telegraf-kafka Documentation

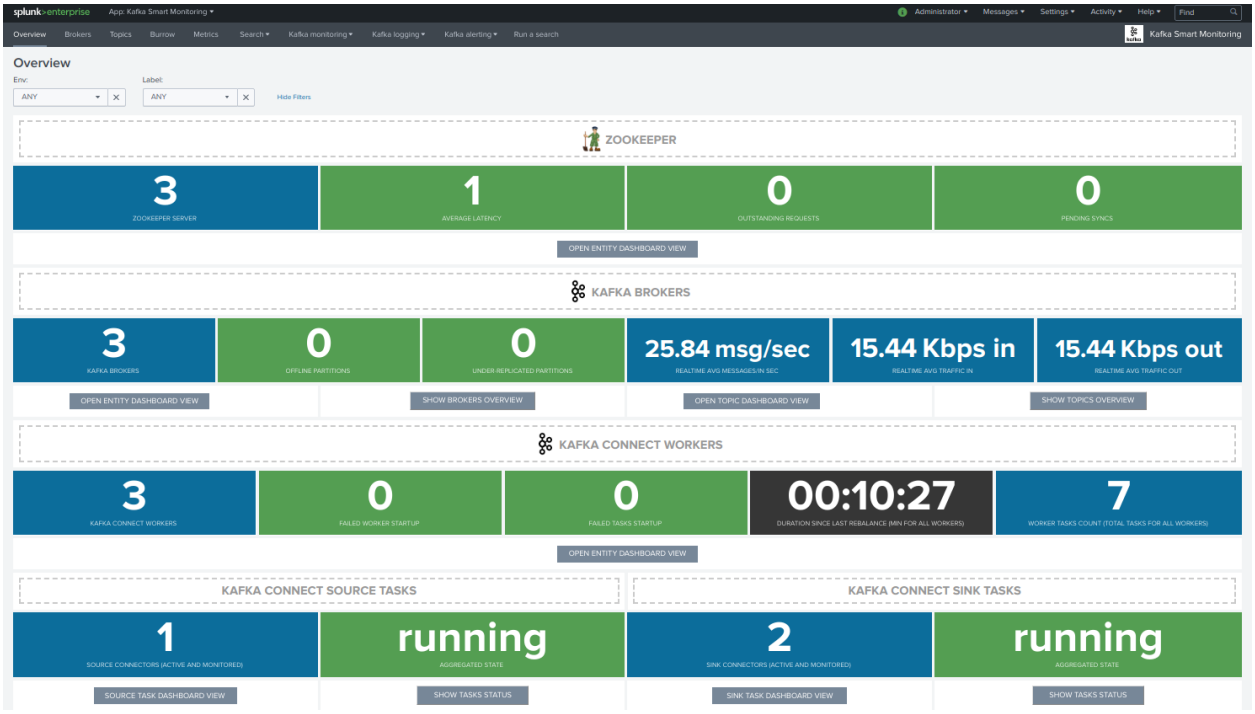
Release 1

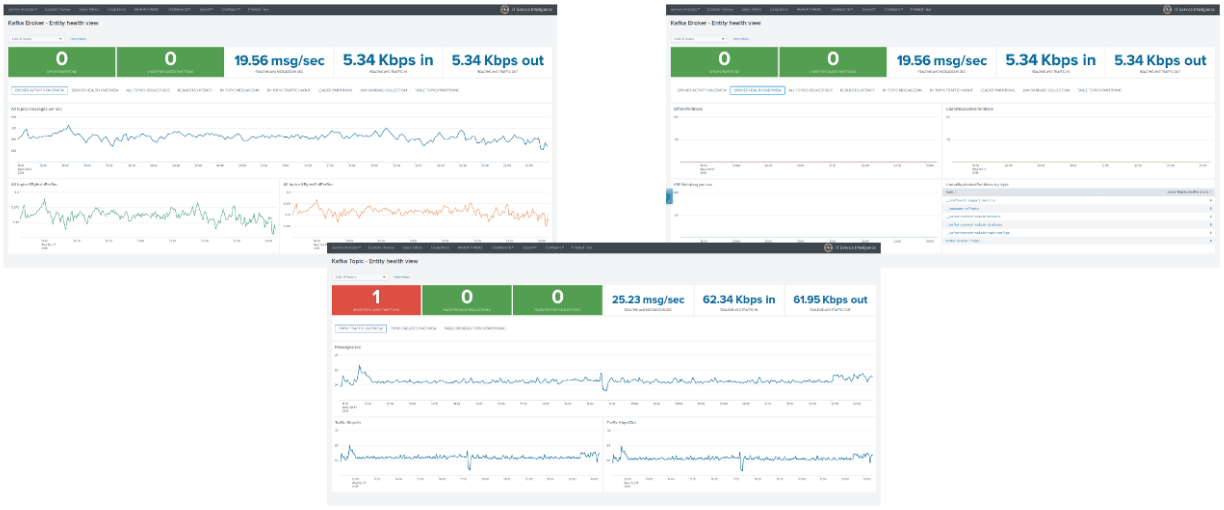
Guilhem Marchand

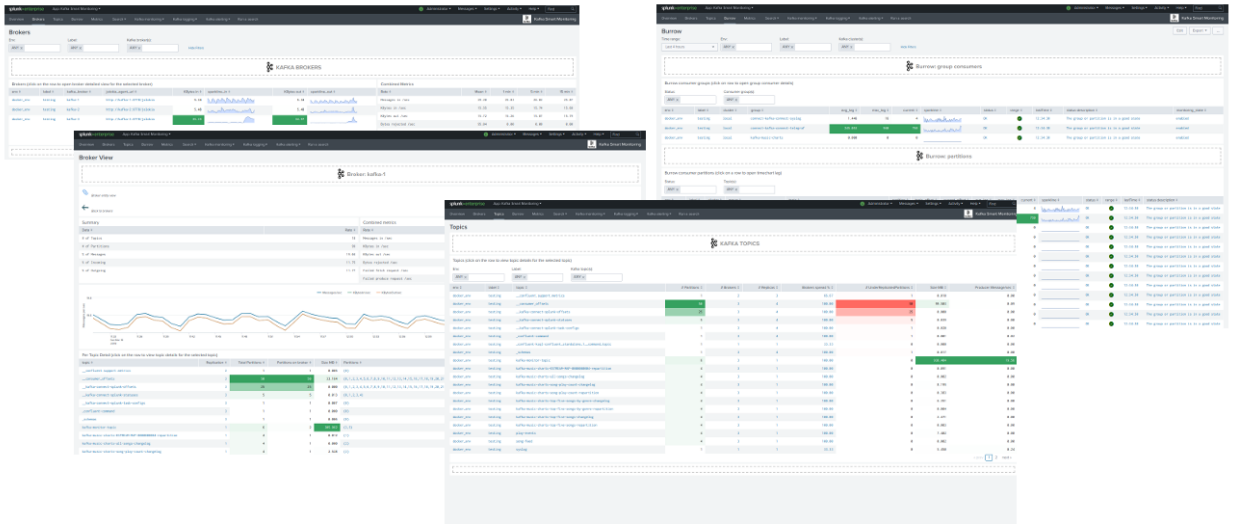
Feb 22, 2020

Contents

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
- 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

CHAPTER 1

Overview:

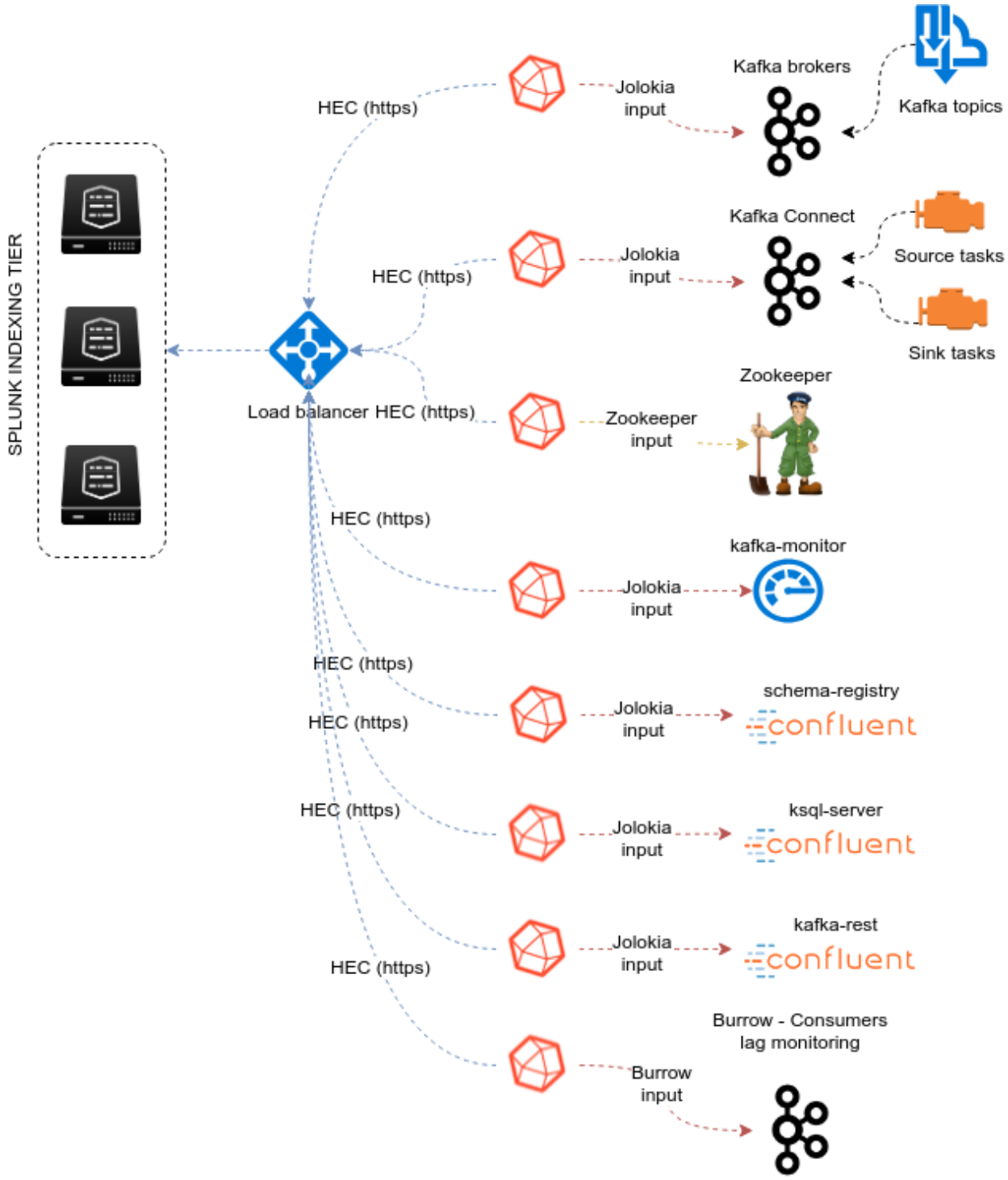
1.1 About

- 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)
- LinkedIn Kafka monitor to provide end to end monitoring (https://github.com/linkedin/kafka-monitor)
- 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

All the metrics are ingested into the high performance Splunk metric store, Splunk 7.0.x or higher is required.

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

Qualification and certification is made against Kafka V2.x and Confluent V5.x, earlier versions might however work with no issues but are not being tested.

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 Answers

Open a question in Splunk answers for the application:

- https://answers.splunk.com/app/questions/4268.html

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 betters, 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

CHAPTER 2

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.

The deployment and configuration requires the creation of a dedicated metric index (by default called **telegraf_kafka**), see the implementation section.

2.1.2 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

2.1.3 Upgrades

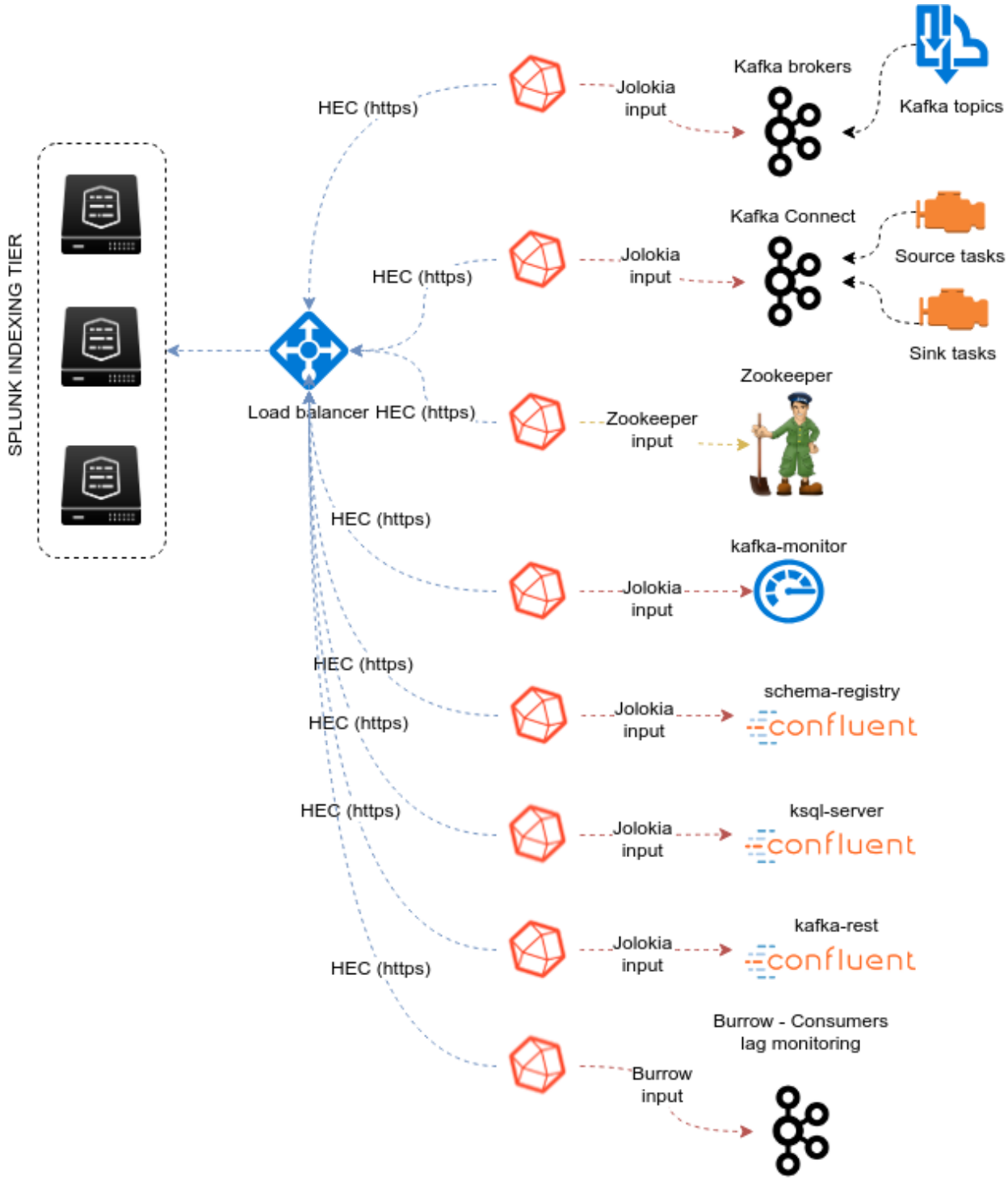
Upgrading the Splunk application is pretty much the same operation than the initial deployment.

2.1.4 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 diagram overview:



2.2.1 Splunk configuration

Index definition

The application relies by default on the creation of a metrics index called "telegraf_kafka":

indexes.conf example with no Splunk volume::

```
[telegraf_kafka]
coldPath = $SPLUNK_DB/telegraf_kafka/colddb
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/colddb
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.

HEC input ingestion and definition

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

inputs.conf example:

```
[http://telegraf_kafka_monitoring]
disabled = 0
index = 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.

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

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/

2.2.3 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 to 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 in a Kafka topic. However, it presents a specific concern for Kafka itself.

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"
```

(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
   splunkmetric_hec_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

2.2.4 Jolokia JVM monitoring

Kafka components are being monitored through the very powerful Jolokia agent:

- https://jolokia.org

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

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

2.2.5 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-jvm-1.6.0-agent.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 Kafka brokers

```
Environment="KAFKA_OPTS=-javaagent:/opt/jolokia/jolokia-jvm-1.6.0-agent.jar=port=8778,
↪host=0.0.0.0"
```

For Kafka Connect

```
Environment="KAFKA_OPTS=-javaagent:/opt/jolokia/jolokia-jvm-1.6.0-agent.jar=port=8778,
↪host=0.0.0.0"
```

For Confluent schema-registry

```
Environment="KAFKA_OPTS=-javaagent:/opt/jolokia/jolokia-jvm-1.6.0-agent.jar=port=8778,
↪host=0.0.0.0"
```

For Confluent ksql-server

```
Environment="KSQL_OPTS=-javaagent:/opt/jolokia/jolokia-jvm-1.6.0-agent.jar=port=8778,
↪host=0.0.0.0"
```

For Confluent kafka-rest

```
Environment="KAFKAREST_OPTS=-javaagent:/opt/jolokia/jolokia-jvm-1.6.0-agent.
↪jar=port=8778,host=0.0.0.0"
```

2.2.6 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-jvm-1.6.0-agent.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.

2.2.7 Zookeeper monitoring

Collecting with Telegraf

The Zookeeper monitoring is very simple and achieved by Telegraf and the Zookeeper input plugin.

The following configuration stands in telegraf.conf and configures the input plugin to monitor multiple Zookeeper servers from one source:

```
# zookeeper metrics
[[inputs.zookeeper]]
  servers = ["zookeeper-1:12181","zookeeper-2:22181","zookeeper-3:32181"]
```

If each server runs an instance of Zookeeper and you deploy Telegraf, you can simply collect from the localhost:

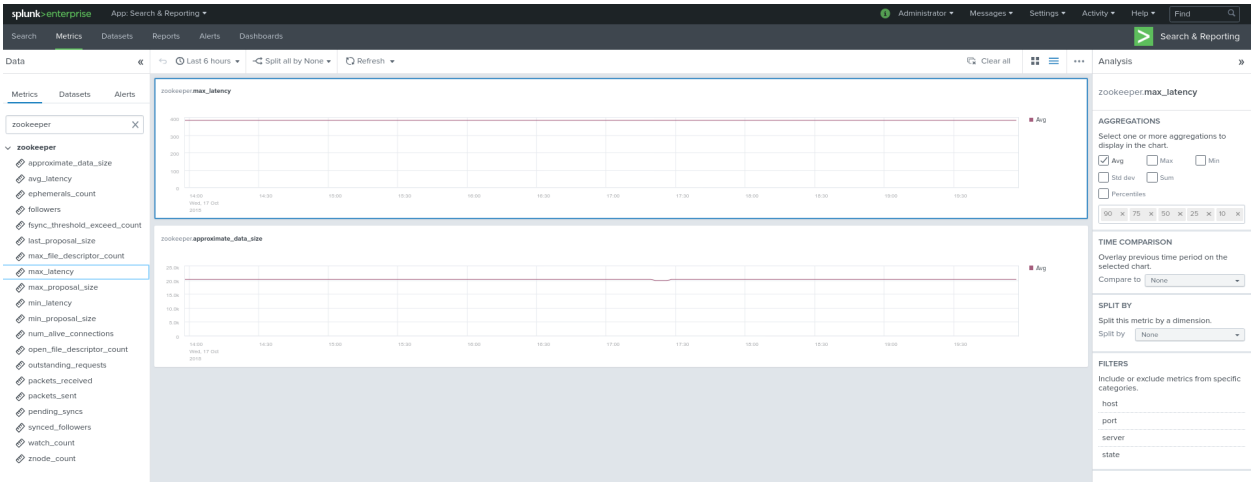
```
# zookeeper metrics
[[inputs.zookeeper]]
  servers = ["$HOSTNAME:2181"]
```

Full telegraf.conf example

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

```
[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
   splunkmetric_hec_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 metrics
[[inputs.zookeeper]]
  servers = ["zookeeper-1:12181","zookeeper-2:22181","zookeeper-3:32181"]
```

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=zookeeper.*
```

2.2.8 Kafka brokers monitoring with Jolokia

Jolokia

example: Jolokia start in docker environment:

```
environment:
  KAFKA_BROKER_ID: 1
  KAFKA_ZOOKEEPER_CONNECT: zookeeper-1:12181,zookeeper-2:12181,zookeeper-3:12181
  KAFKA_ADVERTISED_LISTENERS: PLAINTEXT://kafka-1:19092
  KAFKA_OPTS: "-javaagent:/opt/jolokia/jolokia-jvm-1.6.0-agent.jar=port=8778,host=0.0.
↪0.0"
```

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 brocker machine.

Connecting to multiple remote Jolokia instances:

```
# 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"]
```

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

(continues on next page)

(continued from previous page)

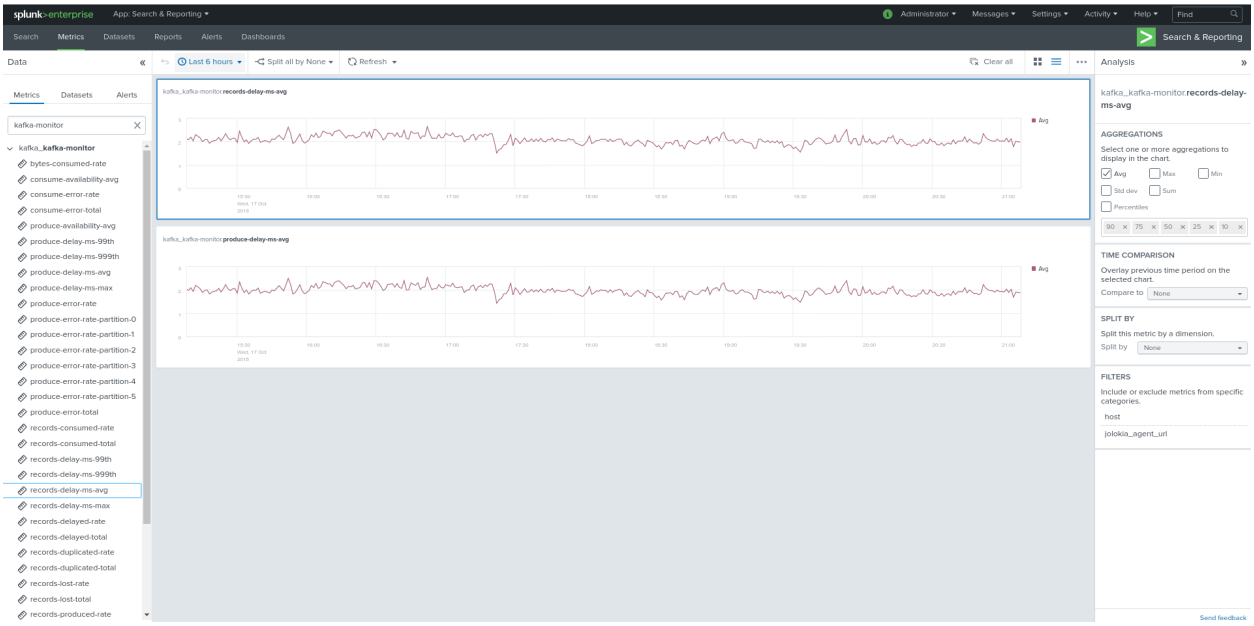
```
# outputs
[[outputs.http]]
   url = "https://splunk:8088/services/collector"
   insecure_skip_verify = true
   data_format = "splunkmetric"
    ## Provides time, index, source overrides for the HEC
   splunkmetric_hec_routing = true
    ## Additional HTTP headers
    [outputs.http.headers]
   # Should 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"
  tag_keys     = ["name"]

[[inputs.jolokia2_agent.metric]]
  name         = "network"
```

(continues on next page)

(continued from previous page)

```
  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_*.*
```

2.2.9 Kafka connect monitoring

Jolokia

example: Jolokia start in docker environment:

```
environment:
  KAFKA_OPTS: "-javaagent:/opt/jolokia/jolokia-jvm-1.6.0-agent.jar=port=18779,host=0.
↪0.0.0"
command: "/usr/bin/connect-distributed /etc/kafka-connect/config/connect-distributed.
↪properties-kafka-connect-1"
```

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

bellow a full telegraf.conf example:

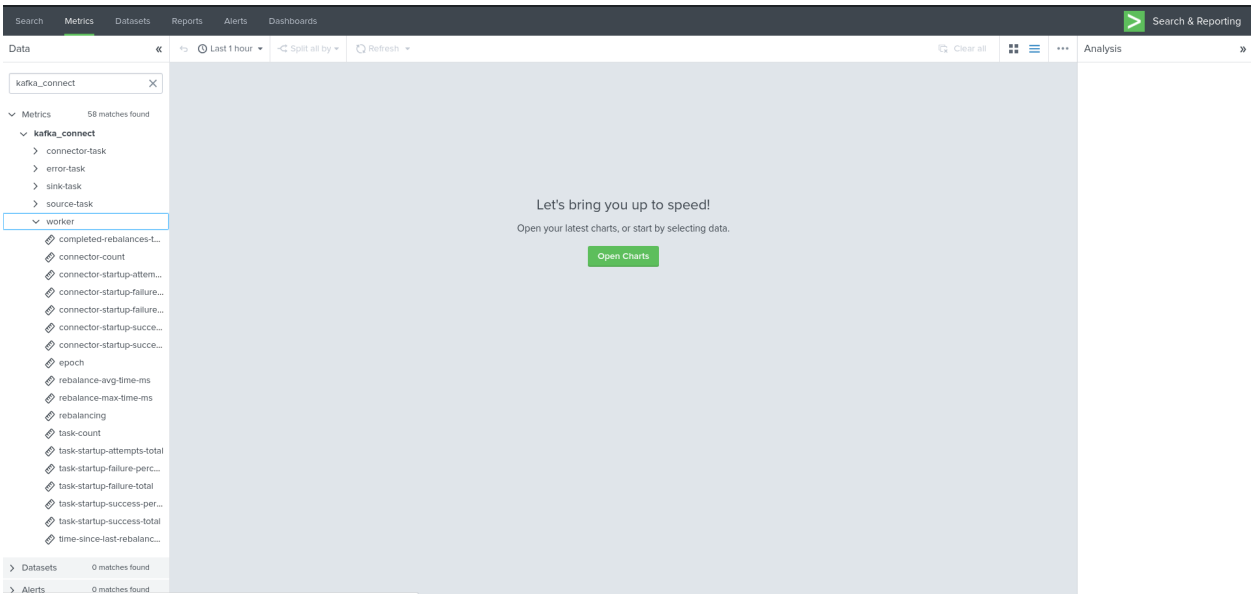
```
[global_tags]
  # the env tag is used by the application for multi-environments management
  env = "my_env"
  # the label tag is an optional tag used by the application that you can use as␣
↪additional label for the services or infrastructure
  label = "my_env_label"

[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
   splunkmetric_hec_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=*"
```

(continues on next page)

(continued from previous page)

```
  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
[[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.*
```

2.2.10 Kafka LinkedIn 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 !

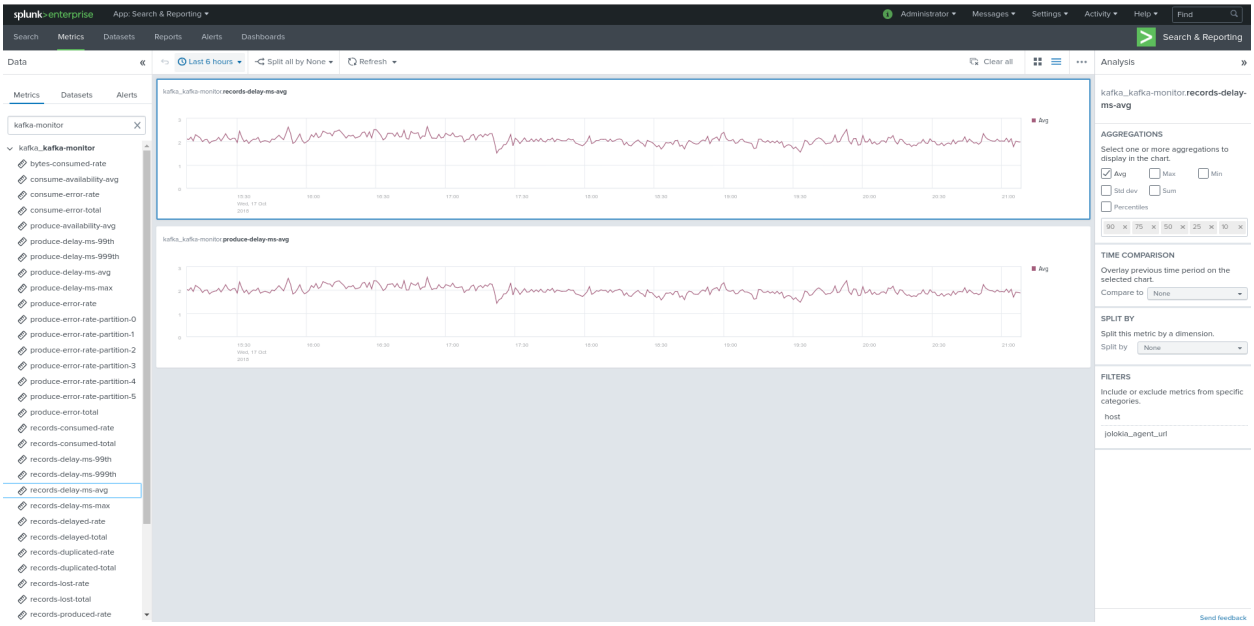
It is very straightforward to run the kafka-monitor in a docker container, first you need to create your own image:

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

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
   splunkmetric_hec_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.*
```

2.2.11 Confluent schema-registry

Jolokia

example: Jolokia start in docker environment:

```
environment:
  SCHEMA_REGISTRY_KAFKASTORE_CONNECTION_URL: zookeeper-1:12181,zookeeper-2:12181,
↪zookeeper-3:12181
  SCHEMA_REGISTRY_HOST_NAME: schema-registry
  SCHEMA_REGISTRY_LISTENERS: "http://0.0.0.0:8081"
  SCHEMA_REGISTRY_OPTS: "-javaagent:/opt/jolokia/jolokia-jvm-1.6.0-agent.
↪jar=port=18783,host=0.0.0.0"
```

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:

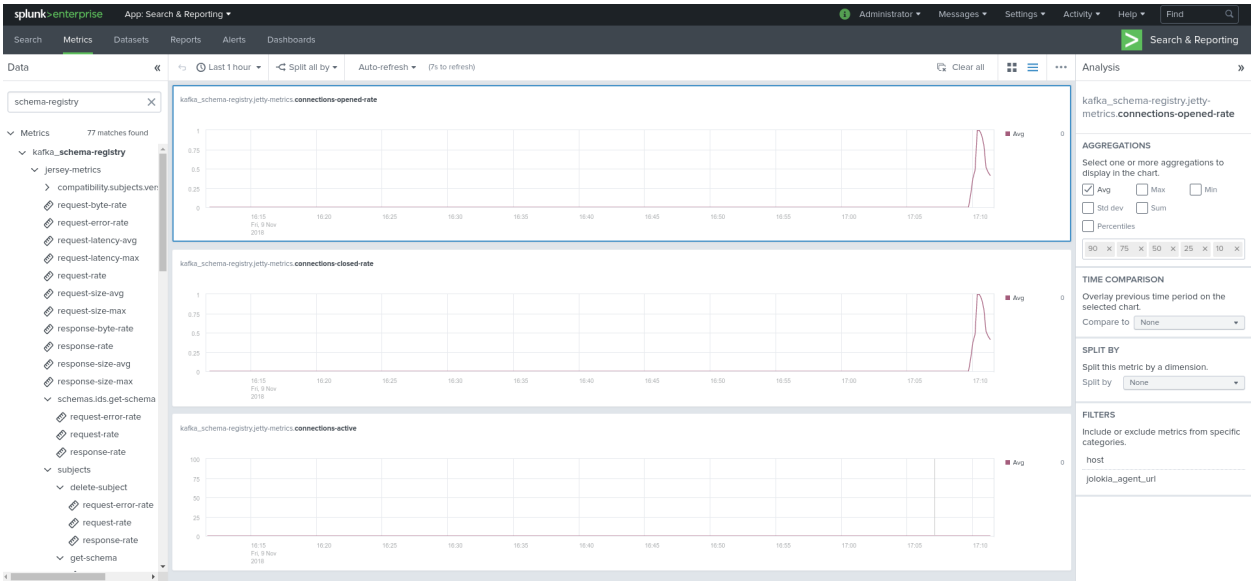
```
# Kafka-connect JVM monitoring
 [[inputs.jolokia2_agent]]
  name_prefix = "kafka_schema-registry."
   urls = ["http://$HOSTNAME:8778/jolokia"]
```

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
  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
   splunkmetric_hec_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.*
```

2.2.12 Confluent ksql-server

Jolokia

example: Jolokia start in docker environment:

```
environment:
  KSQL_BOOTSTRAP_SERVERS: PLAINTEXT://kafka-1:19092,PLAINTEXT://kafka-2:29092,
↪PLAINTEXT://kafka-3:39092
  KSQL_KSQL_SERVICE_ID: confluent_standalone_1_
  SCHEMA_REGISTRY_LISTENERS: "http://0.0.0.0:8081"
  KSQL_OPTS: "-javaagent:/opt/jolokia/jolokia-jvm-1.6.0-agent.jar=port=18784,host=0.0.
↪0.0"
```

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:

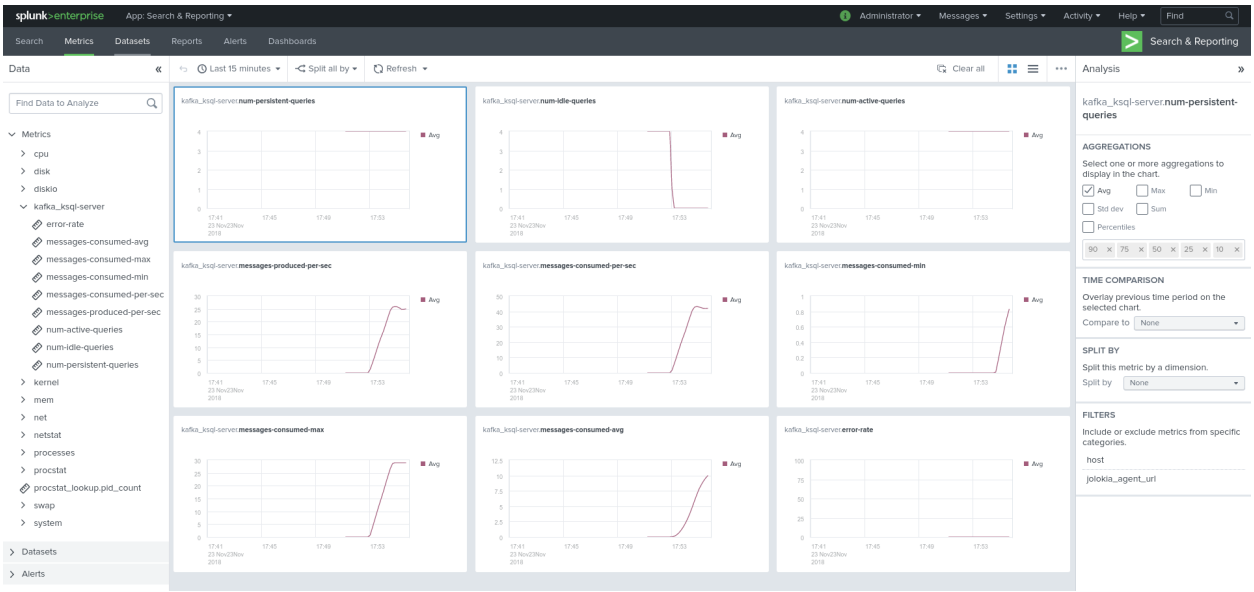
```
[[inputs.jolokia2_agent]]
  name_prefix = "kafka_"
  urls = ["http://$HOSTNAME:18784/jolokia"]
```

Full telegraf.conf example

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
   splunkmetric_hec_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.*
```

2.2.13 Confluent kafka-rest

Jolokia

example: Jolokia start in docker environment:

```
environment:
  KAFKA_REST_ZOOKEEPER_CONNECT: "zookeeper-1:12181,zookeeper-2:22181,zookeeper-3:32181
↪"
  KAFKA_REST_LISTENERS: "http://localhost:18089"
  KAFKA_REST_SCHEMA_REGISTRY_URL: "http://schema-registry-1:18083"
  KAFKAREST_OPTS: "-javaagent:/opt/jolokia/jolokia-jvm-1.6.0-agent.jar=port=18785,
↪host=0.0.0.0"
  KAFKA_REST_HOST_NAME: "kafka-rest"
```

notes: KAFKAREST_OPTS is not a typo, this is (strangely) the right name to configuration java options.

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:

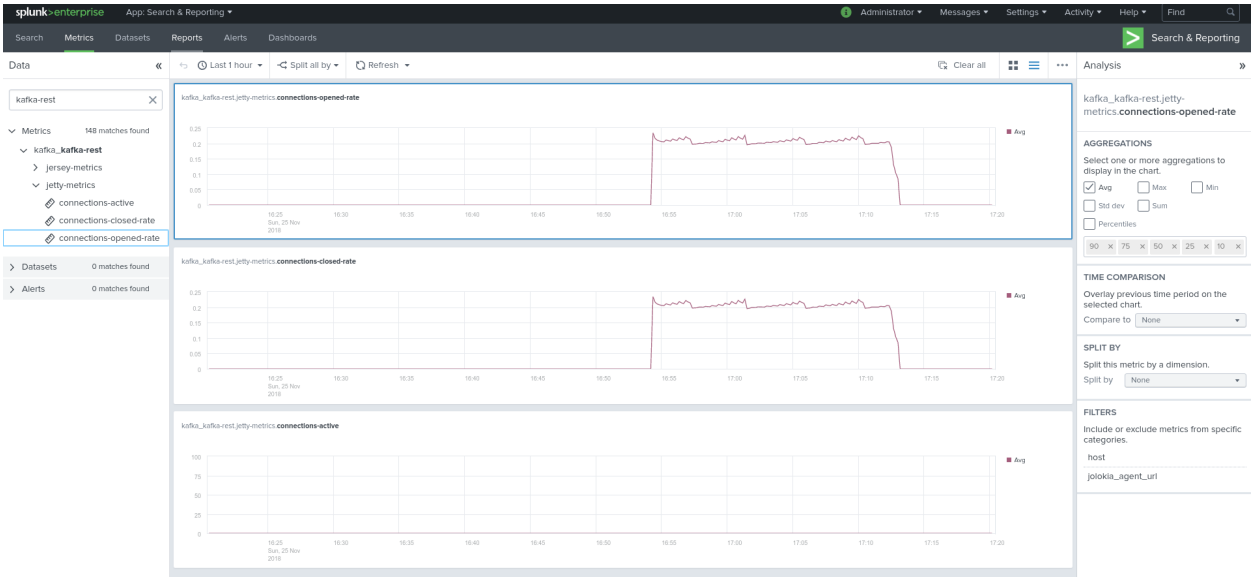
```
[[inputs.jolokia2_agent]]
  name_prefix = "kafka_kafka-rest."
  urls = ["http://$HOSTNAME:18785/jolokia"]
```

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
   splunkmetric_hec_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
↪"]

 [[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:

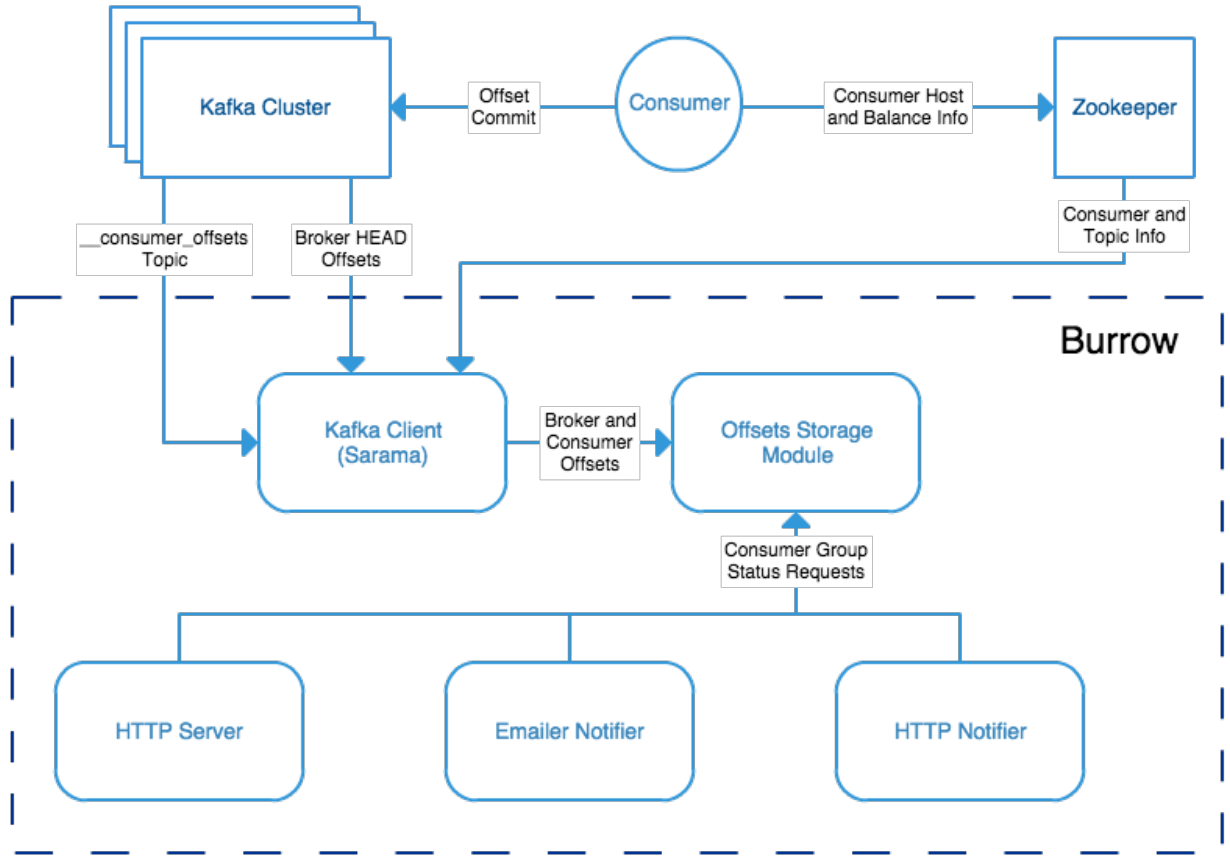
```
| mcatalog values(metric_name) values(_dims) where index=* metric_name=kafka_kafka_
↪kafka-rest.*
```

2.2.14 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

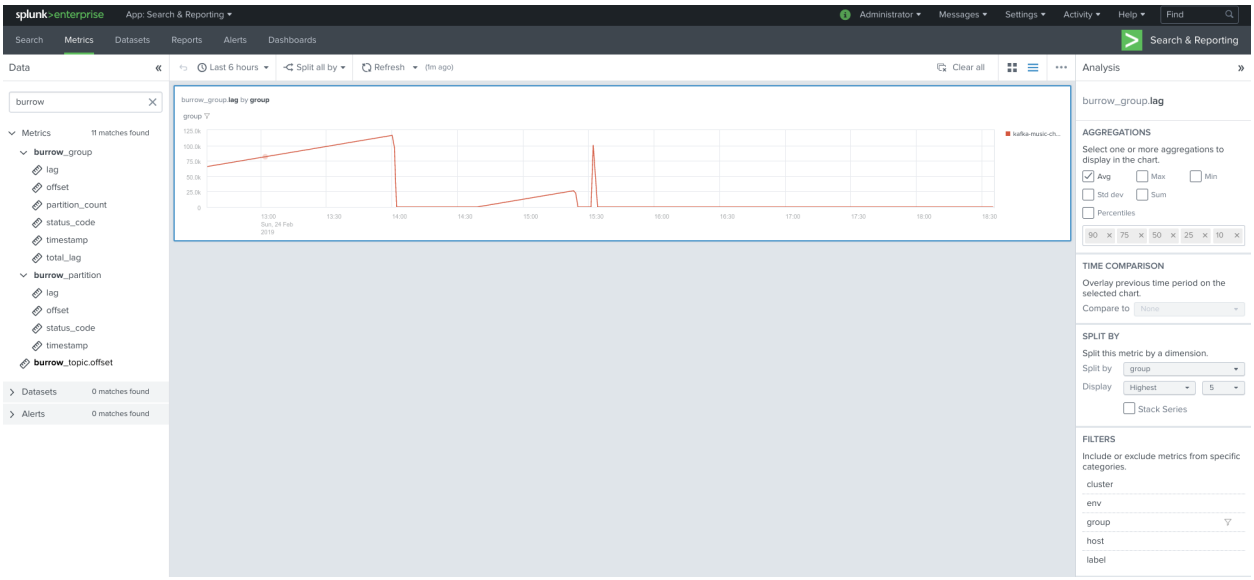
```
[global_tags]
  # the env tag is used by the application for multi-environments management
  env = "my_env"
  # the label tag is an optional tag used by the application that you can use as␣
↪additional label for the services or infrastructure
  label = "my_env_label"

[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
   splunkmetric_hec_routing = true
    ## Additional HTTP headers
```

(continues on next page)

(continued from previous page)

```
    [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 = []

  ## 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

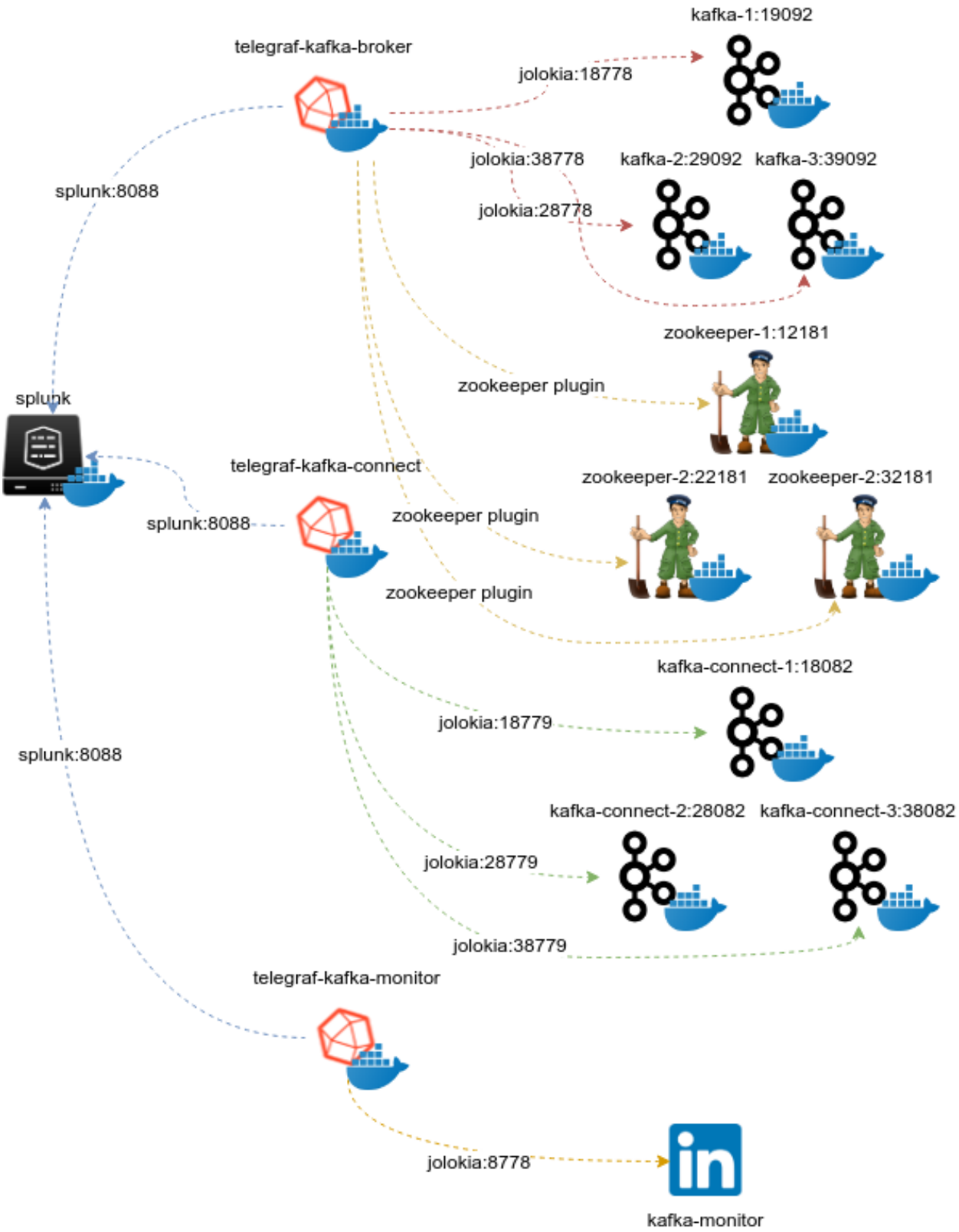
Docker compose templates are provided in the following repository:

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

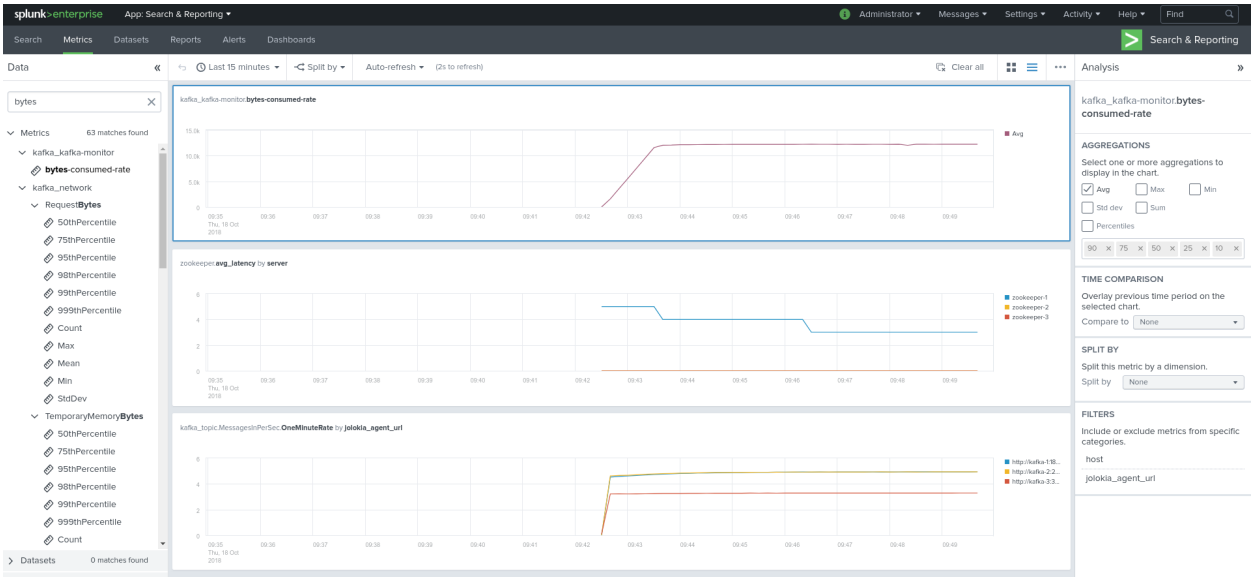
Using the docker templates allows you to create a full pre-configured Kafka environment with docker, just in 30 seconds.

Example:

- 3 x nodes Zookeeper cluster
- 3 x nodes Apache Kafka brokers cluster
- 3 x nodes Apache Kafka connect cluster
- 1 x node Confluent schema-registry
- 1 x Splunk standalone server running in docker
- 1 x LinkedIn Kafka monitor node
- 1 x Telegraf collector container to collect metrics from Zookeeper, Kafka brokers
- 1 x Telegraf collector container to collect metrics from Kafka Connect (including source and sink tasks)
- 1 x Telegraf collector container to collect metrics from LinkedIn Kafka monitor



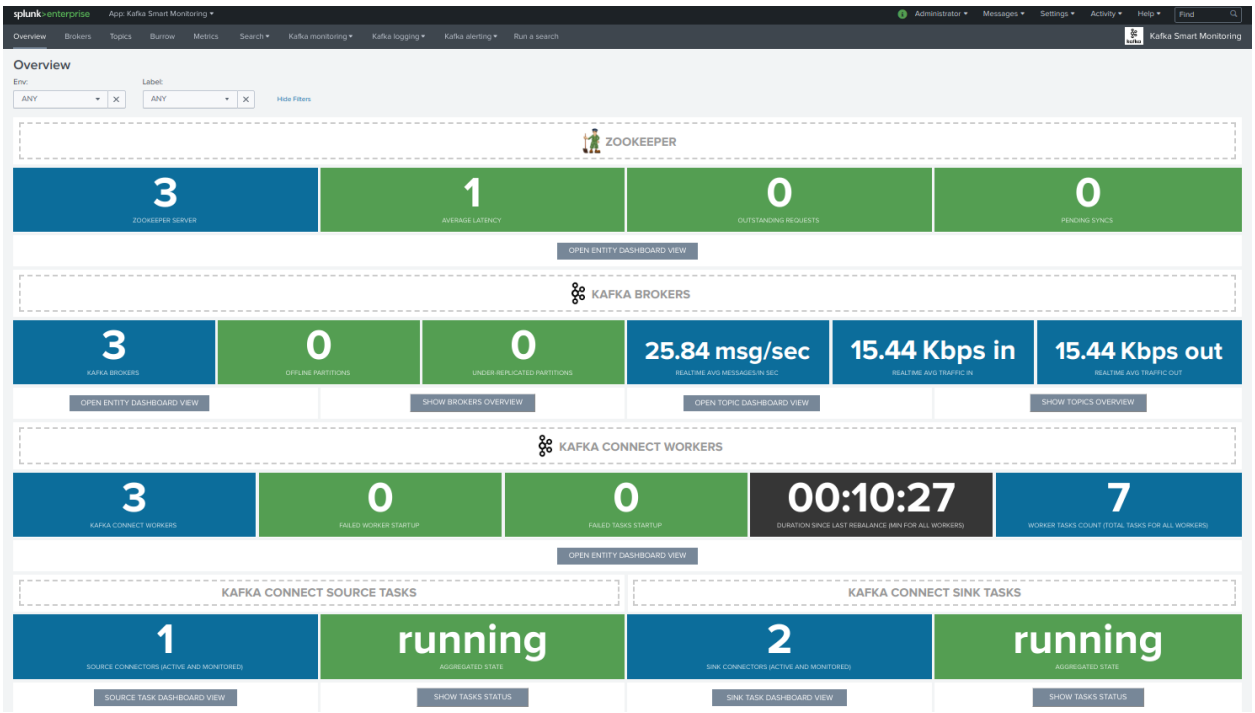
Start the template, have a very short coffee (approx. 30 sec), open Splunk, install the Metrics workspace app and observe the magic happening !



2.4 Splunk dashboards (health views)

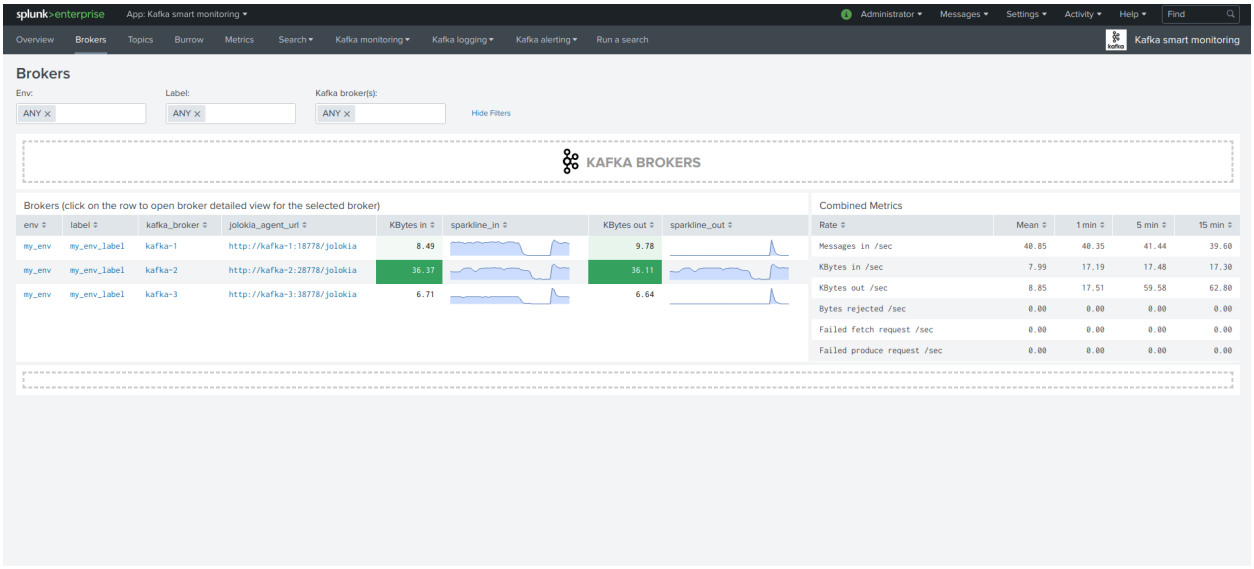
2.4.1 Overview (landing page)

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

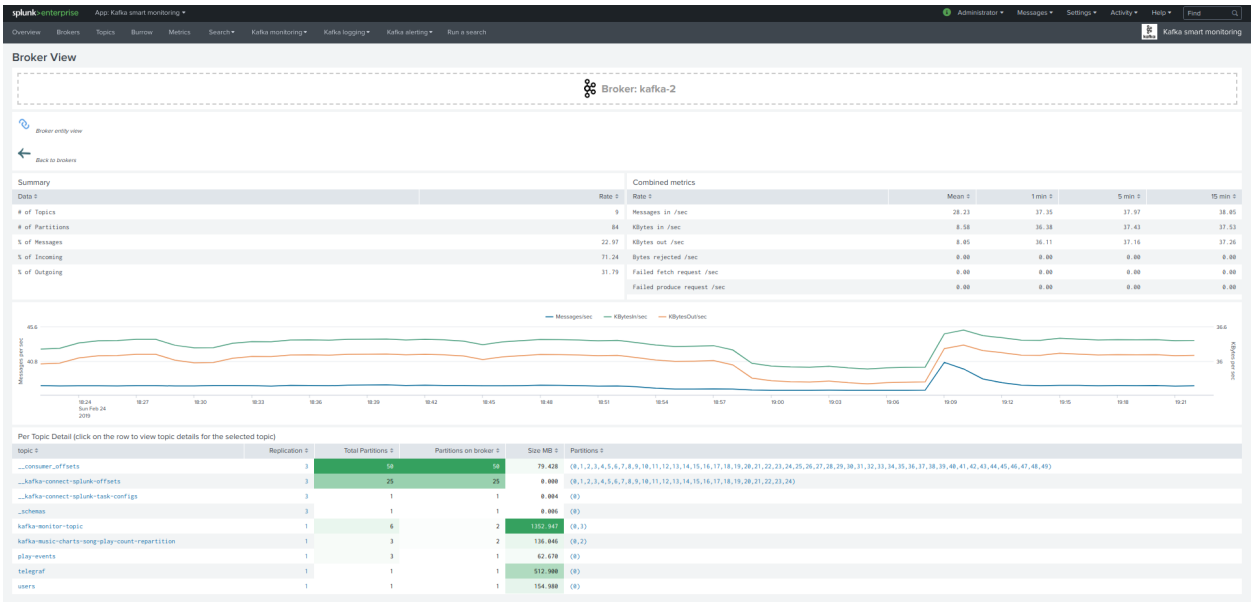


2.4.2 Overview Brokers

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

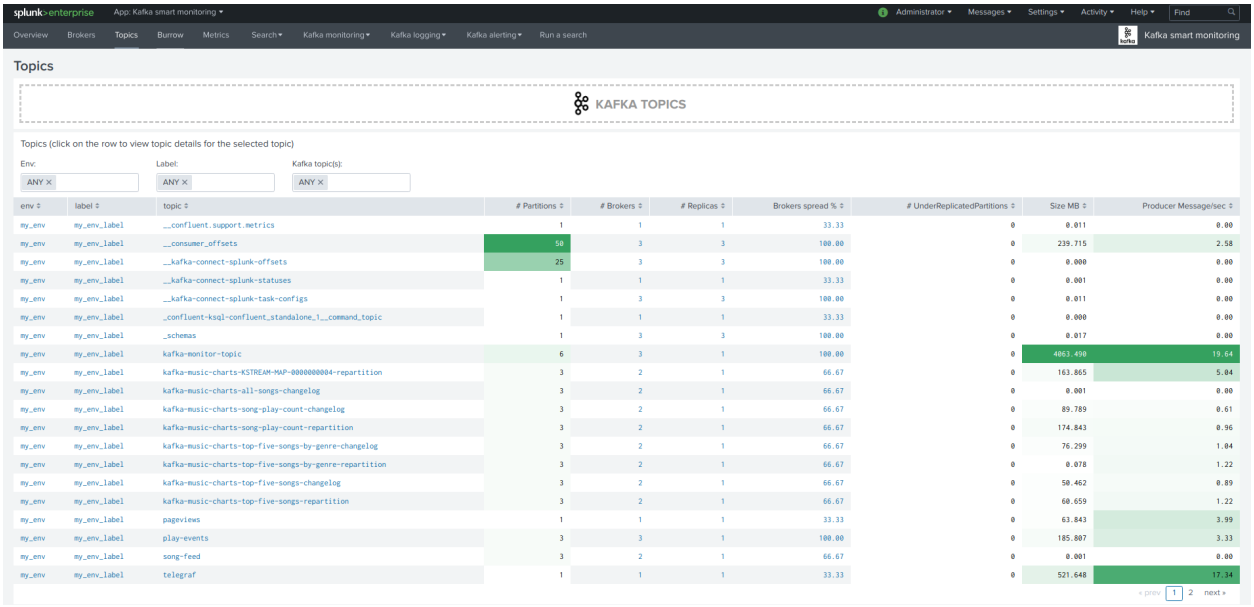


Broker drilldown page:

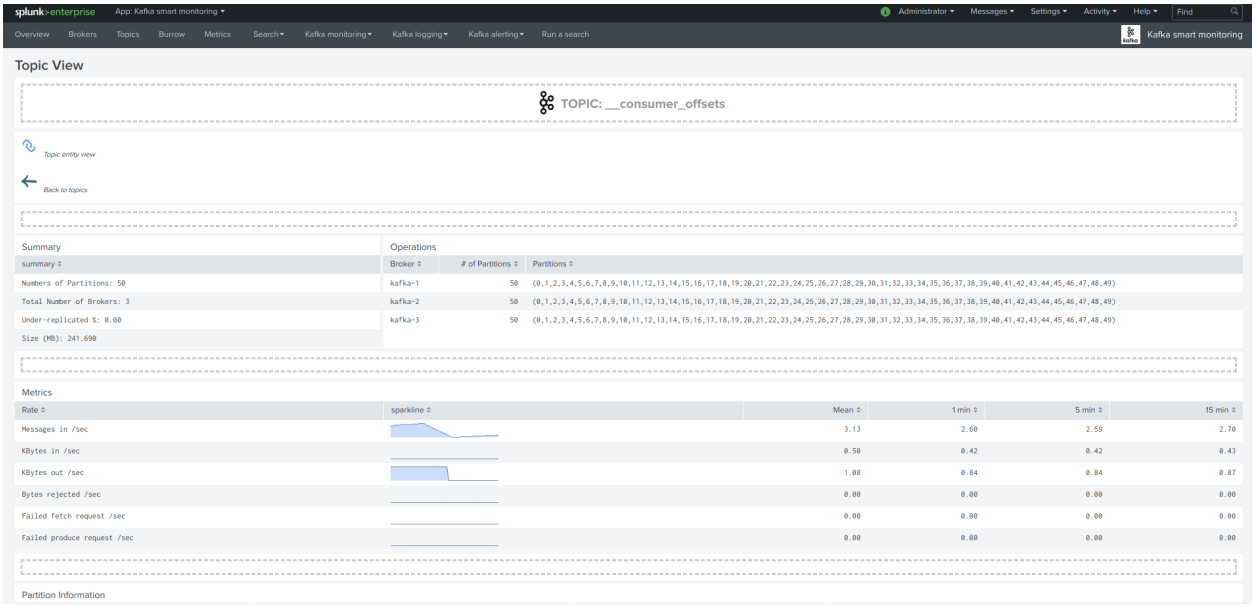


2.4.3 Overview Topics

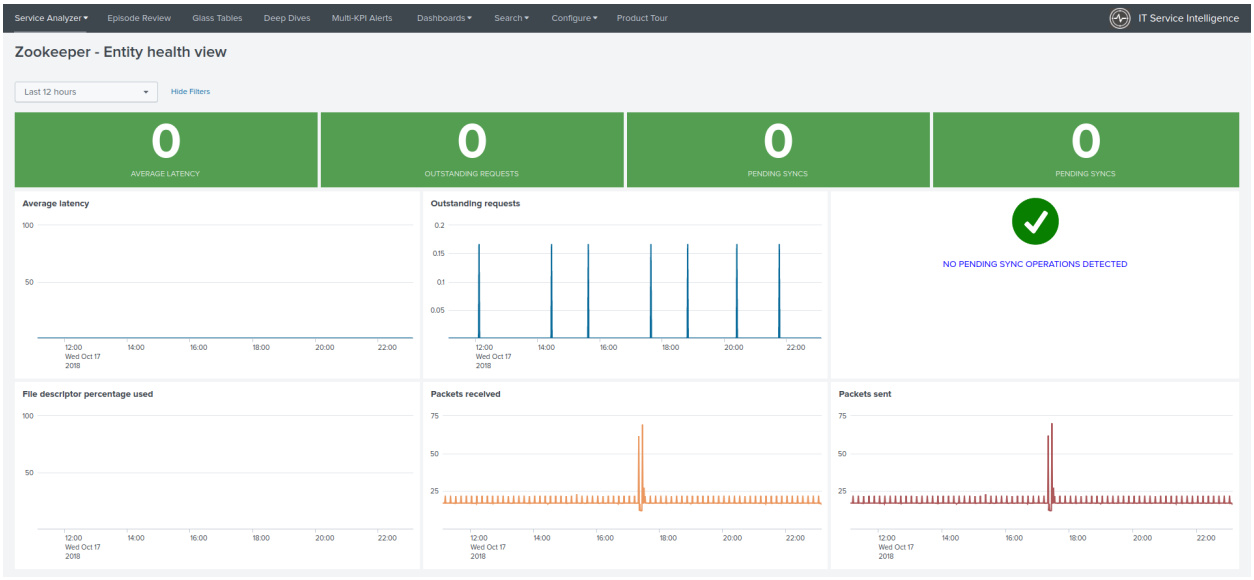
The Topics overview page inspired from Yahoo Kafka manager exposes the Kafka topics and main metrics:



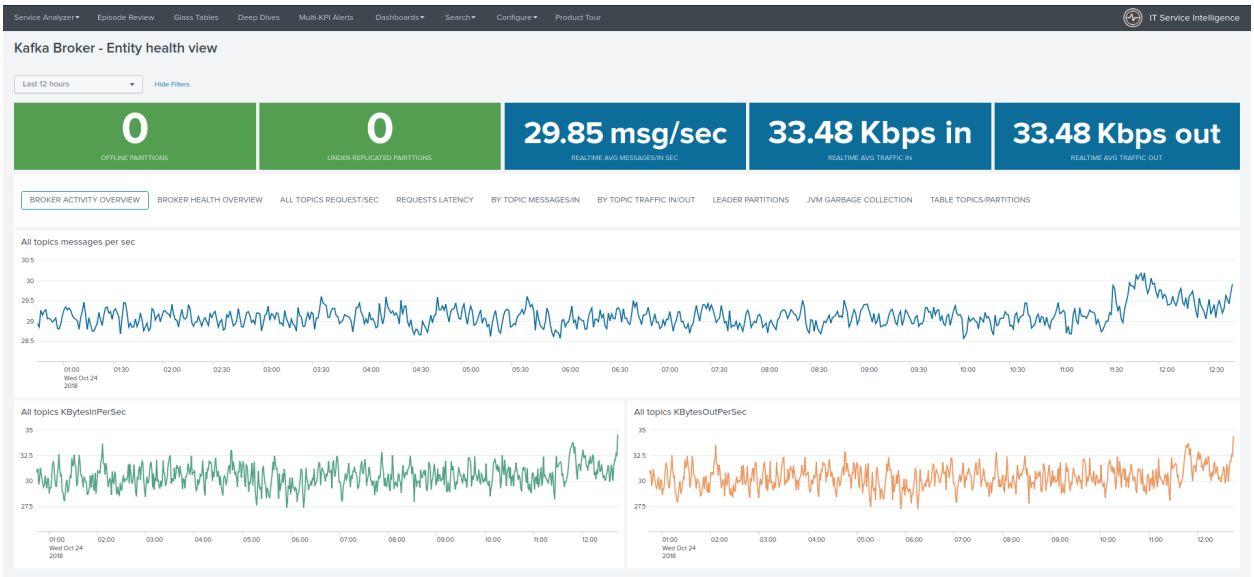
Topic drilldown page:

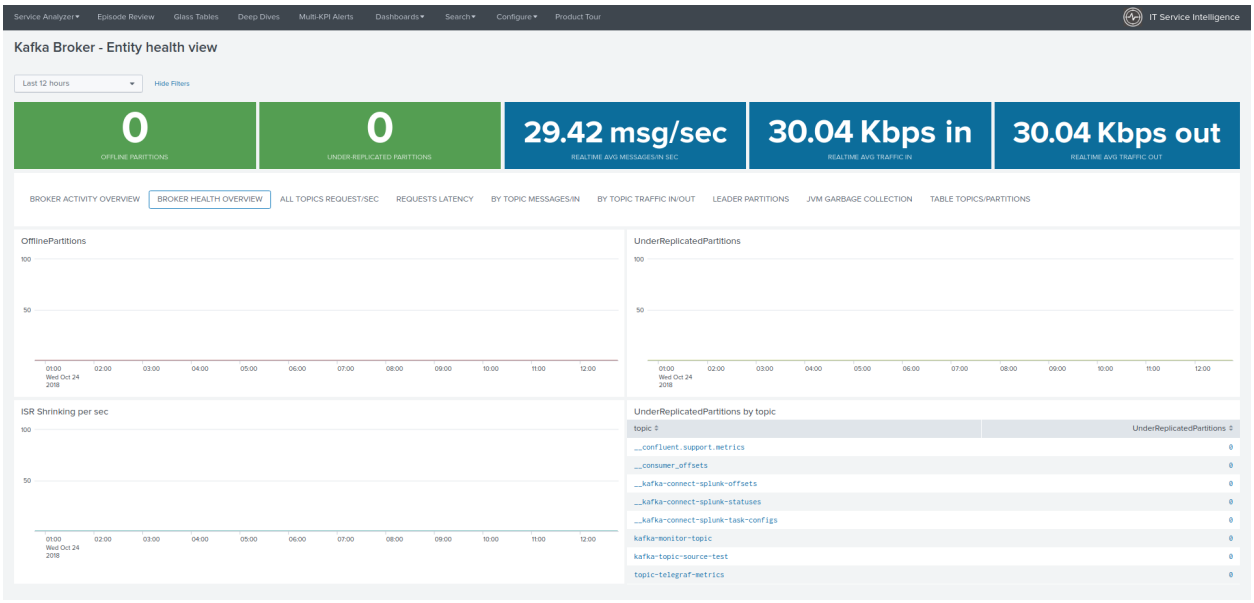


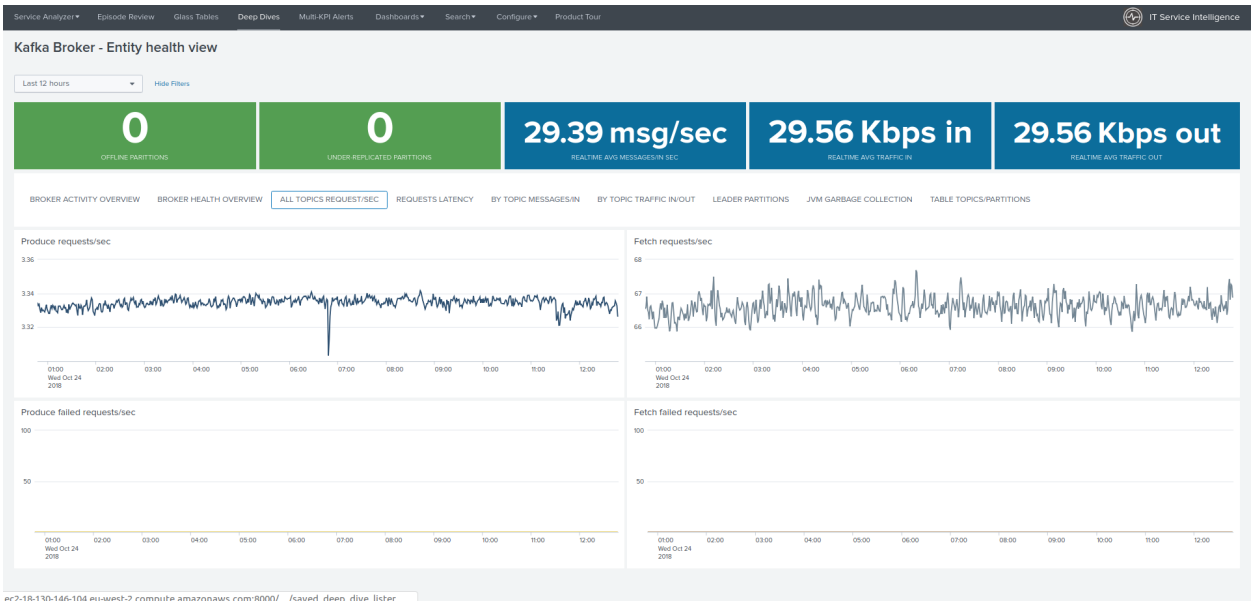
2.4.4 Zookeeper dashboard view

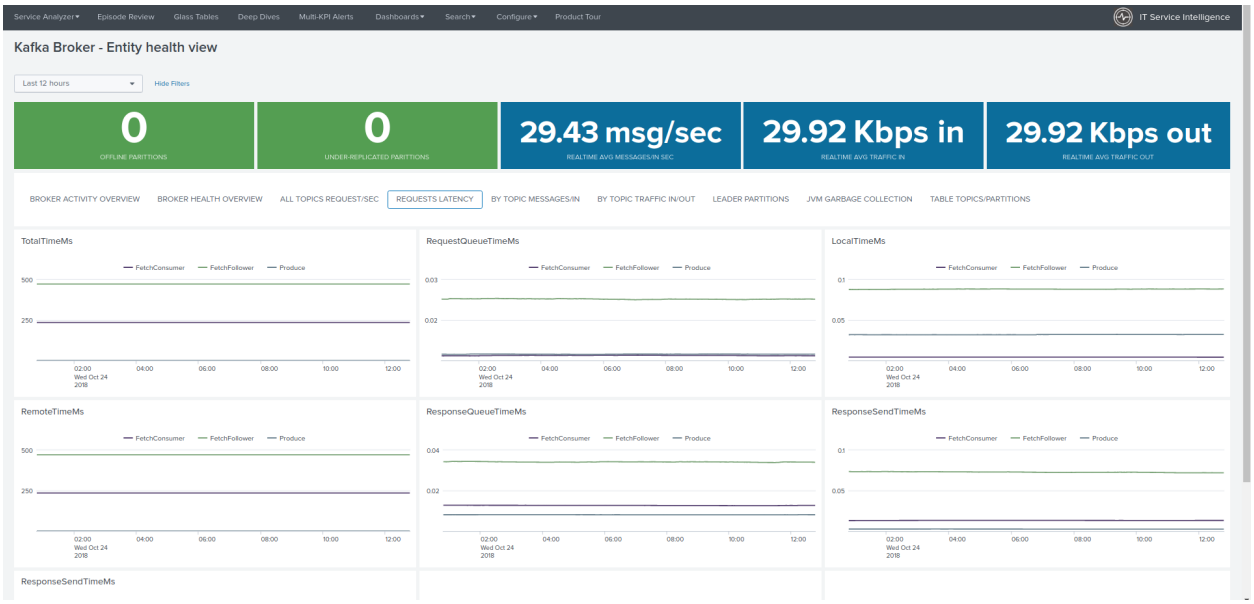


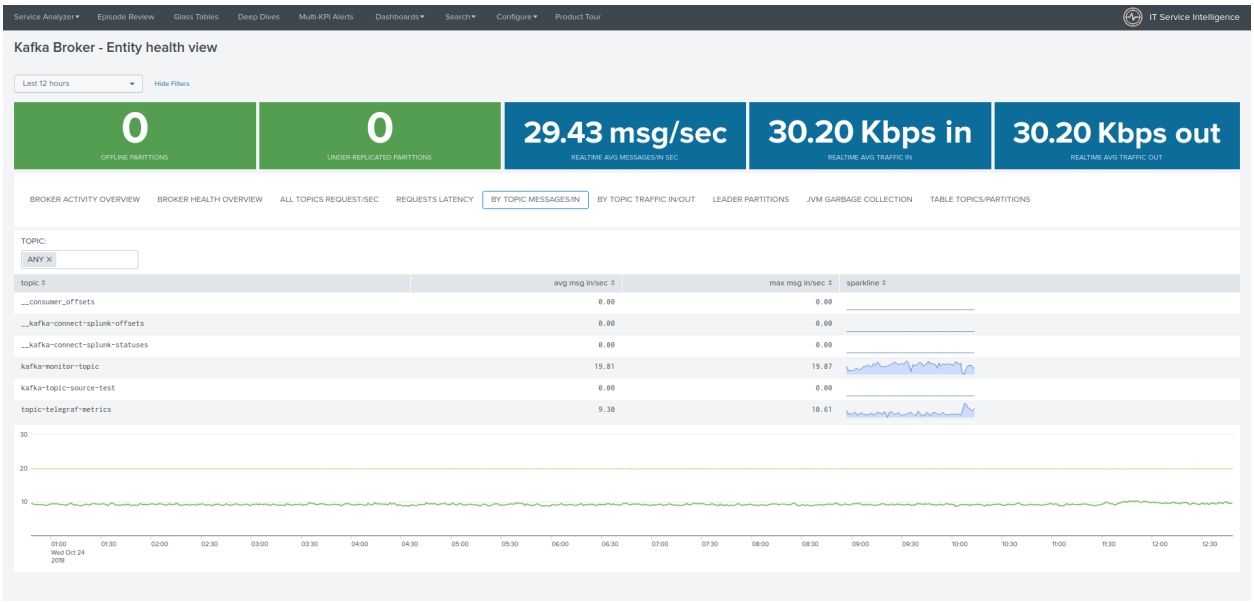
2.4.5 Kafka broker dashboard view

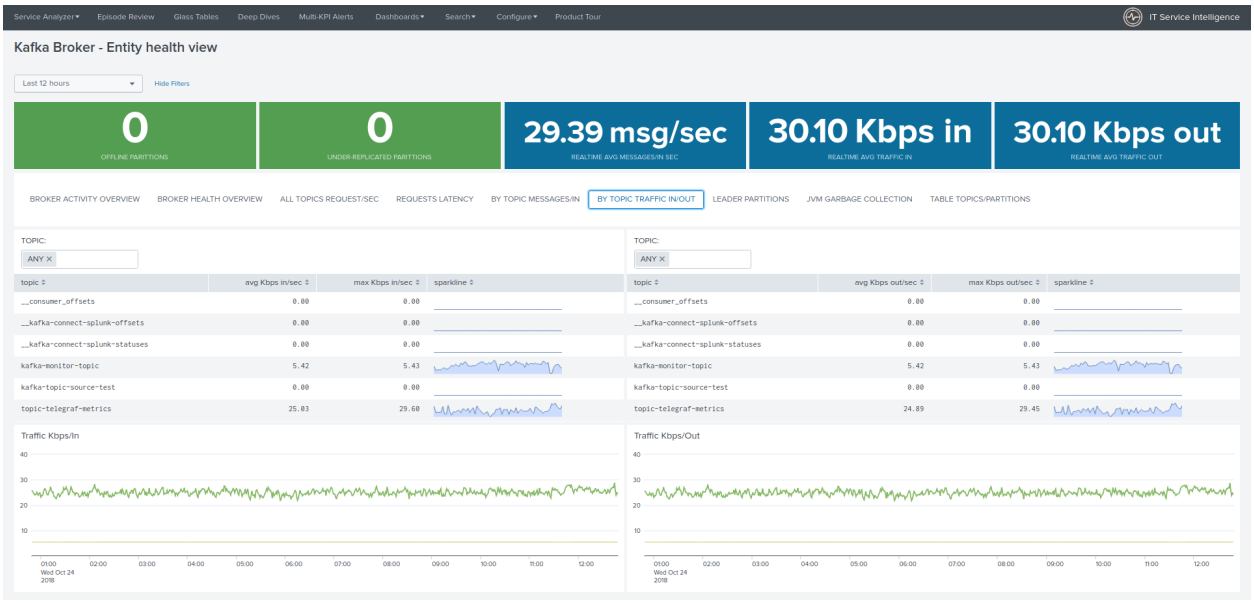


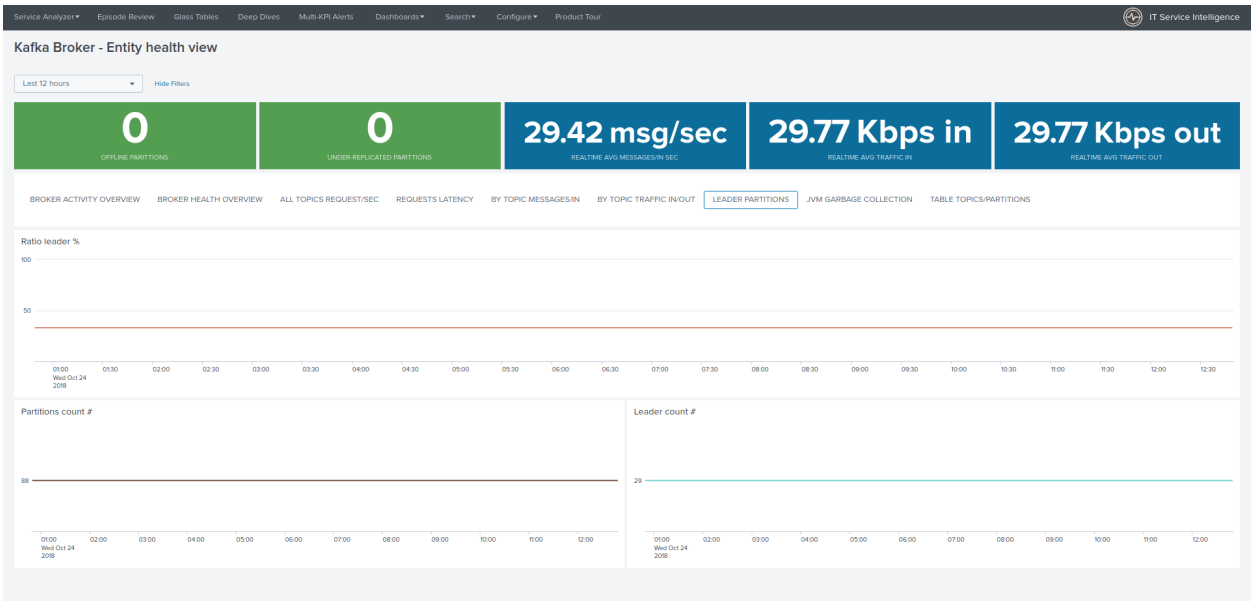

Service Analyzer
Episode Review
Glass Tables
Deep Dives
Multi-KPI Alerts
Dashboards
Search
Configure
Product Tour
IT Service Intelligence
Kafka Broker - Entity health view
Last 12 hours
Hide Filters
0
OFFLINE PARITIONS
0
UNDER-REPLICATED PARITIONS
29.42 msg/sec
REALTIME AVG MESSAGES/IN SEC
30.04 Kbps in
REALTIME AVG TRAFFIC IN
30.04 Kbps out
REALTIME AVG TRAFFIC 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
OfflinePartitions
UnderReplicatedPartitions
ISR Shrinking per sec
UnderReplicatedPartitions by topic
topic
UnderReplicatedPartitions
__confluent.support.metrics
__consumer_offsets
__kafka-connect-splunk-offsets
__kafka-connect-splunk-statuses
__kafka-connect-splunk-task-configs
kafka-monitor-topic
kafka-topic-source-test
topic-telegraf-metrics

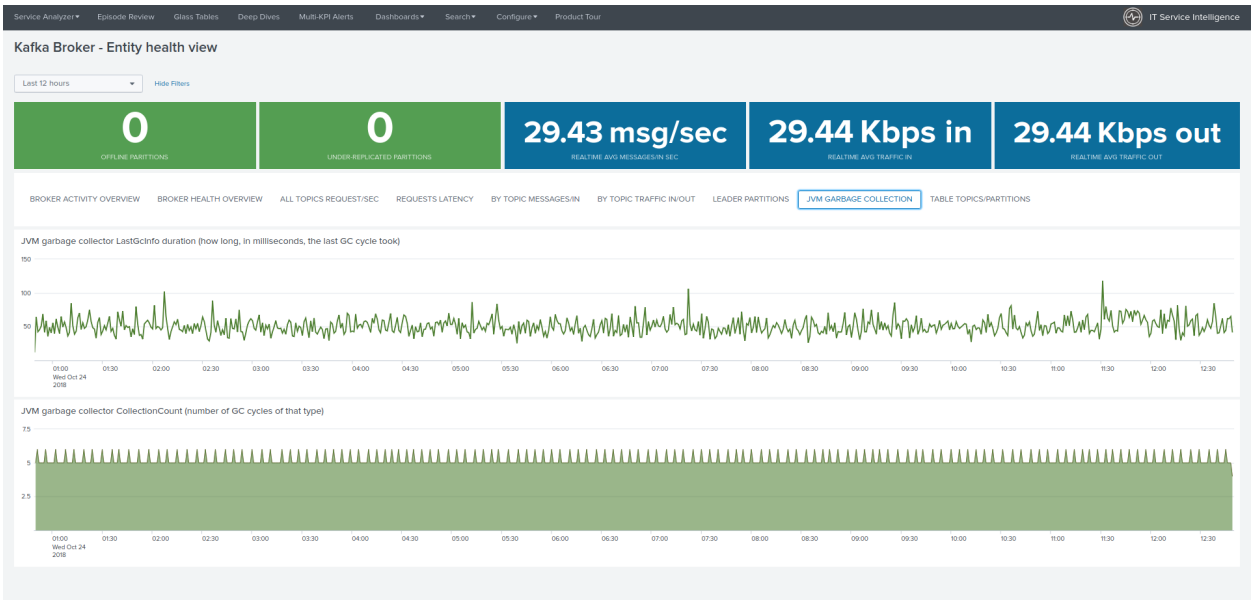

Service Analyzer
Episode Review
Glass Tables
Deep Dives
Multi-KPI Alerts
Dashboards
Search
Configure
Product Tour
IT Service Intelligence
Kafka Broker - Entity health view
Last 12 hours
Hide Filters
0
OFFLINE PARITIONS
0
UNDER-REPLICATED PARITIONS
29.39 msg/sec
REALTIME AVG MESSAGES/IN SEC
29.56 Kbps in
REALTIME AVG TRAFFIC IN
29.56 Kbps out
REALTIME AVG TRAFFIC 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
Produce requests/sec
Fetch requests/sec
Produce failed requests/sec
Fetch failed requests/sec

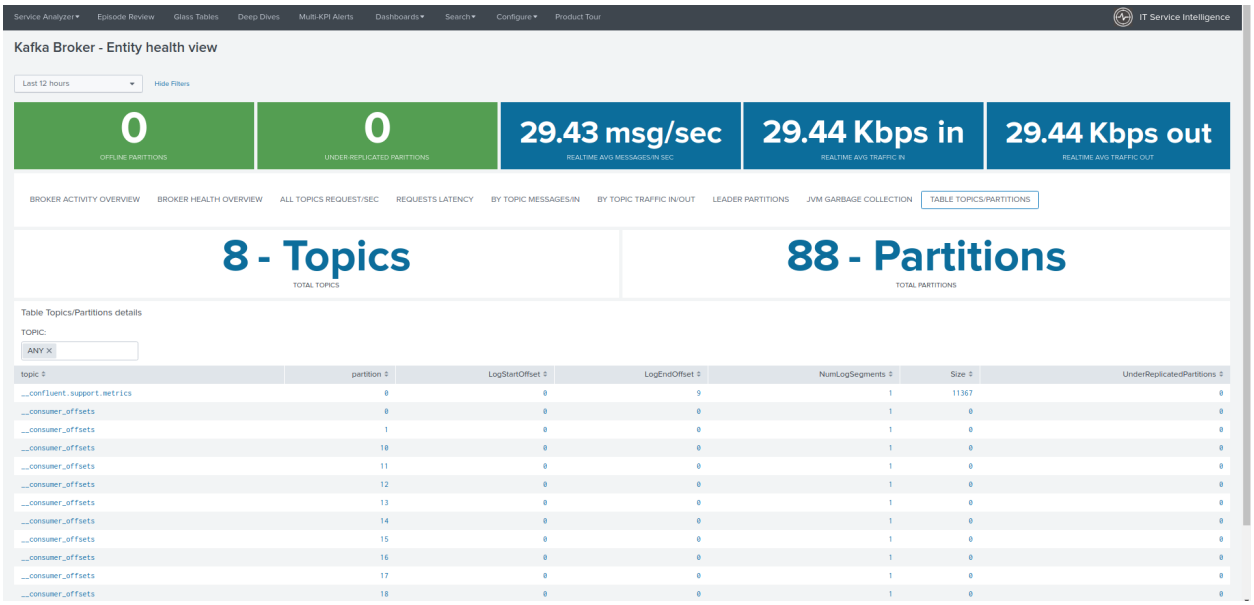

Service Analyzer
Episode Review
Glass Tables
Deep Dives
Multi-KPI Alerts
Dashboards
Search
Configure
Product Tour
IT Service Intelligence
Kafka Broker - Entity health view
Last 12 hours
Hide Filters
0
OFFLINE PARITTIONS
0
UNDER-REPLICATED PARITTIONS
29.43 msg/sec
REALTIME AVG MESSAGES/IN SEC
29.92 Kbps in
REALTIME AVG TRAFFIC IN
29.92 Kbps out
REALTIME AVG TRAFFIC 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
TotalTimeMs
RequestQueueTimeMs
LocalTimeMs
RemoteTimeMs
ResponseQueueTimeMs
ResponseSendTimeMs
FetchConsumer
FetchFollower
Produce


Service Analyzer
Episode Review
Glass Tables
Deep Dives
Multi-KPI Alerts
Dashboards
Search
Configure
Product Tour
IT Service Intelligence
Kafka Broker - Entity health view
Last 12 hours
Hide Filters
0
OFFLINE PARITTIONS
0
UNDER-REPLICATED PARITTIONS
29.43 msg/sec
REALTIME AVG MESSAGES/IN SEC
30.20 Kbps in
REALTIME AVG TRAFFIC IN
30.20 Kbps out
REALTIME AVG TRAFFIC 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
TOPIC:
ANY
topic
avg msg in/sec
max msg in/sec
sparkline
__consumer_offsets 0.00 0.00
__kafka-connect-splunk-offsets 0.00 0.00
__kafka-connect-splunk-statuses 0.00 0.00
kafka-monitor-topic 19.81 19.87
kafka-topic-source-test 0.00 0.00
topic-telegraf-metrics 9.30 10.61

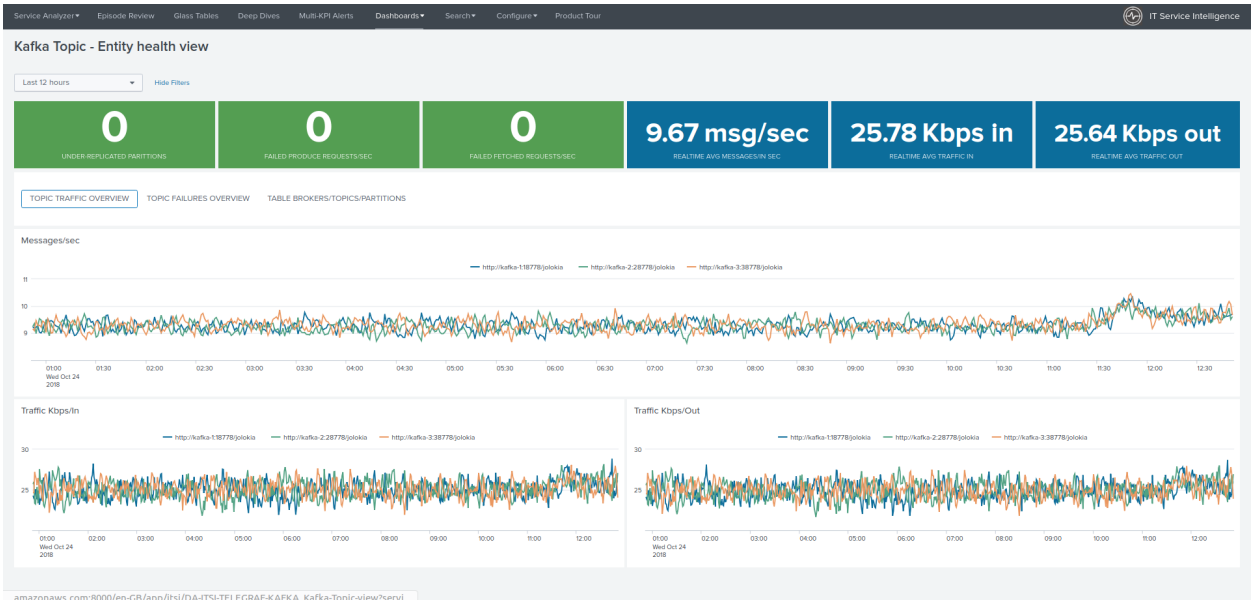

Service Analyzer
Episode Review
Glass Tables
Deep Dives
Multi-KPI Alerts
Dashboards
Search
Configure
Product Tour
IT Service Intelligence
Kafka Broker - Entity health view
Last 12 hours
Hide Filters
0
OFFLINE PARITIONS
0
UNDER-REPLICATED PARITIONS
29.39 msg/sec
REALTIME AVG MESSAGES/IN SEC
30.10 Kbps in
REALTIME AVG TRAFFIC IN
30.10 Kbps out
REALTIME AVG TRAFFIC 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
TOPIC:
ANY
topic
avg Kbps in/sec
max Kbps in/sec
sparkline
__consumer_offsets 0.00 0.00
__kafka-connect-splunk-offsets 0.00 0.00
__kafka-connect-splunk-statuses 0.00 0.00
kafka-monitor-topic 5.42 5.43
kafka-topic-source-test 0.00 0.00
topic-telegraf-metrics 25.03 29.60
TOPIC:
ANY
topic
avg Kbps out/sec
max Kbps out/sec
sparkline
__consumer_offsets 0.00 0.00
__kafka-connect-splunk-offsets 0.00 0.00
__kafka-connect-splunk-statuses 0.00 0.00
kafka-monitor-topic 5.42 5.43
kafka-topic-source-test 0.00 0.00
topic-telegraf-metrics 24.89 29.45
Traffic Kbps/In
Traffic Kbps/Out

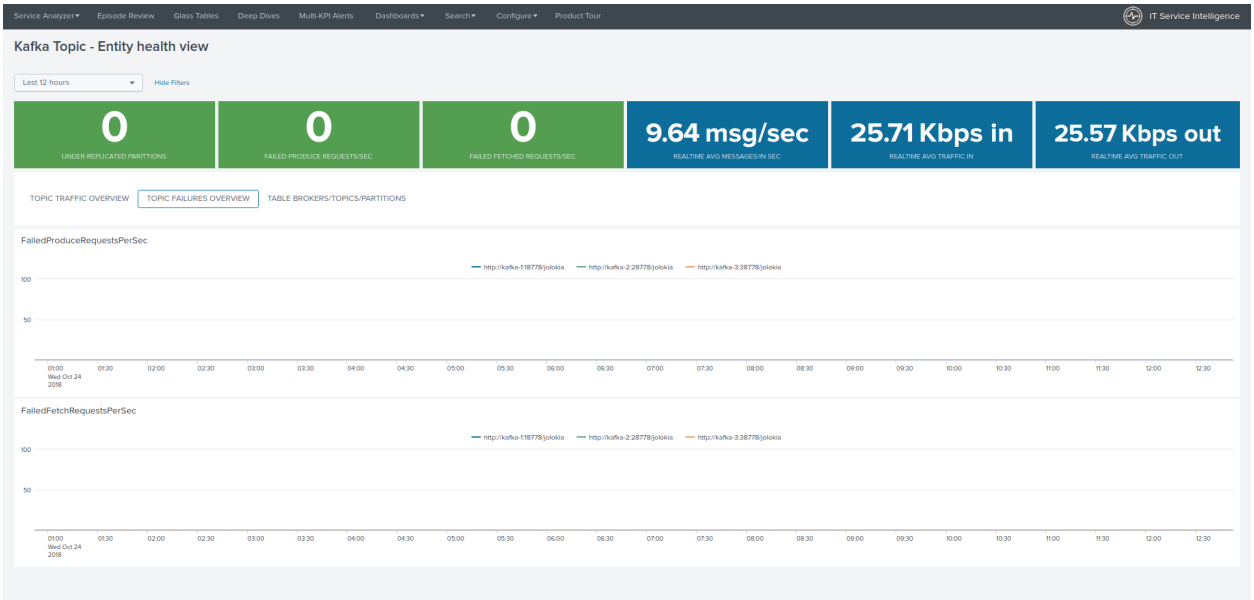

Service Analyzer
Episode Review
Glass Tables
Deep Dives
Multi-KPI Alerts
Dashboards
Search
Configure
Product Tour
IT Service Intelligence
Kafka Broker - Entity health view
Last 12 hours
Hide Filters
0
OFFLINE PARITIONS
0
UNDER-REPLICATED PARITIONS
29.42 msg/sec
REALTIME AVG MESSAGES/IN SEC
29.77 Kbps in
REALTIME AVG TRAFFIC IN
29.77 Kbps out
REALTIME AVG TRAFFIC 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
Ratio leader %
Partitions count #
Leader count #

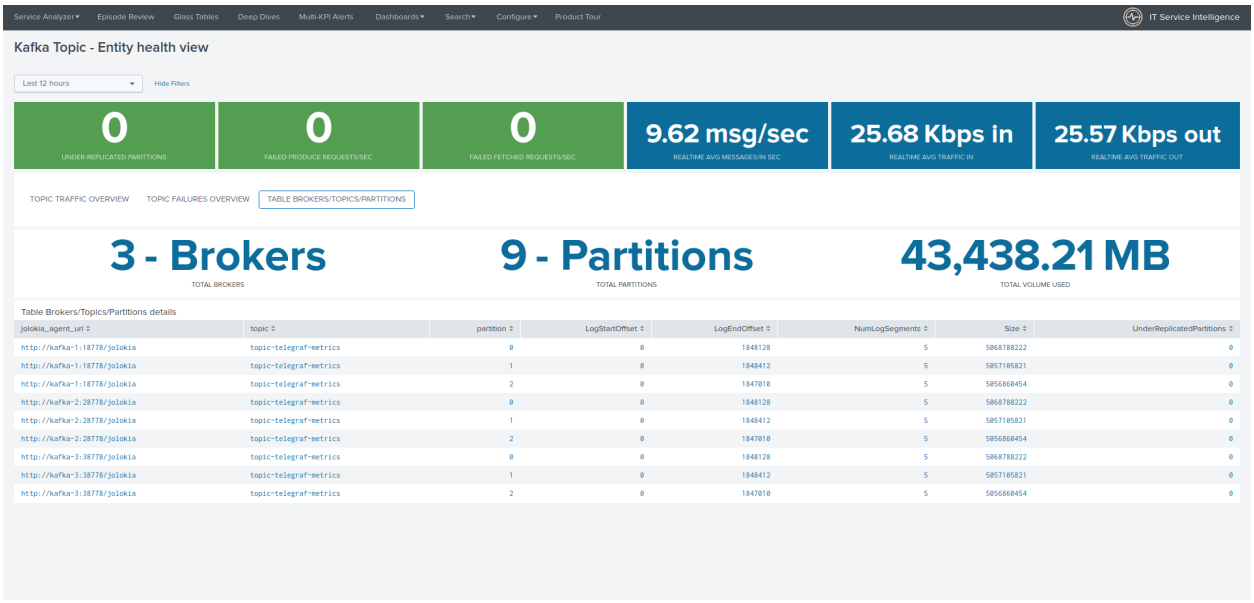

Service Analyzer
Episode Review
Glass Tables
Deep Dives
Multi-KPI Alerts
Dashboards
Search
Configure
Product Tour
IT Service Intelligence
Kafka Broker - Entity health view
Last 12 hours
Hide Filters
0
OFFLINE PARITTIONS
0
UNDER-REPLICATED PARITTIONS
29.43 msg/sec
REALTIME AVG MESSAGES/IN SEC
29.44 Kbps in
REALTIME AVG TRAFFIC IN
29.44 Kbps out
REALTIME AVG TRAFFIC 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
JVM garbage collector LastGcInfo duration (how long, in milliseconds, the last GC cycle took)
JVM garbage collector CollectionCount (number of GC cycles of that type)


Service Analyzer
Episode Review
Glass Tables
Deep Dives
Multi-KPI Alerts
Dashboards
Search
Configure
Product Tour
IT Service Intelligence
Kafka Broker - Entity health view
Last 12 hours
Hide Filters
0
OFFLINE PARITTIONS
0
UNDER-REPLICATED PARITTIONS
29.43 msg/sec
REALTIME AVG MESSAGES/IN SEC
29.44 Kbps in
REALTIME AVG TRAFFIC IN
29.44 Kbps out
REALTIME AVG TRAFFIC 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
8 - Topics
TOTAL TOPICS
88 - Partitions
TOTAL PARTITIONS
Table Topics/Partitions details
TOPIC:
ANY
topic
partition
LogStartOffset
LogEndOffset
NumLogSegments
Size
UnderReplicatedPartitions
__confluent.support.metrics 0 0 9 1 11367 0
__consumer_offsets 0 0 0 1 0 0
__consumer_offsets 1 0 0 1 0 0
__consumer_offsets 10 0 0 1 0 0
__consumer_offsets 11 0 0 1 0 0
__consumer_offsets 12 0 0 1 0 0
__consumer_offsets 13 0 0 1 0 0
__consumer_offsets 14 0 0 1 0 0
__consumer_offsets 15 0 0 1 0 0
__consumer_offsets 16 0 0 1 0 0
__consumer_offsets 17 0 0 1 0 0
__consumer_offsets 18 0 0 1 0 0

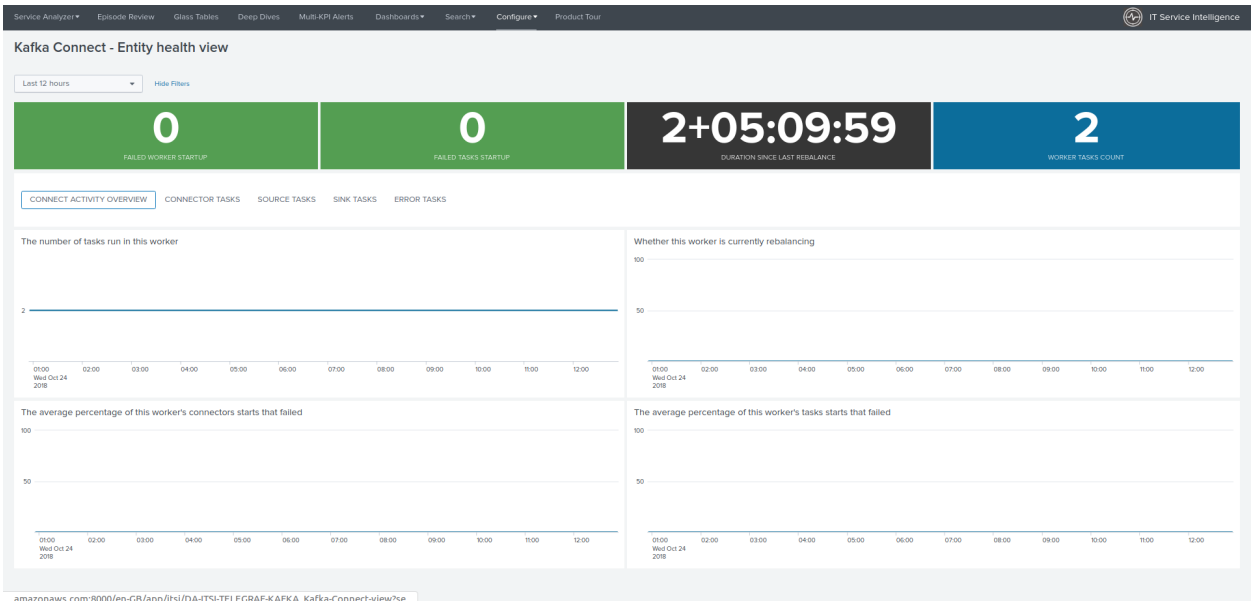
2.4.6 Kafka topic dashboard view

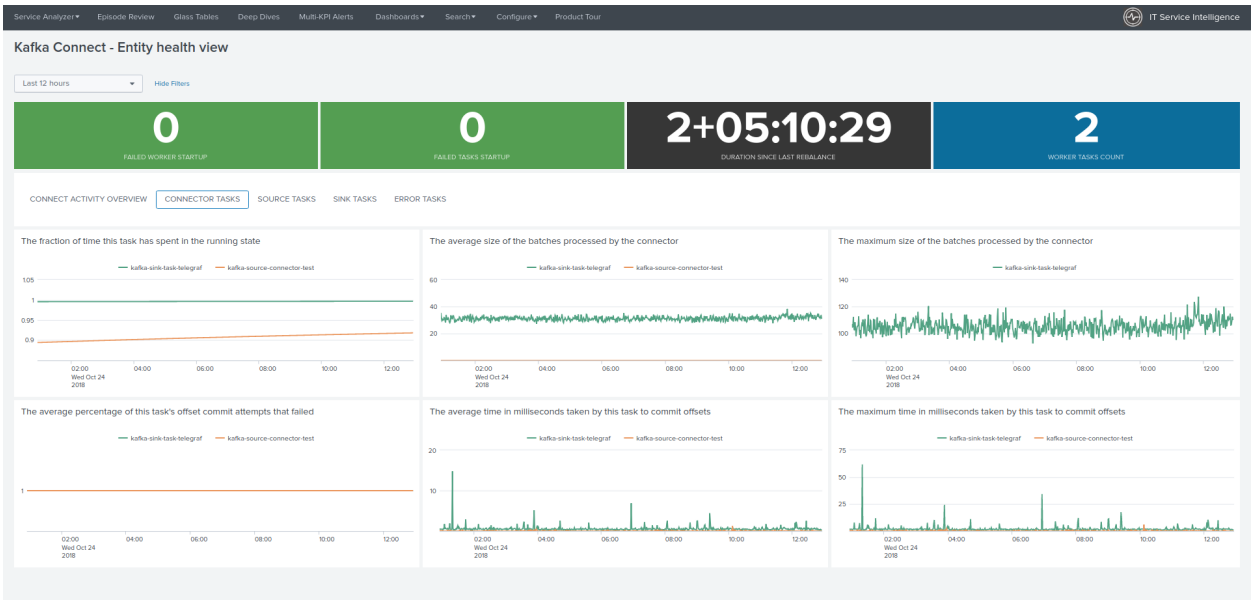


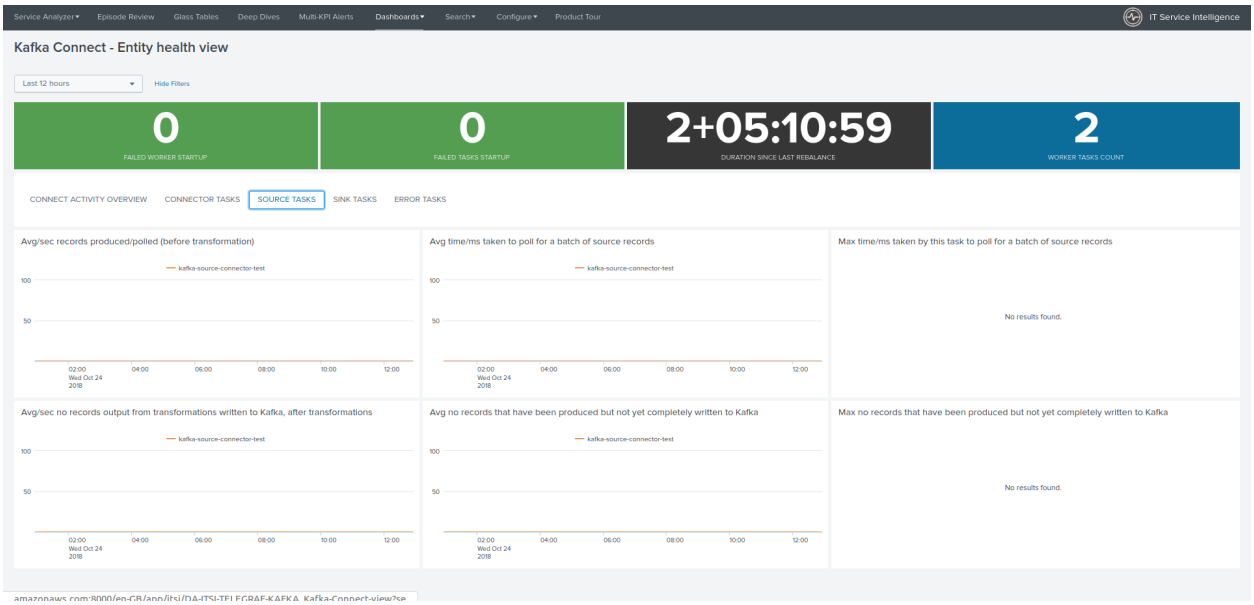


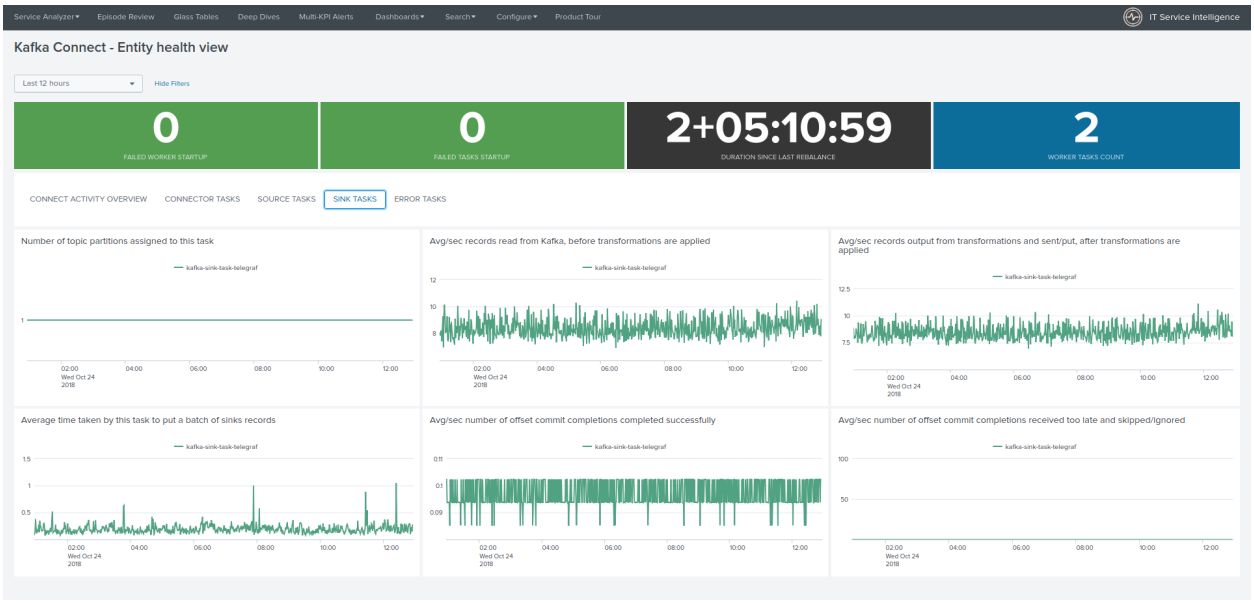


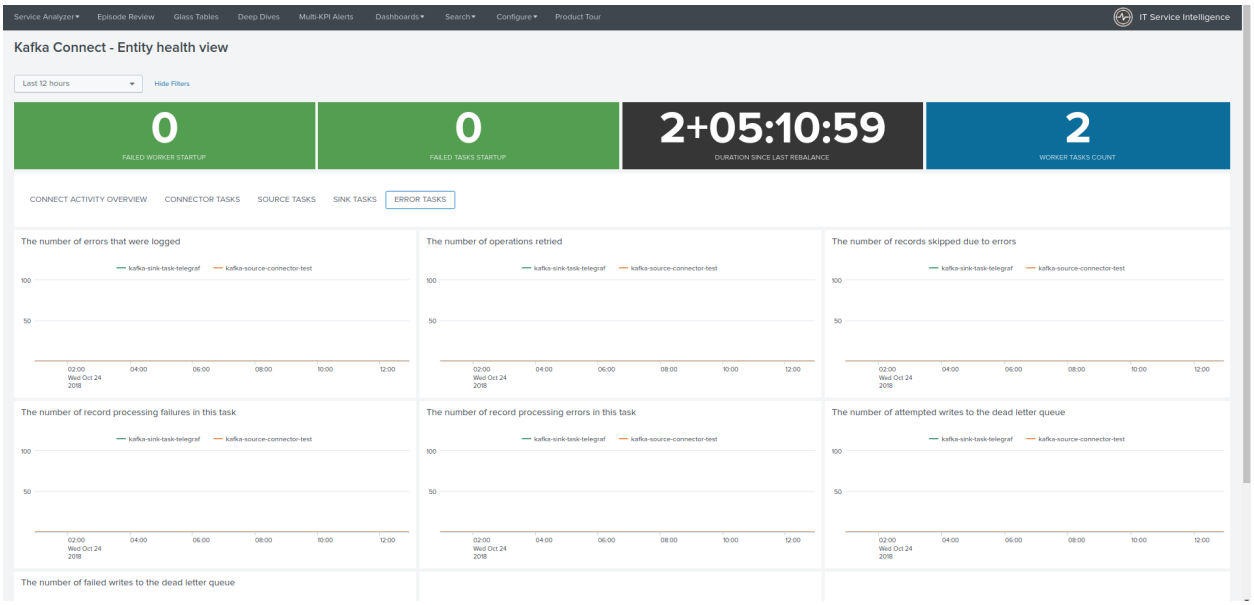
2.4.7 Kafka connect dashboard view



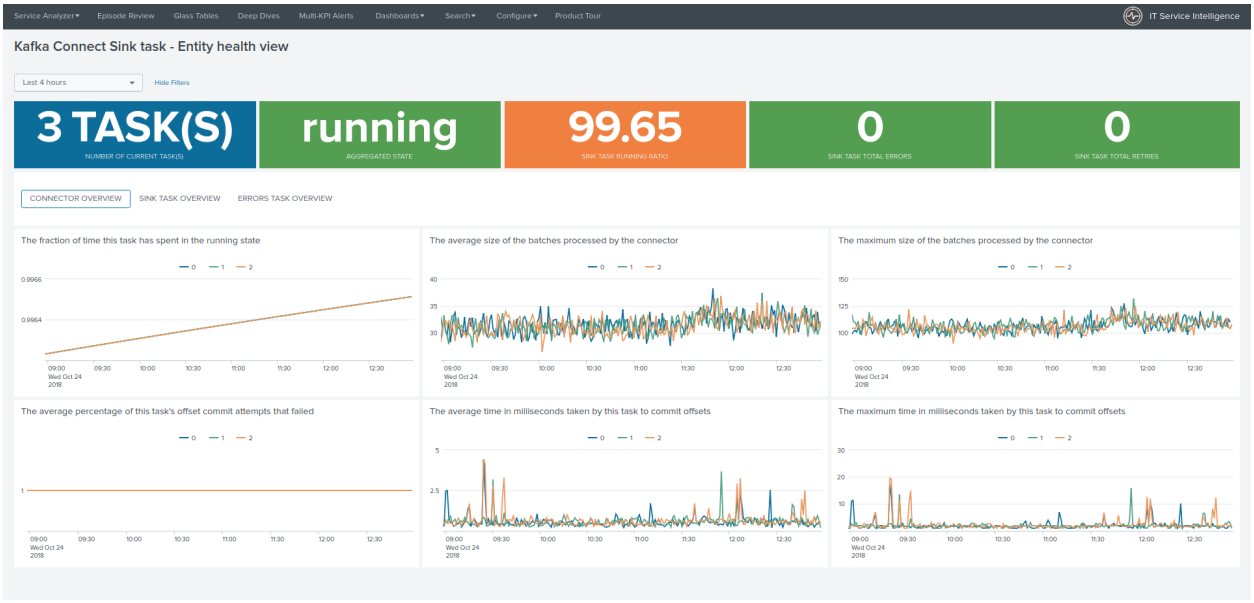

Service Analyzer
Episode Review
Glass Tables
Deep Dives
Multi-KPI Alerts
Dashboards
Search
Configure
Product Tour
IT Service Intelligence
Kafka Connect - Entity health view
Last 12 hours
Hide Filters
0
FAILED WORKER STARTUP
0
FAILED TASKS STARTUP
2+05:10:29
DURATION SINCE LAST REBALANCE
2
WORKER TASKS COUNT
CONNECT ACTIVITY OVERVIEW
CONNECTOR TASKS
SOURCE TASKS
SINK TASKS
ERROR TASKS
The fraction of time this task has spent in the running state
The average size of the batches processed by the connector
The maximum size of the batches processed by the connector
The average percentage of this task's offset commit attempts that failed
The average time in milliseconds taken by this task to commit offsets
The maximum time in milliseconds taken by this task to commit offsets
kafka-sink-task-telegraf
kafka-source-connector-test

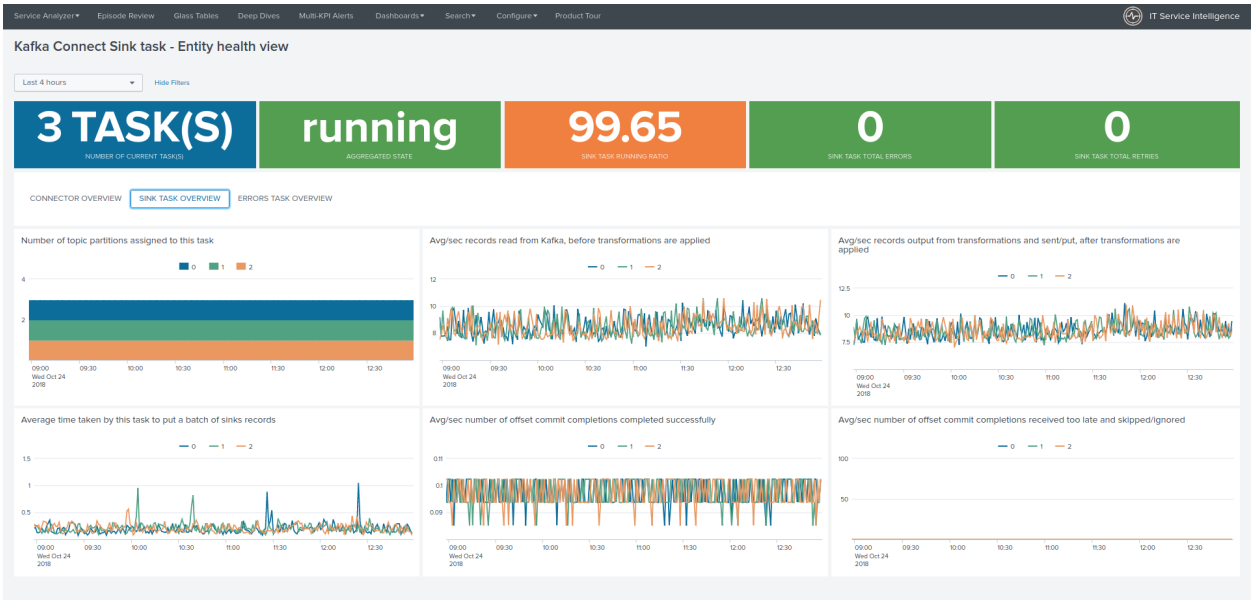

Service Analyzer
Episode Review
Glass Tables
Deep Dives
Multi-KPI Alerts
Dashboards
Search
Configure
Product Tour
IT Service Intelligence
Kafka Connect - Entity health view
Last 12 hours
Hide Filters
0
FAILED WORKER STARTUP
0
FAILED TASKS STARTUP
2+05:10:59
DURATION SINCE LAST REBALANCE
2
WORKER TASKS COUNT
CONNECT ACTIVITY OVERVIEW
CONNECTOR TASKS
SOURCE TASKS
SINK TASKS
ERROR TASKS
Avg/sec records produced/polled (before transformation)
Avg time/ms taken to poll for a batch of source records
Max time/ms taken by this task to poll for a batch of source records
No results found.
Avg/sec no records output from transformations written to Kafka, after transformations
Avg no records that have been produced but not yet completely written to Kafka
Max no records that have been produced but not yet completely written to Kafka
No results found.
kafka-source-connector-test

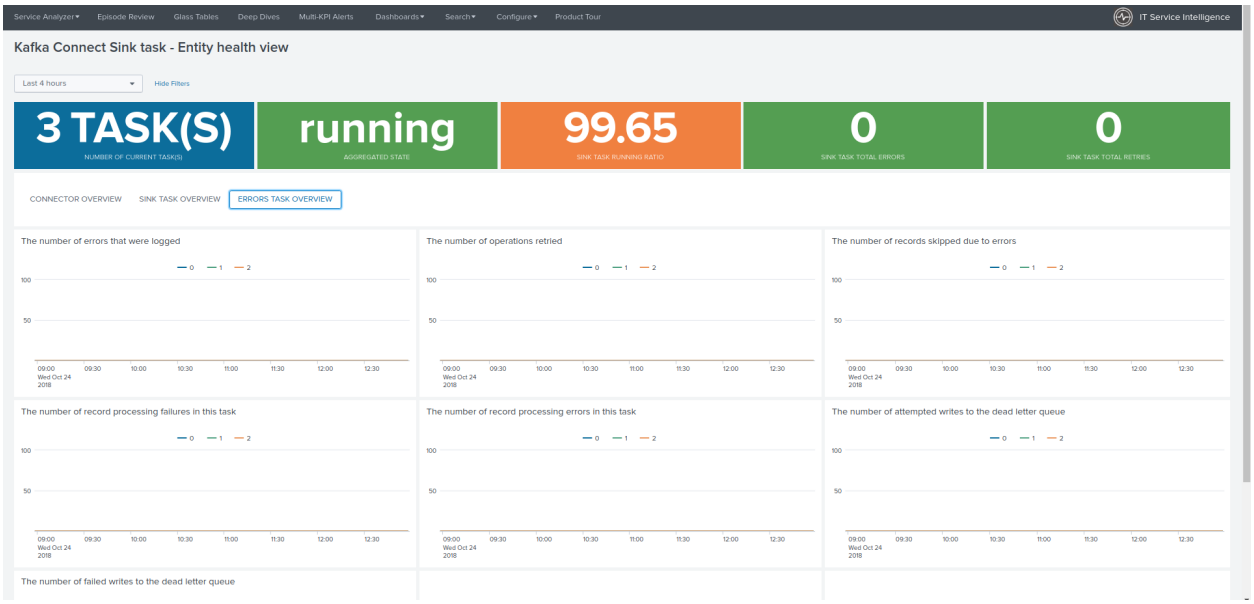

Service Analyzer
Episode Review
Glass Tables
Deep Dives
Multi-KPI Alerts
Dashboards
Search
Configure
Product Tour
IT Service Intelligence
Kafka Connect - Entity health view
Last 12 hours
Hide Filters
0
FAILED WORKER STARTUP
0
FAILED TASKS STARTUP
2+05:10:59
DURATION SINCE LAST REBALANCE
2
WORKER TASKS COUNT
CONNECT ACTIVITY OVERVIEW
CONNECTOR TASKS
SOURCE TASKS
SINK TASKS
ERROR TASKS
Number of topic partitions assigned to this task
Avg/sec records read from Kafka, before transformations are applied
Avg/sec records output from transformations and sent/put, after transformations are applied
Average time taken by this task to put a batch of sinks records
Avg/sec number of offset commit completions completed successfully
Avg/sec number of offset commit completions received too late and skipped/ignored
kafka-sink-task-telegraf


Service Analyzer
Episode Review
Glass Tables
Deep Dives
Multi-KPI Alerts
Dashboards
Search
Configure
Product Tour
IT Service Intelligence
Kafka Connect - Entity health view
Last 12 hours
Hide Filters
0
FAILED WORKER STARTUP
0
FAILED TASKS STARTUP
2+05:10:59
DURATION SINCE LAST REBALANCE
2
WORKER TASKS COUNT
CONNECT ACTIVITY OVERVIEW
CONNECTOR TASKS
SOURCE TASKS
SINK TASKS
ERROR TASKS
The number of errors that were logged
The number of operations retried
The number of records skipped due to errors
The number of record processing failures in this task
The number of record processing errors in this task
The number of attempted writes to the dead letter queue
The number of failed writes to the dead letter queue
kafka-sink-task-telegraf
kafka-source-connector-test

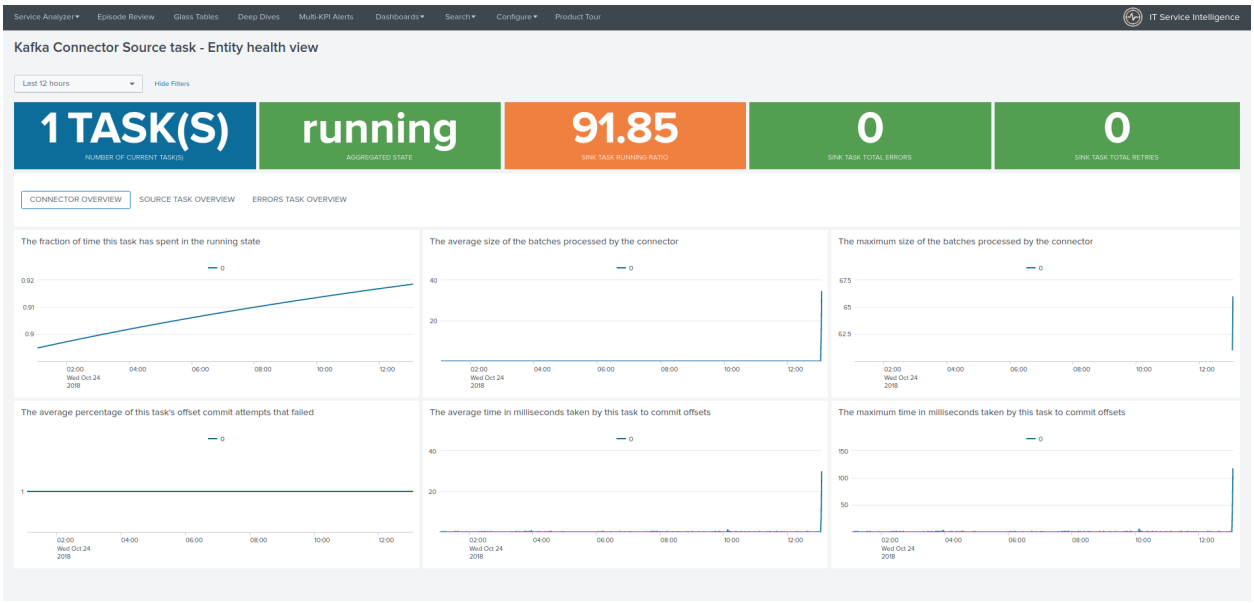
2.4.8 Kafka connect sink task dashboard view

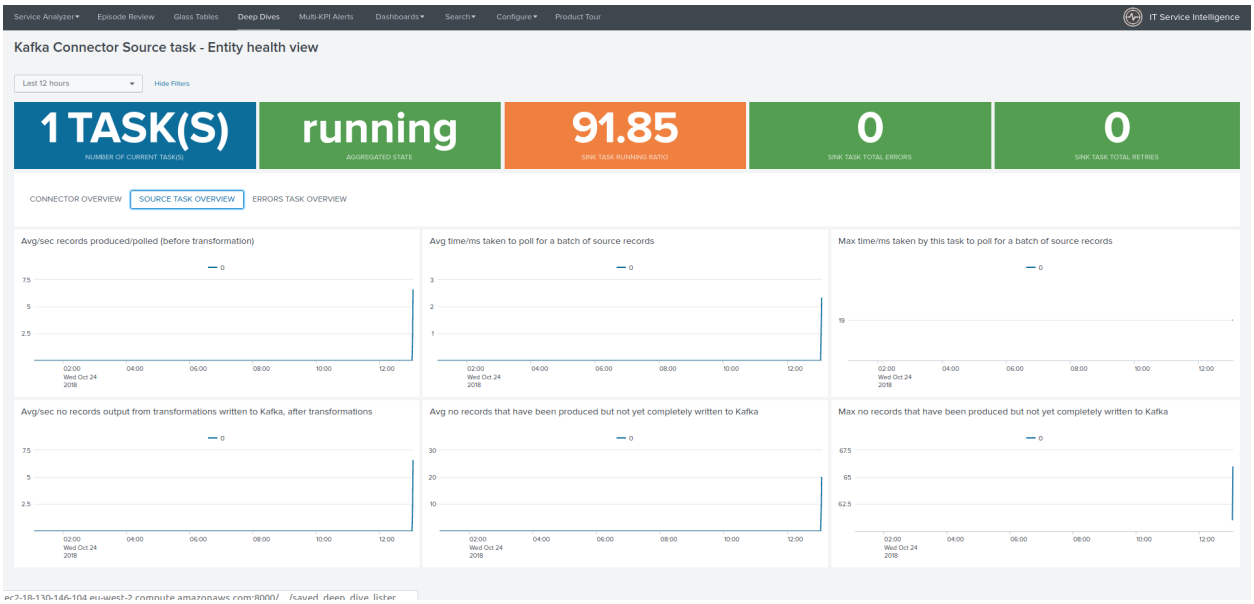


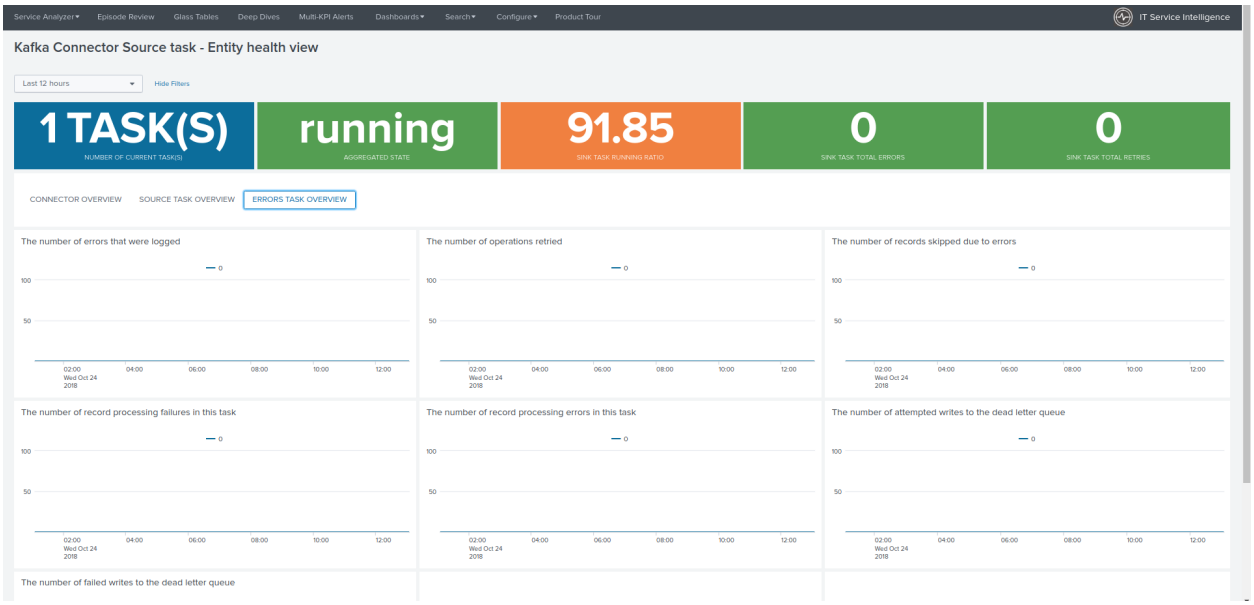




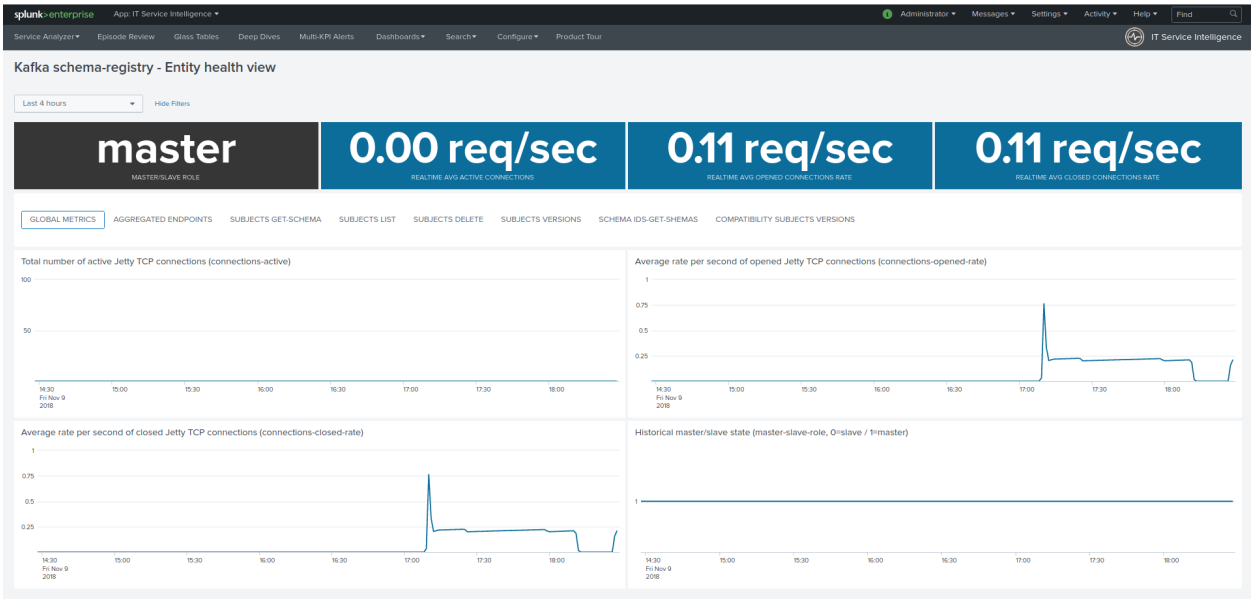
2.4.9 Kafka connect source task dashboard view

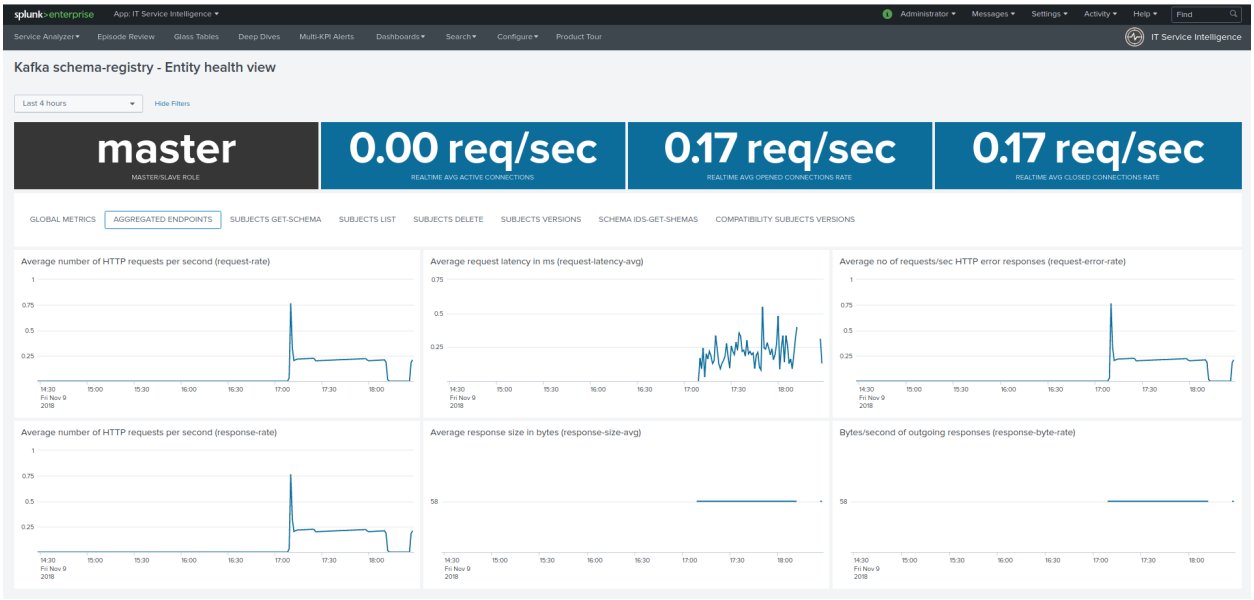



Service Analyzer
Episode Review
Glass Tables
Deep Dives
Multi-KPI Alerts
Dashboards
Search
Configure
Product Tour
IT Service Intelligence
Kafka Connector Source task - Entity health view
Last 12 hours
Hide Filters
1 TASK(S)
NUMBER OF CURRENT TASK(S)
running
AGGREGATED STATE
91.85
SINK TASK RUNNING RATIO
0
SINK TASK TOTAL ERRORS
0
SINK TASK TOTAL RETRIES
CONNECTOR OVERVIEW
SOURCE TASK OVERVIEW
ERRORS TASK OVERVIEW
Avg/sec records produced/polled (before transformation)
Avg time/ms taken to poll for a batch of source records
Max time/ms taken by this task to poll for a batch of source records
Avg/sec no records output from transformations written to Kafka, after transformations
Avg no records that have been produced but not yet completely written to Kafka
Max no records that have been produced but not yet completely written to Kafka

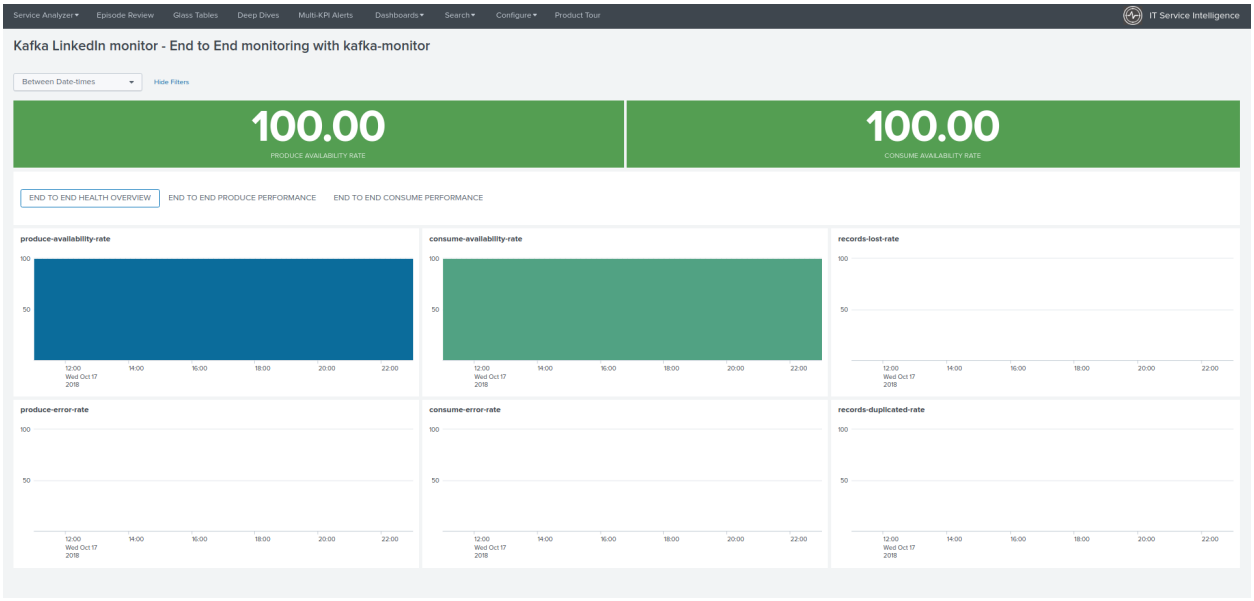

Service Analyzer
Episode Review
Glass Tables
Deep Dives
Multi-KPI Alerts
Dashboards
Search
Configure
Product Tour
IT Service Intelligence
Kafka Connector Source task - Entity health view
Last 12 hours
Hide Filters
1 TASK(S)
NUMBER OF CURRENT TASK(S)
running
AGGREGATED STATE
91.85
SINK TASK RUNNING RATIO
0
SINK TASK TOTAL ERRORS
0
SINK TASK TOTAL RETRIES
CONNECTOR OVERVIEW
SOURCE TASK OVERVIEW
ERRORS TASK OVERVIEW
The number of errors that were logged
The number of operations retried
The number of records skipped due to errors
The number of record processing failures in this task
The number of record processing errors in this task
The number of attempted writes to the dead letter queue
The number of failed writes to the dead letter queue

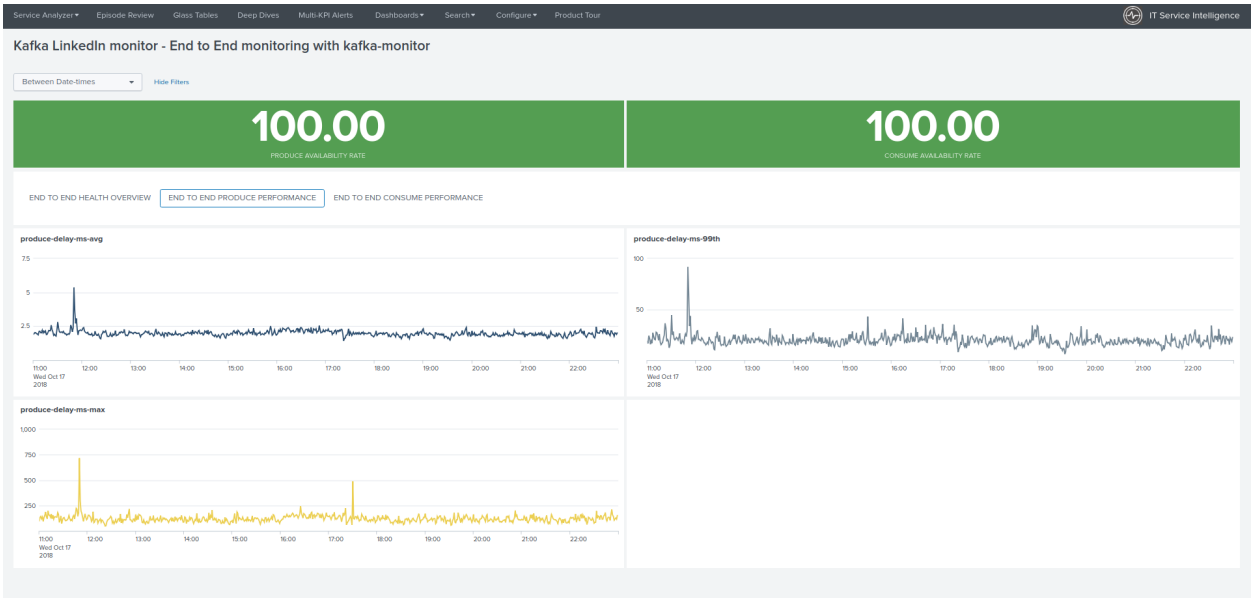
2.4.10 Confluent schema-registry dashboard view

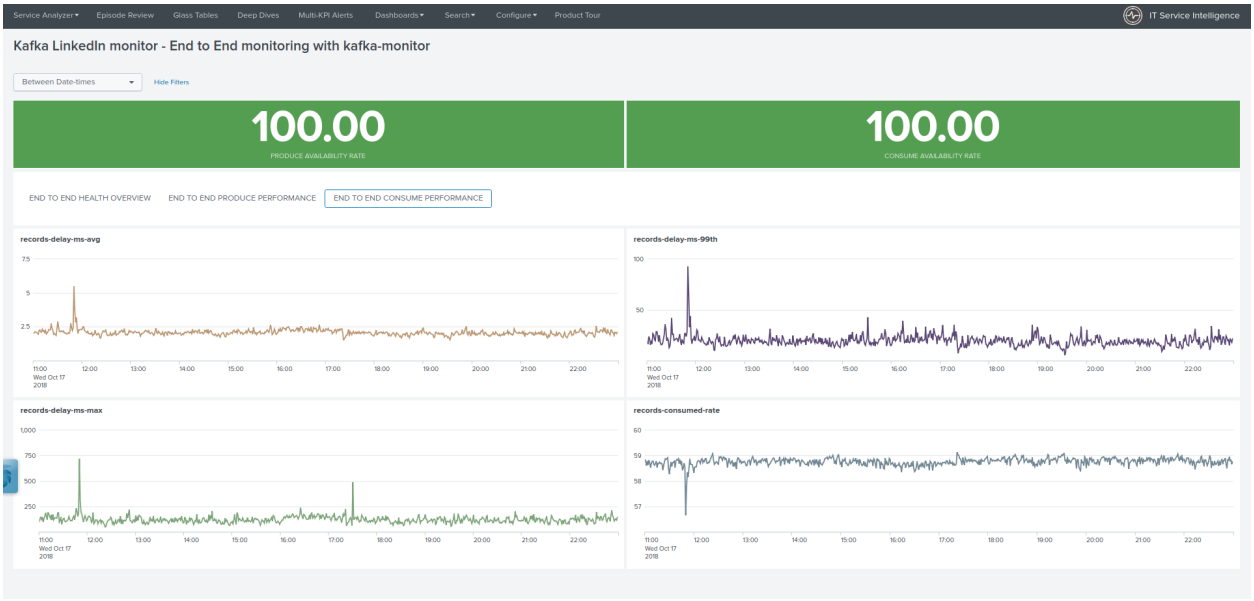




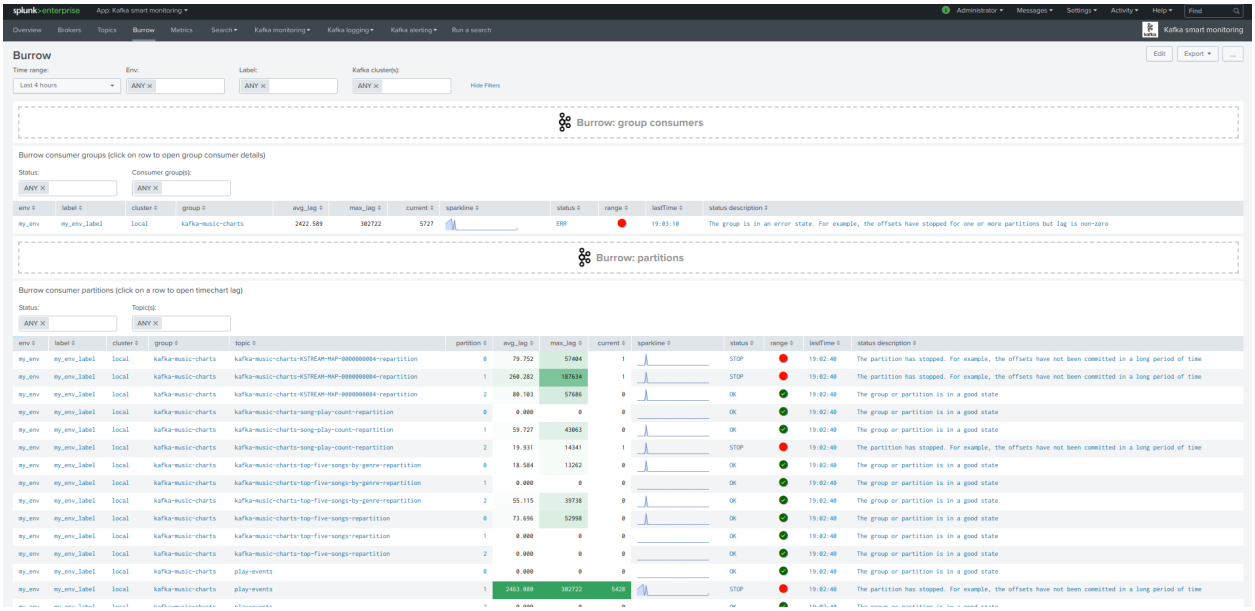
2.4.11 LinkedIn Kafka monitor view

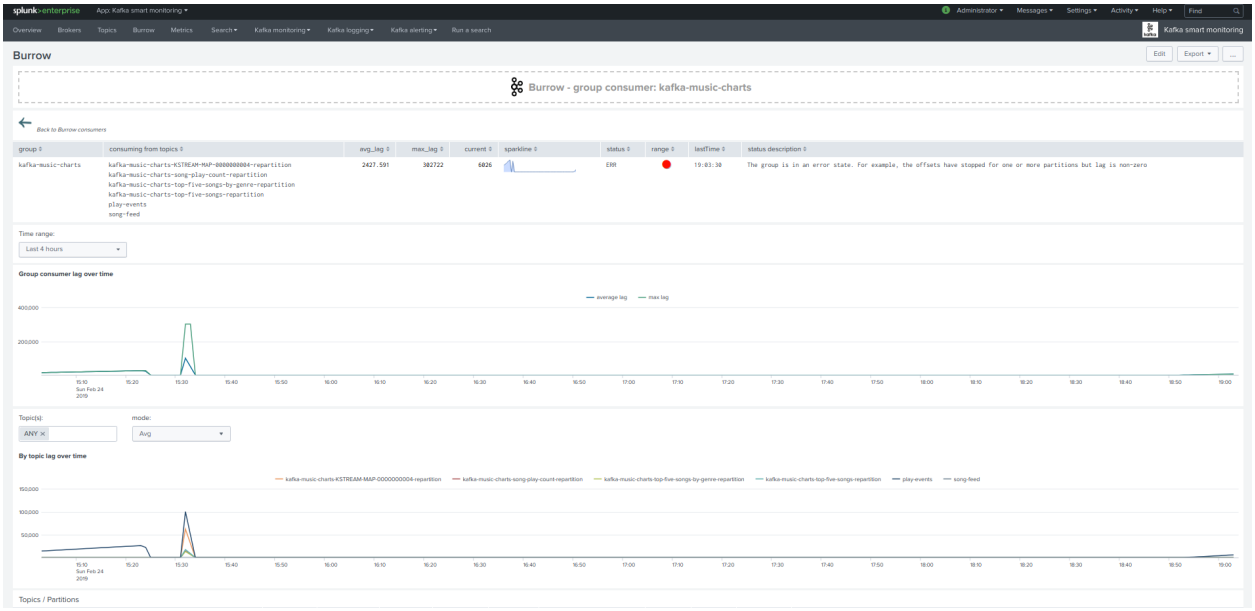


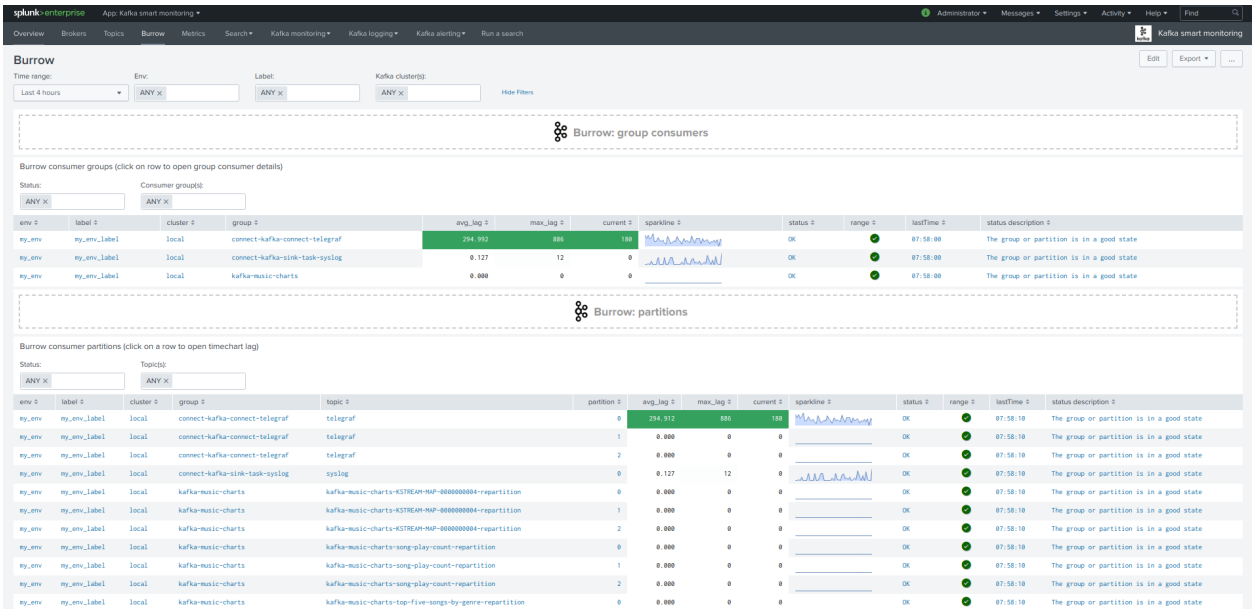


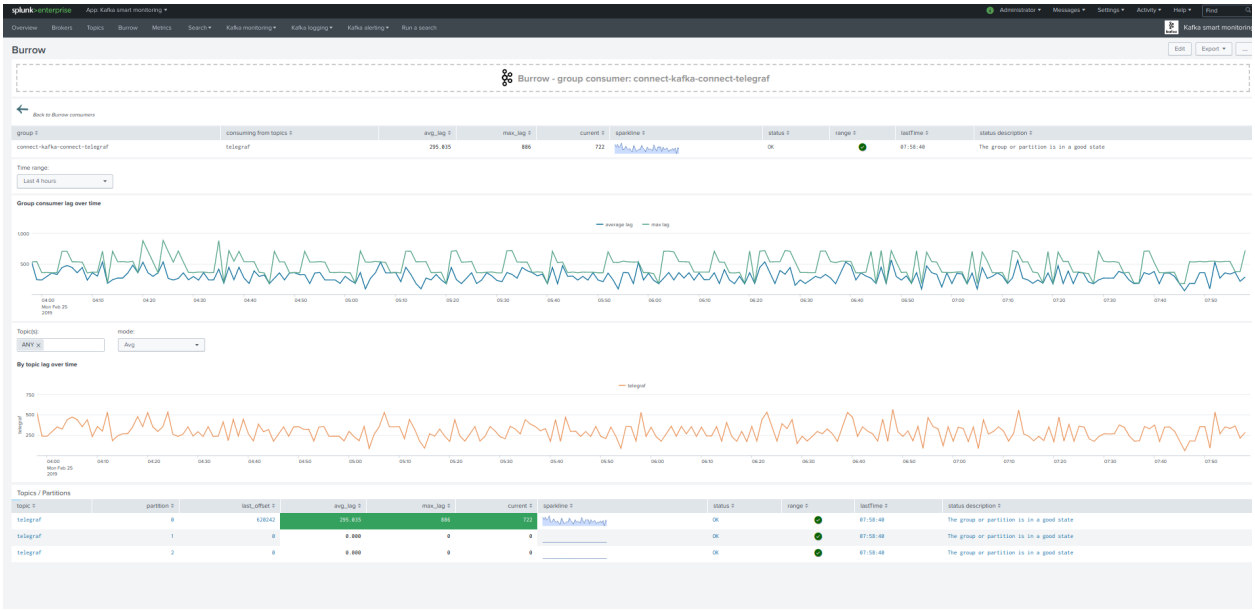


2.4.12 Burrow Kafka Consumers lagging view




splunk>enterprise
App: Kafka smart monitoring
Administrator
Messages
Settings
Activity
Help
Find
Overview
Brokers
Topics
Burrow
Metrics
Search
Kafka monitoring
Kafka logging
Kafka alerting
Run a search
Kafka smart monitoring
Burrow
Edit
Export
Burrow - group consumer: kafka-music-charts
Back to Burrow consumers
group
consuming from topics
avg_lag
max_lag
current
sparkline
status
range
lastTime
status description
kafka-music-charts
kafka-music-charts-KSTREAM-MAP-0000000004-repartition
kafka-music-charts-song-play-count-repartition
kafka-music-charts-top-five-songs-by-genre-repartition
kafka-music-charts-top-five-songs-repartition
play-events
song-feed
2427.591
302722
6026
ERR
19:03:30
The group is in an error state. For example, the offsets have stopped for one or more partitions but lag is non-zero
Time range:
Last 4 hours
Group consumer lag over time
average lag
max lag
400,000
200,000
15:10
Sun Feb 24
2019
15:20
15:30
15:40
15:50
16:00
16:10
16:20
16:30
16:40
16:50
17:00
17:10
17:20
17:30
17:40
17:50
18:00
18:10
18:20
18:30
18:40
18:50
19:00
Topic(s):
ANY ×
mode:
Avg
By topic lag over time
kafka-music-charts-KSTREAM-MAP-0000000004-repartition
kafka-music-charts-song-play-count-repartition
kafka-music-charts-top-five-songs-by-genre-repartition
kafka-music-charts-top-five-songs-repartition
play-events
song-feed
150,000
100,000
50,000
Topics / Partitions

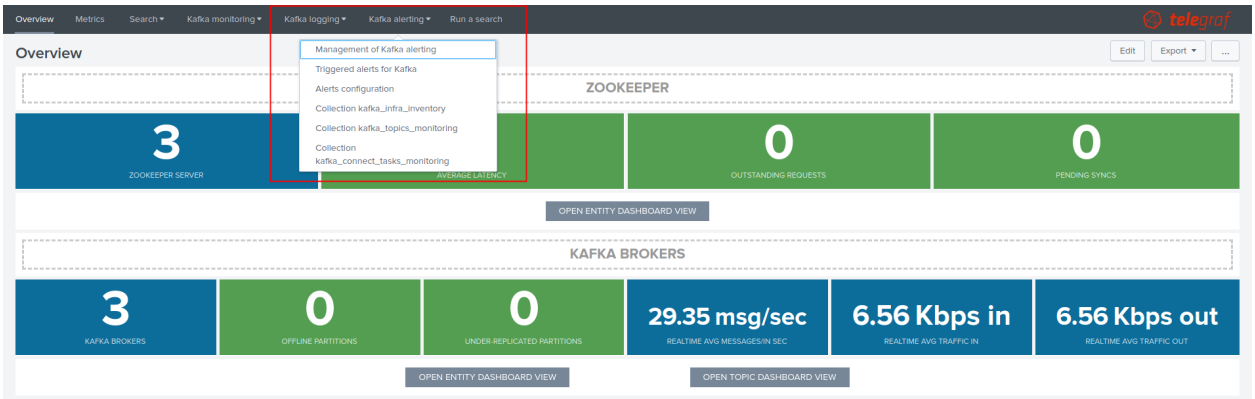

splunk>enterprise
App: Kafka smart monitoring
Administrator
Messages
Settings
Activity
Help
Find
Overview
Brokers
Topics
Burrow
Metrics
Search
Kafka monitoring
Kafka logging
Kafka alerting
Run a search
Kafka smart monitoring
Burrow
Edit
Export
Time range:
Last 4 hours
Env:
ANY ×
Label:
ANY ×
Kafka cluster(s):
ANY ×
Hide Filters
Burrow: group consumers
Burrow consumer groups (click on row to open group consumer details)
Status:
ANY ×
Consumer group(s):
ANY ×
env
label
cluster
group
avg_lag
max_lag
current
sparkline
status
range
lastTime
status description
my_env
my_env_label
local
connect-kafka-connect-telegraf
294.992
886
180
OK
07:58:00
The group or partition is in a good state
connect-kafka-sink-task-syslog
0.127
12
0
kafka-music-charts
0.000
0
Burrow: partitions
Burrow consumer partitions (click on a row to open timechart lag)
Topic(s):
partition
topic
telegraf
syslog
kafka-music-charts-KSTREAM-MAP-0000000004-repartition
kafka-music-charts-song-play-count-repartition
kafka-music-charts-top-five-songs-by-genre-repartition



2.5 Kafka infrastructure OOTB alerting

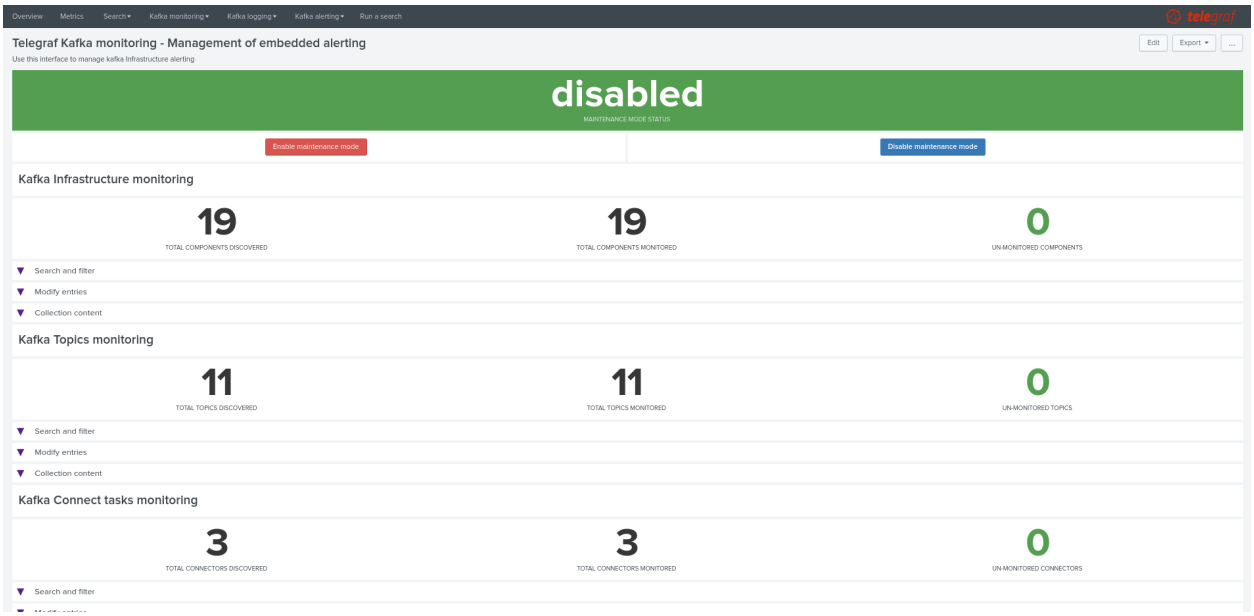
The Splunk application provides out of the box alerting for all the components of the Kafka and Confluent infrastructure.

Go straight to the Kafka alerting in app menu:



2.5.1 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:



- The Kafka infrastructure collection (kv_telegraf_kafka_inventory) contains all the components that were discovered, and is used for the stale metrics life test monitoring
- The Kafka topics collection (kv_telegraf_kafka_topics_monitoring) contains the topics discovered, with their monitoring status (enabled by default)
- The Kafka Connect tasks collection (kv_telegraf_kafka_connect_tasks_monitoring) contains the list of source and sink Connect tasks disovered, with their monitoring status (enabled by default)

Maintenance mode

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

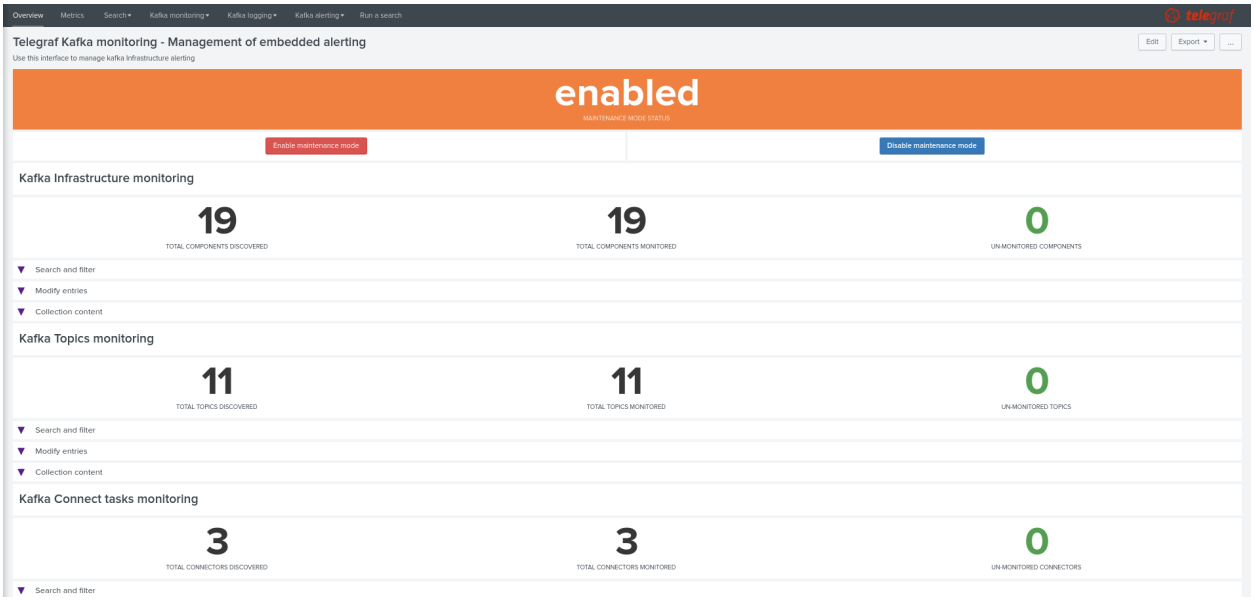
If you action the maintenance mode within the UI, this flushes the KVstore collection and will enable the maintenance mode.

Once the maintenance mode is activated, all alerts will still be running but none of them will be able to trigger during the maintenance time.

When the maintenance period is over, just disable the maintenance mode and any alert that would meet a condition will trigger as normally.

The maintenance mode has been designed to allow you massively disabling all alerts triggering at once from Splunk.

Maintenance mode is activated:



Notes: 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)

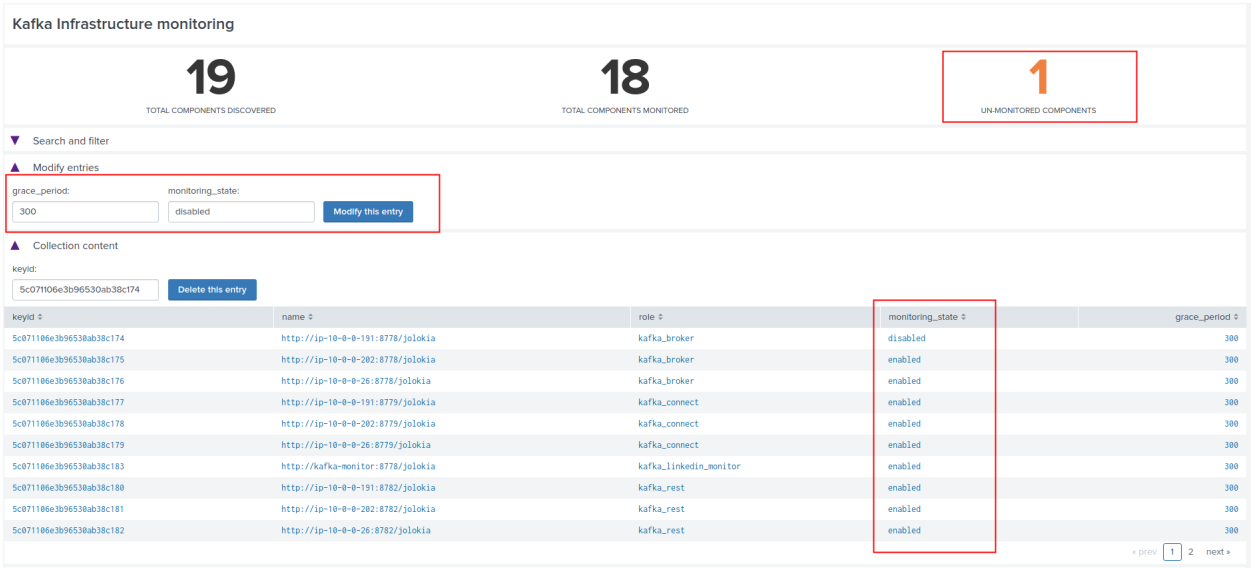
Modifying entries (monitoring state, grace period)

Each of the entities monitored, such as instances, topics and Connect tasks, have a monitoring status stored in the respective collection.

- monitoring_state=”enabled”

You can use the user interface to change this status to disabled, such that even if the conditions are met and the alert is activated, Splunk will not trigger an alert based on the status.

Select an entity to get the fields and key identifier populated automatically, achieve your modification and press the modify button:

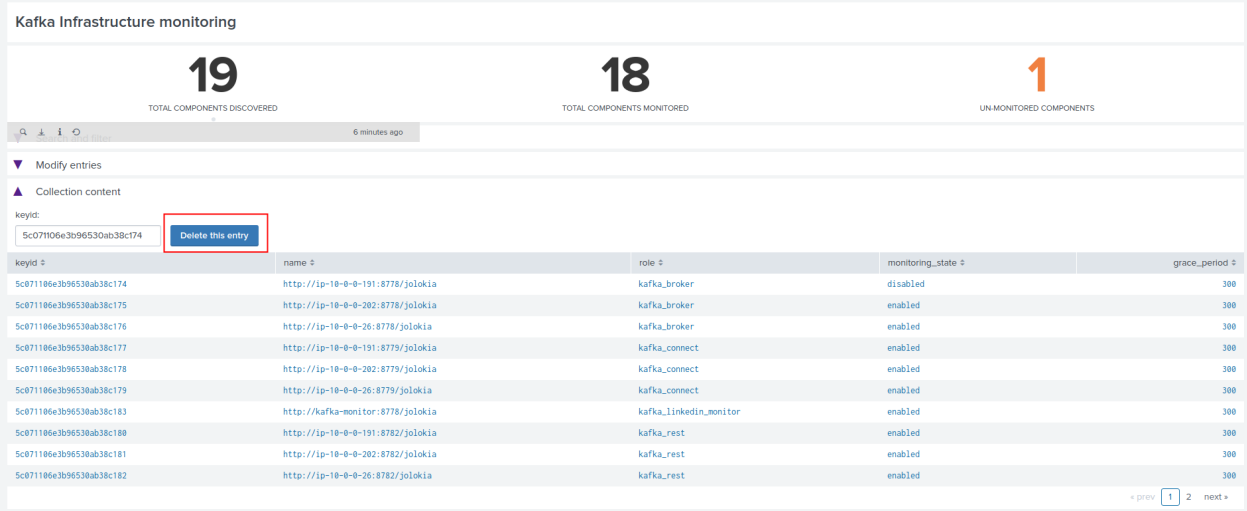


When an entity monitoring status is disabled, the single form will account this information.

Deleting entries

You can use the interface to delete entries from the collections, entries (new components discovered) are added automatically but never deleted.

Select the entity in the table to get the key identifier populated automatically, and use the delete button to remove this entity:



If these entities are still active and reporting metrics, the entity will be re-created automatically. In such a case it is preferable to disable its monitored state as long as the entity remains active.

2.5.2 Enabling OOTB alerts

By default, all alerts are disabled.

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.

Go straight to the Kafka alerting / Alerts configuration in app menu and enable your alerts.

Stale metrics life test by component

Important: All alerts are disabled by default, and must be enabled depending on your needs

Life test monitoring alerts perform a verification of the metric availability to alert on a potential downtime or issue with a component.

- Kafka monitoring - [component] - stale metrics life test

Once activated, stale metrics alert verify the grace period to be applied, and the monitoring state of the component from the KVstore collection.

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

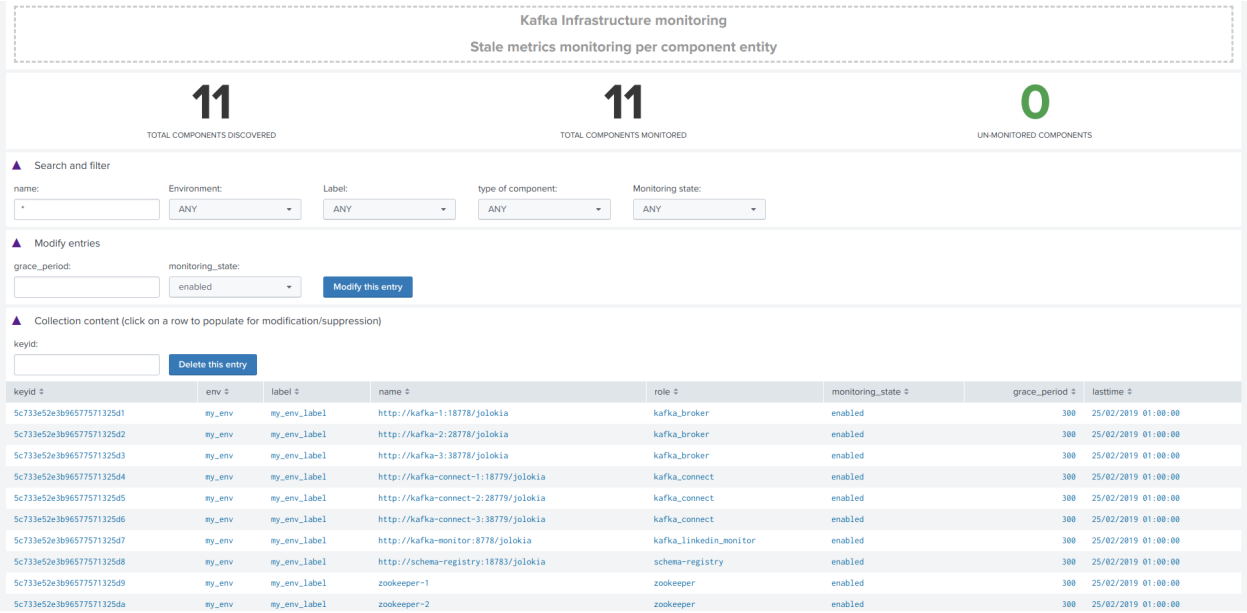
Collection content update:

The content of the collection is automatically generated by the night time scheduled report:

- Update Kafka Infrastructure components inventory

Once a component has been added to the collection, it will not be overwritten nor modified or deleted, and modifications can be made and saved safely.

Use the UI to define which nodes are being monitored:



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

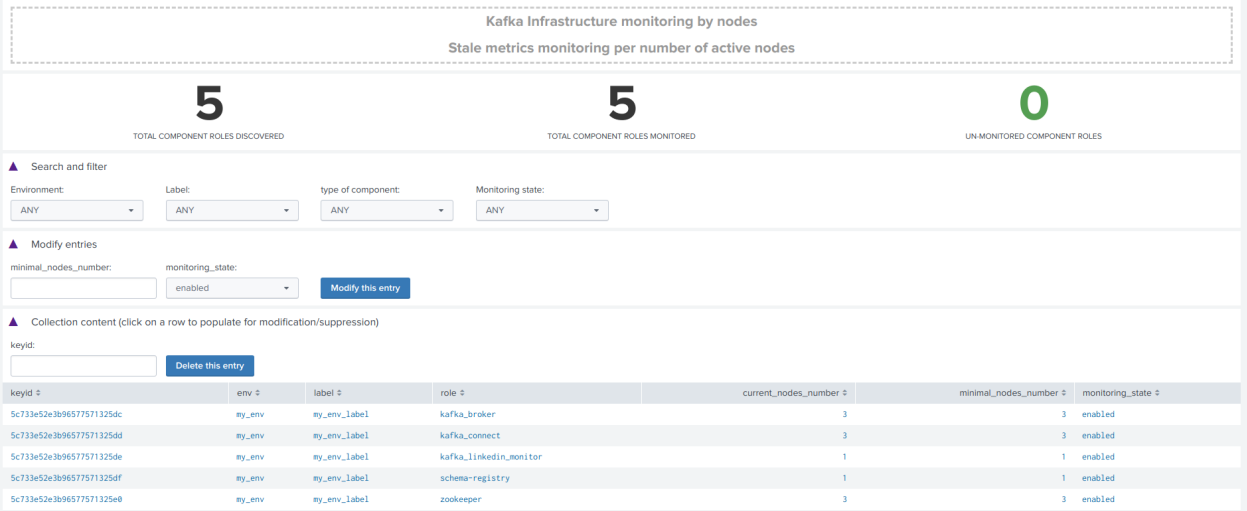
Important: All alerts are disabled by default, and must be enabled depending on your needs

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

Use the UI above to define the minimal number of nodes per component that are expected to be up and running.



Shall an upgrade of a statefullSet 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

Important: All alerts are disabled by default, and must be enabled depending on your needs

Alerts are available to monitor the main and most important items for Kafka brokers:

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

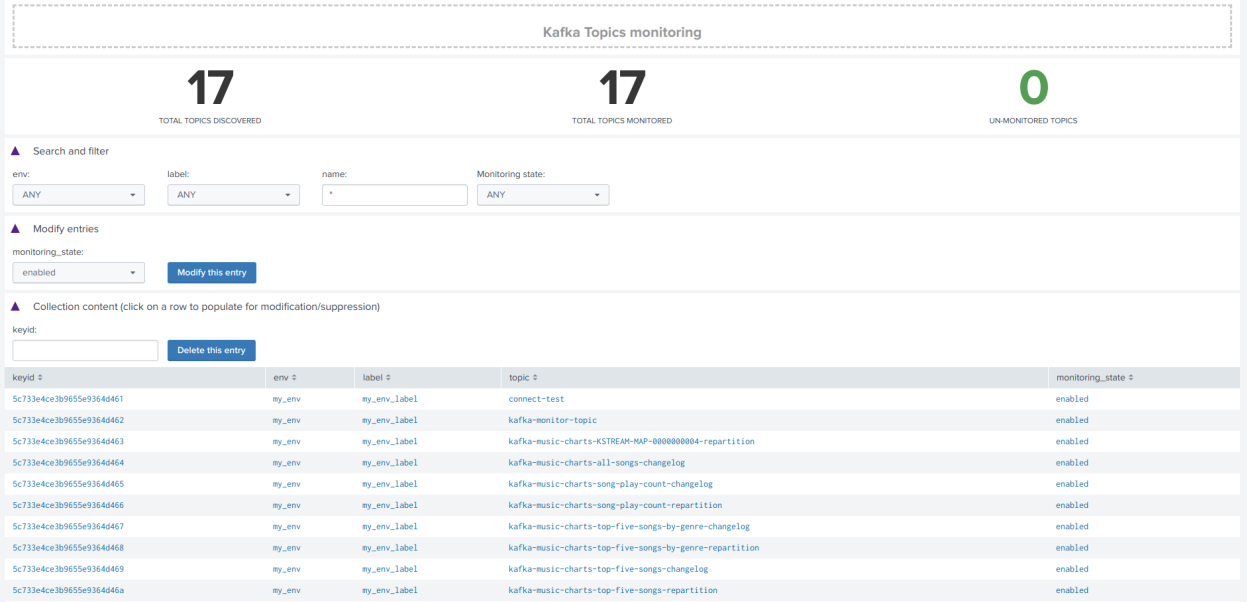
Kafka topics monitoring

Important: All alerts are disabled by default, and must be enabled depending on your needs

Topics are monitored depending on their monitored state:

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

Use the UI to define which topics are being monitored:



Kafka Connect task monitoring

Important: All alerts are disabled by default, and must be enabled depending on your needs

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

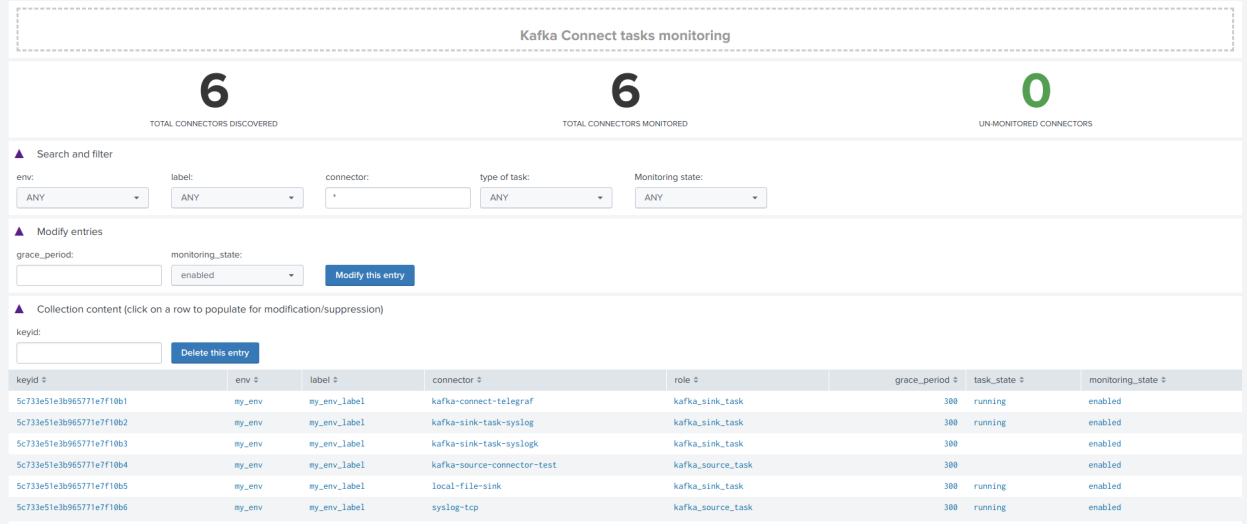
Collection content update:

The content of the collection is automatically generated by the night time scheduled report:

- Update Kafka Infrastructure components inventory

Once a component has been added to the collection, it will not be overwritten nor modified or deleted, and modifications can be made and saved safely.

Use the UI to define which Kafka Connect tasks are being monitored:



Kafka Consumers monitoring with Burrow

Important: All alerts are disabled by default, and must be enabled depending on your needs

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

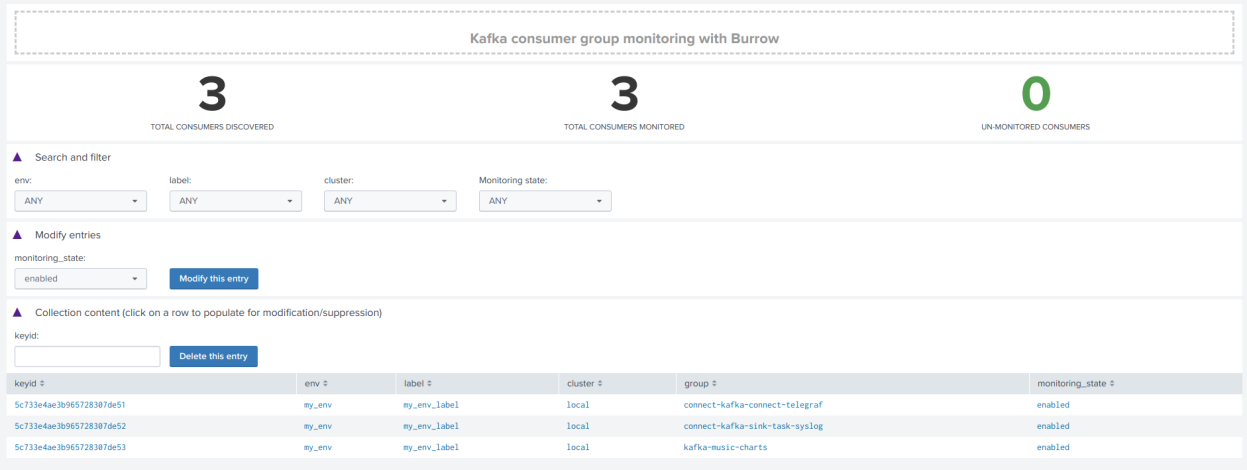
Collection content update:

The content of the collection is automatically generated by the night time scheduled report:

- Update Kafka Burrow group consumers inventory

Once a component has been added to the collection, it will not be overwritten nor modified or deleted, and modifications can be made and saved safely.

Use the UI to define which group consumers are being monitored:



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.

CHAPTER 3

Troubleshoot:

3.1 Troubleshoot & FAQ

CHAPTER 4

Versioniong and build history:

4.1 Release notes

4.1.1 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

4.1.2 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.

4.1.3 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 bootbox 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)

4.1.4 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 bootbox 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

4.1.5 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 dopdown 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 Kvstore 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 Kvstore 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