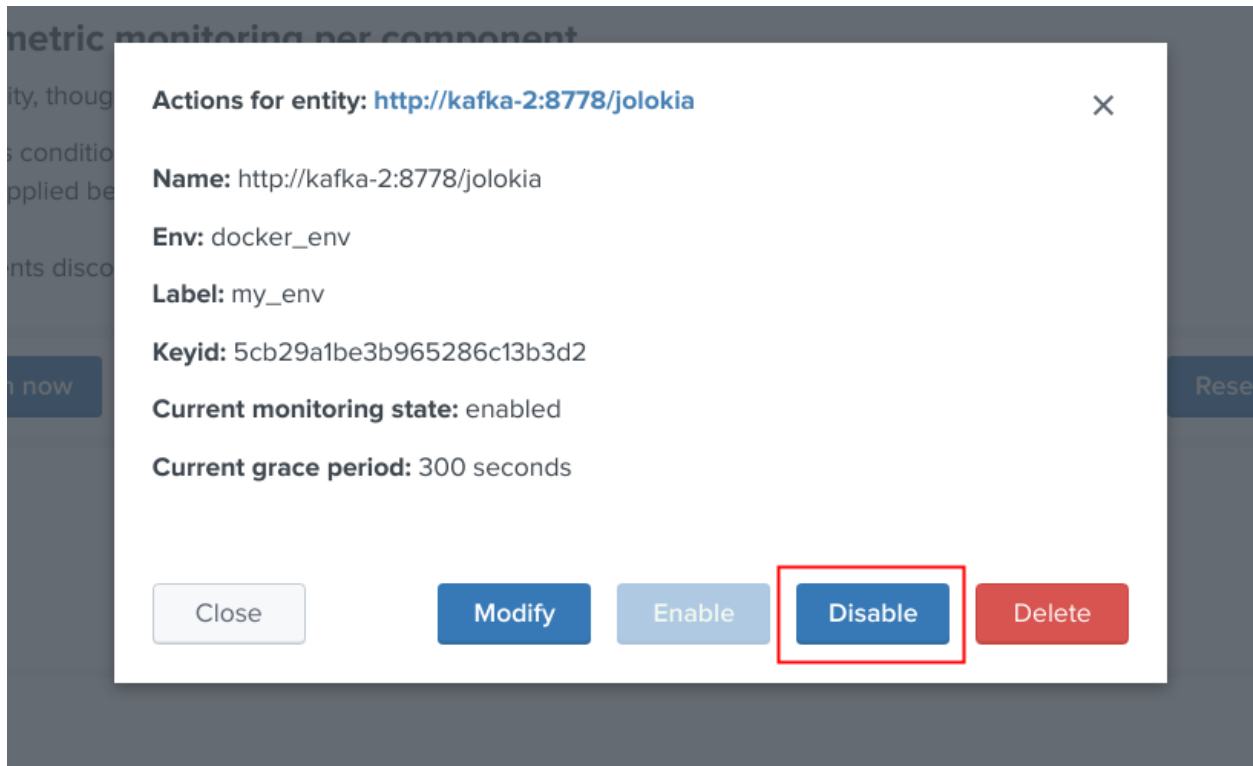


Enable/Disabling monitoring state

When an object has a disabled monitoring state, the button “enable monitoring” is automatically made available:



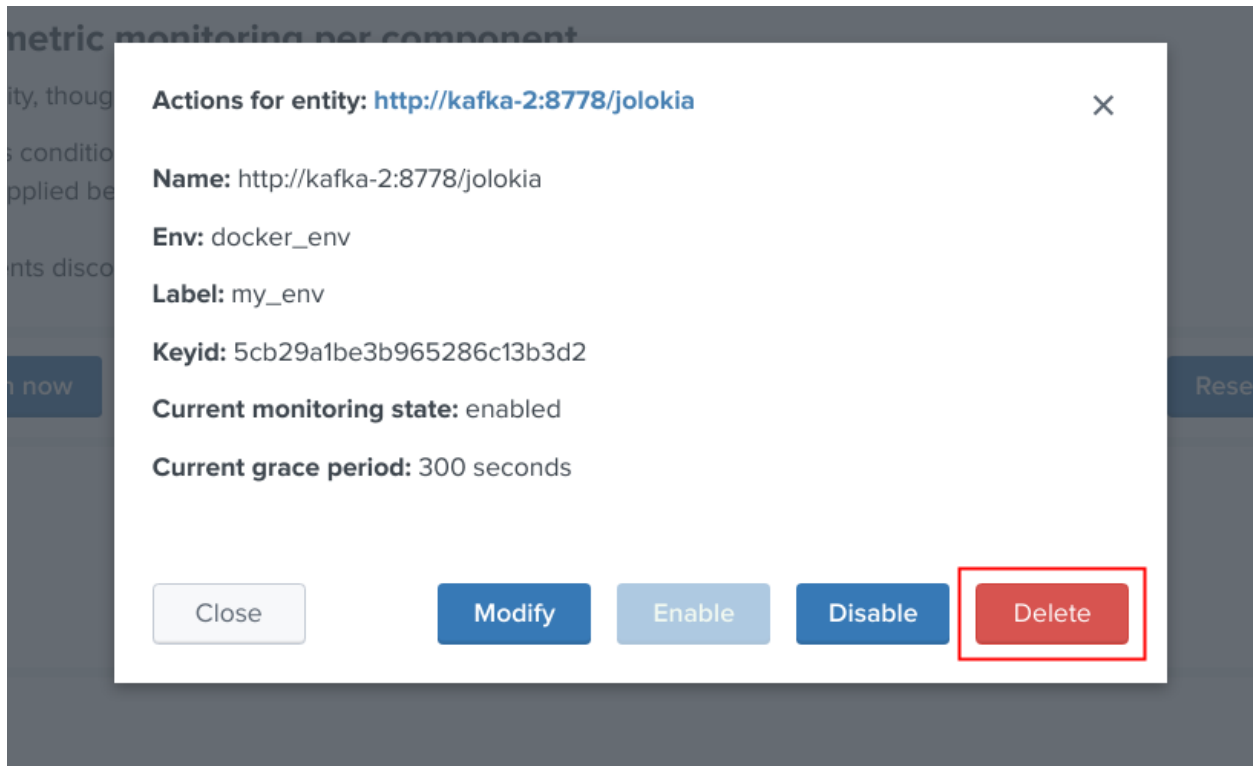
When an object has an enabled monitoring state, the button “disable monitoring” is automatically made available:



Shall the action be requested and confirmed, the object state will be updated, and the table exposing the object definition be refreshed.

Deleting objects in the collection

An object that was discovered and added to the collection automatically can be deleted via the UI:



Shall the action be requested and confirmed, the object state will be entirely removed from the collection, and the table exposing the object definition be refreshed.

Important:

By default, objects are discovered every 4 hours looking at metrics available for the last 4 hours.

This means that if the object has been still generated metrics to Splunk, it will be re-created automatically by the workflow.

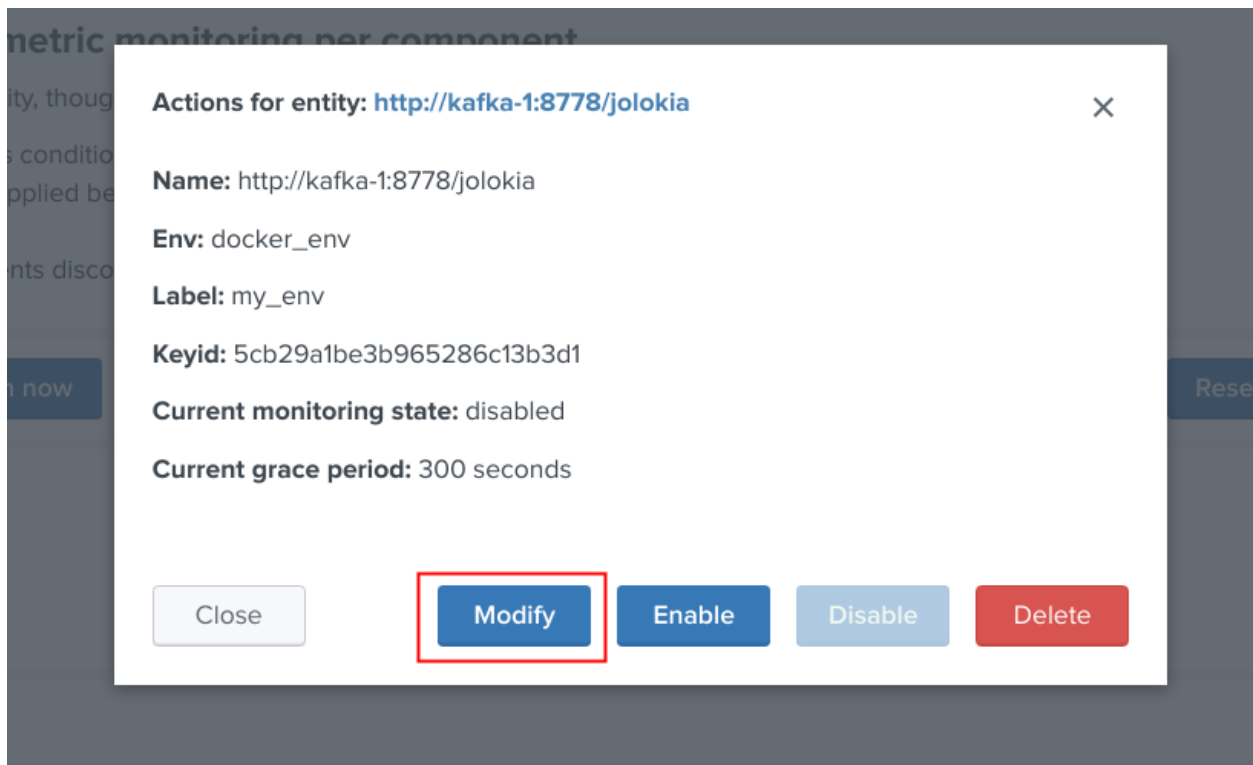
To avoid having to re-delete the same object again, you should wait 4 hours minimum before purging the object that was decommissioned.

Finally, note that if an object has not been generating metrics for a least 24 hours, its monitoring state will be disabled a special "disabled_autoforced" value.

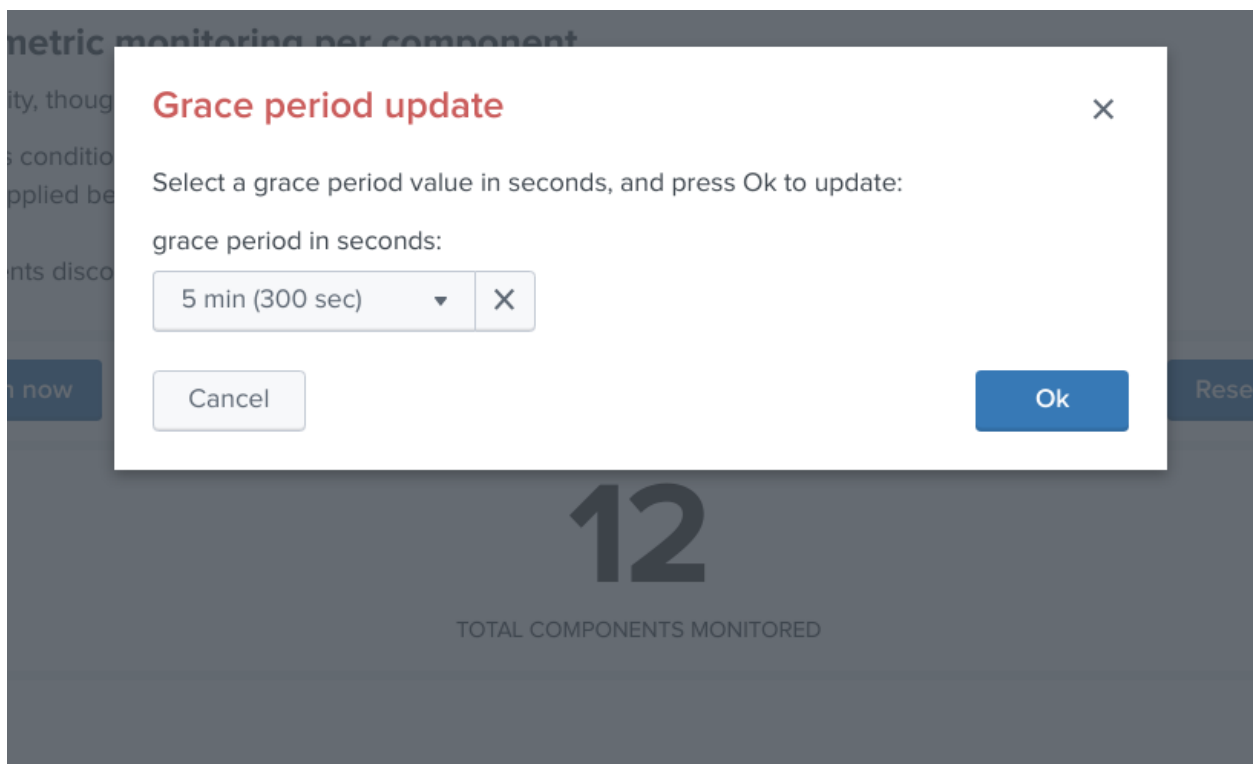
This state can still be manually updated via the UI, to permanently re-enable or disable the monitoring state if the component is still an active component.

Modifying an object in the collection

Depending on the type of object, the modal interaction window can provide a modification button:



The type of modification that can be applied depends on type of component, example:



Manually request a collection update job

A collection update can be requested at any time within the UI:

Kafka infrastructure monitoring help - Stale metric monitoring per component

The KVstore collection defines the state of alerting for a given entity, though the following items:

- monitoring_state**: only enabled entities will trigger when alerts conditions are met, such as a component being down or unreachable.
- grace_period**: a minimal grace period in seconds that will be applied before the alert triggers. (for metrics availability test only)

Use the update collection button to immediately run the components discovery report, or use the reset button to flush and reset the state of the collection.

Update the collection now

Reset the collection now

13

TOTAL COMPONENTS DISCOVERED

12

TOTAL COMPONENTS MONITORED

1

NOT MONITORED COMPONENTS

Search and filter

name:

Environment:

Label:

type of component:

Monitoring state:

*

ANY

ANY

ANY

ANY

Collection content: click on a table row to access object contextual actions

keyid ↕	env ↕	label ↕	name ↕	role ↕	monitoring_state ↕	range ↕	grace_period ↕	lasttime ↕
5cb29a1be3b965286c13b3d1	docker_env	my_env	http://kafka-1:8778/jolokia	kafka_broker	disabled	●	300	14/04/2019 12:00:00
5cb29a1be3b965286c13b3d2	docker_env	my_env	http://kafka-2:8778/jolokia	kafka_broker	enabled	●	300	14/04/2019 12:00:00
5cb29a1be3b965286c13b3d3	docker_env	my_env	http://kafka-3:8778/jolokia	kafka_broker	enabled	●	300	14/04/2019 12:00:00
5cb29a1be3b965286c13b3d4	docker_env	my_env	http://kafka-connect-1:8778/jolokia	kafka_connect	enabled	●	300	14/04/2019 12:00:00
5cb29a1be3b965286c13b3d5	docker_env	my_env	http://kafka-connect-2:8778/jolokia	kafka_connect	enabled	●	300	14/04/2019 12:00:00

Shall the action be requested and confirmed, the UI will automatically run the object discovery report, any new object that was not yet discovered since the last run of the report, will be added to the collection and made available within the UI.

Kafka infrastructure monitoring help - Stale metric monitoring per component

The KVstore collection defines the state of alerting for a given entity, though the following items:

- monitoring_state**: only enabled entities will trigger when alerts conditions are met, such as a component being down or unreachable.
- grace_period**: a minimal grace period in seconds that will be applied before the alert triggers. (for metrics availability test only)

Use the update collection button to immediately run the components discovery report, or use the reset button to flush and reset the state of the collection.

Update the collection now

Reset the collection now

13

TOTAL COMPONENTS DISCOVERED

Updating the KVstore collection...
12

TOTAL COMPONENTS MONITORED

1

NOT MONITORED COMPONENTS

Search and filter

name:

Environment:

Label:

type of component:

Monitoring state:

*

ANY

ANY

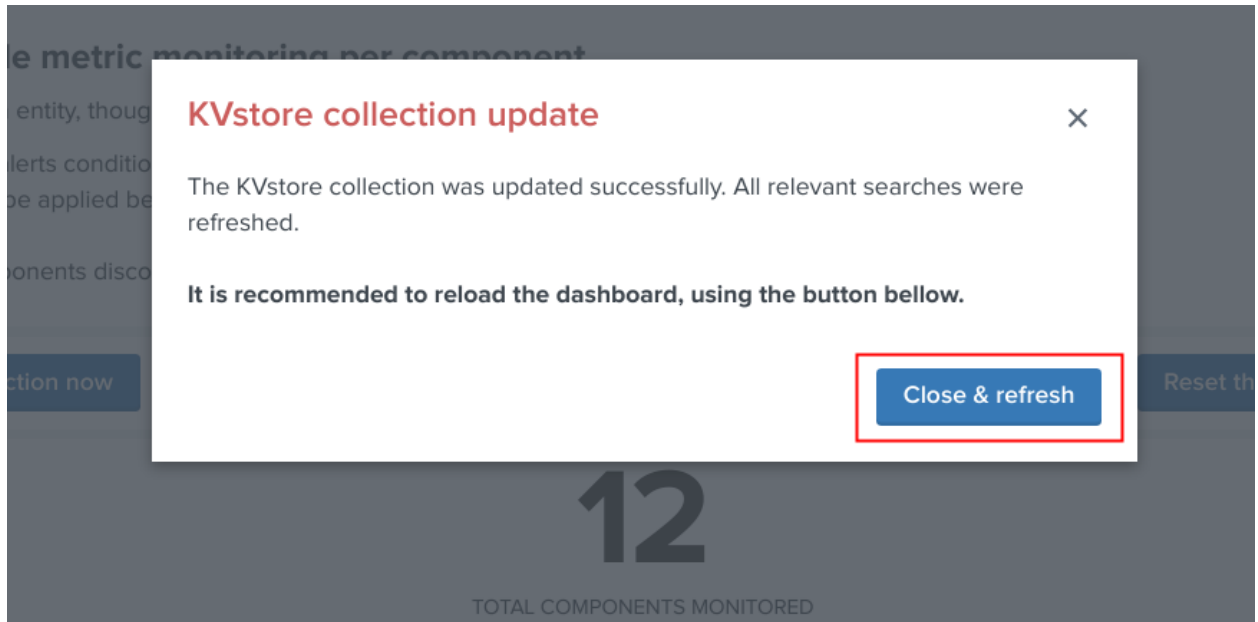
ANY

ANY

Collection content: click on a table row to access object contextual actions

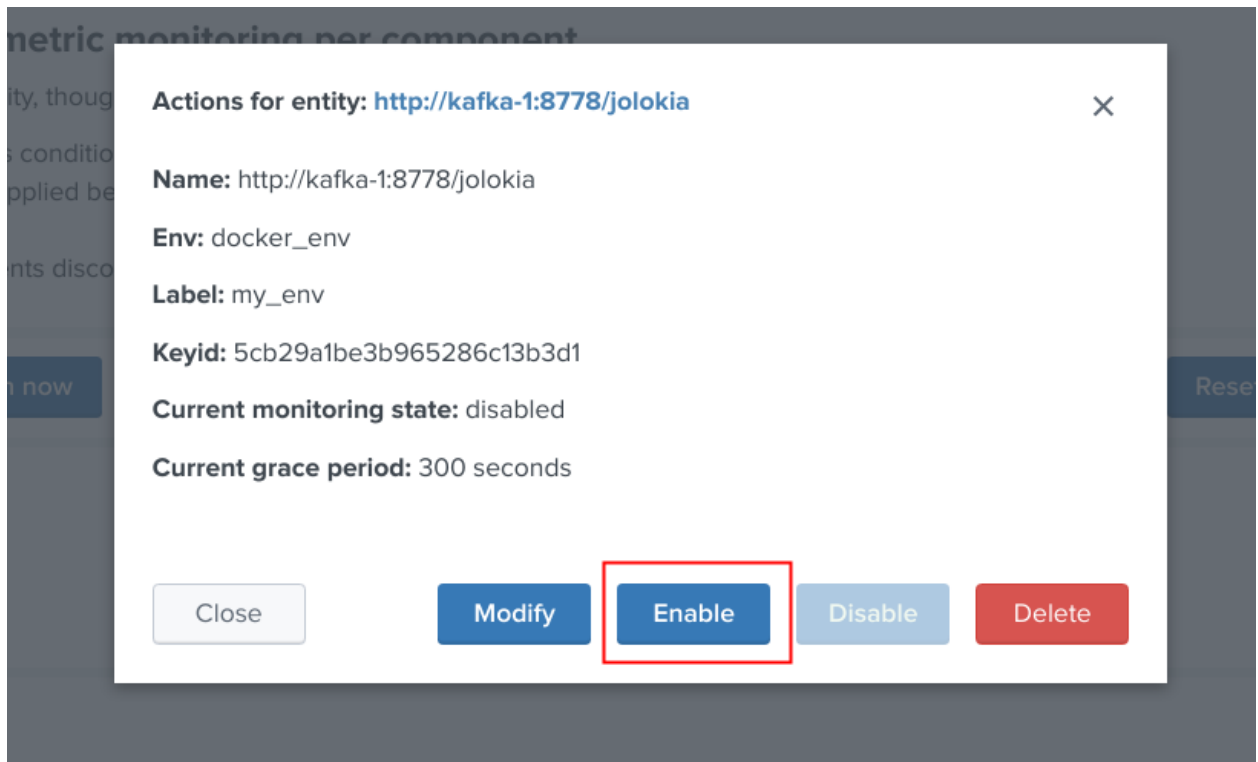
keyid ↕	env ↕	label ↕	name ↕	role ↕	monitoring_state ↕	range ↕	grace_period ↕	lasttime ↕
5cb29a1be3b965286c13b3d1	docker_env	my_env	http://kafka-1:8778/jolokia	kafka_broker	disabled	●	300	14/04/2019 12:00:00
5cb29a1be3b965286c13b3d2	docker_env	my_env	http://kafka-2:8778/jolokia	kafka_broker	enabled	●	300	14/04/2019 12:00:00
5cb29a1be3b965286c13b3d3	docker_env	my_env	http://kafka-3:8778/jolokia	kafka_broker	enabled	●	300	14/04/2019 12:00:00
5cb29a1be3b965286c13b3d4	docker_env	my_env	http://kafka-connect-1:8778/jolokia	kafka_connect	enabled	●	300	14/04/2019 12:00:00
5cb29a1be3b965286c13b3d5	docker_env	my_env	http://kafka-connect-2:8778/jolokia	kafka_connect	enabled	●	300	14/04/2019 12:00:00

Once the job has run, click on the refresh button:

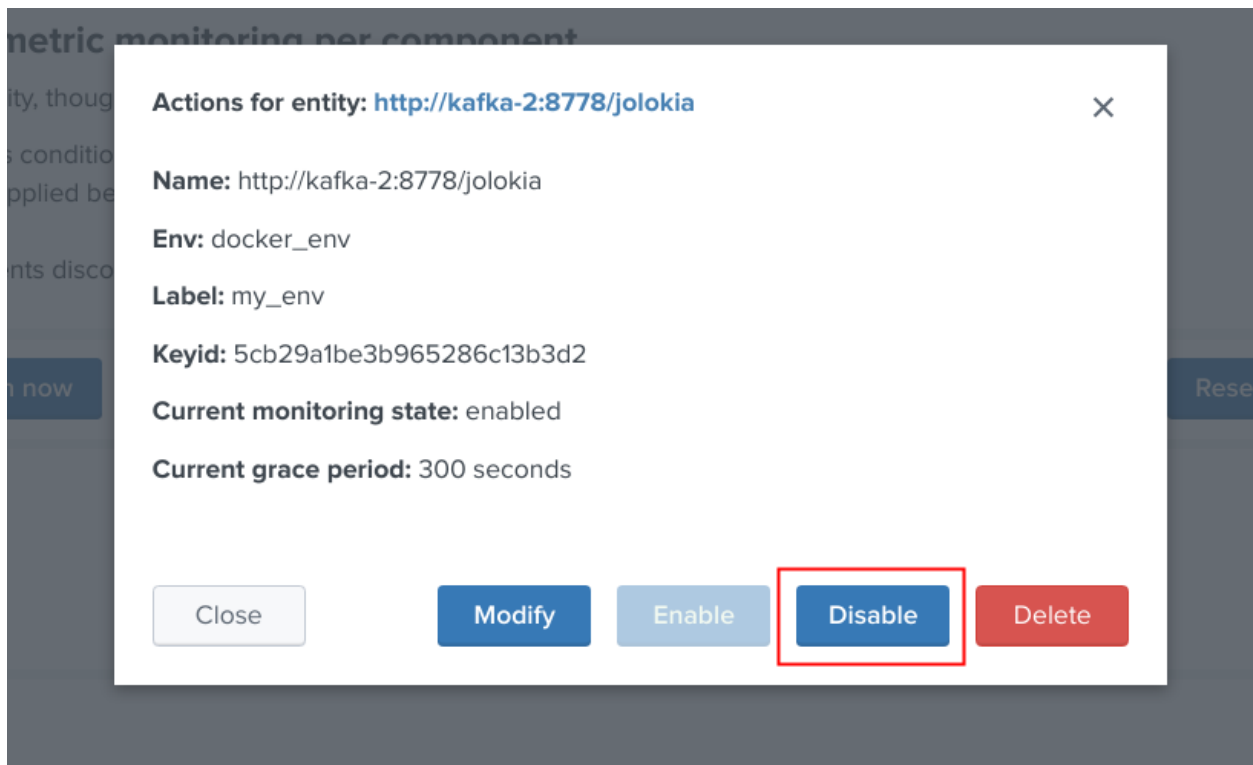


Enable/Disabling monitoring state

When an object has a disabled monitoring state, the button “enable monitoring” is automatically made available:



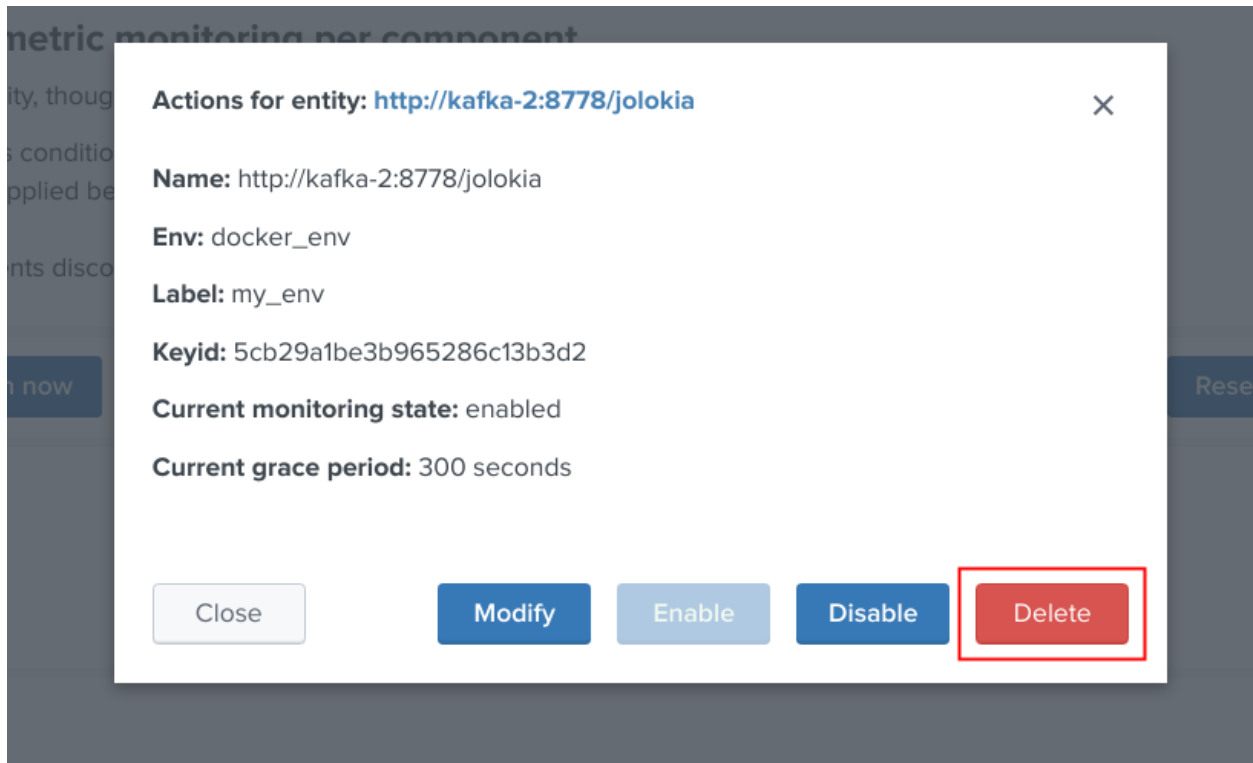
When an object has an enabled monitoring state, the button “disable monitoring” is automatically made available:



Shall the action be requested and confirmed, the object state will be updated, and the table exposing the object definition be refreshed.

Deleting objects in the collection

An object that was discovered and added to the collection automatically can be deleted via the UI:



Shall the action be requested and confirmed, the object state will be entirely removed from the collection, and the table exposing the object definition be refreshed.

Important:

By default, objects are discovered every 4 hours looking at metrics available for the last 4 hours.

This means that if the object has been still generated metrics to Splunk, it will be re-created automatically by the workflow.

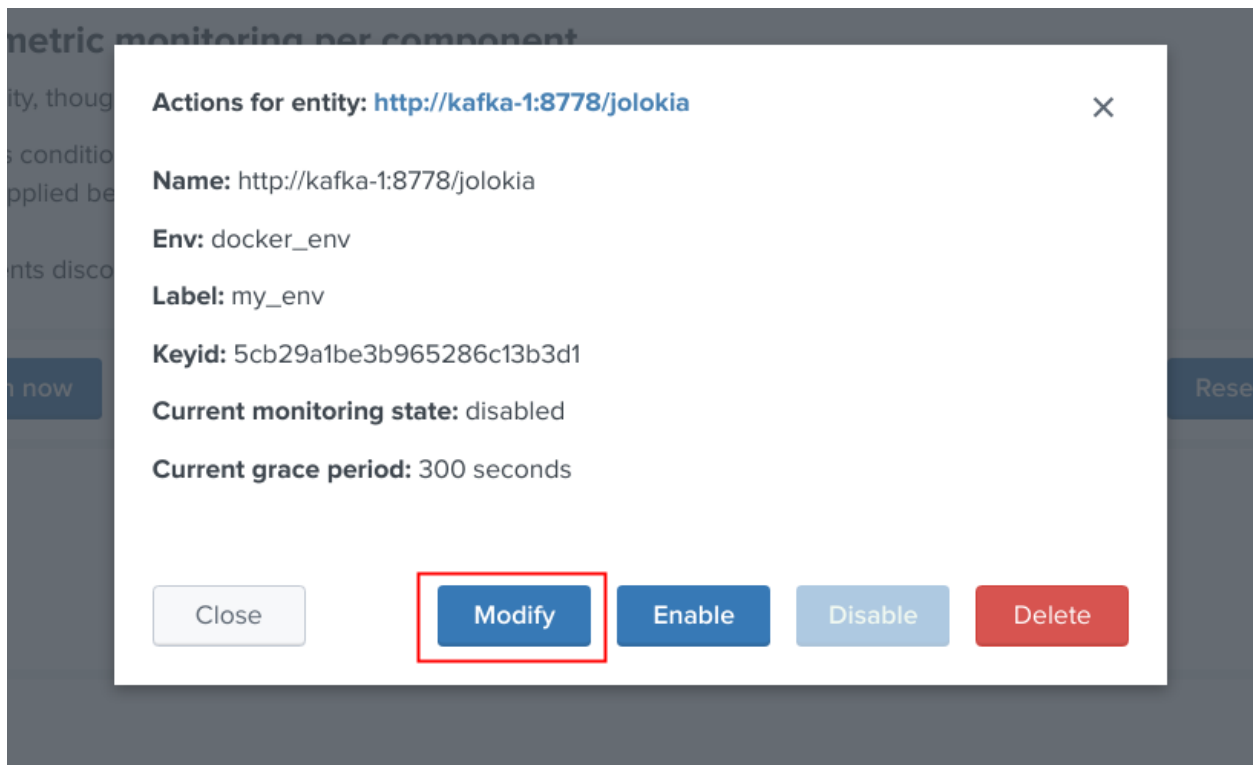
To avoid having to re-delete the same object again, you should wait 4 hours minimum before purging the object that was decommissioned.

Finally, note that if an object has not been generating metrics for a least 24 hours, its monitoring state will be disabled a special “disabled_autoforced” value.

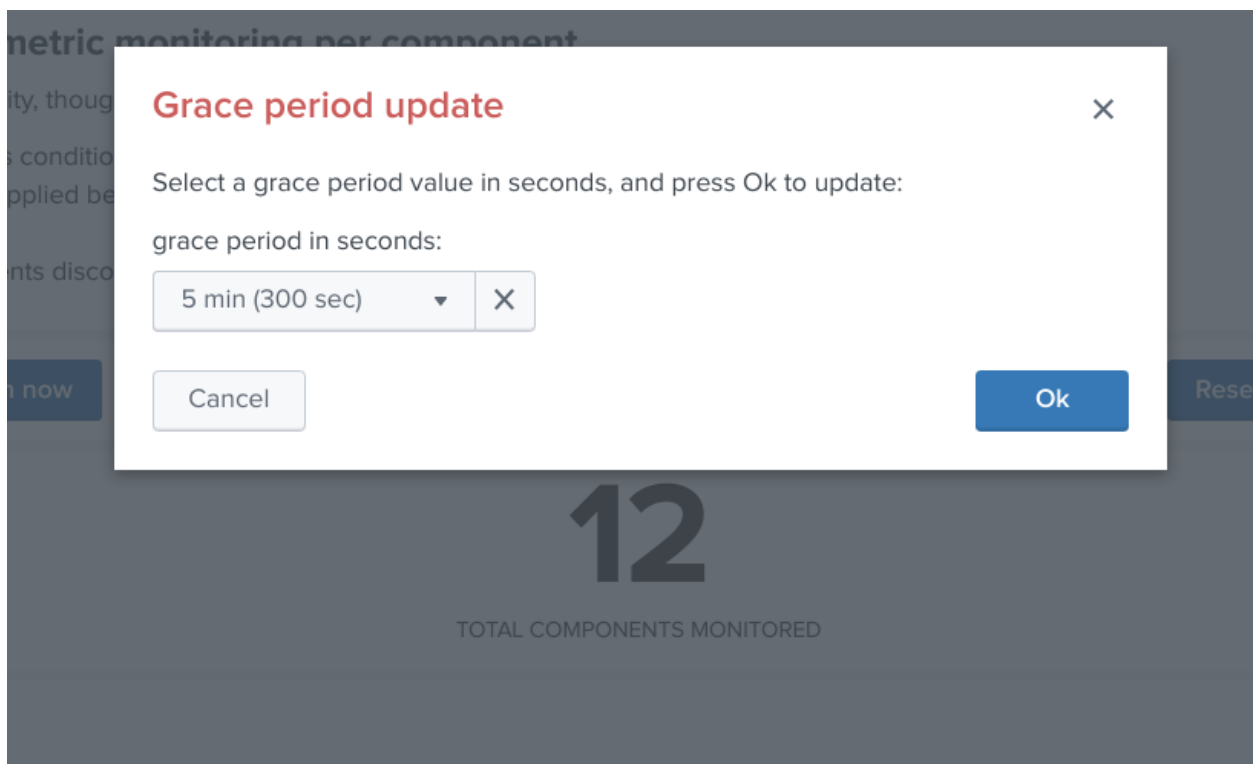
This state can still be manually updated via the UI, to permanently re-enable or disable the monitoring state if the component is still an active component.

Modifying an object in the collection

Depending on the type of object, the modal interaction window can provide a modification button:



The type of modification that can be applied depends on type of component, example:



Manually request a collection update job

A collection update can be requested at any time within the UI:

Kafka infrastructure monitoring help - Stale metric monitoring per component

The KVStore collection defines the state of alerting for a given entity, though the following items:

- monitoring_state**: only enabled entities will trigger when alerts conditions are met, such as a component being down or unreachable.
- grace_period**: a minimal grace period in seconds that will be applied before the alert triggers. (for metrics availability test only)

Use the update collection button to immediately run the components discovery report, or use the reset button to flush and reset the state of the collection.

Update the collection now Reset the collection now

13 TOTAL COMPONENTS DISCOVERED **12** TOTAL COMPONENTS MONITORED **1** NOT MONITORED COMPONENTS

Search and filter

name: Environment: Label: type of component: Monitoring state:

Collection content: click on a table row to access object contextual actions

keyid ↕	env ↕	label ↕	name ↕	role ↕	monitoring_state ↕	range ↕	grace_period ↕	lasttime ↕
5cb29a1be3b965286c13b3d1	docker_env	my_env	http://kafka-1:8778/jolokia	kafka_broker	disabled	●	300	14/04/2019 12:00:00
5cb29a1be3b965286c13b3d2	docker_env	my_env	http://kafka-2:8778/jolokia	kafka_broker	enabled	●	300	14/04/2019 12:00:00
5cb29a1be3b965286c13b3d3	docker_env	my_env	http://kafka-3:8778/jolokia	kafka_broker	enabled	●	300	14/04/2019 12:00:00
5cb29a1be3b965286c13b3d4	docker_env	my_env	http://kafka-connect-1:8778/jolokia	kafka_connect	enabled	●	300	14/04/2019 12:00:00
5cb29a1be3b965286c13b3d5	docker_env	my_env	http://kafka-connect-2:8778/jolokia	kafka_connect	enabled	●	300	14/04/2019 12:00:00

Shall the action be requested and confirmed, the UI will automatically run the object discovery report, any new object that was not yet discovered since the last run of the report, will be added to the collection and made available within the UI.

Kafka infrastructure monitoring help - Stale metric monitoring per component

The KVStore collection defines the state of alerting for a given entity, though the following items:

- monitoring_state**: only enabled entities will trigger when alerts conditions are met, such as a component being down or unreachable.
- grace_period**: a minimal grace period in seconds that will be applied before the alert triggers. (for metrics availability test only)

Use the update collection button to immediately run the components discovery report, or use the reset button to flush and reset the state of the collection.

Update the collection now Reset the collection now

13 TOTAL COMPONENTS DISCOVERED **Updating the KVStore collection...** **1** NOT MONITORED COMPONENTS

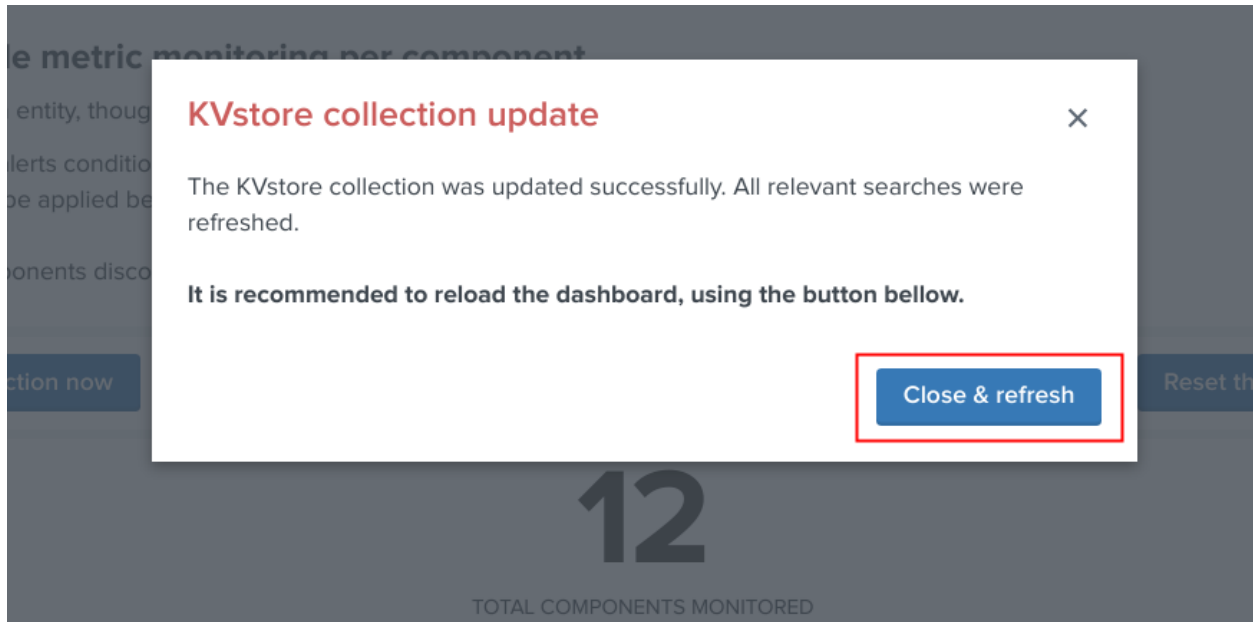
Search and filter

name: Environment: Label: type of component: Monitoring state:

Collection content: click on a table row to access object contextual actions

keyid ↕	env ↕	label ↕	name ↕	role ↕	monitoring_state ↕	range ↕	grace_period ↕	lasttime ↕
5cb29a1be3b965286c13b3d1	docker_env	my_env	http://kafka-1:8778/jolokia	kafka_broker	disabled	●	300	14/04/2019 12:00:00
5cb29a1be3b965286c13b3d2	docker_env	my_env	http://kafka-2:8778/jolokia	kafka_broker	enabled	●	300	14/04/2019 12:00:00
5cb29a1be3b965286c13b3d3	docker_env	my_env	http://kafka-3:8778/jolokia	kafka_broker	enabled	●	300	14/04/2019 12:00:00
5cb29a1be3b965286c13b3d4	docker_env	my_env	http://kafka-connect-1:8778/jolokia	kafka_connect	enabled	●	300	14/04/2019 12:00:00
5cb29a1be3b965286c13b3d5	docker_env	my_env	http://kafka-connect-2:8778/jolokia	kafka_connect	enabled	●	300	14/04/2019 12:00:00

Once the job has run, click on the refresh button:



Shall the job fail for some reasons such as a lack of permissions, an error window with the Splunk error message would be exposed automatically.

Manually request a collection rebuild job

A collection reset can be requested at any time within the UI:

Kafka infrastructure monitoring help - State metric monitoring per component

The KVstore collection defines the state of alerting for a given entity, though the following items:

- monitoring_state:** only enabled entities will trigger when alerts conditions are met, such as a component being down or unreachable.
- grace_period:** a minimal grace period in seconds that will be applied before the alert triggers. (for metrics availability test only)

Use the update collection button to immediately run the components discovery report, or use the reset button to flush and reset the state of the collection.

Update the collection now Reset the collection now

13 TOTAL COMPONENTS DISCOVERED **12** TOTAL COMPONENTS MONITORED **1** NOT MONITORED COMPONENTS

Search and filter

name: Environment: Label: type of component: Monitoring state:

Collection content: click on a table row to access object contextual actions

keyid ↕	env ↕	label ↕	name ↕	role ↕	monitoring_state ↕	range ↕	grace_period ↕	lasttime ↕
5cb29a1be3b965286c13b3d1	docker_env	my_env	http://kafka-1:8778/jolokia	kafka_broker	disabled	●	300	14/04/2019 12:00:00
5cb29a1be3b965286c13b3d2	docker_env	my_env	http://kafka-2:8778/jolokia	kafka_broker	enabled	●	300	14/04/2019 12:00:00
5cb29a1be3b965286c13b3d3	docker_env	my_env	http://kafka-3:8778/jolokia	kafka_broker	enabled	●	300	14/04/2019 12:00:00
5cb29a1be3b965286c13b3d4	docker_env	my_env	http://kafka-connect-1:8778/jolokia	kafka_connect	enabled	●	300	14/04/2019 12:00:00
5cb29a1be3b965286c13b3d5	docker_env	my_env	http://kafka-connect-2:8778/jolokia	kafka_connect	enabled	●	300	14/04/2019 12:00:00

Important: When requesting a reset of the collection, all changes will be irremediably lost. All matching objects will be reset to their default discovered values.

Shall the action be requested and confirmed, the UI will automatically run the object discovery report, any new object that was not yet discovered since the last run of the report, will be added to the collection and made available within the UI.

Kafka infrastructure monitoring help - Stale metric monitoring per component

The KVstore collection defines the state of alerting for a given entity, though the following items:

- monitoring_state**: only enabled entities will trigger when alerts conditions are met, such as a component being down or unreachable.
- grace_period**: a minimal grace period in seconds that will be applied before the alert triggers. (for metrics availability test only)

Use the update collection button to immediately run the components discovery report, or use the reset button to flush and reset the state of the collection.

Update the collection now Reset the collection now

Flushing and updating the KVstore collection...

13 TOTAL COMPONENTS DISCOVERED 12 TOTAL COMPONENTS MONITORED 1 NOT MONITORED COMPONENTS

Search and filter

name: Environment: Label: type of component: Monitoring state:

Collection content: click on a table row to access object contextual actions

keyid	env	label	name	role	monitoring_state	range	grace_period	lasttime
5cb29a1be3b965286c13b3d1	docker_env	my_env	http://kafka-1:8778/jolokia	kafka_broker	disabled	●	300	14/04/2019 14:06:13
5cb29a1be3b965286c13b3d2	docker_env	my_env	http://kafka-2:8778/jolokia	kafka_broker	enabled	●	300	14/04/2019 14:06:13
5cb29a1be3b965286c13b3d3	docker_env	my_env	http://kafka-3:8778/jolokia	kafka_broker	enabled	●	300	14/04/2019 14:06:13
5cb29a1be3b965286c13b3d4	docker_env	my_env	http://kafka-connect-1:8778/jolokia	kafka_connect	enabled	●	300	14/04/2019 14:06:13
5cb29a1be3b965286c13b3d5	docker_env	my_env	http://kafka-connect-2:8778/jolokia	kafka_connect	enabled	●	300	14/04/2019 14:06:13

Once the job has run, click on the refresh button:

12 TOTAL COMPONENTS MONITORED

KVstore collection update

The KVstore collection was updated successfully. All relevant searches were refreshed.

It is recommended to reload the dashboard, using the button below.

Close & refresh

Shall the job fail for some reasons such as a lack of permissions, an error window with the Splunk error message would be exposed automatically.

2.6.2 Enabling OOTB alerts

Important: By default, all alerts are disabled, you must enable the alerts within Splunk Web depending on your needs.

You need to decide which alert must be enabled depending on your needs and environments, and achieve any additional alert actions that would be required such as creating an incident in a ticketing system.

Splunk alerts can easily be extended by alert actions.

Alert configuration summary user interface

The summary alert tab exposes most valuable information about the alerts, and provides a shortcut access to the management of the alerts:

ALERT CONFIGURATION SUMMARY	STATE METRICS MONITORING PER COMPONENT	STATE METRICS MONITORING PER NUMBER OF NODES	KAFKA TOPICS MONITORING	KAFKA CONNECT TASKS MONITORING	KAFKA CONSUMER GROUP MONITORING
-----------------------------	--	--	-------------------------	--------------------------------	---------------------------------

Kafka infrastructure monitoring help - Summary configuration of embedded alerts

Embedded alerts: click on a table row to access object contextual actions

[Refresh this table](#)

title ↕	cron_schedule ↕	schedule_window ↕	alertsuppress.fields ↕	alertsuppress.period ↕	disabled	next_scheduled_time ↕	range ↕
All Kafka components - active node numbers - state metrics life test	+/5 * * * *	0	env, label, role	4h	0	2019-04-14 14:15:00 UTC	🟢
Kafka monitoring - Burrow - group consumers state monitoring	+/5 * * * *	0	env, label, cluster, group	4h	0	2019-04-14 14:20:00 UTC	🟢
Kafka monitoring - Confluent kafka-rest - state metrics life test	+/5 * * * *	0	env, label, name	4h	0	2019-04-14 14:20:00 UTC	🟢
Kafka monitoring - Confluent ksql-server - state metrics life test	+/5 * * * *	0	env, label, name	4h	0	2019-04-14 14:20:00 UTC	🟢
Kafka monitoring - Confluent schema-registry - state metrics life test	+/5 * * * *	0	env, label, name	4h	0	2019-04-14 14:20:00 UTC	🟢
Kafka monitoring - Kafka Brokers - Abnormal number of Active Controllers (2 minutes grace period)	+/5 * * * *	0	env, label, kafka_broker	4h	0	2019-04-14 14:20:00 UTC	🟢
Kafka monitoring - Kafka Brokers - Failed producer or consumer was detected	+/5 * * * *	0	env, label, kafka_broker, metric_name	4h	0	2019-04-14 14:20:00 UTC	🟢
Kafka monitoring - Kafka Brokers - ISR Shrinking detection	+/5 * * * *	0	env, label, kafka_broker	4h	0	2019-04-14 14:20:00 UTC	🟢
Kafka monitoring - Kafka Brokers - Offline or Under-replicated partitions	+/5 * * * *	0	env, label, kafka_broker, metric_name	4h	0	2019-04-14 14:20:00 UTC	🟢
Kafka monitoring - Kafka Brokers - state metrics life test	+/5 * * * *	0	env, label, name	4h	0	2019-04-14 14:20:00 UTC	🟢
Kafka monitoring - Kafka Connect - connector or task startup failure detected	+/5 * * * *	0	env, label, connector	4h	0	2019-04-14 14:20:00 UTC	🟢
Kafka monitoring - Kafka Connect - state metrics life test	+/5 * * * *	0	env, label, name	4h	0	2019-04-14 14:20:00 UTC	🟢
Kafka monitoring - Kafka Connect - tasks status monitoring	+/5 * * * *	0	env, label, connector	4h	0	2019-04-14 14:20:00 UTC	🟢
Kafka monitoring - Kafka Topics - Under-replicated partitions detected on topic	+/5 * * * *	0	env, label, topic	4h	0	2019-04-14 14:20:00 UTC	🟢
Kafka monitoring - Kafka Topics - errors detected on a topic	+/5 * * * *	0	env, label, topic	4h	0	2019-04-14 14:20:00 UTC	🟢
Kafka monitoring - LinkedIn Kafka Monitor - state metrics life test	+/5 * * * *	0	env, label, name	4h	0	2019-04-14 14:20:00 UTC	🟢
Kafka monitoring - Zookeeper - state metrics life test	+/5 * * * *	0	env, label, name	4h	0	2019-04-14 14:20:00 UTC	🟢

Click on any alert to open the modal interaction window:

×

Actions for alert:
All Kafka components - active node numbers - stale metrics life test

Cron Schedule: */5 * * * *

Schedule window: 0

Suppress fields: env, label, role

Suppress period: 4h

Disabled: 0

Next scheduled time: 2019-04-14 14:15:00 UTC

Close

Review and edit alert

Search alert history

Click on the “Review and edit alert” button to open the Splunk alert configuration UI for this alert:

Enterprise

App: Kafka Smart Monitoring

Adminstrator

Messages

Settings

Activity

Help

Find

Overview

Brokers

Topics

Burrow

Metrics

Search

Kafka monitoring

Kafka logging

Kafka alerting

Settings

Run a search

Kafka Smart Monitoring

New Search

index=audit action=alert_fired ss_app=telegraf-kafka ss_name=All Kafka components - active node numbers - stale metrics life test

Last 24 hours

Q

1 event (13/04/2019 17:00:00.000 to 14/04/2019 17:04:36.000)

No Event Sampling

Job

||

Smart Mode

Events (1)

Patterns

Statistics

Visualization

Format Timeline

Zoom Out

Zoom to Selection

Deselect

1 hour per column

List

Format

20 Per Page

< Hide Fields

≡ All Fields

SELECTED FIELDS

host 1

source 1

source_type 1

INTERESTING FIELDS

action 1

alert_actions 1

date_hour 1

date_mday 1

date_minute 1

date_month 1

date_second 1

date_wday 1

date_year 1

date_zone 1

dest 1

digest_mode 1

expiration 1

index 1

linecount 1

sourcecount 1

Time

Event

14/04/2019 14:46:38.119

Audit:[timestamp=14-14-2019 14:46:38.119, user=admin, action=alert_fired, ss_user=nobody, ss_app=telegraf-kafka, ss_name=All Kafka components - active node numbers - stale metrics life test, sid=scheduled_admin_audit2kdyWYta7kaZc__98D59945d93574a85eb7_at_155523180_95189, alert_actions="", severity=4, trigger_time=155523198, expiration=155533998, digest_mode=0, triggered_alerts=1][n/a]

host=ip-10-0-0-173 | source=audittrail | source_type=audittrail

- `grace_period`: The grace value in seconds before assuming a severe status (difference in seconds between the last communication and time of the check)
- `monitoring_state`: A value of “enabled” activates verification, any other value disables it

Stale metrics life test by number of nodes per type of component

If you are running the Kafka components in a container based architecture, you can monitor your infrastructure availability by monitoring the number of active nodes per type of component.

As such, you will be monitoring how many nodes are active at a time, rather than specific nodes identities which will change with the life cycle of the containers.

- All Kafka components - active node numbers - stale metrics life test

Shall an upgrade of a statefulSet or deployment in Kubernetes fail and new containers fail to start, the OOTB alerting will report this bad condition on per type of component basis.

Kafka brokers monitoring

The following alerts are available to monitor the main and most important aspects of Kafka Broker clusters:

- Abnormal number of Active Controllers
- Offline or Under-replicated partitions
- Failed producer or consumer was detected
- ISR Shrinking detection

Kafka topics monitoring

The following alerts are available to monitor Kafka topics:

- Under-replicated partitions detected on topics
- Errors reported on topics (bytes rejected, failed fetch requests, failed produce requests)

Kafka Connect task monitoring

Alerts are available to monitor the state of connectors and tasks for Kafka Connect:

- Kafka monitoring - Kafka Connect - tasks status monitoring

Alerts can be controlled by changing values of the fields:

- `grace_period`: The grace value in seconds before assuming a severe status (difference in seconds between the last communication and time of the check)
- `monitoring_state`: A value of “enabled” activates verification, any other value disables it

Kafka Consumers monitoring with Burrow

Alerts are available to monitor and report the state of Kafka Consumers via Burrow:

- Kafka monitoring - Burrow - group consumers state monitoring

Alerts can be controlled by changing values of the fields:

- `monitoring_state`: A value of “enabled” activates verification, any other value disables it

Notes: Kafka Connect source and sink connectors depending on their type are as well consumers, Burrow will monitor the way the connectors behave by analysing their lagging metrics and type of activity, this is a different, complimentary and advanced type of monitoring than analysing the state of the tasks.

2.6.3 Programmatic access and interactions with external systems

Requirements and recommendations

- Create a Splunk service account user that is member of the builtin **kafka_admin** role
- The builtin **kafka_admin** role provides read and write permission to the different KVstore collections
- Make sure splunkd REST API is reachable from your external tool

References

- <http://dev.splunk.com/view/webframework-developapps/SP-CAAAEZG>
- <https://docs.splunk.com/Documentation/Splunk/latest/RESTREF/RESTprolog>
- <https://docs.splunk.com/Documentation/Splunk/latest/RESTTUT/RESTandCloud>
- <https://www.urlencoder.org/> (example online tool to URLEncode / decode)

For convenience of the documentation bellow

```
export splunk_url="localhost:8089"
```

Authentication

For Splunk prior to 7.3.x

The recommended approach is authentication to Splunk API via a token, see:

Official documentation: [Splunk docs API token](#).

Example authenticating with a user called svc_kafka that is member of the kafka_admin role:

```
curl -k https://$splunk_url/services/auth/login --data-urlencode username=svc_kafka --
↳data-urlencode password=pass

<response>
  <sessionKey>DWGNbGpJgSj30w0GxTAXmj8t0dZKjvjxLYaP^yphdluFN_FGz4gz^
  ↳NhcgPCLDkjWH3BUQa1Vewt8FTF8KXyyfI09HqjOicIthMuBIB70dVJA8Jg</sessionKey>
  <messages>
    <msg code=""></msg>
  </messages>
</response>

export token="DWGNbGpJgSj30w0GxTAXmj8t0dZKjvjxLYaP^yphdluFN_FGz4gz^
↳NhcgPCLDkjWH3BUQa1Vewt8FTF8KXyyfI09HqjOicIthMuBIB70dVJA8Jg"
```

A token remains valid for the time of a session, which is by default valid for 1 hour.

For Splunk 7.3.0 and later

Splunk 7.3.0 introduced the usage of proper authentication tokens, which is the recommended way to authenticate against splunkd API:

Official documentation: [Splunk docs JSON authentication token](#).

Once you have created an authentication token for the user to be used as the service account, using curl specify the bearer token:

```
curl -k -H "Authorization: Bearer <token>"
```

Maintenance mode management

Get the current maintenance mode status

For Splunk 7.3.0 and later, replace with `-H "Authorization: Bearer <token>"`

```
curl -k -H "Authorization: Splunk $token" \
  https://$splunk_url/servicesNS/nobody/telegraf-kafka/storage/collections/data/kv_
  ↳telegraf_kafka_alerting_maintenance
```

Enabling the maintenance mode

Enabling the maintenance mode requires:

- a first operation that flushed any record of the KVstore collection
- a value for the end of the maintenance period in epochtime (field `maintenance_mode_end`)
- the current time in epochtime (field `time_updated`)

Example: Enable the maintenance mode till the 11 of May 2019 at 9.pm

For Splunk 7.3.0 and later, replace with `-H "Authorization: Bearer <token>"`

```
curl -k -H "Authorization: Splunk $token" -X DELETE \
  https://$splunk_url/servicesNS/nobody/telegraf-kafka/storage/collections/data/kv_
  ↳telegraf_kafka_alerting_maintenance

curl -k -H "Authorization: Splunk $token" \
  https://$splunk_url/servicesNS/nobody/telegraf-kafka/storage/collections/data/kv_
  ↳telegraf_kafka_alerting_maintenance \
  -H 'Content-Type: application/json' \
  -d '{"maintenance_mode": "enabled", "maintenance_mode_end": "1557565200", "time_
  ↳updated": "1557509578"}'
```

Disabling the maintenance mode

Disabling the maintenance mode requires:

- a first operation that flushed any record of the KVstore collection
- the current time in epochtime (field `time_updated`)

For Splunk 7.3.0 and later, replace with `-H "Authorization: Bearer <token>"`

```
curl -k -H "Authorization: Splunk $token" -X DELETE \
  https://$splunk_url/servicesNS/nobody/telegraf-kafka/storage/collections/data/kv_
  ↳telegraf_kafka_alerting_maintenance
```

(continues on next page)

(continued from previous page)

```
curl -k -H "Authorization: Splunk $token" \
  https://$splunk_url/servicesNS/nobody/telegraf-kafka/storage/collections/data/kv_
↳telegraf_kafka_alerting_maintenance \
  -H 'Content-Type: application/json' \
  -d '{"maintenance_mode": "disabled", "maintenance_mode_end": "", "time_updated":
↳"1557509578"}'
```

Kafka Connect task monitoring management

Retrieve all the records from the KVstore

For Splunk 7.3.0 and later, replace with `-H "Authorization: Bearer <token>"`

```
curl -k -H "Authorization: Splunk $token" \
  https://$splunk_url/servicesNS/nobody/telegraf-kafka/storage/collections/data/kv_
↳telegraf_kafka_connect_tasks_monitoring
```

Request tasks inventory update: automatically Add any new task to the collection

For Splunk 7.3.0 and later, replace with `-H "Authorization: Bearer <token>"`

```
curl -k -H "Authorization: Splunk $token" https://$splunk_url/servicesNS/nobody/
↳telegraf-kafka/search/jobs -d search="| savedsearch \"Update Kafka Connect tasks_
↳inventory\""
```

Create a new connector to be monitored

Create a new connector entry which enables monitoring for the connector, with recommended fields (env, label, connector, role):

Example:

```
{"env": "docker_env", "label": "testing", "connector": "kafka-connect-my-connector",
↳"role": "kafka_sink_task", "monitoring_state": "enabled", "grace_period": "300"}
```

For Splunk 7.3.0 and later, replace with `-H "Authorization: Bearer <token>"`

```
curl -k -H "Authorization: Splunk $token" \
  https://$splunk_url/servicesNS/nobody/telegraf-kafka/storage/collections/data/kv_
↳telegraf_kafka_connect_tasks_monitoring \
  -H 'Content-Type: application/json' \
  -d '{"env": "docker_env", "label": "testing", "connector": "kafka-connect-my-
↳connector", "role": "kafka_sink_task", "monitoring_state": "enabled", "grace_period
↳": "300"}'
```

Get the entries for a specific connector

example:

```
query={"env": "docker_env", "label": "testing", "connector": "kafka-connect-my-connector"}
```

Encode the URL and use a query:

Notes: URI encode everything after the “query=”

For Splunk 7.3.0 and later, replace with `-H “Authorization: Bearer <token>”`

```
curl -k -H "Authorization: Splunk $token" \
  https://$splunk_url/servicesNS/nobody/telegraf-kafka/storage/collections/data/kv_
  telegraf_kafka_connect_tasks_monitoring?query=%7B%22connector%22%3A%20%22kafka-
  connect-my-connector%22%7D
```

Delete a Kafka connector

Delete the record with the key ID “5410be5441ba15298e4624d1”:

For Splunk 7.3.0 and later, replace with `-H “Authorization: Bearer <token>”`

```
curl -k -H "Authorization: Splunk $token" -X DELETE \
  https://$splunk_url/servicesNS/nobody/telegraf-kafka/storage/collections/data/kv_
  telegraf_kafka_connect_tasks_monitoring/5410be5441ba15298e4624d1
```

Deactivating the monitoring state of a connector

Using a search triggered via rest call: (a different method is possible by altering the record, see after)

For Splunk 7.3.0 and later, replace with `-H “Authorization: Bearer <token>”`

```
curl -k -H "Authorization: Splunk $token" https://$splunk_url/servicesNS/nobody/
  telegraf-kafka/search/jobs -d search="| inputlookup kafka_connect_tasks_monitoring_
  | search env=\"docker_env\" label=\"testing\" connector=\"kafka-connect-syslog\" |
  eval monitoring_state=\"disabled\" | outputlookup kafka_connect_tasks_monitoring_
  append=t key_field=_key"
```

Or using a rest call (all wanted fields have to be mentioned):

- get the key ID, and if required get the current value of every field to be preserved

For Splunk 7.3.0 and later, replace with `-H “Authorization: Bearer <token>”`

```
curl -k -H "Authorization: Splunk $token" \
  https://$splunk_url/servicesNS/nobody/telegraf-kafka/storage/collections/data/kv_
  telegraf_kafka_connect_tasks_monitoring/5cd5a890e3b965791163eb71 \
  -H 'Content-Type: application/json' \
  -d '{"env": "docker_env", "label": "testing", "connector": "kafka-connect-my-
  connector", "role": "kafka_sink_task", "monitoring_state": "disabled", "grace_period
  ": "300"}'
```

Activating the monitoring state of a connector

Using a search triggered via rest call: (a different method is possible by altering the record, see after)

For Splunk 7.3.0 and later, replace with `-H “Authorization: Bearer <token>”`

```
curl -k -H "Authorization: Splunk $token" https://$splunk_url/servicesNS/nobody/
↳telegraf-kafka/search/jobs -d search="| inputlookup kafka_connect_tasks_monitoring_
↳| search env=\"docker_env\" label=\"testing\" connector=\"kafka-connect-syslog\" |_
↳eval monitoring_state=\"enabled\" | outputlookup kafka_connect_tasks_monitoring_
↳append=t key_field=_key"
```

Or using a rest call (all wanted fields have to be mentioned):

- get the key ID, and if required get the current value of every field to be preserved

For Splunk 7.3.0 and later, replace with `-H "Authorization: Bearer <token>"`

```
curl -k -H "Authorization: Splunk $token" \
  https://$splunk_url/servicesNS/nobody/telegraf-kafka/storage/collections/data/kv_
↳telegraf_kafka_connect_tasks_monitoring/5cd5a890e3b965791163eb71 \
  -H 'Content-Type: application/json' \
  -d '{"env": "docker_env", "label": "testing", "connector": "kafka-connect-my-
↳connector", "role": "kafka_sink_task", "monitoring_state": "enabled", "grace_period
↳": "300"}'
```

Delete a connector

example:

```
query={"env": "docker_env", "label": "testing", "connector": "kafka-connect-my-
↳connector"}
```

Encode the URL and use a query:

Notes: URI encode everything after the “query=”

For Splunk 7.3.0 and later, replace with `-H "Authorization: Bearer <token>"`

```
curl -k -H "Authorization: Splunk $token" \
  https://$splunk_url/servicesNS/nobody/telegraf-kafka/storage/collections/data/kv_
↳telegraf_kafka_connect_tasks_monitoring?query=%7B%22connector%22%3A%20%22kafka-
↳connect-my-connector%22%7D
```

Delete the record using the key ID:

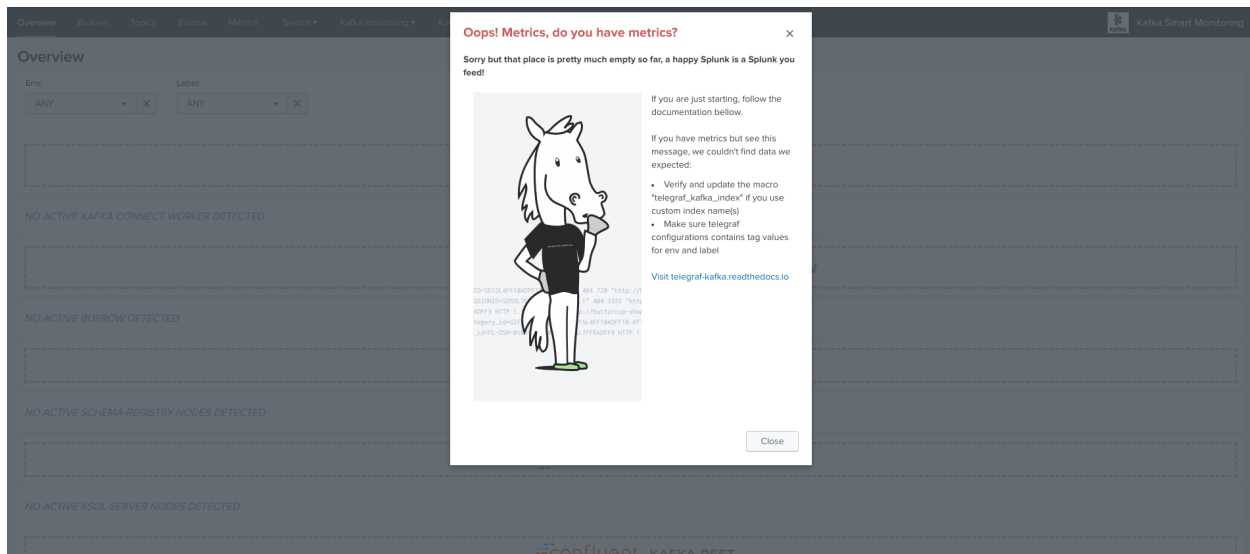
For Splunk 7.3.0 and later, replace with `-H "Authorization: Bearer <token>"`

```
curl -k -H "Authorization: Splunk $token" -X DELETE \
  https://$splunk_url/servicesNS/nobody/telegraf-kafka/storage/collections/data/kv_
↳telegraf_kafka_connect_tasks_monitoring/5410be5441ba15298e4624d1
```


3.1 Troubleshoot & FAQ

3.1.1 No metrics data could be found

Shall the application not be able to find any Kafka metrics in Splunk in respect with your configuration, the Overview landing page will show the following modal message:



Root causes can be:

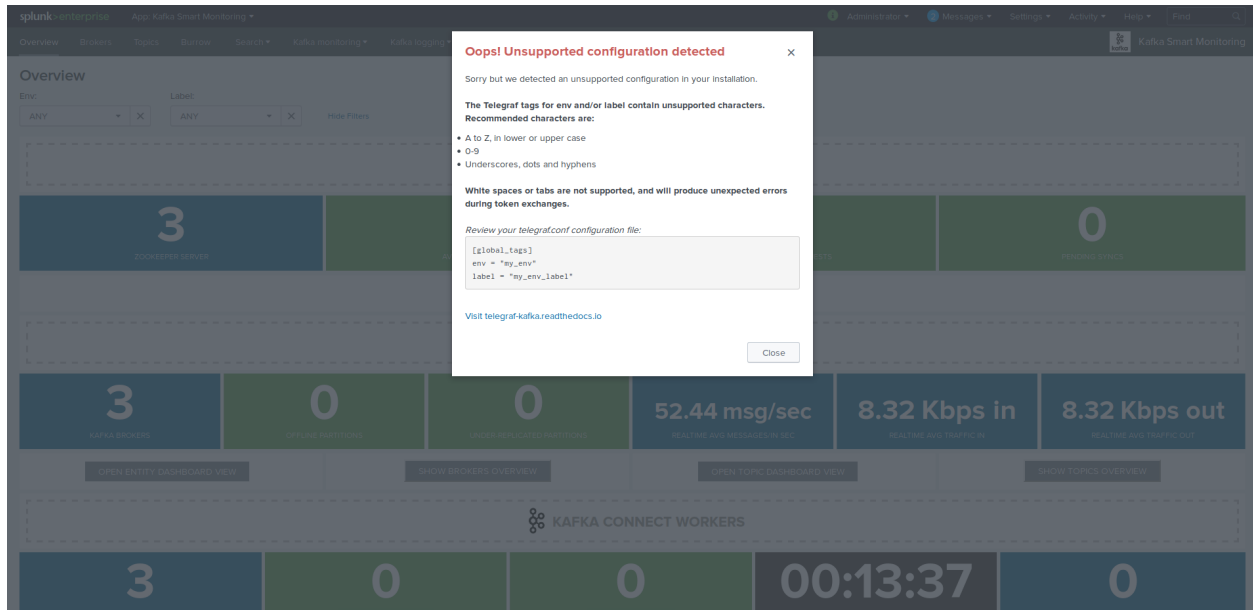
- No metrics could be collected by Telegraf
- Telegraf cannot send the metrics to your Splunk HEC endpoint
- The index name differs from the default “telegraf_kafka” index and the macro “telegraf_kafka_index” was not customised

- Telegraf configuration does not provide a value for env and label tags

3.1.2 Unsupported configuration detected in tags

The value for env and label tags must not contain any white spaces or tabs to avoid token exchange issues in the dashboard navigation.

Shall this be the case, a modal error window will open automatically when opening the Overview page:



If this happens, then your Telegraf tags are incorrect, fix your telegraf.conf configuration files, example:

```
[global_tags]
# the env tag is used by the application for multi-environments management
env = "docker_env"
# the label tag is an optional tag used by the application that you can use as
↳ additional label for the services or infrastructure
label = "testing_env"
```

Once the fix has been applied properly, the error message will disappear.

Versioning and build history:

4.1 Release notes

4.1.1 Version 1.1.39

- Feature: Improvements on Confluent Interceptor UI and related Overview page subcomponents
- Feature: OS Main KPIs views in each component dashboard
- Change: navigation bar review

4.1.2 Version 1.1.38

- Feature: Integration with Confluent Interceptor Monitoring for producers and consumers advanced lag monitoring with Confluent
- Change: Kafka Smart Monitoring goes Dark theme!
- Change: When no Kafka Connect source or sink connector have been detected, show the aggregated single form in black rather than yellow
- Fix: Issue #64 - Missing double quotes in the flush interval telegraf config UI generator #64
- Fix: Issue #66 - Missing dot in telegraf config for Kafka Connect #66
- Fix: Issue #67 - Bad image file path in broker view #67

4.1.3 Version 1.1.37

CAUTION: Zookeeper metric collection switches from Telegraf plugin to JMX for easier and more consistent metric collection across all the components. If you were using the application prior to this version, please update your configuration to collect metrics via JMX.

See: <https://telegraf-kafka.readthedocs.io/en/latest/implementation.html>

- Fix: Show Kafka Connect tasks return empty results in home page
- Fix: The usage of unit makes single form content way too small in different views

4.1.4 Version 1.1.36

CAUTION: Zookeeper metric collection switches from Telegraf plugin to JMX for easier and more consistent metric collection across all the components. If you were using the application prior to this version, please update your configuration to collect metrics via JMX.

See: <https://telegraf-kafka.readthedocs.io/en/latest/implementation.html>

- Fix: Avoid mcatalog calls with metric_name as metric_name, some versions of Splunk will incorrectly complain about this, and this causes appinspect failures in Splunk Base

4.1.5 Version 1.1.35

CAUTION: Zookeeper metric collection switches from Telegraf plugin to JMX for easier and more consistent metric collection across all the components. If you were using the application prior to this version, please update your configuration to collect metrics via JMX.

See: <https://telegraf-kafka.readthedocs.io/en/latest/implementation.html>

- change: Zookeeper metric collection switches from Telegraf plugin to JMX collection via Jolokia Telegraf input
- fix: Topic entity dashboard should break by topic rather than per broker, fix aggregation when any selected
- fix: Burrow does not show up in Overview page
- fix: Telegraf configuration helper UI is broken, and update for Zookeeper collection
- fix: appinspect warnings
- fix: Increase time range of searches in Overview page to better cover longer time between measures
- fix: Kafka Connect connectors discovery does not preserve a non default monitoring_state

4.1.6 Version 1.1.34

- unpublished

4.1.7 Version 1.1.33

CAUTION: Zookeeper metric collection switches from Telegraf plugin to JMX for easier and more consistent metric collection across all the components. If you were using the application prior to this version, please update your configuration to collect metrics via JMX.

See: <https://telegraf-kafka.readthedocs.io/en/latest/implementation.html>

- change: Zookeeper metric collection switches from Telegraf plugin to JMX collection via Jolokia Telegraf input
- fix: Topic entity dashboard should break by topic rather than per broker, fix aggregation when any selected
- fix: Burrow does not show up in Overview page
- fix: Telegraf configuration helper UI is broken, and update for Zookeeper collection
- fix: appinspect warnings

- fix: Increase time range of searches in Overview page to better cover longer time between measures

4.1.8 Version 1.1.32

CAUTION: Zookeeper metric collection switches from Telegraf plugin to JMX for easier and more consistent metric collection across all the components. If you were using the application prior to this version, please update your configuration to collect metrics via JMX.

See: <https://telegraf-kafka.readthedocs.io/en/latest/implementation.html>

- change: Zookeeper metric collection switches from Telegraf plugin to JMX collection via Jolokia Telegraf input
- fix: Topic entity dashboard should break by topic rather than per broker, fix aggregation when any selected
- fix: Burrow does not show up in Overview page
- fix: Telegraf configuration helper UI is broken, and update for Zookeeper collection
- fix: appinspect warnings

4.1.9 Version 1.1.31

CAUTION: Zookeeper metric collection switches from Telegraf plugin to JMX for easier and more consistent metric collection across all the components. If you were using the application prior to this version, please update your configuration to collect metrics via JMX.

See: <https://telegraf-kafka.readthedocs.io/en/latest/implementation.html>

- change: Zookeeper metric collection switches from Telegraf plugin to JMX collection via Jolokia Telegraf input
- fix: Topic entity dashboard should break by topic rather than per broker, fix aggregation when any selected
- fix: Burrow does not show up in Overview page
- fix: Telegraf configuration helper UI is broken, and update for Zookeeper collection

4.1.10 Version 1.1.30

- fix: Realtime traffic In and Out refer to the same field in Overview for Kafka Brokers

4.1.11 Version 1.1.29

- fix: Realtime traffic In and Out refer to the same field in Kafka broker entity view

4.1.12 Version 1.1.28

- feature: Improvement of the maintenance mode with start date time selection capability and automatic scheduling

4.1.13 Version 1.1.27

- fix: Drilldown link broken for Kafka Broker view from dynamic Brokers overview in main Overview dashboards, or Kafka Brokers dashboard (change introduced in 1.1.26)

4.1.14 Version 1.1.26

- fix: The total number of Kafka Connect connectors reported in Alerting management UI is incorrect if connectors have the same ID across multiple tenants
- fix: Cleaning / deletion of unused css and js objects
- feature: Improved table icons rendering with courtesy of Chrys Younger
- feature: Improved Broker overview panels in Overview and Brokers views

4.1.15 Version 1.1.25

- fix: Regression introduced in version 1.1.21 impacts the Kafka Connect tasks inventory if a task is inactive or removed for a long period
- fix: Incorrect number of connectors reported in Alerting managing interface if connectors have the same names across environments

4.1.16 Version 1.1.24

- feature: Introducing logs mapping macros used in entity views to provide customization capabilities for logs integration

4.1.17 Version 1.1.23

- fix: Missing env/label filters in entity views impact results if multiple env/label and ANY selected

4.1.18 Version 1.1.22

- fix: Improves searches for Connected experience dashboard for Kafka Connect (listing connectors in alert)
- fix: Improves Telegraf configuration generator modal window rendering and adds link button to documentation
- fix: Missing env / label filtering in show tasks in alert button from Overview
- feature: Add dynamic view inclusion in menu for Connected Experience custom dashboards

4.1.19 Version 1.1.21

- feature: Introduction of the Telegraf configuration generator, a guided user interface that generates the telegraf.conf configuration files depending on your requirements
- feature: Adding new Audit menu with builtin Audit dashboard for scheduled performance and daily volume indexing analysis
- feature: Use bootstrap buttons in Overview rather than custom buttons design
- feature: Adding active button in Overview to show Kafka Connect tasks in alert (tasks not reporting)
- feature: Adding drilldown from single in Overview / Kafka Connect health views for failed connectors / failed tasks
- feature: Store the last operational time of Kafka Connect connectors in the KVstore, update at inventory / alert run time, return when an alert triggers

- fix: Lag field missing in table from Splunk Connected experience Burrow dashboard due to typo

4.1.20 Version 1.1.20

- fix: Remove any console.log (even while these are commented) in javascript to avoid manual check from appinspect

4.1.21 Version 1.1.19

- fix: Static index reference in new Splunk Connected experience Kafka Connect dashboard
- fix: lag field name type in new Splunk Connected experience Burrow dashboard
- fix: Remove restart required after installation to allow installation in Splunk Cloud via Self-services (SSAI)
- feature: Adding Splunk Connected experience mobile dashboard for Zookeeper health

4.1.22 Version 1.1.18

- feature: Introduction of the builtin kafka_admin role
- feature: Provides default kafka_admin role member write access to the application name space and the KVstore based lookup collections
- feature: Introduction of the Connected Experience dashboards, Health overview dashboards designed to be used with Splunk Connected Experience and Splunk Cloud Gateway (Splunk Mobile, Splunk TV)
- fix: Static indexes references in Kafka Connect and Kafka Burrow dashboards

4.1.23 Version 1.1.17

- fix: Expose units for Zookeeper latency metrics in Overview and entity view
- feature: Introducing the smart component enablement, which allows enabling / disabling a Kafka component to be visible from the Overview, to be managed via the configuration user interface
- feature: Expose Zookeeper leader and Broker active controller in Overview dashboard when mono tenancy (environment) detected or selected
- feature: Configuration checker, detect incomplete installation (Kafka inventory not updated) when loading Overview, and provide modal update user interaction
- fix: Prevents multiple endpoint calls in Alerting User Interface management in Ajax

4.1.24 Version 1.1.16

- feature: Spinner during update / rebuild of KVstore collections within the management of embedded alerting UI
- feature: Manage unprivileged user access to the UI, and proper error handling due to lack of permission against the KVstore collections
- fix: Improved handling of topics / connectors / consumers discovery reports
- feature: Kafka Brokers OOTB alerts and Kafka Connect connector or task startup failure detected are not linked to a monitoring_state that can be deactivated via the KVstore collections

- feature: Configuration error checker which verifies at overview loading page for unsupported tags in env/label such as white spaces.

4.1.25 Version 1.1.15

- feature: Major improvements of the user experience with the management of embedded alerting via modal contextual user interactions
- feature: Maintenance mode is now time conditioned with an end of maintenance period requested via UI calendar during activation
- feature: Migration to native modal windows for user interactions in the alerting management user interface (removal of bootstrap.js plugin)
- feature: Default schedule change of the maintenance mode status verification report
- feature: Request Splunk restart by default in app.conf
- fix: Kafka Connect tasks that are paused do not properly affect the aggregated state single form in Overview
- fix: Burrow task single form in Overview page results in appendcols related error in Overview page within Splunk 7.0.x
- fix: Regression in Kafka Connect task listing for Splunk 7.0.x in PostProcess search due to append (introduced by Alerting Management UI)
- fix: Regression in dynamic table overview for Kafka Connect status per task in Overview (introduced by 1.1.14)

4.1.26 Version 1.1.14

- feature: Major improvements of the user experience with the management of embedded alerting via modal contextual user interactions
- feature: Maintenance mode is now time conditioned with an end of maintenance period requested via UI calendar during activation
- feature: Migration to native modal windows for user interactions in the alerting management user interface (removal of bootstrap.js plugin)
- feature: Default schedule change of the maintenance mode status verification report
- feature: Request Splunk restart by default in app.conf
- fix: Kafka Connect tasks that are paused do not properly affect the aggregated state single form in Overview
- fix: Add Kafka Connect tasks in the dynamic table tasks overview if the tasks are listed as monitored in the collection, and the tasks do not report metrics currently (collection stopped, tasks were removed but not from collection)
- fix: Burrow task single form in Overview page results in appendcols related error in Overview page within Splunk 7.0.x

4.1.27 Version 1.1.13

- fix: Static span is defined in Burrow detailed view charts
- fix: Prevents removed Burrow consumers to appear as low range when latest metrics available are part of the selected time range

- fix: Missing group by statement for Burrow consumers monitoring in OOTB alert, generates unexpected output containing OK consumers, while alerts are correctly justified for ERR consumers

Version 1.1.12

- feature: Adding drilldown to single forms for Offline and Under-replicated partitions in Overview and Kafka Brokers entities views
- fix: ISR Shrinking missing env/label/broker filters in Kafka broker entity view
- feature: Better table rendering in Kafka broker entity view for Under-replicated partitions

Version 1.1.11

- feature: Improvement of the Alerting framework management interface with tabs categorization, capability to update and reset collections on demand, alert activation summary, UI experience greatly improved
- fix: Prevent low range state for Kafka Connect tasks that were recently deleted in tasks overview
- fix: Improve Kafka Connect tasks table in Kafka Connect entity view
- fix: Pastel red color for under-replicated partitions in topics views
- fix: Properly order per topic/partitions in broker entity table view
- fix: Prevents a failing component that was unreachable for a long period to be entirely removed from the infrastructure collection, replaced by a disabled_autoforced monitoring_state value if downtime>24 hours
- fix: Preserve _key_id of KVstore collections during updates for kafka_infra_inventory / kafka_infra_nodes_inventory lookups

Version 1.1.10

- fix: Static index references instead of macro usage in Kafka Connect entity view, Kafka Connect status report and drilldown links
- fix: Switch to dropdown selector for env/label in Overview to avoid multiselect issues with forwarding tokens to dashboards

Version 1.1.9

- fix: Static index reference instead of macro usage in Kafka Connect report

Version 1.1.8

- feature: Improvements of the Kafka Connect task status overview report
- feature: Add icon ranges and filters for Kafka Connect task status overview from Overview main dashboard, configure drilldown from table to entity views

Version 1.1.7

- feature: Add input text filter for Consumers in UI Monitoring management
- fix: Non working filters for Consumers / partitions in UI Burrow
- feature: Map monitoring_state in Consumers status preview in Overview

Version 1.1.6

- fix: incompatibility for ksql-server with latest Confluent release (5.1.x) due to metric name changes in JMX model
- feature: avoid no results returned by single in Overview page for Burrow when no consumers are yet added to the monitored collection

Version 1.1.5

Burrow integration: Kafka Consumer Lag monitoring

- feature: Integration of Burrow, new Burrow consumer lag monitoring UIs
- feature: Management of Kafka consumers state within the alerting framework
- feature: Integration of Burrow consumers state within the Overview UI
- feature: Schedule Kystore collection update reports (infra, topics, tasks, consumers) on a per 4 hours basis
- fix: Prevents user from attempting to disable maintenance mode when already disabled, and vice-versa
- fix: Properly sort Connect tasks statuses on Overview page to show Unknown status when tasks are missing but monitored

The Burrow integration provides advanced threshold less lag monitoring for Kafka Consumers, such as Kafka Connect connectors and Kafka Streams.

Version 1.1.4

Burrow integration: Kafka Consumer Lag monitoring

- feature: Integration of Burrow, new Burrow consumer lag monitoring UIs
- feature: Management of Kafka consumers state within the alerting framework
- feature: Integration of Burrow consumers state within the Overview UI
- feature: Schedule Kystore collection update reports (infra, topics, tasks, consumers) on a per 4 hours basis
- fix: Prevents user from attempting to disable maintenance mode when already disabled, and vice-versa

The Burrow integration provides advanced threshold less lag monitoring for Kafka Consumers, such as Kafka Connect connectors and Kafka Streams.

Version 1.1.3

- fix: Properly order partitions in new Brokers detailed UI
- fix: Allows selection of special topics in entity topic view

Version 1.1.2

- feature: New Brokers/Brokers details, Topics/Topics details UIs inspired from Yahoo kafka-manager
- feature: Allows environment and label selection from Overview, propagates tokens across all UIs
- fix: Incorrect number of partitions reported within Brokers entity view when multiple Brokers are selected

Version 1.1.1

- fix: Static index called in report Kafka monitoring - tasks status report

Version 1.1.0

CAUTION: Breaking changes, telegraf modification is required to provide global tags for env and label dimensions!

https://da-itsi-telegraf-kafka.readthedocs.io/en/latest/kafka_monitoring.html#telegraf-output-configuration

Upgrade path:

- Upgrade telegraf configuration to provide the env and label tags
- Upgrade the application

Features/fixes:

- feature: Multi-environment / Multi-dc support via env and label tagging at Telegraf metric level, allows embedded management of any number of environment and/or additional sub-dividing notion (multi-env, multi-dc. . .)
- feature: New kvstore collection to allow monitoring of services in a container environment philosophy based on the number of active nodes per role rather than their identity
- feature: Update of the Alerting Management User Interface
- feature: New OOTB Alerting based on active nodes numbers per role
- feature: Refreshed Overview page with layers icons, additional overview in page views
- feature: New applications icons
- fix: Various fixes and improvements

Version 1.0.12

- fix: Improve detection of Kafka Connect tasks not successfully running on the Overview page
- fix: Drilldown on single forms for Kafka Connect tasks

Version 1.0.11

- fix: Management interface toggle panels not working (bad reference in js)
- fix: Management interface disable maintenance button not showing up properly in Splunk 7.0.x
- fix: Preset a default value for maintenance mode status
- fix: share lookups, transforms and macros at system level by default

Version 1.0.10

- Unpublished

Version 1.0.9

- feature: Added OOTB Alert for under-replicated partitions per topics
- feature: Management interface for embedded Kafka alerting
- feature: Enabling / Deactivating maintenance mode through UI for alerting management

Version 1.0.8

- feature: Out of the box alerting templates for Kafka infrastructure
- fix: Kafka Connect aggregated states issues in Overview page

Version 1.0.7

- feature: Out of the box alerts for Kafka Infrastructure
- feature: Support for Confluent ksql-server
- feature: Support for Confluent kafka-rest
- feature: Overview home page improvements
- feature: event logging integration with the TA-kafka-streaming-platform
- fix: minor fixes and improvements in views

Version 1.0.6

- fix: Typo in Overview

Version 1.0.5

- feature: Confluent schema-registry support

Version 1.0.4

- fix: inverted filters for source/task in Overview
- fix: dropdown replaced by multiselect and key per connector/task in source/sink views

Version 1.0.3

- fix: Overview page, link for topic management should be under brokers category

Version 1.0.2

- various: logo update

Version 1.0.1

- fix: missing link for Kafka topics reporting

Version 1.0.0

- initial and first public release