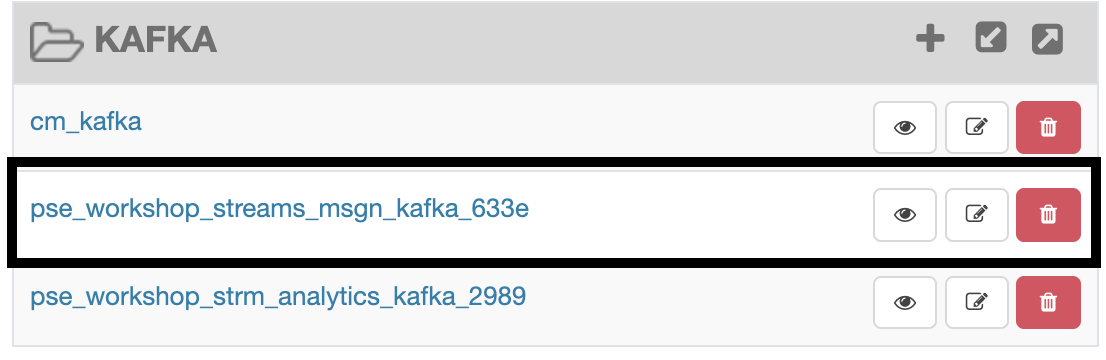
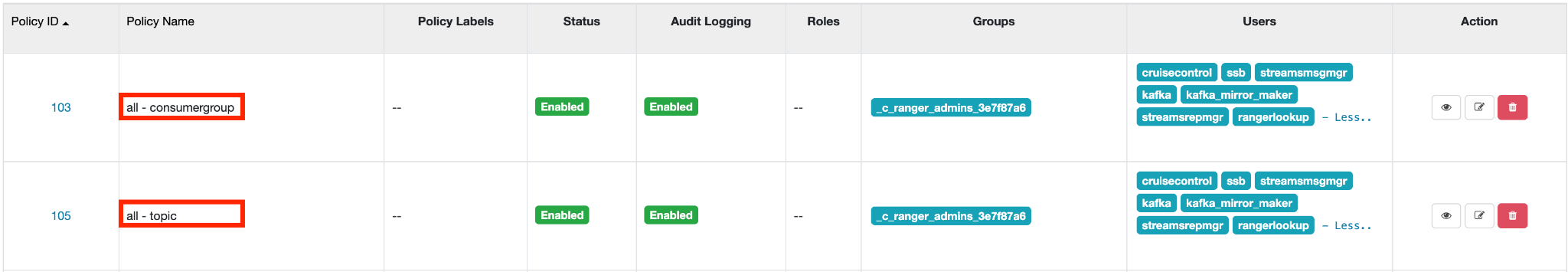
Cloudera Data Flow Workshop Guide

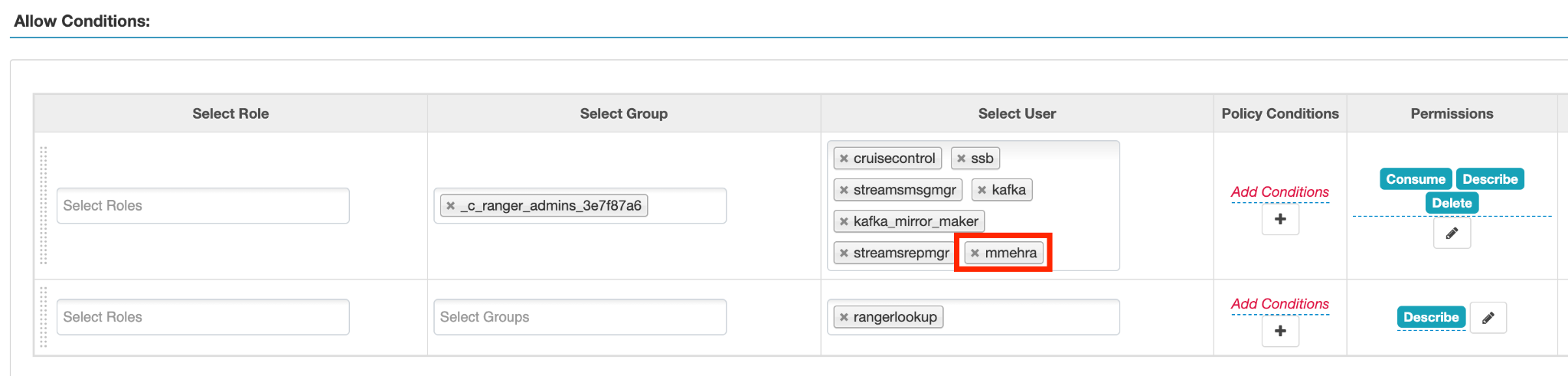
# CDF Lab - Introduction and setup

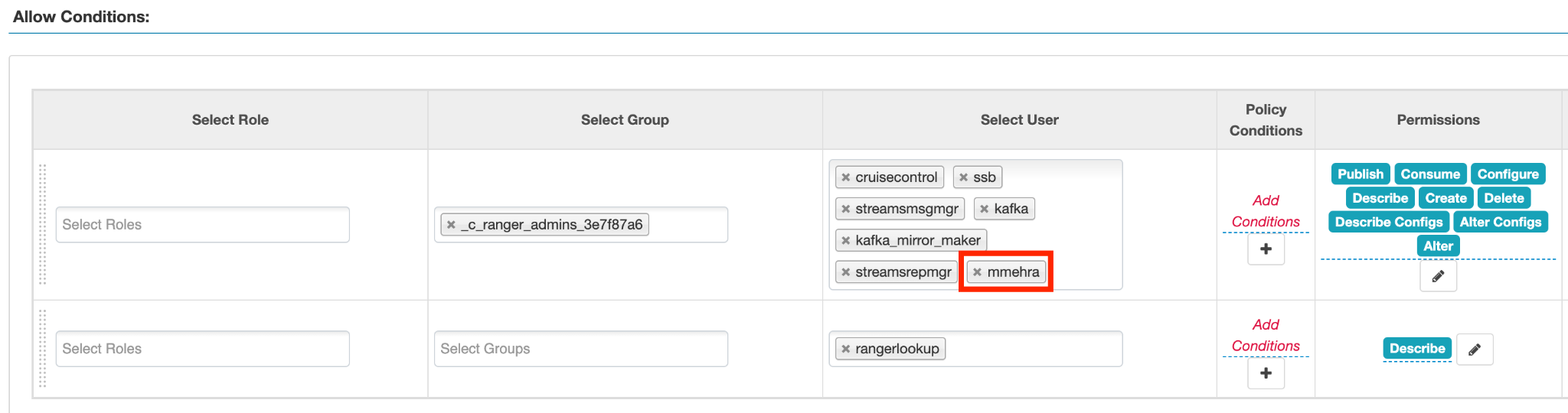
## 1. Configure permissions in Apache Ranger

### 1.1 Kafka Permissions

1. In Ranger, select the Kafka repository that’s associated with the stream messaging datahub.
2. Add the user who will be performing the workshop to the existing permissions in both **all-consumergroup** and **all-topic** and click Save

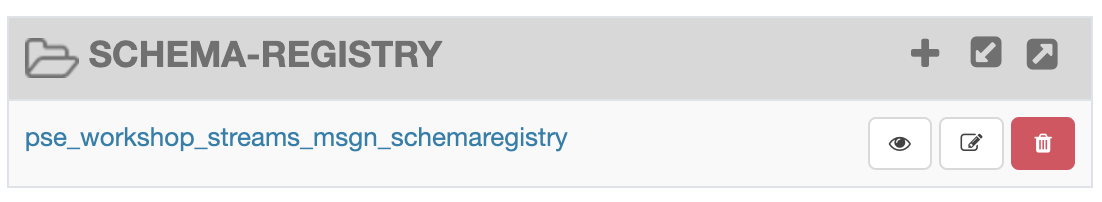




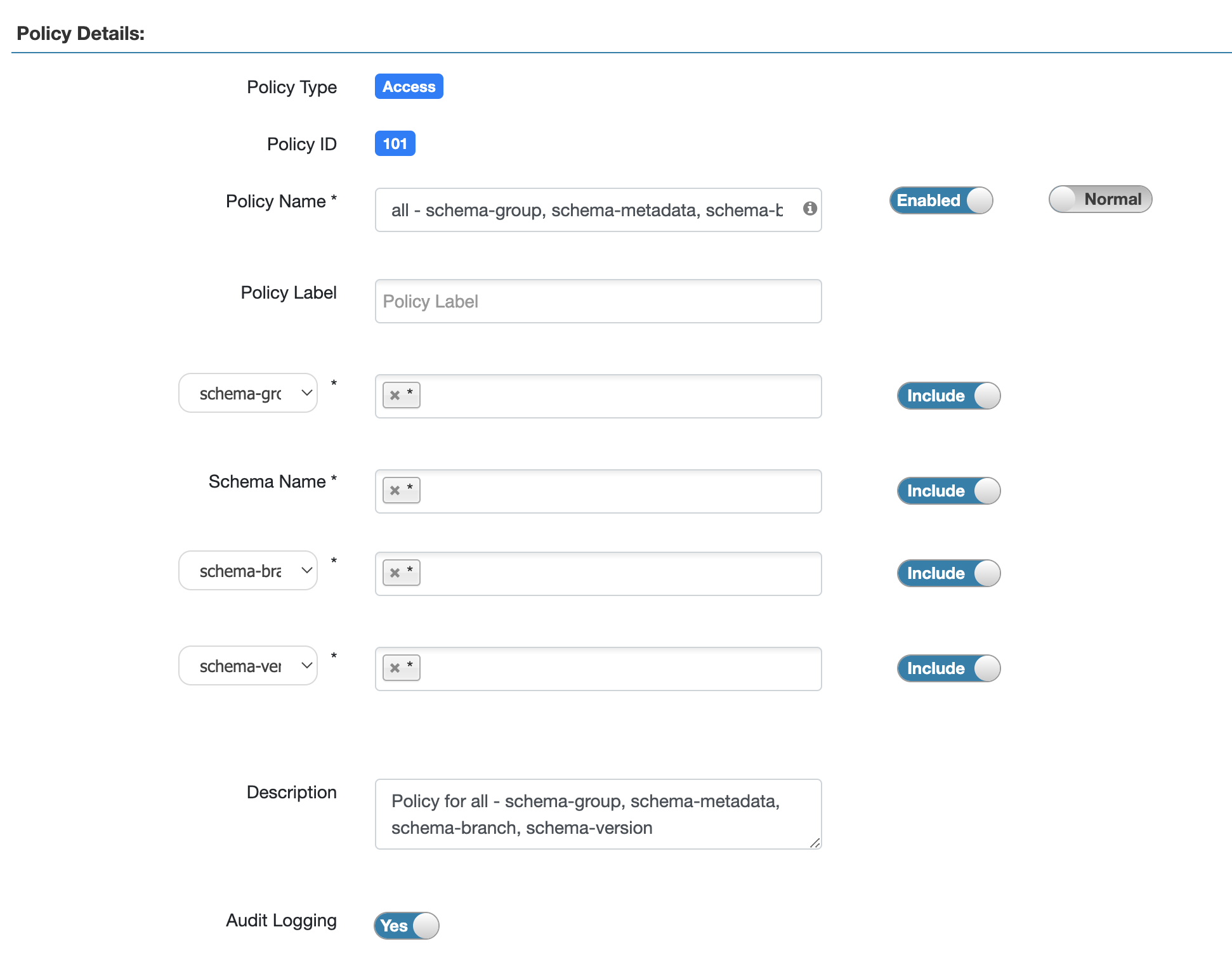


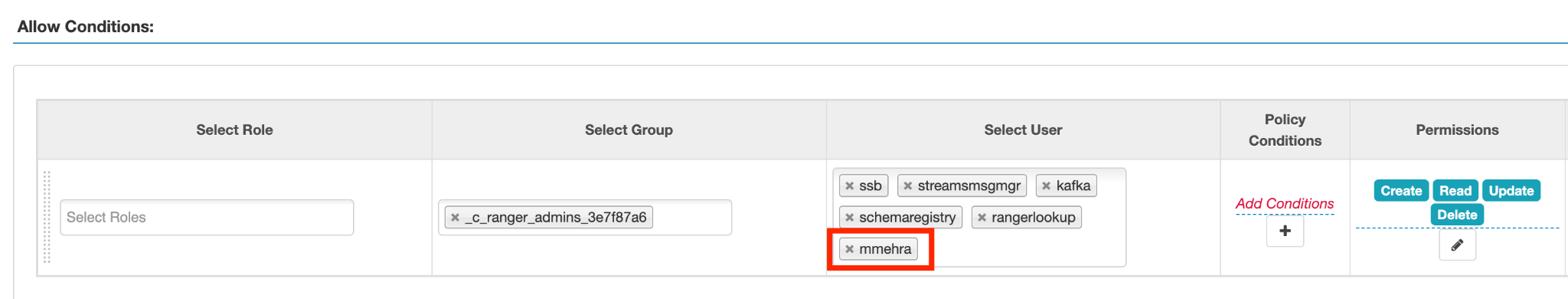
### 1.2 Schema Registry Permissions

1. In Ranger, select the Schema Registry repository that’s associated with the stream messaging datahub.



1. Add the user who will be performing the workshop to the existing permissions in the Policy for: **all - schema-group, schema-metadata, schema-branch, schema-version** and click Save.





# 

# CDF Lab : Migrating Existing Data Flows to CDF-PC

## 1. Overview

The purpose of this workshop is to demonstrate how existing NiFi flows can be migrated to the Data Flow Experience. This workshop will leverage an existing NiFi flow template that has been designed with the best practices for CDF-PC flow deployment.

The existing NiFi Flow will perform the following actions:

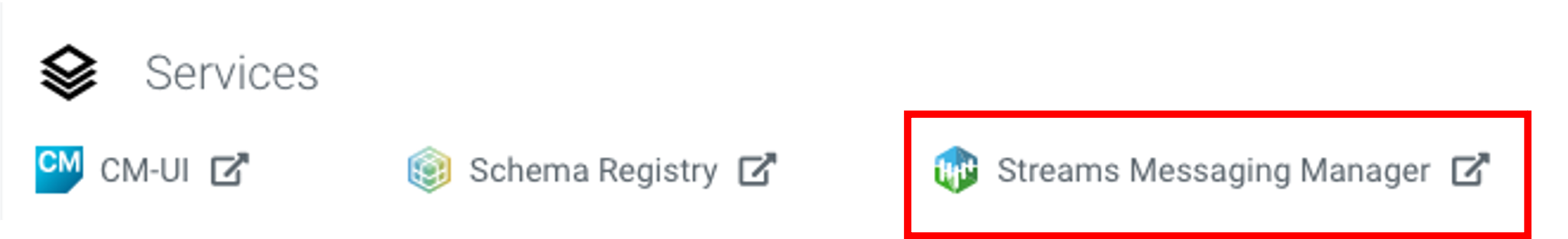
1. Generate random syslogs in 5424 Format
2. convert the incoming data to a JSON using record writers
3. Apply a SQL filter to the JSON records
4. Send the transformed syslog messages to Kafka

Note that a parameter context has already been defined in the flow and the queues have been uniquely named.

## 2. Running the Workshop

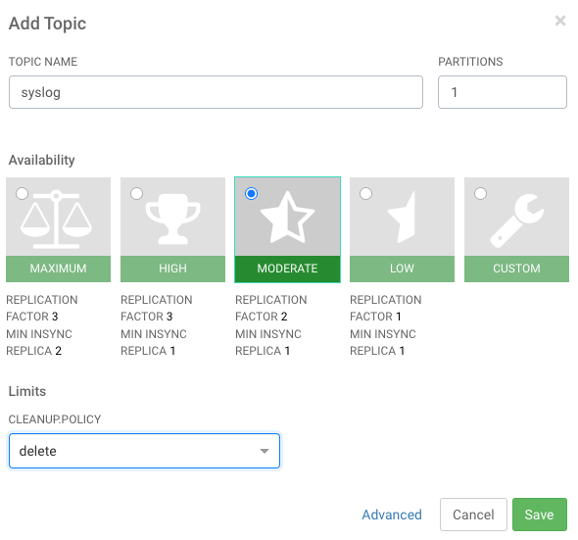
### 2.1. Create a Kafka Topic

1. Login to Streams Messaging Manager by clicking the appropriate hyperlink in the Streams Messaging Datahub



1. Click on Topics in the right tab
2. Click on Add New
3. Create a Topic with the following parameters then click Save:

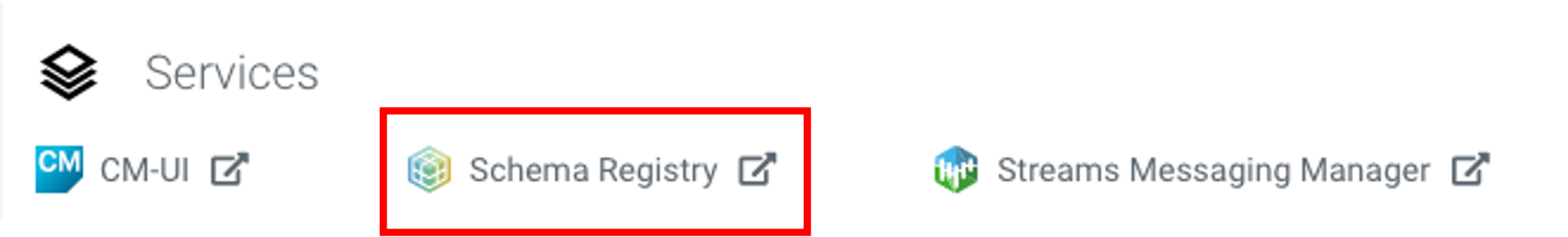
* **Name**: <username>-syslog
* **Partitions**: 1
* **Availability**: Moderate
* **Cleanup Policy**: Delete



**Note**: The Flow will not work if you set the Cleanup Policy to anything other than **Delete**. This is because we are not specifying keys when writing to Kafka.

### 2.2. Create a Schema in Schema Registry

1. Login to Schema Registry by clicking the appropriate hyperlink in the Streams Messaging Datahub.



1. Click on the + button on the top right to create a new schema.
2. Create a new schema with the following information:

* **Name**: <username>-syslog
* **Description**: syslog schema for dataflow workshop
* **Type**: Avro schema provider
* **Schema Group**: Kafka
* **Compatibility**: Backward
* **Evolve**: True
* **Schema** Text:

|  |
| --- |
| {  "name": "syslog",  "type": "record",  "namespace": "com.cloudera",  "fields": [  {  "name": "priority",  "type": "int"  },  {  "name": "severity",  "type": "int"  },  {  "name": "facility",  "type": "int"  },  {  "name": "version",  "type": "int"  },  {  "name": "timestamp",  "type": "long"  },  {  "name": "hostname",  "type": "string"  },  {  "name": "body",  "type": "string"  },  {  "name": "appName",  "type": "string"  },  {  "name": "procid",  "type": "string"  },  {  "name": "messageid",  "type": "string"  },  {  "name": "structuredData",  "type": {  "name": "structuredData",  "type": "record",  "fields": [  {  "name": "SDID",  "type": {  "name": "SDID",  "type": "record",  "fields": [  {  "name": "eventId",  "type": "string"  },  {  "name": "eventSource",  "type": "string"  },  {  "name": "iut",  "type": "string"  }  ]  }  }  ]  }  }  ]  } |

**Note**: The name of the Kafka Topic and the Schema Name must be the same.

### **Discuss the NiFi Flow**

**High Level Overview**

The NiFi flow follows the design methodology behind Cloudera’s Streaming Reference Architecture. In this architecture, we capture data from edge devices, transform and filter it with NiFi, and then distribute it into Kafka for downstream consumption.

In this flow we will simulate reading syslogs in the RFC 5424 Standard. This is done with the Generate Syslog RFC5424 processor (leveraging python script). This processor is configured to run on the primary node, with 4 concurrent threads, and a round-robin load balance success queue.

Concurrency is added to the source processor to demonstrate autoscaling based on CPU utilization when deployed to CDF-PC.

Next, we will natively read this data as syslog and apply a SQL filter to each record in the flowfile. The default SQL filter selects all records.

Afterwards we convert the data to JSON using NiFI record writers. The record readers and writers leverage a defined schema within Schema Registry to understand and process the incoming data.

Finally, we will take the syslog data that has been converted to JSON and send it to Kafka for downstream processing.

**Deep Dive**

Capturing data from edge devices is represented in the flow with the Generate Syslog RFC5424 Processor. Underneath the covers, there is a python based ExecuteScript which will generate 10 random log messages at a time that are consistent with the syslog RFC5424 format. Here is an example output from this processor:

|  |
| --- |
| <75>1 2021-09-22T17:38:52.611Z host2.example.com application1 8518 ID34 [SDID iut="3" eventSource="application" eventId="92"] application1 has exited cleanly  <174>1 2021-09-22T17:38:52.623Z host4.example.com application7 3891 ID4 [SDID iut="8" eventSource="python" eventId="87"] application7 has stopped unexpectedly  <71>1 2021-09-22T17:38:52.624Z host3.example.com application6 1237 ID16 [SDID iut="8" eventSource="python" eventId="52"] application6 has completed gracefully  <37>1 2021-09-22T17:38:52.624Z host6.example.com application8 3409 ID27 [SDID iut="3" eventSource="application" eventId="25"] application8 has started successfully  <53>1 2021-09-22T17:38:52.624Z host2.example.com application6 8645 ID16 [SDID iut="4" eventSource="kernel" eventId="70"] application6 has exited cleanly  <79>1 2021-09-22T17:38:52.625Z host5.example.com application5 9688 ID9 [SDID iut="9" eventSource="python" eventId="32"]  application5 has started successfully  <19>1 2021-09-22T17:38:52.625Z host8.example.com application9 7773 ID44 [SDID iut="6" eventSource="python" eventId="82"] application9 has exited cleanly  <149>1 2021-09-22T17:38:52.625Z host7.example.com application7 5858 ID6 [SDID iut="3" eventSource="kernel" eventId="45"] application7 has completed gracefully  <186>1 2021-09-22T17:38:52.626Z host1.example.com application7 5795 ID20 [SDID iut="4" eventSource="kernel" eventId="62"] application7 has started successfully |

Going a bit deeper, the RFC5424 format is defined as:



Where:

* **PRI** — or "priority", is a number calculated from Facility (what kind of message) code and Severity (how urgent is the message) code: PRI = **Facility** \* 8 + **Severity**
* **VERSION** — version is always "1" for RFC 5424
* **TIMESTAMP** — valid timestamp examples (must follow ISO 8601 format with uppercase "T" and "Z")
* **HOSTNAME** — using FQDN (fully qualified domain name) is recommended, e.g. mymachine.example.com
* **APP-NAME** — usually the name of the device or application that provided the message
* **PROCID** — often used to provide the process name or process ID (is - "nil" in the example)
* **MSGID** — should identify the type of message,
* **STRUCTURED-DATA** — named lists of key-value pairs for easy parsing and searching
* **MSG** — details about the event

Dowstream, the Write to Kafka processor will publish the syslog records to Kafka in JSON format. This processor uses the syslog record reader to read the incoming syslog messages and the Avro record writer to convert the messages to Avro before publishing to the Kafka topic.

Within the Query Record processor, we define a Syslog 5424 Record Reader and a JSON Record writer. These record based processors leverage the syslog schema that was previously created in the Schema Registry.

Each of the name fields within the schema can be used with the SQL FIter. For the workshop, focus specifically on the **severity** field.

RFC 5424 defines severity as follows:

* 0 - Emergency: system is unusable
* 1 - Alert: action must be taken immediately
* 2 - Critical: critical conditions
* 3 - Error: error conditions
* 4 - Warning: warning conditions
* 5 - Notice: normal but significant condition
* 6 - Informational: informational messages
* 7 - Debug: debug-level messages

The Filter Rule Parameter can therefore be updated to something like:

|  |
| --- |
| SELECT \* FROM FLOWFILE WHERE severity <=3 |

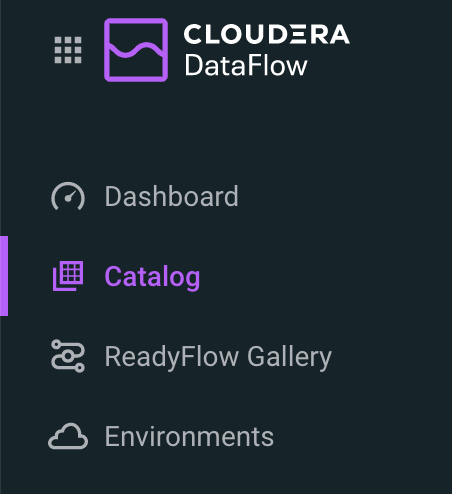
This would filter all incoming syslog messages and only convert and transfer those with severity levels of 0, 1, 2, and 3.

We will change the default filter during CDF-PC deployment. For now, keep the default filter intact.

## 

## Import the Flow into the CDF-PC Catalog

1. Download Nifi Flow from Github - <https://github.com/amishra1006/CDP-Enablement-session-2024/blob/main/syslog-to-kafka.json>
2. Open the CDF-PC data service and click on Catalog in the left tab.



1. Select Import Flow Definition on the Top Right

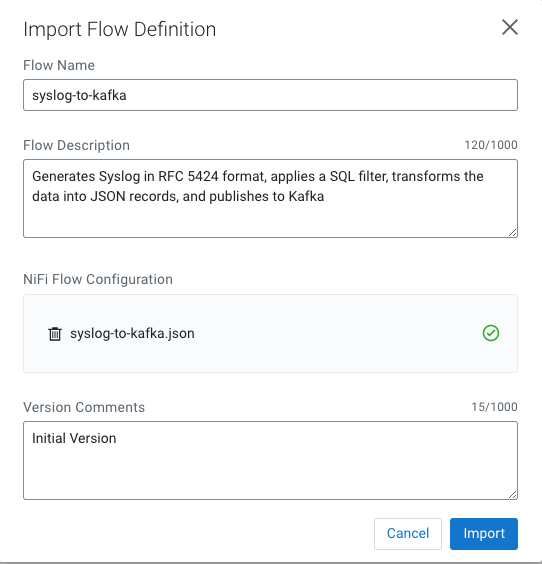


1. Add the following information:

* **Flow Name:** syslog-to-kafka
* **Flow Description:**

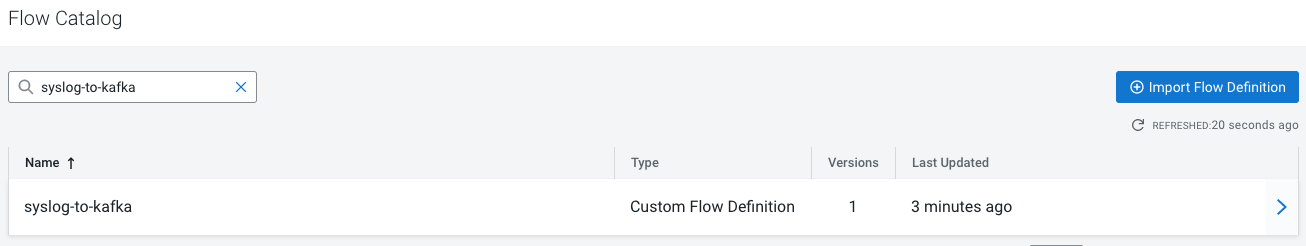
|  |
| --- |
| Reads Syslog in RFC 5424 format, applies a SQL filter, transforms the data into JSON records, and publishes to Kafka |

* **NiFi Flow Configuration:** syslog-to-kafka.json (upload the Flow Definition)
* **Version Comments:** Initial Version

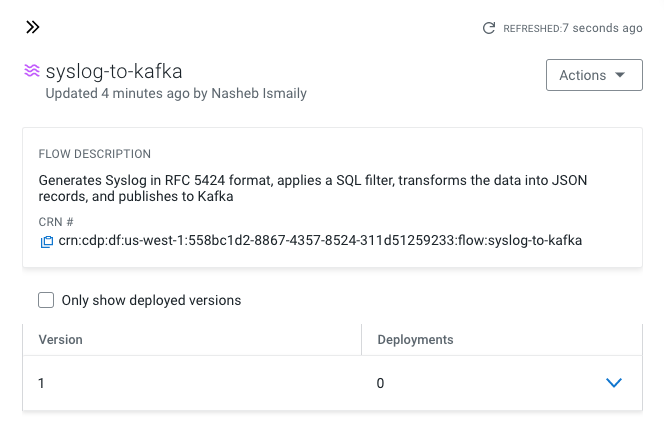


## 2. Deploy the Flow in CDF-PC

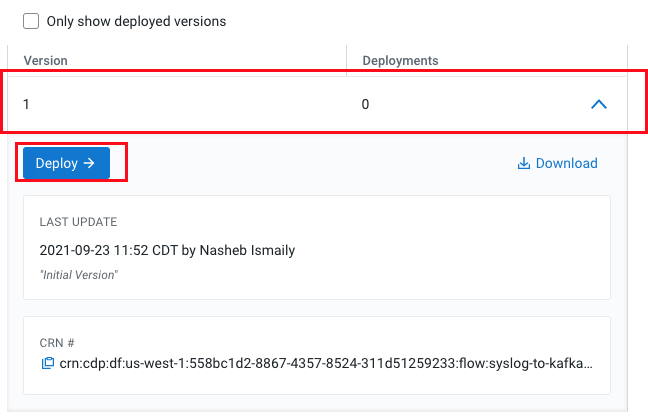
1. Search for the flow in the Flow Catalog



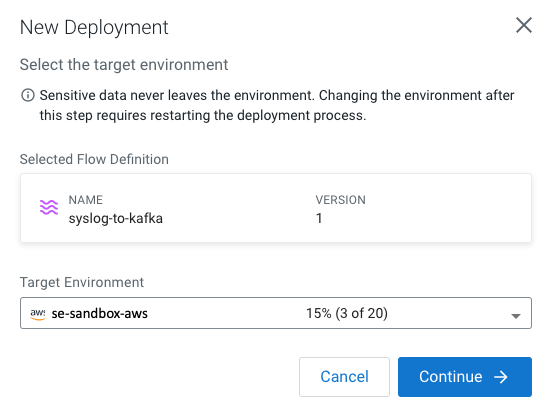
1. Click on the Flow, you should see the following:



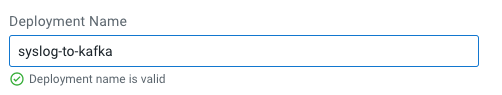
1. Click on **Version 1**, you should see a **Deploy** Option appear shortly. Then click on **Deploy**.



1. Select the CDP environment where this flow will be deployed.



1. Give the deployment a unique name, then click Next.



1. Add the Flow Parameters. These should be the same values that were used to successfully run the flow earlier in the Nif DataHub.

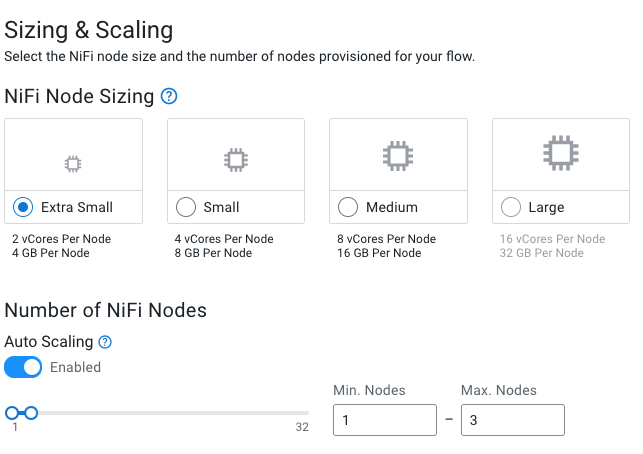
* **CDP Workload User** - The workload username for the current user
* **CDP Workload Password** - The workload password for the current user
* **Kafka Broker Endpoint** - A comma separated list of Kafka Brokers.
* **Kafka Destination Topic -** syslog
* ***Kafka Producer ID*** *- nifi\_dfx\_p1*
* **Schema Name** - syslog
* **Schema Registry Hostname** - The hostname of the master server in the Kafka Datahub. Do NOT use the URL hostname for schema registry, that one is for Knox.
* **Filter Rule -** SELECT \* FROM FLOWFILE

**Note:** The only difference between the parameter entries in CDF-PC as compared

to NiFi Datahub is the Kafka Producer ID

1. On the next page, define the Sizing and Scaling as follows

* **Size:** Extra Small
* **Enable Auto Scaling:** True
* **Min Nodes:** 1
* **Max Nodes:** 3



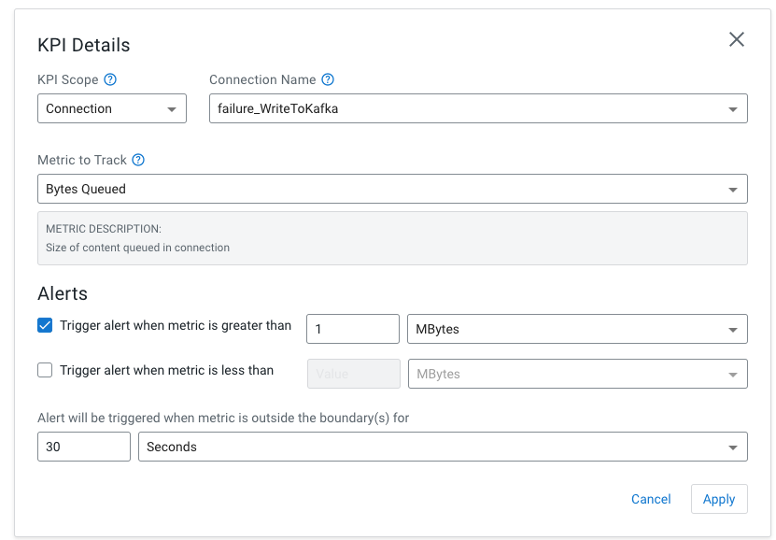
1. On the next page, select the Add New KPI Option.



1. Add the following KPI

* **KPI Scope:** Connection
* **Connection Name:** failure\_WriteToKafka
* **Metrics to Track:** Bytes Queued
* **Alerts:**

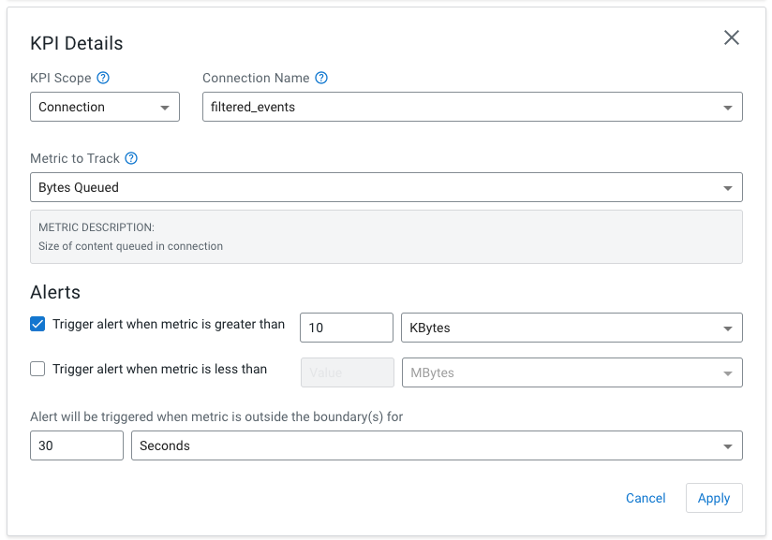
|  |
| --- |
| Trigger alert when metric is greater than 1 MB  Alert will be triggered when metrics is outside the boundary(s) for 30 seconds |



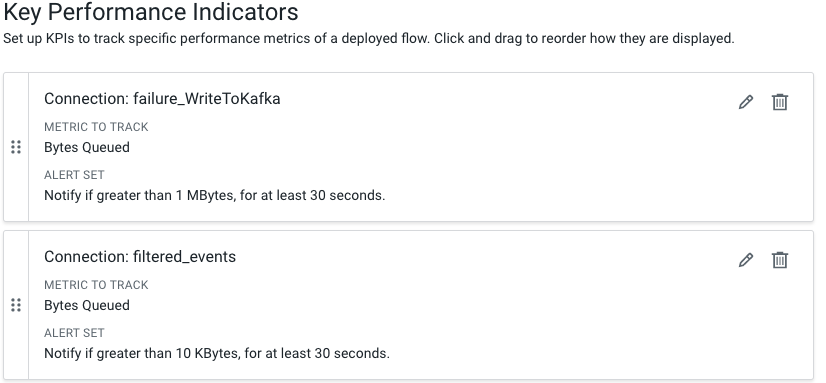
1. Add the following KPI

* **KPI Scope:** Connection
* **Processor Name:** filtered\_events
* **Metrics to Track:** Bytes Queued
* **Alerts:**

|  |
| --- |
| Trigger alert when metric is greater than 10 KB  Alert will be triggered when metrics is outside the boundary(s) for 30 seconds |



1. Click Apply, you will see your defined KPI



1. Click Next, and Review your deployment. Then Click Deploy.



1. Proceed to the CDF-PC Dashboard and wait for your flow to deploy to complete. A Green Check Mark will appear once complete.



1. Click into your deployment and then Click **Manage Deployment** to view metrics, KPI alerts, and autoscaling. The Flow can be examined through the NiFi UI hyperlink.