2 Hours to Data Innovation - CDF

Preparation **4**

Module 0 - NiFi Overview **6**

1. Anatomy of a Dataflow **6**

1.1 Processor 7

1.2 Controller Services 8

1.3 FlowFile 9

1.4 Process Group 9

1.5 Parameters 9

1.6 Relationships 10

Module 1 - Data Distribution with Cloudera DataFlow **13**

1. Overview 13

2. Create a schema in Schema Registry 13

3. Design the flow in the Flow Designer 15

3.1 Create a new flow draft 15

3.2 Configure parameters for the flow 16

3.3 Create Controller Services 19

3.4 Design the flow 21

3.5 Naming the queues 23

4. Interactively test the flow 25

5. Version Control the flow into the DataFlow Catalog 30

6. Deploy the flow in Production 32

7. Deploy the Flow using CLI 38

Module 2 - Streaming Analytics with Cloudera Stream Processing **40**

1. Overview 40

2. Inspect Kafka topics in SMM 40

3. Create a Project in SSB and configure GitHub repo for it 43

4. Register a Kafka Data Source 48

5. Register a Schema Registry catalog 50

6. Register a Kudu catalog 53

7. Create a Kafka Table manually 55

8. Unlock your keytab 58

9. Create Streaming SQL jobs 59

9.1. Job 1: Selecting records from a Kafka virtual table 59

9.2. Job 2: Selecting records from a Schema Registry virtual table 64

9.3. Job 3: Using a windowing function 65

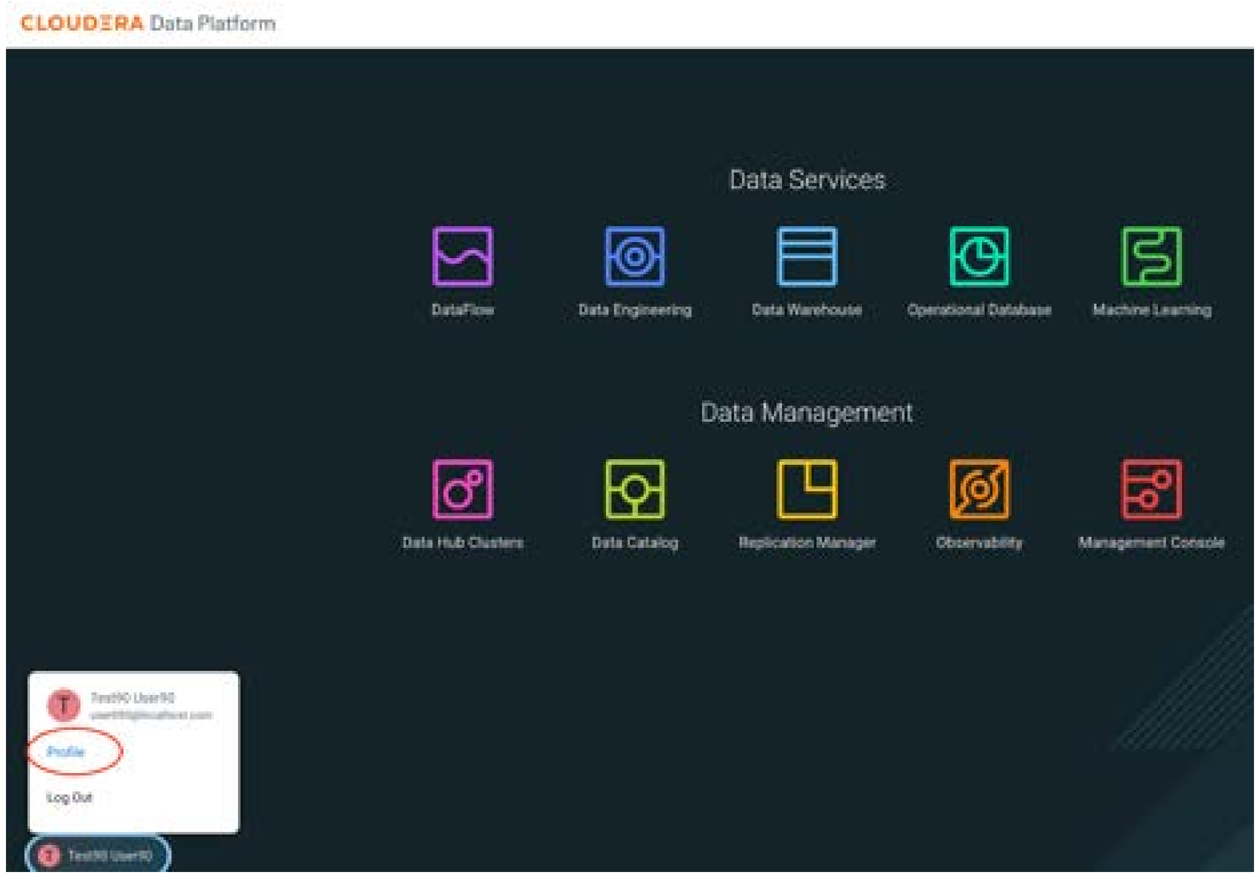
9.4. Job 4: Enriching streaming data with data-at-rest data sources 66

10. Create a Production job sending data to Kafka 66
11. Versioning your project changes into GitHub 72
12. Troubleshooting Flink/SSB jobs 73
13. Common issues 76

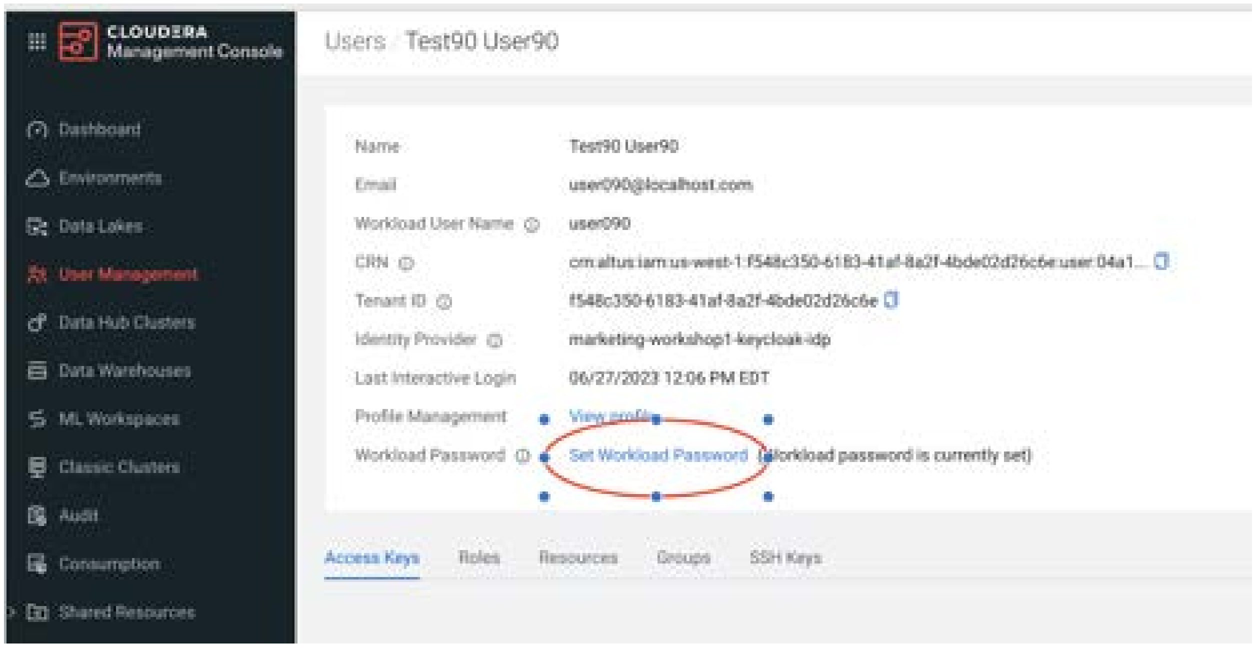
Authors	Email	Date Updated	Contributions
Richard Walden, Hima Lanka, Nagaraj Jayakumar, Christopher Perro, André Araújo	rwalden, hlanka, njayakumar, cperro, araujo @cloudera.com	TBD	Full document review and updates to prepare it for SKO FY24
Richard Walden, Hima Lanka, Steven Matison	rwalden, hlanka, steven.matision @cloudera.com	June 2023	New Module 0 (High Level overview of NiFi)
Steven Matison	steven.matison@cloudera.com	March 2024	Cleanup/Modernize

Preparation

Setting the workload password:
To access non-SSO interfaces, each user and machine user must set a workload password. After opening the CDP Control Plane, click on the username -> Profile on the bottom left corner as shown in the image.



This opens the user management page. Click on the “Set Workload Password” link as highlighted below and set the password to `G0yvxvdms5srhyKF`



Users / Test90 User90

Name	Test90 User90
Email	user090@localhost.com
Workload User Name	user090
CRN	crn:altus:iam:us-west-1:f548c350-6183-41af-8a2f-4bde02d26c6e:user:04a1...
Tenant ID	f548c350-6183-41af-8a2f-4bde02d26c6e
Identity Provider	marketing-workshop1-keycloak-idp
Last Interactive Login	06/27/2023 12:06 PM EDT
Profile Management	View profile
Workload Password	Set Workload Password (Workload password is currently set)

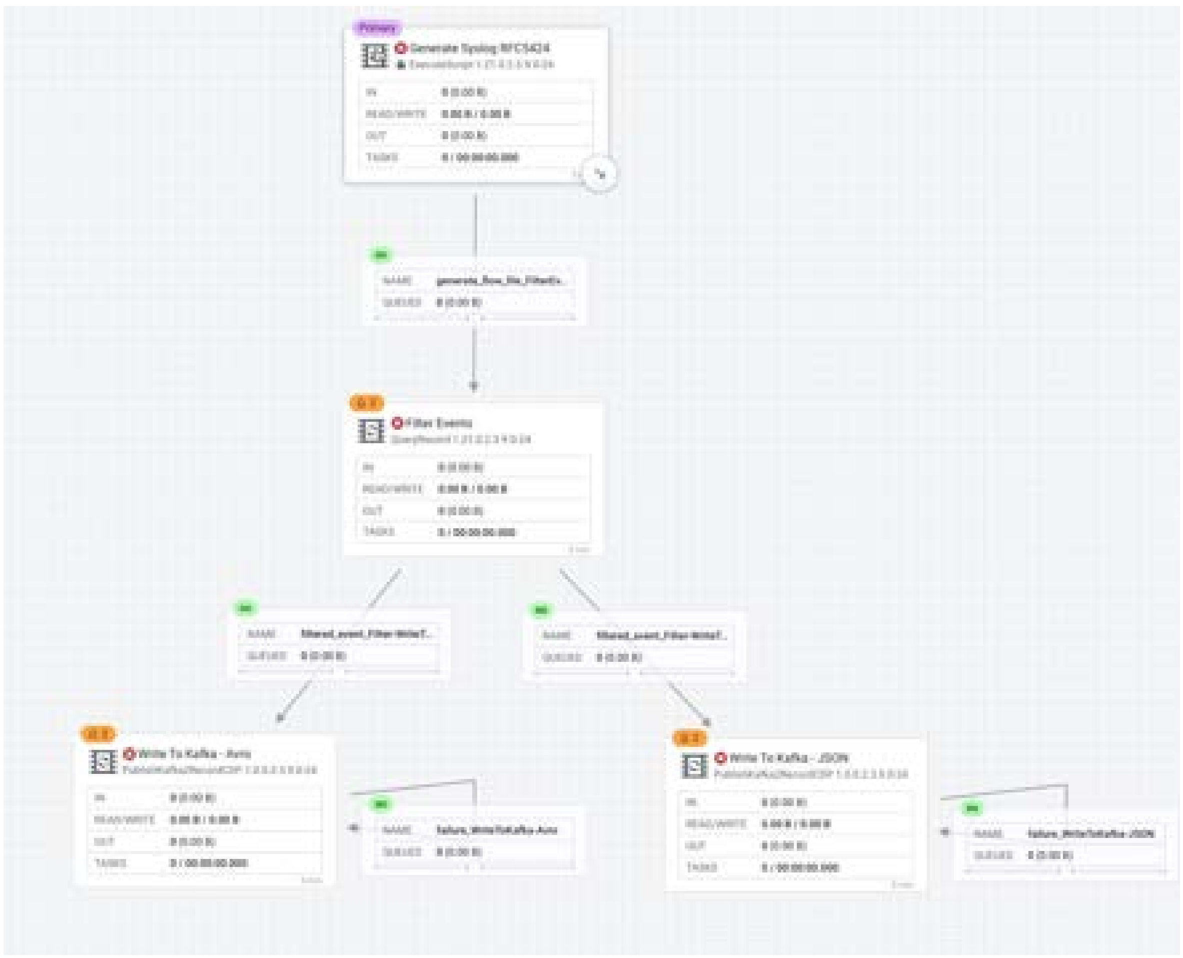
This is the password you will be using when configuring the data flow and SSB.

Module 0 - NiFi Overview

Apache NiFi is a dataflow system based on the concepts of flow-based programming. It supports powerful and scalable directed graphs of data routing, transformation, and system mediation logic. NiFi has a web-based user interface for design, control, feedback, and monitoring of dataflows. It is highly configurable along several dimensions of quality of service, such as loss-tolerant versus guaranteed delivery, low latency versus high throughput, and priority-based queuing. NiFi provides fine-grained data provenance for all data received, forked, joined, clone, modified, sent, and ultimately dropped upon reaching its configured end-state.

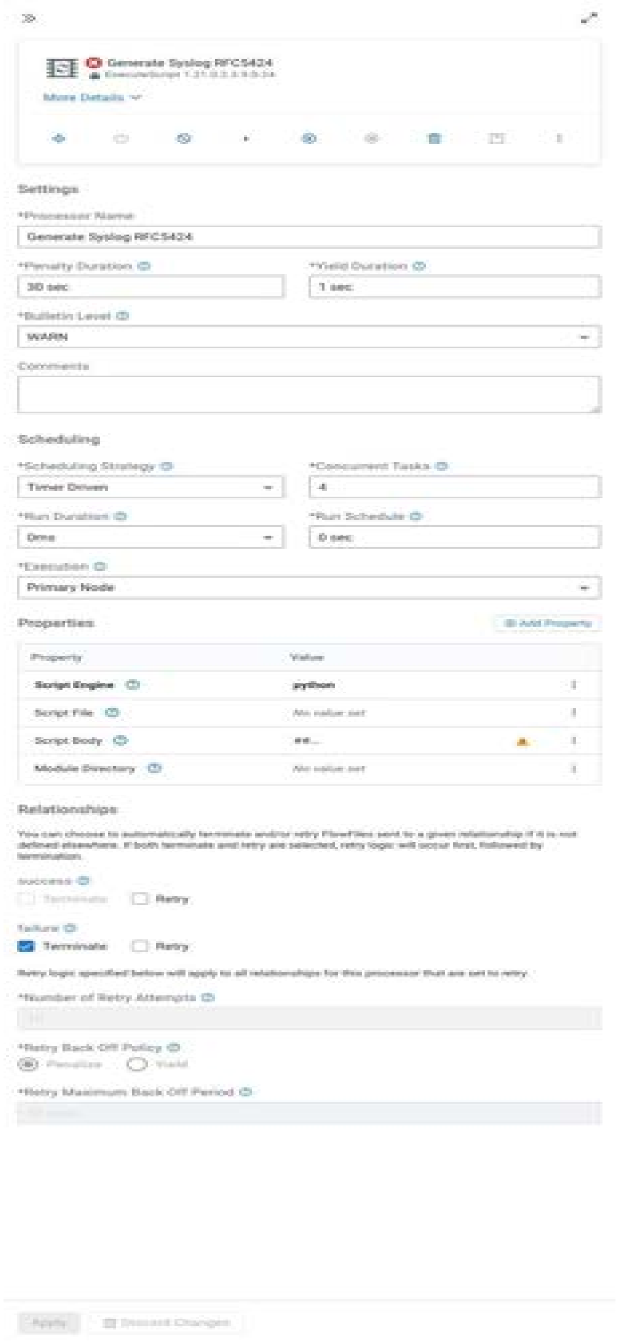
1. Anatomy of a Dataflow

A dataflow is a collection of processors that are connected together to perform a unit of work. At its basic level, it will get data from a source, do some processing of the data and then send that data to a destination (e.g. could be a file into S3, JSON data into Kafka, etc.). Let’s take a look at the different components of a dataflow.



1.1 Processor

The processor is the most common component of NiFi. Processors a small unit of work that has already been coded and tested for you. You will drag a processor onto the canvas and then be able to configure the processor, start, stop, view history, etc.



Generate Syslog RFC5424
ExecuteScript 1.21.0.2.3.9.0-34
More Details

Settings

*Processor Name
Generate Syslog RFC5424

*Penalty Duration
30 sec

*Yield Duration
1 sec

*Bulletin Level
WARN

Comments

Scheduling

*Scheduling Strategy
Timer Driven

*Concurrent Tasks
4

*Run Duration
0ms

*Run Schedule
0 sec

*Execution
Primary Node

Properties

Add Property

Property	Value
Script Engine	**python**
Script File	No value set
Script Body	##...
Module Directory	No value set

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
10

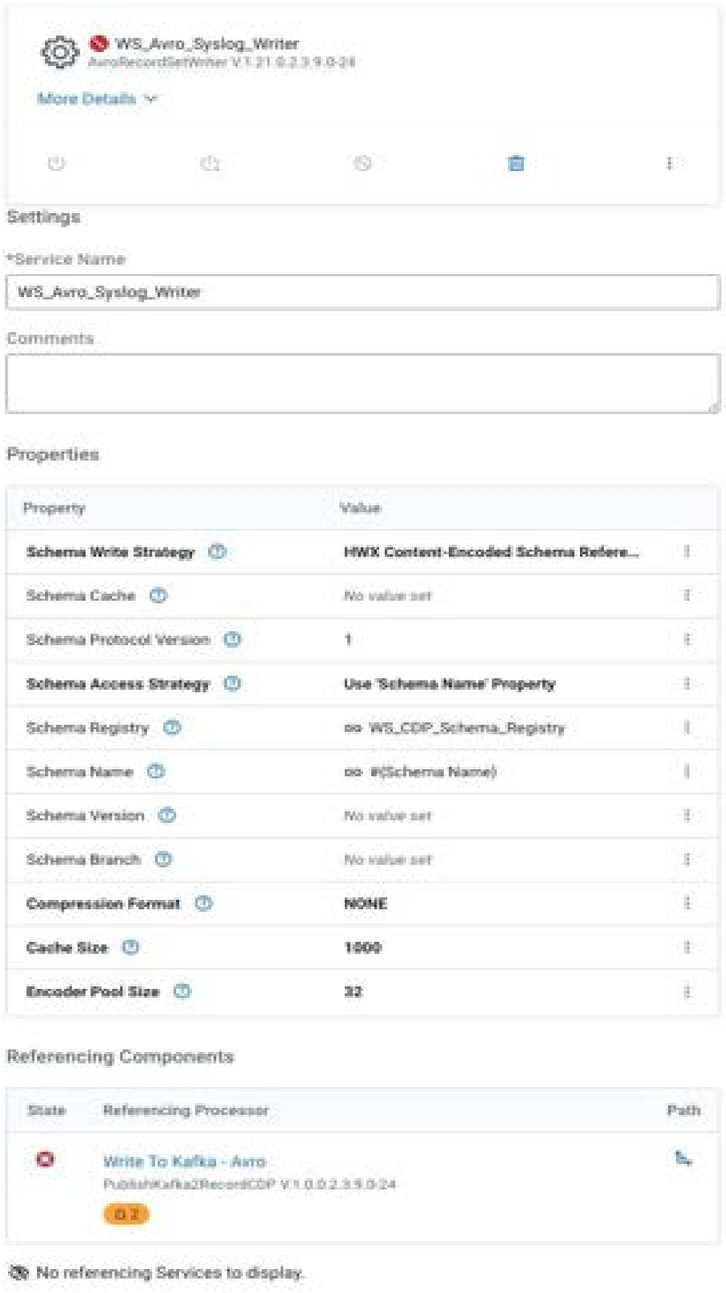
*Retry Back Off Policy
[✓ Penalize] Yield

*Retry Maximum Back Off Period
10 mins

Apply
Discard Changes

1.2 Controller Services

Controller Services are extension points that, after being added and configured in the User Interface, will start up when NiFi starts up and provide information for use by other components (such as processors or other controller services). A common Controller Service used by several components is the StandardSSLContextService. It provides the ability to configure keystore and/or truststore properties once and reuse that configuration throughout the application. 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.



WS_Avro_Syslog_Writer
AvroRecordSetWriter V.1.21.0.2.3.9.0-24

More Details

Settings

*Service Name

WS_Avro_Syslog_Writer

Comments

Properties

Property	Value
Schema Write Strategy	**HWX Content-Encoded Schema Refere...**
Schema Cache	No value set
Schema Protocol Version	1
Schema Access Strategy	**Use 'Schema Name' Property**
Schema Registry	WS_CDP_Schema_Registry
Schema Name	#{Schema Name}
Schema Version	No value set
Schema Branch	No value set
Compression Format	**NONE**
Cache Size	**1000**
Encoder Pool Size	**32**

Referencing Components

State	Referencing Processor	Path
	Write To Kafka - Avro PublishKafka2RecordCDP V.1.0.0.2.3.9.0-24 2	

No referencing Services to display.

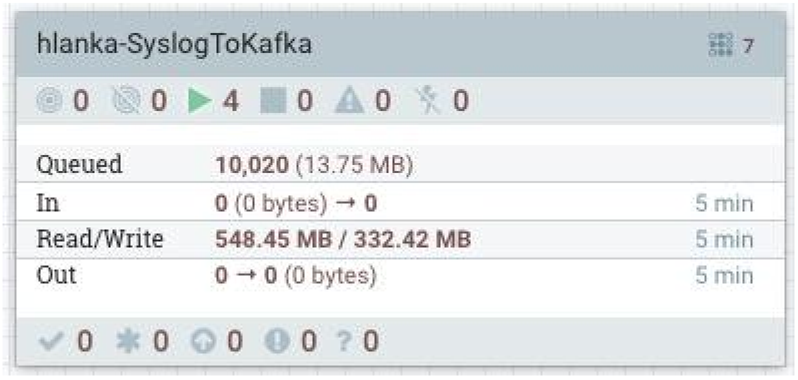
1.3 FlowFile

The FlowFile represents a single piece of data in NiFi. A FlowFile is made up of two components: FlowFile Attributes and FlowFile Content. Content is the data that is represented by the FlowFile. Attributes are characteristics that provide information or context about the data; they are made up of key-value pairs. All FlowFiles have the following Standard Attributes:

- uuid: A Universally Unique Identifier that distinguishes the FlowFile from other FlowFiles in the system.
- filename: A human-readable filename that may be used when storing the data to disk or in an external service
- path: A hierarchically structured value that can be used when storing data to disk or an external service so that the data is not stored in a single directory

1.4 Process Group

When a dataflow becomes complex, it often is beneficial to reason about the dataflow at a higher, more abstract level. NiFi allows multiple components, such as Processors, to be grouped together into a Process Group. The NiFi User Interface then makes it easy for a user to connect together multiple Process Groups into a logical dataflow, as well as allowing the user to enter a Process Group in order to see and manipulate the components within the Process Group.



hlanka-SyslogToKafka 7

0 0 ▶ 4 ■ 0 ⚠ 0 0

Queued	10,020 (13.75 MB)	
In	0 (0 bytes) → 0	5 min
Read/Write	548.45 MB / 332.42 MB	5 min
Out	0 → 0 (0 bytes)	5 min

✓ 0 ✱ 0 0 0 ? 0

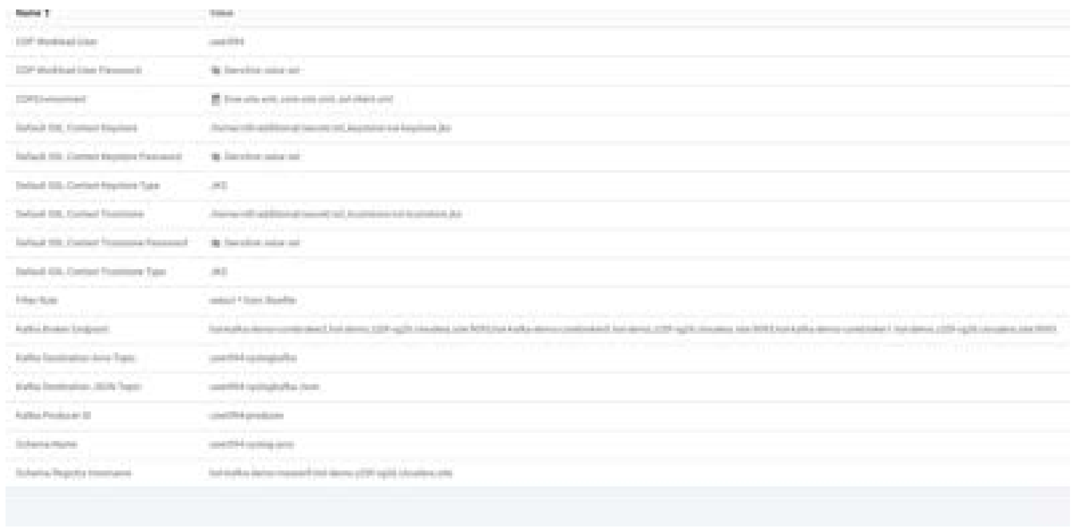
1.5 Parameters

The values of properties in the flow, including sensitive properties, can be parameterized using Parameters. Parameters are created and configured within the NiFi UI. Any property can be configured to reference a Parameter with the following conditions:

- A sensitive property can only reference a Sensitive Parameter
- A non-sensitive property can only reference a Non-Sensitive Parameter

- Properties that reference Controller Services can not use Parameters
- Parameters cannot be referenced in Reporting Tasks or in Management Controller Services

Parameters are grouped together into a Parameter Context.



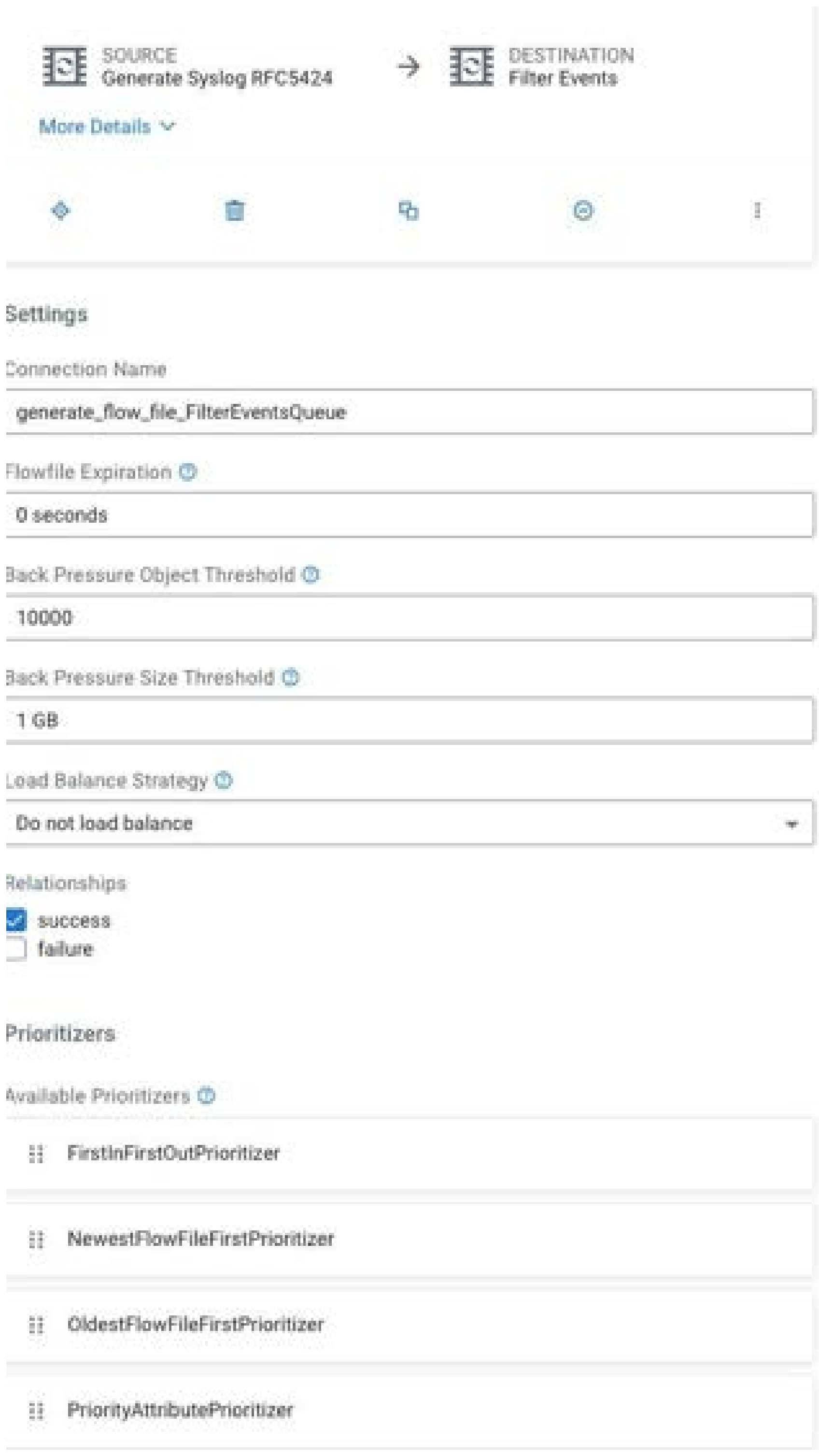
Name ↑	Value
CDP Workload User	[illegible]
CDP Workload User Password	Sensitive value set
CDPEnvironment	[illegible]
Default SSL Context Keystore	[illegible]
Default SSL Context Keystore Password	Sensitive value set
Default SSL Context Keystore Type	JKS
Default SSL Context Truststore	[illegible]
Default SSL Context Truststore Password	Sensitive value set
Default SSL Context Truststore Type	JKS
Filter Rule	select * from flowfile
Kafka Broker Endpoint	[illegible]
Kafka Destination Avro Topic	[illegible]
Kafka Destination JSON Topic	[illegible]
Kafka Producer ID	[illegible]
Schema Name	[illegible]
Schema Registry Hostname	[illegible]

1.6 Relationships

When designing a flow, you need to connect one processor to the next. When you do that, you connect them via relationships. At a minimum, a processor has 2 relationships, success and failure. Some processors have more static relationships, others have dynamically created relationships. These connections between two processors serve a very valuable service and are also configurable as well. These configurable items include:

- FlowFile expiration: Is a concept by which data that cannot be processed in a timely fashion can be automatically removed from the flow
- Back Pressure: This option dictates how many flowfiles can exist in a connection before a processor is no longer scheduled to run. There are two different properties that can be set:
    - Back pressure object threshold - The number of flow files a connection can hold. The default is 10000
    - Back pressure data size threshold - The amount of data, in size, that the connection can hold before the component is no longer scheduled to run. The default is 1 GB.
- Load Balancing - There are two components to Load Balancing:
    - Load Balancing Strategy - This is how flowfiles are distributed across nodes of a cluster. These strategies are:
        - Do not load balance: Do not load balance FlowFiles between nodes in the cluster. This is the default.

    - Partition by attribute: Determines which node to send a given FlowFile to based on the value of a user-specified FlowFile Attribute. All FlowFiles that have the same value for the Attribute will be sent to the same node in the cluster. If the destination node is disconnected from the cluster or if unable to communicate, the data does not fail over to another node. The data will queue, waiting for the node to be available again. Additionally, if a node joins or leaves the cluster necessitating a rebalance of the data, consistent hashing is applied to avoid having to redistribute all of the data.
    - Round robin: FlowFiles will be distributed to nodes in the cluster in a round-robin fashion. If a node is disconnected from the cluster or if unable to communicate with a node, the data that is queued for that node will be automatically redistributed to another node(s). If a node is not able to receive the data as fast other nodes in the cluster, the node may also be skipped for one or more iterations in order to maximize throughput of data distribution across the cluster.
    - Single node: All FlowFiles will be sent to a single node in the cluster. Which node they are sent to is not configurable. If the node is disconnected from the cluster or if unable to communicate with the node, the data that is queued for that node will remain queued until the node is available again.
  - Load Balancing Compression - This defines if flowfile data is compressed when transferring data between nodes on a cluster. There are 3 types:
    - Do not compress: FlowFiles will not be compressed. This is the default.
    - Compress attributes only: FlowFile attributes will be compressed, but FlowFile contents will not.
    - Compress attributes and content: FlowFile attributes and contents will be compressed.



Settings

Connection Name: generate_flow_file_FilterEventsQueue

Flowfile Expiration: 0 seconds

Back Pressure Object Threshold: 10000

Back Pressure Size Threshold: 1 GB

Load Balance Strategy: Do not load balance

Relationships

- [✓] success
- [ ] failure

Prioritizers

Available Prioritizers

- FirstInFirstOutPrioritizer
- NewestFlowFileFirstPrioritizer
- OldestFlowFileFirstPrioritizer
- PriorityAttributePrioritizer

Module 1 - Data Distribution with Cloudera DataFlow

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.

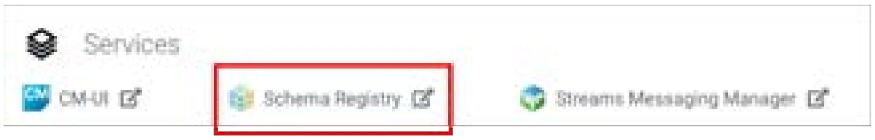
You'll also learn flow design best practices like:

- 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:

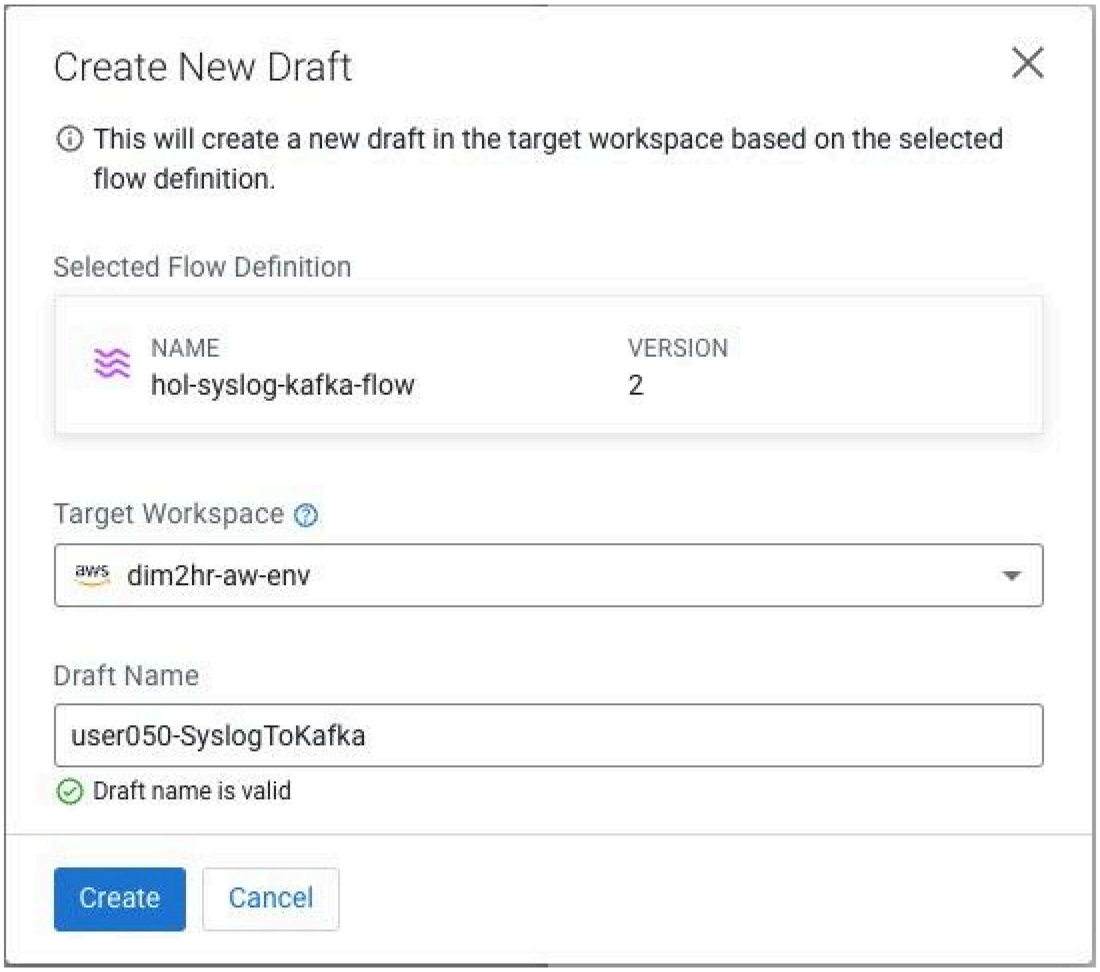
```
{
  "name": "syslog",
  "type": "record",
  "namespace": "com.cloudera",
  "fields": [
    { "name": "priority", "type": "int" },
    { "name": "severity", "type": "int" },
    { "name": "facility", "type": "int" },
    { "name": "version", "type": "int" },
    { "name": "timestamp", "type": "long" },
    { "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. Design the flow in the Flow Designer

3.1 Create a new flow draft

1. In Cloudera DataFlow, open the catalog and find the flow named HoL-SyslogToKafka and select create a new draft. Then give the draft a new name: <user>-SyslogToKafka



Create New Draft

This will create a new draft in the target workspace based on the selected flow definition.

Selected Flow Definition

NAME	VERSION
hol-syslog-kafka-flow	2

Target Workspace: dim2hr-aw-env

Draft Name: user050-SyslogToKafka

Draft name is valid

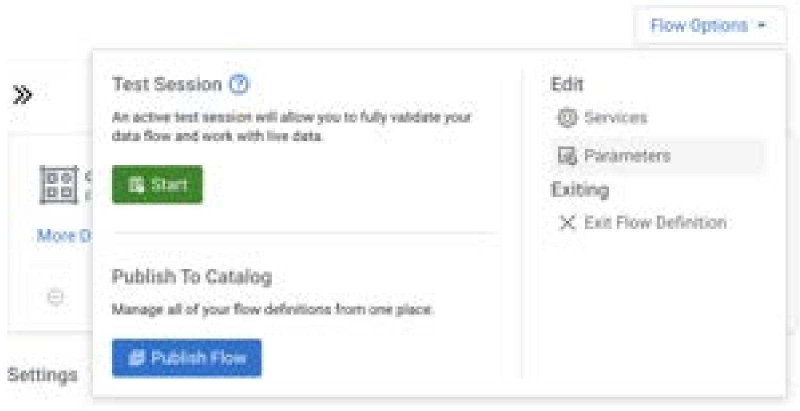
3.2 Configure parameters for the flow

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


Flow Options
Test Session
An active test session will allow you to fully validate your data flow and work with live data.
Start
Publish To Catalog
Manage all of your flow definitions from one place.
Publish Flow
Edit
Services
Parameters
Exiting
Exit Flow Definition
Settings

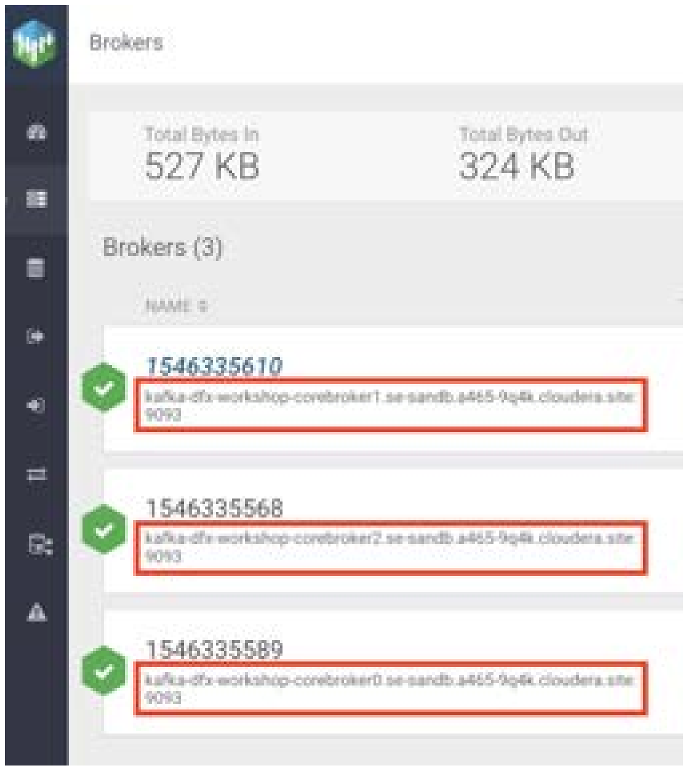
2. Update these parameters. Click on each parameter to be updated and enter the appropriate details:

Name	Description	Value
CDP Workload User	CDP Workload User	<Your own workload user name> e.g. user050
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:

- **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.



Brokers

Total Bytes In 527 KB

Total Bytes Out 324 KB

Brokers (3)

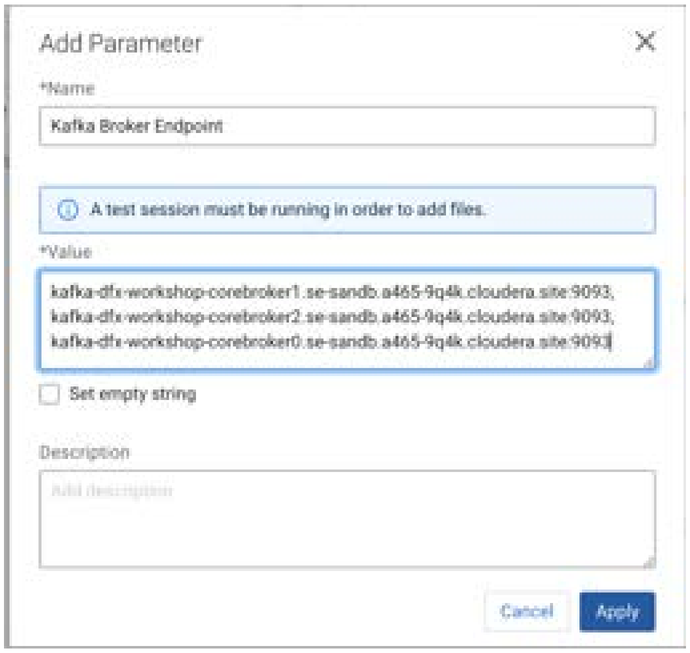
NAME

1546335610
kafka-dfx-workshop-corebroker1.se-sandb.a465-9q4k.cloudera.site:9093

1546335568
kafka-dfx-workshop-corebroker2.se-sandb.a465-9q4k.cloudera.site:9093

1546335589
kafka-dfx-workshop-corebroker0.se-sandb.a465-9q4k.cloudera.site:9093

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



Add Parameter

*Name

Kafka Broker Endpoint

A test session must be running in order to add files.

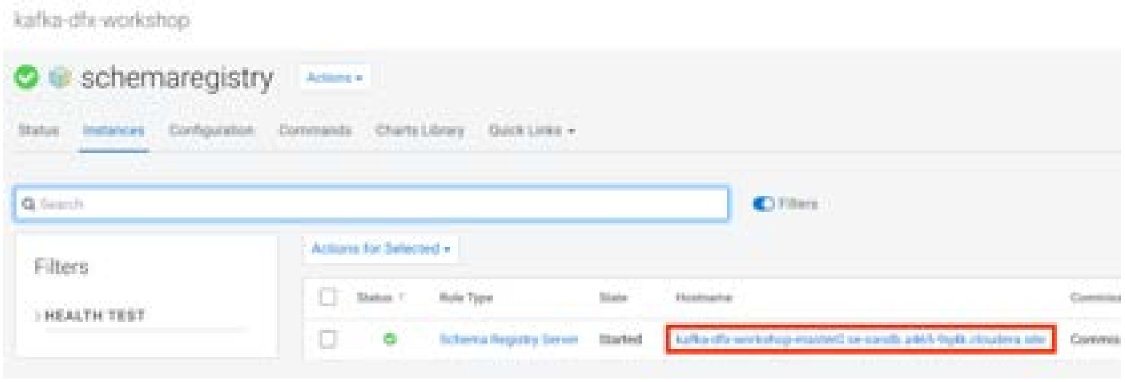
*Value

```
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
```

Set empty string

Description

- **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:

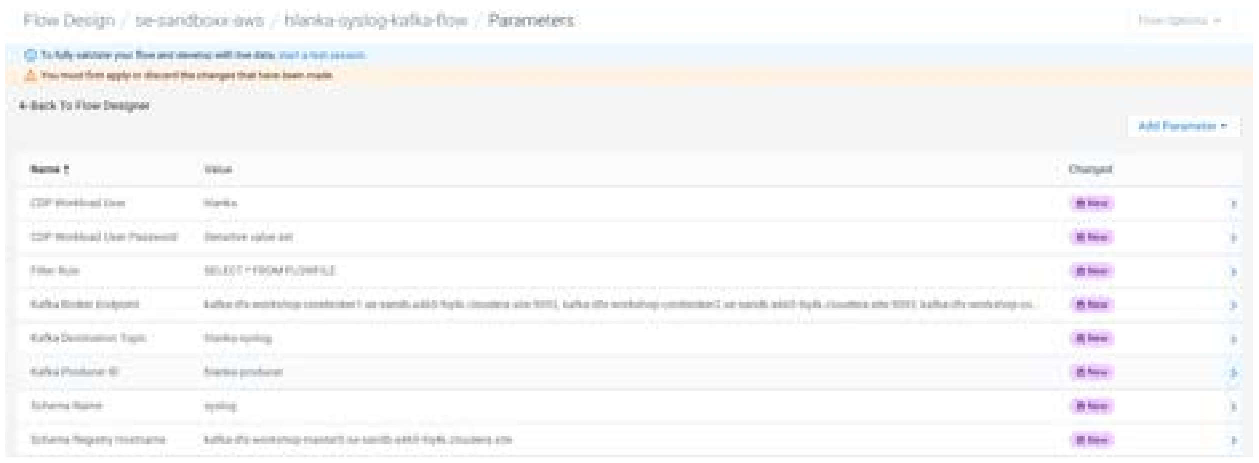


	Status	Role Type	State	Hostname	Commis
	●	Schema Registry Server	Started	kafka-dfx-workshop-master0.se-sandb.a465-9q4k.cloudera.site	Commis

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>

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



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

← Back To Flow Designer

Name	Value	Changed
CDP Workload User	hlanka	New
CDP Workload User Password	Sensitive value set	New
Filter Rule	SELECT * FROM FLOWFILE	New
Kafka Broker Endpoint	[illegible]	New
Kafka Destination Topic	hlanka-syslog	New
Kafka Producer ID	hlanka-producer	New
Schema Name	syslog	New
Schema Registry Hostname	[illegible]	New

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

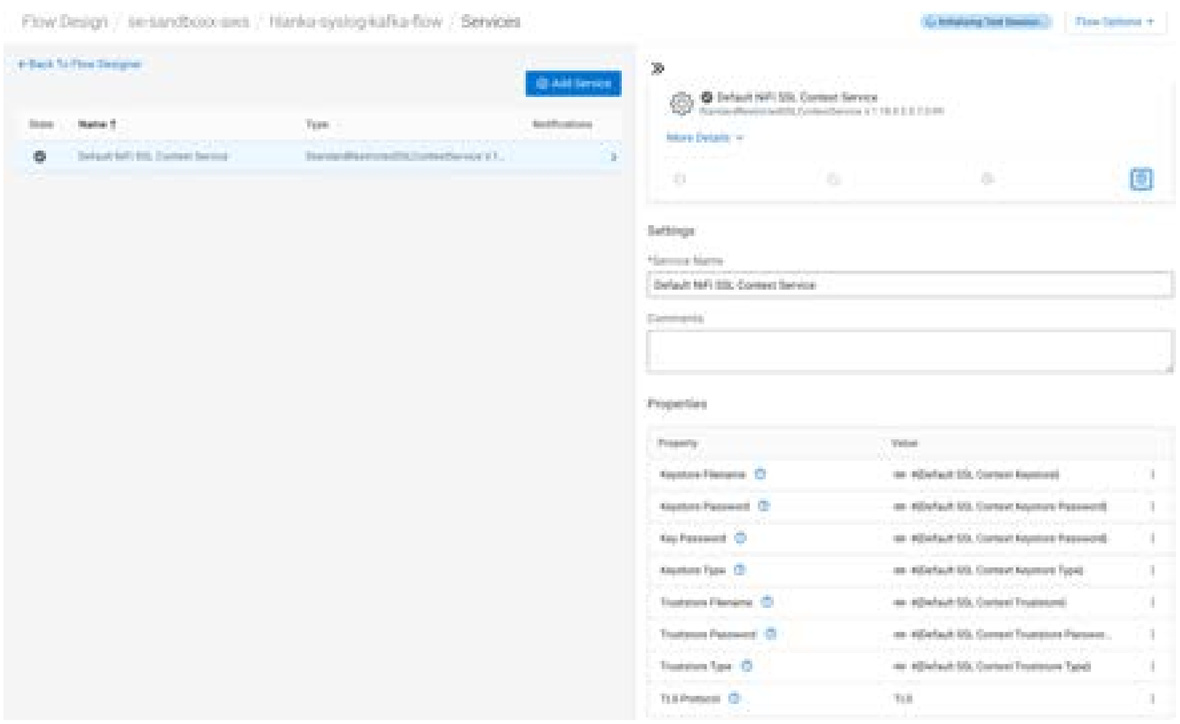
3.3 Create Controller Services

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.



Flow Design / se-sandbox-aws / Hanks-syslog-kafka-flow / Services

Initializing Test Session

Flow Options

← Back To Flow Designer

Add Service

State	Name	Type	Notifications
●	Default NiFi SSL Context Service	StandardRestrictedSSLContextService v.1...	

Default NiFi SSL Context Service

StandardRestrictedSSLContextService v.1.18.0.2.2.7.0-80

More Details

Settings

*Service Name

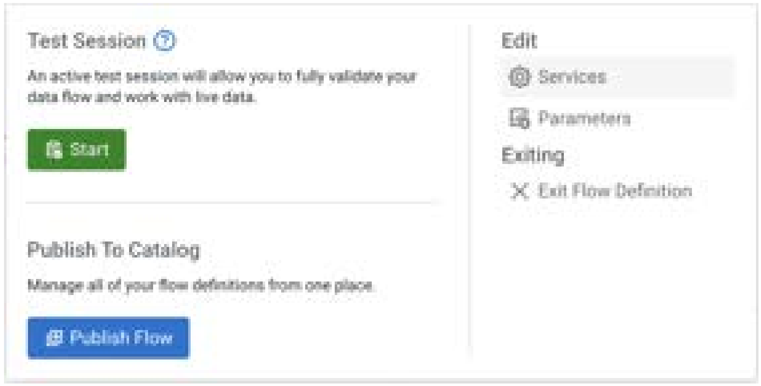
Default NiFi SSL Context Service

Comments

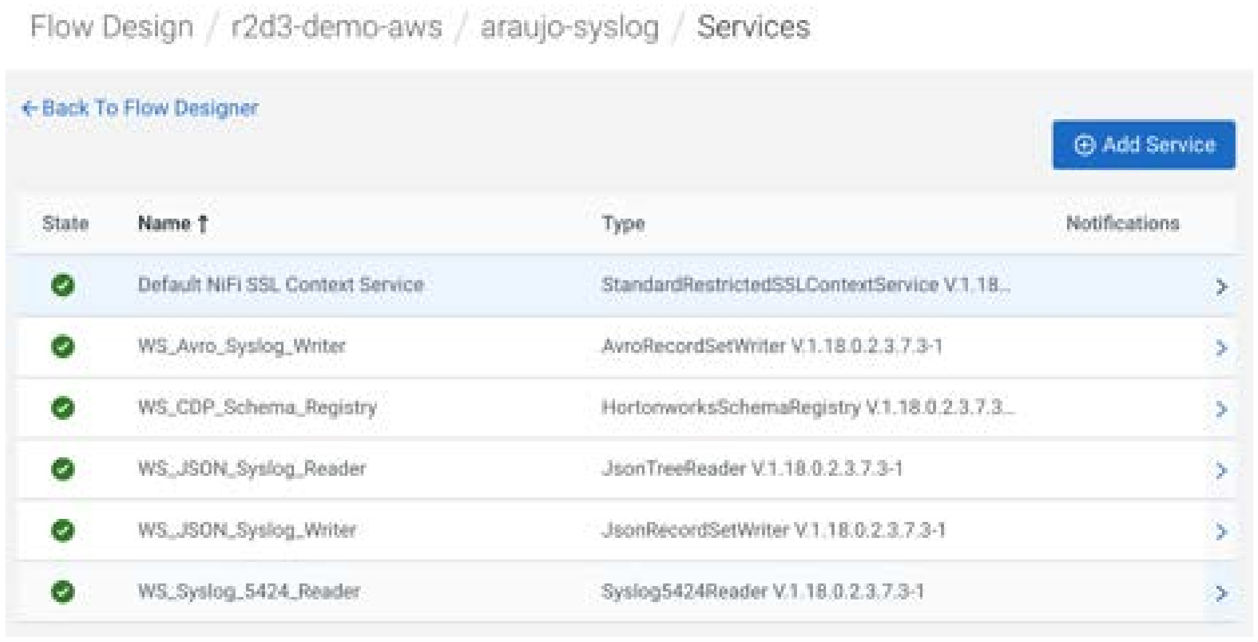
Properties

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 Passwo...
Truststore Type	#{Default SSL Context Truststore Type}
TLS Protocol	TLS

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


Test Session
An active test session will allow you to fully validate your data flow and work with live data.
Start
Publish To Catalog
Manage all of your flow definitions from one place.
Publish Flow
Edit
Services
Parameters
Exiting
Exit Flow Definition

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


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

← Back To Flow Designer

⊕ Add Service

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	

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

3.4 Design the flow

1. Open the Flow Design page

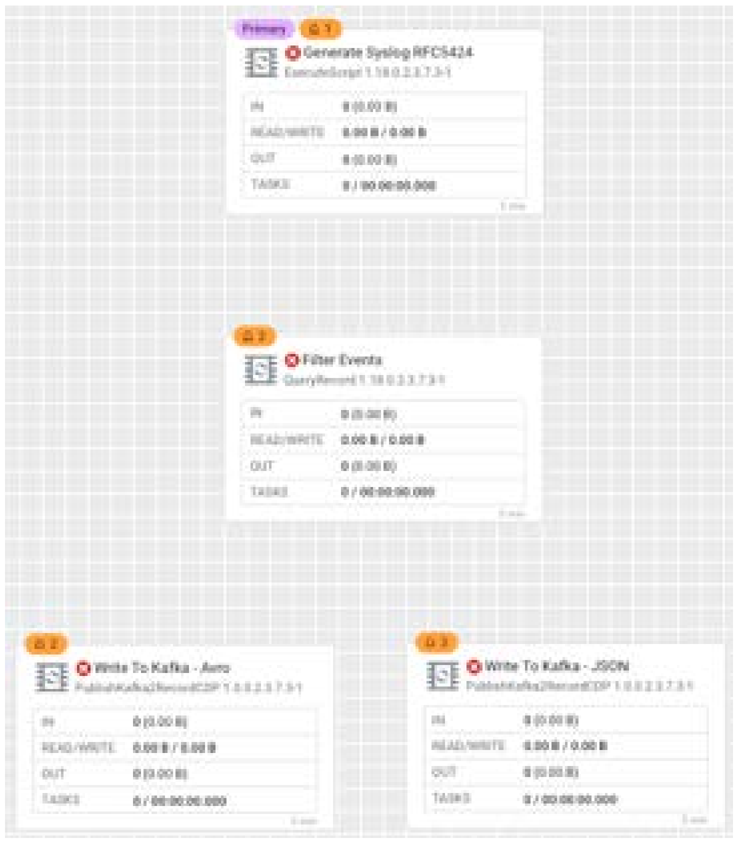
2. Create the **Write To Kafka - JSON** processor

    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}**

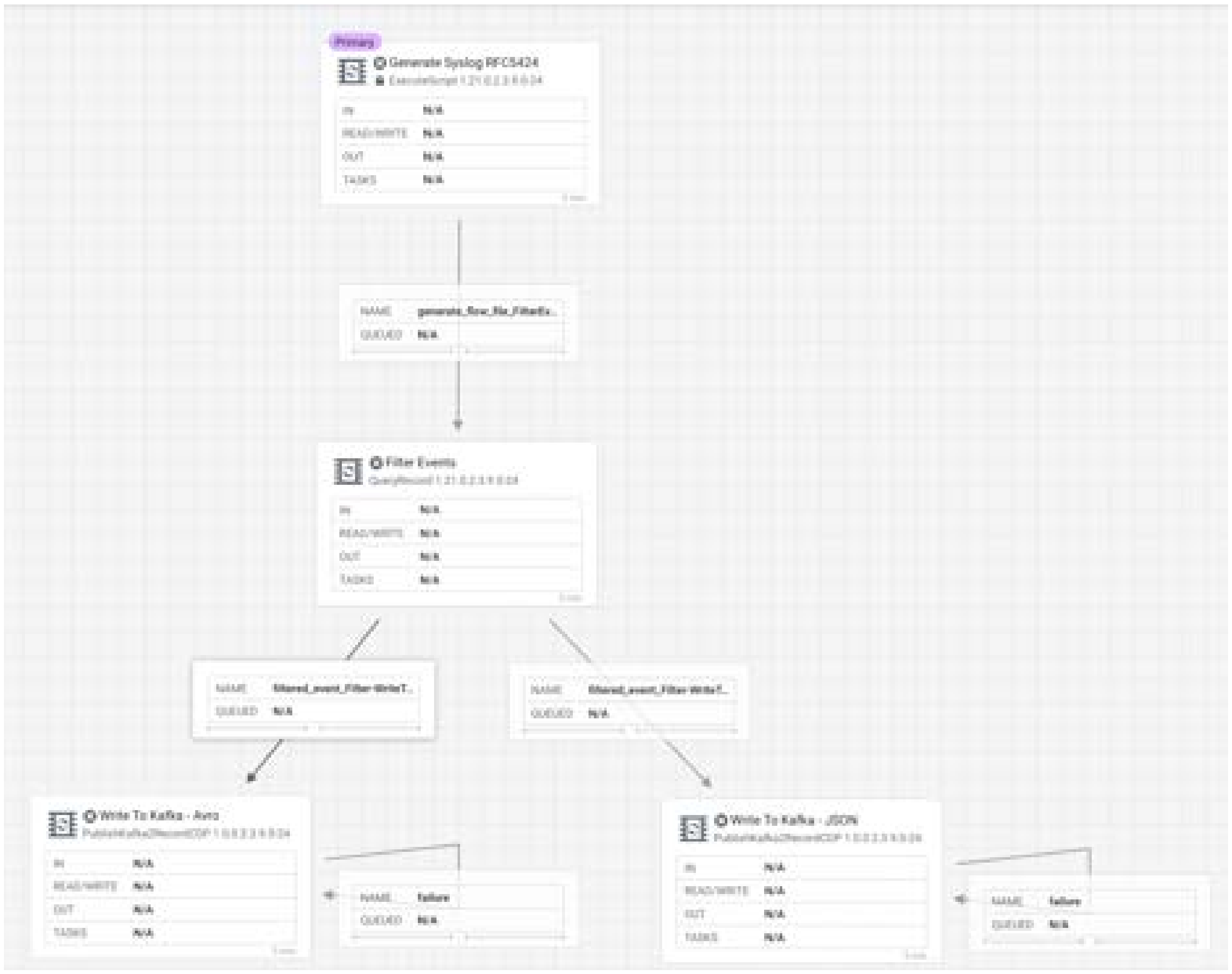
iii. Relationships: check **Terminate** for the **success** relationship

d. Click **Apply**


Primary
Generate Syslog RFC5424
ExecuteScript 1.18.0.2.3.7.3-1
Filter Events
QueryRecord 1.18.0.2.3.7.3-1
Write To Kafka - Avro
PublishKafka2RecordCDP 1.0.0.2.3.7.3-1
Write To Kafka - JSON
PublishKafka2RecordCDP 1.0.0.2.3.7.3-1
IN
READ/WRITE
OUT
TASKS

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

- Connect Processors **Filter Events** & **Write To Kafka - JSON** for Relationship **filtered_events**
- Connect **Write To Kafka - JSON** to itself to retry for **failures**



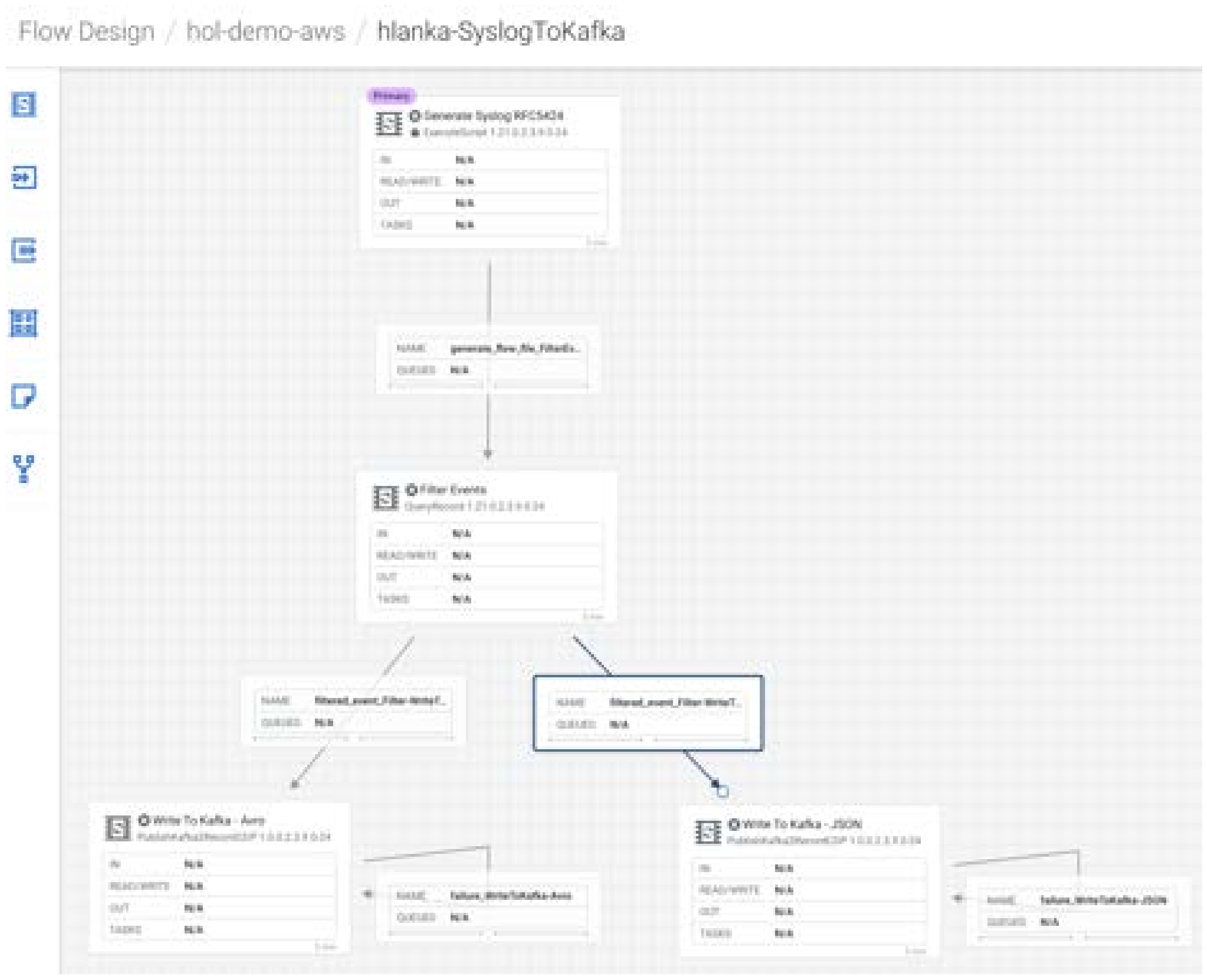
3.5 Naming the queues

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 **failure** retry queue from **Write To Kafka - JSON** to itself should be named **failure_WriteToKafka-JSON**

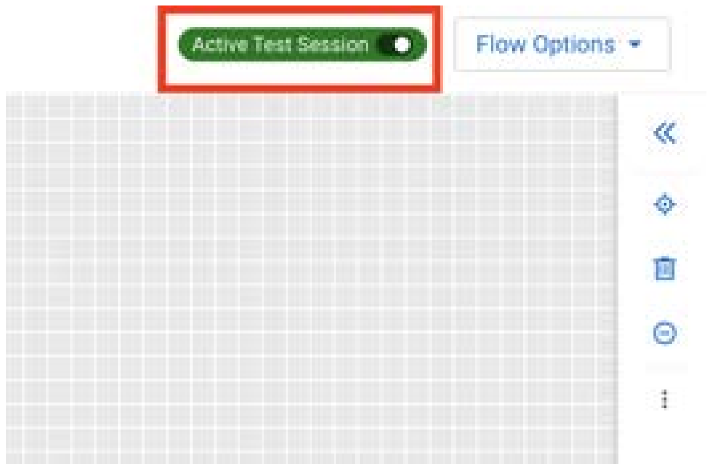

Flow Design / hol-demo-aws / hlanka-SyslogToKafka
Primary
Generate Syslog RFC5424
Filter Events
Write To Kafka - Avro
Write To Kafka - JSON
IN
READ/WRITE
OUT
TASKS
N/A
NAME
QUEUED
generate_flow_file_FilterEv...
filtered_event_Filter-WriteT...
failure_WriteToKafka-Avro
failure_WriteToKafka-JSON

4. 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, if you don't already have an active 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…**

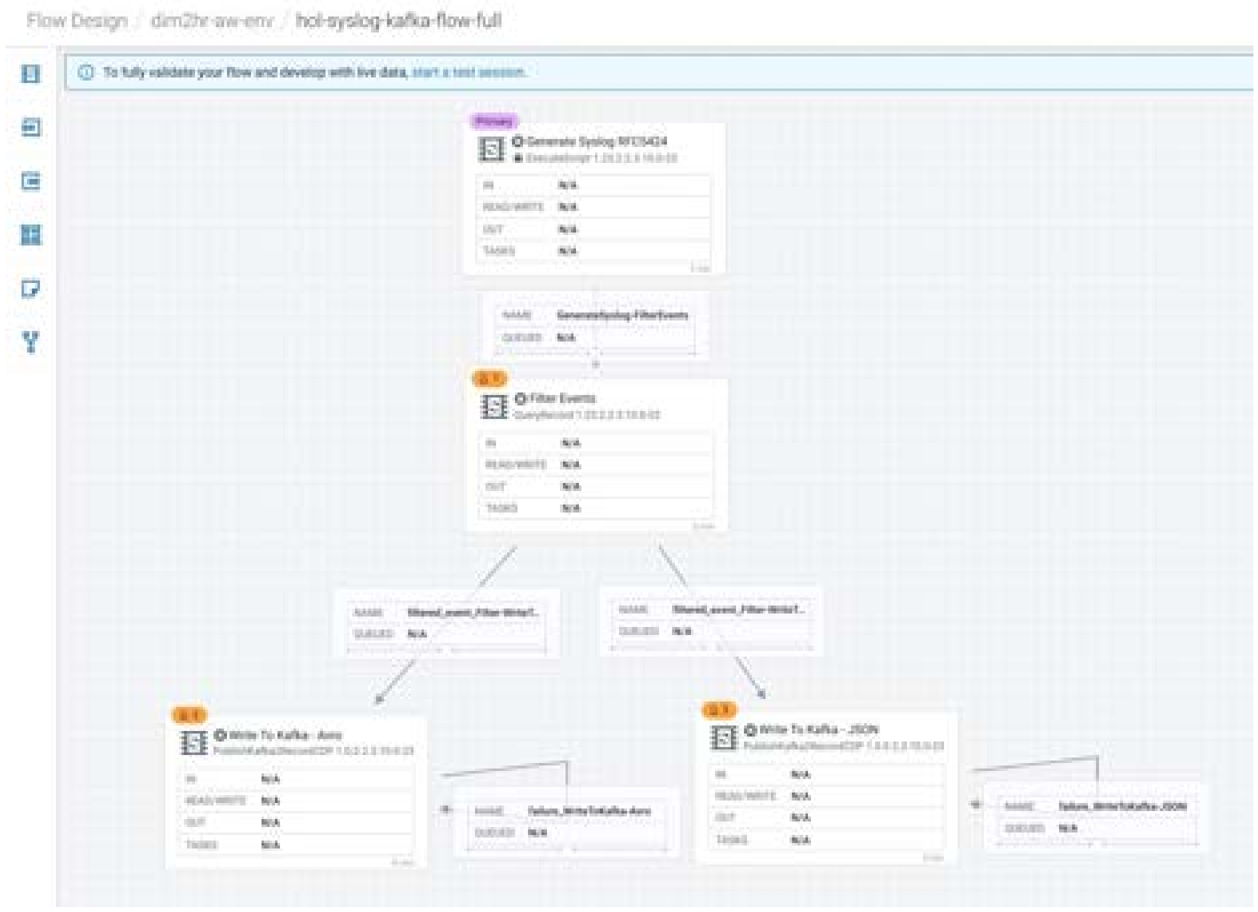
3. Wait for the status to change to **Active Test Session**.

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

5. Start all processors that are not running.

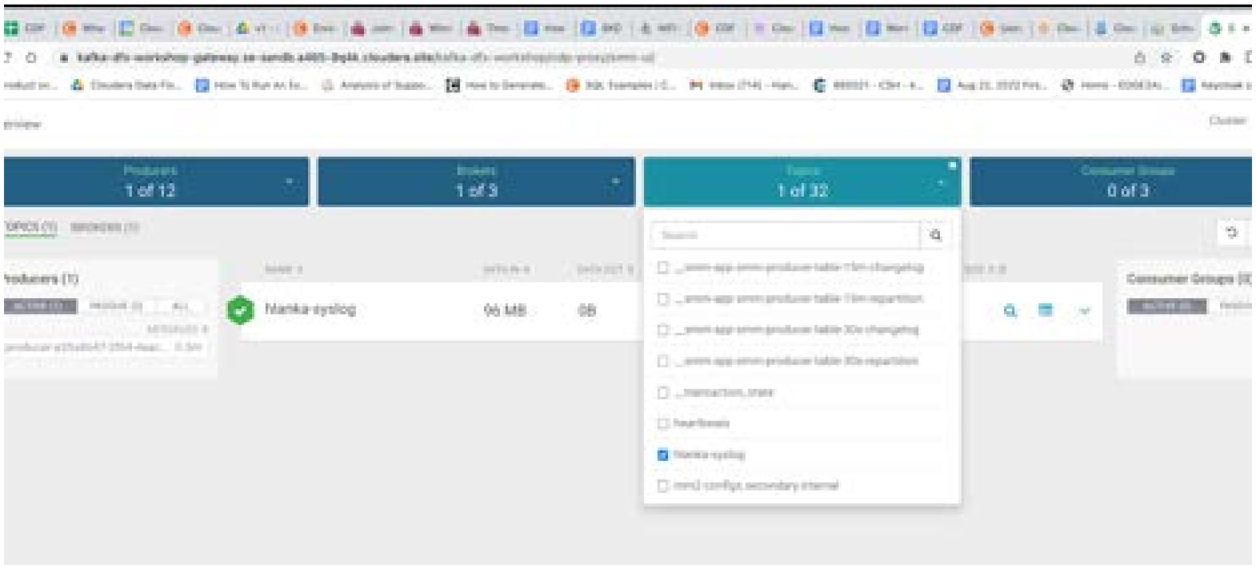
   Stopped processors are identified by the icon ✖. Right-click on a stopped processor and select **Start** to start it.

   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.

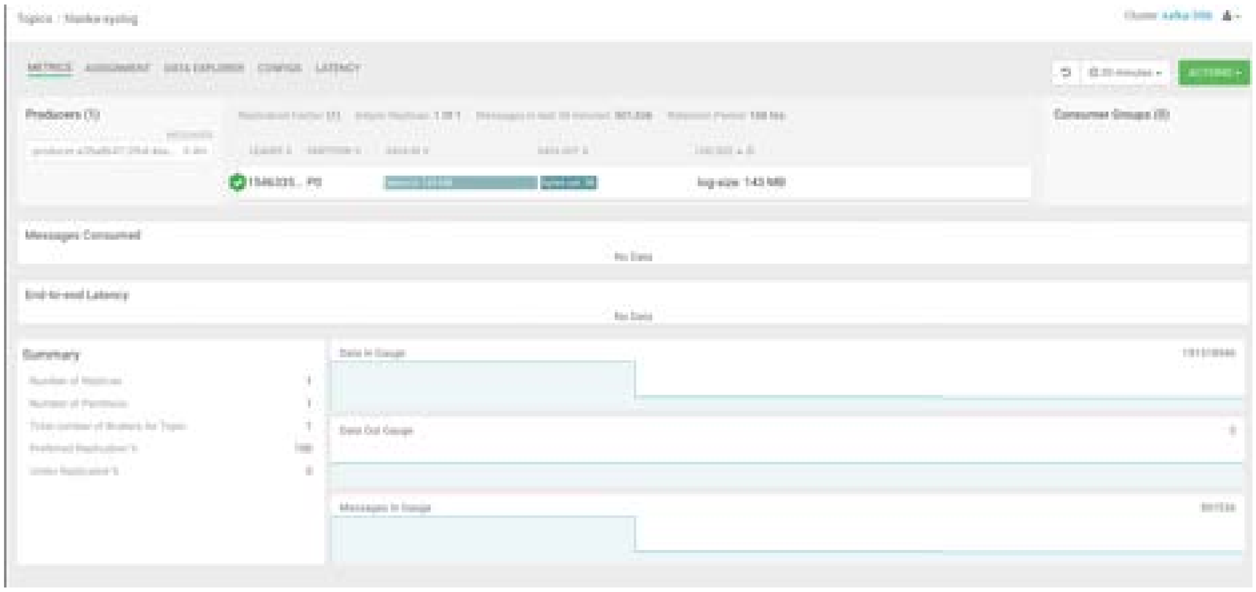

Flow Design / dim2hr-aw-env / hol-syslog-kafka-flow-full
To fully validate your flow and develop with live data, start a test session.
Primary
Generate Syslog RFC5424
IN N/A
READ/WRITE N/A
OUT N/A
TASKS N/A
NAME GenerateSyslog-FilterEvents
QUEUED N/A
Filter Events
QueryRecord
NAME filtered_event_Filter-WriteT...
QUEUED N/A
Write To Kafka - Avro
PublishKafka2RecordCDP
NAME failure_WriteToKafka-Avro
Write To Kafka - JSON
NAME failure_WriteToKafka-JSON

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

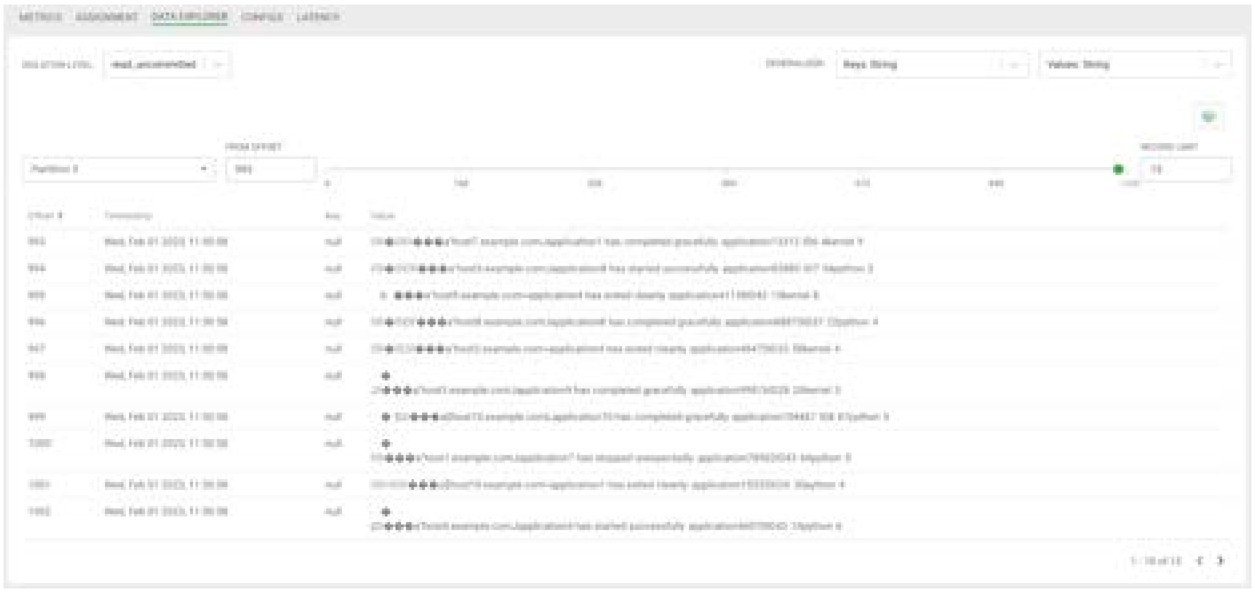
6. Access the SMM UI from the Streams Messaging DataHub page and check the topic metrics and data:


Producers
1 of 12
Brokers
1 of 3
Topics
1 of 32
Consumer Groups
0 of 3
fdanka-syslog
96 MB
0B
__smm-app-smm-producer-table-15m-changelog
__smm-app-smm-producer-table-15m-repartition
__smm-app-smm-producer-table-30s-changelog
__smm-app-smm-producer-table-30s-repartition
__transaction_state
heartbeats
fdanka-syslog
mm2-configs.secondary.internal

Here are the topic metrics:



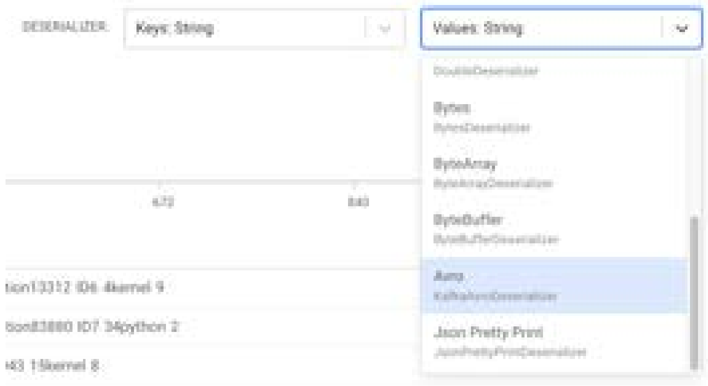
You can also view the data from the data explorer:



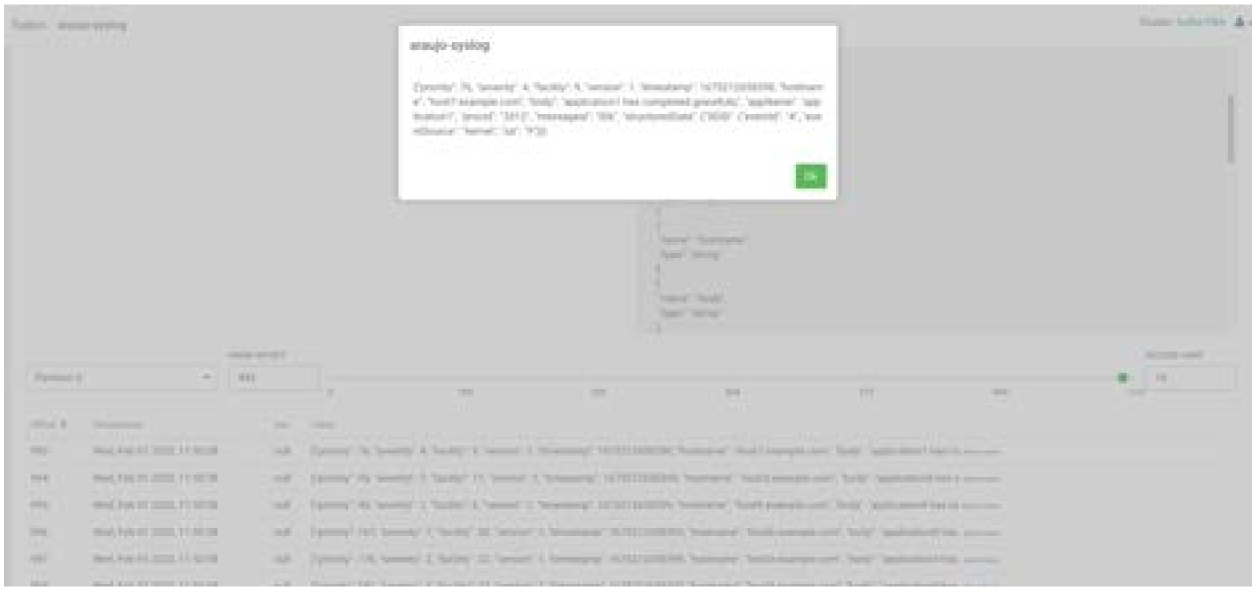
7. You can see that the data in that topic is binary (lots of garbage-like characters on the screen. This is because the topic contains Avro messages, which is a binary serialization format.

   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:



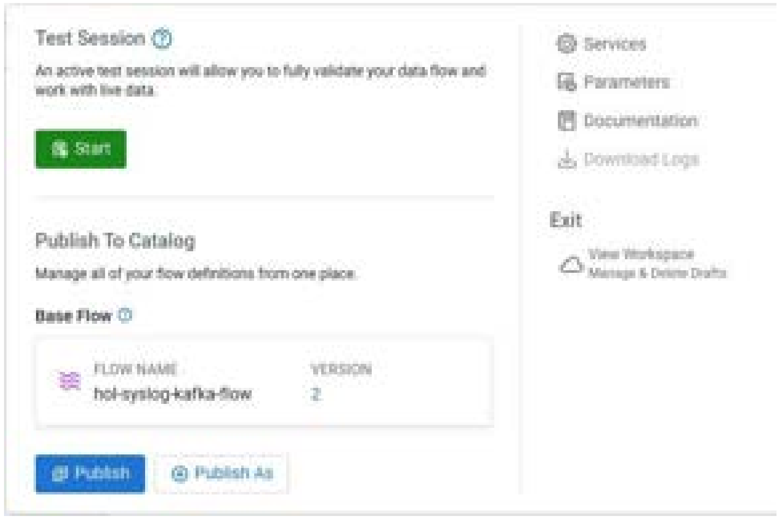
5. Version Control the flow into the DataFlow Catalog

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 As.


Test Session

An active test session will allow you to fully validate your data flow and work with live data.

Publish To Catalog

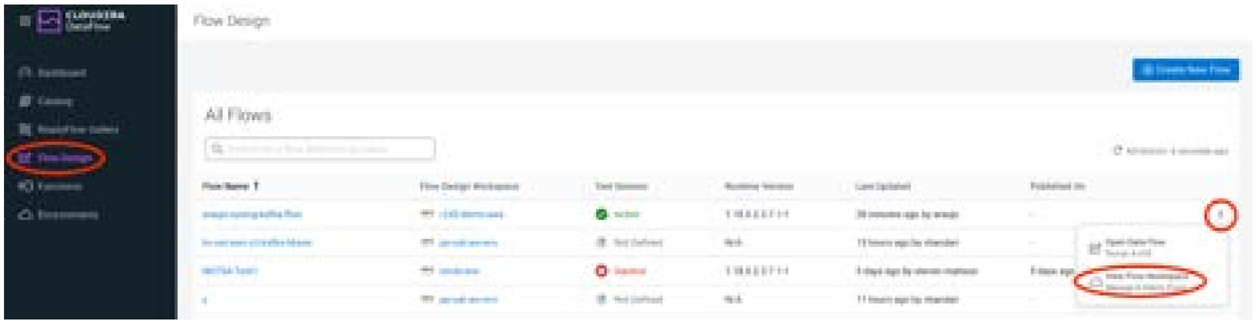
Manage all of your flow definitions from one place.

Base Flow

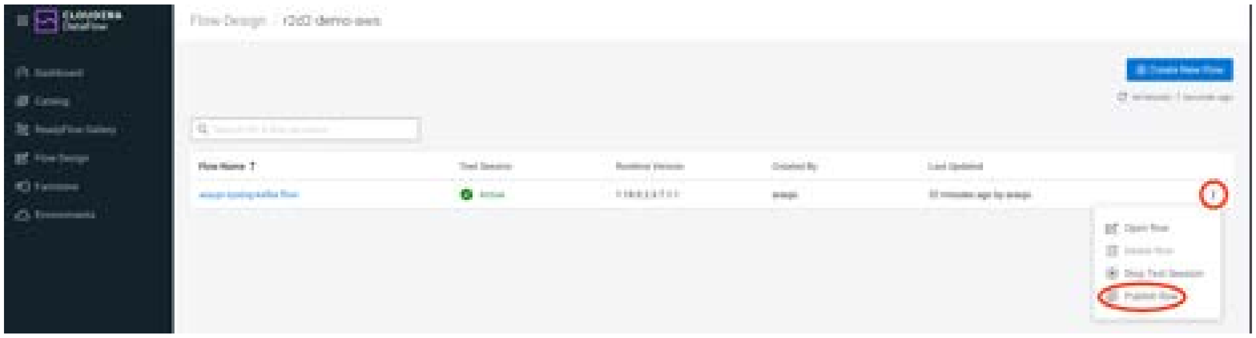
FLOW NAME	VERSION
hol-syslog-kafka-flow	2

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 As**.
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.
8. Now that you tested your flow and published to the catalog, you can end the test session to release the resources as a best practice. You have two options to end an active test session:

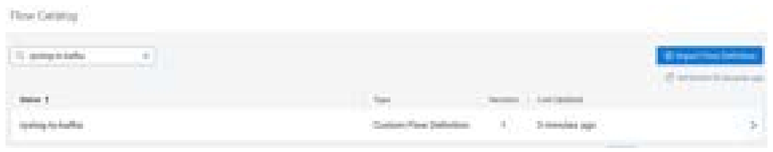
a. Click the toggle next to the Active Test Session notification

b. Click Flow Options then click

Stop under Test Session.

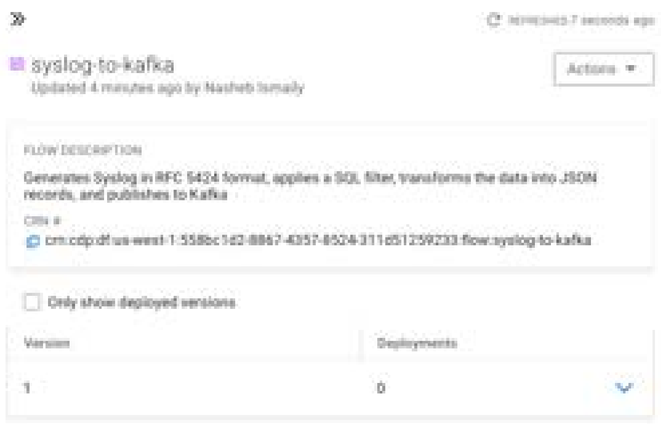
6. Deploy the flow in Production

1. Search for the flow in the Flow Catalog



Name ↑	Type	Versions	Last Updated
syslog-to-kafka	Custom Flow Definition	1	3 minutes ago

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



syslog-to-kafka
Updated 4 minutes ago by Nasheb Ismaily

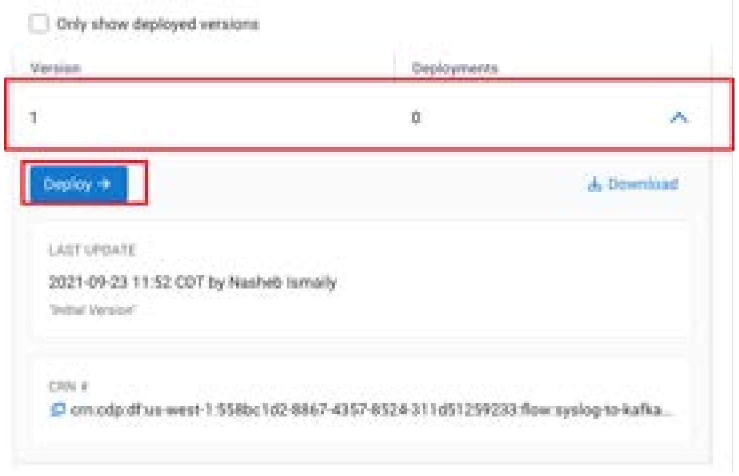
FLOW DESCRIPTION
Generates Syslog in RFC 5424 format, applies a SQL filter, transforms the data into JSON records, and publishes to Kafka

CRN #
crn:cdp:df:us-west-1:558bc1d2-8867-4357-8524-311d51259233:flow:syslog-to-kafka

Only show deployed versions

Version	Deployments
1	0

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



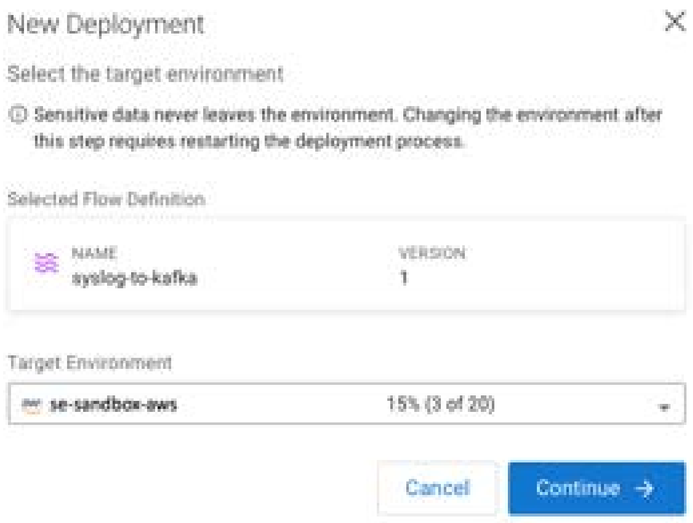
Only show deployed versions

Version	Deployments
1	0

LAST UPDATE
2021-09-23 11:52 CDT by Nasheb Ismaily
"Initial Version"

CRN #
crn:cdp:df:us-west-1:558bc1d2-8867-4357-8524-311d51259233:flow:syslog-to-kafka...

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



New Deployment

Select the target environment

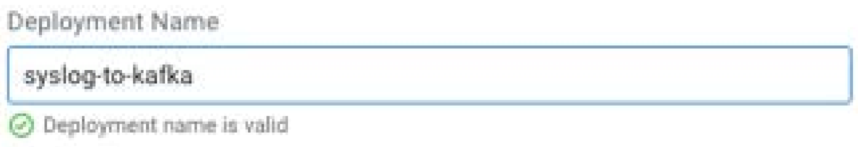
Sensitive data never leaves the environment. Changing the environment after this step requires restarting the deployment process.

Selected Flow Definition

NAME	VERSION
syslog-to-kafka	1

Target Environment
se-sandbox-aws 15% (3 of 20)

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

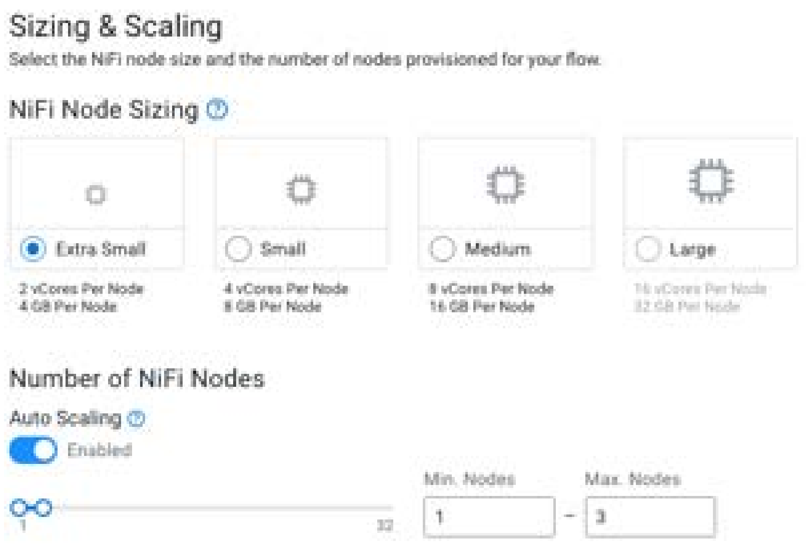


Deployment Name
syslog-to-kafka
Deployment name is valid

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

● Extra Small	○ Small	○ Medium	○ Large
2 vCores Per Node 4 GB Per Node	4 vCores Per Node 8 GB Per Node	8 vCores Per Node 16 GB Per Node	16 vCores Per Node 32 GB Per Node

Number of NiFi Nodes

Auto Scaling: Enabled

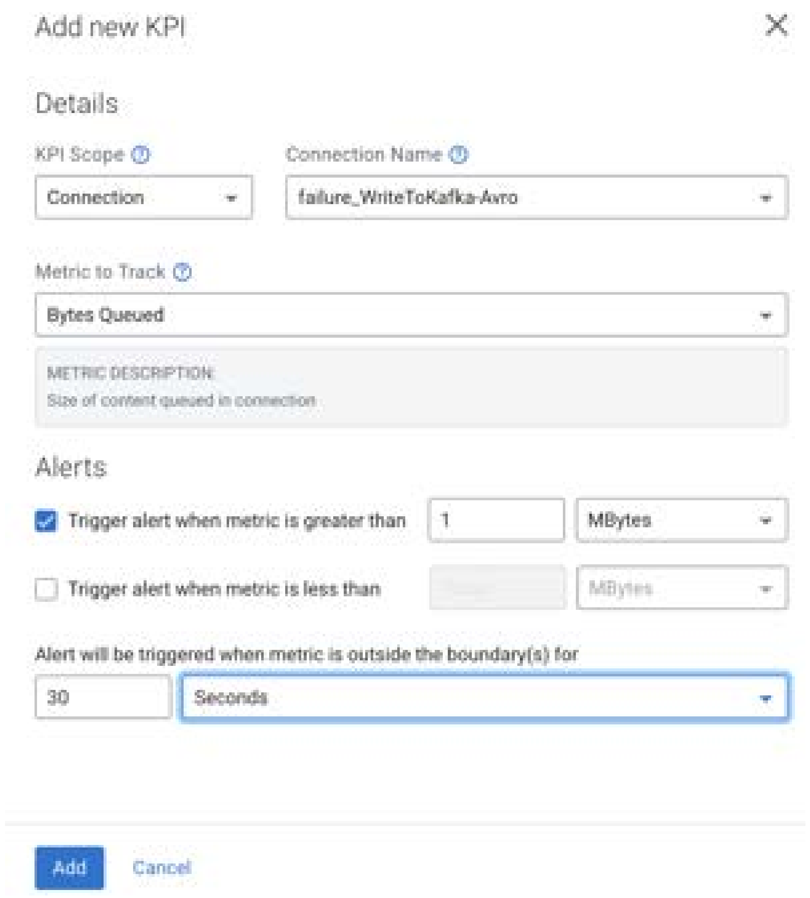
Min. Nodes: 1 – Max. Nodes: 3

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-Avro**
        - 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

Details

KPI Scope: Connection

Connection Name: failure_WriteToKafka-Avro

Metric to Track: Bytes Queued

METRIC DESCRIPTION:
Size of content queued in connection

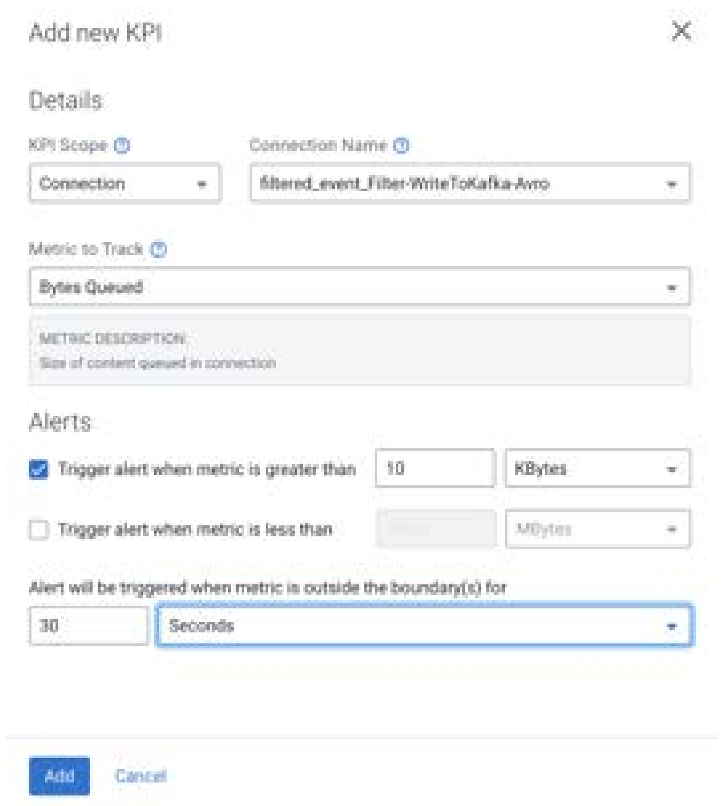
Alerts

☑ Trigger alert when metric is greater than 1 MBytes

☐ Trigger alert when metric is less than MBytes

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

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

Connection Name: filtered_event_Filter-WriteToKafka-Avro

Metric to Track: Bytes Queued

METRIC DESCRIPTION:
Size of content queued in connection

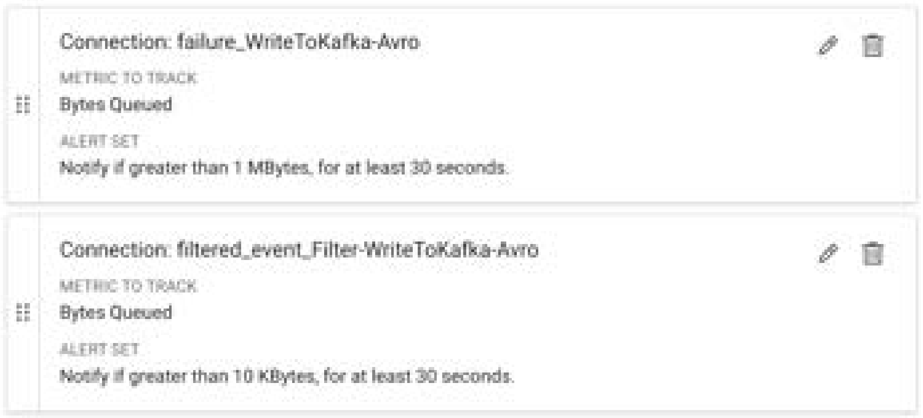
Alerts

[✓] Trigger alert when metric is greater than 10 KBytes

[ ] Trigger alert when metric is less than MBytes

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

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


Connection: failure_WriteToKafka-Avro
METRIC TO TRACK
Bytes Queued
ALERT SET
Notify if greater than 1 MBytes, for at least 30 seconds.

Connection: filtered_event_Filter-WriteToKafka-Avro
METRIC TO TRACK
Bytes Queued
ALERT SET
Notify if greater than 10 KBytes, for at least 30 seconds.

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

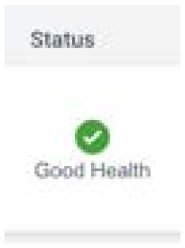
    Notice that in this page there's a **>_ View CLI Command** link. You will use the information in the page in the next section to deploy a flow using the CLI. For now you just need to save the script and dependencies provided there:

    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.

7. Deploy the Flow using CLI

In this section you will use the CDP CLI to deploy a flow in Cloudera DataFlow. For this you will use the following files created in the last section:

- **deploy.sh**: file created with the CLI command copied from the CLI Command page.
- **<deployment_name>-parameter-groups.json**: file downloaded from the CLI Command page, containing the parameter definitions for the deployment
- **<deployment_name>-kpis.json**: file downloaded from the CLI Command page, containing the KPI definitions for the deployment

1. If you don't have the CDP CLI installed on your laptop, follow the documentation to install and configure it.
2. To test that the CDP CLI is working properly, execute the command below. If the command executes successfully, it will show the details of your CDP account:

   ```
   cdp iam get-user
   ```

   The output should be a JSON like the one below:

   ```
   {
       "user": {
           "userId": "...",
           "crn": "...",
           "email": "...",
           "firstName": "...",
           "lastName": "...",
           "creationDate": "...",
           "accountAdmin": false,
           "identityProviderCrn": "...",
           "lastInteractiveLogin": "...",
           "workloadUsername": "...",
           "workloadPasswordDetails": {
               "isPasswordSet": true
           }
       }
   }
   ```

3. Edit the **deploy.sh** file and modify the details for the following parameters:
   a. **--deployment-name**: change the deployment name to something unique. Since you already deployed this flow from the UI you already have a deployment with the name in this file.

   b. **--parameter-groups**: replace the <<PATH_TO_UPDATE>> string with the full path of the parameter group definition file. E.g.:

      ```
      file:///Users/rbearden/Downloads/syslog001-parameter-groups.json
      ```

   c. **--kpis**: replace the <<PATH_TO_UPDATE>> string with the full path of the KPI definition file. E.g.:

      ```
      file:///Users/rbearden/Downloads/syslog001-kpis.json
      ```

4. Edit the **<deployment_name>-parameter-groups.json** file and replace the **<<CDP_MISSING_SENSITIVE_VALUE>>** string with the proper value for the corresponding parameter(s).

   You can also change values of other parameters if wanted/needed.

5. Execute the deploy.sh script to create the new flow deployment:

   ```
   bash deploy.sh
   ```

6. Go to the Cloudera DataFlow Dashboard, monitor the flow deployment and verify that the deployment completes successfully.

Module 2 - Streaming Analytics with Cloudera Stream Processing

1. Overview

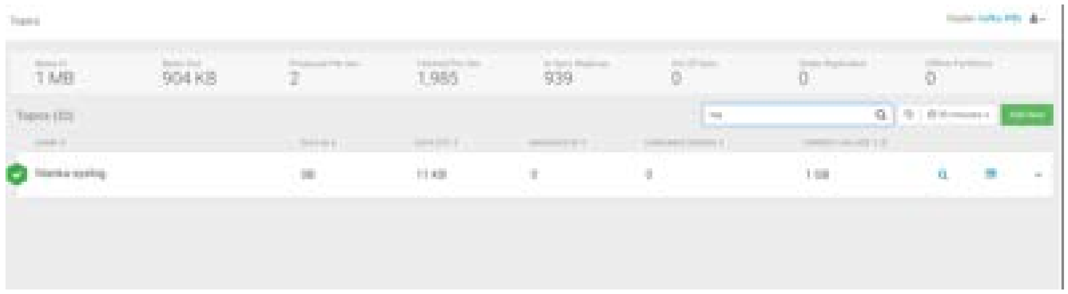
In this module you will practice the streaming analytic capabilities of Cloudera Stream Processing using SQL Stream Builder to develop and deploy a real-time streaming job in production. You will leverage the NiFi Flow deployed in CDF-PC from the previous workshop and demonstrate how to query live data and subsequently sink it to another location. You will learn to:

- Use Stream Messaging Manager to inspect Kafka topics.
- Create a Project in SQL Stream Builder (SSB).
- Register Schema Registry and Kudu Catalogs in SSB.
- Create Virtual Tables in SSB.
- Create and execute Jobs in SSB.

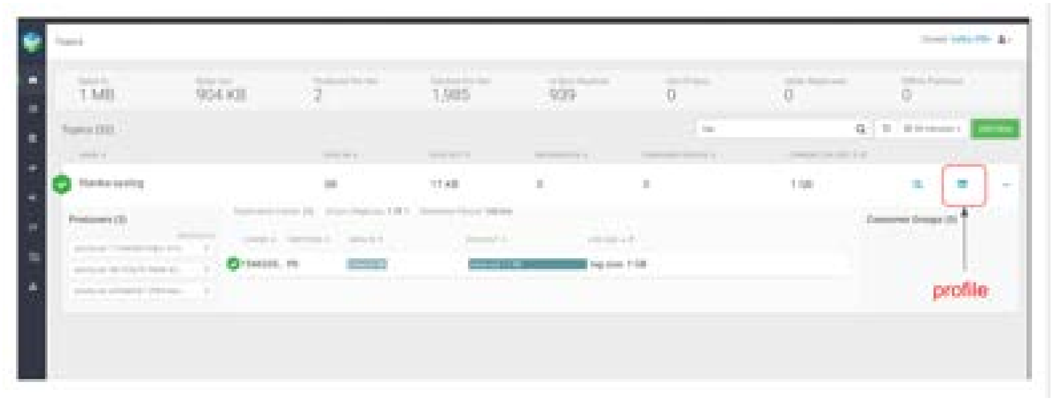
2. Inspect Kafka topics in SMM

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

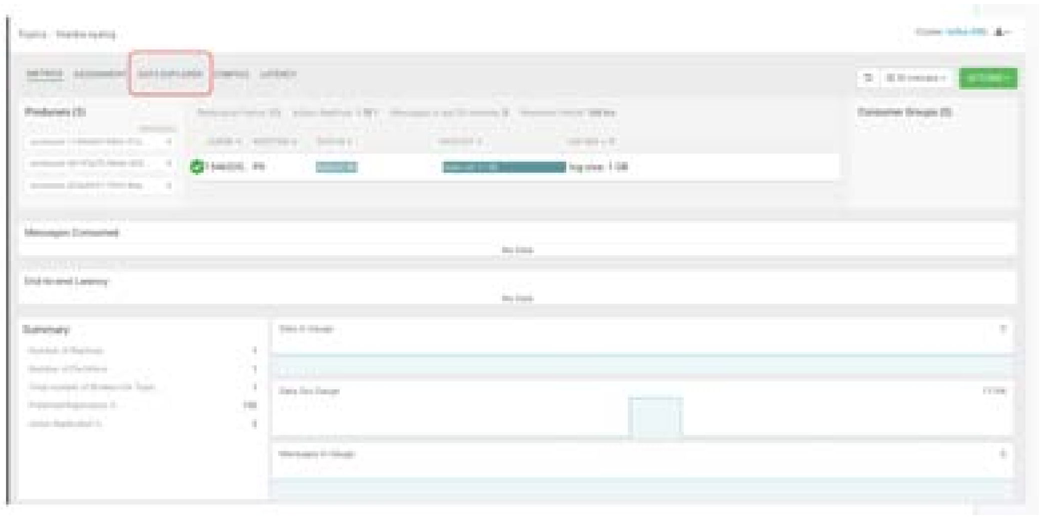
1. Login to **SMM**.
2. Click on the **Topics icon** (🛢) and filter the topic **<userid>-syslog-avro**, which was created by the NiFi flow in Module 1



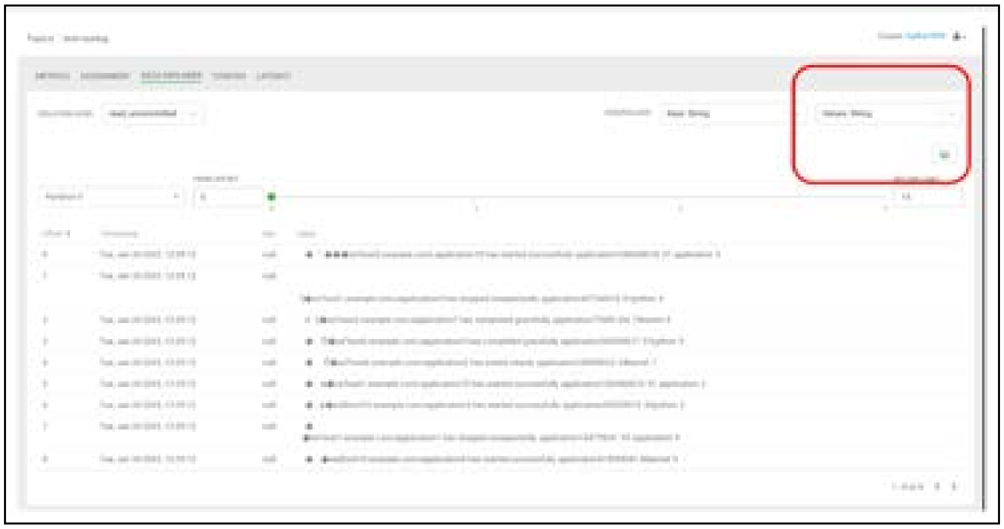
3. Click on the **Profile** icon for the topic, as shown below:

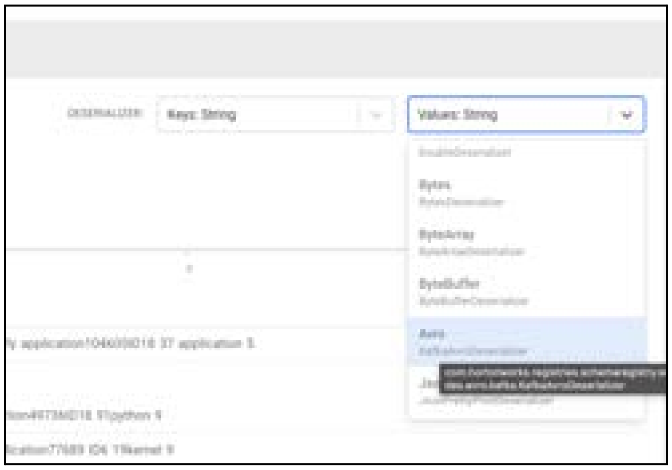


4. On the Topic page, click on the **DATA EXPLORER** tab:



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

Bytes
BytesDeserializer

ByteArray
ByteArrayDeserializer

ByteBuffer
ByteBufferDeserializer

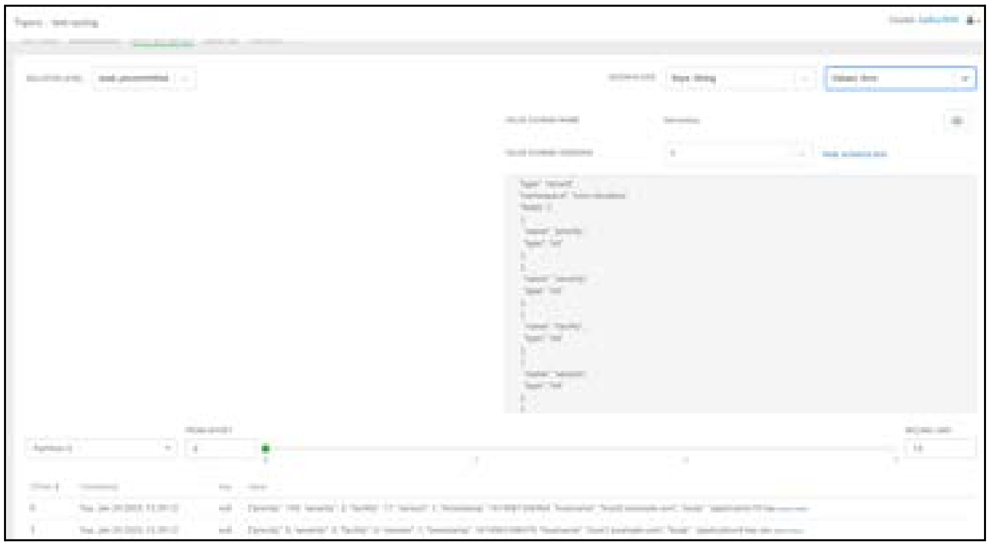
Avro
KafkaAvroDeserializer

com.hortonworks.registries.schemaregistry.serdes.avro.kafka.KafkaAvroDeserializer

ly application104K00D18 37 application 5

tion49736D18 91python 9

lication77689 ID6 19kernel 9



3. Create a Project in SSB and configure GitHub repo for it

In this section you will start getting familiar with SQL Stream Builder (SSB) and will create a project in it to use throughout the remainder of this workshop. Projects in SSB can be stored in a Git repository and you will learn how to configure your project to use one.

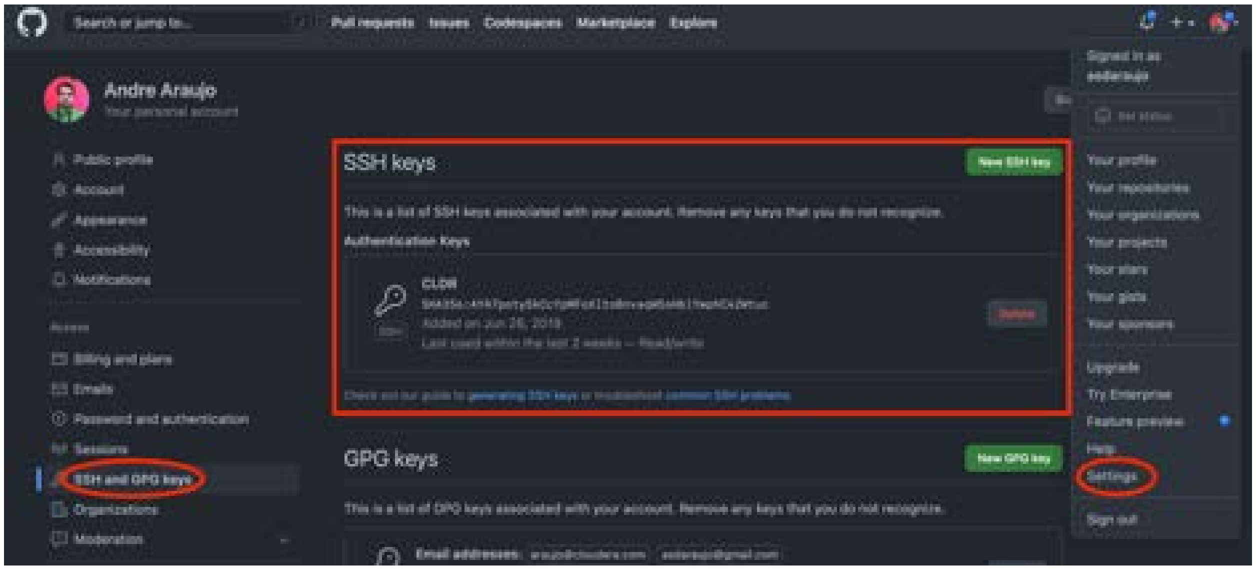
Before you do that, though, you need to create a Git repository under your own GitHub account. It is recommended that you *register a SSH key with your GitHub account* so that you can use SSH authentication. Using BASIC authentication (username and password) is not recommended since that will not work if you have two-factor authentication enabled for it.

1. Login to github.com with your own personal user.

   NOTE: Use an account in the public github.com website. The internal Cloudera GitHub will not be reachable by SSB.

2. Click on **Settings** for your account and then on the **SSH and GPG keys** menu. If you already have a SSH key registered and you have a copy of the private key in your computer, you can use it when configuring your Project in SSB.

   If you don't have a SSH key register or don't know where it's the private key file associated with the registered key, click on the **New SSH key** button and create/register a new one.



This is a list of SSH keys associated with your account. Remove any keys that you do not recognize.

Authentication Keys

CLDR

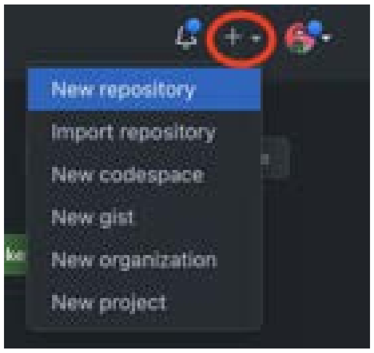
Added on Jun 26, 2019

Last used within the last 2 weeks — Read/write

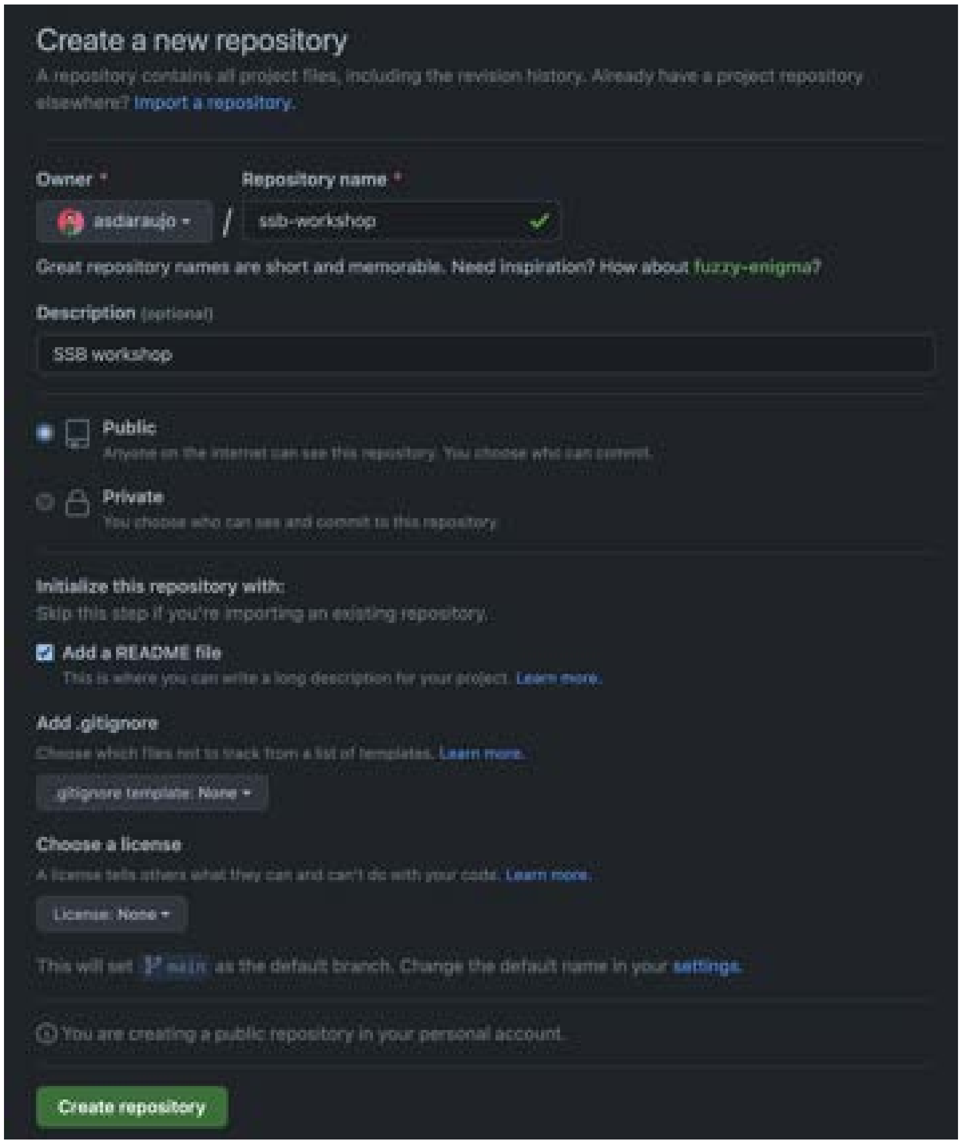
Check out our guide to generating SSH keys or troubleshoot common SSH problems.

This is a list of GPG keys associated with your account. Remove any keys that you do not recognize.

3. Click the "plus" icon at the top right of the page and select **New repository** to add a new repository. Call it **ssb-workshop** and set it like the one below:



4. Make sure that the “Add a README file” option is checked.



Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? Import a repository.

Owner * asdaraujo / Repository name * ssb-workshop

Great repository names are short and memorable. Need inspiration? How about fuzzy-enigma?

Description (optional)

SSB workshop

[✓ Public] Anyone on the internet can see this repository. You choose who can commit.

Private You choose who can see and commit to this repository.

Initialize this repository with:

Skip this step if you're importing an existing repository.

[✓] Add a README file
This is where you can write a long description for your project. Learn more.

Add .gitignore

Choose which files not to track from a list of templates. Learn more.

.gitignore template: None

Choose a license

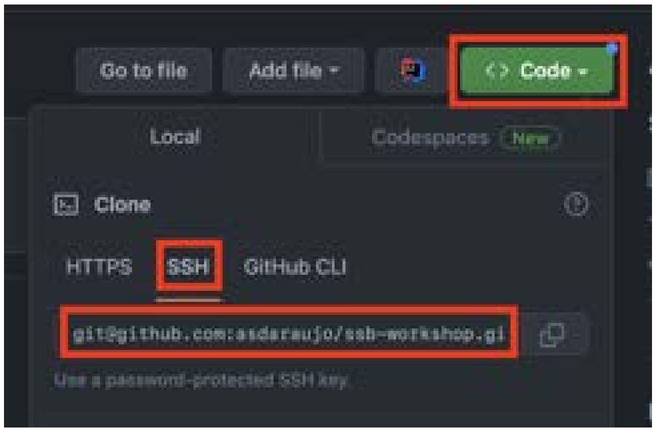
A license tells others what they can and can't do with your code. Learn more.

License: None

This will set main as the default branch. Change the default name in your settings.

You are creating a public repository in your personal account.

5. Once the repository is created, click on the Code button on the repository page, select the SSH tab and copy the associated URL. This is the URL you will use to configure the SSB project:

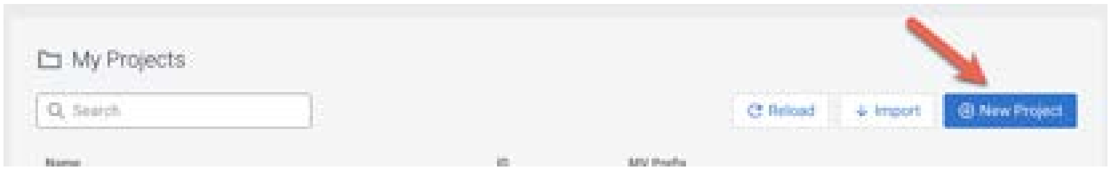


```
git@github.com:asdaraujo/ssb-workshop.gi
```

6. In the CDP Console, find the **Streaming Analytics (Flink) DataHub** and click on it to open the DataHub page.
7. Click on the Streaming SQL Console icon to open the SSB UI. You will notice that there are already two projects created by default:
   a. **ssb_default** - a global project accessible to all users
   b. **<userid>_default** - a private project accessible only to your user

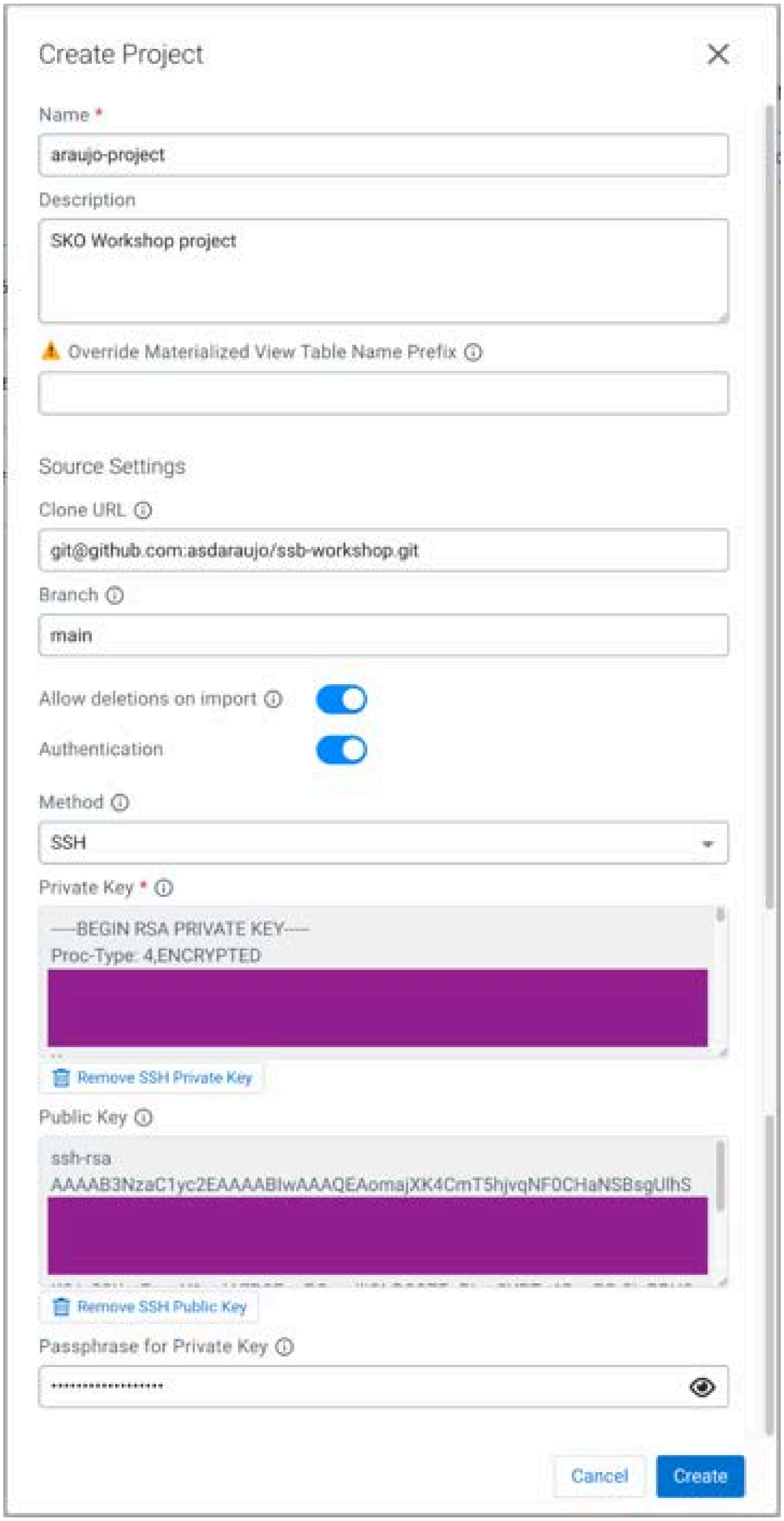
   You won't use any of those. Instead, you will create a GitHub-backed repository next.

8. In the **Projects** page, click on the **New Project** button:



9. Enter the following details in the Create Project dialog box:
   a. Name: **<an unique name for your project>**
   b. Clone URL: **<the URL you copied in step 4>**
   c. Branch: **main** (by default a GitHub repository is created with a **main** branch. If you modified that, make sure you specify the correct branch name here)
   d. Authentication: **Enabled**
   e. Method: **SSH**
   f. Private Key: **<upload the private key file associated with the key in your GitHub account>**

g. Public Key: **<upload the public key file associated with the key in your GitHub account>**
h. Passphrase for Private Key: **<if your private key is protected by a passphrase, enter it here; otherwise, leave it blank>**


Create Project

Name *

araujo-project

Description

SKO Workshop project

Override Materialized View Table Name Prefix

Source Settings

Clone URL

git@github.com:asdaraujo/ssb-workshop.git

Branch

main

Allow deletions on import: on

Authentication: on

Method

SSH

Private Key *

```
-----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4,ENCRYPTED
```
[redacted]

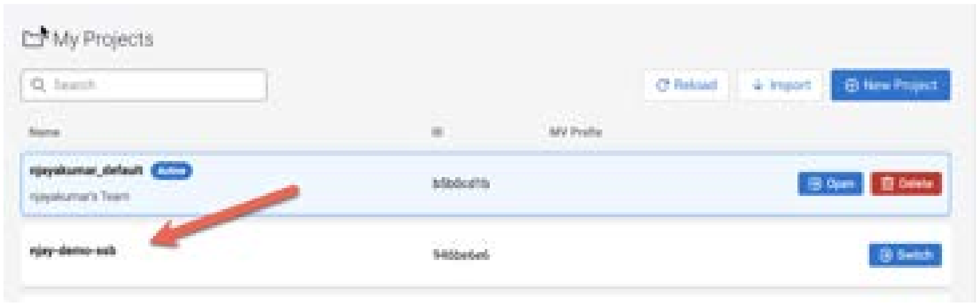
Public Key

```
ssh-rsa
AAAAB3NzaC1yc2EAAAABIwAAAQEAomajXK4CmT5hjvqNF0CHaNSBsgUlhS
```
[redacted]

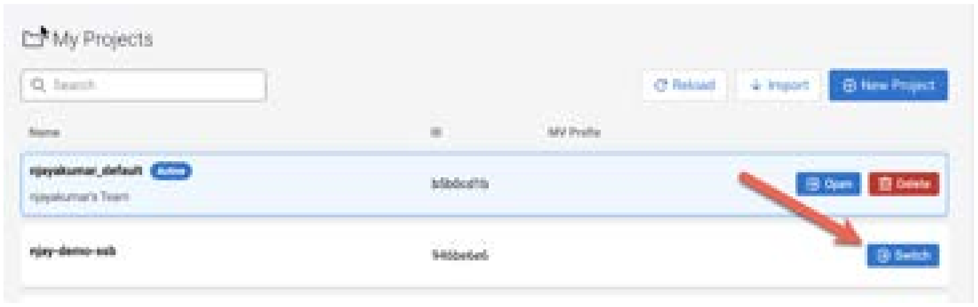
Passphrase for Private Key

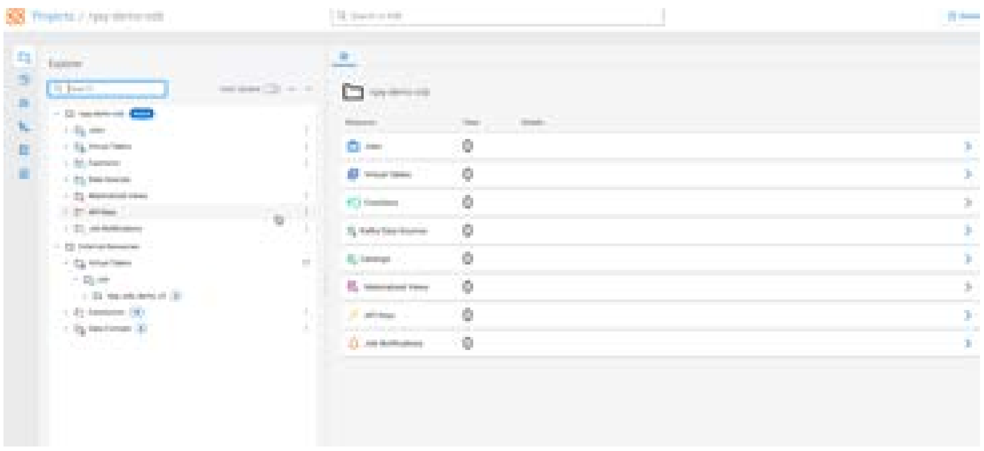
••••••••••••••••••••

10. The new project will be listed in the Projects page:



11. Click on the **Switch** button of the new project to switch to it, and you will see the main page for your project::

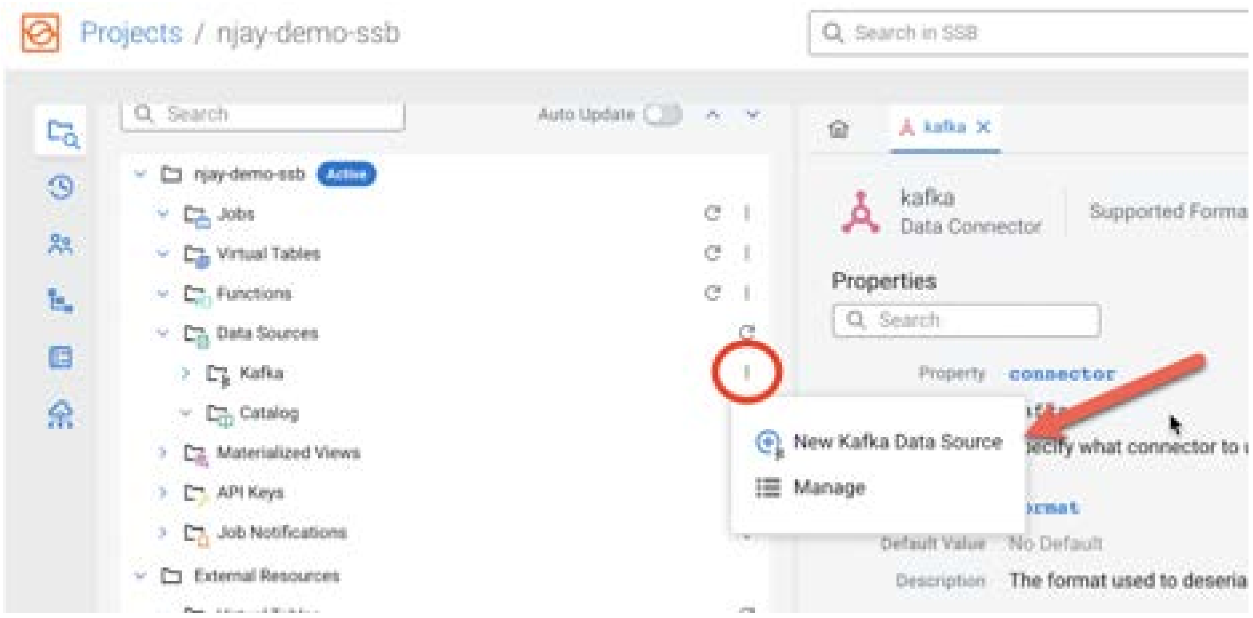




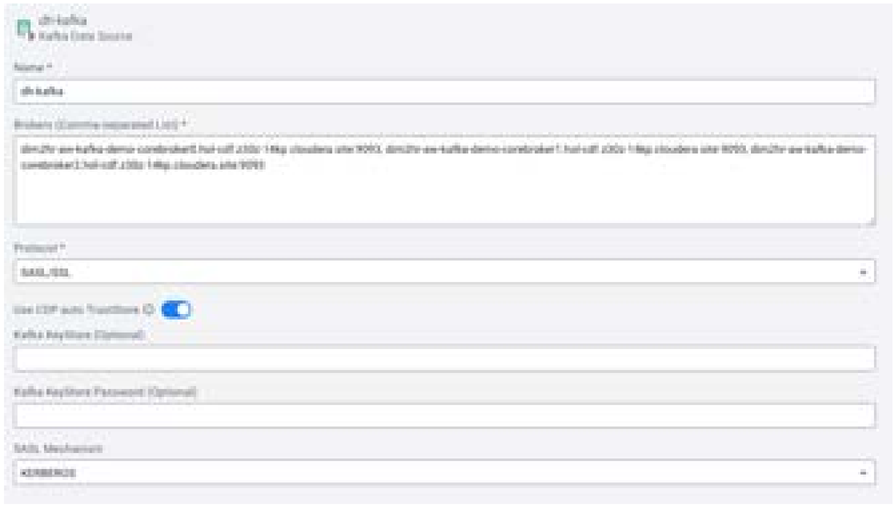
4. Register a Kafka Data Source

Before you can use SSB to process data from Kafka topics you must register one or more Kafka clusters as Data Sources in your project. In this section you will register the Kafka DataHub as your data source.

1. In your project workspace, expand the Data Sources item in the navigation tree. You will see two items under it: "Kafka" and "Catalog"
2. Click on the kebab menu (⋮) of the **Kafka** node to create a new Kafka data source:



3. Provide the following details in the Kafka Data Source dialog box:
   a. Name: **<name of your data source. E.g. dh-kafka>**
   b. Brokers: **<comma-separated list of brokers>** (See the Notes under Module 1's section 3.2.2 for more details on how to find the Kafka broker addresses)
   c. Use CDP Auto TrustStore
   d. Protocol: **SASL/SSL**
   e. SASL Mechanism: **KERBEROS**


dh-kafka
Kafka Data Source

Name *
dh-kafka

Brokers (Comma separated List) *
[illegible]

Protocol *
SASL/SSL

Use CDP auto TrustStore ⓘ ●

Kafka KeyStore (Optional)

Kafka KeyStore Password (Optional)

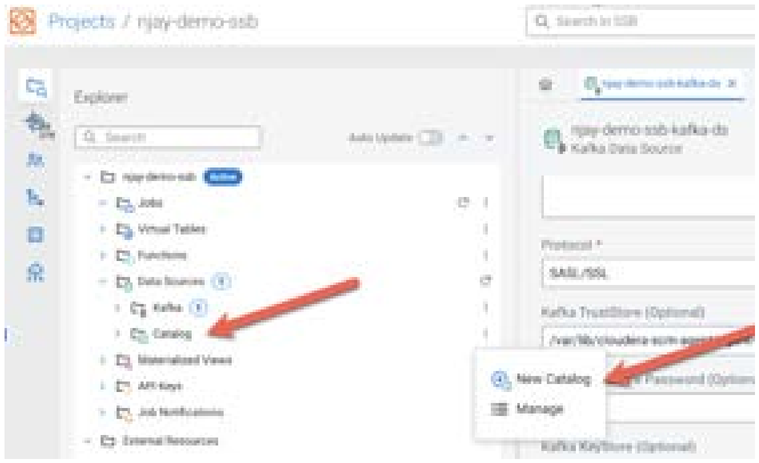
SASL Mechanism
KERBEROS

5. Register a Schema Registry catalog

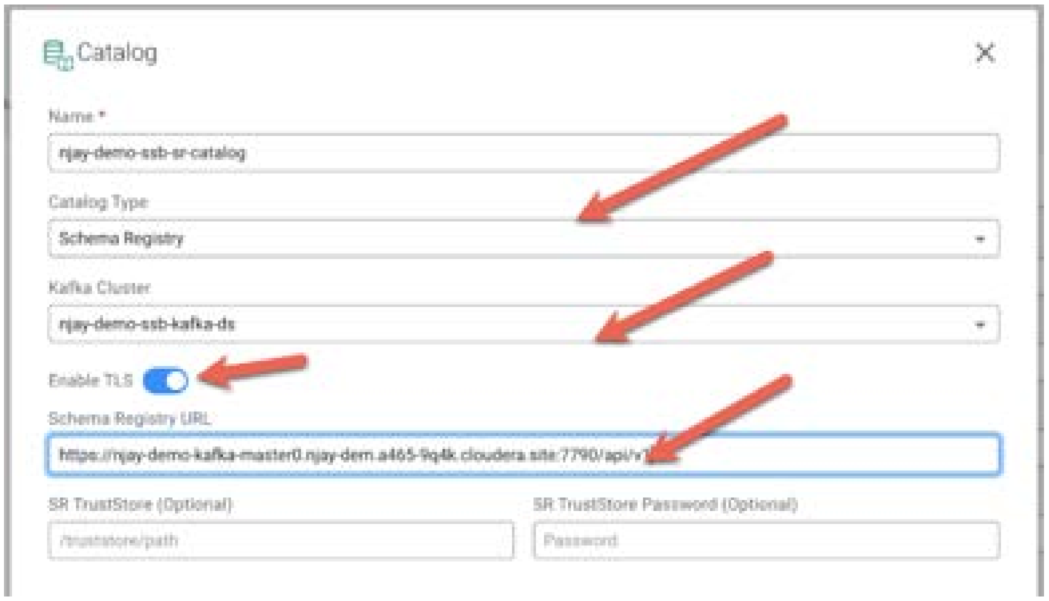
Schema Registry stores schemas that are used to read/write data from/to Kafka topics. Registering a Schema Registry instance in SSB automatically creates SSB tables for the schemas retrieved from it, making it simpler to start accessing data from those topics.

Note: For this to work the schemas stored in Schema Registry need to be stored with the same name of the Kafka topic.

1. In your project workspace, navigate to **Data Sources -> Catalog.** Click the kebab menu of the Catalog item and select **New Catalog** to create a new catalog.



2. Enter the following details in the Catalog dialog box:
    a. Name: **<name of the catalog. E.g. dh-schreg>**
    b. Catalog Type: **Schema Registry**
    c. Kafka Cluster: **<select the Kafka data source you created the previous section>**
    d. Enable TLS: **Enabled**
    e. Schema Registry URL: **https://<schreg_hostname>:7790/api/v1** (See the Notes under Module 1's section 3.2.2 for more details on how to find the Schema Registry hostname)


Catalog

Name *
njay-demo-ssb-sr-catalog

Catalog Type
Schema Registry

Kafka Cluster
njay-demo-ssb-kafka-ds

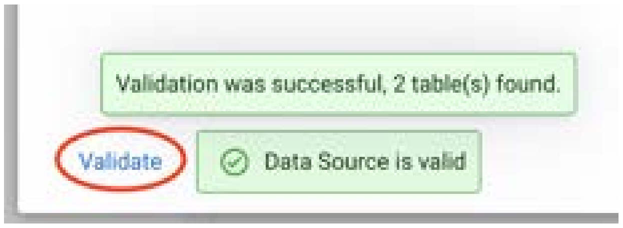
Enable TLS

Schema Registry URL
https://njay-demo-kafka-master0.njay-dem.a465-9q4k.cloudera.site:7790/api/v1

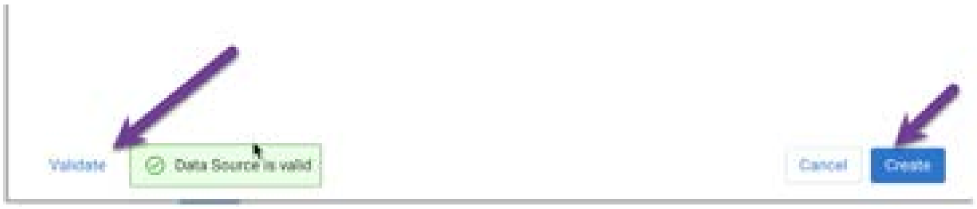
SR TrustStore (Optional)
/truststore/path

SR TrustStore Password (Optional)
Password

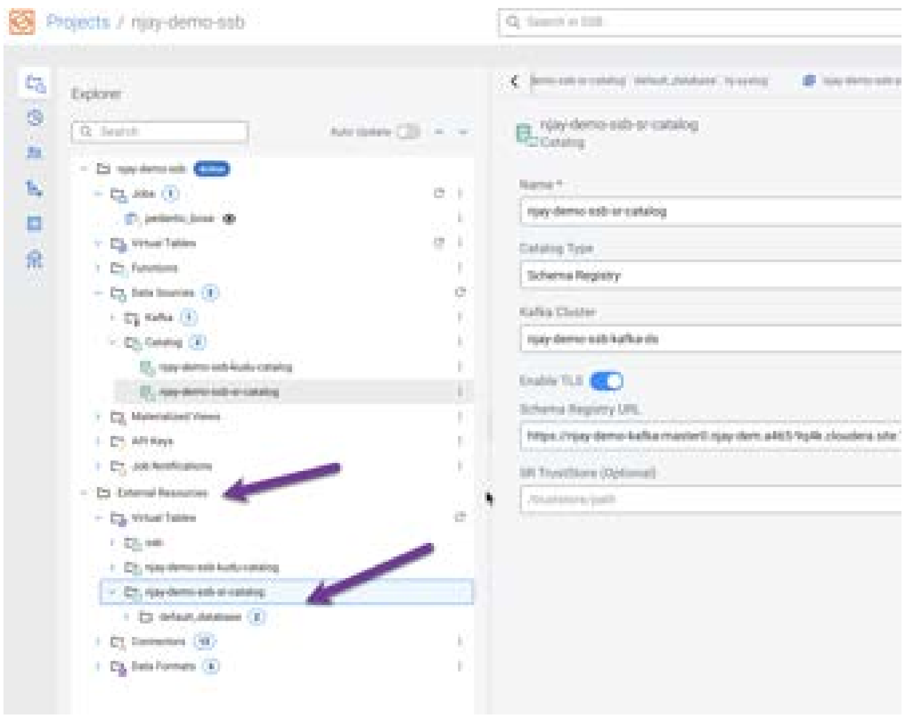
3. Click on the **Validate** link to validate the Catalog. If the catalog is successfully validated you will see a message saying "Data Source is valid". If you hover the mouse over that message you will see how many tables were discovered from Schema Registry during the validation:


Validation was successful, 2 table(s) found.

4. Once the catalog is validated. Press the **Create** button to register the Schema Registry catalog.



5. After the registration is completed you can browse the Schema Registry tables by navigating to **External Resources> Virtual Tables > [schema_registry_catalog_name] > default_database**:



Name *
njay-demo-ssb-sr-catalog

Catalog Type
Schema Registry

Kafka Cluster
njay-demo-ssb-kafka-ds

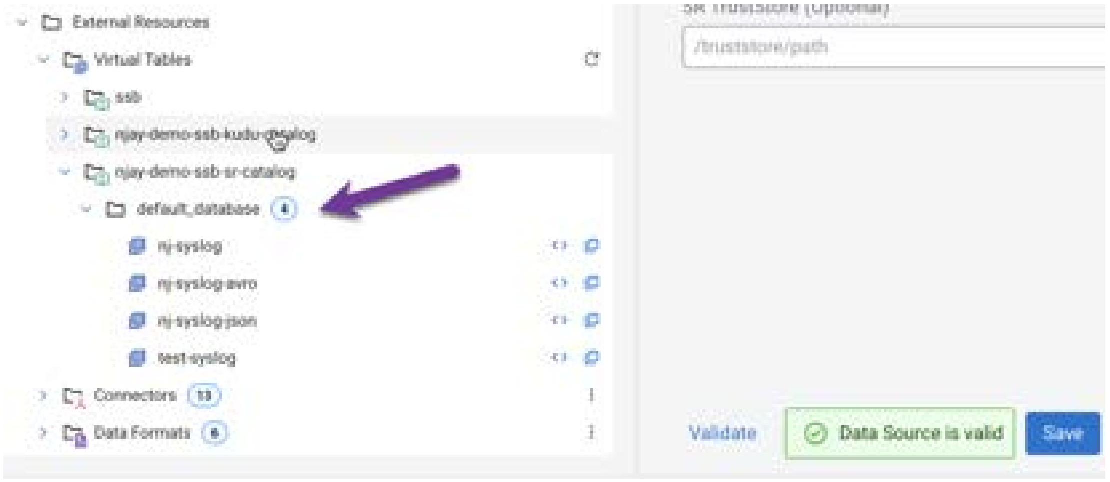
Enable TLS

Schema Registry URL
https://njay-demo-kafka-master0.njay-dem.a465-9q4k.cloudera.site

SR TrustStore (Optional)
/truststore/path

6. The Virtual Tables are created based on the schemas retrieved from Schema Registry. Virtual table names are the same as that of schema names.

   Note: The topic names in Kafka must also be the same as the name of the schema associated with it.



External Resources
- Virtual Tables
  - ssb
  - njay-demo-ssb-kudu-catalog
  - njay-demo-ssb-sr-catalog
    - default_database (4)
      - nj-syslog
      - nj-syslog-avro
      - nj-syslog-json
      - test-syslog
- Connectors (13)
- Data Formats (6)

SR TrustStore (Optional)
/truststore/path

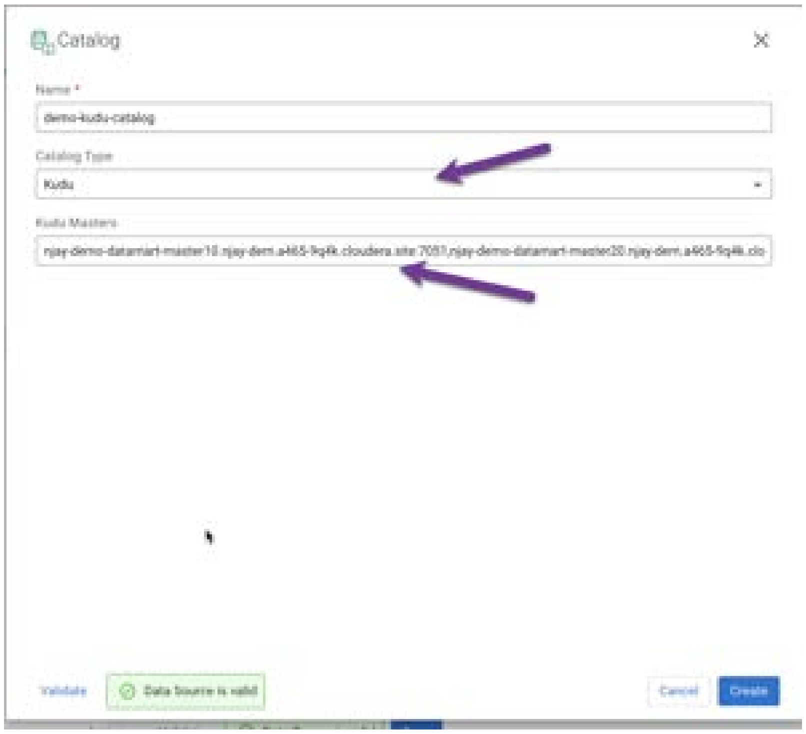
6. Register a Kudu catalog

Similar to a Schema Registry catalog, a Kudu catalog makes Kudu tables automatically available for use in SSB, simplifying the process of mapping Virtual Tables to the actual Kudu ones. In this section you will register the DataHub Kudu cluster as a catalog in your SSB project.

This will make the **syslog_severity** table, which has already been created in Kudu, available for queries in SSB.

1. In your project workspace, navigate to **Data Sources -> Catalog.** Click the kebab menu of the Catalog item and select **New Catalog** to create a new catalog.

2. Enter the following details in the Catalog dialog box:
    a. Name: **<name of the catalog. E.g. dh-kudu>**
    b. Catalog Type: **Kudu**
    c. Kudu Masters: **<kudu_master1>:7051,<kudu_master2>:7051,<kudu_master3>:7051** *(see **Notes** below)*



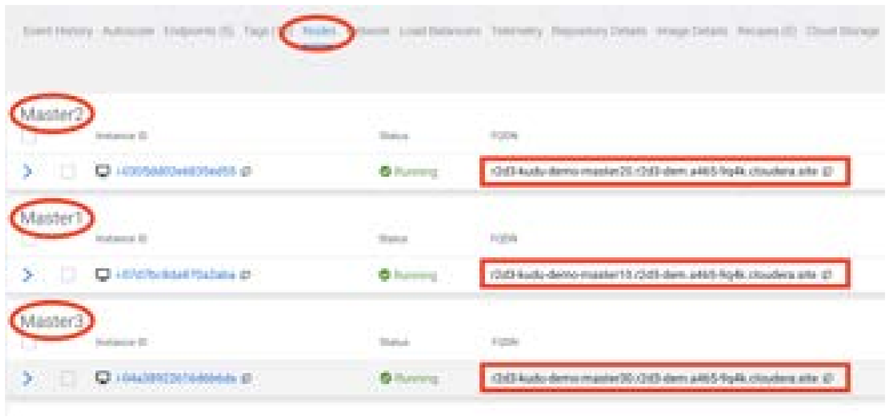
Name *
demo-kudu-catalog

Catalog Type
Kudu

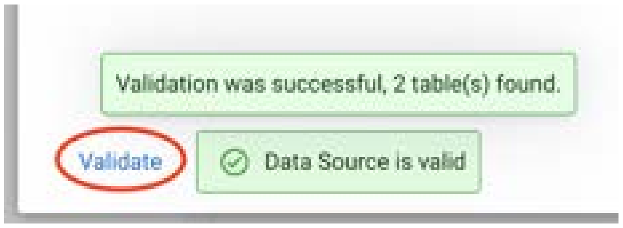
Kudu Masters
njay-demo-datamart-master10.njay-dem.a465-9q4k.cloudera.site:7051,njay-demo-datamart-master20.njay-dem.a465-9q4k.clo

Notes:
- **Kudu Masters:** The hostname of the Kudu master nodes can be found in the Real-time Data Mart (Kudu) DataHub page. In the CDP Console, find your Real-time Data Mart DataHub and click on the **Nodes** tab. Copy the FQDN for the 3 masters, as shown in the screenshot below.

The full masters address to be provided during the catalog registration is the following:

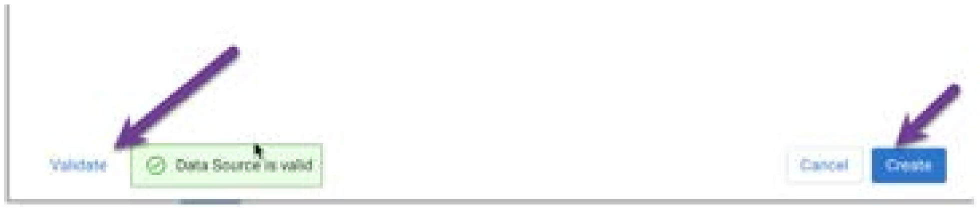


3. Click on the **Validate** link to validate the Catalog. If the catalog is successfully validated you will see a message saying "Data Source is valid". If you hover the mouse over that message you will see how many tables were discovered from Schema Registry during the validation:

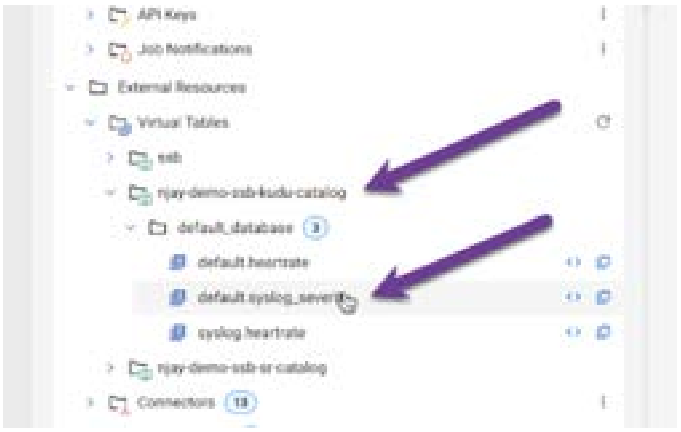


Validation was successful, 2 table(s) found.

4. Once the catalog is validated. Press the **Create** button to register the Kudu catalog.



5. After the registration is completed you can browse the Kudu tables by navigating to **External Resources> Virtual Tables > [kudu_catalog_name] > default_database**:



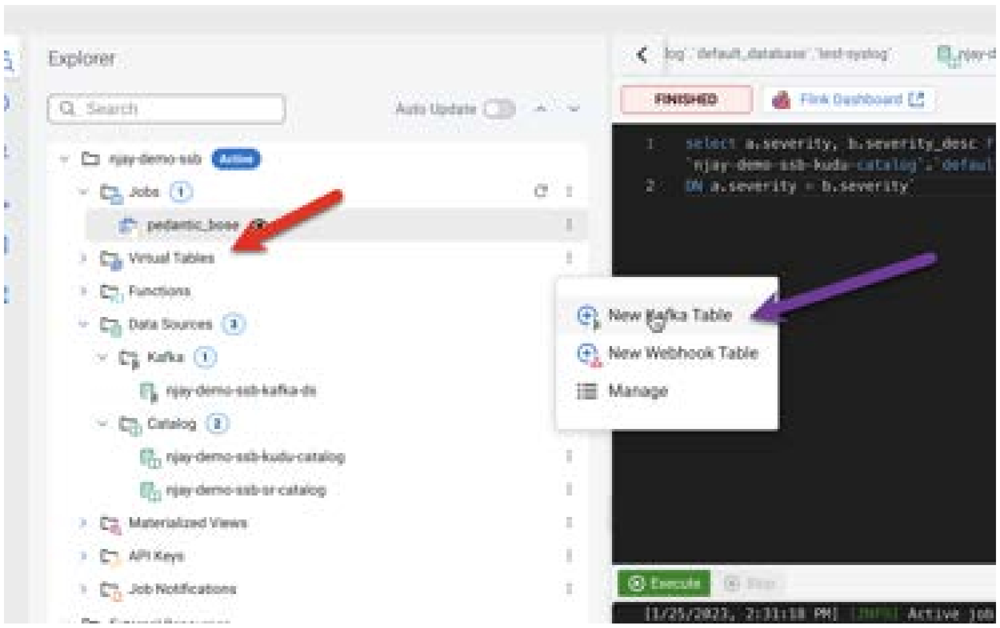
7. Create a Kafka Table manually

You saw that when you register a Schema Registry catalog SSB automatically maps all the schemas found in there as SSB tables that can be used to access data in the Kafka topics named after the schemas.

Sometimes, though, you don't have a schema for a particular topic and you may still want to consume or produce data to that topic. In SSB you can manually create a table and specify its schema directly, as well as the mapping to an existing Kafka topic. In this section you will practice this using the Add Table wizard.

1. In your project workspace, navigate to **Data Sources -> Virtual Tables.**

2. Click the kebab menu of the Virtual Tables item and select **New Kafka Table** to create a new virtual table.



```
select a.severity, b.severity_desc f
`njay-demo-ssb-kudu-catalog`.`defaul
ON a.severity = b.severity`
```

```
[11/25/2023, 2:31:18 PM] [INFO] Active job
```

3. Enter the following details in the Kafka Table dialog box:
    - Table Name: **syslog_data**
    - Kafka Cluster: **<select the Kafka data source you created previously>**
    - Data Format: **JSON**
    - Topic Name: **<userid>-syslog-json**

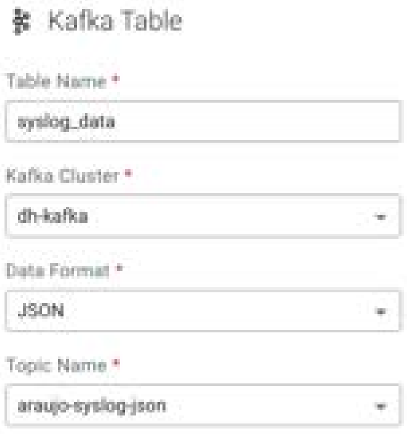


Table Name *	syslog_data
Kafka Cluster *	dh-kafka
Data Format *	JSON
Topic Name *	araujo-syslog-json

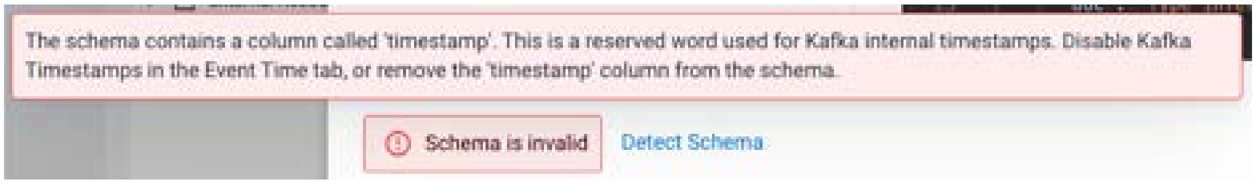
4. When you select Data Format as AVRO, you must provide the correct Schema Definition when creating the table for SSB to be able to successfully process the topic data.

    For JSON tables, though, SSB can look at the data flowing through the topic and try to infer the schema automatically, which is quite handy at times. Obviously, there must be data in the topic already for this feature to work correctly.

Note: SSB tries its best to infer the schema correctly, but this is not always possible and sometimes data types are inferred incorrectly. You should always review the inferred schemas to check if it's correctly inferred and make the necessary adjustments.

Since you are reading data from a JSON topic, go ahead and click on **Detect Schema** to get the schema inferred. You should see the schema be updated in the **Schema Definition** tab.

5. You will also notice that a "Schema is invalid" message appears upon the schema detection. If you hover the mouse over the message it shows the reason:



The schema contains a column called 'timestamp'. This is a reserved word used for Kafka internal timestamps. Disable Kafka Timestamps in the Event Time tab, or remove the 'timestamp' column from the schema.

You will fix this in the next step.

6. Each record read from Kafka by SSB has an associated timestamp column of data type TIMESTAMP ROWTIME. By default, this timestamp is sourced from the internal timestamp of the Kafka message and is exposed through a column called eventTimestamp.

   However, if your message payload already contains a timestamp associated with the event (event time), you may want to use that instead of the Kafka internal timestamp.

   In this case, the syslog message has a field called "**timestamp**" that contains the timestamp you should use. You want to expose this field as the table's "**event_time**" column. To do this, click on the Event Time tab and enter the following properties:
   - Use Kafka Timestamps: **Disable**
   - Input Timestamp Column: **timestamp**
   - Event Time Column: **event_time**
   - Watermark Seconds: **3**

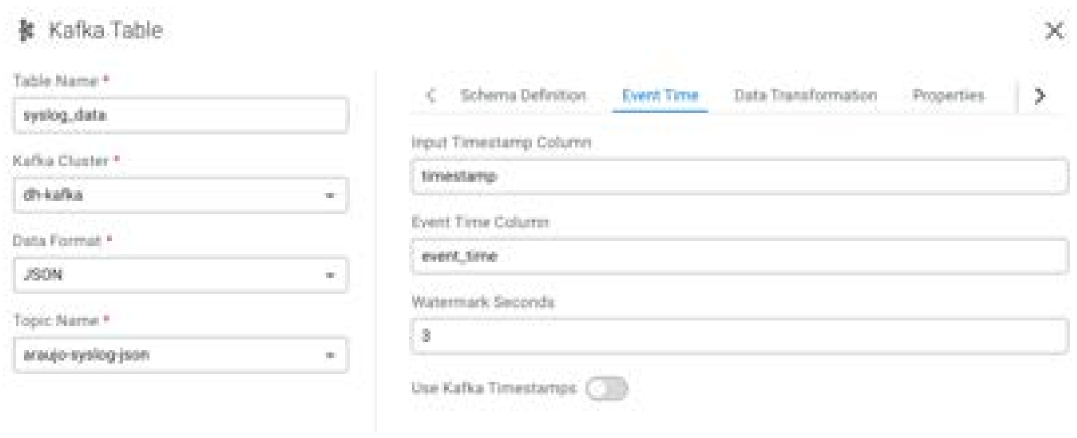

Kafka Table

Table Name *
syslog_data

Kafka Cluster *
dh-kafka

Data Format *
JSON

Topic Name *
araujo-syslog-json

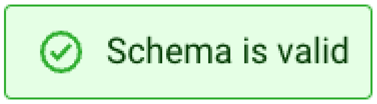
Input Timestamp Column
timestamp

Event Time Column
event_time

Watermark Seconds
3

Use Kafka Timestamps

7. Now that you have configured the event time column, click on **Detect Schema** again. You should see the schema turn valid:

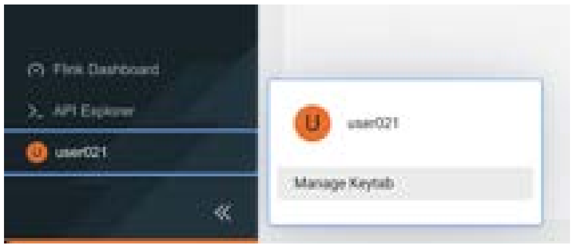

Schema is valid

8. Click the **Create and Review** button to create the table.
9. Review the table's DDL and click **Close**.

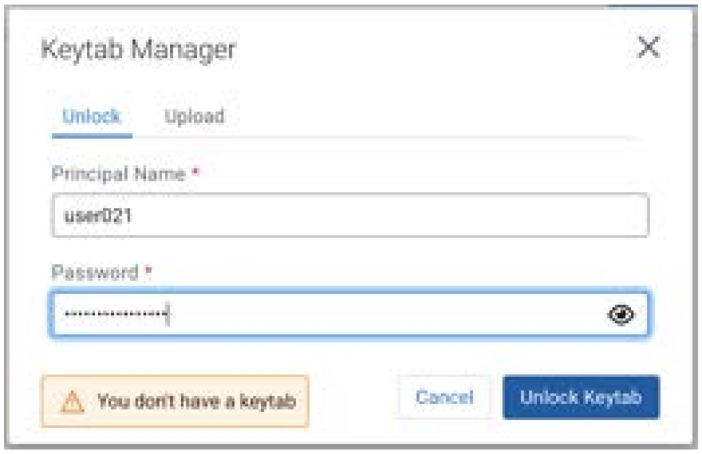
8. Unlock your keytab

Before running jobs in SSB you must unlock your keytab following the steps below:

1. Ensure you are at the SSB home page. If not in the home page, click the Projects link at the top of the screen
2. Click on your user's icon/name at the bottom-left of the screen and select **Manage Keytab**:



3. Enter your username and password and click on **Unlock Keytab**:



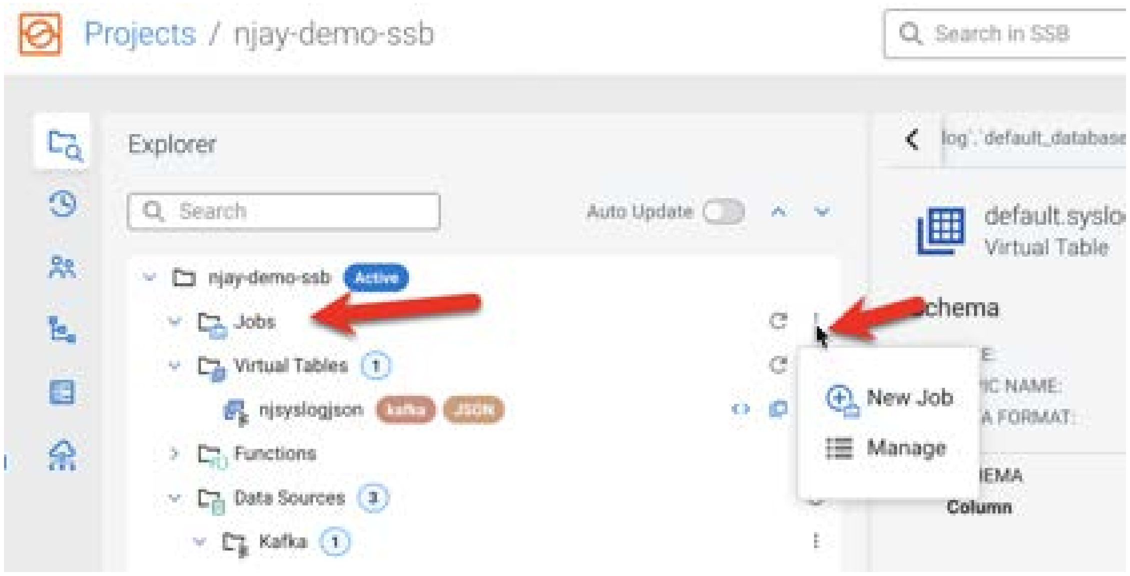
9. Create Streaming SQL jobs

In this section you will practice creating and executing SQL Streaming jobs to implement different types of use cases. You will create the following jobs:

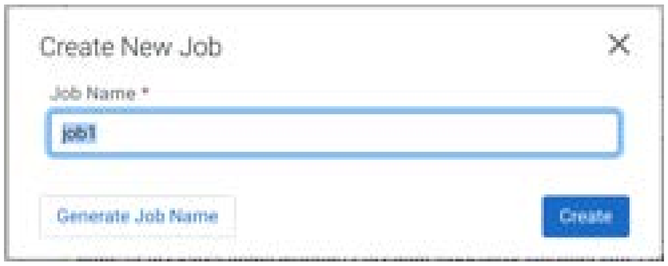
- Job 1: Selecting records from a Kafka virtual table
- Job 2: Selecting records from a Schema Registry virtual table
- Job 3: Using a windowing function
- Job 4: Enriching streaming data with data-at-rest data sources

9.1. Job 1: Selecting records from a Kafka virtual table

1. In the SSB console, ensure you have switched to your project and you are at your project's main page.
2. On the Explorer view, click the kebab menu (⋮) of the **Jobs** item and select **New Job** to create a new one.

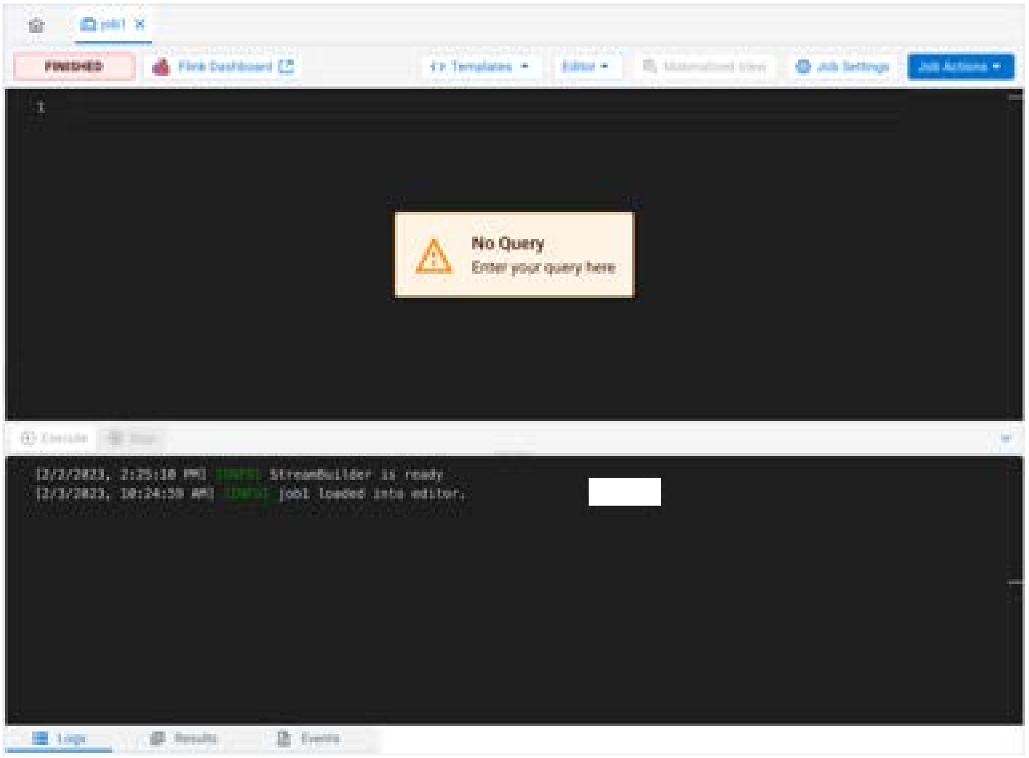


3. Enter **job1** as a **Job Name** and click on the **Create** button:



You are redirected to the SQL Editor where you can perform a number of tasks, including:

a. Compose and execute SQL queries
b. Create and manage Materialized Views
c. Configure your SQL job



No Query
Enter your query here

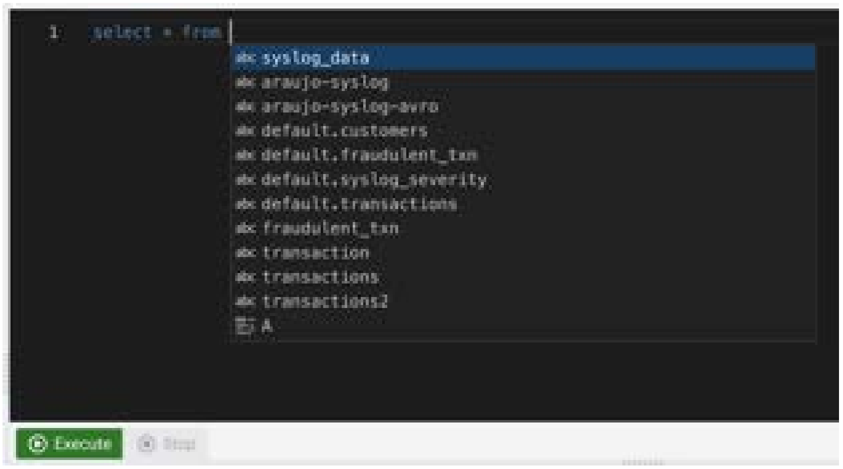
```
[2/2/2023, 2:25:10 PM] [INFO] StreamBuilder is ready
[2/3/2023, 10:24:59 AM] [INFO] job1 loaded into editor.
```

4. Click on the SQL editor area and type the job query.

   In this first job you will simply select everything from the **syslog_data** table that you created previously. Notice that the editor has auto-completion for keywords, table names, etc. As you type you will see suggestions for completion being shown. To select one of the options, simply use the arrow keys to scroll to the option and press **<ENTER>**.

   If the list of options doesn't not show at a particular position of the cursor you can press <CTRL>+<SPACE> to show the list of completion suggestions.

5. In the editor type "**SELECT * FROM** " (with a space in the end) and then press <CTRL>+<SPACE>. Select the syslog_data table from the list.

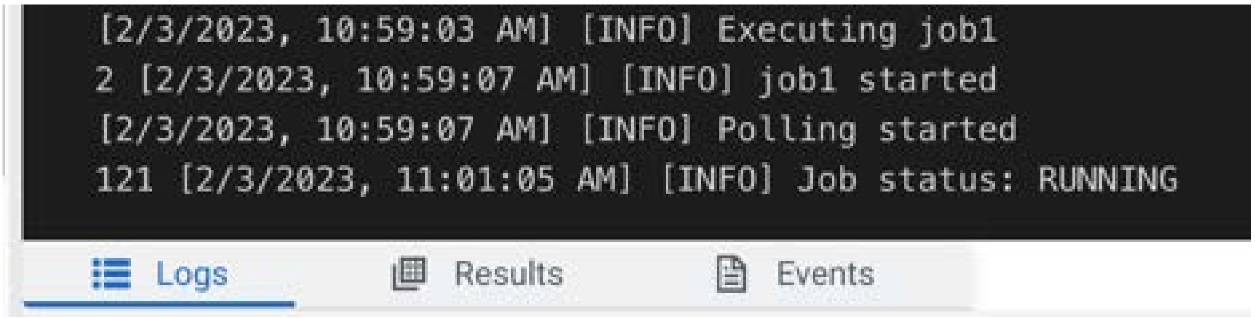


```
select * from
```

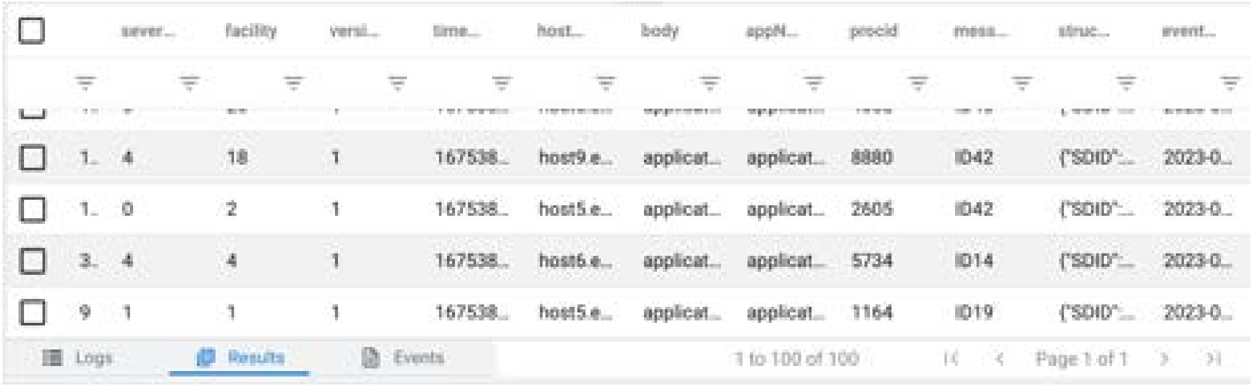
```
syslog_data
araujo-syslog
araujo-syslog-avro
default.customers
default.fraudulent_txn
default.syslog_severity
default.transactions
fraudulent_txn
transaction
transactions
transactions2
A
```

6. Click **Execute**.

   The Logs tab will show that the job execution has started and once data is retrieved it will automatically switch to the Results tab to show the results:



```
[2/3/2023, 10:59:03 AM] [INFO] Executing job1
2 [2/3/2023, 10:59:07 AM] [INFO] job1 started
[2/3/2023, 10:59:07 AM] [INFO] Polling started
121 [2/3/2023, 11:01:05 AM] [INFO] Job status: RUNNING
```

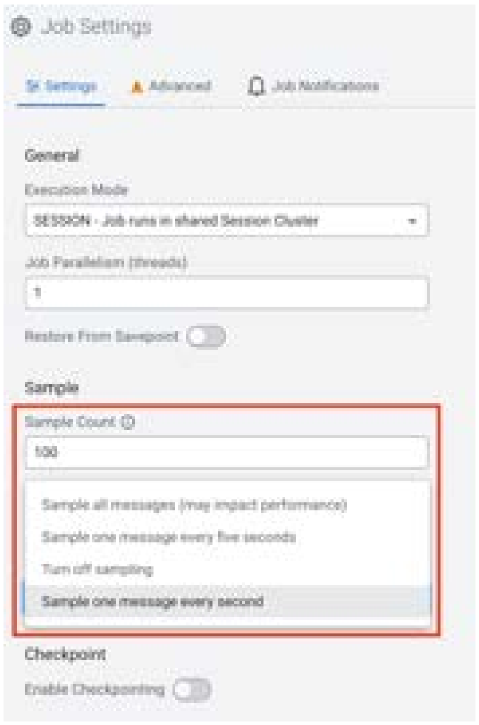


	sever...	facility	versi...	time...	host...	body	appN...	procid	mesa...	struc...	event...
1..	4	18	1	167538...	host9.e...	applicat...	applicat...	8880	ID42	{"SDID"...	2023-0...
1..	0	2	1	167538...	host5.e...	applicat...	applicat...	2605	ID42	{"SDID"...	2023-0...
3..	4	4	1	167538...	host6.e...	applicat...	applicat...	5734	ID14	{"SDID"...	2023-0...
9	1	1	1	167538...	host5.e...	applicat...	applicat...	1164	ID19	{"SDID"...	2023-0...

7. You will notice that the Results tab shows approximately one record every second, even though there is a lot more data flowing through the Kafka topic. This is because, by default, SSB only shows a sample of the data on the screen to give the user some feedback about what the query is retrieving. If you leave the query running you will also

notice that the sample polling will automatically stop once 100 records have been retrieved.

You can change the sample behavior and other settings by clicking on the Job Settings button, as shown below:



General

Execution Mode: SESSION - Job runs in shared Session Cluster

Job Parallelism (threads): 1

Restore From Savepoint

Sample

Sample Count: 100

- Sample all messages (may impact performance)
- Sample one message every five seconds
- Turn off sampling
- Sample one message every second

Checkpoint

Enable Checkpointing

NOTE: Selecting "Sample all messages" can have a negative effect in performance and will increase the resource consumption of your browser as well. Use this only when necessary to check or troubleshoot data.

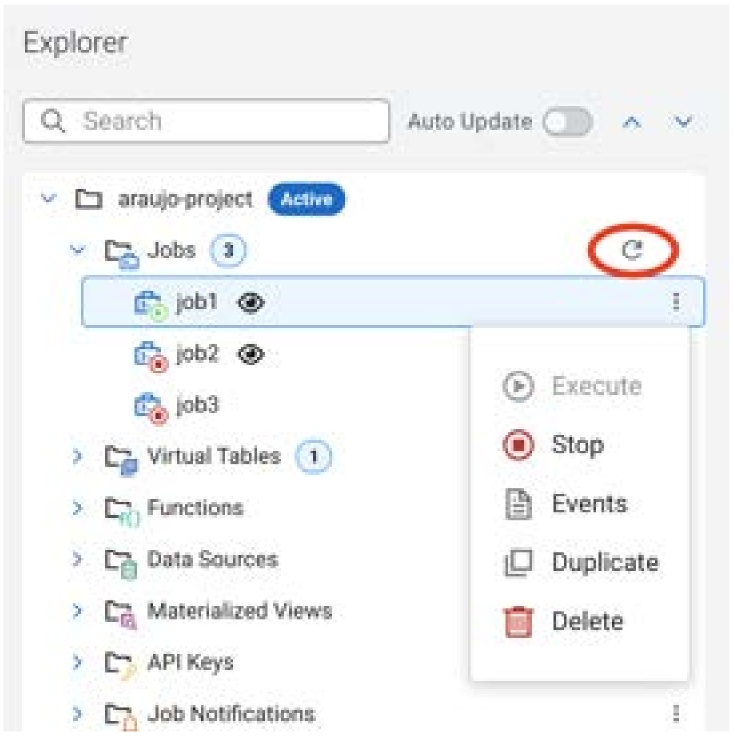
8. From the Results tab you can see that the data contains records of all severities. If you want to change that you can simply stop and edit your query and run it again.

   Click **Stop** to stop the job and edit the query so that it retrieves only records with severity 3:

   ```
   SELECT *
   FROM syslog_data
   WHERE severity = 3
   ```

   Click **Execute** and check the results to confirm that the change was effective.

9. Once you're done, **Stop** your job. In the Explorer tree you can see all the jobs and which ones are running. You can also stop jobs from there by right-clicking on the job item. If the job status doesn't look correct, try click on the Reload icon, shown below:



9.2. Job 2: Selecting records from a Schema Registry virtual table

Once virtual tables are created in SSB, their backend and data format are transparent to the user. Tables can be queried in SSB in exactly the same way. In this section you will execute the same query from job1 against a table that comes from an Kafka topic with AVRO format and will see that everything works the same:

1. On the Explorer view, right-click on the **Jobs** item and select **New Job** to create a new job. Call it **job2**.

2. Enter the same query from the previous job, but using the **<userid>-syslog-avro** table (use the auto-completion to find the correct table name - **<CTRL>+<SPACE>**). You will notice that the auto-completion will prefix the table name with the catalog and database names:

```
SELECT *
FROM [auto-complete the table <userid>-syslog-avro]
WHERE severity = 3
```

Click **Execute** and check the results to verify that everything works as expected.

3. Once you're done, **Stop** your job.

9.3. Job 3: Using a windowing function

One powerful feature of Flink and SSB is the ability to perform windowing aggregations on streaming data. For more information on all the aggregation options available, please check the Flink documentation about table-valued functions, window aggregation, group aggregation and over aggregation.

In this section you will write a query that defines a sliding (HOP) window on the syslog stream to count how many events of each severity are being generated in intervals of 30 seconds. The 30-second window slides forward every 5 seconds, so the query produces results every 5 seconds for the previous window that ended at that point in time.

1. On the Explorer view, right-click on the **Jobs** item and select **New Job** to create a new job. Call it **job3**.

2. Enter the following query in the SQL editor:

```
SELECT
  window_start, window_end,
  severity,
  count(*) as severity_count
FROM
  TABLE(
    HOP(TABLE syslog_data,
        DESCRIPTOR(event_time),
        INTERVAL '5' SECOND,
        INTERVAL '30' SECOND))
GROUP BY
  window_start, window_end, severity
```

Click **Execute** and check the results to verify that everything works as expected.

Tip: If you're not seeing all the counts for all the severities for each one of the aggregation windows, try setting the sample behavior to "Sample all messages"

3. Once you're done, **Stop** your job.

9.4. Job 4: Enriching streaming data with data-at-rest data sources

Another powerful feature of Flink/SSB is that you can write queries joining tables that have different backends. A Kafka streaming table, for example, can be joined with batch tables stored in relational databases, Kudu or even HDFS.

This opens a number of possibilities like using data-at-rest tables to enrich streaming data. In this section you will create a job that enriches the aggregation stream produced by the query from job3 by joining it with the **syslog_severity** table, which is stored in Kudu and has the definition of each one of the severity levels.

1. On the Explorer view, right-click on the **Jobs** item and select **New Job** to create a new job. Call it **job4**.

2. Enter the following query in the SQL editor:

```
SELECT
  a.window_start, a.window_end,
  a.severity,
  b.severity_desc,
  count(*) as severity_count
FROM
  TABLE(
    HOP(TABLE syslog_data,
        DESCRIPTOR(event_time),
        INTERVAL '5' SECOND,
        INTERVAL '30' SECOND)) a
  JOIN [auto-complete the table default.syslog_severity] b
    ON a.severity = b.severity
GROUP BY
  a.window_start, a.window_end, a.severity, b.severity_desc
```

   Click **Execute** and check the results to verify that everything works as expected.

3. Once you're done, **Stop** your job.

10. Create a Production job sending data to Kafka

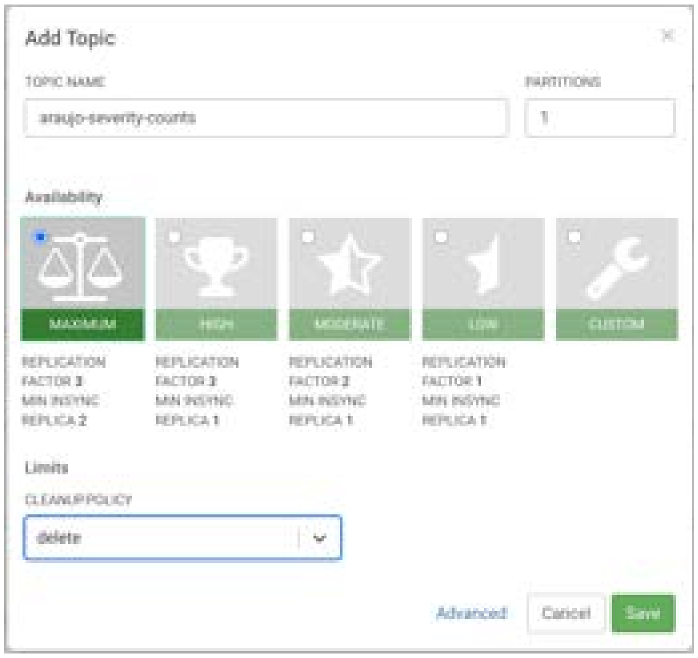
So far, the results of all the jobs that you created were displayed on the screen but not saved anywhere. When you run jobs in production you will want to send the results somewhere (Kafka, databases, Kudu, HDFS, S3, etc.)

SSB makes it really easy to write data to different locations. All you have to do is to create a Virtual Table mapped to that location and then INSERT into that table the results of the query that you are running. You will practice this in this section by modifying job 4 to send its results to a Kafka topic.

You will start by creating a topic to store the results of your query and then modify job4 so that it sends its aggregated and enriched stream output to that topic.

1. Open the SMM UI (link is on the Streams Messaging DataHub page)
2. Click on the **Topics icon** (🛢) and then on the **Add New** button to create a new topic

3. Enter the following properties for your topic:
    a. Topic Name: **<userid>-severity-counts**
    b. Partitions: **1**
    c. Availability: **MAXIMUM**
    d. Cleanup Policy: **delete**



Add Topic

TOPIC NAME: araujo-severity-counts

PARTITIONS: 1

Availability: [✓ MAXIMUM] HIGH MODERATE LOW CUSTOM

MAXIMUM	HIGH	MODERATE	LOW
REPLICATION FACTOR 3 MIN INSYNC REPLICA 2	REPLICATION FACTOR 3 MIN INSYNC REPLICA 1	REPLICATION FACTOR 2 MIN INSYNC REPLICA 1	REPLICATION FACTOR 1 MIN INSYNC REPLICA 1

Limits

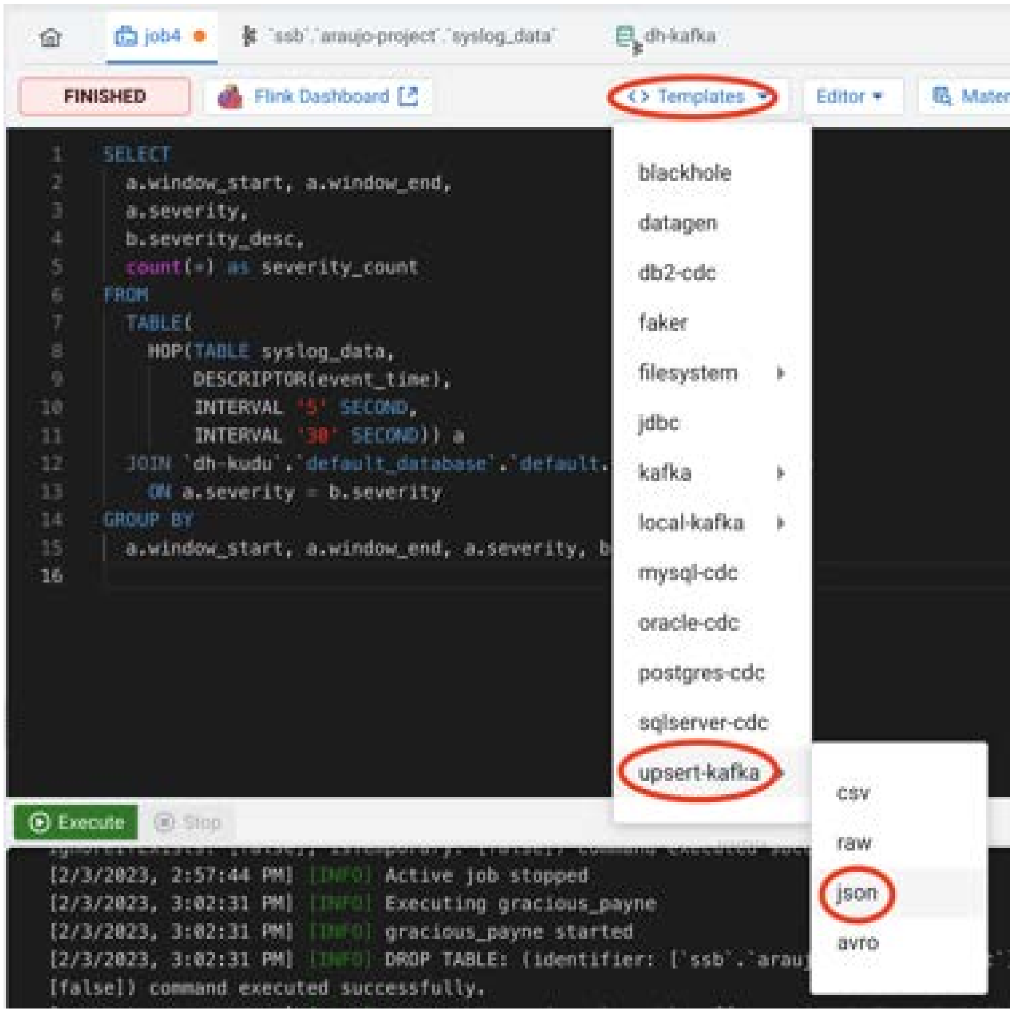
CLEANUP POLICY: delete

4. Back in the SSB UI, select **job4** in the Explorer pane to open the SQL editor for it. You should see the aggregation query that you had created in the previous section.

5. Before you can run this job in production you need to create a Virtual Table in SSB and map it to the **<userid>-severity-counts** Kafka topic that was created above.

    The structure/schema of this table must match the structure of the output of the query that is being executed. Thankfully, SSB has a template feature that makes this very easy to do.

Click on the **Templates** button above the SQL editor and select **upsert-kafka > json**.



```
SELECT
  a.window_start, a.window_end,
  a.severity,
  b.severity_desc,
  count(*) as severity_count
FROM
  TABLE(
    HOP(TABLE syslog_data,
        DESCRIPTOR(event_time),
        INTERVAL '5' SECOND,
        INTERVAL '30' SECOND)) a
  JOIN `dh-kudu`.`default_database`.`default.
    ON a.severity = b.severity
GROUP BY
  a.window_start, a.window_end, a.severity, b
```

```
[2/3/2023, 2:57:44 PM] [INFO] Active job stopped
[2/3/2023, 3:02:31 PM] [INFO] Executing gracious_payne
[2/3/2023, 3:02:31 PM] [INFO] gracious_payne started
[2/3/2023, 3:02:31 PM] [INFO] DROP TABLE: (identifier: [`ssb`.`arau
[false]) command executed successfully.
```

You will notice that a **CREATE TABLE** template was inserted at the beginning of the editor. Scroll up and down to check the contents of the editor. **Please do NOT execute it yet**.

6. The added template is a CREATE TABLE statement to create a Virtual Table of the type you selected. Before you can execute it you have to complete the template with the missing information.

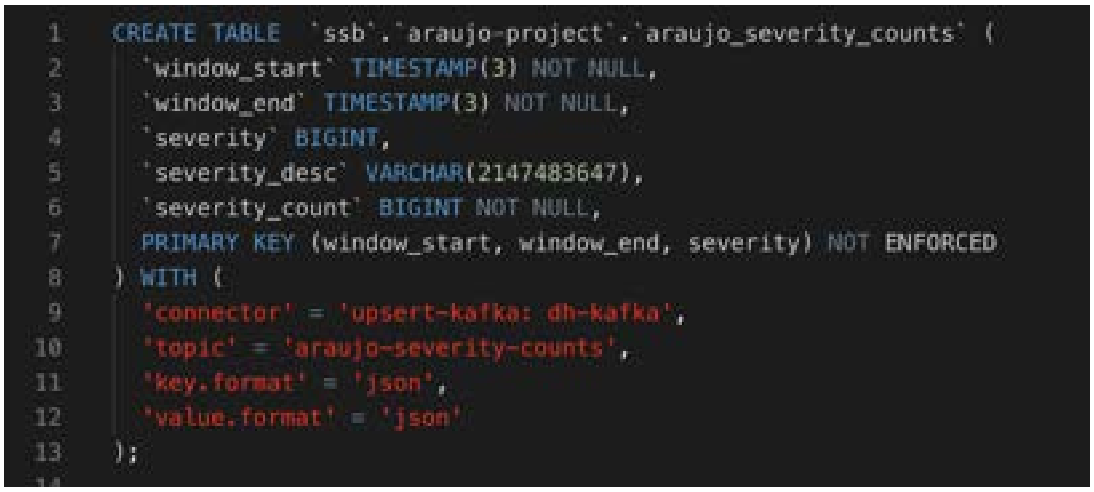
   Make the following changes to the template:
   a. Table name: change the table name to **<userid>_severity_counts**
   b. Primary key: add the following primary key clause after the last column in the table (severity_count):

c. Properties: replace all the properties in the WITH clause with the following:

```
'connector' = 'upsert-kafka: <data_source_name>',
'topic' = '<userid>-severity-counts',
'key.format' = 'json',
'value.format' = 'json'
```

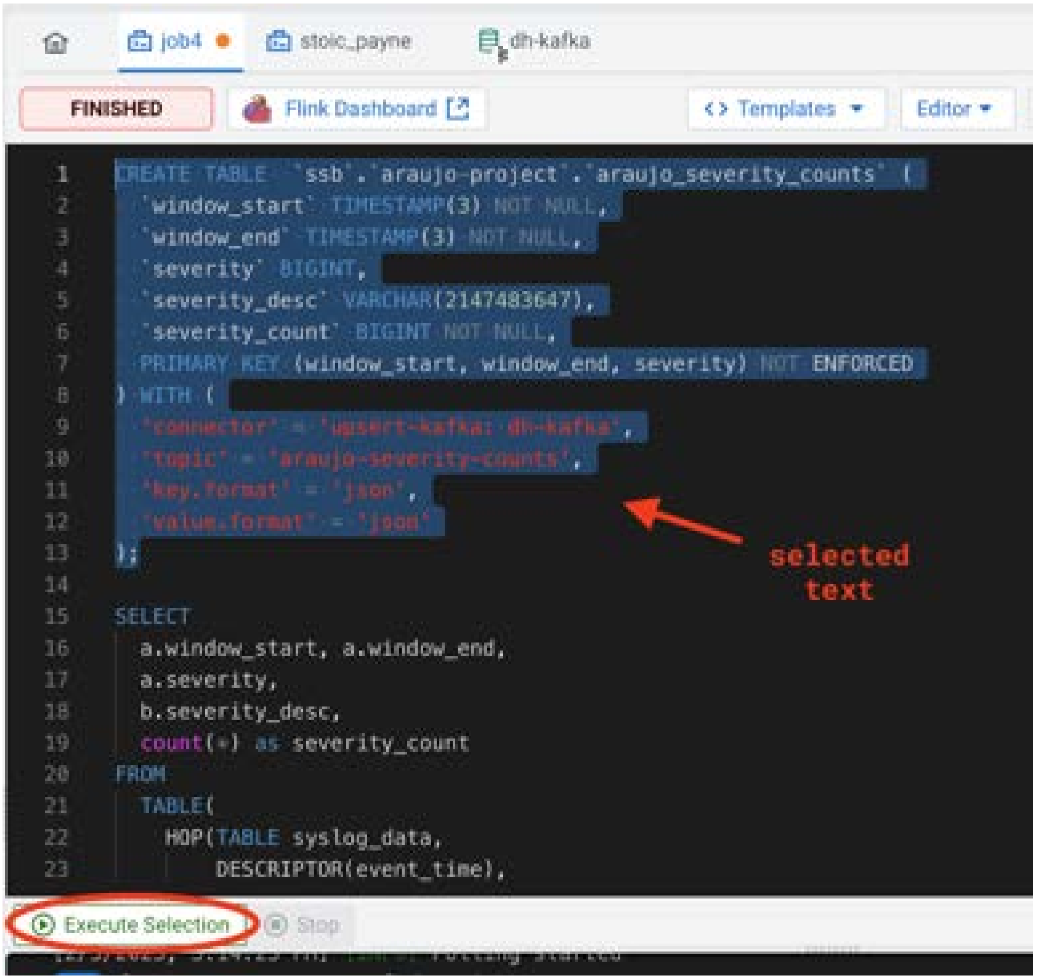
d. topic: **<userid>-severity-counts**

There are other optional properties commented out in the template that you can ignore. The completed template should look like the one below:



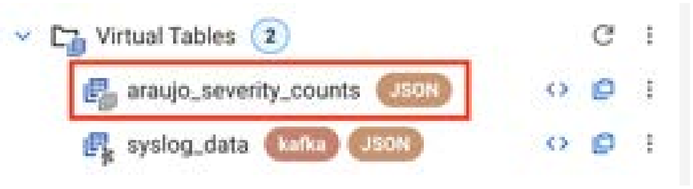
```
CREATE TABLE  `ssb`.`araujo-project`.`araujo_severity_counts` (
  `window_start` TIMESTAMP(3) NOT NULL,
  `window_end` TIMESTAMP(3) NOT NULL,
  `severity` BIGINT,
  `severity_desc` VARCHAR(2147483647),
  `severity_count` BIGINT NOT NULL,
  PRIMARY KEY (window_start, window_end, severity) NOT ENFORCED
) WITH (
  'connector' = 'upsert-kafka: dh-kafka',
  'topic' = 'araujo-severity-counts',
  'key.format' = 'json',
  'value.format' = 'json'
);
```

7. With your mouse, select only the text of the entire CREATE TABLE statement and click on the **Execute Selection** button to run only that statement and create the Virtual Table.



```sql
CREATE TABLE `ssb`.`araujo-project`.`araujo_severity_counts` (
  `window_start` TIMESTAMP(3) NOT NULL,
  `window_end` TIMESTAMP(3) NOT NULL,
  `severity` BIGINT,
  `severity_desc` VARCHAR(2147483647),
  `severity_count` BIGINT NOT NULL,
  PRIMARY KEY (window_start, window_end, severity) NOT ENFORCED
) WITH (
  'connector' = 'upsert-kafka: dh-kafka',
  'topic' = 'araujo-severity-counts',
  'key.format' = 'json',
  'value.format' = 'json'
);

SELECT
  a.window_start, a.window_end,
  a.severity,
  b.severity_desc,
  count(*) as severity_count
FROM
  TABLE(
    HOP(TABLE syslog_data,
        DESCRIPTOR(event_time),
```

After the CREATE TABLE execution you can see the created table in the Explorer tree:

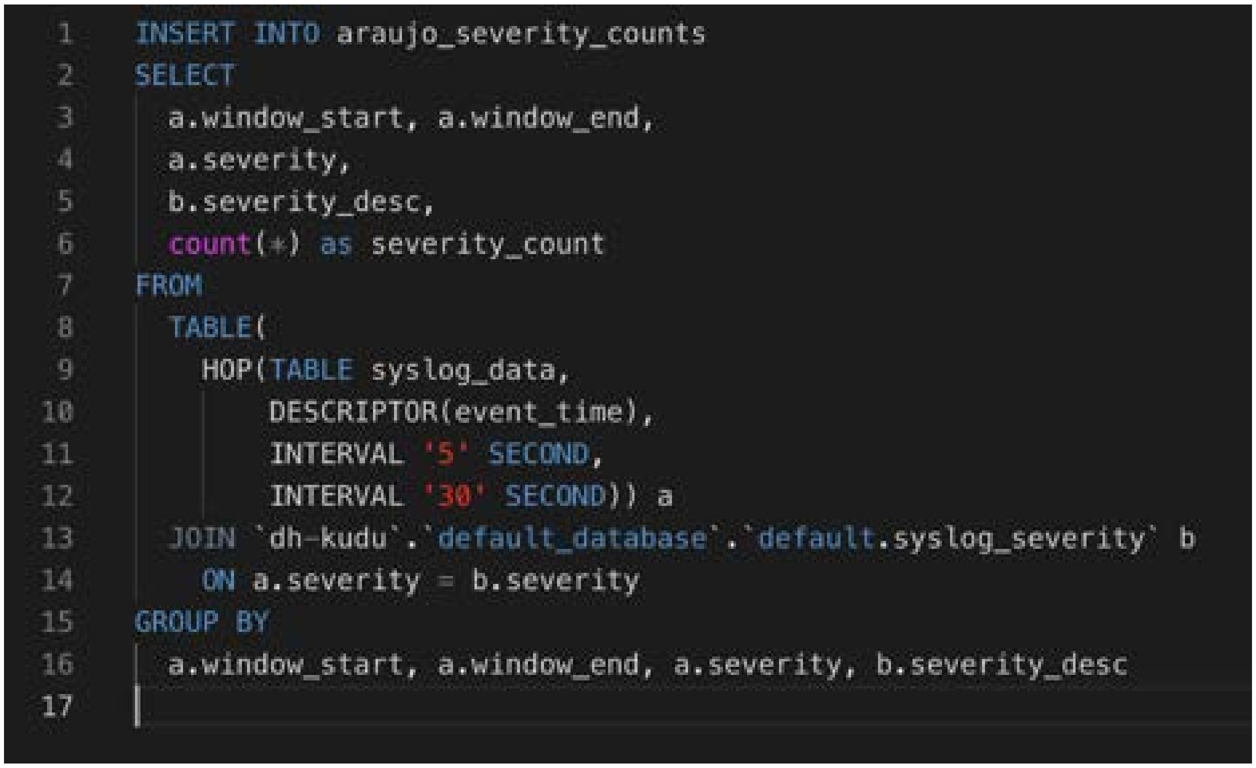


8. Once the table is created successfully, delete the CREATE TABLE statement from the SQL editor.

9. Modify the original query in the editor by adding the following line before the SELECT keyword:

```
INSERT INTO `<userid>_severity_counts`
```

Your final statement should look like this:



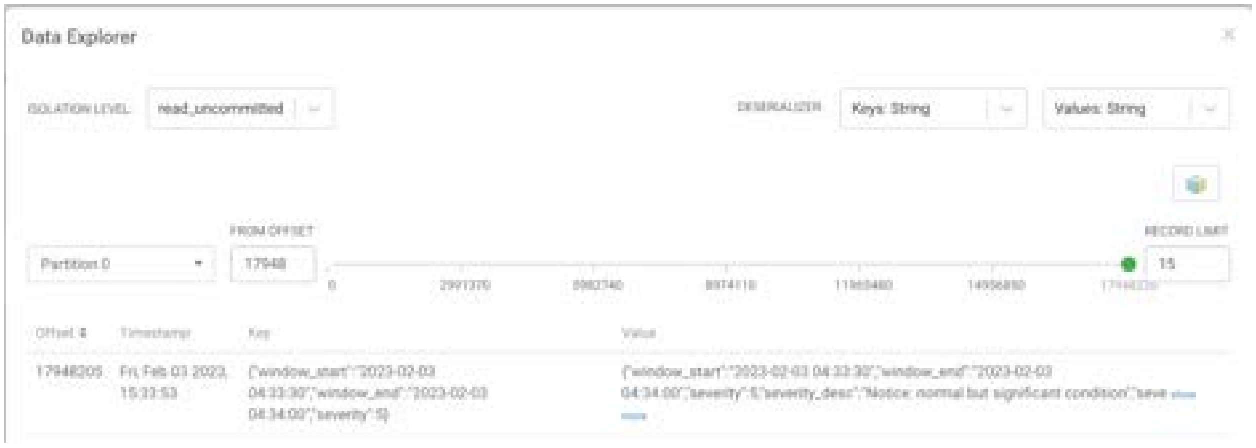
```
INSERT INTO araujo_severity_counts
SELECT
  a.window_start, a.window_end,
  a.severity,
  b.severity_desc,
  count(*) as severity_count
FROM
  TABLE(
    HOP(TABLE syslog_data,
        DESCRIPTOR(event_time),
        INTERVAL '5' SECOND,
        INTERVAL '30' SECOND)) a
  JOIN `dh-kudu`.`default_database`.`default.syslog_severity` b
    ON a.severity = b.severity
GROUP BY
  a.window_start, a.window_end, a.severity, b.severity_desc

```

10. Click on the **Execute** button to submit your job for execution.

11. You should see the output of the query on the Results tab once the job starts executing. You can close the tab or window at any time and the job will continue running on the Flink cluster.

12. Open the SMM UI again, click on the **Topics icon** (🗄) and search for the **<userid>-severity-counts** topic.

13. Click on the Data Explorer icon (🔍) for the topic to visualize the data in the topic. You should be able to see recently added data with the aggregations produced by **job4**:



Data Explorer

ISOLATION LEVEL: read_uncommitted

DESERIALIZER: Keys: String | Values: String

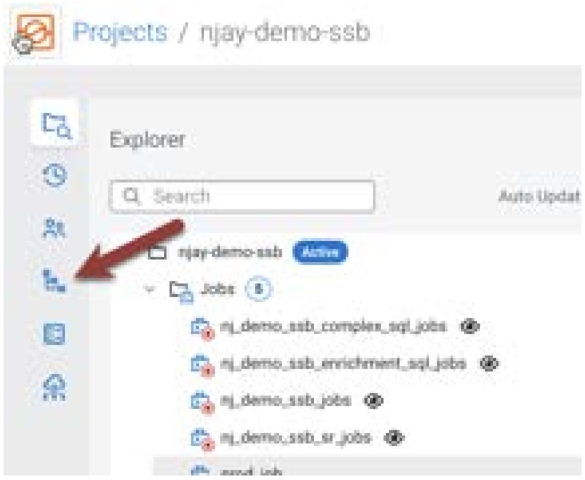
Partition 0 | FROM OFFSET: 17948 | RECORD LIMIT: 15

Offset	Timestamp	Key	Value
17948205	Fri, Feb 03 2023, 15:33:53	{"window_start":"2023-02-03 04:33:30","window_end":"2023-02-03 04:34:00","severity":5}	{"window_start":"2023-02-03 04:33:30","window_end":"2023-02-03 04:34:00","severity":5,"severity_desc":"Notice: normal but significant condition","seve

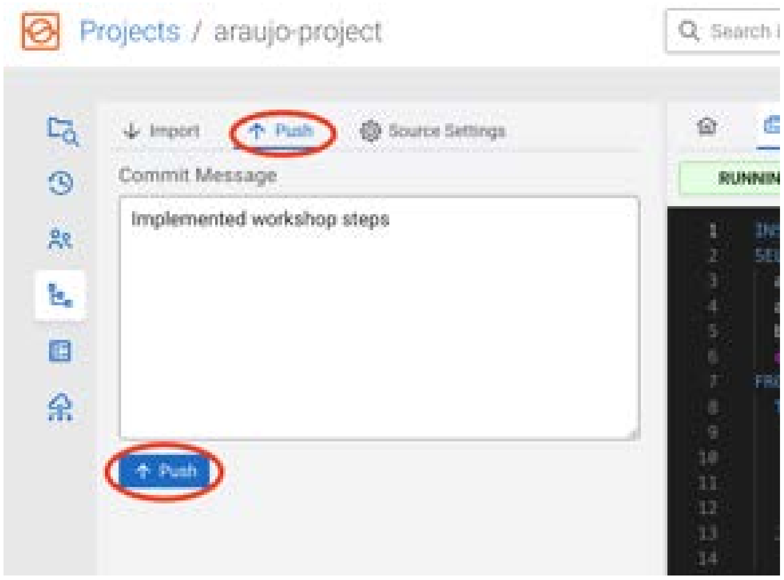
11. Versioning your project changes into GitHub

Now that you have completed the changes to your project, at least for now, you can commit the changes and version that into your GitHub repository.

1. Click on the Source Control icon, as shown below:

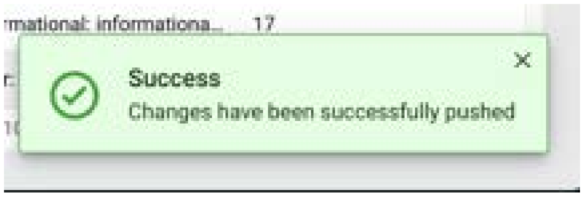


2. Click on the **Push** tab at the top, enter your commit message and click on the **Push** button.

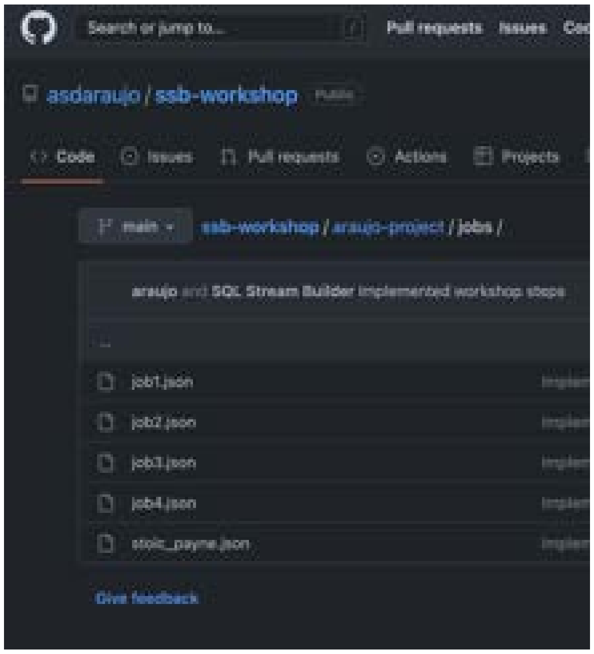


Implemented workshop steps

If your push is successful you should see the following message:


Success
Changes have been successfully pushed

3. Verify the github repository. Changes will be pushed to the remote repository.


asdaraujo / ssb-workshop

main ssb-workshop / araujo-project / jobs /

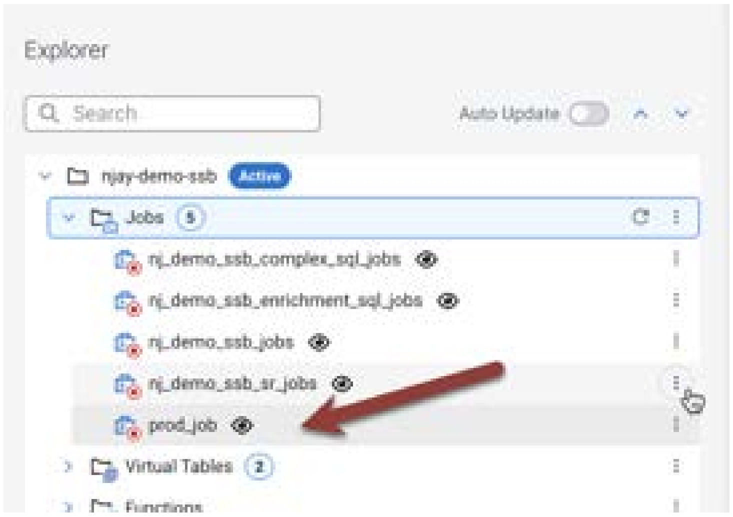
araujo and SQL Stream Builder implemented workshop steps

job1.json
job2.json
job3.json
job4.json
stoic_payne.json

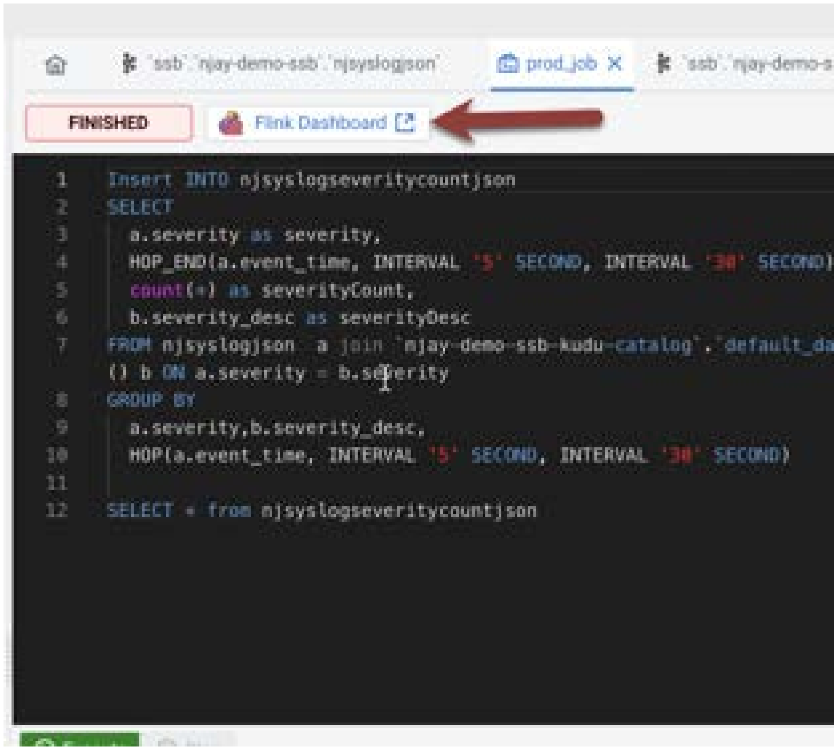
12. Troubleshooting Flink/SSB jobs

If you have problems with running your jobs, follow the steps below to check job details in SSB and on the Flink Dashboard.

1. Click on your job (under Jobs on the left bar) to open the job editor:

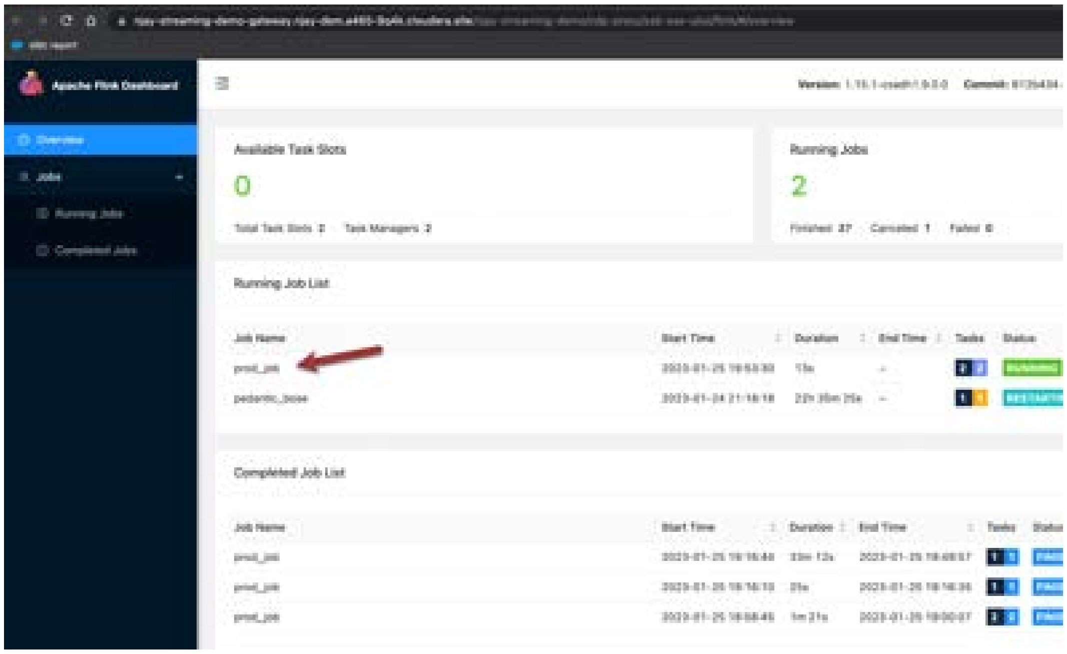


2. Click on the **Flink Dashboard** link to open the Flink Dashboard for job



```
Insert INTO njsyslogseveritycountjson
SELECT
  a.severity as severity,
  HOP_END(a.event_time, INTERVAL '5' SECOND, INTERVAL '30' SECOND)
  count(*) as severityCount,
  b.severity_desc as severityDesc
FROM njsyslogjson  a join `njay-demo-ssb-kudu-catalog`.`default_da
() b ON a.severity = b.severity
GROUP BY
  a.severity,b.severity_desc,
  HOP(a.event_time, INTERVAL '5' SECOND, INTERVAL '30' SECOND)

SELECT * from njsyslogseveritycountjson
```

3. Navigate the dashboard pages to explore details and metrics of the job execution:



Apache Flink Dashboard

Version: 1.15.1-csadh1.9.0.0 Commit: 8123434

Available Task Slots

0

Total Task Slots 2 Task Managers 2

Running Jobs

2

Finished 27 Canceled 1 Failed 0

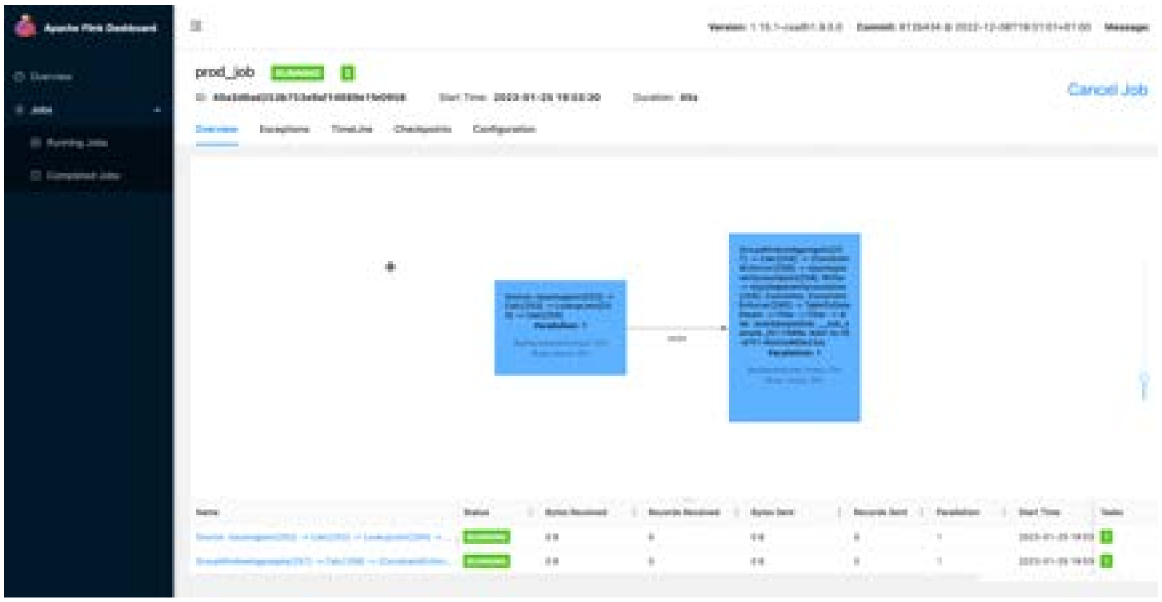
Running Job List

Job Name	Start Time	Duration	End Time	Tasks	Status
prod_job	2023-01-25 19:53:30	13s	-	2 2	RUNNING
pedantic_jesse	2023-01-24 21:18:18	22h 35m 25s	-	1 1	RESTARTI

Completed Job List

Job Name	Start Time	Duration	End Time	Tasks	Status
prod_job	2023-01-25 19:16:44	33m 12s	2023-01-25 19:49:57	1 1	FINI
prod_job	2023-01-25 19:16:10	25s	2023-01-25 19:16:35	1 1	FINI
prod_job	2023-01-25 18:58:45	1m 21s	2023-01-25 19:00:07	2 2	FINI

4. Browse the job's DAG. Click on each of the operators (blue boxes) to see metrics and details of each one, including their individual tasks.



Version: 1.15.1-csadh1.9.0.0 Commit: 8123434 @ 2022-12-08T19:51:01+01:00 Message:

prod_job RUNNING

ID: 40a3d6ad2126753e0af14606e1fe0908 Start Time: 2023-01-25 19:53:30 Duration: 40s

Cancel Job

Name	Status	Bytes Received	Records Received	Bytes Sent	Records Sent	Parallelism	Start Time	Tasks
Source: [illegible] -> Calc[253] -> LookupJoin[255] -> ...	RUNNING	0 B	0	0 B	0	1	2023-01-25 19:53	1
GroupWindowAggregate[257] -> Calc[258] -> ConstraintEnfor...	RUNNING	0 B	0	0 B	0	1	2023-01-25 19:53	1