DataFlow Workshop

Module 1 -Building Flow with NiFi & Interacting with Kafka

1. Overview

In this module you will practice creating, testing and deploying a flow in CDF PC. You will learn to:

- Develop flows using the low-code Flow Designer.
- Create a test session and interactively test a flow.
- Version-control a flow into the DataFlow Catalog.
- Deploy a flow on an auto-scaling production cluster on k8s.

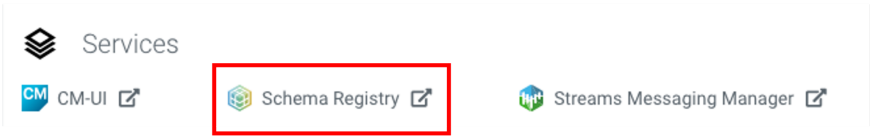
You'll also learn flow design best practices like:

- Create self-contained flow within a Process Group
- Parameterize flows for reusability
- Set meaningful names for flow components, including queues (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.

2. Create a schema in Schema Registry

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



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

   Note: The name of the schema ("syslog") must match the name of the Kafka topic you will create later.

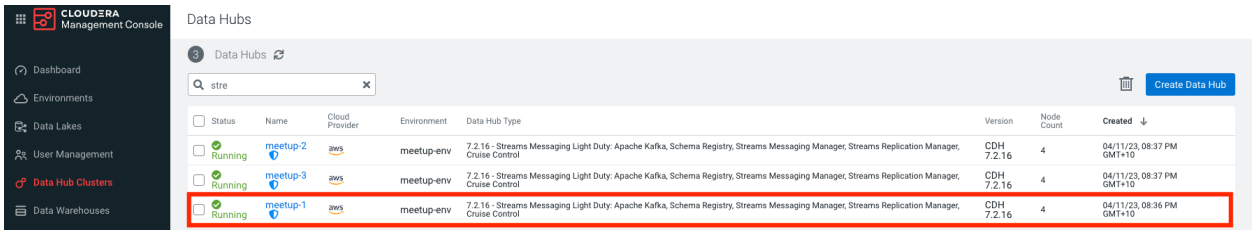
   - Name: **<userid>-syslog-avro**
   - Description: **syslog schema for dataflow workshop**
   - Type: **Avro schema provider**
   - Schema Group: **Kafka**
   - Compatibility: **Backward**
   - Evolve: **True**
   - Schema Text:

```
{
  "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" }
              ]
            }
          }
```

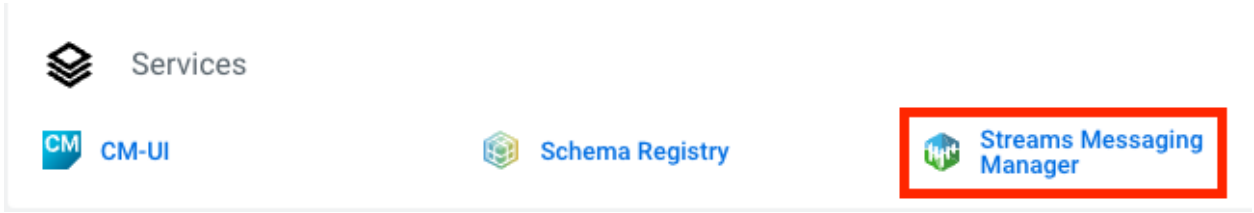
```
            ]
          }
        }
    ]
}
```

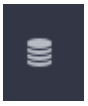
3. Create a Kafka topic manually

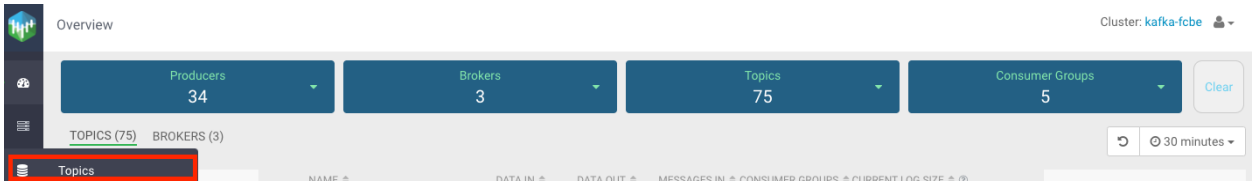
In Data Hub Clusters, search for streaming and click on the cluster allocated to you example meetup-1



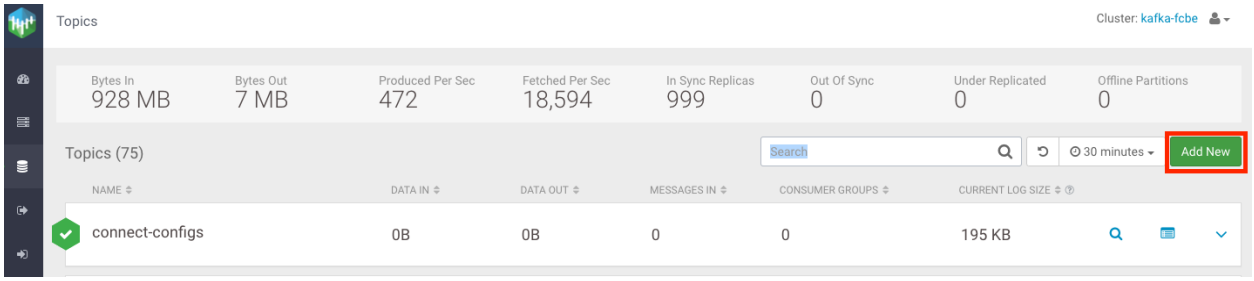
In the services section of the cluster click on “Streams Messaging Manager”



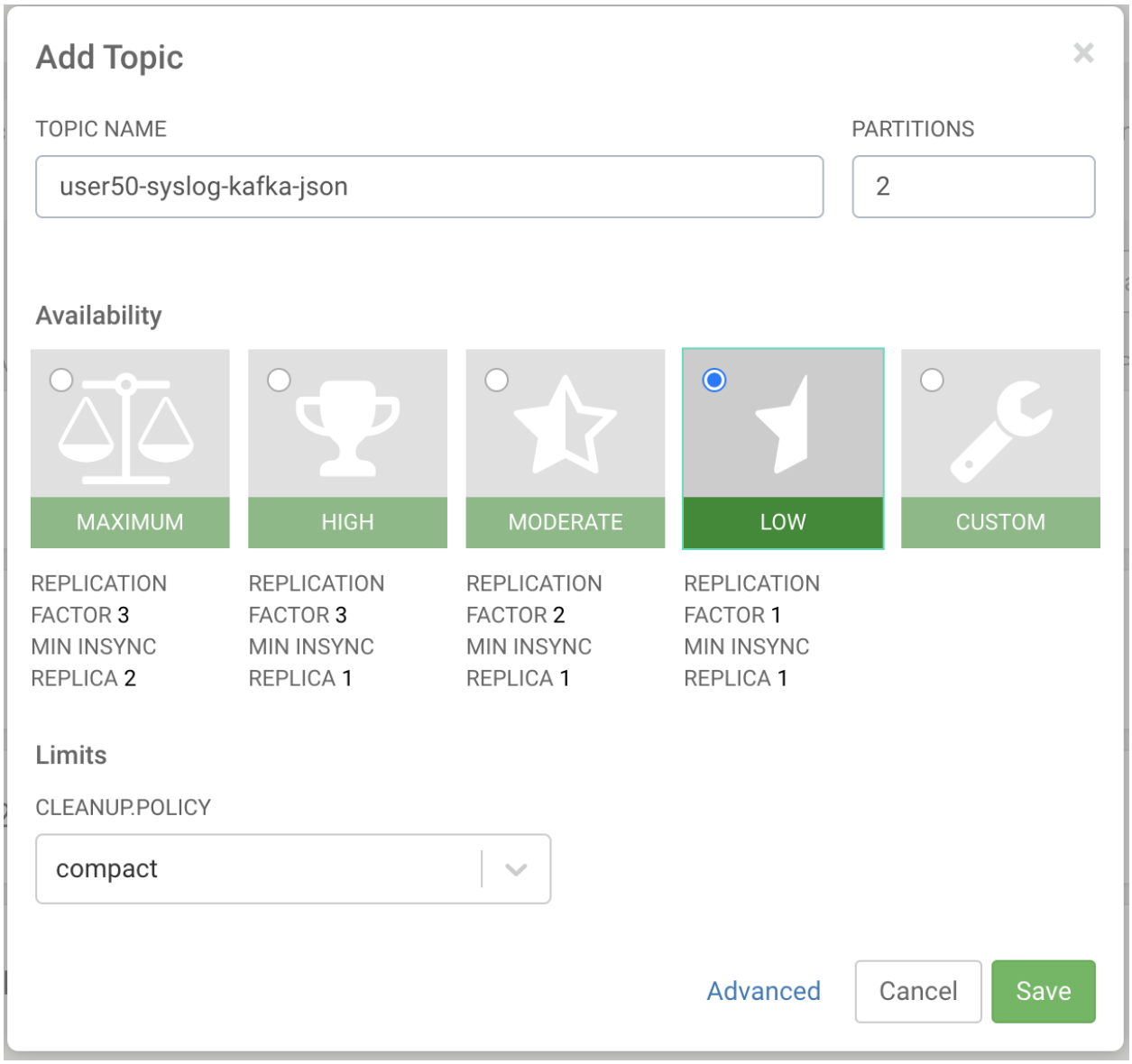
In the Streams Messaging Manager home page click on “Topics” which is on the left corner of your screen and represented by this icon. 



In the topics page, click on “Add New” button to create a new topic manually



Once you click on on “Add New” you will be taken to the the following screen



Create a new topic with the following information:

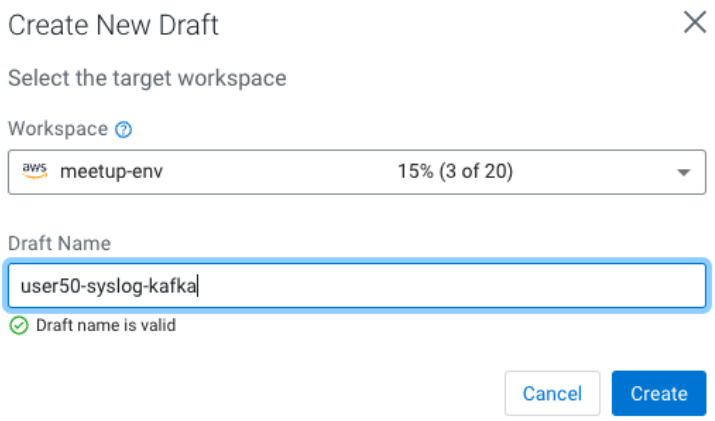
- Name: **<userid>-syslog-kafka-json**
- Availability: **Low**
- Cleanup Policy: **Compact**
- Partitions: **2**

Click Save.

4. Design the flow in the Flow Designer

4.1 Create a new flow

1. In Cloudera DataFlow, open the Flow Designer and create a new draft flow called **<userid>-syslog-kafka-flow**



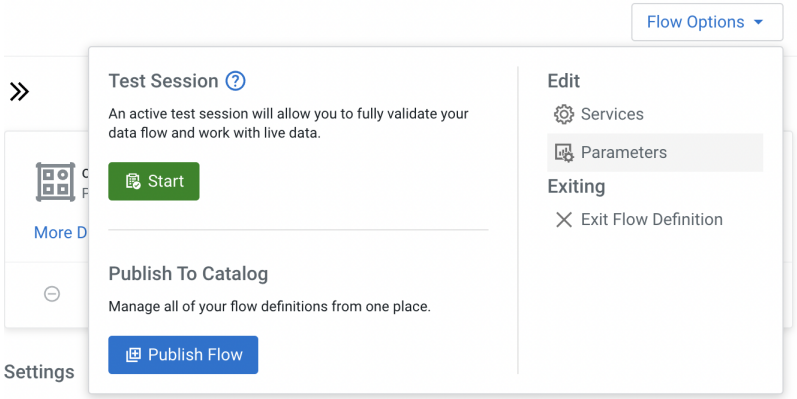
4.2 Configure parameters for the flow

Parameters are created within Parameter Contexts. In the context of CDP Flow Designer, one default Parameter Context is auto-created when you create a new draft flow.

You can then add parameters to this one Context, you cannot create additional ones.

This default context is automatically bound to each Process Group you create within your draft Flow, making all parameters available to be used in any process group.

1. To create a Parameter, click on Flow Options in the top right corner and select Parameters

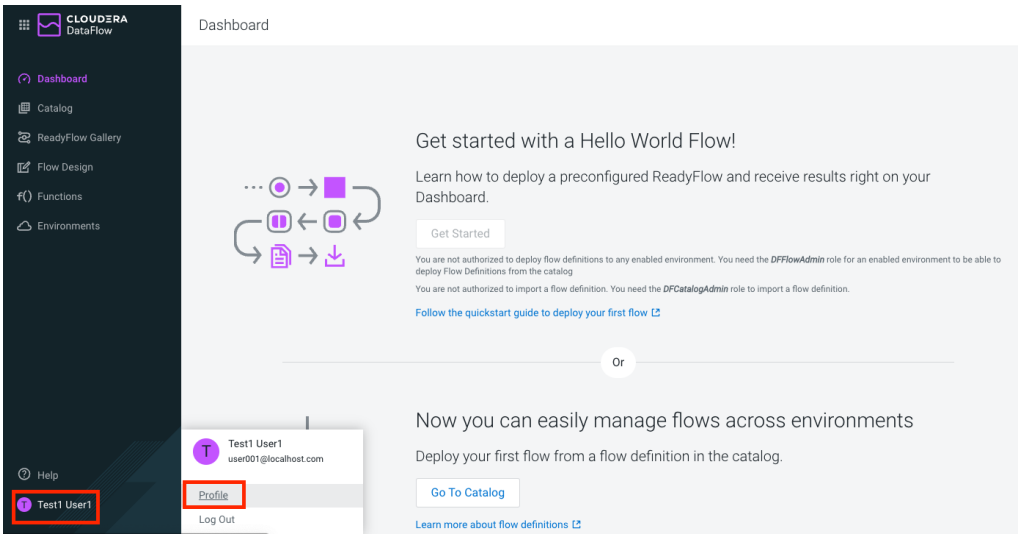


2. Add the parameters. Click on **Add Parameter > Add Parameter** for each parameter to be added and enter the appropriate details:

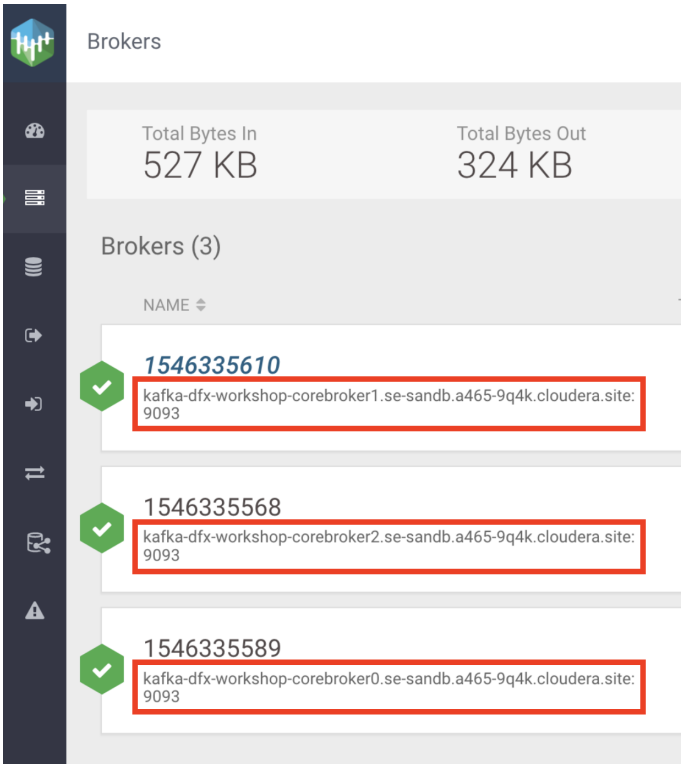
Name	Description	Value
CDP Workload User	CDP Workload User	<Your own workload user name>
Filter Rule	Filter Rule	SELECT * FROM FLOWFILE
Kafka Broker Endpoint	Comma-separated list of Kafka Broker addresses	<Comma-separated list of Kafka Broker addresses. See notes below>
Kafka Destination Avro Topic		<userid>-syslog-avro
Kafka Destination JSON Topic		<userid>-syslog-json
Kafka Producer ID		<userid>-producer
Schema Name		<userid>-syslog-avro
Schema Registry Hostname		<Hostname of Schema Registry service. See notes below>

Notes:

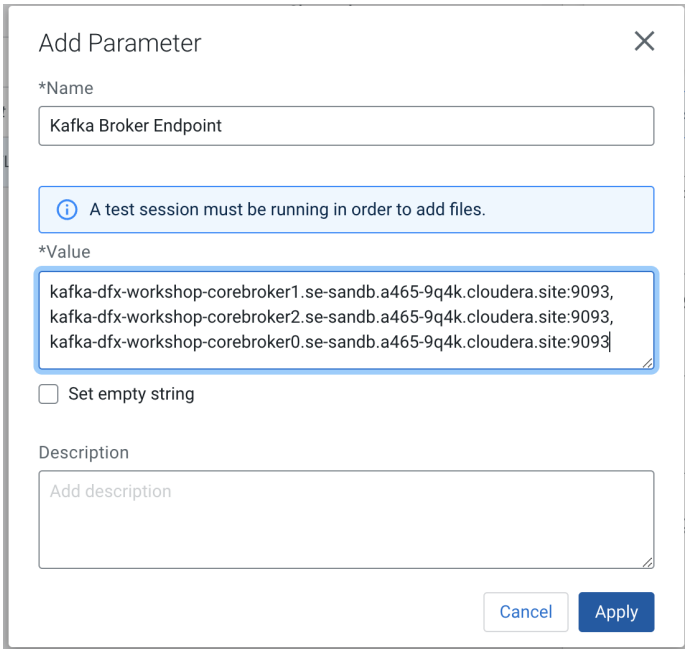
- **CDP Workload User:** The workload user can be found at the bottom left of your screen for instance “Test1 User1” you will be allocated a user and the information will be shared with you.



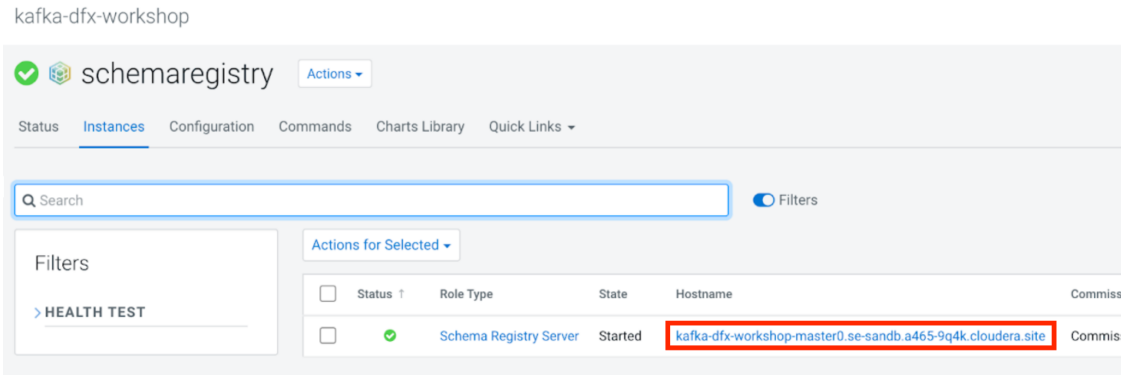
- **Kafka Broker Endpoint:** The Kafka brokers' addresses can be found in the Brokers page of the SMM UI. To get there, find your Streams Messaging DataHub and click on the Streams Messaging Manager (SMM) link (Streams Messaging Manager). On the SMM UI, click on the Brokers icon ( ). The address of each broker will be shown in this page and consists of the broker host name and port number, as shown below.



The value for the Kafka Broker Endpoint parameter must be a comma-separated list of the broker addresses, as shown in the example below:



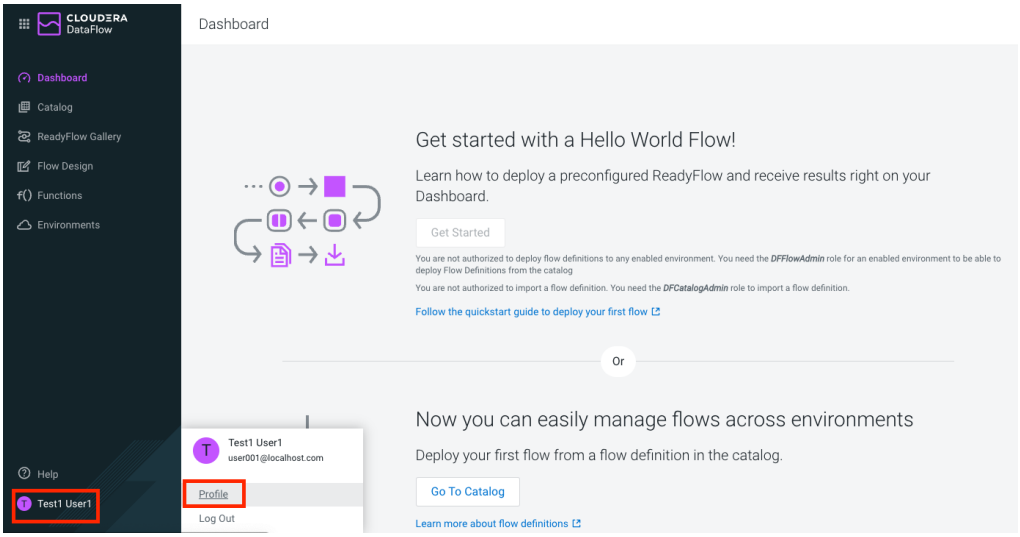
- **Schema Registry Hostname**: to identify the name of the Schema Registry host using Cloudera Manager: on the Streams Messaging DataHub page, click on **CM-UI > Clusters > schemaregistry > Instances** and copy the host name from there:



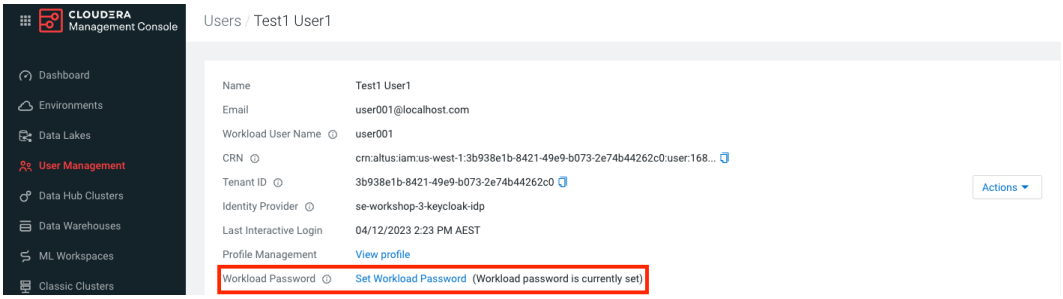
3. Add the *sensitive* parameters. Click on **Add Parameter > Add Sensitive Parameter** for each parameter to be added and enter the appropriate details:

Name	Description	Value
CDP Workload User Password	CDP Workload User Password	<Your own workload password for the environment. See notes below>

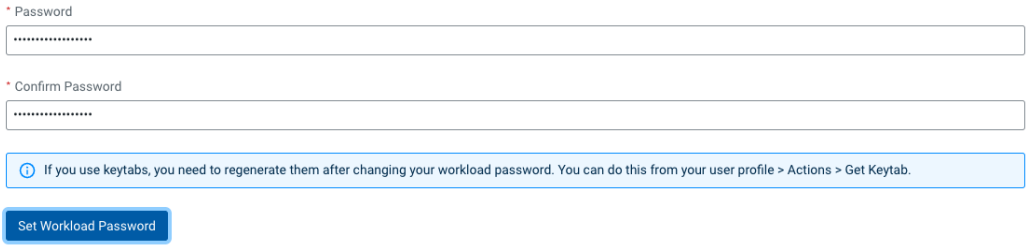
- **CDP Workload User Password:** The workload user password can be set using the following instructions. At the bottom left of your screen click on the **Username-> Profile**



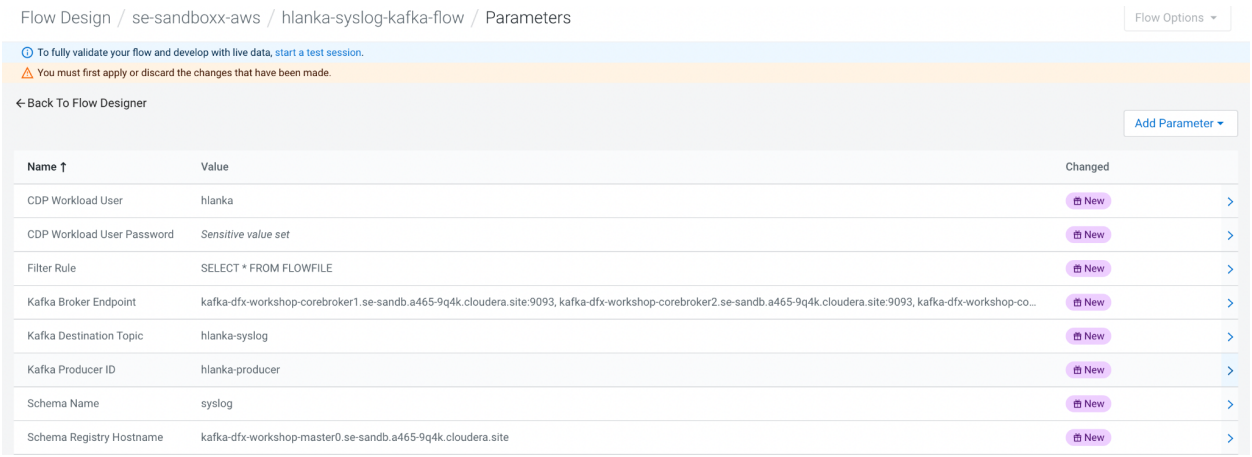
- You will be taken to the following screen where you can set the workload password



- Click on set workload password, provide a password and then click on **"Set Workload Password"**



4. Once all the parameters have been created verify with the list below



5. Click **Apply Changes** and then **Back To Flow Designer** to go back to the Flow Designer main page.

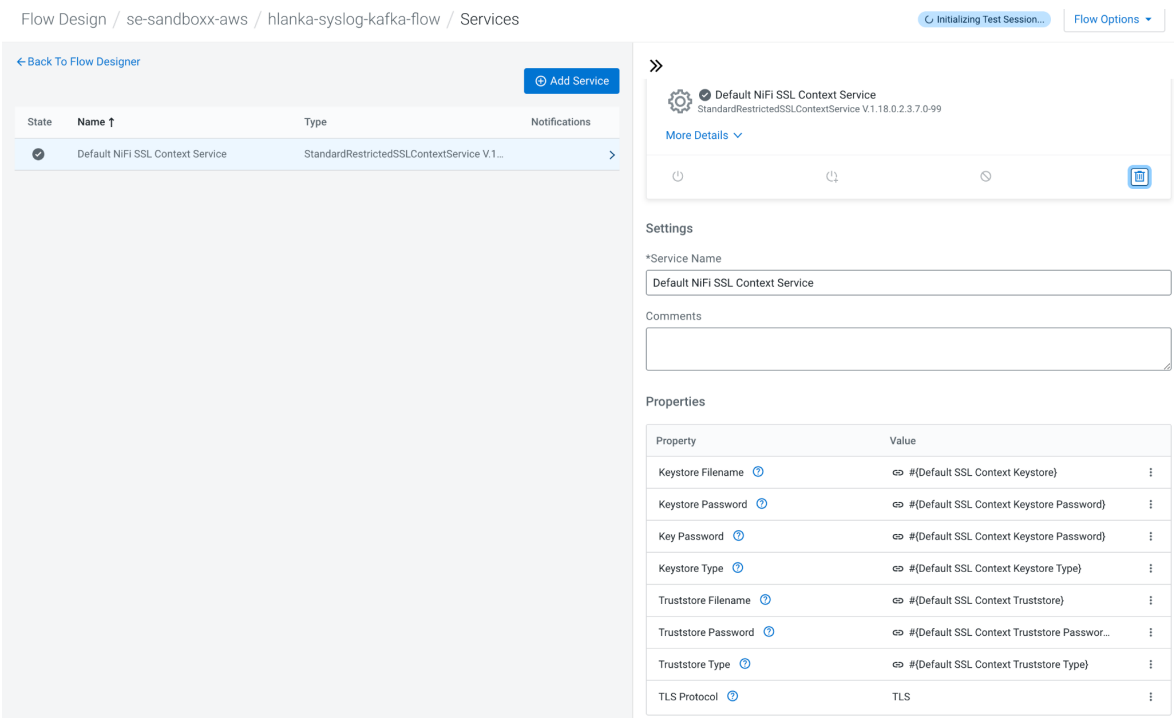
4.3 Create Controller Services

Controller Services are extension points that provide information for use by other components (such as processors or other controller services). The idea is that, rather than configure this information in every processor that might need it, the controller service provides it for any processor to use as needed.

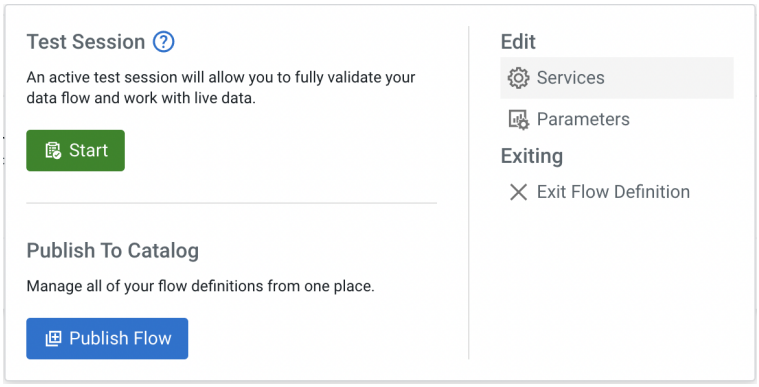
1. Enable the Test Session before configuring the services.

    a. Select **Flow Options > Test Session**. Use the latest NiFi version (default).
    b. Click **Start Test Session**. Test session status changes to Initializing Test Session…
    c. Wait for the status to change to **Active Test Session**.

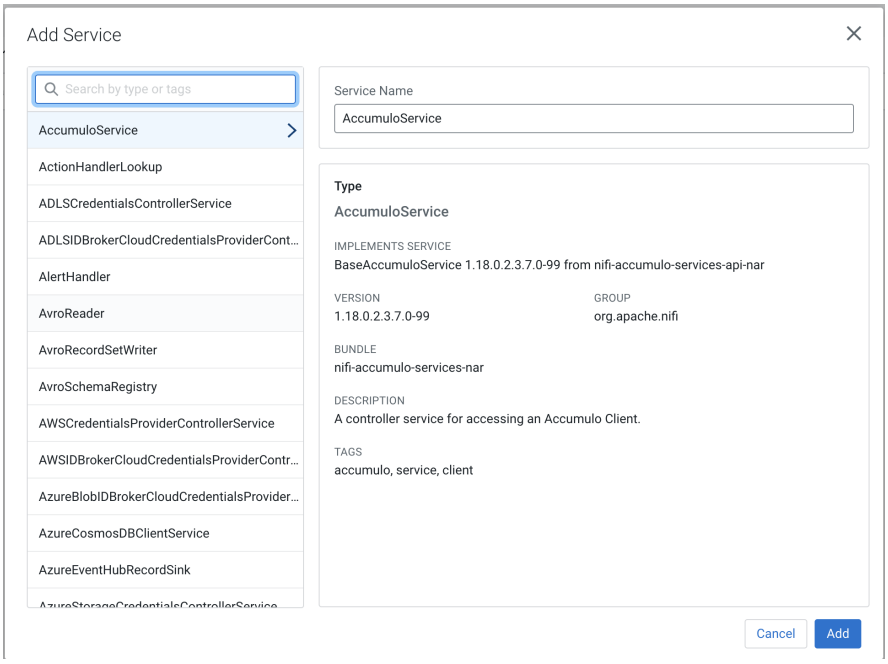
    You are not yet at the point of testing the flow, but enabling a test session forces the **Default Nifi SSL Context Service** controller service to be created, which you will need in the next steps.



2. Click **Flow Options > Services.**

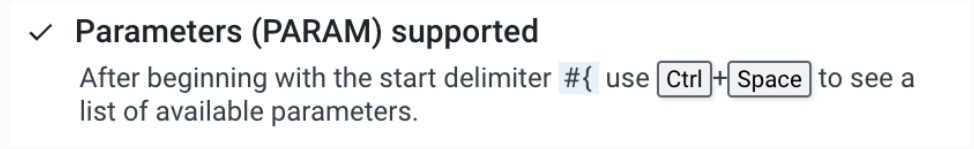


3. Add a new **HortonworksSchemaRegistry** controller service
    a. Click **Add Service**. The **Add Service** page opens.



    b. In the text box, filter for **HortonworksSchemaRegistry**, select it and click **Add**
    c. Configure the HortonworksSchemaRegistry service:

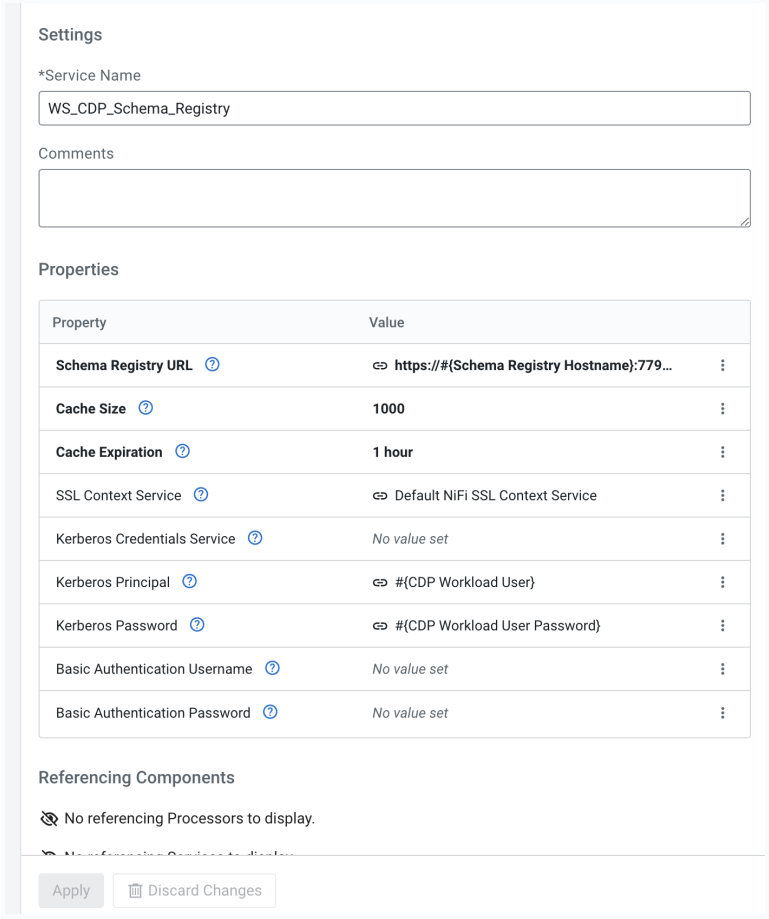
    Tip: When type parameters into a property value you can use the following technique to see the list of available parameters:



i. Service Name: **WS_CDP_Schema_Registry**

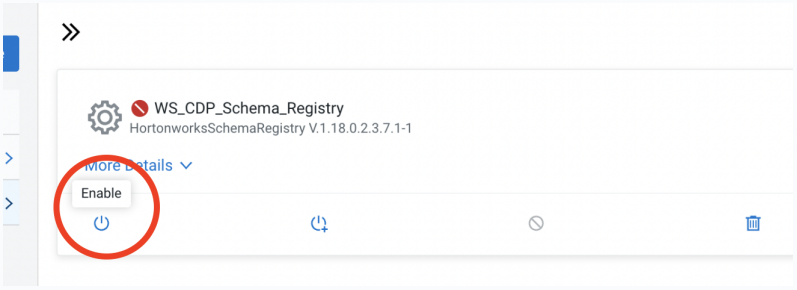
ii. Properties:

1. Schema Registry URL: **https://#{Schema Registry Hostname}:7790/api/v1**
2. SSL Context Service: **Default Nifi SSL Context Service**
3. Kerberos Principal: **#{CDP Workload User}**
4. Kerberos Password: **#{CDP Workload User Password}**



d. Click **Apply**

e. Activate the service by clicking on the **Enable** Icon



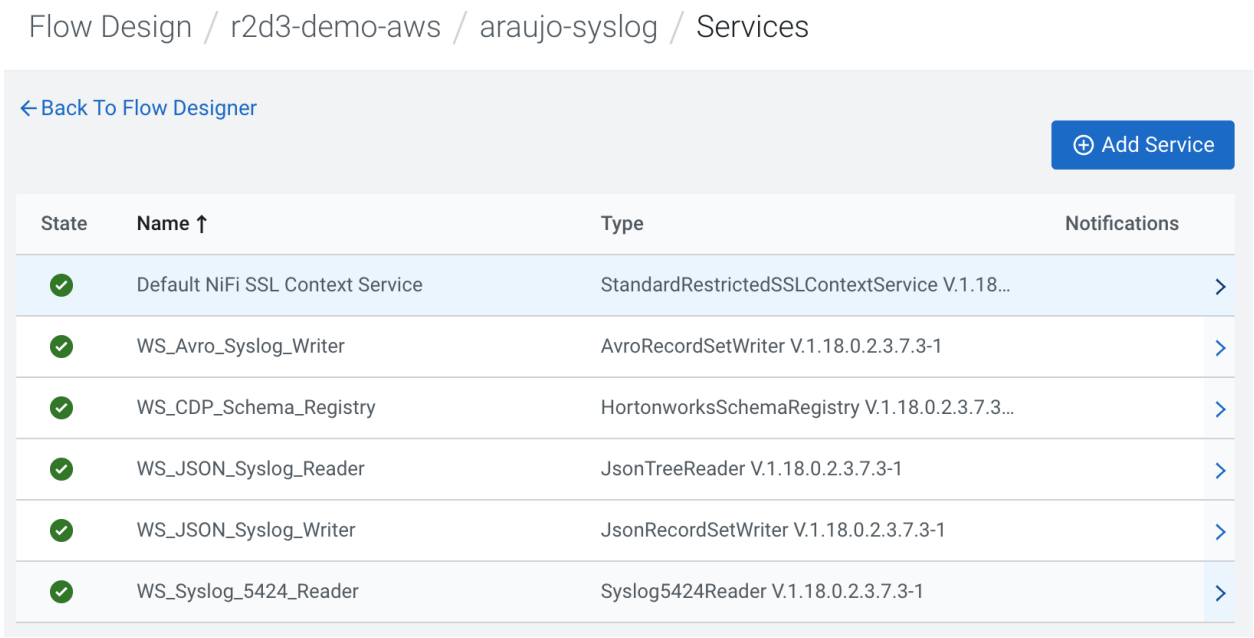
4. Add a new **Syslog5424Reader** controller service
    f. Click **Add Service**. The **Add Service** page opens.
    g. In the text box, filter for **Syslog5424Reader**, select it and click **Add**
    h. Configure the Syslog5424Reader service:
        i. Service Name: **WS_Syslog_5424_Reader**
    i. Click **Apply**
    j. Activate the service by clicking on the **Enable** Icon

5. Add a new **JsonTreeReader** controller service
    k. Click **Add Service**. The **Add Service** page opens.
    l. In the text box, filter for **JsonTreeReader**, select it and click **Add**
    m. Configure the JsonTreeReader service:
        i. Service Name: **WS_JSON_Syslog_Reader**
        ii. Properties:
            1. Schema Access Strategy: **Use 'Schema Name' Property**
            2. Schema Registry: **WS_CDP_Schema_Registry**
            3. Schema Name: **#{Schema Name}**
    n. Click **Apply**
    o. Activate the service by clicking on the **Enable** Icon

6. Add a new **JsonRecordSetWriter** controller service
    p. Click **Add Service**. The **Add Service** page opens.
    q. In the text box, filter for **JsonRecordSetWriter**, select it and click **Add**
    r. Configure the JsonRecordSetWriter service:
        i. Service Name: **WS_JSON_Syslog_Writer**
        ii. Properties:
            1. Schema Access Strategy: **Use 'Schema Name' Property**
            2. Schema Registry: **WS_CDP_Schema_Registry**
            3. Schema Name: **#{Schema Name}**
    s. Click **Apply**
    t. Activate the service by clicking on the **Enable** Icon

7. Add a new **AvroRecordSetWriter** controller service
    a. Click **Add Service**. The **Add Service** page opens.
    b. In the text box, filter for **AvroRecordSetWriter**, select it and click **Add**
    c. Configure the JsonRecordSetWriter service:
        i. Service Name: **WS_Avro_Syslog_Writer**
        ii. Properties:
            1. Schema Write Strategy: **HWX Content-Encoded Schema Reference**
            2. Schema Access Strategy: **Use 'Schema Name' Property**
            3. Schema Registry: **WS_CDP_Schema_Registry**
            4. Schema Name: **#{Schema Name}**
    d. Click **Apply**
    e. Activate the service by clicking on the **Enable** Icon

8. This completes the configurations of all the controller services. Please verify with the list below:

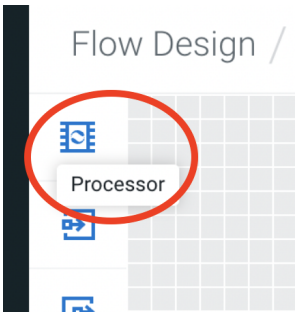


9. Click on the **Back to Flow Designer** link to go back to the flow canvas

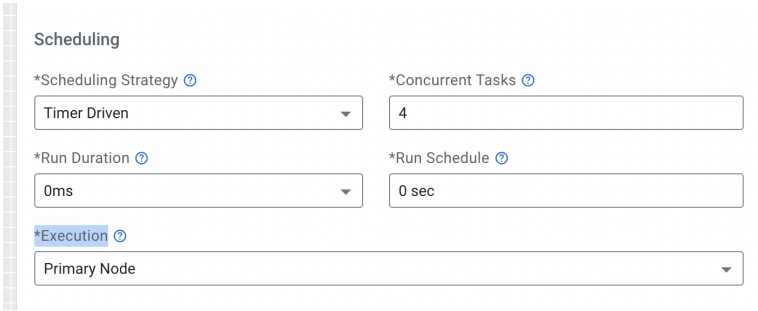
4.4 Design the flow

1. Open the Flow Design page
2. Create the **Generate Syslog RFC5424** processor

    a. Drag the Processor icon onto the canvas



    b. Select the **ExecuteScript** processor.
    c. Configure the processor as follows:
        i. Processor Name: **Generate Syslog RFC5424**
        ii. Concurrent Tasks: **4**
        iii. Run Schedule: **1 sec**
        iv. Execution: **Primary Node**

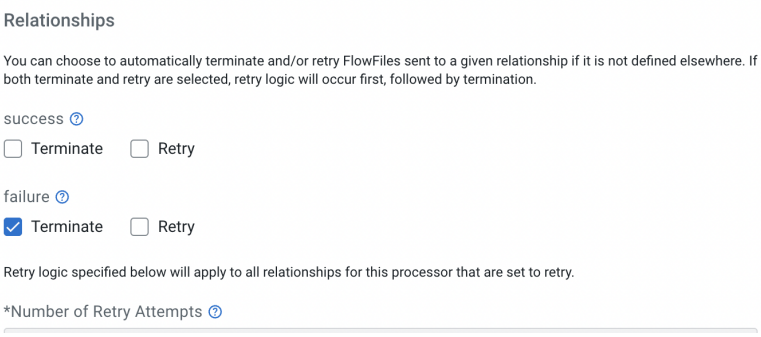


        v. Properties:
            1. Script Engine: **python**
            2. Script Body: <copy and paste the following script>

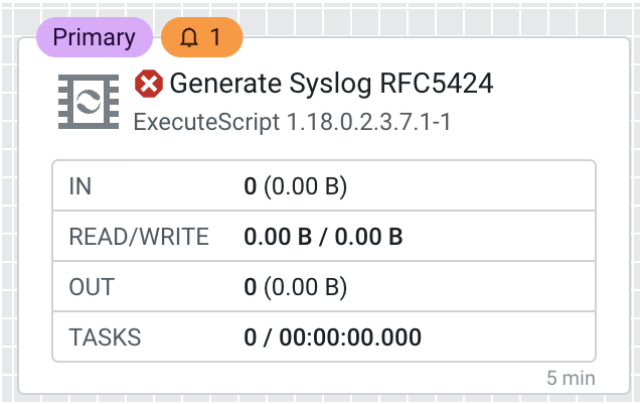
```
##
## Author: Nasheb Ismaily
## Description: Generates a random RFC5424 format syslog message
##

from org.apache.commons.io import IOUtils
from java.nio.charset import StandardCharsets
from org.apache.nifi.processor.io import OutputStreamCallback
import random
from datetime import datetime

class PyOutputStreamCallback(OutputStreamCallback):
```

```
    def __init__(self):
        pass

    def process(self, outputStream):
        hostname = "host"
        domain_name = ".example.com"
        tag = ["kernel", "python", "application"]
        version = "1"
        nouns = "application"
        verbs = ("started", "stopped", "exited", "completed")
        adv = ("successfully", "unexpectedly", "cleanly", "gracefully")

        for i in range(1,10):
            application = "{0}{1}".format(nouns, random.choice(range(1,
11)))
            message_id = "ID{0}".format(random.choice(range(1, 50)))
            random_tag = random.choice(tag)
            structured_data = "[SDID iut=\"{0}\" eventSource=\"{1}\"
eventId=\"{2}\"]".format(random.choice(range(1, 10)), random_tag,
random.choice(range(1, 100)))
            time_output =
datetime.utcnow().strftime('%Y-%m-%dT%H:%M:%S.%f')[:-3] + 'Z'
            random_host = random.choice(range(1, 11))
            fqdn = "{0}{1}{2}".format(hostname, random_host, domain_name)
            random_pid = random.choice(range(500, 9999))
            priority = random.choice(range(0, 191))
            num = random.randrange(0, 4)
            random_message = application + ' has ' + verbs[num] + ' ' +
adv[num]

            syslog_output = ("<{0}>{1} {2} {3} {4} {5} {6} {7}
{8}\n".format(priority, version, time_output, fqdn,
application,random_pid, message_id,structured_data, random_message))
            outputStream.write(bytearray(syslog_output.encode('utf-8')))

flowFile = session.create()
if (flowFile != None):
    flowFile = session.write(flowFile, PyOutputStreamCallback())
    session.transfer(flowFile, REL_SUCCESS)
```

vi. Relationships: check **Terminate** for the **failure** relationship



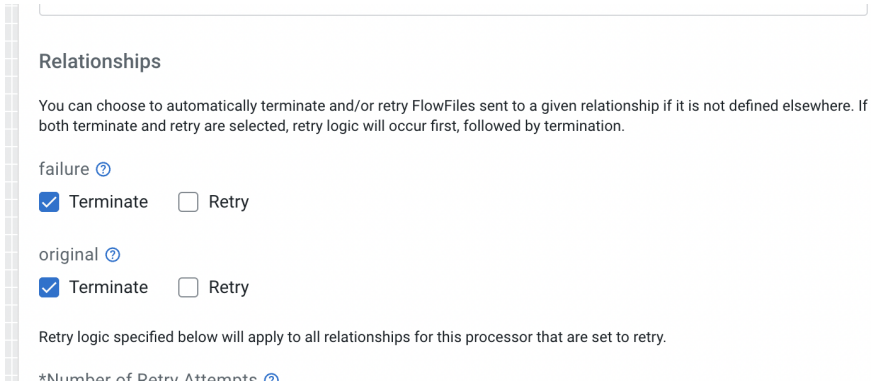
d. Click **Apply**



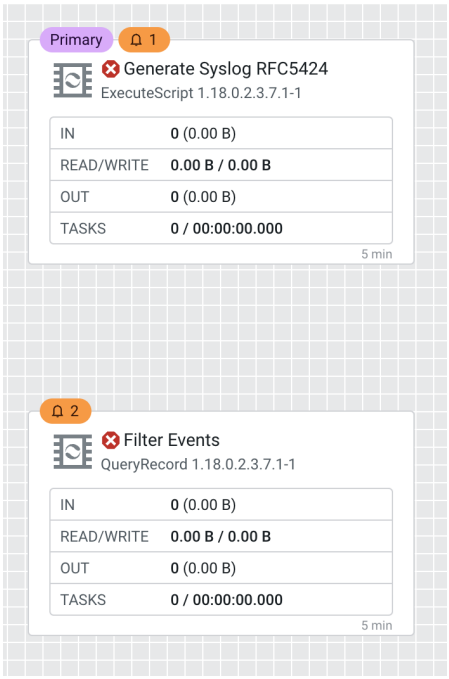
3. Create the **Filter Events** processor

   a. Drag the Processor icon onto the canvas
   b. Select the **QueryRecord** processor.
   c. Configure the processor as follows:
      i. Processor Name: **Filter Events**
      ii. Properties:
         1. Record Reader: **WS_Syslog_5424_Reader**
         2. Record Writer: **WS_JSON_Syslog_Writer**
      iii. Click on the **Add Property** button to add a new property:
         1. Property Name: **filtered_event**
         2. Property Value: **#{Filter Rule}**

iv. Relationships: check **Terminate** for the **failure** and **original** relationships



d. Click **Apply**



4. Create the **Write To Kafka - Avro** processor

   a. Drag the Processor icon onto the canvas
   b. Select the **PublishKafka2RecordCDP** processor.
   c. Configure the processor as follows:
      i. Processor Name: **Write To Kafka - Avro**
      ii. Properties:
         1. Kafka Brokers: **#{Kafka Broker Endpoint}**
         2. Topic Name: **#{Kafka Destination Avro Topic}**
         3. Record Reader: **WS_JSON_Syslog_Reader**
         4. Record Writer: **WS_Avro_Syslog_Writer**

5. Use Transactions: **false**
6. Security Protocol: **SASL_SSL**
7. SASL Mechanism: **PLAIN**
8. Username: **#{CDP Workload User}**
9. Password: **#{CDP Workload User Password}**
10. SSL Context Service: **Default NiFi SSL Context Service**
11. To set a specific id for the producer, click on the **Add Property** button to add a property, call it **client.id** and set the value to **#{Kafka Producer ID}**

iii. Relationships: check **Terminate** for the **success** relationship

d. Click **Apply**

5. Create the **Write To Kafka - JSON** processor

a. Drag the Processor icon onto the canvas
b. Select the **PublishKafka2RecordCDP** processor.
c. Configure the processor as follows:

i. Processor Name: **Write To Kafka - JSON**
ii. Properties:
    1. Kafka Brokers: **#{Kafka Broker Endpoint}**
    2. Topic Name: **#{Kafka Destination JSON Topic}**
    3. Record Reader: **WS_JSON_Syslog_Reader**
    4. Record Writer: **WS_JSON_Syslog_Writer**
    5. Use Transactions: **false**
    6. Security Protocol: **SASL_SSL**
    7. SASL Mechanism: **PLAIN**
    8. Username: **#{CDP Workload User}**
    9. Password: **#{CDP Workload User Password}**
    10. SSL Context Service: **Default NiFi SSL Context Service**
    11. To set a specific id for the producer, click on the **Add Property** button to add a property, call it **client.id** and set the value to **#{Kafka Producer ID}**
iii. Relationships: check **Terminate** for the **success** relationship

d. Click **Apply**

6. Connect the processors as shown in the diagram below:

- Connect the Processors **Generate Syslog RFC5424** & **Filter Events** for Relationship **success**
- Connect Processors **Filter Events** & **Write To Kafka - Avro** for Relationship **filtered_events**
- Connect Processors **Filter Events** & **Write To Kafka - JSON** for Relationship **filtered_events**
- Connect **Write To Kafka** to itself to retry for **failures**


Primary
Generate Syslog RFC5424
ExecuteScript 1.18.0.2.3.7.1-1
IN 0 (0.00 B)
READ/WRITE 0.00 B / 0.00 B
OUT 0 (0.00 B)
TASKS 0 / 00:00:00.000
5 min
NAME success
QUEUED 0 (0.00 B)
Filter Events
QueryRecord 1.18.0.2.3.7.1-1
IN 0 (0.00 B)
READ/WRITE 0.00 B / 0.00 B
OUT 0 (0.00 B)
TASKS 0 / 00:00:00.000
5 min
NAME filtered_events
QUEUED 0 (0.00 B)
WriteToKafka
PublishKafka2RecordCDP 1.0.0.2.3.7.1-1
IN 0 (0.00 B)
READ/WRITE 0.00 B / 0.00 B
OUT 0 (0.00 B)
TASKS 0 / 00:00:00.000
5 min
NAME failure
QUEUED 0 (0.00 B)

4.5 Naming the queues

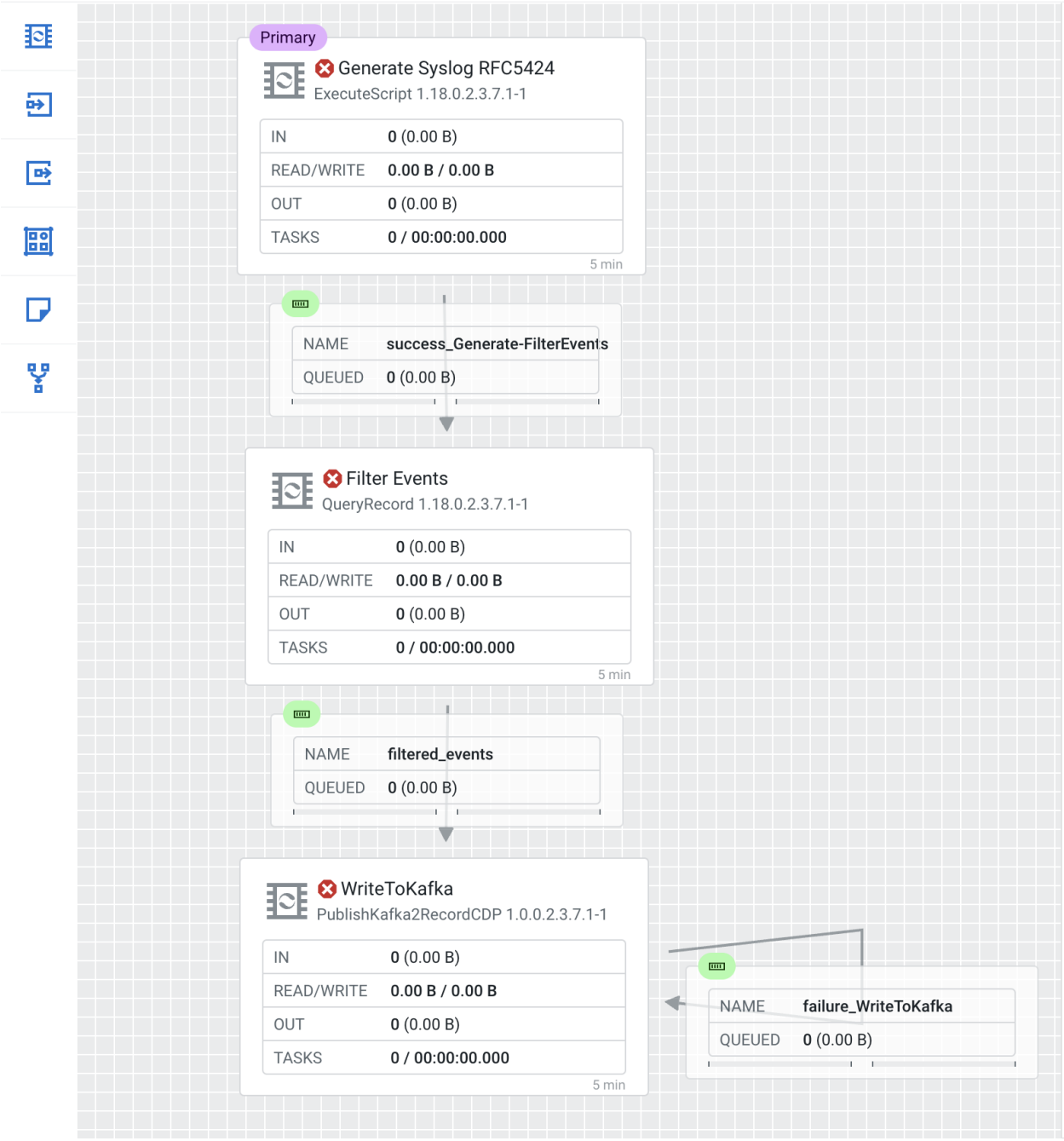
Providing unique names to all queues is important when creating flows in CDP-PC. The queue names are used to define Key Performance Indicators and having unique names for them make it easier to understand and create CDF-PC dashboards.

To name a queue, double-click the queue and give it a unique name. The best practice here is to start with the existing queue name (i.e. success, failure, retry, etc…) and, if that name is not unique, add the source and destination processor to the name.

For example:

- The **success** queue between **Generate Flow File** and **Filter Events** should be named
- The **success** queue between **Filter Events** and **Write To Kafka** should be named **filtered_event_Filter-WriteToKafka**
- The **failure** retry queue from **Write To Kafka** to itself should be named

Flow Design / se-sandboxx-aws / hlanka-syslog-kafka-flow / Canvas

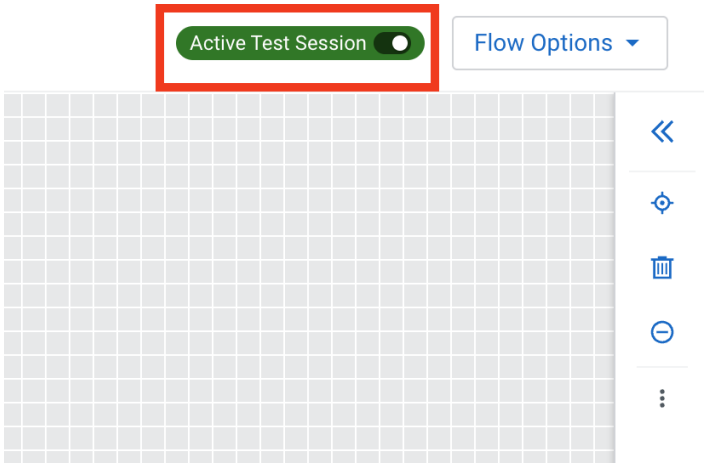


5. Interactively test the flow

Test sessions are a feature in the CDF Flow Designer that allow you to start/stop processors and work with live data to validate your data flow logic.

To test your draft flow, start a Test Session by clicking **Flow Options > Test Session > Start Test Session**. This launches a NiFi sandbox to enable you to validate your draft flow and interactively work with live data by starting and stopping components.

Tip: You can check the status of your Test Session in the upper right corner of your workspace. That is where you can also deactivate your Test Session.



1. If your test session is not yet started, start one by clicking on **Flow Options > Test Session > Start**.

   Note: If you had previously started and stopped a test session, the **Start** button is replaced with the **Restart** button. Clicking on **Restart** restarts the session with the same settings used previously. If you want to change the settings, click on the **Edit Settings** button.

2. Click **Start Test Session** (accept all defaults). Test session status changes to **Initializing Test Session…**

3. Wait for the status to change to **Active Test Session**.

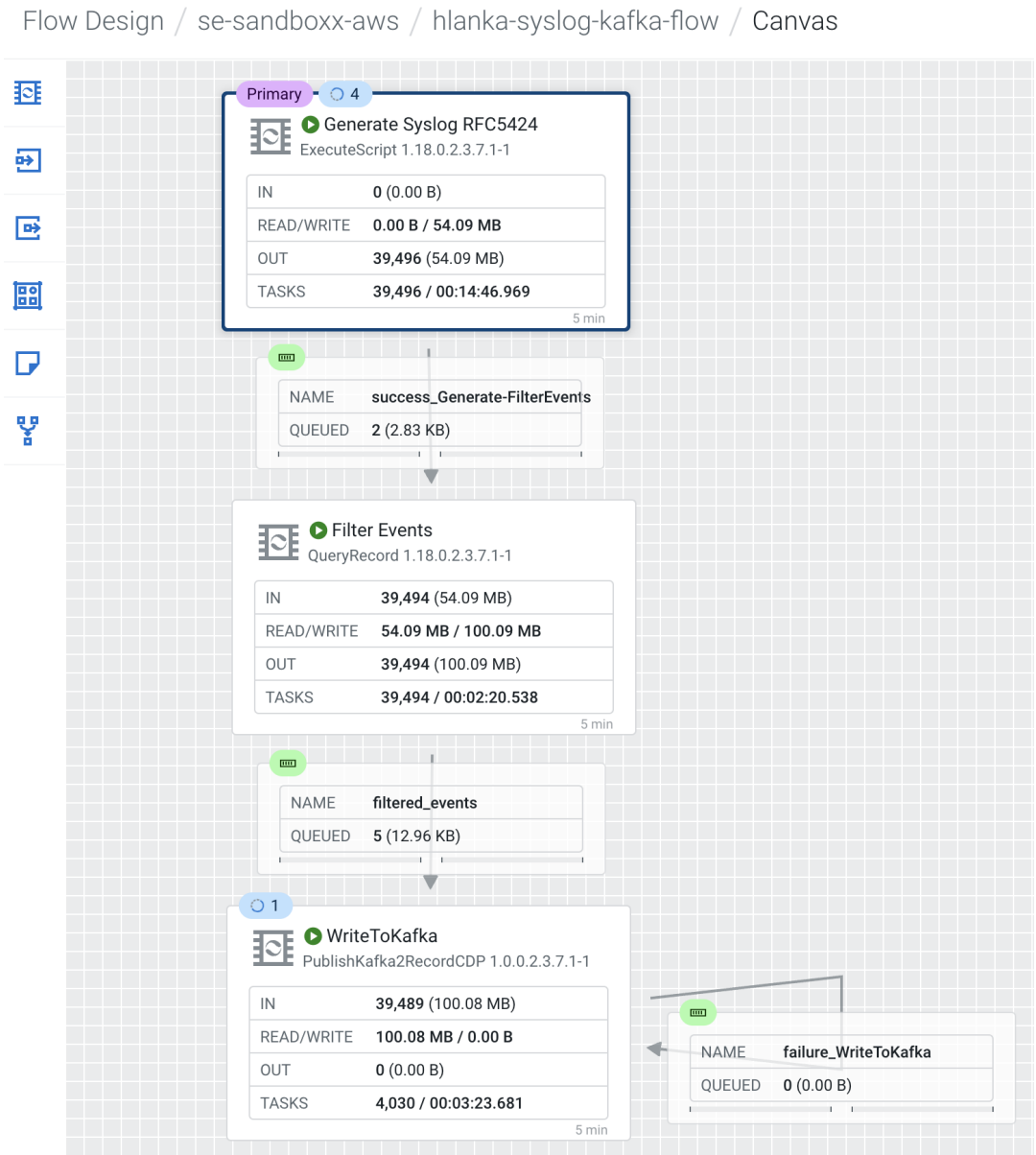
4. Click **Flow Options > Services** to enable Controller Services, if not already enabled.

5. Start all processors that are not running.

   Stopped processors are identified by the icon . Right-click on a stopped processor

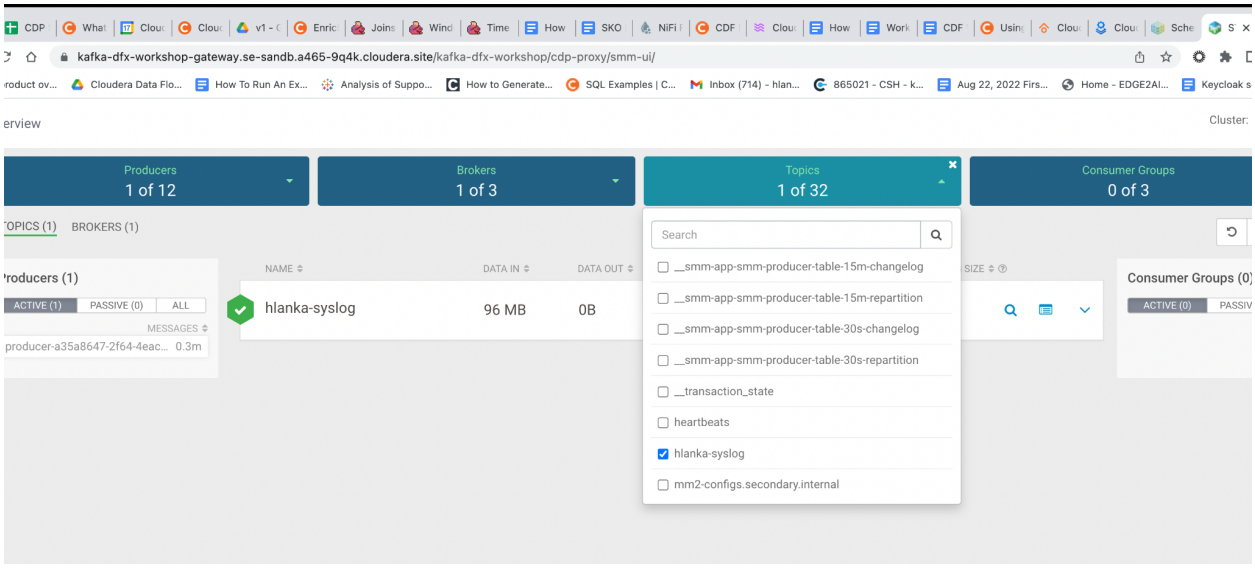
and select **Start** to start it.

The flow starts executing. On the Flow Design Canvas you can observe statistics on your processors change as they consume data and execute their respective tasks.

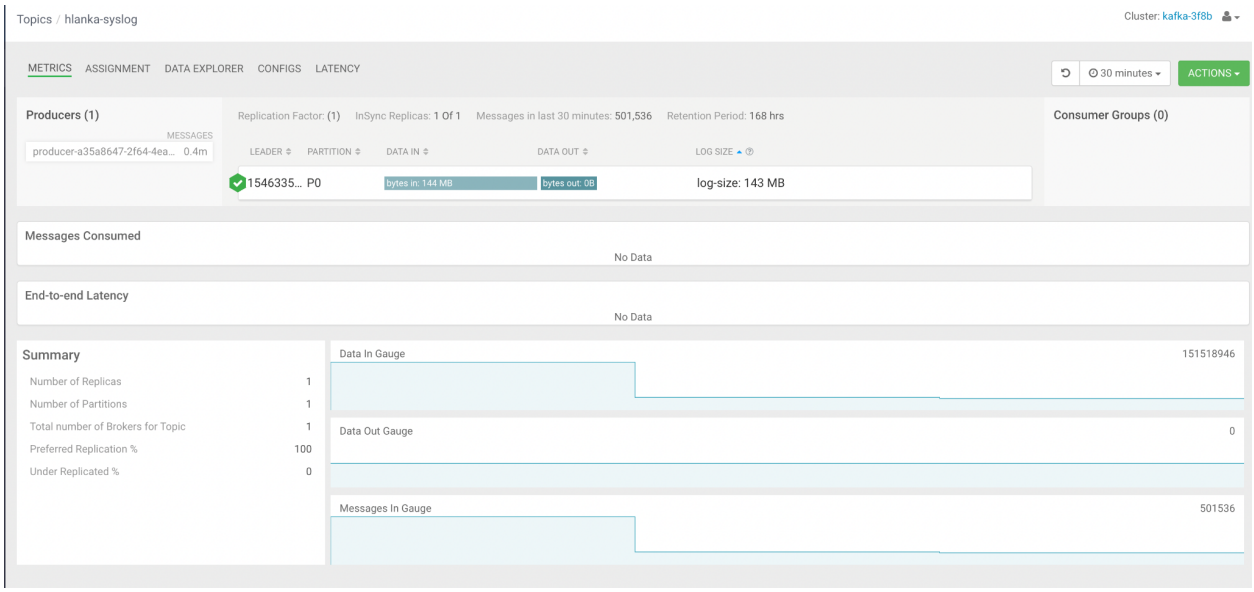


You should see the flow running successfully and writing records to Kafka.

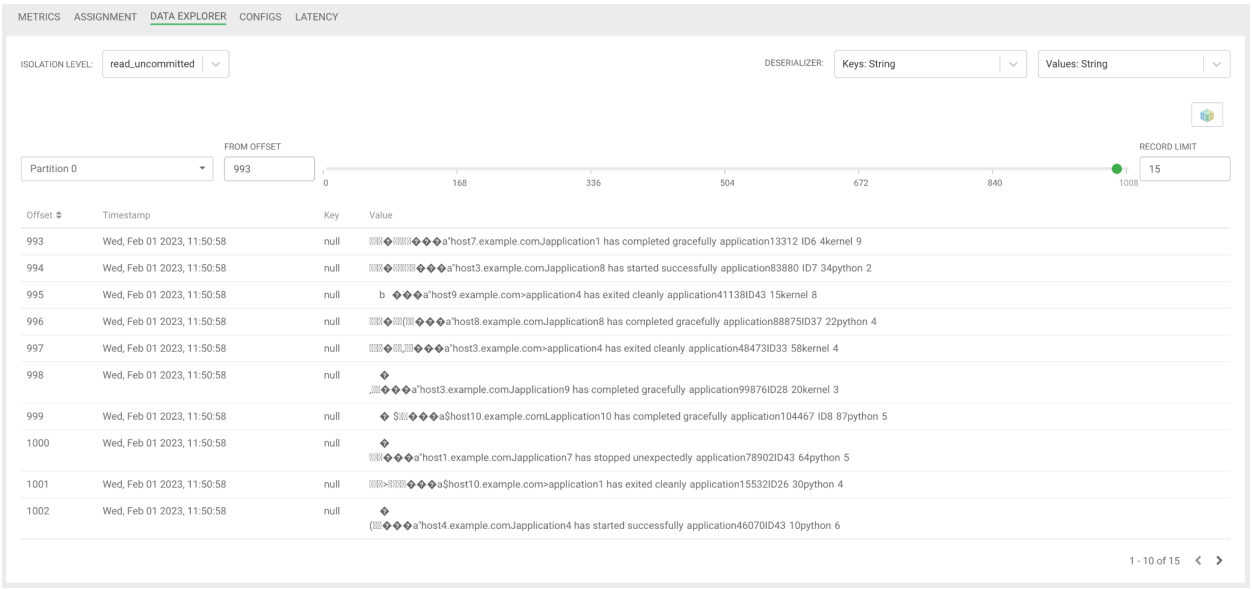
6. Access the SMM UI from the Streams Messaging DataHub page and check the topic metrics and data:



Here are the topic metrics:



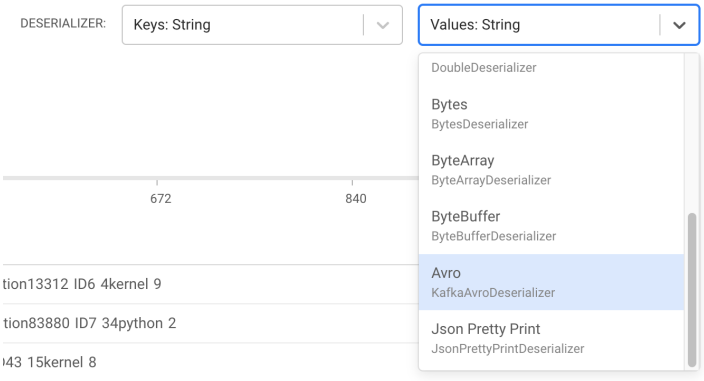
You can also view the data from the data explorer:



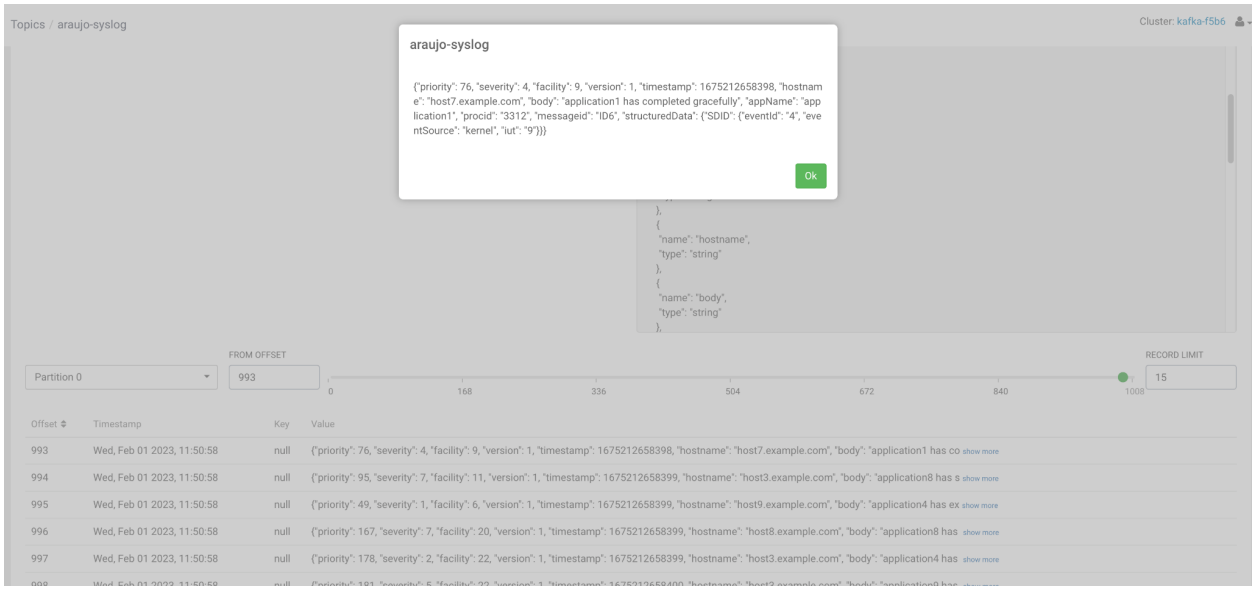
7. You can see that the data in that topic is binary (lots of garbage-like characters on the screen. This is because the topic contains Avro messages, which is a binary serialization format.

   Fortunately, SMM is integrated to Schema Registry and can fetch the correct schema to properly deserialize the data and present a human-readable form of it.

   To do that select **Avro** as the **Values** Deserializer for the topic:



   You will notice that after Avro is selected the messages are shown with a readable JSON encoding:



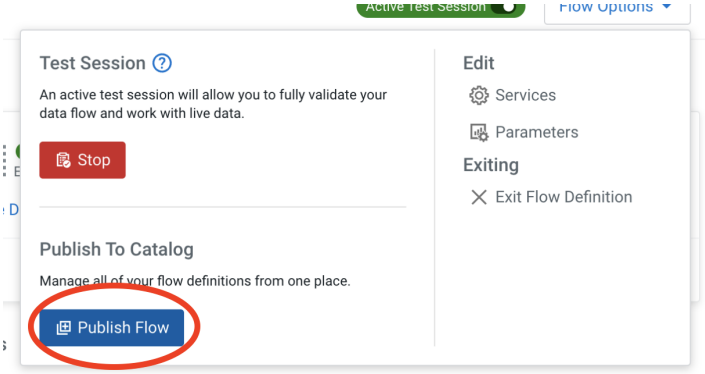
6. Version Control the flow into the DataFlow Catalog

When your flow draft has been tested and is ready to be used you can publish it to the DataFlow Catalog so that you can deploy it from there.

On the first time a flow draft is exported to the catalog you are asked to provide a name for the flow. This name must be unique and must not already exist in the catalog. After a flow is published, subsequent changes that are published will be saved in the catalog as new versions using the same name provided the first time. This name cannot be changed.

When you want to publish a draft flow as a flow definition, you have two options:

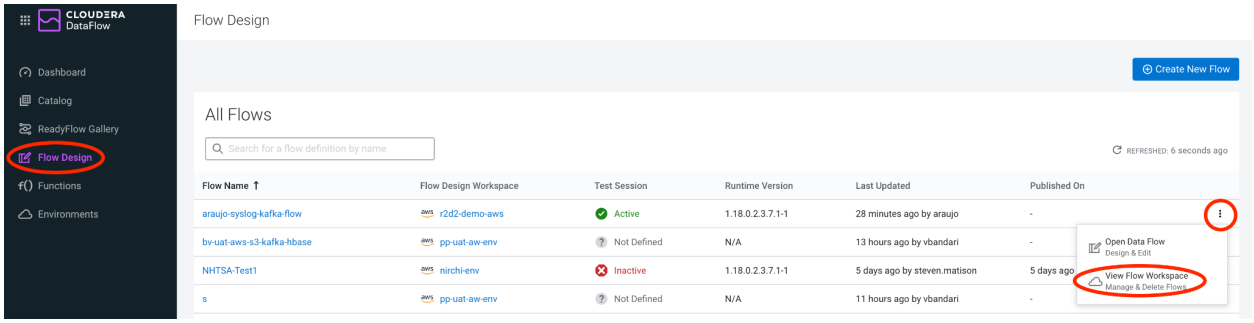
- On the Flow Design Canvas, click Flow Options > Publish To Catalog > Publish Flow.



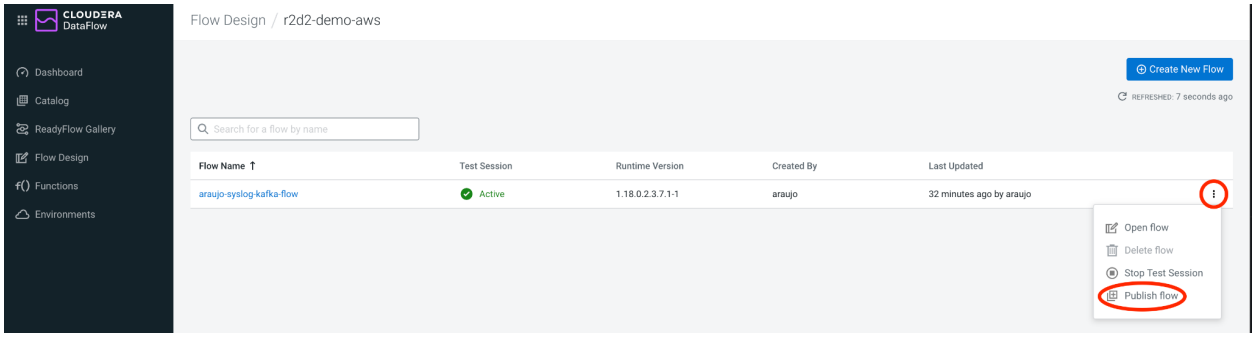
OR

- Click on **Flow Design** (left-hand side) to see the **All Flows** page, which lists all the flows. Then click on the ⋮ menu for the flow you want to publish and select **View Flow**

Workspace.



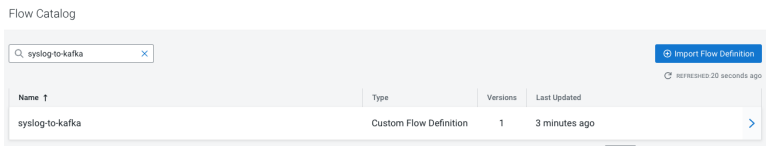
This will take you to the list of all flows running on the same environment (workspace) as the flow you selected. To publish the flow, Workspace view, click again on the ⋮ menu for the desired flow and select **Publish Flow**.



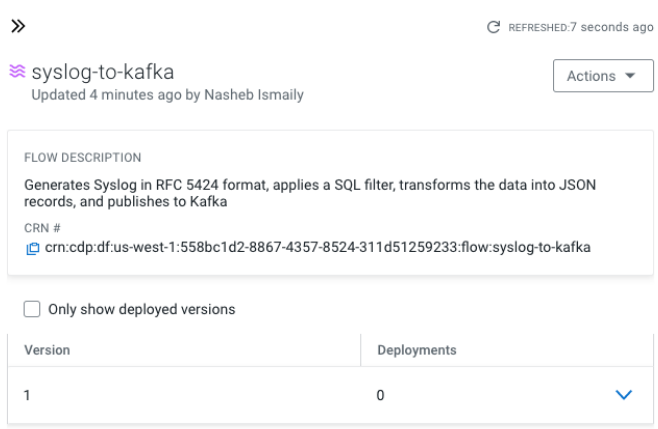
1. Choose one of the methods explained above and publish your flow.
2. In the Publish Flow dialog box, enter the following details:
    a. Flow Name: only when you publish your flow for the first time.
    b. Flow Description: only when you publish your flow for the first time.
    c. Version Comments: every time a flow version is published.
3. Click **Publish**.
4. Click on **Catalog** and verify that your flow was successfully published.
5. Make a simple change to your flow (e.g move processors to different positions)
6. Publish your flow again.
7. Check that your flow in the Catalog has multiple versions now.

7. Deploy the flow in Production

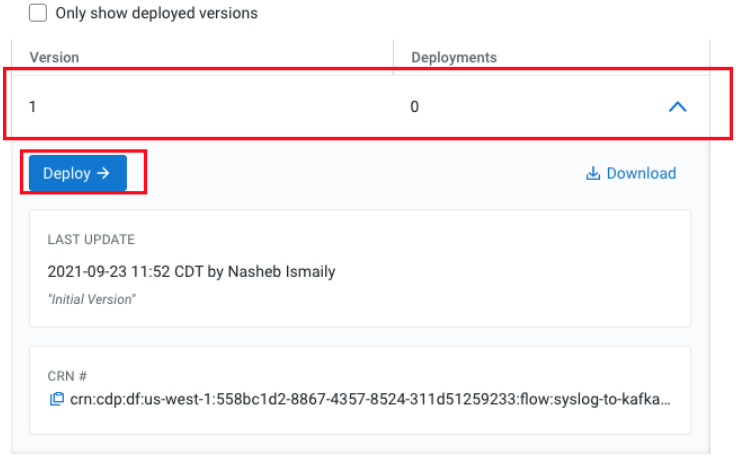
1. Search for the flow in the Flow Catalog



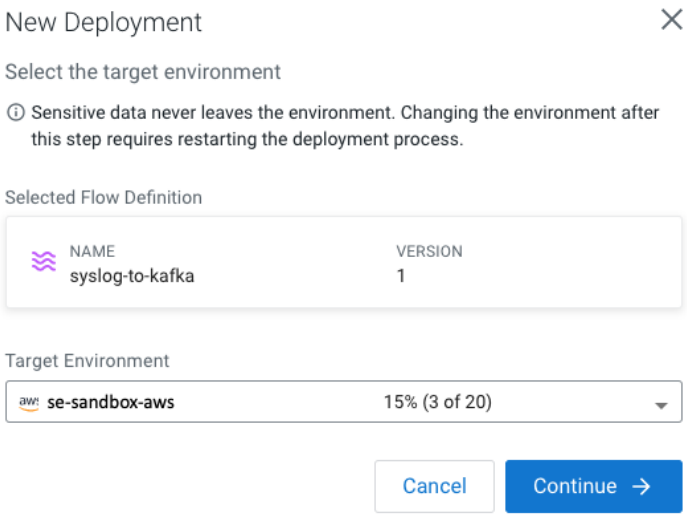
2. Click on the Flow to see its details, including the list of versions:



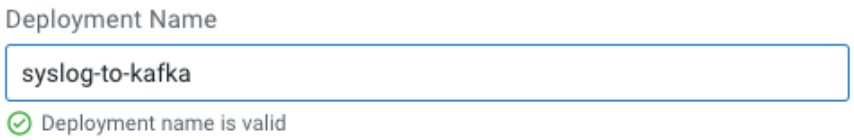
3. Click on **Version 1**, you should see a **Deploy** Option appear shortly. Then click on **Deploy**.



4. Select the CDP environment where this flow will be deployed and click **Continue**

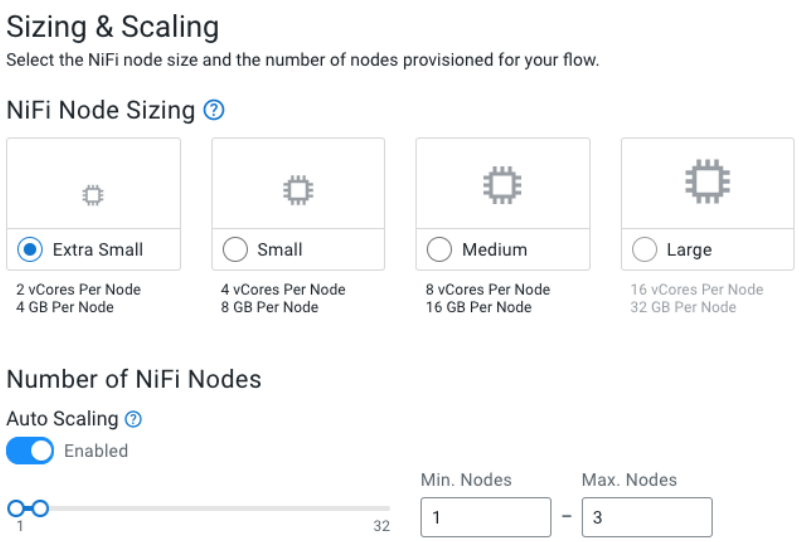


5. Give the deployment a unique name (e.g **<userid>-syslog-to-kafka-001**), then click **Next**



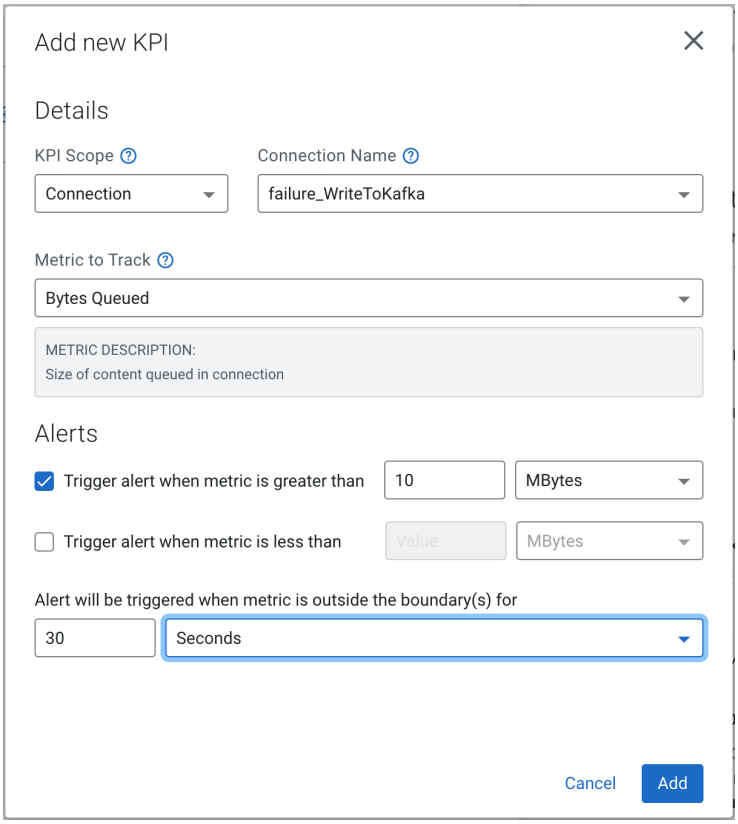
6. In the **NiFi Configuration** page, accept all the defaults (runtime version, autostart behavior, inbound connections and custom NAR) and click **Next**.
7. In the **Parameters** page, provide the correct values for the parameter for the production run and then click **Next**. Most of the parameters already have good defaults and you only need to change them if needed. However, you must re-enter the **CDP Workload User Password**.
    - 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 Avro Topic: **<userid>-syslog-avro**
    - Kafka Destination JSON Topic: **<userid>-syslog-json**
    - Kafka Producer ID: **<userid>-producer**
    - Schema Name: **<userid>-syslog-avro**
    - Schema Registry Hostname: The hostname of the master server in the Kafka Datahub.
    - Filter Rule: **SELECT * FROM FLOWFILE**

8. In the **Sizing & Scaling**, select the following and then click **Next**:
    - Size: **Extra Small**
    - Enable Auto Scaling: **True**
    - Min Nodes: **1**
    - Max Nodes: **3**

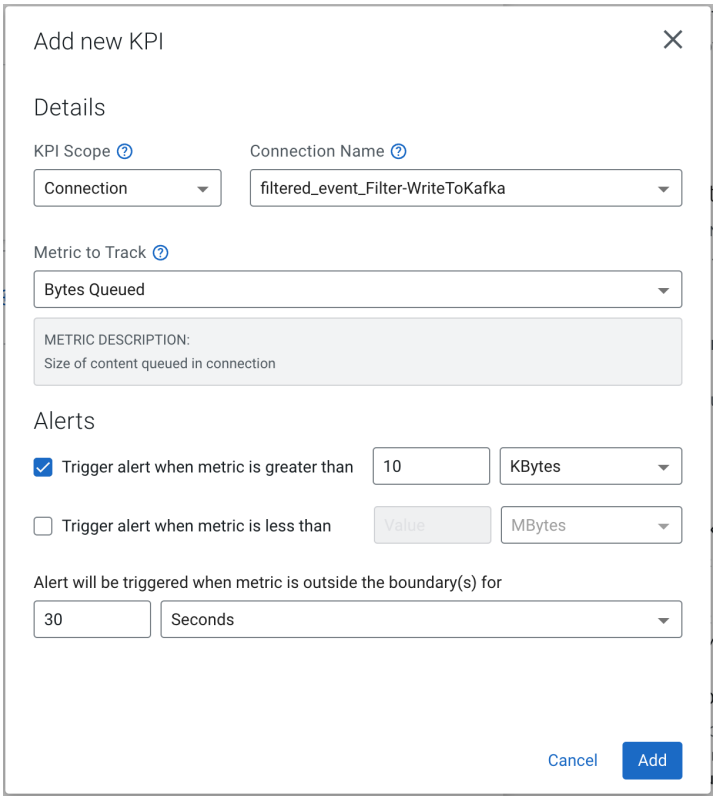


9. In the Key Performance Indicators page, click on Add New KPI to add the following KPIs.

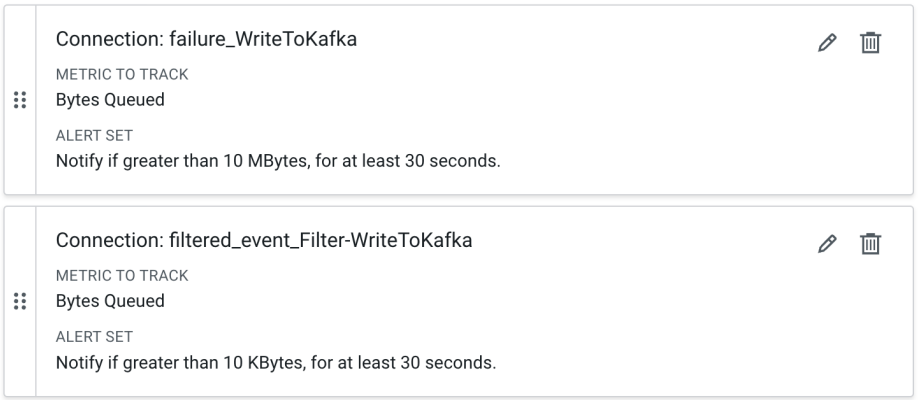
   a. Add the following KPI
      - KPI Scope: **Connection**
      - Connection Name: **failure_WriteToKafka**
      - Metrics to Track: **Bytes Queued**
      - Alerts:
        - Trigger alert when metric is greater than: **1 MB**
        - Alert will be triggered when metrics is outside the boundary(s) for: **30 seconds**



b. Add the following KPI
   - KPI Scope: **Connection**
   - Connection Name: **filtered_event_Filter-WriteToKafka**
   - 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**



10. Review the KPIs and click **Next**.



11. In the **Review** page, review your deployment details.

    Notice that in this page there's a **>_ View CLI Command** link. You will use the information in the page in the next section to deploy a flow using the CLI. For now you just need to save the script and dependencies provided there:

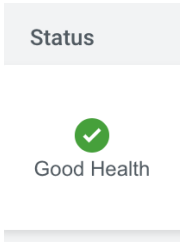
    a. Click on the **>_ View CLI Command** link and familiarize yourself with the content.
    b. Download the 2 JSON dependency files by click on the download button:

i. Flow Deployment Parameters JSON
ii. Flow Deployment KPIs JSON

c. Copy the command at the end of this page and save that in a file called **deploy.sh**
d. Close the **Equivalent CDP CLI Command** tab.

12. Click Deploy to initiate the flow deployment.

13. In the DataFlow **Dashboard,** monitor your flow until it's running successfully (a green check mark will appear once the deployment has completed)



14. Click on your deployment and explore the flow details and monitor the KPI metrics that are shown on this page.

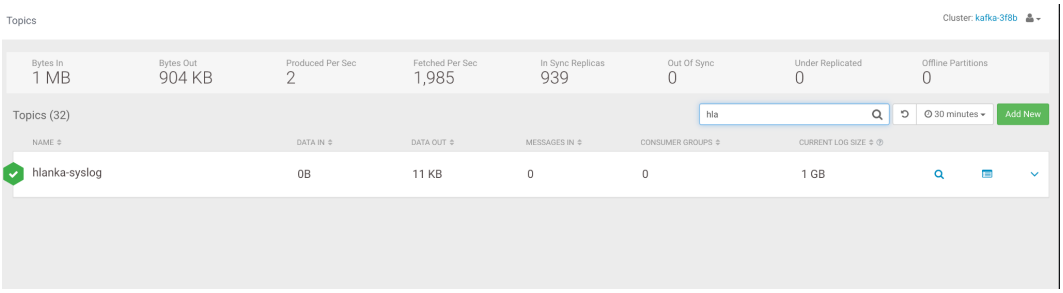
15. Then click on **Manage Deployment** and explore the options under Deployment Settings, which allow you to manage your production flow deployment.

16. Explore the links in the **Actions** menu on this page.

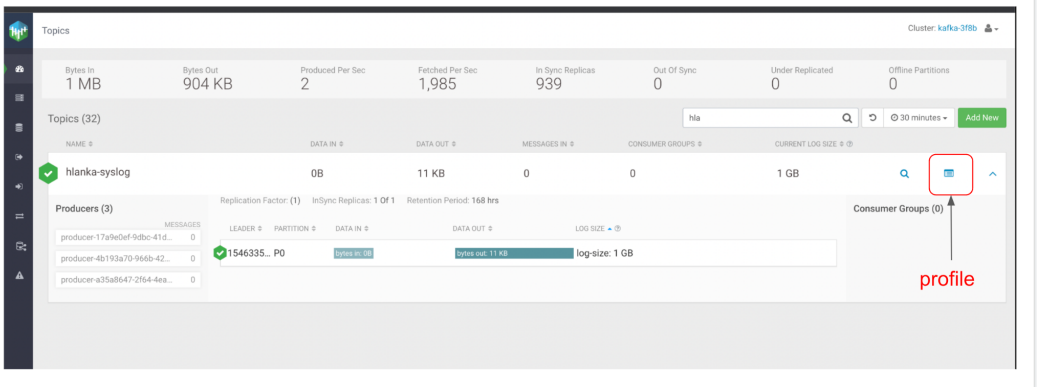
8. Inspect Kafka topics in SMM

Use SMM to to show the kafka topics **<userid>-syslog-avro** that is being written to from the NiFi flow in module 1

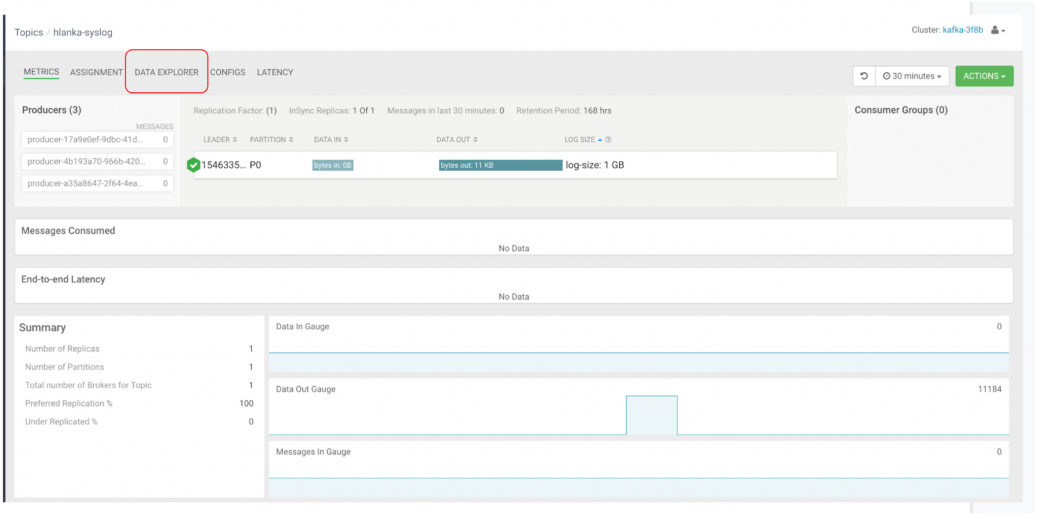
1. Login to **SMM**.
2. Click on the **Topics icon** ( ) and filter the topic **<userid>-syslog-avro**, which was created by the NiFi flow in Module 1



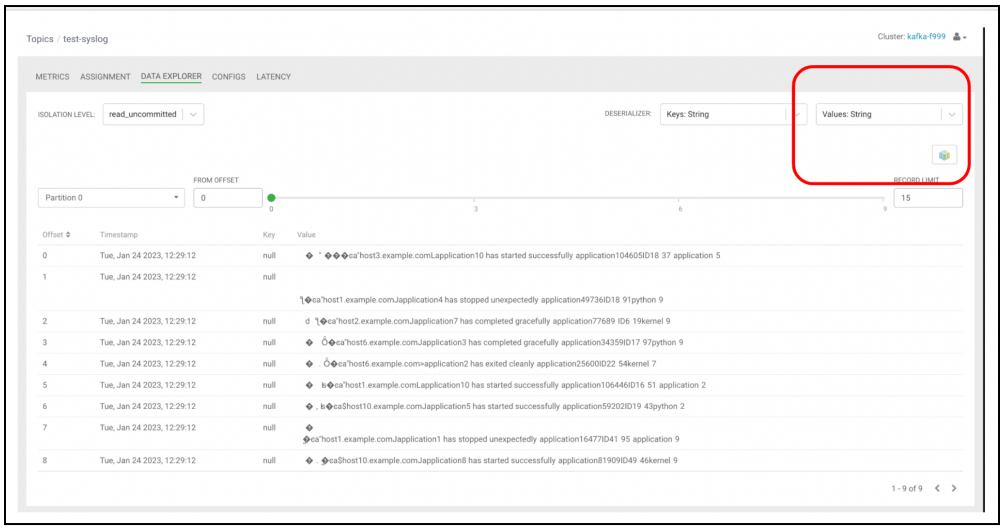
3. Click on the **Profile** icon for the topic, as shown below:

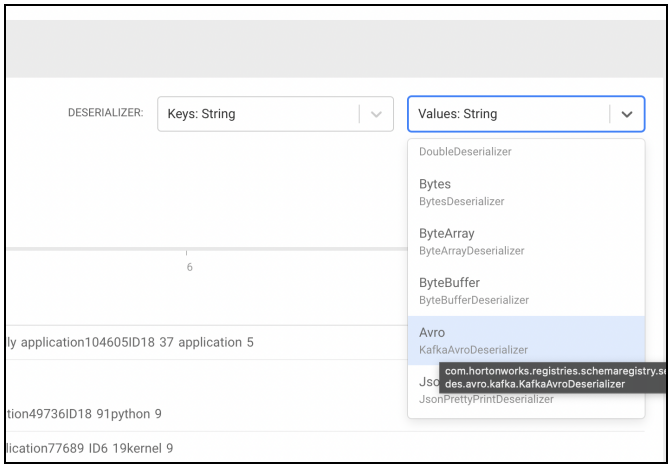


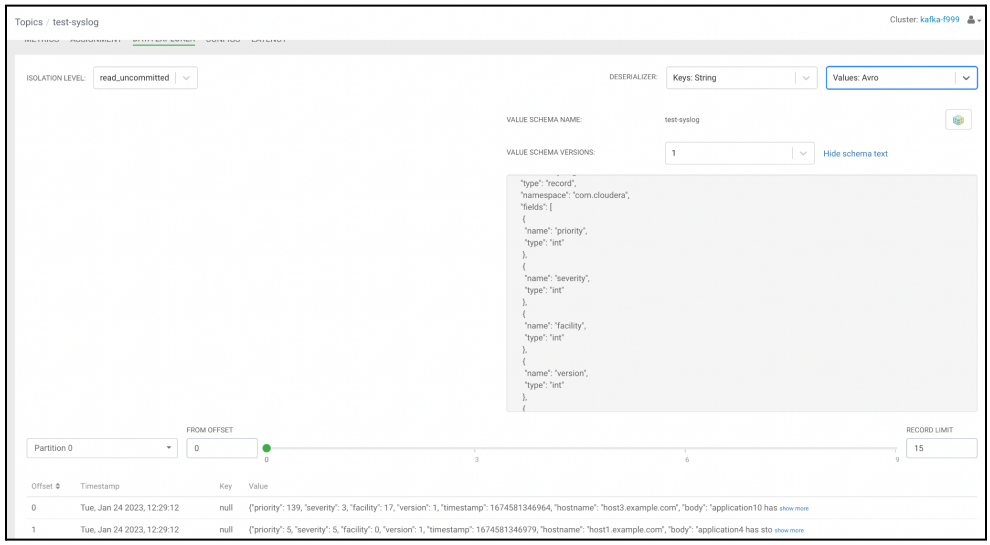
4. On the Topic page, click on the **DATA EXPLORER** tab:



5. Use the Data Explorer tool for the Avro topic and verify that the data is binary. Change the Value Deserializer to **Avro** to use the Schema Registry integration and verify that the data was deserialized correctly:




DESERIALIZER:
Keys: String
Values: String
DoubleDeserializer
Bytes
BytesDeserializer
ByteArray
ByteArrayDeserializer
ByteBuffer
ByteBufferDeserializer
Avro
KafkaAvroDeserializer
com.hortonworks.registries.schemaregistry.s
des.avro.kafka.KafkaAvroDeserializer
JsonPrettyPrintDeserializer
6
ly application104605ID18 37 application 5
tion49736ID18 91python 9
lication77689 ID6 19kernel 9


Topics / test-syslog
Cluster: kafka-f999
ISOLATION LEVEL:
read_uncommitted
DESERIALIZER:
Keys: String
Values: Avro
VALUE SCHEMA NAME:
test-syslog
VALUE SCHEMA VERSIONS:
1
Hide schema text
"type": "record",
"namespace": "com.cloudera",
"fields": [
{
"name": "priority",
"type": "int"
},
{
"name": "severity",
"type": "int"
},
{
"name": "facility",
"type": "int"
},
{
"name": "version",
"type": "int"
},
FROM OFFSET
RECORD LIMIT
Partition 0
0
15
0
3
6
9
Offset
Timestamp
Key
Value
0
Tue, Jan 24 2023, 12:29:12
null
{"priority": 139, "severity": 3, "facility": 17, "version": 1, "timestamp": 1674581346964, "hostname": "host3.example.com", "body": "application10 has
show more
1
Tue, Jan 24 2023, 12:29:12
null
{"priority": 5, "severity": 5, "facility": 0, "version": 1, "timestamp": 1674581346979, "hostname": "host1.example.com", "body": "application4 has sto
show more