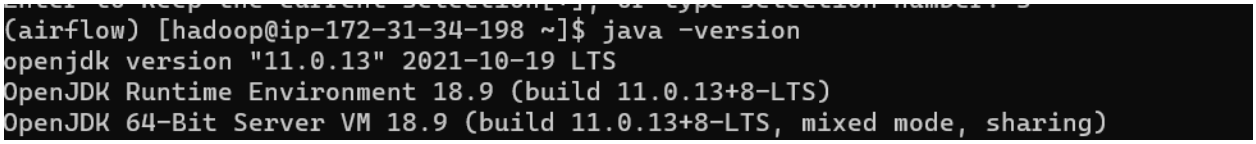
Final Demonstration

This document will guide you through the demonstration of the real-world use case of Airflow in Session 3 of the Airflow module.

Prerequisites:

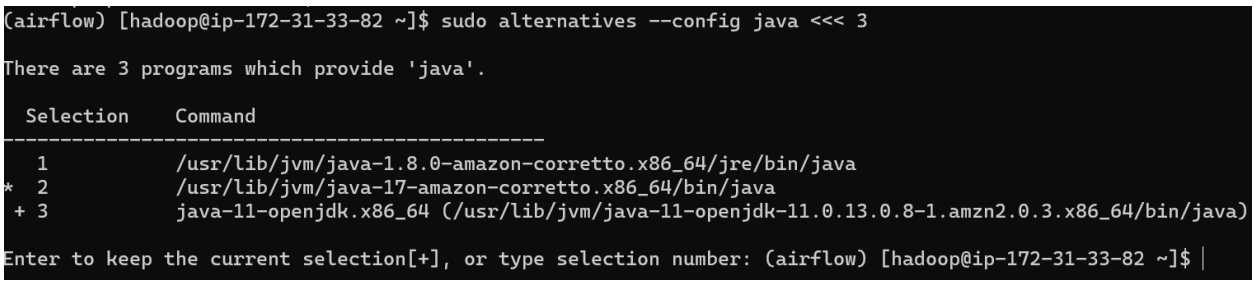
- mysql.dump (MySQL queries for table creation and record)
- etl_dag.py(the code for the DAG explained in the video)
- filter_trip.py (Spark application)
- generate_trip_throughput.py (Spark application)
- filter_booking.py (Spark application)
- generate_car_with_most_trips.py (Spark application)
- Make sure that the current Java version is Java 11(11.x.x). **Note**: This is only needed to run the Sqoop operator and you will need to switch back to Java 8 after you are done with the Sqoop operator. The steps to switch back to Java 8 can be found at the end of this document.
  You can check this by running the following command:

```
java -version
```



If you still have Java 8, then you need to switch to Java 11 by running the following command.

```
sudo alternatives --config java <<< 3
```



What are we doing?
We will we creating a DAG for the ride-hailing problem statement explained in Session 3

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

   ```
   [hadoop@ip-172-31-33-82 ~]$ source /home/hadoop/airflow/bin/activate
   (airflow) [hadoop@ip-172-31-33-82 ~]$
   ```

3. Now you need to load the data into your local MySQL, firstly you need to place the mysql.dump file in some location in your EMR machine. We will store in the /home/hadoop/ directory

   (You can use WinSCP or create a new file called mysql.dump in the /home/hadoop/ directory and paste the contents in that file)

4. Now run the following command to execute the SQL commands in /tmp/transactions.dump and create our tables :

   mysql -u root -p123 < /home/hadoop/mysql.dump

5. Next, you need to set up the different connections using 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))

   Edit the following connections:

   Sqoop:

   Conn Id: sqoop_default

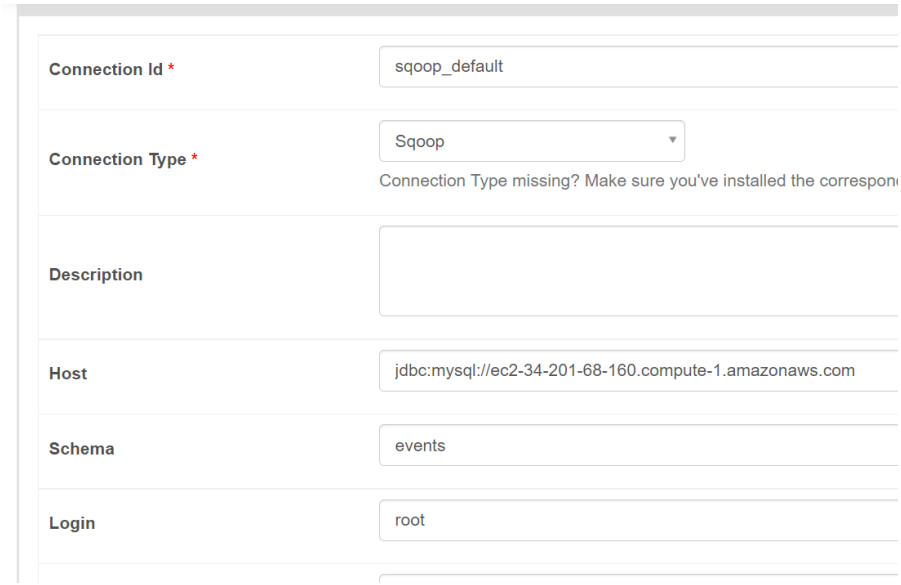
   Conn Type: Sqoop (Select from the drop-down)

   Connection URL: **jdbc:mysql://<public DNS>**

   Schema: **events**

   Login: **root**

   Password: **123**



Hive:

Conn Id: hive_cli_default

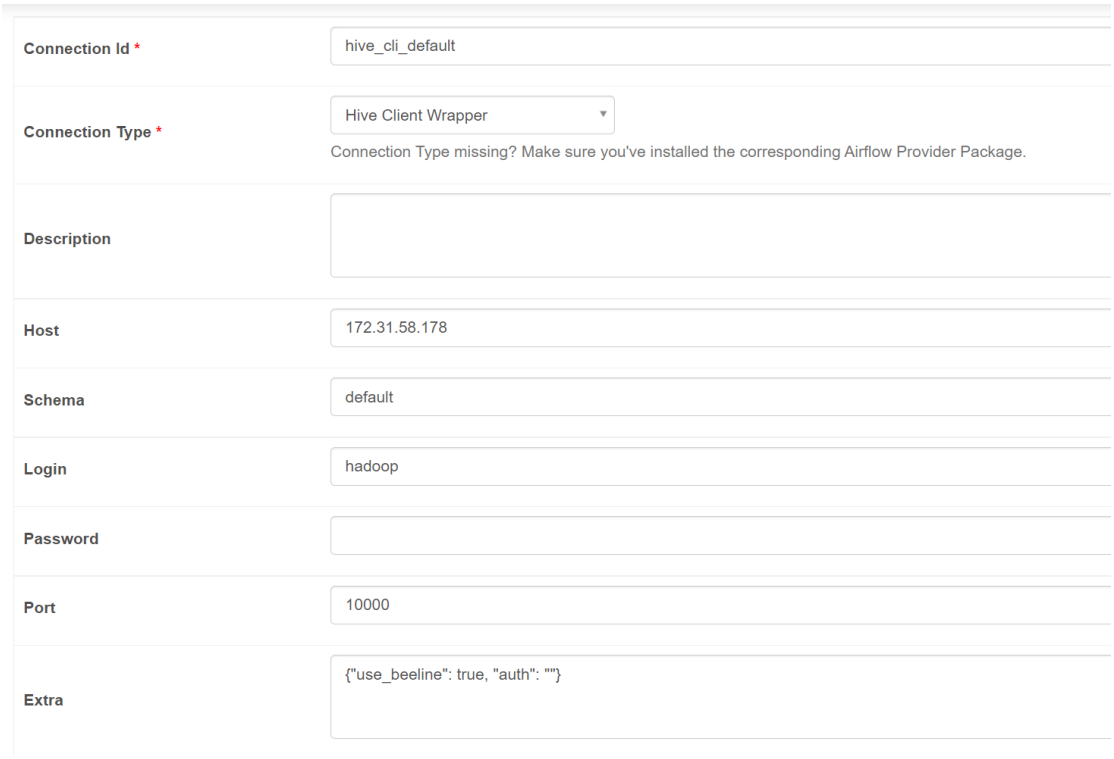
Conn Type: Hive Client Wrapper (Select from the drop-down)

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

Login: **hadoop**

Port: 10000

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



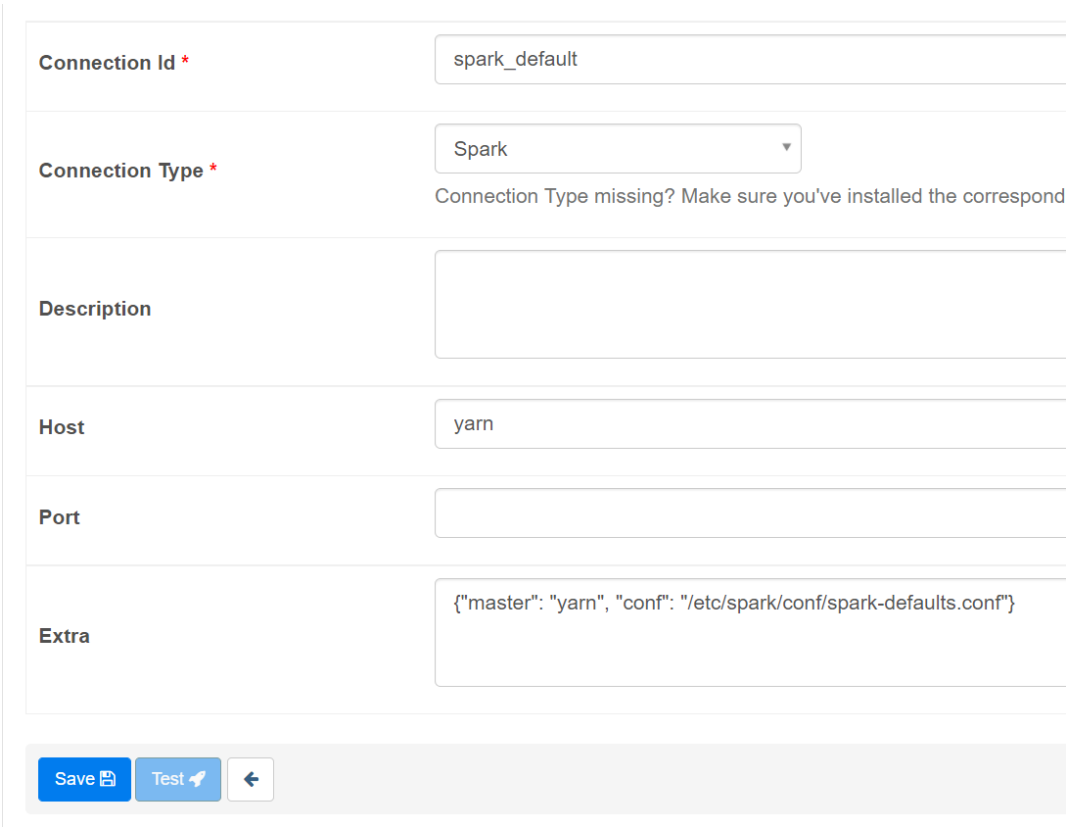
Spark:

Conn Id: spark_default

Conn Type: Spark (Select from the drop-down)

Host: yarn

Extra: **{"master": "yarn", "conf": "/etc/spark/conf/spark-defaults.conf"}**



6. Create a directory called **uber** in the airflow_codes directory using the following command:

   mkdir -p /home/hadoop/airflow_codes/uber

7. Now place the following file inside the **uber directory** you just created:
   - filter_trip.py
   - generate_trip_throughput.py
   - filter_booking.py
   - generate_car_with_most_trips.py

   (You can use WinSCP or create a new file and paste the code in that file)

```
(airflow) [hadoop@ip-172-31-58-178 ~]$ ls airflow_codes/uber/
filter_booking.py  generate_car_with_most_trips.py
filter_trip.py     generate_trip_throughput.py
(airflow) [hadoop@ip-172-31-58-178 ~]$
```

8. Now you need to place the **etl_dag.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)

9. 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 etl_dag.py

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

    airflow dags list

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

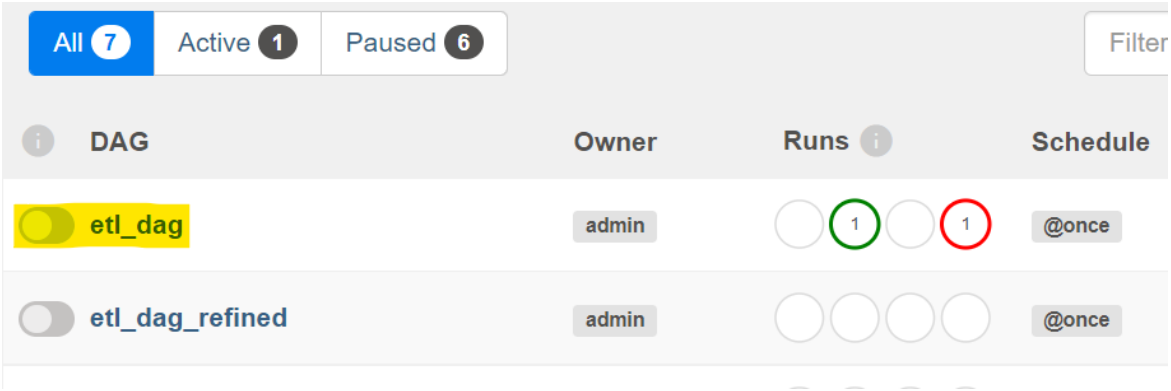
12. In case you are re-running this DAG, you will have to delete the target_dir of the sqoop task as we did in the SqoopOperator demonstration to avoid any errors.

    You can enter the following command to do so:

    sudo hdfs dfs -rm -r -skipTrash /data/raw

    Also, clear the task/DAG before re-running it.

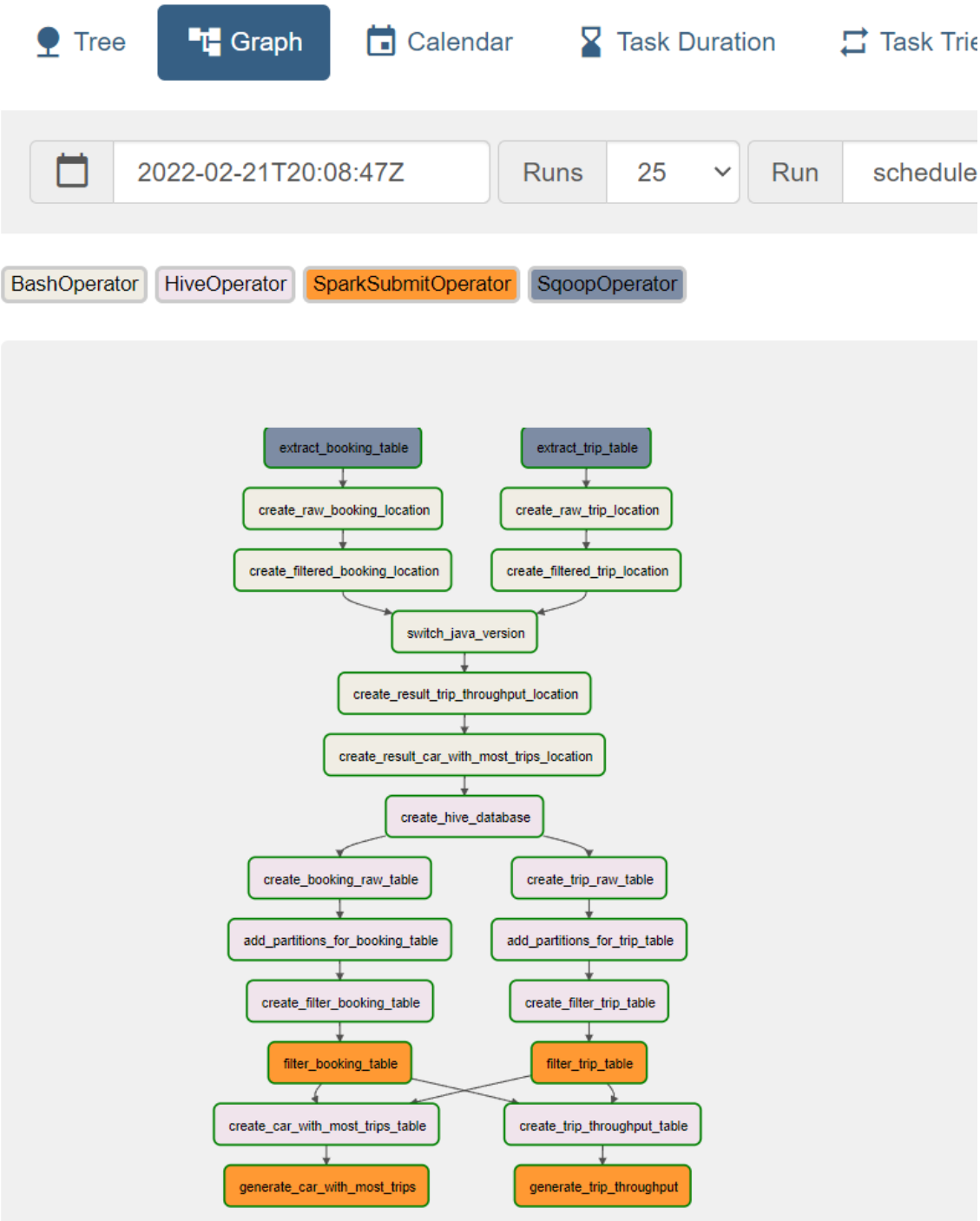
13. Switch ON the DAG(etl_dag)



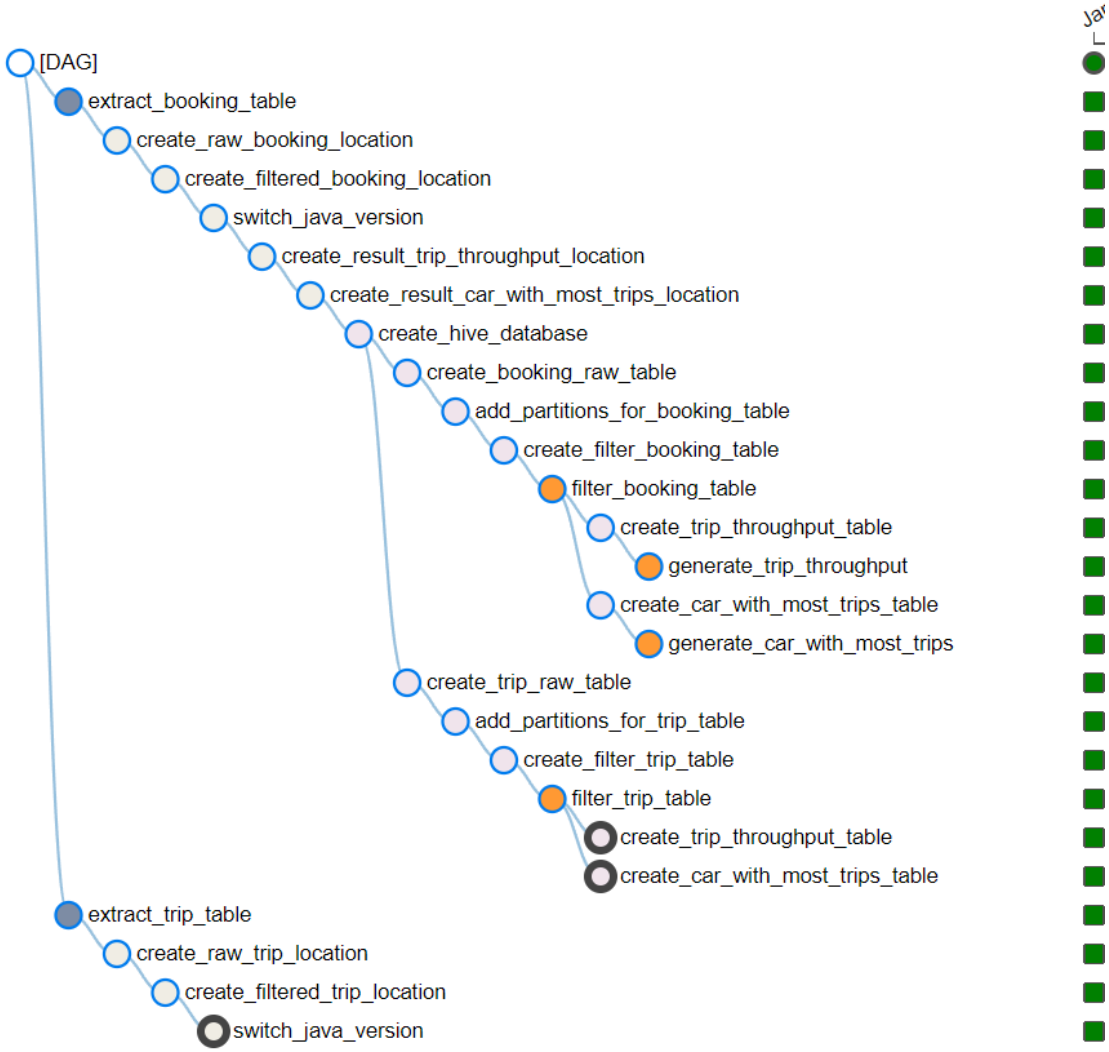
    (Note: The DAG might take a while to show up on the UI. Keep refreshing and wait patiently)

14. Click on the etl_dag and go to the graph view

    You will see the task is running



This may take a while. So wait patiently.



Click on refresh and eventually, all tasks will have successfully completed

15. Once the DAG has completed execution, the output will be generated in the tables **trip_troughput** and **car_with_most_trips** inside the **events** database

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

    For trip_throughtput enter command:

    hive -e "select * from events.trip_throughput;"

```
mumbai  0.75    2022-01-16
bangalore       0.9     2022-01-16
chennai 0.8     2022-01-16
Time taken: 6.207 seconds, Fetched: 3 row(s)
```

For car_with_most_trips enter command:

hive -e "select * from events.car_with_most_trips ;"

```
bangalore       sedan   5       2022-01-16
mumbai  economy 3       2022-01-16
mumbai  sedan   3       2022-01-16
chennai sedan   5       2022-01-16
Time taken: 5.442 seconds, Fetched: 4 row(s)
```

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