This document guides students through the Cloudera Data Flow workshop. It will take you step by step to completing the Prerequisites and each Lab activity

Cloudera Data Flow Workshop

[**Pre-requisites**](#_17dp8vu) **6**

[**Lab 0 - Introduction and setup**](#_26in1rg) **7**

[1. Configure permissions in Apache Ranger](#_lnxbz9) 7

[1.1 Kafka Permissions](#_35nkun2) 7

[1.2 Schema Registry Permissions](#_1ksv4uv) 7

[1.3 SQL Stream Builder Permissions](#_44sinio) 8

[1.3.1 Kafka - Streaming Analytics Datahub](#_2jxsxqh) 8

[1.3.2 YARN - Streaming Analytics Datahub](#_z337ya) 9

[1.3.3 Kafka - Streams Messaging Datahub](#_3j2qqm3) 10

[2. Configure Data Providers in SQL Stream Builder](#_1y810tw) 11

[**Lab 1 : Create and Export Data Flow using DataHub**](#_2xcytpi) **12**

[1. Overview](#_1ci93xb) 12

[2. Building the Data Flow](#_3whwml4) 12

[2.1. Create Process Group](#_2bn6wsx) 12

[2.2. Creating the parameter context](#_qsh70q) 13

[2.3. Create the Flow](#_3as4poj) 14

[2.4. Add Parameters to NiFi Flow](#_1pxezwc) 16

[2.5. Naming the queues](#_49x2ik5) 19

[3. Running the Data Flow](#_2p2csry) 20

[4. Exporting the Data Flow](#_147n2zr) 20

[**Lab 2 : Migrating Existing Data Flows to CDF-PC**](#_3o7alnk) **21**

[1. Overview](#_23ckvvd) 21

[2. Running the Workshop](#_ihv636) 21

[2.1. Create a Kafka Topic](#_32hioqz) 21

[2.2. Create a Schema in Schema Registry](#_1hmsyys) 22

[2.3. Upload the Existing NiFi Flow Definition to NiFi Datahub](#_41mghml) 25

[2.4. Discuss the NiFi Flow](#_2grqrue) 27

[2.5. Update Parameter Context](#_vx1227) 30

[2.6. Configure CDP\_Schema\_Registry Controller Service](#_3fwokq0) 32

[2.7. Start the Controller Services](#_1v1yuxt) 32

[2.8. Run the NiFi Flow](#_4f1mdlm) 32

[2.9. Export the NiFi Flow](#_2u6wntf) 33

[**Lab 3 : Operationalizing Data Flows with CDF-PC**](#_19c6y18) **34**

[1. Import the Flow into the CDF-PC Catalog](#_3tbugp1) 34

[2. Deploy the Flow in CDF-PC](#_28h4qwu) 35

[**Lab 4 : SQL Stream Builder**](#_nmf14n) **42**

[1. Overview](#_37m2jsg) 42

[2. Running the workshop](#_1mrcu09) 42

[**Lab 5 : Leveraging Data Flow Catalog in CDF-PC**](#_46r0co2) **46**

[1. Overview](#_2lwamvv) 46

[2. Running the workshop](#_111kx3o) 46

# Pre-requisites

For the ease of carrying out the workshop and considering the time at hand, we have already taken care of some of the steps that need to be considered before we can start with the actual Lab steps. The prerequisites that need to be in place are:

1. Flow Management Data Hub Cluster should be created and running.
2. Streams Messaging Data Hub Cluster should be created and running.
3. Stream analytics Data Hub cluster should be created and running.
4. Data provider should be configured in SQL Stream Builder.
5. Have access to the file syslog-to-kafka.json.
6. Environment should be enabled as part of the CDF Data Service.

## Verify access to the environment

* Provide your public IP address to Cloudera team:

To get the public IP address, you can **follow any of the** following approaches.

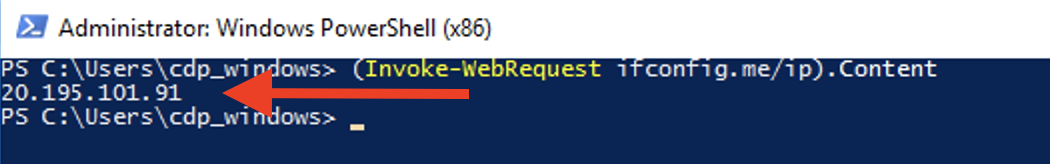
Using a Browser

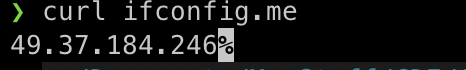
* + Open <https://whatismyipaddress.com/> and IPv4 is what we are looking for.



Using Powershell/Terminal

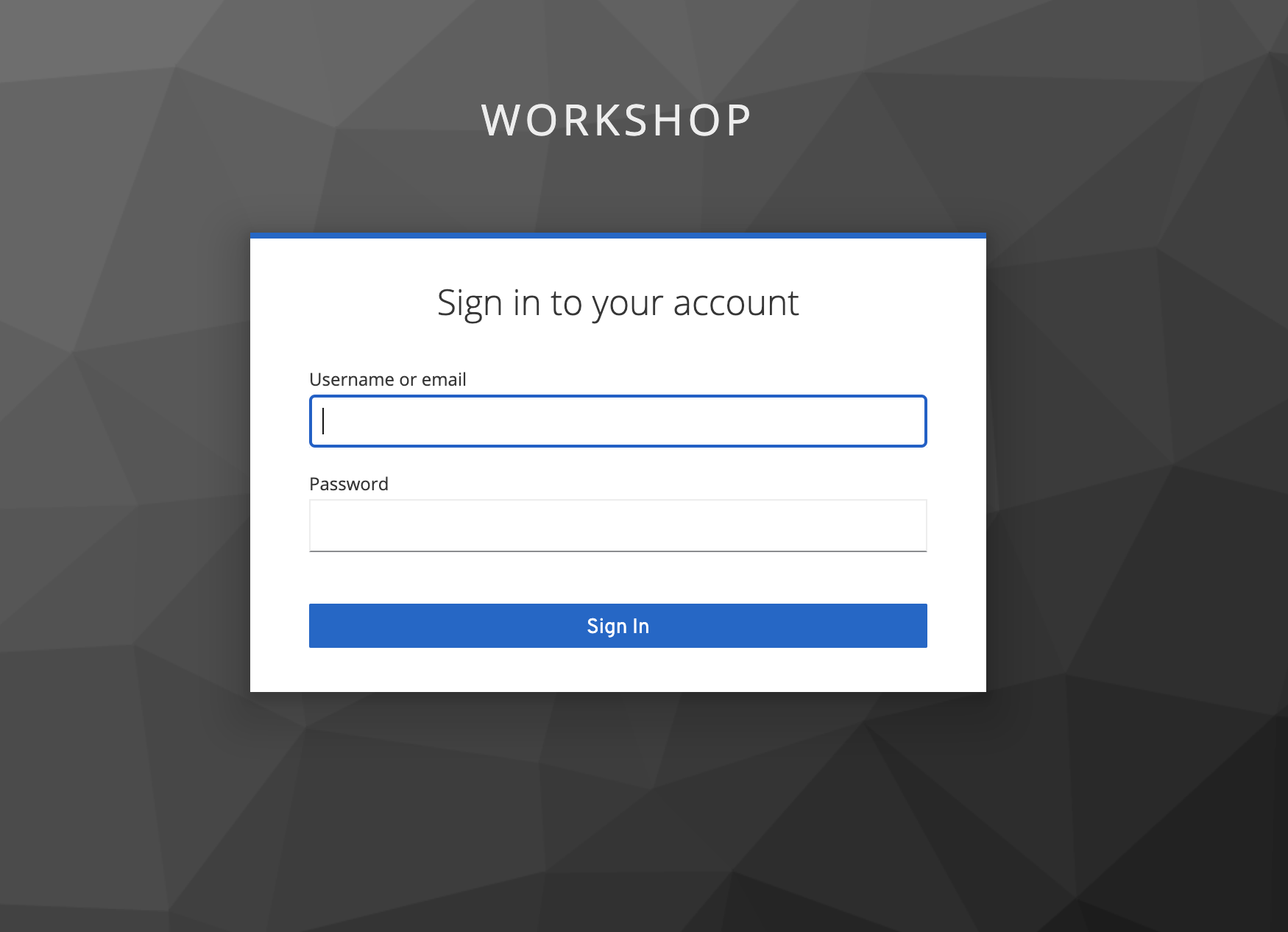
* + **WINDOWS:** Open Powershell and run the below command. You will get the public IP address as the output. Copy the entire command including both the parentheses.  
    (Invoke-WebRequest ifconfig.me/ip).Content



* + **MAC:** Open Terminal and run the below command. You will get the public IP address as the output.  
      
    curl ifconfig.me  
      
    

Open the below link and login with the credentials assigned to you.

[Workshop Login](http://3.109.161.118/auth/realms/workshop/protocol/saml/clients/samlclient)



* You should land on the CDP Console as shown below.



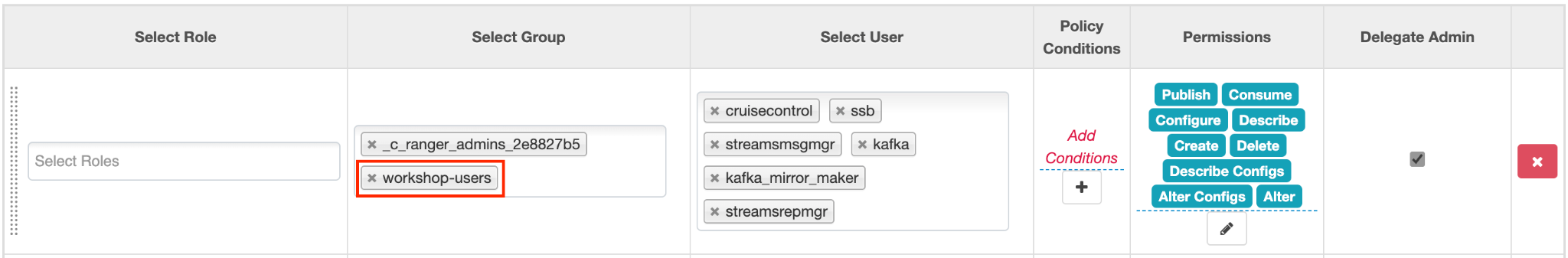
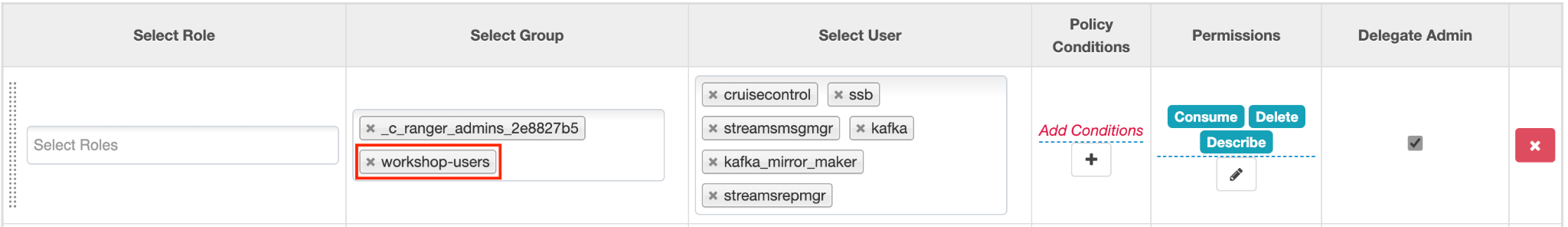
# Lab 0 - Introduction and setup

## 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

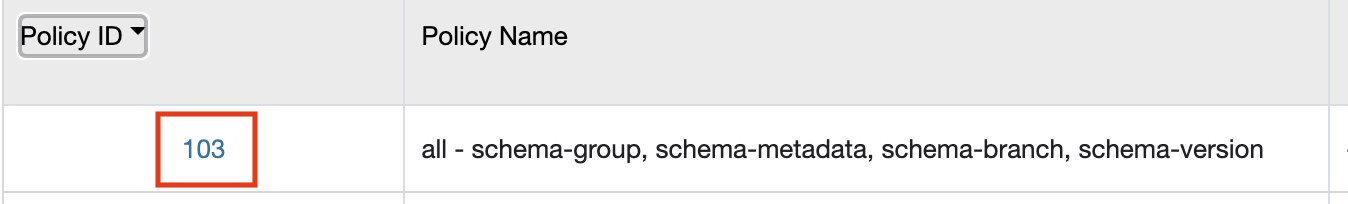


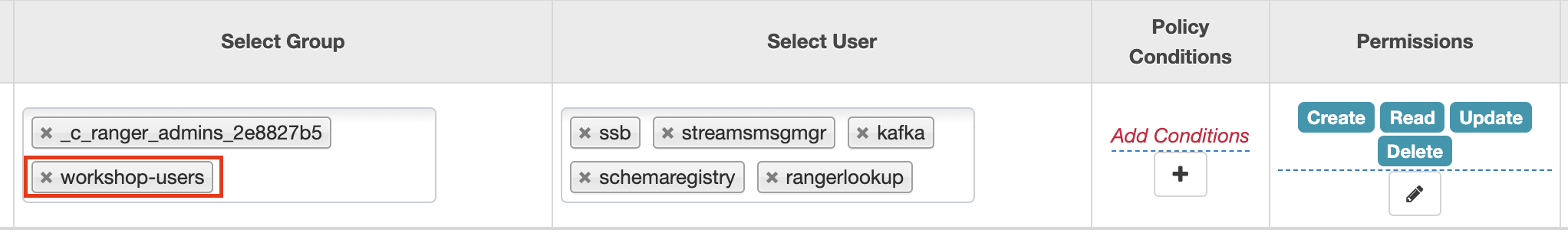


### 1.2 Schema Registry Permissions

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



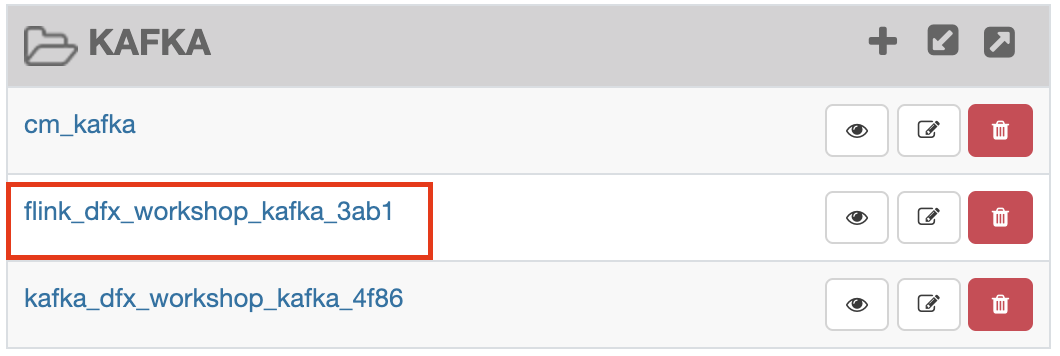
1. Add the user who will be performing the workshop to the existing permissions in the Policy for: **all - schema-group, schema-metadata, schema-branch, schema-version** and click Save.
2. 



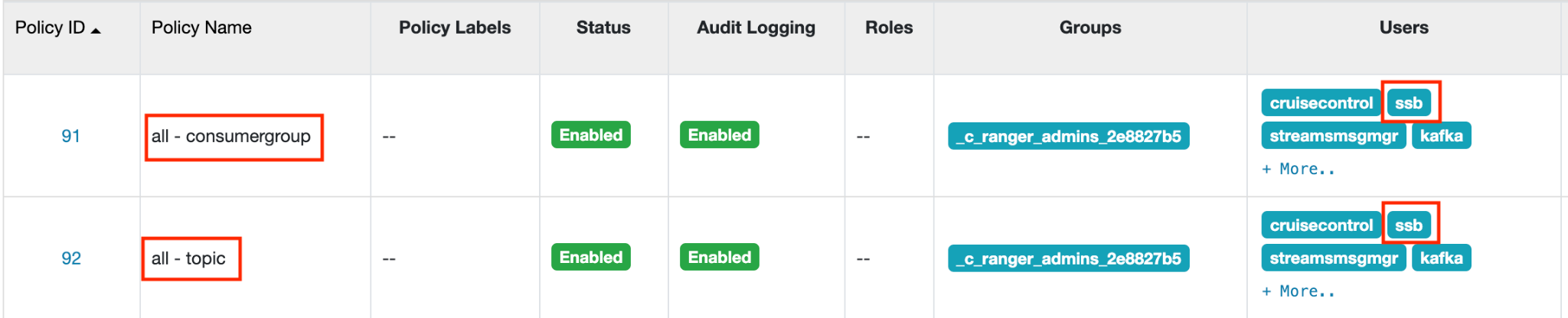
### 1.3 SQL Stream Builder Permissions

#### 1.3.1 Kafka - Streaming Analytics Datahub

1. In Ranger, select the Kafka repository that’s associated with the streaming analytics datahub



1. Add the internal **ssb** user to the existing permissions in both **all-consumergroup** and **all-topic** and click Save, if it hasn’t already been added.

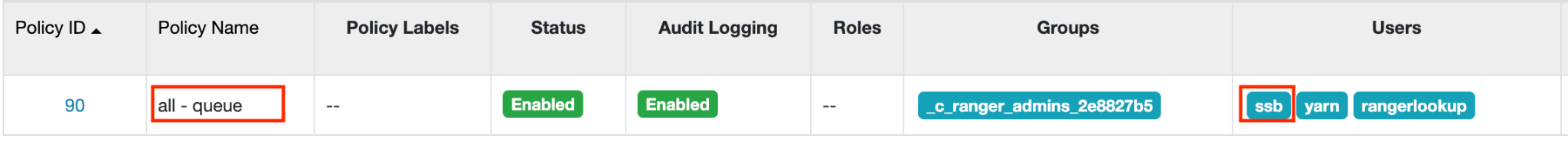


#### 1.3.2 YARN - Streaming Analytics Datahub

1. In Ranger, select the YARN repository that’s associated with the streaming analytics datahub.

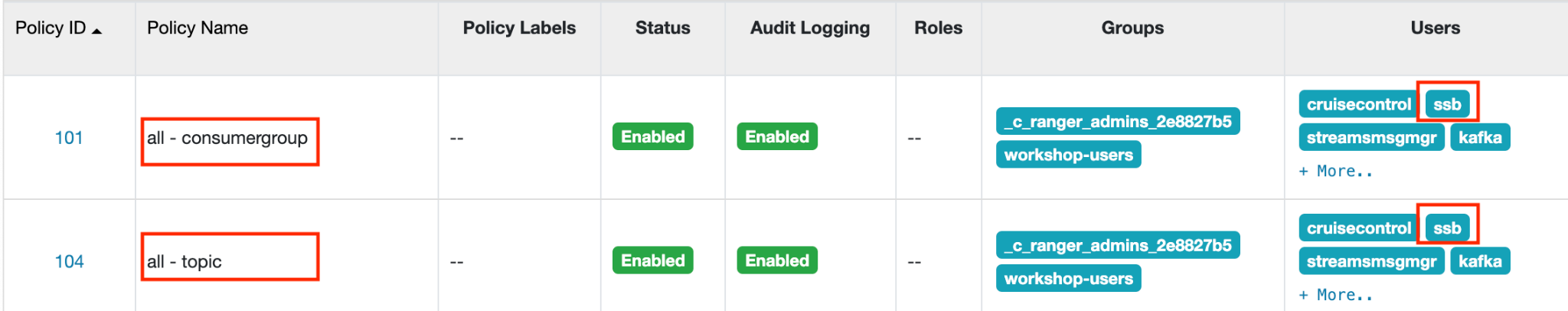


1. Add the internal **ssb** user to the existing permissions in both **all-queue** and click Save, if it hasn’t already been added.



#### 1.3.3 Kafka - Streams Messaging Datahub

1. In Ranger, select the Kafka repository that’s associated with the stream messaging datahub.
2. Add the internal **ssb** user to the existing permissions in both **all-consumergroup** and **all-topic** and click Save, if it hasn’t already been added.

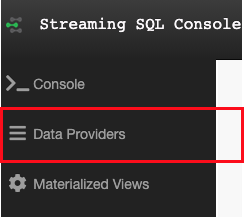


## 2. Configure Data Providers in SQL Stream Builder

1. Open the Streaming SQL Console from the Streaming Analytics datahub

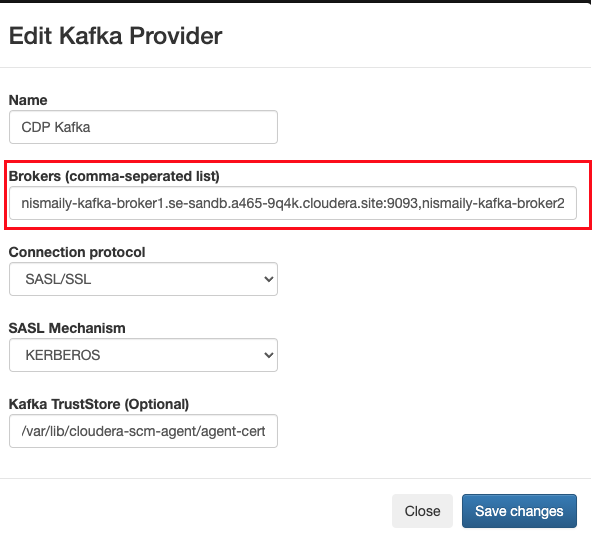


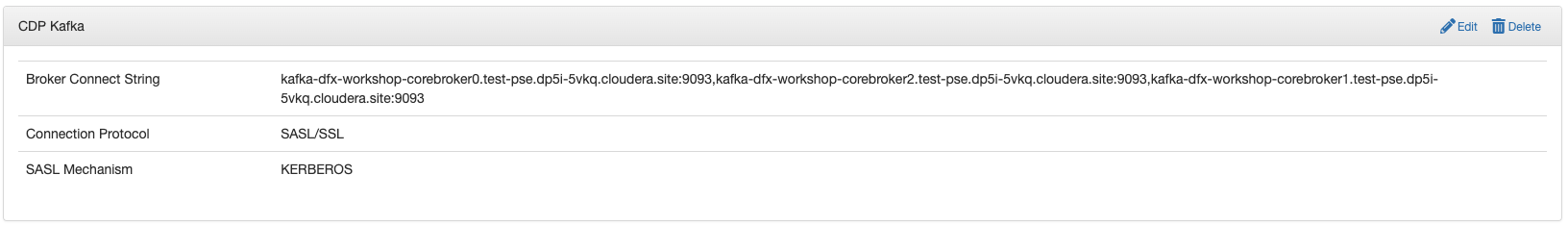
1. Click on Data Providers



1. Edit the **Brokers** in the CDP Kafka Data Provider

The default brokers point to an internal Kafka cluster deployed as part of the streaming analytics datahub. Update the broker list to point to your brokers in the streams messaging datahub. Configure the appropriate Connection Protocol and SASL Mechanism. Broker information can be obtained from the Brokers tab in Streams Messaging Manager.





# 

# Lab 1 : Create and Export Data Flow using DataHub

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

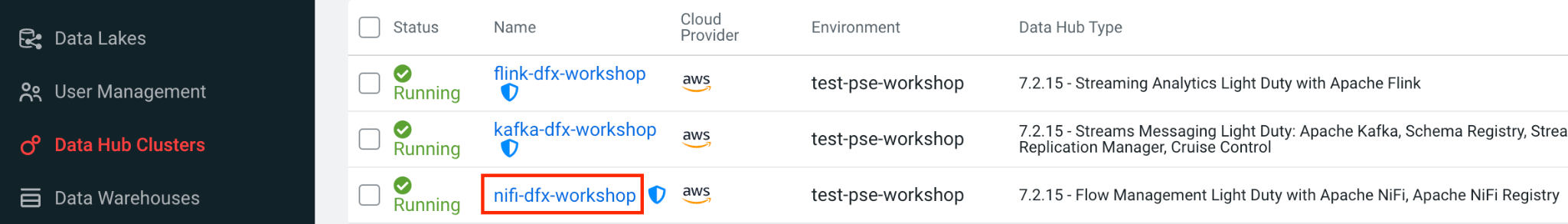
## 1. Overview

Creating a data flow for CDF-PC is the same process as creating any data flow within Nifi with 3 very important steps:

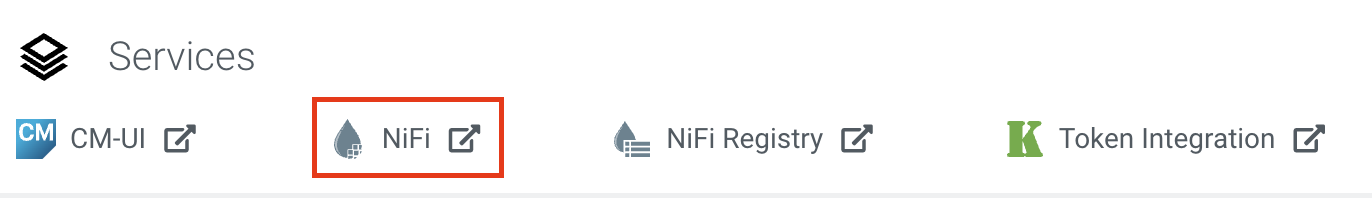
* The data flow that would be used for CDF-PC must be self contained within a process group
* Data flows for CDF-PC must use parameters for any property on a processor that is modifiable, e.g. user names, Kafka topics, etc.
* All queues need to have meaningful names (inplace of Success, Fail, and Retry). These names will be used to define Key Performance Indicators in CDF-PC.

The following is a step by step guide in building a data flow for use within CDF-PC.

We’ll be using the Nifi Data Hub to build our flows.



Click on the NiFi service to get started.

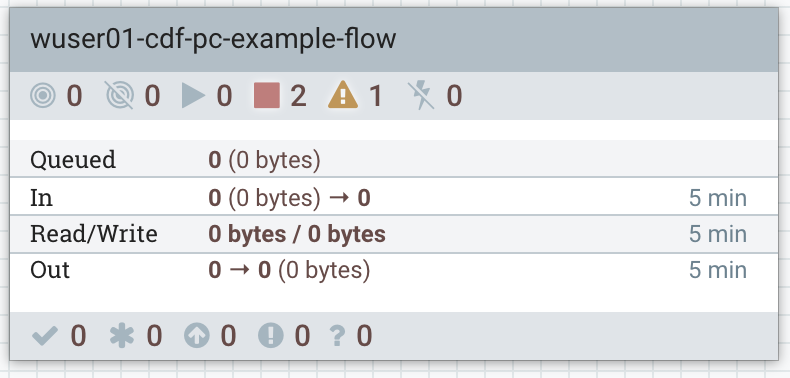


## 2. Building the Data Flow

### 2.1. Create Process Group

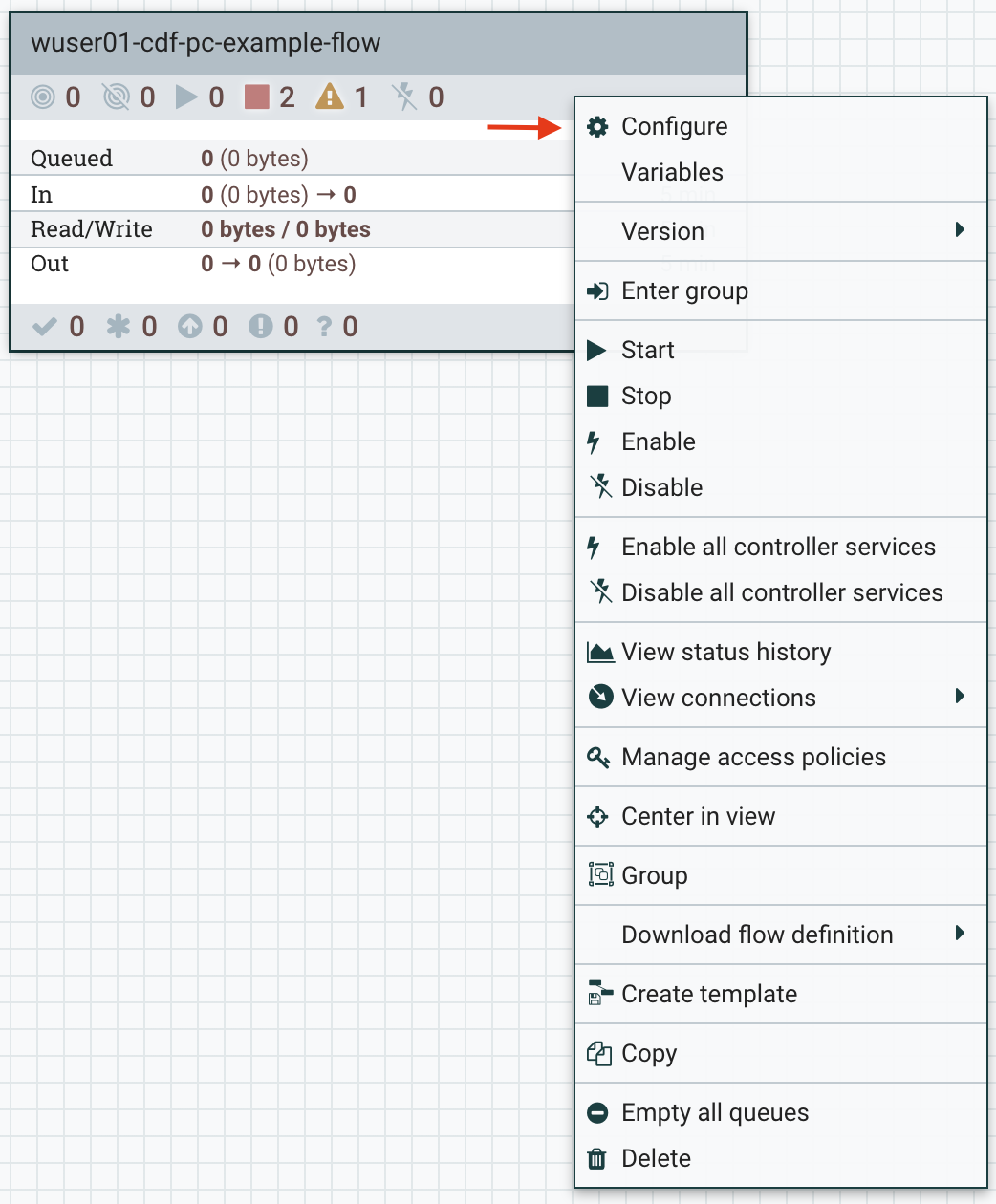
Drag a process group onto the NiFi palette and name it: **<username>-cdf-pc-example-flow**

**Example: wuser01-cdf-pc-example-flow**

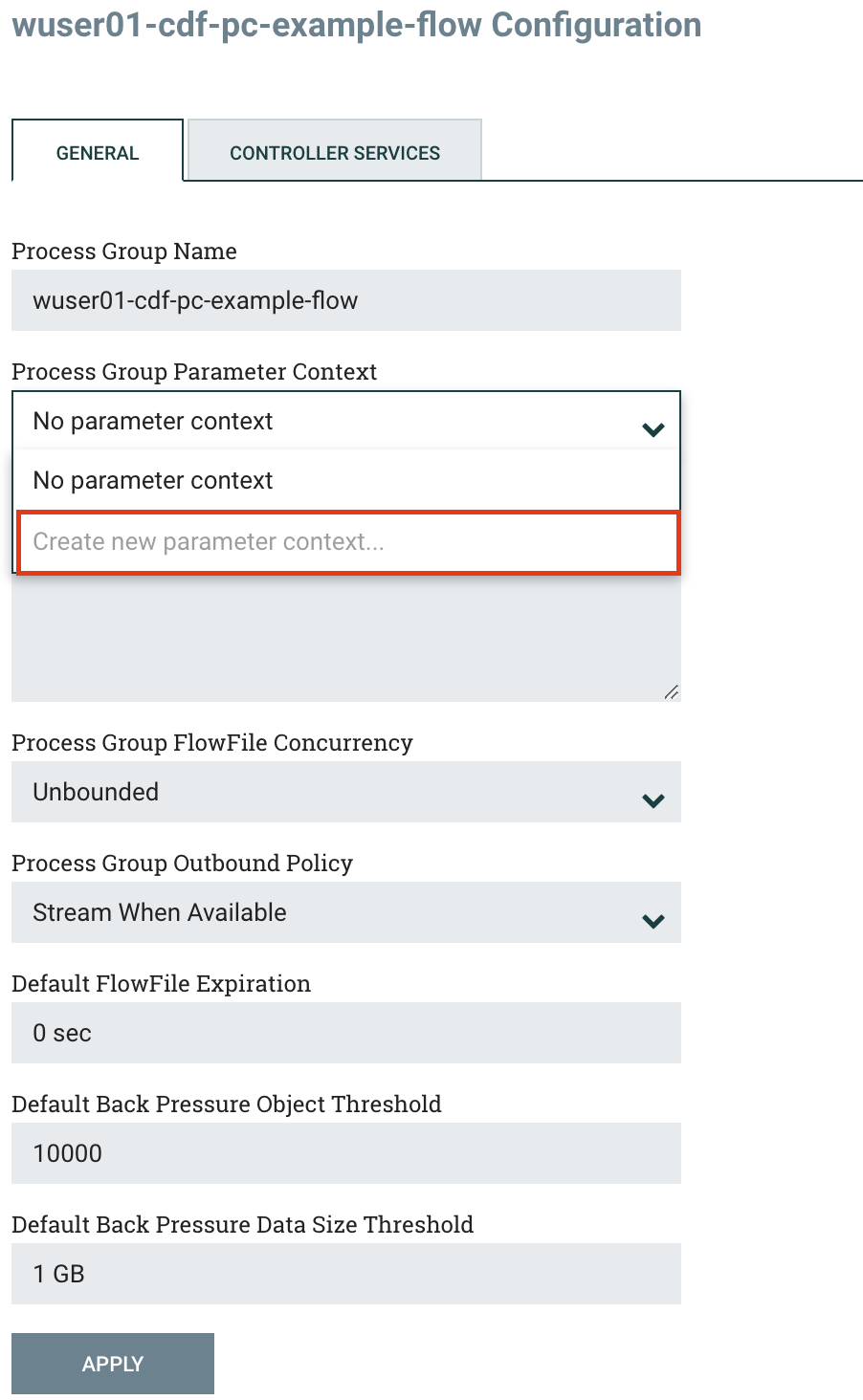


### 2.2. Creating the parameter context

The next step is to configure what is called a parameter context. You can create many parameter contexts within NiFi but only 1 parameter context can be assigned to a given process group. To create a parameter context, right click on the palette and click Configure:

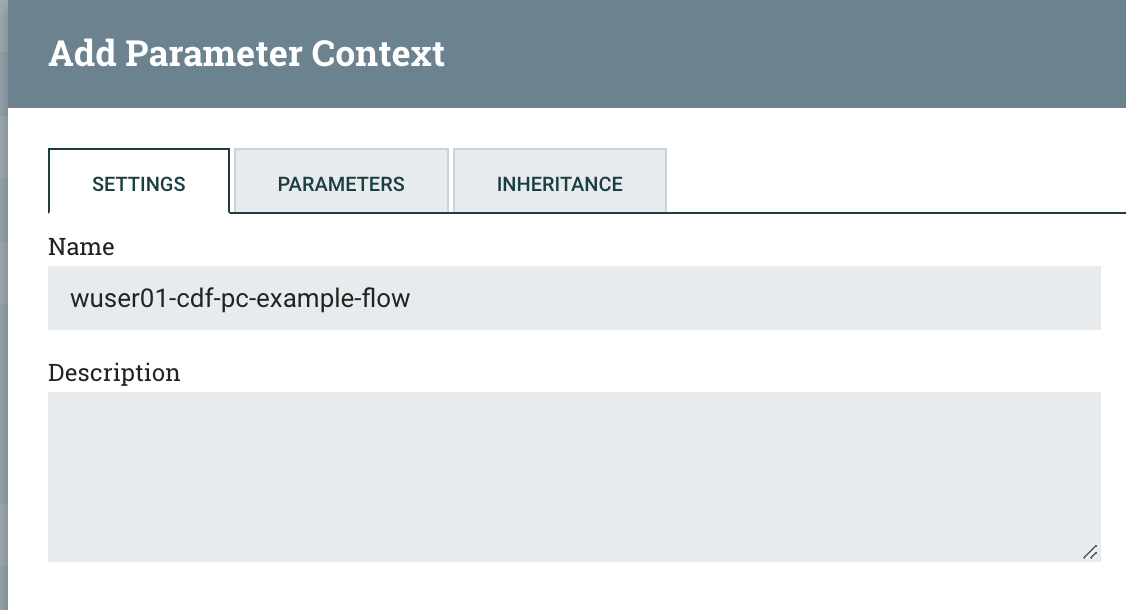


Next, you select create new parameter context under the general tab:



Give the parameter context the same name as the process group: **wuser01-cdf-pc-example-flow**

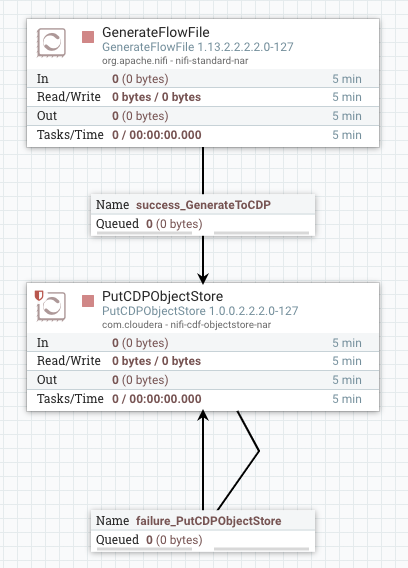
Then hit the APPLY button and close the configuration window.

****

### 2.3. Create the Flow

Double click the process group created above and begin to design the flow. This flow will contain 2 Processors:

* **GenerateFlowFile** - Generates random data
* **PutCDPObjectStore** - Loads data into HDFS



The GenerateFlowFile Processor needs to be configured as follows:

* **Scheduling - Run Schedule:** 30 sec
* **Properties - Custom Text**:

| <26>1 2021-09-21T21:32:43.967Z host1.example.com application4 3064 ID42 [exampleSDID@873 iut="4" eventSource="application" eventId="58"] application4 has  stopped unexpectedly |
| --- |

This represents a syslog out in RFC5424 format. Subsequent portions of this workshop will leverage this same syslog format.

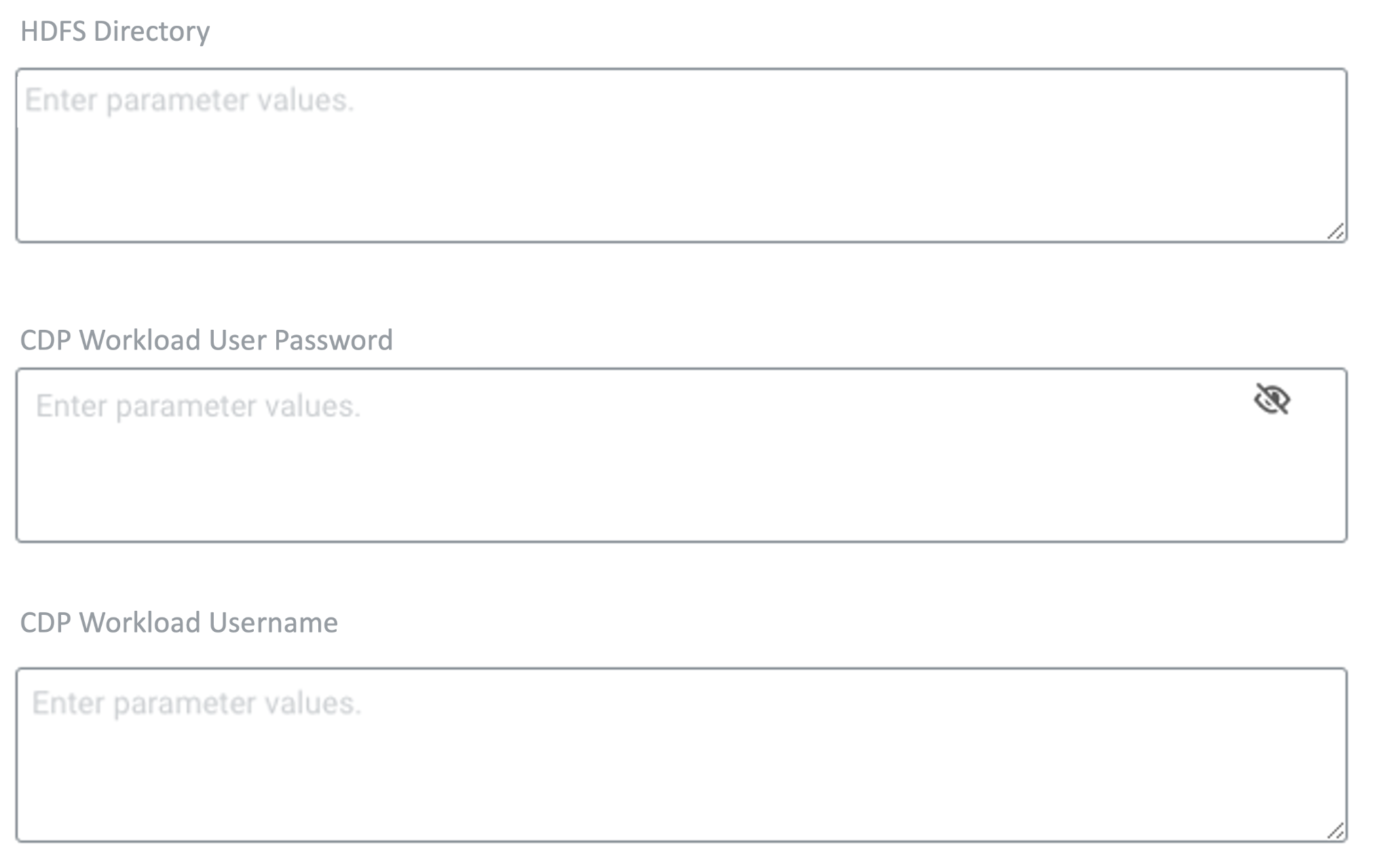
The PutCDPObjectStore Processor needs to be configured as follows:

* **Properties - Directory**: #{HDFS Directory}
* **Properties - CDP Username:** #{CDP Workload User}
* **Properties - CDP Password:** #{CDP Workload User Password}
  + Mark as sensitive property
* **Relationships - Automatically Terminate /Retry Relationships:** Check the Success Terminate box

Notice the #{ } representation, this is the identifying marker for parameters in NiFi.

During the flow deployment process in CDF-PC, every parameterized entry becomes a field that users must fill out. Think of them as globalized variables that can be altered during the deployment of a flow.

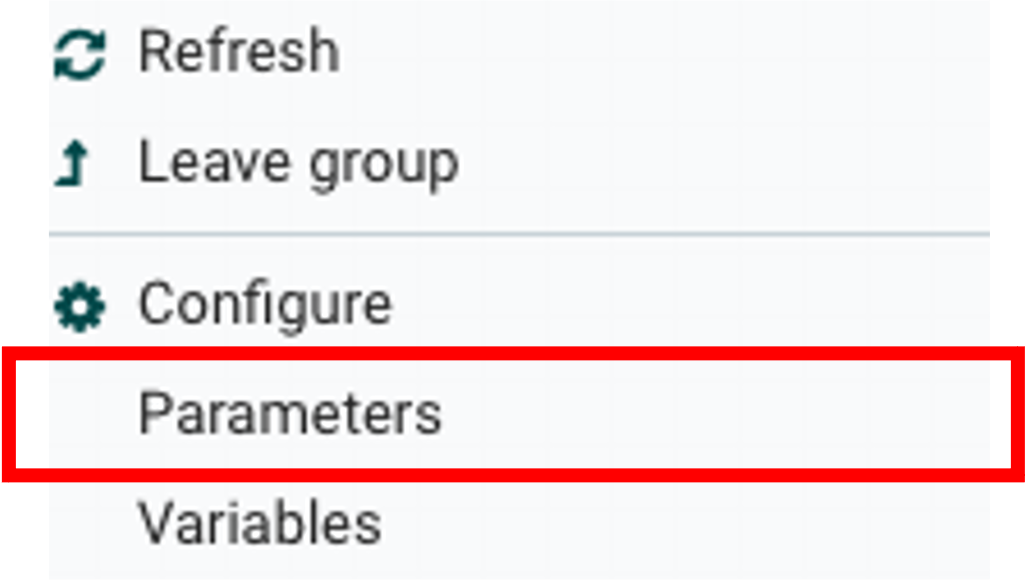
The parameters defined above will result in the following during deployment within CDF-PC:



### 2.4. Add Parameters to NiFi Flow

Now that your flow has been parameterized, you will need to add some additional information about these parameters.

First, Right click on the canvas and select Parameters:



Next, you select the parameters tab and add your parameters to the context. To add a parameter, you hit the PLUS button:



You will need to add parameters with the exact same name as you defined in the NiFi Flow. Recall the parameters we configured earlier:

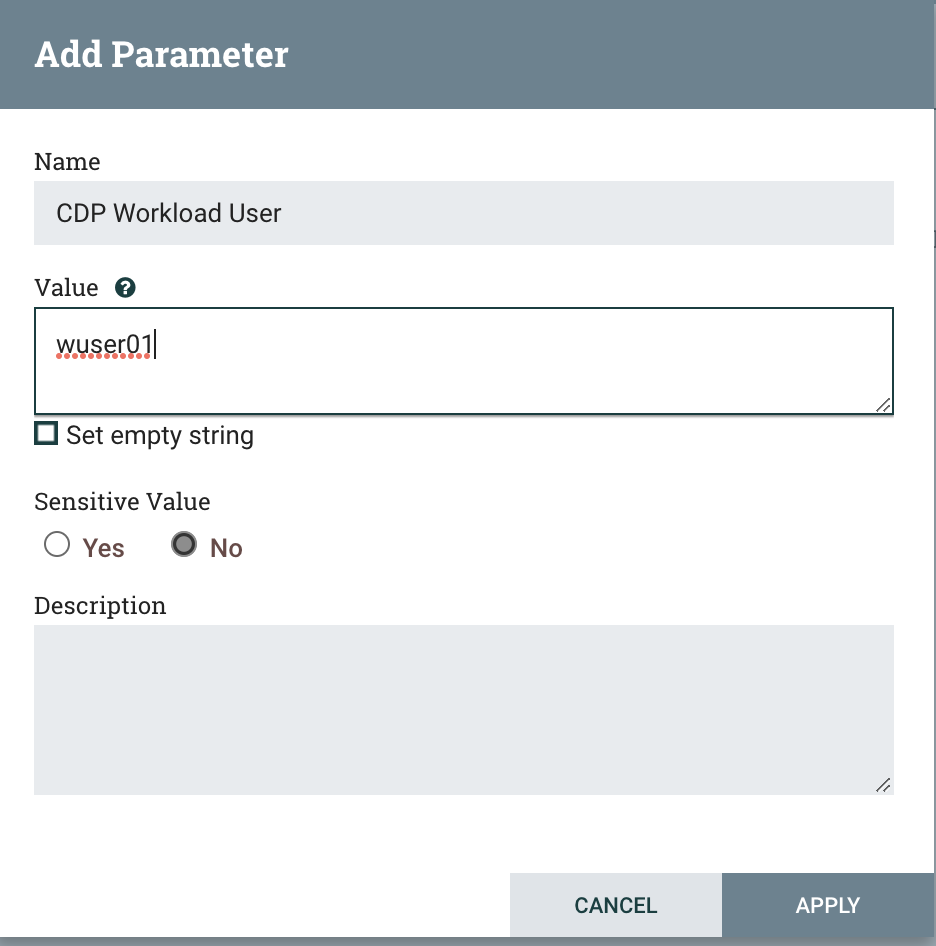
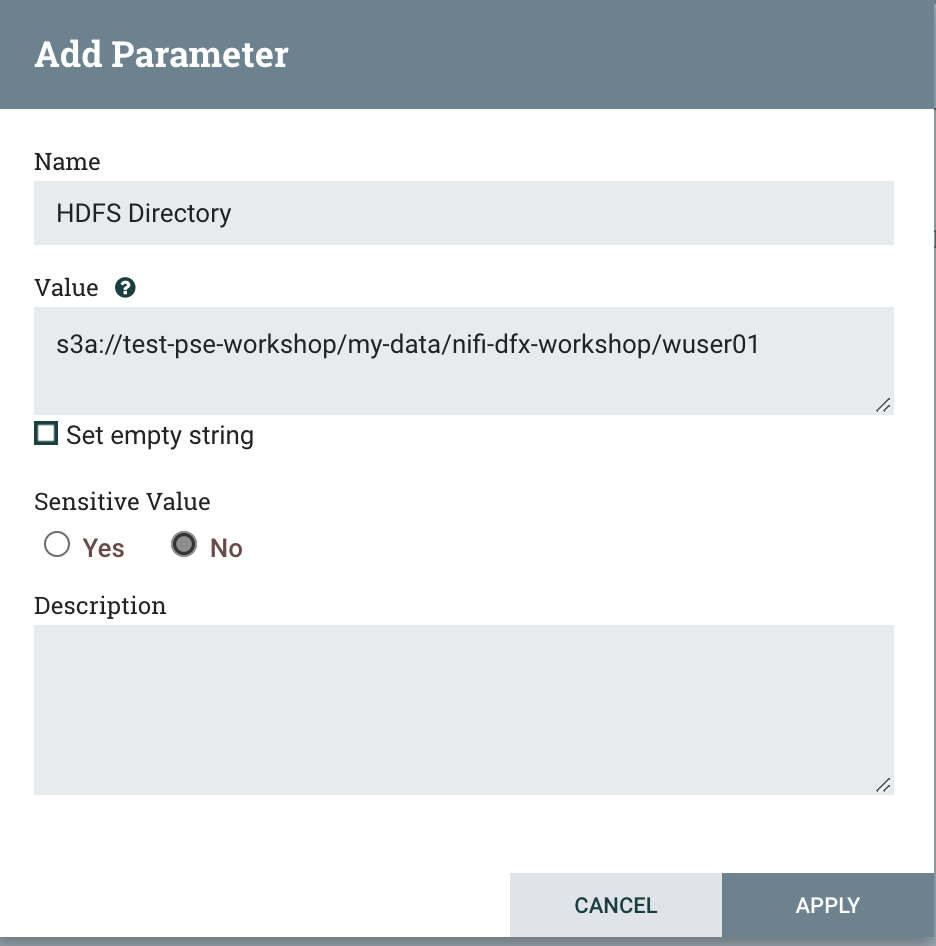
* #{HDFS Directory}
* #{CDP Workload User}
* #{CDP Workload User Password}

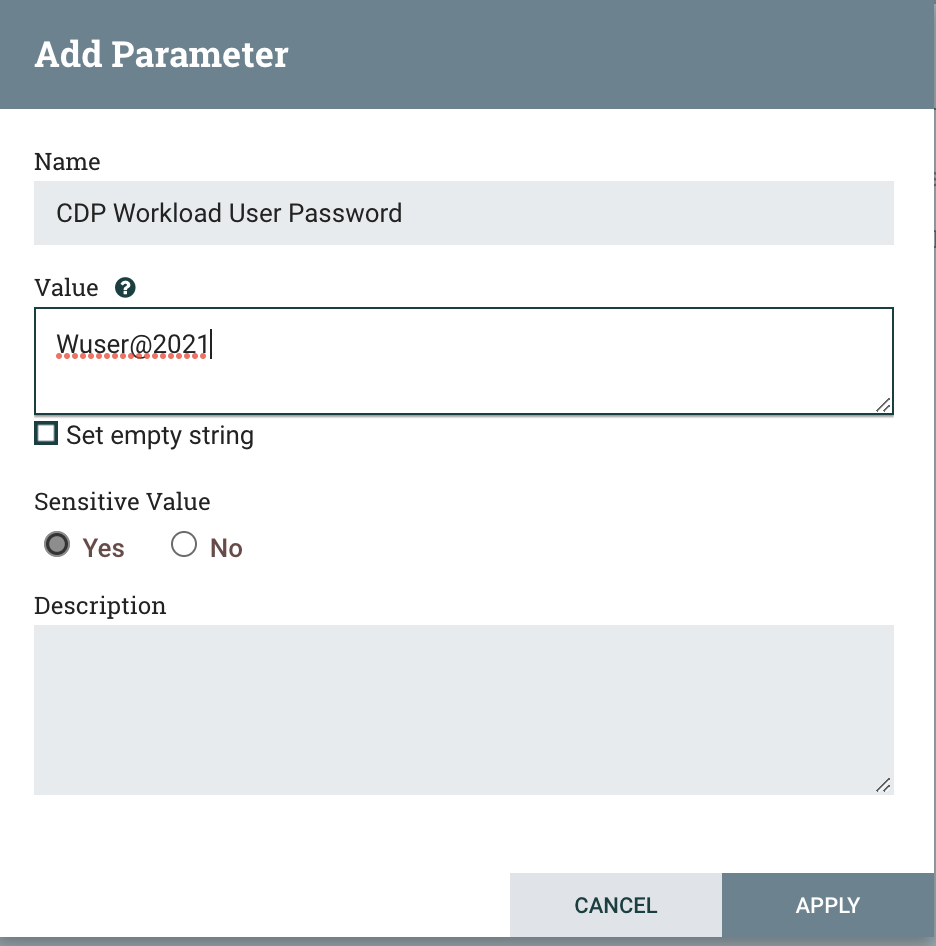
When adding parameters, the information that is entered in the **Value** field will appear in CDF-PC as the default entry during flow deployment.

The **Sensitive Value** field needs to be selected for any information that should be obfuscated from the end users (e.x, passwords).

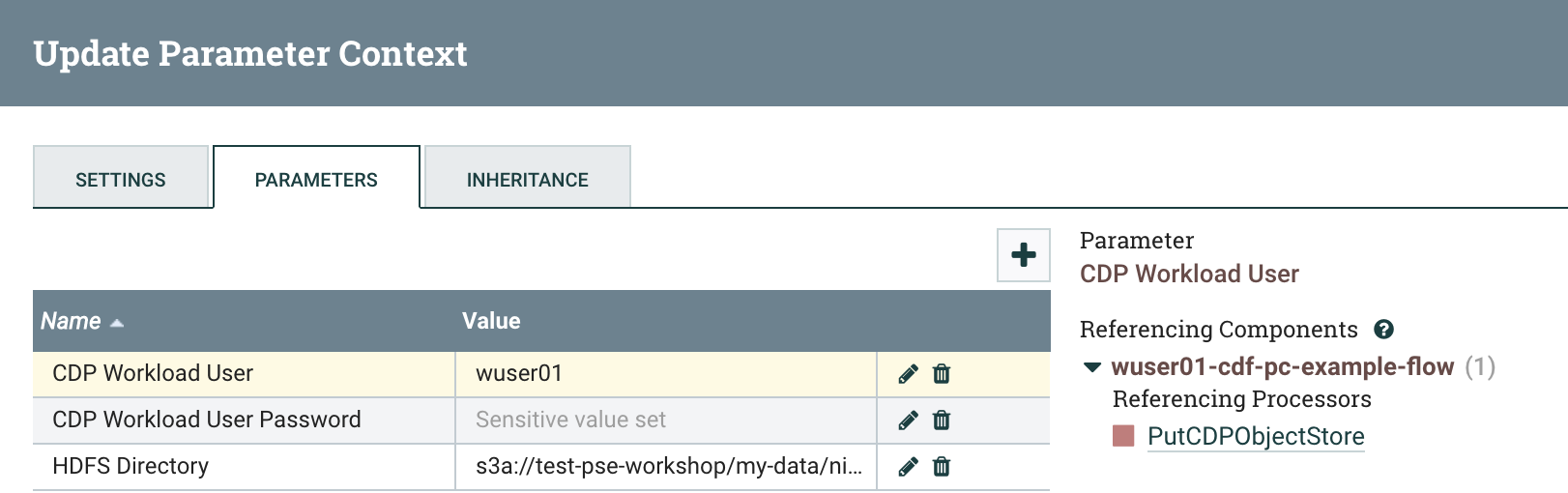
The information added to the **Description** field will also appear to the end user during flow deployment.

Here are example configurations for the parameters:





Once you are done adding parameters to the context, click Apply. The finished process context would look something like this:

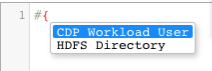


Now that we have created these parameters, we can easily search and reuse them within our dataflow. This is especially useful for **CDP Workload User** and **CDP Workload User Password**.

To search for existing parameters:

1. Open a processor's configuration and proceed to the properties tab.
2. Enter: **#{**
3. Hit ‘control+spacebar’

This will bring up a list of existing parameters that are not tagged as sensitive:



### 2.5. Naming the queues

Providing unique names to all queues is a very important concept for DataHub to CDF-PC migration. The queue names are used to define Key Performance Indicators upon which CDF-PC will auto-scale.

To name a queue, double-click the queue and give it a unique name. A best practice here is to start the existing queue name (i.e. success, failure, retry, etc…) and add the source and destination processor information.

For example, the success queue between GenerateFlowFile and PutCDPObjectStore is named **success\_GenerateToCDP.** The failure queue for PutCDPObjectStore is named **failure\_PutCDPObjectStore**.

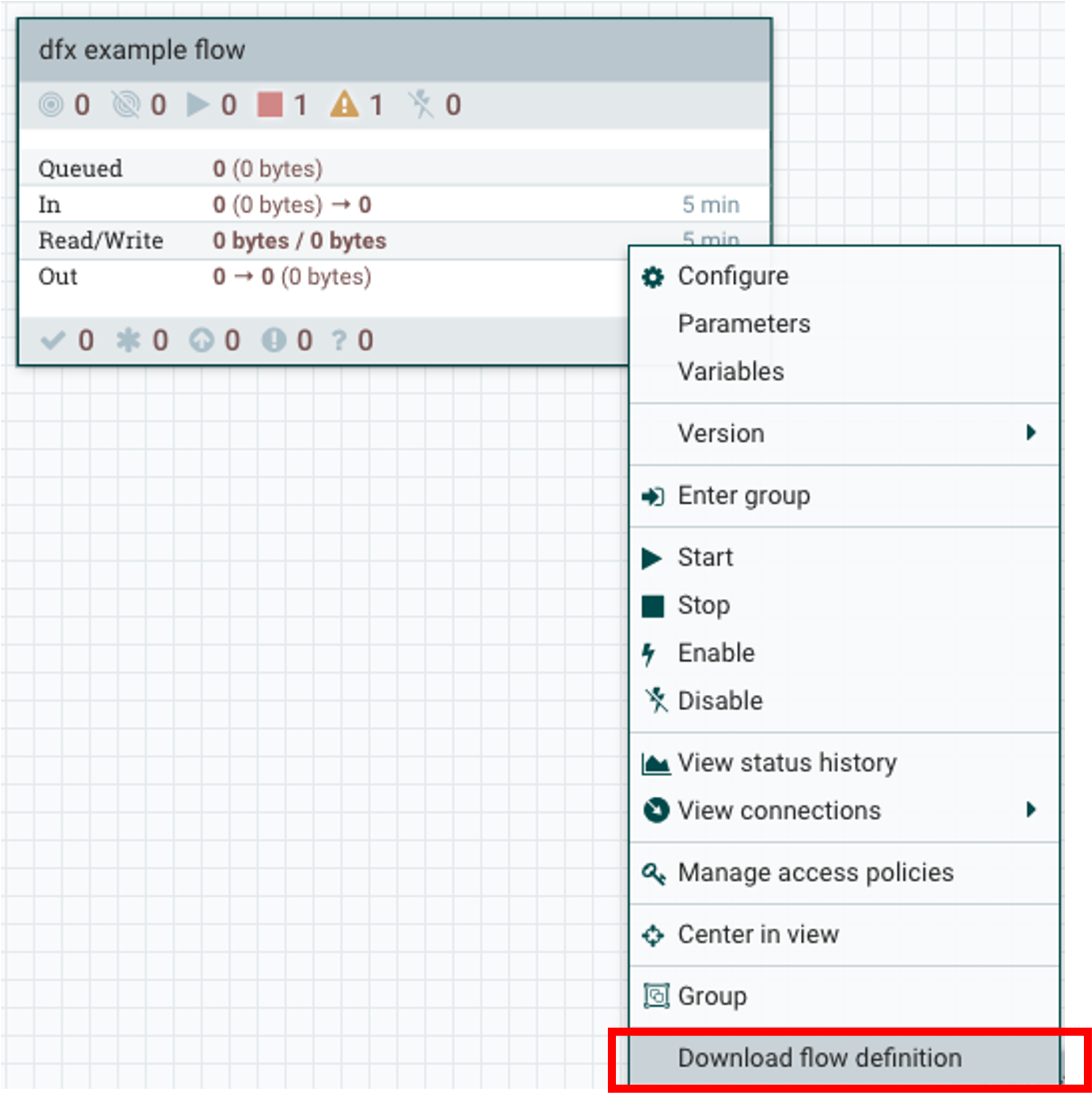
 

## 3. Running the Data Flow

When you run the flow, please note that you may get an error on the PutCDPObjectStore if the specified path does not exist in s3. This will happen for the first file load. To circumvent this, create the appropriate directory in s3 before running the flow.

## 4. Exporting the Data Flow

After the flow has been created and tested in DataHub, we can export it as a Flow Definition. The Flow Definitions can later be imported into CDF-PC for deployment. To export a Flow, go back out to the process group view, right click the process group and select ‘Download Flow Definition’.



# Lab 2 : 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.

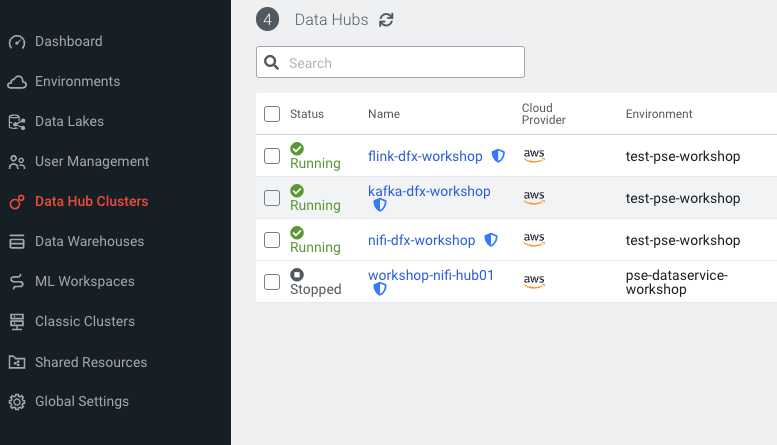
The existing NiFi Flow will perform the following actions:

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.

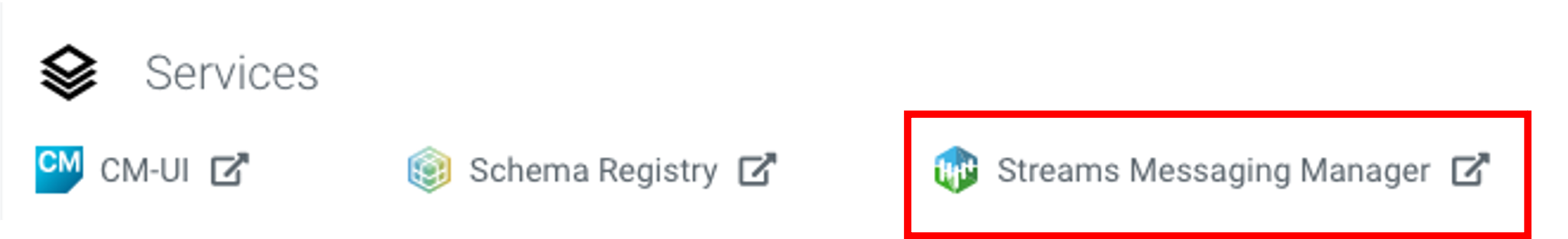
## 2. Running the Lab

Click on Data Hub Cluster on the Management Console and Click on the kafka-dfx-workshop Data Hub



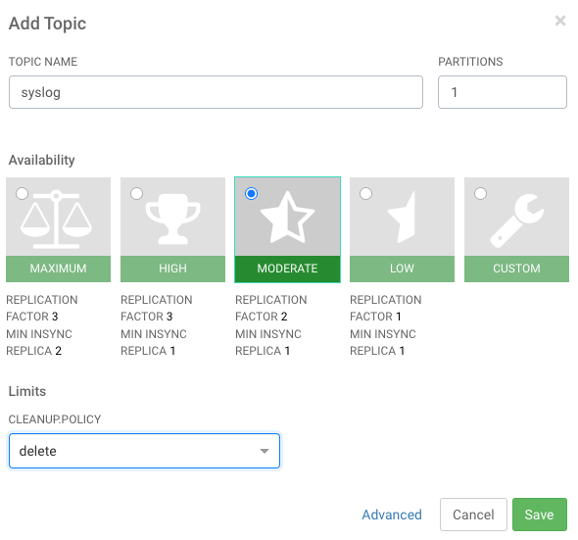
### 2.1. Create a Kafka Topic

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:

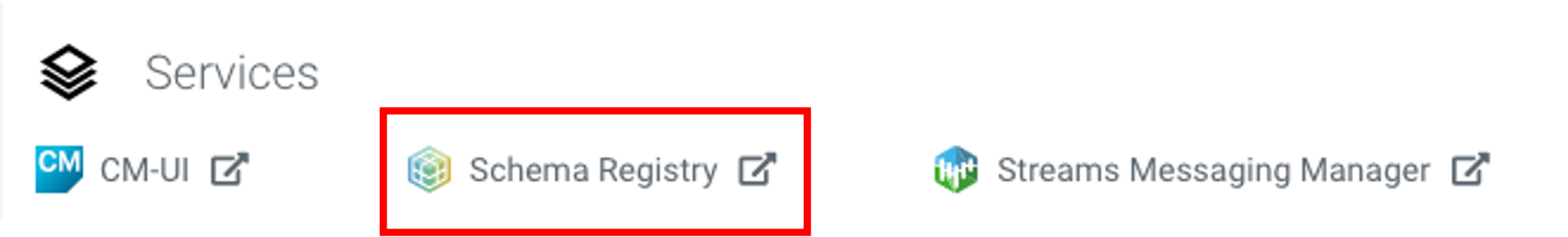
* **Name**: <username>-syslog
* **Partitions**: 1
* **Availability**: Moderate
* **Cleanup Policy**: Delete



**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"  }  ]  }  }  ]  }  }  ]  } |
| --- |

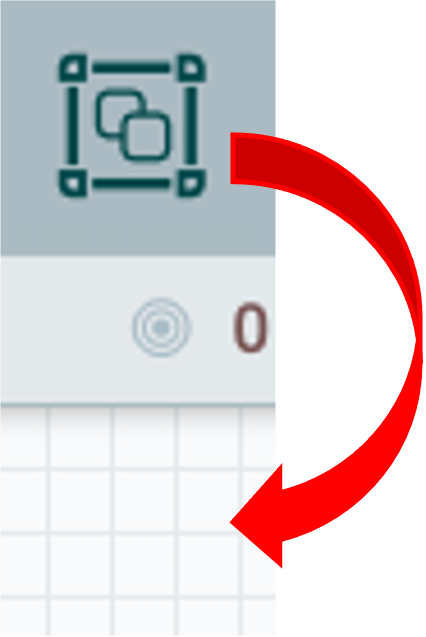
**Note**: The name of the Kafka Topic and the Schema Name must be the same.

### 2.3. Upload the Existing NiFi Flow Definition to NiFi Datahub

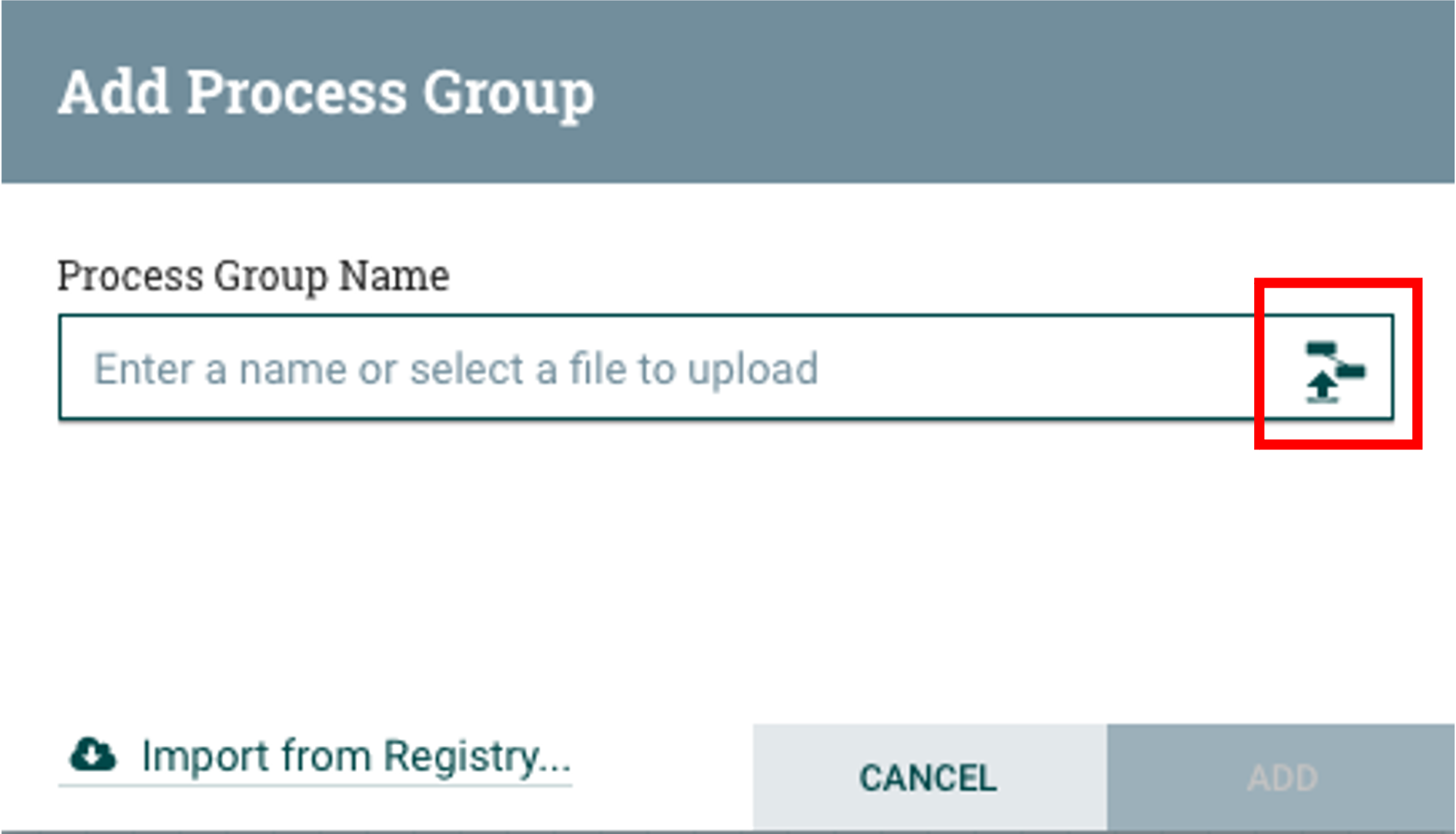
1. Begin by downloading the pre-built **syslog-to-kafka.json** NiFi template from the Cloudera Workshop material.
2. Login to the NiFI Datahub UI by selecting the appropriate hyperlink in the datahub cluster.



1. In the NiFi UI, drag a new process group onto the canvas:

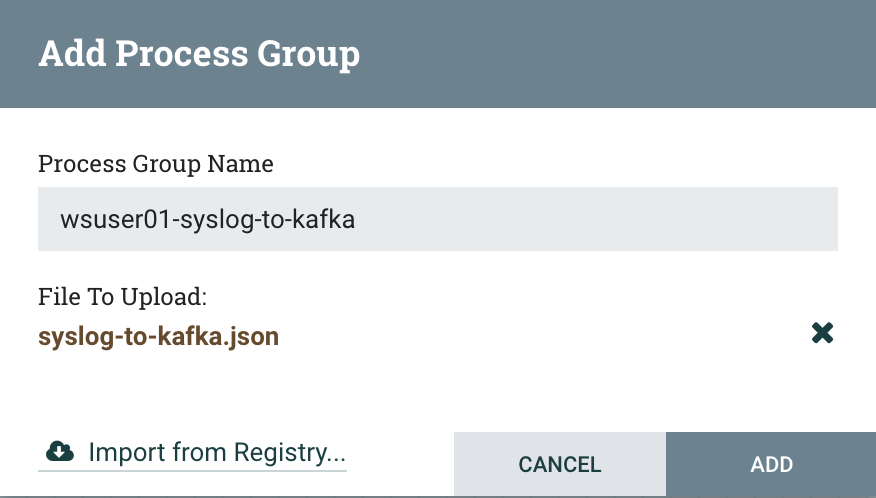


1. Select the upload button

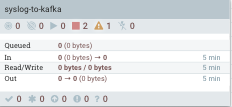


1. Select the **syslog-to-kafka.json** file you downloaded earlier, change the process group name to

**<USERNAME>-syslog-to-kafka** then click ADD.



1. At this point you will see a **<USERNAME>-sylog-to-kafka** process group on the NiFi Canvas. Click into it.



### 2.4. Overview of 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

Downstream, 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:

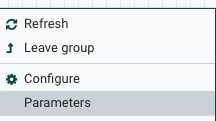
| SELECT \* FROM FLOWFILE WHERE severity <=3 |
| --- |

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.

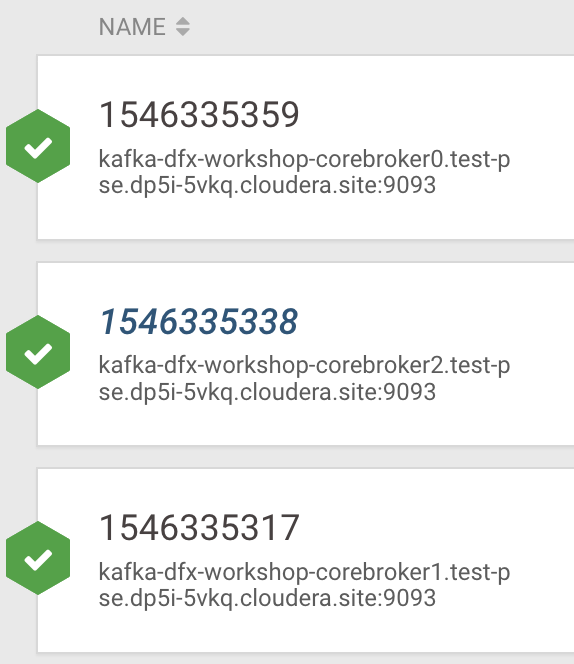
### 2.5. Update Parameter Context

Before we can run the flow, we need to update the parameters within the context to work with the customer's CDP Environment. To do this, right click on the canvas and select parameters.



Next, update each of the existing parameters as follows:

* **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. This can be obtained through Streams Messaging Manager(SMM). Within SMM, click on the brokers tab on the left and copy each of the brokers URL:Port and create a comma separated list. For example, consider the following in SMM:



The Kafka Broker Endpoint parameter would thus be set to:

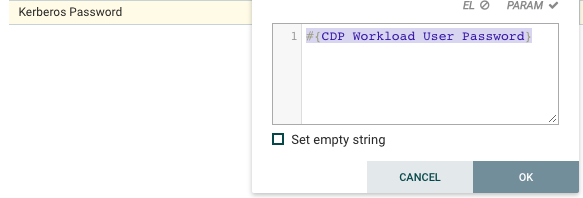
| nismaily-kafka-broker1.se-sandb.a465-9q4k.cloudera.site:9093,nismaily-kafka-broker2.se-sandb.a465-9q4k.cloudera.site:9093,nismaily-kafka-broker0.se-sandb.a465-9q4k.cloudera.site:9093 |
| --- |

* **Kafka Destination Topic - <username>-**syslog
* **Kafka Producer ID** - nifi\_dh\_p1
* **Schema Name** - <username>-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

### 2.6. Configure CDP\_Schema\_Registry Controller Service

When the Flow definition is downloaded the Parameter #{CDP Workload User Password} does not get saved in the Controller Service. To fix this for CDP\_Schema\_Registry do the following:

1. Right Click on the canvas and select Configure
2. Click the gear next to CDP\_Schema\_Registry
3. Update the Kerberos password as follows and hit save

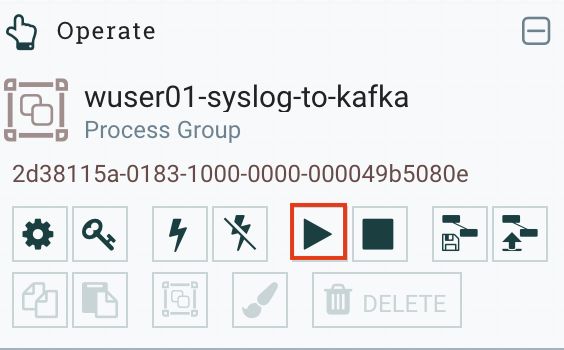


### 2.7. Start the Controller Services

1. Right Click on the canvas and select Configure
2. Click the Enable (Lightning button) on all Controller Services.

### 2.8. Run the NiFi Flow

1. Start the NiFi Flow by clicking on the start button in the operate pane



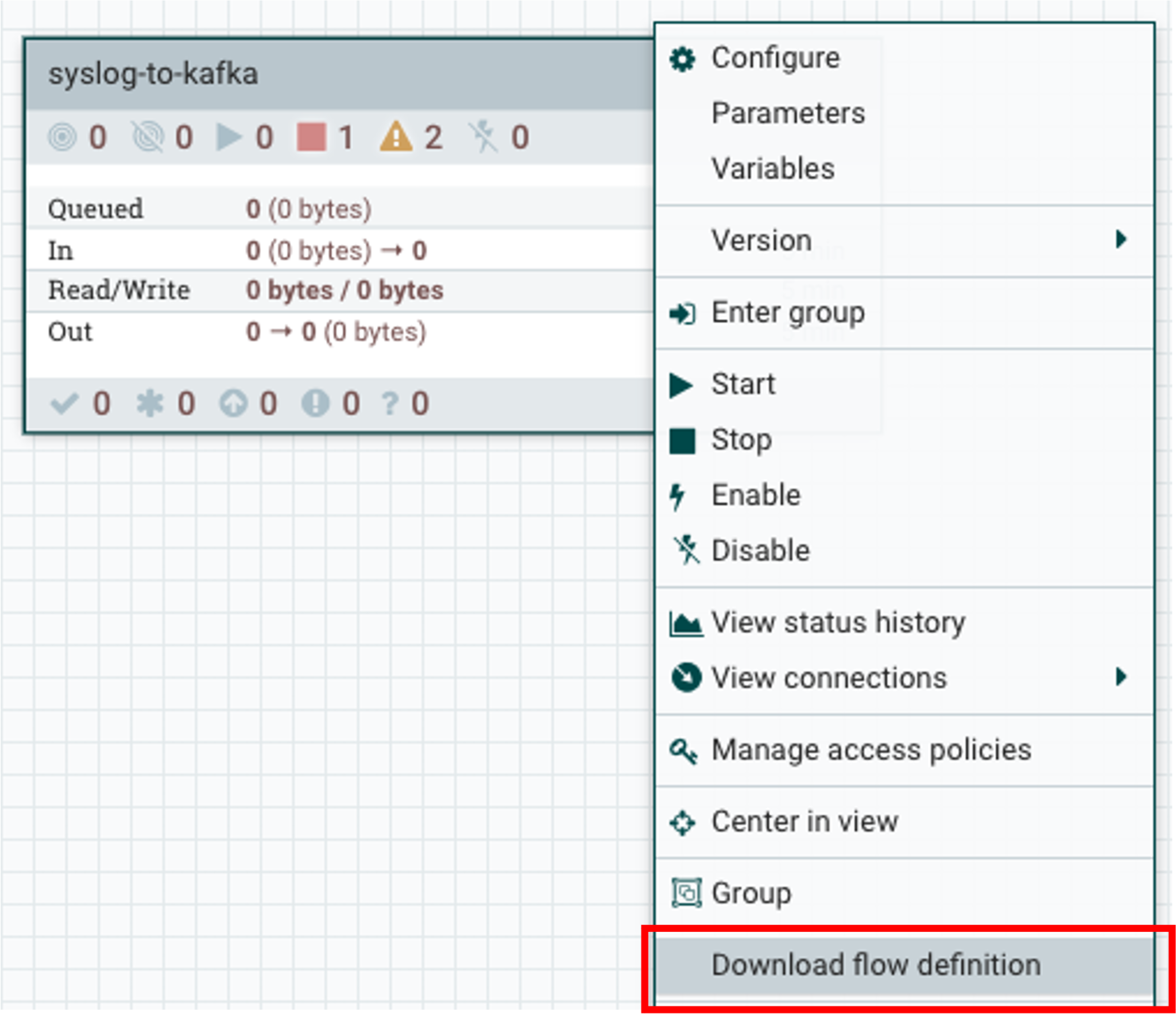
1. Proceed to SMM and examine the contents of the Topic



1. Stop the Flow

### 2.9. Export the NiFi Flow

After the flow has successfully tested in DataHub, we can export it as a Flow Definition. To export a Flow, go back out to the process group view, right click the process group and select ‘Download Flow Definition’.

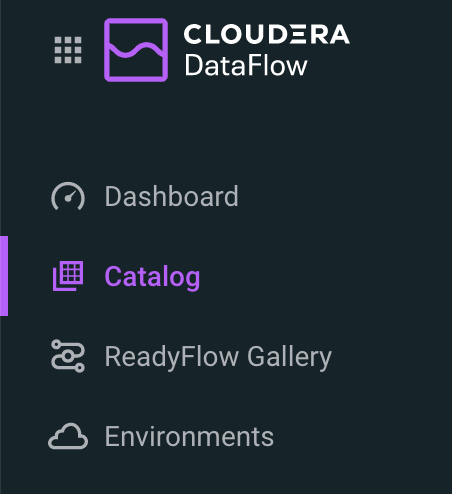


**Note**: We can just reuse the original downloaded Flow Definition. We’re asking the customer to do this so they can practice what they learned in the previous workshop.

# Lab 3 : Operationalizing Data Flows with CDF-PC

## 1. Import the Flow into the CDF-PC Catalog

1. Open the CDF-PC data service and click on Catalog in the left tab.



1. Select Import Flow Definition on the Top Right

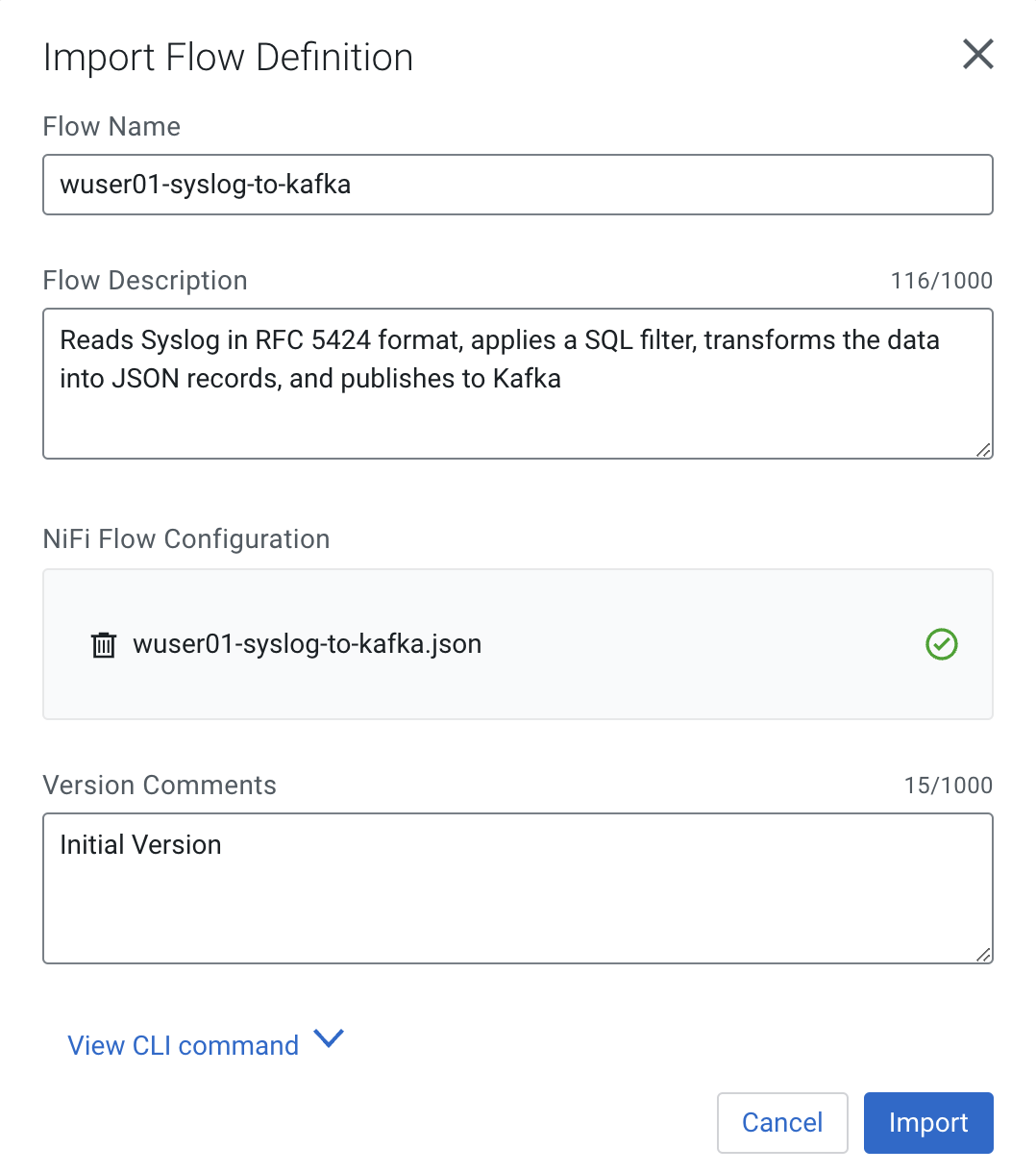


1. Add the following information:

* **Flow Name: <username>-**syslog-to-kafka
* **Flow Description:**

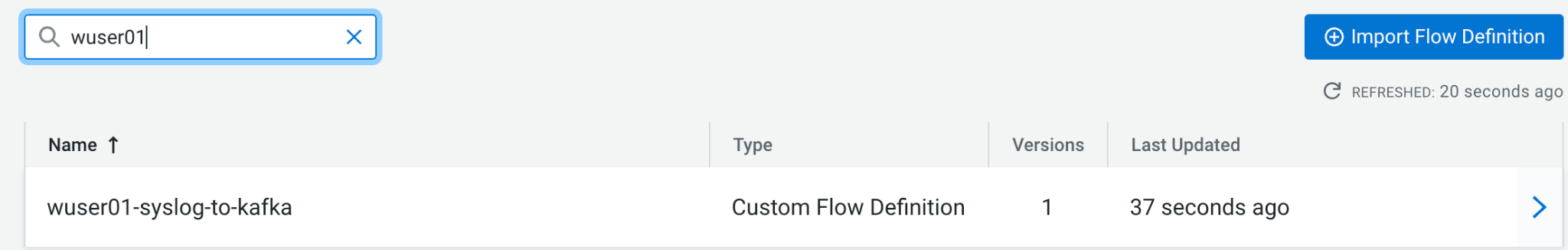
| Reads Syslog in RFC 5424 format, applies a SQL filter, transforms the data into JSON records, and publishes to Kafka |
| --- |

* **NiFi Flow Configuration:** *wuser01-syslog-to-kafka.json* (upload the Flow Definition)
* **Version Comments:** Initial Version

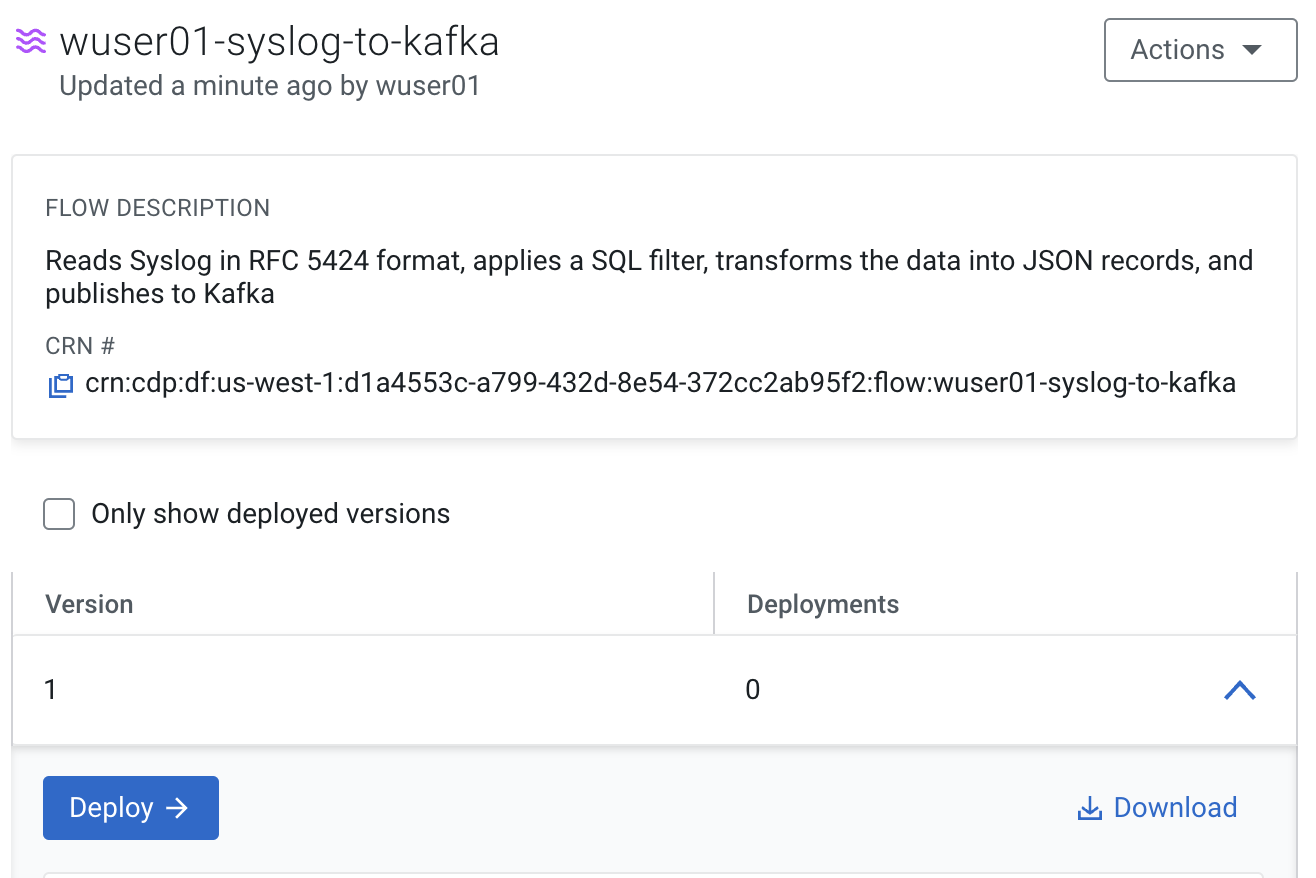


## 2. Deploy the Flow in CDF-PC

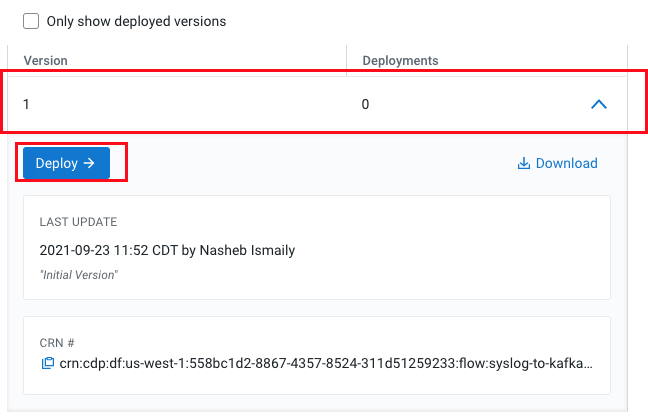
1. Search for the flow in the Flow Catalog



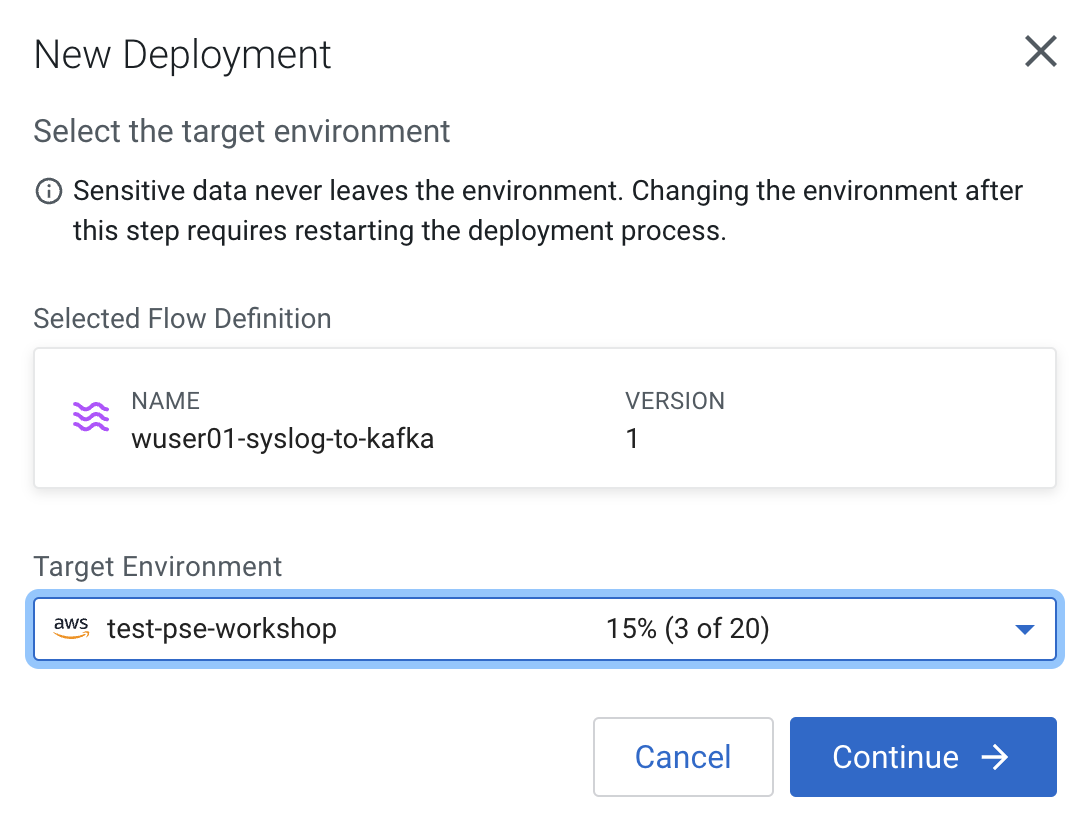
1. Click on the Flow, you should see the following:



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.

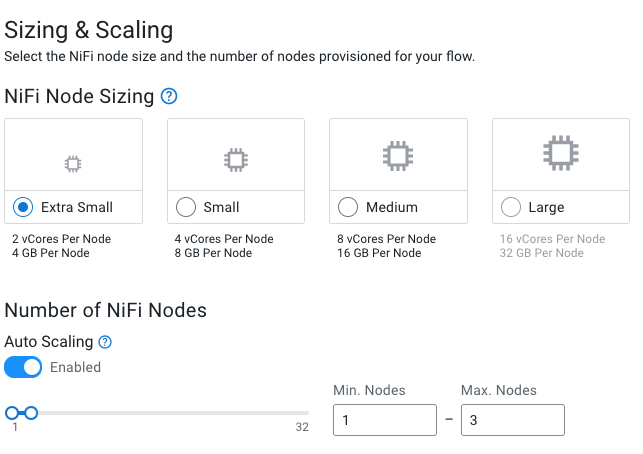
* **CDP Workload User** - The workload username for the current user
* **CDP Workload Password** - The workload password for the current user
* **Kafka Broker Endpoint** - A comma separated list of Kafka Brokers.
* **Kafka Destination Topic -** wuser01-syslog
* ***Kafka Producer ID*** *- nifi\_dfx\_p1*
* **Schema Name** - wuser01-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



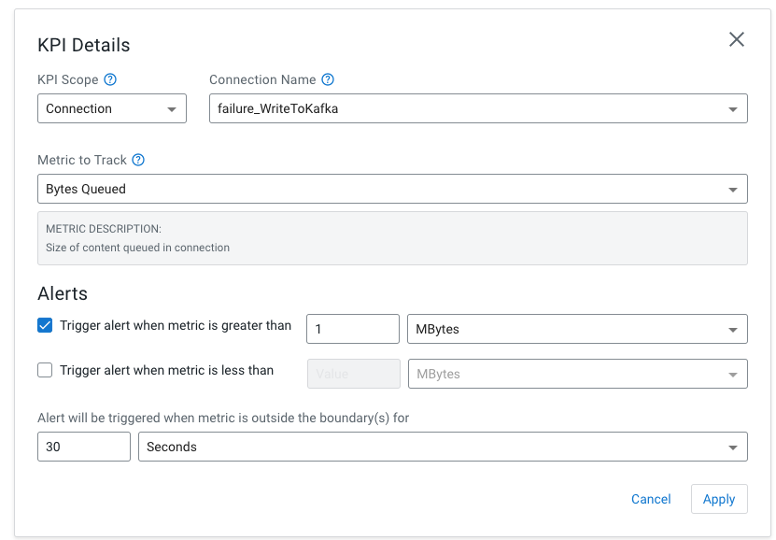
1. On the next page, select the Add New KPI Option.



1. Add the following KPI

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

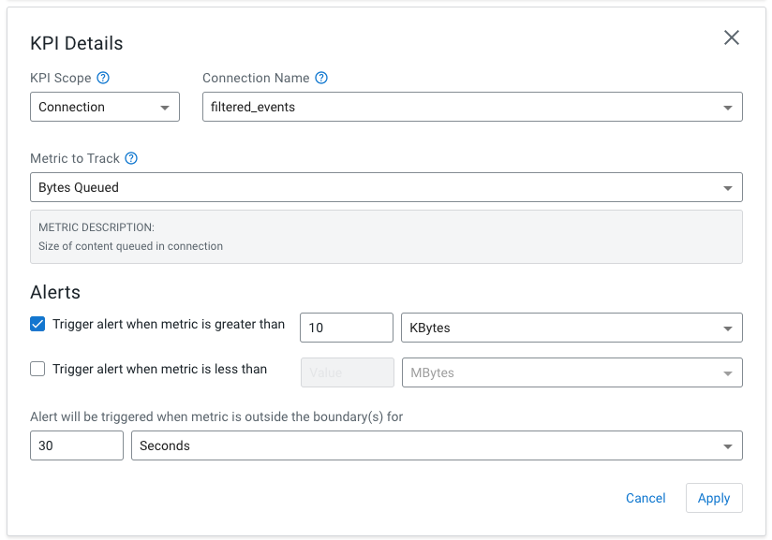
| Trigger alert when metric is greater than 1 MB  Alert will be triggered when metrics is outside the boundary(s) for 30 seconds |
| --- |



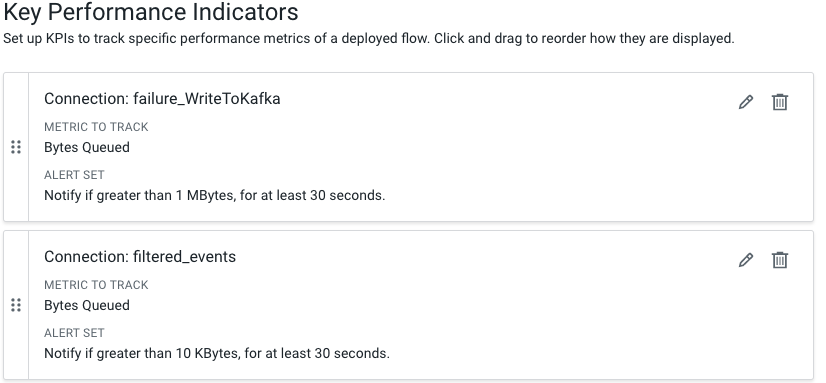
1. Add the following KPI

* **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.

# Lab 4 : SQL Stream Builder

## 1. Overview

The purpose of this workshop is to demonstrate streaming analytic capabilities using SQL Stream Builder. We will leverage the NiFi Flow deployed in CDF-PC from the previous workshop and demonstrate how to query live data and subsequently sink it to another location. The SQL query will leverage the existing syslog schema in Schema Registry.

## 2. Running the lab

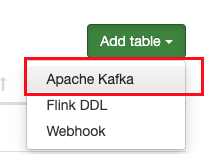
1. Open the SQL Stream Builder Interface



1. Click on Tables

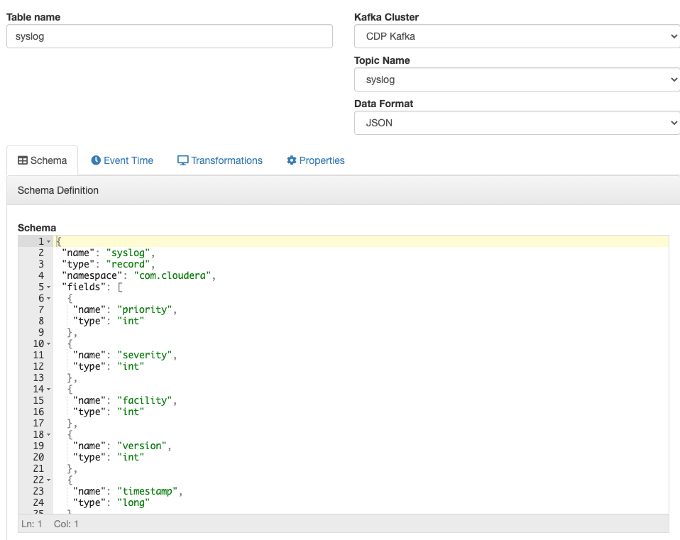


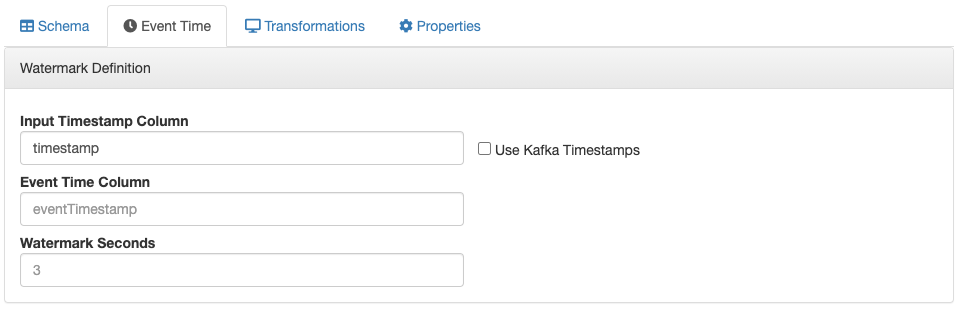
1. Add a new Kafka Table



1. Provide the following information

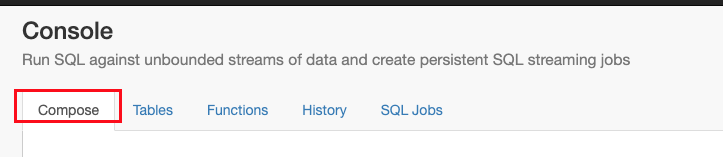
* **Table Name:<username>**\_syslog
* **Kafka Cluster:** CDP Kafka
* **Topic Name:** wuser01-syslog
* **Data Format:** JSON
* **Schema:** *Copy the syslog schema from Schema Registry*
* **Event Time Tab** - Deselect Use Kafka Timestamps
* **Event Time Tab - Input Timestamp Column:** timestamp

**

**

**Note:** At this point you can also discuss the detect schema functionality.

1. Click Save Changes
2. Click on the Compose Tab



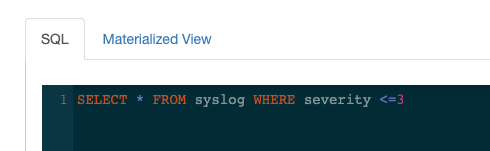
1. Add the following SQL Statement

| SELECT \* FROM **<username>**\_syslog WHERE severity <=3 |
| --- |

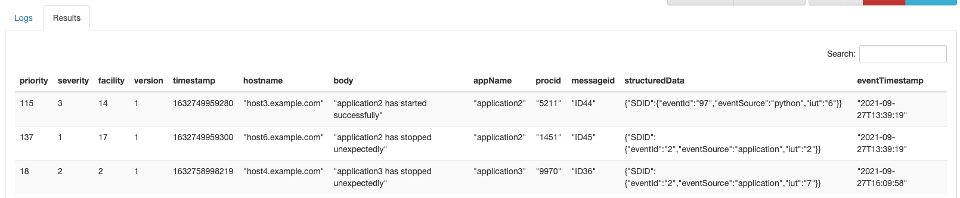
1. Provide a SQL Job Name and leave Sink Table disabled



1. Run the Streaming SQL Job by clicking Execute. Also, ensure your syslog-to-kafka flow is running in CDF-PC.



1. You should see syslog messages with severity levels <=3



1. Other information: Sink’s

The results of the continuous SQL can be sent to a sink of some type for further processing or stage.

1. Other Information: Materialized Views

SSB has the capability to materialize results from a Streaming SQL query to a persistent view of the data that can be read through REST and over the PG wire protocol. Applications can use this mechanism to query streams of data in a way of high performance without deploying additional database systems. Materialized Views are built into the SQL Stream Builder service, and require no configuration or maintenance. The Materialized Views act like a special kind of sink, and can even be used in place of a sink. They require no indexing, storage allocation, or specific management.

# Lab 5 : Leveraging Data Flow Catalog in CDF-PC

## 1. Overview

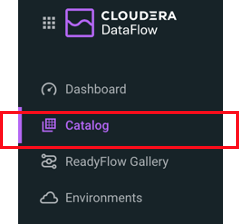
The purpose of this workshop is to demonstrate how to leverage existing flow definitions from Data Flow’s ReadyFlow Gallery, specifically the “Kafka filter to Kafka” flow which allows you to quickly consume events in JSON, CSV or Avro format from a kafka topic, filter them as necessary and eventually write them back to kafka.

**Note**: For your convenience, ReadyFlow Gallery flows are pre-defined in the Catalog

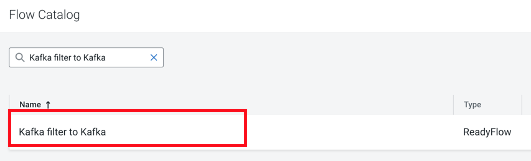
This ReadyFlow consumes JSON, CSV or Avro data from a source Kafka topic and parses the schema by looking up the schema name in the CDP Schema Registry. You can filter events by specifying a SQL query in the 'Filter Rule' parameter. The filtered events are then converted to the specified output data format and written to the destination Kafka topic. Failed Kafka write operations are retried automatically to handle transient issues. Define a KPI on the 'failure\_WriteToKafka' connection to monitor failed write operations.

## 2. Running the lab

1. Go to Cloudera CDP Home -> Data Flow -> Catalog



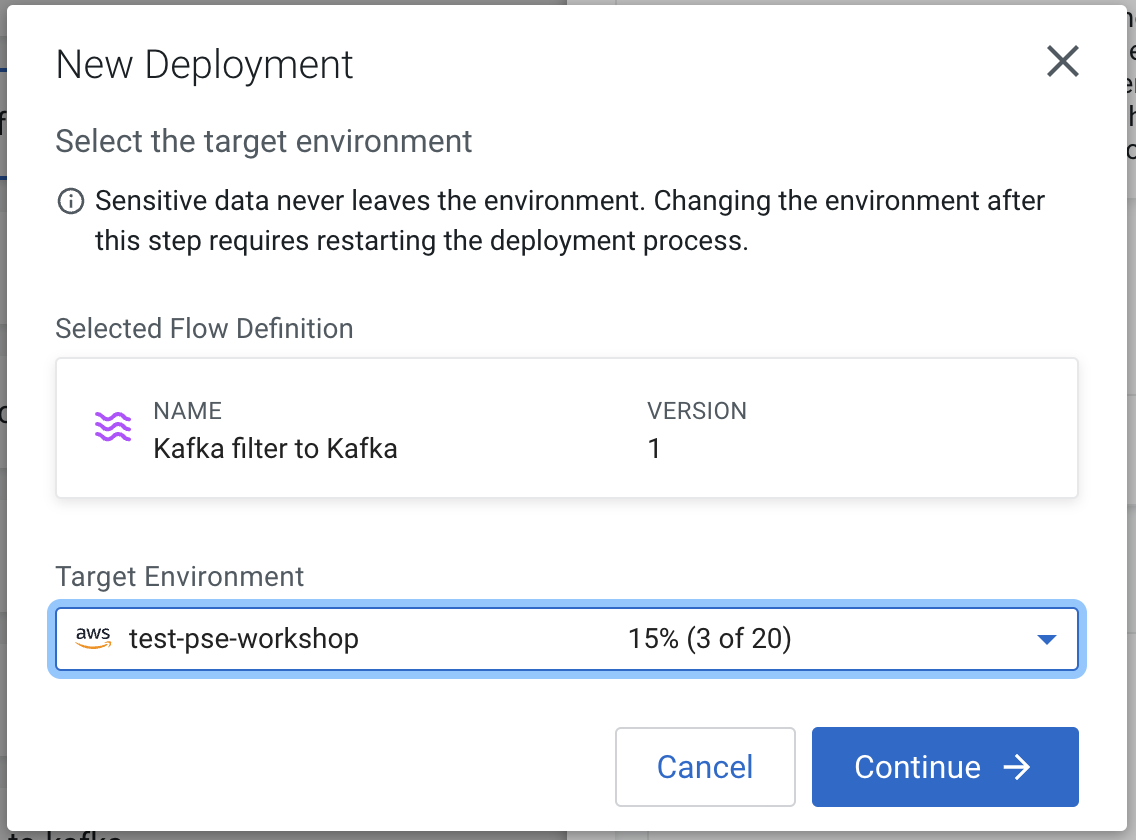
1. In the search box, type “Kafka filter to Kafka” and click on the ReadyFlow



1. Click on the latest version, then Click Deploy



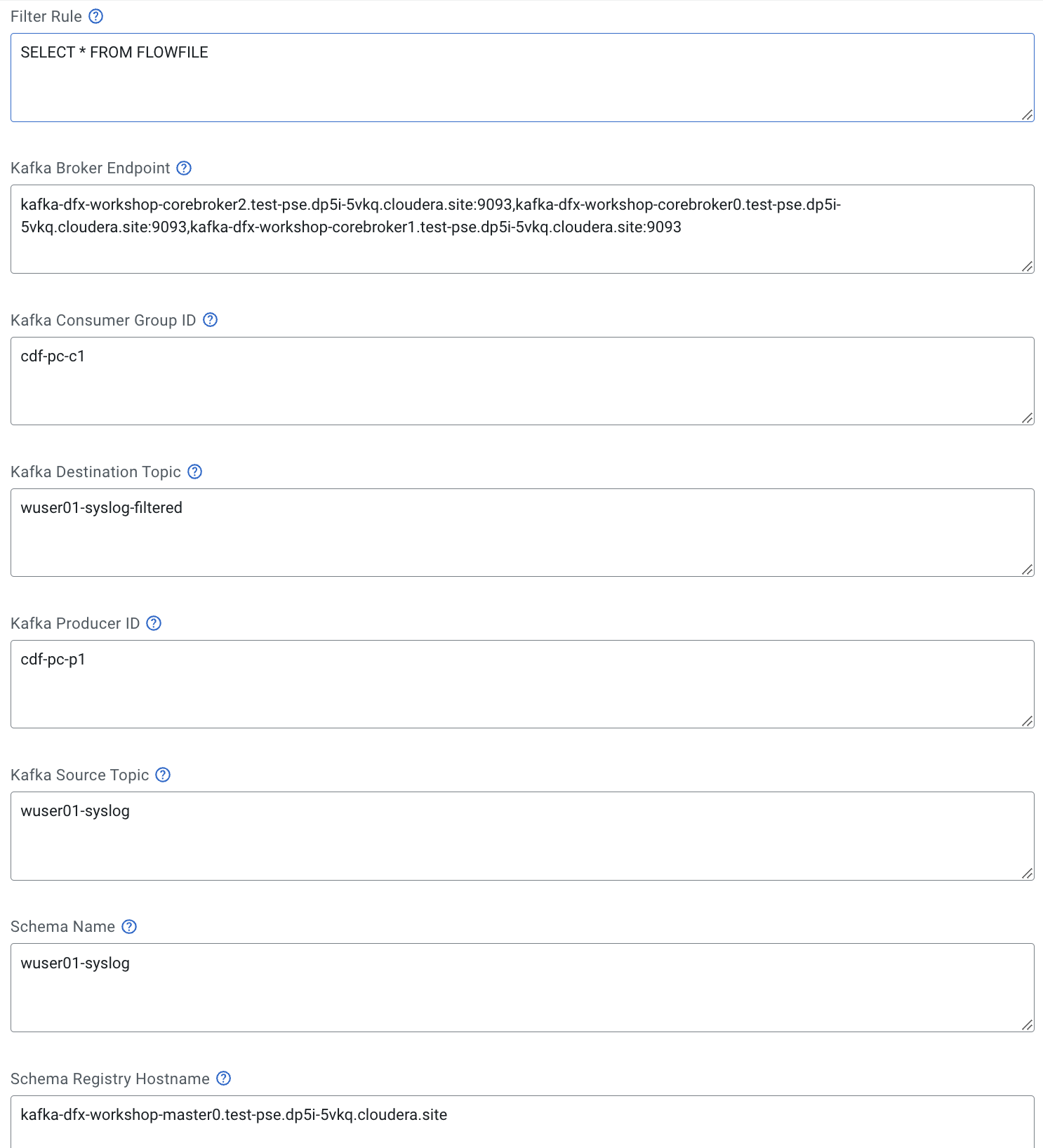
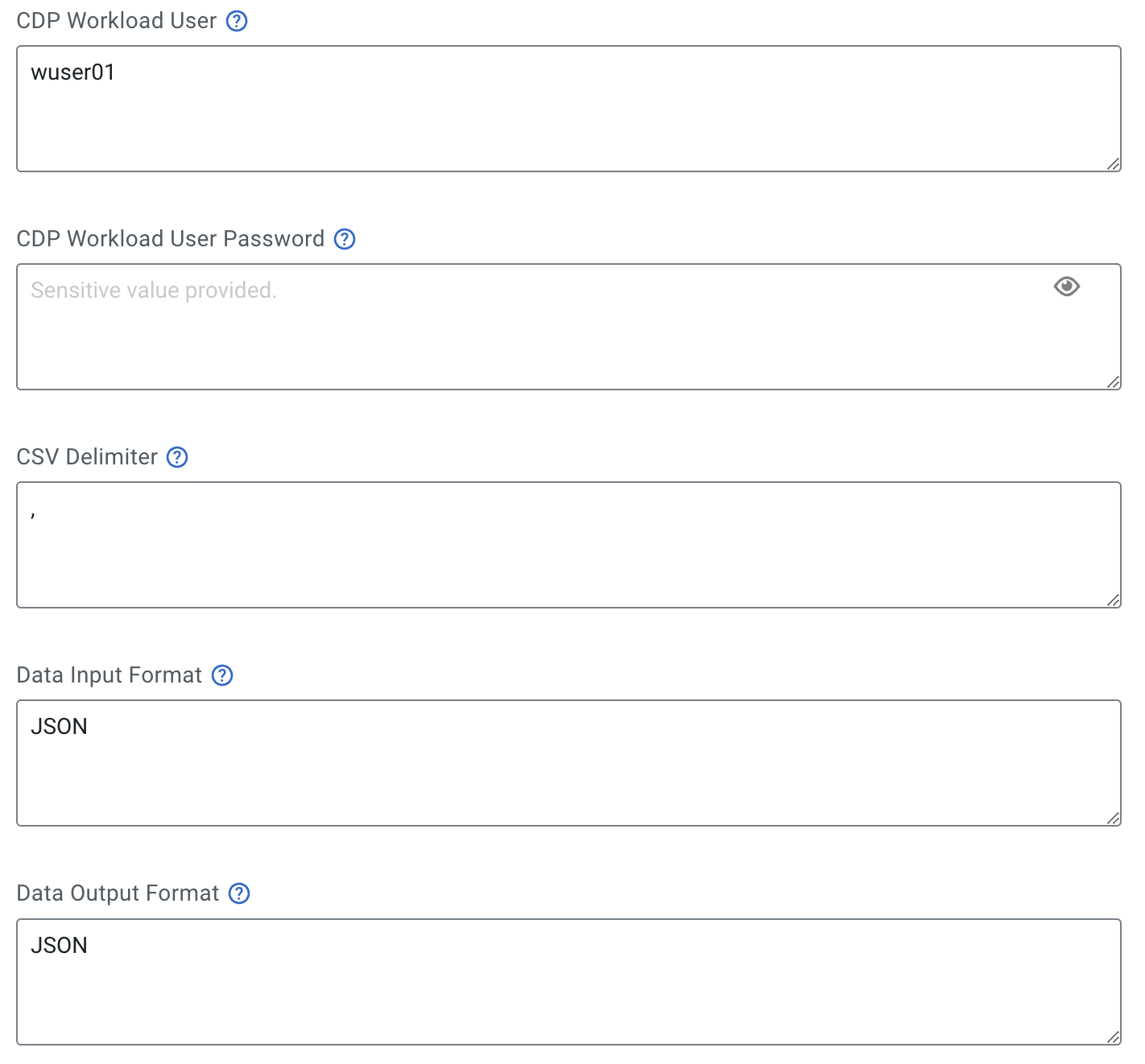
1. Select the Target Environment you wish to deploy to and click Continue



1. You will be presented with the New Flow Deployment wizard. In the Overview page, give your deployment a unique valid name and click Next

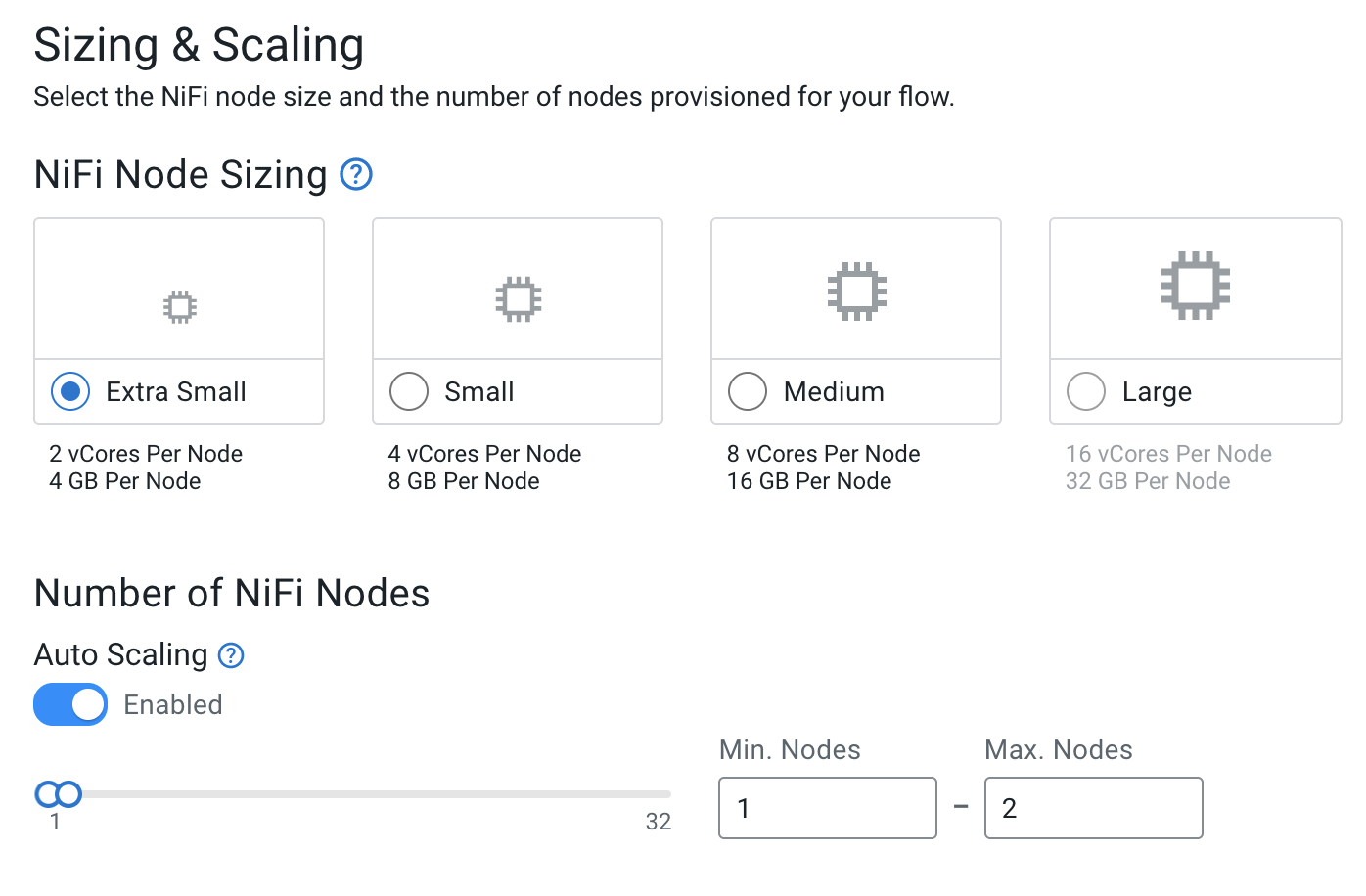


1. In the Parameters page, fill in the required parameters and click Next



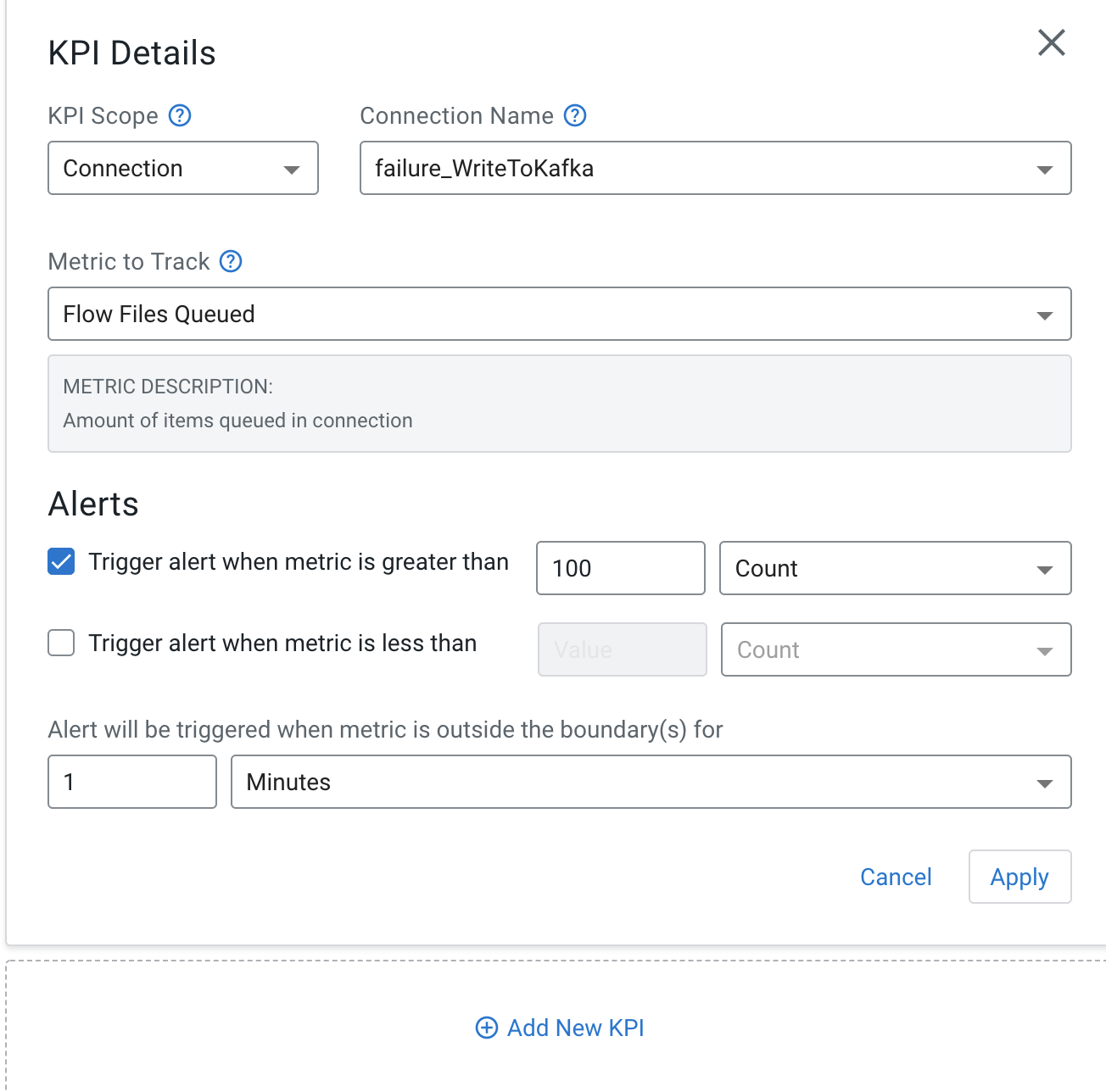
1. In the Sizing & Scaling page, set the following settings:

* Extra Small
* Auto Scaling: Enabled
* Min Nodes: 1
* Max Nodes: 2

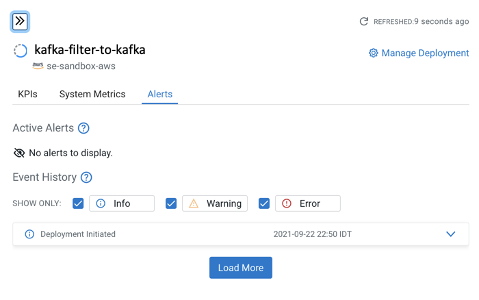


1. In the Key Performance Indicators page, click Add New KPI and set the following settings, then click Apply:

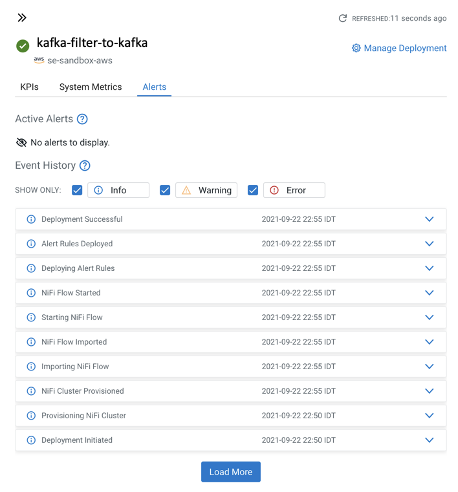
* KPI Scope: Connection
* Connection Name: failure\_WriteToKafka
* Metric To Track: Flow Files Queued
* Set the alerts according to the following image



1. In the Review page, verify that the configuration is correct and click Previous to go back and change a setting or Deploy to start the deployment process
2. The flow is now being deployed, this process may take a few minutes to complete, you can click on Load More to get the latest events for the deployment.



1. Once the deployment process is complete, you should see an output similar to the following.



1. To view your flow, click on Manage Deployment, Actions, then View in NiFi

