

# **Real-time Credit card Fraud detection using Spark Streaming, Spark ML, Spark SQL, Kafka, Cassandra and Airflow**

<b>1. Image and Code Setup</b>	<b>2</b>
1.1 Clone Code	2
1.2 Import to IntelliJ	3
<b>2. Initial Import Spark Job</b>	<b>9</b>
2.1 Demonstration	9
<b>3. Spark Training Job</b>	<b>15</b>
3.1 Demonstration	15
<b>4. Spark Streaming</b>	<b>17</b>
4.1 Demonstration	17
<b>5. Airflow Automation</b>	<b>22</b>
5.1 Configure Airflow	22
5.1.1 Create hduser and airflow database	22
5.1.2 Modify airflow.cfg	23
5.2 Build all Projects	25
5.3 Start all the Servers	26
5.4 Airflow Automation	29

**Note: In the video course, I am automating both Spark ML and Spark Streaming Job using Apache Airflow. This is just to show how multiple Spark Jobs are automated using Apache Airflow.**

**In reality, the Spark ML Job will not be automated. It will be manually run once a week, or once a month to create a new model. The model will be tested for efficiency. Also, it will be compared with the efficiency of the previous model. If the new model is better than the previous model then the new model will be deployed. Spark Streaming Job will be restarted to pick the new model. Hence only Spark Streaming Job must be automated in a real scenario**

<https://pixipanda.com/>

Refer to this website for complete training content on Kafka, Spark, Spark SQL, Spark Streaming, Structured Streaming, Hive, Sqoop, HDFS, and MapReduce. Along with training content, I have also uploaded code and image for practice

# 1. Image and Code Setup

Download and Install Oracle VirtualBox

<https://www.virtualbox.org/wiki/Downloads>

Import Ubuntu image.

[http://bit.ly/udemy\\_ubuntu\\_image](http://bit.ly/udemy_ubuntu_image)

Username: hduser

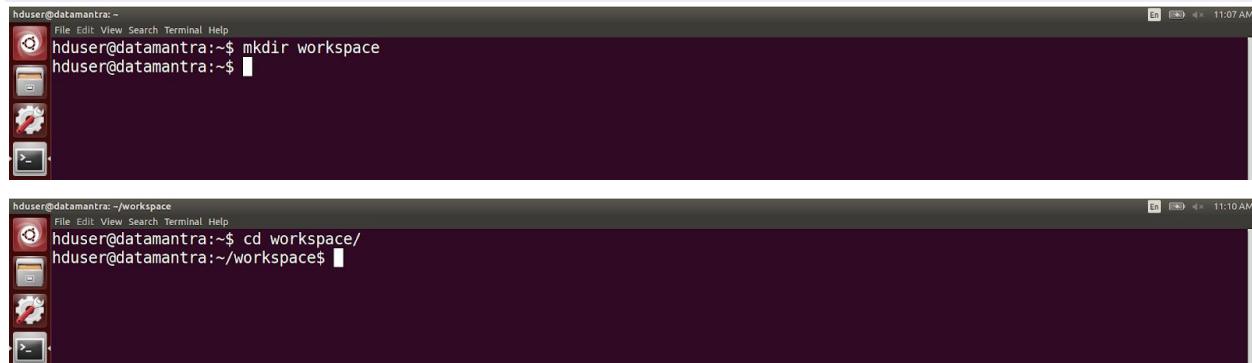
Password: hadoop123

## 1.1 Clone Code

### Create a workspace directory

Create a workspace directory in your home directory

```
mkdir workspace  
cd workspace
```



The image contains two vertically stacked screenshots of a terminal window. Both screenshots show a dark-themed terminal interface with a red header bar. The top screenshot shows the command 'mkdir workspace' being typed at the prompt 'hduser@datamantra:~\$'. The bottom screenshot shows the command 'cd workspace' being typed at the prompt 'hduser@datamantra:~/workspace\$'. Both screenshots include a dock with various icons at the bottom.

### Clone FraudDetection from GitHub

This is a branch code. It is a bit different from the code shown in the video.

Here we have VectorSlicer, StandardScaler and Confusion matrix.

```
git clone  
https://github.com/pramoddattamantra/FraudDetection/tree/FraudDetection\_ML
```

```
hduser@datamantra:~/workspace$ git clone https://github.com/pramoddatamantra/FraudDetection
Cloning into 'FraudDetection'...
remote: Enumerating objects: 504, done.
remote: Total 504 (delta 0), reused 0 (delta 0), pack-reused 504
Receiving objects: 100% (504/504), 808.39 KiB | 362.00 KiB/s, done.
Resolving deltas: 100% (199/199), done.
Checking connectivity... done.
hduser@datamantra:~/workspace$
```

## Clone Credit card producer from GitHub

```
git clone https://github.com/pramoddatamantra/CreditcardProducer
```

```
hduser@datamantra:~/workspace$ git clone https://github.com/pramoddatamantra/CreditcardProducer
Cloning into 'CreditcardProducer'...
remote: Enumerating objects: 96, done.
remote: Total 96 (delta 0), reused 0 (delta 0), pack-reused 96
Unpacking objects: 100% (96/96), done.
Checking connectivity... done.
hduser@datamantra:~/workspace$
```

## Clone Fraud-alert-dashboard

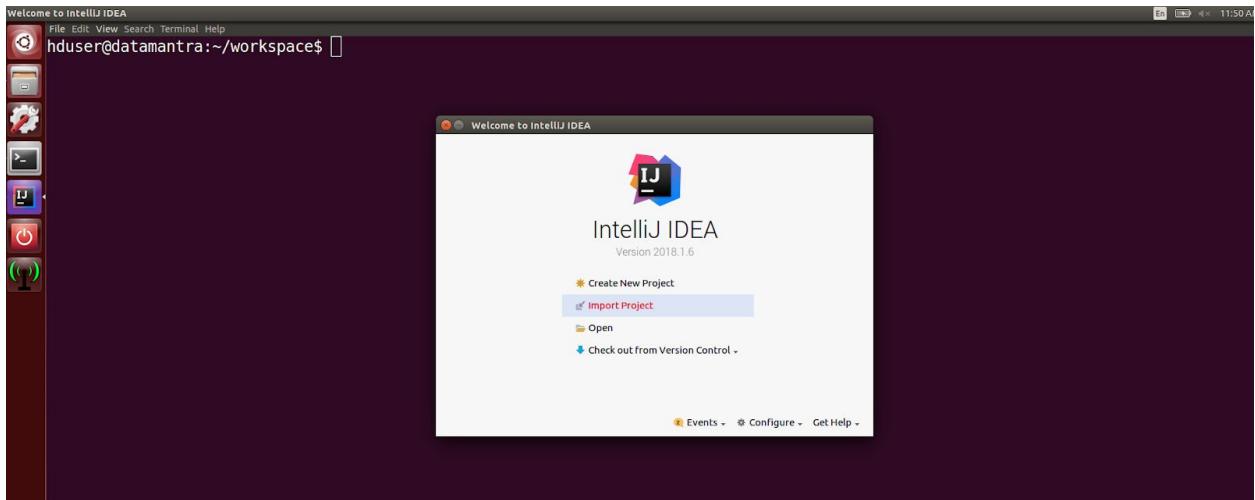
```
git clone https://github.com/pramoddatamantra/Fraud-alert-dashboard
```

```
hduser@datamantra:~/workspace$ git clone https://github.com/pramoddatamantra/Fraud-alert-dashboard
Cloning into 'Fraud-alert-dashboard'...
remote: Enumerating objects: 56, done.
remote: Total 56 (delta 0), reused 0 (delta 0), pack-reused 56
Unpacking objects: 100% (56/56), done.
Checking connectivity... done.
hduser@datamantra:~/workspace$
```

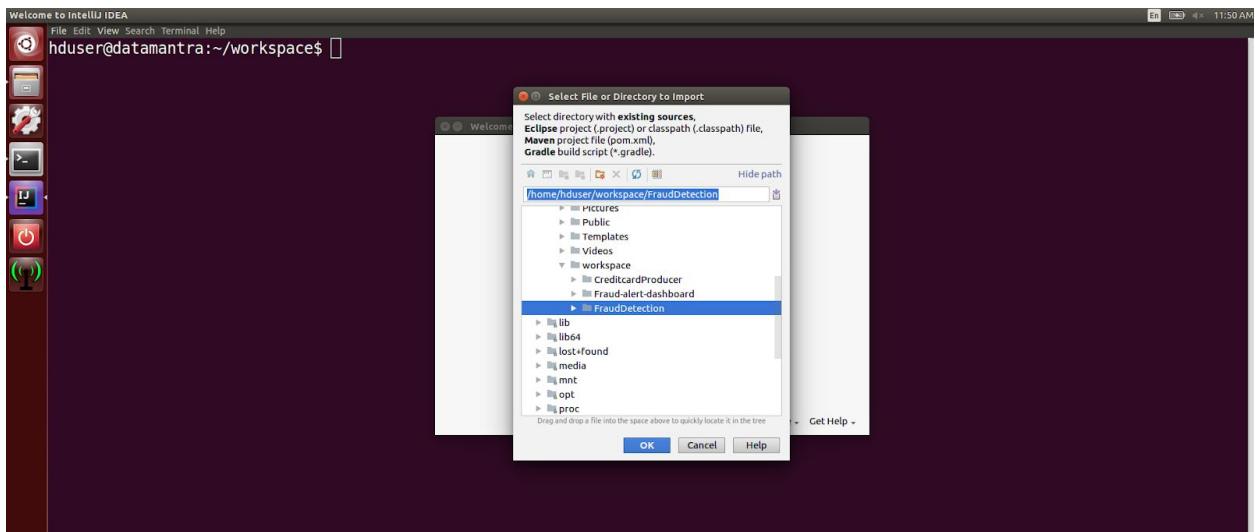
## 1.2 Import to IntelliJ

### Import FraudDetection to IntelliJ

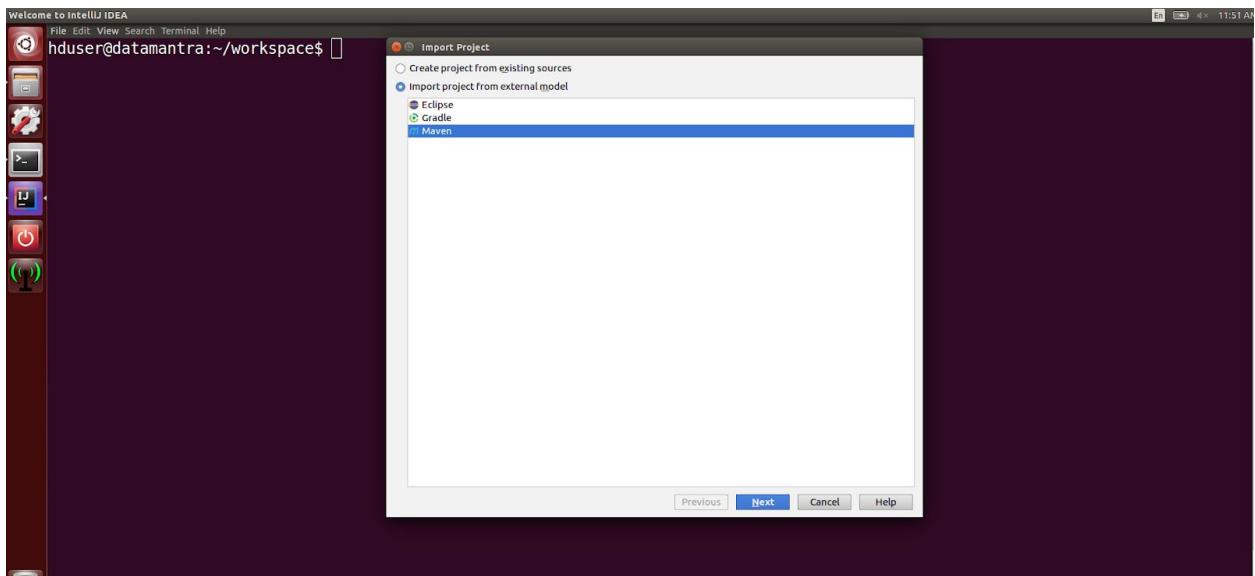
Click Import



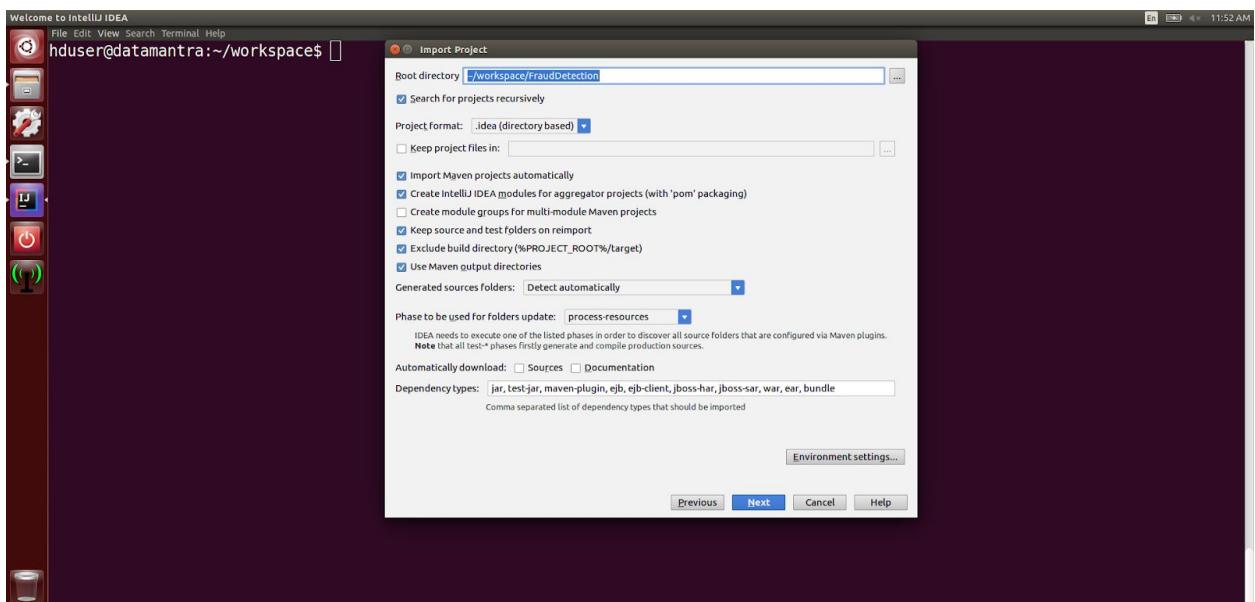
Select FraudDetection Project. Click OK. Click Next



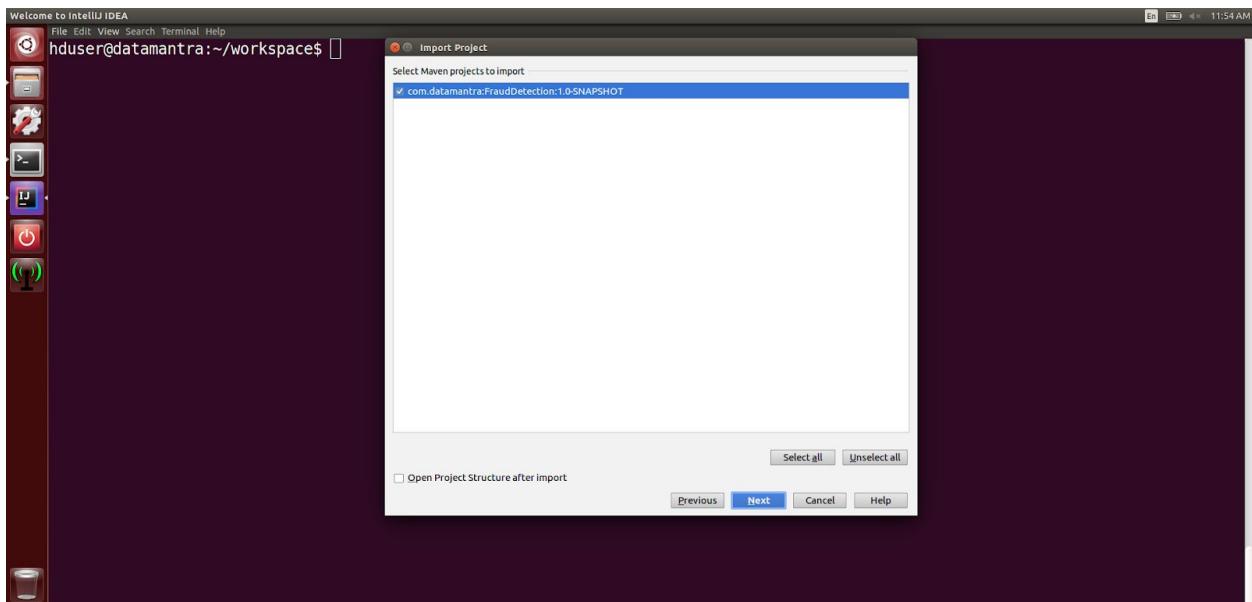
Select Maven. Click Next



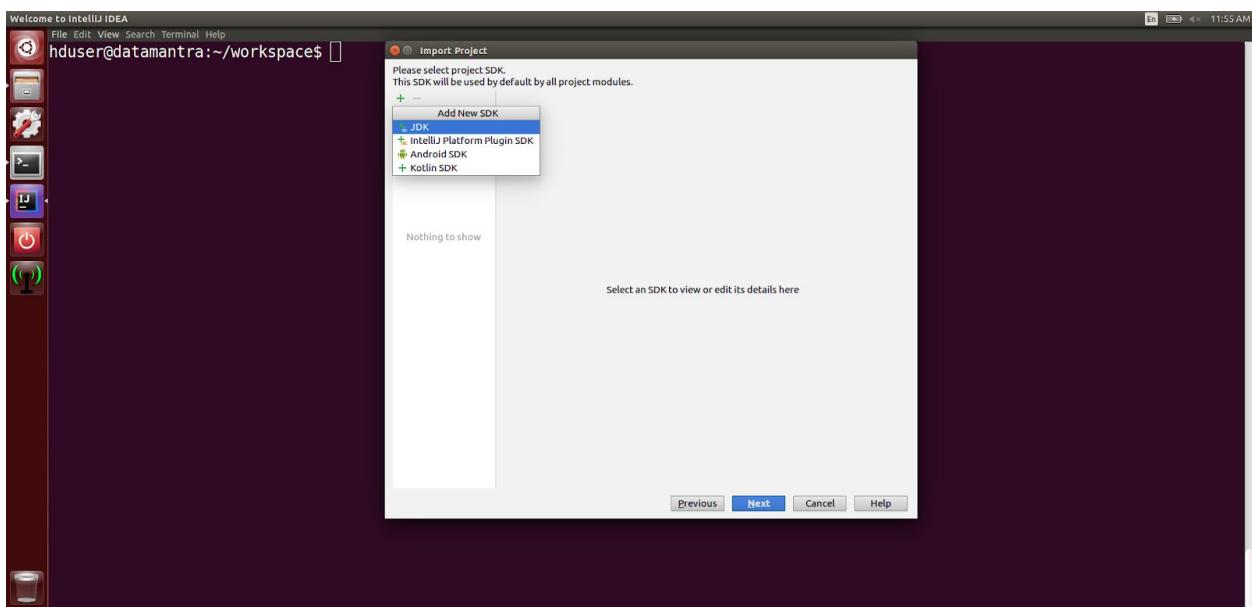
Checkmark “Search for projects recursively” and “Import Maven projects automatically”  
Click Next



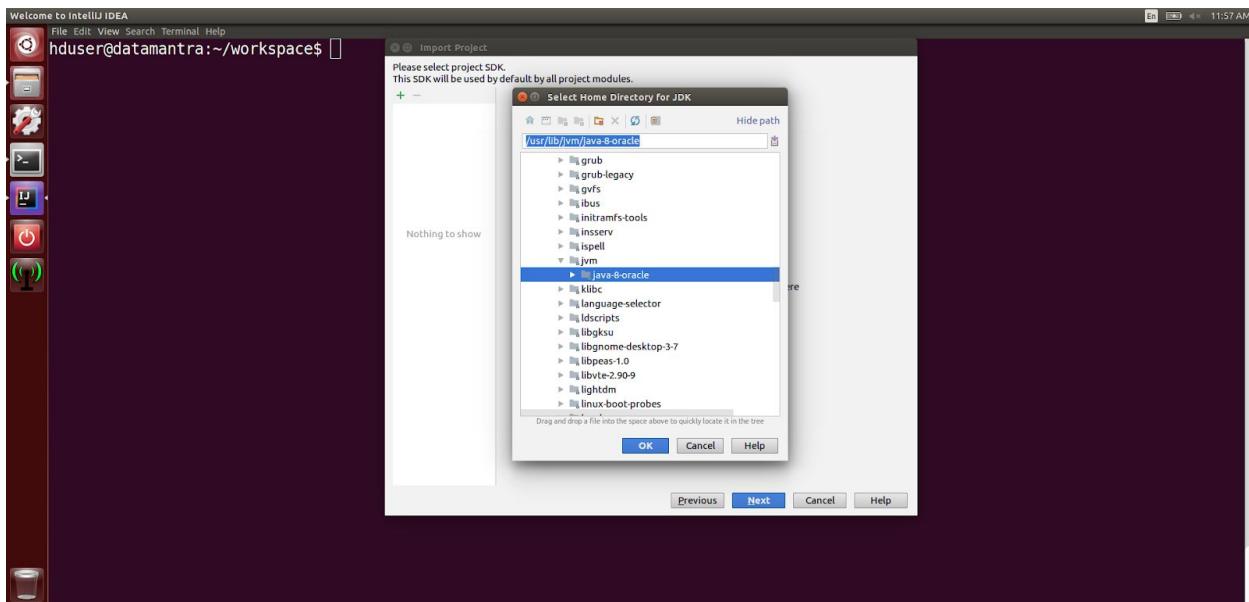
By default, the FraudDetection project will be selected. Click Next



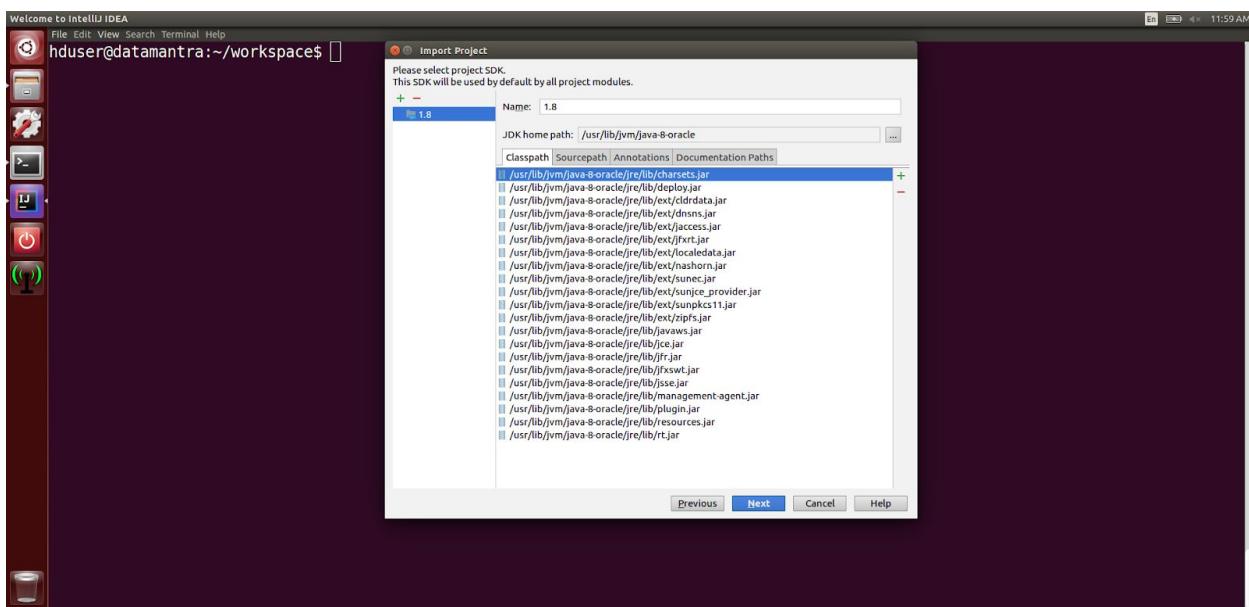
## Select JDK



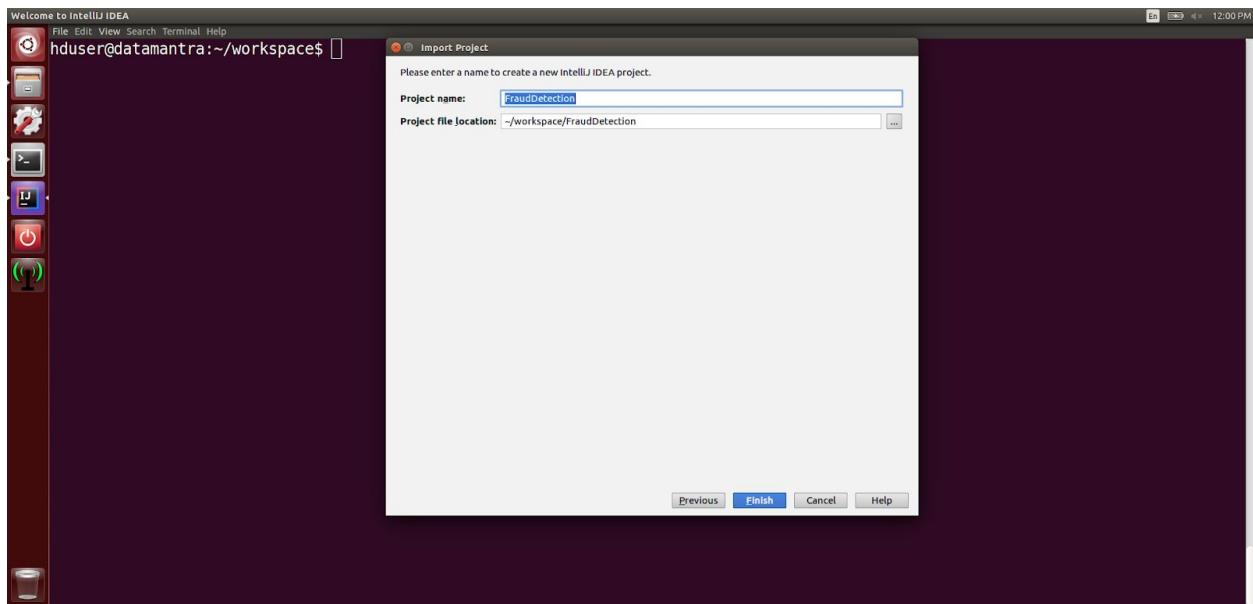
Select the java-8-oracle directory. Click OK and Click Next



Java 1.8 SDK is selected. Click Next



Let the project name and project path be as it is. Click Finish



Follow the above steps to import CreditcardProducer and Fraud-alert-dashboard project into IntelliJ

## 2. Initial Import Spark Job

### 2.1 Demonstration

#### Start Cassandra Server

```
Cassandra -f

hduser@datamantra:~$ cassandra -f
```

#### Connect to Cassandra Server

Connect to Cassandra Server using Cassandra Cqlsh client

```
cqlsh

hduser@datamantra:~$ cqlsh
Connected to Test Cluster at 127.0.0.1:9042.
[cqlsh 5.0.1 | Cassandra 3.11.1 | CQL spec 3.4.4 | Native protocol v4]
Use HELP for help.
cqlsh>
```

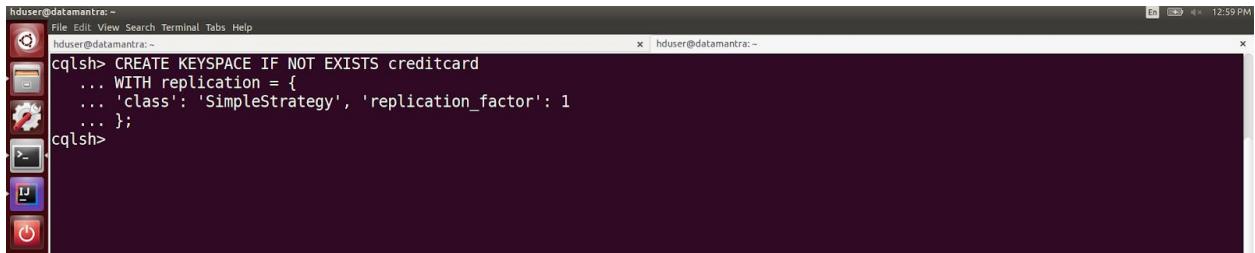
#### Create keyspace and tables

<https://github.com/pramoddatamantra/FraudDetection/blob/master/src/main/resources/cassandra/creditcard.cql>

Copy the contents of this file and paste it on cqlsh Cassandra client

Create creditcard keyspace

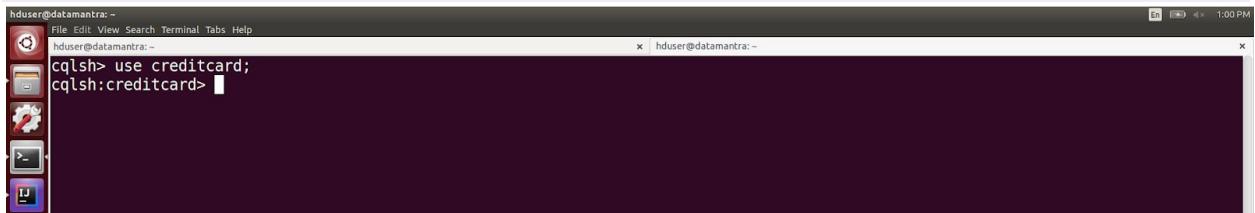
```
CREATE KEYSPACE IF NOT EXISTS creditcard
WITH replication = {
  'class': 'SimpleStrategy', 'replication_factor': 1
};
```



```
hduser@datamantra:~  
File Edit View Search Terminal Tabs Help  
hduser@datamantra:~  
cqlsh> CREATE KEYSPACE IF NOT EXISTS creditcard  
... WITH replication = {  
... 'class': 'SimpleStrategy', 'replication_factor': 1  
... };  
cqlsh>
```

## Use creditcard keyspace

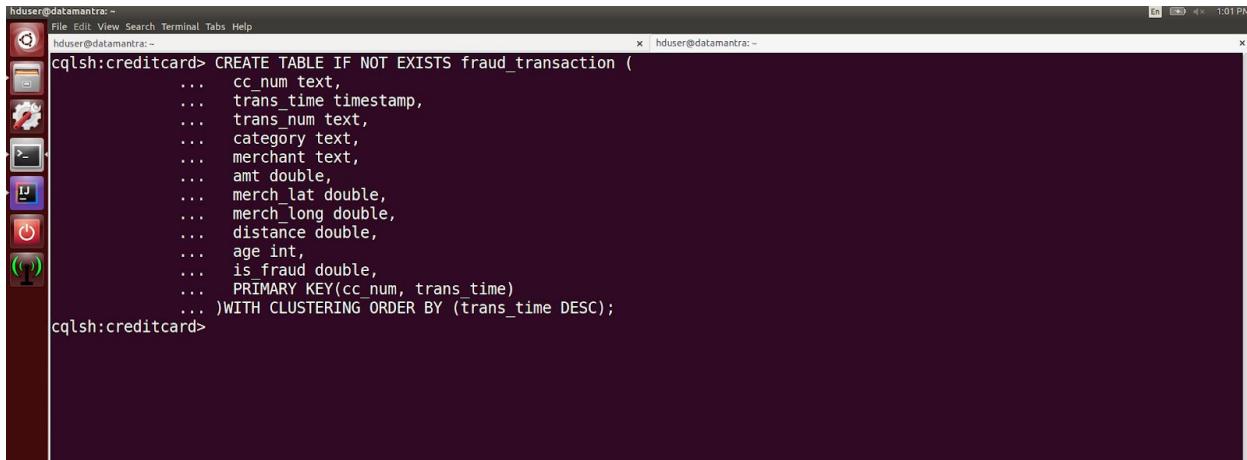
```
USE creditcard;
```



```
hduser@datamantra:~  
File Edit View Search Terminal Tabs Help  
hduser@datamantra:~  
cqlsh> use creditcard;  
cqlsh:creditcard> █
```

## Create fraud\_transaction table

```
CREATE TABLE IF NOT EXISTS fraud_transaction (  
    cc_num text,  
    trans_time timestamp,  
    trans_num text,  
    category text,  
    merchant text,  
    amt double,  
    merch_lat double,  
    merch_long double,  
    distance double,  
    age int,  
    is_fraud double,  
    PRIMARY KEY(cc_num, trans_time)  
)WITH CLUSTERING ORDER BY (trans_time DESC);
```



```
hduser@datamantra:~
```

```
File Edit View Search Terminal Tabs Help
```

```
hduser@datamantra:~
```

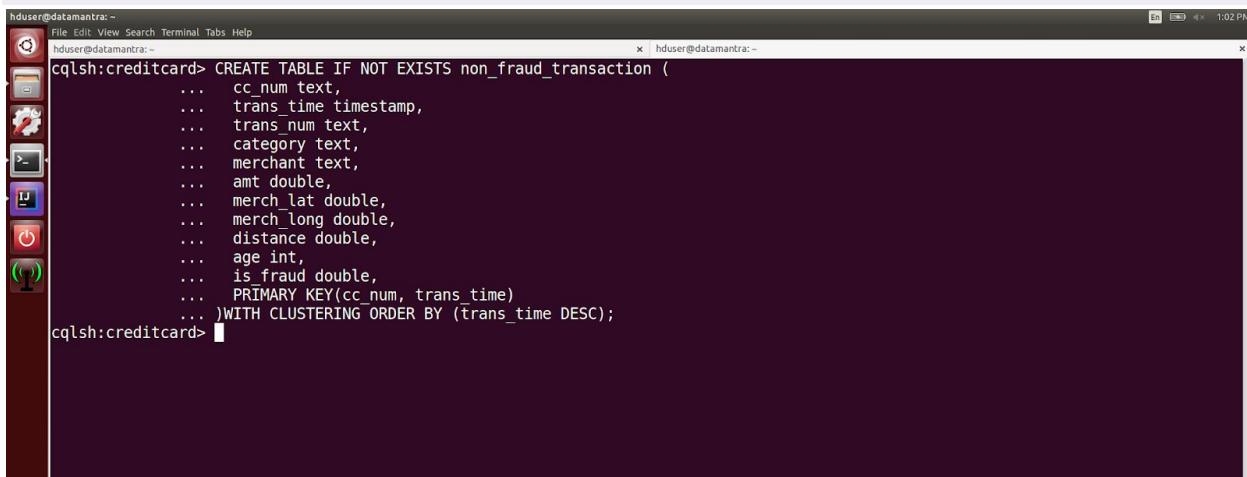
```
x | hduser@datamantra:~
```

```
cqlsh:creditcard> CREATE TABLE IF NOT EXISTS fraud_transaction (
...     cc_num text,
...     trans_time timestamp,
...     trans_num text,
...     category text,
...     merchant text,
...     amt double,
...     merch_lat double,
...     merch_long double,
...     distance double,
...     age int,
...     is_fraud double,
...     PRIMARY KEY(cc_num, trans_time)
... )WITH CLUSTERING ORDER BY (trans_time DESC);
```

```
cqlsh:creditcard>
```

## Create not\_fraud\_transaction table

```
CREATE TABLE IF NOT EXISTS non_fraud_transaction (
    cc_num text,
    trans_time timestamp,
    trans_num text,
    category text,
    merchant text,
    amt double,
    merch_lat double,
    merch_long double,
    distance double,
    age int,
    is_fraud double,
    PRIMARY KEY(cc_num, trans_time)
)WITH CLUSTERING ORDER BY (trans_time DESC);
```



```
hduser@datamantra:~
```

```
File Edit View Search Terminal Tabs Help
```

```
hduser@datamantra:~
```

```
x | hduser@datamantra:~
```

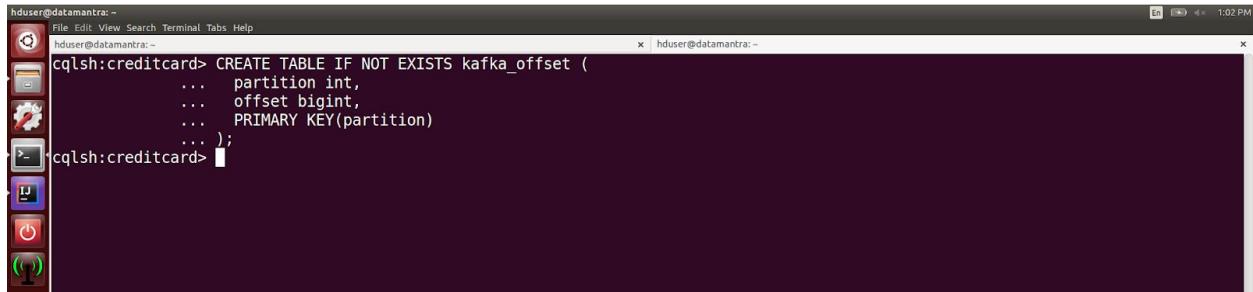
```
cqlsh:creditcard> CREATE TABLE IF NOT EXISTS non_fraud_transaction (
...     cc_num text,
...     trans_time timestamp,
...     trans_num text,
...     category text,
...     merchant text,
...     amt double,
...     merch_lat double,
...     merch_long double,
...     distance double,
...     age int,
...     is_fraud double,
...     PRIMARY KEY(cc_num, trans_time)
... )WITH CLUSTERING ORDER BY (trans_time DESC);
```

```
cqlsh:creditcard>
```

## Create a kafka\_offset table

Offset per partition will be saved in this table

```
CREATE TABLE IF NOT EXISTS kafka_offset (
    partition int,
    offset bigint,
    PRIMARY KEY(partition)
);
```



The screenshot shows a terminal window titled 'hduser@datamantra:~'. The user is in the 'creditcard' keyspace, as indicated by the prompt 'cqlsh:creditcard>'. They are running the command 'CREATE TABLE IF NOT EXISTS kafka\_offset (...)'. The table definition includes columns for 'partition' (int) and 'offset' (bigint), with 'partition' as the primary key. The command is partially typed, with the closing ')' character highlighted.

## Create a customer table

This table will have customer details

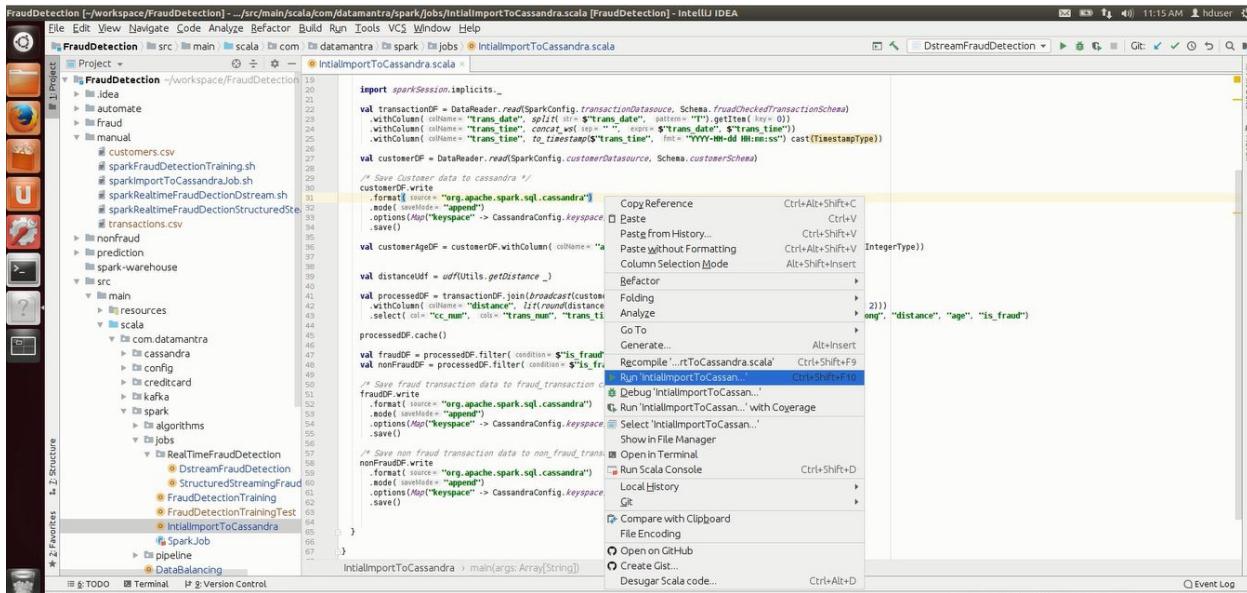
```
CREATE TABLE IF NOT EXISTS customer (
    cc_num text,
    first text,
    last text,
    gender text,
    street text,
    city text,
    state text,
    zip text,
    lat double,
    long double,
    job text,
    dob timestamp,
    PRIMARY KEY(cc_num));
```

```

hduser@datamantra:~ 
File Edit View Search Terminal Tabs Help 
hduser@datamantra:~ 
cqlsh:creditcard> CREATE TABLE IF NOT EXISTS customer (
...     cc_num text,
...     first text,
...     last text,
...     gender text,
...     street text,
...     city text,
...     state text,
...     zip text,
...     lat double,
...     long double,
...     job text,
...     dob timestamp,
...     PRIMARY KEY(cc_num)
... );
cqlsh:creditcard>

```

## Run IntialImportToCassandra.scala from IntelliJ No Input parameter



## Output

Transaction data and customer data must be saved in the corresponding table in Cassandra

```
SELECT * FROM customer limit 5
```

```
hduser@datamantra:~
```

```
cqlsh:creditcard> SELECT * FROM customer limit 5;
```

cc_num	city	dob	first	gender	job	last	lat
long	state	street		zip			
180036251237802	Salt Lick	1956-07-19 18:30:00.000000+0000	Melissa	F	Psychologist, forensic	James	38.
104	-83.6316	KY	537 Bryant Mall	40371			
676165681542	Salem	1954-07-03 18:30:00.000000+0000	John	M	Human resources officer	Garcia	44.9
039	-123.0445	OR	34153 Maria Mountain	97302			
30157941709315	Caledonia	1974-11-03 18:30:00.000000+0000	Maurice	M	Hydrographic surveyor	Simon	43.6
221	-91.4837	MN	031 Jessica Harbor Suite 104	55921			
180094108369013	Zephyr Cove	1949-12-28 18:30:00.000000+0000	John	M	Geophysical data processor	Holland	39.0
204	-119.9114	NV	630 Christina Harbor	89448			
4048725581466255	Chicago	1950-12-27 18:30:00.000000+0000	Theresa	F	Retail banker	Cole	41.8
858	-87.6181	IL	431 Walker Via	60601			

```
(5 rows)
cqlsh:creditcard>
```

```
SELECT * FROM fraud_transaction limit 5;
```

```
hduser@datamantra:~
```

```
cqlsh:creditcard> SELECT * FROM fraud_transaction limit 5;
```

cc_num	trans_time	age	amt	category	distance	is_fraud	merch_lat	merch_long	merchant	trans_num
180036251237802	2012-01-01 17:54:16.000000+0000	63	2168	shopping_net	157.66	1	37.13679	-84.94059	Fisher Inc	f5961217ad20bcb888558e451c0555b8
180036251237802	2012-01-01 17:41:52.000000+0000	63	2583	shopping_net	59.92	1	38.4867	-84.11508	Stamm-Witting	122b29e9fd69ed43223ad7a56cf38881
180036251237802	2012-01-01 17:28:32.000000+0000	63	262	gas_transport	80.43	1	38.76958	-83.27012	Streich, Hansen and Veun	f5ebbe07eb6fbcf3f194cd24f4b9bf8
180036251237802	2012-01-01 17:07:28.000000+0000	63	2361	grocery_pos	118.8	1	39.13225	-84.00291	Mosciski, Gislason and Mertz	f95517fbc6167e00f6adfb3c6c1fd03
180036251237802	2012-01-01 16:38:01.000000+0000	63	1723	grocery_pos	232.03	1	36.42305	-82.07787	Barton Inc	e8b482a3eb37655c0ce434ca49798458

```
(5 rows)
cqlsh:creditcard>
```

```
SELECT * FROM fraud_transaction limit 5;
```

All the data is successfully imported to Cassandra Keyspace

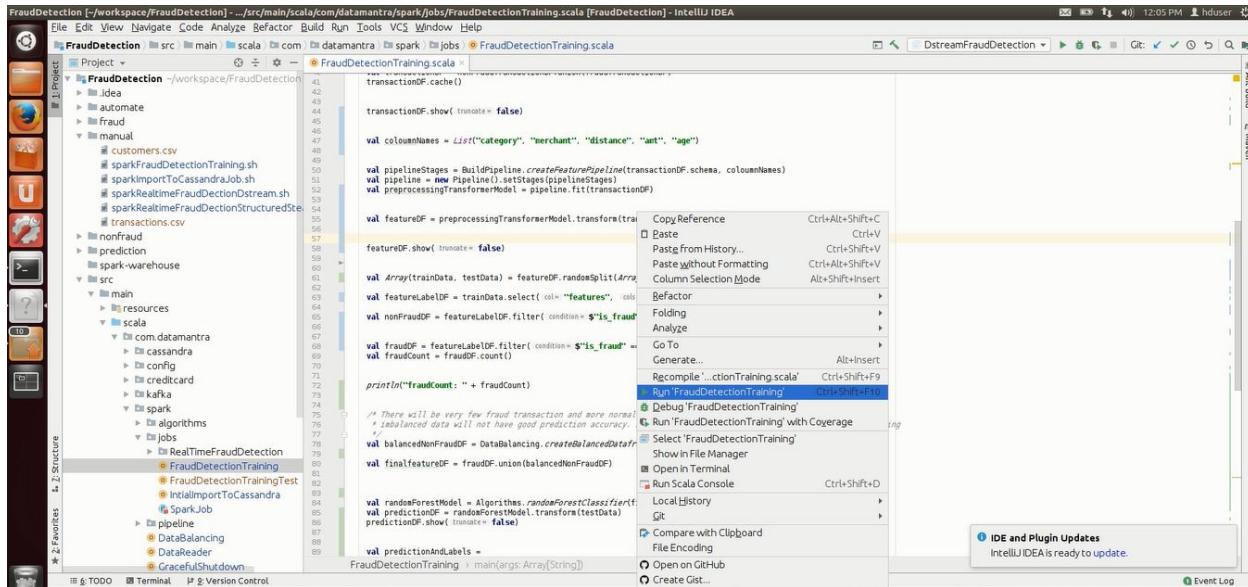
# 3. Spark Training Job

## 3.1 Demonstration

Run FraudDetectionTraining.scala from IntelliJ

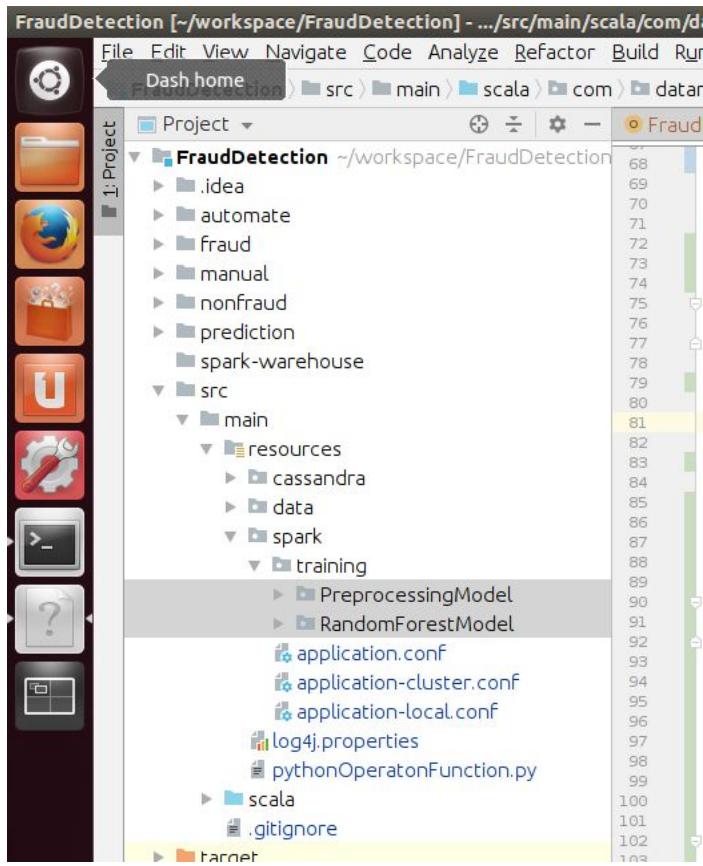
No Input Parameters

Right-click on FraudDetectionTraining.scala file and Click Run



## Output

```
===== Confusion matrix =====
#####| Actual = 1 | Actual = 0 |
-----+-----+
Predicted = 1| 91.000000 | 83.000000 |
Predicted = 0| 0.000000 | 2548.000000 |
-----+
True Positive Rate: 1.0
False Positive Rate: 0.03154694
Precision: 0.5229885
```



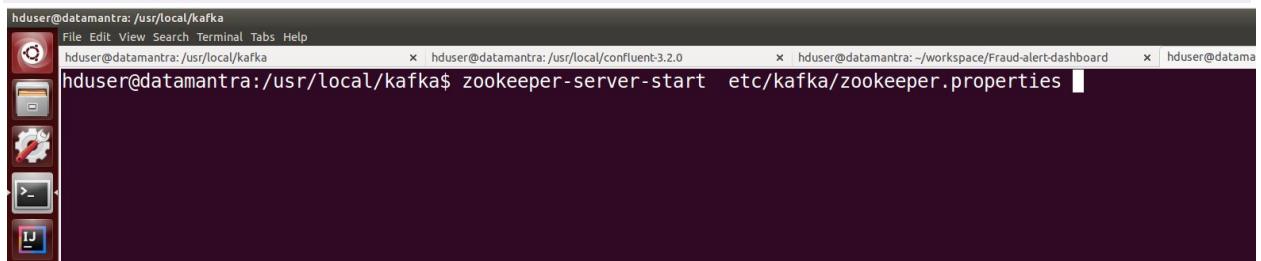
Preprocessing Model and Random Forest Model is saved to the File System

# 4. Spark Streaming

## 4.1 Demonstration

### Start Zookeeper

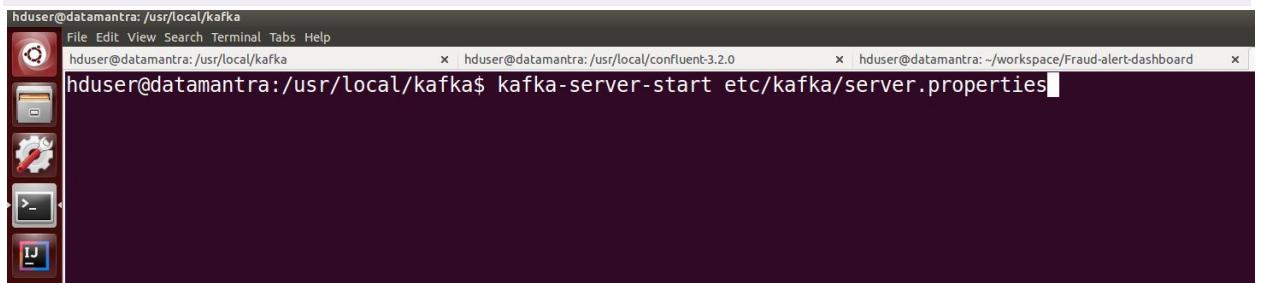
```
cd /usr/local/kafka  
zookeeper-server-start etc/kafka/zookeeper.properties
```



A screenshot of a terminal window titled "hduser@datamantra: /usr/local/kafka". The window contains three tabs: "hduser@datamantra:/usr/local/kafka", "hduser@datamantra:/usr/local/confluent-3.2.0", and "hduser@datamantra:~/workspace/Fraud-alert-dashboard". The active tab shows the command "zookeeper-server-start etc/kafka/zookeeper.properties" being typed.

### Start Kafka Server

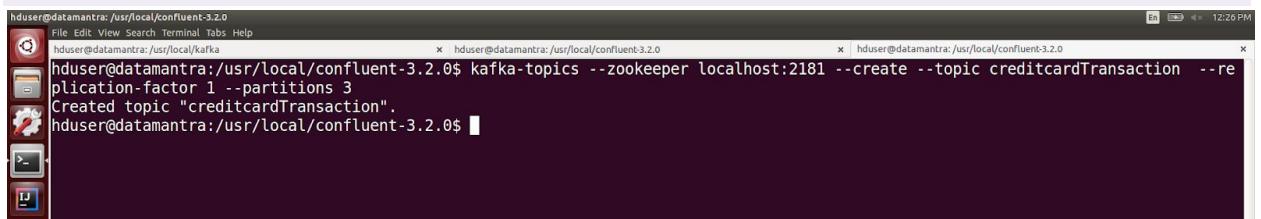
```
kafka-server-start etc/kafka/server.properties
```



A screenshot of a terminal window titled "hduser@datamantra: /usr/local/kafka". The window contains three tabs: "hduser@datamantra:/usr/local/kafka", "hduser@datamantra:/usr/local/confluent-3.2.0", and "hduser@datamantra:~/workspace/Fraud-alert-dashboard". The active tab shows the command "kafka-server-start etc/kafka/server.properties" being typed.

### Create transaction topic

```
kafka-topics --zookeeper localhost:2181 --create --topic creditcardTransaction --replication-factor 1 --partitions 3
```

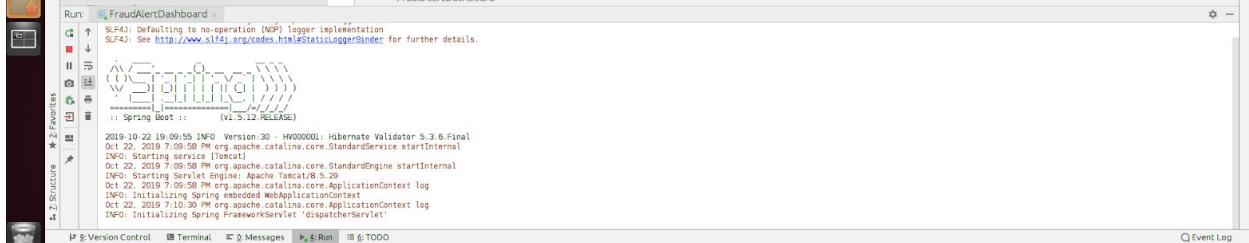
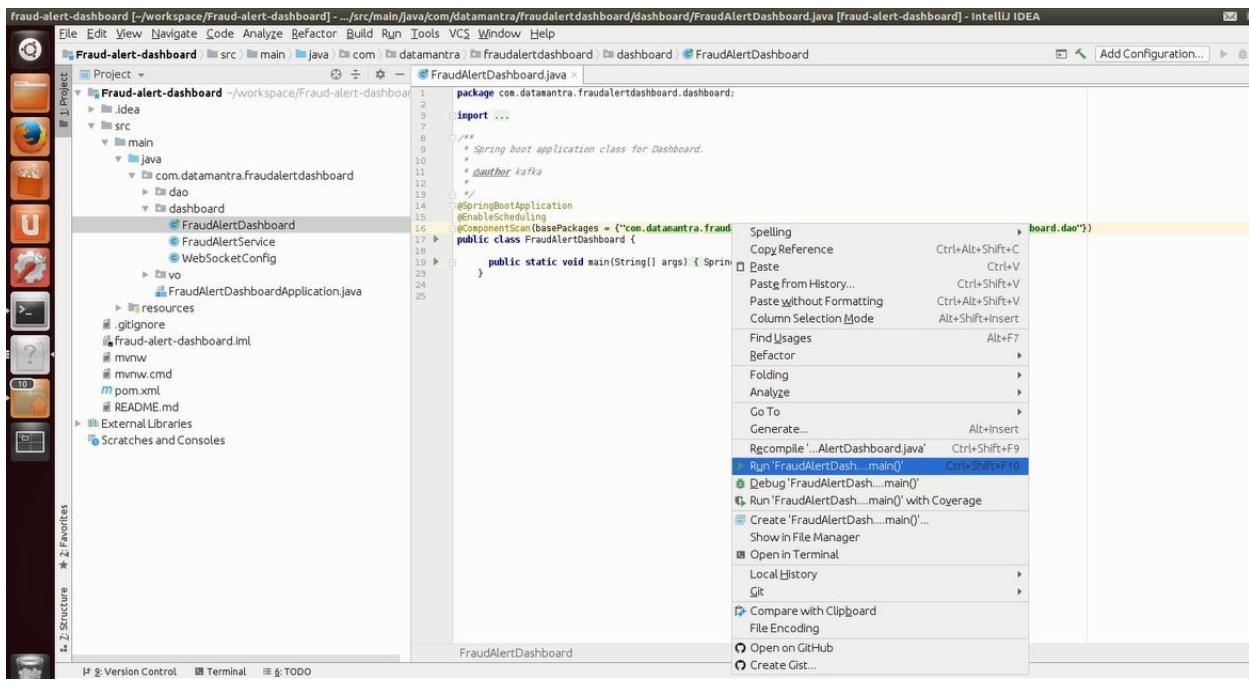


A screenshot of a terminal window titled "hduser@datamantra: /usr/local/confluent-3.2.0". The window contains three tabs: "hduser@datamantra:/usr/local/kafka", "hduser@datamantra:/usr/local/confluent-3.2.0", and "hduser@datamantra:/usr/local/confluent-3.2.0". The active tab shows the command "kafka-topics --zookeeper localhost:2181 --create --topic creditcardTransaction --replication-factor 1 --partitions 3" being typed. The output shows the topic "creditcardTransaction" was created.

## Run Fraud-alert-dashboard project from IntelliJ

No input parameters

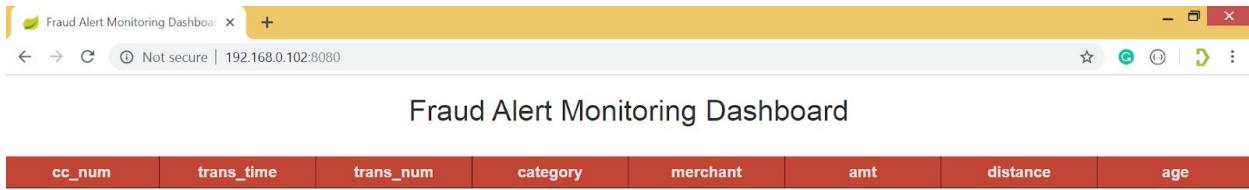
Right Click on FraudAlertDashboard.java file and Run



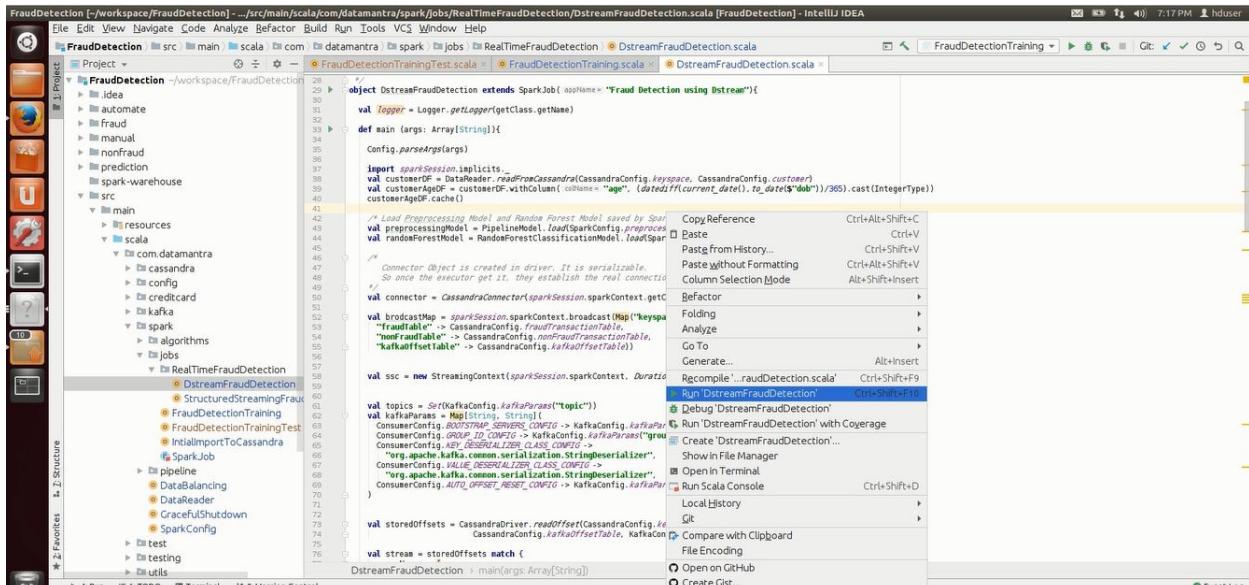
## Output

The fraud alert dashboard web server is up and running. Verify this by accessing through the web browser

<http://<ubuntu-ipaddr>:8080>



## Run Spark Streaming Job from IntelliJ

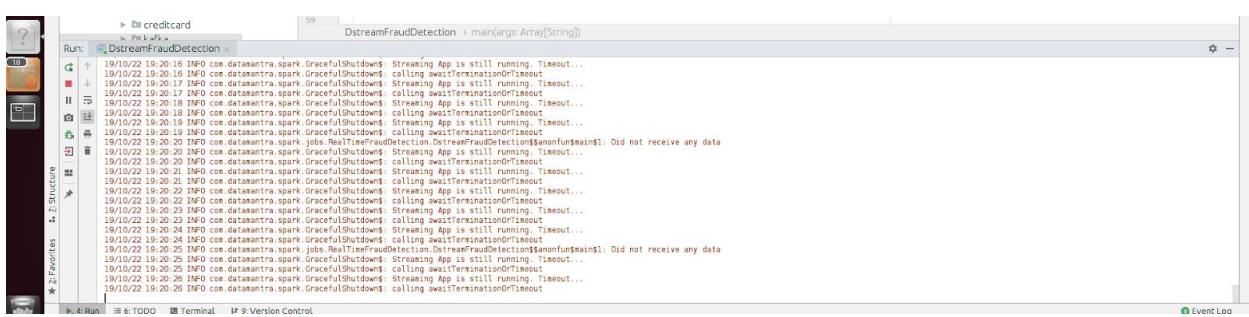


The screenshot shows the IntelliJ IDEA interface with the 'FraudDetection' project open. A context menu is displayed over the code editor for the file 'DstreamFraudDetection.scala'. The 'Run' option is highlighted in blue, indicating it is the selected action.

```

object DstreamFraudDetection extends SparkJob(appName = "Fraud Detection using Dstream"){
  ...
}
  
```

Run 'DstreamFraudDetection' is the selected option in the context menu.



The Run tool window displays the log output for the 'DstreamFraudDetection' job. The log entries show the application is still running and awaiting termination:

```

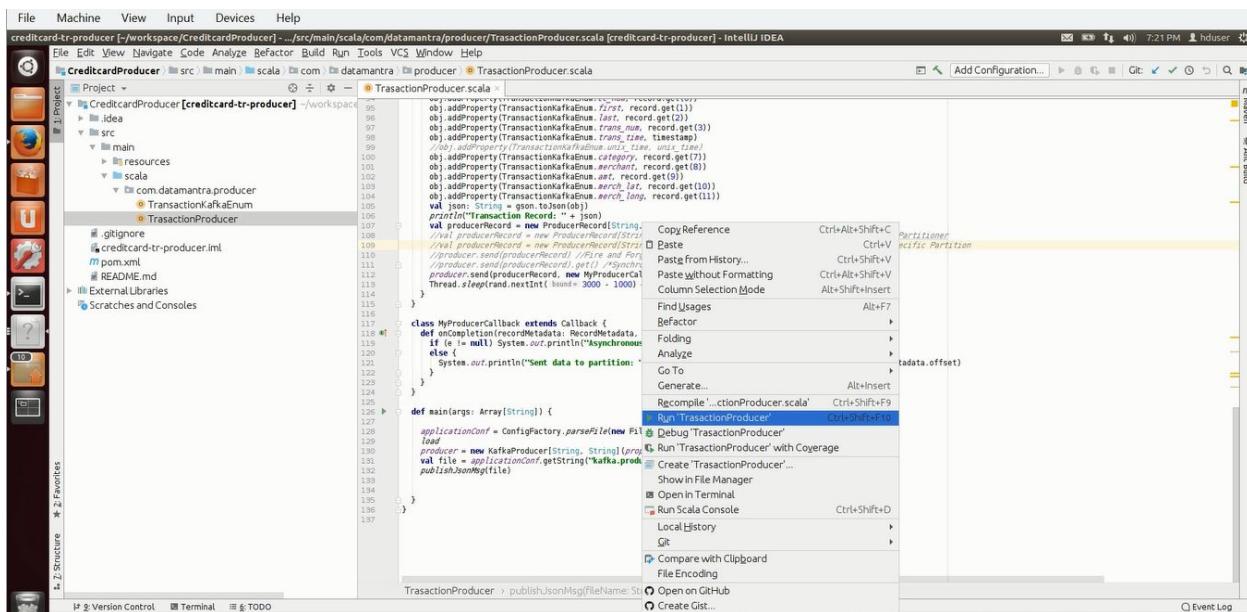
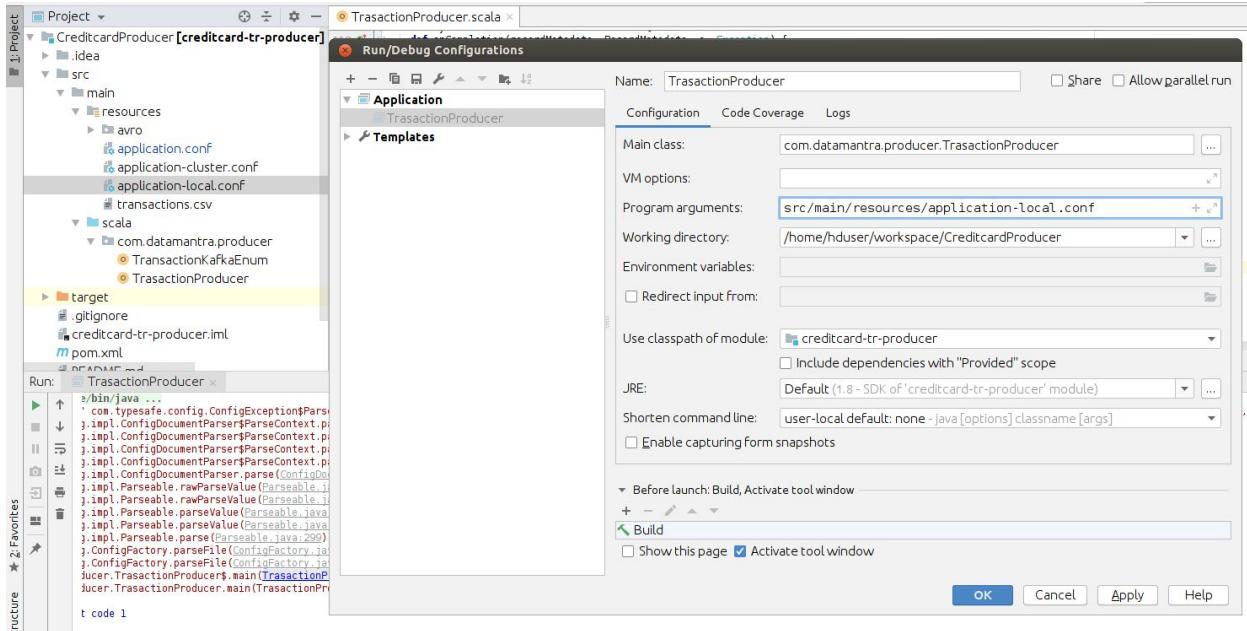
19/10/22 19:20:16 INFO com.datamantra.spark.GracefulShutdown: Streaming App is still running. Timeout...
19/10/22 19:20:16 INFO com.datamantra.spark.GracefulShutdown: calling awaitTerminationOrTimeout...
19/10/22 19:20:17 INFO com.datamantra.spark.GracefulShutdown: Streaming App is still running. Timeout...
19/10/22 19:20:18 INFO com.datamantra.spark.GracefulShutdown: Streaming App is still running. Timeout...
19/10/22 19:20:18 INFO com.datamantra.spark.GracefulShutdown: calling awaitTerminationOrTimeout...
19/10/22 19:20:19 INFO com.datamantra.spark.GracefulShutdown: Streaming App is still running. Timeout...
19/10/22 19:20:19 INFO com.datamantra.spark.GracefulShutdown: calling awaitTerminationOrTimeout...
19/10/22 19:20:20 INFO com.datamantra.spark.GracefulShutdown: Streaming App is still running. Timeout...
19/10/22 19:20:20 INFO com.datamantra.spark.GracefulShutdown: calling awaitTerminationOrTimeout...
19/10/22 19:20:21 INFO com.datamantra.spark.GracefulShutdown: Streaming App is still running. Timeout...
19/10/22 19:20:22 INFO com.datamantra.spark.GracefulShutdown: Streaming App is still running. Timeout...
19/10/22 19:20:22 INFO com.datamantra.spark.GracefulShutdown: calling awaitTerminationOrTimeout...
19/10/22 19:20:23 INFO com.datamantra.spark.GracefulShutdown: Streaming App is still running. Timeout...
19/10/22 19:20:23 INFO com.datamantra.spark.GracefulShutdown: calling awaitTerminationOrTimeout...
19/10/22 19:20:24 INFO com.datamantra.spark.GracefulShutdown: Streaming App is still running. Timeout...
19/10/22 19:20:24 INFO com.datamantra.spark.GracefulShutdown: calling awaitTerminationOrTimeout...
19/10/22 19:20:25 INFO com.datamantra.spark.GracefulShutdown: Streaming App is still running. Timeout...
19/10/22 19:20:25 INFO com.datamantra.spark.GracefulShutdown: calling awaitTerminationOrTimeout...
19/10/22 19:20:26 INFO com.datamantra.spark.GracefulShutdown: Streaming App is still running. Timeout...
19/10/22 19:20:26 INFO com.datamantra.spark.GracefulShutdown: calling awaitTerminationOrTimeout...
  
```

## Run CreditcardProducer from IntelliJJ

<https://github.com/pramoddatamantra/CreditcardProducer>

Input parameter is the configuration file

```
src/main/resources/application-local.conf
```



## Output

Whenever Spark Streaming Job predicts a fraud transaction it will be displayed on Fraud Alert Dashboard



Fraud Alert Monitoring Dashboard

cc_num	trans_time	trans_num	category	merchant	amt	distance	age
374115112731710	2019-10-22 19:30:50	11efb54624f40fe0...	entertainment	Johns-Hoeger	64	4.16	41

# 5. Airflow Automation

## 5.1 Configure Airflow

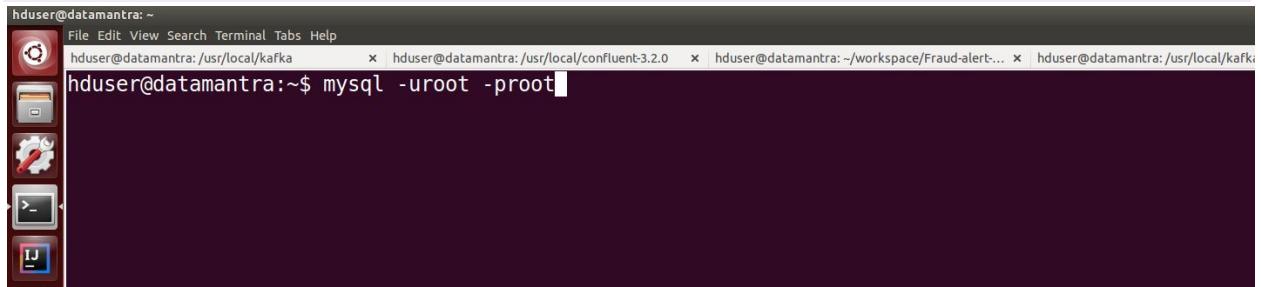
By default, apache airflow is configured with SQLite. But SQLite is not suitable for the Production environment. Hence modify configure airflow to use MySQL database

### 5.1.1 Create hduser and airflow database

Create a new user and database in Mysql

Login to Mysql with root credentials

```
mysql -uroot -proot
```



```
hduser@datamantra:~
```

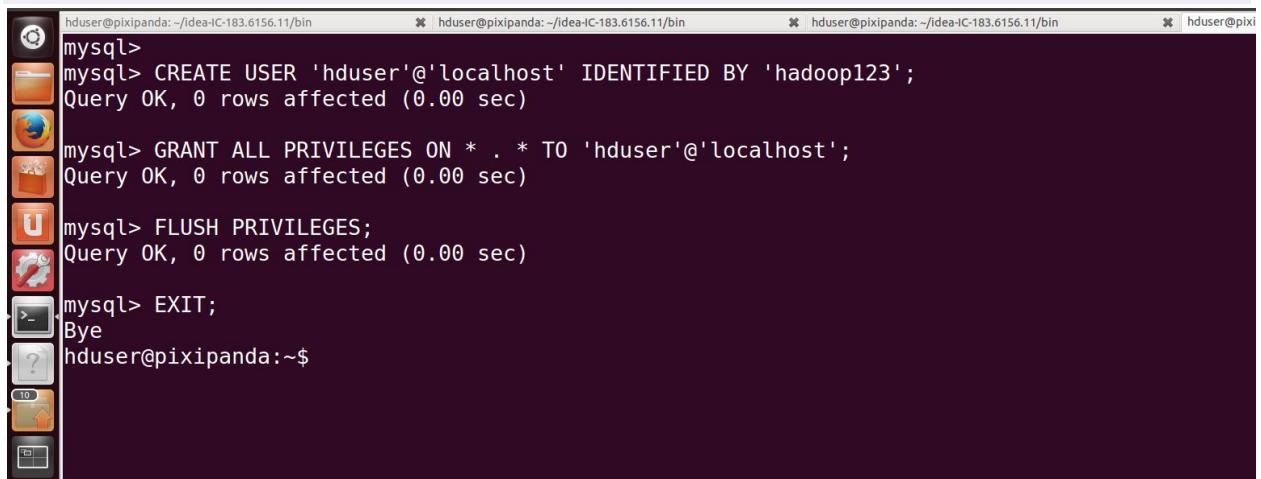
```
File Edit View Search Terminal Tabs Help
```

```
hduser@datamantra:/usr/local/kafka x hduser@datamantra:/usr/local/confluent-3.2.0 x hduser@datamantra:~/workspace/Fraud-alert... x hduser@datamantra:/usr/local/kafka
```

```
hduser@datamantra:~$ mysql -uroot -proot
```

### Create hduser with hadoop123 as password

```
CREATE USER 'hduser'@'localhost' IDENTIFIED BY 'hadoop123';
GRANT ALL PRIVILEGES ON * . * TO 'hduser'@'localhost';
FLUSH PRIVILEGES;
EXIT;
```



```
hduser@pixipanda:~/idea-IC-183.6156.11/bin
```

```
hduser@pixipanda:~/idea-IC-183.6156.11/bin
```

```
hduser@pixipanda:~/idea-IC-183.6156.11/bin
```

```
hduser@pixipanda:~/idea-IC-183.6156.11/bin
```

```
mysql>
mysql> CREATE USER 'hduser'@'localhost' IDENTIFIED BY 'hadoop123';
Query OK, 0 rows affected (0.00 sec)

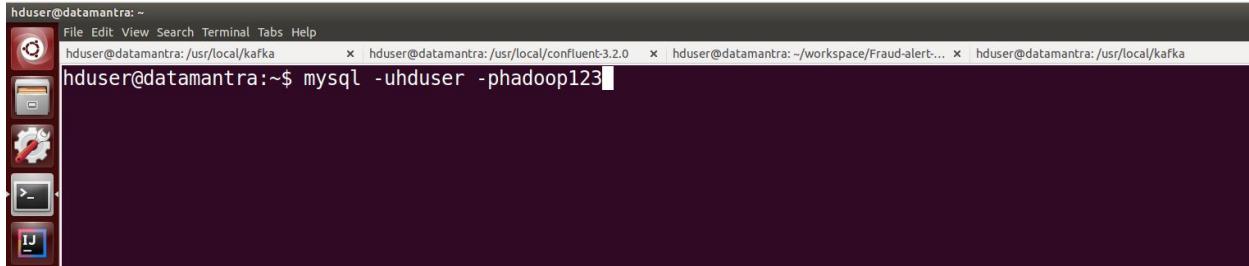
mysql> GRANT ALL PRIVILEGES ON * . * TO 'hduser'@'localhost';
Query OK, 0 rows affected (0.00 sec)

mysql> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.00 sec)

mysql> EXIT;
Bye
hduser@pixipanda:~$
```

## Login as hduser

```
mysql -uhduser -phadoop123
```



A screenshot of a terminal window titled "hduser@datamantra: ~". The window has a dark background and a light gray header bar. In the header bar, there are several tabs: "File Edit View Search Terminal Tabs Help", "hduser@datamantra:/usr/local/kafka", "hduser@datamantra:/usr/local/confluent-3.2.0", "hduser@datamantra:~/workspace/Fraud-alert...", and "hduser@datamantra:/usr/local/kafka". The main area of the terminal shows the command "mysql -uhduser -phadoop123" being typed.

## Create Airflow database and exit

```
CREATE DATABASE airflow;  
EXIT;
```

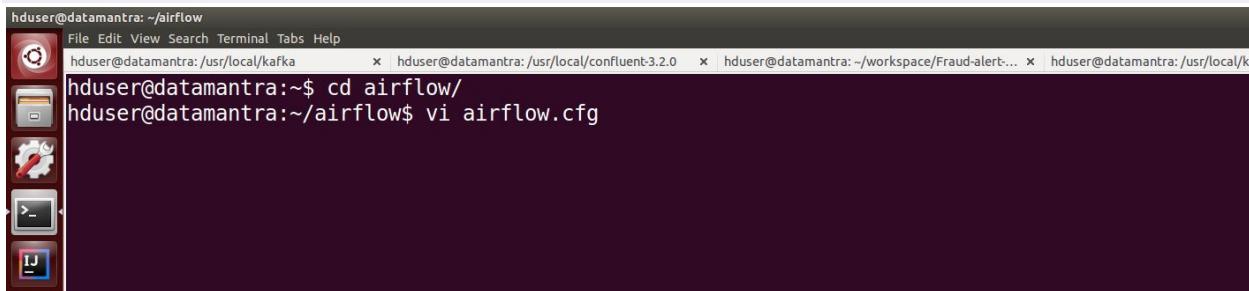


A screenshot of a terminal window titled "hduser@pixipanda: ~". The window has a dark background and a light gray header bar. In the header bar, there are three tabs: "File Edit View Search Terminal Tabs Help", "hduser@pixipanda:~/idea-IC-183.6156.11/bin", and "hduser@pixipanda:~/idea-IC-183.6156.11/bin". The main area of the terminal shows the MySQL command "CREATE DATABASE airflow;" followed by "Query OK, 1 row affected (0.00 sec)". Then, the command "mysql> exit;" is run, resulting in "Bye". The prompt "hduser@pixipanda:~\$" is shown at the bottom.

### 5.1.2 Modify airflow.cfg

Modify the airflow.cfg file to use MySQL database

```
cd ~/airflow  
vi airflow.cfg
```



A screenshot of a terminal window titled "hduser@datamantra:~/airflow". The window has a dark background and a light gray header bar. In the header bar, there are several tabs: "File Edit View Search Terminal Tabs Help", "hduser@datamantra:/usr/local/kafka", "hduser@datamantra:/usr/local/confluent-3.2.0", "hduser@datamantra:~/workspace/Fraud-alert...", and "hduser@datamantra:/usr/local/kafka". The main area of the terminal shows the commands "cd airflow/" and "vi airflow.cfg" being run sequentially.

Modify following fields, save and exit

```
sql_alchemy_conn = mysql://hduser:hadoop123@localhost:3306/airflow
load_examples = False
```

Here we are configuring MySQL instead of SQLite.

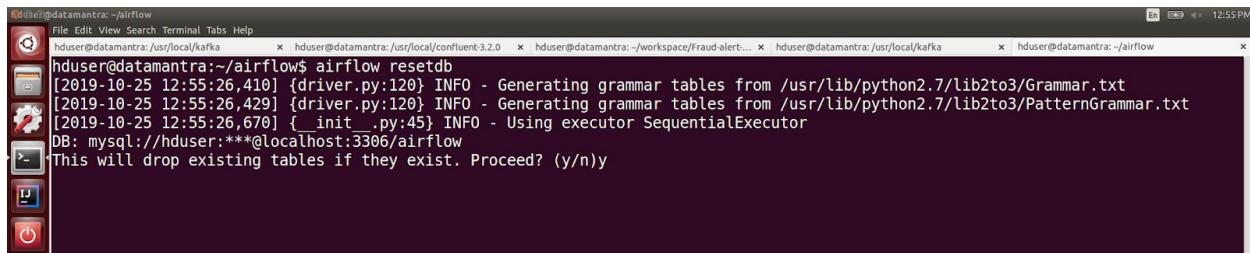
Also instead of loading all the built-in example scripts of Airflow, only our script will be loaded.

```
# The SQLAlchemy connection string to the metadata database.
# SQLAlchemy supports many different database engine, more information
# their website
#sql_alchemy_conn = sqlite:///home/hduser/airflow/airflow.db
sql_alchemy_conn = mysql://hduser:hadoop123@localhost:3306/airflow

# Whether to load the examples that ship with Airflow. It's good to
# get started, but you probably want to set this to False in a production
# environment
load_examples = False
```

## Rest Airflow Database

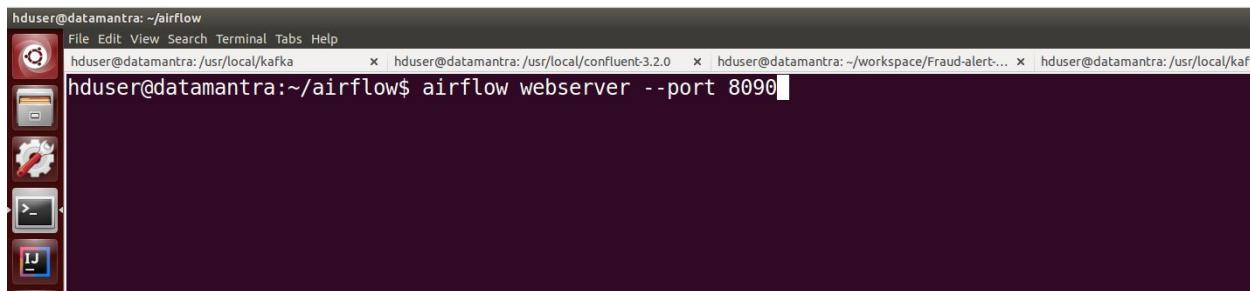
```
airflow resetdb
```



```
hduser@datamantra:~/airflow
File Edit View Search Terminal Tabs Help
hduser@datamantra:/usr/local/kafka x hduser@datamantra:/usr/local/confluent-3.2.0 x hduser@datamantra:~/workspace/Fraud-alert... x hduser@datamantra:/usr/local/kafka x hduser@datamantra:~/airflow
[2019-10-25 12:55:26,410] {driver.py:120} INFO - Generating grammar tables from /usr/lib/python2.7/lib2to3/Grammar.txt
[2019-10-25 12:55:26,429] {driver.py:120} INFO - Generating grammar tables from /usr/lib/python2.7/lib2to3/PatternGrammar.txt
[2019-10-25 12:55:26,670] {__init__.py:45} INFO - Using executor SequentialExecutor
DB: mysql://hduser:***@localhost:3306/airflow
This will drop existing tables if they exist. Proceed? (y/n)
```

## Start Airflow Webserver

```
airflow webserver --port 8090
```



```
hduser@datamantra:~/airflow
File Edit View Search Terminal Tabs Help
hduser@datamantra:/usr/local/kafka x hduser@datamantra:/usr/local/confluent-3.2.0 x hduser@datamantra:~/workspace/Fraud-alert... x hduser@datamantra:/usr/local/kafka x hduser@datamantra:~/airflow
hduser@datamantra:~/airflow$ airflow webserver --port 8090
```

## Access Airflow Web Server

Access [https://<ubuntu\\_ipaddress>:8080](https://<ubuntu_ipaddress>:8080) to verify whether the airflow web server is up and running

DAGs

Search:

DAG	Schedule	Owner	Recent Tasks	Last Run	DAG Runs	Links
No data available in table						

Showing 1 to 0 of 0 entries

Hide Paused DAGs

Airflow is configured successfully

## Stop Airflow Webserver

Stop Airflow Webserver by pressing CTRL C on the terminal where it was started. This was just to test whether Airflow is working. We will start again when we start automation

```
[2019-10-25 17:13:09 +0000] [7171] [INFO] Shutting up the baggage from /home/hduser/airflow/dags
^C[2019-10-25 17:13:09 +0000] [7171] [INFO] Handling signal: int
[2019-10-25 17:13:09 +0000] [7179] [INFO] Worker exiting (pid: 7179)
[2019-10-25 17:13:09 +0000] [7178] [INFO] Worker exiting (pid: 7178)
[2019-10-25 17:13:09 +0000] [7182] [INFO] Worker exiting (pid: 7182)
[2019-10-25 17:13:09 +0000] [7180] [INFO] Worker exiting (pid: 7180)
[2019-10-25 17:13:09 +0000] [7171] [INFO] Shutting down: Master
hduser@datamantra:~$
```

## 5.2 Build all Projects

### Build FraudDetection Project

```
cd ~/workspace/FraudDetection
./deployInLocal.sh
```

```
hduser@datamantra: ~/workspace/FraudDetection
File Edit View Search Terminal Tabs Help
hduser@datamantra:/usr/local/kafka
hduser@datamantra:~/workspace/FraudDetection
hduser@datamantra:~/workspace/FraudDetection$ ./deployInLocal.sh
```

## Build Creditcard Producer Project

```
cd ~/workspace/CreditcardProducer  
mvn package
```

A screenshot of a terminal window titled "Terminal". The window shows a command-line interface with a dark background and light-colored text. The user has navigated to the directory ~/workspace/CreditcardProducer and run the command mvn package. The output of the command is visible at the bottom of the terminal window.

## Build Fraud-Alert-Dashboard Project

```
cd ~/workspace/Fraud-alert-dashboard  
mvn package
```

A screenshot of a terminal window titled "Terminal". The window shows a command-line interface with a dark background and light-colored text. The user has navigated to the directory ~/workspace/Fraud-alert-dashboard and run the command mvn package. The output of the command is visible at the bottom of the terminal window.

## 5.3 Start all the Servers

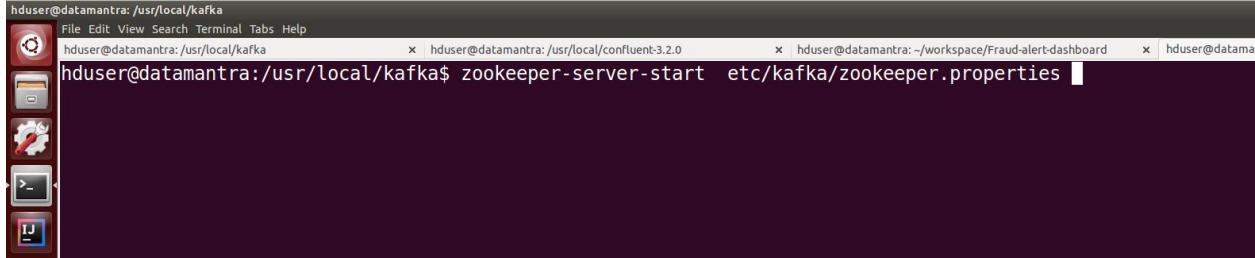
### Start Cassandra Server

```
cassandra -f
```

A screenshot of a terminal window titled "Terminal". The window shows a command-line interface with a dark background and light-colored text. The user has run the command cassandra -f to start the Cassandra server. The output of the command is visible at the bottom of the terminal window.

## Start Zookeeper Server

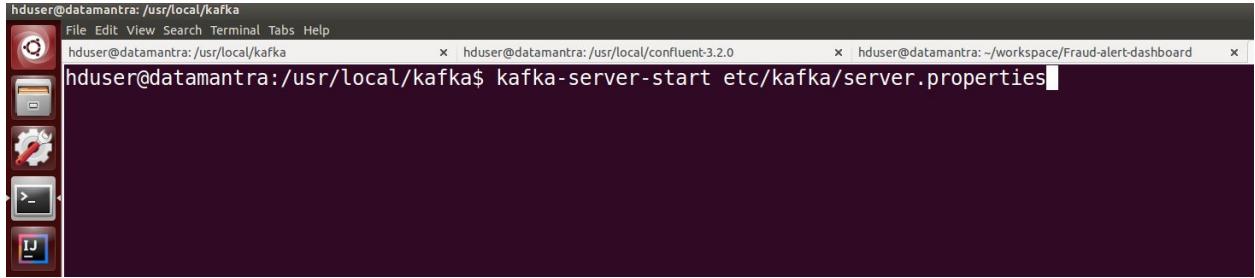
```
cd /usr/local/kafka  
zookeeper-server-start etc/kafka/zookeeper.properties
```



A screenshot of a terminal window titled "hduser@datamantra: /usr/local/kafka". The window contains three tabs: "hduser@datamantra: /usr/local/kafka", "hduser@datamantra: /usr/local/confluent-3.2.0", and "hduser@datamantra: ~/workspace/Fraud-alert-dashboard". The active tab shows the command "zookeeper-server-start etc/kafka/zookeeper.properties" being typed.

## Start Kafka Server

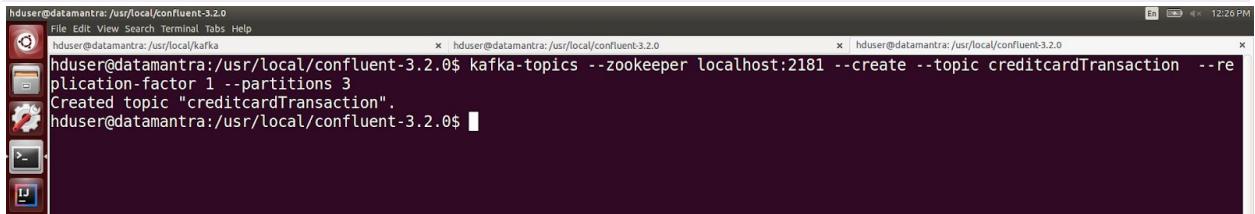
```
kafka-server-start etc/kafka/server.properties
```



A screenshot of a terminal window titled "hduser@datamantra: /usr/local/kafka". The window contains three tabs: "hduser@datamantra: /usr/local/kafka", "hduser@datamantra: /usr/local/confluent-3.2.0", and "hduser@datamantra: ~/workspace/Fraud-alert-dashboard". The active tab shows the command "kafka-server-start etc/kafka/server.properties" being typed.

## Create transaction topic

```
kafka-topics --zookeeper localhost:2181 --create --topic creditcardTransaction --replication-factor 1 --partitions 3
```



A screenshot of a terminal window titled "hduser@datamantra: /usr/local/confluent-3.2.0". The window contains three tabs: "hduser@datamantra: /usr/local/kafka", "hduser@datamantra: /usr/local/confluent-3.2.0", and "hduser@datamantra: /usr/local/confluent-3.2.0". The active tab shows the command "kafka-topics --zookeeper localhost:2181 --create --topic creditcardTransaction --replication-factor 1 --partitions 3" being run. The output indicates that the topic "creditcardTransaction" was created.

## Spark Cluster Setup

### Start Spark Master

```
start-master.sh
```

```
hduser@datamantra:~$ start-master.sh
starting org.apache.spark.deploy.master.Master, logging to /usr/local/spark/logs/spark-hduser-org.apache.spark.deploy.master.Master-1-datamantra.out
hduser@datamantra:~$
```

## Start Spark Slave

```
start-slave.sh spark://datamantra:7077
```

```
hduser@datamantra:~$ start-slave.sh spark://datamantra:7077
starting org.apache.spark.deploy.worker.Worker, logging to /usr/local/spark/logs/spark-hduser-org.apache.spark.deploy.worker.Worker-1-datamantra.out
hduser@datamantra:~$
```

## Access Spark Web UI

Check whether Spark Cluster is up and running

[http://ubuntu\\_ip:8080/](http://ubuntu_ip:8080/)

The screenshot shows the Apache Spark Web UI interface. At the top, there's a navigation bar with tabs for Airflow - DAGs, Fraud Alert Monitoring Dashboard, and Spark Master at spark://pixipanda:7077. The main content area has a yellow header bar with the title "Spark Master at spark://pixipanda:7077". Below this, there's a summary section with the following data:

URL: spark://pixipanda:7077	REST URL: spark://pixipanda:6066 (cluster mode)
Alive Workers: 1	Cores in use: 2 Total, 0 Used
Memory in use: 3.8 GB Total, 0.0 B Used	Applications: 0 Running, 0 Completed
Drivers: 0 Running, 0 Completed	Status: ALIVE

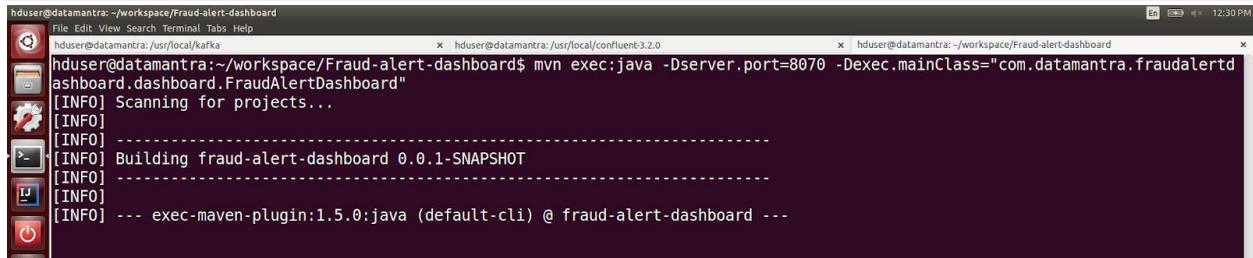
Below the summary, there are three main sections:

- Workers**: A table showing one worker entry: worker-20191023200726-192.168.0.109-36277.
- Running Applications**: An empty table with columns: Application ID, Name, Cores, Memory per Executor, Submitted Time, User, State, Duration.
- Completed Applications**: An empty table with columns: Application ID, Name, Cores, Memory per Executor, Submitted Time, User, State, Duration.

## Start the dashboard for visualization

```
cd ~/workspace/Fraud-alert-dashboard/
mvn exec:java -Dserver.port=8070
```

```
-Dexec.mainClass="com.datamantra.fraudalertdashboard.dashboard.FraudAlertDa  
shboard"
```

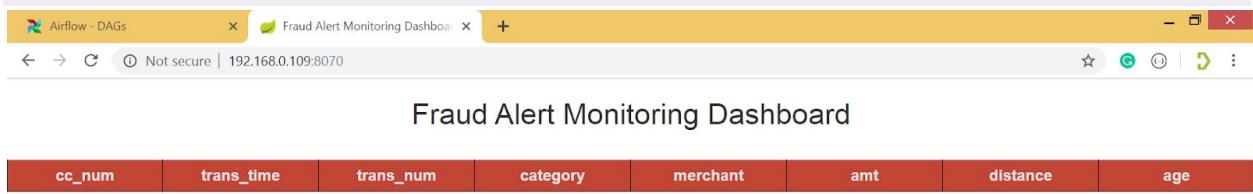


A terminal window titled "hduser@datamantra: ~/workspace/Fraud-alert-dashboard". It shows the command "mvn exec:java -Dserver.port=8070 -Dexec.mainClass="com.datamantra.fraudalertdashboard.FraudAlertDashboard"" being run. The output shows the application is scanning for projects, building the fraud-alert-dashboard 0.0.1-SNAPSHOT, and executing the main class.

```
hduser@datamantra:~/workspace/Fraud-alert-dashboard$ mvn exec:java -Dserver.port=8070 -Dexec.mainClass="com.datamantra.fraudalertdashboard.FraudAlertDashboard"  
[INFO] Scanning for projects...  
[INFO] [INFO]  
[INFO] Building fraud-alert-dashboard 0.0.1-SNAPSHOT  
[INFO] [INFO]  
[INFO] --- exec-maven-plugin:1.5.0:java (default-cli) @ fraud-alert-dashboard ---
```

## Access Dashboard Web UI

[http://ubuntu\\_ip:8070](http://ubuntu_ip:8070)

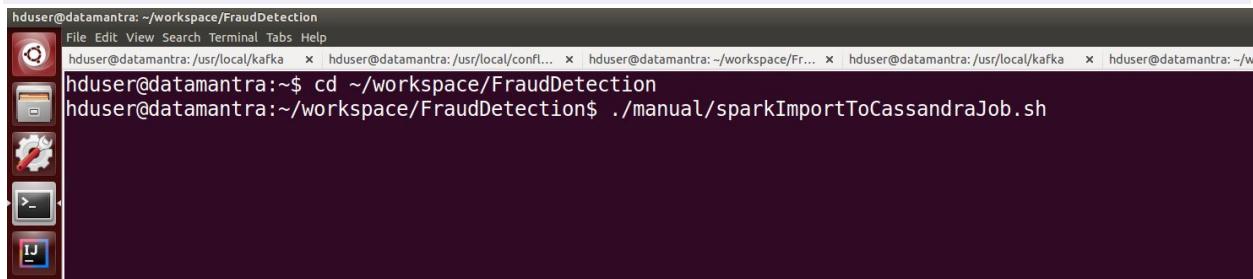


## 5.4 Airflow Automation

### Start Spark Import Job

This Job will be started manually to import customer data and transaction data from the filesystem to Cassandra Keyspace.

```
cd ~/workspace/FraudDetection  
../manual/sparkImportToCassandraJob.sh
```



A terminal window titled "hduser@datamantra: ~/workspace/FraudDetection". It shows the command "cd ~/workspace/FraudDetection" and then "../manual/sparkImportToCassandraJob.sh" being run. The output shows the job has been executed.

```
hduser@datamantra:~/workspace/FraudDetection$ cd ~/workspace/FraudDetection  
hduser@datamantra:~/workspace/FraudDetection$ ./manual/sparkImportToCassandraJob.sh
```

Spark Master at spark://datamantra:7077

URL: spark://datamantra:7077  
REST URL: spark://datamantra:6066 (cluster mode)  
Alive Workers: 1  
Cores in use: 2 Total, 2 Used  
Memory in use: 3.8 GB Total, 1024.0 MB Used  
Applications: 1 Running, 0 Completed  
Drivers: 0 Running, 0 Completed  
Status: ALIVE

**Workers**

Worker Id	Address	State	Cores	Memory
worker-20191025122856-192.168.0.100-40121	192.168.0.100:40121	ALIVE	2 (2 Used)	3.8 GB (1024.0 MB Used)

**Running Applications**

Application ID	Name	Cores	Memory per Executor	Submitted Time	User	State	Duration
app-20191025142701-0000 (kill)	Import Data to Cassandra	2	1024.0 MB	2019/10/25 14:27:01	hduser	RUNNING	2 s

**Completed Applications**

Application ID	Name	Cores	Memory per Executor	Submitted Time	User	State	Duration
----------------	------	-------	---------------------	----------------	------	-------	----------

## Output

Transaction data and customer data is imported to Cassandra Keyspace

```
hduser@pixipanda:~$ cqlsh:creditcard> SELECT * from fraud_transaction limit 4 ;
cc_num | trans_time | age | amt | category | distance | is_fraud | merch_lat | merch_long | merchant
       | trans_num
-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
180036251237802 | 2012-01-01 17:54:16.000000+0000 | 63 | 2108 | shopping_net | 157.66 | 1 | 37.13679 | -84.94059 | Fisher Inc | f5961217ad20bcb888558e451c0555b8
180036251237802 | 2012-01-01 17:41:52.000000+0000 | 63 | 2583 | shopping_net | 59.92 | 1 | 38.4867 | -84.11508 | Stamm-Witting | 122b29e9fd69ed43223ad7a56cf38881
180036251237802 | 2012-01-01 17:28:32.000000+0000 | 63 | 262 | gas_transport | 80.43 | 1 | 38.76958 | -83.27012 | Streich, Hansen and Veum | f5ebbe07eb6fbct3f194cd24f4b9bf8
180036251237802 | 2012-01-01 17:07:28.000000+0000 | 63 | 2361 | grocery_pos | 118.8 | 1 | 39.13225 | -84.00291 | osciski, Gislason and Mertz | f95517fb6167e00f6adfb3c6c1f1d03
(4 rows)
cqlsh:creditcard>
```

## Create an Airflow DAG directory.

Automation scripts must be kept in this directory. Airflow will look for scripts in this directory and it will load the scripts to the web server. So copy frauddetection.py file from FraudDetection Project to airflow dags directory

```
mkdir ~/airflow/dags
cp ~/workspace/FraudDetection/automate/airflow/frauddetection.py
~/airflow/dags/.
```

```

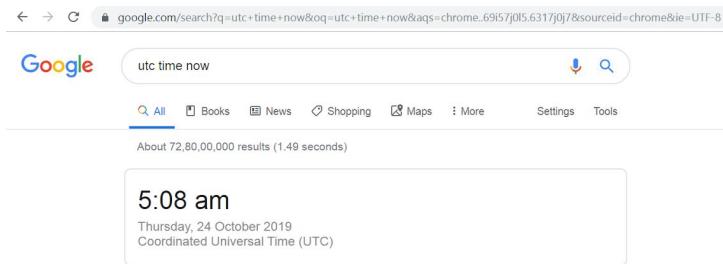
hduser@datamantra:~ 
hduser@datamantra:/usr/local/kafka x hduser@datamantra:/usr/local/confl... x hduser@datamantra:~/workspace/FraudDetection/automate/airflow/frauddetection.py ~/airflow/dags/. 
hduser@datamantra:~$ 

```

## Change the start time in [frauddetection.py](#)

Airflow uses UTC timezone

UTC time in my system is 5:08 am. We will kick start the automation at 5:20 am



start\_date is set to 2019, 10, 24, 5 i.e start date is set to 5:00 am

schedule\_interval is set to “\*/20 \* \* \* \*”. It means every 20 mins automation step will be evaluated.

**Actual start time = start\_date + schedule\_interval = 5:00 + 20 mins = 5:20 am**

```

default_args = {
    'owner': 'me',
    'start_date': dt.datetime(2019, 10, 24, 5),
    'retries': 1,
    'retry_delay': dt.timedelta(minutes=5),
}

with DAG(DAG_NAME,
         default_args=default_args,
         schedule_interval='*/20 * * * *',
         ) as dag:

```

## Compile the changes

```
python ~/airflow/dags/fraudetection.py
```



```
hduser@datamantra:~$ python ~/airflow/dags/fraudetection.py
[2019-10-25 16:02:00,865] {driver.py:120} INFO - Generating grammar tables from /usr/lib/python2.7/lib2to3/Grammar.txt
[2019-10-25 16:02:00,917] {driver.py:120} INFO - Generating grammar tables from /usr/lib/python2.7/lib2to3/PatternGrammar.txt
hduser@datamantra:~$
```

## Reset airflow database

Reset is done so that airflow tables are created in MySQL database

```
airflow resetdb
```

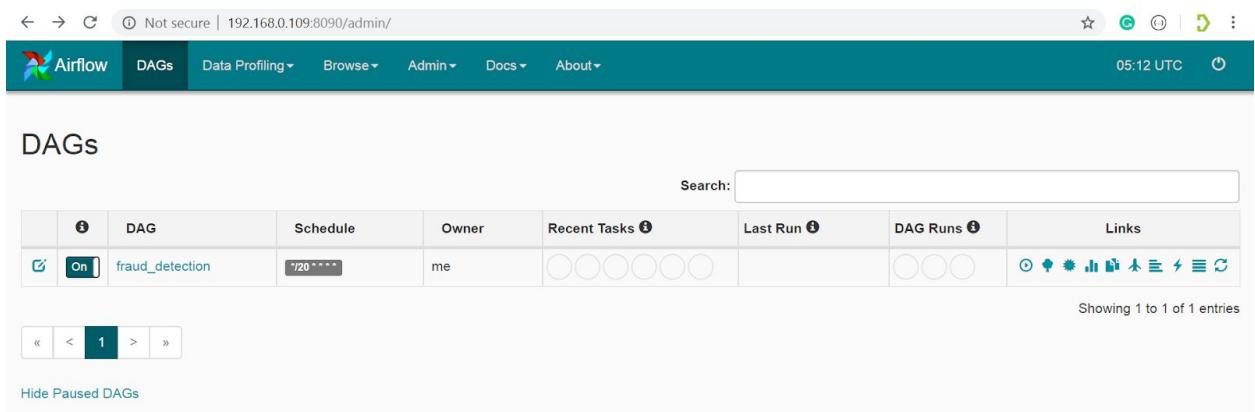
## Start Airflow Webserver

```
airflow webserver --port 8090
```

## Access Airflow Webserver

```
http://ubuntu_ipaddr:8090
```

Click on OFF button to set it ON



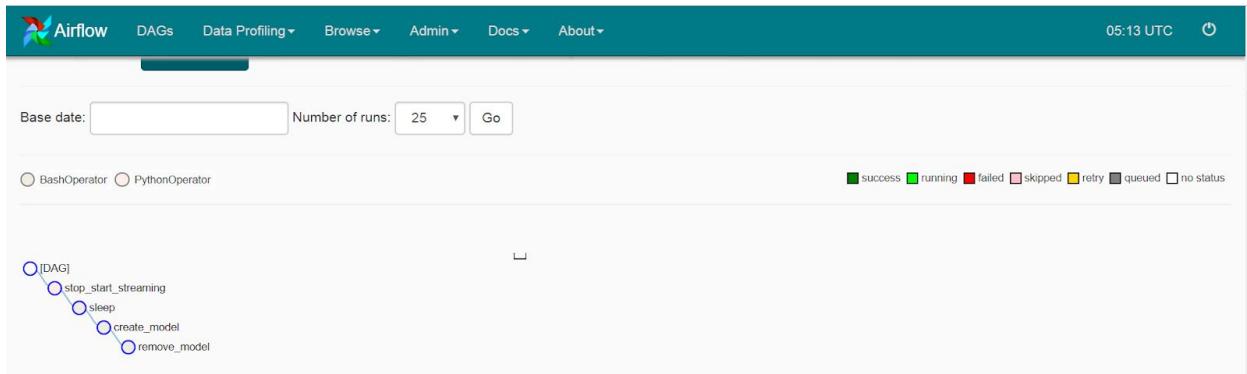
	DAG	Schedule	Owner	Recent Tasks	Last Run	DAG Runs	Links
<input checked="" type="checkbox"/>	<input checked="" type="button"/> On <a href="#">fraud_detection</a>	<a href="#">*/20 * * * *</a>	me	<span>○ ○ ○ ○ ○</span>		<span>○ ○ ○</span>	<a href="#">Logs</a> <a href="#">Metrics</a> <a href="#">Timeline</a> <a href="#">Task Instances</a> <a href="#">DAG Runs</a> <a href="#">File System</a>

Showing 1 to 1 of 1 entries

If you click on fraud\_detection you will see the automation steps.

1. Remove Model is any exist
2. Create New Model by running Spark ML Job
3. Sleep for 5 seconds(This step is optional)

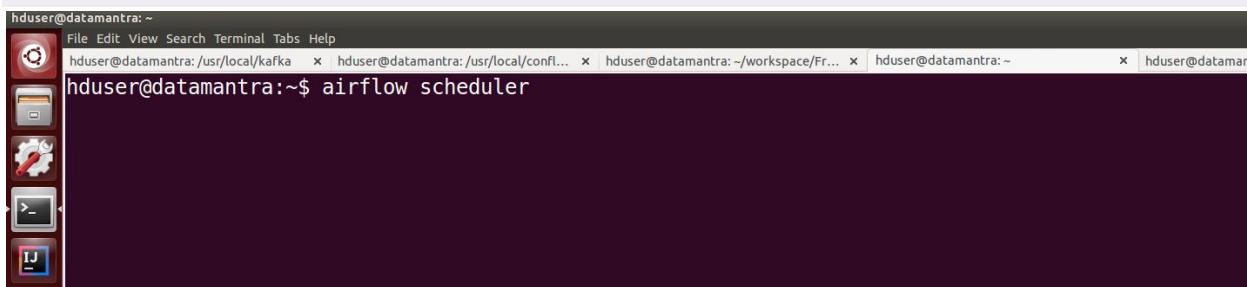
#### 4. Stop and Start Spark Streaming Job



### Start Airflow Scheduler

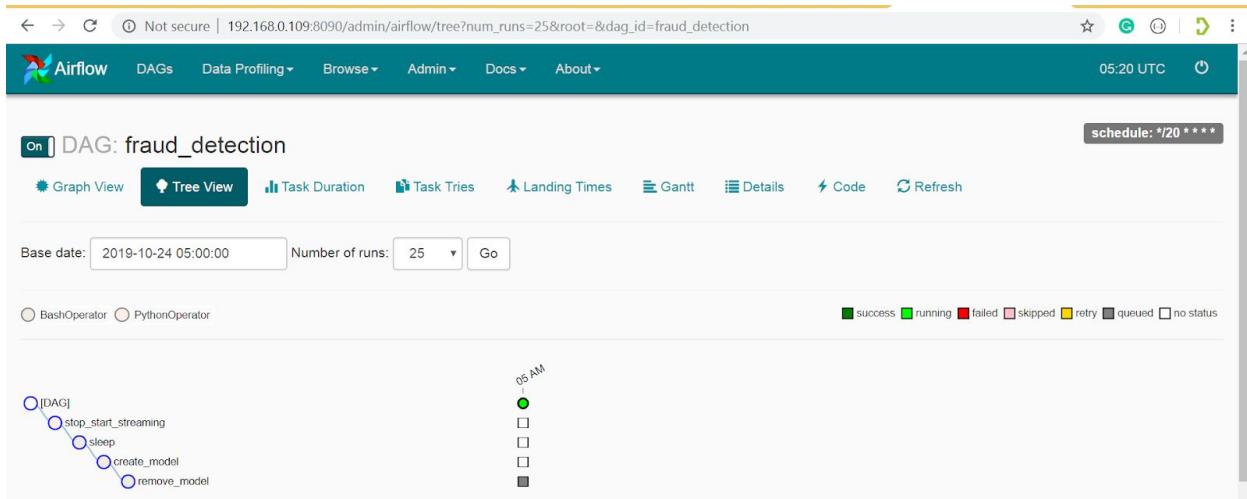
Airflow Scheduler is the one which will actually kick start your automation

```
airflow scheduler
```



### Output

Automation start at 5:20 am UTC



## Spark ML Job Scheduled

← → ⌂ Not secure | 192.168.0.109:8080 ⋮

 **Spark Master at spark://pixipanda:7077**

URL: `spark://pixipanda:7077`  
REST URL: `spark://pixipanda:6066 (cluster mode)`  
Alive Workers: 1  
Cores in use: 3 Total, 1 Used  
Memory in use: 3.8 GB Total, 1024.0 MB Used  
Applications: 1 Running, 1 Completed  
Drivers: 0 Running, 0 Completed  
Status: ALIVE

**Workers**

Worker Id	Address	State	Cores	Memory
worker-20191024102917-192.168.0.109-46654	192.168.0.109:46654	ALIVE	3 (1 Used)	3.8 GB (1024.0 MB Used)

**Running Applications**

Application ID	Name	Cores	Memory per Executor	Submitted Time	User	State	Duration
app-20191024105014-0001	(kill) Fraud Detection Spark ML Training	1	1024.0 MB	2019/10/24 10:50:14	hduser	RUNNING	12 s

## Spark ML Job Completed

### Completed Applications

Application ID	Name	Cores	Memory per Executor	Submitted Time	User	State	Duration
app-20191024105014-0001	Fraud Detection Spark ML Training	1	1024.0 MB	2019/10/24 10:50:14	hduser	FINISHED	52 s
app-20191024103614-0000	Import Data to Cassandra	3	1024.0 MB	2019/10/24 10:36:14	hduser	FINISHED	30 s

## All automation steps completed.

This means Spark Streaming is up and running

Airflow DAGs Data Profiling ▾ Browse ▾ Admin ▾ Docs ▾ About ▾ 05:22 UTC ⌂

**DAG: fraud\_detection** schedule: \*/20 \* \* \* \*

Graph View Tree View Task Duration Task Tries Landing Times Gantt Details Code Refresh

Base date: 2019-10-24 05:00:00 Number of runs: 25 Go

BashOperator PythonOperator success running failed skipped retry queued no status

```
graph TD; DAG([DAG]) --> stop_start_streaming[stop_start_streaming]; stop_start_streaming --> sleep[sleep]; sleep --> create_model[create_model]; create_model --> remove_model[remove_model];
```

05 AM

## Spark Streaming Job is up and running

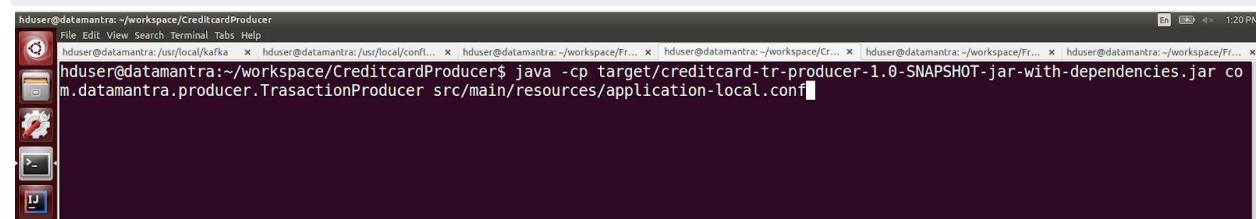
The screenshot shows the Spark Master UI at `spark://pixipanda:7077`. It displays the following information:

- Cluster Metrics:** URL: `spark://pixipanda:7077`, REST URL: `spark://pixipanda:6066 (cluster mode)`, Alive Workers: 1, Cores in use: 3 Total, 2 Used, Memory in use: 3.8 GB Total, 2.0 GB Used, Applications: 1 Running, 2 Completed, Drivers: 1 Running, 0 Completed, Status: ALIVE.
- Workers:** A table showing one worker: worker-20191024102917-192.168.0.109-46654, Address: 192.168.0.109:46654, State: ALIVE, Cores: 3 (2 Used), Memory: 3.8 GB (2.0 GB Used).
- Running Applications:** A table showing one application: app-20191024105132-0002 (kill) RealTime Creditcard FraudDetection, Application ID: app-20191024105132-0002, Name: RealTime Creditcard FraudDetection, Cores: 1, Memory per Executor: 1024.0 MB, Submitted Time: 2019/10/24 10:51:32, User: hduser, State: RUNNING, Duration: 47 s.

## Start Kafka Producer

Spark Streaming Job is up and running, now we will start Kafka Producer

```
java -cp
target/creditcard-tr-producer-1.0-SNAPSHOT-jar-with-dependencies.jar
com.datamantra.producer.TrasactionProducer
src/main/resources/application-local.conf
```



## Output

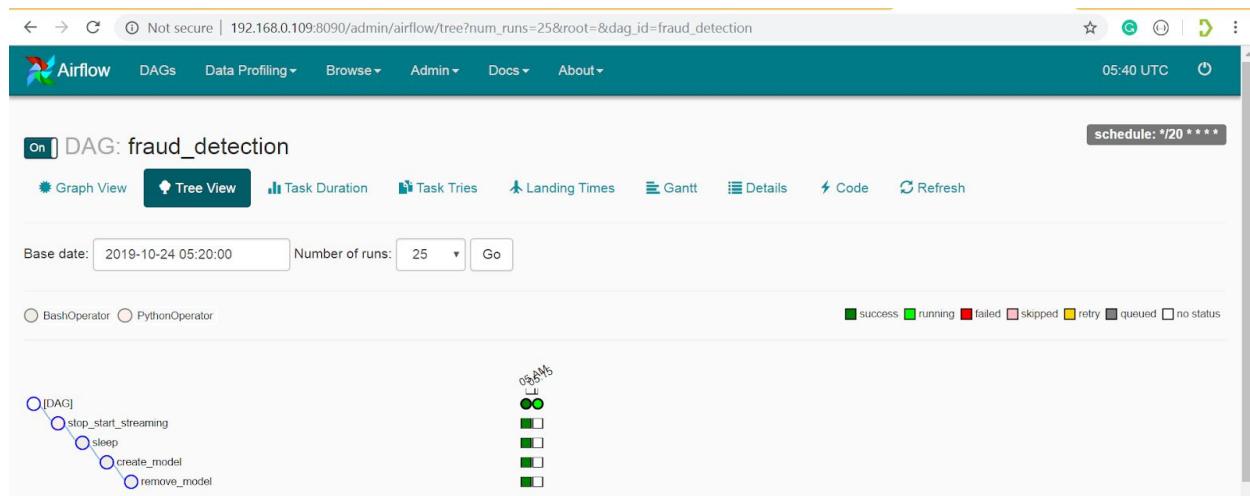
A fraud alert is displayed on the dashboard

The screenshot shows a browser tab titled "Fraud Alert Monitoring Dashboard". The page displays a single row of data from a table:

cc_num	trans_time	trans_num	category	merchant	amt	distance	age
374115112731710	2019-10-24 10:54:17	11efb54624f40fe0...	entertainment	Johns-Hoeger	64	4.16	41

## Next set of automation

Next set of automation steps scheduled at 5:40 am(every 20 mins)



## Spark ML Job is running

This will create new model

Apache Spark 2.2.1

Spark Master at spark://pixipanda:7077

URL: spark://pixipanda:7077  
REST URL: spark://pixipanda:8066 (cluster mode)  
Alive Workers: 1  
Cores in use: 3 Total, 3 Used  
Memory in use: 3.8 GB Total, 3.0 GB Used  
Applications: 2 Running, 2 Completed  
Drivers: 1 Running, 0 Completed  
Status: ALIVE

Workers

Worker Id	Address	State	Cores	Memory
worker-20191024102917-192.168.0.109-46654	192.168.0.109:46654	ALIVE	3 (3 Used)	3.8 GB (3.0 GB Used)

Running Applications

Application ID	Name	Cores	Memory per Executor	Submitted Time	User	State	Duration
app-20191024111015-0003 (kill)	Fraud Detection Spark ML Training	1	1024.0 MB	2019/10/24 11:10:15	hduser	RUNNING	6 s
app-20191024105132-0002 (kill)	RealTime Creditcard FraudDetection	1	1024.0 MB	2019/10/24 10:51:32	hduser	RUNNING	19 min

## Completed Applications

Application ID	Name	Cores	Memory per Executor	Submitted Time	User	State	Duration
app-20191024111015-0003	Fraud Detection Spark ML Training	1	1024.0 MB	2019/10/24 11:10:15	hduser	FINISHED	1.1 min
app-20191024105014-0001	Fraud Detection Spark ML Training	1	1024.0 MB	2019/10/24 10:50:14	hduser	FINISHED	52 s
app-20191024103614-0000	Import Data to Cassandra	3	1024.0 MB	2019/10/24 10:36:14	hduser	FINISHED	30 s

## Spark Streaming Job Stopped

### Completed Applications

Application ID	Name	Cores	Memory per Executor	Submitted Time	User	State	Duration
app-20191024105132-0002	RealTime Creditcard FraudDetection	1	1024.0 MB	2019/10/24 10:51:32	hduser	FINISHED	20 min
app-20191024111015-0003	Fraud Detection Spark ML Training	1	1024.0 MB	2019/10/24 11:10:15	hduser	FINISHED	1.1 min
app-20191024105014-0001	Fraud Detection Spark ML Training	1	1024.0 MB	2019/10/24 10:50:14	hduser	FINISHED	52 s
app-20191024103614-0000	Import Data to Cassandra	3	1024.0 MB	2019/10/24 10:36:14	hduser	FINISHED	30 s

## New Streaming Job Started

This will load the new model created by the previous Spark ML Job. Transactions will be predicted using this new model.

Screenshot of the Spark Master UI at `spark://pixipanda:7077`. The UI shows the following information:

- URL: `spark://pixipanda:7077`
- REST URL: `spark://pixipanda:6066 (cluster mode)`
- Alive Workers: 1
- Cores in use: 3 Total, 2 Used
- Memory in use: 3.8 GB Total, 2.0 GB Used
- Applications: 1 Running, 4 Completed
- Drivers: 1 Running, 1 Completed
- Status: ALIVE

**Workers**

Worker Id	Address	State	Cores	Memory
worker-20191024102917-192.168.0.109-46654	192.168.0.109:46654	ALIVE	3 (2 Used)	3.8 GB (2.0 GB Used)

**Running Applications**

Application ID	Name	Cores	Memory per Executor	Submitted Time	User	State	Duration
app-20191024111156-0004 (kill)	RealTime Creditcard FraudDetection	1	1024.0 MB	2019/10/24 11:11:56	hduser	RUNNING	25 s

Screenshot of the Airflow UI showing the DAG `fraud_detection`. The DAG has the following tasks:

- [DAG]
- stop\_start\_streaming
- sleep
- create\_model
- remove\_model

The DAG is scheduled to run every 20 minutes. The current run status is shown in the Gantt chart:

Task	Run 1	Run 2	Run 3	Run 4	Run 5
[DAG]	success	success	success	success	success
stop_start_streaming	running	running	running	running	running
sleep	success	success	success	success	success
create_model	success	success	success	success	success
remove_model	success	success	success	success	success

Legend: success (green), running (yellow), failed (red), skipped (pink), retry (orange), queued (grey), no status (white).