HiveOperator Demonstration

This document will guide you through the demonstration of the HiveOperator in Session 2 of the Airflow module.

Prerequisites:

- Data in the HDFS location from the SqoopOperator demonstration
- sample_hive.py(the code explained in the video)
- JDK version is 8. Steps to change the version of JDK are present in the Airflow installation segment

What are we doing?
In this demonstration, we need to create a DAG with four Hive tasks.
They will perform the following functions:

- create_hive_database - Creates the Hive database
- create_raw_table - Creates the raw Hive table
- create_filtered_table - Creates the filtered Hive table
- load_filtered_table - Load filtered records into a different hive table in the Parque file format

Please follow the instructions below:

1. Login to your EMR instance.

2. Activate the Python virtual environment using the following command:

   source /home/hadoop/airflow/bin/activate

3. Now in this demonstration, we will be using the data we transferred into HDFS in the Sqoop demonstration. You can use the following command to check the same

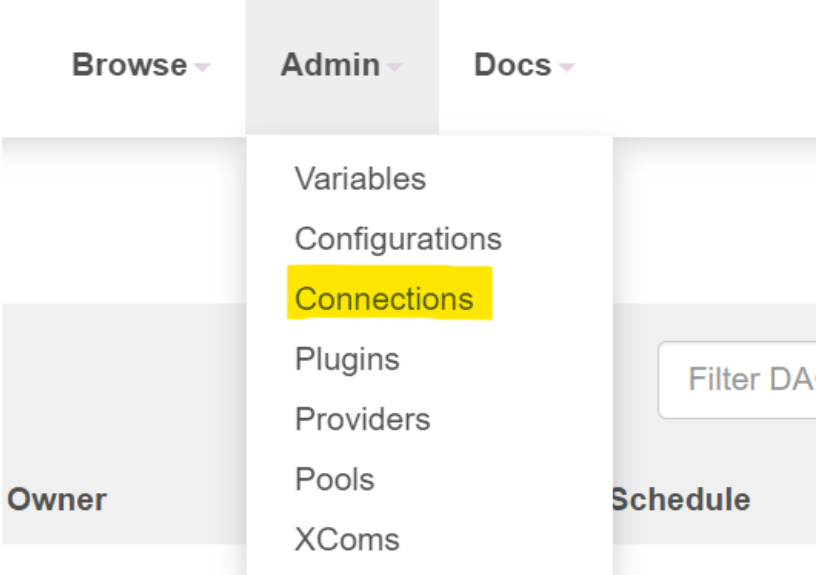
   hdfs dfs -cat hdfs:///data/credit_card/transactions/*

   If you don't have this data, you will have to finish the Sqoop demonstration first to get the data in the above HDFS location

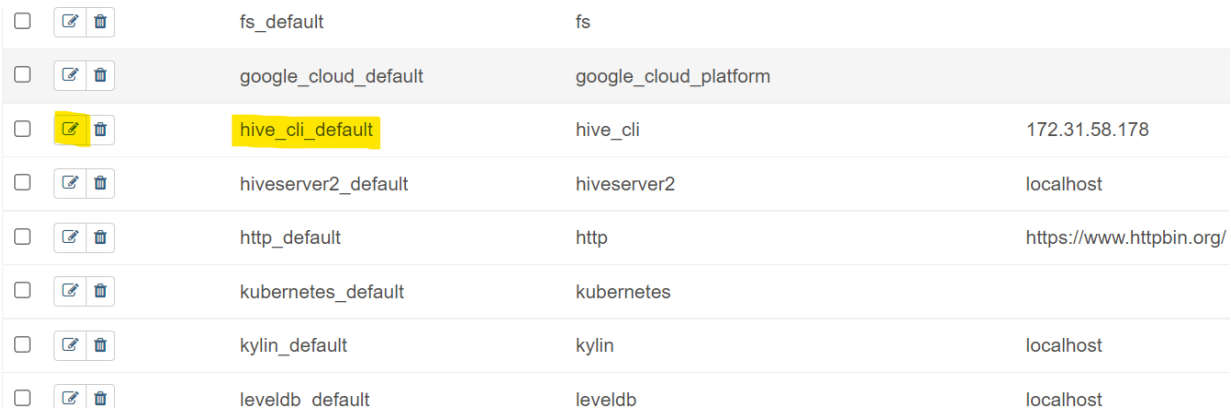
4. Next, you need to set up the Hive connection from the Airflow UI which is hosted in the URL : **your_public_dns:8082**

   Note: You can find you_publin_ip in your AWS EMR dashboard (IPv4 Public DNS))

Go to the Admin tab and click on Connections

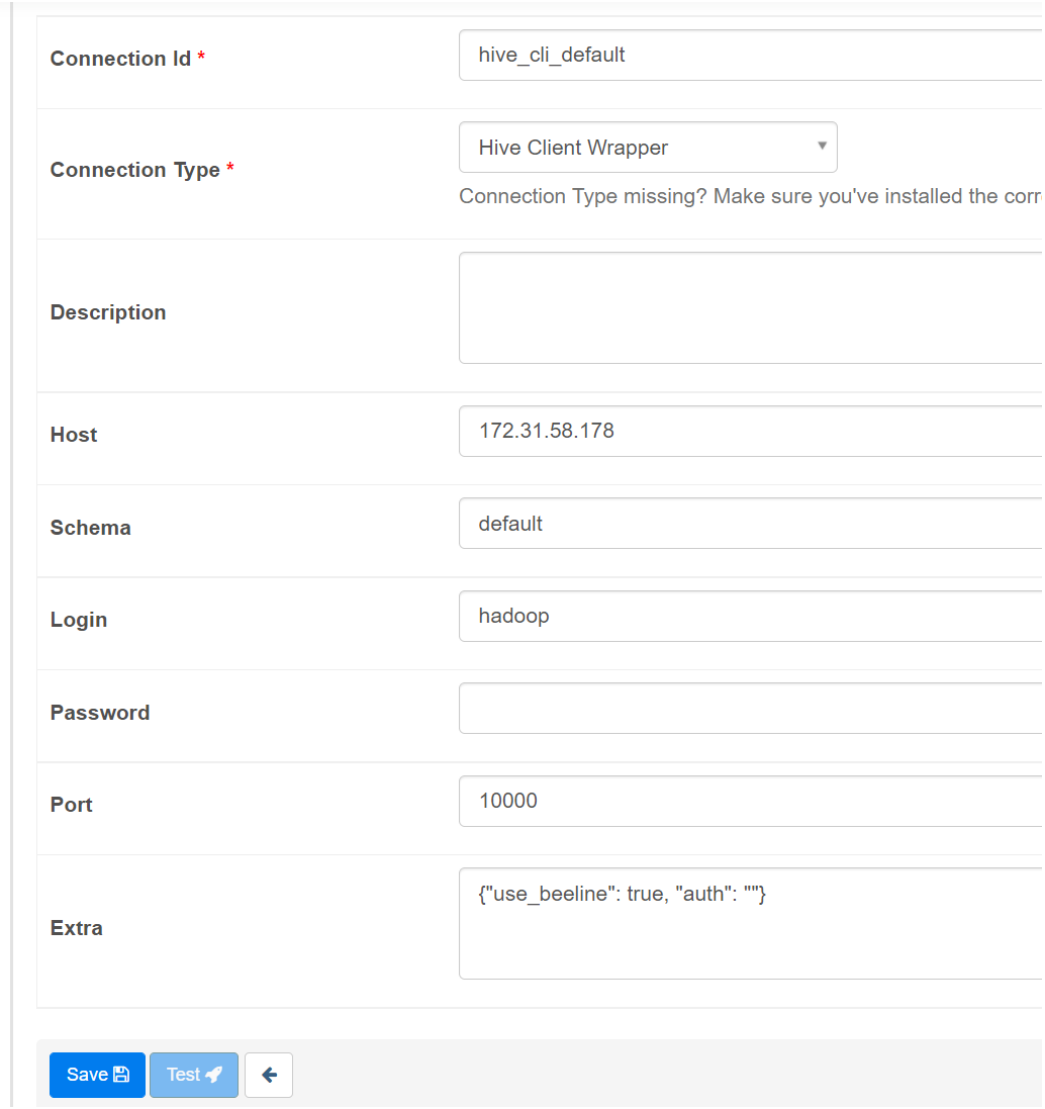


Now click on the edit button to the left of the **hive_cli_default** connection.



fs_default	fs		
google_cloud_default	google_cloud_platform		
hive_cli_default	hive_cli	172.31.58.178	
hiveserver2_default	hiveserver2	localhost	
http_default	http	https://www.httpbin.org/	
kubernetes_default	kubernetes		
kylin_default	kylin	localhost	
leveldb_default	leveldb	localhost	

Next, fill in the following details and click on Save



Conn Id: hive_cli_default

Conn Type: Hive Client Wrapper (Select from the drop-down)

Host: **<private IP of master node>**

Login: **hadoop**

Port: 10000

Extra: {“use_beeline”: true, “auth”: “”}

5. Now you need to place the **sample_hive.py** file in the **/home/hadoop/airflow/dags** directory. (You can use WinSCP or create a new file and paste the code in that file)

6. To ensure that the file there are no issues/errors with the file is it considered good practice to compile the program using the following command:

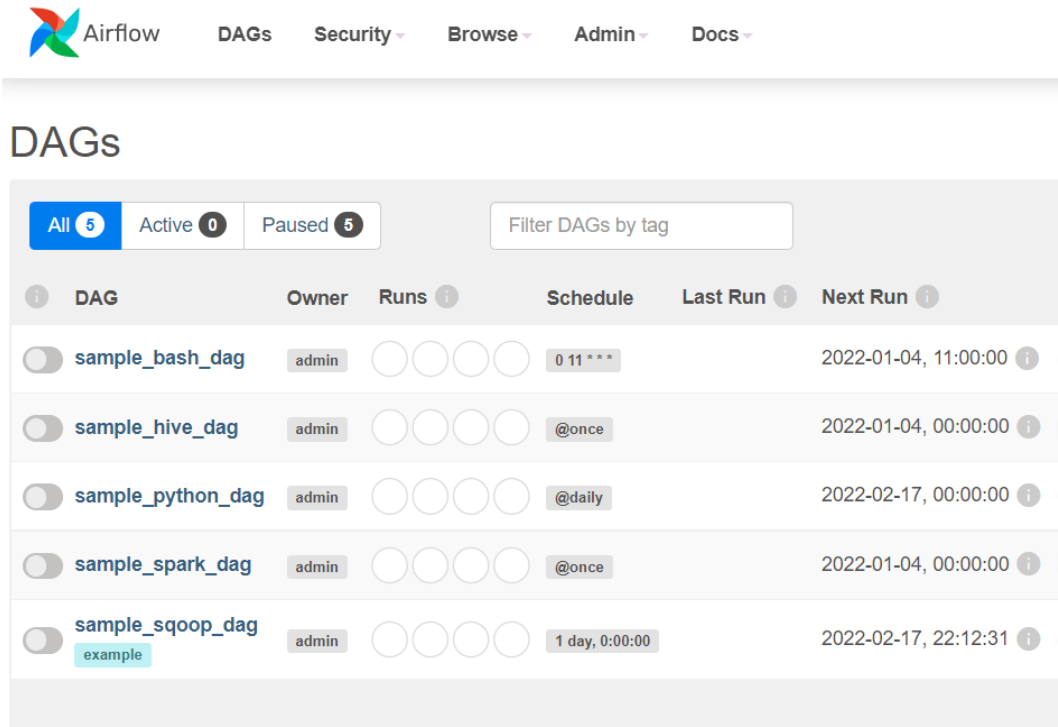
   python sample_hive.py

7. You can also use the following command to list the dags in your instance:

   airflow dags list

8. Once you have made sure that your dag file has no issues you can go back to the Airflow UI

9. Switch ON the DAG(sample_hive_dag)

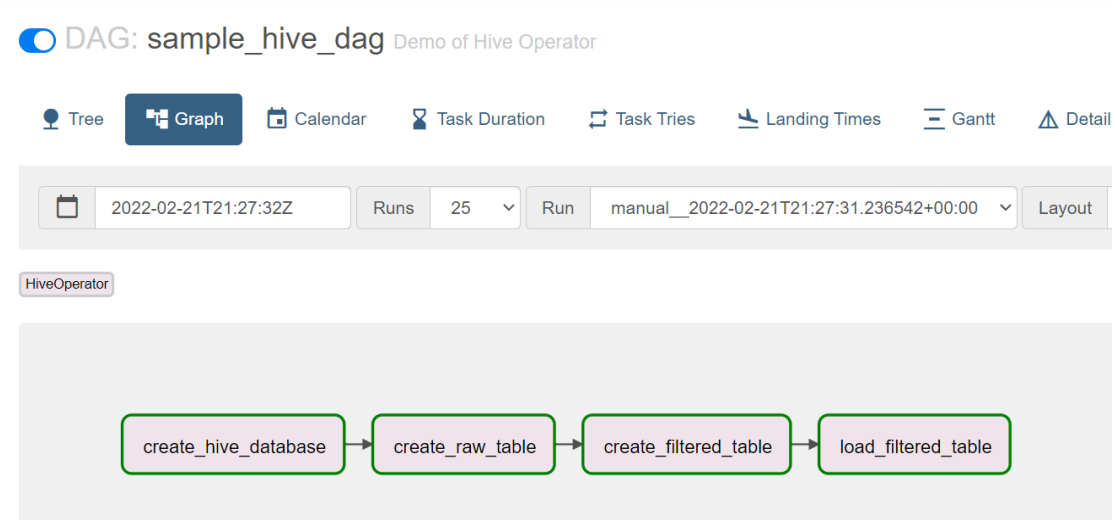


   (Note: The DAG might take a while to show up on the UI. Keep refreshing and wait patiently)

10. Click on the sample_hive_dag and go to the graph view

    You will see the task is running

    Click on refresh and eventually all files will have successfully completed



11. Once the DAG has completed execution, the output will be generated in the tables transactions and filtered_transactions inside the credit_card database

12. You can view the results in the CLI by following the steps below:

    (**Note:** in the video, Amit used beeline whereas here we will use the hive shell. You can whatever you find comfortable)

    Enter **hive**

```
(airflow) [hadoop@ip-172-31-58-178 ~]$ hive

Logging initialized using configuration in file:/etc/hive/conf.dist/hive-log4j2.properties Async: true
hive>
```

    Enter **show databases;** to see the available databases

```
hive> show databases;
OK
credit_card
default
events
Time taken: 1.146 seconds, Fetched: 3 row(s)
hive>
```

    We need the credit_card database, So enter **use credit_card;**

```
hive> use credit_card;
OK
Time taken: 0.049 seconds
hive>
```

Use **show tables;** to view the tables present

```
hive> show tables;
OK
filtered_transactions
transactions
Time taken: 0.056 seconds, Fetched: 2 row(s)
hive>
```

Use the following queries to see the records in these tables:

select * from transactions;

```
hive> select * from transactions;
OK
1       U101    I301    1600598377      1600599217      20.0
2       U102    I302    1600588362      1600588361      60.0
3       U102    I305    1600588312      1600599326      -100.0
4       U103    I307    1600588342      1600599332      20.0
5       U105    I303    1600588361      1600599325      40.0
6       U106    I304    1600588325      1600599356      NULL
7       U107    I302    1600588352      1600599337      60.0
8       U103    I305    1600588336      1600599353      30.0
9       U107    I302    1600588354      1600599338      10.0
10      U105    I302    1600588317      1600599326      50.0
Time taken: 3.62 seconds, Fetched: 10 row(s)
hive>
```

select * from filtered_transactions;

```
hive> select * from filtered_transactions;
OK
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for
1       U101    I301    1600598377      1600599217      20.0
3       U102    I305    1600588312      1600599326      -100.0
4       U103    I307    1600588342      1600599332      20.0
5       U105    I303    1600588361      1600599325      40.0
7       U107    I302    1600588352      1600599337      60.0
8       U103    I305    1600588336      1600599353      30.0
9       U107    I302    1600588354      1600599338      10.0
10      U105    I302    1600588317      1600599326      50.0
Time taken: 0.298 seconds, Fetched: 8 row(s)
hive>
```

13. You can switch off your DAG if you don't want it to run anymore.