

Best iBest in Flow Competition Tutorials

Author: Michael Kohs George Vetticaden Timothy Spann

Date: 04/18/2023

Last Updated: 4/27/2023

Useful Data Assets

Setting Your Workload Password

Creating a Kafka Topic

Use Case walkthrough	9
1. Reading and filtering a stream of syslog data	9
2. Writing critical syslog events to Apache Iceberg for analysis	29
3. Resize image flow deployed as serverless function	56

Use Case Walkthrough for Competition

Notice

This document assumes that you have registered for an account, activated it and logged into the CDP Sandbox. This is for authorized users only who have attended the webinar and have read the training materials.

A short guide and references are listed [here](#).

Competition Resources

Login to the Cluster

<https://login.cdpworkshops.cloudera.com/auth/realms/se-workshop-5/protocol/saml/clients/cdp-sso>

Kafka Broker connection string

- oss-kafka-demo-corebroker2.oss-demo.qsm5-opic.cloudera.site:9093,
- oss-kafka-demo-corebroker1.oss-demo.qsm5-opic.cloudera.site:9093,
- oss-kafka-demo-corebroker0.oss-demo.qsm5-opic.cloudera.site:9093

Kafka Topics

- syslog_json
- syslog_avro
- syslog_critical

Schema Registry Hostname

- oss-kafka-demo-master0.oss-demo.qsm5-opic.cloudera.site

Schema Name

- syslog
- syslog_avro
- syslog_transformed

Syslog Filter Rule

- SELECT * FROM FLOWFILE WHERE severity <= 2

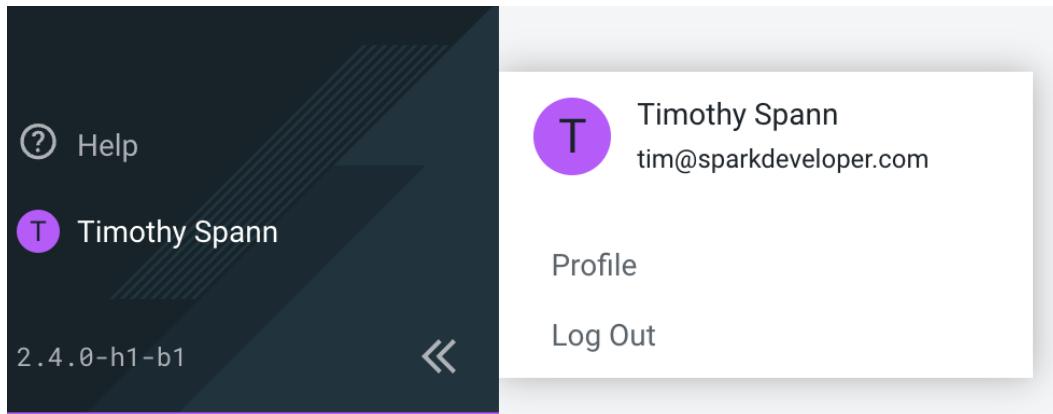
Access Key and Private Key for Machine User in DataFlow Function

- Access Key: eda9f909-d1c2-4934-bad7-95ec6e326de8
- Private Key: eon6eFzLlxZI/gpU0dWht21DI60MkSQZlzeWSGBSI=

The following keys are needed if you want to deploy a DataFlow Function that you build during the Best in Flow Competition.

Your Workflow User Name and Password

1. Click on your name at the bottom left corner of the screen for a menu to pop up.



2. Click on *Profile* to be redirected to your user's profile page with important information.

The screenshot shows the Cloudera Management Console interface. On the left is a dark sidebar with various navigation options: Dashboard, Environments, Data Lakes, **User Management** (which is currently selected), Data Hub Clusters, Data Warehouses, ML Workspaces, Classic Clusters, Audit, and Shared Resources. The main content area is titled "Users / Timothy Spann". It displays a table of user information:

Name	Timothy Spann
Email	tim@sparkdeveloper.com
Workload User Name	tim
CRN	crn:altus:iam:us-west-1:5251b921-84c5-45c4-af51-c0b8a6ebd1c9:user:b8f9... Copy
Tenant ID	5251b921-84c5-45c4-af51-c0b8a6ebd1c9 Copy
Identity Provider	se-workshop-5-keycloak-idp
Last Interactive Login	04/19/2023 8:14 AM EDT
Profile Management	View profile
Workload Password	Set Workload Password (Workload password is currently set)

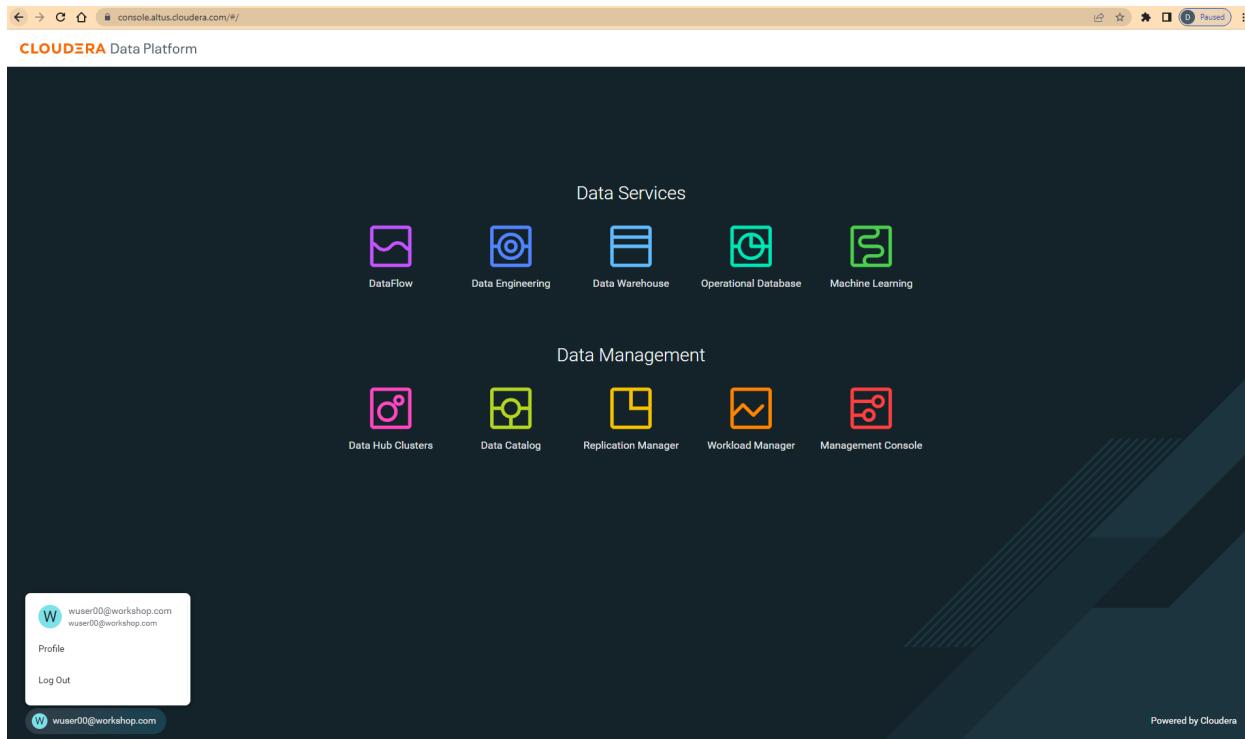
Below the table, there is a navigation bar with links: Access Keys (which is underlined to indicate it's active), Roles, Resources, Groups, and SSH Keys.

If your Workload Password does not say currently set or you forgot it, follow the steps below to reset it. Your **userid** is shown above at **Workload User Name**.

Setting Workload Password

You will need to define your workload password that will be used to access non-SSO interfaces. You may read more about it [here](#). Please keep it with you. If you have forgotten it, you will be able to repeat this process and define another one.

1. From the **Home Page**, click on your User Name (Ex: `tim`) at the lower left corner.
2. Click on the **Profile** option.



1. Click option **Set Workload Password**.
2. Enter a suitable Password and Confirm Password.
3. Click the button **Set Workload Password**.

The screenshot shows the Cloudera Management Console interface. The left sidebar includes options like Dashboard, Environments, Data Lakes, User Management (which is selected), Data Hub Clusters, Data Warehouses, ML Workspaces, Classic Clusters, Audit, Shared Resources, and Global Settings. The main content area displays user details for "wuser00@workshop.com".

Name	wuser00@workshop.com
Email	wuser00@workshop.com
Workload User Name	wuser00
CRN	cm:altus:iam:us-west-1:d1a4553c-a799-432d-8e54-372cc2ab95f2:user:a03...
Tenant ID	d1a4553c-a799-432d-8e54-372cc2ab95f2
Identity Provider	partner-workshops
Last Interactive Login	02/28/2023 9:21 AM +04
Profile Management	View profile
Workload Password	Set Workload Password (Workload password is currently set)

Below the user details, there are tabs for Access Keys, Roles, Resources, Groups, and SSH Keys. Under the Access Keys tab, it says "No access keys found." and has a "Generate Access Key" button. The bottom of the sidebar includes Help and the user profile again.

Users / wuser00@workshop.com / Workload Password

● Password

● Confirm Password

If you use keytabs, you need to regenerate them after changing your workload password. You can do this from your user profile > Actions > Get Keytab.

Set Workload Password

Check that you got the message - Workload password is currently set or alternatively, look for a message next to Workload Password which says (Workload password is currently set). Save the password you configured as well as the workload user name for use later.

Users / wuser00@workshop.com

Name	wuser00@workshop.com
Email	wuser00@workshop.com
Workload User Name	wuser00
CRN	cm.altusiam.us-west-1:d1a4553c-a799-432d-8e54-372cc2ab95f2:users:a03...
Tenant ID	d1a4553c-a799-432d-8e54-372cc2ab95f2
Identity Provider	partner-workshops
Last Interactive Login	02/28/2023 9:21 AM +04
Profile Management	View profile
Workload Password	(Workload password is currently set)

Access Keys Roles Resources Groups SSH Keys

No access keys found.

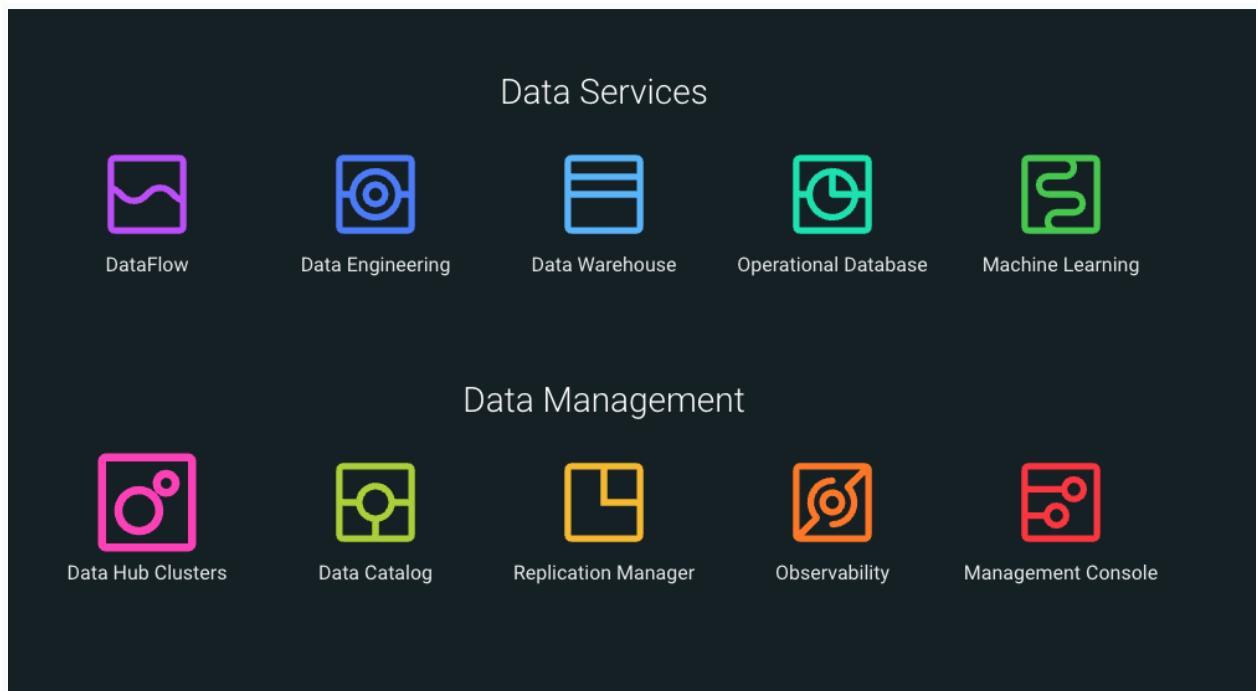
Generate Access Key

Create a Kafka Topic

The tutorials require you to create an Apache Kafka topic to send your data to, this is how you can create that topic. You will also need this information to create topics for any of your own custom applications for the competition.

1. Navigate to Data Hub Clusters from the Home Page

Info: You can always navigate back to the home page by clicking the app switcher icon at the top left of your screen.



2. Navigate to the **oss-kafka-demo** cluster

The screenshot shows the Cloudera Management Console Data Hubs page. It lists four running clusters:

Status	Name	Cloud Provider	Environment	Data Hub Type	Version	Node Count	Created
Running	oss-flink-demo	aws	oss-demo-aws	7.2.16 - Streaming Analytics Light Duty with Apache Flink	CDH 7.2.16	6	03/21/23, 03:28 PM EDT
Running	oss-kafka-demo	aws	oss-demo-aws	7.2.16 - Streams Messaging Light Duty: Apache Kafka, Schema Registry, Streams Messaging Manager, Streams Replication Manager, Cruise Control	CDH 7.2.16	4	03/21/23, 03:29 PM EDT
Running	oss-kudu-demo	aws	oss-demo-aws	7.2.16 - Real-time Data Mart: Apache Impala, Hue, Apache Kudu, Apache Spark	CDH 7.2.16	7	03/21/23, 03:29 PM EDT
Running	oss-nifi-demo	aws	oss-demo-aws	7.2.16 - Flow Management Light Duty with Apache Nifi, Apache NiFi Registry	CDH 7.2.16	4	03/21/23, 03:29 PM EDT

Streams Messaging Manager

3. Navigate to Streams Messaging Manager

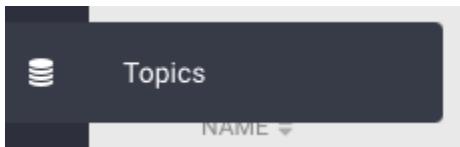
The screenshot shows the Cloudera Management Console with the 'oss-kafka-demo' cluster selected. The top navigation bar includes 'Data Hubs / oss-kafka-demo / Event History'. Below the navigation is a summary card for the cluster, showing 'STATUS: Running', 'NODES: 4', 'CREATED AT: 03/21/23, 03:20 PM EDT', 'CLUSTER TEMPLATE: 7.2.16 - Streams Messaging Light Duty: Apache Kafka, Schema Registry, Streams Messaging Manager, Streams Replication Manager, Cruise Control', and 'STATUS REASON: Synced instance states with the cloud provider'. The main content area is titled 'oss-kafka-demo' and contains sections for 'Environment Details', 'Services' (with tabs for CM UI, Schema Registry, Streams Messaging Manager, and Token Integration), and 'Cloudera Manager Info'. The 'Event History' tab is active, displaying a log of events from 3/22/2023, including logs for Cloudera Manager, COP services, and various pre-flight and post-flight checks. A download link for the logs is available at the bottom.

Info: Streams Messaging Manager (SMM) is a tool for working with Apache Kafka.

4. Now that you are in SMM.

The screenshot shows the Streams Messaging Manager (SMM) Overview page. At the top, there are four tabs: 'Producers' (14), 'Topics' (35), 'Brokers' (3), and 'Consumer Groups' (6). The 'Topics' tab is active. The main content area displays a table for topics, with columns for NAME, DATA IN, DATA OUT, MESSAGES IN, CONSUMER GROUPS, and CURRENT LOG SIZE. The table lists various topics such as '_consumer_offsets', '_CruiseControlMetrics', '_KafkaCruiseControlModelTrainingSamples', '_KafkaCruiseControlPartitionMetricSamples', '_smm_alert_notifications', '_smm_consumer_metrics', '_smm_producer_metrics', '_smm-app-consumer-metrics-keys-index-chan', '_smm-app-producer-metrics-keys-index-chang', '_smm-app-smm-consumer-table-15m-changelog', '_smm-app-smm-consumer-table-15m-repartition', and '_smm-app-smm-consumer-table-30s-changelog'. To the right of the table, there is a sidebar for 'Consumer Groups' with tabs for ACTIVE (6), PASSIVE (0), and ALL (6). The sidebar lists consumer groups: 'smm-service-status' (LAG 8), '_smm-app' (LAG 3), and 'cdf' (LAG 0).

5. Navigate to the round icon third from the top, click this **Topic** button.

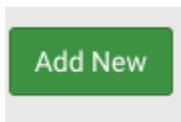


6. You are now in the Topic browser.

Topics

NAME	BYTES IN	BYTES OUT	PRODUCED PER SEC	FETCHED PER SEC	IN SYNC REPLICAS	OUT OF SYNC	UNDER REPLICATED	OFFLINE PARTITIONS
	DATA IN #	DATA OUT #		MESSAGES IN #	CONSUMER GROUPS #		CURRENT LOG SIZE # (B)	
Topics (34)								
smm-app-smm-producer-table-30s-repartition	11 KB	11 KB	3	237	1	0	445 KB	
smm-app-smm-producer-table-15m-changelog	10 KB	0B		236	0		512 KB	
heartbeats	172 KB	0B		1.8k	0		1 MB	
smr-service-status-connector-metrics-minutes-store-changelog	18 KB	0B		90	0		6 MB	
tim_syslog_critical	0B	0B		0	0		N/A	
smm-app-smm-producer-table-15m-repartition	11 KB	11 KB	3	237	1	0	445 KB	
smr-service-cluster-metrics-minutes-store-changelog	0B	0B		0	0		0B	
smm-app-smm-consumer-table-15m-repartition	0B	0B		0	1		0B	
smm-app-smm-consumer-table-30s-repartition	0B	0B		0	1		0B	
smm_consumer_metrics	0B	0B		0	1		0B	
KafkaCruiseControlPartitionMetricSamples	171 KB	0B		8.9k	0		2 MB	
smr-metrics.secondary.internal	0B	0B		0	1		0B	
smm-app-smm-consumer-table-15m-changelog	0B	0B		0	0		0B	

7. Click **Add New** to build a new topic.



8. Enter the name of your topic prefixed with your Workload User Name, ex: <>replace_with_userid>>_syslog_critical.

Add Topic

X

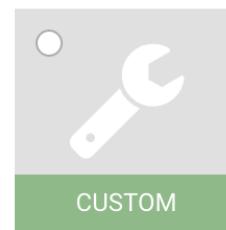
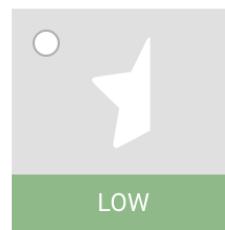
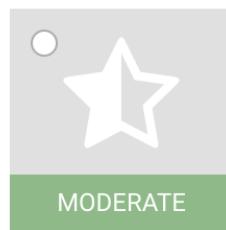
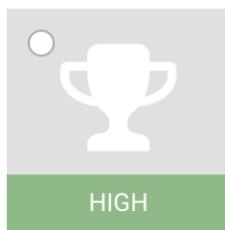
TOPIC NAME

tim_syslog_critical

PARTITIONS

3

Availability



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



Advanced

Cancel

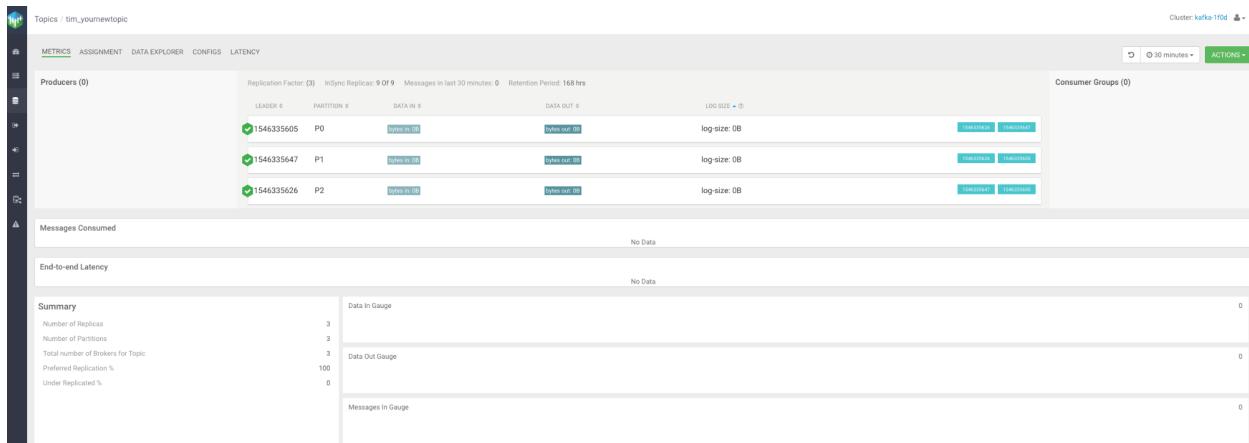
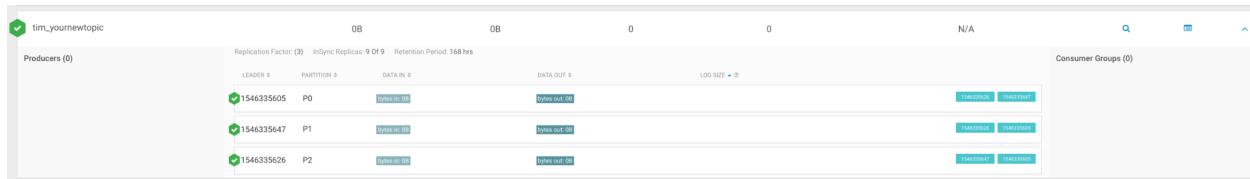
Save

9. For settings you should create it with **(3 partitions, cleanup.policy: delete, availability maximum)** as shown above.

After successfully creating a topic, close the tab that opened when navigating to Streams Messaging Manager

Congratulations! You have built a new topic.



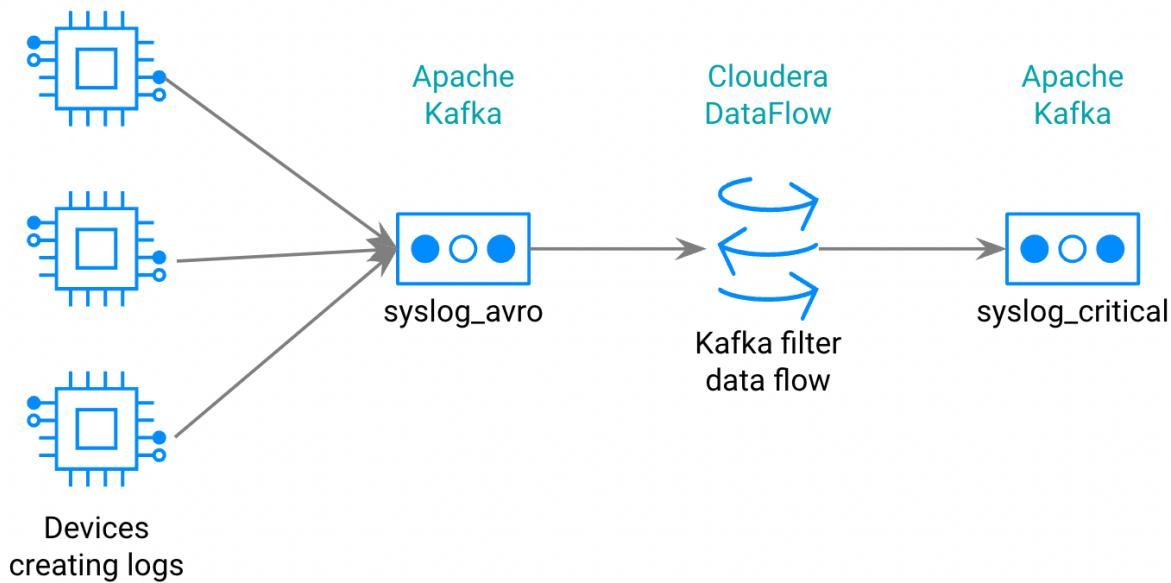


10. After successfully creating a topic, close the tab that opened when navigating to Streams Messaging Manager

1. Reading and filtering a stream of syslog data

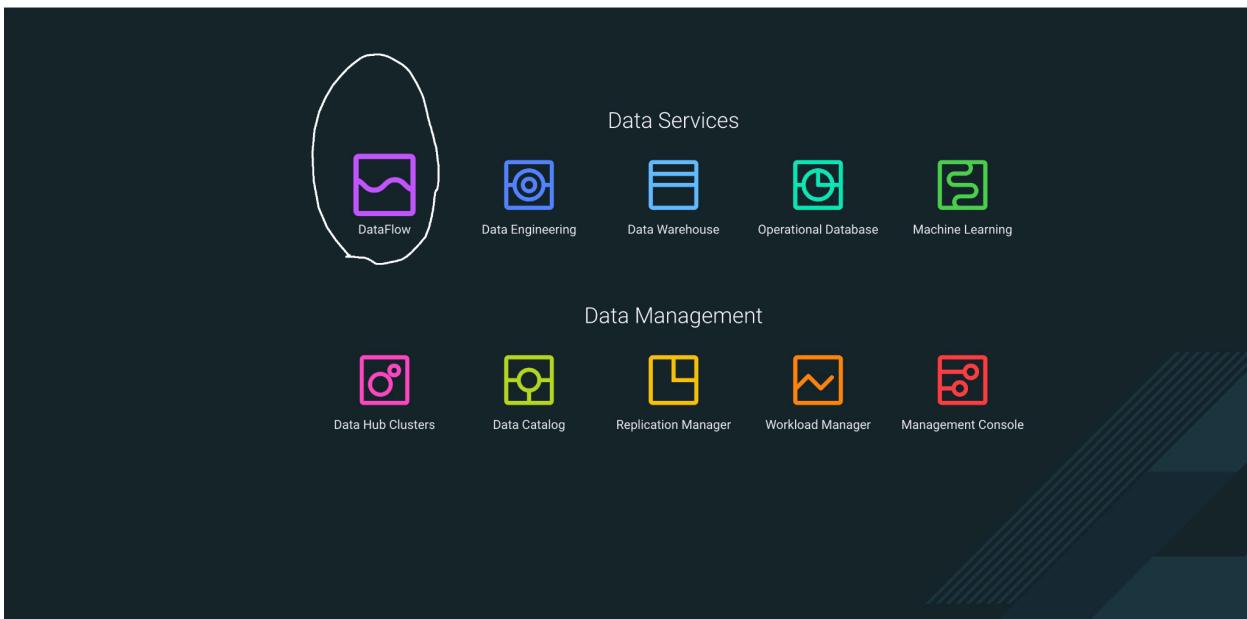
You have been tasked with filtering a noisy stream of syslog events which are available in a Kafka topic. The goal is to identify critical events and write them to the Kafka topic you just created.

Related documentation is [here](#).



1.1 Open ReadyFlow & start Test Session

1. Navigate to **DataFlow** from the Home Page



2. Navigate to the **ReadyFlow Gallery**
3. Explore the ReadyFlow Gallery

Info:

The ReadyFlow Gallery is where you can find out-of-box templates for common data movement use cases. You can directly create deployments from a ReadyFlow or create new drafts and modify the processing logic according to your needs before deploying.

4. Select the “Kafka filter to Kafka” ReadyFlow.
5. Get your user id from your profile, it is usually the first part of your email, so my email is tim@sparkdeveloper.com so my user id is **tim**. It is your “**Workload User Name**” that you are going to need for several things, remember that.
6. You already created a new **topic** to receive data in the setup section.
<>replace_with_userid>>_syslog_critical Ex: tim_syslog_critical.
7. Click on “Create New Draft” to open the ReadyFlow in the Designer with the name **youruserid_kafkafilterkafka**, for example **tim_kafkafilterkafka**. If your name has periods, underscores or other non-alphanumeric characters just leave those out. Select from the available workspaces in the dropdown, you should only have one available.

Create New Draft

Select the target workspace

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

Selected Flow Definition

NAME	VERSION
Kafka filter to Kafka	1

Workspace [?](#)

aws oss-demo-aws	60% (3 of 5)
------------------	--------------

Draft Name

✓ Draft name is valid

[Cancel](#) [Create](#)

8. Start a Test Session by either clicking on the *start a test session* link in the banner or going to *Flow Options* and selecting *Start* in the Test Session section.
9. In the Test Session creation wizard, select the latest NiFi version and click *Start Test Session*. Leave the other options to its default values. Notice how the status at the top now says “Initializing Test Session”.

Info:

Test Sessions provision infrastructure on the fly and allow you to start and stop individual processors and send data through your flow. By running data through processors step by step and using the data viewer as needed, you’re able to validate your processing logic during development in an iterative way without having to treat your entire data flow as one deployable unit.

1.2 Modifying the flow to read syslog data

The flow consists of three processors and looks very promising for our use case. The first processor reads data from a Kafka topic, the second processor allows us to filter the events before the third processor writes the filtered events to another Kafka topic.

All we have to do now to reach our goal is to customize its configuration to our use case.

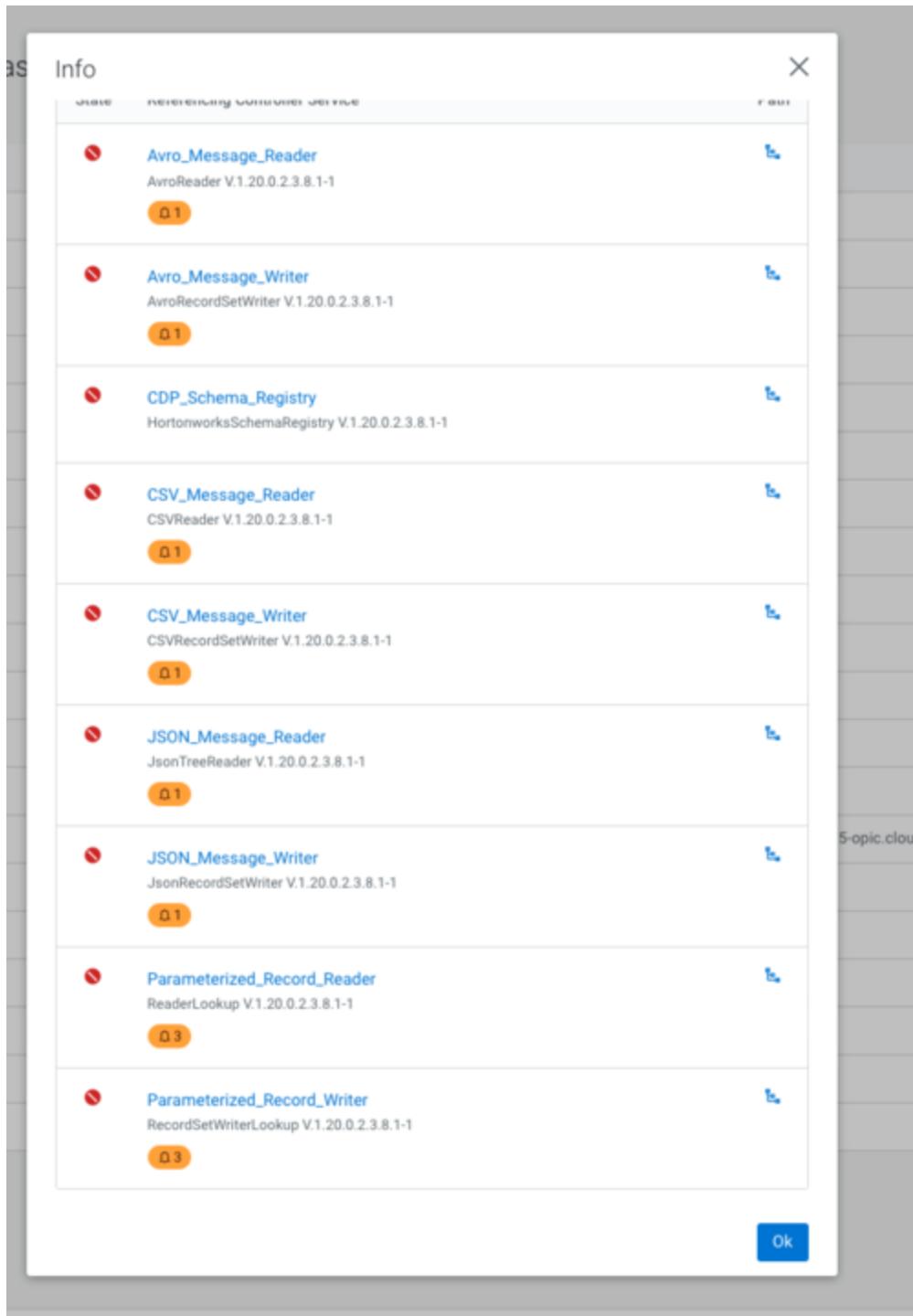
1. Provide values for predefined parameters

- a. Navigate to *Flow Options*→ *Parameters*
- b. For some settings there are some that are set already as parameters, for others they are not, you can set them manually. Make sure you create a parameter for the **Group Id**.
- c. Configure the following parameters:

Name	Description	Value
CDP Workload User	CDP Workload User	<Your own workload user ID that you saved when you configured your workload password>
CDP Workload User Password	CDP Workload User Password	<Your own workload user password you configured in the earlier step>
Filter Rule	Filter Rule	SELECT * FROM FLOWFILE WHERE severity <= 2
Data Input Format		AVRO
Data Output Format		JSON
Kafka Consumer Group ID	ConsumeFromKafka	<>replace_with_userid>>_cdf Ex: tim_cdf
Group ID	ConsumeFromKafka	<>replace_with_userid>>_cdf Ex: tim_cdf
Kafka Broker Endpoint	Comma-separated list of Kafka Broker addresses	oss-kafka-demo-corebroker2.oss-demo.qsm5-opic.cloud era.site:9093, oss-kafka-demo-corebroker1.oss-demo.qsm5-opic.cloud era.site:9093, oss-kafka-demo-corebroker0.oss-demo.qsm5-opic.cloud era.site:9093
Kafka Destination Topic	Must be unique	<>replace_with_userid>>_syslog_critical Ex: tim_syslog_critical
Kafka Producer ID	Must be unique	<>replace_with_userid>>_cdf_producer1

		Ex: tim_cdf_producer1
Kafka Source Topic		syslog_avro
Schema Name		syslog
Schema Registry Hostname	Hostname from Kafka cluster	oss-kafka-demo-master0.os-s-demo.qsm5-opic.cloudera.site

- d. Click *Apply Changes* to save the parameter values
- e. If confirmation is requested, Click *Ok*.



2. Start Controller Services

- Navigate to *Flow Options* → *Services*
- Select *CDP_Schema_Registry* service and click *Enable Service and Referencing Components* action. If this is not enabled, it may be an error or an extra space in any of the parameters for example **AVRO** must not have a new line or blank spaces. The first thing to try if you have an issue is to stop the Design

environment and then restart the test session. Check the Tips guide for more help or contact us in the bestinflow.slack.com.

Service and Referencing Con



- c. Start from the top of the list and enable all remaining Controller services
- d. Make sure all services have been enabled. You may need to reload the page or try it in a new tab.

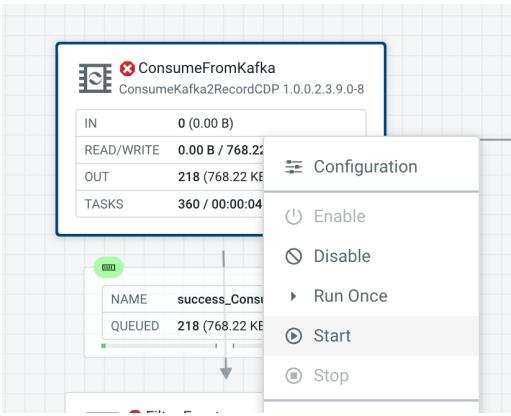
All Services for Kafka filter to Kafka

[+ Add Service](#)

State	Name ↑	Type	Notifications
✓	Avro_Message_Reader	AvroReader V.1.20.0.2.3.9.0-8	>
✓	Avro_Message_Writer	AvroRecordSetWriter V.1.20.0.2.3.9.0-8	>
✓	CDP_Schema_Registry	HortonworksSchemaRegistry V.1.20.0.2...	>
✓	CSV_Message_Reader	CSVReader V.1.20.0.2.3.9.0-8	>
✓	CSV_Message_Writer	CSVRecordSetWriter V.1.20.0.2.3.9.0-8	>
✓	Default NiFi SSL Context Service	StandardRestrictedSSLContextService ...	>
✓	JSON_Message_Reader	JsonTreeReader V.1.20.0.2.3.9.0-8	>
✓	JSON_Message_Writer	JsonRecordSetWriter V.1.20.0.2.3.9.0-8	>
✓	Parameterized_Record_Reader	ReaderLookup V.1.20.0.2.3.9.0-8	>
✓	Parameterized_Record_Writer	RecordSetWriterLookup V.1.20.0.2.3.9.0...	>

- 3. If your processors have all **started** because you started your controller services, it is best to stop them all by right clicking on each one and clicking 'Stop' and then start them one at a time so you can follow the process easier. **Start the *ConsumeFromKafka processor*** using the right click action menu or the *Start* button in the configuration

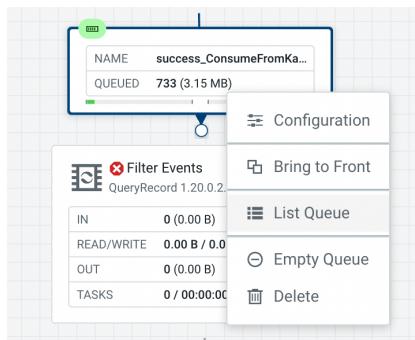
drawer.



After starting the processor, you should see events starting to queue up in the *success_ConsumeFromKafka-FilterEvents* connection.

4. Verify data being consumed from Kafka

- Right-click on the *success_ConsumeFromKafka-FilterEvents* connection and select *List Queue*



Info:

The *List Queue* interface shows you all flow files that are being queued in this connection. Click on a flow file to see its metadata in the form of *attributes*. In our use case, the attributes tell us a lot about the Kafka source from which we are consuming the data. Attributes change depending on the source you're working with and can also be used to store additional metadata that you generate in your flow.

- b. Select any flow file in the queue and click the *book* icon to open it in the *Data Viewer*

More Details ▾

FILE SIZE 5 KB	MIME TYPE application/json
LINEAGE DURATION 00:06:32.413	QUEUE POSITION No value set
PENALIZED No	QUEUE DURATION 00:06:32.378

[Open in Data Viewer](#)

Info: The *Data Viewer* displays the content of the selected flow file and shows you the events that we have received from Kafka. It automatically detects the data format - in this case JSON - and presents it in human readable format.

- c. Scroll through the content and note how we are receiving syslog events with varying severity.

□ Data Viewer

Content

View as: Formatted / JSON

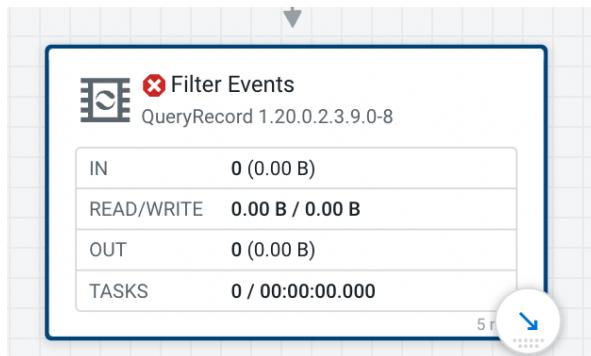
```

1 v [ {
2   "priority" : 54,
3   "severity" : 6,
4   "facility" : 6,
5   "version" : 1,
6   "timestamp" : 1680908330914,
7   "hostname" : "host8.example.com",
8   "body" : "application8 has exited cleanly",
9   "appName" : "application8",
10  "procid" : "6086",
11  "messageid" : "ID44",
12  "structuredData" : {
13    "SDID" : {
14      "eventId" : "88",
15      "eventSource" : "application",
16      "iut" : "2"
17    }
18  }
19 ]

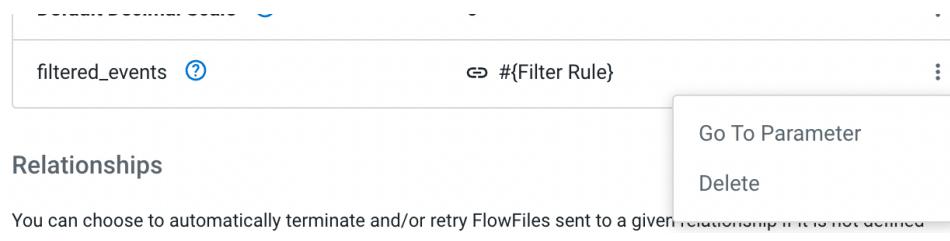
```

5. Define filter rule to filter out low severity events

- Return to the Flow Designer by closing the Data Viewer tab and clicking *Back To Flow Designer* in the *List Queue* screen.
- Select the *Filter Events* processor on the canvas. We are using a *QueryRecord* processor to filter out low severity events. The *QueryRecord* processor is very flexible and can run several filtering or routing rules at once.



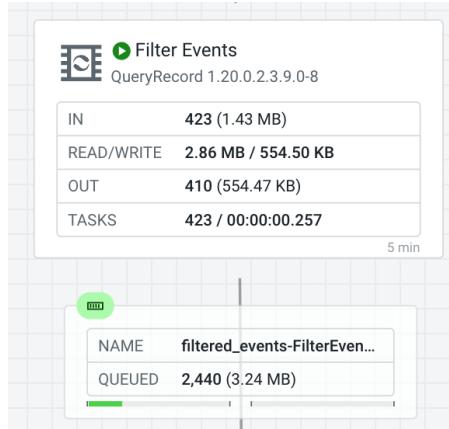
- In the configuration drawer, scroll down until you see the *filtered_events* property. We are going to use this property to filter out the events. Click on the menu at the end of the row and select *Go To Parameter*.



- If you wish to change this, you can change the Parameter value.
- Click *Apply Changes* to update the parameter value. Return back to the Flow Designer.
- Start the *Filter Events* processor using the right-click menu or the *Start* icon in the configuration drawer.

6. Verify that the filter rule works

- After starting the *Filter Events* processor, flow files will start queueing up in the *filtered_events-FilterEvents-WriteToKafka* connection



- Right click the *filtered_events-FilterEvents-WriteToKafka* connection and select *List Queue*.
- Select a few random flow files and open them in the Data Viewer to verify that only events with severity <=2 are present.

The screenshot shows the Data Viewer interface. The title bar says 'Data Viewer'. The content area displays two JSON flow files. The first file contains the following event data:

```
1 * [ {  
2     "priority" : 66,  
3     "severity" : 2,  
4     "facility" : 8,  
5     "version" : 1,  
6     "timestamp" : 1680909990832,  
7     "hostname" : "host4.example.com",  
8     "body" : "application1 has exited cleanly",  
9     "appName" : "application1",  
10    "procid" : "8932",  
11    "messageid" : "ID42",  
12    "structuredData" : {  
13        "SDID" : {  
14            "eventId" : "95",  
15            "eventSource" : "kernel",  
16            "iut" : "6"  
17        }  
18    },  
19  }, {  
20     "priority" : 0,  
21     "severity" : 0,  
22     "facility" : 0,  
23     "version" : 1,  
24     "timestamp" : 1680909990840,  
25     "hostname" : "host5.example.com",  
26     "body" : "application6 has stopped unexpectedly",  
27     "appName" : "application6",  
28     "procid" : "6003",  
29     "messageid" : "ID35",  
30     "structuredData" : {  
31         "SDID" : {  
32             "eventId" : "33",  
33             "eventSource" : "kernel",  
34             "iut" : "3"  
35         }  
36     }  
37 }
```

The second file contains similar event data with different severity levels. The 'View as:' dropdown menu at the top right is set to 'Formatted / JSON'.

- Navigate back to the Flow Designer canvas.

7. Write the filtered events to the Kafka alerts topic

Now all that is left is to start the *WriteToKafka* processor to write our filtered high severity events to *syslog_critical* Kafka topic.

- a. Select the *WriteToKafka* processor and explore its properties in the configuration drawer.
- b. Note how we are plugging in many of our parameters to configure this processor. Values like *Kafka Brokers*, *Topic Name*, *Username*, *Password* and the *Record Writer* have all been parameterized and use the values that we provided in the very beginning.
- c. Start the *WriteToKafka* processor using the right-click menu or the *Start* icon in the configuration drawer.

Congratulations! You have successfully customized this ReadyFlow and achieved your goal of sending critical alerts to a dedicated topic! Now that you are done with developing your flow, it is time to deploy it in production!

1.3 Publishing your flow to the catalog

1. Stop the Test Session

- a. Click the toggle next to *Active Test Session* to stop your Test Session



- b. Click “End” in the dialog to confirm. The Test Session is now stopping and allocated resources are being released



2. Publish your modified flow to the Catalog

- a. Open the “Flow Options” menu at the top
- b. Click “Publish” to make your modified flow available in the Catalog
- c. Prefix your username to the Flow Name and provide a Flow Description. Click **Publish**.

The screenshot shows a modal dialog titled "Publish A New Flow". It has three input fields:

- Flow Name:** vetticadentest1_kafkafilterkafka-vetticadentest1 (highlighted with a blue border)
- Flow Description:** Flow that identifies high severity events and writes them to a dedicated Kafka topic.
- Version Comments:** Initial Version

At the bottom of the dialog are two buttons: "Cancel" and "Publish" (highlighted with a blue border).

i.

- d. You are now redirected to your published flow definition in the Catalog.

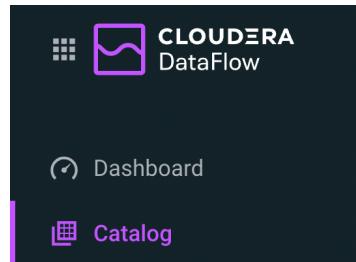
The screenshot shows the Cloudera DataFlow Catalog interface. At the top, there's a header with a refresh icon and the text "REFRESHED: 18 seconds ago". Below the header, the flow name is "vetticadentest1_kafkafilterkafka-vetticadentest1" with a subtitle "Updated a minute ago by George Vetticaden". To the right is an "Actions" dropdown menu. A "FLOW DESCRIPTION" section follows, containing a brief description: "Flow that identifies high severity events and writes them to a dedicated Kafka topic." Below this is a CRN field: "crn:cdp:df:us-west-1:5251b921-84c5-45c4-af51-c0b8a6ebd1c9:flow:vetticadentest1_kafkafilterkafka-vetticadentest1". Underneath the description is a checkbox labeled "Only show deployed versions". Below this is a table with three columns: "Version", "Deployments", and "Associated Drafts". The first row shows "1", "0", and "1" respectively. At the bottom of the table are buttons for "Deploy →", "Download", and "Create New Draft".

Info: The *Catalog* is the central repository for all your deployable flow definitions. From here you can create auto-scaling deployments from any version or create new drafts and update your flow processing logic to create new versions of your flow.

1.4 Creating an auto-scaling flow deployment

1. As soon as you publish your flow, it should take you to the Catalog. If it does not then locate your flow definition in the Catalog

- a. Make sure you have navigated to the *Catalog*



- b. If you have closed the sidebar, search for your published flow <>yourid<> into the search bar in the Catalog. Click on the flow definition that matches the name you

gave it earlier.

The screenshot shows the NiFi Flow Catalog interface. On the left is a sidebar with icons for creating new flows, managing environments, and other actions. The main area has a search bar at the top with the query 'Kafka filter to Kafka - mkohs'. Below the search is a table with a single row: 'Name ↑' followed by 'Kafka filter to Kafka - mkohs'. To the right of the table is a detailed view of the flow. It includes a 'REFRESHED: 5 seconds ago' message, the flow's name 'Kafka filter to Kafka - mkohs', its last update time 'Updated 8 minutes ago by Michael Kohs', and an 'Actions' dropdown menu. Below this is a 'FLOW DESCRIPTION' section with a brief description: 'Flow that identifies high severity alerts and writes them to a dedicated Kafka topic.' and a CRN number: 'crm:cdf:us-west-1:9d74eee4-1cad-45d7-b645-7ccf9edbb73d:flow...'. There is also a checkbox for 'Only show deployed versions'. At the bottom are buttons for 'Deploy →', 'Download', and 'Create New Draft'.

- c. After opening the side panel, click *Deploy*, select the available environment from the drop down menu and click *Continue* to start the Deployment Wizard.
- d.

The screenshot shows the 'New Deployment' wizard. The title is 'New Deployment' and there is a close button 'X'. The first section is 'Select the target environment' with a note: '① Sensitive data never leaves the environment. Changing the environment after this step requires restarting the deployment process.' The second section is 'Selected Flow Definition' which shows a table with one row: NAME 'tim_kafkafilterkafka' and VERSION '1'. The third section is 'Target Environment' which shows a dropdown menu with 'aws oss-demo-aws' selected and '60% (3 of 5)' progress. At the bottom are 'Cancel' and 'Continue →' buttons.

If you have any issues, log out, close your browser, restart your browser, try an incognito window and re-login. Also see the “Best Practices Guide”.

2. Complete the Deployment Wizard

The Deployment Wizard guides you through a six step process to create a flow deployment. Throughout the six steps you will choose the NiFi configuration of your flow,

provide parameters and define KPIs. At the end of the process, you are able to generate a CLI command to automate future deployments.

Note: The Deployment name has a cap of 27 characters which needs to be considered as you write the prod name.

- a. Provide a name such as <>your_username>>_kafkatokafka_prod to indicate the use case and that you are deploying a production flow. Click *Next*.

Overview

Deployment Name

 Deployment name is valid

Selected Flow Definition

NAME	VERSION
tim_kafkafilterkafka	1

Target Environment

aws	NAME
	oss-demo-aws

- b. The **NiFi Configuration** screen allows you to customize the runtime that will execute your flow. You have the opportunity to pick from various released NiFi versions.

Select the *Latest Version* and make sure *Automatically start flow upon successful deployment* is checked.

Click *Next*.

- c. The **Parameters** step is where you provide values for all the parameters that you defined in your flow. In this example, you should recognize many of the prefilled values from the previous exercise - including the *Filter Rule* and our *Kafka Source* and *Kafka Destination Topics*.

To advance, you have to provide values for all parameters. Select the *No Value* option to only display parameters without default values.

SHOW: Sensitive No value

You should now only see one parameter - the *CDP Workload User Password* parameter which is sensitive. Sensitive parameter values are removed when you

publish a flow to the catalog to make sure passwords don't leak.

Provide your *CDP Workload User Password* and click *Next* to continue.

Kafka filter to Kafka (12)

CDP Workload User Password ⓘ 13/100K

.....

SHOW: Sensitive No value

Cancel ← Previous Next →

- d. The **Sizing & Scaling** step lets you choose the resources that you want to allocate for this deployment. You can choose from several node configurations and turn on Auto-Scaling.

Let's choose the *Extra Small* Node Size and turn on *Auto-Scaling* from 1-3 nodes. Click *Next* to advance.

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

NiFi Node Sizing ⓘ

<input checked="" type="radio"/> Extra Small	<input type="radio"/> Small	<input type="radio"/> Medium	<input type="radio"/> Large
2 vCores Per Node 4 GB Per Node	3 vCores Per Node 6 GB Per Node	6 vCores Per Node 12 GB Per Node	12 vCores Per Node 24 GB Per Node

Number of NiFi Nodes

Auto Scaling ⓘ

Enabled

Min. Nodes: 1 - Max. Nodes: 3

Cancel ← Previous Next →

- e. The **Key Performance Indicators (KPI)** step allows you to monitor flow performance. You can create KPIs for overall flow performance metrics or

in-depth processor or connection metrics.

Add the following KPI

- KPI Scope: **Entire Flow**
- Metric to Track: **Data Out**
- Alerts:
 - Trigger alert when metric is less than: **1 MB/sec**
 - Alert will be triggered when metrics is outside the boundary(s) for: **1 Minute**

Add new KPI X

Details

KPI Scope [?](#)

Metric to Track [?](#)

METRIC DESCRIPTION:
Rate of data sent to external source in bytes

Alerts

Trigger alert when metric is greater than

Trigger alert when metric is less than

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

[Cancel](#)

Add the following KPI

- KPI Scope: **Processor**
- Processor Name: **ConsumeFromKafka**
- Metric to Track: **Bytes Received**
- Alerts:
 - Trigger alert when metric is less than: **512 KBytes/sec**
 - Alert will be triggered when metrics is outside the boundary(s) for: **30 seconds**

Add new KPI

Details

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

Metric to Track [?](#)

METRIC DESCRIPTION:
Number of bytes received from a source that is external from the flows

Alerts

Trigger alert when metric is greater than
 Trigger alert when metric is less than

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

Review the KPIs and click **Next**.

	Entire Flow METRIC TO TRACK Data Out ALERT SET Notify if less than 1 MB/sec, for at least 1 minutes.	 
	Processor: ConsumeFromKafka METRIC TO TRACK Bytes Received ALERT SET Notify if less than 512 KBytes, for at least 30 seconds.	 

- f. 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:

- i. Click on the **>_ View CLI Command** link and familiarize yourself with the content.
- ii. Download the 2 JSON dependency files by click on the download button:
 1. Flow Deployment Parameters JSON

2. Flow Deployment KPIs JSON

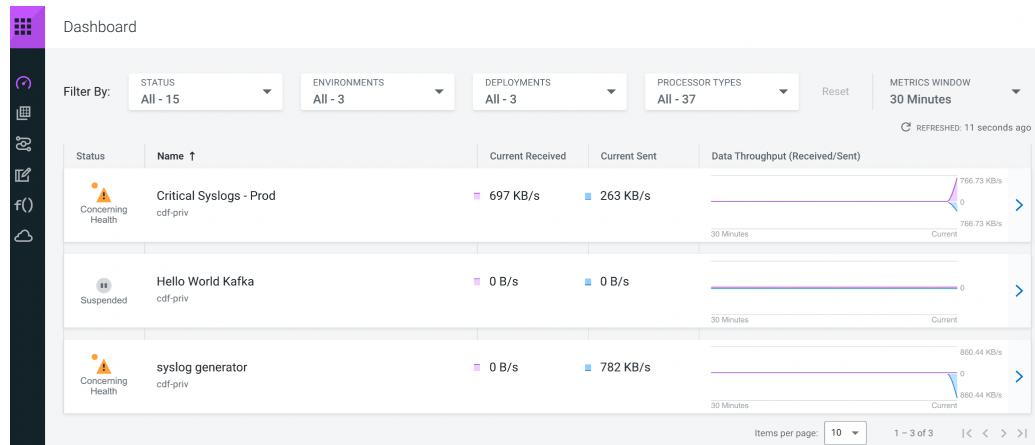
- iii. Copy the command at the end of this page and save that in a file called **deploy.sh**
- iv. Close the **Equivalent CDP CLI Command** tab.
- g. Click **Deploy** to initiate the flow deployment!
- h. You are redirected to the Deployment Dashboard where you can monitor the progress of your deployment. Creating the deployment should only take a few minutes.

The screenshot shows the Cloudera DataFlow Deployment Dashboard. On the left sidebar, there are links for Dashboard, Catalog, ReadyFlow Gallery, Flow Design, Functions, and Environments. The main dashboard area has a 'Filter By' dropdown set to 'STATUS All - 15'. It displays several deployment cards:

- Critical Syslogs - Prod**: Status is 'Deploying', Processor Type is 'aws cdf-priv'.
- Hello World Kafka**: Status is 'Suspended', Processor Type is 'cdf-priv'.
- syslog generator**: Status is 'Concerning Health', Processor Type is 'cdf-priv'.

On the right side, there's a section for 'Critical Syslogs - Prod' with tabs for 'KPIs', 'System Metrics', and 'Alerts' (which is selected). It shows 'No alerts to display'. Below that is an 'Event History' section with checkboxes for 'Info', 'Warning', and 'Error' levels, and a timestamp 'Deployment Initiated 2023-04-07 17:49 PDT'. A 'Load More' button is at the bottom.

- i. Congratulations! Your flow deployment has been created and is already processing Syslog events!



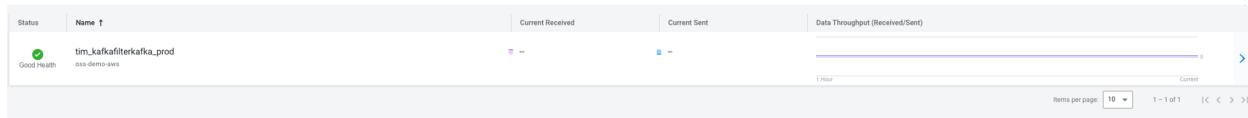
Please wait until your application is done **Deploying, Importing Flow. Wait for Good Health.**

This screenshot shows the Deployment Dashboard with the following filters: 'ENVIRONMENTS All - 1', 'DEPLOYMENTS 1 of 2', and 'PROCESSOR TYPES All - 32'. There is one active deployment:

- tim_kafkafilterkafka_prod**: Status is 'Deploying', Processor Type is 'cdf-demo-aws'. Current Received: --, Current Sent: --.

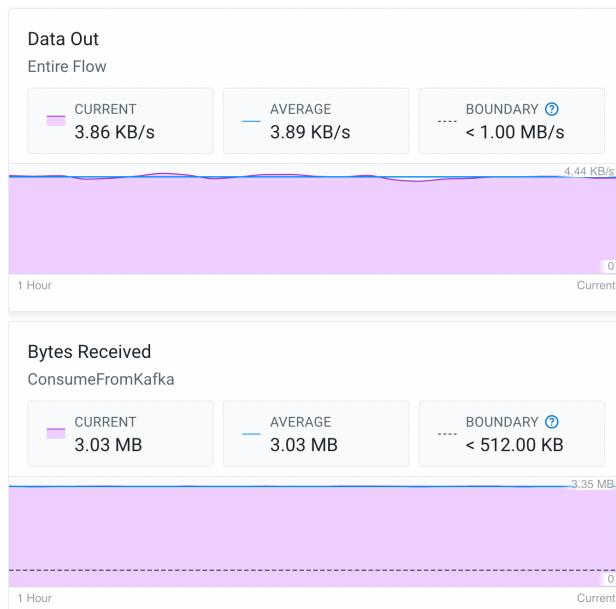
The interface includes a 'Metrics Window' dropdown set to '1 Day', a 'Reset' button, and a 'Load More' button at the bottom.

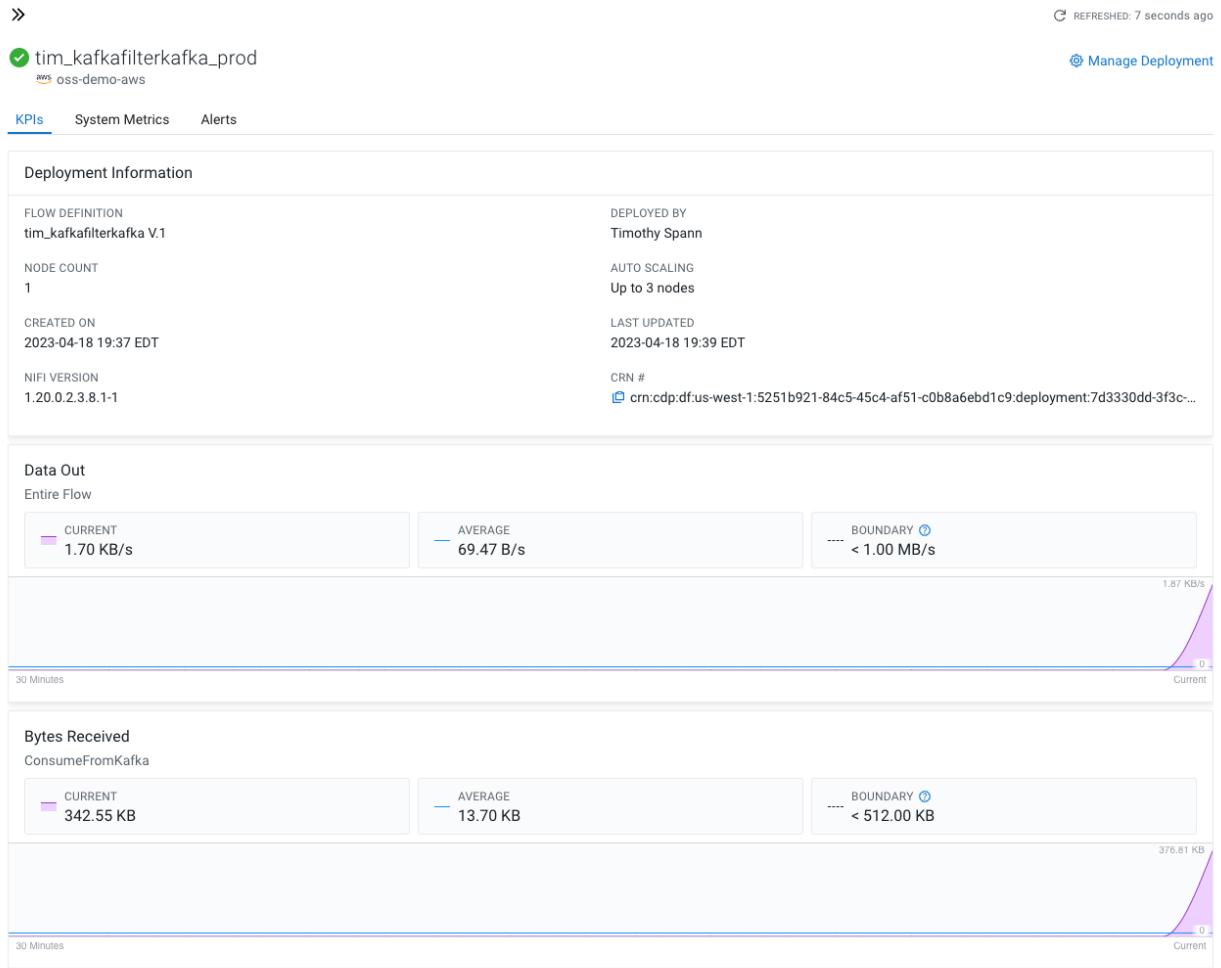
1.5 Monitoring your flow deployment



1. Notice how the dashboard shows you the data rates at which a deployment currently receives and sends data. The data is also visualized in a graph that shows the two metrics over time.
2. Change the *Metrics Window* setting at the top right. You can visualize as much as 1 Day.
3. Click on the **yourid_kafkafilterkafka_prod** deployment. The side panel opens and shows more detail about the deployment. On the *KPIs* tab it will show information about the KPIs that you created when deploying the flow.

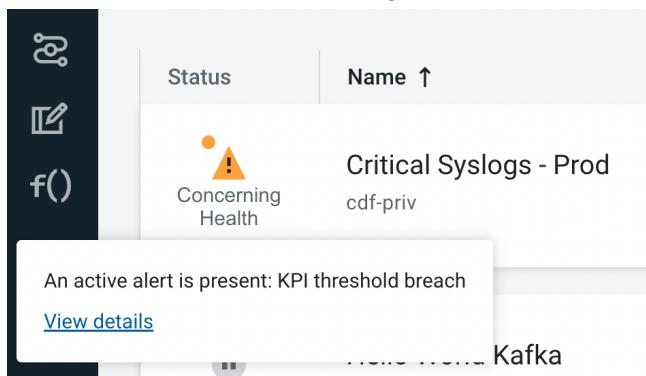
Using the two KPIs *Bytes Received* and *Data Out* we can observe that our flow is filtering out data as expected since it reads more than it sends out.





Wait a number of minutes so some data and metrics can be generated.

- Switch to the *System Metrics* tab where you can observe the current CPU utilization rate for the deployment. Our flow is not doing a lot of heavy transformation, so it should hover around at ~10% CPU usage.
- Close the side panel by clicking anywhere on the Dashboard.
- Notice how your `yourid_Critical/SyslogsProd` deployment shows *Concerning Health* status. Hover over the warning icon and click *View Details*.



7. You will be redirected to the *Alerts* tab of the deployment. Here you get an overview of active and past alerts and events. Expand the Active Alert to learn more about its cause.

The screenshot shows a table titled "Active Alerts". The columns are "Alert Type" and "Alert Time ↓". There is one row visible, which is expanded. The alert type is "KPI Threshold Breach" and it occurred "3 days ago". The alert details state: "Rate of data sent to external source in bytes is 0.00 B/s, less than the boundary of 1MB/s". A link "Learn more" is provided. The first occurrence was on "2023-04-07 17:53 PDT".

After expanding the alert, it is clear that it is caused by a KPI threshold breach for sending less than 1MB/s to external systems as defined earlier when you created the deployment.

1.6 Managing your flow deployment

1. Click on the *yourid_kafkafilterkafka_prod* deployment in the Dashboard. In the side panel, click *Manage Deployment* at the top right.

The screenshot shows the Deployment Manager interface. At the top, there is a header with two arrows and a refresh icon. Below the header, there is a section for "Critical Syslogs - Prod" with an "aws cdf-priv" badge. To the right, there is a link to "Manage Deployment". At the bottom, there are tabs for "KPIs", "System Metrics", and "Alerts", with "Alerts" being the active tab.

2. You are now being redirected to the *Deployment Manager*. The Deployment Manager allows you to reconfigure the deployment and modify KPIs, modify the number of NiFi nodes or turn auto-scaling on/off or update parameter values.

Dashboard / Critical Syslogs - Prod / Deployment Manager

Deployment Manager

DEPLOYMENT NAME: Critical Syslogs - Prod

FLOW DEFINITION: Kafka filter to Kafka - mkohs V1

DEPLOYED BY: Michael Kohs

LAST UPDATED: 2023-04-07 17:51 PDT

CRN #: crn:cdp:df:us-west-1:9d74eee4-1cad-45d7-b645...

Parameters

Running Processors that are affected by the Parameter changes will automatically be restarted.

Data entered here never leaves the environment in your cloud account. Provide parameter values directly in the text input or upload a file for parameters that expect a file.

The selected flow definition references an external Default NiFi SSL Context Service. Hence, DataFlow will automatically create a matching SSL Context Service with a keystore and truststore generated from the target environment's FreeIPEA certificate.

Kafka filter to Kafka (12)

CDP Workload User

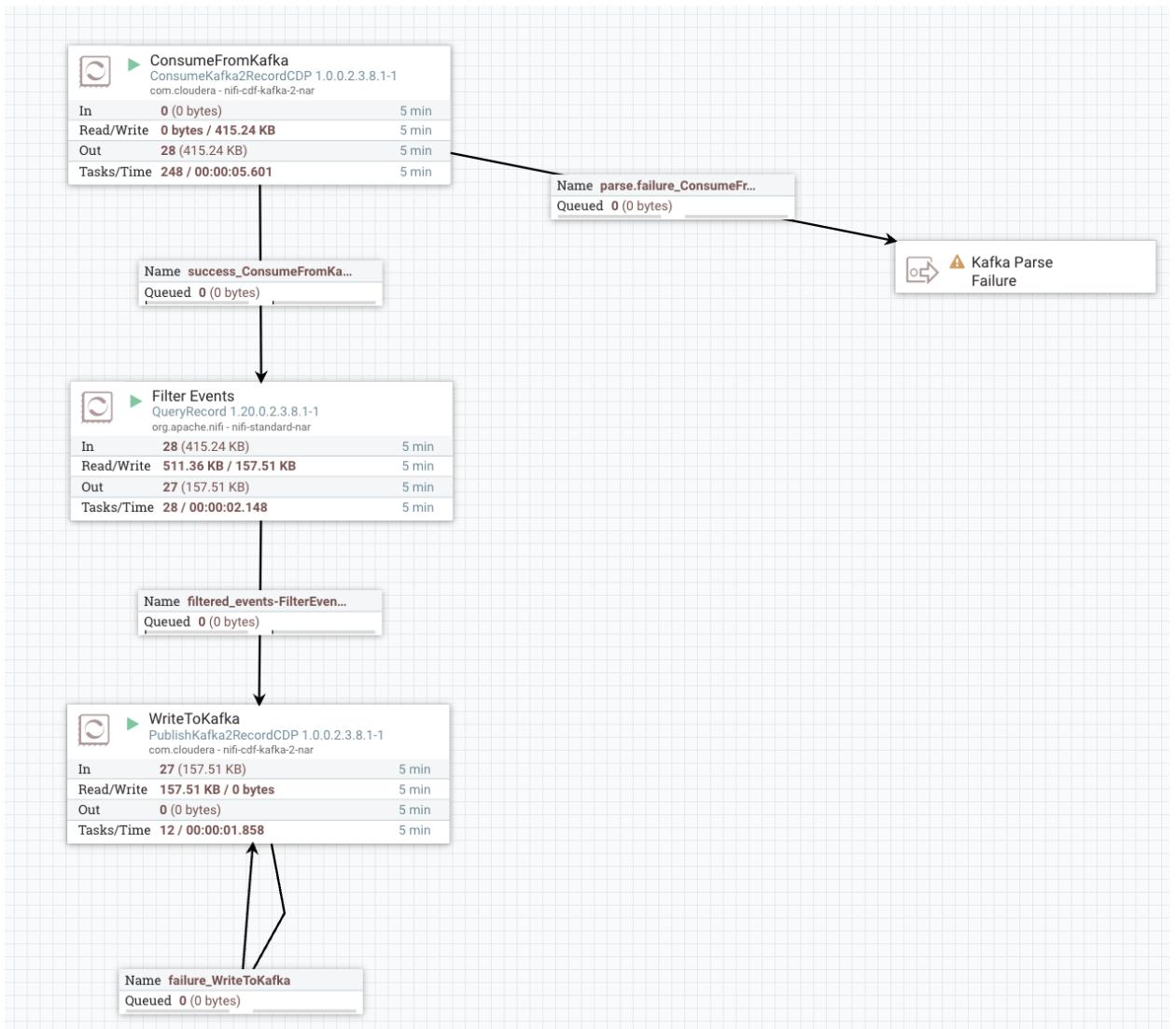
srv_nifi-kafka-ingest

3. Explore NiFi UI for deployment. Click the *Actions* menu and click on *View in NiFi*.

Actions

- View in NiFi
- Suspend flow
- Change NiFi Runtime Version
- Restart Deployment
- Terminate

4. You are being redirected to the NiFi cluster running the flow deployment. You can use this view for in-depth troubleshooting. Users can have read-only or read/write permissions to the flow deployment.



»

REFRESHED: 3 seconds ago

tim_kafkafilterkafka_prod

aws oss-demo-aws

Manage Deployment

KPIs System Metrics Alerts

Deployment Information

FLOW DEFINITION

tim_kafkafilterkafka V.1

DEPLOYED BY

Timothy Spann

NODE COUNT

1

AUTO SCALING

Up to 3 nodes

CREATED ON

2023-04-18 19:37 EDT

LAST UPDATED

2023-04-18 19:39 EDT

NIFI VERSION

1.20.0.2.3.8.1-1

CRN

[crn:cdp:df:us-west-1:5251b921-84c5-45c4-af51-c0b8a6ebd1c9:deployment:7d3330dd-3f3c...](#)

Data Out

Entire Flow

CURRENT
114.23 B/s

AVERAGE
88.23 B/s

BOUNDARY ⓘ
< 1.00 MB/s

30 Minutes

1.94 KB/s
0 Current

Bytes Received

ConsumeFromKafka

CURRENT
425.60 KB

AVERAGE
62.29 KB

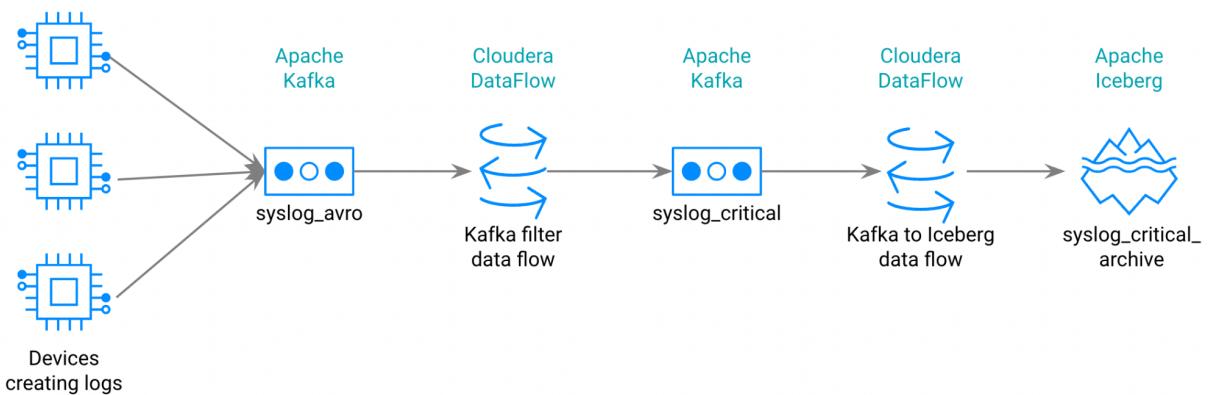
BOUNDARY ⓘ
< 512.00 KB

30 Minutes

468.16 KB
0 Current

2. Writing critical syslog events to Apache Iceberg for analysis

A few weeks have passed since you built your data flow with DataFlow Designer to filter out critical syslog events to a dedicated Kafka topic. Now that everyone has better visibility into real-time health, management wants to do historical analysis on the data. Your company is evaluating Apache Iceberg to build an open data lakehouse and you are tasked with building a flow that ingests the most critical syslog events into an Iceberg table.



Ensure your table is built and accessible.

Create an Apache Iceberg Table

1. From the Home page, click the Data Hub Clusters. Navigate to **oss-kudu-demo** from the Data Hubs list

The screenshot shows the Cloudera Management Console interface with the following details:

- Left Sidebar:** Includes links for Dashboard, Environments, Data Lakes, User Management, Data Hub Clusters (highlighted), Data Warehouses, MC Workspaces, Classic Clusters, Audit, Shared Resources, and Global Settings.
- Data Hubs Section:** A table listing running data hubs:

Status	Name	Cloud Provider	Environment	Data Hub Type	Version	Node Count	Created
Running	oss-kafka-dagthen	aws	oss-demo-aws	7.2.16 - Streams Messaging Light Duty, Apache Kafka, Schema Registry, Streams Messaging Manager, Streams Replication Manager, Cruise Control	CDH 7.2.16	4	04/19/23, 12:52 PM EDT
Running	oss-flink-demo	aws	oss-demo-aws	7.2.16 - Streaming Analytics Light Duty with Apache Flink	CDH 7.2.16	6	03/21/23, 03:29 PM EDT
Running	oss-kafka-demo	aws	oss-demo-aws	7.2.16 - Streams Messaging Light Duty, Apache Kafka, Schema Registry, Streams Messaging Manager, Streams Replication Manager, Cruise Control	CDH 7.2.16	4	03/21/23, 03:29 PM EDT
Running	oss-kudu-demo	aws	oss-demo-aws	7.2.16 - Real-time Data Mkt, Apache Impala, Hue, Apache Kudu, Apache Spark	CDH 7.2.16	7	03/21/23, 03:29 PM EDT
Running	oss-nifi-demo	aws	oss-demo-aws	7.2.16 - Flow Management Light Duty with Apache Nifi, Apache Nifi Registry	CDH 7.2.16	4	03/21/23, 03:29 PM EDT
- Bottom Right:** Buttons for 'Create Data Hub' and 'Create Cluster'.

2. Navigate to **Hue** from the Kudu Data Hub.



- Inside of **Hue** you can now create your table. You will have your own database to work with. To get to your database, click on the '<' icon next to default database. You should see your specific database in the format: <YourEmailWithUnderscores>_db. Click on your database to go to the SQL Editor.

- Create your Apache Iceberg table with the sql below and clicking the play icon to execute the sql query. Note that the the table name must prefixed with your **Work Load User Name (userid)**.

```
CREATE TABLE <>userid>>_syslog_critical_archive
(priority int, severity int, facility int, version int, event_timestamp bigint, hostname string,
body string, appName string, procid string, messageid string,
structureddata struct<sdid:struct<eventid:string,eventsources:string,iut:string>>)
STORED BY ICEBERG;
```

The screenshot shows the Impala UI interface. In the top navigation bar, there is a search bar with placeholder text "Search data and saved documents...". Below the search bar, the title "Impala" is followed by "Add a name..." and "Add a description...". On the far right of the header are icons for copy, paste, and more. The main workspace has a sidebar on the left containing icons for file operations like create, delete, and refresh, along with a "Tables" section showing "(1) +". The main area displays the following SQL code:

```
1 CREATE TABLE vetticadentest3_syslog_critical_archive
2   (priority int, severity int, facility int, version int, event_timestamp bigint, hostname string,
3   body string, appName string, procid string, messageid string,
4   structureddata struct<sdid:string,struct<eventid:string,eventsources:string,iut:string>>)
5   STORED BY 'ICEBERG';
6
```

Below the code, a message states "No logs available at this moment." To the right of the message is a green URL link: "7748d23082ef6bb7:9ae2365a00000000". At the bottom of the screen, there are tabs for "Query History", "Saved Queries", and "Results (1)". The "Results (1)" tab is selected, showing a summary message: "1 Table has been created.".

5. Once you have sent data to your table, you can query it.

The screenshot shows the Impala UI interface. The top navigation bar includes a search bar, the title "Impala", and buttons for "Add a name..." and "Add a description...". The main workspace shows the same table creation code as the previous screenshot. The main area displays the following SQL query:

```
1
2
3 select * from vetticadentest3_syslog_critical_archive;
```

Below the query, a message indicates "Query c941b4cad5a8f212:d48d4ae500000000 100% Complete (0 out of 0)". To the right of the message is a green URL link: "c941b4cad5a8f212:d48d4ae500000000". At the bottom of the screen, a message states "Done. 0 results.".

Additional Documentation

- [Create a Table](#)
- [Query a Table](#)
- [Apache Iceberg Table Properties](#)

2.1 Open ReadyFlow & start Test Session

1. Navigate to **DataFlow** from the Home Page
2. Navigate to the **ReadyFlow Gallery**
3. Explore the ReadyFlow Gallery

4. Search for the “Kafka to Iceberg” ReadyFlow.

The screenshot shows the NiFi ReadyFlow Gallery interface. On the left is a sidebar with icons for Home, Recent, Catalog, and a search bar. The main area has a search bar at the top with the text 'Iceberg'. Below it, a flow titled 'Kafka to Iceberg' is listed under the 'Added' category. The flow diagram consists of three nodes: a 'Kafka' source node, a 'Batching' processor, and a 'PutIceberg' target node. Below the flow, the description reads: 'Consumes JSON, CSV or Avro events from Kafka and writes them as Parquet files to a destination Iceberg table.' At the bottom of the card are two buttons: 'View Added Flow Definition' and 'Create New Draft'.

5. Click on “Create New Draft” to open the ReadyFlow in the Designer named **yourid_kafkatoiceberg Ex: tim_kafkatoiceberg**
6. Start a Test Session by either clicking on the *start a test session* link in the banner or going to *Flow Options* and selecting *Start* in the Test Session section.
7. In the Test Session creation wizard, select the latest NiFi version and click *Start Test Session*. Notice how the status at the top now says “Initializing Test Session”.

2.2 Modifying the flow to read syslog data

The flow consists of three processors and looks very promising for our use case. The first processor reads data from a Kafka topic, the second processor gives us the option to batch up events and create larger files which are then written out to Iceberg by the *PutIceberg* processor. All we have to do now to reach our goal is to customize its configuration to our use case.

1. Provide values for predefined parameters

- a. Navigate to *Flow Options*→ *Parameters*
- b. Select all parameters that show *No value set* and provide the following values

Name	Description	Value
CDP Workload User	CDP Workload User	<Your own workload user name>
CDP Workload User Password	CDP Workload User Password	<Your own workload user password>
Data Input Format	This flow supports AVRO, JSON and CSV	JSON

Hive Catalog Namespace		<YourEmailWithUnderScores_db>
Iceberg Table Name		<>replace_with_userid>>_syslog_critical_archive
Kafka Broker Endpoint	Comma-separated list of Kafka Broker addresses	oss-kafka-demo-corebroker2.oss-demo.qsm5-opic.cloudera.site:9093, oss-kafka-demo-corebroker1.oss-demo.qsm5-opic.cloudera.site:9093, oss-kafka-demo-corebroker0.oss-demo.qsm5-opic.cloudera.site:9093
Kafka Consumer Group Id		<>replace_with_userid>>_cdf Ex: tim_cdf
Kafka Source Topic		<>replace_with_userid>>_syslog_critical Ex: tim_syslog_critical
Schema Name		syslog
Schema Registry Hostname		oss-kafka-demo-master0.oss-demo.qsm5-opic.cloudera.site

c. Click *Apply Changes* to save the parameter values

2. Start Controller Services

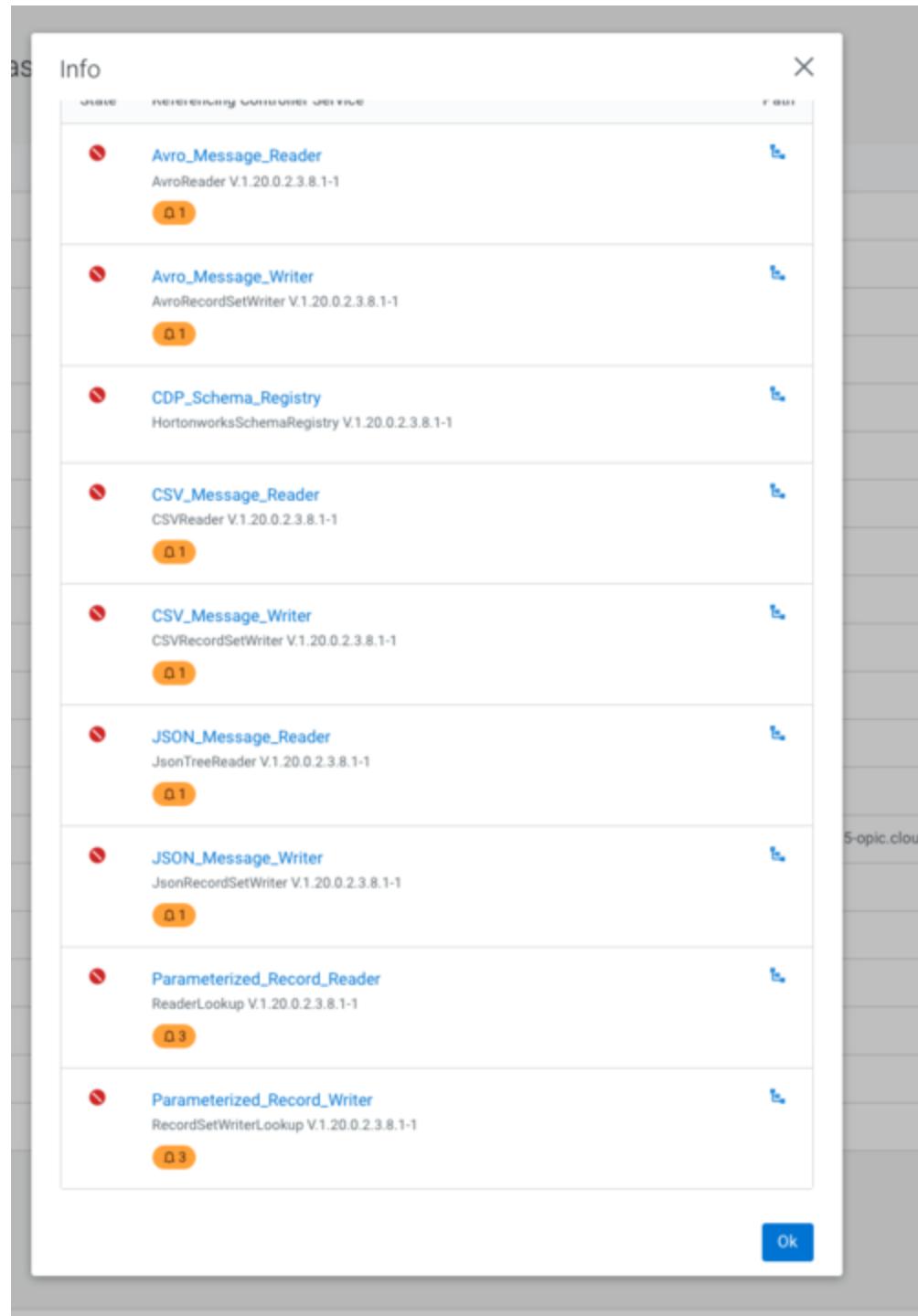
a. Navigate to *Flow Options → Services*

- b. Select *CDP_Schema_Registry* service and click *Enable Service and Referencing Components* action

Service and Referencing Con



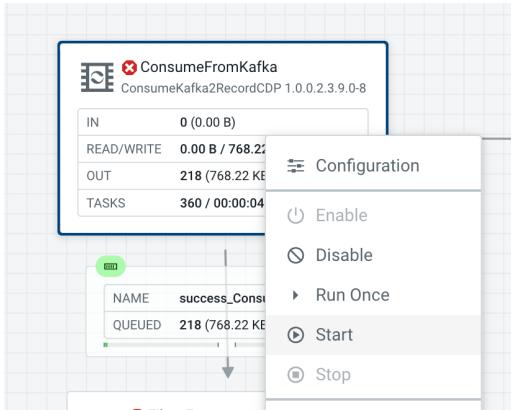
- c. Start from the top of the list and enable all remaining Controller services including KerberosPasswordUserService, HiveCatalogService, AvroReader, ...
- d. Click *Ok* if confirmation is asked.



- e. Make sure all services have been enabled

All Services for Kafka to Iceberg			
State	Name ↑	Type	Notifications
✓	AvroReader	AvroReader V.1.20.0.2.3.8.1-1	>
✓	Avro_Message_Reader	AvroReader V.1.20.0.2.3.8.1-1	>
✓	Avro_Message_Writer	AvroRecordSetWriter V.1.20.0.2.3.8.1-1	>
✓	CDP_Schema_Registry	HortonworksSchemaRegistry V.1.20.0.2.3.8.1-1	>
✓	CSV_Message_Reader	CSVReader V.1.20.0.2.3.8.1-1	>
✓	Default NiFi SSL Context Service	StandardRestrictedSSLContextService V.1.20.0.2....	>
✓	HiveCatalogService	HiveCatalogService V.1.20.0.2.3.8.1-1	>
✓	JSON_Message_Reader	JsonTreeReader V.1.20.0.2.3.8.1-1	>
✓	JsonRecordSetWriterOriginalSchema	JsonRecordSetWriter V.1.20.0.2.3.8.1-1	>
✓	KerberosPasswordUserService	KerberosPasswordUserService V.1.20.0.2.3.8.1-1	>
✓	Parameterized_Message_Reader	ReaderLookup V.1.20.0.2.3.8.1-1	>

3. Start the **ConsumeFromKafka processor** using the right click action menu or the *Start* button in the configuration drawer. It might already be started.



After starting the processor, you should see events starting to queue up in the *success_ConsumeFromKafka-FilterEvents* connection.

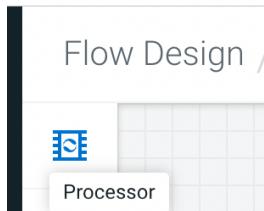
NOTE:

To receive data from your topic, you will need either the first deployment still running or to run it from another Flow Designer Test Session.

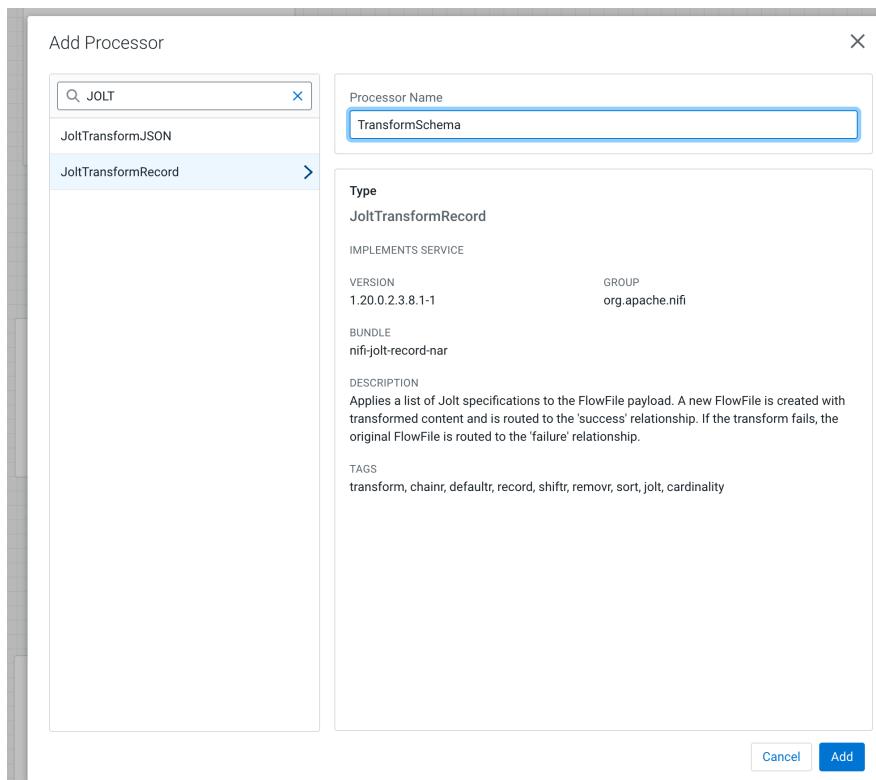
2.3 Changing the flow to modify the schema for Iceberg integration

Our data warehouse team has created an Iceberg table that they want us to ingest the critical syslog data in. A challenge we are facing is that not all column names in the Iceberg table match our syslog record schema. So we have to add functionality to our flow that allows us to change the schema of our syslog records. To do this, we will be using the *JoltTransformRecord* processor.

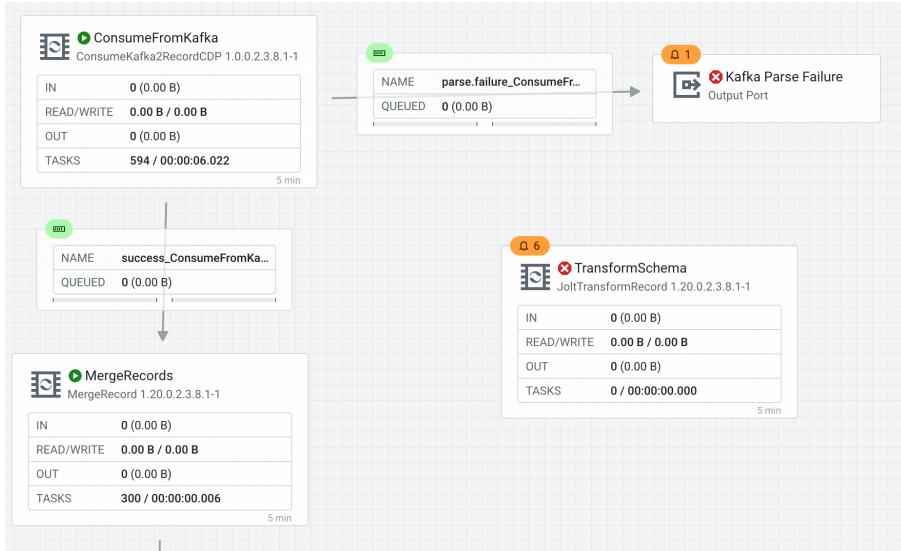
1. Add a new *JoltTransformRecord* to the canvas by dragging the processor icon to the canvas.



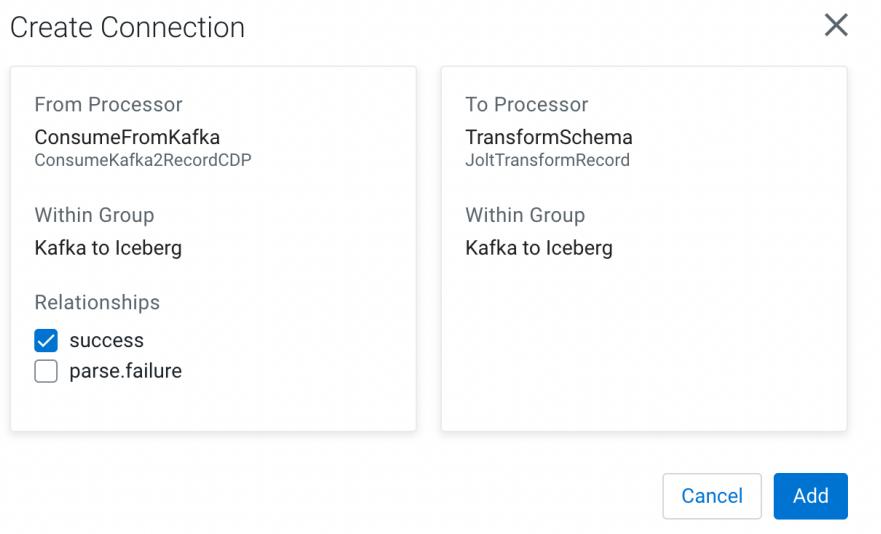
2. In the *Add Processor* window, select the *JoltTransformRecord* type and name the processor *TransformSchema*.



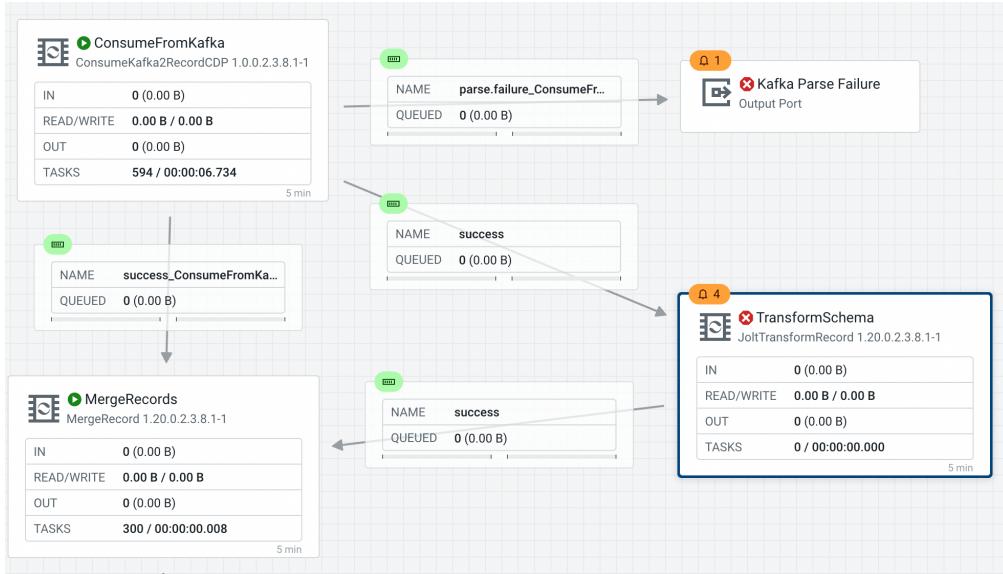
3. Validate that your new processor now appears on the canvas.



4. Create connections from *ConsumeFromKafka* to *TransformSchema* by hovering over the *ConsumeFromKafka* processor and dragging the arrow that appears to *TransformSchema*. Pick the **success** relationship to connect.

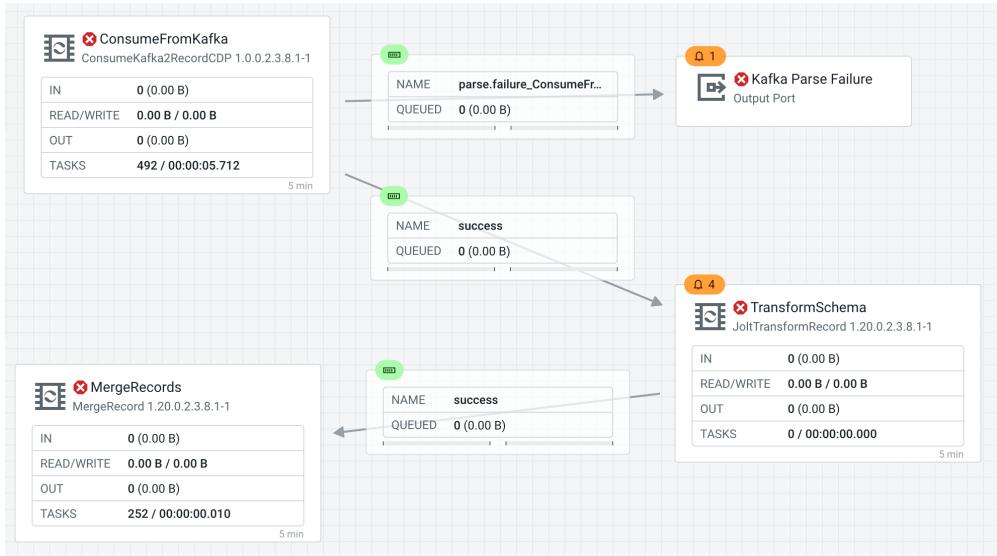


Now connect the *success* relationship of *TransformSchema* to the *MergeRecords* processor.



- Now that we have connected our new *TransformSchema* processor, we can delete the original connection between *ConsumeFromKafka* and *MergeRecords*.

Make sure that the *ConsumeFromKafka* processor is stopped. Then select the connection, empty the queue if needed, and then delete it. Now all syslog events that we receive, will go through the *TransformSchema* processor.



- To make sure that our schema transformation works, we have to create a new *Record Writer Service* and use it as the Record Writer for the *TransformSchema* processor.

Select the *TransformSchema* processor and open the configuration panel. Scroll to the

Properties section, click the three dot menu in the *Record Writer* row and select *Add Service* to create a new Record Writer.

Properties		
Property	Value	
Record Reader ⓘ	No value set	⋮
Record Writer ⓘ	No value set	⋮
Jolt Transformation DSL ⓘ	Chain	Add Service Go To Service
Jolt Specification ⓘ	No value set	
Transform Cache Size ⓘ	1	⋮

- Select *AvroRecordSetWriter*, name it *TransformedSchemaWriter* and click *Add*.

Click *Apply* in the configuration panel to save your changes.

Add Service

>

Service Name

- Now click the three dot menu again and select *Go To Service* to configure our new Avro Record Writer.

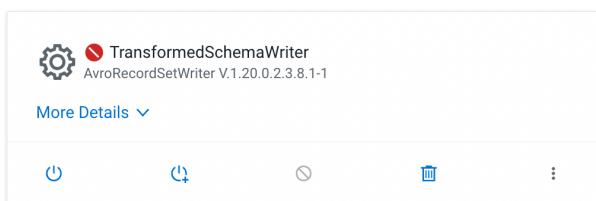
Properties

Properties		
Property	Value	
Record Reader ⓘ	No value set	⋮
Record Writer ⓘ	↳ TransformedSchemaWriter	⋮
Jolt Transformation DSL ⓘ	Chain	Add Service Go To Service
Jolt Specification ⓘ	No value set	

- To configure our new Avro Record Writer, provide the following values:

Name	Description	Value

Schema Write Strategy	Specify whether/how CDF should write schema information	Embed Avro Schema
Schema Access Strategy	Specify how CDF identifies the schema to apply.	Use 'Schema Name' Property
Schema Registry	Specify the Schema Registry that stores our schema	CDP_Schema_Registry
Schema Name	The schema name to look up in the Schema Registry	syslog_transformed



The screenshot shows the configuration page for the **TransformedSchemaWriter**. At the top, it displays the service name **TransformedSchemaWriter** and its version **AvroRecordSetWriter V.1.20.0.2.3.8.1-1**. Below this are standard UI controls: a power icon, a refresh icon, a settings gear icon, a trash bin icon, and a three-dot menu icon. The main area is divided into sections:

- Settings**: Contains a field labeled ***Service Name** with the value **TransformedSchemaWriter**.
- Comments**: A text input field.
- Properties**: A table showing configuration properties:

Property	Value
Schema Write Strategy ?	Embed Avro Schema :
Schema Cache ?	No value set :
Schema Access Strategy ?	Use 'Schema Name' Property :
Schema Registry ?	CDP_Schema_Registry :
Schema Name ?	syslog_transformed :

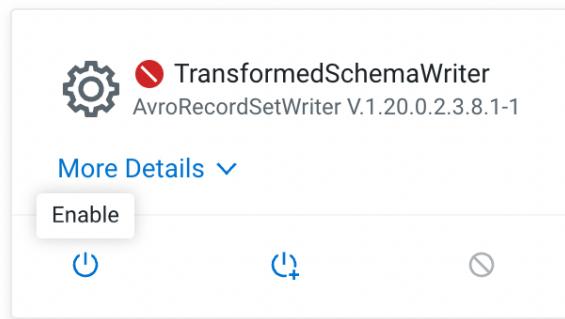
10. Convert the value that you provided for *Schema Name* into a parameter. Click on the three dot menu in the *Schema Name* row and select *Convert To Parameter*.

Schema Name	?	syslog_transformed	⋮
Schema Version	?	No value set	Convert To Parameter
Default Reader	⋮	All Readers	⋮

11. Give the parameter the name *Schema Name Transformed* and click “add”. You have now created a new parameter from a value that can be used in more places in your data flow.

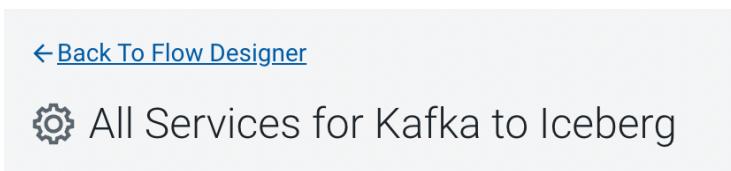
Schema Name	?	↪ #{{Schema Name Transformed}}	⋮
-------------	-------------------	--------------------------------	---

12. **Apply** your configuration changes and *Enable* the Service by clicking the power icon. Now you have configured our new Schema Writer and we can return back to the Flow Designer canvas.



If you have any issues, end the test session and restart. If your login timed out, close your browser and re login.

13. Click *Back to Flow Designer* to navigate back to the canvas.



14. Select *TransformSchema* to configure it and provide the following values:

Name	Description	Value
------	-------------	-------

Record Reader	Service used to parse incoming events	AvroReader
Record Writer	Service used to format outgoing events	TransformedSchemaWriter
Jolt Specification	The specification that describes how to modify the incoming JSON data. We are standardizing on lower case field names and renaming the <i>timestamp</i> field to <i>event_timestamp</i> .	[{ "operation": "shift", "spec": { "appName": "appname", "timestamp": "event_timestamp", "structuredData": { "SDID": { "eventId": "structureddata.sdid.eventid", "eventSource": "structureddata.sdid.eventsource", "iut": "structureddata.sdid.iut" } }, "*": { "@": "&" } } }]]

15. Scroll to *Relationships* and select *Terminate* for the *failure*, *original* relationships and click *Apply*.

Relationships

You can choose to automatically term both terminate and retry are selected,

success [?](#)

Terminate Retry

failure [?](#)

Terminate Retry

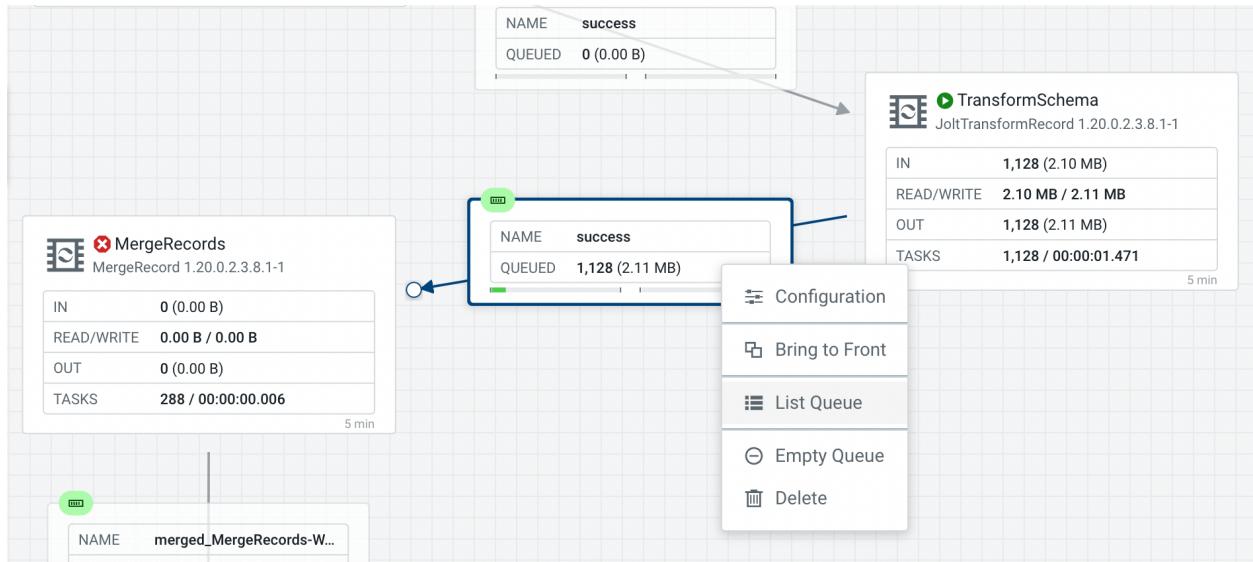
original [?](#)

Terminate Retry

Retry logic specified below will apply t

16. Start your *ConsumeFromKafka* and *TransformSchema* processor and validate that the transformed data matches our Iceberg table schema.

17. Once events are queuing up in the connection between *TransformSchema* and *MergeRecord*, right click the connection and select *List Queue*.



18. Select any of the queued files and select the book icon to open it in the Data Viewer

The screenshot shows the 'List Queue' view for the 'success' queue. It displays a table of queued files:

Position	Filename	UUID	File Size	Queue Duration	Lineage Duration	Penalized
1	6b59473b-ff7e-41d9-89c1-473a67a6f2cf	aa0fadfb-18ec-4458-a2c3-2141cdb509f6	3 KB	00:01:07.868	00:07:25.966	No

To the right, the 'Data Viewer' is open, showing the contents of the selected file:

```

Content
View as: Formatted / Avro
1 v [ {
2   "priority" : 89,
3   "severity" : 1,
4   "facility" : 11,
5   "version" : 1,
6   "event_timestamp" : 1681340412516,
7   "hostname" : "host3.example.com",
8   "body" : "application6 has exited cleanly",
9   "appname" : "application6",
10  "procid" : "7556",
11  "messageid" : "ID20",
12  "structureddata" : {
13    "sdid" : {
14      "eventid" : "79",
15      "eventsources" : "kernel",
16      "iut" : "2"
17    }
18  }
}

```

19. Notice how all field names have been transformed to lower case and how the *timestamp* field has been renamed to *event_timestamp*.

The screenshot shows the 'Data Viewer' displaying the transformed JSON data. The output is:

```

Content
View as: Formatted / Avro
1 v [ {
2   "priority" : 89,
3   "severity" : 1,
4   "facility" : 11,
5   "version" : 1,
6   "event_timestamp" : 1681340412516,
7   "hostname" : "host3.example.com",
8   "body" : "application6 has exited cleanly",
9   "appname" : "application6",
10  "procid" : "7556",
11  "messageid" : "ID20",
12  "structureddata" : {
13    "sdid" : {
14      "eventid" : "79",
15      "eventsources" : "kernel",
16      "iut" : "2"
17    }
18  }
}

```

2.4 Merging records and start writing to Iceberg

Now that we have verified that our schema is being transformed as needed, it's time to start the remaining processors and write our events into the Iceberg table. The *MergeRecords* processor is configured to batch events up to increase efficiency when writing to Iceberg. The final processor, *WriteToIceberg* takes our Avro records and writes them into a Parquet formatted table.

1. Tip: You can change the configuration to something like “**30 sec**” to speed up processing.
2. Select the *MergeRecords* processor and explore its configuration. It is configured to batch events up for at least 30 seconds or until the queued up events have reached *Maximum Bin Size* of 1GB. You will want to lower these for testing.

Properties

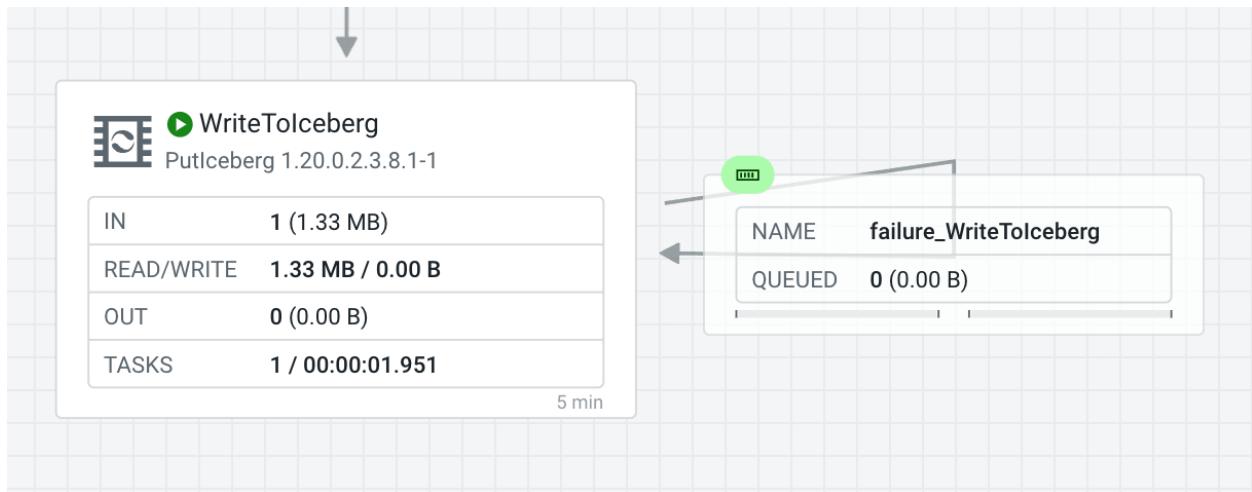
Property	Value
Record Reader ?	AvroReader
Record Writer ?	Avro_Message_Writer
Merge Strategy ?	Bin-Packing Algorithm
Correlation Attribute Name ?	No value set
Attribute Strategy ?	Keep Only Common Attributes
Minimum Number of Records ?	1
Maximum Number of Records ?	100000000
Minimum Bin Size ?	100 MB
Maximum Bin Size ?	1 GB
Max Bin Age ?	5 min
Maximum Number of Bins ?	5

3. Start the *MergeRecords* processor and verify that it batches up events and writes them out after 30 seconds.
4. Select the *WriteToIceberg* processor and explore its configuration. Notice how it relies on several parameters to establish a connection to the right database and table.

Properties

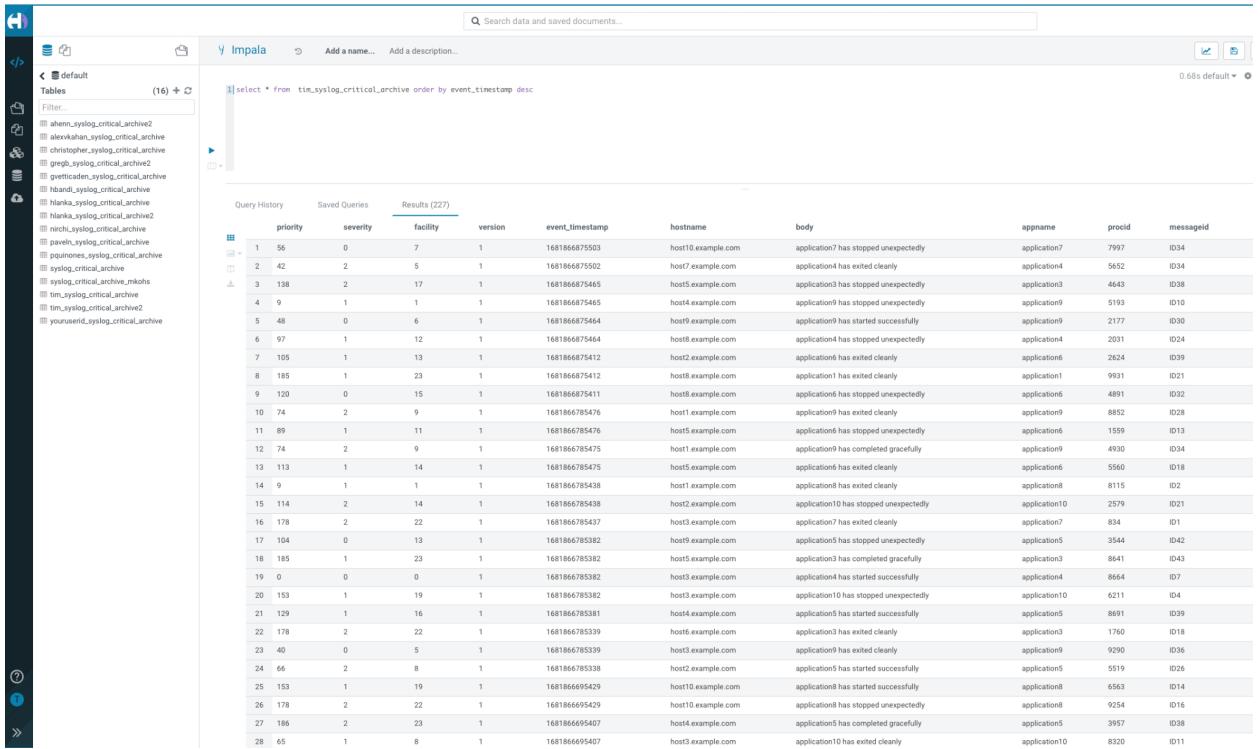
Property	Value
Record Reader ?	↳ AvroReader
Catalog Service ?	↳ HiveCatalogService
Catalog Namespace ?	↳ #{Hive Catalog Namespace}
Table Name ?	↳ #{Iceberg Table Name}
File Format ?	PARQUET
Maximum File Size ?	No value set
Kerberos User Service ?	↳ KerberosPasswordUserService

5. Start the *WriteTolceberg* processor and verify that it writes records successfully to Iceberg. If the metrics on the processor increase and you don't see any warnings or events being written to the *failure_WriteTolceberg* connection, your writes are successful!



Congratulations! With this you have completed the second use case.

You may want to log into Hue to check your data has loaded.



The screenshot shows the DataBrick interface with the following details:

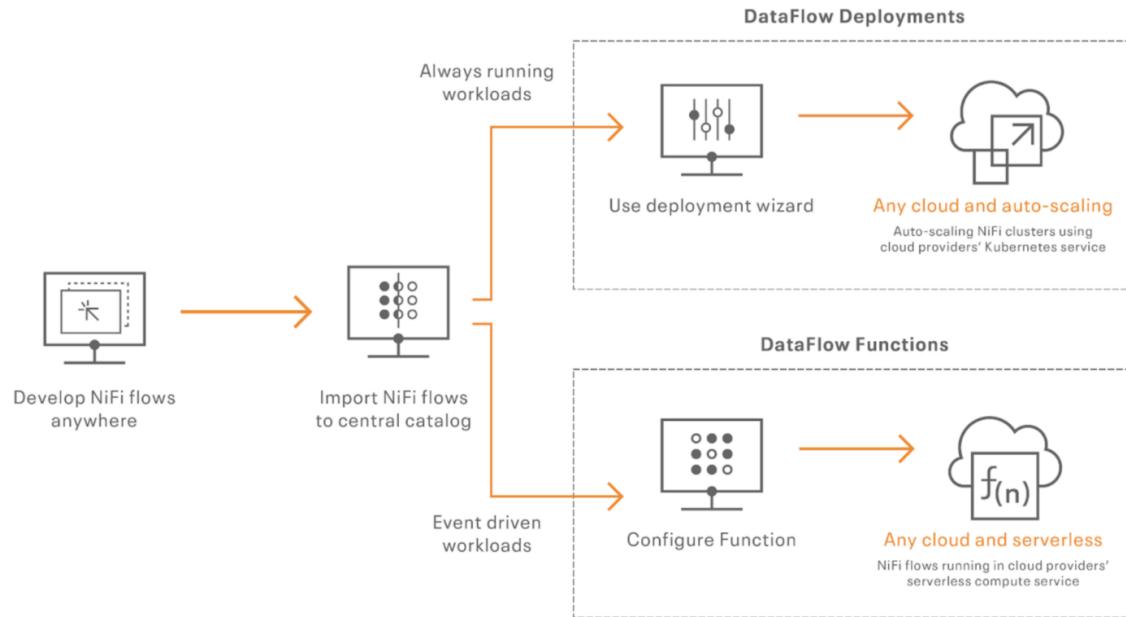
- Left Sidebar:** Contains icons for Home, Cluster, Databricks ML, Databricks ML Flow, Databricks ML Pipeline, Databricks ML Model, Databricks ML Model Flow, Databricks ML Pipeline Flow, and Help.
- Top Bar:** Shows the cluster name "Impala" and a search bar with placeholder "Search data and saved documents...".
- Table List:** A sidebar titled "Tables" showing 16 entries, including "ahern_syslog_critical_archive2", "alekhanan_syslog_critical_archive", "christophersyslog_critical_archive", "gregs_syslog_critical_archive2", "jason_syslog_critical_archive", "lauridsyslog_critical_archive", "hankasyslog_critical_archive", "nirchisyslog_critical_archive2", "pavinsyslog_critical_archive", "poumonesyslog_critical_archive", "syslog_critical_archive", "syslog_critical_archive_mikohs", "tim_syslog_critical_archive", "tim_syslog_critical_archive2", and "youuserid_syslog_critical_archive".
- Query Editor:** Displays the query: "select * from tim_syslog_critical_archive order by event_timestamp desc".
- Results:** A table titled "Results (227)" with the following columns: priority, severity, facility, version, event_timestamp, hostname, body, appname, procid, and messageid. The table contains 227 rows of log data.

	priority	severity	facility	version	event_timestamp	hostname	body	appname	procid	messageid
1	56	0	7	1	1681866875503	host10.example.com	application7 has stopped unexpectedly	application7	7997	ID34
2	42	2	5	1	1681866875502	host7.example.com	application4 has exited cleanly	application4	5652	ID34
3	138	2	17	1	1681866875465	host3.example.com	application3 has stopped unexpectedly	application3	4643	ID38
4	9	1	1	1	1681866875465	host4.example.com	application9 has stopped unexpectedly	application9	5193	ID10
5	48	0	6	1	1681866875464	host9.example.com	application9 has started successfully	application9	2177	ID30
6	97	1	12	1	1681866875464	host1.example.com	application4 has stopped unexpectedly	application4	2031	ID24
7	105	1	13	1	1681866875412	host2.example.com	application6 has exited cleanly	application6	2624	ID39
8	185	1	23	1	1681866875412	host3.example.com	application1 has exited cleanly	application1	9931	ID21
9	120	0	15	1	1681866875411	host1.example.com	application6 has stopped unexpectedly	application6	4891	ID32
10	74	2	9	1	1681866785476	host1.example.com	application9 has exited cleanly	application9	8852	ID28
11	89	1	11	1	1681866785476	host1.example.com	application6 has stopped unexpectedly	application6	1599	ID13
12	74	2	9	1	1681866785475	host1.example.com	application9 has completed gracefully	application9	4930	ID34
13	113	1	14	1	1681866785475	host1.example.com	application6 has exited cleanly	application6	5560	ID18
14	9	1	1	1	1681866785438	host1.example.com	application9 has exited cleanly	application9	8115	ID2
15	114	2	14	1	1681866785438	host2.example.com	application10 has stopped unexpectedly	application10	2579	ID21
16	178	2	22	1	1681866785437	host3.example.com	application7 has exited cleanly	application7	834	ID1
17	104	0	13	1	1681866785382	host9.example.com	application5 has stopped unexpectedly	application5	3544	ID42
18	185	1	23	1	1681866785382	host1.example.com	application3 has completed gracefully	application3	8641	ID43
19	0	0	0	1	1681866785382	host3.example.com	application4 has started successfully	application4	8664	ID7
20	153	1	19	1	1681866785382	host3.example.com	application10 has stopped unexpectedly	application10	6211	ID4
21	129	1	16	1	1681866785381	host1.example.com	application5 has started successfully	application5	8691	ID39
22	178	2	22	1	1681866785339	host3.example.com	application3 has exited cleanly	application3	1760	ID18
23	40	0	5	1	1681866785339	host3.example.com	application9 has exited cleanly	application9	9290	ID36
24	66	2	8	1	1681866785338	host2.example.com	application5 has started successfully	application5	5519	ID26
25	153	1	19	1	1681866995429	host10.example.com	application8 has started successfully	application8	6563	ID14
26	178	2	22	1	1681866995429	host10.example.com	application8 has stopped unexpectedly	application8	9254	ID16
27	186	2	23	1	1681866695407	host4.example.com	application5 has completed gracefully	application5	3957	ID38
28	65	1	8	1	1681866695407	host3.example.com	application10 has exited cleanly	application10	8320	ID11

Feel free to publish your flow to the catalog and create a deployment just like you did for the first one.

3. Resize image flow deployed as serverless function

DataFlow Functions provides a new, efficient way to run your event-driven Apache NiFi data flows. You can have your flow executed within AWS Lambda, Azure Functions or Google Cloud Functions and define the trigger that should start its execution.



DataFlow Functions is perfect for use cases such as:

- Processing files as soon as they land into the cloud provider object store
- Creating microservices over HTTPS
- CRON driven use cases
- etc

In this use case, we will be deploying a NiFi flow that will be triggered by HTTPS requests to resize images. Once deployed, the cloud provider will provide an HTTPS endpoint that you'll be able to call to send an image, it will trigger the NiFi flow that will return a resized image based on your parameters.

The deployment of the flow as a function will have to be done within your cloud provider.

The below tutorial will use AWS as the cloud provider. If you're using Azure or Google Cloud, you can still refer to this [documentation](#) to deploy the flow as a [function](#).

3.1 Designing the flow for AWS Lambda

1. Go into Cloudera DataFlow / Flow Design and create a new draft with a name of your choice.
2. Drag and drop an Input Port named **input** onto the canvas. When triggered, AWS Lambda is going to inject into that input port a FlowFile containing the information about the HTTPS call that has been made.

Example of payload that will be injected by AWS Lambda as a FlowFile:

```

1  {
2    "version": "2.0",
3    "routeKey": "ANY /NaaF_S3_compression",
4    "rawPath": "/default/NaaF_S3_compression",
5    "rawQueryString": "",
6    "headers": {
7      "accept-encoding": "gzip",
8      "content-length": "34",
9      "content-type": "application/json",
10     "date": "Fri, 06 Aug 2021 12:35:29 GMT",
11     "host": "ns9dgd2tw6.execute-api.us-east-2.amazonaws.com",
12     "x-amzn-trace-id": "Root=1-610d2c92-7651f91a5dcbab8671048936",
13     "x-forwarded-for": "3.19.60.232",
14     "x-forwarded-port": "443",
15     "x-forwarded-proto": "https"
16   },
17   "requestContext": {
18     "accountId": "381358652250",
19     "apiId": "ns9dgd2tw6",
20     "domainName": "ns9dgd2tw6.execute-api.us-east-2.amazonaws.com",
21     "domainPrefix": "ns9dgd2tw6",
22     "http": {
23       "method": "POST",
24       "path": "/default/NaaF_S3_compression",
25       "protocol": "HTTP/1.1",
26       "sourceIp": "3.19.60.232",
27       "userAgent": ""
28     },
29     "requestId": "DpPm3j83iYcEP7w=",
30     "routeKey": "ANY /NaaF_S3_compression",
31     "stage": "default",
32     "time": "06/Aug/2021:12:35:30 +0000",
33     "timeEpoch": 1628253330104
34   },
35   "body": "{\n    \"data\": \"this is my payload\"\n}",
36   "isBase64Encoded": false
37 }
```

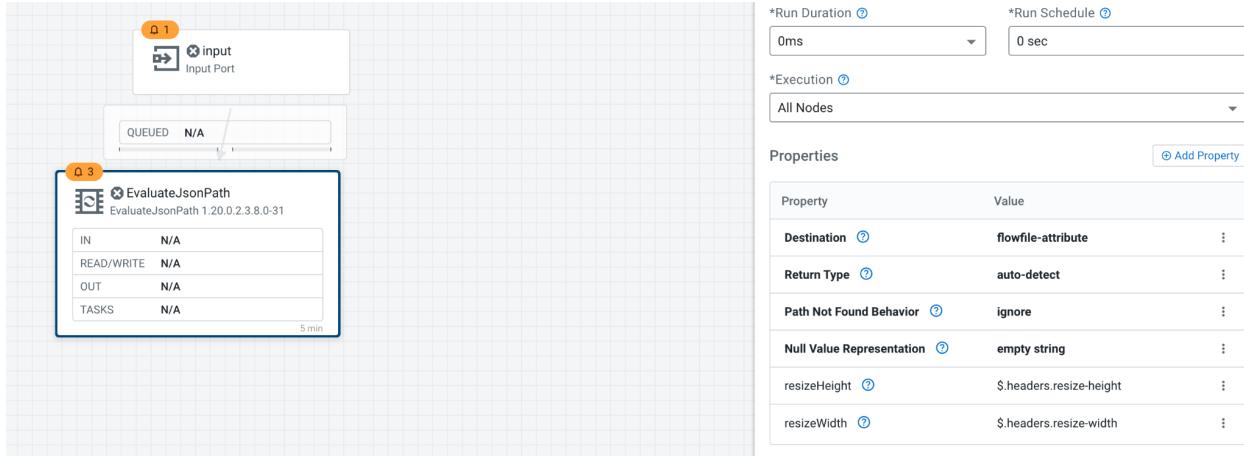
3. Drag and drop an EvaluateJsonPath processor, call it ExtractHTTPHeaders. We're going to use this to extract the HTTP headers that we want to keep in our flow. Add two properties configured as below. It'll save as FlowFile's attributes the HTTP headers (resize-height and resize-width) that we will be adding when making a call with our image to specify the dimensions of the resized image.

```

resizeHeight => $.headers.resize-height
resizeWidth => $.headers.resize-width

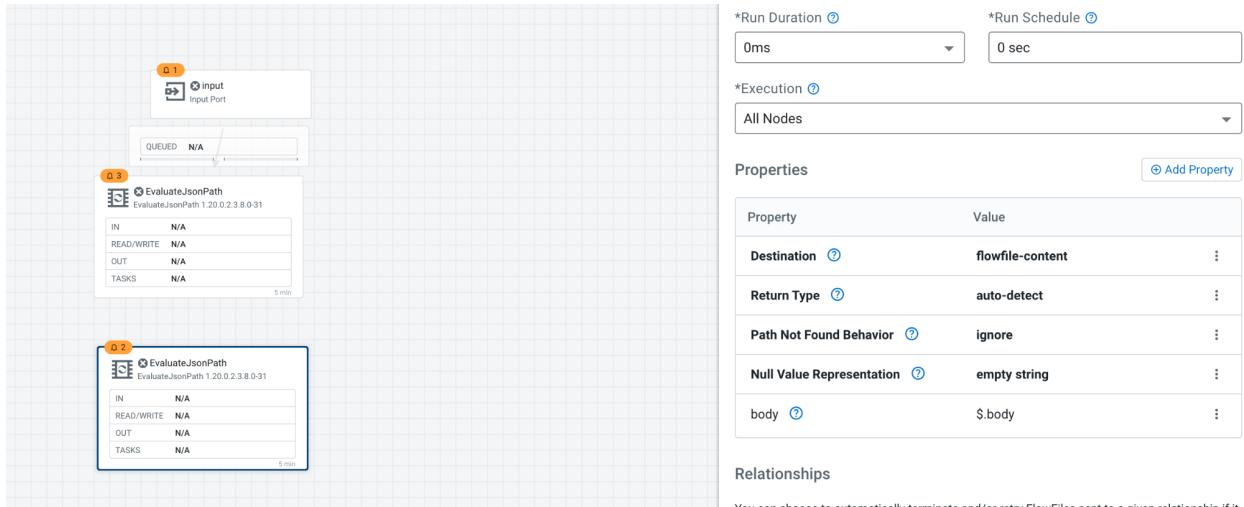
```

Note: don't forget to change **Destination** as "flowfile-attribute" and Click **Apply**.



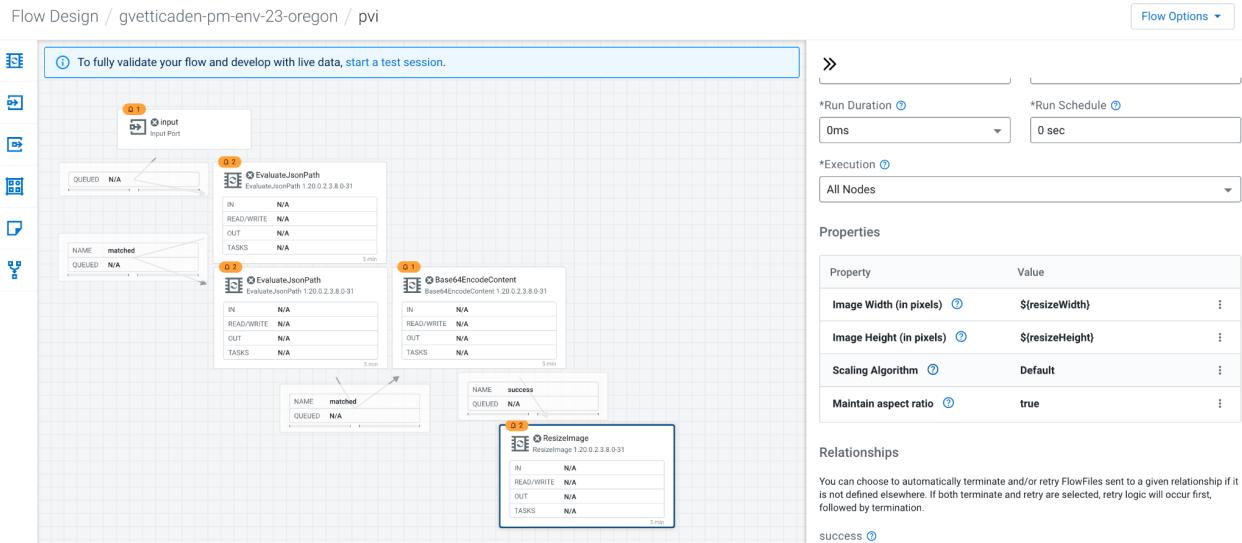
- Drag and drop another EvaluateJsonPath processor and then change its name to a unique one. This one will be used to retrieve the content of the body field from the payload we received and use it as the new content of the FlowFile. This field contains the actual representation of the image we have been sending over HTTP with Base 64 encoding.

```
body => $.body
```



- Drag and drop a Base64EncodeContent processor and change the mode to Decode. This will Base64 decode the content of the FlowFile to retrieve its binary format.

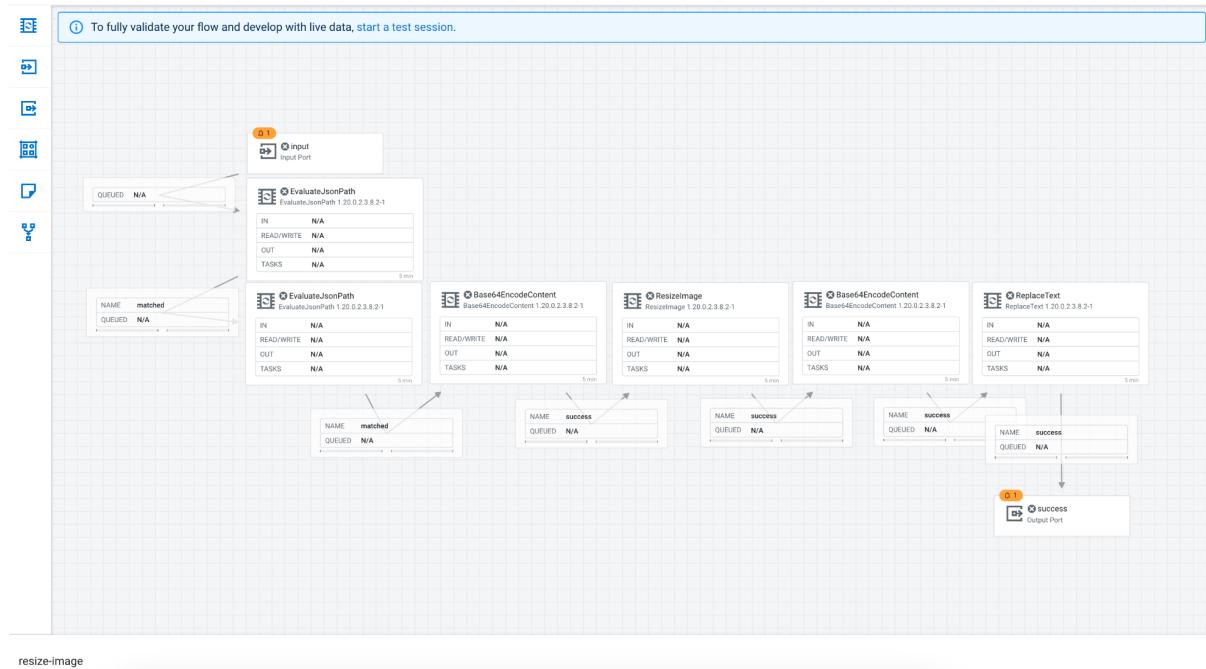
6. Drag and drop a ResizeImage processor. Use the previously created FlowFile attributes to specify the new dimensions of the image. Also, specify true for maintaining the ratio.



7. Drag and drop a Base64EncodeContent processor. To send back the resized image to the user, AWS Lambda expects us to send back a specific JSON payload with the Base 64 encoding of the image.
8. Drag and drop a ReplaceText processor. We use it to extract the Base 64 representation of the resized image and add it in the expected JSON payload. Add the below JSON in "Replacement Value" and change "Evaluation Mode" to "Entire text".

```
{
  "statusCode": 200,
  "headers": { "Content-Type": "image/png" },
  "isBase64Encoded": true,
  "body": "$1"
}
```

9. Drag and drop an output port.
10. Connect all the components together, you can auto-terminate the unused relationships.
This should look like this:



resize-image

You can now publish the flow into the DataFlow Catalog in the Flow Options menu:

The screenshot shows the "Flow Options" menu open over the flow. The menu includes sections for "Edit", "View", and "Exiting". It also lists "Services" and "Parameters". The "Edit" section contains "Services" and "Parameters". The "View" section contains "Documentation". The "Exiting" section contains "Exit Draft". The "Test Session" section contains a "Start" button. The "Publish To Catalog" section contains a "Publish" button and a tooltip "Publish the flow to the catalog".

Make sure to give it a name that is unique (you can prefix it with your name):

Flow Catalog

Search by name

Import Flow Definition

REFRESHED: 25 seconds ago

Name ↑	Type	Versions	Last Updated
000-abc1	Custom Flow Definition	1	a year ago
000-import-test-1	Custom Flow Definition	1	a year ago
000-test-1	Custom Flow Definition	2	a year ago
1-2 automatic autoscaling	Custom Flow Definition	4	a year ago
aaa	Custom Flow Definition	1	a year ago
aaaa	Custom Flow Definition	1	6 months ago
abchs	Custom Flow Definition	1	8 months ago
abchs1	Custom Flow Definition	1	8 months ago
ABROWN DFX Test	Custom Flow Definition	2	2 years ago

Publish A New Flow

Flow Name: **pvillard-meet-the-committers-resize-image**

Flow Description: Add description

Version Comments: Initial Version

Cancel Publish

Once the flow is published, make sure to copy the CRN of the published version (it will end by /v.1):

The screenshot shows the DataFlow Functions interface. At the top right, there is a refresh icon and the text "REFRESHED: 9 seconds ago". Below the title, there is a "Actions" button with a dropdown arrow. The flow name is "pvillard-meet-the-committers-resize-image" and it was updated 10 seconds ago by Pierre Villard. The flow description is "No description specified". The CRN is listed as "crn:cdp:df:us-west-1:558bc1d2-8867-4357-8524-311d51259233:flow:pvillard-meet-the-committers-resize-image". There is a checkbox labeled "Only show deployed versions". A table below shows one version (v1) with 0 deployments and 1 associated draft. Buttons for "Deploy →", "Download", and "Create New Draft" are present. The "Associated Drafts" section shows one draft named "prm-sandbox-aws" with a "resize-image" step. The "CRN #" is also listed. The "Created" section shows the date and time as "2023-04-28 17:49 EEST" by Pierre Villard, with a note "Initial Version".

3.2 Deploying the flow as a function in AWS Lambda

First thing first, go into DataFlow Functions and download the binary for running DataFlow Functions in AWS Lambda:

The screenshot shows the 'Functions' section of the Cloudera DataFlow interface. On the left is a dark sidebar with navigation links: Dashboard, Catalog, ReadyFlow Gallery, Flow Design, f() Functions (which is highlighted in purple), and Environments. Below these are links for Get Started, Help, and a user profile for Pierre Villard. At the bottom of the sidebar are version information (2.4.1-b22) and a double-left arrow icon. The main content area is titled 'Functions' and contains three cards:

- DataFlow Functions for AWS Lambda**: Features an orange Lambda icon. Below it are 'Download' and 'Documentation' buttons.
- DataFlow Functions for Azure Functions**: Features a blue lightning bolt icon. Below it are 'Download' and 'Documentation' buttons.
- DataFlow Functions for Google Cloud Functions**: Features a blue hexagon icon. Below it are 'Download' and 'Documentation' buttons.

A note at the bottom says: 'Download the binary for executing flows in these Cloud Providers' with a 'Learn More' link.

This screenshot shows a detailed view of the 'DataFlow Functions for AWS Lambda' section. The sidebar on the left is identical to the one in the previous screenshot. The main content area is titled 'Functions' and features a large card for the AWS Lambda function:

DataFlow Functions for AWS Lambda

Below the title are 'Download' and 'Documentation' buttons. A note at the bottom says: 'Download the binary for executing flows in these Cloud Providers' with a 'Learn More' link.

This should download a binary with a name similar to:
naaf-aws-lambda-1.0.0.2.3.7.0-100-bin.zip

Once you have the binary, make sure you also have:

- The CRN of the flow you published in the DataFlow Catalog
- The Access Key that has been provided with these instructions in "Competition Resources" section

- The Private Key that has been provided with these instructions in “Competition Resources” section

In order to speed up the deployment, we’re going to leverage some scripts to automate the deployment. It assumes that your AWS CLI is properly configured locally on your laptop and you can use the `jq` command for reading JSON payloads. You can now follow [the instructions from this page here](#).

However, if you wish to deploy the flow in AWS Lambda manually through the AWS UI, you can follow [the steps described here](#).

n Flow Competition Tutorials

Author: Michael Kohs George Vetticaden Timothy Spann

Date: 04/18/2023

Last Updated: 4/27/2023

Useful Data Assets

Setting Your Workload Password

Creating a Kafka Topic

Use Case walkthrough

1. Reading and filtering a stream of syslog data	9
2. Writing critical syslog events to Apache Iceberg for analysis	29
3. Resize image flow deployed as serverless function	56

Use Case Walkthrough for Competition

Notice

This document assumes that you have registered for an account, activated it and logged into the CDP Sandbox. This is for authorized users only who have attended the webinar and have read the training materials.

A short guide and references are listed [here](#).

Competition Resources

Login to the Cluster

<https://login.cdpworkshops.cloudera.com/auth/realms/se-workshop-5/protocol/saml/clients/cdp-sso>

Kafka Broker connection string

- oss-kafka-demo-corebroker2.oss-demo.qsm5-opic.cloudera.site:9093,
- oss-kafka-demo-corebroker1.oss-demo.qsm5-opic.cloudera.site:9093,
- oss-kafka-demo-corebroker0.oss-demo.qsm5-opic.cloudera.site:9093

Kafka Topics

- syslog_json
- syslog_avro
- syslog_critical

Schema Registry Hostname

- oss-kafka-demo-master0.oss-demo.qsm5-opic.cloudera.site

Schema Name

- syslog
- syslog_avro
- syslog_transformed

Syslog Filter Rule

- SELECT * FROM FLOWFILE WHERE severity <= 2

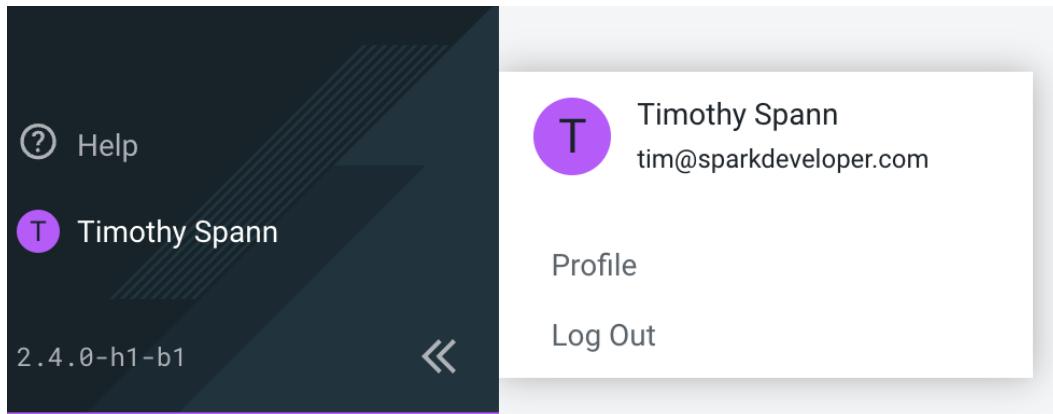
Access Key and Private Key for Machine User in DataFlow Function

- Access Key: eda9f909-d1c2-4934-bad7-95ec6e326de8
- Private Key: eon6eFzLlxZI/gpU0dWht21DI60MkSQZlzeWSGBSI=

The following keys are needed if you want to deploy a DataFlow Function that you build during the Best in Flow Competition.

Your Workflow User Name and Password

1. Click on your name at the bottom left corner of the screen for a menu to pop up.



2. Click on *Profile* to be redirected to your user's profile page with important information.

The screenshot shows the Cloudera Management Console interface. On the left is a dark sidebar with various navigation options: Dashboard, Environments, Data Lakes, **User Management** (which is currently selected), Data Hub Clusters, Data Warehouses, ML Workspaces, Classic Clusters, Audit, and Shared Resources. The main content area is titled "Users / Timothy Spann". It displays a table of user information:

Name	Timothy Spann
Email	tim@sparkdeveloper.com
Workload User Name	tim
CRN	crn:altus:iam:us-west-1:5251b921-84c5-45c4-af51-c0b8a6ebd1c9:user:b8f9... Copy
Tenant ID	5251b921-84c5-45c4-af51-c0b8a6ebd1c9 Copy
Identity Provider	se-workshop-5-keycloak-idp
Last Interactive Login	04/19/2023 8:14 AM EDT
Profile Management	View profile
Workload Password	Set Workload Password (Workload password is currently set)

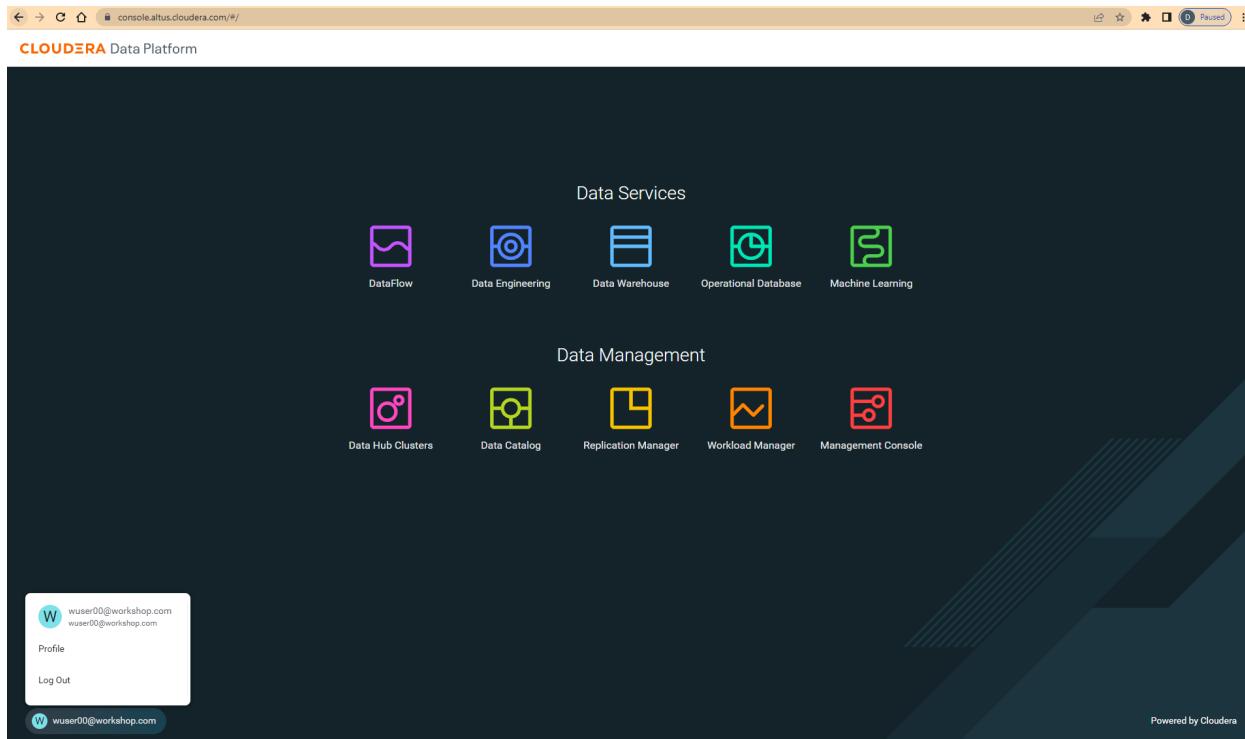
Below the table, there is a navigation bar with links: Access Keys (which is underlined to indicate it's active), Roles, Resources, Groups, and SSH Keys.

If your Workload Password does not say currently set or you forgot it, follow the steps below to reset it. Your **userid** is shown above at **Workload User Name**.

Setting Workload Password

You will need to define your workload password that will be used to access non-SSO interfaces. You may read more about it [here](#). Please keep it with you. If you have forgotten it, you will be able to repeat this process and define another one.

1. From the **Home Page**, click on your User Name (Ex: `tim`) at the lower left corner.
2. Click on the **Profile** option.



1. Click option **Set Workload Password**.
2. Enter a suitable Password and Confirm Password.
3. Click the button **Set Workload Password**.

The screenshot shows the Cloudera Management Console interface. The left sidebar includes options like Dashboard, Environments, Data Lakes, User Management (which is selected), Data Hub Clusters, Data Warehouses, ML Workspaces, Classic Clusters, Audit, Shared Resources, and Global Settings. The main content area displays user details for "wuser00@workshop.com".

Name	wuser00@workshop.com
Email	wuser00@workshop.com
Workload User Name	wuser00
CRN	cm:altus:iam:us-west-1:d1a4553c-a799-432d-8e54-372cc2ab95f2:user:a03...
Tenant ID	d1a4553c-a799-432d-8e54-372cc2ab95f2
Identity Provider	partner-workshops
Last Interactive Login	02/28/2023 9:21 AM +04
Profile Management	View profile
Workload Password	Set Workload Password (Workload password is currently set)

Below the user details, there are tabs for Access Keys, Roles, Resources, Groups, and SSH Keys. Under the Access Keys tab, it says "No access keys found." and has a "Generate Access Key" button. The bottom of the sidebar includes Help and the user profile again.

Users / wuser00@workshop.com / Workload Password

● Password

● Confirm Password

If you use keytabs, you need to regenerate them after changing your workload password. You can do this from your user profile > Actions > Get Keytab.

Set Workload Password

Check that you got the message - Workload password is currently set or alternatively, look for a message next to Workload Password which says (Workload password is currently set). Save the password you configured as well as the workload user name for use later.

Users / wuser00@workshop.com

Name	wuser00@workshop.com
Email	wuser00@workshop.com
Workload User Name	wuser00
CRN	cm.altusiam.us-west-1:d1a4553c-a799-432d-8e54-372cc2ab95f2:users:a03...
Tenant ID	d1a4553c-a799-432d-8e54-372cc2ab95f2
Identity Provider	partner-workshops
Last Interactive Login	02/28/2023 9:21 AM +04
Profile Management	View profile
Workload Password	(Workload password is currently set)

Access Keys Roles Resources Groups SSH Keys

No access keys found.

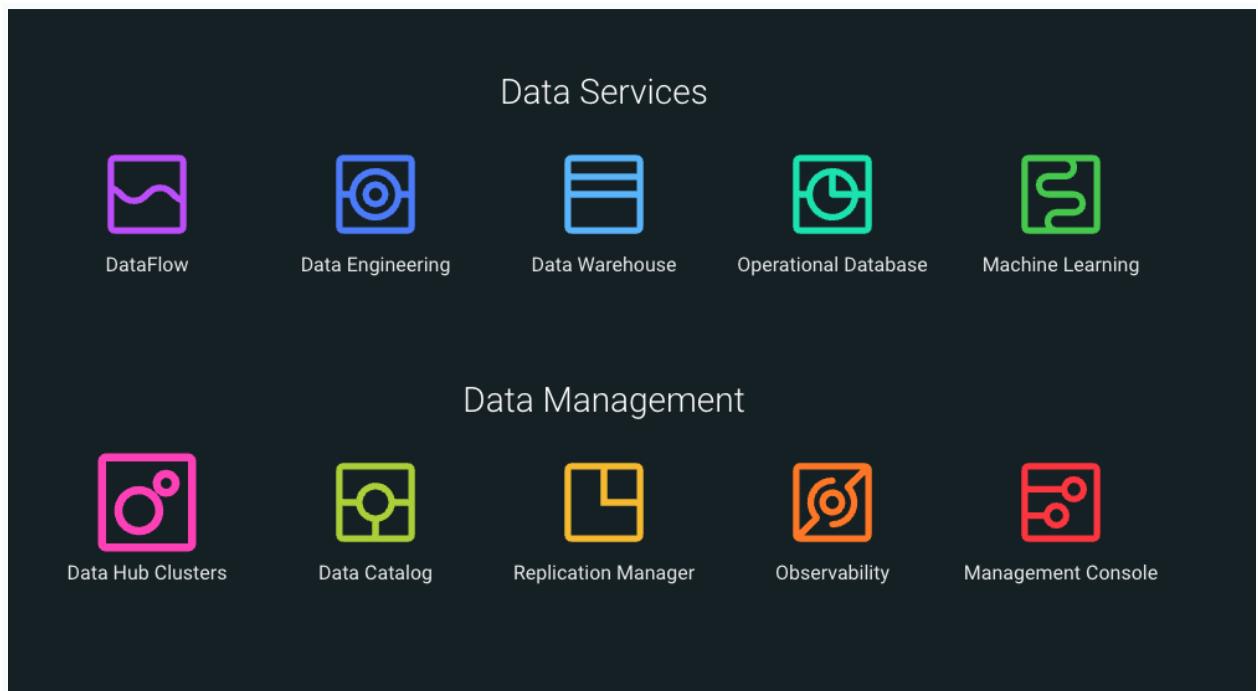
Generate Access Key

Create a Kafka Topic

The tutorials require you to create an Apache Kafka topic to send your data to, this is how you can create that topic. You will also need this information to create topics for any of your own custom applications for the competition.

1. Navigate to Data Hub Clusters from the Home Page

Info: You can always navigate back to the home page by clicking the app switcher icon at the top left of your screen.



2. Navigate to the **oss-kafka-demo** cluster

The screenshot shows the "Data Hubs" page in the Cloudera Management Console. It lists four running clusters:

Status	Name	Cloud Provider	Environment	Data Hub Type	Version	Node Count	Created
Running	oss-flink-demo	aws	oss-demo-aws	7.2.16 - Streaming Analytics Light Duty with Apache Flink	CDH 7.2.16	6	03/21/23, 03:28 PM EDT
Running	oss-kafka-demo	aws	oss-demo-aws	7.2.16 - Streams Messaging Light Duty: Apache Kafka, Schema Registry, Streams Messaging Manager, Streams Replication Manager, Cruise Control	CDH 7.2.16	4	03/21/23, 03:29 PM EDT
Running	oss-kudu-demo	aws	oss-demo-aws	7.2.16 - Real-time Data Mart: Apache Impala, Hue, Apache Kudu, Apache Spark	CDH 7.2.16	7	03/21/23, 03:29 PM EDT
Running	oss-nifi-demo	aws	oss-demo-aws	7.2.16 - Flow Management Light Duty with Apache Nifi, Apache NiFi Registry	CDH 7.2.16	4	03/21/23, 03:29 PM EDT

Streams Messaging Manager

3. Navigate to Streams Messaging Manager

The screenshot shows the Cloudera Management Console interface for the 'oss-kafka-demo' cluster. The main header reads 'Data Hubs / oss-kafka-demo / Event History'. Below it, the cluster summary shows 'STATUS: Running', 'NODES: 4', 'CREATED AT: 03/21/23, 03:20 PM EDT', and 'CLUSTER TEMPLATE: 7.2.16 - Streams Messaging Light Duty: Apache Kafka, Schema Registry, Streams Messaging Manager, Streams Replication Manager, Cruise Control'. The 'Event History' tab is selected, displaying a list of recent events:

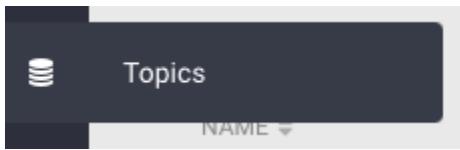
- Cloudera Manager reported that node(s) oss-kafka-demo-master-0.oss-demo.qsm3-optic.cloudera.site status became HEALTHY. (3/22/2023, 1:48:31 AM)
- COP services have been installed. (3/21/2023, 14:27:07 PM)
- Installing COP services. (3/21/2023, 14:00:00 PM)
- Pre-flight STS endpoint accessibility check result: OK (3/21/2023, 13:57:09 PM)
- Pre-flight S3 endpoint accessibility check result: OK (3/21/2023, 13:57:09 PM)
- Pre-flight archive.cloudera.com accessibility check result: OK (3/21/2023, 13:57:09 PM)
- Pre-flight Datatrans S3 API (<https://cloudera-dbus-prod.s3.amazonaws.com>) accessibility check result: OK (3/21/2023, 13:57:09 PM)

Info: Streams Messaging Manager (SMM) is a tool for working with Apache Kafka.

4. Now that you are in SMM.

Producers			Topics			Consumer Groups		
TOPICS (35)	BROKERS (3)		35			6		
ACTIVE (11)	PASSIVE (0)	ALL	MESSAGES #			ACTIVE (0)	PASSIVE (0)	ALL
producer-1	90		375			smr-service-status	8	
... (list continues)						... (list continues)		

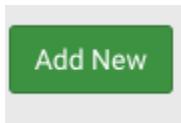
5. Navigate to the round icon third from the top, click this **Topic** button.



6. You are now in the Topic browser.

NAME	DATA IN #	DATA OUT #	MESSAGES IN #	CONSUMER GROUPS #	CURRENT LOG SIZE # (B)	Search	30 minutes	Add New
__smm-app-smm-producer-table-30s-repartition	11 KB	11 KB	237	1	445 KB			
__smm-app-smm-producer-table-15m-changelog	10 KB	0B	236	0	512 KB			
heartbeats	172 KB	0B	1.8k	0	1 MB			
srn-service-status-connector-metrics-minutes-store-changelog	18 KB	0B	90	0	6 MB			
tim_syslog_critical	0B	0B	0	0	N/A			
__smm-app-smm-producer-table-15m-repartition	11 KB	11 KB	237	1	445 KB			
srn-service-cluster-metrics-minutes-store-changelog	0B	0B	0	0	0B			
__smm-app-smm-consumer-table-15m-repartition	0B	0B	0	1	0B			
__smm-app-smm-consumer-table-30s-repartition	0B	0B	0	1	0B			
__smm_consumer_metrics	0B	0B	0	1	0B			
KafkaCruiseControlPartitionMetricSamples	171 KB	0B	8.9k	0	2 MB			
srn-metrics.secondary.internal	0B	0B	0	1	0B			
__smm-app-smm-consumer-table-15m-changelog	0B	0B	0	0	0B			

7. Click **Add New** to build a new topic.



8. Enter the name of your topic prefixed with your Workload User Name, ex: <>replace_with_userid>>_syslog_critical.

Add Topic

X

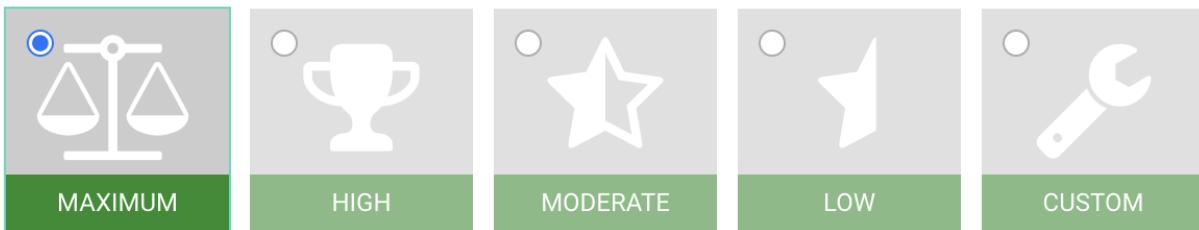
TOPIC NAME

tim_syslog_critical

PARTITIONS

3

Availability



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



Advanced

Cancel

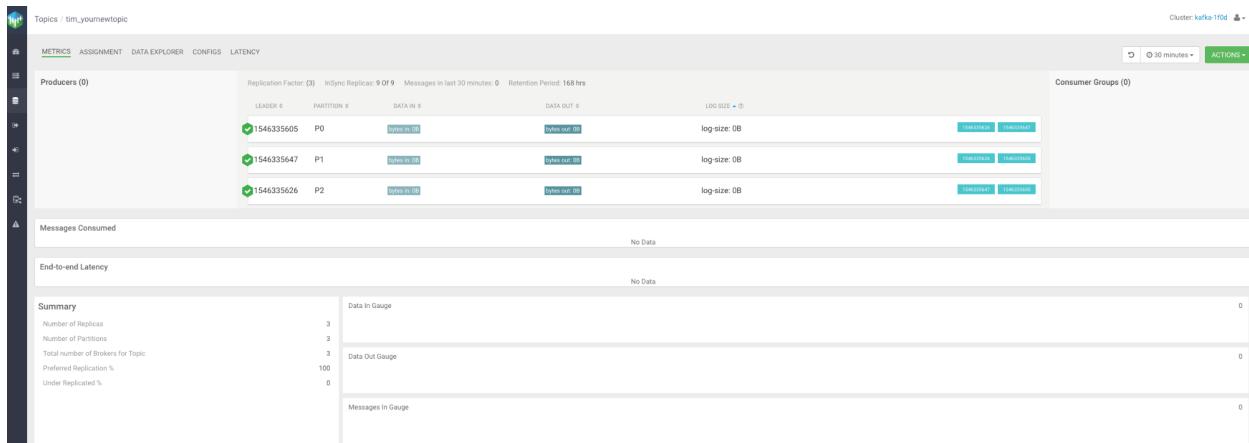
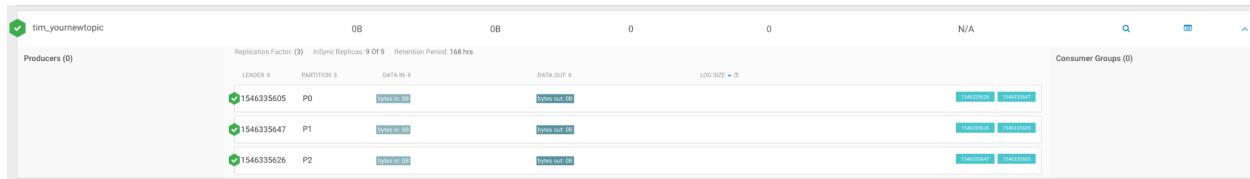
Save

9. For settings you should create it with **(3 partitions, cleanup.policy: delete, availability maximum)** as shown above.

After successfully creating a topic, close the tab that opened when navigating to Streams Messaging Manager

Congratulations! You have built a new topic.



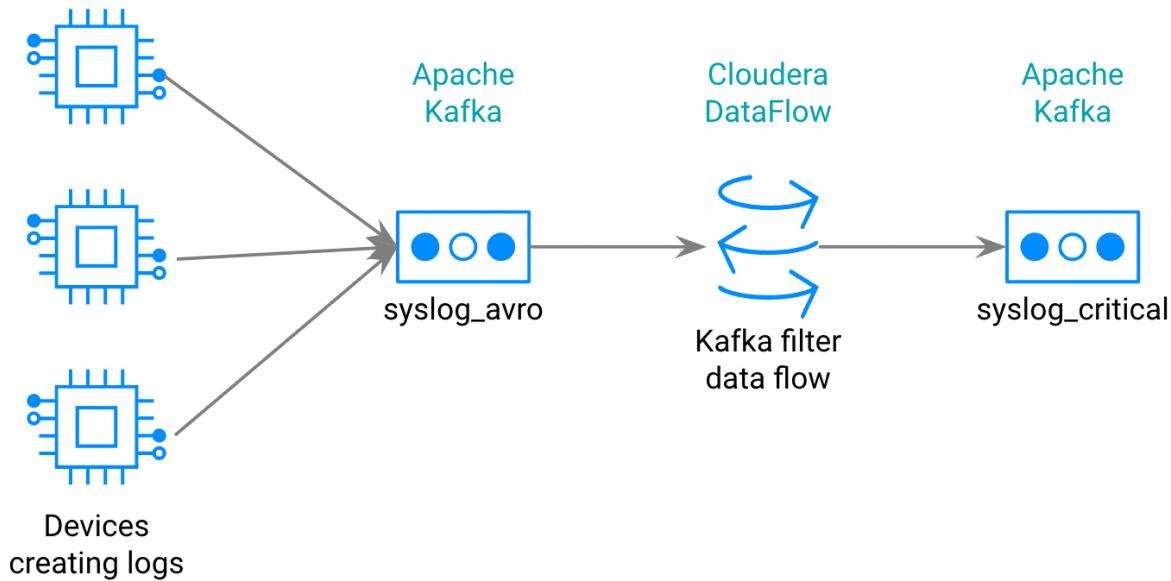


10. After successfully creating a topic, close the tab that opened when navigating to Streams Messaging Manager

1. Reading and filtering a stream of syslog data

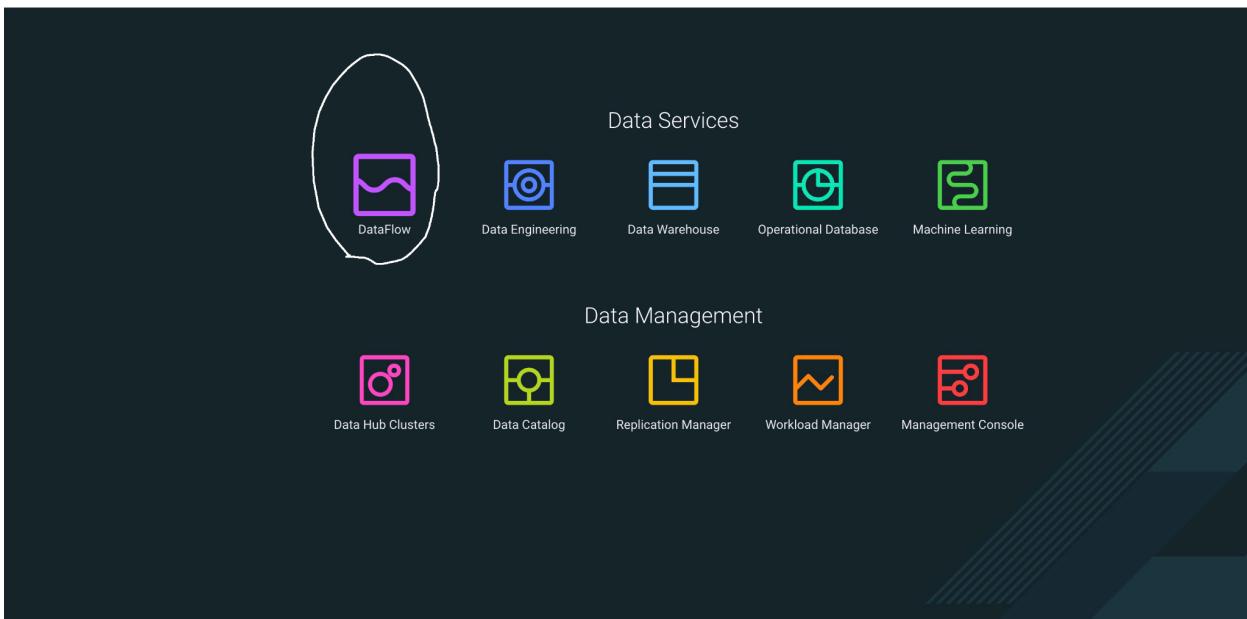
You have been tasked with filtering a noisy stream of syslog events which are available in a Kafka topic. The goal is to identify critical events and write them to the Kafka topic you just created.

Related documentation is [here](#).



1.1 Open ReadyFlow & start Test Session

1. Navigate to **DataFlow** from the Home Page



2. Navigate to the **ReadyFlow Gallery**
3. Explore the ReadyFlow Gallery

Info:

The ReadyFlow Gallery is where you can find out-of-box templates for common data movement use cases. You can directly create deployments from a ReadyFlow or create new drafts and modify the processing logic according to your needs before deploying.

4. Select the “Kafka filter to Kafka” ReadyFlow.
5. Get your user id from your profile, it is usually the first part of your email, so my email is tim@sparkdeveloper.com so my user id is **tim**. It is your “**Workload User Name**” that you are going to need for several things, remember that.
6. You already created a new **topic** to receive data in the setup section.
<>replace_with_userid>>_syslog_critical Ex: tim_syslog_critical.
7. Click on “Create New Draft” to open the ReadyFlow in the Designer with the name **youruserid_kafkafilterkafka**, for example **tim_kafkafilterkafka**. If your name has periods, underscores or other non-alphanumeric characters just leave those out. Select from the available workspaces in the dropdown, you should only have one available.

Create New Draft

Select the target workspace

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

Selected Flow Definition

NAME	VERSION
Kafka filter to Kafka	1

Workspace [?](#)

aws oss-demo-aws	60% (3 of 5)
------------------	--------------

Draft Name

✓ Draft name is valid

[Cancel](#) [Create](#)

8. Start a Test Session by either clicking on the *start a test session* link in the banner or going to *Flow Options* and selecting *Start* in the Test Session section.
9. In the Test Session creation wizard, select the latest NiFi version and click *Start Test Session*. Leave the other options to its default values. Notice how the status at the top now says “Initializing Test Session”.

Info:

Test Sessions provision infrastructure on the fly and allow you to start and stop individual processors and send data through your flow. By running data through processors step by step and using the data viewer as needed, you’re able to validate your processing logic during development in an iterative way without having to treat your entire data flow as one deployable unit.

1.2 Modifying the flow to read syslog data

The flow consists of three processors and looks very promising for our use case. The first processor reads data from a Kafka topic, the second processor allows us to filter the events before the third processor writes the filtered events to another Kafka topic.

All we have to do now to reach our goal is to customize its configuration to our use case.

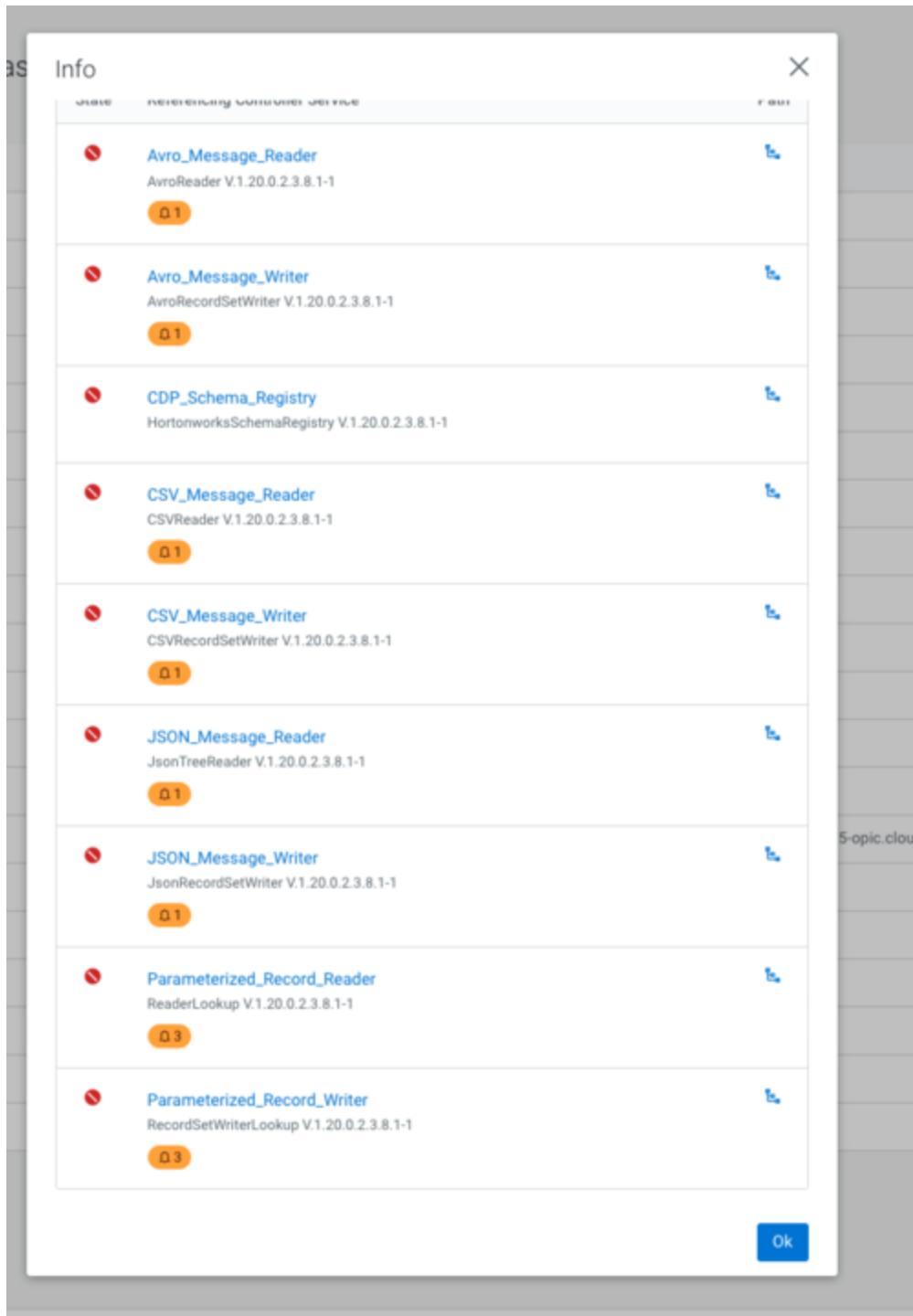
1. Provide values for predefined parameters

- a. Navigate to *Flow Options*→ *Parameters*
- b. For some settings there are some that are set already as parameters, for others they are not, you can set them manually. Make sure you create a parameter for the **Group Id**.
- c. Configure the following parameters:

Name	Description	Value
CDP Workload User	CDP Workload User	<Your own workload user ID that you saved when you configured your workload password>
CDP Workload User Password	CDP Workload User Password	<Your own workload user password you configured in the earlier step>
Filter Rule	Filter Rule	SELECT * FROM FLOWFILE WHERE severity <= 2
Data Input Format		AVRO
Data Output Format		JSON
Kafka Consumer Group ID	ConsumeFromKafka	<>replace_with_userid>>_cdf Ex: tim_cdf
Group ID	ConsumeFromKafka	<>replace_with_userid>>_cdf Ex: tim_cdf
Kafka Broker Endpoint	Comma-separated list of Kafka Broker addresses	oss-kafka-demo-corebroker2.oss-demo.qsm5-opic.cloud era.site:9093, oss-kafka-demo-corebroker1.oss-demo.qsm5-opic.cloud era.site:9093, oss-kafka-demo-corebroker0.oss-demo.qsm5-opic.cloud era.site:9093
Kafka Destination Topic	Must be unique	<>replace_with_userid>>_syslog_critical Ex: tim_syslog_critical
Kafka Producer ID	Must be unique	<>replace_with_userid>>_cdf_producer1

		Ex: tim_cdf_producer1
Kafka Source Topic		syslog_avro
Schema Name		syslog
Schema Registry Hostname	Hostname from Kafka cluster	oss-kafka-demo-master0.os-s-demo.qsm5-opic.cloudera.site

- d. Click *Apply Changes* to save the parameter values
- e. If confirmation is requested, Click *Ok*.



2. Start Controller Services

- Navigate to *Flow Options* → *Services*
- Select *CDP_Schema_Registry* service and click *Enable Service and Referencing Components* action. If this is not enabled, it may be an error or an extra space in any of the parameters for example **AVRO** must not have a new line or blank spaces. The first thing to try if you have an issue is to stop the Design

environment and then restart the test session. Check the Tips guide for more help or contact us in the bestinflow.slack.com.

Service and Referencing Con



- c. Start from the top of the list and enable all remaining Controller services
- d. Make sure all services have been enabled. You may need to reload the page or try it in a new tab.

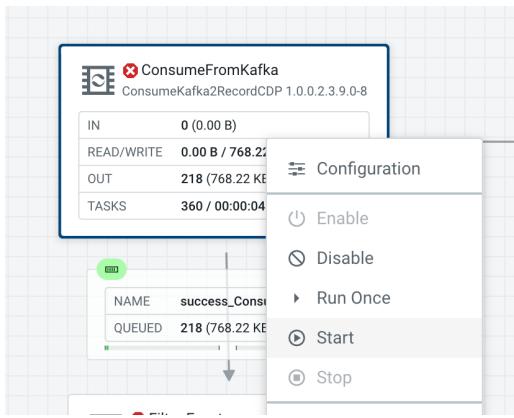
All Services for Kafka filter to Kafka

[+ Add Service](#)

State	Name ↑	Type	Notifications
✓	Avro_Message_Reader	AvroReader V.1.20.0.2.3.9.0-8	>
✓	Avro_Message_Writer	AvroRecordSetWriter V.1.20.0.2.3.9.0-8	>
✓	CDP_Schema_Registry	HortonworksSchemaRegistry V.1.20.0.2...	>
✓	CSV_Message_Reader	CSVReader V.1.20.0.2.3.9.0-8	>
✓	CSV_Message_Writer	CSVRecordSetWriter V.1.20.0.2.3.9.0-8	>
✓	Default NiFi SSL Context Service	StandardRestrictedSSLContextService ...	>
✓	JSON_Message_Reader	JsonTreeReader V.1.20.0.2.3.9.0-8	>
✓	JSON_Message_Writer	JsonRecordSetWriter V.1.20.0.2.3.9.0-8	>
✓	Parameterized_Record_Reader	ReaderLookup V.1.20.0.2.3.9.0-8	>
✓	Parameterized_Record_Writer	RecordSetWriterLookup V.1.20.0.2.3.9.0...	>

- 3. If your processors have all **started** because you started your controller services, it is best to stop them all by right clicking on each one and clicking 'Stop' and then start them one at a time so you can follow the process easier. **Start the *ConsumeFromKafka processor*** using the right click action menu or the *Start* button in the configuration

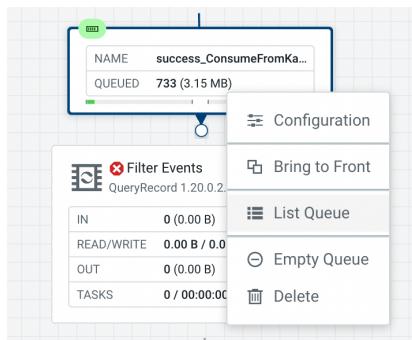
drawer.



After starting the processor, you should see events starting to queue up in the *success_ConsumeFromKafka-FilterEvents* connection.

4. Verify data being consumed from Kafka

- Right-click on the *success_ConsumeFromKafka-FilterEvents* connection and select *List Queue*



Info:

The *List Queue* interface shows you all flow files that are being queued in this connection. Click on a flow file to see its metadata in the form of *attributes*. In our use case, the attributes tell us a lot about the Kafka source from which we are consuming the data. Attributes change depending on the source you're working with and can also be used to store additional metadata that you generate in your flow.

- b. Select any flow file in the queue and click the *book* icon to open it in the *Data Viewer*

Info: The *Data Viewer* displays the content of the selected flow file and shows you the events that we have received from Kafka. It automatically detects the data format - in this case JSON - and presents it in human readable format.

- c. Scroll through the content and note how we are receiving syslog events with varying severity.

□ Data Viewer

Content

View as: Formatted / JSON

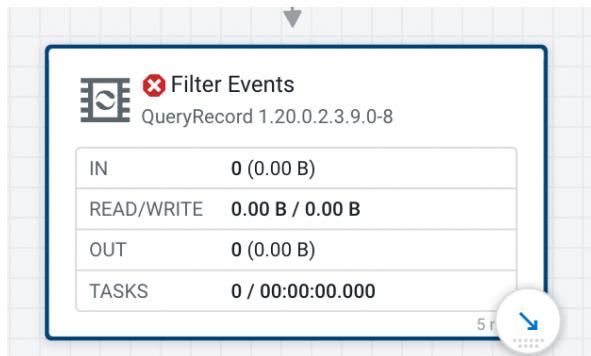
```

1 v [ {
2   "priority" : 54,
3   "severity" : 6,
4   "facility" : 6,
5   "version" : 1,
6   "timestamp" : 1680908330914,
7   "hostname" : "host8.example.com",
8   "body" : "application8 has exited cleanly",
9   "appName" : "application8",
10  "procid" : "6086",
11  "messageid" : "ID44",
12  "structuredData" : {
13    "SDID" : {
14      "eventId" : "88",
15      "eventSource" : "application",
16      "iut" : "2"
17    }
18  }
19 ]

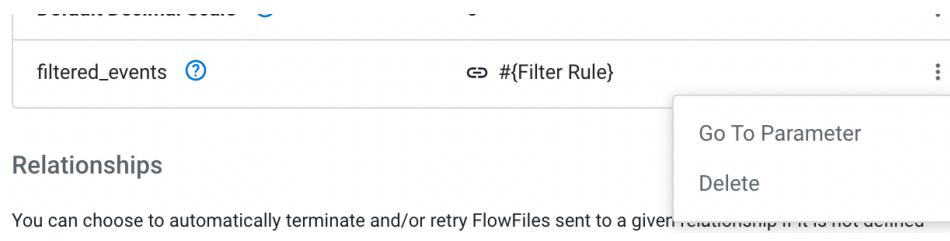
```

5. Define filter rule to filter out low severity events

- Return to the Flow Designer by closing the Data Viewer tab and clicking *Back To Flow Designer* in the *List Queue* screen.
- Select the *Filter Events* processor on the canvas. We are using a *QueryRecord* processor to filter out low severity events. The *QueryRecord* processor is very flexible and can run several filtering or routing rules at once.



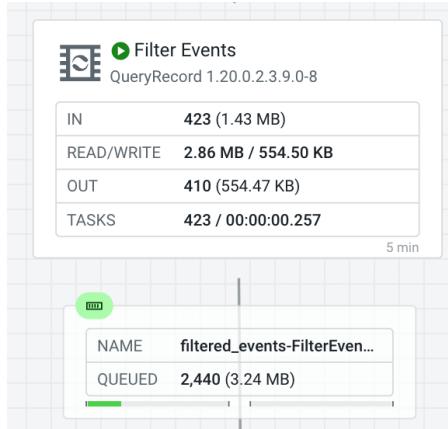
- In the configuration drawer, scroll down until you see the *filtered_events* property. We are going to use this property to filter out the events. Click on the menu at the end of the row and select *Go To Parameter*.



- If you wish to change this, you can change the Parameter value.
- Click *Apply Changes* to update the parameter value. Return back to the Flow Designer.
- Start the *Filter Events* processor using the right-click menu or the *Start* icon in the configuration drawer.

6. Verify that the filter rule works

- After starting the *Filter Events* processor, flow files will start queueing up in the *filtered_events-FilterEvents-WriteToKafka* connection



- Right click the *filtered_events-FilterEvents-WriteToKafka* connection and select *List Queue*.
- Select a few random flow files and open them in the Data Viewer to verify that only events with severity <=2 are present.

The screenshot shows the Data Viewer interface. The title bar says 'Data Viewer'. The content area displays two JSON objects representing flow files. The first object has lines 1 through 35 numbered on the left. The second object starts at line 36. Both objects contain fields like priority, severity, facility, version, timestamp, hostname, body, appName, procid, messageid, and structuredData. The structuredData field contains an SDID object with eventId, eventSource, and iut fields.

```
Content View as: Formatted / JSON
1 * [ {
2   "priority" : 66,
3   "severity" : 2,
4   "facility" : 8,
5   "version" : 1,
6   "timestamp" : 1680909990832,
7   "hostname" : "host4.example.com",
8   "body" : "application1 has exited cleanly",
9   "appName" : "application1",
10  "procid" : "8932",
11  "messageid" : "ID42",
12  "structuredData" : {
13    "SDID" : {
14      "eventId" : "95",
15      "eventSource" : "kernel",
16      "iut" : "6"
17    }
18  },
19  },
20  {
21   "priority" : 0,
22   "severity" : 0,
23   "facility" : 0,
24   "version" : 1,
25   "timestamp" : 1680909990840,
26   "hostname" : "host5.example.com",
27   "body" : "application6 has stopped unexpectedly",
28   "appName" : "application6",
29   "procid" : "6003",
30   "messageid" : "ID35",
31   "structuredData" : {
32     "SDID" : {
33       "eventId" : "33",
34       "eventSource" : "kernel",
35       "iut" : "3"
36     }
37   }
38 } ]
```

- Navigate back to the Flow Designer canvas.

7. Write the filtered events to the Kafka alerts topic

Now all that is left is to start the *WriteToKafka* processor to write our filtered high severity events to *syslog_critical* Kafka topic.

- a. Select the *WriteToKafka* processor and explore its properties in the configuration drawer.
- b. Note how we are plugging in many of our parameters to configure this processor. Values like *Kafka Brokers*, *Topic Name*, *Username*, *Password* and the *Record Writer* have all been parameterized and use the values that we provided in the very beginning.
- c. Start the *WriteToKafka* processor using the right-click menu or the *Start* icon in the configuration drawer.

Congratulations! You have successfully customized this ReadyFlow and achieved your goal of sending critical alerts to a dedicated topic! Now that you are done with developing your flow, it is time to deploy it in production!

1.3 Publishing your flow to the catalog

1. Stop the Test Session

- a. Click the toggle next to *Active Test Session* to stop your Test Session



- b. Click “End” in the dialog to confirm. The Test Session is now stopping and allocated resources are being released



2. Publish your modified flow to the Catalog

- a. Open the “Flow Options” menu at the top
- b. Click “Publish” to make your modified flow available in the Catalog
- c. Prefix your username to the Flow Name and provide a Flow Description. Click **Publish**.

The screenshot shows the 'Publish A New Flow' dialog box. It has three main input fields:

- Flow Name:** vetticadentest1_kafkafilterkafka-vetticadentest1 (highlighted with a blue border)
- Flow Description:** Flow that identifies high severity events and writes them to a dedicated Kafka topic.
- Version Comments:** Initial Version

At the bottom of the dialog are two buttons: 'Cancel' and 'Publish' (highlighted with a blue border).

i.

- d. You are now redirected to your published flow definition in the Catalog.

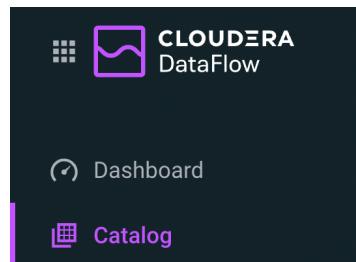
The screenshot shows the Cloudera DataFlow Catalog interface. At the top, there's a header with a refresh icon and the text "REFRESHED: 18 seconds ago". Below the header, the flow name is "vetticadentest1_kafkafilterkafka-vetticadentest1" with a subtitle "Updated a minute ago by George Vetticaden". To the right is an "Actions" dropdown menu. A "FLOW DESCRIPTION" box contains the text: "Flow that identifies high severity events and writes them to a dedicated Kafka topic." and the CRN: "crn:cdp:df:us-west-1:5251b921-84c5-45c4-af51-c0b8a6ebd1c9:flow:vetticadentest1_kafkafilterkafka-vetticadentest1". Below the description is a checkbox "Only show deployed versions". A table below shows one version (Version 1, Deployments 0, Associated Drafts 1). At the bottom are buttons for "Deploy →", "Download", and "Create New Draft".

Info: The Catalog is the central repository for all your deployable flow definitions. From here you can create auto-scaling deployments from any version or create new drafts and update your flow processing logic to create new versions of your flow.

1.4 Creating an auto-scaling flow deployment

1. As soon as you publish your flow, it should take you to the Catalog. If it does not then locate your flow definition in the Catalog

- a. Make sure you have navigated to the Catalog



- b. If you have closed the sidebar, search for your published flow <>yourid<> into the search bar in the Catalog. Click on the flow definition that matches the name you

gave it earlier.

The screenshot shows the NiFi Flow Catalog interface. On the left is a sidebar with icons for creating new flows, managing environments, and other actions. The main area has a search bar at the top with the query 'Kafka filter to Kafka - mkohs'. Below the search is a table with a single row: 'Name ↑' followed by 'Kafka filter to Kafka - mkohs'. To the right of the table is a detailed view of the flow. It includes a 'REFRESHED: 5 seconds ago' timestamp, a 'Actions' dropdown, a 'FLOW DESCRIPTION' section with a brief summary, a 'CRN #', and a link to the full CRN. There is also a checkbox for 'Only show deployed versions'. Below this is a table with three columns: 'Version' (1), 'Deployments' (0), and 'Associated Drafts' (1). At the bottom are buttons for 'Deploy →', 'Download', and 'Create New Draft'.

- c. After opening the side panel, click *Deploy*, select the available environment from the drop down menu and click *Continue* to start the Deployment Wizard.
- d.

The screenshot shows the 'New Deployment' wizard. The title is 'New Deployment' and there is a close button 'X'. The first section is 'Select the target environment' with a note: '① Sensitive data never leaves the environment. Changing the environment after this step requires restarting the deployment process.' The second section is 'Selected Flow Definition' which shows a table with one row: NAME 'tim_kafkafilterkafka' and VERSION '1'. The third section is 'Target Environment' which shows a dropdown menu with 'aws oss-demo-aws' selected and '60% (3 of 5)' progress. At the bottom are 'Cancel' and 'Continue →' buttons.

If you have any issues, log out, close your browser, restart your browser, try an incognito window and re-login. Also see the “Best Practices Guide”.

2. Complete the Deployment Wizard

The Deployment Wizard guides you through a six step process to create a flow deployment. Throughout the six steps you will choose the NiFi configuration of your flow,

provide parameters and define KPIs. At the end of the process, you are able to generate a CLI command to automate future deployments.

Note: The Deployment name has a cap of 27 characters which needs to be considered as you write the prod name.

- a. Provide a name such as <>your_username>>_kafkatokafka_prod to indicate the use case and that you are deploying a production flow. Click *Next*.

Overview

Deployment Name

 Deployment name is valid

Selected Flow Definition

NAME	VERSION
tim_kafkafilterkafka	1

Target Environment

aws	NAME
	oss-demo-aws

- b. The **NiFi Configuration** screen allows you to customize the runtime that will execute your flow. You have the opportunity to pick from various released NiFi versions.

Select the *Latest Version* and make sure *Automatically start flow upon successful deployment* is checked.

Click *Next*.

- c. The **Parameters** step is where you provide values for all the parameters that you defined in your flow. In this example, you should recognize many of the prefilled values from the previous exercise - including the *Filter Rule* and our *Kafka Source* and *Kafka Destination Topics*.

To advance, you have to provide values for all parameters. Select the *No Value* option to only display parameters without default values.

SHOW: Sensitive No value

You should now only see one parameter - the *CDP Workload User Password* parameter which is sensitive. Sensitive parameter values are removed when you

publish a flow to the catalog to make sure passwords don't leak.

Provide your *CDP Workload User Password* and click *Next* to continue.

SHOW: Sensitive No value

Kafka filter to Kafka (12)

CDP Workload User Password [?](#)

13/100K

Cancel [← Previous](#) [Next →](#)

- d. The **Sizing & Scaling** step lets you choose the resources that you want to allocate for this deployment. You can choose from several node configurations and turn on Auto-Scaling.

Let's choose the *Extra Small* Node Size and turn on *Auto-Scaling* from 1-3 nodes. Click *Next* to advance.

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

NiFi Node Sizing [?](#)

<input checked="" type="radio"/> Extra Small	<input type="radio"/> Small	<input type="radio"/> Medium	<input type="radio"/> Large
2 vCores Per Node 4 GB Per Node	3 vCores Per Node 6 GB Per Node	6 vCores Per Node 12 GB Per Node	12 vCores Per Node 24 GB Per Node

Number of NiFi Nodes

Auto Scaling [?](#)

Enabled

Min. Nodes: - Max. Nodes:

Cancel [← Previous](#) [Next →](#)

- e. The **Key Performance Indicators (KPI)** step allows you to monitor flow performance. You can create KPIs for overall flow performance metrics or

in-depth processor or connection metrics.

Add the following KPI

- KPI Scope: **Entire Flow**
- Metric to Track: **Data Out**
- Alerts:
 - Trigger alert when metric is less than: **1 MB/sec**
 - Alert will be triggered when metrics is outside the boundary(s) for: **1 Minute**

Add new KPI X

Details

KPI Scope [?](#)

Metric to Track [?](#)

METRIC DESCRIPTION:
Rate of data sent to external source in bytes

Alerts

Trigger alert when metric is greater than

Trigger alert when metric is less than

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

[Cancel](#)

Add the following KPI

- KPI Scope: **Processor**
- Processor Name: **ConsumeFromKafka**
- Metric to Track: **Bytes Received**
- Alerts:
 - Trigger alert when metric is less than: **512 KBytes/sec**
 - Alert will be triggered when metrics is outside the boundary(s) for: **30 seconds**

Add new KPI

Details

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

Metric to Track [?](#)

METRIC DESCRIPTION:
Number of bytes received from a source that is external from the flows

Alerts

Trigger alert when metric is greater than
 Trigger alert when metric is less than

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

Review the KPIs and click **Next**.

	Entire Flow METRIC TO TRACK Data Out ALERT SET Notify if less than 1 MB/sec, for at least 1 minutes.	 
	Processor: ConsumeFromKafka METRIC TO TRACK Bytes Received ALERT SET Notify if less than 512 KBytes, for at least 30 seconds.	 

- f. 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:

- i. Click on the **>_ View CLI Command** link and familiarize yourself with the content.
- ii. Download the 2 JSON dependency files by click on the download button:
 1. Flow Deployment Parameters JSON

2. Flow Deployment KPIs JSON

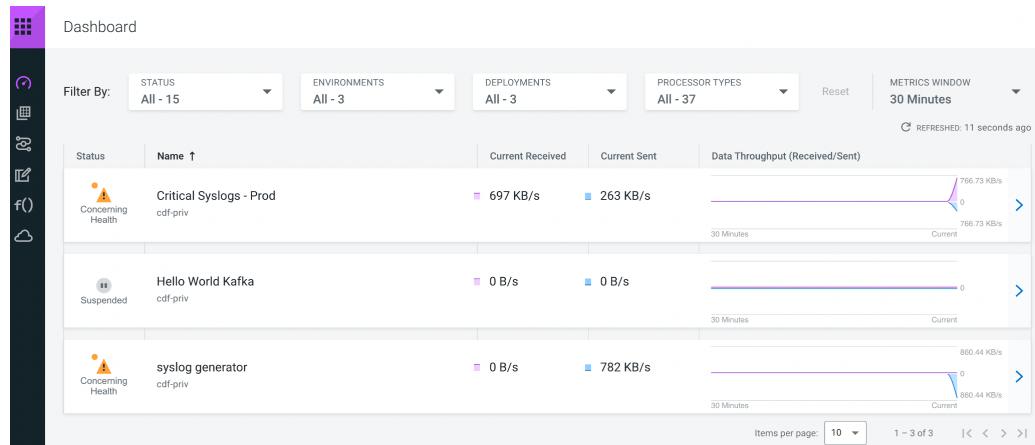
- iii. Copy the command at the end of this page and save that in a file called **deploy.sh**
- iv. Close the **Equivalent CDP CLI Command** tab.
- g. Click **Deploy** to initiate the flow deployment!
- h. You are redirected to the Deployment Dashboard where you can monitor the progress of your deployment. Creating the deployment should only take a few minutes.

The screenshot shows the Cloudera DataFlow Deployment Dashboard. On the left sidebar, there are links for Dashboard, Catalog, ReadyFlow Gallery, Flow Design, Functions, and Environments. The main dashboard area has a 'Filter By' dropdown set to 'STATUS All - 15'. It displays several deployment cards:

- Critical Syslogs - Prod**: Status is 'Deploying', Processor Type is 'aws cdf-priv'.
- Hello World Kafka**: Status is 'Suspended', Processor Type is 'cdf-priv'.
- syslog generator**: Status is 'Concerning Health', Processor Type is 'cdf-priv'.

On the right side, there's a section for 'Critical Syslogs - Prod' with tabs for 'KPIs', 'System Metrics', and 'Alerts' (which is selected). It shows 'No alerts to display.' Below that is an 'Event History' section with checkboxes for 'Info', 'Warning', and 'Error' levels, and a timestamp 'Deployment Initiated 2023-04-07 17:49 PDT'.

- i. Congratulations! Your flow deployment has been created and is already processing Syslog events!



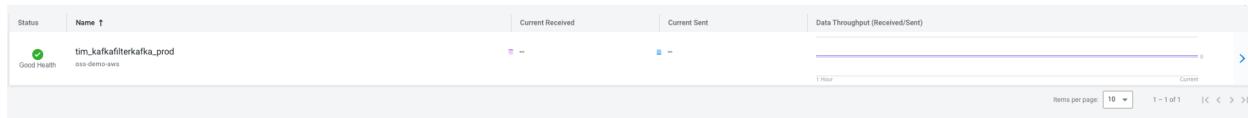
Please wait until your application is done **Deploying, Importing Flow. Wait for Good Health.**

This screenshot shows the Deployment Dashboard with the following filters: 'STATUS All - 15', 'ENVIRONMENTS All - 1', 'DEPLOYMENTS 1 of 2', and 'PROCESSOR TYPES All - 32'. There is one active deployment:

- tim_kafkafilterkafka_prod**: Status is 'Deploying', Processor Type is 'cdf-demo-aws'. Current Received: --, Current Sent: --.

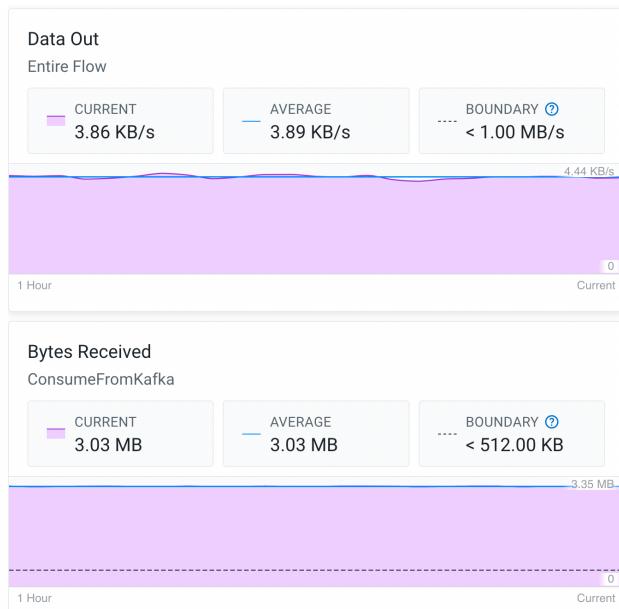
The interface includes a 'Metrics Window' dropdown set to '1 Day' and a 'Load More' button.

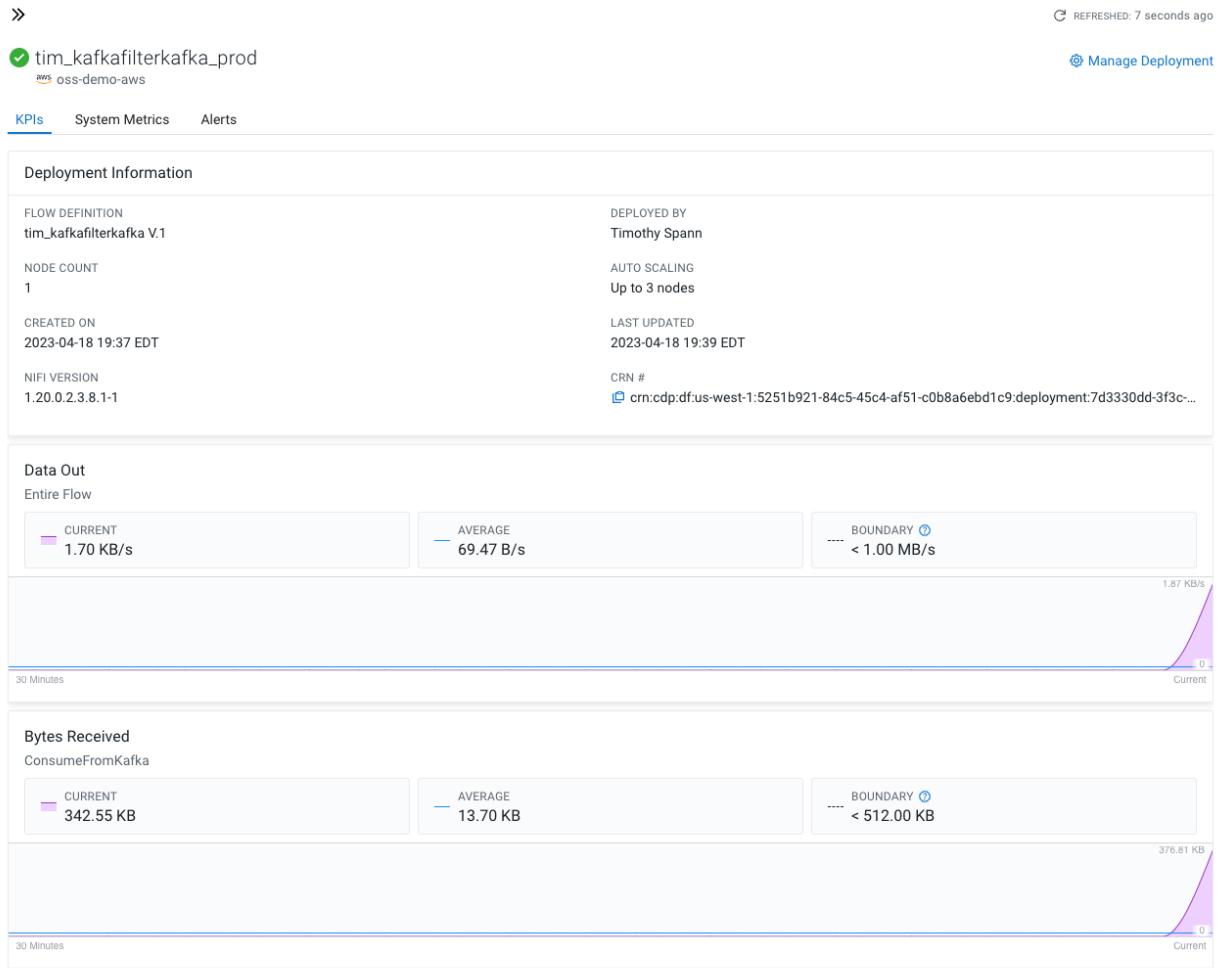
1.5 Monitoring your flow deployment



1. Notice how the dashboard shows you the data rates at which a deployment currently receives and sends data. The data is also visualized in a graph that shows the two metrics over time.
2. Change the *Metrics Window* setting at the top right. You can visualize as much as 1 Day.
3. Click on the **yourid_kafkafilterkafka_prod** deployment. The side panel opens and shows more detail about the deployment. On the *KPIs* tab it will show information about the KPIs that you created when deploying the flow.

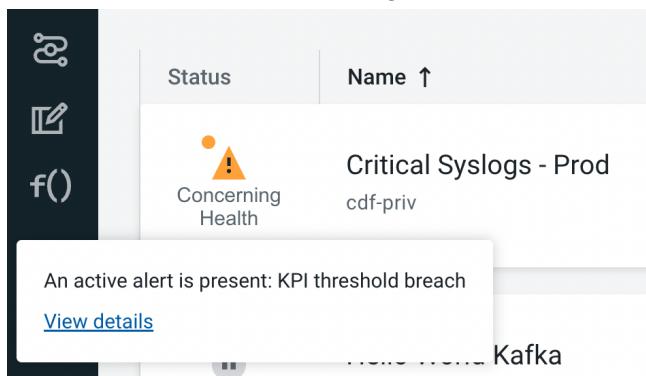
Using the two KPIs *Bytes Received* and *Data Out* we can observe that our flow is filtering out data as expected since it reads more than it sends out.





Wait a number of minutes so some data and metrics can be generated.

- Switch to the *System Metrics* tab where you can observe the current CPU utilization rate for the deployment. Our flow is not doing a lot of heavy transformation, so it should hover around at ~10% CPU usage.
- Close the side panel by clicking anywhere on the Dashboard.
- Notice how your `yourid_Critical/SyslogsProd` deployment shows *Concerning Health* status. Hover over the warning icon and click *View Details*.



7. You will be redirected to the *Alerts* tab of the deployment. Here you get an overview of active and past alerts and events. Expand the Active Alert to learn more about its cause.

The screenshot shows a table titled "Active Alerts". The columns are "Alert Type" and "Alert Time ↓". There is one row visible, which is expanded to show details. The alert type is "KPI Threshold Breach" and it occurred "3 days ago". The alert details state: "Rate of data sent to external source in bytes is 0.00 B/s, less than the boundary of 1MB/s". A link "Learn more" is provided. The first occurrence was on "2023-04-07 17:53 PDT".

After expanding the alert, it is clear that it is caused by a KPI threshold breach for sending less than 1MB/s to external systems as defined earlier when you created the deployment.

1.6 Managing your flow deployment

1. Click on the *yourid_kafkafilterkafka_prod* deployment in the Dashboard. In the side panel, click *Manage Deployment* at the top right.

The screenshot shows the Deployment Manager interface. At the top, there is a header with two arrows and a refresh icon. Below the header, there is a section for "Critical Syslogs - Prod" with an "aws cdf-priv" badge. To the right, there is a link to "Manage Deployment". At the bottom, there are tabs for "KPIs", "System Metrics", and "Alerts", with "Alerts" being the active tab.

2. You are now being redirected to the *Deployment Manager*. The Deployment Manager allows you to reconfigure the deployment and modify KPIs, modify the number of NiFi nodes or turn auto-scaling on/off or update parameter values.

Dashboard / Critical Syslogs - Prod / Deployment Manager

Deployment Manager

DEPLOYMENT NAME: Critical Syslogs - Prod

FLOW DEFINITION: Kafka filter to Kafka - mkohs V1

DEPLOYED BY: Michael Kohs

LAST UPDATED: 2023-04-07 17:51 PDT

CRN #: crn:cdp:df:us-west-1:9d74eee4-1cad-45d7-b645...

STATUS: Concerning Health

NODE COUNT: 1

AUTO SCALING: Up to 3 nodes

CREATED ON: 2023-04-07 17:49 PDT

REGION: US West (Oregon)

NIFI RUNTIME VERSION: 1.20.0.2.3.9.0-8

ENVIRONMENT: cdf-priv

Recreate Deployment CLI Command

Deployment Settings

Parameters

Running Processors that are affected by the Parameter changes will automatically be restarted.

Data entered here never leaves the environment in your cloud account. Provide parameter values directly in the text input or upload a file for parameters that expect a file.

The selected flow definition references an external Default Nifi SSL Context Service. Hence, DataFlow will automatically create a matching SSL Context Service with a keystore and truststore generated from the target environment's FreeIpa certificate.

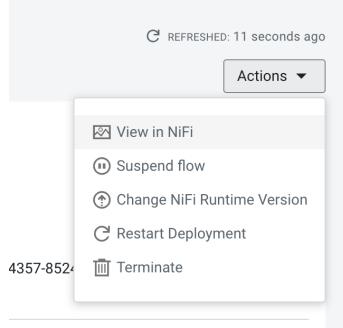
Kafka filter to Kafka (12)

CDP Workload User

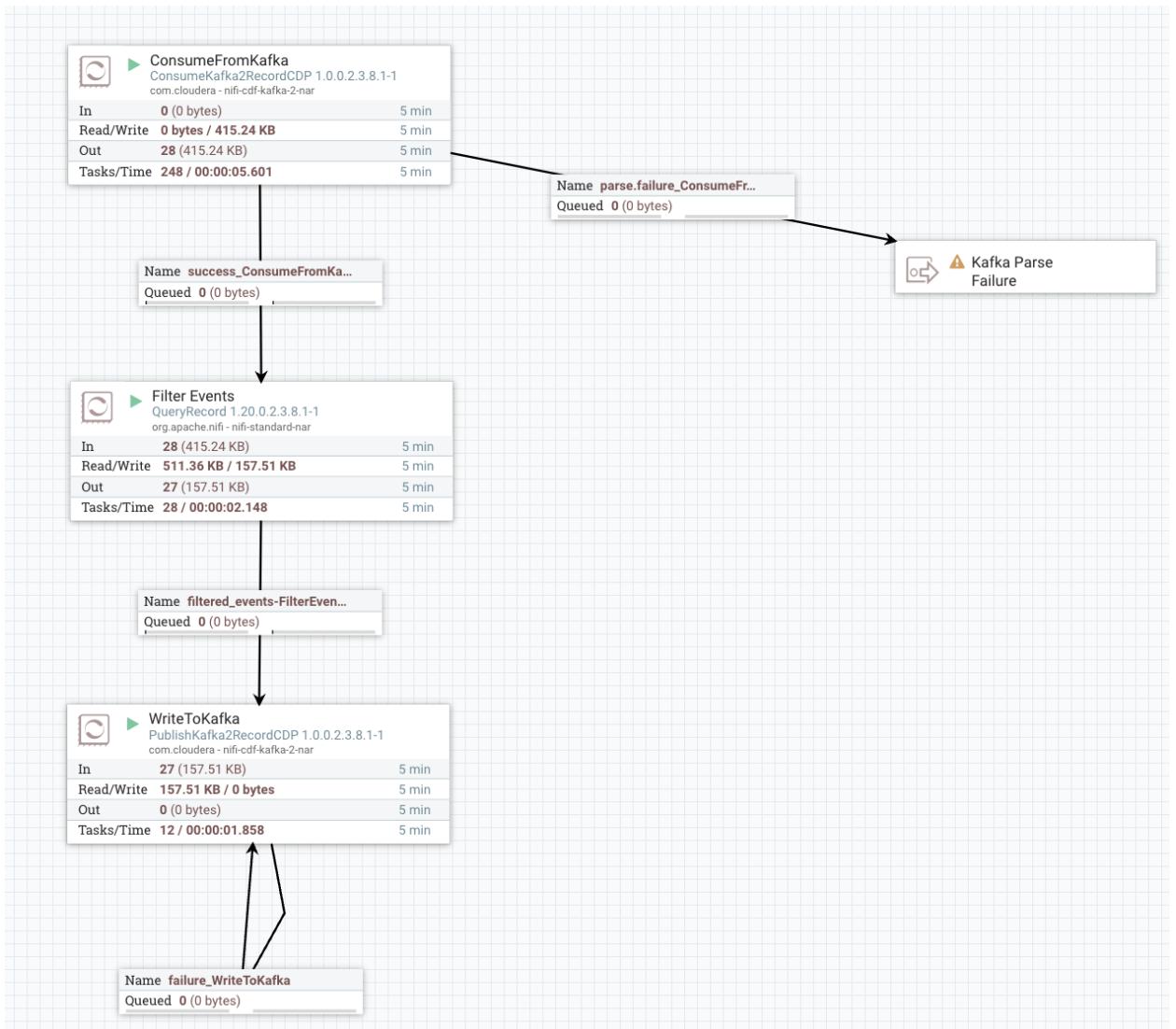
srv_nifi-kafka-ingest

21/100K

3. Explore NiFi UI for deployment. Click the *Actions* menu and click on *View in NiFi*.



4. You are being redirected to the NiFi cluster running the flow deployment. You can use this view for in-depth troubleshooting. Users can have read-only or read/write permissions to the flow deployment.



»

REFRESHED: 3 seconds ago

tim_kafkafilterkafka_prod

aws oss-demo-aws

Manage Deployment

KPIs System Metrics Alerts

Deployment Information

FLOW DEFINITION

tim_kafkafilterkafka V.1

DEPLOYED BY

Timothy Spann

NODE COUNT

1

AUTO SCALING

Up to 3 nodes

CREATED ON

2023-04-18 19:37 EDT

LAST UPDATED

2023-04-18 19:39 EDT

NIFI VERSION

1.20.0.2.3.8.1-1

CRN

[crn:cdp:df:us-west-1:5251b921-84c5-45c4-af51-c0b8a6ebd1c9:deployment:7d3330dd-3f3c...](#)

Data Out

Entire Flow

CURRENT
114.23 B/s

AVERAGE
88.23 B/s

BOUNDARY ⓘ
< 1.00 MB/s

30 Minutes

1.94 KB/s
0 Current

Bytes Received

ConsumeFromKafka

CURRENT
425.60 KB

AVERAGE
62.29 KB

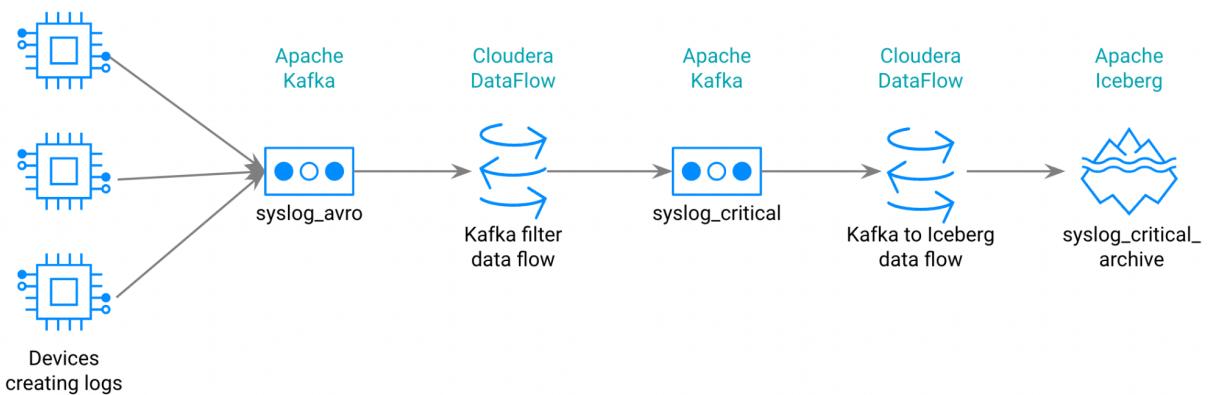
BOUNDARY ⓘ
< 512.00 KB

30 Minutes

468.16 KB
0 Current

2. Writing critical syslog events to Apache Iceberg for analysis

A few weeks have passed since you built your data flow with DataFlow Designer to filter out critical syslog events to a dedicated Kafka topic. Now that everyone has better visibility into real-time health, management wants to do historical analysis on the data. Your company is evaluating Apache Iceberg to build an open data lakehouse and you are tasked with building a flow that ingests the most critical syslog events into an Iceberg table.



Ensure your table is built and accessible.

Create an Apache Iceberg Table

1. From the Home page, click the Data Hub Clusters. Navigate to **oss-kudu-demo** from the Data Hubs list

The screenshot shows the Cloudera Management Console interface with the following details:

- Left Sidebar:** Includes links for Dashboard, Environments, Data Lakes, User Management, Data Hub Clusters (highlighted), Data Warehouses, MC Workspaces, Classic Clusters, Audit, Shared Resources, and Global Settings.
- Data Hubs Section:** A table listing running Data Hubs:

Status	Name	Cloud Provider	Environment	Data Hub Type	Version	Node Count	Created
Running	oss-kafka-dagthen	aws	oss-demo-aws	7.2.16 - Streams Messaging Light Duty, Apache Kafka, Schema Registry, Streams Messaging Manager, Streams Replication Manager, Cruise Control	CDH 7.2.16	4	04/19/23, 12:52 PM EDT
Running	oss-flink-demo	aws	oss-demo-aws	7.2.16 - Streaming Analytics Light Duty with Apache Flink	CDH 7.2.16	6	03/21/23, 03:29 PM EDT
Running	oss-kafka-demo	aws	oss-demo-aws	7.2.16 - Streams Messaging Light Duty, Apache Kafka, Schema Registry, Streams Messaging Manager, Streams Replication Manager, Cruise Control	CDH 7.2.16	4	03/21/23, 03:29 PM EDT
Running	oss-kudu-demo	aws	oss-demo-aws	7.2.16 - Real-time Data Mkt, Apache Impala, Hue, Apache Kudu, Apache Spark	CDH 7.2.16	7	03/21/23, 03:29 PM EDT
Running	oss-nifi-demo	aws	oss-demo-aws	7.2.16 - Flow Management Light Duty with Apache Nifi, Apache Nifi Registry	CDH 7.2.16	4	03/21/23, 03:29 PM EDT
- Bottom Right:** Buttons for 'Create Data Hub' and 'Create Cluster'.

2. Navigate to **Hue** from the Kudu Data Hub.



- Inside of **Hue** you can now create your table. You will have your own database to work with. To get to your database, click on the '<' icon next to default database. You should see your specific database in the format: <YourEmailWithUnderscores>_db. Click on your database to go to the SQL Editor.

- Create your Apache Iceberg table with the sql below and clicking the play icon to execute the sql query. Note that the the table name must prefixed with your **Work Load User Name (userid)**.

```
CREATE TABLE <>userid>>_syslog_critical_archive
(priority int, severity int, facility int, version int, event_timestamp bigint, hostname string,
body string, appName string, procid string, messageid string,
structureddata struct<sdid:struct<eventid:string,eventsources:string,iut:string>>)
STORED BY ICEBERG;
```

The screenshot shows the Impala interface. In the top navigation bar, there is a search bar with the placeholder "Search data and saved documents...". Below the search bar, the database name "vetticadentest3_gmail_com_db" is selected. The main area displays a table named "vetticadentest3_syslog_critical_archive". The table definition is shown in the code editor:

```
1 CREATE TABLE vetticadentest3_syslog_critical_archive
2   priority int, severity int, facility int, version int, event_timestamp bigint, hostname string,
3   body string, appName string, procid string, messageid string,
4   structureddata struct<sdid:string,struct<eventid:string,eventsources:string,iut:string>>
5   STORED BY 'ICEBERG';
6
```

Below the code editor, a message states "No logs available at this moment." To the right, a green URL link is visible: 7748d23082ef6bb7:9ae2365a00000000. At the bottom, there are tabs for "Query History", "Saved Queries", and "Results (1)". The "Results (1)" tab is active, showing a summary message: "Table has been created."

5. Once you have sent data to your table, you can query it.

The screenshot shows the Impala interface. The database "vetticadentest3_gmail_com_db" is selected. A query is being run against the table "vetticadentest3_syslog_critical_archive":

```
1
2
3 select * from vetticadentest3_syslog_critical_archive;
```

The results of the query are displayed in a message box:

Query c941b4cad5a8f212:d48d4ae500000000 100% Complete (0 out of 0)
Query c941b4cad5a8f212:d48d4ae500000000 100% Complete (0 out of 0)

At the bottom, a message indicates "Done. 0 results." and a green URL link is visible: c941b4cad5a8f212:d48d4ae500000000.

Additional Documentation

- [Create a Table](#)
- [Query a Table](#)
- [Apache Iceberg Table Properties](#)

2.1 Open ReadyFlow & start Test Session

1. Navigate to **DataFlow** from the Home Page
2. Navigate to the **ReadyFlow Gallery**
3. Explore the ReadyFlow Gallery

4. Search for the “Kafka to Iceberg” ReadyFlow.

The screenshot shows the NiFi ReadyFlow Gallery interface. On the left is a sidebar with icons for Home, Recent, Catalog, and a search bar. The main area has a search bar at the top with the text 'Iceberg'. Below it, a flow titled 'Kafka to Iceberg' is listed under the 'Added' category. The flow diagram consists of three nodes: a 'Kafka' source node, a 'Batching' processor, and a 'PutIceberg' target node. Below the flow, the description reads: 'Consumes JSON, CSV or Avro events from Kafka and writes them as Parquet files to a destination Iceberg table.' At the bottom of the card are two buttons: 'View Added Flow Definition' and 'Create New Draft'.

5. Click on “Create New Draft” to open the ReadyFlow in the Designer named **yourid_kafkatoiceberg Ex: tim_kafkatoiceberg**
6. Start a Test Session by either clicking on the *start a test session* link in the banner or going to *Flow Options* and selecting *Start* in the Test Session section.
7. In the Test Session creation wizard, select the latest NiFi version and click *Start Test Session*. Notice how the status at the top now says “Initializing Test Session”.

2.2 Modifying the flow to read syslog data

The flow consists of three processors and looks very promising for our use case. The first processor reads data from a Kafka topic, the second processor gives us the option to batch up events and create larger files which are then written out to Iceberg by the *PutIceberg* processor. All we have to do now to reach our goal is to customize its configuration to our use case.

1. Provide values for predefined parameters

- a. Navigate to *Flow Options*→ *Parameters*
- b. Select all parameters that show *No value set* and provide the following values

Name	Description	Value
CDP Workload User	CDP Workload User	<Your own workload user name>
CDP Workload User Password	CDP Workload User Password	<Your own workload user password>
Data Input Format	This flow supports AVRO, JSON and CSV	JSON

Hive Catalog Namespace		<YourEmailWithUnderScores_db>
Iceberg Table Name		<>replace_with_userid>>_syslog_critical_archive
Kafka Broker Endpoint	Comma-separated list of Kafka Broker addresses	oss-kafka-demo-corebroker2.oss-demo.qsm5-opic.cloudera.site:9093, oss-kafka-demo-corebroker1.oss-demo.qsm5-opic.cloudera.site:9093, oss-kafka-demo-corebroker0.oss-demo.qsm5-opic.cloudera.site:9093
Kafka Consumer Group Id		<>replace_with_userid>>_cdf Ex: tim_cdf
Kafka Source Topic		<>replace_with_userid>>_syslog_critical Ex: tim_syslog_critical
Schema Name		syslog
Schema Registry Hostname		oss-kafka-demo-master0.oss-demo.qsm5-opic.cloudera.site

c. Click *Apply Changes* to save the parameter values

2. Start Controller Services

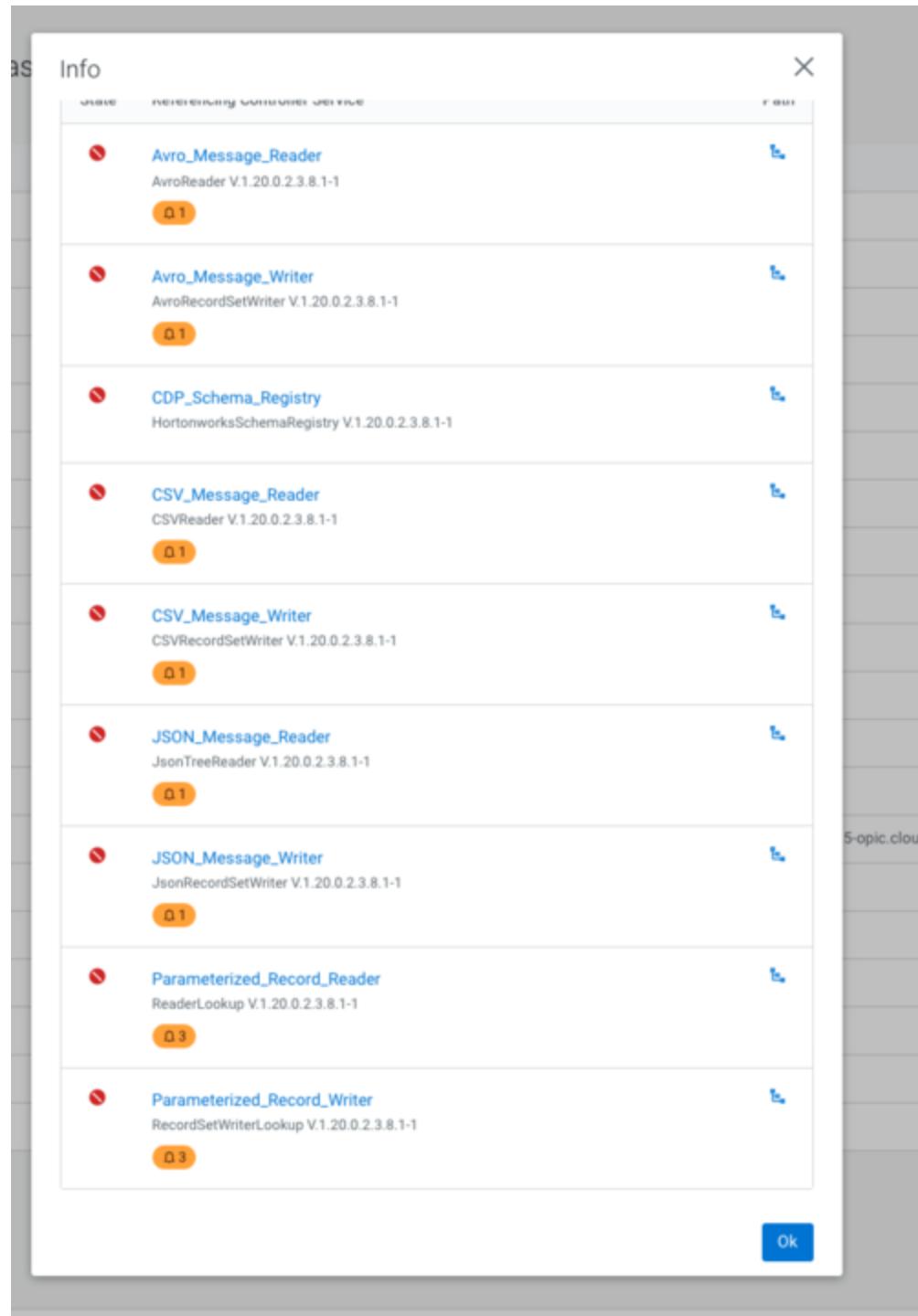
a. Navigate to *Flow Options → Services*

- b. Select *CDP_Schema_Registry* service and click *Enable Service and Referencing Components* action

Service and Referencing Con



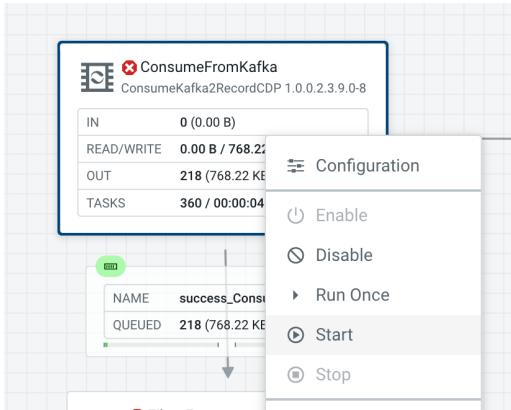
- c. Start from the top of the list and enable all remaining Controller services including KerberosPasswordUserService, HiveCatalogService, AvroReader, ...
- d. Click *Ok* if confirmation is asked.



- e. Make sure all services have been enabled

All Services for Kafka to Iceberg			
State	Name ↑	Type	Notifications
✓	AvroReader	AvroReader V.1.20.0.2.3.8.1-1	>
✓	Avro_Message_Reader	AvroReader V.1.20.0.2.3.8.1-1	>
✓	Avro_Message_Writer	AvroRecordSetWriter V.1.20.0.2.3.8.1-1	>
✓	CDP_Schema_Registry	HortonworksSchemaRegistry V.1.20.0.2.3.8.1-1	>
✓	CSV_Message_Reader	CSVReader V.1.20.0.2.3.8.1-1	>
✓	Default NiFi SSL Context Service	StandardRestrictedSSLContextService V.1.20.0.2....	>
✓	HiveCatalogService	HiveCatalogService V.1.20.0.2.3.8.1-1	>
✓	JSON_Message_Reader	JsonTreeReader V.1.20.0.2.3.8.1-1	>
✓	JsonRecordSetWriterOriginalSchema	JsonRecordSetWriter V.1.20.0.2.3.8.1-1	>
✓	KerberosPasswordUserService	KerberosPasswordUserService V.1.20.0.2.3.8.1-1	>
✓	Parameterized_Message_Reader	ReaderLookup V.1.20.0.2.3.8.1-1	>

3. Start the **ConsumeFromKafka processor** using the right click action menu or the *Start* button in the configuration drawer. It might already be started.



After starting the processor, you should see events starting to queue up in the *success_ConsumeFromKafka-FilterEvents* connection.

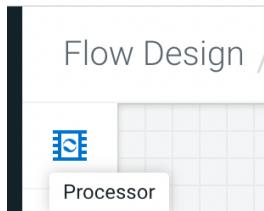
NOTE:

To receive data from your topic, you will need either the first deployment still running or to run it from another Flow Designer Test Session.

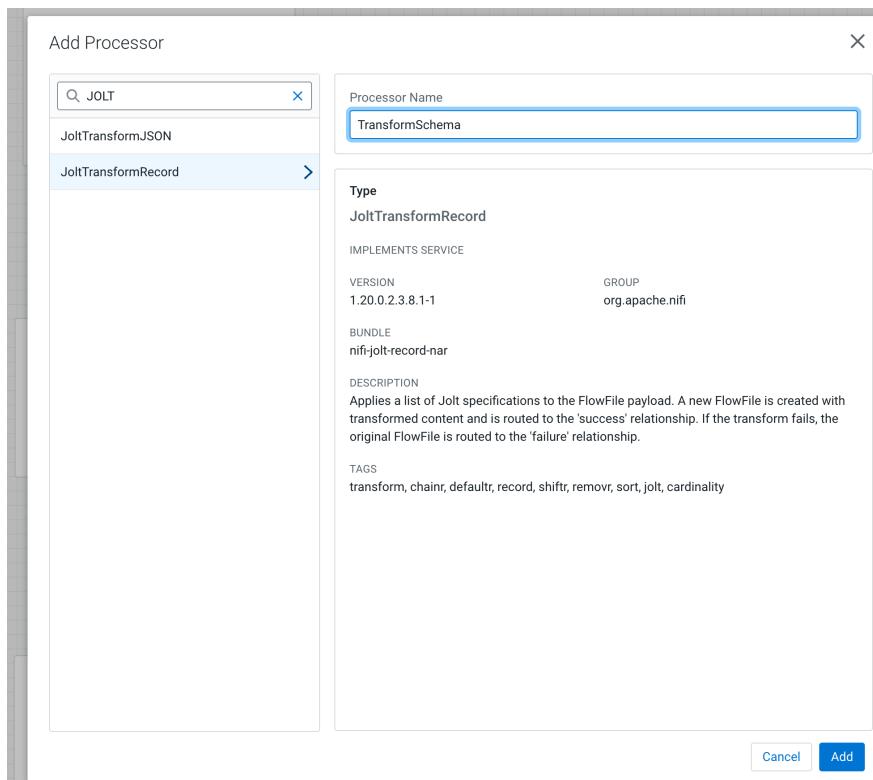
2.3 Changing the flow to modify the schema for Iceberg integration

Our data warehouse team has created an Iceberg table that they want us to ingest the critical syslog data in. A challenge we are facing is that not all column names in the Iceberg table match our syslog record schema. So we have to add functionality to our flow that allows us to change the schema of our syslog records. To do this, we will be using the *JoltTransformRecord* processor.

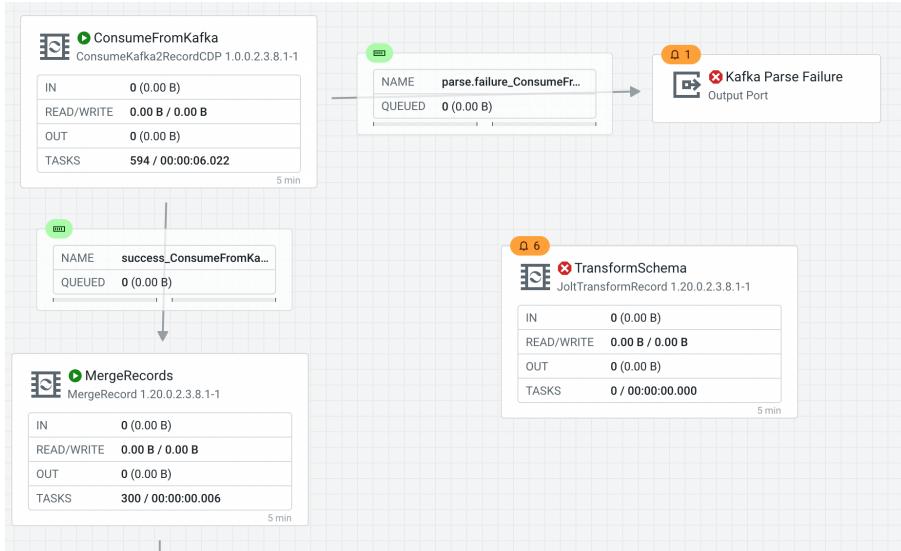
1. Add a new *JoltTransformRecord* to the canvas by dragging the processor icon to the canvas.



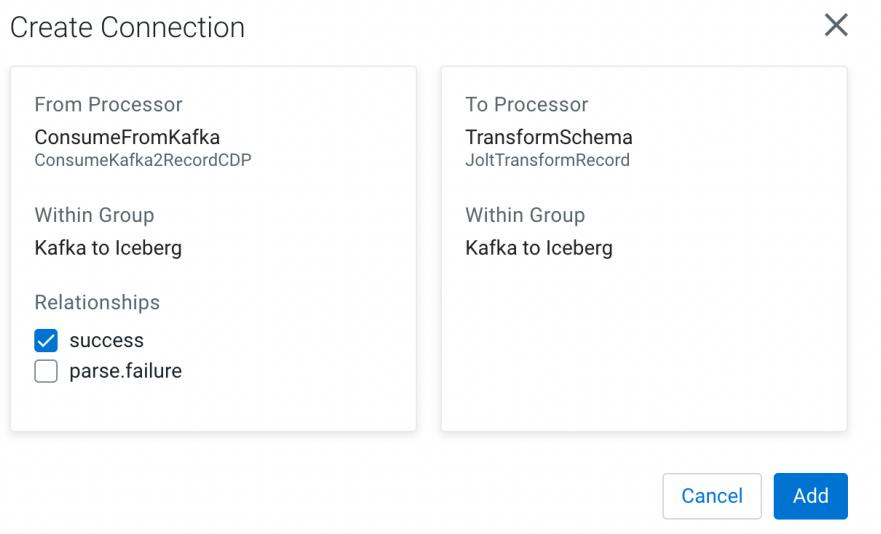
2. In the *Add Processor* window, select the *JoltTransformRecord* type and name the processor *TransformSchema*.



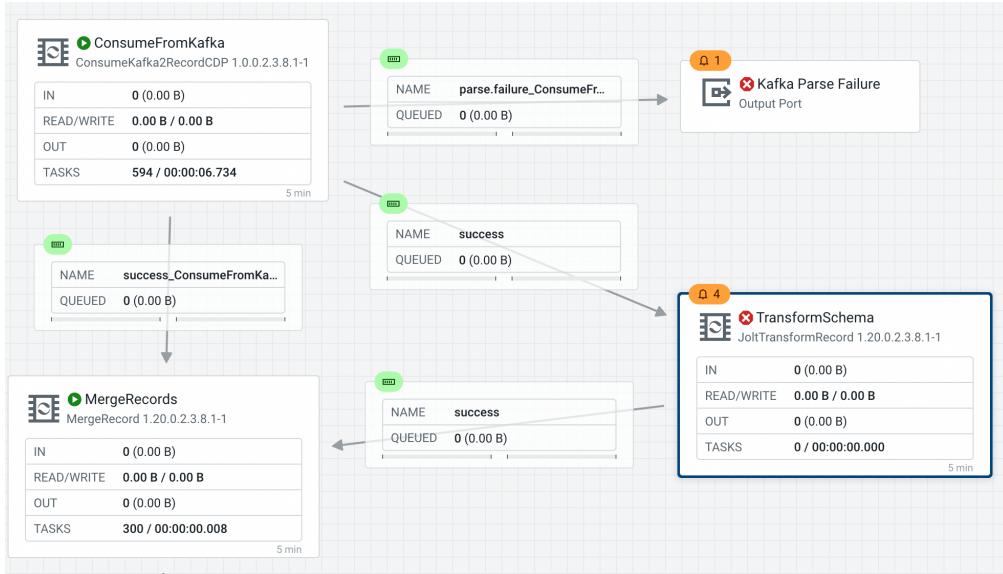
3. Validate that your new processor now appears on the canvas.



4. Create connections from *ConsumeFromKafka* to *TransformSchema* by hovering over the *ConsumeFromKafka* processor and dragging the arrow that appears to *TransformSchema*. Pick the **success** relationship to connect.

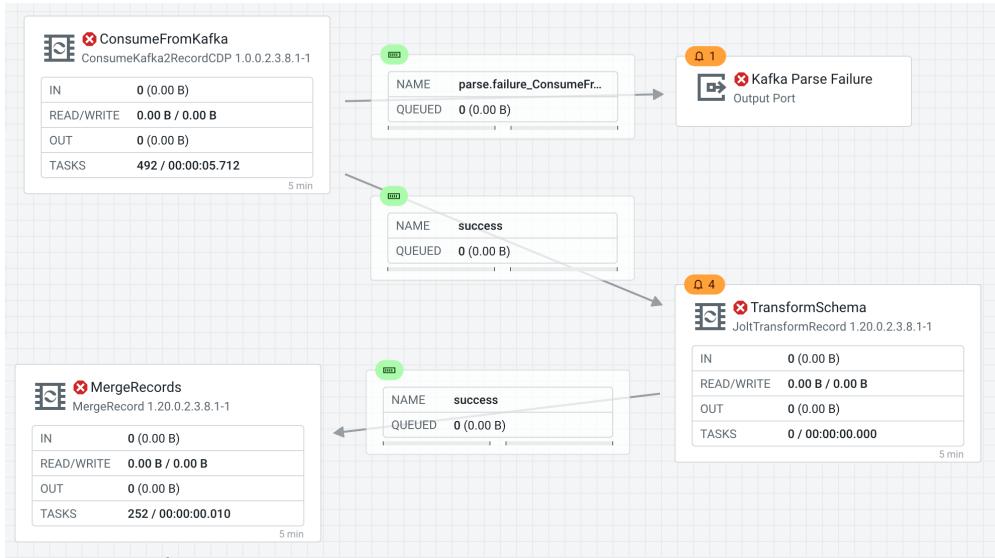


Now connect the *success* relationship of *TransformSchema* to the *MergeRecords* processor.



- Now that we have connected our new *TransformSchema* processor, we can delete the original connection between *ConsumeFromKafka* and *MergeRecords*.

Make sure that the *ConsumeFromKafka* processor is stopped. Then select the connection, empty the queue if needed, and then delete it. Now all syslog events that we receive, will go through the *TransformSchema* processor.



- To make sure that our schema transformation works, we have to create a new *Record Writer Service* and use it as the Record Writer for the *TransformSchema* processor.

Select the *TransformSchema* processor and open the configuration panel. Scroll to the

Properties section, click the three dot menu in the *Record Writer* row and select *Add Service* to create a new Record Writer.

Properties		
Property	Value	
Record Reader ?	No value set	⋮
Record Writer ?	No value set	⋮
Jolt Transformation DSL ?	Chain	Add Service
Jolt Specification ?	No value set	Go To Service
Transform Cache Size ?	1	⋮

7. Select *AvroRecordSetWriter*, name it *TransformedSchemaWriter* and click *Add*.

Click *Apply* in the configuration panel to save your changes.

Add Service

🔍

🔍

🔍

🔍

🔍

🔍

🔍

🔍

🔍

🔍

🔍

🔍

🔍

🔍

🔍

🔍

🔍

🔍

🔍

🔍

🔍

🔍

🔍

🔍

🔍

🔍

🔍

🔍

🔍

🔍

🔍

🔍

🔍

🔍

🔍

🔍

🔍

🔍

🔍

🔍

🔍

🔍

🔍

🔍

🔍

🔍

🔍

🔍

🔍

🔍

🔍

🔍

🔍

🔍

🔍

🔍

🔍

🔍

🔍

🔍

🔍

🔍

🔍

🔍

🔍

🔍

🔍

🔍

🔍

🔍

🔍

🔍

🔍

🔍

🔍

🔍

🔍

🔍

🔍

🔍

🔍

🔍

🔍

🔍

🔍

🔍

🔍

🔍

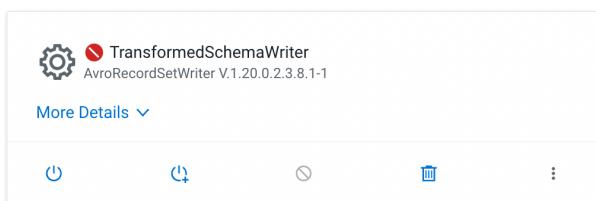
🔍

🔍

🔍

<input type="text" style="width: 100%; height: 30px; border: none; outline: none; padding: 5px 10px; font-size: 14px; color: #333; border-bottom:

Schema Write Strategy	Specify whether/how CDF should write schema information	Embed Avro Schema
Schema Access Strategy	Specify how CDF identifies the schema to apply.	Use 'Schema Name' Property
Schema Registry	Specify the Schema Registry that stores our schema	CDP_Schema_Registry
Schema Name	The schema name to look up in the Schema Registry	syslog_transformed



The screenshot shows the configuration page for the TransformedSchemaWriter service. At the top, it displays the service icon (gear with a red dot), the name 'TransformedSchemaWriter', and the version 'AvroRecordSetWriter V.1.20.0.2.3.8.1-1'. Below this are standard UI controls: a 'More Details' link, a power button, a refresh button, a settings gear, a trash can, and a three-dot menu.

Settings

*Service Name: TransformedSchemaWriter

Comments: (empty text area)

Properties

Property	Value
Schema Write Strategy ?	Embed Avro Schema :
Schema Cache ?	No value set :
Schema Access Strategy ?	Use 'Schema Name' Property :
Schema Registry ?	CDP_Schema_Registry :
Schema Name ?	syslog_transformed :

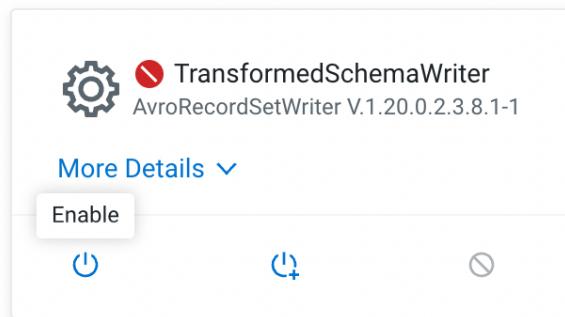
10. Convert the value that you provided for *Schema Name* into a parameter. Click on the three dot menu in the *Schema Name* row and select *Convert To Parameter*.

Schema Name	?	syslog_transformed	⋮
Schema Version	?	No value set	Convert To Parameter
Default Reader	⋮	All Readers	⋮

11. Give the parameter the name *Schema Name Transformed* and click “add”. You have now created a new parameter from a value that can be used in more places in your data flow.

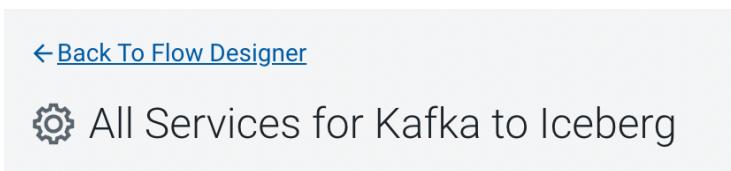
Schema Name	?	↪ #{{Schema Name Transformed}}	⋮
-------------	-------------------	--------------------------------	---

12. **Apply** your configuration changes and *Enable* the Service by clicking the power icon. Now you have configured our new Schema Writer and we can return back to the Flow Designer canvas.



If you have any issues, end the test session and restart. If your login timed out, close your browser and re login.

13. Click *Back to Flow Designer* to navigate back to the canvas.



14. Select *TransformSchema* to configure it and provide the following values:

Name	Description	Value
------	-------------	-------

Record Reader	Service used to parse incoming events	AvroReader
Record Writer	Service used to format outgoing events	TransformedSchemaWriter
Jolt Specification	The specification that describes how to modify the incoming JSON data. We are standardizing on lower case field names and renaming the <i>timestamp</i> field to <i>event_timestamp</i> .	[{ "operation": "shift", "spec": { "appName": "appname", "timestamp": "event_timestamp", "structuredData": { "SDID": { "eventId": "structureddata.sdid.eventid", "eventSource": "structureddata.sdid.eventsource", "iut": "structureddata.sdid.iut" } }, "*": { "@": "&" } } }]]

15. Scroll to *Relationships* and select *Terminate* for the *failure*, *original* relationships and click *Apply*.

Relationships

You can choose to automatically term both terminate and retry are selected,

success [?](#)

Terminate Retry

failure [?](#)

Terminate Retry

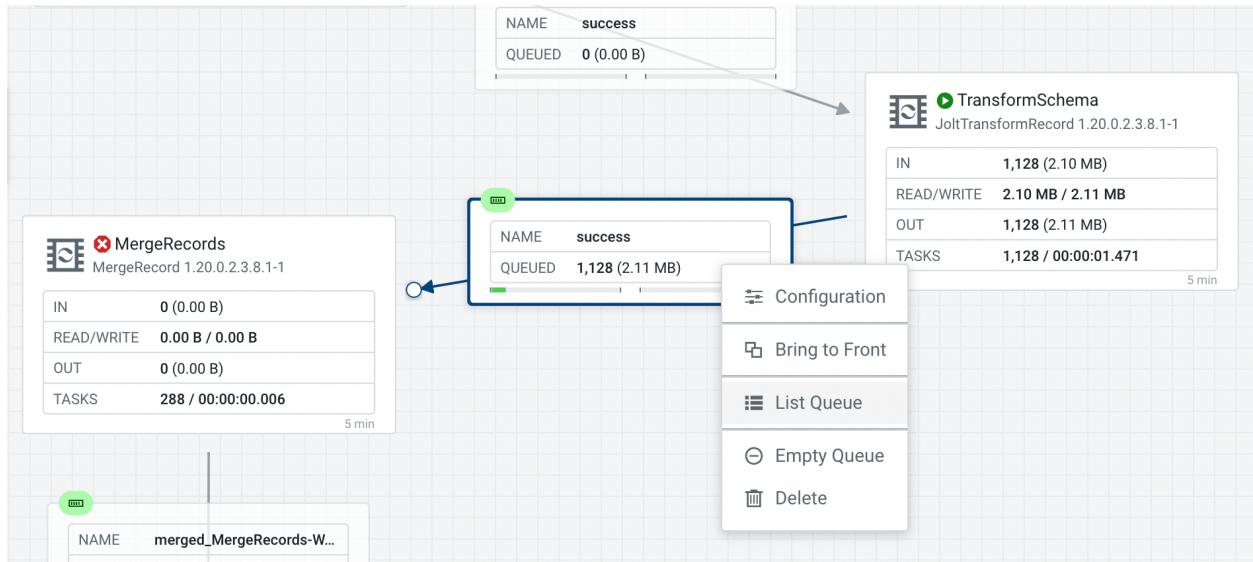
original [?](#)

Terminate Retry

Retry logic specified below will apply t

16. Start your *ConsumeFromKafka* and *TransformSchema* processor and validate that the transformed data matches our Iceberg table schema.

17. Once events are queuing up in the connection between *TransformSchema* and *MergeRecord*, right click the connection and select *List Queue*.



18. Select any of the queued files and select the book icon to open it in the Data Viewer

The screenshot shows the Flink Data Stream Processing interface with the 'List Queue' view open. The top bar shows the path: Flow Design / pm-sandbox-aws / Kafka to Iceberg / List Queue. The main area displays a table of queued files:

All Queued Flow Files for Success Relationship(s)						
Position	Filename	UUID	File Size	Queue Duration	Lineage Duration	Penalized
1	6b59473b-ff7e-41d9-89c1-473a67a6f2cf	aa0fadfb-18ec-4458-a2c3-2141cdb509f6	3 KB	00:01:07.868	00:07:25.966	No

To the right, the Data Viewer panel shows the contents of the selected file (the first row from the table):

```

6b59473b-ff7e-41d9-89c1-473a67a6f2cf
aa0fadfb-18ec-4458-a2c3-2141cdb509f6
More Details ▾
Open in Data Viewer

```

19. Notice how all field names have been transformed to lower case and how the *timestamp* field has been renamed to *event_timestamp*.

The screenshot shows the Flink Data Stream Processing interface with the Data Viewer panel open. The content pane displays the following JSON data:

```

1 v [ {
2   "priority" : 89,
3   "severity" : 1,
4   "facility" : 11,
5   "version" : 1,
6   "event_timestamp" : 1681340412516,
7   "hostname" : "host3.example.com",
8   "body" : "application6 has exited cleanly",
9   "appname" : "application6",
10  "procid" : "7556",
11  "messageid" : "ID20",
12  "structureddata" : {
13    "sdid" : {
14      "eventid" : "79",
15      "eventsource" : "kernel",
16      "iut" : "2"
17    }
18  }
}

```

2.4 Merging records and start writing to Iceberg

Now that we have verified that our schema is being transformed as needed, it's time to start the remaining processors and write our events into the Iceberg table. The *MergeRecords* processor is configured to batch events up to increase efficiency when writing to Iceberg. The final processor, *WriteToIceberg* takes our Avro records and writes them into a Parquet formatted table.

1. Tip: You can change the configuration to something like “**30 sec**” to speed up processing.
2. Select the *MergeRecords* processor and explore its configuration. It is configured to batch events up for at least 30 seconds or until the queued up events have reached *Maximum Bin Size* of 1GB. You will want to lower these for testing.

Properties

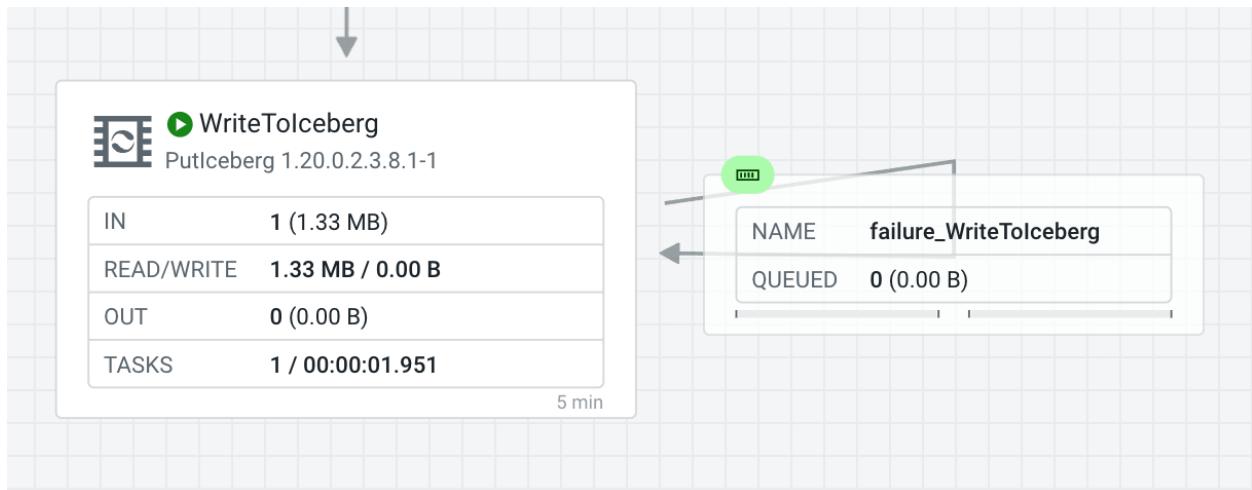
Property	Value
Record Reader ?	AvroReader
Record Writer ?	Avro_Message_Writer
Merge Strategy ?	Bin-Packing Algorithm
Correlation Attribute Name ?	No value set
Attribute Strategy ?	Keep Only Common Attributes
Minimum Number of Records ?	1
Maximum Number of Records ?	100000000
Minimum Bin Size ?	100 MB
Maximum Bin Size ?	1 GB
Max Bin Age ?	5 min
Maximum Number of Bins ?	5

3. Start the *MergeRecords* processor and verify that it batches up events and writes them out after 30 seconds.
4. Select the *WriteToIceberg* processor and explore its configuration. Notice how it relies on several parameters to establish a connection to the right database and table.

Properties

Property	Value
Record Reader ?	↳ AvroReader
Catalog Service ?	↳ HiveCatalogService
Catalog Namespace ?	↳ #{Hive Catalog Namespace}
Table Name ?	↳ #{Iceberg Table Name}
File Format ?	PARQUET
Maximum File Size ?	No value set
Kerberos User Service ?	↳ KerberosPasswordUserService

5. Start the *WriteTolceberg* processor and verify that it writes records successfully to Iceberg. If the metrics on the processor increase and you don't see any warnings or events being written to the *failure_WriteTolceberg* connection, your writes are successful!



Congratulations! With this you have completed the second use case.

You may want to log into Hue to check your data has loaded.

The screenshot shows the DataBrick interface with the following details:

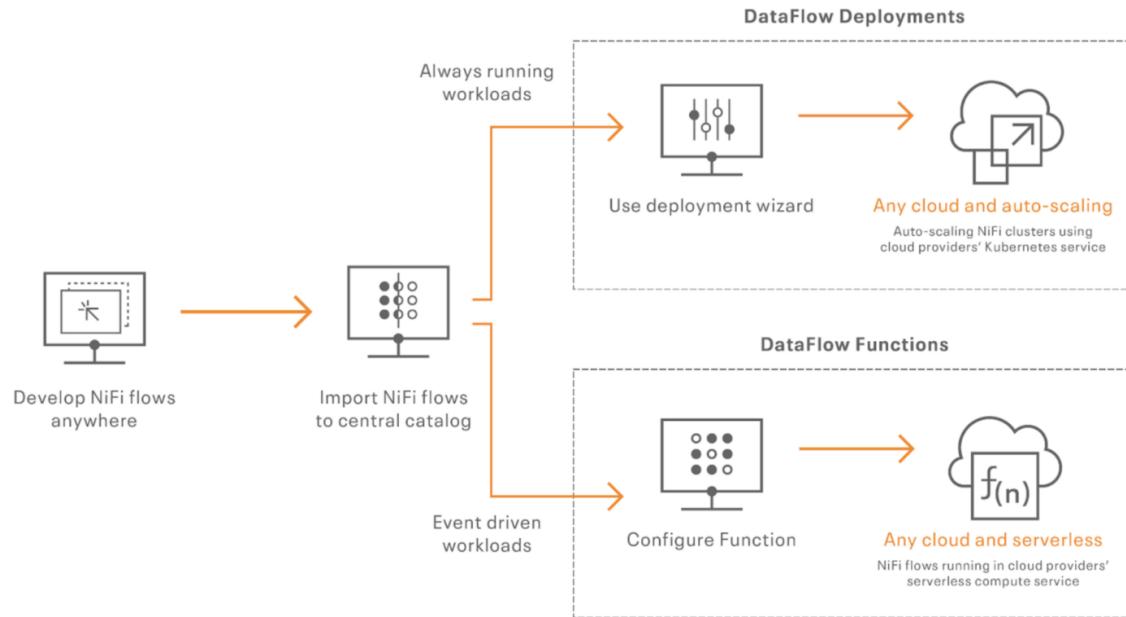
- Left Sidebar:** Contains icons for Home, Cluster, Databricks ML, Databricks ML Flow, Databricks ML Pipeline, Databricks ML Model, Databricks ML Model Flow, Databricks ML Pipeline Flow, and Help.
- Top Bar:** Shows the cluster name "Impala" and a search bar with placeholder "Search data and saved documents...".
- Cluster Status:** Shows "0.68s default" with a gear icon.
- Tables:** A list of tables under the "default" database, including "ahen_syslog_critical_archive2", "alekhanan_syslog_critical_archive", "christophersyslog_critical_archive", "gregs_syslog_critical_archive2", "jason_syslog_critical_archive", "lauridsyslog_critical_archive", "hankasyslog_critical_archive", "nirchisyslog_critical_archive2", "pavinsyslog_critical_archive", "poumonesyslog_critical_archive", "syslog_critical_archive", "syslog_critical_archive_mikohs", "tim_syslog_critical_archive", "tim_syslog_critical_archive2", and "youuserid_syslog_critical_archive".
- Query Editor:** Displays the SQL query: "select * from tim_syslog_critical_archive order by event_timestamp desc".
- Results:** A table titled "Results (227)" showing 227 rows of log entries. The columns are:

	priority	severity	facility	version	event_timestamp	hostname	body	appname	procid	messagedId
1	56	0	7	1	1681866875503	host10.example.com	application7 has stopped unexpectedly	application7	7997	ID34
2	42	2	5	1	1681866875502	host7.example.com	application4 has exited cleanly	application4	5652	ID34
3	138	2	17	1	1681866875465	host3.example.com	application3 has stopped unexpectedly	application3	4643	ID38
4	9	1	1	1	1681866875465	host4.example.com	application9 has stopped unexpectedly	application9	5193	ID10
5	48	0	6	1	1681866875464	host9.example.com	application9 has started successfully	application9	2177	ID30
6	97	1	12	1	1681866875464	host1.example.com	application4 has stopped unexpectedly	application4	2031	ID24
7	105	1	13	1	1681866875412	host2.example.com	application6 has exited cleanly	application6	2624	ID39
8	185	1	23	1	1681866875412	host3.example.com	application1 has exited cleanly	application1	9931	ID21
9	120	0	15	1	1681866875411	host1.example.com	application6 has stopped unexpectedly	application6	4891	ID32
10	74	2	9	1	1681866785476	host1.example.com	application9 has exited cleanly	application9	8852	ID28
11	89	1	11	1	1681866785476	host1.example.com	application6 has stopped unexpectedly	application6	1599	ID13
12	74	2	9	1	1681866785475	host1.example.com	application9 has completed gracefully	application9	4930	ID34
13	113	1	14	1	1681866785475	host1.example.com	application6 has exited cleanly	application6	5560	ID18
14	9	1	1	1	1681866785438	host1.example.com	application9 has exited cleanly	application9	8115	ID2
15	114	2	14	1	1681866785438	host2.example.com	application10 has stopped unexpectedly	application10	2579	ID21
16	178	2	22	1	1681866785437	host3.example.com	application7 has exited cleanly	application7	834	ID1
17	104	0	13	1	1681866785382	host9.example.com	application5 has stopped unexpectedly	application5	3544	ID42
18	185	1	23	1	1681866785382	host1.example.com	application3 has completed gracefully	application3	8641	ID43
19	0	0	0	1	1681866785382	host3.example.com	application4 has started successfully	application4	8664	ID7
20	153	1	19	1	1681866785382	host3.example.com	application10 has stopped unexpectedly	application10	6211	ID4
21	129	1	16	1	1681866785381	host1.example.com	application5 has started successfully	application5	8691	ID39
22	178	2	22	1	1681866785339	host3.example.com	application3 has exited cleanly	application3	1760	ID18
23	40	0	5	1	1681866785339	host3.example.com	application9 has exited cleanly	application9	9290	ID36
24	66	2	8	1	1681866785338	host2.example.com	application5 has started successfully	application5	5519	ID26
25	153	1	19	1	1681866995429	host10.example.com	application8 has started successfully	application8	6563	ID14
26	178	2	22	1	1681866995429	host10.example.com	application8 has stopped unexpectedly	application8	9254	ID16
27	186	2	23	1	1681866695407	host4.example.com	application5 has completed gracefully	application5	3957	ID38
28	65	1	8	1	1681866695407	host3.example.com	application10 has exited cleanly	application10	8320	ID11

Feel free to publish your flow to the catalog and create a deployment just like you did for the first one.

3. Resize image flow deployed as serverless function

DataFlow Functions provides a new, efficient way to run your event-driven Apache NiFi data flows. You can have your flow executed within AWS Lambda, Azure Functions or Google Cloud Functions and define the trigger that should start its execution.



DataFlow Functions is perfect for use cases such as:

- Processing files as soon as they land into the cloud provider object store
- Creating microservices over HTTPS
- CRON driven use cases
- etc

In this use case, we will be deploying a NiFi flow that will be triggered by HTTPS requests to resize images. Once deployed, the cloud provider will provide an HTTPS endpoint that you'll be able to call to send an image, it will trigger the NiFi flow that will return a resized image based on your parameters.

The deployment of the flow as a function will have to be done within your cloud provider.

The below tutorial will use AWS as the cloud provider. If you're using Azure or Google Cloud, you can still refer to this [documentation](#) to deploy the flow as a [function](#).

3.1 Designing the flow for AWS Lambda

1. Go into Cloudera DataFlow / Flow Design and create a new draft with a name of your choice.
2. Drag and drop an Input Port named **input** onto the canvas. When triggered, AWS Lambda is going to inject into that input port a FlowFile containing the information about the HTTPS call that has been made.

Example of payload that will be injected by AWS Lambda as a FlowFile:

```

1  {
2    "version": "2.0",
3    "routeKey": "ANY /NaaF_S3_compression",
4    "rawPath": "/default/NaaF_S3_compression",
5    "rawQueryString": "",
6    "headers": {
7      "accept-encoding": "gzip",
8      "content-length": "34",
9      "content-type": "application/json",
10     "date": "Fri, 06 Aug 2021 12:35:29 GMT",
11     "host": "ns9dgd2tw6.execute-api.us-east-2.amazonaws.com",
12     "x-amzn-trace-id": "Root=1-610d2c92-7651f91a5dcbab8671048936",
13     "x-forwarded-for": "3.19.60.232",
14     "x-forwarded-port": "443",
15     "x-forwarded-proto": "https"
16   },
17   "requestContext": {
18     "accountId": "381358652250",
19     "apiId": "ns9dgd2tw6",
20     "domainName": "ns9dgd2tw6.execute-api.us-east-2.amazonaws.com",
21     "domainPrefix": "ns9dgd2tw6",
22     "http": {
23       "method": "POST",
24       "path": "/default/NaaF_S3_compression",
25       "protocol": "HTTP/1.1",
26       "sourceIp": "3.19.60.232",
27       "userAgent": ""
28     },
29     "requestId": "DpPm3j83iYcEP7w=",
30     "routeKey": "ANY /NaaF_S3_compression",
31     "stage": "default",
32     "time": "06/Aug/2021:12:35:30 +0000",
33     "timeEpoch": 1628253330104
34   },
35   "body": "{\n    \"data\": \"this is my payload\"\n}",
36   "isBase64Encoded": false
37 }
```

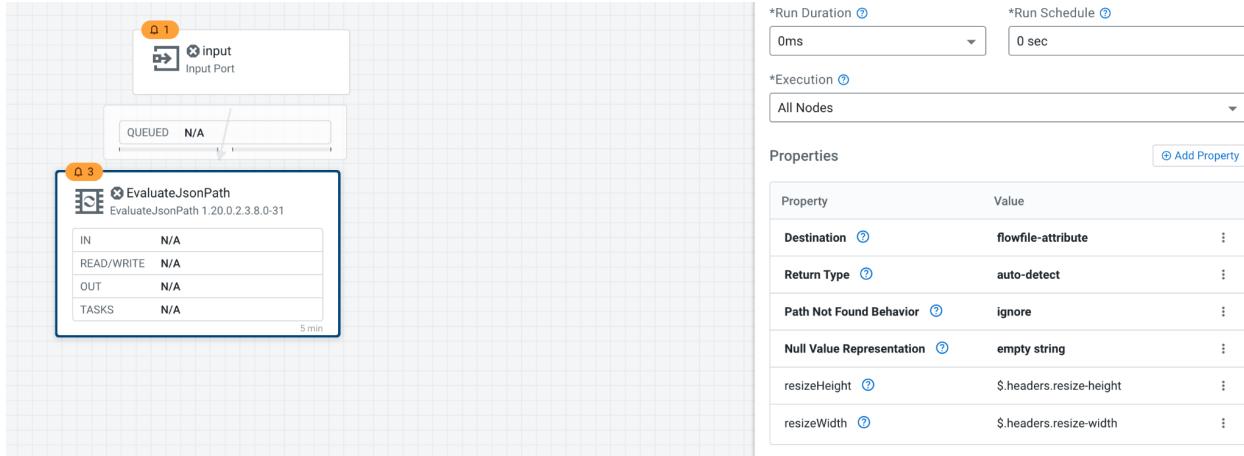
3. Drag and drop an EvaluateJsonPath processor, call it ExtractHTTPHeaders. We're going to use this to extract the HTTP headers that we want to keep in our flow. Add two properties configured as below. It'll save as FlowFile's attributes the HTTP headers (resize-height and resize-width) that we will be adding when making a call with our image to specify the dimensions of the resized image.

```

resizeHeight => $.headers.resize-height
resizeWidth => $.headers.resize-width

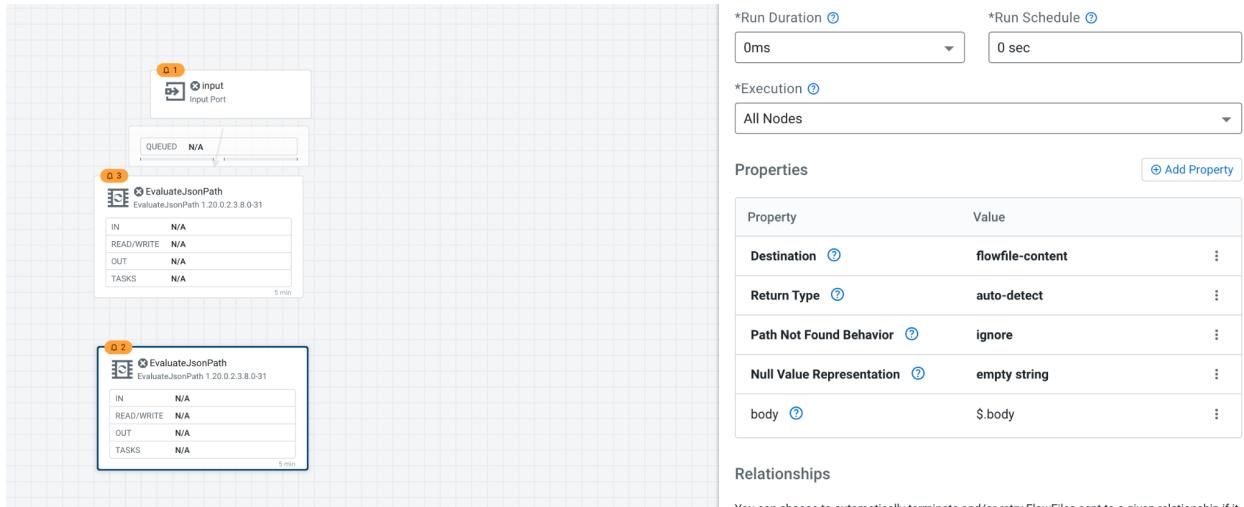
```

Note: don't forget to change **Destination** as "flowfile-attribute" and Click **Apply**.



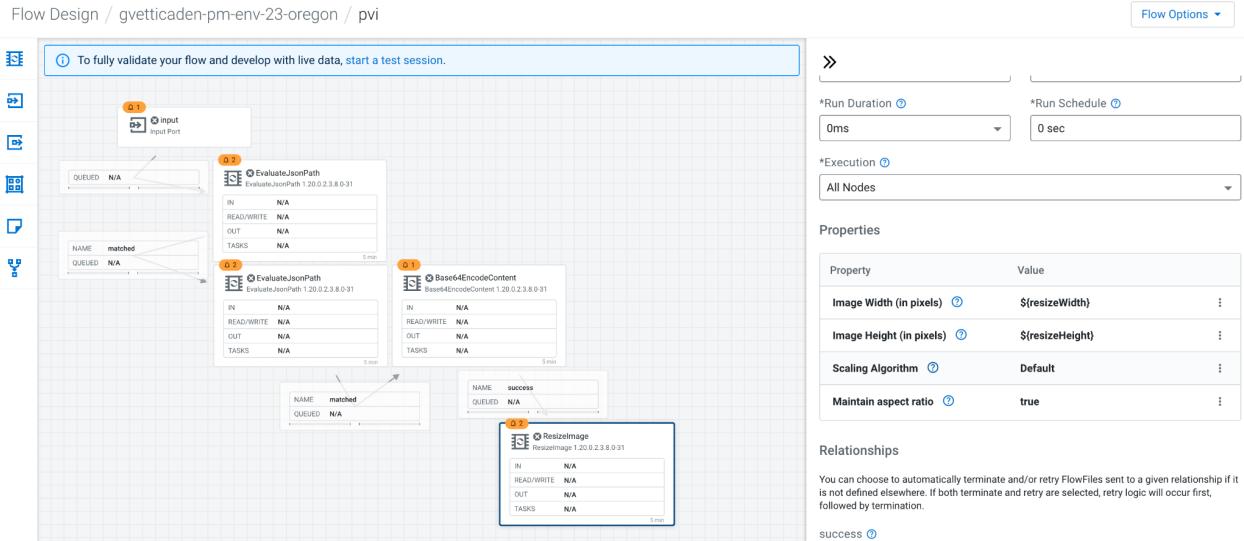
- Drag and drop another EvaluateJsonPath processor and then change its name to a unique one. This one will be used to retrieve the content of the body field from the payload we received and use it as the new content of the FlowFile. This field contains the actual representation of the image we have been sending over HTTP with Base 64 encoding.

```
body => $.body
```



- Drag and drop a Base64EncodeContent processor and change the mode to Decode. This will Base64 decode the content of the FlowFile to retrieve its binary format.

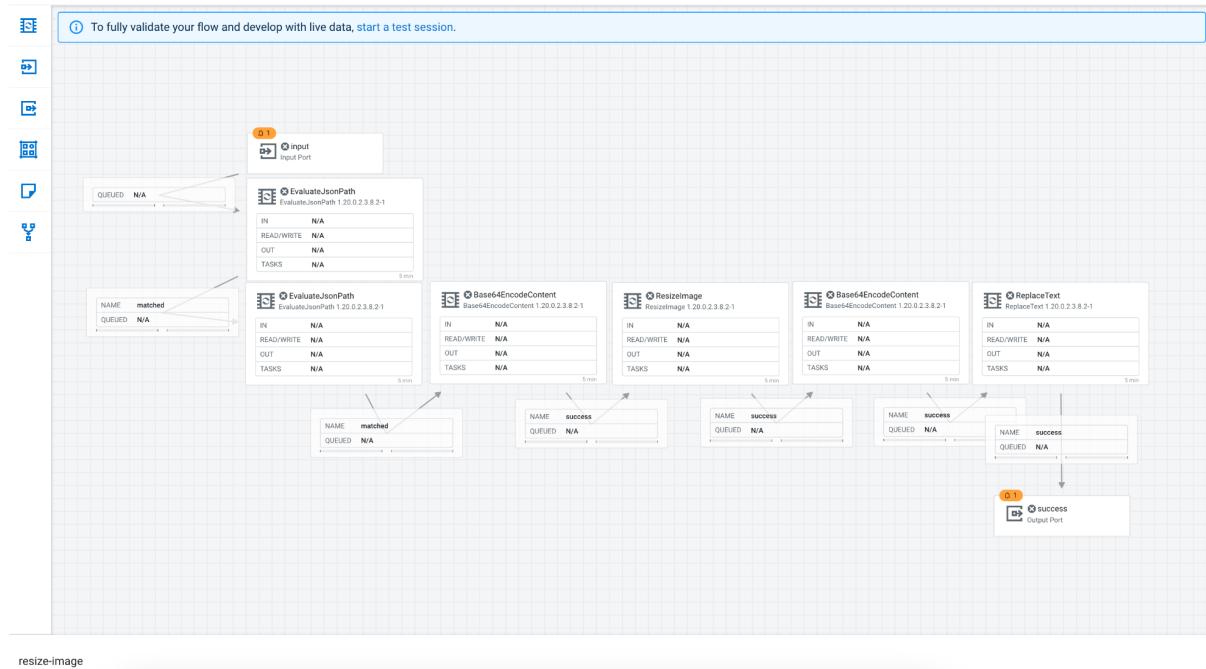
6. Drag and drop a ResizeImage processor. Use the previously created FlowFile attributes to specify the new dimensions of the image. Also, specify true for maintaining the ratio.



7. Drag and drop a Base64EncodeContent processor. To send back the resized image to the user, AWS Lambda expects us to send back a specific JSON payload with the Base 64 encoding of the image.
8. Drag and drop a ReplaceText processor. We use it to extract the Base 64 representation of the resized image and add it in the expected JSON payload. Add the below JSON in "Replacement Value" and change "Evaluation Mode" to "Entire text".

```
{
    "statusCode": 200,
    "headers": { "Content-Type": "image/png" },
    "isBase64Encoded": true,
    "body": "$1"
}
```

9. Drag and drop an output port.
10. Connect all the components together, you can auto-terminate the unused relationships. This should look like this:



resize-image

You can now publish the flow into the DataFlow Catalog in the Flow Options menu:

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](#)

[Publish the flow to the catalog](#)

Edit

- [Services](#)
- [Parameters](#)

View

- [Documentation](#)

Exiting

- [Exit Draft](#)

Resizelimage
Resizelimage 1.20.0.2.3.8.2-1

Base64EncodeContent
Base64EncodeContent 1.20.0.2.3.8.2-1

ReplaceText
ReplaceText 1.20.0.2.3.8.2-1

Make sure to give it a name that is unique (you can prefix it with your name):

Flow Catalog

Search by name

Import Flow Definition

REFRESHED: 25 seconds ago

Name ↑	Type	Versions	Last Updated
000-abc1	Custom Flow Definition	1	a year ago
000-import-test-1	Custom Flow Definition	1	a year ago
000-test-1	Custom Flow Definition	2	a year ago
1-2 automatic autoscaling	Custom Flow Definition	4	a year ago
aaa	Custom Flow Definition	1	a year ago
aaaa	Custom Flow Definition	1	6 months ago
abchs	Custom Flow Definition	1	8 months ago
abchs1	Custom Flow Definition	1	8 months ago
ABROWN DFX Test	Custom Flow Definition	2	2 years ago

Publish A New Flow

Flow Name: **pvillard-meet-the-committers-resize-image**

Flow Description: Add description

Version Comments: Initial Version

Cancel Publish

Once the flow is published, make sure to copy the CRN of the published version (it will end by /v.1):

The screenshot shows the DataFlow Functions interface. At the top right, there is a refresh icon and the text "REFRESHED: 9 seconds ago". Below the title, there is a "Actions" button with a dropdown arrow. The flow name is "pvillard-meet-the-committers-resize-image" and it was updated 10 seconds ago by Pierre Villard. The flow description is "No description specified". The CRN is listed as "crn:cdp:df:us-west-1:558bc1d2-8867-4357-8524-311d51259233:flow:pvillard-meet-the-committers-resize-image". There is a checkbox labeled "Only show deployed versions". A table below shows one version (v1) with 0 deployments and 1 associated draft. Buttons for "Deploy →", "Download", and "Create New Draft" are present. The "Associated Drafts" section shows one draft named "prm-sandbox-aws" with a "resize-image" step. The "CRN #" is also listed. The "Created" section shows the date and time as "2023-04-28 17:49 EEST" by Pierre Villard, with a note "Initial Version".

3.2 Deploying the flow as a function in AWS Lambda

First thing first, go into DataFlow Functions and download the binary for running DataFlow Functions in AWS Lambda:

The screenshot shows the 'Functions' section of the Cloudera DataFlow interface. It features three cards for different cloud providers:

- AWS Lambda**: Represented by an orange Lambda icon. Below it is the text "DataFlow Functions for AWS Lambda". At the bottom are "Download" and "Documentation" buttons.
- Azure Functions**: Represented by a blue lightning bolt icon. Below it is the text "DataFlow Functions for Azure Functions". At the bottom are "Download" and "Documentation" buttons.
- Google Cloud Functions**: Represented by a blue hexagonal icon. Below it is the text "DataFlow Functions for Google Cloud Functions". At the bottom are "Download" and "Documentation" buttons.

Below these cards is a note: "💡 Download the binary for executing flows in these Cloud Providers" followed by a "Learn More" link.

This screenshot shows a detailed view of the AWS Lambda function setup within the Cloudera DataFlow interface. The left sidebar includes links for Dashboard, Catalog, ReadyFlow Gallery, Flow Design, Functions (which is highlighted in purple), and Environments.

The main content area is titled "Functions" and displays the "AWS Lambda" card from the previous screenshot. It includes the Lambda icon, the text "DataFlow Functions for AWS Lambda", a "Download" button, and a "Documentation" button.

At the bottom of the card, there is a note: "💡 Download the binary for executing flows in these Cloud Providers" followed by a "Learn More" link.

This should download a binary with a name similar to:
naaf-aws-lambda-1.0.0.2.3.7.0-100-bin.zip

Once you have the binary, make sure you also have:

- The CRN of the flow you published in the DataFlow Catalog
- The Access Key that has been provided with these instructions in “Competition Resources” section

- The Private Key that has been provided with these instructions in “Competition Resources” section

In order to speed up the deployment, we’re going to leverage some scripts to automate the deployment. It assumes that your AWS CLI is properly configured locally on your laptop and you can use the `jq` command for reading JSON payloads. You can now follow [the instructions from this page here](#).

However, if you wish to deploy the flow in AWS Lambda manually through the AWS UI, you can follow [the steps described here](#).