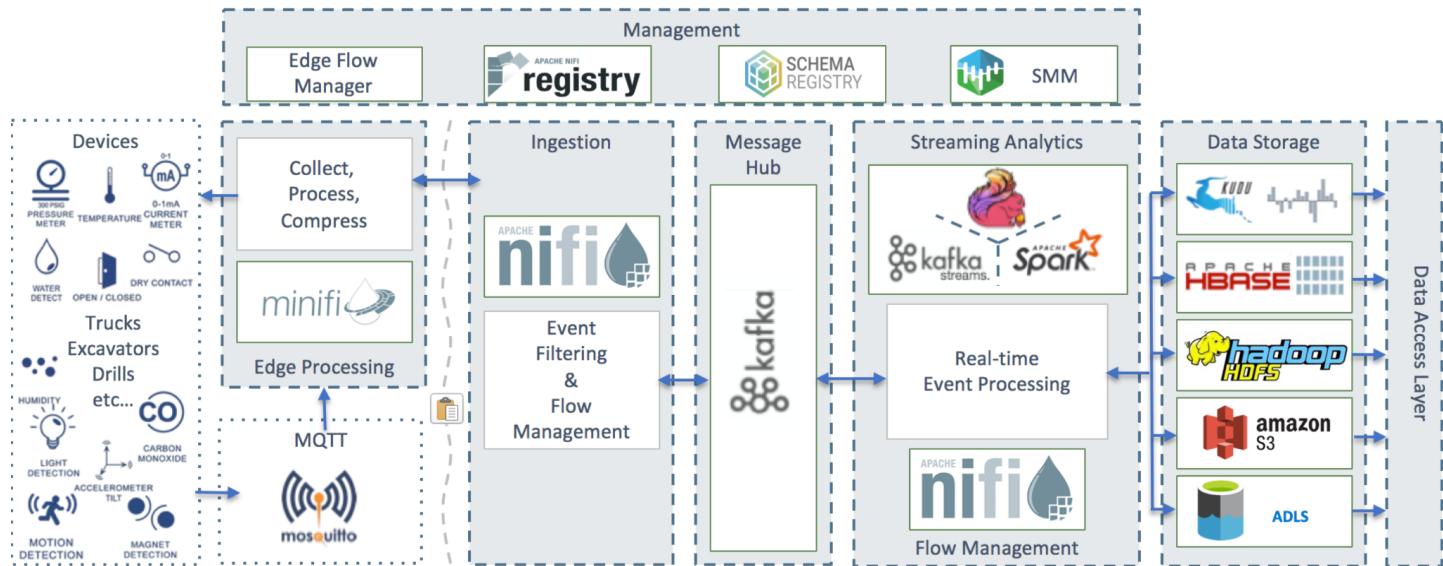


# CDF Workshop Student Guide

<b>Introduction</b>	<b>3</b>
Labs summary	4
<b>Lab 0 - Initial setup</b>	<b>5</b>
Pre-requisites	5
Cloudera Manager and other services	5
<b>Lab 1 - Apache NiFi: setup machine sensors simulator</b>	<b>7</b>
<b>Lab 2 - Configuring Edge Flow Management</b>	<b>14</b>
<b>Lab 3 - Configuring the NiFi flow and pushing data to Kafka</b>	<b>30</b>
<b>Lab 4 - Streams Messaging Manager (SMM) to check and monitor Kafka</b>	<b>36</b>
<b>Lab 5 - From Kafka to Kudu</b>	<b>39</b>
Create Schema Registry	45
<b>Lab 6 - Fast analytics on fast data with Kudu and Impala</b>	<b>52</b>
This concludes Cloudera Data Flow Workshop	53
Thanks for your participation..	53

# Introduction

In this hands-on workshop, you will build a full OT to IT workflow for an IoT Predictive Maintenance use case. Below is the architecture diagram, showing all the components you will set up over the lab exercises. While the diagram divides the components according to their location (factory, regional or datacenter level) in this workshop all such components will reside in one single host.



## Labs summary

S.No	Description
<b>Lab 1</b>	On the Apache NiFi, run a simulator to send IoT sensor data to the MQTT broker.
<b>Lab 2</b>	Create the MiNiFi flow on the Edge Flow Manager and publish it for the MiNiFi agent to start sending data to the NiFi cluster.
<b>LAB 3</b>	Configuring the NiFi flow and pushing data to Kafka
<b>LAB 4</b>	Streams Messaging Manager (SMM) to check and monitor Kafka
<b>LAB 5</b>	From Kafka to Kudu
<b>LAB 6</b>	Fast analytics on fast data with Kudu and Impala

## Lab 0 - Initial setup

### Pre-requisites

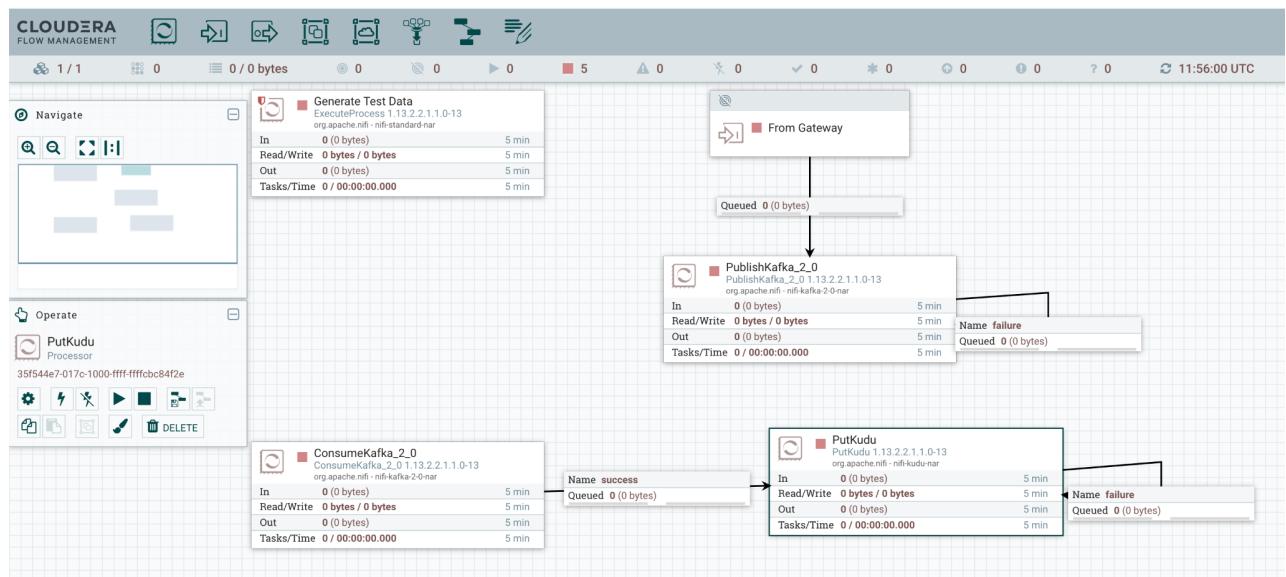
1. Laptop with a supported OS (Windows 7 not supported).
2. A modern browser like Google Chrome (IE not supported).

### Cloudera Manager and other services

Login into Cloudera Manager with username/password `admin/supersecret1`, and familiarize yourself with the services installed. The URLs to access the services are listed below

Service	URL
Cloudera Manager	<a href="http://&lt;your vm-public-IP&gt;:7180">http://&lt;your vm-public-IP&gt;:7180</a>
Edge Flow Manager	<a href="http://&lt;your vm-public-IP&gt;:10088/efm/ui">http://&lt;your vm-public-IP&gt;:10088/efm/ui</a>
NiFi	<a href="http://&lt;your vm-public-IP&gt;:8080/nifi/">http://&lt;your vm-public-IP&gt;:8080/nifi/</a>
NiFi Registry	<a href="http://&lt;your vm-public-IP&gt;:18080/nifi-registry">http://&lt;your vm-public-IP&gt;:18080/nifi-registry</a>
Hue	<a href="http://&lt;your vm-public-IP&gt;:8888">http://&lt;your vm-public-IP&gt;:8888</a>
SMM	<a href="http://&lt;your vm-public-IP&gt;:9991/#/">http://&lt;your vm-public-IP&gt;:9991/#/</a>

Below is a screenshot of flow which we are going to define in this workshop.



Let's Begin...

## Lab 1 - Apache NiFi: setup machine sensors simulator

In this lab you will run a simple Python script that simulates IoT sensor data from some hypothetical machines, and send the data to a MQTT broker ([mosquitto](#)). The MQTT broker plays the role of a gateway that is connected to many and different types of sensors through the "mqtt" protocol. Your cluster comes with an embedded MQTT broker that the simulation script publishes to. For convenience, we will use NiFi to run the script rather than Shell commands.

The VM provided in this lab is already having mosquitto installed and configured.

Below are the outputs from one of such VM's.

```
$ systemctl status mosquitto
```

```
[root@cdp ~]# systemctl status mosquitto
● mosquitto.service - Mosquitto MQTT v3.1/v3.1.1 Broker
  Loaded: loaded (/usr/lib/systemd/system/mosquitto.service; enabled; vendor preset: disabled)
  Active: active (running) since Wed 2021-09-29 12:18:33 UTC; 6min ago
    Docs: man:mosquitto.conf(5)
          man:mosquitto(8)
   Main PID: 5317 (mosquitto)
     CGroup: /system.slice/mosquitto.service
             └─5317 /usr/sbin/mosquitto -c /etc/mosquitto/mosquitto.conf
```

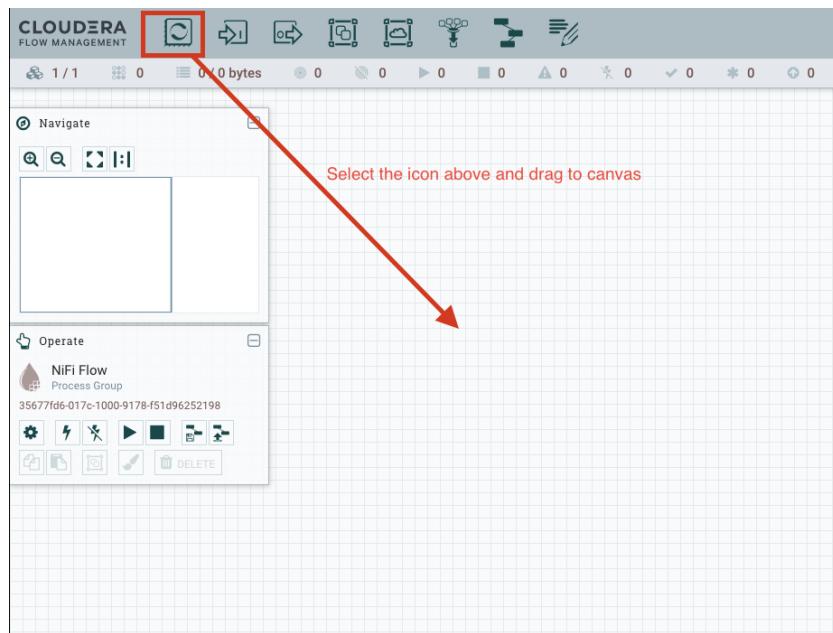
Below is the output of the sample simulator python programme.

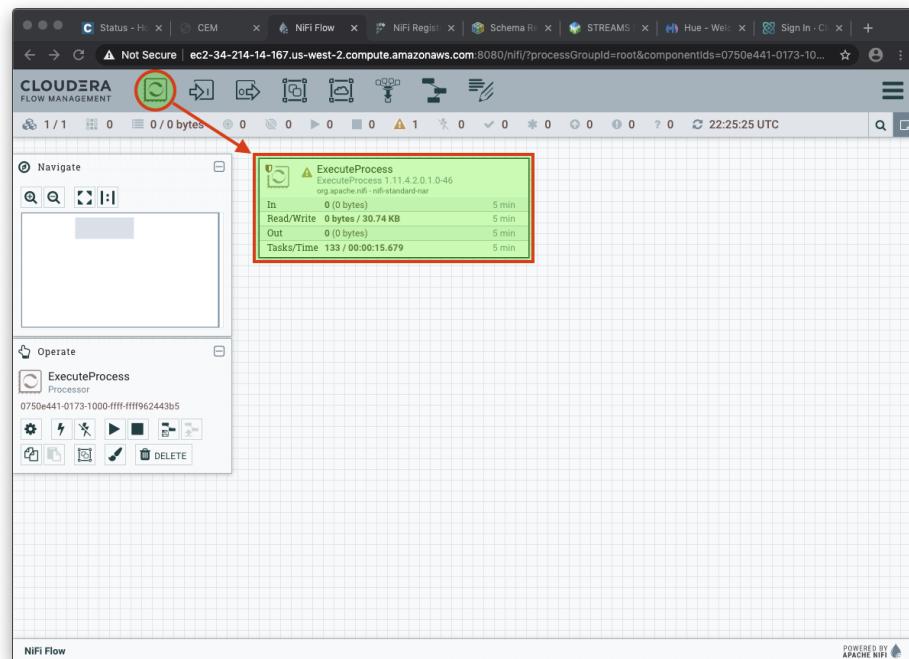
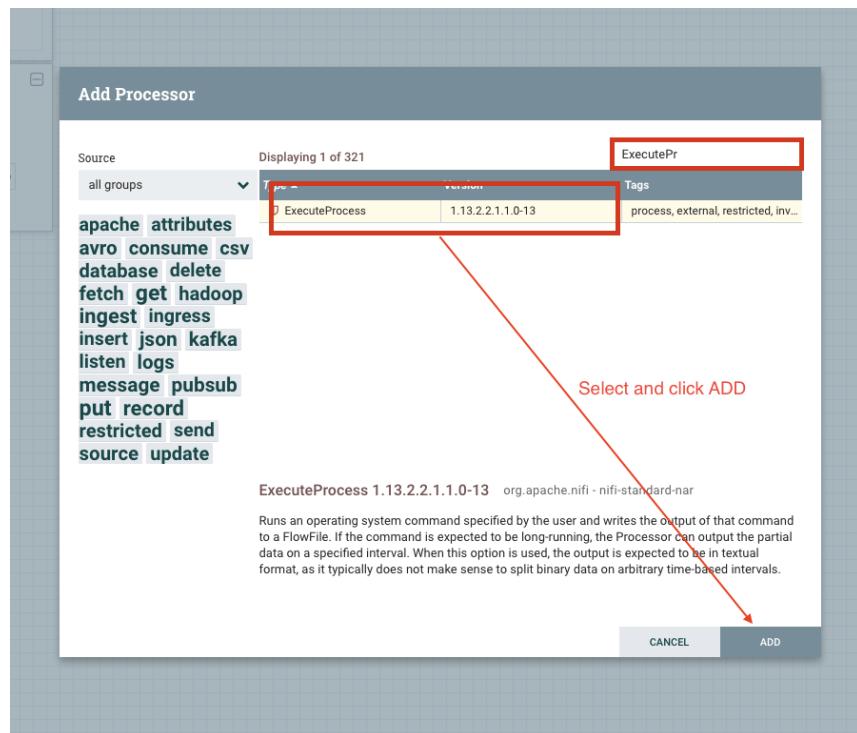
```
$ cd /opt/demo
$ python3 simulate.py
```

```
[centos@cdp demo]$ cd /opt/demo
[centos@cdp demo]$ python3 simulate.py
[{"sensor_id": 98, "sensor_ts": 1632918949154217, "sensor_0": 2, "sensor_1": 5, "sensor_2": 2, "sensor_3": 40, "sensor_4": 47, "sensor_5": 62, "sensor_6": 25, "sensor_7": 109, "sensor_8": 36, "sensor_9": 7, "sensor_10": 5, "sensor_11": 10}
```

## Step 1: Setup the processors on NiFi

- a. Go to Apache NiFi (<http://<public-ip>:8080/nifi>) and add a Processor (ExecuteProcess) to the canvas.





## Step 2: Configure the processor

- a. Right-click the processor, select Configure (or, alternatively, just double-click the processor). On the PROPERTIES tab, set the properties shown below to run our Python simulate script.

Property	Value
Command	python3
Command Argument	/opt/demo/simulate.py

The screenshot shows the 'PROPERTIES' tab of a processor configuration. A yellow box highlights the entire table. The table has two columns: 'Property' and 'Value'. The 'Property' column contains: Command, Command Arguments, Batch Duration, and Redirect Error Stream. The 'Value' column contains: python3, /opt/demo/simulate.py, No value set, and false respectively. The 'python3' and '/opt/demo/simulate.py' entries are highlighted with a red box.

Property	Value
Command	python3
Command Arguments	/opt/demo/simulate.py
Batch Duration	No value set
Redirect Error Stream	false

- b. In the **SCHEDULING** tab, set to Run Schedule: 1 sec

Alternatively, you could set that to other time intervals: 1 sec, 30 sec, 1 min, etc...

The screenshot shows the 'SCHEDULING' tab of a processor configuration. A yellow box highlights the entire tab. It includes fields for Scheduling Strategy (Timer driven), Run Duration (00:00:00.000), Concurrent Tasks (1), Run Schedule (1 sec), Execution (All nodes), and a note about the Run Duration being determined by the Run Schedule.

Scheduling Strategy	Run Duration
Timer driven	00:00:00.000
Concurrent Tasks	Run Schedule
1	1 sec
Execution	All nodes

- c. In the SETTINGS tab:

- Check the "success" relationship in the AUTOMATICALLY TERMINATED RELATIONSHIPS section.
- Set the processor name to "Generate Test Data"
- Click Apply

## Configure Processor

SETTINGS    SCHEDULING    PROPERTIES    COMMENTS

Name: **Generate Test Data**  Enabled  success

Automatically Terminate Relationships [?](#)  
All created FlowFiles are routed to this relationship

Id: 7ff13bdc-123e-1d34-a73d-b40563ab753f

Type: ExecuteProcess 1.9.0.1.0.1.0-12

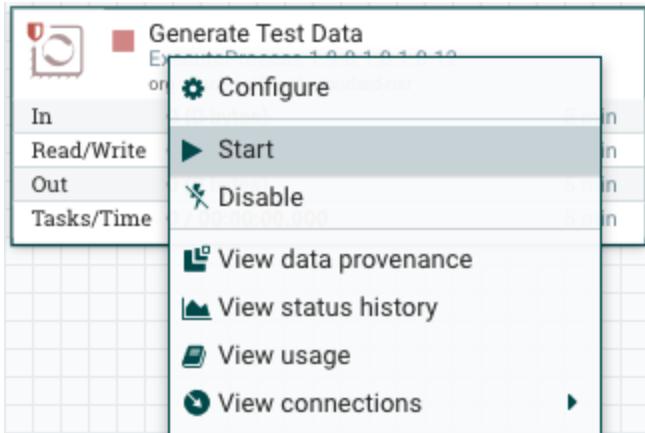
Bundle: org.apache.nifi - nifi-standard-nar

Penalty Duration [?](#): 30 sec

Yield Duration [?](#): 1 sec

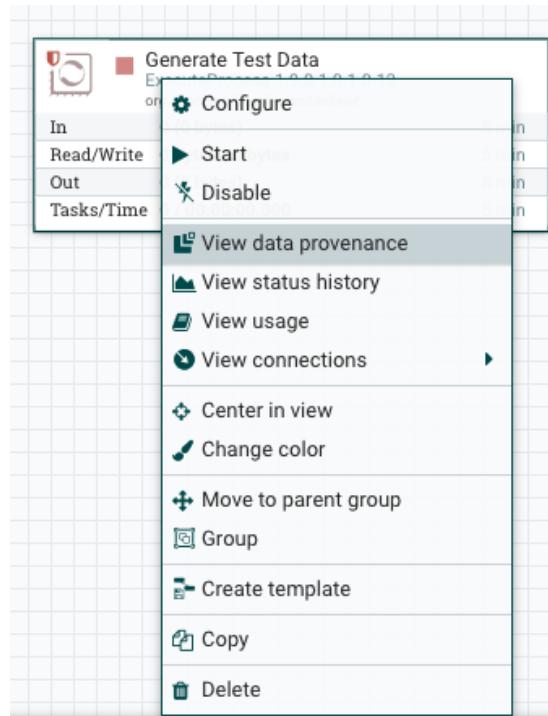
Bulletin Level [?](#): WARN

- d. You can then right-click to Start this simulator runner.



### Step 3: Validate the processor

- a. Right-click and select Stop after a few seconds and look at the provenance. You'll see that it has run a number of times and produced results.



## NiFi Data Provenance

Displaying 1,000 of 1,000

Oldest event available: 10/29/2019 02:44:35 UTC

Showing 1,000 of 1,000+ events that match the specified query, please refine the search. [Clear search](#)

Filter		by component name ▾					
Date/Time ▾	Type	FlowFileUuid	Size	Component Name	Component Type	Node	Actions
10/29/2019 14:15:01.707...	DROP	f8cb0afc-ef58-4128-af1d...	238 bytes	Generate Test Data	ExecuteProcess	ip-10-0-1-192.us-west-2.c...	
10/29/2019 14:15:01.707...	CREATE	f8cb0afc-ef58-4128-af1d...	238 bytes	Generate Test Data	ExecuteProcess	ip-10-0-1-192.us-west-2.c...	
10/29/2019 14:15:00.648...	DROP	a8593d6e-7014-4276-8b...	239 bytes	Generate Test Data	ExecuteProcess	ip-10-0-1-192.us-west-2.c...	
10/29/2019 14:15:00.648...	CREATE	a8593d6e-7014-4276-8b...	239 bytes	Generate Test Data	ExecuteProcess	ip-10-0-1-192.us-west-2.c...	
10/29/2019 14:14:59.588...	DROP	7bae8a48-24e9-4706-93...	239 bytes	Generate Test Data	ExecuteProcess	ip-10-0-1-192.us-west-2.c...	
10/29/2019 14:14:59.588...	CREATE	7bae8a48-24e9-4706-93...	239 bytes	Generate Test Data	ExecuteProcess	ip-10-0-1-192.us-west-2.c...	
10/29/2019 14:14:58.529...	DROP	b727f9d0-b4c7-40c1-b8d...	238 bytes	Generate Test Data	ExecuteProcess	ip-10-0-1-192.us-west-2.c...	
10/29/2019 14:14:58.529...	CREATE	b727f9d0-b4c7-40c1-b8d...	238 bytes	Generate Test Data	ExecuteProcess	ip-10-0-1-192.us-west-2.c...	
10/29/2019 14:14:57.471...	DROP	013119a1-2f59-42ea-a88...	241 bytes	Generate Test Data	ExecuteProcess	ip-10-0-1-192.us-west-2.c...	
10/29/2019 14:14:57.471...	CREATE	013119a1-2f59-42ea-a88...	241 bytes	Generate Test Data	ExecuteProcess	ip-10-0-1-192.us-west-2.c...	
10/29/2019 14:14:56.413...	DROP	c1ce6530-8eb1-4a9e-816...	238 bytes	Generate Test Data	ExecuteProcess	ip-10-0-1-192.us-west-2.c...	
10/29/2019 14:14:56.413...	CREATE	c1ce6530-8eb1-4a9e-816...	238 bytes	Generate Test Data	ExecuteProcess	ip-10-0-1-192.us-west-2.c...	
10/29/2019 14:14:55.357...	DROP	09d42dde-272f-4b0c-807...	237 bytes	Generate Test Data	ExecuteProcess	ip-10-0-1-192.us-west-2.c...	
10/29/2019 14:14:55.357...	CREATE	09d42dde-272f-4b0c-807...	237 bytes	Generate Test Data	ExecuteProcess	ip-10-0-1-192.us-west-2.c...	
10/29/2019 14:14:54.291...	DROP	6edbadee-2182-487a-96...	238 bytes	Generate Test Data	ExecuteProcess	ip-10-0-1-192.us-west-2.c...	

# Lab 1 Completed

## Lab 2 - Configuring Edge Flow Management

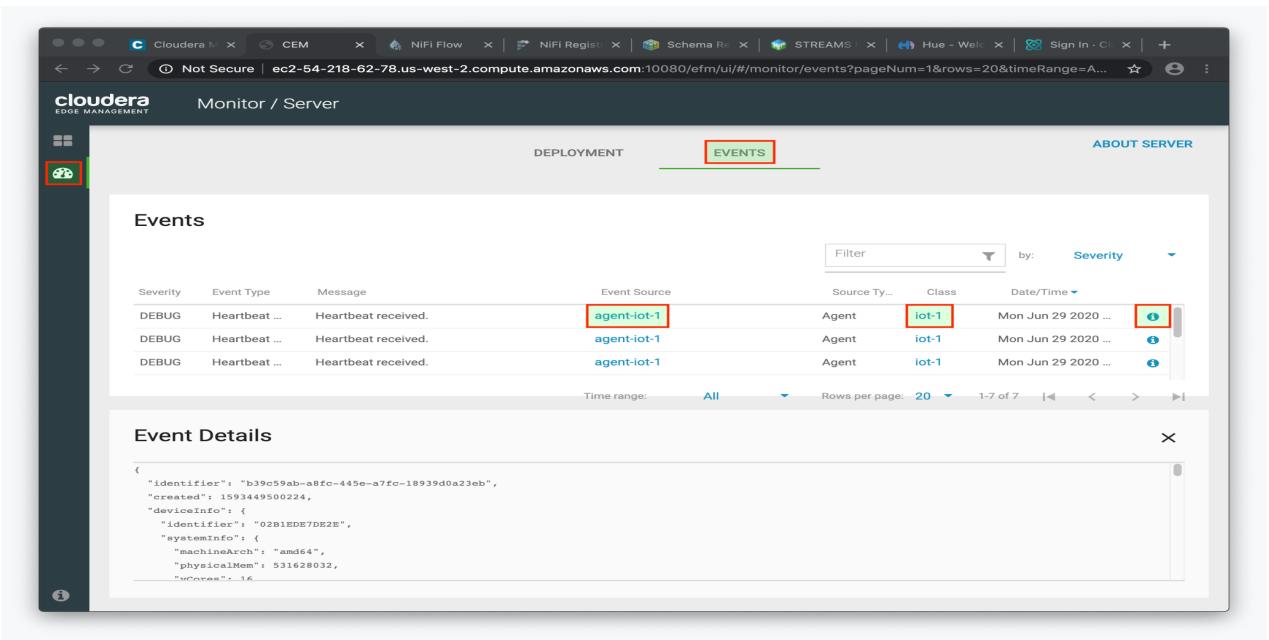
Cloudera Edge Flow Management (EFM) gives you a visual overview of all MiNiFi agents in your environment, and allows you to update the flow configuration for each one, with versioning control thanks to the NiFi Registry integration. In this lab, you will create the MiNiFi flow and publish it for the MiNiFi agent to pick it up.

### Step1: Verify heartbeats from MiNiFi agents

- Open the EFM Web UI at [http://<public\\_dns>:10088/efm/ui/](http://<public_dns>:10088/efm/ui/) and select the Dashboard tab



Click on the EVENTS header and verify that your EFM server is receiving heartbeats from the MiNiFi agent. Click on the info icon on a heartbeat record to see the details of the heartbeat.



Severity	Event Type	Message	Event Source	Source Ty...	Class	Date/Time
DEBUG	Heartbeat ...	Heartbeat received.	agent-iot-1	Agent	iot-1	Mon Jun 29 2020 ...
DEBUG	Heartbeat ...	Heartbeat received.	agent-iot-1	Agent	iot-1	Mon Jun 29 2020 ...
DEBUG	Heartbeat ...	Heartbeat received.	agent-iot-1	Agent	iot-1	Mon Jun 29 2020 ...

**Event Details**

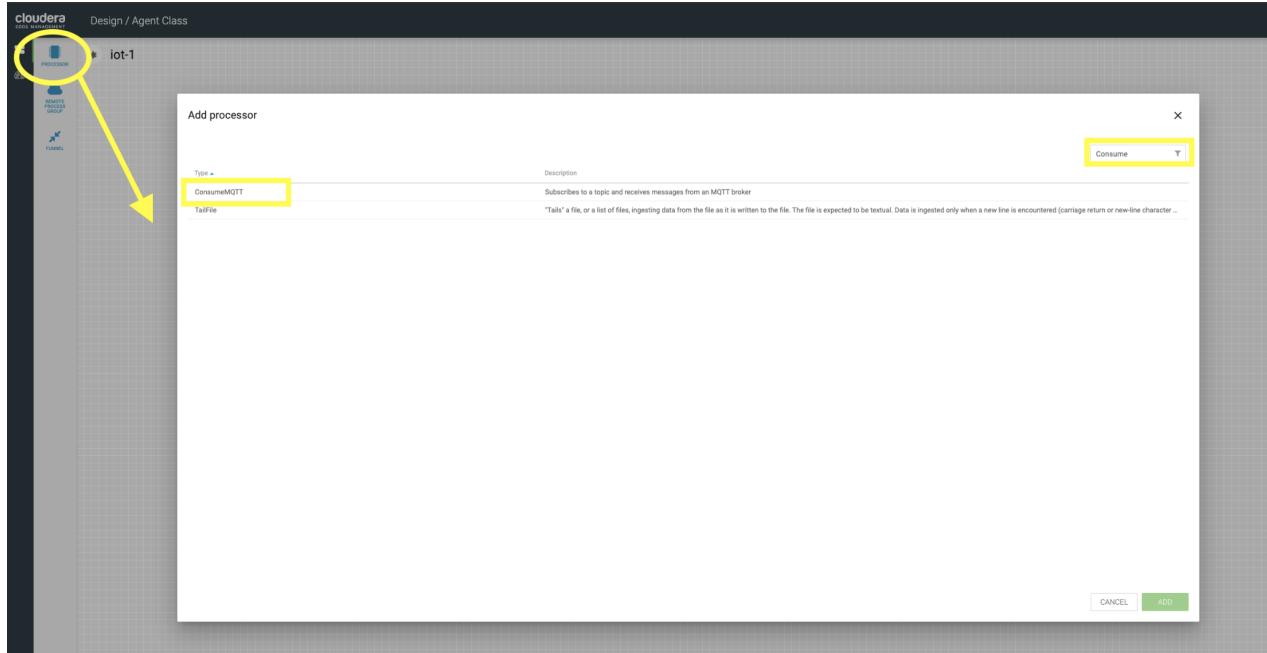
```
{ "identifier": "b39c59ab-a8fc-445e-a7fc-18939d0a23eb", "created": 1593449500224, "deviceInfo": { "identifier": "02B1EDE7DE2E", "systemInfo": { "machineArch": "amd64", "physicalMem": 531628032, "nCores": 16 }}
```

Severity	Event Type	Message	Event Source	Source Type	Class	Date/Time
DEBUG	Heartbeat Re...	Heartbeat received.	agent-iot-1	Agent	iot-1	Thu May 02 2019 10:2...
DEBUG	Heartbeat Re...	Heartbeat received.	agent-iot-1	Agent	iot-1	Thu May 02 2019 10:2...
DEBUG	Heartbeat Re...	Heartbeat received.	agent-iot-1	Agent	iot-1	Thu May 02 2019 10:2...
DEBUG	Heartbeat Re...	Heartbeat received.	agent-iot-1	Agent	iot-1	Thu May 02 2019 10:2...
DEBUG	Heartbeat Re...	Heartbeat received.	agent-iot-1	Agent	iot-1	Thu May 02 2019 10:2...
DEBUG	Heartbeat Re...	Heartbeat received.	agent-iot-1	Agent	iot-1	Thu May 02 2019 10:2...

## Step 2 : Designing the EFM flow

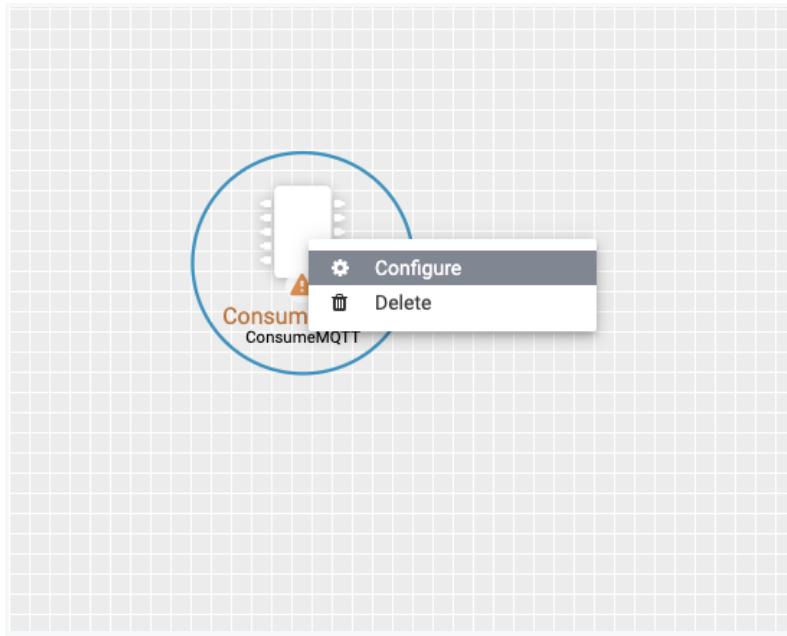
- Select the Flow Designer tab ( ). To build a dataflow, select the desired class ( `iot-1` ) from the table and click OPEN. Alternatively, you can double-click on the desired class.

- Add a *ConsumeMQTT* Processor to the canvas, by dragging the processor icon to the canvas, selecting the *ConsumeMQTT* processor type and clicking on the Add button. Once the processor is on the canvas, double-click it and configure it with below settings:



Select the processor and click ADD.

- c. Right Click and configure the *ConsumeMQTT* Processor with the below settings:



```

Broker URI: tcp://localhost:1883 OR tcp://<internal-hostname>:1883
Client ID: minifi-iot
Topic Filter: iot/# 
Max Queue Size = 60
  
```

Processor Name \*: ConsumeMQTT

Penalty Duration \*: 0 ms

Yield Duration \*: 0 ms

Automatically Terminated Relationships: Message

Scheduling Strategy: Timer Driven

Concurrent Tasks \*: 10

Run Schedule \*: 0 ms

Run Duration: 0ms, 25ms, 50ms, 100ms, 250ms, 500ms, 1s, 2s

Properties:

Property	Value
Broker URI	tcp://localhost:1883
Client ID	minifi-iot
Username	No value set
Password	No value set

**APPLY**

And ensure you scroll down on the properties page to set the Topic Filter and Max Queue Size:

Topic Filter	iot/#
Quality of Service(QoS)	0 - At most once
Max Queue Size	60

**APPLY**

Click on APPLY.

- d. Add a *Remote Process Group* to the canvas and configure it as follows and click on ADD.

URL = http://<public-IP>:8080/nifi

Add Remote Process Group

URL \*

CANCEL

ADD

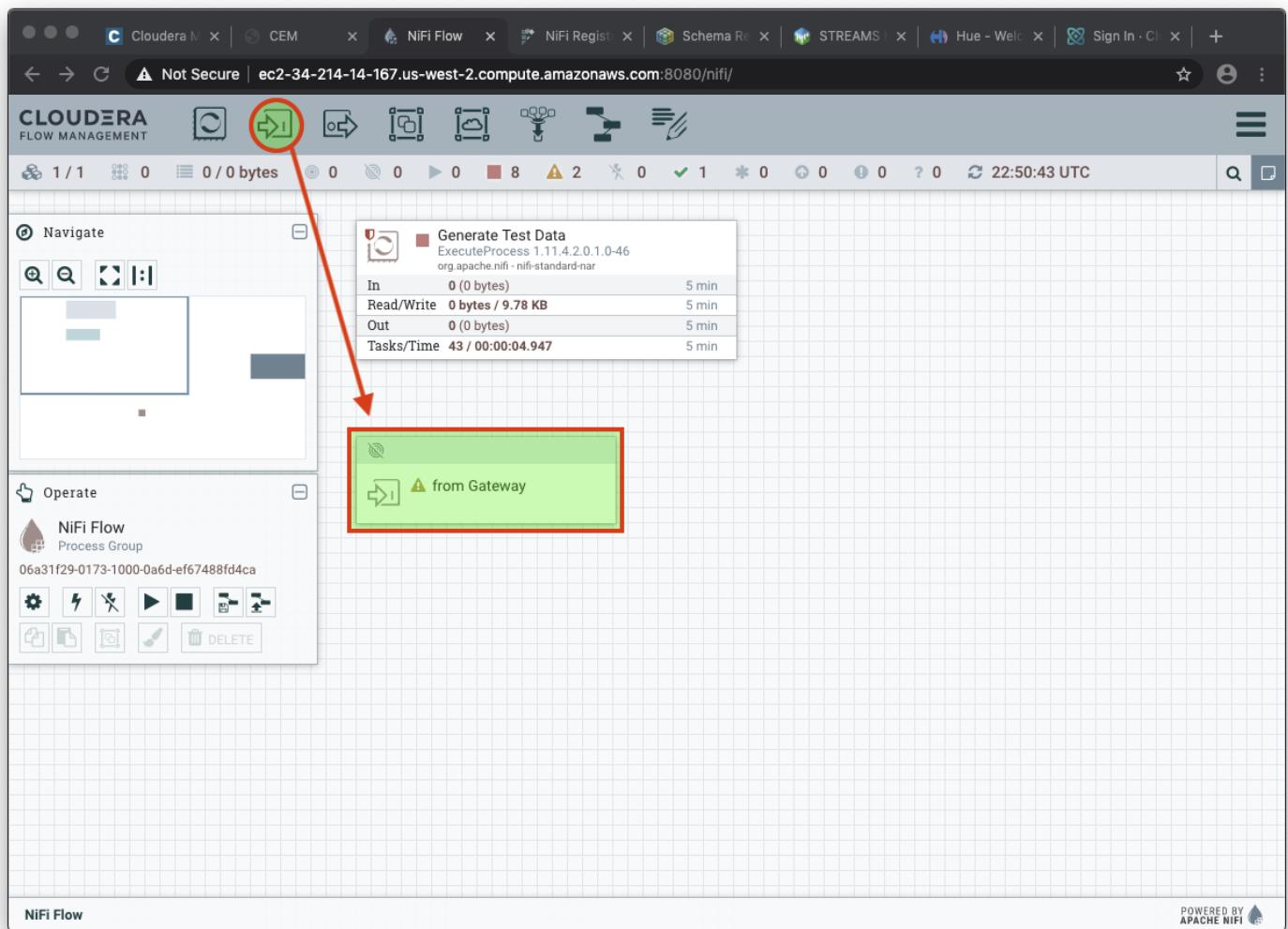
- e. Right click on the newly created Remote Process Group and click on CONFIGURE

The screenshot shows the Cloudera Edge Management interface. On the left, a flow diagram titled 'iot-1' is displayed. It contains a 'REMOTE PROCESS GROUP' node (highlighted with a red circle) and a 'ConsumeMQTT' node. A red arrow points from the 'REMOTE PROCESS GROUP' node to its configuration dialog on the right. The configuration dialog is titled 'Configuration' and has the URL 'http://edge2ai-1.dim.local:8080/nifi' entered. Other settings include 'HTTP' selected for 'TRANSPORT PROTOCOL', '30 secs' for 'COMMUNICATIONS TIMEOUT', and '10 sec' for 'YIELD DURATION'. The 'About' section displays the 'REMOTE PROCESS GROUP ID' as '260a79b2-dd7c-4df5-ba29-bda51d3a4944'. At the bottom right of the dialog is a green 'APPLY' button.

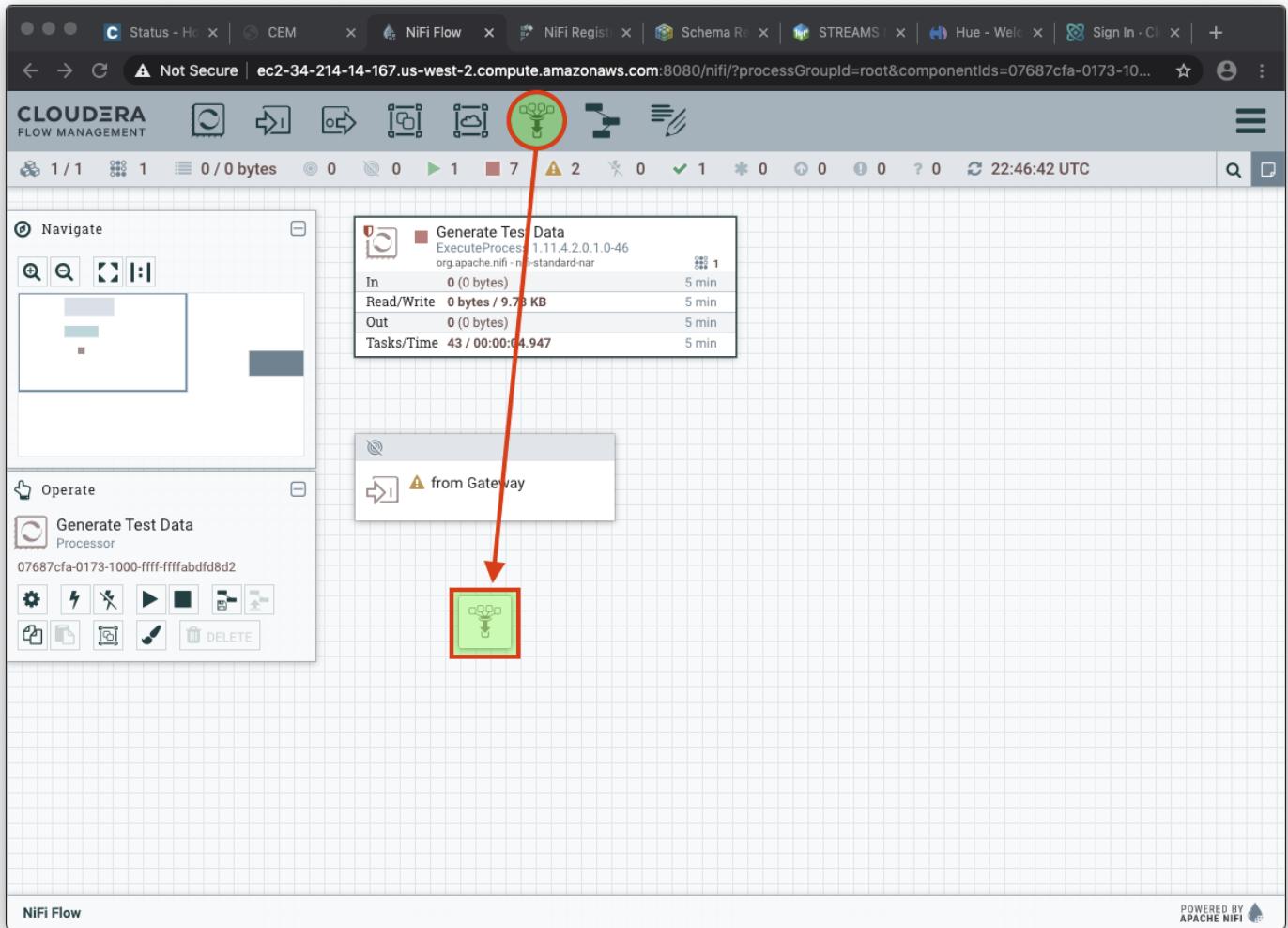
### Step 3 : Designing the NiFi flow

At this point you need to connect the ConsumerMQTT processor to the RPG, however, you first need the ID of the NiFi entry port.

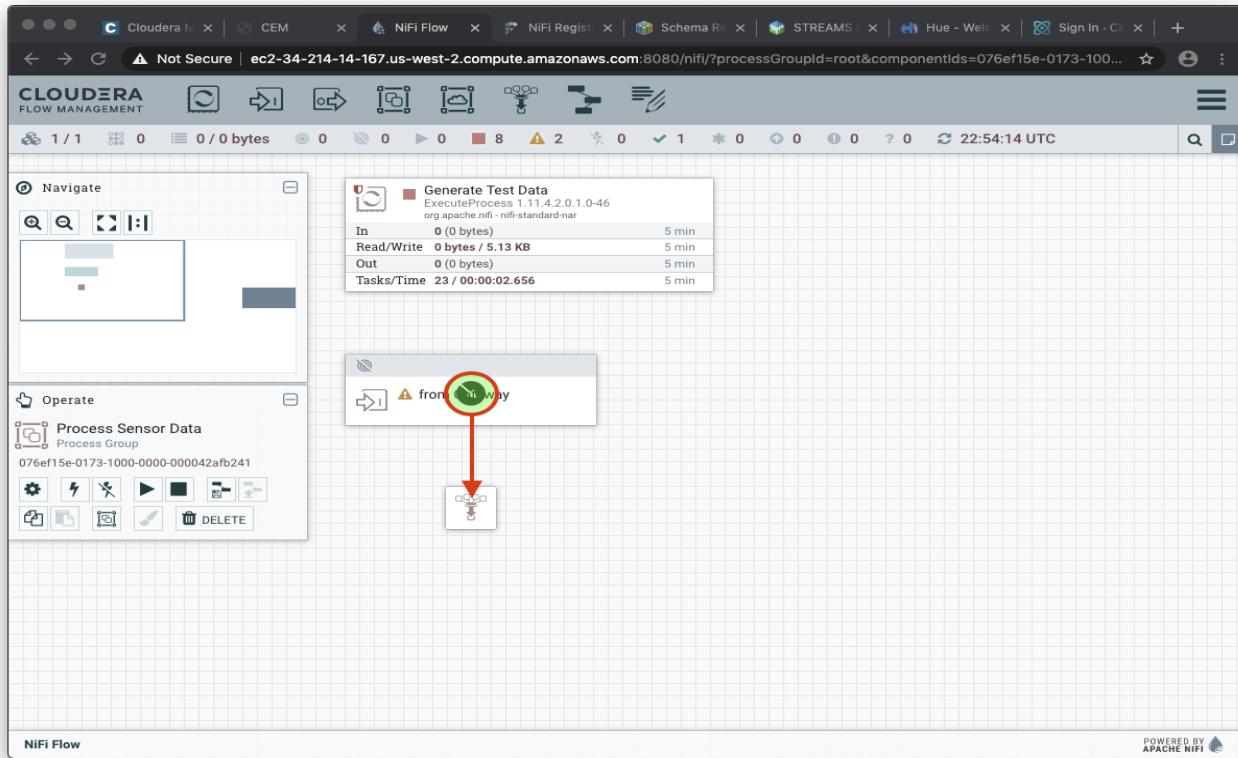
Open NiFi Web UI at <http://<public-IP>:8080/nifi/> and add an *Input Port* to the canvas. Call it something like "from Gateway" and copy the ID of the input port, as you will soon need it.



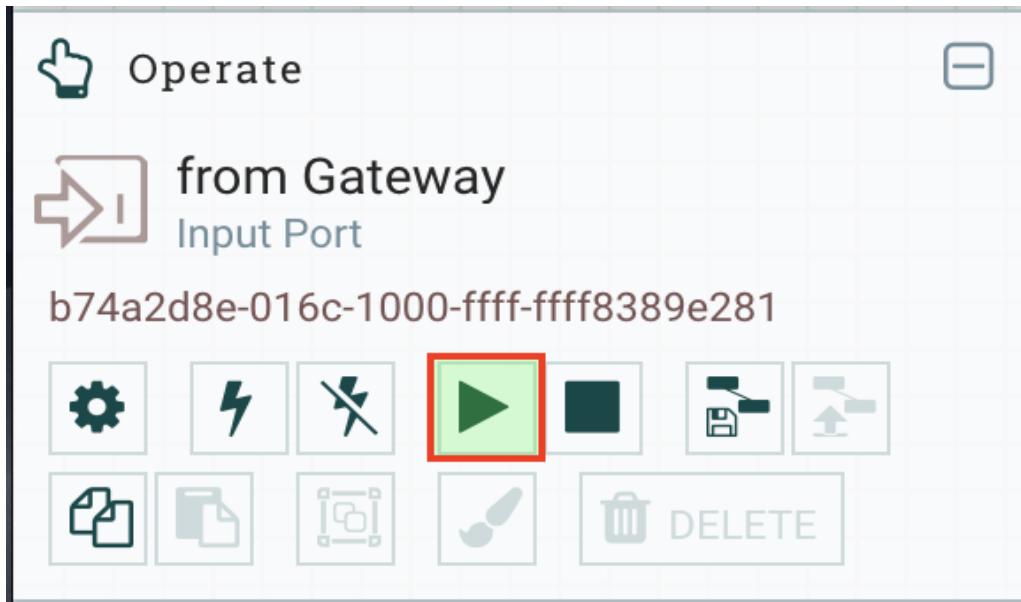
To terminate the NiFi *Input Port* let's, for now, add a *Funnel* to the canvas...



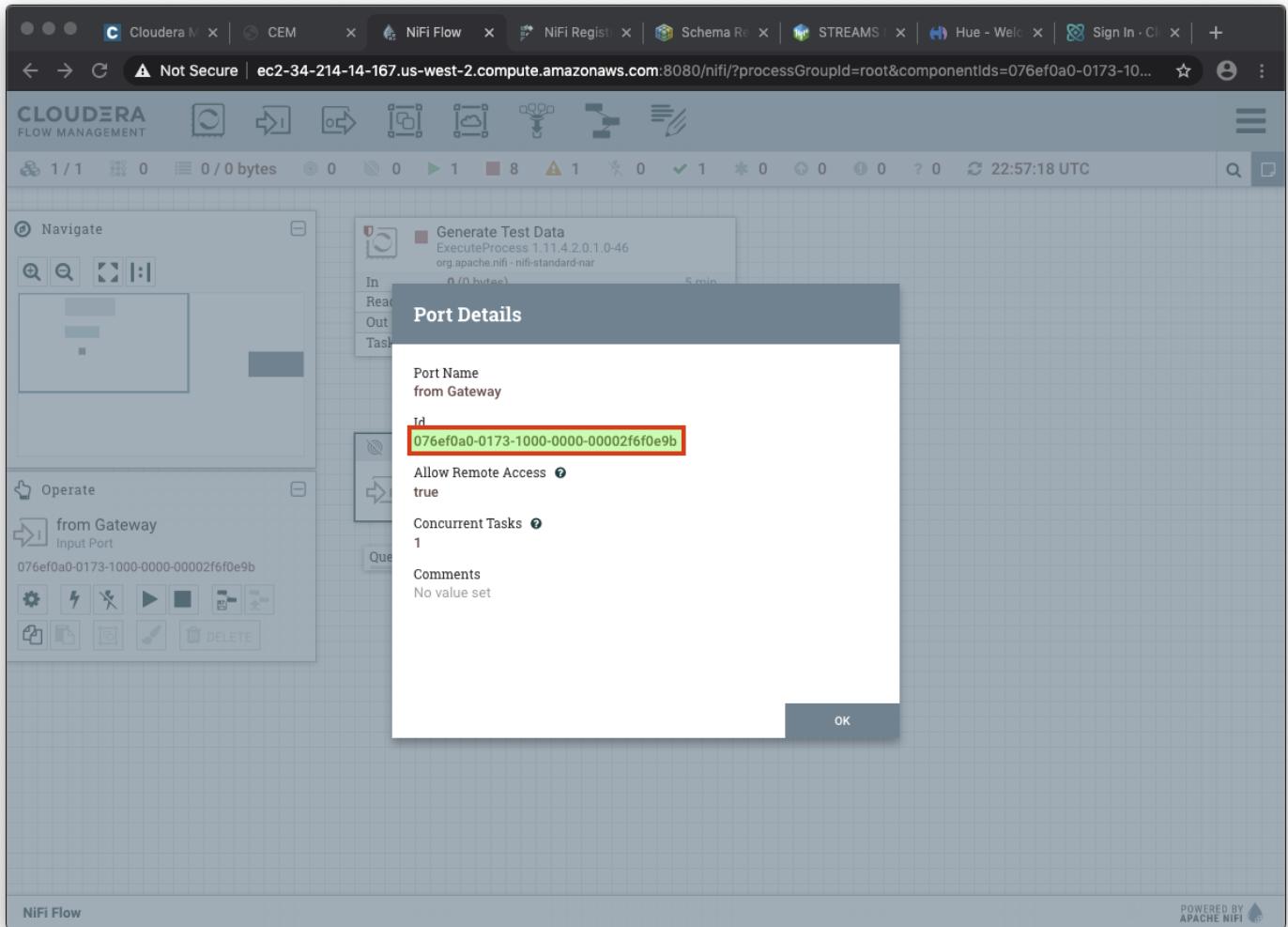
set up a connection from the Input Port to it. To set up a connection, hover the mouse over the Input Port until an arrow symbol is shown in the center. Click on the arrow, drag it and drop it on the Funnel to connect the two elements.



Right-click on the Input Port and start it. Alternatively, click on the Input Port to select it and then press the start ("play") button on the Operate panel:

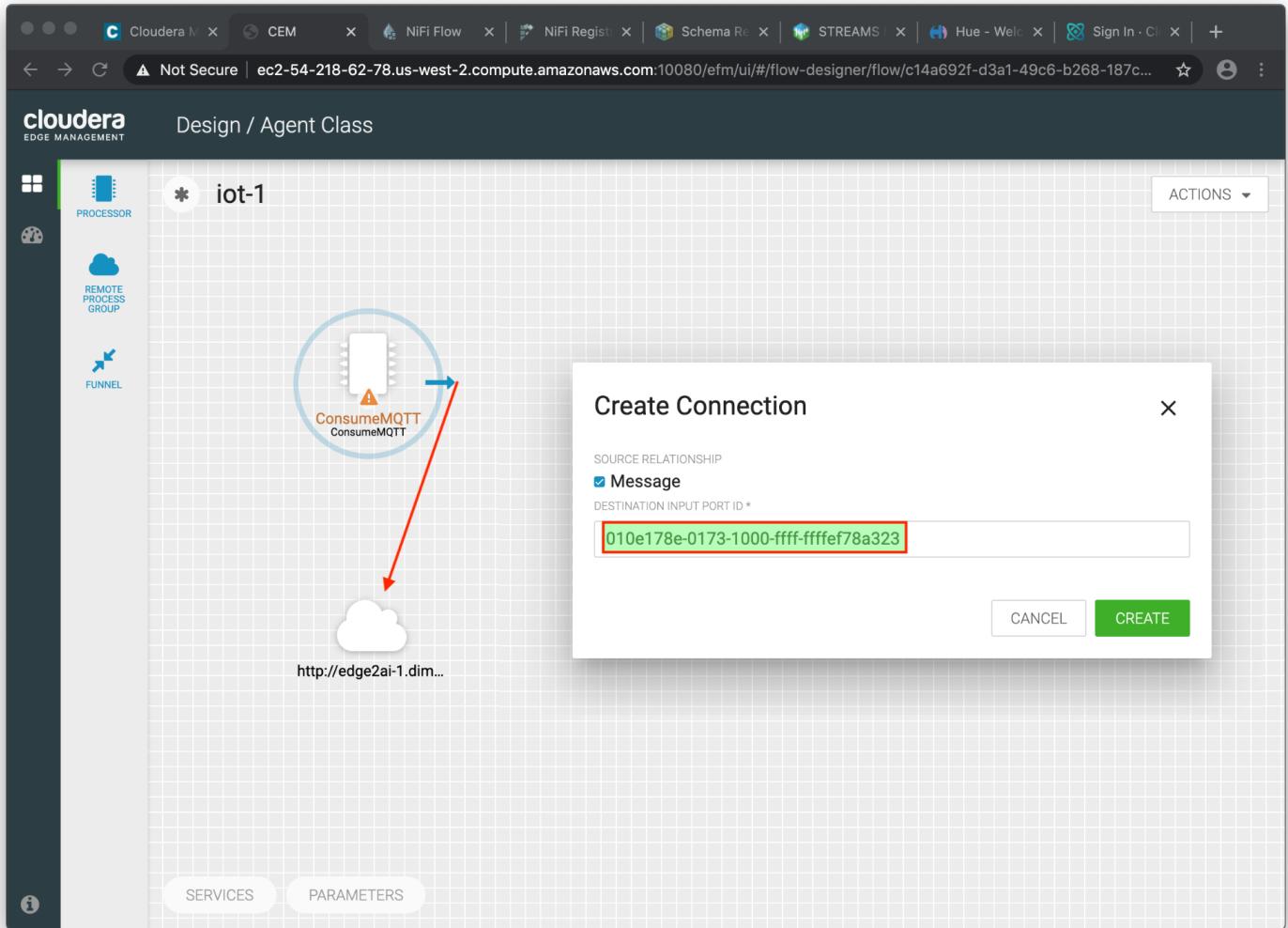


You will need the ID of the *Input Port* to complete the connection of the *ConsumeMQTT* processor to the RPG (NiFi). Double-click on the *Input Port* and copy its ID.



#### Step 4 : Integration between EFM and NiFi

Back to the Flow Designer, connect the *ConsumeMQTT* processor to the RPG. The connection requires an ID and you can paste here the ID you copied from the Input Port. Make sure that there are NO SPACES!



Double-click the connection and update the following configuration:

Property	Value
Flowfile Expiration	60 seconds
Back Pressure Object Threshold	10000
Connection Name	Sensor Data

Sensor data (Connection)X

## Configuration

 SOURCE  
ConsumeMQTT

 DESTINATION  
http://edge2ai-1.dim.local:8080/nifi

### Settings

SOURCE RELATIONSHIP  Message

DESTINATION INPUT PORT ID \*

FLOWFILE EXPIRATION \*

BACK PRESSURE OBJECT THRESHOLD \*

BACK PRESSURE SIZE THRESHOLD \*

CONNECTION NAME

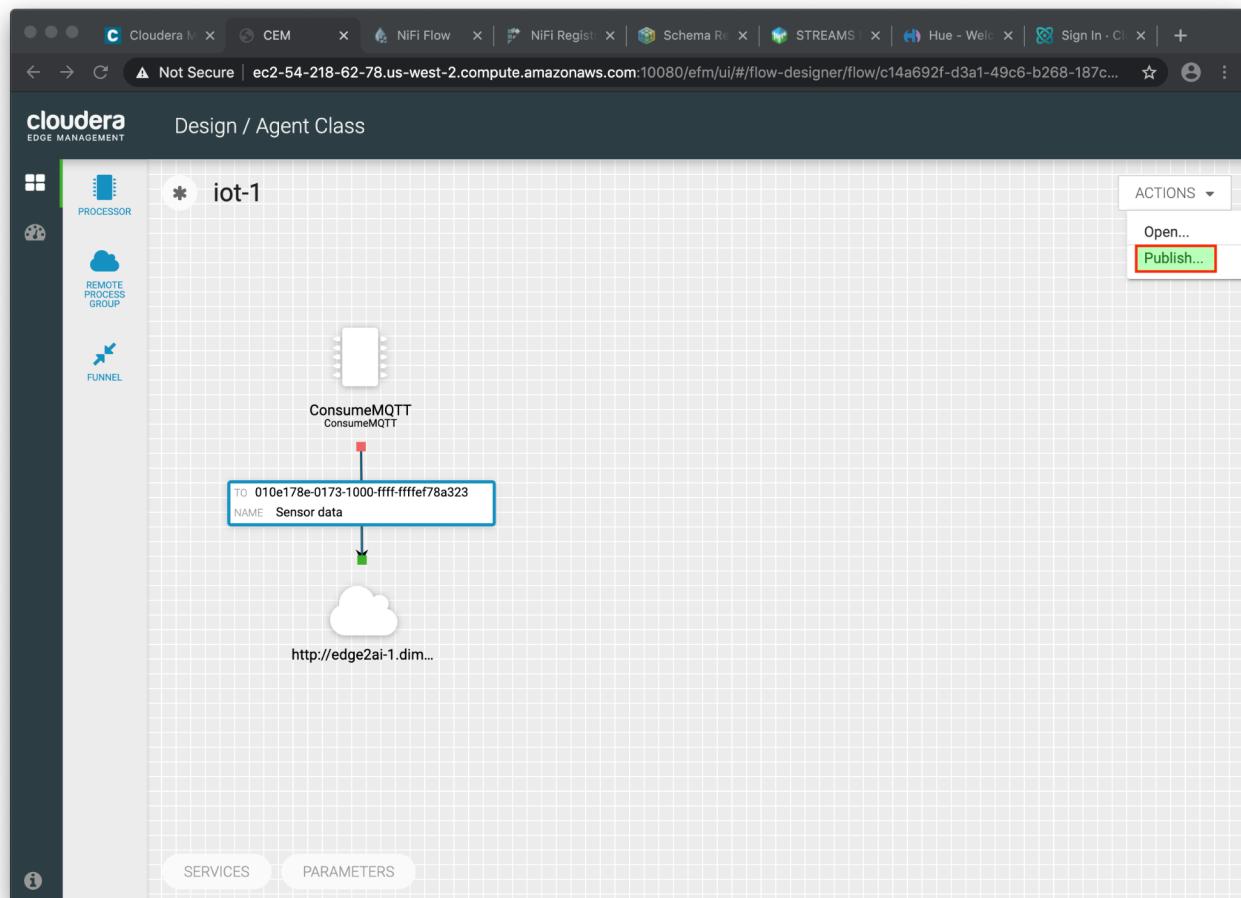
**APPLY**

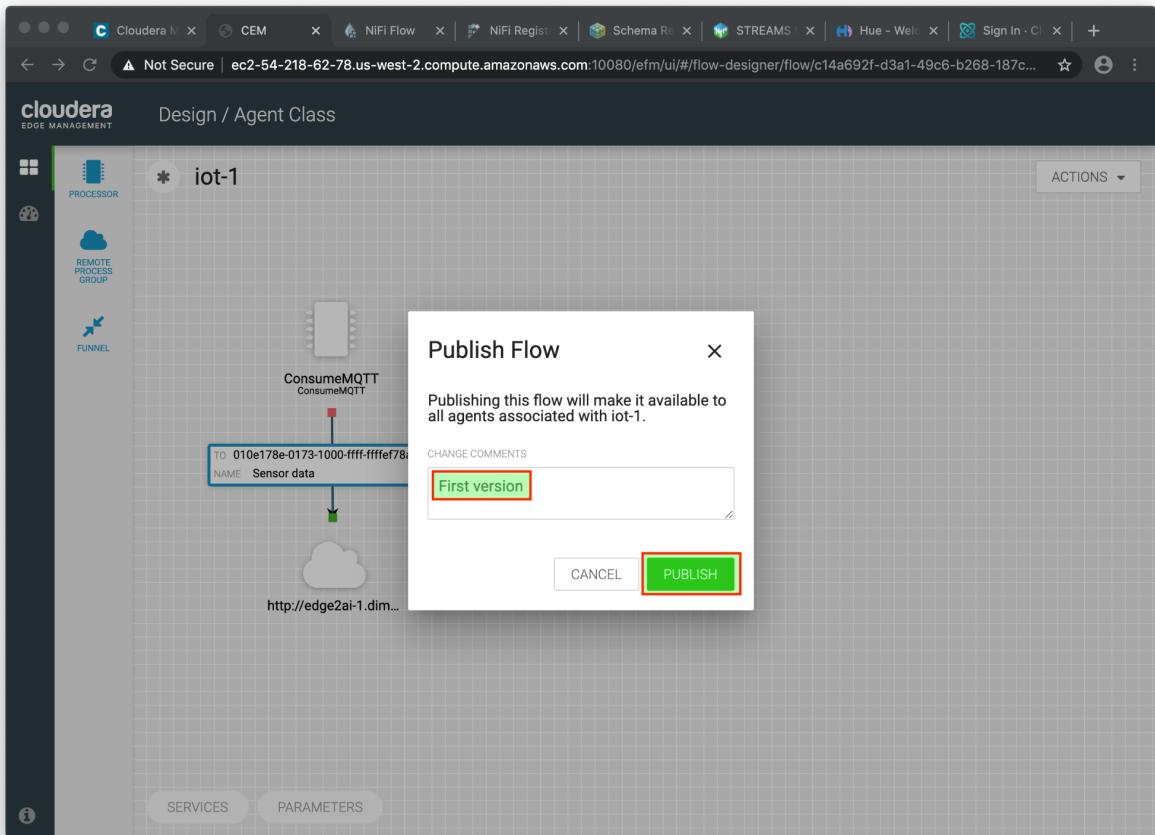
The Flow is now complete, but before publishing it, create the Bucket in the *NiFi Registry* so that all versions of your flows are stored for review and audit. Open the NiFi Registry at

[http://<public\\_dns>:18080/nifi-registry](http://<public_dns>:18080/nifi-registry), click on the wrench/spanner icon (🔧) on the top-right corner and create a bucket called IoT (ATTENTION: the bucket name is CASE-SENSITIVE).

The screenshot shows the NiFi Registry / Administration interface. At the top, there are several tabs: Cloudera Manager, Hue - Editor, NiFi Flow, CEM, NiFi Registry (which is the active tab), and IoT Prediction Model. Below the tabs, the URL is displayed as Not secure | 18.215.124.254:18080/nifi-registry/administration/workflow. The main content area is titled 'NiFi Registry / Administration' and shows 'BUCKETS' and 'USERS'. A sub-section titled 'Buckets (1)' is shown, with a single entry 'IoT'. To the right of the list is a 'NEW BUCKET' button, which is circled in red. A modal dialog box titled 'New Bucket' is open, prompting for a 'Bucket Name' (with 'IoT' entered) and an optional checkbox 'Keep this dialog open after creating bucket'. It also contains 'CANCEL' and 'CREATE' buttons.

You can now publish the flow for the MiNiFi agent to automatically pick up. Click Publish, add a descriptive comment for your changes and click Apply.





Go back to the NiFi Registry Web UI and click on the NiFi Registry name, next to the Cloudera logo. If the flow publishing was successful, you should see the flow's version details in the NiFi Registry.

The screenshot shows the Cloudera NiFi Registry interface. At the top, there is a navigation bar with various links like Status, CEM, NiFi Flow, NiFi Registry (which is highlighted with a red box), Schema Registry, STREAMS, Hue - Web, and Sign In. Below the navigation bar, the title is "NiFi Registry / All". On the right side, there is a user status "anonymous" and some icons. The main content area displays a single flow entry:

VERSIONS
1

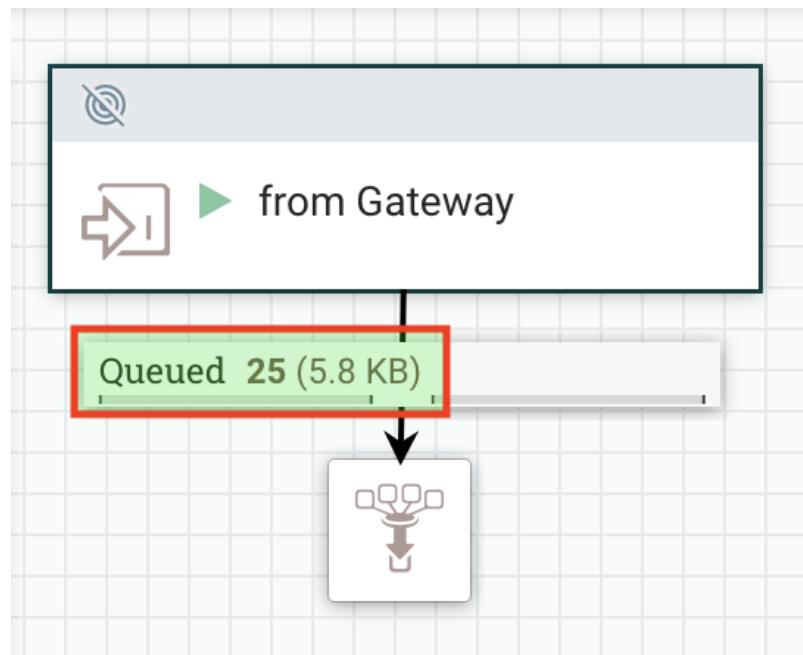
**BUCKET IDENTIFIER**: b00b69db-0583-40f7-bb57-f57d57cc4... **CHANGE LOG**: Version 1 - a few seconds ago by anonymous

**FLOW IDENTIFIER**: 35bf15d4-2ffd-4141-8958-6079522166... **First version**: Aug-15-2019 at 7:35 AM

**DESCRIPTION**: Created by MINIFI C2 Flow Designer

On the right side of the entry, there is a "ACTIONS" button.

At this point, you can test the edge flow up until NiFi. Start the NiFi simulator (ExecuteProcess processor) again and confirm you can see the messages queued in NiFi.



You can stop the simulator (Stop the NiFi processor) once you confirm that the flow is working correctly.

Before we move on to the next lab we will have to stop all the processors currently on the canvas and clear the connection queue which has flow files which we saw in the previous step.

To clear the queue, right click on the connection and click on “Clear Queue”

Once the queue is cleared, we can delete the connection from the INPUT PORT to the FUNNEL.

Once the connection is deleted we can now also delete the FUNNEL.

**LAB 2 completed**

## Lab 3 - Configuring the NiFi flow and pushing data to Kafka

In this lab, you will create a NiFi flow to receive the data from all gateways and push it to Kafka.

Step 1 : Setup the NiFi flow

Open the NiFi web UI and add a *PublishKafka\_2.0* processor and configure it as follows.



Make sure to use the 'hostname of the VM assigned to you.(Refer excel sheet shared by the instructor)

```
Kafka Brokers: <hostname>:9092
Topic Name: iot
Use Transactions: False
```

**Configure Processor**

■ Stopped

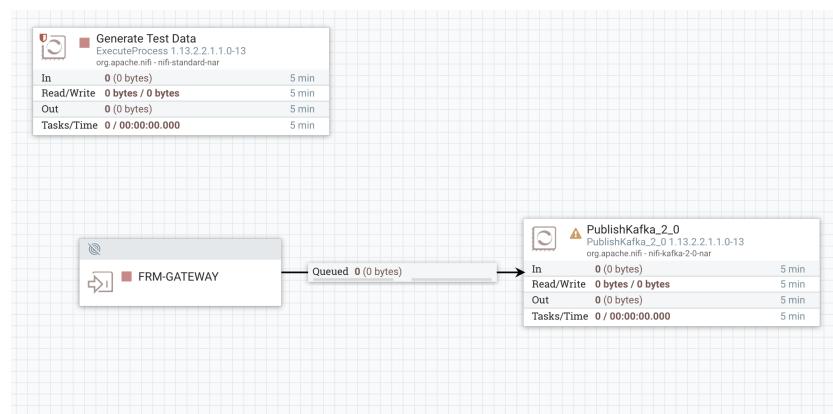
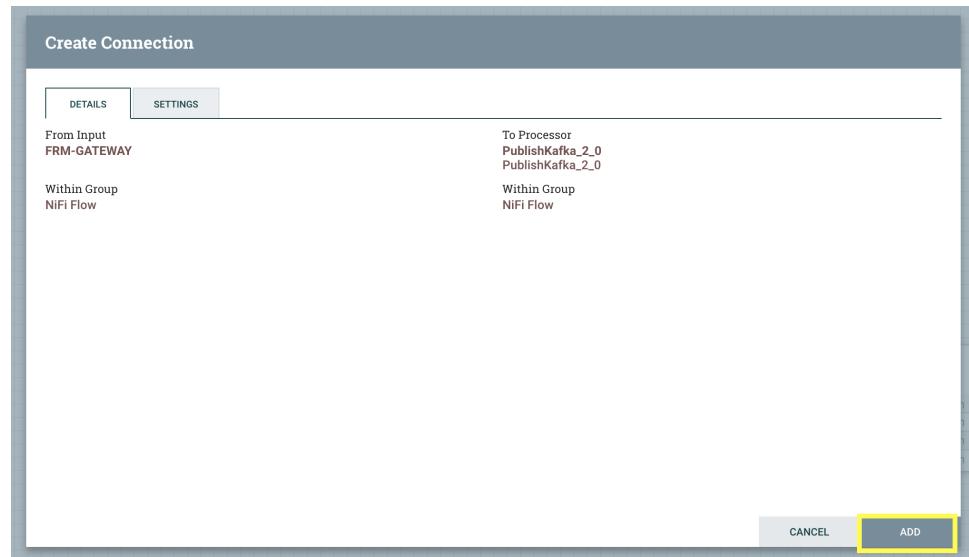
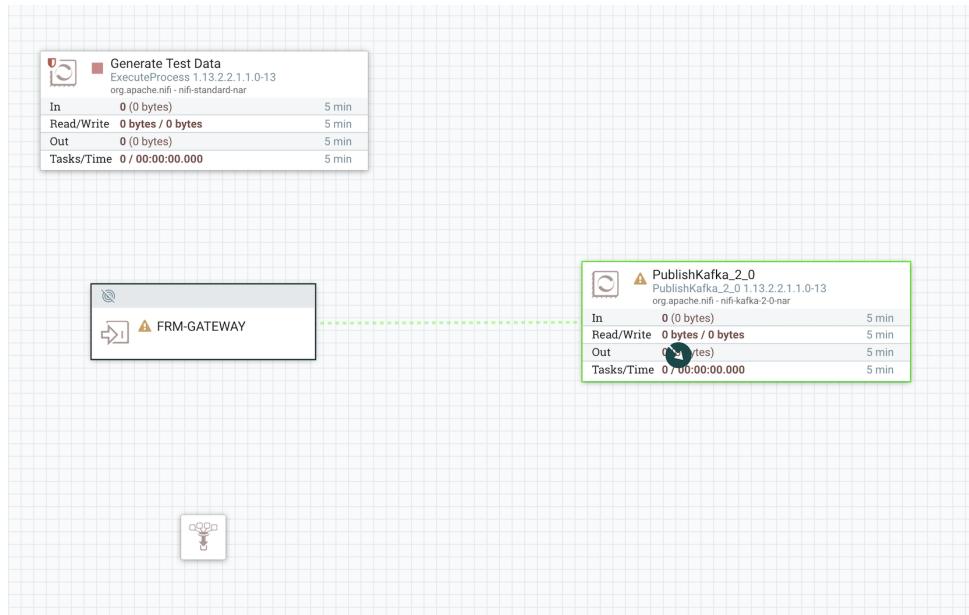
SETTINGS	SCHEDULING	PROPERTIES	COMMENTS																										
<b>Required field</b>																													
<table border="1"> <thead> <tr> <th>Property</th> <th>Value</th> </tr> </thead> <tbody> <tr> <td>Kafka Brokers</td> <td>cdp.3.108.43.134.nip.io:9092</td> </tr> <tr> <td>Security Protocol</td> <td>PLAINTEXT</td> </tr> <tr> <td>SASL Mechanism</td> <td>GSSAPI</td> </tr> <tr> <td>Kerberos Service Name</td> <td>No value set</td> </tr> <tr> <td>Kerberos Credentials Service</td> <td>No value set</td> </tr> <tr> <td>Kerberos Principal</td> <td>No value set</td> </tr> <tr> <td>Kerberos Keytab</td> <td>No value set</td> </tr> <tr> <td>Username</td> <td>No value set</td> </tr> <tr> <td>Password</td> <td>No value set</td> </tr> <tr> <td>Token Auth</td> <td>false</td> </tr> <tr> <td>SSL Context Service</td> <td>No value set</td> </tr> <tr> <td><b>Topic Name</b></td> <td><b>iot</b></td> </tr> </tbody> </table>				Property	Value	Kafka Brokers	cdp.3.108.43.134.nip.io:9092	Security Protocol	PLAINTEXT	SASL Mechanism	GSSAPI	Kerberos Service Name	No value set	Kerberos Credentials Service	No value set	Kerberos Principal	No value set	Kerberos Keytab	No value set	Username	No value set	Password	No value set	Token Auth	false	SSL Context Service	No value set	<b>Topic Name</b>	<b>iot</b>
Property	Value																												
Kafka Brokers	cdp.3.108.43.134.nip.io:9092																												
Security Protocol	PLAINTEXT																												
SASL Mechanism	GSSAPI																												
Kerberos Service Name	No value set																												
Kerberos Credentials Service	No value set																												
Kerberos Principal	No value set																												
Kerberos Keytab	No value set																												
Username	No value set																												
Password	No value set																												
Token Auth	false																												
SSL Context Service	No value set																												
<b>Topic Name</b>	<b>iot</b>																												
<input type="button" value="CANCEL"/> <input type="button" value="APPLY"/>																													

**Configure Processor**

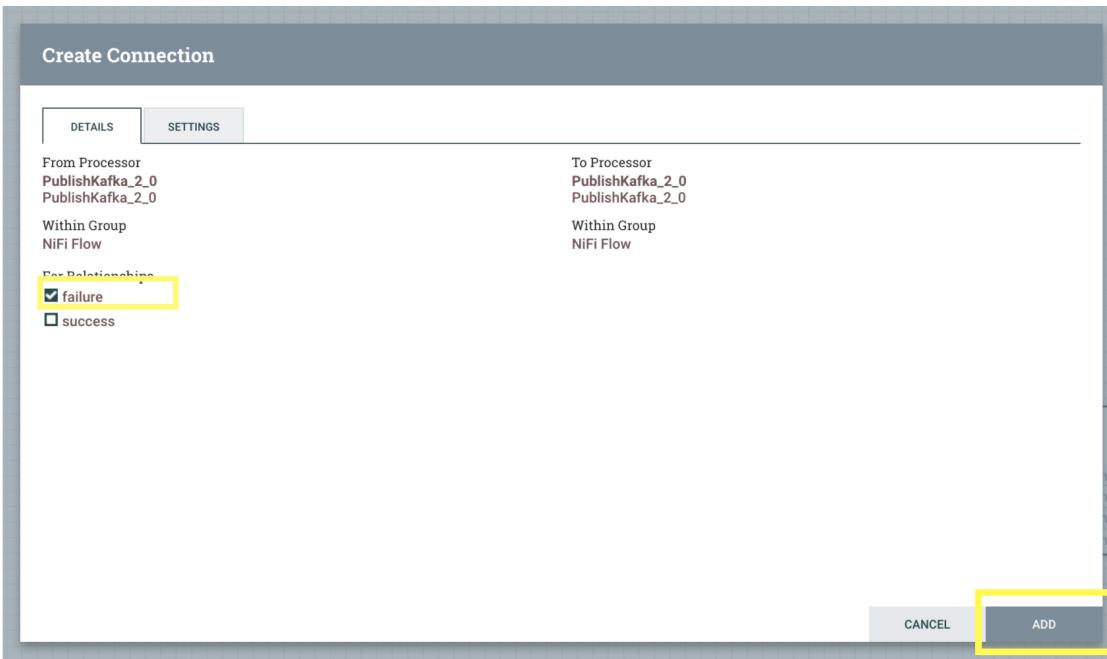
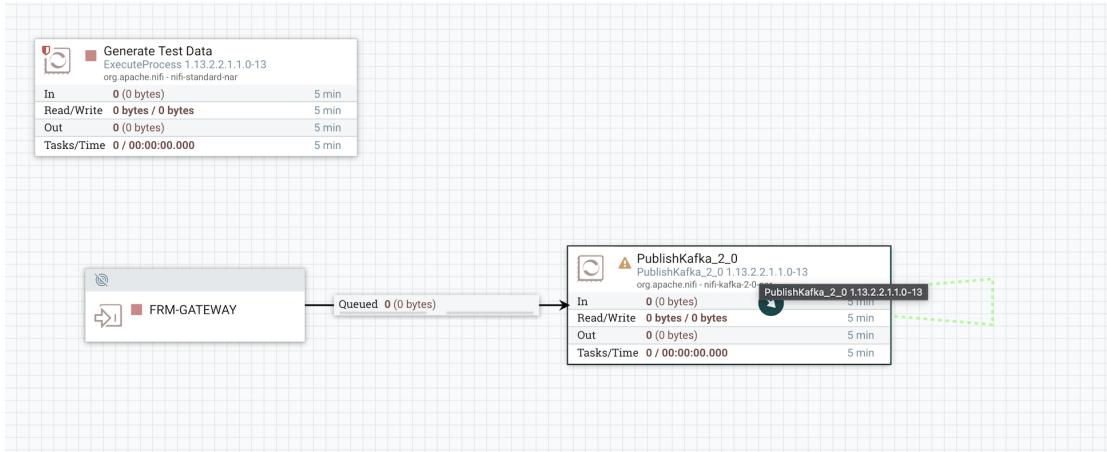
■ Stopped

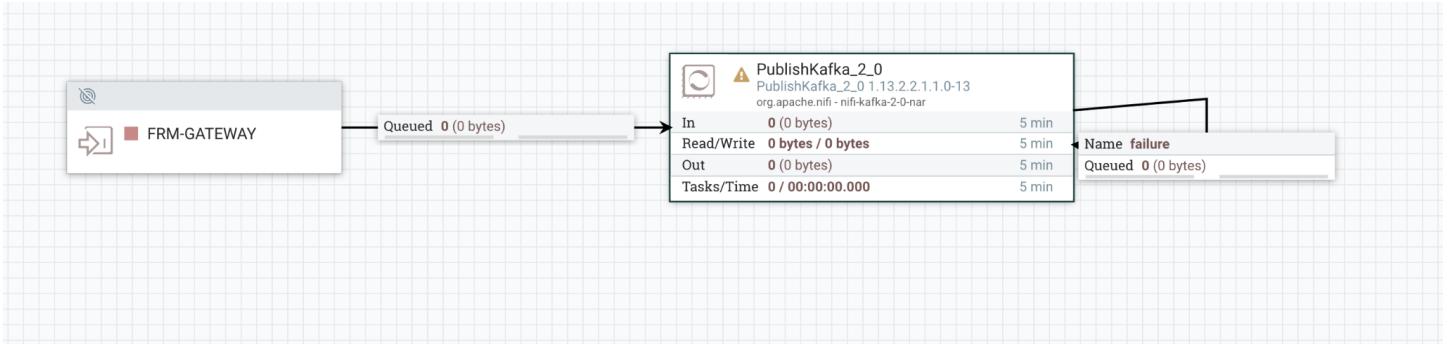
SETTINGS	SCHEDULING	PROPERTIES	COMMENTS																												
<b>Required field</b>																															
<table border="1"> <thead> <tr> <th>Property</th> <th>Value</th> </tr> </thead> <tbody> <tr> <td>Kerberos Keytab</td> <td>No value set</td> </tr> <tr> <td>Username</td> <td>No value set</td> </tr> <tr> <td>Password</td> <td>No value set</td> </tr> <tr> <td>Token Auth</td> <td>false</td> </tr> <tr> <td>SSL Context Service</td> <td>No value set</td> </tr> <tr> <td><b>Topic Name</b></td> <td><b>iot</b></td> </tr> <tr> <td>Delivery Guarantee</td> <td>Best Effort</td> </tr> <tr> <td>Failure Strategy</td> <td>Route to Failure</td> </tr> <tr> <td><b>Use Transactions</b></td> <td><b>false</b></td> </tr> <tr> <td>Transactional Id Prefix</td> <td>No value set</td> </tr> <tr> <td>Attributes to Send as Headers (Regex)</td> <td>No value set</td> </tr> <tr> <td>Message Header Encoding</td> <td>UTF-8</td> </tr> <tr> <td>Kafka Key</td> <td>No value set</td> </tr> </tbody> </table>				Property	Value	Kerberos Keytab	No value set	Username	No value set	Password	No value set	Token Auth	false	SSL Context Service	No value set	<b>Topic Name</b>	<b>iot</b>	Delivery Guarantee	Best Effort	Failure Strategy	Route to Failure	<b>Use Transactions</b>	<b>false</b>	Transactional Id Prefix	No value set	Attributes to Send as Headers (Regex)	No value set	Message Header Encoding	UTF-8	Kafka Key	No value set
Property	Value																														
Kerberos Keytab	No value set																														
Username	No value set																														
Password	No value set																														
Token Auth	false																														
SSL Context Service	No value set																														
<b>Topic Name</b>	<b>iot</b>																														
Delivery Guarantee	Best Effort																														
Failure Strategy	Route to Failure																														
<b>Use Transactions</b>	<b>false</b>																														
Transactional Id Prefix	No value set																														
Attributes to Send as Headers (Regex)	No value set																														
Message Header Encoding	UTF-8																														
Kafka Key	No value set																														
<input type="button" value="CANCEL"/> <input type="button" value="APPLY"/>																															

Connect the Input Port(From Gateway) to the PublishKafka processor by dragging the destination of the current connection from the Funnel to the PublishKafka.

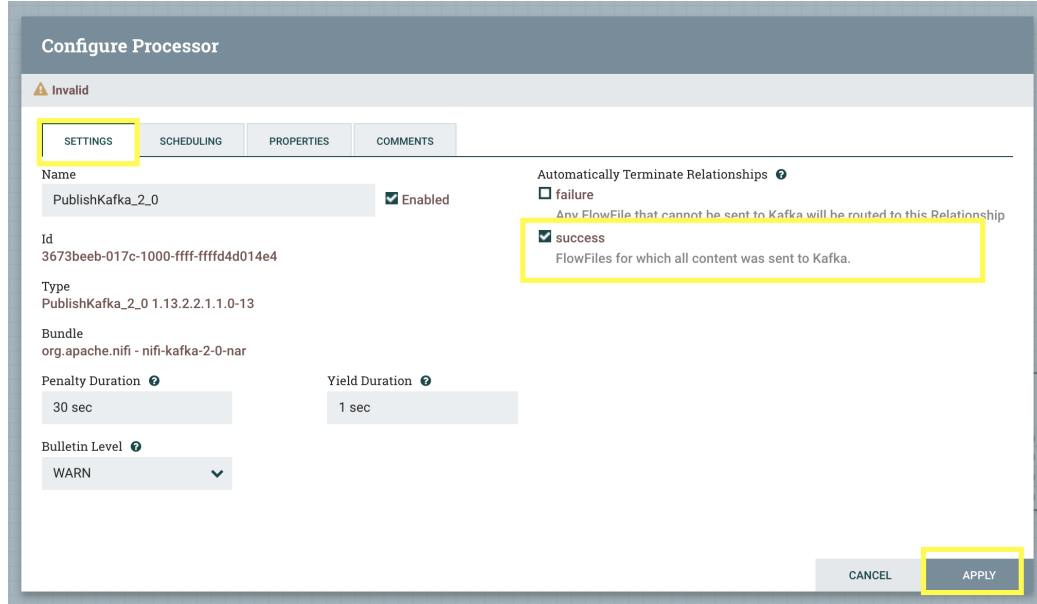


Create a connection from the PublishKafka to itself for failures (successes go to Kafka). Then you can start the Kafka processor.

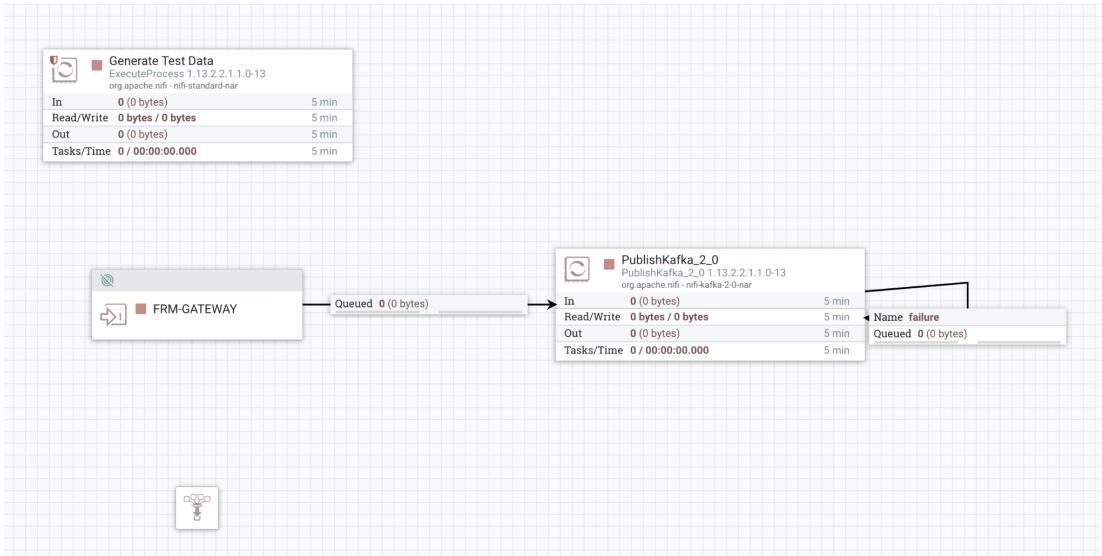




As the PublishKafka\_2.0 processor still shows an invalid mark we need to auto-terminate the processor on Success. Right click the process and click Configure, navigate to the Settings tab and check the box next to Success and click APPLY



The final flow on NiFi should look something like this



You can add more processors as needed to process, split, duplicate or re-route your FlowFiles to all other destinations and processors.

## Lab 4 - Streams Messaging Manager (SMM) to check and monitor Kafka

Now that our NiFi flow is pushing data to Kafka, it would be good to have a confirmation that everything is running as expected. In this lab you will use Streams Messaging Manager (SMM) to check and monitor Kafka.

1. Start the NiFi ExecuteProcess simulator again and confirm you can see the messages queued in NiFi. Leave it running.
2. To access the SMM Web UI use the following URL

`http://<PUBLIC-IP>:9991/#/`

3. Go to the Stream Messaging Manager (SMM) Web UI and familiarize yourself with the options there. Notice the filters (blue boxes) at the top of the screen.

NAME	DATA IN	DATA OUT	MESSAGES IN	CONSUMER GROUPS	Actions
iot	39 KB	0B	100	0	
_smm-app-sm... (1)	0B	0B	0	1	
_smm-app-sm... (1)	0B	0B	0	0	
_smm-app-sm... (1)	0B	0B	0	1	
_smm-app-sm... (1)	0B	0B	0	0	
_smm-app-sm... (1)	0B	0B	0	1	
_smm-app-sm... (1)	0B	0B	0	0	

- ❖ Filter by Topic and select the `iot` topic, you'll be able to see all the producers and consumers that are writing to and reading from it, respectively. Since we haven't implemented any consumers yet, the consumer list should be empty.
- ❖ Click on the topic to explore its details. You can see more details, metrics and the breakdown per partition. Click on one of the partitions and you'll see additional information and which producers and consumers interact with that partition.

The screenshot shows the Cloudera Data Flow interface for a Kafka topic named 'iot'. At the top, there are four summary cards: Producers (1 of 1), Brokers (1 of 1), Topics (1 of 15), and Consumer Groups (1). Below these, the 'TOPICS (1)' section displays 'iot' with a replication factor of 1. The 'MESSAGES' card shows 100 messages. A tooltip for partition P0 provides detailed metrics: DATA IN 4037, MESSAGES IN 100, and DATA OUT 0. Below the topic summary, there is a table for 'ALL PARTITIONS' with 8 rows, each representing a partition (P0 through P7) with 4 KB in and 0B out. A blue arrow points from the 'EXPLORER' link in the tooltip to the 'ALL PARTITIONS' button.

Click on the EXPLORE link to visualize the data in a particular partition. Confirm that there's data in the Kafka topic and it looks like the JSON produced by the sensor simulator.

The screenshot shows the Cloudera Data Flow interface for the 'iot' topic. The top navigation bar includes links for Status, CEM, NiFi Flow, NiFi Reg, Schema, STREAM, Hue - W, and cdsw.3d. The current page is 'Topics / iot'. A modal window titled 'iot' displays a JSON message: {"sensor\_id":46,"sensor\_ts":1565985015434644,"sensor\_0":1,"sensor\_1":3,"sensor\_2":0,"sensor\_3":23,"sensor\_4":15,"sensor\_5":75,"sensor\_6":42,"sensor\_7":80,"sensor\_8":46,"sensor\_9":7,"sensor\_10":4,"sensor\_11":1}. Below the modal, a data preview section shows a range from 'FROM OFFSET' (Partition 0, Offset 15) to 'TO OFFSET' (Offset 30). The main table lists five log entries:

Offset	Timestamp	Key	Value
15	Fri, Aug 16 2019, 12:50:15	null	{"sensor_id":46,"sensor_ts":1565985015434644,"sensor_0":1,"sensor_1":3,"sensor_2":0,"sensor_3":23,"sensor_4":15,"sensor_5":75,"sensor_6":...}
16	Fri, Aug 16 2019, 12:50:25	null	{"sensor_id":80,"sensor_ts":1565985025445219,"sensor_0":3,"sensor_1":16,"sensor_2":0,"sensor_3":27,"sensor_4":14,"sensor_5":97,"sensor_6":...}
17	Fri, Aug 16 2019, 12:50:35	null	{"sensor_id":88,"sensor_ts":1565985035457017,"sensor_0":1,"sensor_1":17,"sensor_2":0,"sensor_3":34,"sensor_4":13,"sensor_5":72,"sensor_6":...}
18	Fri, Aug 16 2019, 12:50:45	null	{"sensor_id":62,"sensor_ts":1565985045469664,"sensor_0":1,"sensor_1":7,"sensor_2":1,"sensor_3":25,"sensor_4":2,"sensor_5":62,"sensor_6":2...}
19	Fri, Aug 16 2019,	null	{"sensor_id":23,"sensor_ts":1565985055479293,"sensor_0":1,"sensor_1":6,"sensor_2":3,"sensor_3":29,"sensor_4":45,"sensor_5":89,"sensor_6":...}

## Lab 5 - From Kafka to Kudu

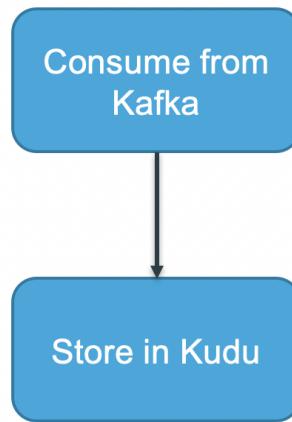
1. Open the browser and navigate to the HUE UI using the URL  
<http://<your vm-public-IP>:8888>

Create your account using the following credentials:

UserName : hueuser

Password: hueuser

2. Until lab 4 we have built our first flow. The next flow will read the Kafka records and store them in Kudu. The high level view of this flow looks like the following:



3. First, create the Kudu table. Login into Hue, and in the Impala Query, run this statement:

```
CREATE TABLE sensors
(
    sensor_id INT,
    sensor_ts TIMESTAMP,
    sensor_0 DOUBLE,
    sensor_1 DOUBLE,
    sensor_2 DOUBLE,
    sensor_3 DOUBLE,
    sensor_4 DOUBLE,
    sensor_5 DOUBLE,
    sensor_6 DOUBLE,
    sensor_7 DOUBLE,
    sensor_8 DOUBLE,
    sensor_9 DOUBLE,
    sensor_10 DOUBLE,
    sensor_11 DOUBLE,
    is_healthy INT,
    PRIMARY KEY (sensor_ID, sensor_ts)
)
PARTITION BY HASH PARTITIONS 16
STORED AS KUDU
TBLPROPERTIES ('kudu.num_tablet_replicas' = '1');
```

The screenshot shows the Cloudera Data Flow interface. On the left, there's a sidebar with various icons. In the center, under the 'Tables' section, a table named 'sensors' is listed. A yellow box highlights this table. To the right of the table, the table definition is shown in SQL:

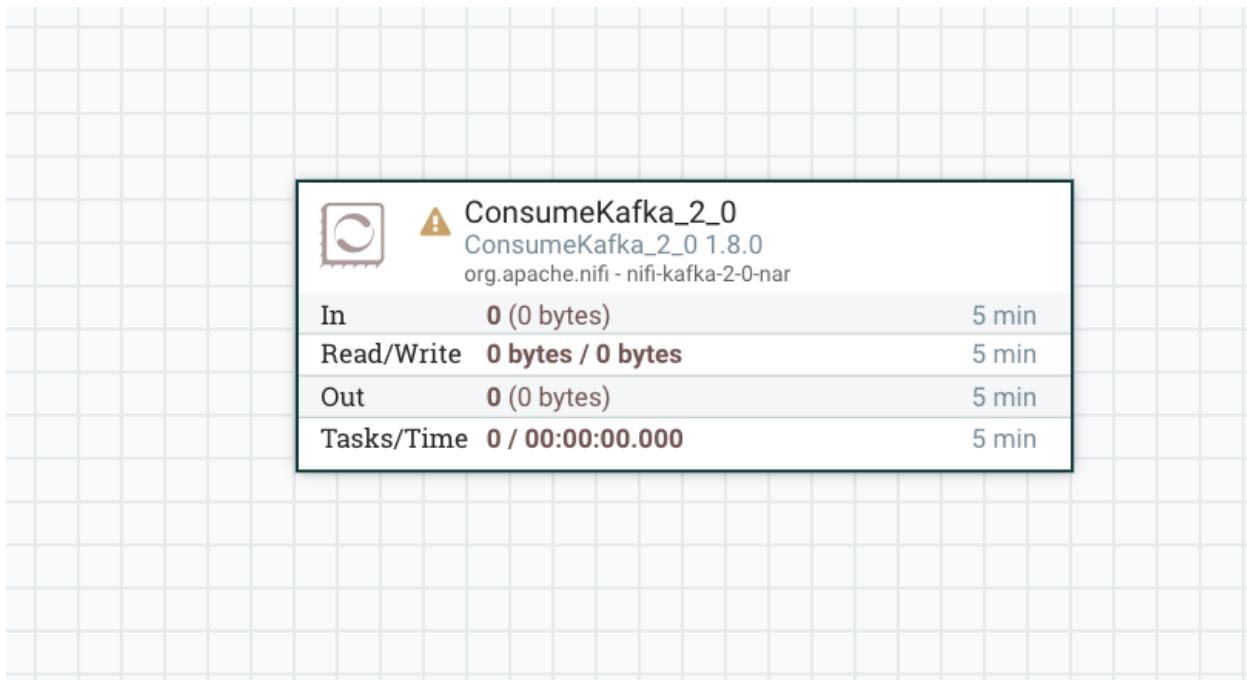
```

1 CREATE TABLE sensors
2 (
3   sensor_id INT,
4   sensor_ts TIMESTAMP,
5   sensor_0 DOUBLE,
6   sensor_1 DOUBLE,
7   sensor_2 DOUBLE,
8   sensor_3 DOUBLE,
9   sensor_4 DOUBLE,
10  sensor_5 DOUBLE,
11  sensor_6 DOUBLE,
12  sensor_7 DOUBLE,
13  sensor_8 DOUBLE,
14  sensor_9 DOUBLE,
15  sensor_10 DOUBLE,
16  sensor_11 DOUBLE,
17  is_healthy INT,
18 ) PRIMARY KEY (sensor_ID, sensor_ts)
19 )
20 PARTITION BY HASH PARTITIONS 16
21 STORED AS KUDU
22 TBLPROPERTIES ('kudu.num_tablet_replicas' = '1');

```

Below the table definition, it says 'No logs available at this moment.' At the bottom, there are tabs for 'Query History', 'Saved Queries', and 'Results (1)'. The 'Results (1)' tab is selected, showing a summary with one entry: '1 Table has been created.' This entry is also highlighted with a yellow box.

Let's start! Now you should already be familiar with how to add a new processor to the NiFi canvas. Add the **ConsumeKafka\_2\_0** processor:



Double click on the **ConsumeKafka\_2\_0** and go to the **PROPERTIES** tab

And add the value for the following properties:

- **Kafka Brokers:** <internal-hostname>:9092
- **Topic Name(s):** iot
- **Group ID:** nifi\_consumer

Once done, click **APPLY**

Now add a **PutKudu** processor to the canvas and connect the **ConsumeKafka** to the **PutKudu** component, then double click on **PutKudu** and go to the **SETTINGS** tab.

### Configure Processor

SETTINGS	SCHEDULING	PROPERTIES	COMMENTS
Name PutKudu	<input checked="" type="checkbox"/> Enabled	Automatically Terminate Relationships <small>?</small> <input type="checkbox"/> failure A FlowFile is routed to this relationship if it cannot be sent to Kudu <input checked="" type="checkbox"/> success A FlowFile is routed to this relationship after it has been successfully stored in Kudu	
Id 79ba58dc-0548-3d54-c320-04e5b3883b5b			
Type PutKudu 1.8.0			
Bundle org.apache.nifi - nifi-kudu-nar			
Penalty Duration <small>?</small> 30 sec	Yield Duration <small>?</small> 1 sec		
Bulletin Level <small>?</small> WARN			
<b>CANCEL</b> <b>APPLY</b>			

Check the **success** box and go to the **PROPERTIES** tab

The screenshot shows the 'Configure Processor' dialog box for a Kudu sink. The 'PROPERTIES' tab is selected. The table below lists the configuration properties:

Property	Value
Kudu Masters	No value set
Table Name	No value set
Skip head line	false
Record Reader	No value set
Insert Operation	INSERT
Flush Mode	AUTO_FLUSH_BACKGROUND
Batch Size	100

At the bottom right are 'CANCEL' and 'APPLY' buttons.

Set:

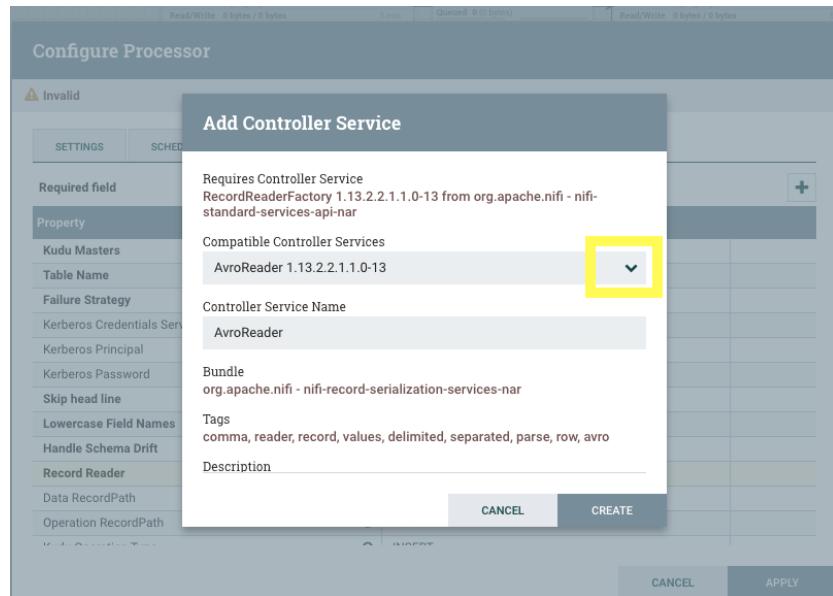
- **Kudu Masters:** <internal-hostname>:7051
- **Table Name:** default.sensors
- **Insert Operation:** UPSERT

Then for the **Record Reader**, click on **Create new service...**

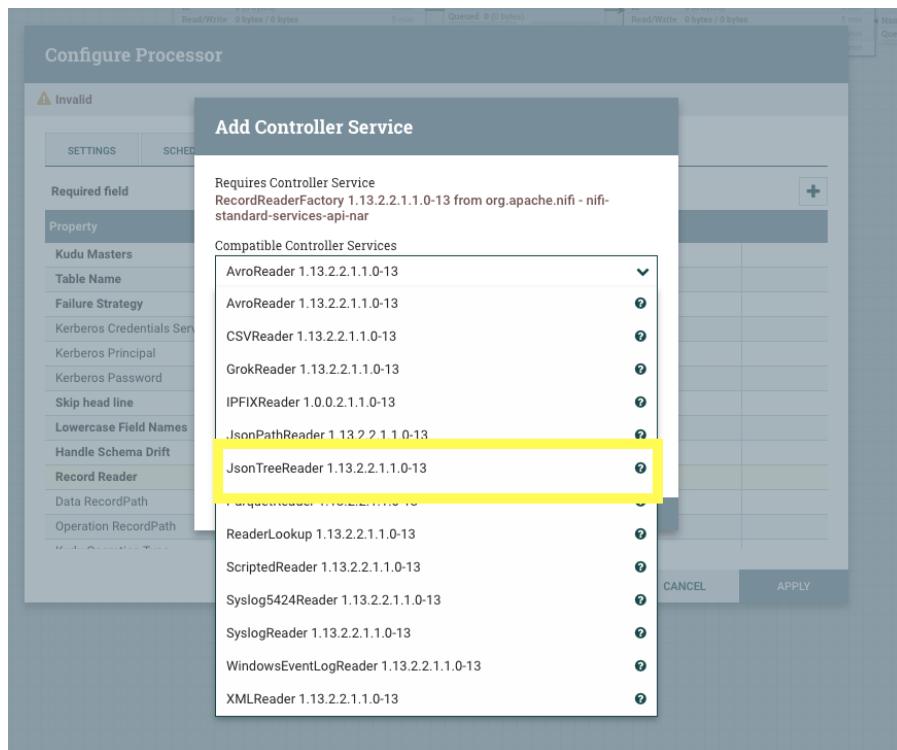
The screenshot shows the 'Configure Processor' dialog box for a Kudu source. The 'PROPERTIES' tab is selected. The table lists the properties, and the 'Record Reader' row has a dropdown menu open with the option 'Create new service...' highlighted by a yellow box.

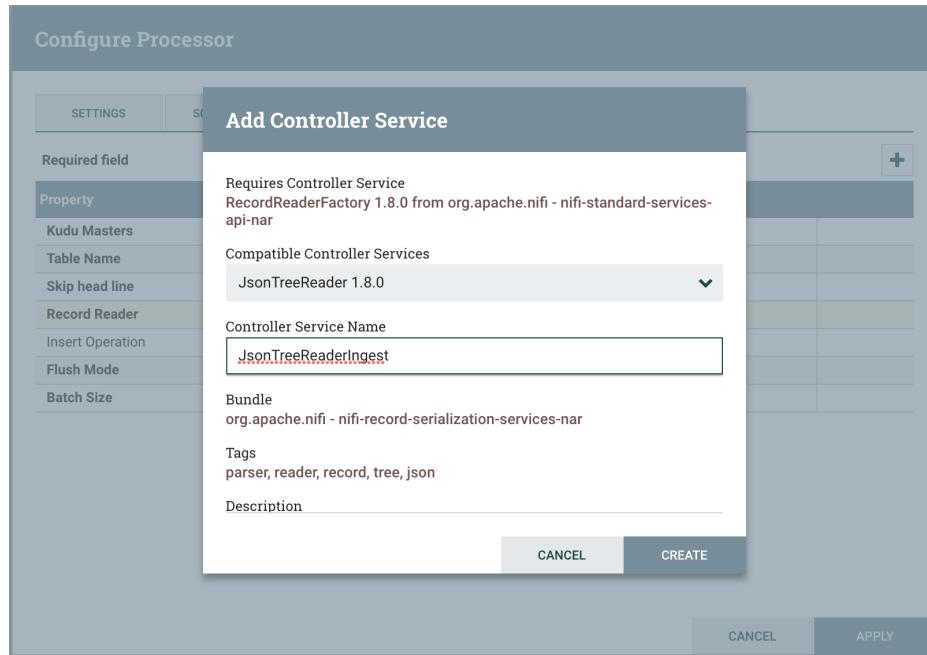
Property	Value
Kudu Masters	3.109.253.138:7051
Table Name	default.sensors
Failure Strategy	Route to Failure
Kerberos Credentials Service	No value set
Kerberos Principal	No value set
Kerberos Password	No value set
Skip head line	false
Lowercase Field Names	false

At the bottom right are 'CANCEL' and 'APPLY' buttons.



In the **Compatible Controller Services** select **JsonTreeReader** and give it a name (**JsonTreeReaderIngest**)





Then click **CREATE**, and click on the small arrow to bring you to the **Controller Services**.

Property	Value
Kudu Masters	cdp.3.109.253.138.nip.io:7051
Table Name	default.sensors
Failure Strategy	Route to Failure
Kerberos Credentials Service	No value set
Kerberos Principal	No value set
Kerberos Password	No value set
Skip head line	false
Lowercase Field Names	false
Handle Schema Drift	false
Record Reader	JsonTreeReaderIngest
Data RecordPath	No value set
Operation RecordPath	No value set

Click on the **configure** icon on the right end of the **JsonTreeReaderIngest** row and go to the **PROPERTIES** tab. Set



- Schema Access Strategy = Use 'Schema Name' Property
- Schema Name = demo-schema

## Create Schema Registry

The JsonReader needs to know how to interpret the json string it will receive, that is, it needs to know the schema. We will now create another **controller service** that will store the schema.

Property	Value
Schema Access Strategy	No value Create new service...
Schema Registry	
Schema Name	
Schema Version	
Schema Branch	
Date Format	No value set
Time Format	No value set
Timestamp Format	No value set

In the **Schema Registry** field click to the value field, click **Create new service** and select **AvroSchemaRegistry** Call it *AvroSchemaRegistryIngestion*.

Configure Controller Service

**Add Controller Service**

Requires Controller Service  
SchemaRegistry 1.13.2.2.1.1.0-13 from org.apache.nifi - nifi-standard-services-api-nar

Compatible Controller Services  
AvroSchemaRegistry 1.13.2.2.1.1.0-13

Controller Service Name  
**AvroSchemaRegistryIngestion**

Bundle  
org.apache.nifi - nifi-registry-nar

Tags  
schema, registry, csv, json, avro

Description

CANCEL CREATE

CANCEL APPLY

Then click **CREATE**.

Now click on the arrow icon in the **Schema Registry** row

Configure Controller Service

SETTINGS PROPERTIES COMMENTS

Required field

Property	Value
Schema Access Strategy	Use 'Schema Name' Property
Schema Registry	<b>AvroSchemaRegistryIngestion</b>
Schema Name	demo-schema
Schema Version	No value set
Schema Branch	No value set
Date Format	No value set
Time Format	No value set
Timestamp Format	No value set

Click →

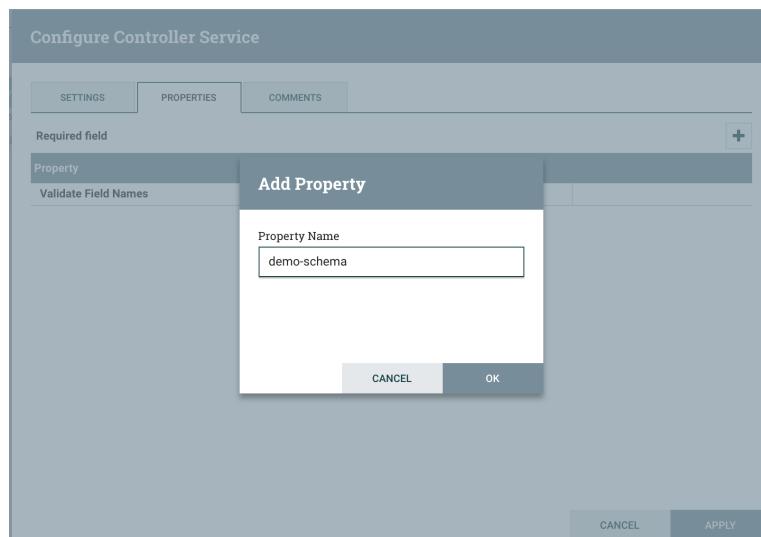
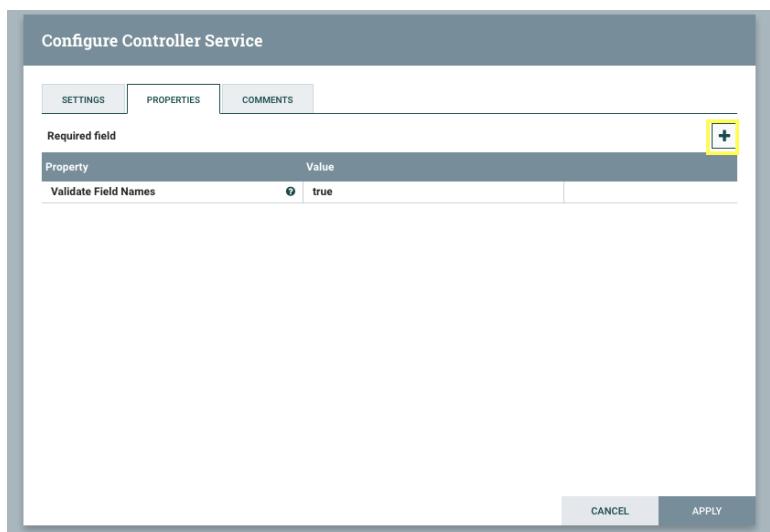
CANCEL APPLY

Click YES to save the changes.

Click on the **Configure** icon for the **AvroSchemaRegistryIngestion**



then click on the plus sign and add value *demo-schema*



## Insert the below schema:

```
{
  "type": "record",
  "name": "SensorReading",
  "namespace": "com.cloudera.example",
  "doc": "This is a sample sensor reading",
  "fields": [
    {
      "name": "sensor_id",
      "doc": "Sensor identification number.",
      "type": "int"
    },
    {
      "name": "sensor_ts",
      "doc": "Timestamp of the collected readings.",
      "type": "long"
    },
    {
      "name": "is_healthy",
      "doc": "Flag indicating health (healthy == 1)",
      "type": "int"
    },
    {
      "name": "response",
      "doc": "response record",
      "type": {
        "type": "record",
        "name": "CDSWResponse",
        "namespace": "com.cloudera.example",
        "doc": "This is a CDSW Model response",
        "fields": [
          {
            "name": "result",
            "doc": "Result",
            "type": "int"
          }
        ]
      }
    },
    {
      "name": "sensor_0",
      "doc": "Reading #0.",
      "type": "int"
    },
    {
      "name": "sensor_1",
      "doc": "Reading #1.",
      "type": "int"
    },
    {
      "name": "sensor_2",
      "doc": "Reading #2.",
      "type": "int"
    },
    {
      "name": "sensor_3",
      "doc": "Reading #3.",
      "type": "int"
    },
    {
      "name": "sensor_4",
      "doc": "Reading #4.",
      "type": "int"
    },
    {
      "name": "sensor_5",
      "doc": "Reading #5.",
      "type": "int"
    },
    {
      "name": "sensor_6",
      "doc": "Reading #6.",
      "type": "int"
    },
    {
      "name": "sensor_7",
      "doc": "Reading #7.",
      "type": "int"
    },
    {
      "name": "sensor_8",
      "doc": "Reading #8.",
      "type": "int"
    },
    {
      "name": "sensor_9",
      "doc": "Reading #9."
    }
  ]
}
```

```
"doc": "Reading #9.",  
"type": "int"  
},  
{  
"name": "sensor_10",  
"doc": "Reading #10.",  
"type": "int"  
},  
{  
"name": "sensor_11",  
"doc": "Reading #11.",  
"type": "int"  
}  
]  
}
```

And click **OK**, then click **APPLY**.

NiFi Flow Configuration

GENERAL CONTROLLER SERVICES

Name	Type	Bundle	State	Scope
AvroSchemaRegistryIngestion	AvroSchemaRegistry 1.8.0	org.apache.nifi - nifi-registry-nar	Disabled	NiFi Flow
JsonTreeReaderIngestion	JsonTreeReader 1.8.0	org.apache.nifi - nifi-record-serialization-service...	Invalid	NiFi Flow

On the right, click the **Enable** icon for the **AvroSchemaRegistryIngestion**



## Enable Controller Service

Service AvroSchemaRegistryIngestion

Scope **Service only**

Referencing Components **Controller Services (1)**  
**JsonTreeReaderIngest** *JsonTreeReader*

**CANCEL** **ENABLE**

Click **ENABLE** and then **CLOSE**, and the warning icon should disappear

NiFi Flow Configuration

**GENERAL** **CONTROLLER SERVICES**

Name	Type	Bundle	State	Scope
AvroSchemaRegistryIngestion	AvroSchemaRegistry 1.8.0	org.apache.nifi - nifi-registry-nar	Enabled	NiFi Flow
JsonTreeReaderIngest	JsonTreeReader 1.8.0	org.apache.nifi - nifi-record-serialization-service...	Disabled	NiFi Flow

Then enable also the **JsonTreeReaderIngest**.

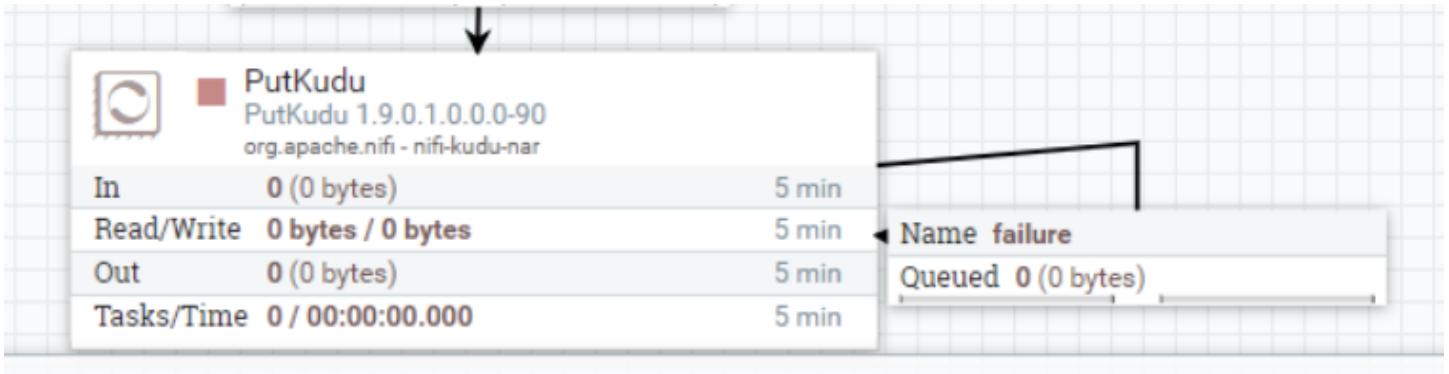
NiFi Flow Configuration

**GENERAL** **CONTROLLER SERVICES**

Name	Type	Bundle	State	Scope
AvroSchemaRegistryIngestion	AvroSchemaRegistry 1.8.0	org.apache.nifi - nifi-registry-nar	Enabled	NiFi Flow
JsonTreeReaderIngest	JsonTreeReader 1.8.0	org.apache.nifi - nifi-record-serialization-service...	Enabled	NiFi Flow

You can now close the **NiFi Flow Configuration** and go back to the main NiFi canvas

Now drag a connection from the **PutKudu** to itself



Now we are ready to test the end2end flows. Start each component and verify that data is stored in Kudu:

A screenshot of the Cloudera Kudu UI. At the top, a query window shows two rows of code: "11" and "12 SELECT \* FROM sensors;". Below the query window, a log pane displays two messages: "Query 4541e78baa8961ae:bafe5c9000000000: 0% Complete (0 out of 16)" and "Query 4541e78baa8961ae:bafe5c9000000000 100% Complete (16 out of 16)". At the bottom, a results table titled "Results (4)" is shown with four rows of data. A red bracket on the left groups the query window and log pane, while a red bracket on the right groups the results table.

	id	val	ts
1	4634	102	2019-08-12 15:37:45.271000000
2	7850	207	2019-08-12 15:36:41.528000000
3	6640	24	2019-08-12 15:37:33.268000000
4	4548	162	2019-08-12 15:37:39.270000000

## Lab 6 - Fast analytics on fast data with Kudu and Impala

In this lab, you will run some SQL queries using the Impala engine. You can run a report to inform you which machines are likely to break in the near future.

Login into Hue, and run the following statement in the Impala Query

```
select sensor_id, sensor_ts from sensors where is_healthy = 0;
```

Run a few times a SQL statement to count all rows in the table to confirm the latest inserts are always picked up by Impala. This allows you to build real-time reports for fast action.

The screenshot shows the Cloudera Manager interface with several tabs at the top: Cloudera Manager, Hue - Editor (which is active), NiFi Flow, and CEM. Below the tabs, a URL bar shows 'Not secure | 18.215.124.254:8888/hue/editor?editor=5'. A message box says 'You are accessing a non-optimized Hue, please switch to one of the available addresses'. The main area is titled 'HUE' with a 'Query' dropdown set to 'Impala'. On the left, there's a sidebar with 'Tables' (default, 1 table) and a 'sensors' table listed. The main query editor window contains the SQL statement: '1 select sensor\_id, sensor\_ts from sensors where is\_healthy = 0;'. The results section shows a table with 5 rows:

	sensor_id	sensor_ts
1	67	2019-05-02 15:57:02.947224000
2	2	2019-05-02 15:56:57.941931000
3	12	2019-05-02 15:56:07.904867000
4	67	2019-05-02 15:57:12.957365000
5	63	2019-05-02 15:55:52.892034000

This concludes Cloudera Data Flow Workshop

---

Thanks for your participation..