

DataFlow Workshop

Module 1 -Building Flow with NiFi & Interacting with Kafka	2
1. Overview	2
2. Create a schema in Schema Registry	2
3. Create a Kafka topic manually	4
4. Design the flow in the Flow Designer	6
4.1 Create a new flow	6
4.2 Configure parameters for the flow	6
4.3 Create Controller Services	12
4.4 Design the flow	16
4.5 Naming the queues	25
5. Interactively test the flow	27
6. Version Control the flow into the DataFlow Catalog	31
7. Deploy the flow in Production	32
8. Inspect Kafka topics in SMM	39

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.

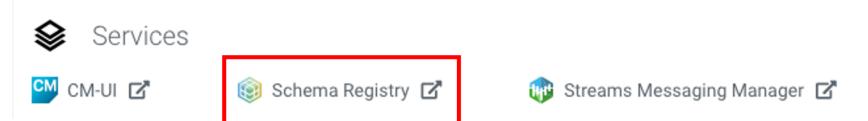
You'll also learn flow design best practices like:

- 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

Status	Name	Cloud Provider	Environment	Data Hub Type	Version	Node Count	Created
Running	meetup-2	AWS	meetup-env	7.2.16 - Streams Messaging Light Duty: Apache Kafka, Schema Registry, Streams Messaging Manager, Streams Replication Manager, Cruise Control	CDH 7.2.16	4	04/11/23, 08:37 PM GMT+10
Running	meetup-3	AWS	meetup-env	7.2.16 - Streams Messaging Light Duty: Apache Kafka, Schema Registry, Streams Messaging Manager, Streams Replication Manager, Cruise Control	CDH 7.2.16	4	04/11/23, 08:37 PM GMT+10
Running	meetup-1	AWS	meetup-env	7.2.16 - Streams Messaging Light Duty: Apache Kafka, Schema Registry, Streams Messaging Manager, Streams Replication Manager, Cruise Control	CDH 7.2.16	4	04/11/23, 08:36 PM GMT+10

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

The screenshot shows the Apache Kafka 'Topics' page. At the top, there are summary statistics: Bytes In (928 MB), Bytes Out (7 MB), Produced Per Sec (472), Fetched Per Sec (18,594), In Sync Replicas (999), Out Of Sync (0), Under Replicated (0), and Offline Partitions (0). Below this is a table titled 'Topics (75)' with columns: NAME, DATA IN, DATA OUT, MESSAGES IN, CONSUMER GROUPS, and CURRENT LOG SIZE. A specific topic, 'connect-configs', is selected and highlighted with a green checkmark. At the bottom right of the table, there is a red box around the 'Add New' button.

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

The screenshot shows the 'Add Topic' dialog box. It has two main sections: 'TOPIC NAME' and 'PARTITIONS'. The 'TOPIC NAME' field contains 'user50-syslog-kafka-json'. The 'PARTITIONS' field contains '2'. Below these, there is a section titled 'Availability' with five options: 'MAXIMUM', 'HIGH', 'MODERATE', 'LOW' (which is selected and highlighted with a green box), and 'CUSTOM'. Underneath the availability section, there are four pairs of replication settings: 'REPLICATION FACTOR 3 MIN INSYNC REPLICA 2', 'REPLICATION FACTOR 3 MIN INSYNC REPLICA 1', 'REPLICATION FACTOR 2 MIN INSYNC REPLICA 1', and 'REPLICATION FACTOR 1 MIN INSYNC REPLICA 1'. At the bottom left is a 'Limits' section with a 'CLEANUP.POLICY' dropdown set to 'compact'. At the bottom right are three buttons: 'Advanced', 'Cancel', and 'Save' (which is highlighted with a green box).

Create a new topic with the following information:

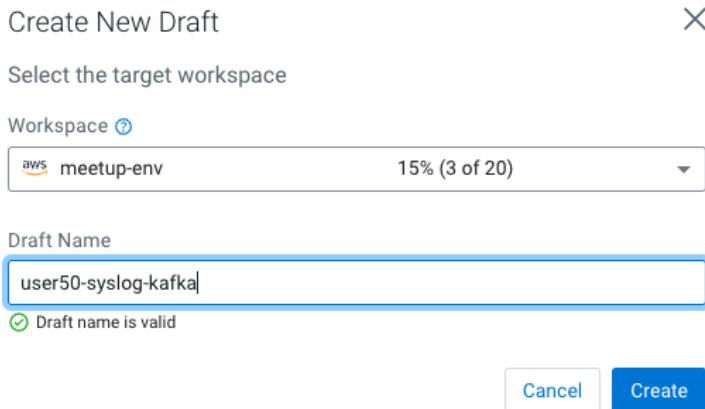
- Name: **<userid>-syslog-kafka-json**
- Availability: **Low**
- Cleanup Policy: **Compact**
- Partitions: **2**

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**



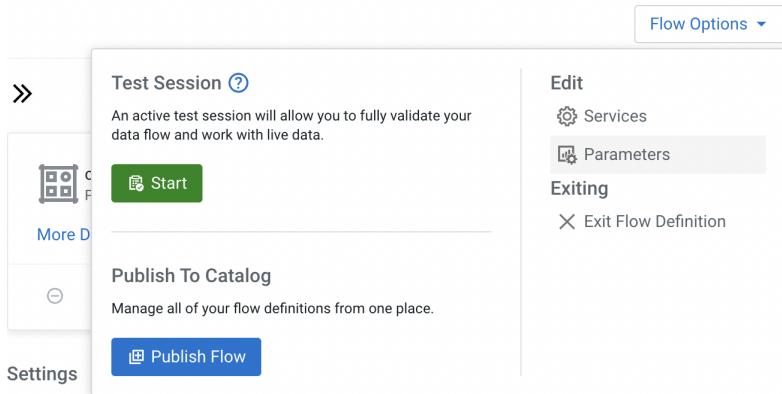
4.2 Configure parameters for the flow

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.

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

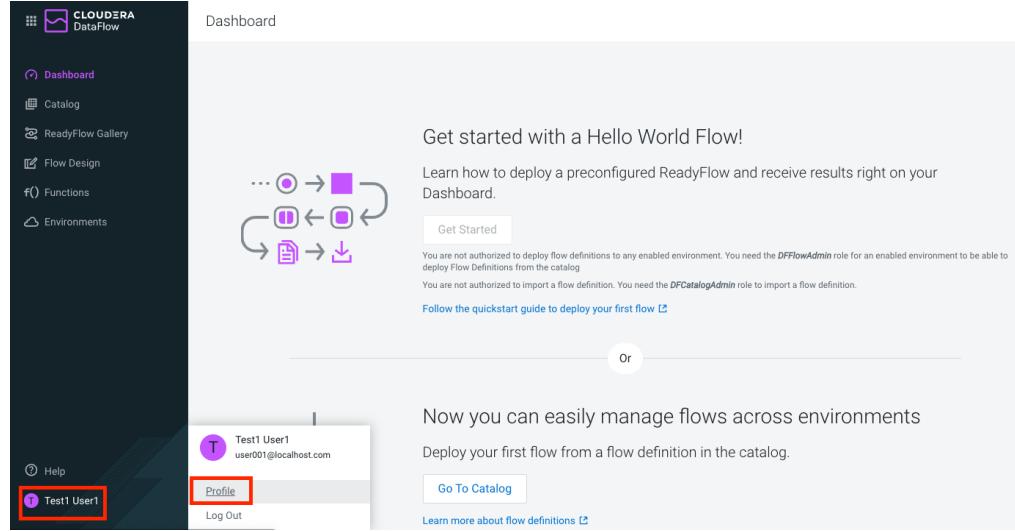


- Add the parameters. Click on **Add Parameter > Add Parameter** for each parameter to be added and enter the appropriate details:

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 screenshot shows the Apache Kafka 'Brokers' page. At the top, there are two summary metrics: 'Total Bytes In' (527 KB) and 'Total Bytes Out' (324 KB). Below this, a section titled 'Brokers (3)' lists three brokers. Each broker entry includes a green checkmark icon, a unique ID, and the broker's name and port number. The broker names are 'kafka-dfx-workshop-corebroker1.se-sandb.a465-9q4k.cloudera.site:9093', 'kafka-dfx-workshop-corebroker2.se-sandb.a465-9q4k.cloudera.site:9093', and 'kafka-dfx-workshop-corebroker0.se-sandb.a465-9q4k.cloudera.site:9093'. These entries are highlighted with red boxes.

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

The screenshot shows the 'Add Parameter' dialog box. The 'Name' field is set to 'Kafka Broker Endpoint'. The 'Value' field contains the text: 'kafka-dfx-workshop-corebroker1.se-sandb.a465-9q4k.cloudera.site:9093, kafka-dfx-workshop-corebroker2.se-sandb.a465-9q4k.cloudera.site:9093, kafka-dfx-workshop-corebroker0.se-sandb.a465-9q4k.cloudera.site:9093'. A note at the bottom of the 'Value' field says: 'A test session must be running in order to add files.' There is also a checkbox labeled 'Set empty string' and a 'Description' field with placeholder text 'Add description'. At the bottom right are 'Cancel' and 'Apply' buttons.

- **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:

kafka-dfx-workshop

Status	Instances	Configuration	Commands	Charts Library	Quick Links
	schemaregistry				
<input type="text" value="Search"/> <input checked="" type="checkbox" value="Filters"/> Filters					
Actions for Selected <input type="checkbox"/>		Status ↑	Role Type	State	Hostname
<input type="checkbox"/> HEALTH TEST					Commiss
		<input type="checkbox"/>	Schema Registry Server	Started	kafka-dfx-workshop-master0.se-sandb.a465-9q4k.cloudera.site
Commiss					

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**

Get started with a Hello World Flow!

Learn how to deploy a preconfigured ReadyFlow and receive results right on your Dashboard.

Get Started

You are not authorized to deploy flow definitions to any enabled environment. You need the **DFFlowAdmin** role for an enabled environment to be able to deploy Flow Definitions from the catalog

You are not authorized to import a flow definition. You need the **DFCatalogAdmin** role to import a flow definition.

Follow the quickstart guide to deploy your first flow [»](#)

Or

Now you can easily manage flows across environments

Deploy your first flow from a flow definition in the catalog.

[Go To Catalog](#)

Learn more about flow definitions [»](#)

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

The screenshot shows the Cloudera Management Console interface. On the left, there's a sidebar with links like Dashboard, Environments, Data Lakes, User Management (which is currently selected), Data Hub Clusters, Data Warehouses, ML Workspaces, and Classic Clusters. The main area is titled 'Users / Test1 User1'. It displays user details: Name (Test1 User1), Email (user001@localhost.com), Workload User Name (user001), CRN (cm:altus:iam:us-west-1:3b938e1b-8421-49e9-b073-2e74b44262c0:user:168...), Tenant ID (3b938e1b-8421-49e9-b073-2e74b44262c0), Identity Provider (se-workshop-3-keycloak-idp), Last Interactive Login (04/12/2023 2:23 PM AEST), Profile Management (View profile), and Workload Password (Set Workload Password (Workload password is currently set)). An 'Actions' dropdown menu is visible on the right.

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

The screenshot shows a 'Set Workload Password' dialog. It has two password input fields: one for 'Password' containing '*****' and one for 'Confirm Password' also containing '*****'. Below the fields is a note: ' ⓘ If you use keytabs, you need to regenerate them after changing your workload password. You can do this from your user profile > Actions > Get Keytab.' At the bottom is a blue 'Set Workload Password' button.

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

The screenshot shows the 'Flow Design / se-sandbox-aws / hlanka-syslog-kafka-flow / Parameters' page. At the top, there are validation messages: ' ⓘ To fully validate your flow and develop with live data, start a test session.' and ' ⚠ You must first apply or discard the changes that have been made.' Below these are buttons for 'Back To Flow Designer' and 'Add Parameter'. The main area is a table with columns 'Name ↑', 'Value', and 'Changed'. Each row has a 'New' button with a trash icon. The parameters listed are: CDP Workload User (hlanka), CDP Workload User Password (Sensitive value set), Filter Rule (SELECT * FROM FLOWFILE), Kafka Broker Endpoint (kafka-dfx-workshop-corebroker1.se-sandb.a465-9q4k.cloudera.site:9093, kafka-dfx-workshop-corebroker2.se-sandb.a465-9q4k.cloudera.site:9093, kafka-dfx-workshop-co...), Kafka Destination Topic (hlanka-syslog), Kafka Producer ID (hlanka-producer), Schema Name (syslog), and Schema Registry Hostname (kafka-dfx-workshop-master0.se-sandb.a465-9q4k.cloudera.site).

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**.

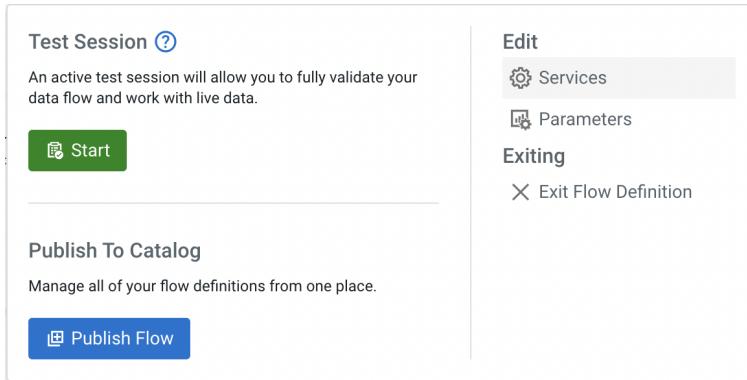
You are not yet at the point of testing the flow, but enabling a test session forces the **Default Nifi SSL Context Service** controller service to be created, which you will need in the next steps.

The screenshot shows the NiFi interface with the following details:

- Header:** Flow Design / se-sandboxx-aws / hlanka-syslog-kafka-flow / Services
- Toolbar:** Back To Flow Designer, Add Service, Initializing Test Session..., Flow Options ▾
- Table View:** Shows a single row for "Default Nifi SSL Context Service" of type "StandardRestrictedSSLContextService V1.1...".
- Details Panel:** Displays the service configuration:
 - Default Nifi SSL Context Service** (StandardRestrictedSSLContextService V1.1.0.2.3.7.0-99)
 - More Details ▾**
 - Settings:** Service Name: Default Nifi SSL Context Service
 - Comments:** (empty)
 - Properties:** A table with columns "Property" and "Value":

Property	Value
Keystore Filename ⓘ	\$(Default SSL Context Keystore)
Keystore Password ⓘ	\$(Default SSL Context Keystore Password)
Key Password ⓘ	\$(Default SSL Context Keystore Password)
Keystore Type ⓘ	\$(Default SSL Context Keystore Type)
Truststore Filename ⓘ	\$(Default SSL Context Truststore)
Truststore Password ⓘ	\$(Default SSL Context Truststore Passwor...)
Truststore Type ⓘ	\$(Default SSL Context Truststore Type)
TLS Protocol ⓘ	TLS

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



3. Add a new HortonworksSchemaRegistry controller service

- Click **Add Service**. The **Add Service** page opens.

- In the text box, filter for **HortonworksSchemaRegistry**, select it and click **Add**
- 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:

✓ **Parameters (PARAM) supported**

After beginning with the start delimiter `#{` use `Ctrl + Space` to see a 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}}**

Settings

*Service Name
WS_CDP_Schema_Registry

Comments

Properties

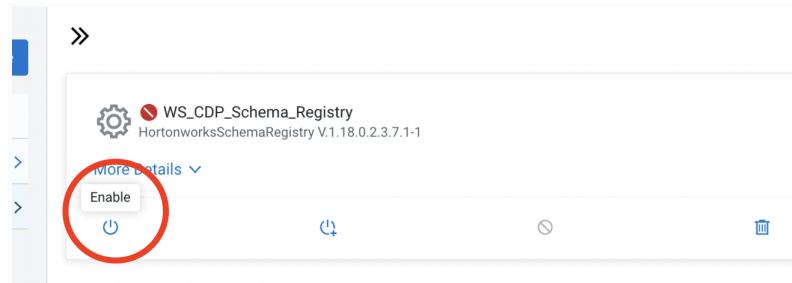
Property	Value
Schema Registry URL ?	https://#{Schema Registry Hostname}:7790/api/v1
Cache Size ?	1000
Cache Expiration ?	1 hour
SSL Context Service ?	Default NiFi SSL Context Service
Kerberos Credentials Service ?	No value set
Kerberos Principal ?	#{{CDP Workload User}}
Kerberos Password ?	#{{CDP Workload User Password}}
Basic Authentication Username ?	No value set
Basic Authentication Password ?	No value set

Referencing Components

No referencing Processors to display.

[Apply](#) [Discard Changes](#)

- 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**
 - j. Click **Apply**
 - k. 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:

Flow Design / r2d3-demo-aws / araujo-syslog / Services

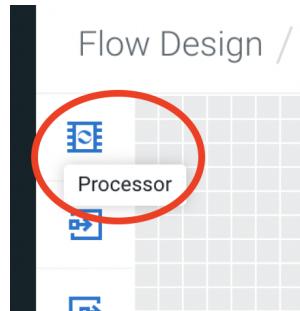
State	Name ↑	Type	Notifications
✓	Default NiFi SSL Context Service	StandardRestrictedSSLContextService V.1.18...	>
✓	WS_Avro_Syslog_Writer	AvroRecordSetWriter V.1.18.0.2.3.7.3-1	>
✓	WS_CDP_Schema_Registry	HortonworksSchemaRegistry V.1.18.0.2.3.7.3...	>
✓	WS_JSON_Syslog_Reader	JsonTreeReader V.1.18.0.2.3.7.3-1	>
✓	WS_JSON_Syslog_Writer	JsonRecordSetWriter V.1.18.0.2.3.7.3-1	>
✓	WS_Syslog_5424_Reader	Syslog5424Reader V.1.18.0.2.3.7.3-1	>

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**

A screenshot of the configuration dialog for the ExecuteScript processor. It has a tab labeled 'Scheduling'. Under 'Scheduling Strategy', it is set to 'Timer Driven'. Under 'Concurrent Tasks', it is set to '4'. Under 'Run Duration', it is set to '0ms'. Under 'Run Schedule', it is set to '0 sec'. Under 'Execution', it is set to '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(bytarray(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

Relationships

You can choose to automatically terminate and/or retry FlowFiles sent to a given relationship if it is not defined elsewhere. If both terminate and retry are selected, retry logic will occur first, followed by termination.

success ⓘ

Terminate Retry

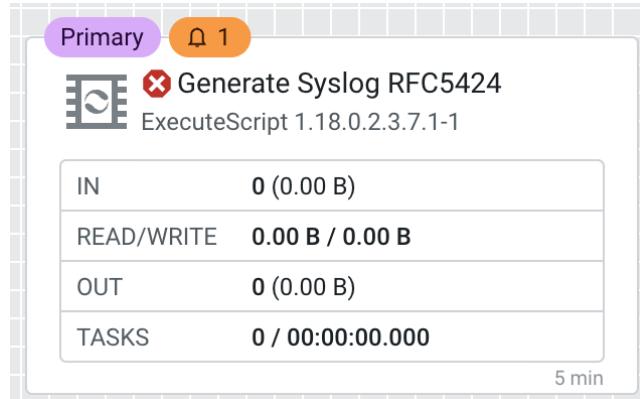
failure ⓘ

Terminate Retry

Retry logic specified below will apply to all relationships for this processor that are set to retry.

*Number of Retry Attempts ⓘ

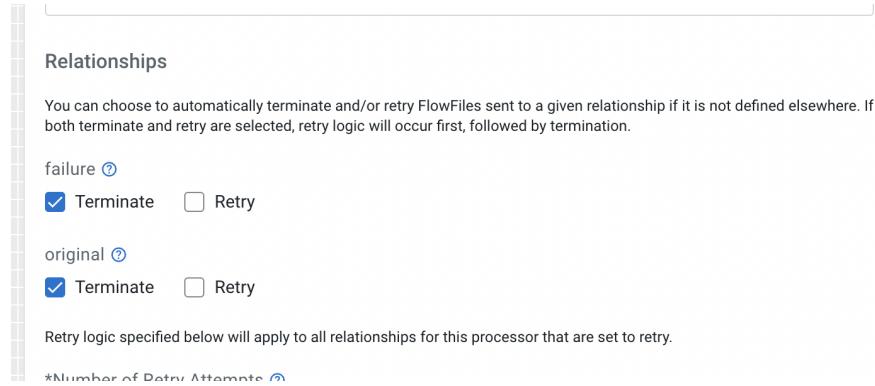
d. Click **Apply**



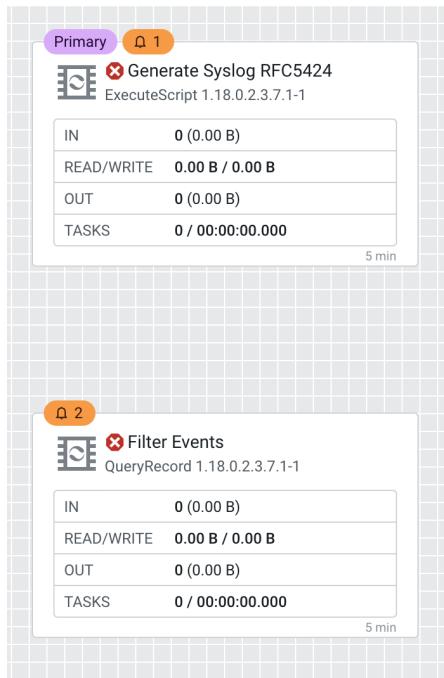
3. Create the **Filter Events** processor

- 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**



4.5 Naming the queues

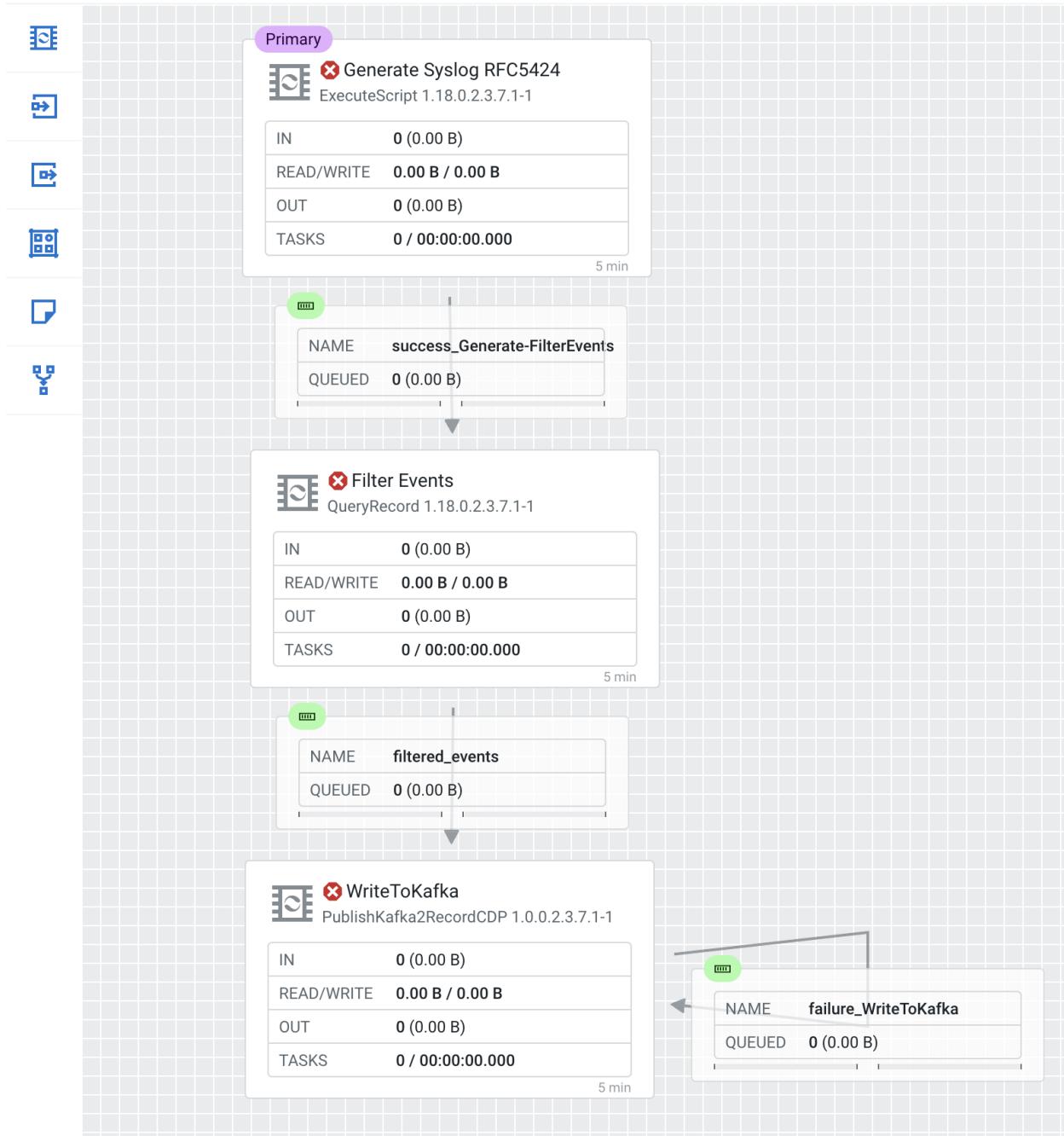
Providing unique names to all queues is important when creating flows in CDP-PC. The queue names are used to define Key Performance Indicators and having unique names for them make it easier to understand and create CDF-PC dashboards.

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

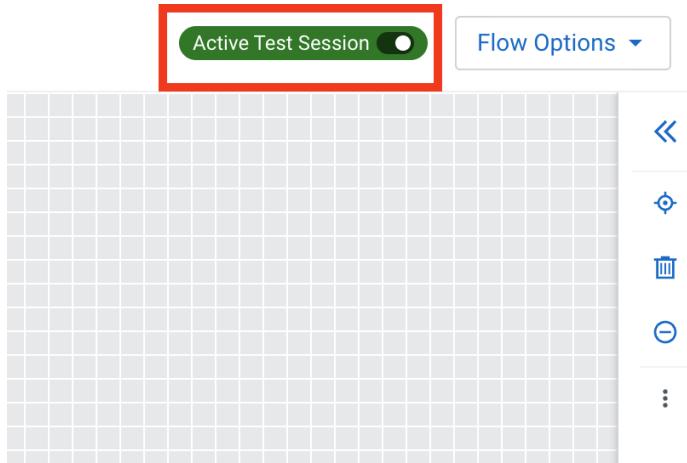


5. Interactively test the flow

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.



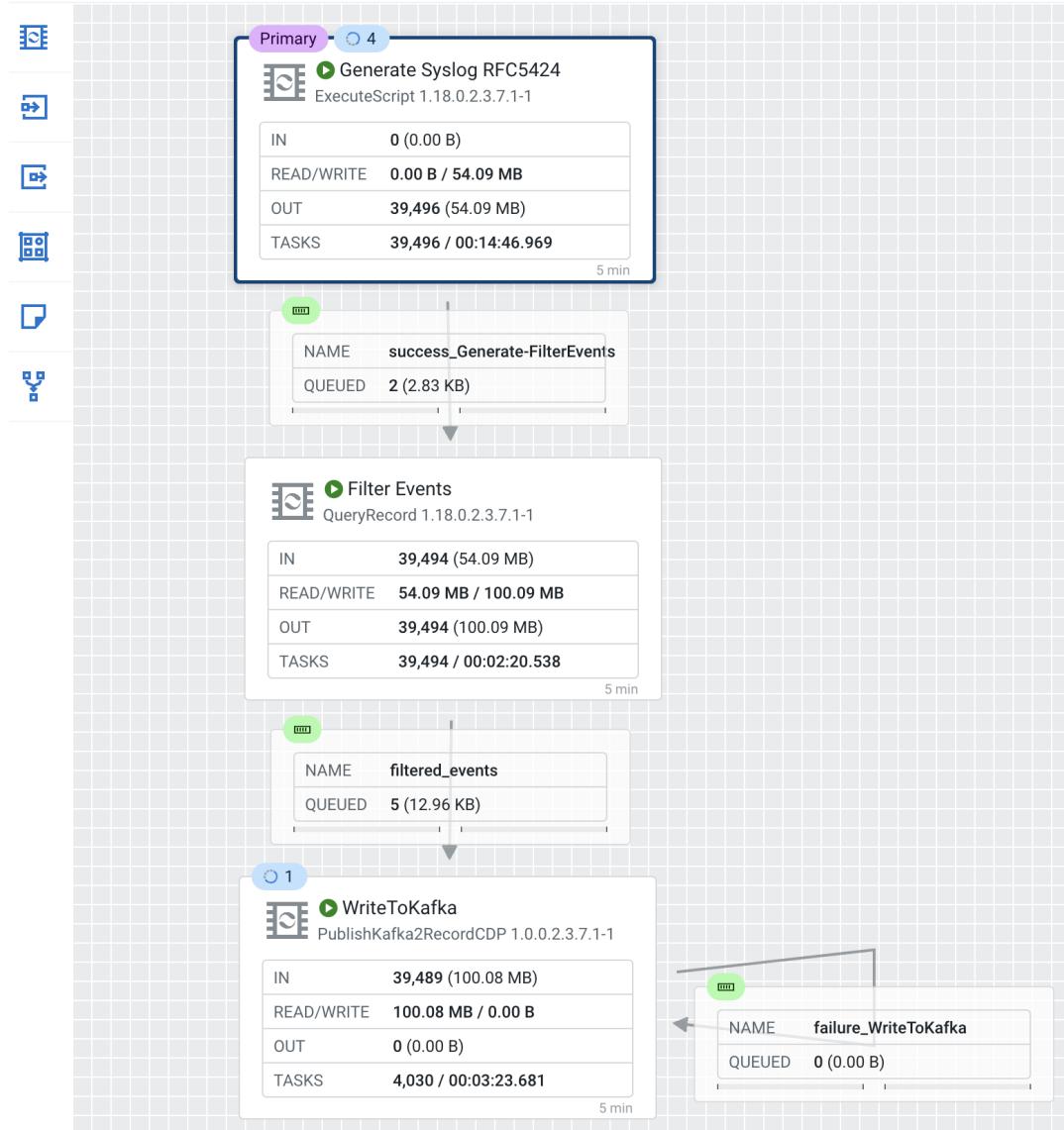
1. If your test session is not yet started, start one by clicking on **Flow Options > Test Session > Start**.
- Note:** If you had previously started and stopped a test session, the **Start** button is replaced with the **Restart** button. Clicking on **Restart** restarts the session with the same settings 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.

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



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:

The screenshot shows the SMM Data Explorer interface. At the top, there are tabs for METRICS, ASSIGNMENT, DATA EXPLORER (which is selected), CONFIGS, and LATENCY. Below the tabs, there are two dropdown menus: 'ISOLATION LEVEL' set to 'read_uncommitted' and 'DESERIALIZER' set to 'Keys: String | Values: String'. A 'RECORD LIMIT' dropdown is also present, set to 15. The main area displays a table with columns: Offset, Timestamp, Key, and Value. The table shows 10 records from offset 993 to 1002. The 'Value' column contains binary data which is partially decoded into readable JSON. For example, the first record's value is 'host7.example.com>application1 has completed gracefully application13312 ID6 4kernel 9'. The last record's value is 'host4.example.com>application4 has started successfully application46070ID43 10python 6'. The bottom right corner of the table indicates '1 - 10 of 15'.

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:

This screenshot shows a dropdown menu for selecting a Deserializer. The 'Values' dropdown is highlighted. The options listed are: DoubleDeserializer, Bytes, BytesDeserializer, ByteArray, ByteArrayDeserializer, ByteBuffer, ByteBufferDeserializer, Avro (which is selected and highlighted in blue), KafkaAvroDeserializer, and Json Pretty Print. The background of the interface shows some log entries: 'tion13312 ID6 4kernel 9', 'tion83880 ID7 34python 2', and 'i43 15kernel 8'.

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

The screenshot shows the Apache Kafka UI interface. At the top, there's a modal window titled "araudo-syslog" displaying a JSON message. The message content is partially visible, showing fields like "priority", "severity", "facility", "version", "timestamp", "hostname", "body", and "procid". A green "Ok" button is at the bottom right of the modal. Below the modal is a table titled "Partition 0" showing log entries from offset 993 to 1008. The table has columns for Offset, Timestamp, Key, and Value. The "Value" column displays the raw JSON messages. A "RECORD LIMIT" dropdown is set to 15.

Offset	Timestamp	Key	Value
993	Wed, Feb 01 2023, 11:50:58	null	{"priority": 76, "severity": 4, "facility": 9, "version": 1, "timestamp": 1675212658398, "hostname": "host7.example.com", "body": "application1 has completed gracefully", "appName": "application1", "procid": "3312", "messageId": "ID6", "structuredData": {"SDID": "event1", "eventSource": "kernel", "iut": "9"}})
994	Wed, Feb 01 2023, 11:50:58	null	{"priority": 95, "severity": 7, "facility": 11, "version": 1, "timestamp": 1675212658399, "hostname": "host3.example.com", "body": "application8 has ex show more", "appName": "application8", "procid": "3312", "messageId": "ID7", "structuredData": {"SDID": "event2", "eventSource": "kernel", "iut": "9"}})
995	Wed, Feb 01 2023, 11:50:58	null	{"priority": 49, "severity": 1, "facility": 6, "version": 1, "timestamp": 1675212658399, "hostname": "host9.example.com", "body": "application4 has ex show more", "appName": "application4", "procid": "3312", "messageId": "ID8", "structuredData": {"SDID": "event3", "eventSource": "kernel", "iut": "9"}})
996	Wed, Feb 01 2023, 11:50:58	null	{"priority": 167, "severity": 7, "facility": 20, "version": 1, "timestamp": 1675212658399, "hostname": "host8.example.com", "body": "application8 has show more", "appName": "application8", "procid": "3312", "messageId": "ID9", "structuredData": {"SDID": "event4", "eventSource": "kernel", "iut": "9"}})
997	Wed, Feb 01 2023, 11:50:58	null	{"priority": 178, "severity": 2, "facility": 22, "version": 1, "timestamp": 1675212658399, "hostname": "host3.example.com", "body": "application4 has show more", "appName": "application4", "procid": "3312", "messageId": "ID10", "structuredData": {"SDID": "event5", "eventSource": "kernel", "iut": "9"}})
1008	Wed, Feb 01 2023, 11:50:58	null	{"priority": 105, "severity": 1, "facility": 1, "version": 1, "timestamp": 1675212658399, "hostname": "host10.example.com", "body": "application10 has show more", "appName": "application10", "procid": "3312", "messageId": "ID11", "structuredData": {"SDID": "event6", "eventSource": "kernel", "iut": "9"}})

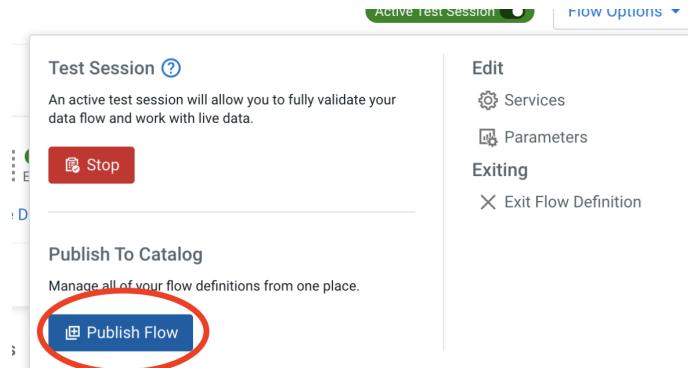
6. Version Control the flow into the DataFlow Catalog

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.

Flow Name	Flow Design Workspace	Test Session	Runtime Version	Last Updated	Published On
araudo-syslog-kafka-flow	r2d2-demo-aws	Active	1.18.0.2.3.7.1-1	28 minutes ago by araujo	-
bv-uat-aws-s3-kafka-hbase	pp-uat-aw-env	Not Defined	N/A	13 hours ago by vbandari	-
NHTSA-Test1	nirchi-env	Inactive	1.18.0.2.3.7.1-1	5 days ago by steven.mattison	5 days ago
s	pp-uat-aw-env	Not Defined	N/A	11 hours ago by vbandari	-

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**.

Flow Name	Test Session	Runtime Version	Created By	Last Updated
araudo-syslog-kafka-flow	Active	1.18.0.2.3.7.1-1	araujo	32 minutes ago by araujo

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:

Version	Deployments
1	0

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**

Deployment Name

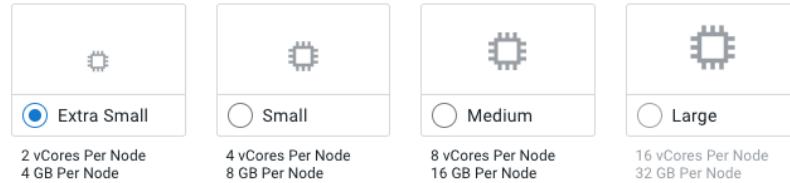
 Deployment name is valid

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**

Sizing & Scaling

Select the NiFi node size and the number of nodes provisioned for your flow.

NiFi Node Sizing [?](#)



Number of NiFi Nodes

Auto Scaling [?](#)

Enabled



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

[+ Add New KPI](#)

- 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**

Add new KPI

X

Details

KPI Scope ? Connection Name ?

Connection failure_WriteToKafka

Metric to Track ?

Bytes Queued

METRIC DESCRIPTION:
Size of content queued in connection

Alerts

Trigger alert when metric is greater than 10 MBytes

Trigger alert when metric is less than Value MBytes

Alert will be triggered when metric is outside the boundary(s) for 30 Seconds

Cancel Add

This screenshot shows the configuration interface for a new KPI. The 'Details' section includes 'KPI Scope' set to 'Connection' and 'Connection Name' set to 'failure_WriteToKafka'. The 'Metric to Track' is 'Bytes Queued'. The 'METRIC DESCRIPTION' is 'Size of content queued in connection'. In the 'Alerts' section, the 'Trigger alert when metric is greater than' option is checked, with a value of '10' and a unit of 'MBytes'. There is also an unchecked option for 'Trigger alert when metric is less than' with a dropdown for 'Value' and 'MBytes'. A note indicates that alerts are triggered when metrics are outside boundaries for a specified duration. The 'Seconds' dropdown is set to '30'. At the bottom are 'Cancel' and 'Add' buttons.

- 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**

Add new KPI

Details

KPI Scope [?](#) Connection Name [?](#)

Connection filtered_event_Filter-WriteToKafka

Metric to Track [?](#)

Bytes Queued

METRIC DESCRIPTION:
Size of content queued in connection

Alerts

Trigger alert when metric is greater than KBytes

Trigger alert when metric is less than Value

Alert will be triggered when metric is outside the boundary(s) for Seconds

[Cancel](#) [Add](#)

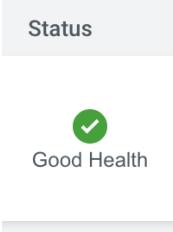
10. Review the KPIs and click **Next**.

<p>Connection: failure_WriteToKafka</p> <p>METRIC TO TRACK</p> <p>Bytes Queued</p> <p>ALERT SET</p> <p>Notify if greater than 10 MBytes, for at least 30 seconds.</p>	
<p>Connection: filtered_event_Filter-WriteToKafka</p> <p>METRIC TO TRACK</p> <p>Bytes Queued</p> <p>ALERT SET</p> <p>Notify if greater than 10 KBytes, for at least 30 seconds.</p>	

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 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** (_topics) and filter the topic **<userid>-syslog-avro**, which was created by the NiFi flow in Module 1

The screenshot shows the SMM interface for managing Kafka topics. At the top, there's a summary bar with metrics like Bytes In (1 MB), Bytes Out (904 KB), and Fetched Per Sec (2). Below this is a search bar with the term "hlanka". The main area displays a table of topics, with one row highlighted for "hlanka-syslog". The table columns include NAME, DATA IN, DATA OUT, MESSAGES IN, CONSUMER GROUPS, and CURRENT LOG SIZE. The "hlanka-syslog" row shows 0B for DATA IN, 11 KB for DATA OUT, 0 for MESSAGES IN, 0 for CONSUMER GROUPS, and 1 GB for CURRENT LOG SIZE.

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

This screenshot shows the same SMM interface as above, but with a red box highlighting the "profile" icon next to the "hlanka-syslog" topic row. The "profile" icon is a small blue square with a white icon inside. A red arrow points from the text "profile" at the bottom right towards this icon. The rest of the interface is identical to the previous screenshot, showing the topic details and cluster statistics.

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

The screenshot shows the Kafka Topic Data Explorer interface for the 'hanka-syslog' topic. The 'DATA EXPLORER' tab is active. Key information displayed includes:

- Producers (3):** Shows three producers: 'producer-17a9e0ef-9dbc-41d...', 'producer-4b193a70-966b-420...', and 'producer-a35a647-2f64-4ea...'. The first one is highlighted.
- MESSAGES:** Leader partition 0 has bytes in: 0B, bytes out: 11 KB, and log-size: 1 GB.
- Consumer Groups (0):** No consumer groups are listed.
- Messages Consumed:** No data.
- End-to-end Latency:** No data.
- Summary:**
 - Number of Replicas: 1
 - Number of Partitions: 1
 - Total number of Brokers for Topic: 1
 - Preferred Replication %: 100
 - Under Replicated %: 0
- Data Gauges:** Data In Gauge (0), Data Out Gauge (11184), and Messages In Gauge (0).

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:

The screenshot shows the Kafka Topic Data Explorer interface for the 'test-syslog' topic. The 'DATA EXPLORER' tab is active. A red box highlights the 'DESERIALIZER' dropdown, which is currently set to 'Values: String'. Below the table, there is a 'RECORD LIMIT' input field set to 15. The table displays log entries:

Offset	Timestamp	Key	Value
0	Tue, Jan 24 2023, 12:29:12	null	application10 has started successfully application104605ID18 37 application 5
1	Tue, Jan 24 2023, 12:29:12	null	application4 has stopped unexpectedly application49736ID18 9!python 9
2	Tue, Jan 24 2023, 12:29:12	null	application7 has completed gracefully application7789ID18 9!kernel 9
3	Tue, Jan 24 2023, 12:29:12	null	application3 has completed gracefully application34359ID17 9!python 9
4	Tue, Jan 24 2023, 12:29:12	null	application2 has exited cleanly application25600ID22 54!kernel 7
5	Tue, Jan 24 2023, 12:29:12	null	application10 has started successfully application106444ID16 51 application 2
6	Tue, Jan 24 2023, 12:29:12	null	application5 has started successfully application59720ID19 43!python 2
7	Tue, Jan 24 2023, 12:29:12	null	application1 has stopped unexpectedly application16477ID41 95 application 9
8	Tue, Jan 24 2023, 12:29:12	null	application8 has started successfully application81909ID49 46!kernel 9

Page navigation: 1 - 9 of 9

DESERIALIZER: Keys: String | ▾ Values: String | ▾

- DoubleDeserializer
- Bytes
BytesDeserializer
- ByteArray
ByteArrayDeserializer
- ByteBuffer
ByteBufferDeserializer
- Avro**
KafkaAvroDeserializer
- com.hortonworks.registries.schemaregistry.s...
Json deserializer
- JsonPrettyPrintDeserializer

1 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: Partition 0 | ▾ 0 | ▾ 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 RECORD LIMIT

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"}