DA-ITSI-TELEGRAF-KAFKA Documentation

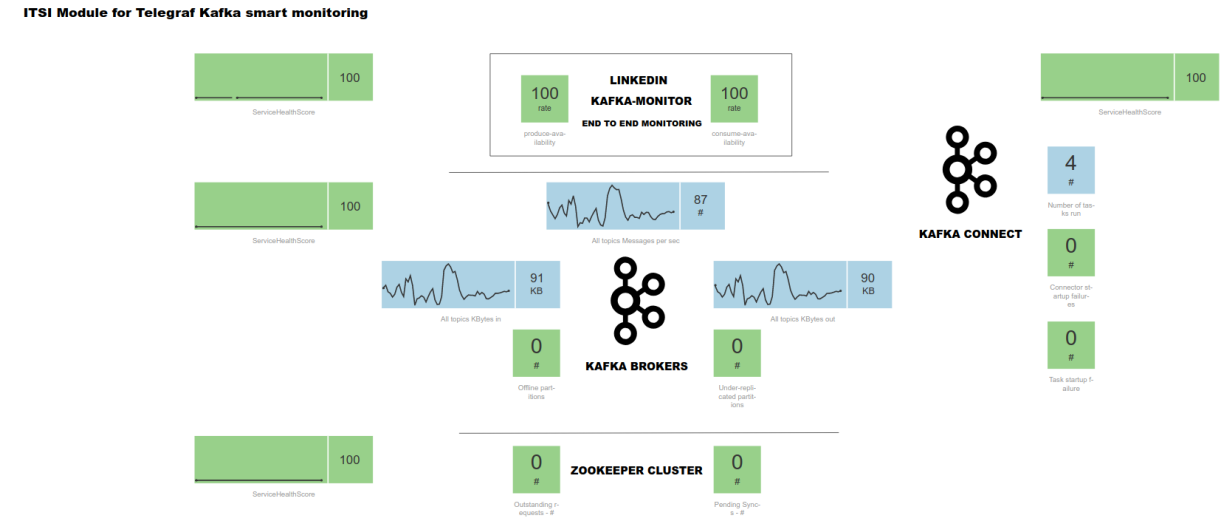
Release 1

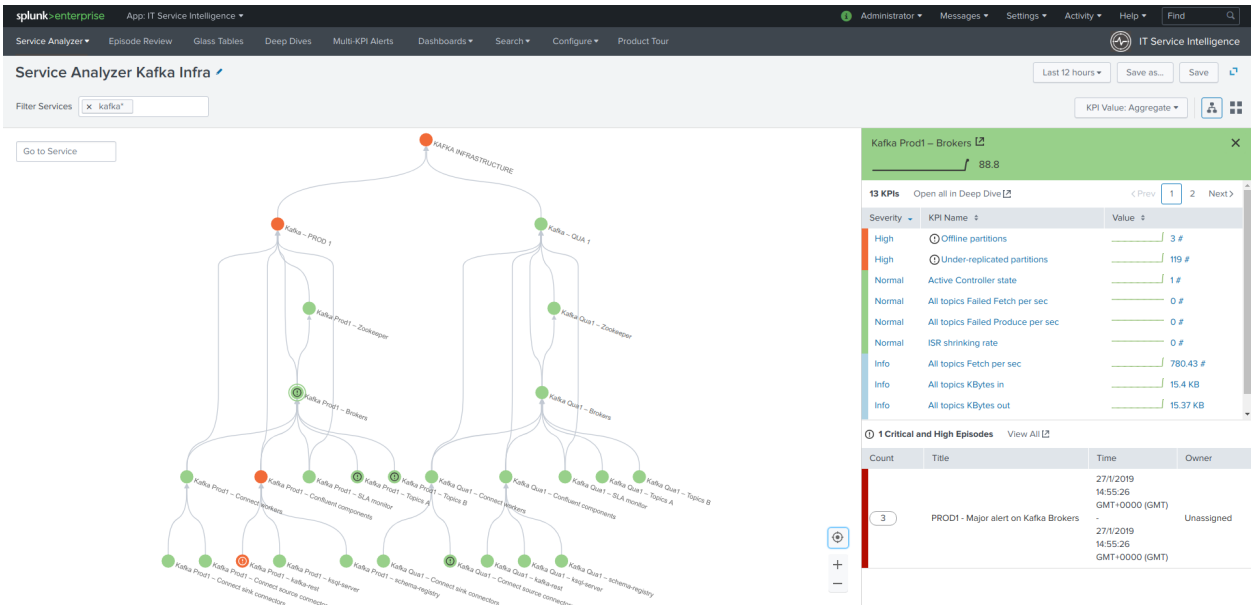
Guilhem Marchand

Feb 28, 2020

Contents

The ITSI module for Telegraf Kafka monitoring provides smart insight monitoring for Apache Kafka monitoring, on top of Splunk and ITSI.





The ITSI provides builtin and native monitoring for all 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 ITSI module can manage different environments or data-centers 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

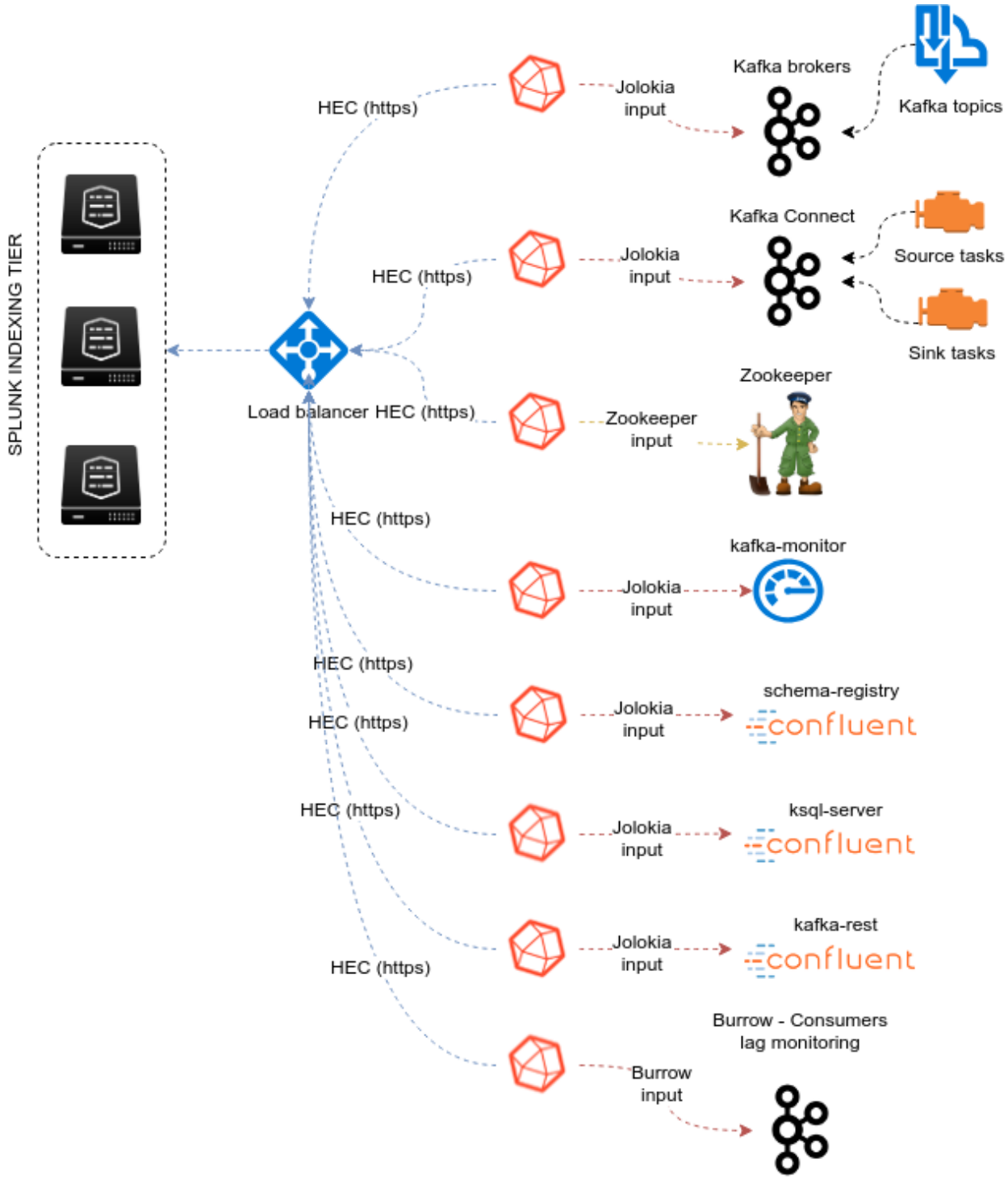
- Author: Guilhem Marchand
- First release published in October 2018
- Purposes:

The ITSI module for Apache Kafka end to end 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)

The ITSI module provides a native and builtin integration with Splunk and ITSI:

- Builtin entities discovery for Zookeeper servers, Kafka brokers, Kafka connect nodes, Kafka connect source and sink tasks, Kafka-monitor, Kafka topics, Kafka Consumers, Confluent schema-registry/ksql-servers/kafka-rest
- Services templates and KPI base searches for Zookeeper, Kafka brokers, Kafka connect and source/sink tasks, Kafka LinkedIn monitor, Kafka topics, Kafka Consumers Lag monitoring, Confluent schema-registry
- Rich entity health views to manage Operating System metrics ingested in the Splunk metric store



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 ITSI compatibility

The ITSI module has been tested and qualified against reasonably fresh versions of ITSI, recommended version is 3.1.0 and higher, previous versions may work as well although it has not and will not be tested.

1.2.3 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

1.2.4 Containers compatibility

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

1.2.5 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 ITSI module for Telegraf Apache Kafka smart monitoring 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/4261.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/DA-ITSI-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 ITSI Module for Telegraf Apache Kafka smart monitoring

The ITSI Module for Telegraf Apache Kafka can be downloaded from:

Splunk base

- https://splunkbase.splunk.com/app/4261

GitHub

- https://github.com/guilhemmarchand/DA-ITSI-TELEGRAF-KAFKA

CHAPTER 2

Deployment and configuration:

2.1 Deployment & Upgrades

2.1.1 Deployment matrix

Splunk roles	required
ITSI Search head	yes
Indexer tiers	no

If ITSI is running in Search Head Cluster (SHC), the ITSI module must be deployed by the SHC deployer.

The deployment and configuration of the ITSI module 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 ITSI module for Telegraf Kafka is straight forward.

Deploy the ITSI module using one of the following options:

- 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

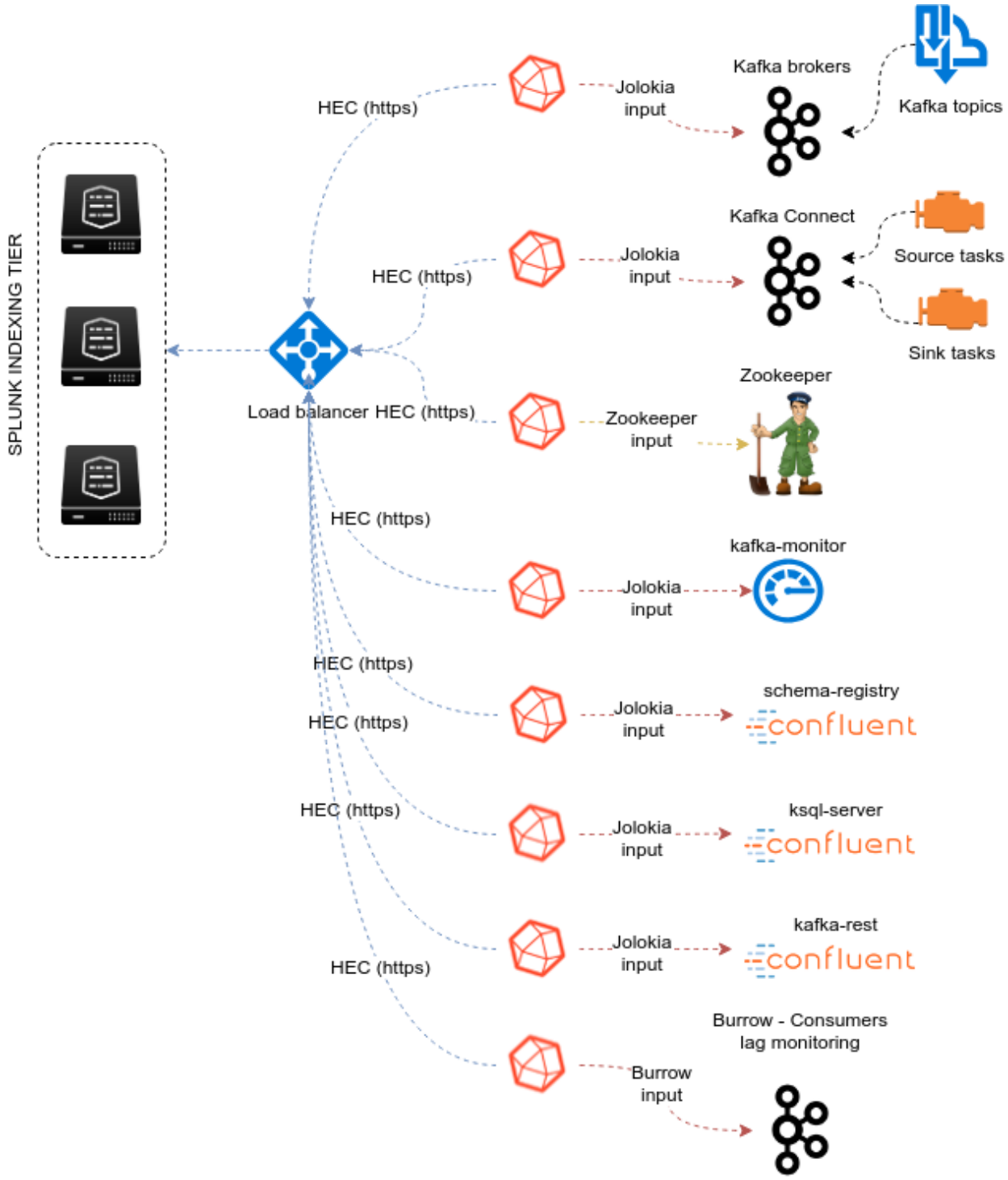
Upgrading the ITSI module is pretty much the same operation than the initial deployment.

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 ITSI module 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"** included in:

- DA-ITSI-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:

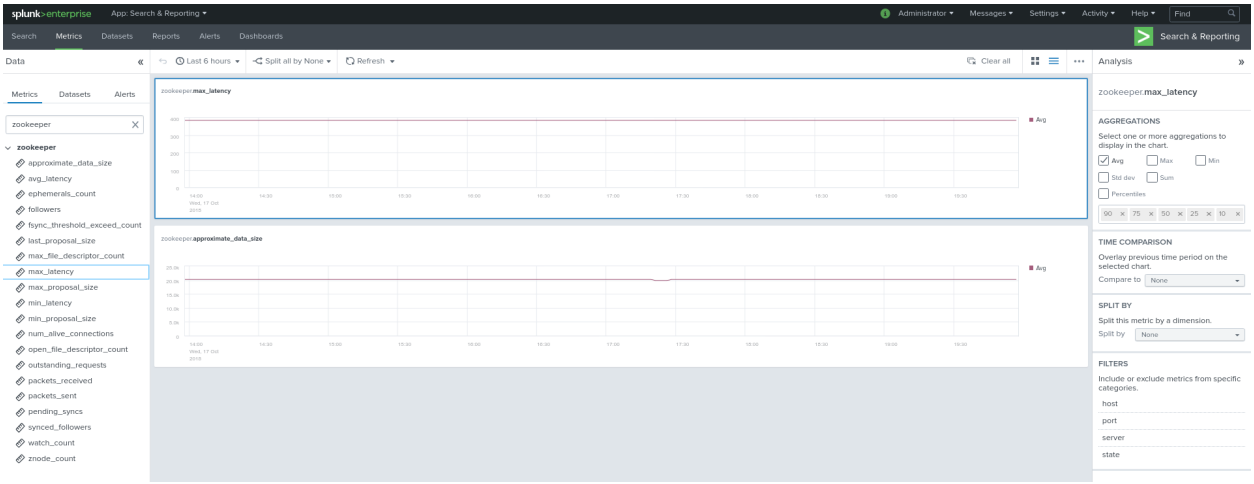
```
# zookeeper metrics
[[inputs.zookeeper]]
  servers = ["$HOSTNAME:2181"]
```

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)

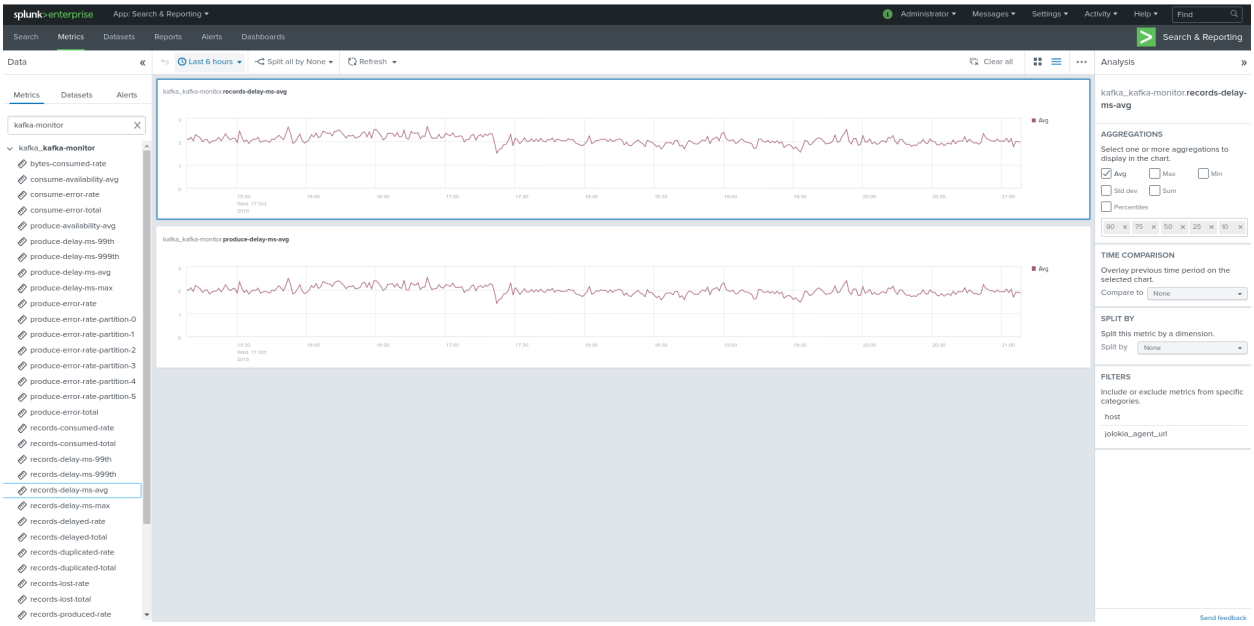
```
# outputs
[[outputs.http]]
   url = "https://splunk:8088/services/collector"
   insecure_skip_verify = true
   data_format = "splunkmetric"
    ## Provides time, index, source overrides for the HEC
   splunkmetric_hec_routing = true
    ## Additional HTTP headers
    [outputs.http.headers]
   # Should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

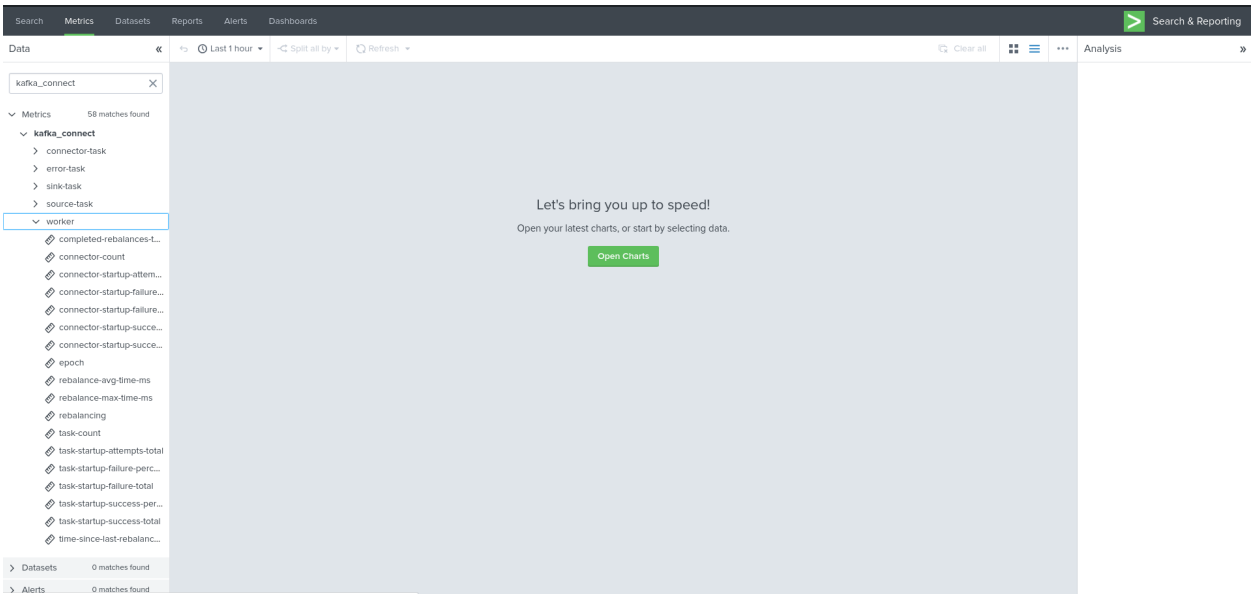
```
[global_tags]
  # the env tag is used by the application for multi-environments management
  env = "my_env"
  # the label tag is an optional tag used by the application that you can use as␣
↪additional label for the services or infrastructure
  label = "my_env_label"

[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 !

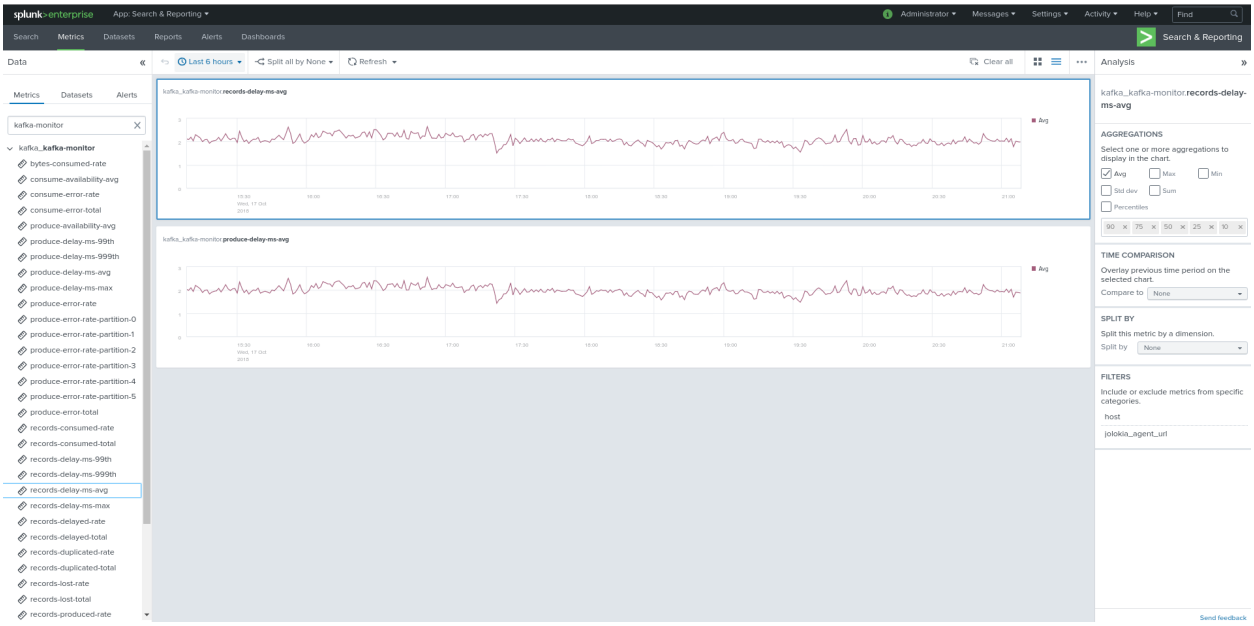
It is very straightforward to run the kafka-monitor in a docker container, first you need to create your own image:

- 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:

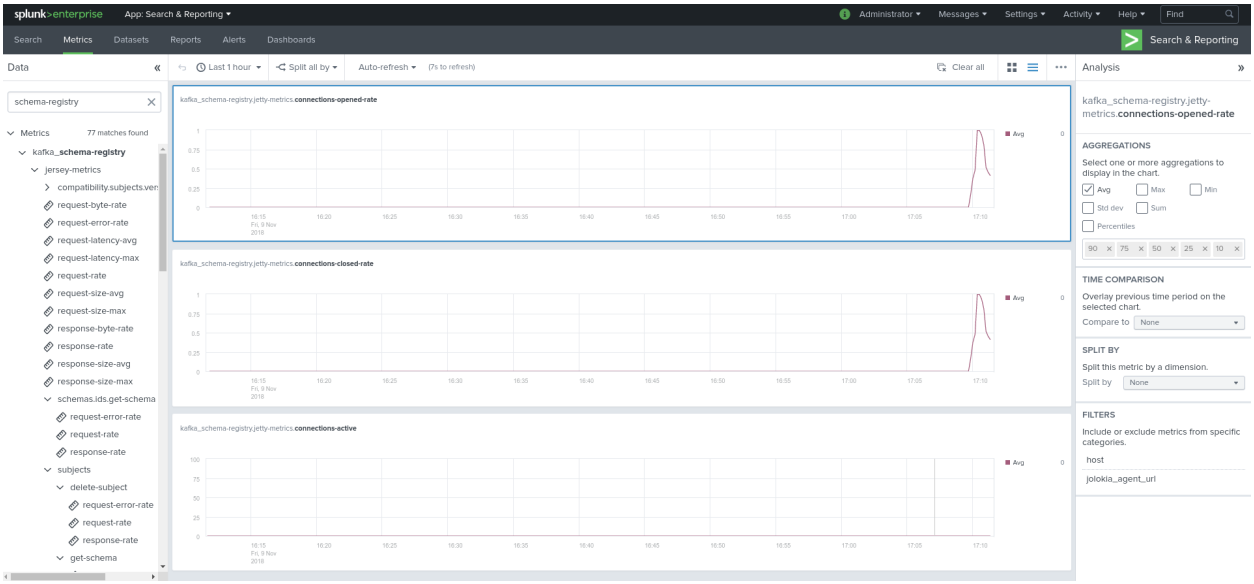
```
# Kafka-connect JVM monitoring
 [[inputs.jolokia2_agent]]
  name_prefix = "kafka_schema-registry."
   urls = ["http://$HOSTNAME:8778/jolokia"]
```

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:

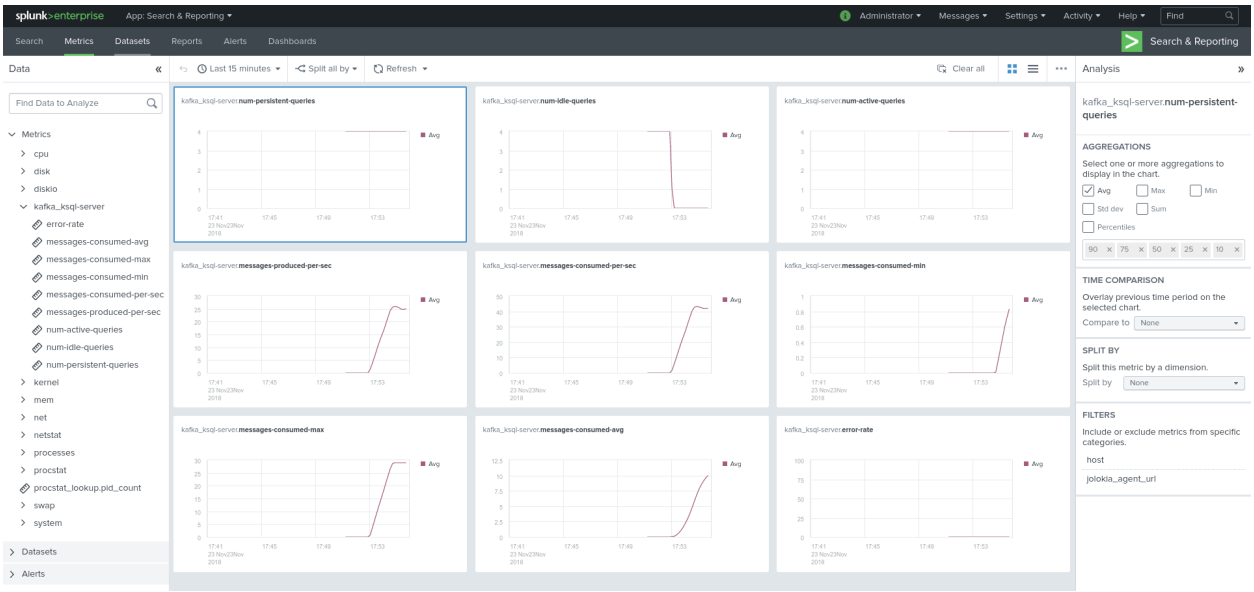
```
[[inputs.jolokia2_agent]]
  name_prefix = "kafka_"
  urls = ["http://$HOSTNAME:18784/jolokia"]
```

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:

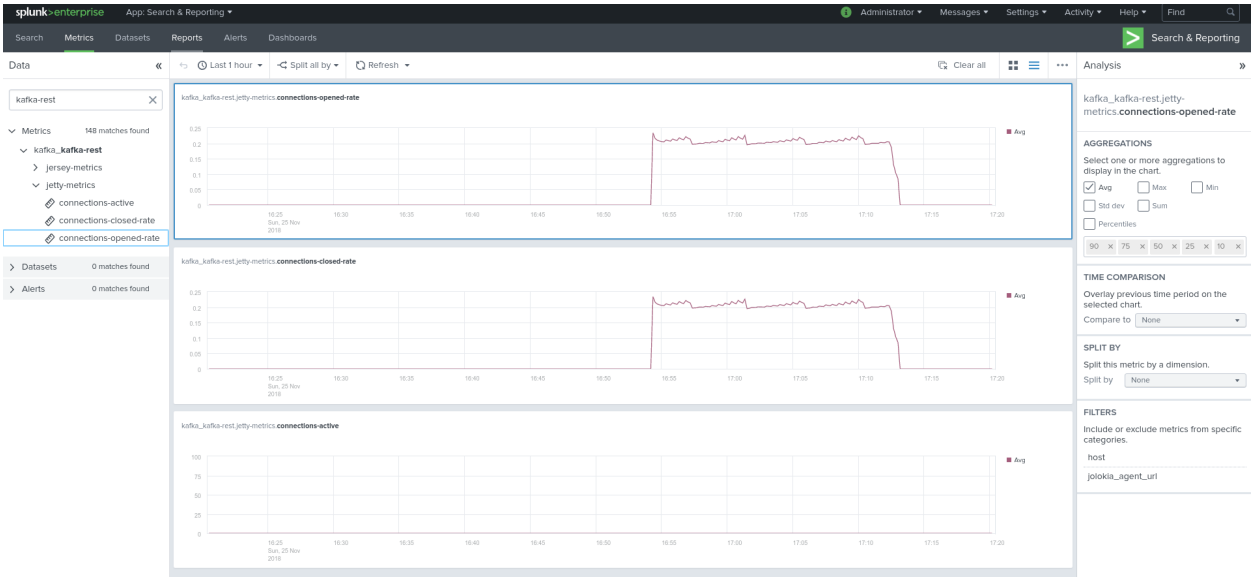
```
[[inputs.jolokia2_agent]]
  name_prefix = "kafka_kafka-rest."
  urls = ["http://$HOSTNAME:18785/jolokia"]
```

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:

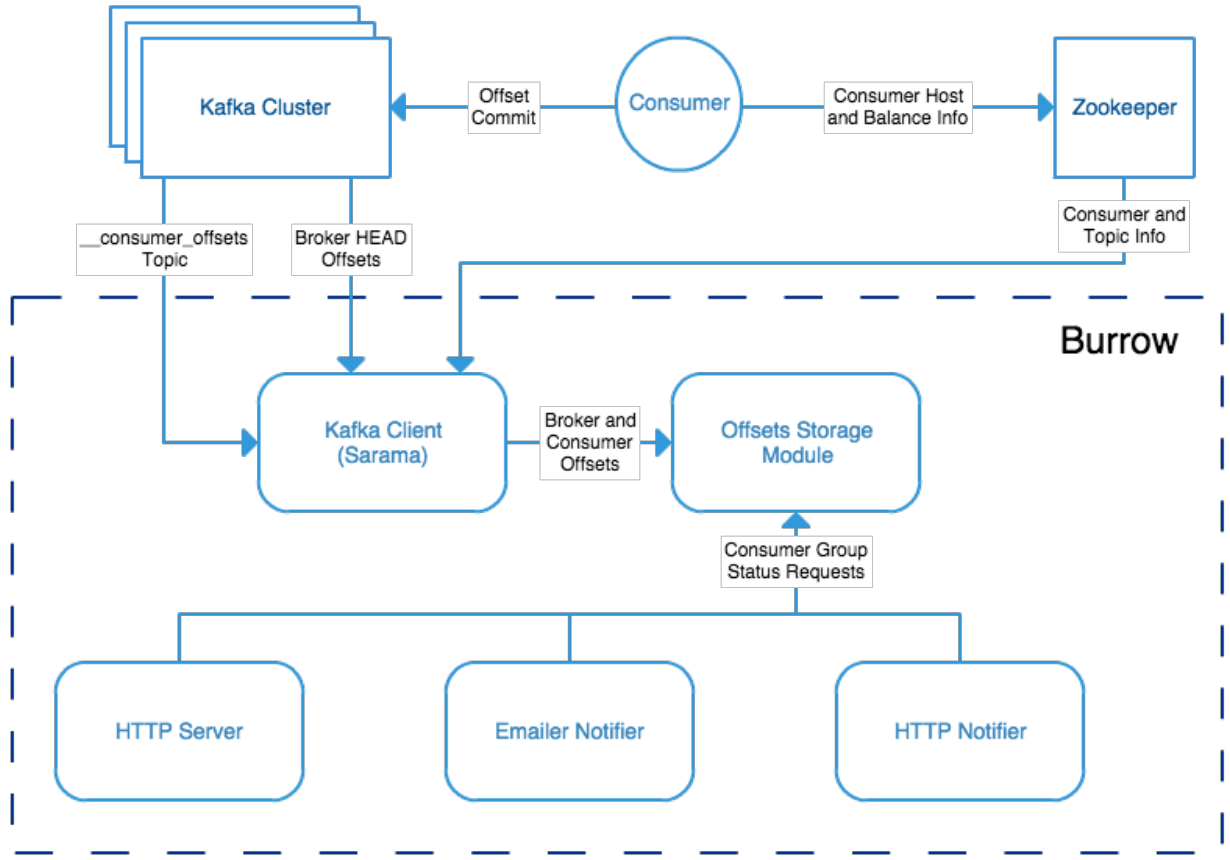
```
| mcatalog values(metric_name) values(_dims) where index=* metric_name=kafka_kafka_
↪kafka-rest.*
```

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

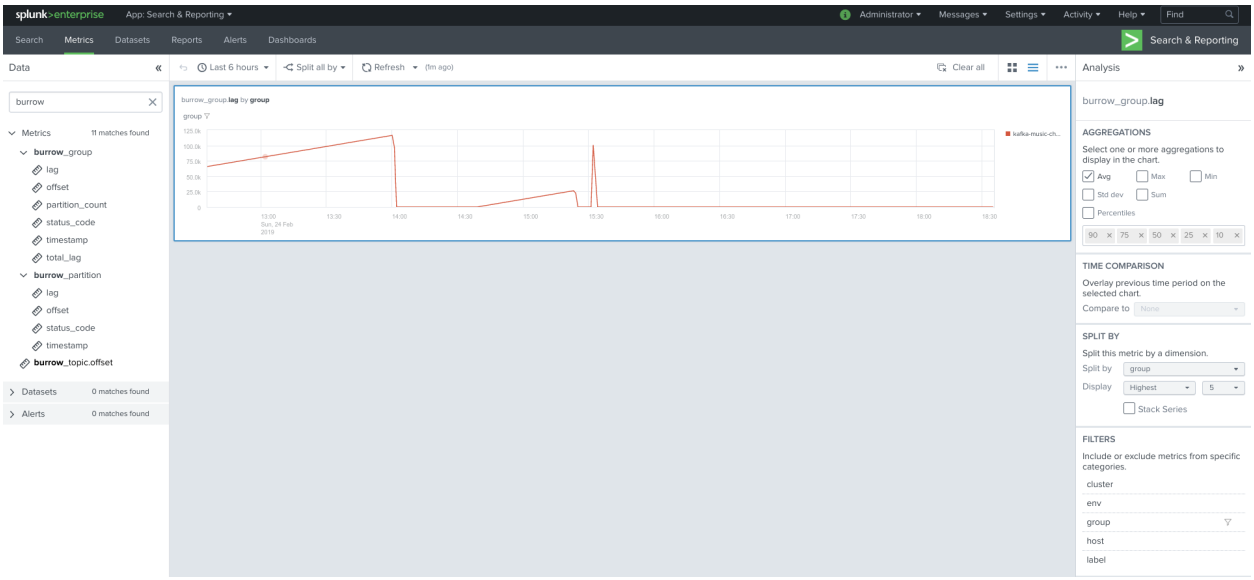
```
[global_tags]
  # the env tag is used by the application for multi-environments management
  env = "my_env"
  # the label tag is an optional tag used by the application that you can use as␣
↪additional label for the services or infrastructure
  label = "my_env_label"

[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_*
```

2.2.15 Operating System level metrics

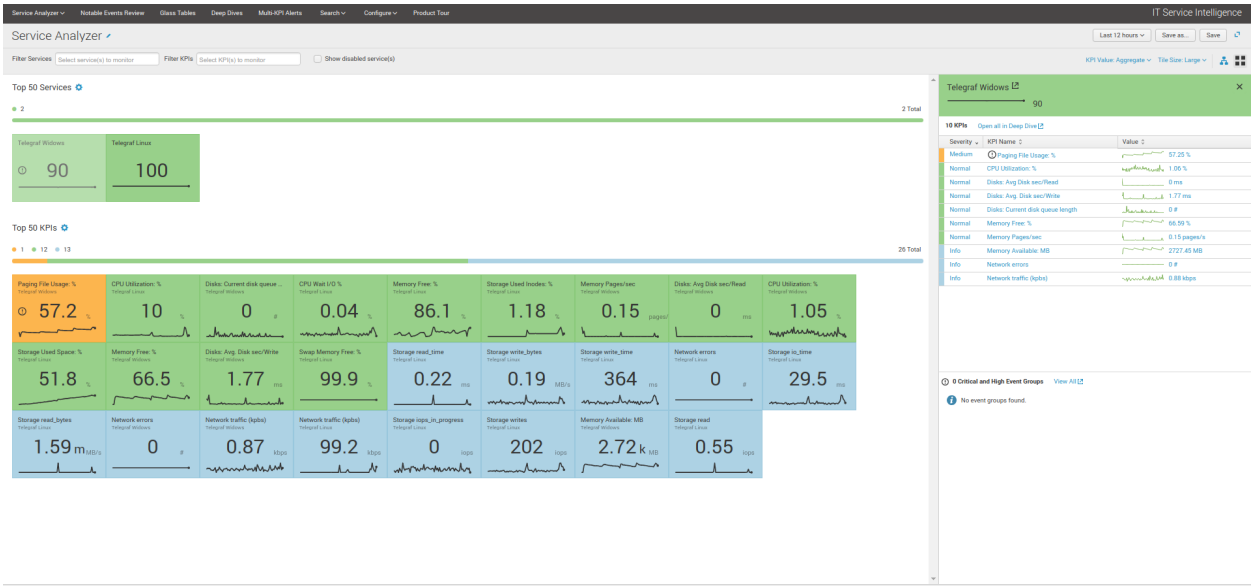
Monitoring the Operating System level metrics is fully part of the monitoring requirements of a Kafka infrastructure.

Bare metal servers and virtual machines

ITSI module for Telegraf Operating System

Telegraf has very powerful Operating System level metrics capabilities, checkout the ITSI module for Telegraf Operating System monitoring !

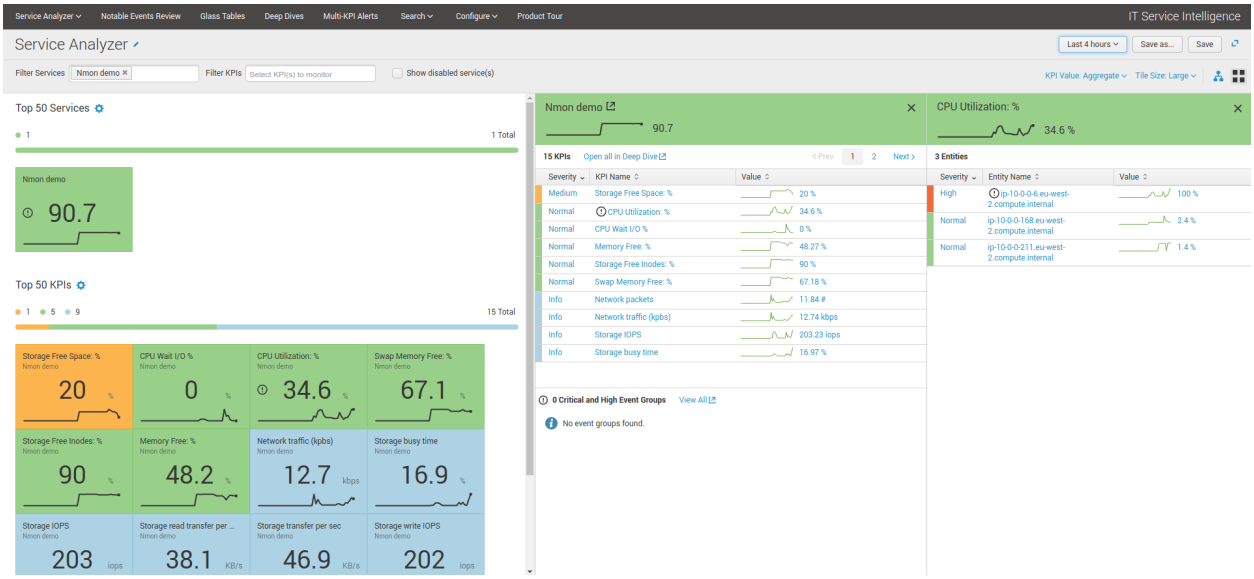
https://da-itsi-telegraf-os.readthedocs.io



ITSI module for metricator Nmon

Another very powerful way of monitoring Operating System level metrics with a builtin ITSI module and the excellent nmon monitoring:

https://www.octamis.com/metricator-docs/itsi_module.html



ITSI module for OS

Last option is using the builtin ITSI module for OS which relies on the TA-nix or TA-Windows:

http://docs.splunk.com/Documentation/ITSI/latest/IModules/AbouttheOperatingSystemModule

Containers with Docker and container orchestrators

Telegraf docker monitoring

Telegraf has very powerful inputs for Docker and is natively compatible with a container orchestrator such as Kubernetes.

Specially with Kubernetes, it is very easy to run a Telegraf container as a daemonset in Kubernetes and retrieve all the performance metrics of the containers.

2.3 Docker testing templates

Docker compose templates are provided in the following repository:

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

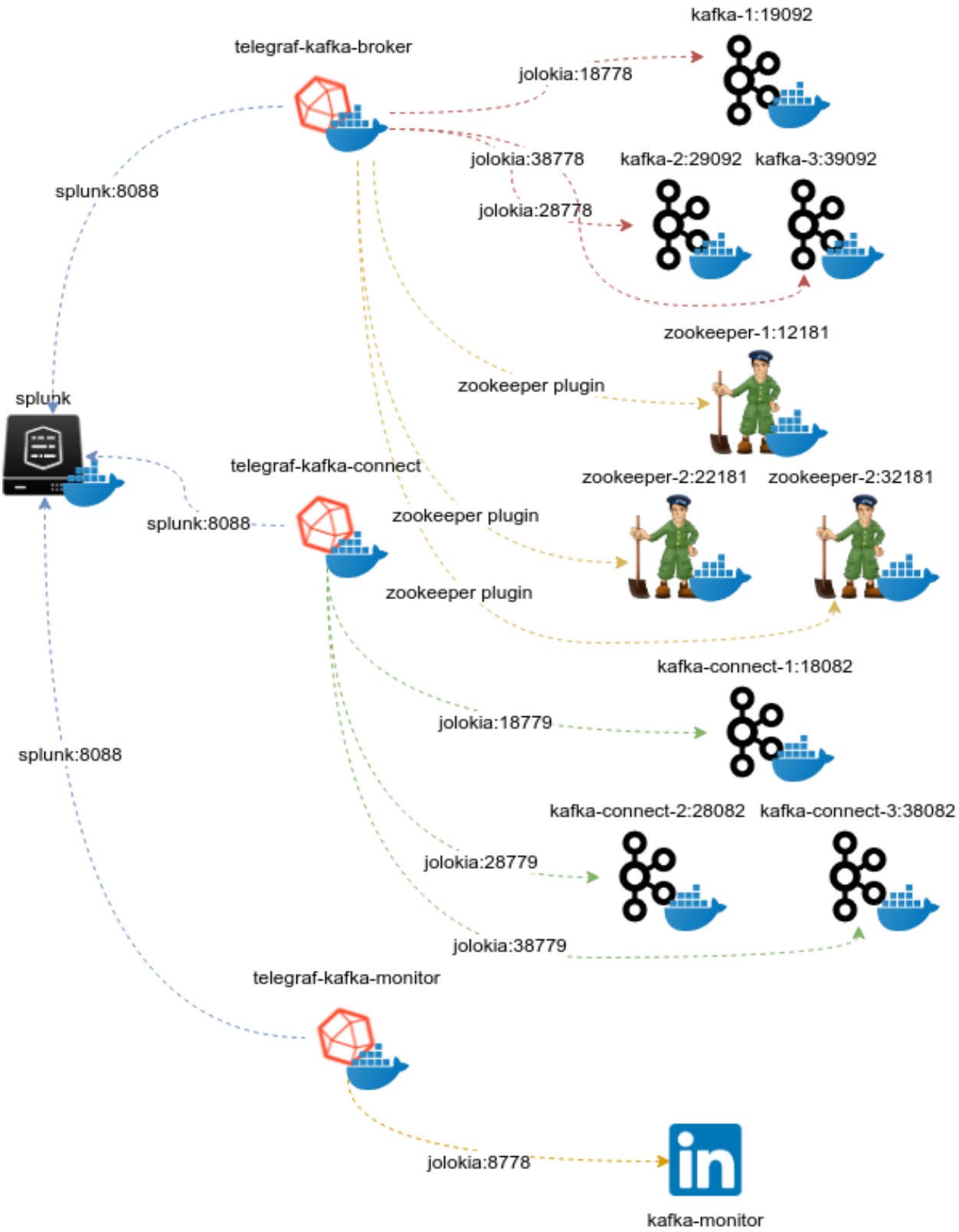
Using the docker templates allows you to create a full pre-configured Kafka environment with docker, just in 30 seconds.

Integration with Kubernetes is documented here:

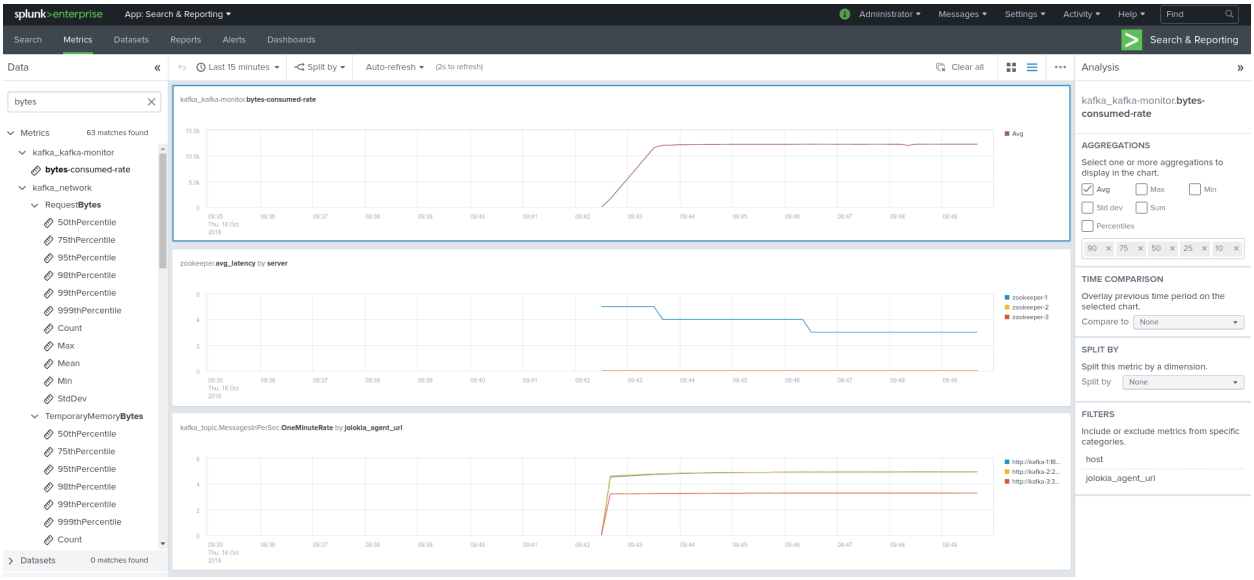
https://splunk-guide-for-kafka-monitoring.readthedocs.io

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 Entities discovery

The ITSI entities discovery is a fully automated process that will discover and properly configure your entities in ITSI depending on the data availability in Splunk.

All report rely on extremely fast and optimized queries with mcatalog, which has a negligible processing cost for the Splunk infrastructure.

2.4.1 Entities automatic import

In a nutshell, the following reports are automatically scheduled:

Purpose	Report
Zookeeper servers detection	DA-ITSI-TELEGRAF-KAFKA-Inventory_Search_zookeeper
Kafka brokers detection	DA-ITSI-TELEGRAF-KAFKA-Inventory_Search_kafka_brokers
Kafka topics detection	DA-ITSI-TELEGRAF-KAFKA-Inventory_Search_kafka_topics
Kafka connect detection	DA-ITSI-TELEGRAF-KAFKA-Inventory_Search_kafka_connect
Kafka connect tasks detection	DA-ITSI-TELEGRAF-KAFKA-Inventory_Search_kafka_connect_tasks
Kafka monitors detection	DA-ITSI-TELEGRAF-KAFKA-Inventory_Search_linkedin_kafka_monitors
Kafka Consumers detection	DA-ITSI-TELEGRAF-KAFKA-Inventory_Search_kafka_burrow_group_consumers
Confluent schema-registry	DA-ITSI-TELEGRAF-KAFKA-Inventory_Search_kafka_schema-registry
Confluent ksql-server	DA-ITSI-TELEGRAF-KAFKA-Inventory_Search_kafka-ksql-server
Confluent kafka-rest	DA-ITSI-TELEGRAF-KAFKA-Inventory_Search_kafka-kafka-rest

When entities are discovered, entities will be added automatically using the itsi_role information field, in addition with several other info fields depending on the components.

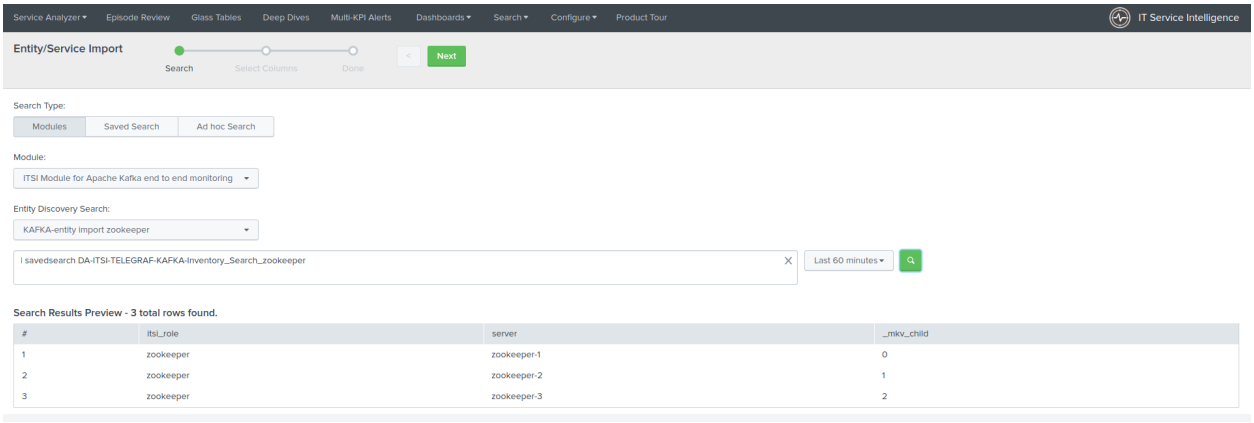
2.4.2 Manual entities import

It is possible to manually import the entities in ITSI, and use the searches above:

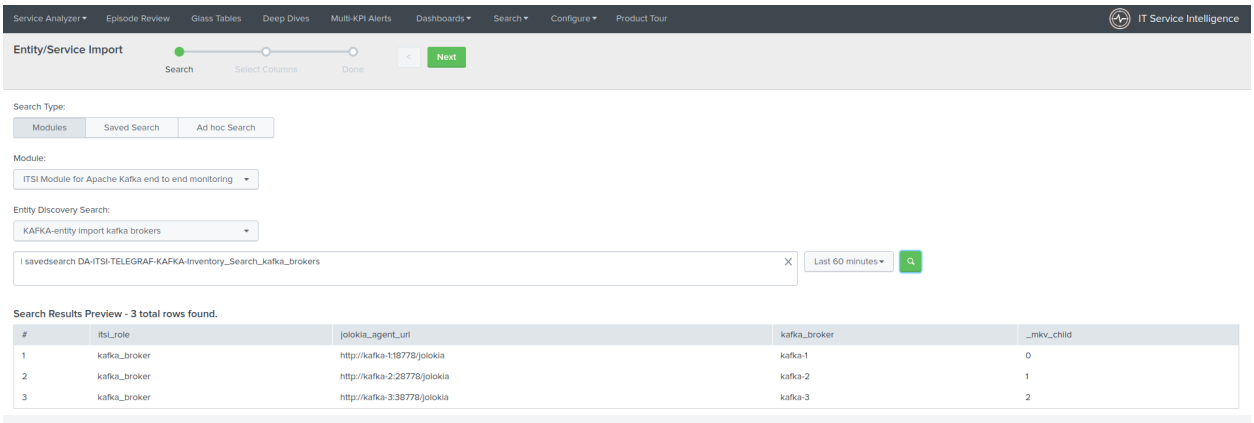
Configure / Entities / New Entity / Import from Search

Then select the module name, and depending on your needs select the relevant search.

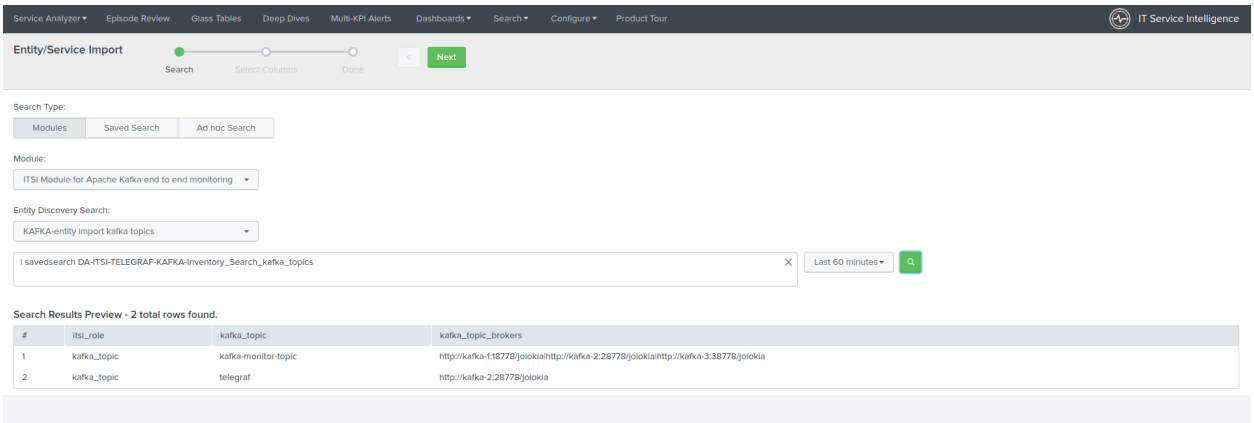
Zookeeper server detection



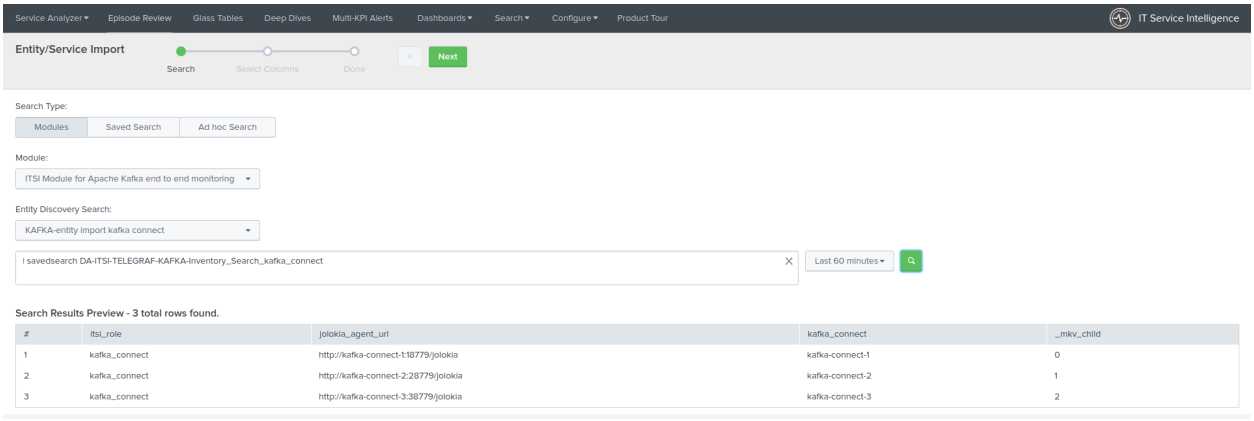
Kafka brokers detection



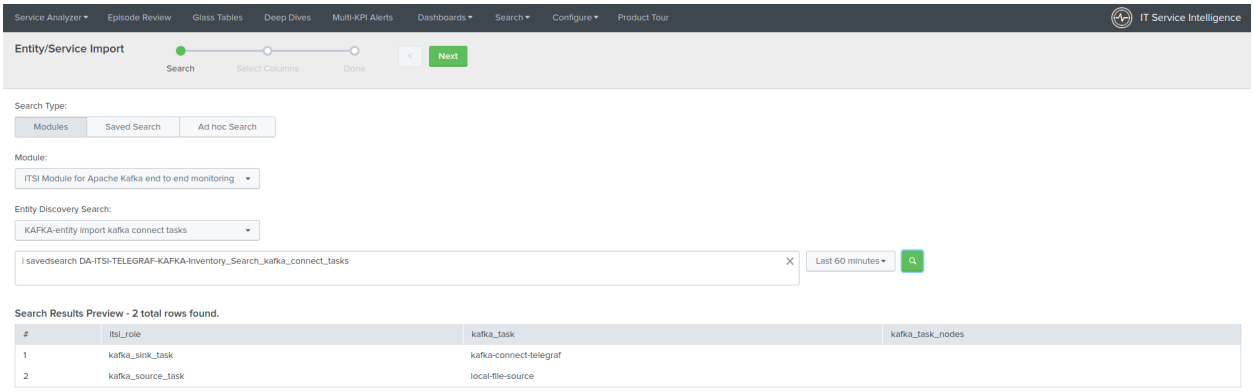
Kafka topics detection



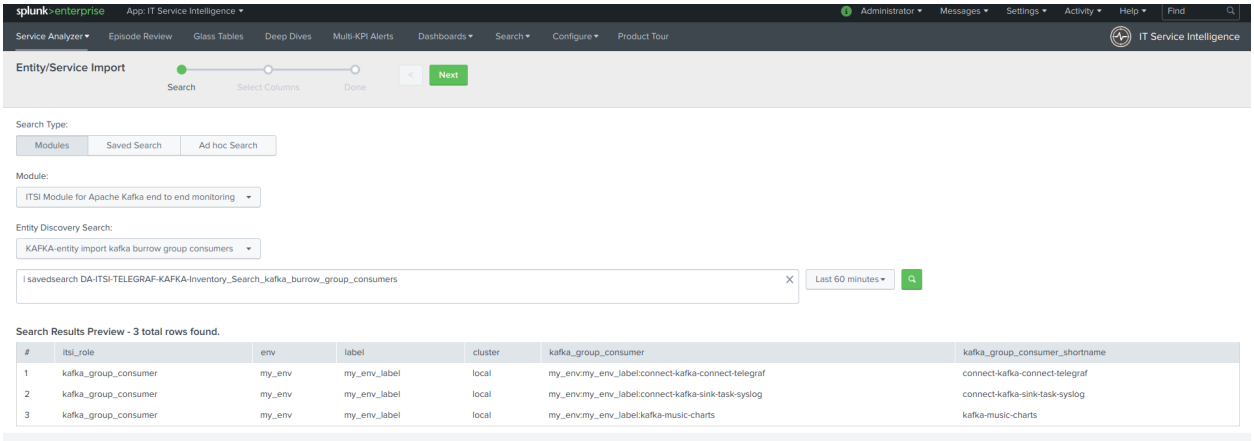
Kafka connect detection



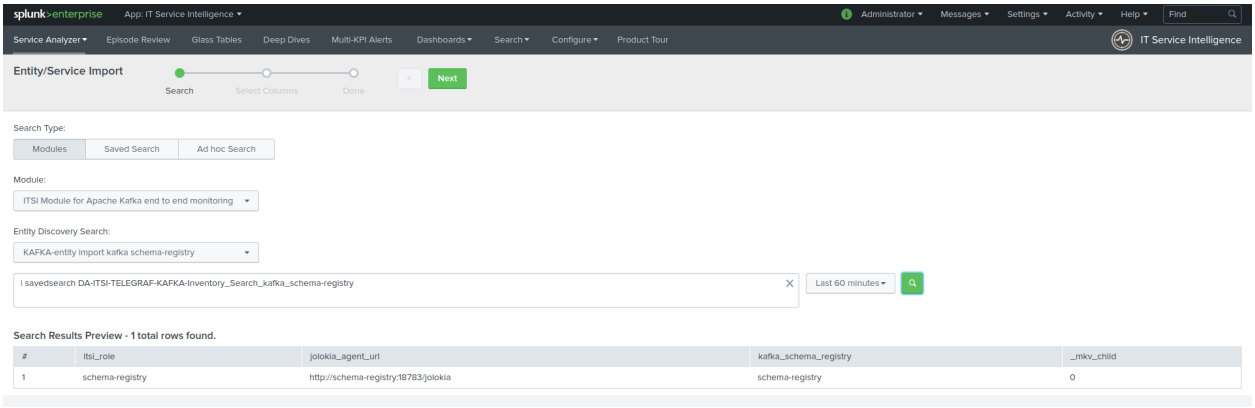
Kafka connect tasks detection



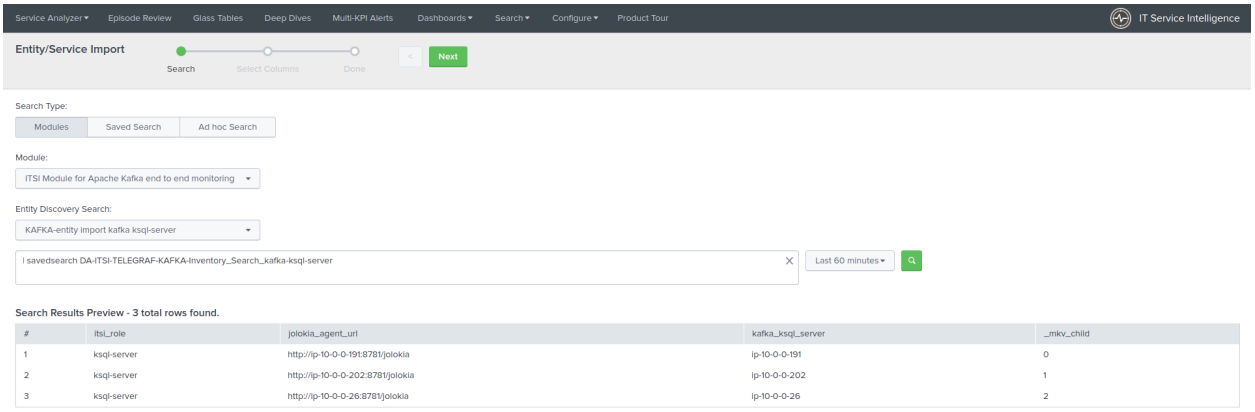
Kafka consumers detection (Burrow)



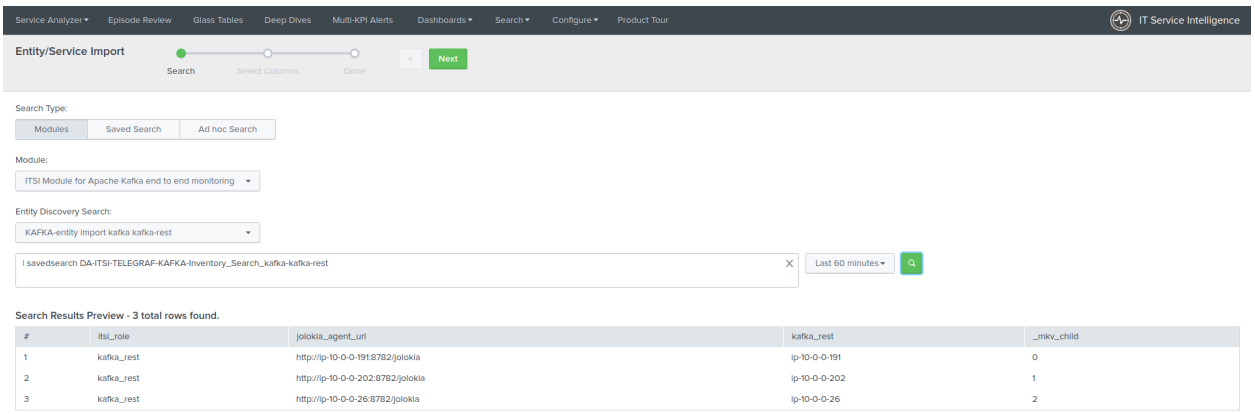
Confluent schema-registry nodes detection



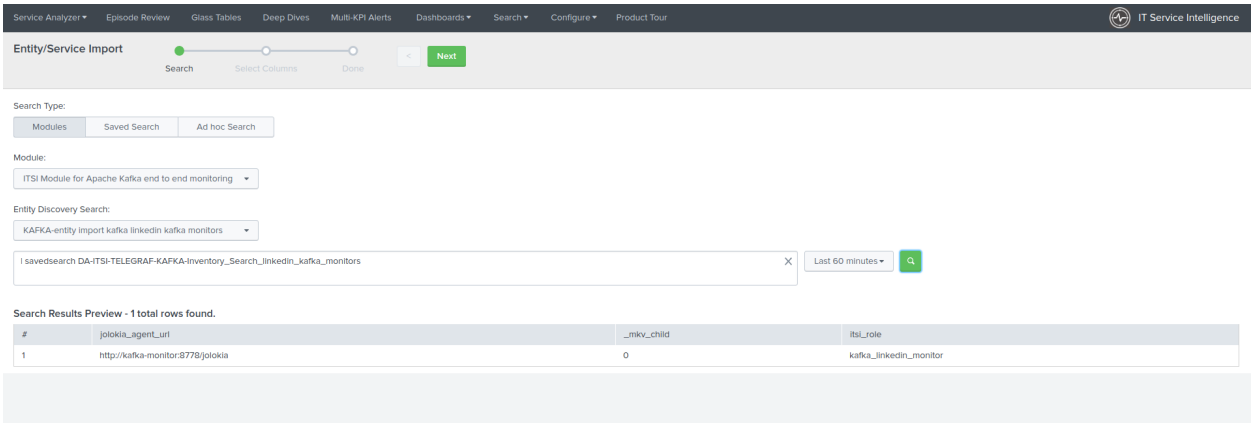
Confluent ksql-server nodes detection



Confluent kafka-rest nodes detection



LinkedIn Kafka monitor nodes detection



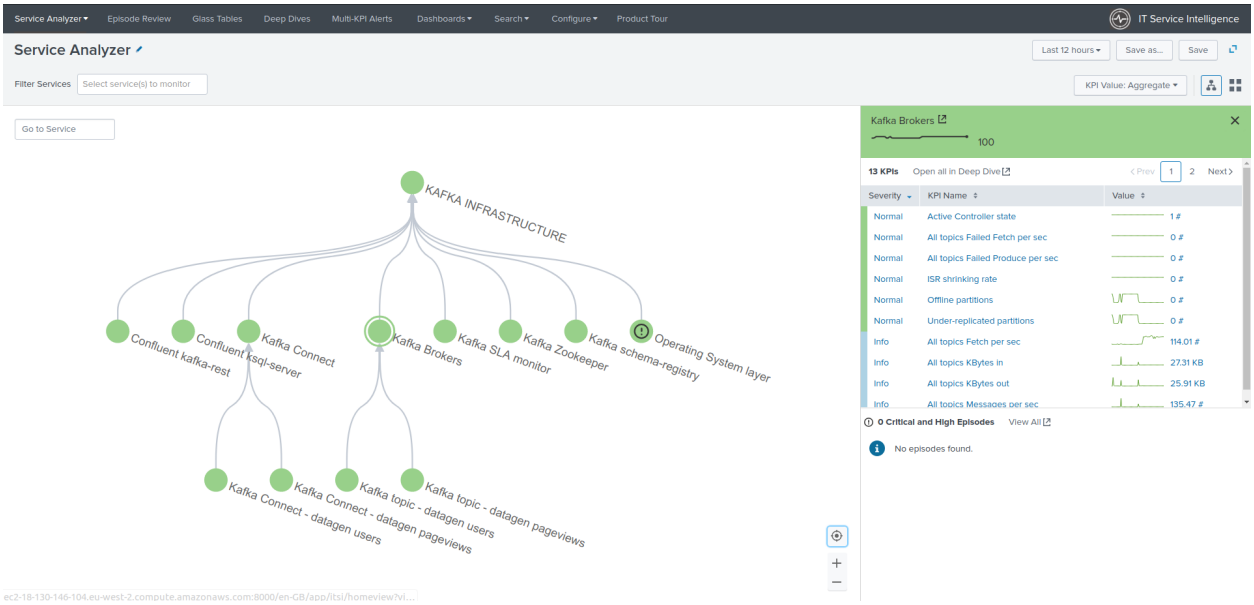
2.5 Services creation

The ITSI module for Telegraf Kafka smart monitoring provides builtin services templates, relying on several base KPIs retrieving data from the metric store.

- **Zookeeper monitoring**
- **Kafka brokers monitoring**
- **Kafka LinkedIn monitor**
- **Kafka topic monitoring**
- **Kafka connect monitoring**
- **Kafka sink task monitoring**
- **Kafka source task monitoring**
- **Kafka Consumers lag monitoring**
- **Confluent schema-registry monitoring**
- **Confluent Confluent ksql-server monitoring**
- **Confluent kafka-rest monitoring**

As a general practice, if you first goal is designing the IT infrastructure in ITSI, a good generic recommendation is to create a main service container for your Kafka infrastructure.

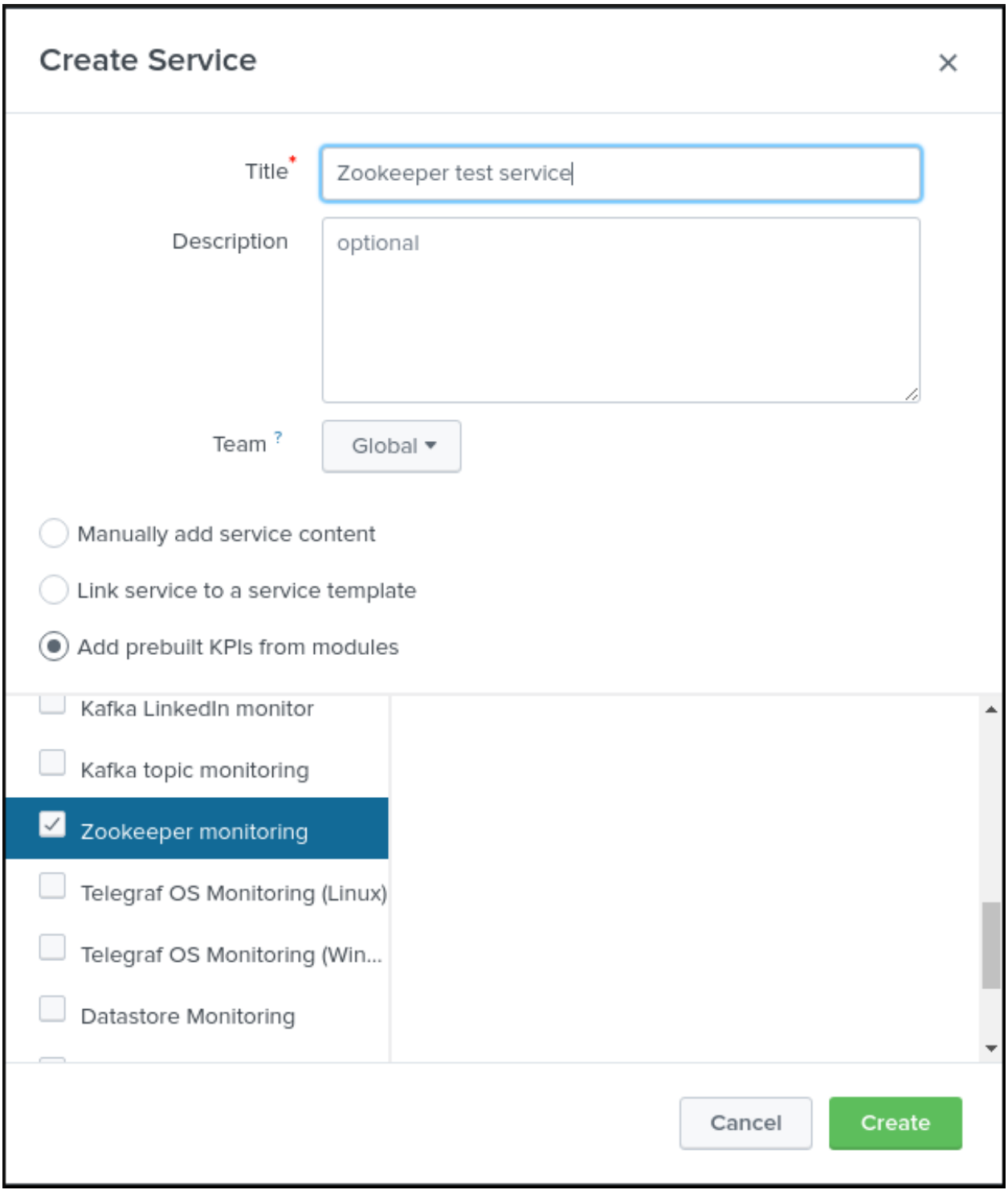
As such, every service that will be designed will be linked to the main service. (the main service depends on them)

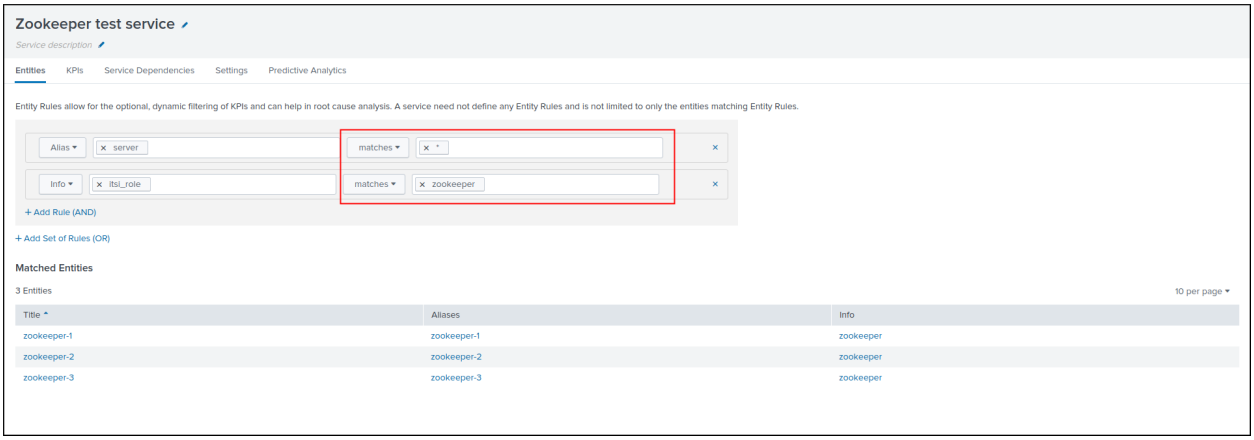


2.5.1 Monitoring Zookeeper servers

To monitor your Zookeeper servers, create a new service using the "Zookeeper monitoring" template service and select the proper filters for your entities:

- Configure / Service / Create new service / Zookeeper monitoring

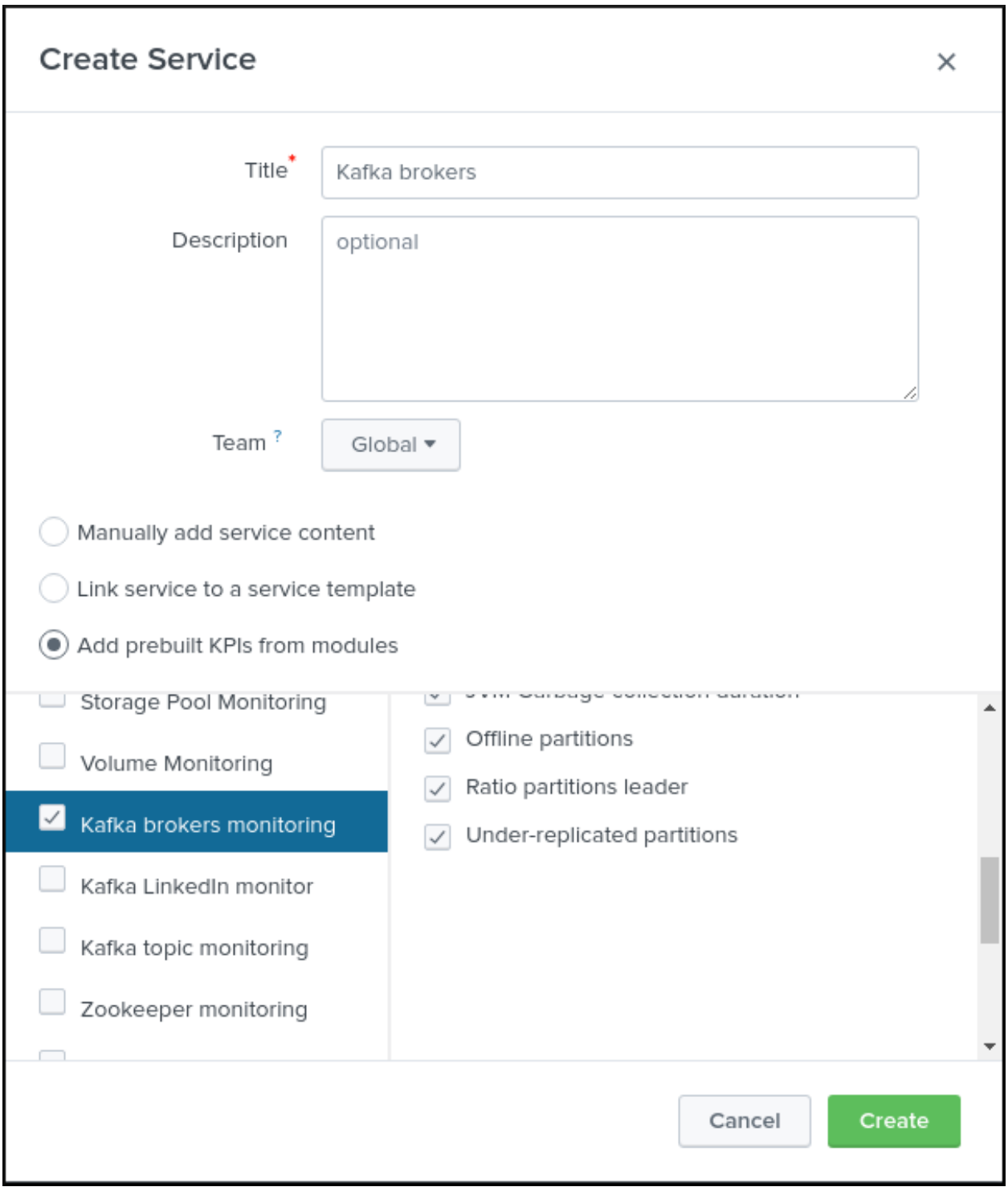

Create Service
Title
Zookeeper test service
Description
optional
Team
Global
Manually add service content
Link service to a service template
Add prebuilt KPIs from modules
Kafka LinkedIn monitor
Kafka topic monitoring
Zookeeper monitoring
Telegraf OS Monitoring (Linux)
Telegraf OS Monitoring (Win...
Datastore Monitoring
Cancel
Create

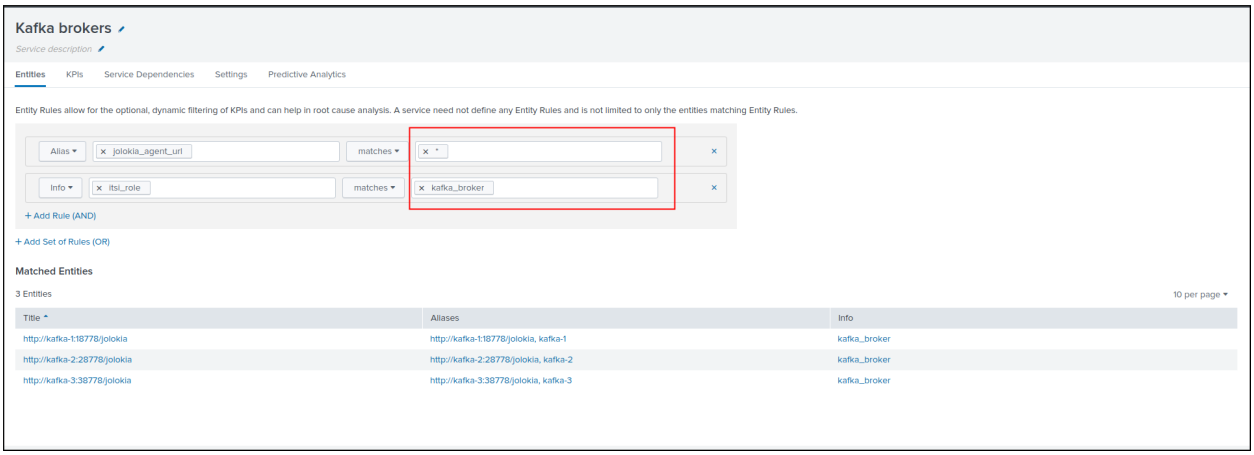


2.5.2 Monitoring Kafka Brokers

To monitor your Zookeeper servers, create a new service using the "Kafka brokers monitoring" template service and select the proper filters for your entities:

- Configure / Service / Create new service / Kafka brokers monitoring

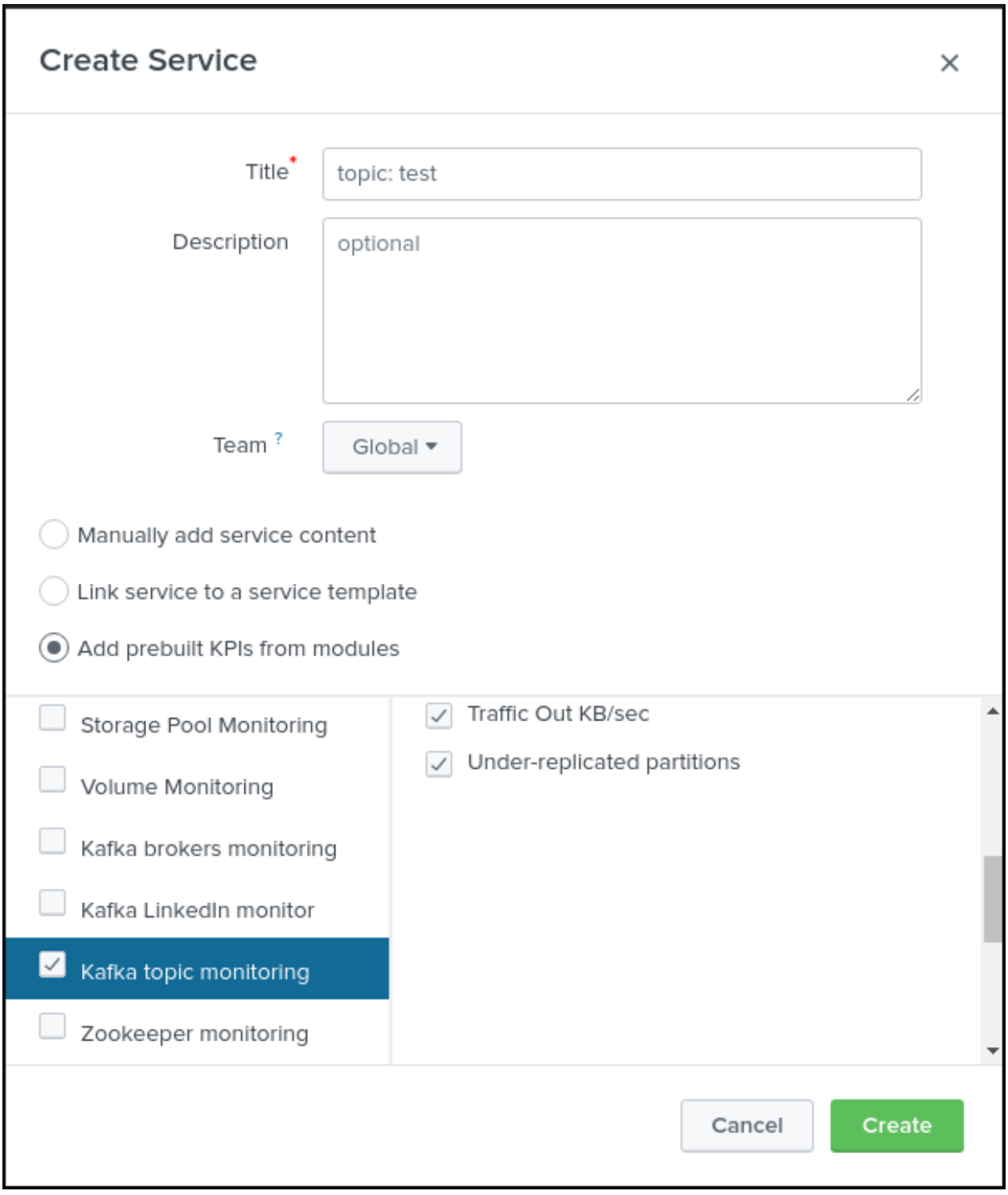

Create Service
Title*
Kafka brokers
Description
optional
Team ?
Global
Manually add service content
Link service to a service template
Add prebuilt KPIs from modules
Storage Pool Monitoring
Volume Monitoring
Kafka brokers monitoring
Kafka LinkedIn monitor
Kafka topic monitoring
Zookeeper monitoring
Offline partitions
Ratio partitions leader
Under-replicated partitions
Cancel
Create

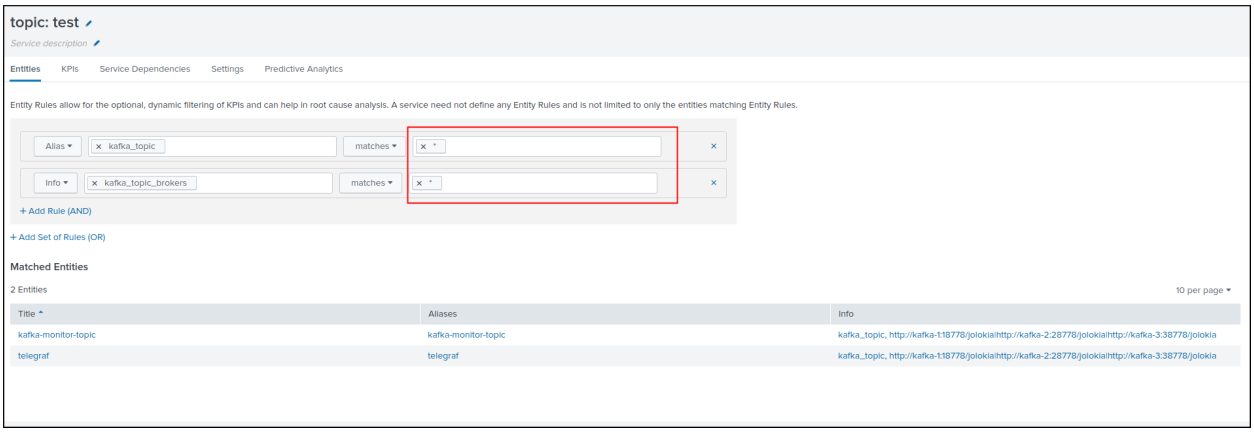


2.5.3 Monitoring Kafka Topics

To monitor one or more Kafka topics, create a new service using the "Kafka topic monitoring" template service and select the proper filters for your entities corresponding to your topics:

- Configure / Service / Create new service / Kafka topic monitoring

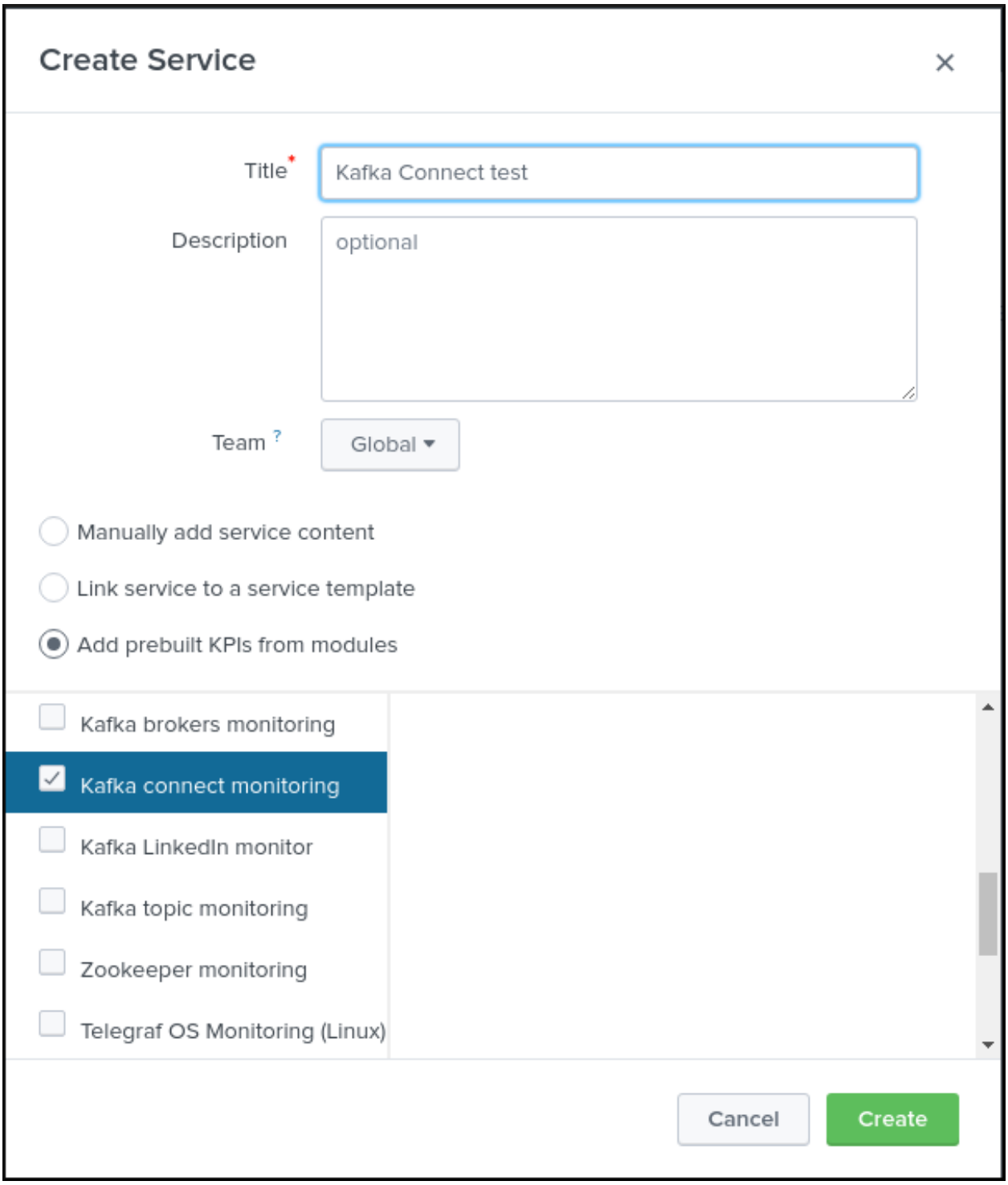

Create Service
Title
topic: test
Description
optional
Team
Global
Manually add service content
Link service to a service template
Add prebuilt KPIs from modules
Storage Pool Monitoring
Volume Monitoring
Kafka brokers monitoring
Kafka LinkedIn monitor
Kafka topic monitoring
Zookeeper monitoring
Traffic Out KB/sec
Under-replicated partitions
Cancel
Create

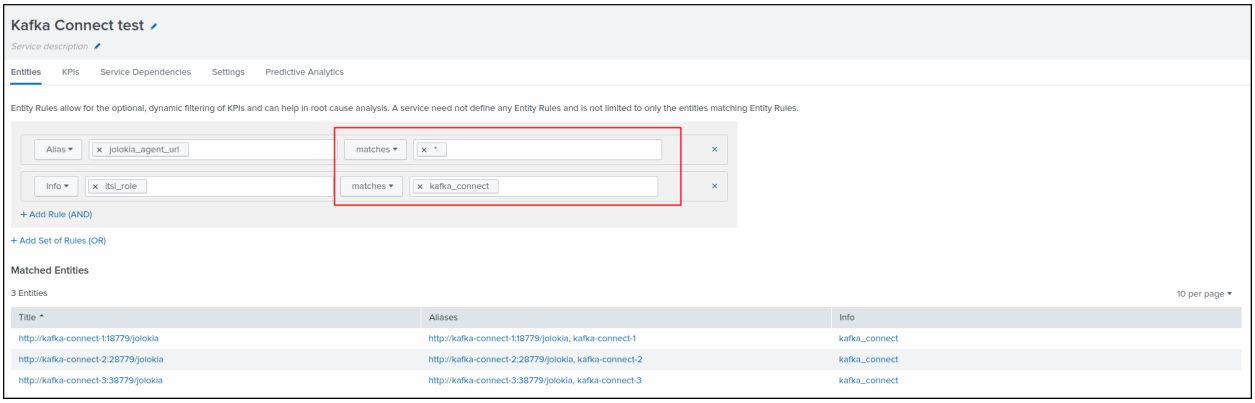


2.5.4 Monitoring Kafka Connect

To monitor Kafka Connect, create a new service using the "Kafka connect monitoring" template service and select the proper filters for your entities:

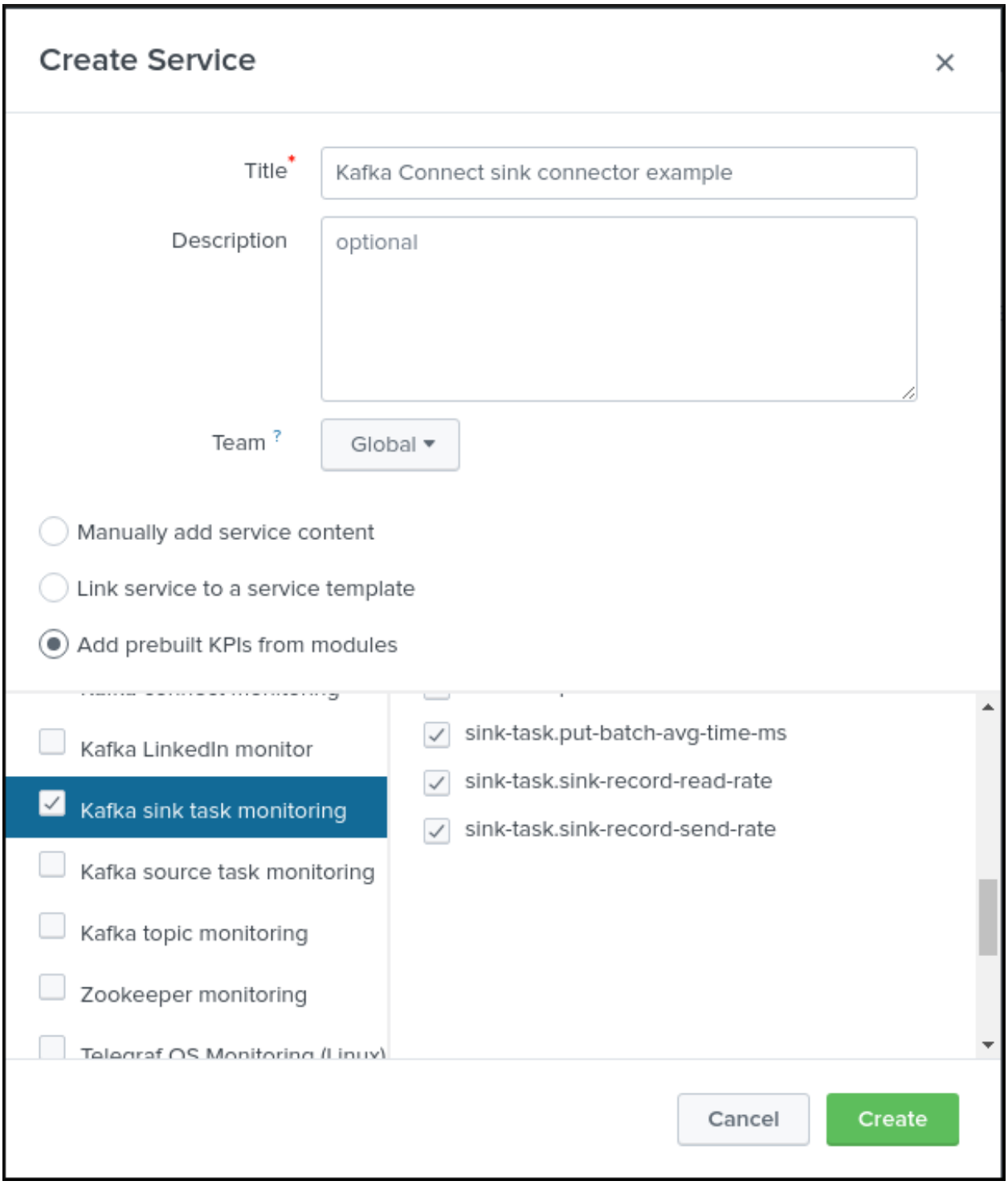
- Configure / Service / Create new service / Kafka connect monitoring

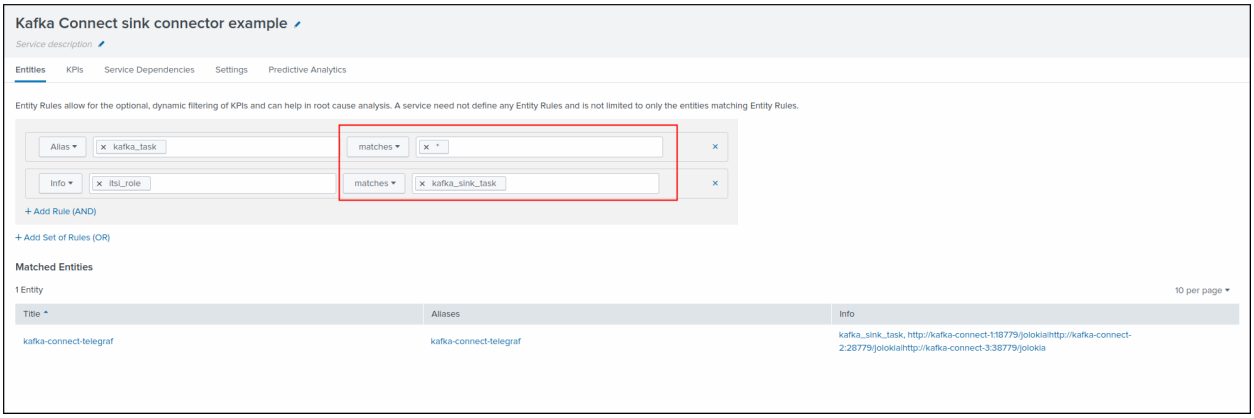

Create Service
Title
Kafka Connect test
Description
optional
Team
Global
Manually add service content
Link service to a service template
Add prebuilt KPIs from modules
Kafka brokers monitoring
Kafka connect monitoring
Kafka LinkedIn monitor
Kafka topic monitoring
Zookeeper monitoring
Telegraf OS Monitoring (Linux)
Cancel
Create



2.5.5 Monitoring Kafka Connect Sink taks

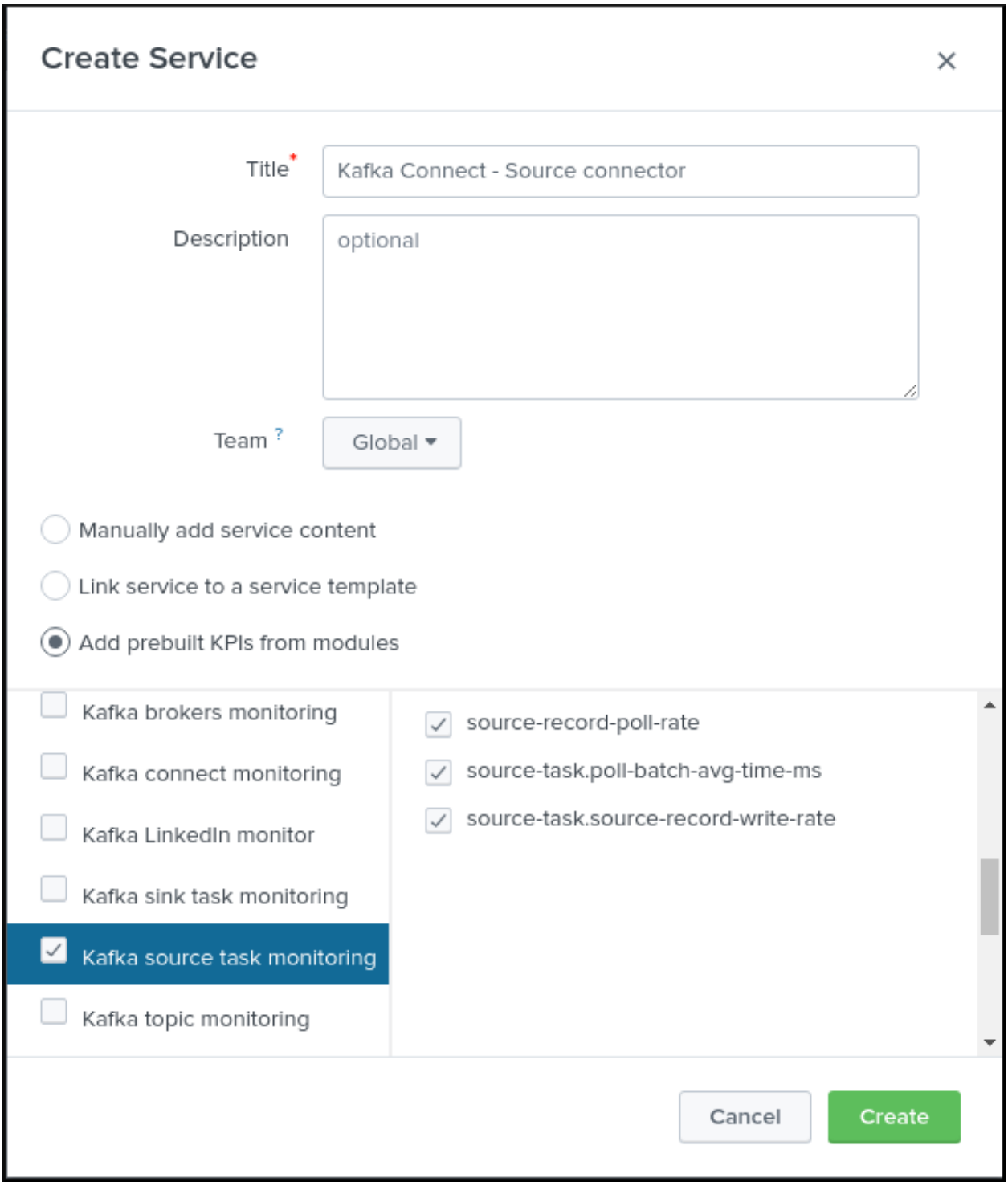
To monitor one of more Kafka Connect Sink connectors, create a new service using the "Kafka sink task monitoring" template service and select the proper filters for your entities:

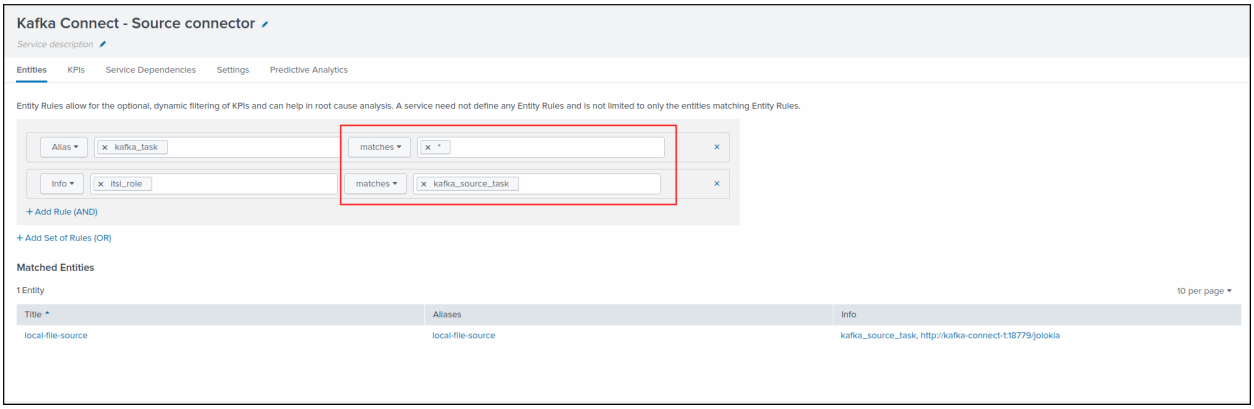

Create Service
Title
Kafka Connect sink connector example
Description
optional
Team
Global
Manually add service content
Link service to a service template
Add prebuilt KPIs from modules
Kafka LinkedIn monitor
Kafka sink task monitoring
Kafka source task monitoring
Kafka topic monitoring
Zookeeper monitoring
sink-task.put-batch-avg-time-ms
sink-task.sink-record-read-rate
sink-task.sink-record-send-rate
Cancel
Create



2.5.6 Monitoring Kafka Connect Source taks

To monitor one of more Kafka Connect Source connectors, create a new service using the "Kafka source task monitoring" template service and select the proper filters for your entities:

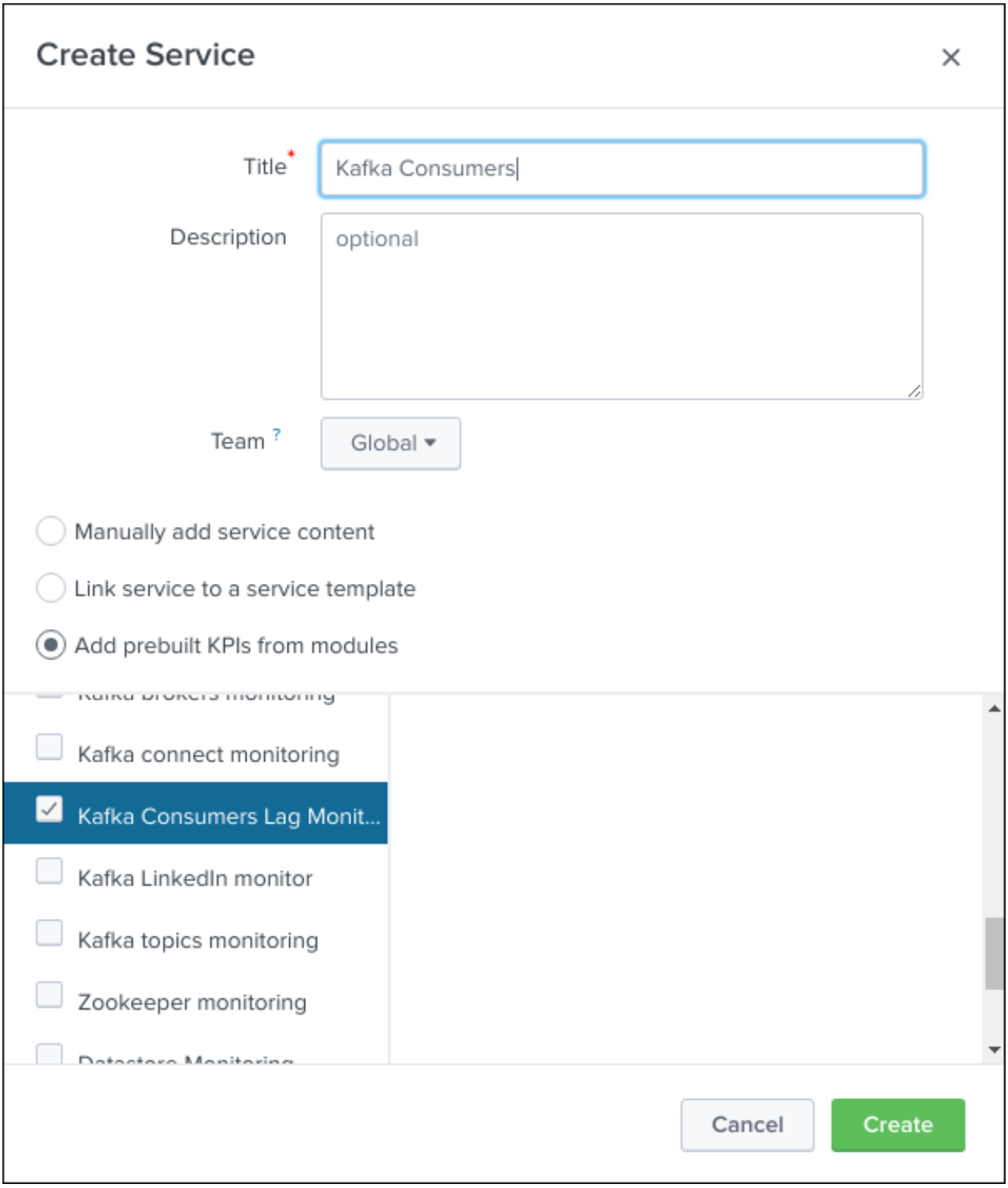

Create Service
Title
Kafka Connect - Source connector
Description
optional
Team
Global
Manually add service content
Link service to a service template
Add prebuilt KPIs from modules
Kafka brokers monitoring
Kafka connect monitoring
Kafka LinkedIn monitor
Kafka sink task monitoring
Kafka source task monitoring
Kafka topic monitoring
source-record-poll-rate
source-task.poll-batch-avg-time-ms
source-task.source-record-write-rate
Cancel
Create

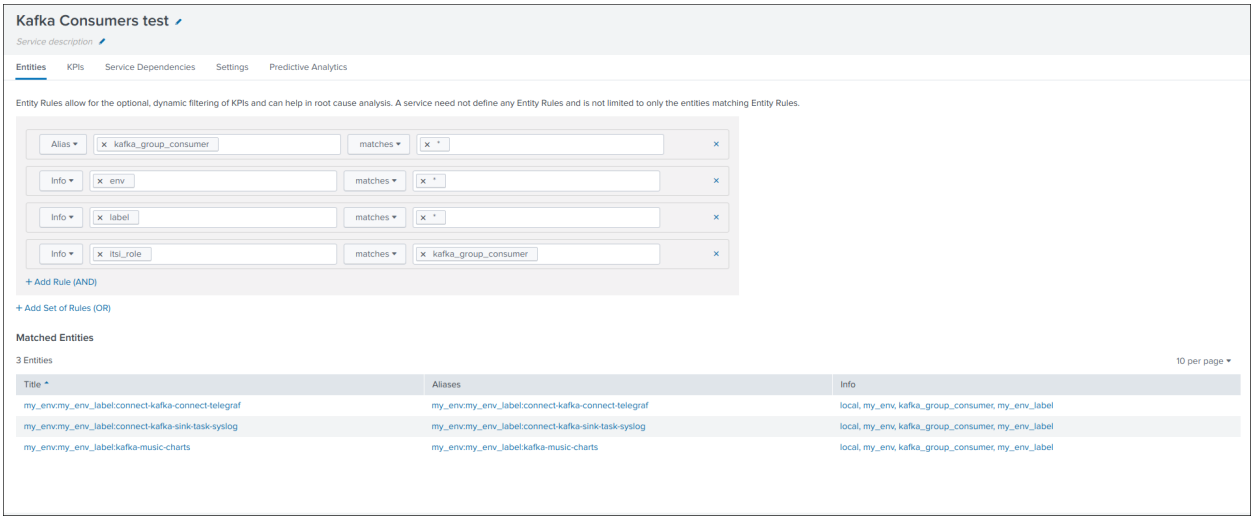


2.5.7 Monitoring Kafka Consumers

To monitor one or more Kafka Consumers, create a new service using the "Kafka Consumers lag monitoring" template service and select the proper filters for your entities corresponding to your topics:

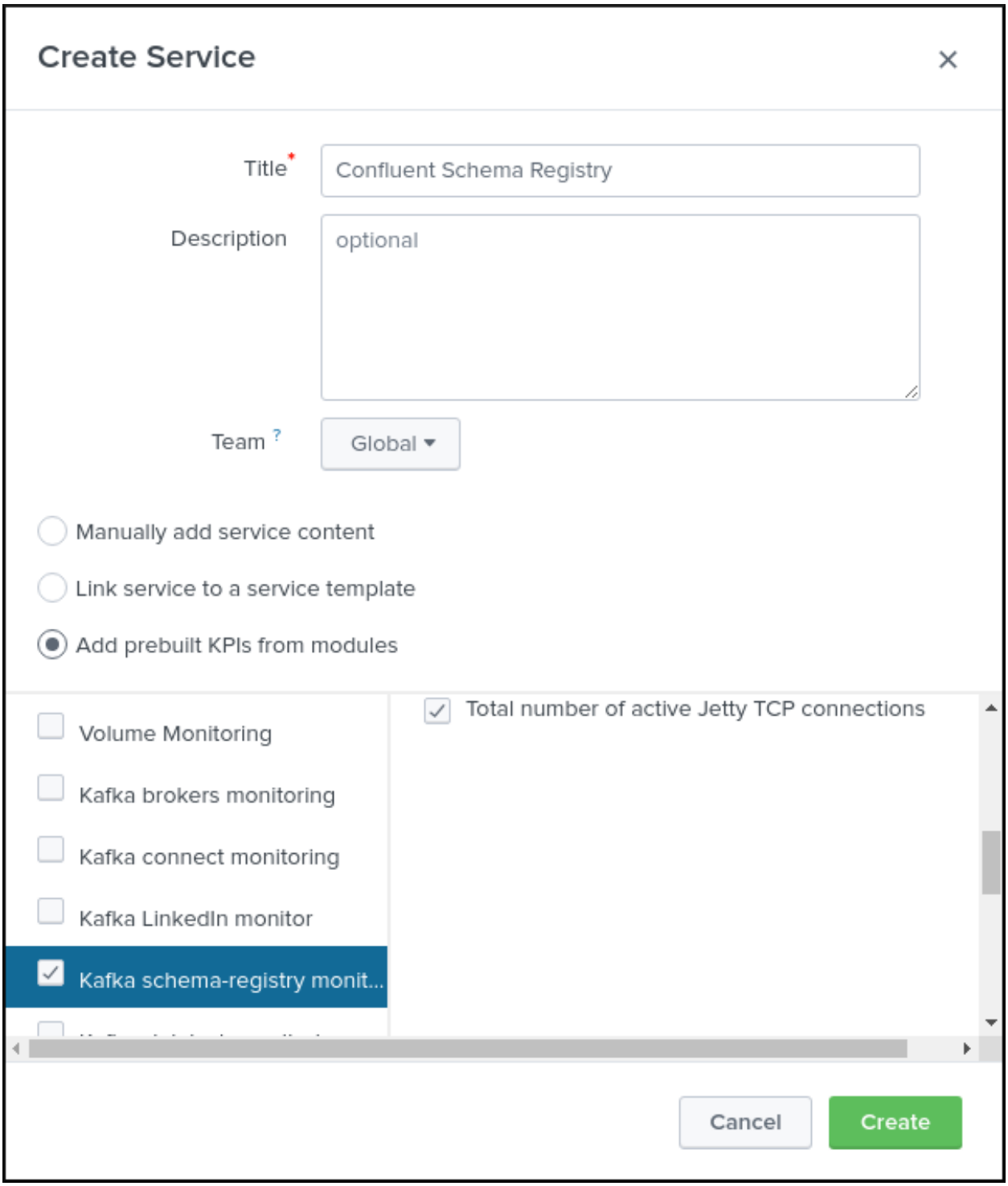
- Configure / Service / Create new service / Kafka lag monitoring

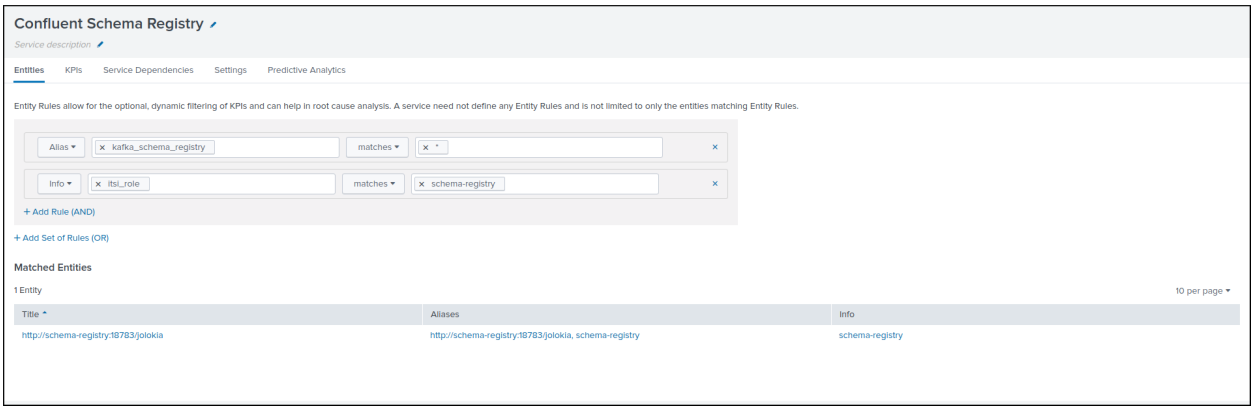

Create Service
Title *
Kafka Consumers
Description
optional
Team ?
Global
Manually add service content
Link service to a service template
Add prebuilt KPIs from modules
Kafka connect monitoring
Kafka Consumers Lag Monit...
Kafka LinkedIn monitor
Kafka topics monitoring
Zookeeper monitoring
Cancel
Create



2.5.8 Monitoring Confluent schema-registry

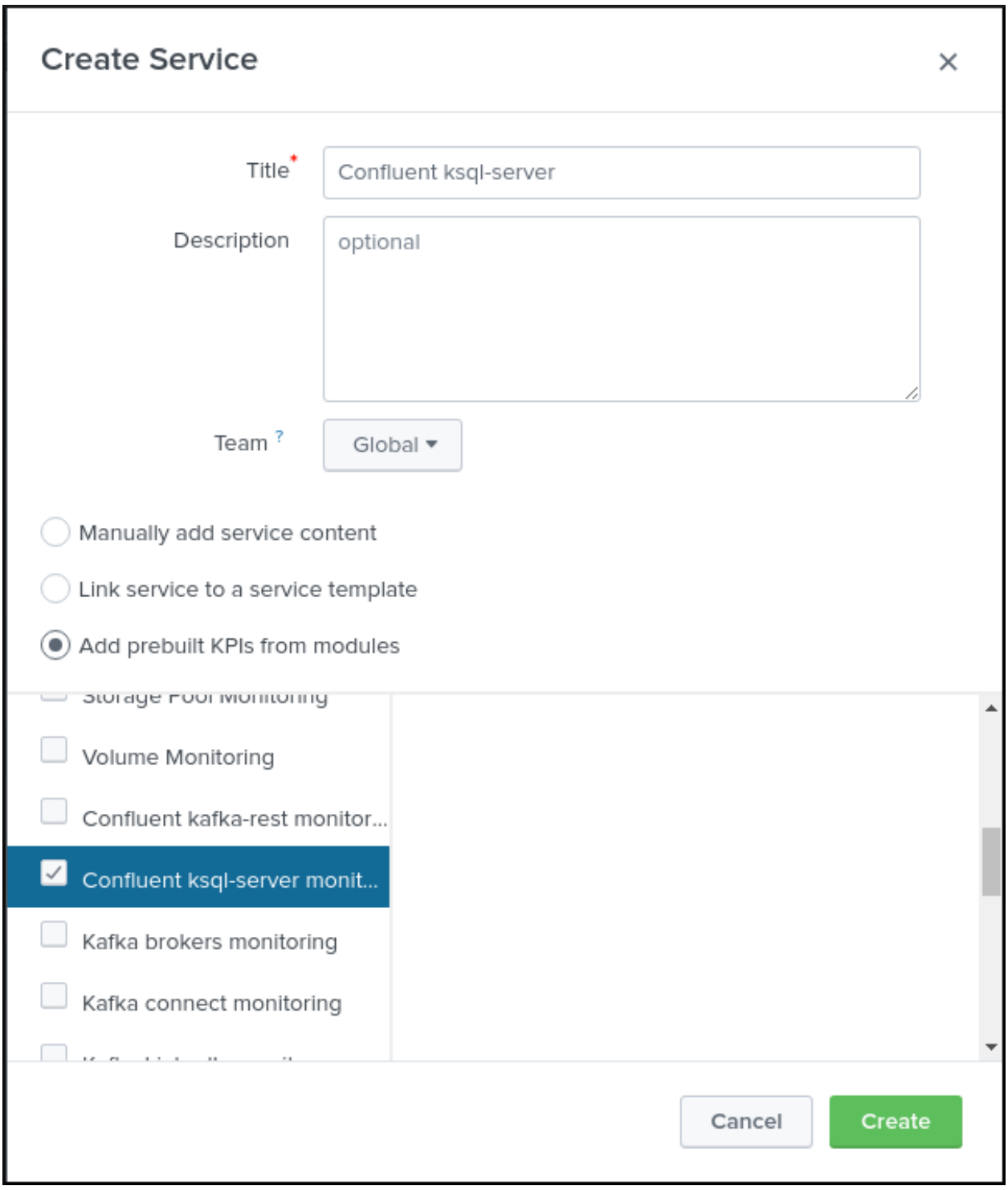
To monitor one of more Confluent schema-registry nodes, create a new service using the "Kafka schema-registry monitoring" template service and select the proper filters for your entities:

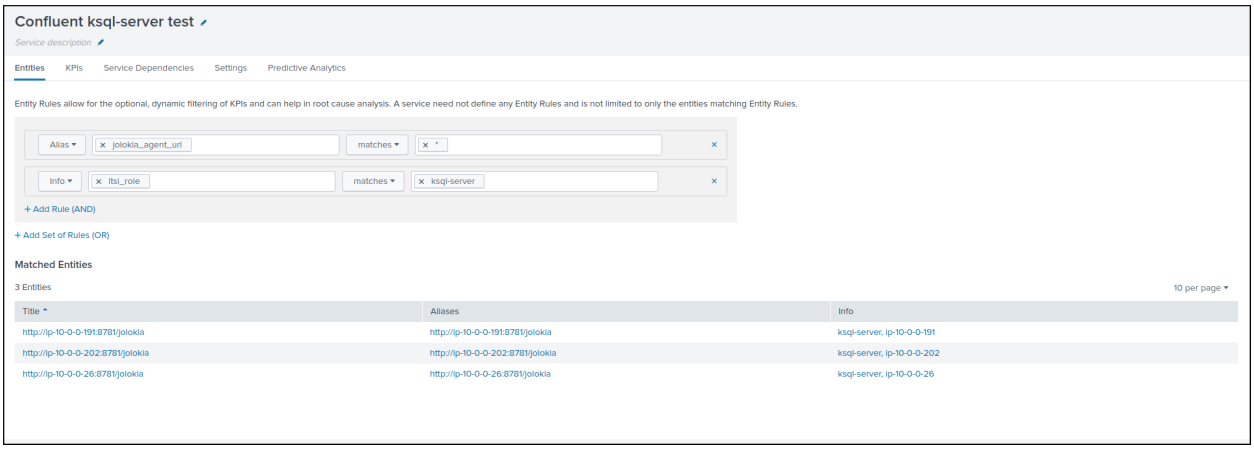

Create Service
Title
Confluent Schema Registry
Description
optional
Team
Global
Manually add service content
Link service to a service template
Add prebuilt KPIs from modules
Volume Monitoring
Kafka brokers monitoring
Kafka connect monitoring
Kafka LinkedIn monitor
Kafka schema-registry monit...
Total number of active Jetty TCP connections
Cancel
Create



2.5.9 Monitoring Confluent ksql-server

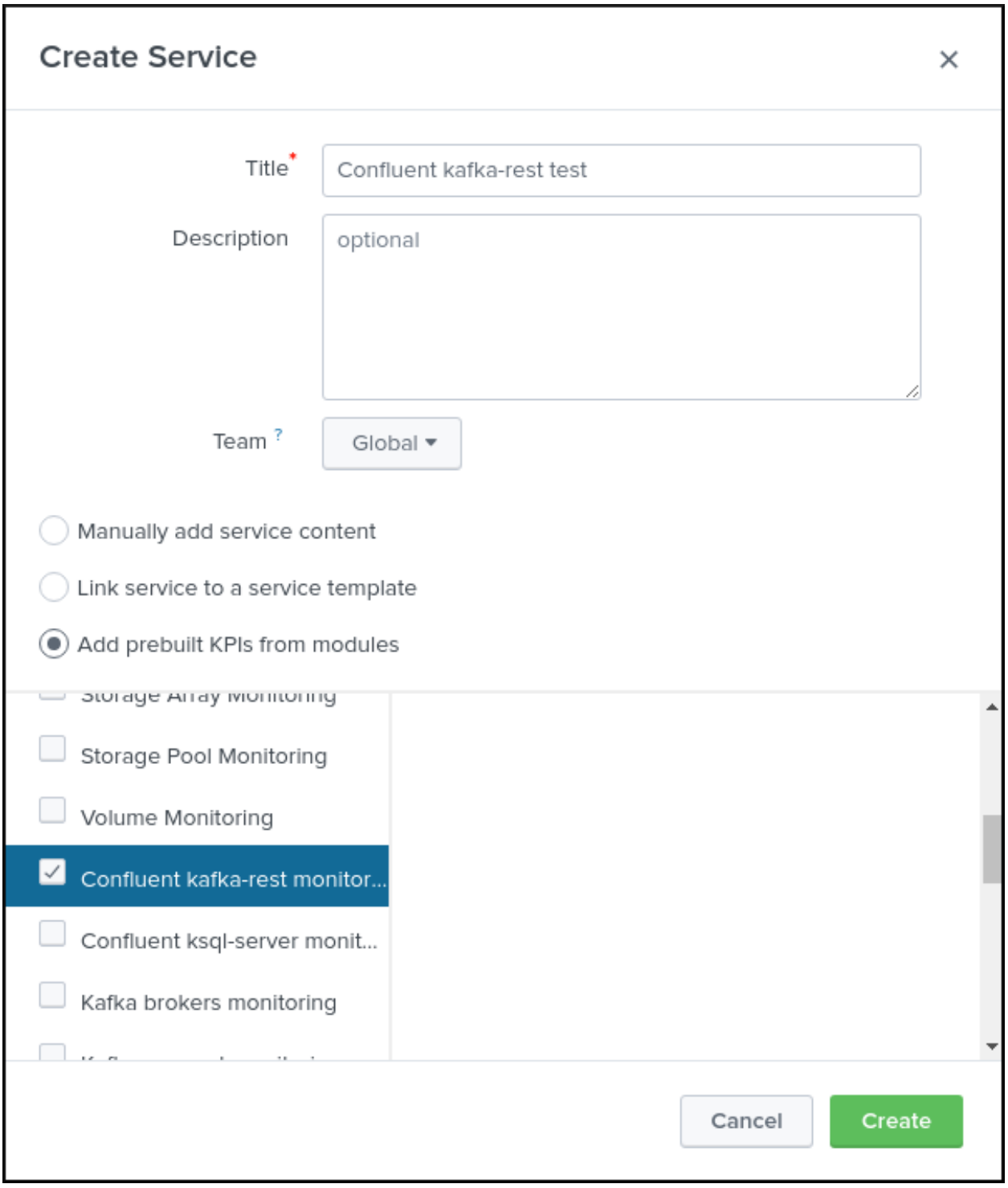
To monitor one of more Confluent ksql servers, create a new service using the "Confluent ksql-server monitoring" template service and select the proper filters for your entities:

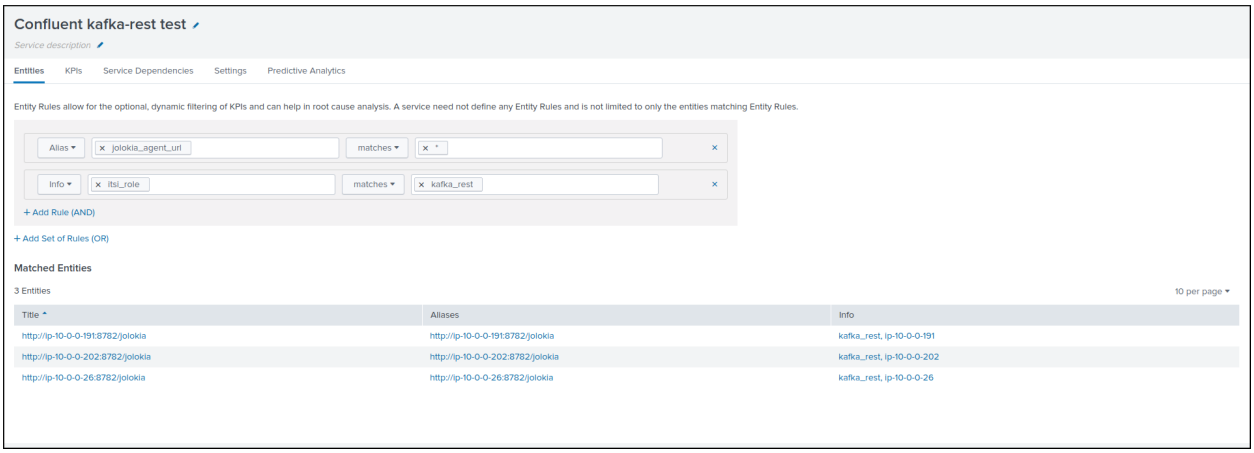

Create Service
Title*
Confluent ksql-server
Description
optional
Team ?
Global
Manually add service content
Link service to a service template
Add prebuilt KPIs from modules
Volume Monitoring
Confluent kafka-rest monitor...
Confluent ksql-server monit...
Kafka brokers monitoring
Kafka connect monitoring
Cancel
Create



2.5.10 Monitoring Confluent kafka-rest

To monitor one of more Confluent kafka-rest nodes, create a new service using the "Confluent kafka-rest monitoring" template service and select the proper filters for your entities:

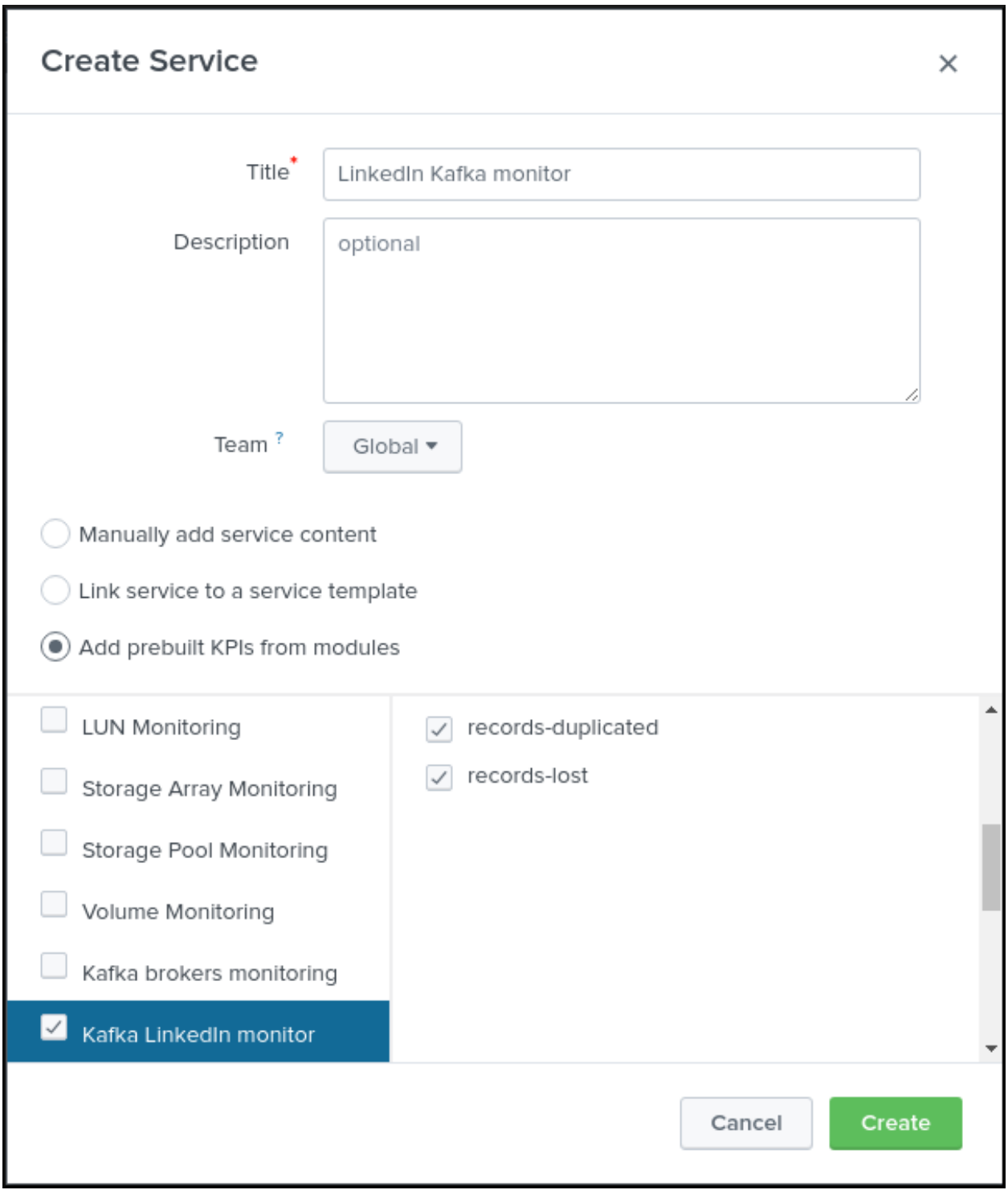

Create Service
Title*
Confluent kafka-rest test
Description
optional
Team ?
Global
Manually add service content
Link service to a service template
Add prebuilt KPIs from modules
Storage Pool Monitoring
Volume Monitoring
Confluent kafka-rest monitor...
Confluent ksql-server monit...
Kafka brokers monitoring
Cancel
Create

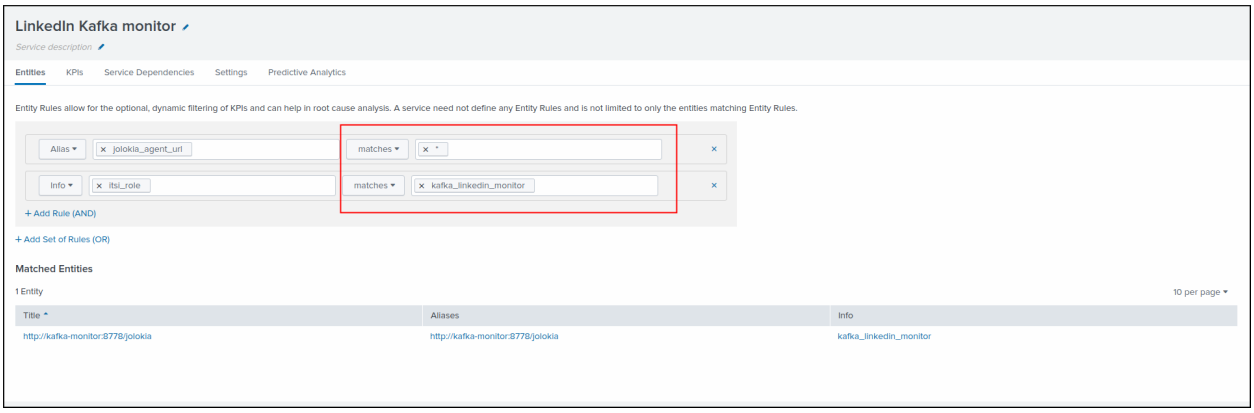


2.5.11 End to end monitoring with LinkedIn Kafka monitor

To monitor your Kafka deployment using the LinkedIn Kafka monitor, create a new service using the "Kafka LinkedIn monitor" template service and select the proper filters for your entities:

- Configure / Service / Create new service / Kafka LinkedIn monitor


Create Service
Title*
LinkedIn Kafka monitor
Description
optional
Team ?
Global
Manually add service content
Link service to a service template
Add prebuilt KPIs from modules
LUN Monitoring
Storage Array Monitoring
Storage Pool Monitoring
Volume Monitoring
Kafka brokers monitoring
Kafka LinkedIn monitor
records-duplicated
records-lost
Cancel
Create

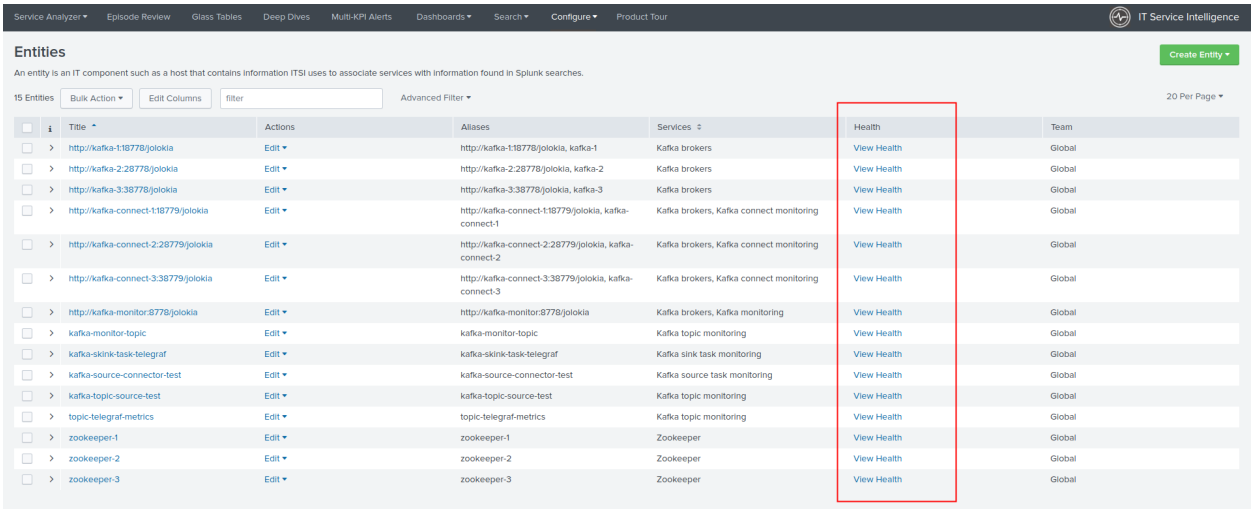


2.6 ITSI Entities dashboard (health views)

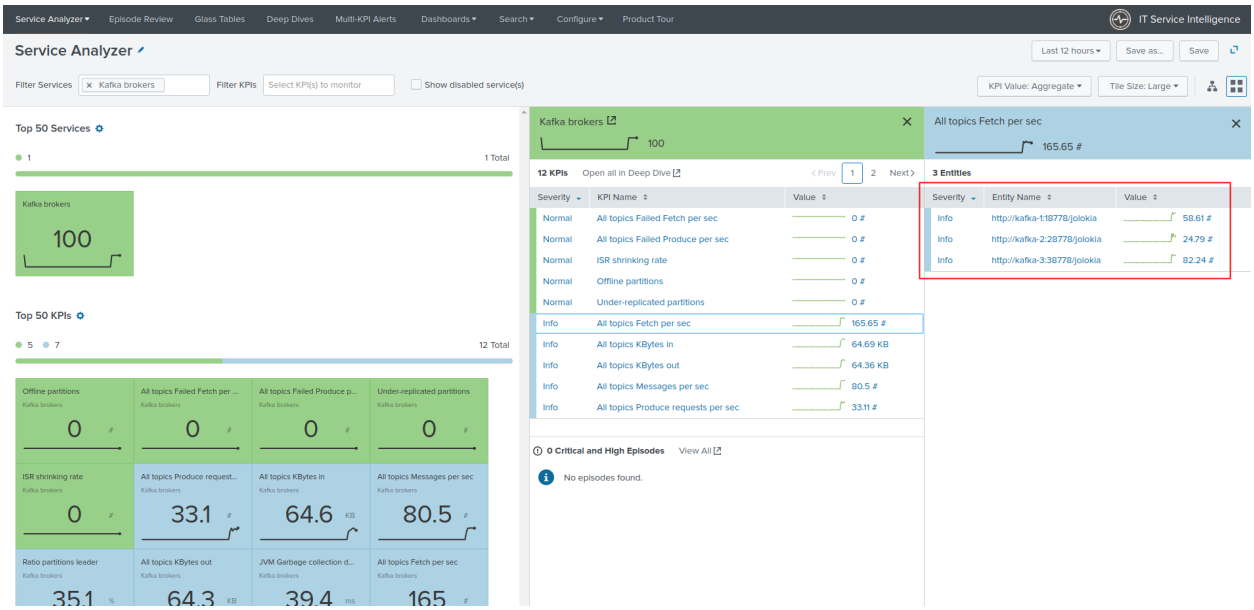
Through builtin ITSI deepdive links, you can automatically and easily access to an efficient dashboard that provides insight analytic for the component.

Accessing entities health views is a native ITSI feature, either by:

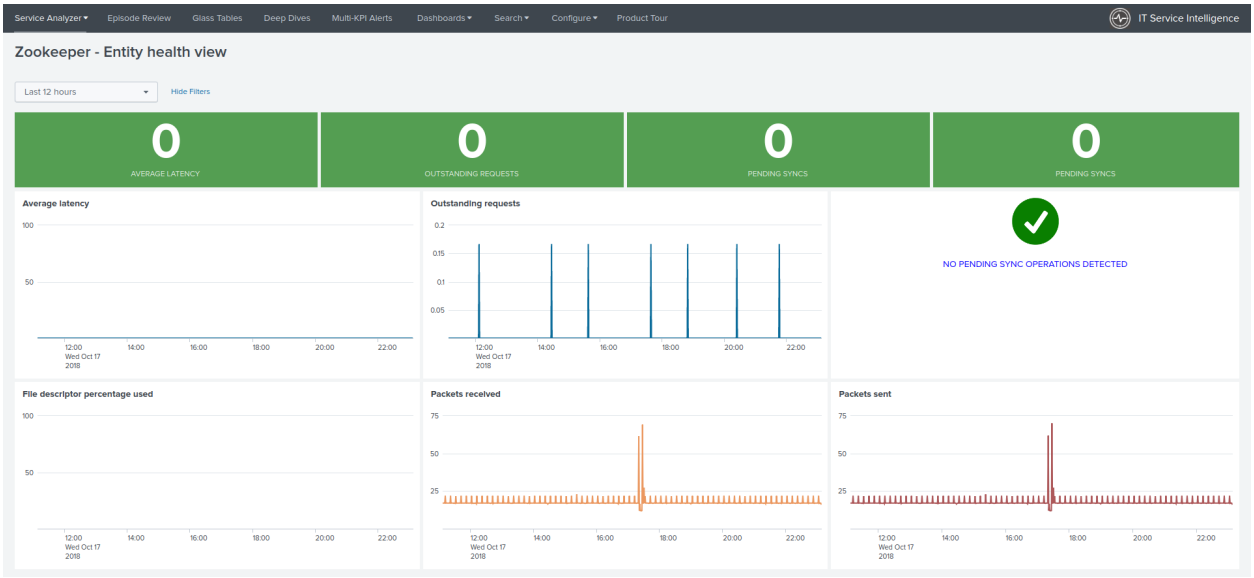
- **Entities lister** (Configure / Entities)



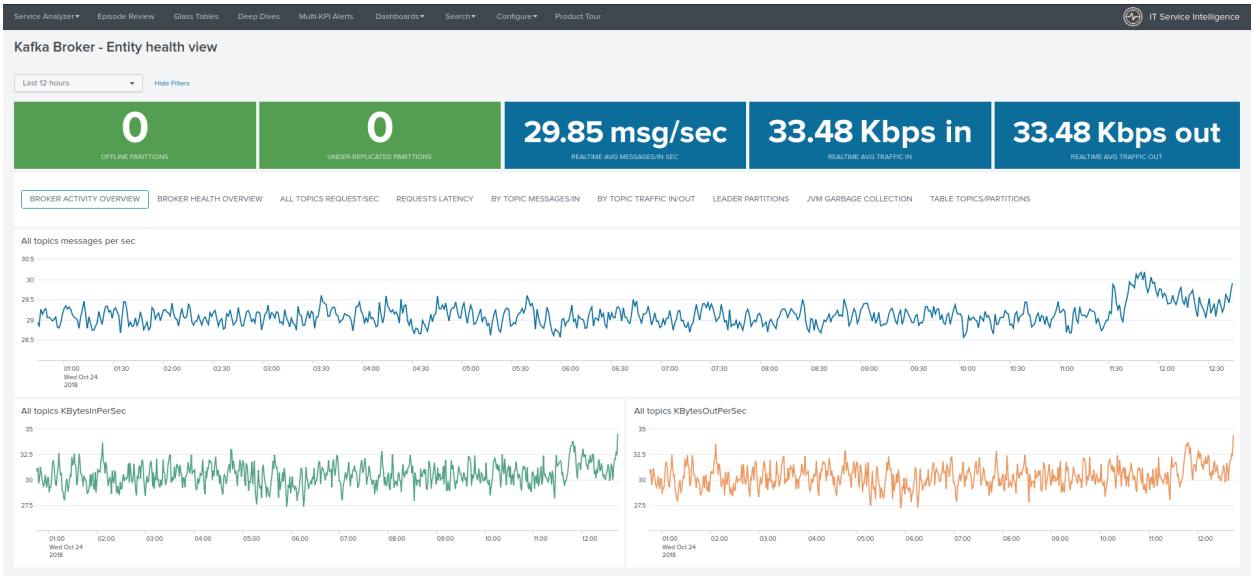
- **deepdive link**

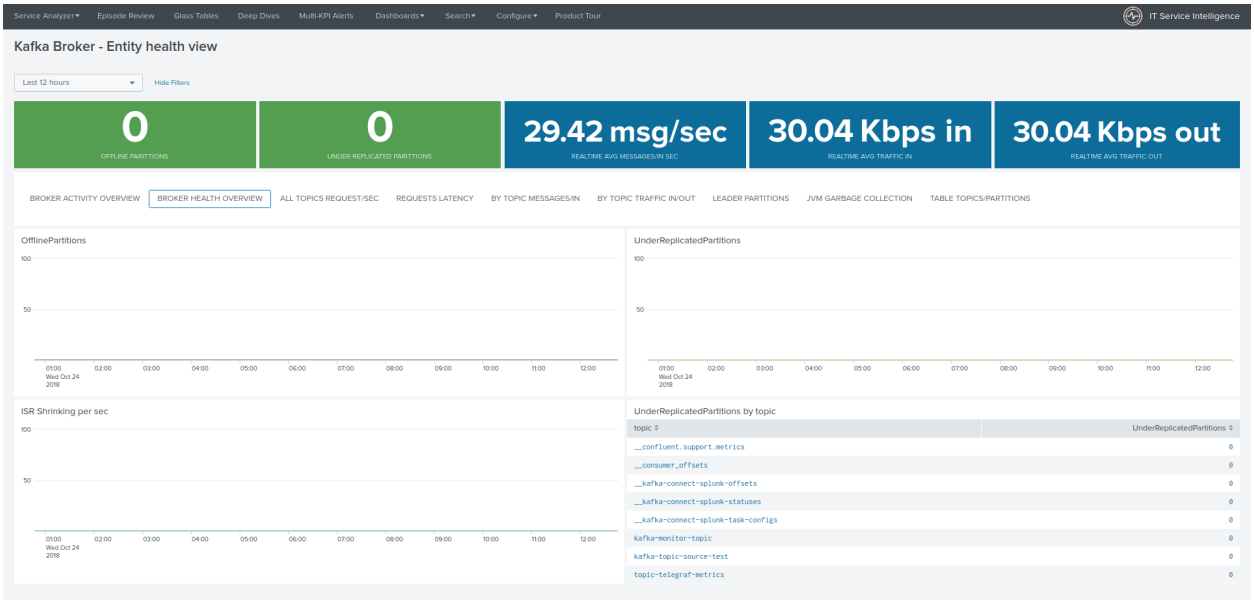


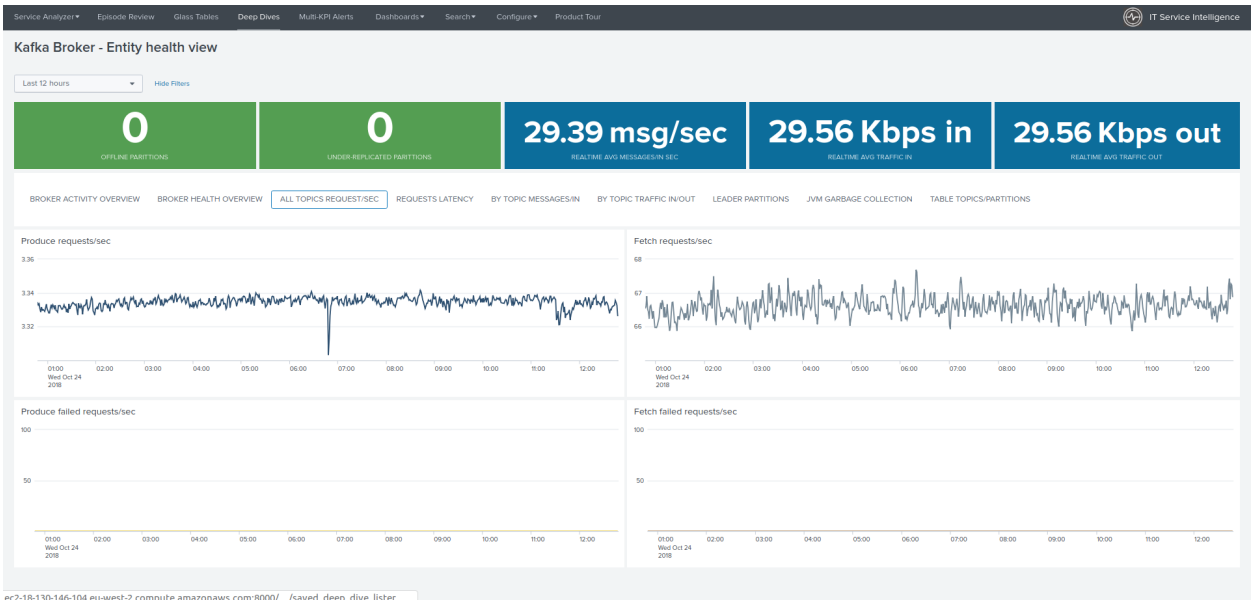
2.6.1 Zookeeper dashboard view

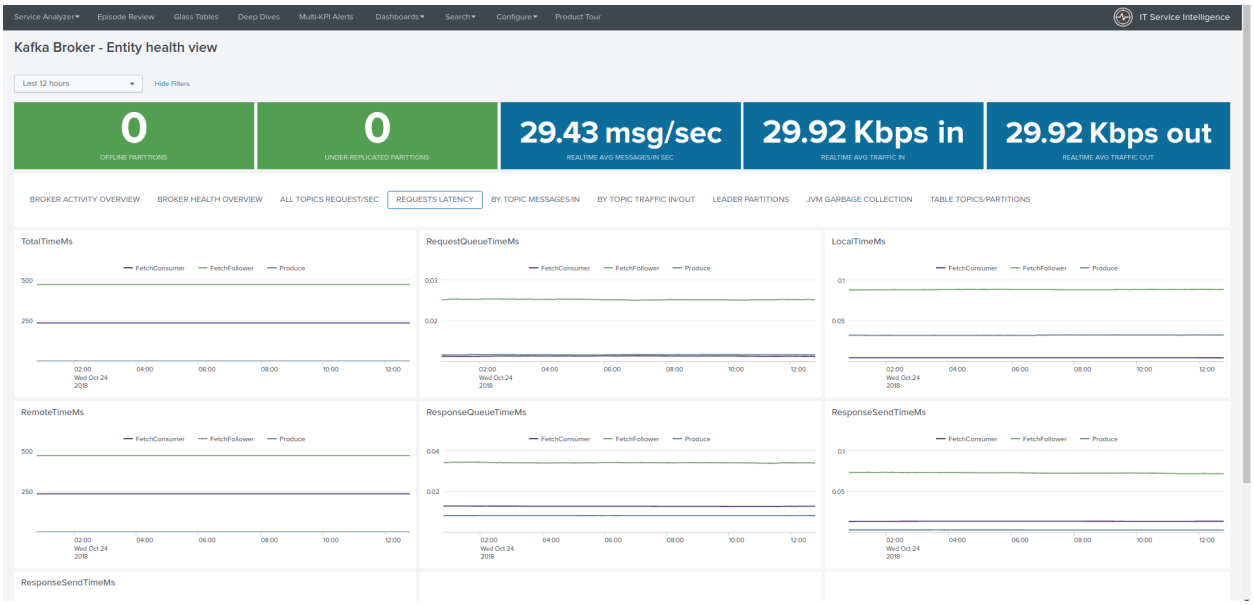


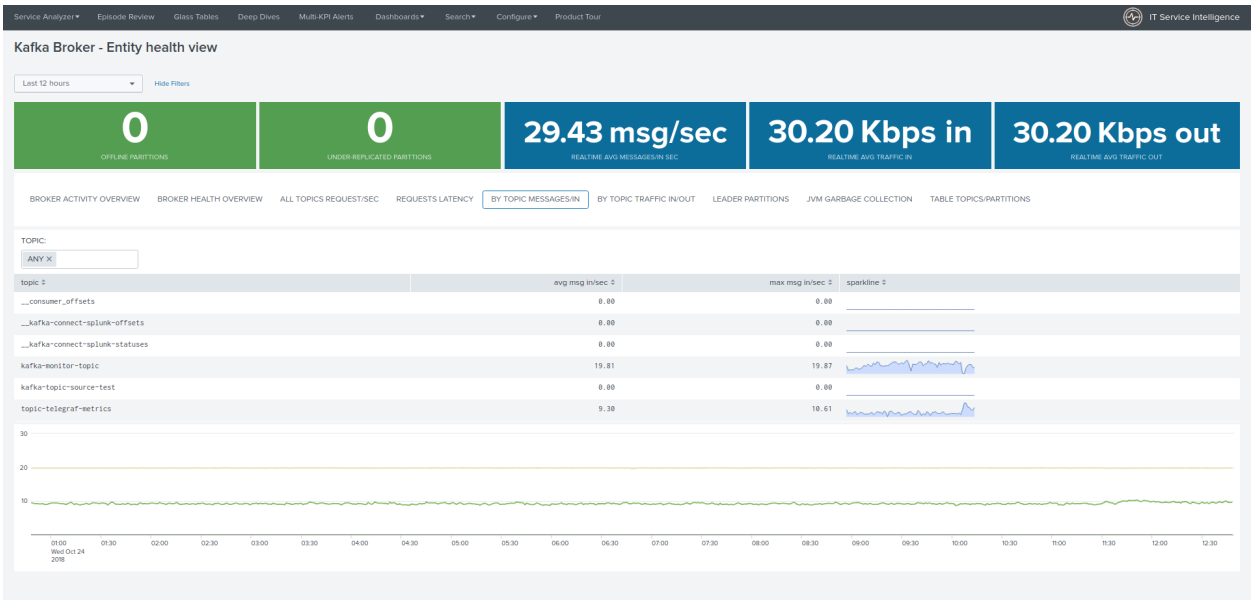
2.6.2 Kafka broker dashboard view

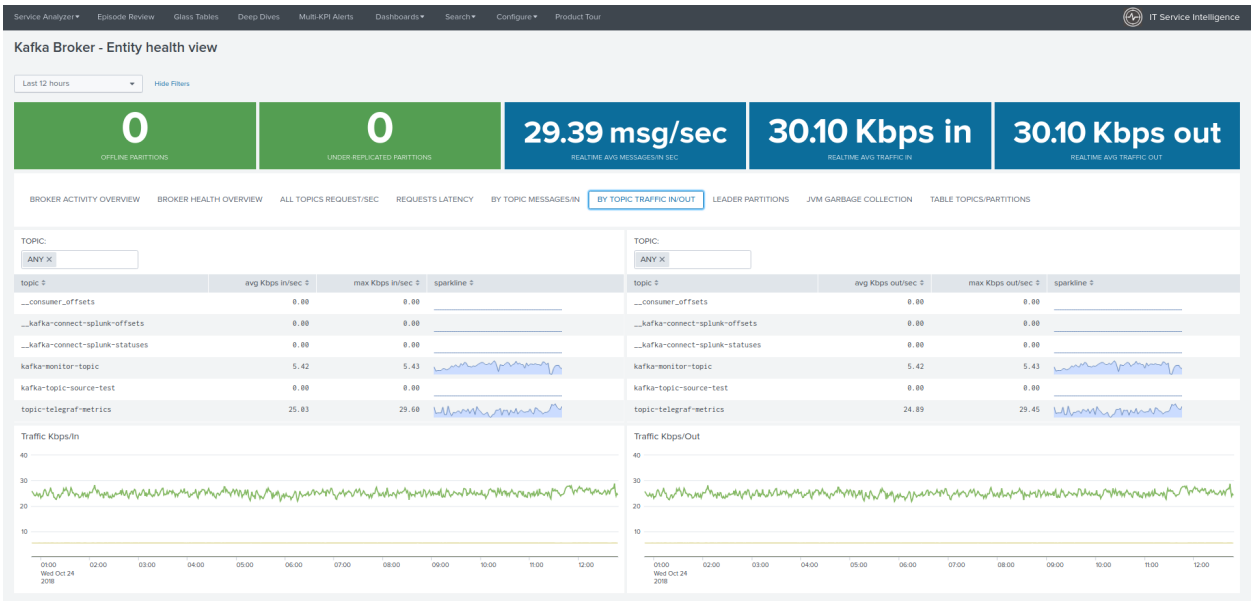


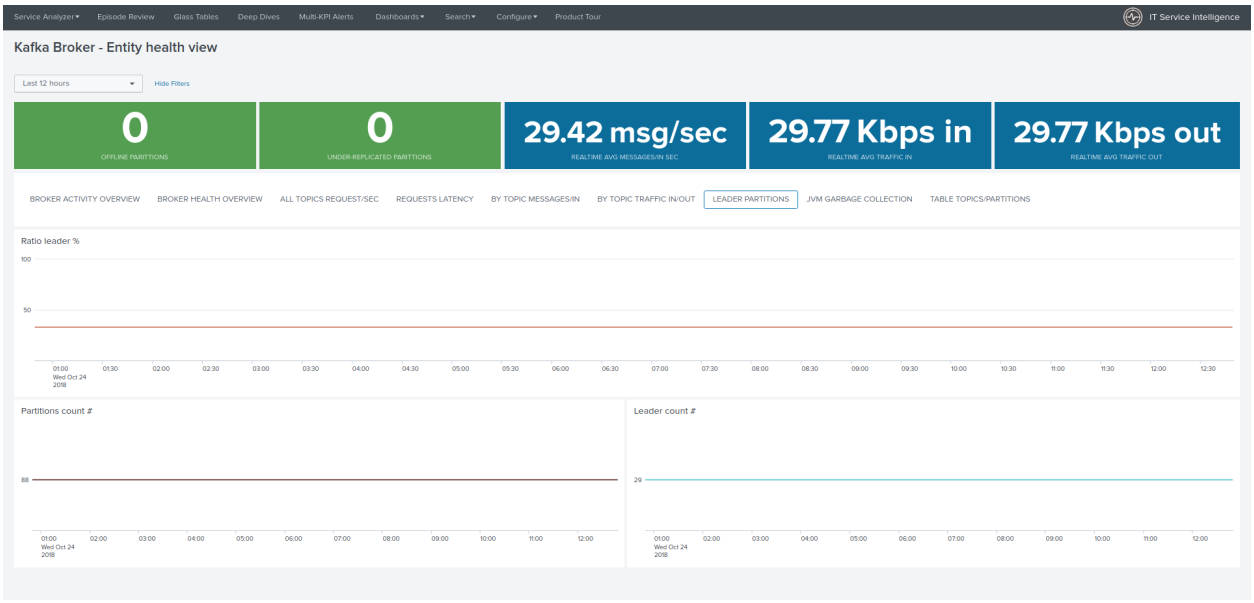


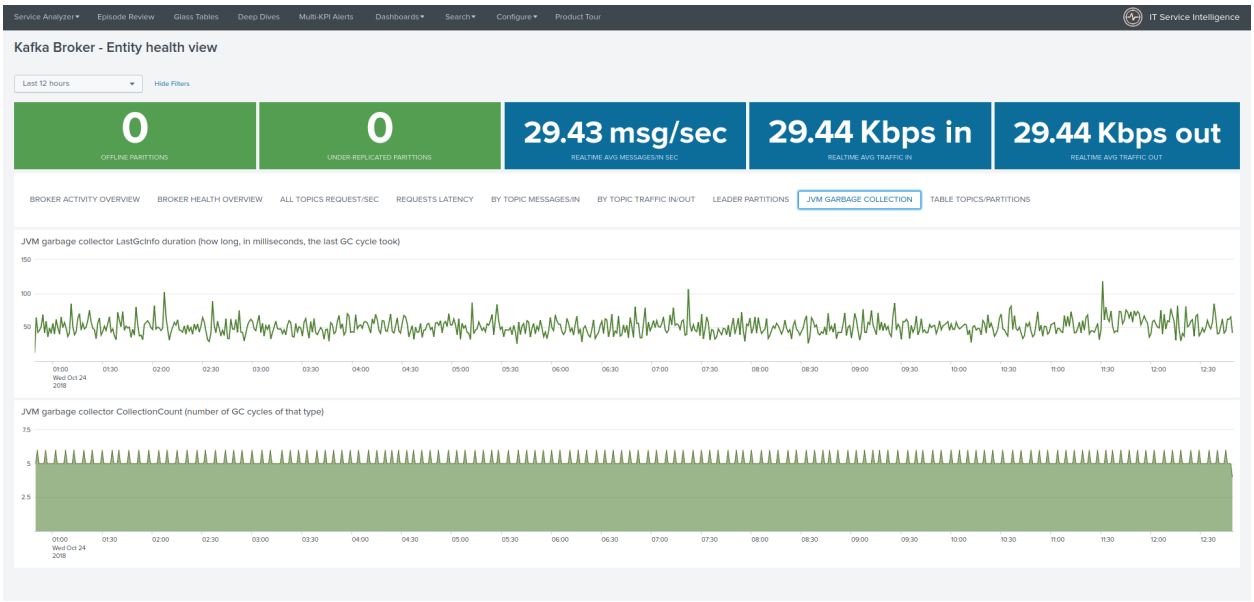

Service Analyzer
Episode Review
Glass Tables
Deep Dives
Multi-KPI Alerts
Dashboards
Search
Configure
Product Tour
IT Service Intelligence
Kafka Broker - Entity health view
Last 12 hours
Hide Filters
0
OFFLINE PARITTIONS
0
UNDER-REPLICATED PARITTIONS
29.39 msg/sec
REALTIME AVG MESSAGES/IN SEC
29.56 Kbps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

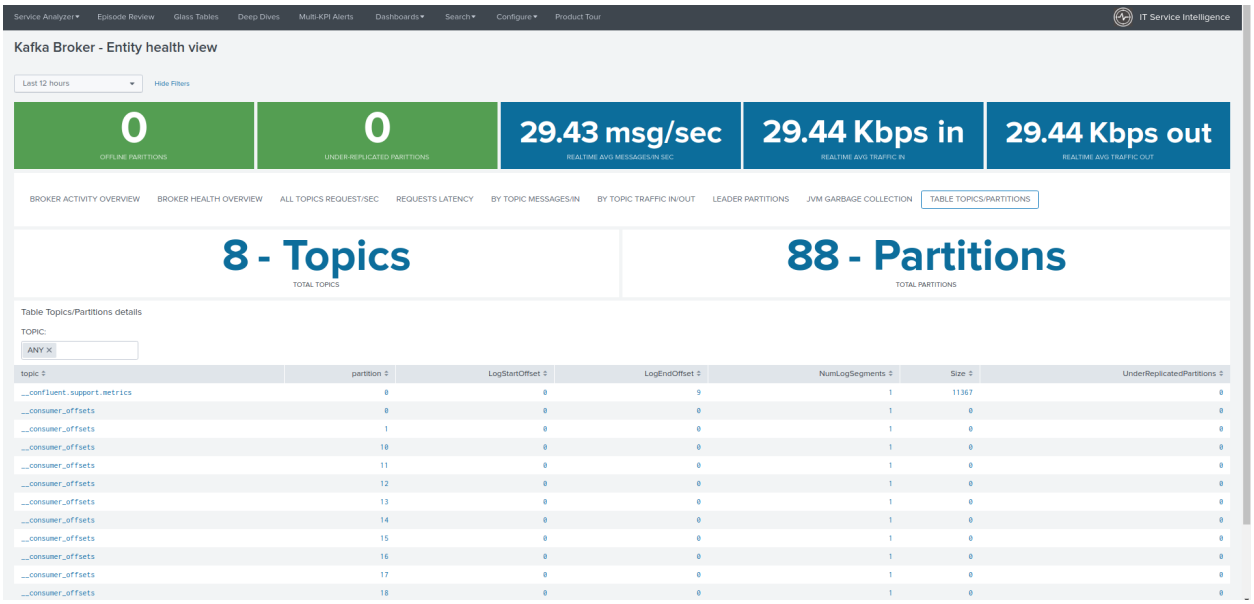

Service Analyzer
Episode Review
Glass Tables
Deep Dives
Multi-KPI Alerts
Dashboards
Search
Configure
Product Tour
IT Service Intelligence
Kafka Broker - Entity health view
Last 12 hours
Hide Filters
0
OFFLINE PARITTIONS
0
UNDER-REPLICATED PARITTIONS
29.43 msg/sec
REALTIME AVG MESSAGES/IN SEC
29.92 Kbps 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
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 PARITIONS
0
UNDER-REPLICATED PARTITIONS
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

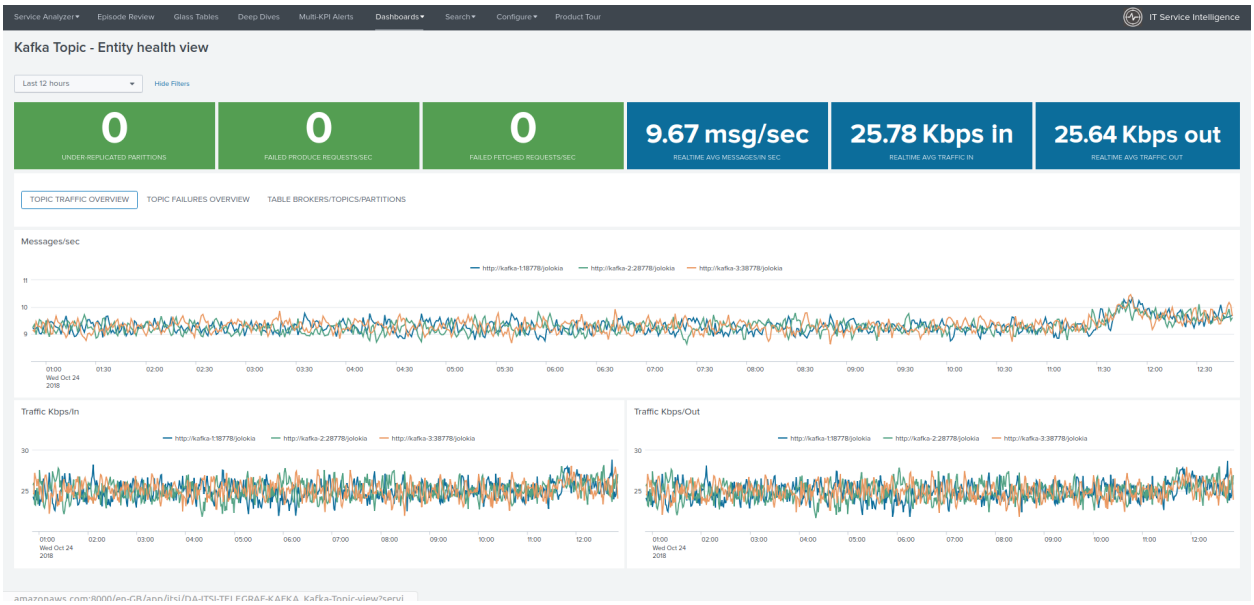

Service Analyzer
Episode Review
Glass Tables
Deep Dives
Multi-KPI Alerts
Dashboards
Search
Configure
Product Tour
IT Service Intelligence
Kafka Broker - Entity health view
Last 12 hours
Hide Filters
0
OFFLINE PARITIONS
0
UNDER-REPLICATED PARTITIONS
29.39 msg/sec
REALTIME AVG MESSAGES/IN SEC
30.10 Kbps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

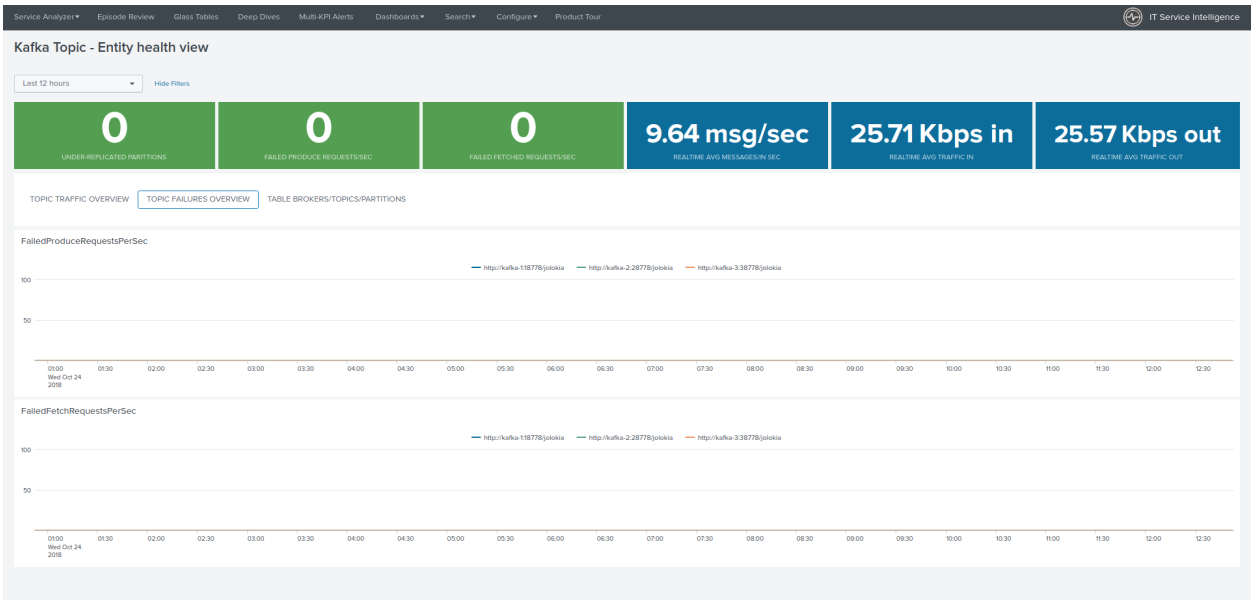

Service Analyzer
Episode Review
Glass Tables
Deep Dives
Multi-KPI Alerts
Dashboards
Search
Configure
Product Tour
IT Service Intelligence
Kafka Broker - Entity health view
Last 12 hours
Hide Filters
0
OFFLINE PARITTIONS
0
UNDER-REPLICATED PARITTIONS
29.42 msg/sec
REALTIME AVG MESSAGES/IN SEC
29.77 Kbps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

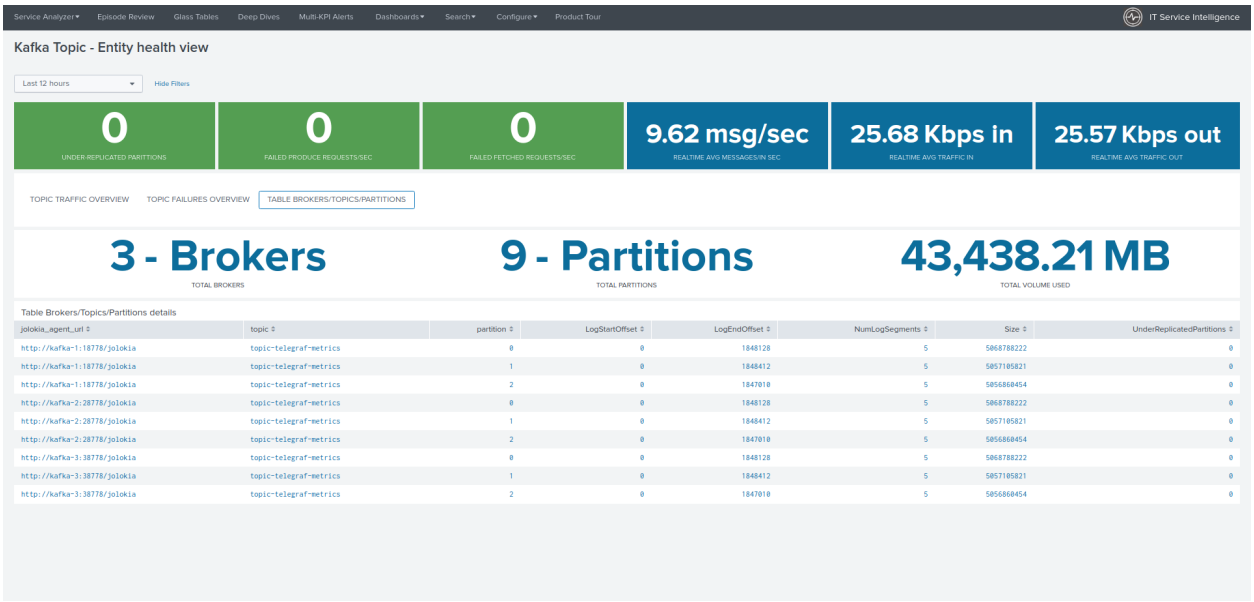

Service Analyzer
Episode Review
Glass Tables
Deep Dives
Multi-KPI Alerts
Dashboards
Search
Configure
Product Tour
IT Service Intelligence
Kafka Broker - Entity health view
Last 12 hours
Hide Filters
0
OFFLINE PARITTIONS
0
UNDER-REPLICATED PARITTIONS
29.43 msg/sec
REALTIME AVG MESSAGES/IN SEC
29.44 Kbps 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)



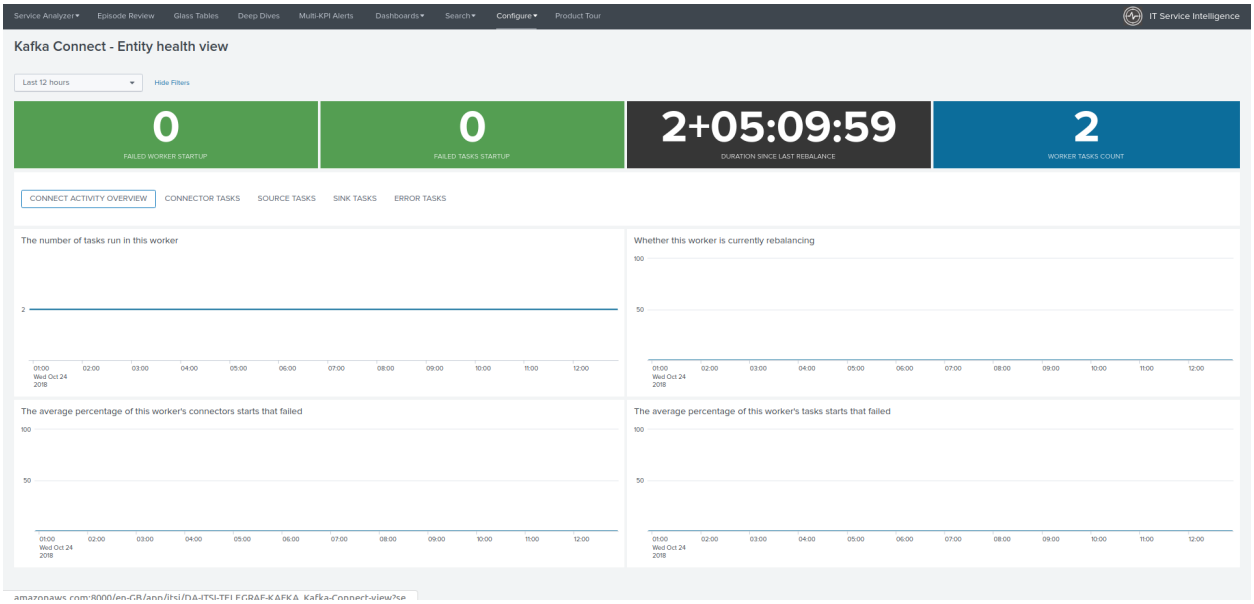
2.6.3 Kafka topic dashboard view

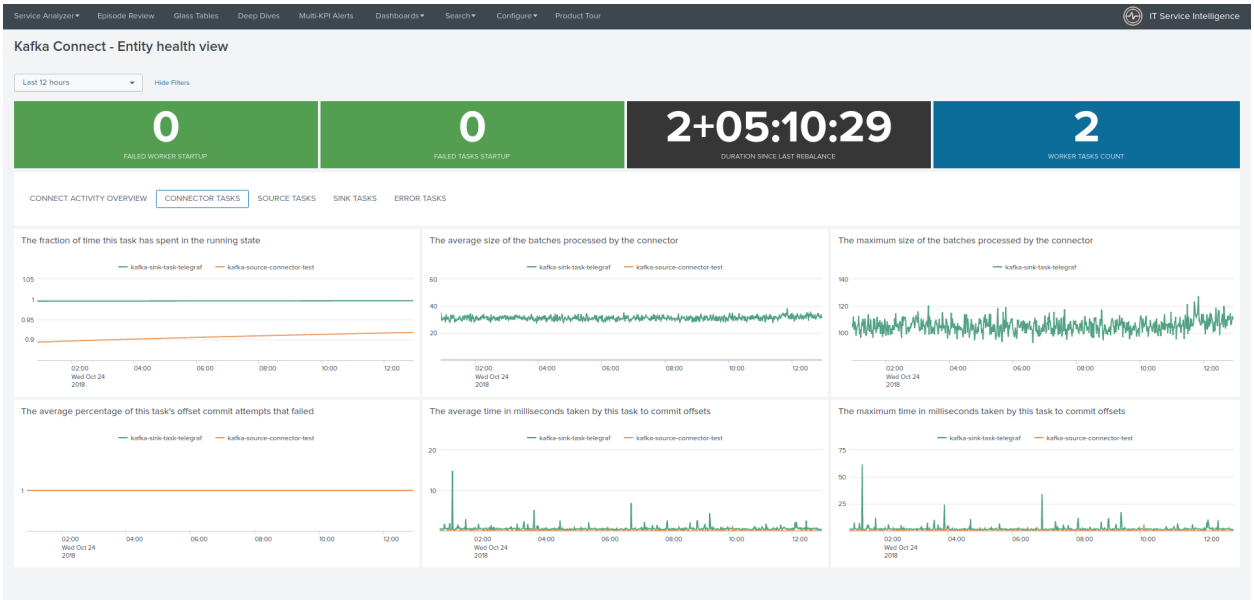


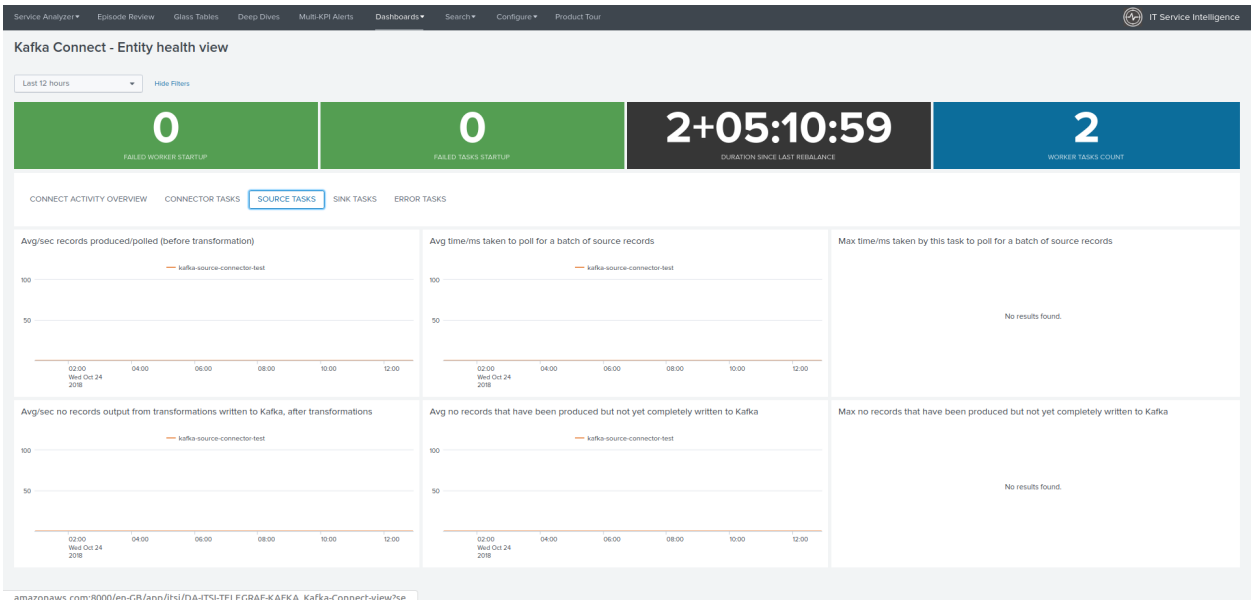

Service Analyzer
Episode Review
Glass Tables
Deep Dives
Multi-KPI Alerts
Dashboards
Search
Configure
Product Tour
IT Service Intelligence
Kafka Topic - Entity health view
Last 12 hours
Hide Filters
0
UNDER-REPLICATED PARITIONS
0
FAILED PRODUCE REQUESTS/SEC
0
FAILED FETCHED REQUESTS/SEC
9.64 msg/sec
REALTIME AVG MESSAGES/IN SEC
25.71 Kbps in
REALTIME AVG TRAFFIC IN
25.57 Kbps out
REALTIME AVG TRAFFIC OUT
TOPIC TRAFFIC OVERVIEW
TOPIC FAILURES OVERVIEW
TABLE BROKERS/TOPICS/PARTITIONS
FailedProduceRequestsPerSec
http://kafka-1:18778/jolokia
http://kafka-2:28778/jolokia
http://kafka-3:38778/jolokia
100
50
01:00
Wed Oct 24
2018
01:30
02:00
02:30
03:00
03:30
04:00
04:30
05:00
05:30
06:00
06:30
07:00
07:30
08:00
08:30
09:00
09:30
10:00
10:30
11:00
11:30
12:00
12:30
FailedFetchRequestsPerSec
http://kafka-1:18778/jolokia
http://kafka-2:28778/jolokia
http://kafka-3:38778/jolokia
100
50
01:00
Wed Oct 24
2018
01:30
02:00
02:30
03:00
03:30
04:00
04:30
05:00
05:30
06:00
06:30
07:00
07:30
08:00
08:30
09:00
09:30
10:00
10:30
11:00
11:30
12:00
12:30

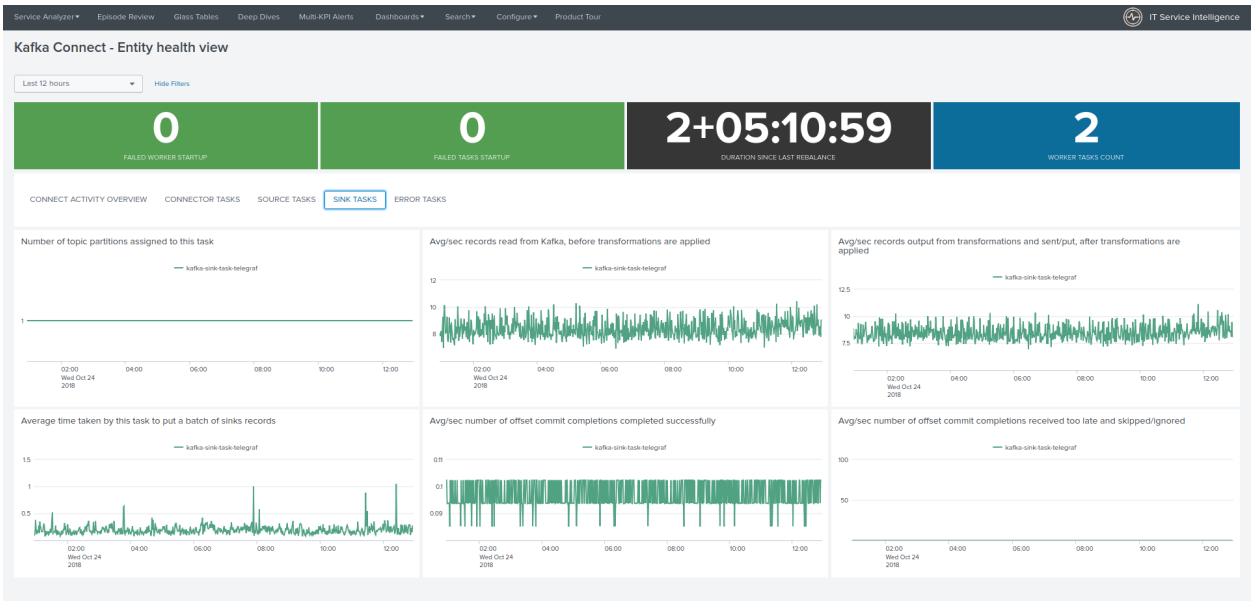

Service Analyzer
Episode Review
Glass Tables
Deep Dives
Multi-KPI Alerts
Dashboards
Search
Configure
Product Tour
IT Service Intelligence
Kafka Topic - Entity health view
Last 12 hours
Hide Filters
0
UNDER-REPLICATED PARITIONS
0
FAILED PRODUCE REQUESTS/SEC
0
FAILED FETCHED REQUESTS/SEC
9.62 msg/sec
REALTIME AVG MESSAGES/IN SEC
25.68 Kbps in
REALTIME AVG TRAFFIC IN
25.57 Kbps out
REALTIME AVG TRAFFIC OUT
TOPIC TRAFFIC OVERVIEW
TOPIC FAILURES OVERVIEW
TABLE BROKERS/TOPICS/PARTITIONS
3 - Brokers
TOTAL BROKERS
9 - Partitions
TOTAL PARTITIONS
43,438.21 MB
TOTAL VOLUME USED
Table Brokers/Topics/Partitions details
jolokia_agent_url
topic
partition
LogStartOffset
LogEndOffset
NumLogSegments
Size
UnderReplicatedPartitions
http://kafka-1:18778/jolokia topic-telegraf-metrics 0 0 1848128 5 5068788222 0
http://kafka-1:18778/jolokia topic-telegraf-metrics 1 0 1848412 5 5057105821 0
http://kafka-1:18778/jolokia topic-telegraf-metrics 2 0 1847010 5 5056860454 0
http://kafka-2:28778/jolokia topic-telegraf-metrics 0 0 1848128 5 5068788222 0
http://kafka-2:28778/jolokia topic-telegraf-metrics 1 0 1848412 5 5057105821 0
http://kafka-2:28778/jolokia topic-telegraf-metrics 2 0 1847010 5 5056860454 0
http://kafka-3:38778/jolokia topic-telegraf-metrics 0 0 1848128 5 5068788222 0
http://kafka-3:38778/jolokia topic-telegraf-metrics 1 0 1848412 5 5057105821 0
http://kafka-3:38778/jolokia topic-telegraf-metrics 2 0 1847010 5 5056860454 0

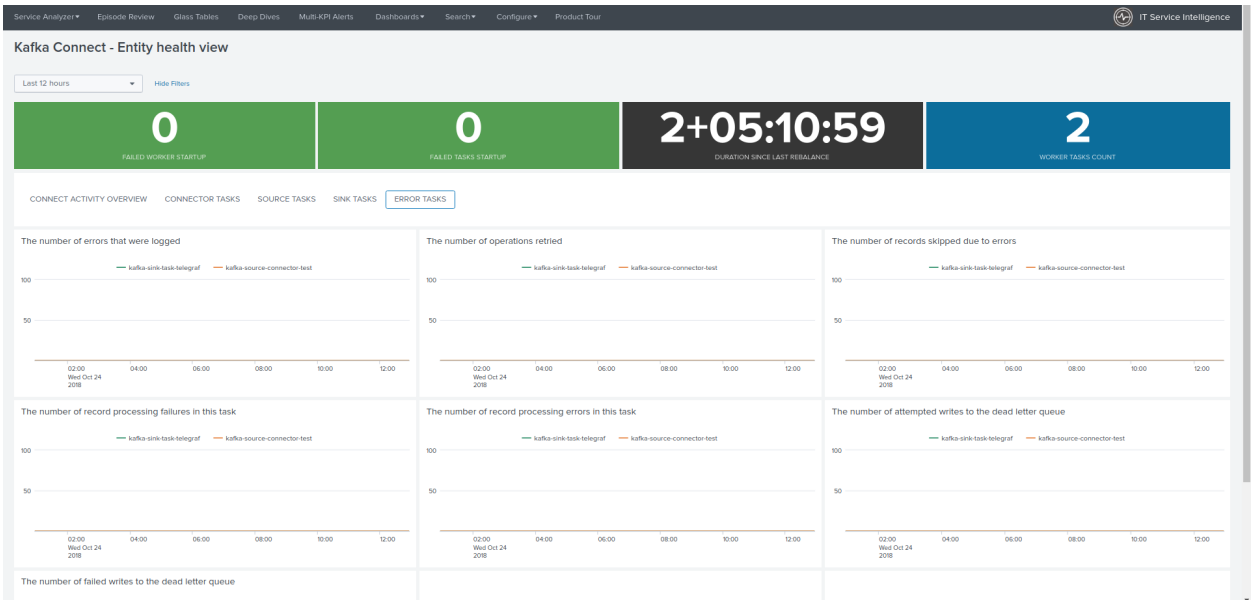
2.6.4 Kafka connect dashboard view



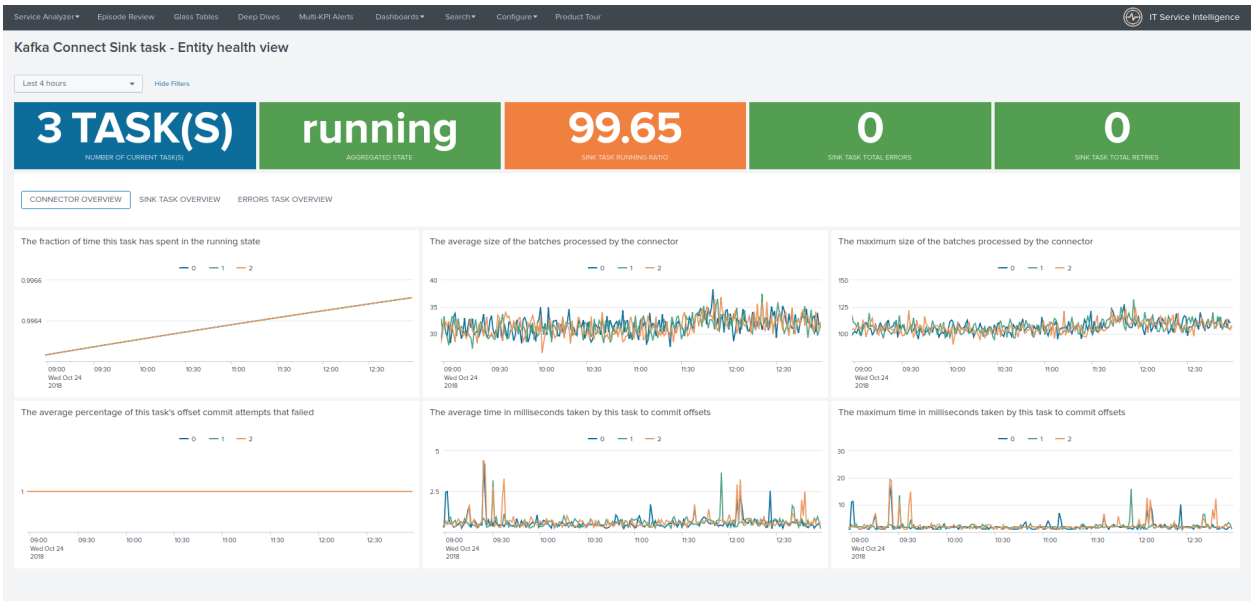


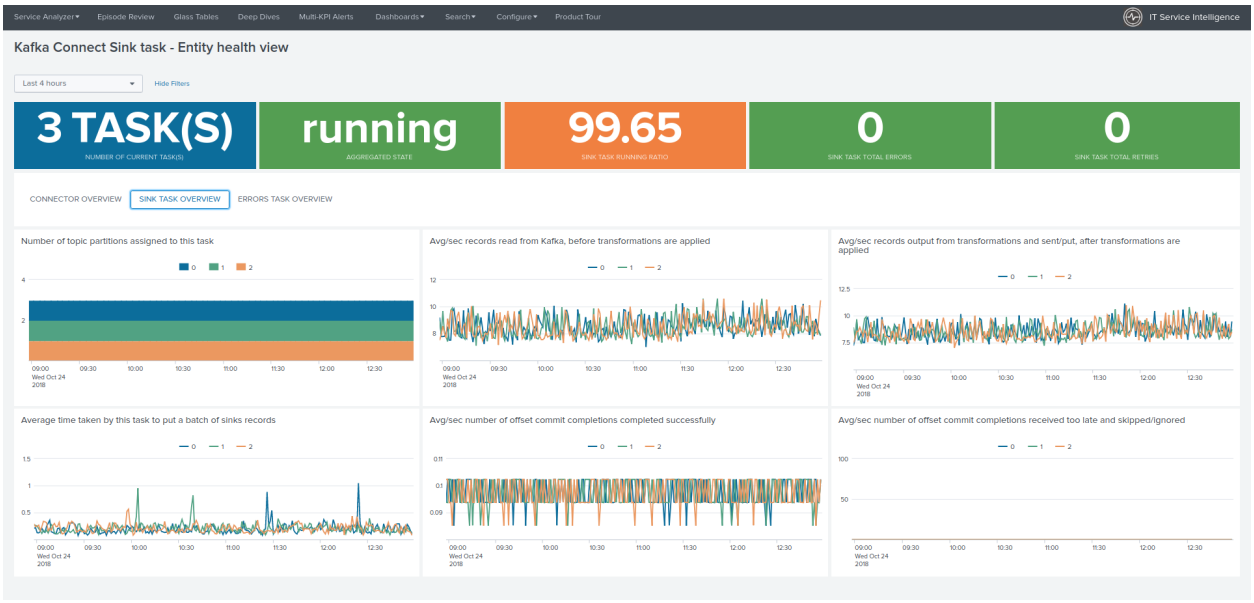

Service Analyzer
Episode Review
Glass Tables
Deep Dives
Multi-KPI Alerts
Dashboards
Search
Configure
Product Tour
IT Service Intelligence
Kafka Connect - Entity health view
Last 12 hours
Hide Filters
0
FAILED WORKER STARTUP
0
FAILED TASKS STARTUP
2+05:10:59
DURATION SINCE LAST REBALANCE
2
WORKER TASKS 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.

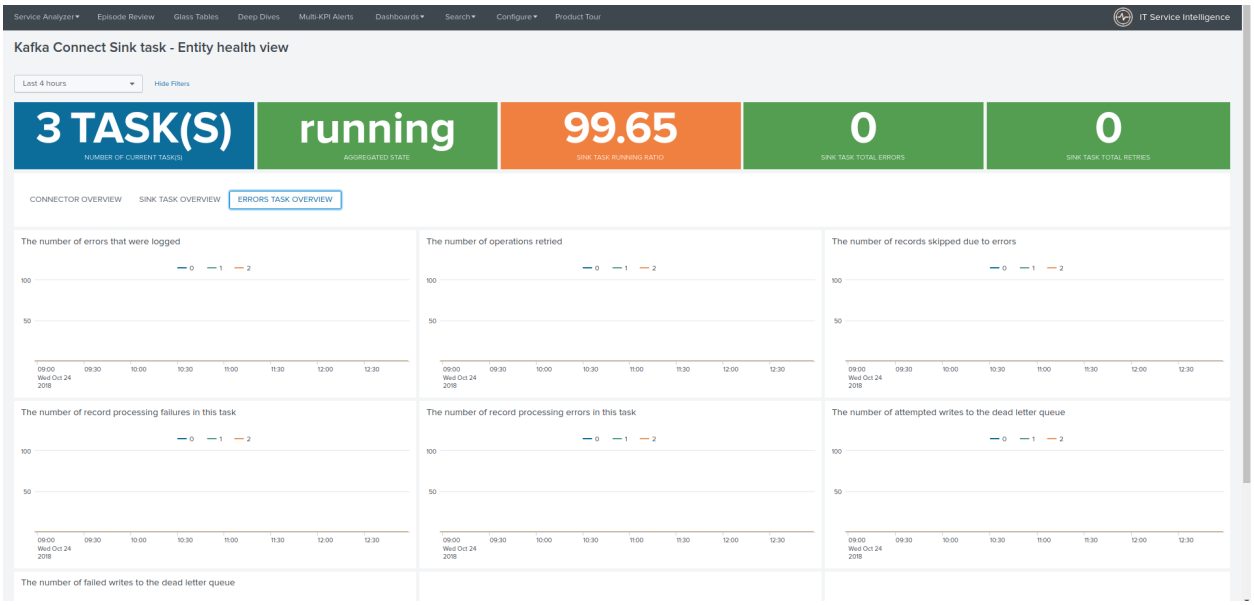

Service Analyzer
Episode Review
Glass Tables
Deep Dives
Multi-KPI Alerts
Dashboards
Search
Configure
Product Tour
IT Service Intelligence
Kafka Connect - Entity health view
Last 12 hours
Hide Filters
0
FAILED WORKER STARTUP
0
FAILED TASKS STARTUP
2+05:10:59
DURATION SINCE LAST REBALANCE
2
WORKER TASKS 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



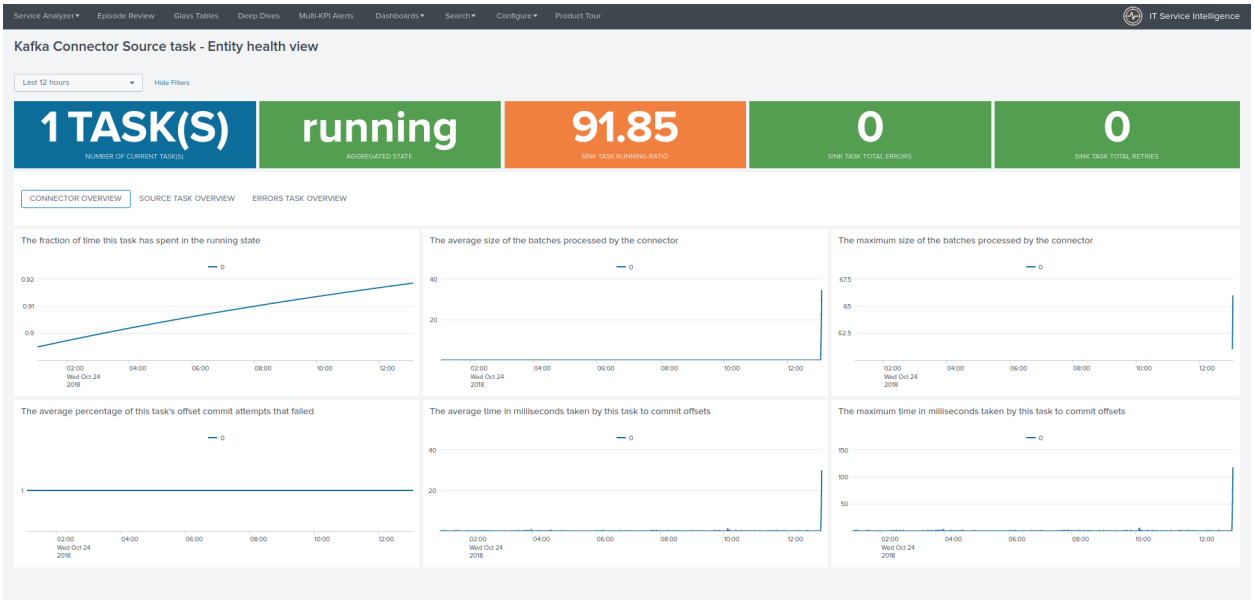
2.6.5 Kafka connect sink task dashboard view

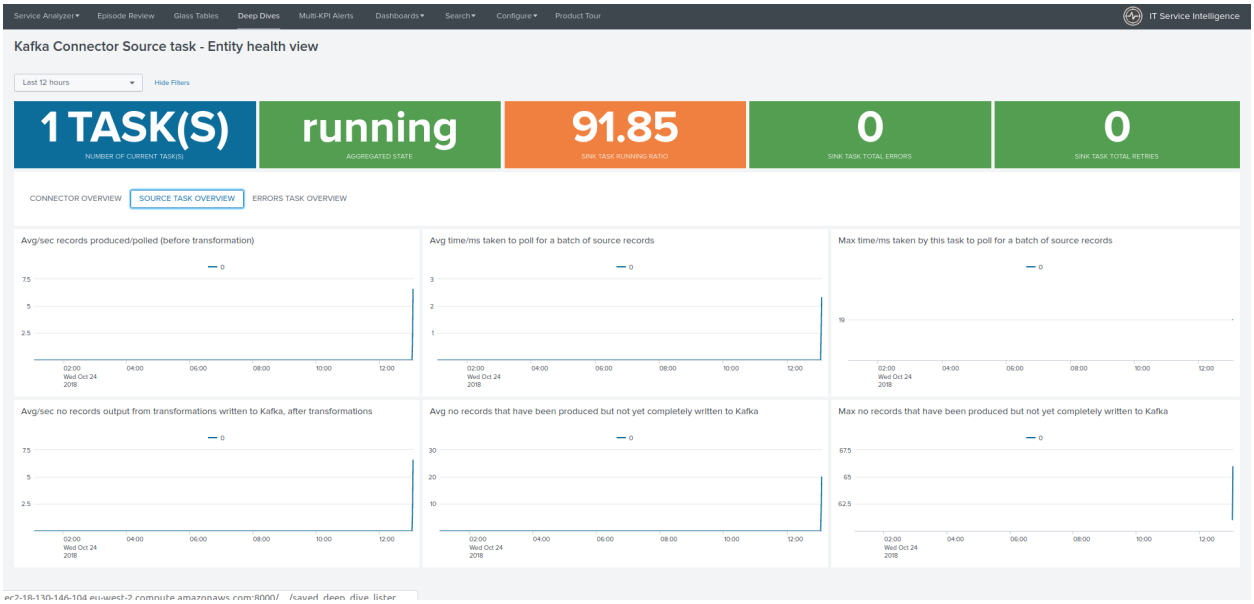


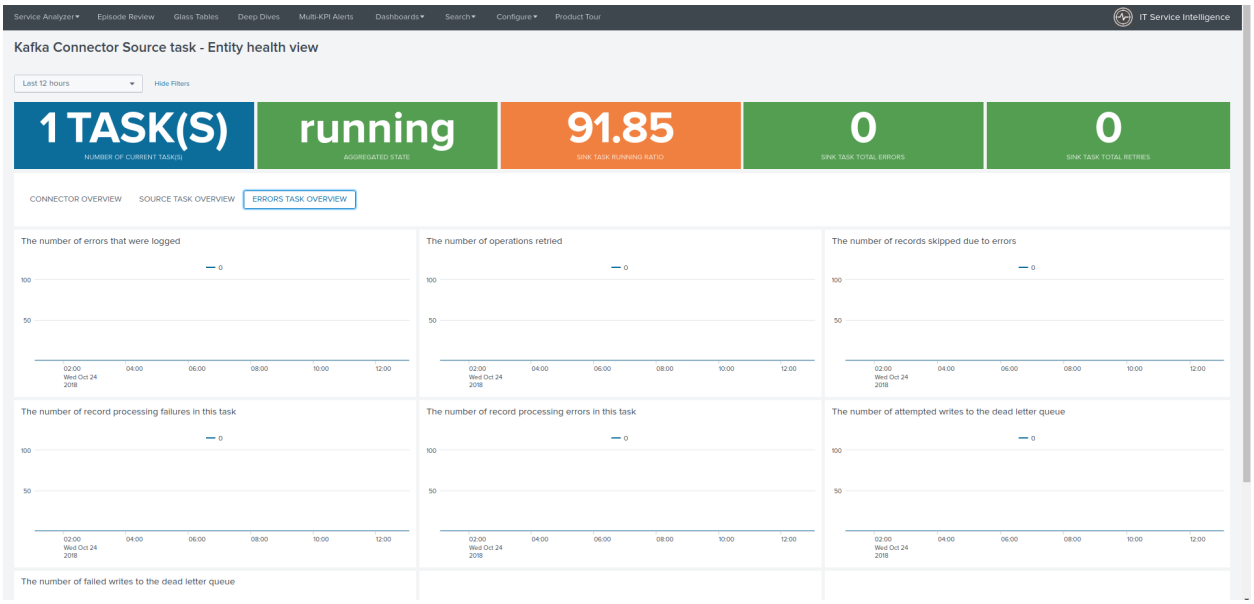

Service Analyzer
Episode Review
Glass Tables
Deep Dives
Multi-KPI Alerts
Dashboards
Search
Configure
Product Tour
IT Service Intelligence
Kafka Connect Sink task - Entity health view
Last 4 hours
Hide Filters
3 TASK(S)
NUMBER OF CURRENT TASK(S)
running
AGGREGATED STATE
99.65
SINK TASK RUNNING RATIO
0
SINK TASK TOTAL ERRORS
0
SINK TASK TOTAL RETRIES
CONNECTOR OVERVIEW
SINK TASK OVERVIEW
ERRORS TASK OVERVIEW
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


Service Analyzer
Episode Review
Glass Tables
Deep Dives
Multi-KPI Alerts
Dashboards
Search
Configure
Product Tour
IT Service Intelligence
Kafka Connect Sink task - Entity health view
Last 4 hours
Hide Filters
3 TASK(S)
NUMBER OF CURRENT TASK(S)
running
AGGREGATED STATE
99.65
SINK TASK RUNNING RATIO
0
SINK TASK TOTAL ERRORS
0
SINK TASK TOTAL RETRIES
CONNECTOR OVERVIEW
SINK 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

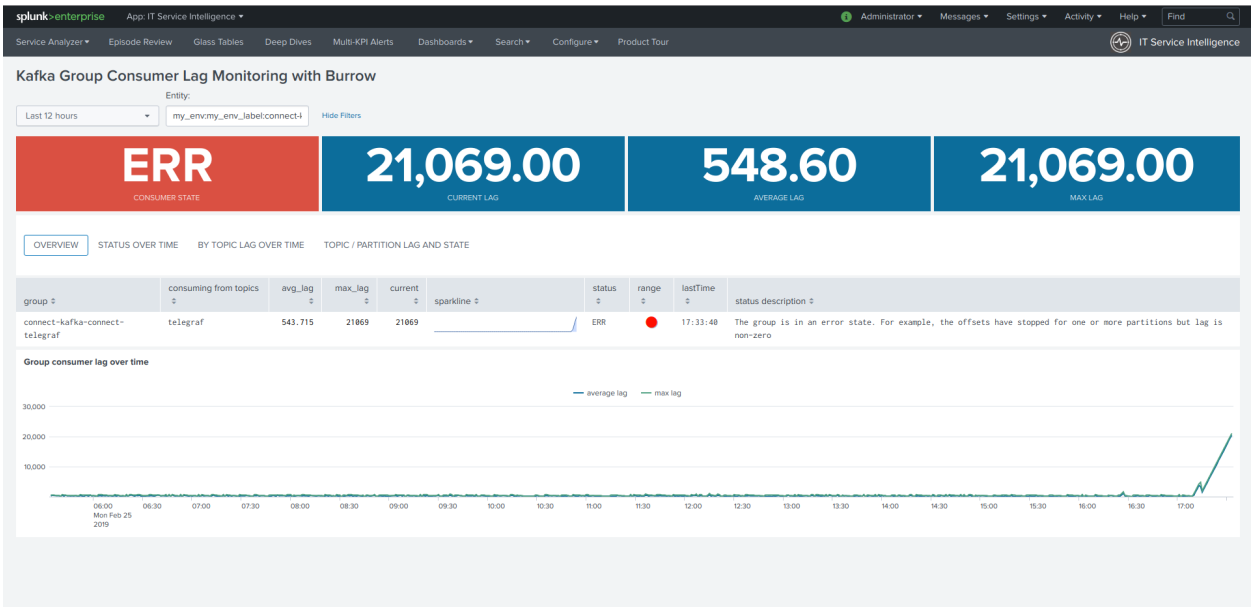
2.6.6 Kafka connect source task dashboard view

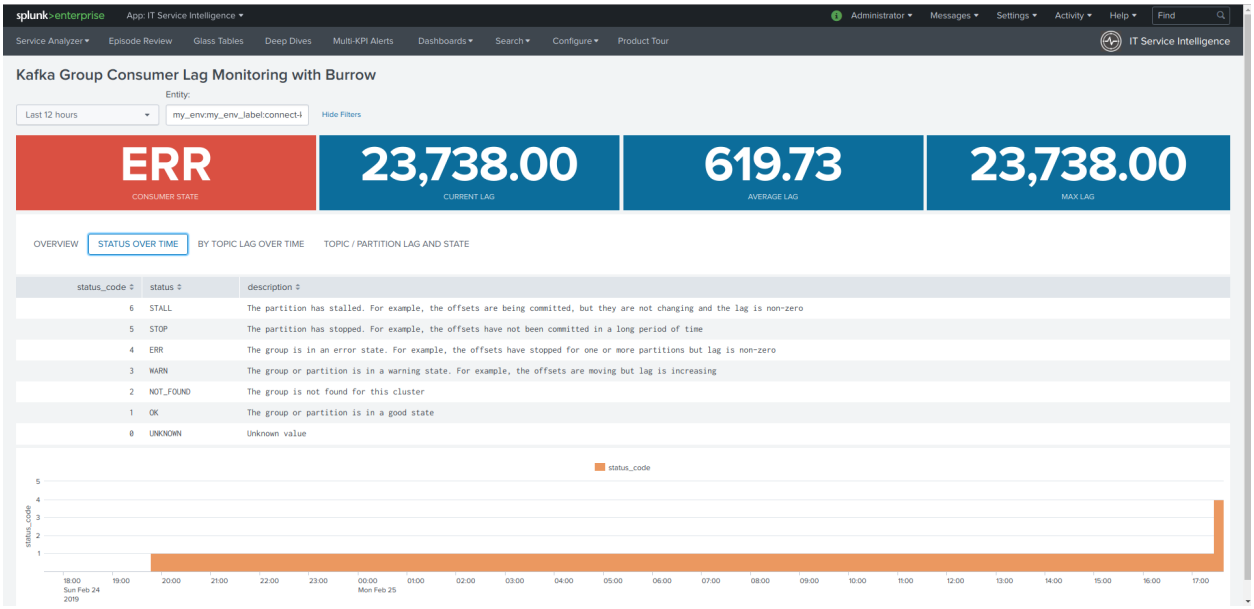


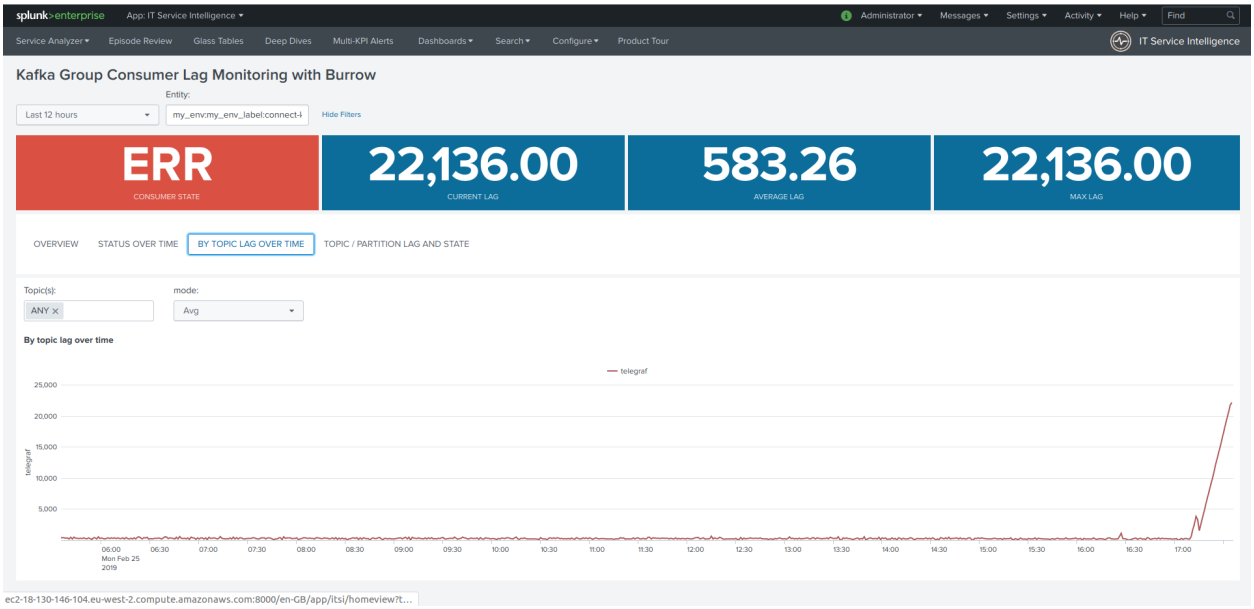


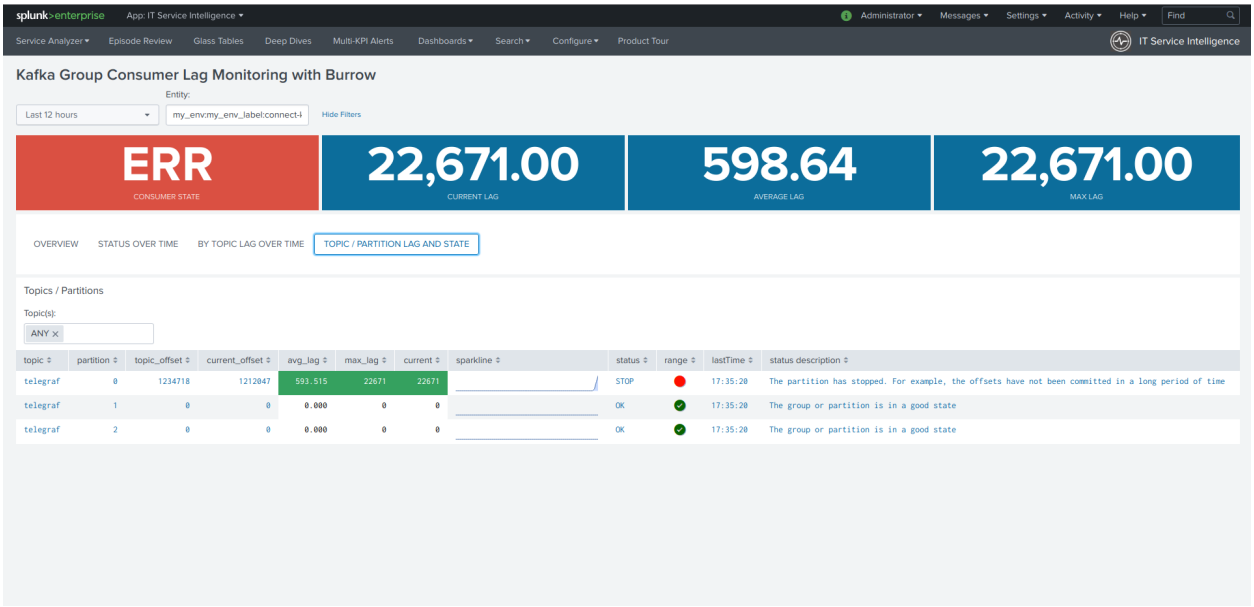


2.6.7 Kafka consumers lag monitoring dashboard view (Burrow)

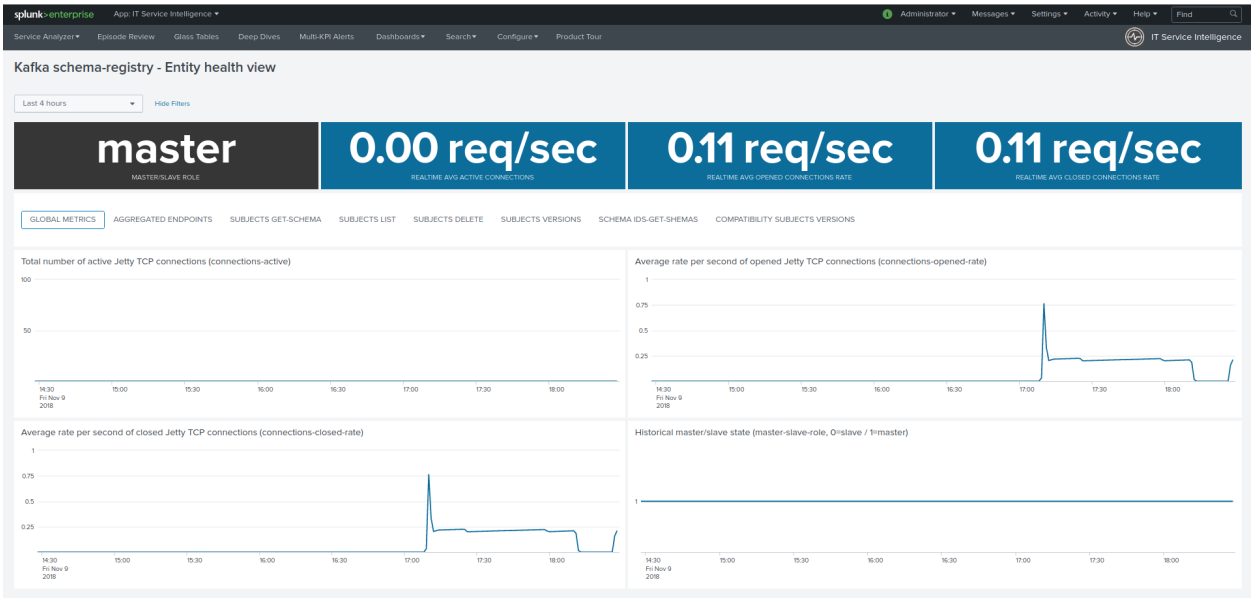


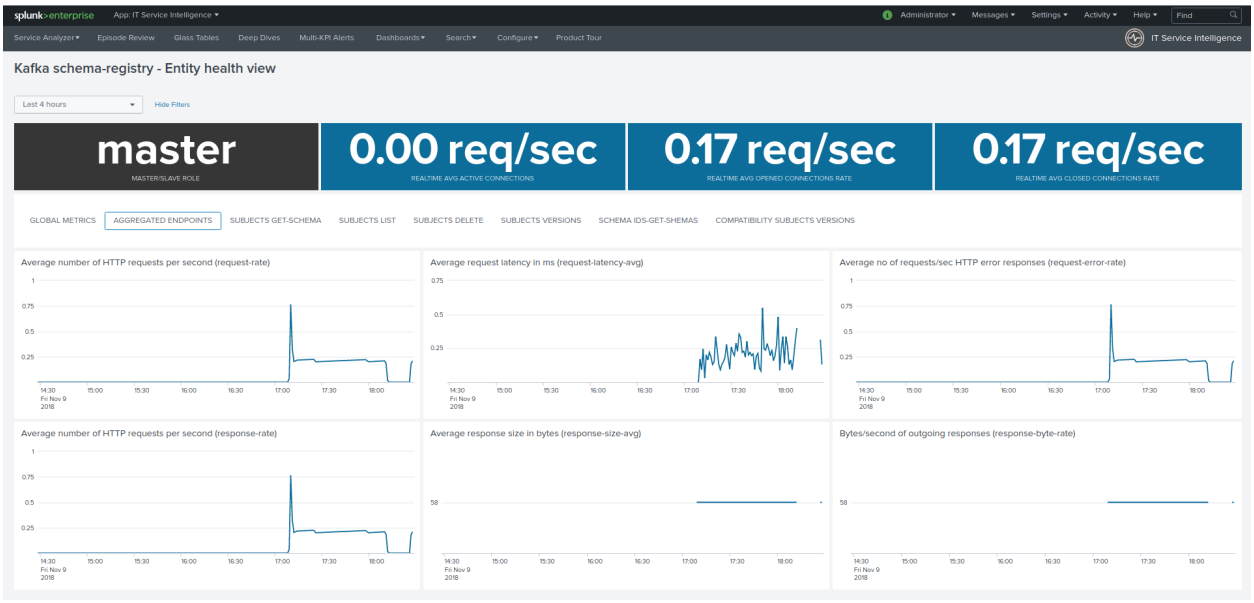

splunk>enterprise
App: IT Service Intelligence
Administrator
Messages
Settings
Activity
Help
Find
Service Analyzer
Episode Review
Glass Tables
Deep Dives
Multi-KPI Alerts
Dashboards
Search
Configure
Product Tour
IT Service Intelligence
Kafka Group Consumer Lag Monitoring with Burrow
Entity:
Last 12 hours
my_env:my_env_label:connect-i
Hide Filters
ERR
CONSUMER STATE
23,738.00
CURRENT LAG
619.73
AVERAGE LAG
23,738.00
MAX LAG
OVERVIEW
STATUS OVER TIME
BY TOPIC LAG OVER TIME
TOPIC / PARTITION LAG AND STATE
status_code
status
description
6 STALL The partition has stalled. For example, the offsets are being committed, but they are not changing and the lag is non-zero
5 STOP The partition has stopped. For example, the offsets have not been committed in a long period of time
4 ERR The group is in an error state. For example, the offsets have stopped for one or more partitions but lag is non-zero
3 WARN The group or partition is in a warning state. For example, the offsets are moving but lag is increasing
2 NOT_FOUND The group is not found for this cluster
1 OK The group or partition is in a good state
0 UNKNOWN Unknown value
status_code
Sun Feb 24 2019
Mon Feb 25


splunk>enterprise
App: IT Service Intelligence
Administrator
Messages
Settings
Activity
Help
Find
Service Analyzer
Episode Review
Glass Tables
Deep Dives
Multi-KPI Alerts
Dashboards
Search
Configure
Product Tour
IT Service Intelligence
Kafka Group Consumer Lag Monitoring with Burrow
Entity:
Last 12 hours
my_env:my_env_label:connect-i
Hide Filters
ERR
CONSUMER STATE
22,136.00
CURRENT LAG
583.26
AVERAGE LAG
22,136.00
MAX LAG
OVERVIEW
STATUS OVER TIME
BY TOPIC LAG OVER TIME
TOPIC / PARTITION LAG AND STATE
Topic(s):
ANY
mode:
Avg
By topic lag over time
telegraf
Mon Feb 25 2019
ec2-18-130-146-104.eu-west-2.compute.amazonaws.com:8000/en-GB/app/itsi/homeview?t...

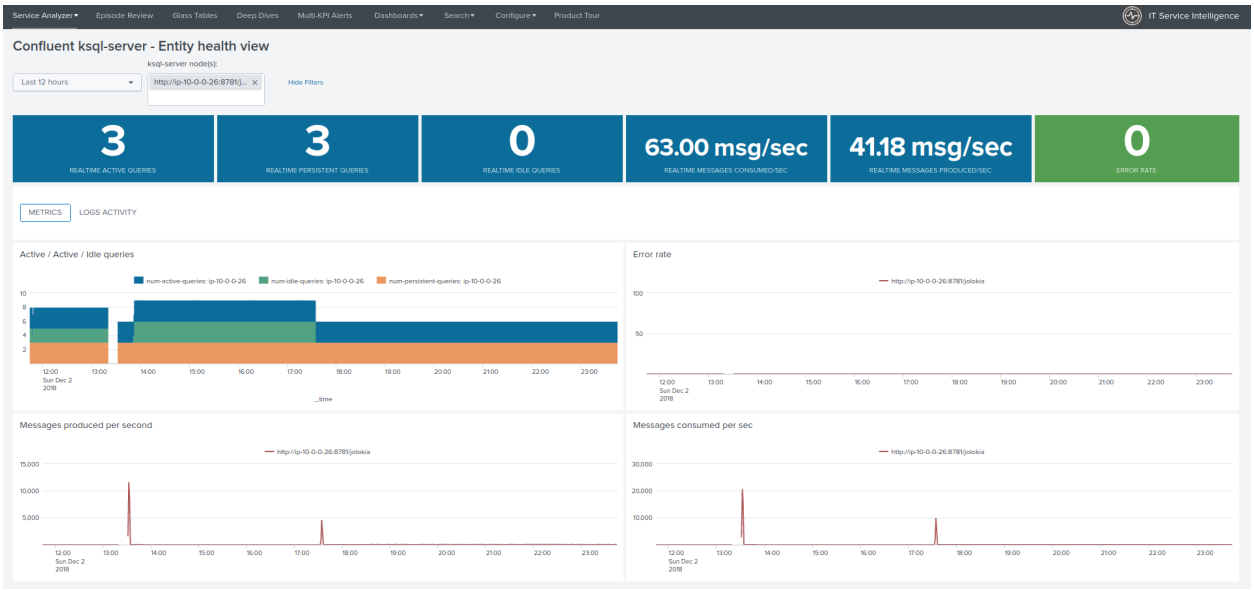


2.6.8 Confluent schema-registry dashboard view

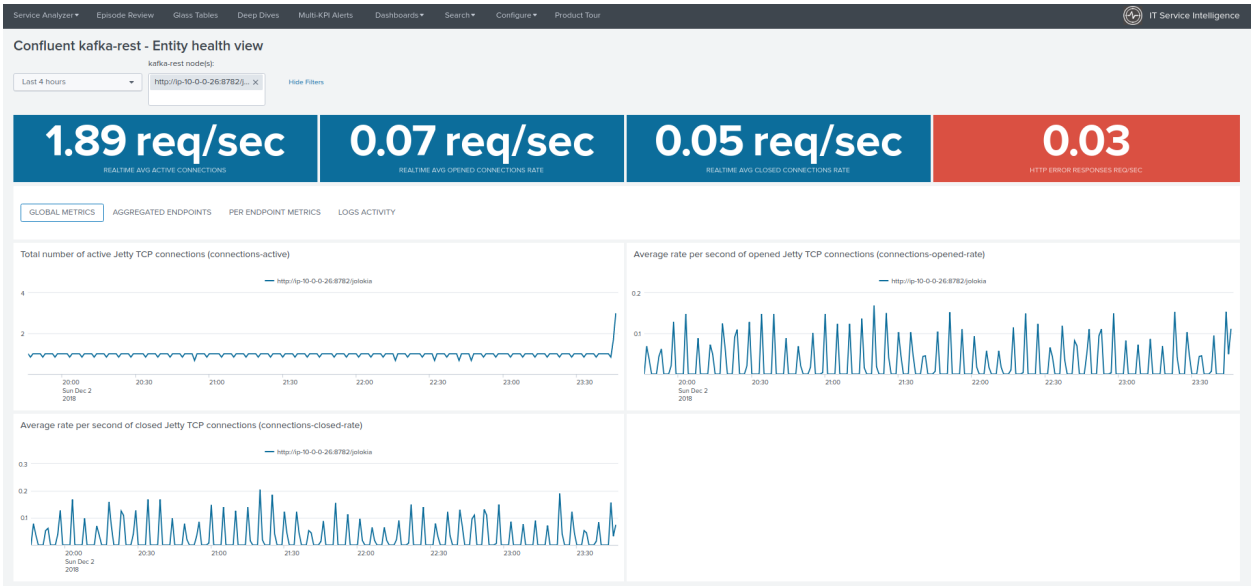


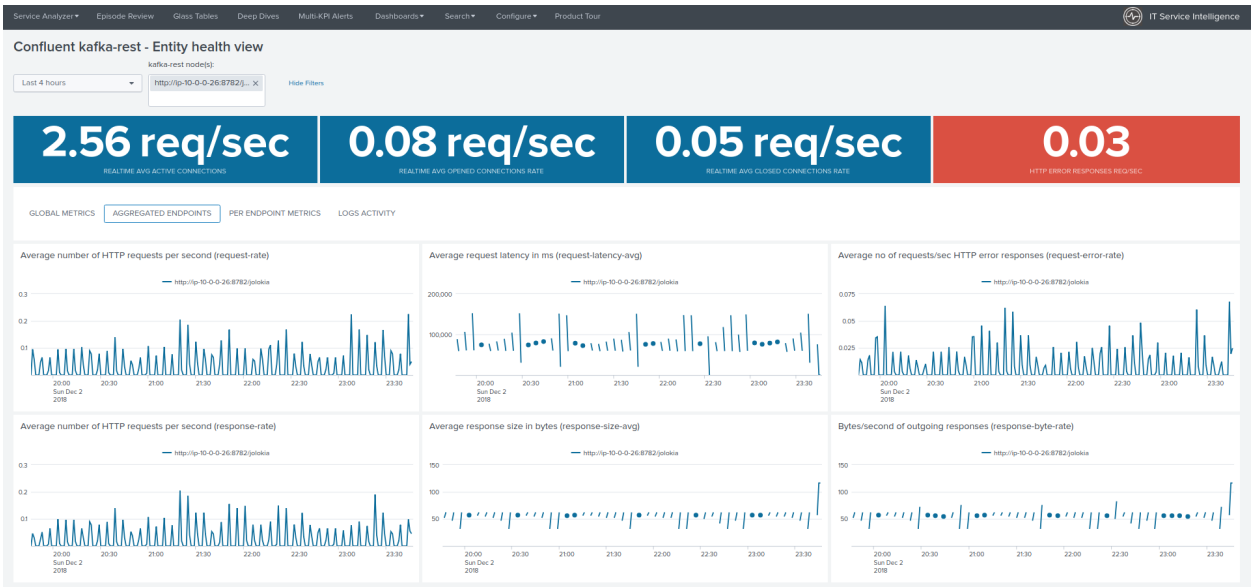


2.6.9 Confluent ksql-server dashboard view

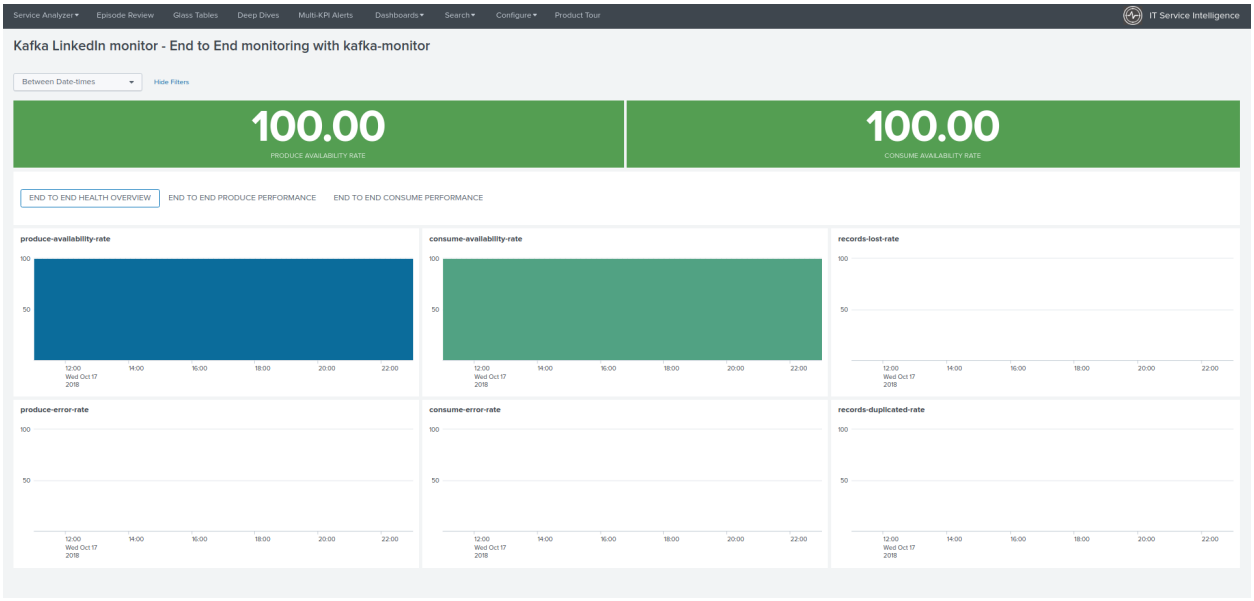


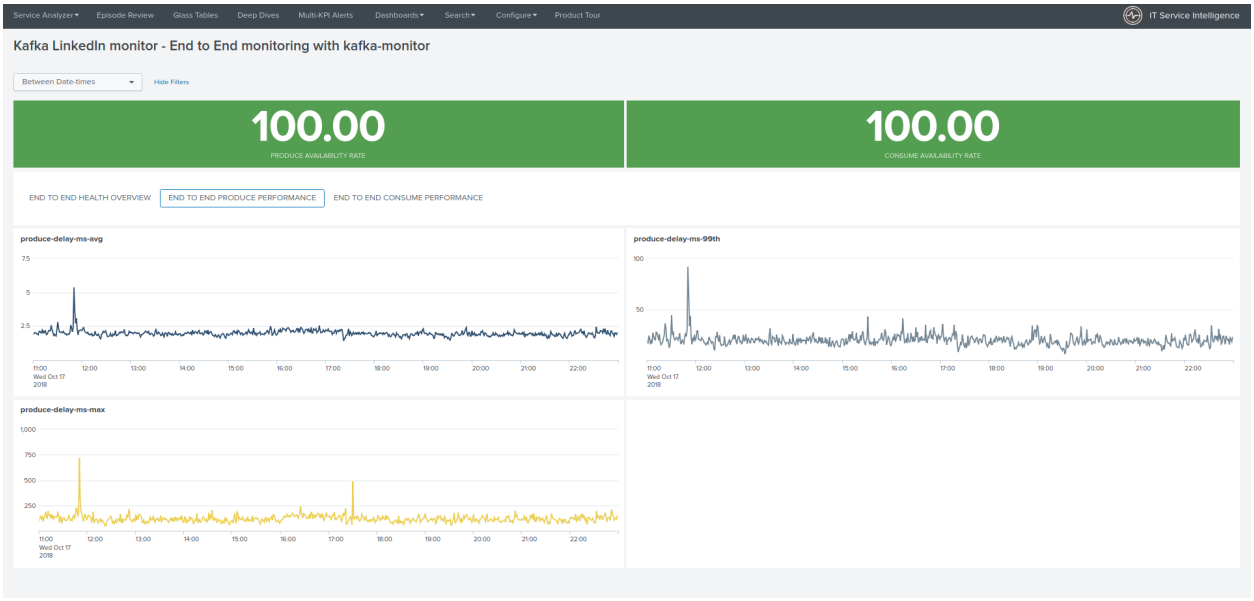
2.6.10 Confluent kafka-rest dashboard view

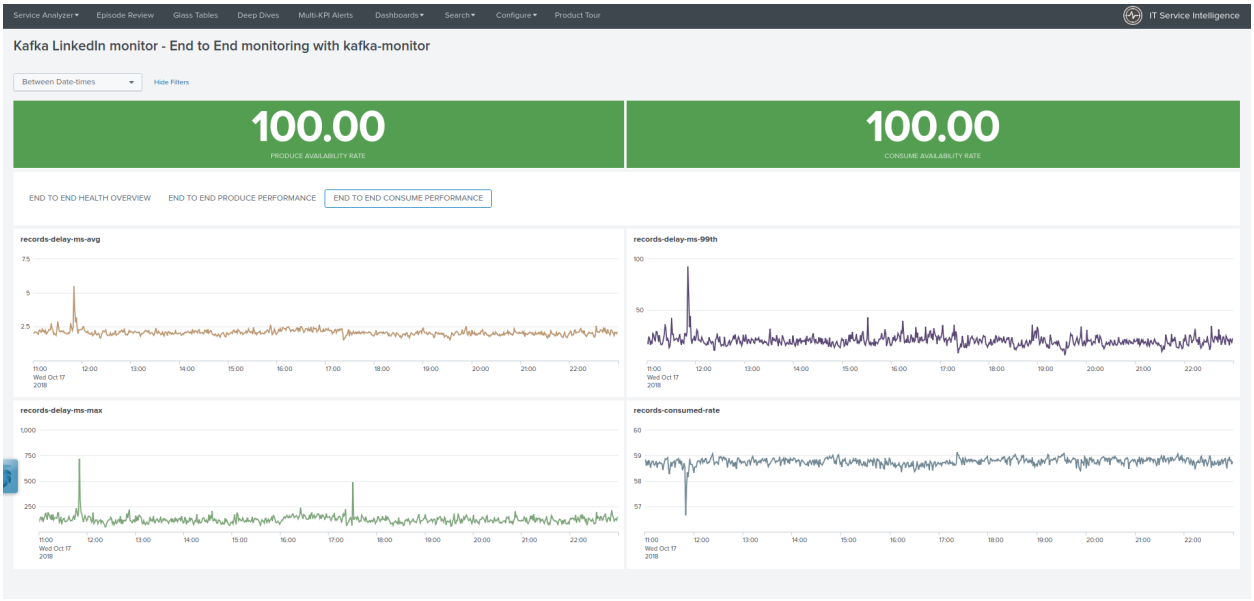




2.6.11 LinkedIn Kafka monitor view






Service Analyzer
Episode Review
Glass Tables
Deep Dives
Multi-KPI Alerts
Dashboards
Search
Configure
Product Tour
IT Service Intelligence
Kafka LinkedIn monitor - End to End monitoring with kafka-monitor
Between Date-times
Hide Filters
100.00
PRODUCE AVAILABILITY RATE
100.00
CONSUME AVAILABILITY RATE
END TO END HEALTH OVERVIEW
END TO END PRODUCE PERFORMANCE
END TO END CONSUME PERFORMANCE
records-delay-ms-avg
records-delay-ms-99th
records-delay-ms-max
records-consumed-rate

CHAPTER 3

Troubleshoot:

3.1 Troubleshoot & FAQ

CHAPTER 4

Versioniong and build history:

4.1 Release notes

4.1.1 Version 1.1.5

- fix: Static index reference in Kafka Brokers entities discovery report
- feature: Drilldown to single forms for Offline and Under-replicated partitions in Overview and Kafka Brokers entities views

4.1.2 Version 1.1.4

- fix: incompatibility for ksql-server with latest Confluent release (5.1.x) due to metric name changes in JMX model

4.1.3 Version 1.1.3

Burrow integration: Kafka Consumer Lag monitoring

- feature: New KPI basesearch and Service Template for Kafka Consumers Lag Monitoring with Burrow
- feature: New entity view for Kafka Consumers Lag monitoring

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

4.1.4 Version 1.1.2

- unpublished

4.1.5 Version 1.1.1

CAUTION: Breaking changes and major release, 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-installation-and-configuration

Upgrade path:

- Upgrade telegraf configuration to provide the env and label tags
- Upgrade the module, manage entities and rebuild your services

release notes:

- fix: duplicated KPI id for topic/brokers under replicated replication leads in KPI rendering issues
- fix: entity rendering issue with Kafka SLA monitor health view

4.1.6 Version 1.1.0

CAUTION: Breaking changes and major release, 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-installation-and-configuration

Upgrade path:

- Upgrade telegraf configuration to provide the env and label tags
- Upgrade the module, manage entities and rebuild your services

release notes:

- feature: Support for multi-environments / multi-dc deployments with metrics tagging
- feature: Global rewrite of entities management and identification
- fix: Moved from second interval to cron schedule for entities import to avoid dup entities at addon installation time
- fix: Various fixes and improvements

4.1.7 Version 1.0.6

- feature: Support for Confluent ksql-server
- feature: Support for Confluent kafka-rest
- feature: event logging integration with the TA-kafka-streaming-platform

4.1.8 Version 1.0.5

- feature: Support for Confluent schema-registry
- feature: Adding follower/leader info in Zookeeper entity view

4.1.9 Version 1.0.4

- fix: typo on partitions in Kafka brokers view

4.1.10 Version 1.0.3

- fix: missing entity filter in latency from Zookeeper view
- fix: incorrect static filter in state from Sink task view

4.1.11 Version 1.0.2

- fix: incorrect duration shown in Kafka Connect entity view
- feature: minor improvements in UIs

4.1.12 Version 1.0.1

- fix: error in state of Source/Sink connector in dashboards

4.1.13 Version 1.0.0

- initial and first public release