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

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
Username: hduser
Password: hadoop123

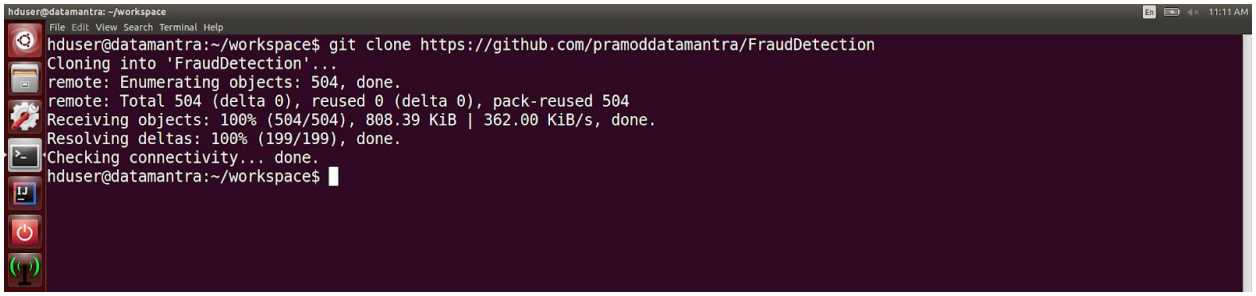
1.1 Clone Code

Create a workspace directory
Create a workspace directory in your home directory

```
mkdir workspace
cd workspace
```

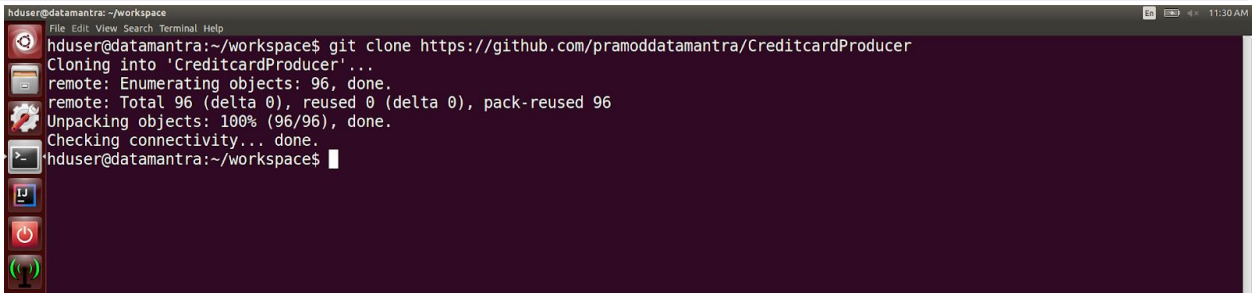
Clone FraudDection from GitHub
This is a branch code. It is a bit different from the code shown in the video.
Here we have VectorSlicer, StandardScaler and Confusion matrix.

```
git clone
https://github.com/pramoddatamantra/FraudDetection/tree/FraudDetection_ML
```



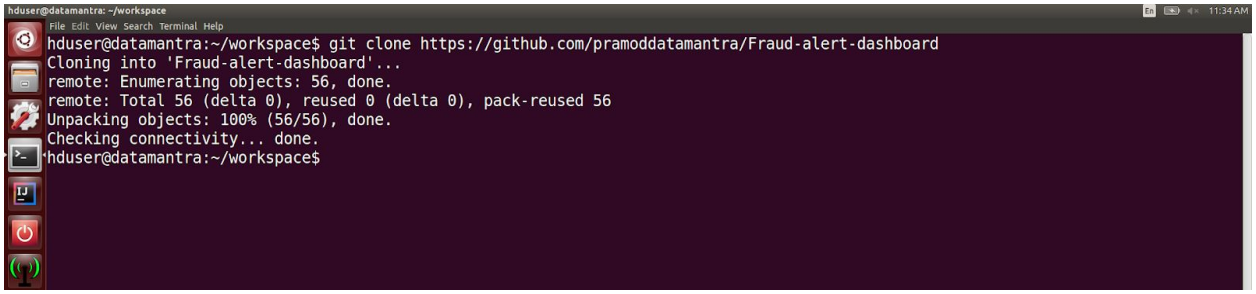
Clone Credit card producer from GitHub

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



Clone Fraud-alert-dashboard

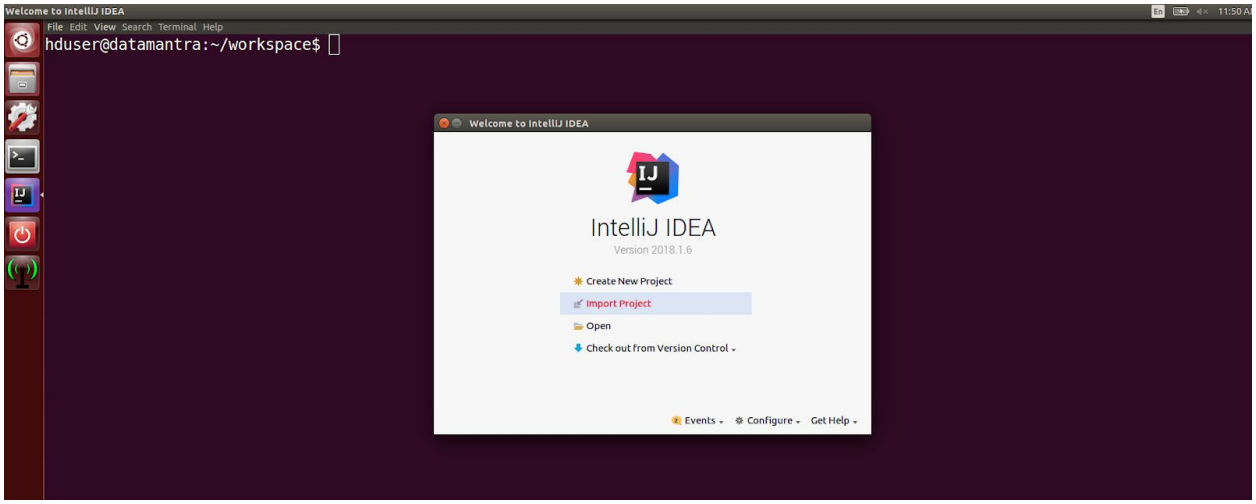
```
git clone https://github.com/pramoddatamantra/Fraud-alert-dashboard
```



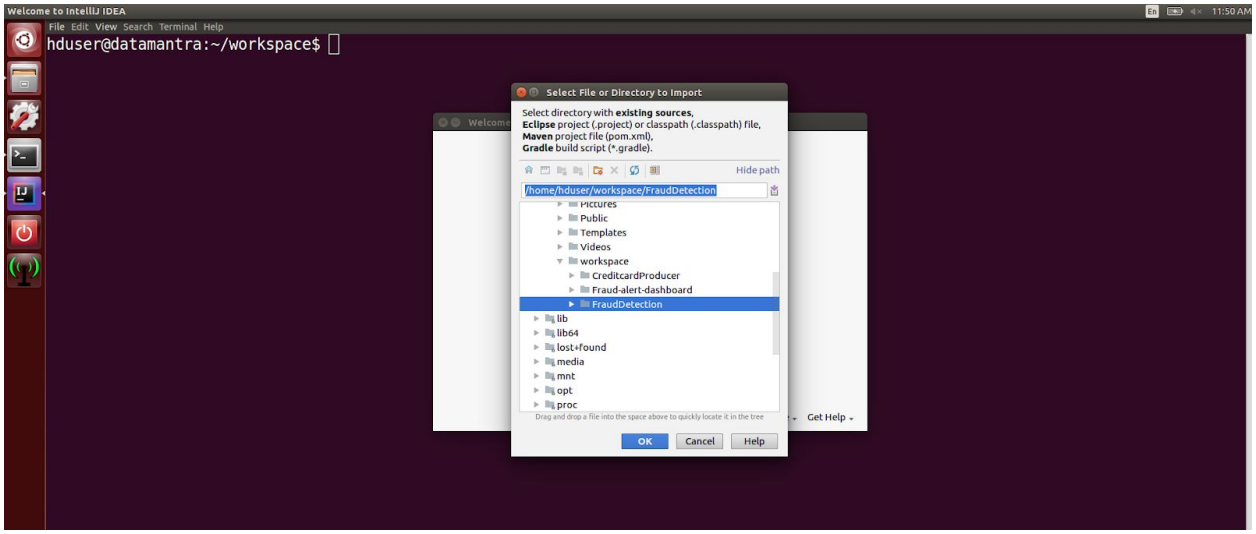
1.2 Import to IntelliJ

Import FraudDection to IntelliJ

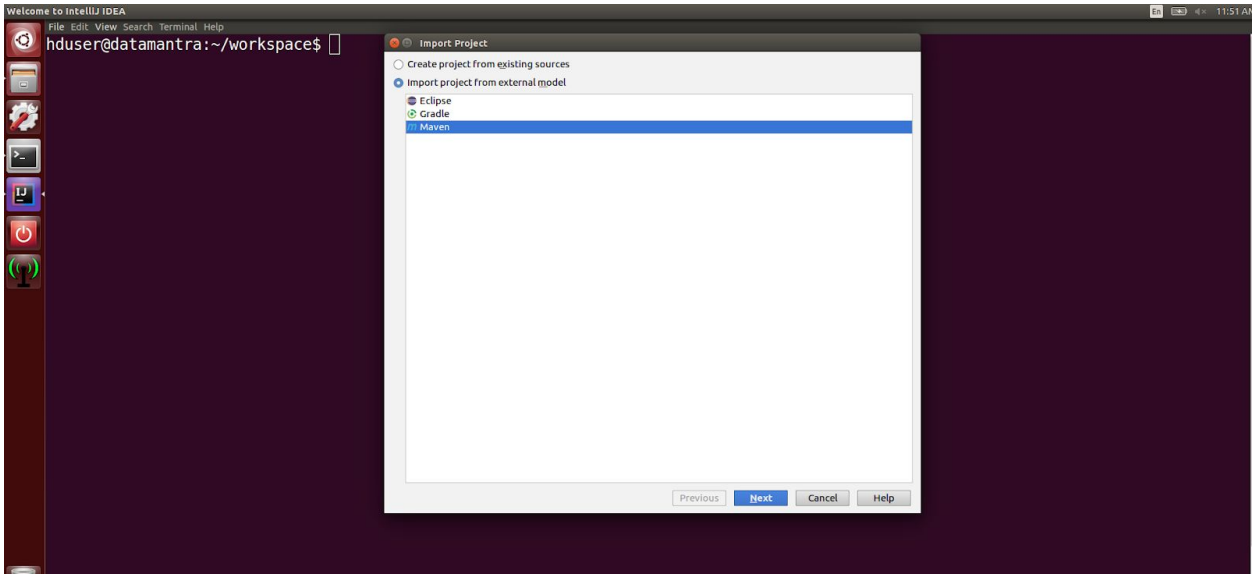
Click Import



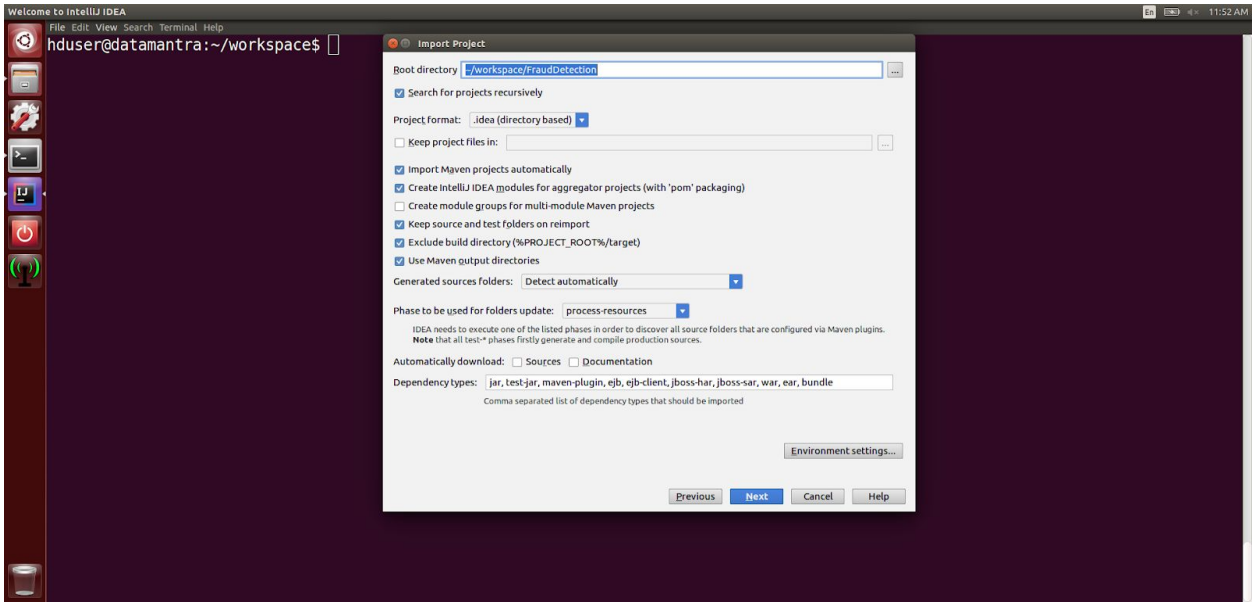
Select FraudDetection Project. Click OK. Click Next



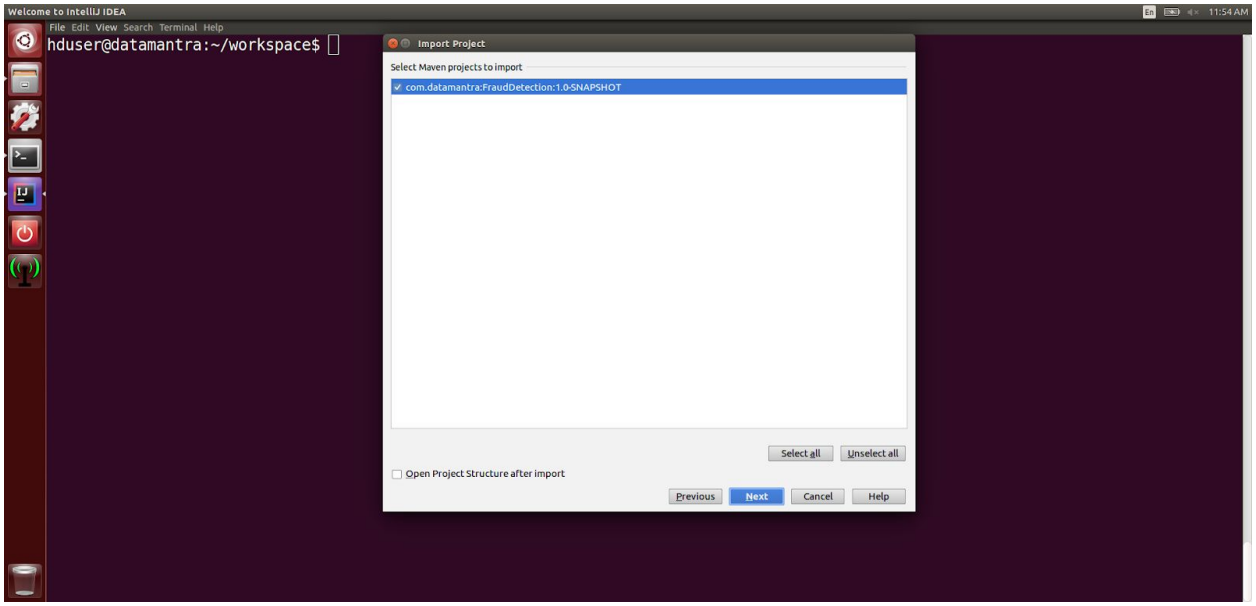
Select Maven. Click Next



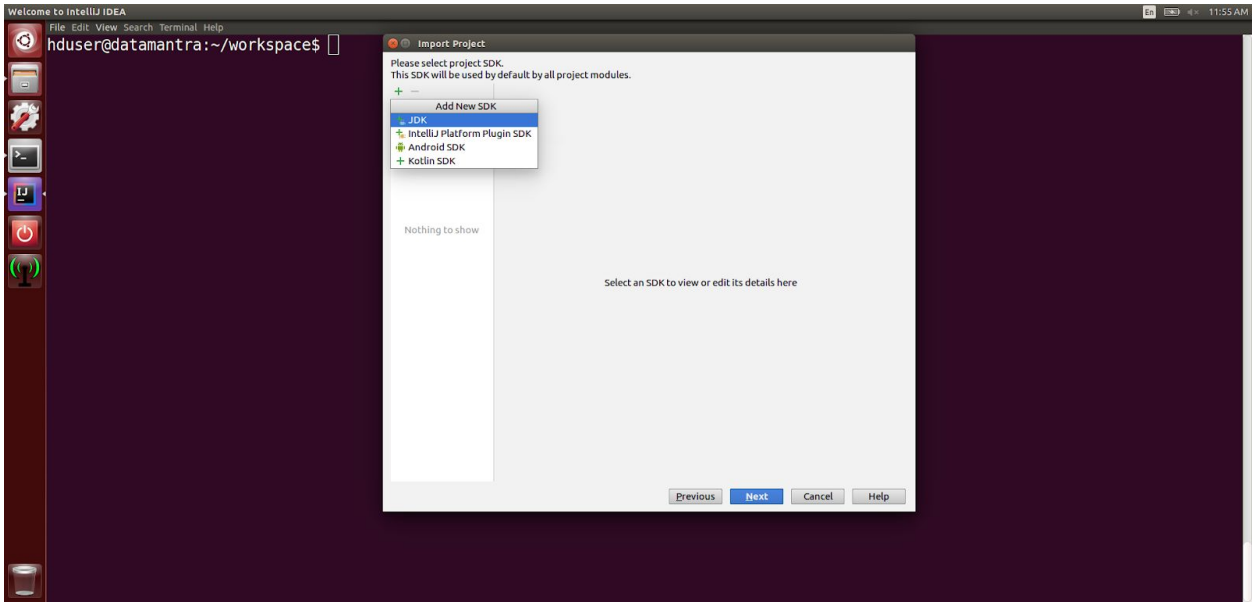
Checkmark “Search for projects recursively” and “Import Maven projects automatically” Click Next



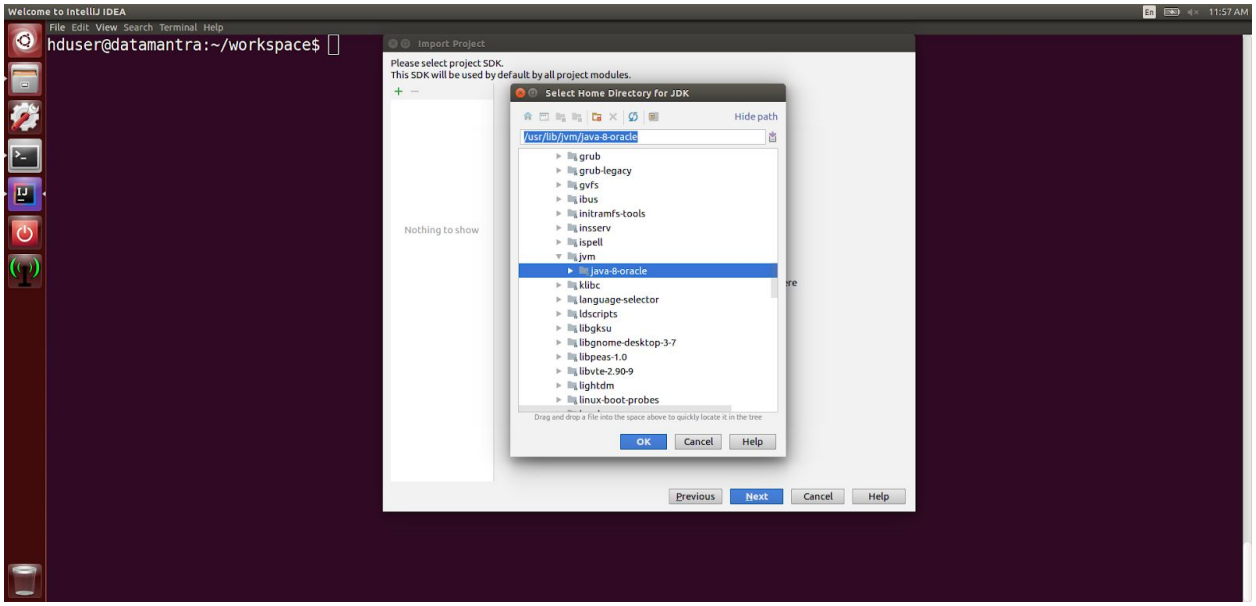
By default, the FraudDetection project will be selected. Click Next



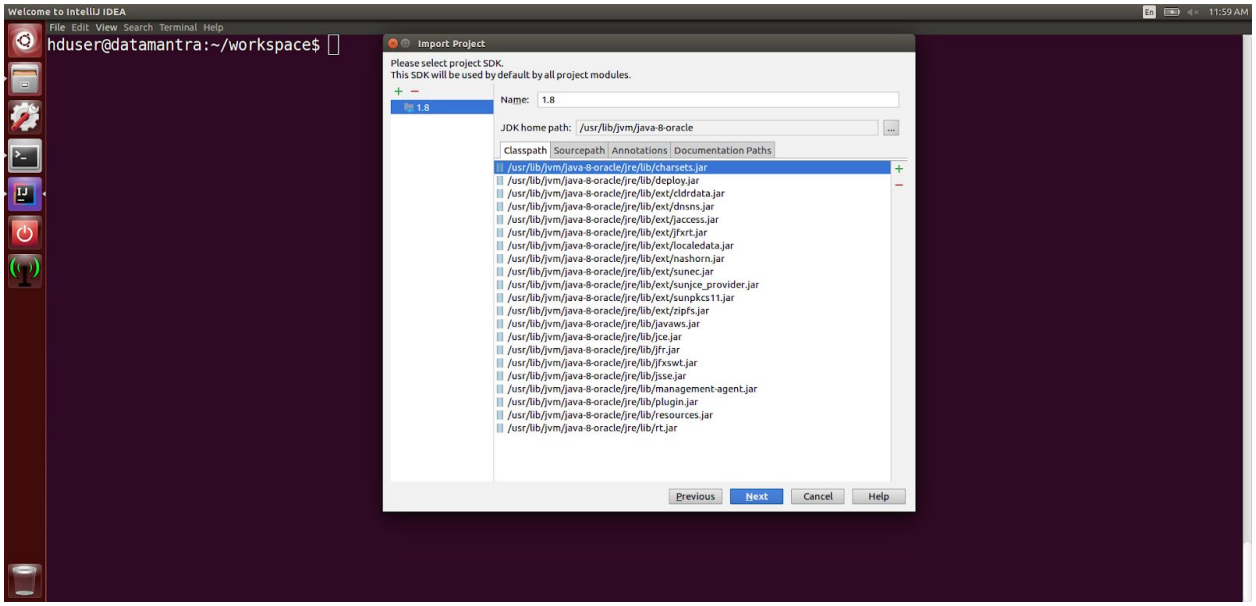
Select JDK



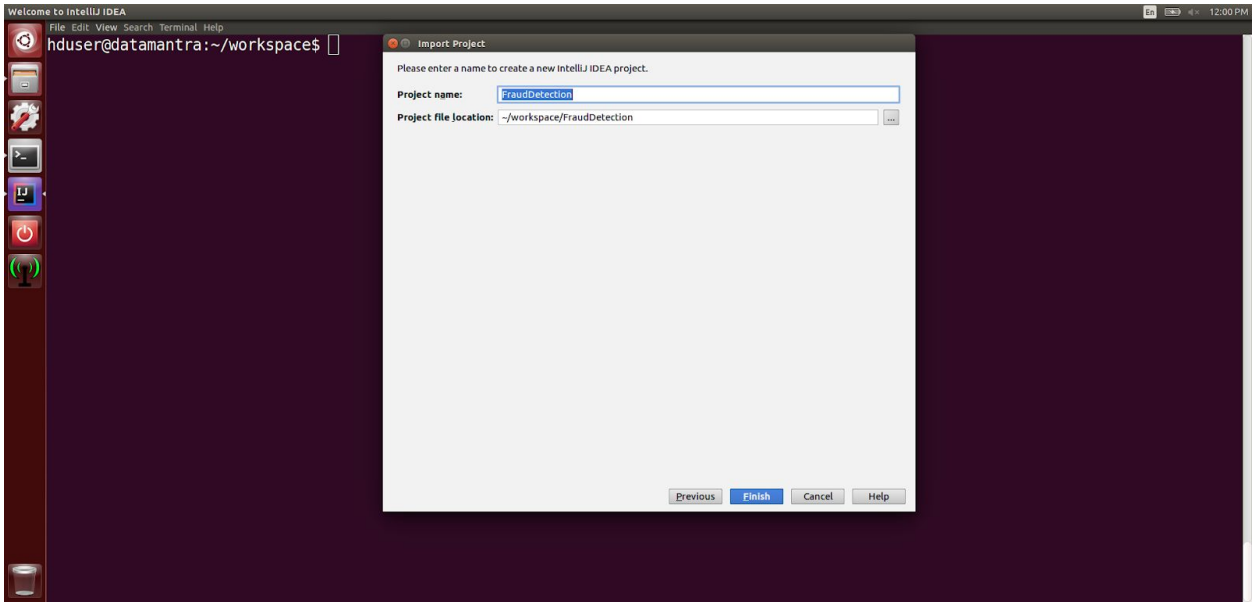
Select the java-8-oracle directory. Click OK and Click Next



Java 1.8 SDK is selected. Click Next



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



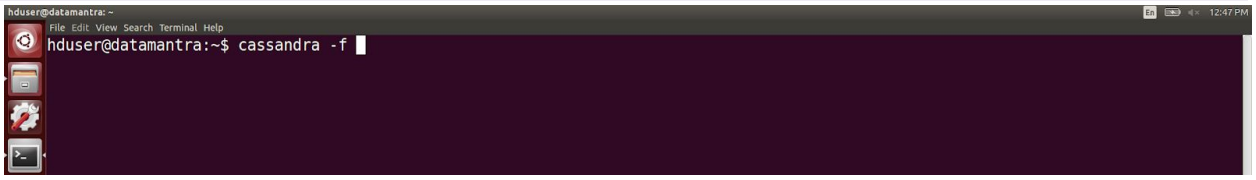
Follow the above steps to import CreditcardProducer and Fraud-alert-dashboard project into IntelliJ

2.Initial Import Spark Job

2.1 Demonstration

Start Cassandra Server

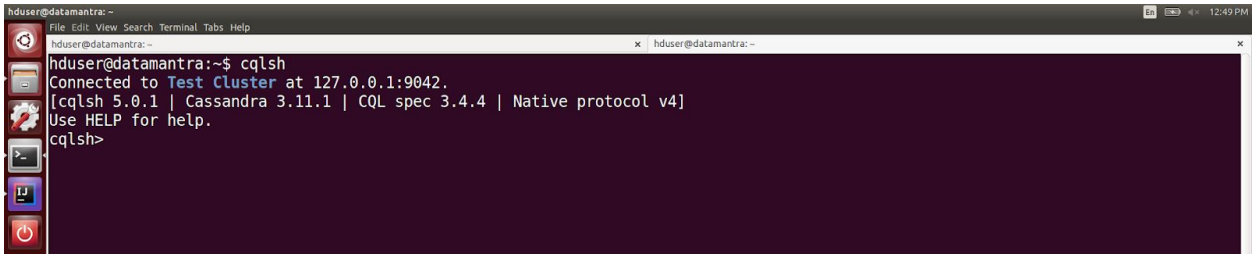
```
Cassandra -f
```



Connect to Cassandra Server

Connect to Cassandra Server using Cassandra Cqlsh client

```
cqlsh
```



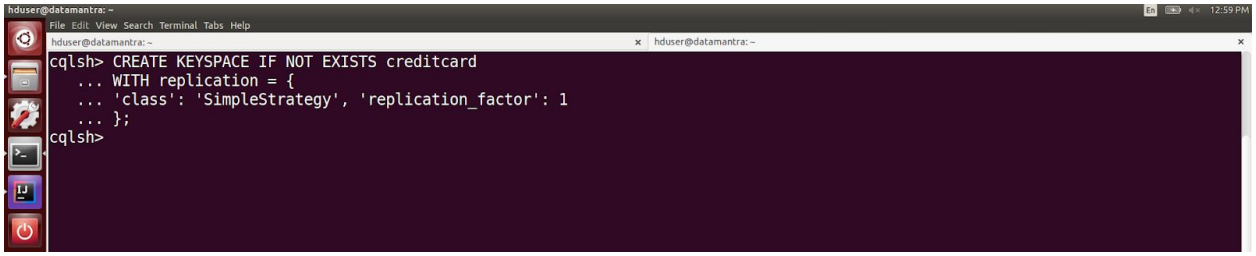
Create keyspace and tables

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

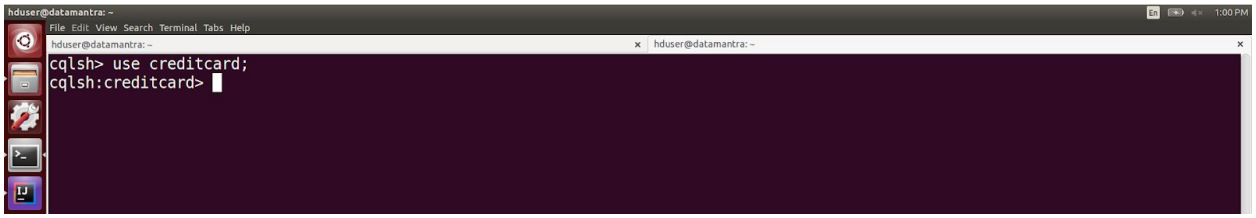
Create creditcard keyspace

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



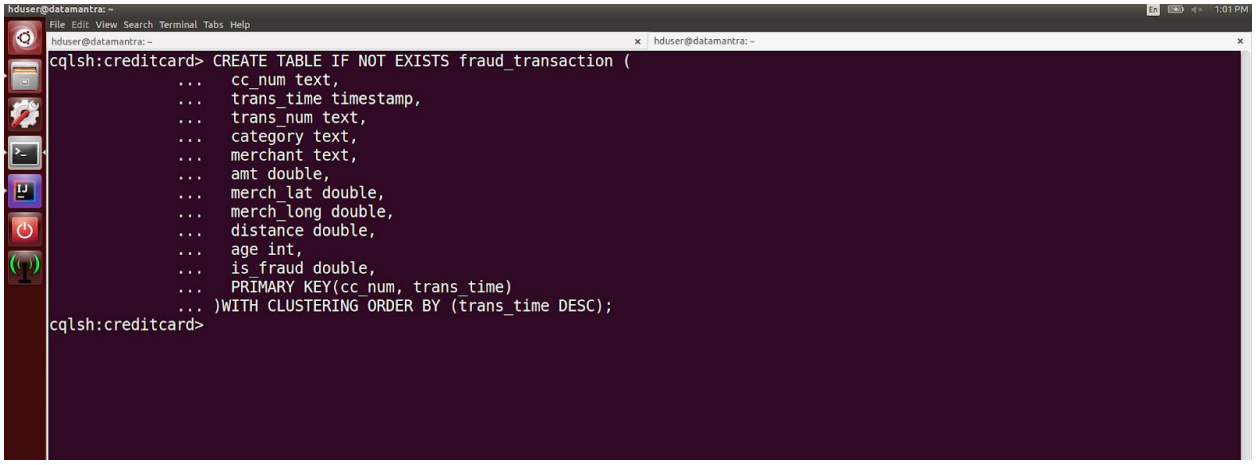
Use creditcard keyspace

```
USE creditcard;
```



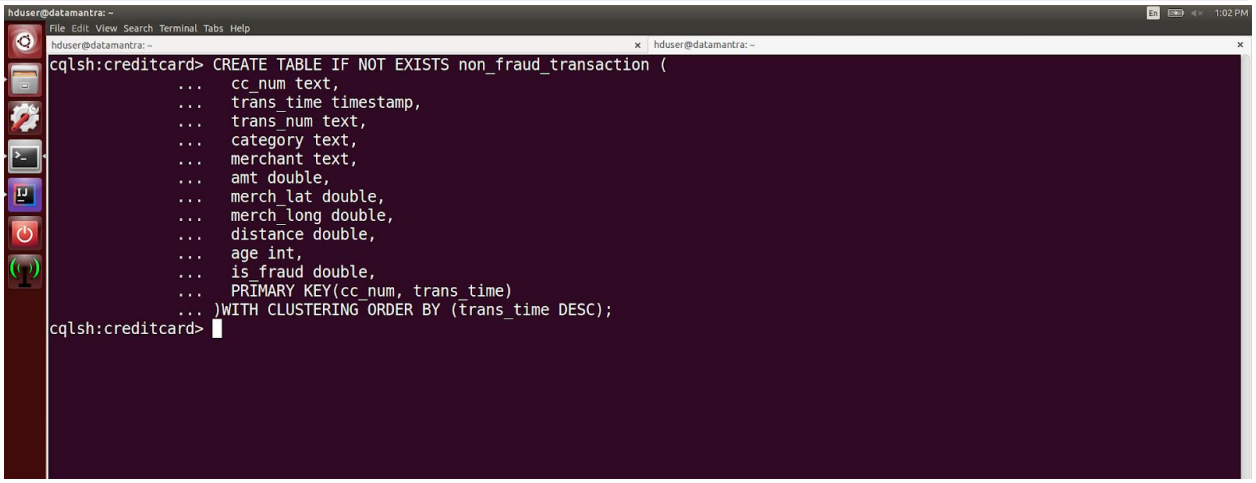
Create fraud_transaction table

```
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);
```



Create not_fraud_transaction table

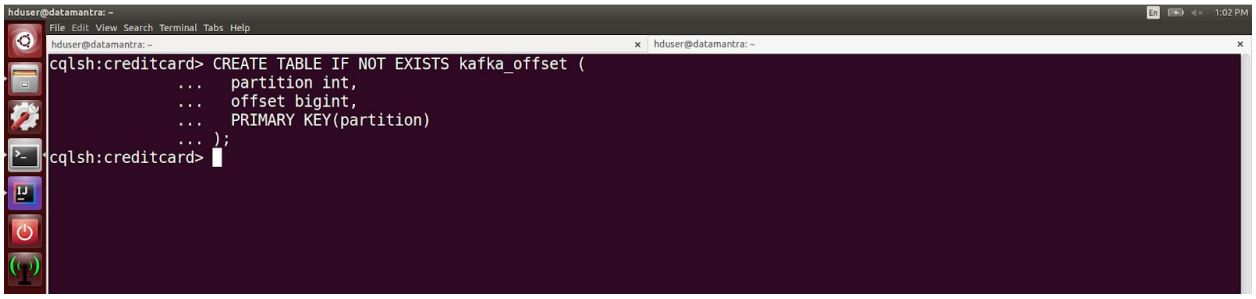
```
CREATE TABLE IF NOT EXISTS non_fraud_transaction (
  cc_num text,
  trans_time timestamp,
  trans_num text,
  category text,
  merchant text,
  amt double,
  merch_lat double,
  merch_long double,
  distance double,
  age int,
  is_fraud double,
  PRIMARY KEY(cc_num, trans_time)
)WITH CLUSTERING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



Create a kafk_offset table

Offset per partition will be saved in this table

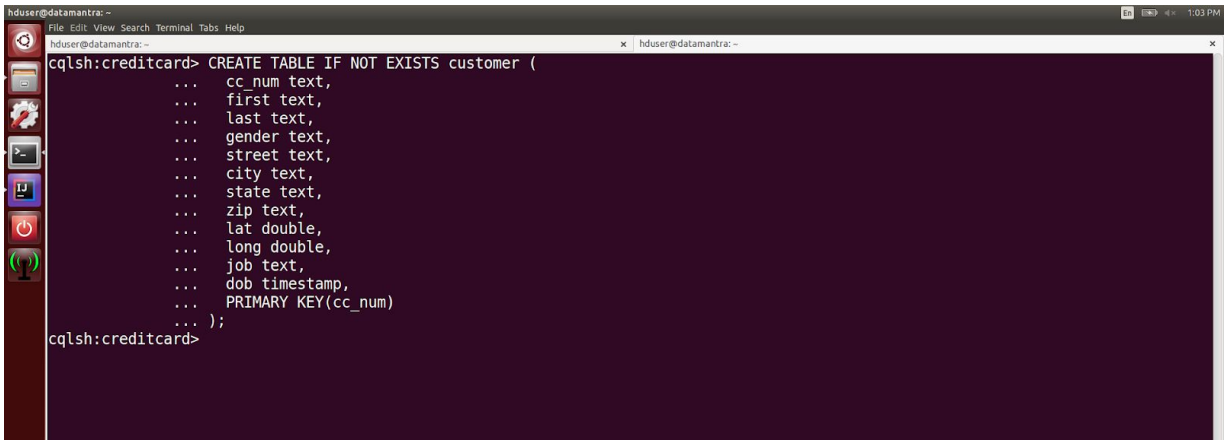
```
CREATE TABLE IF NOT EXISTS kafka_offset (
  partition int,
  offset bigint,
  PRIMARY KEY(partition)
);
```



Create a customer table

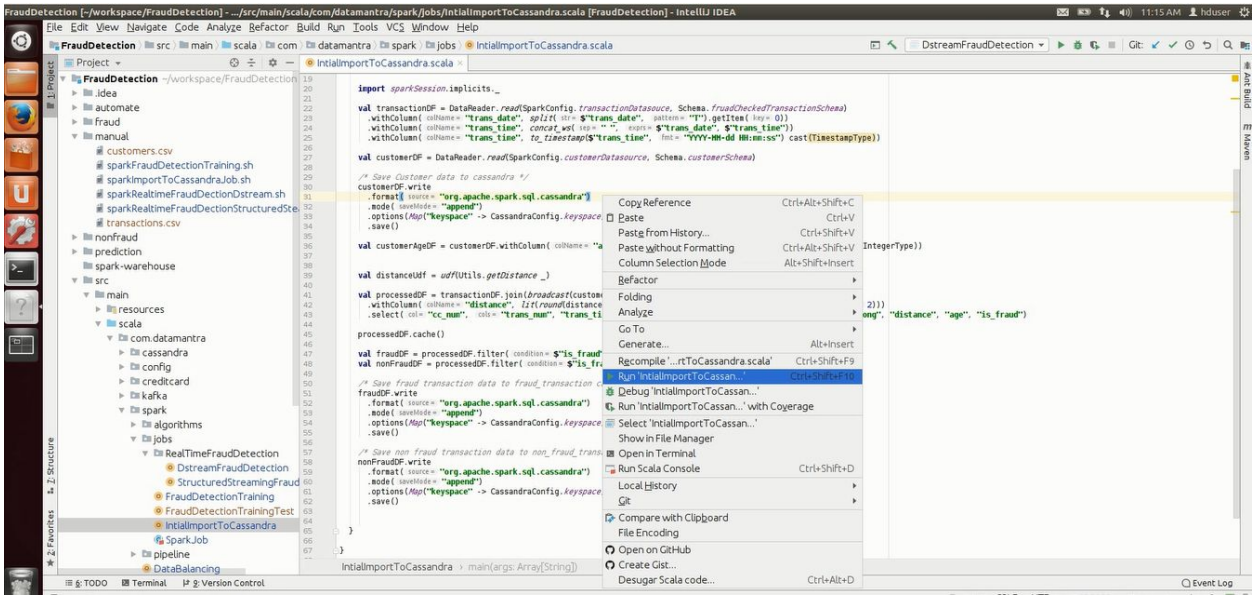
This table will have customer details

```
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));
```



Run IntialImportToCassandra.scala from IntelliJ

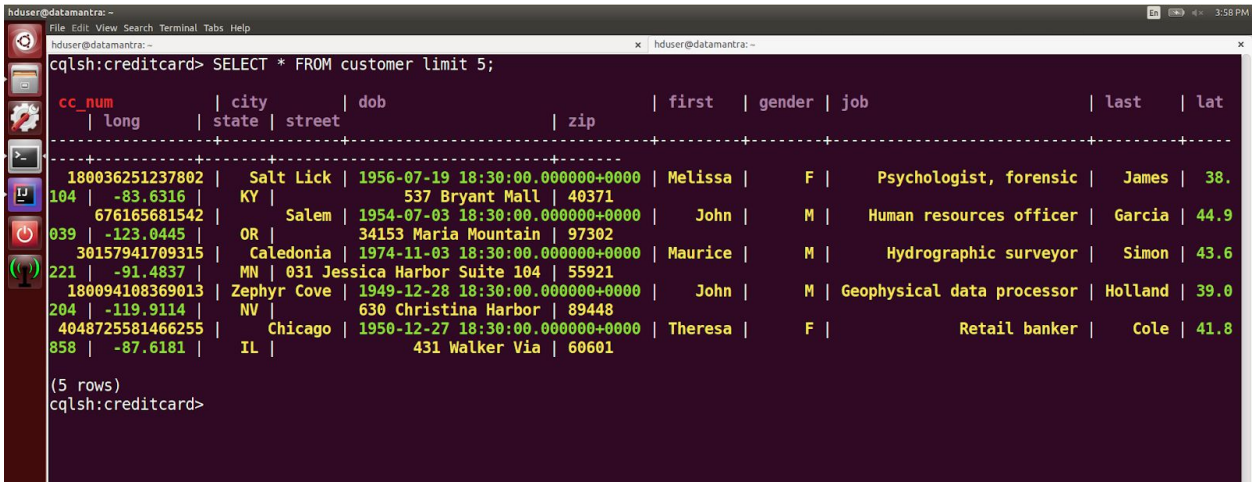
No Input parameter



Output

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

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

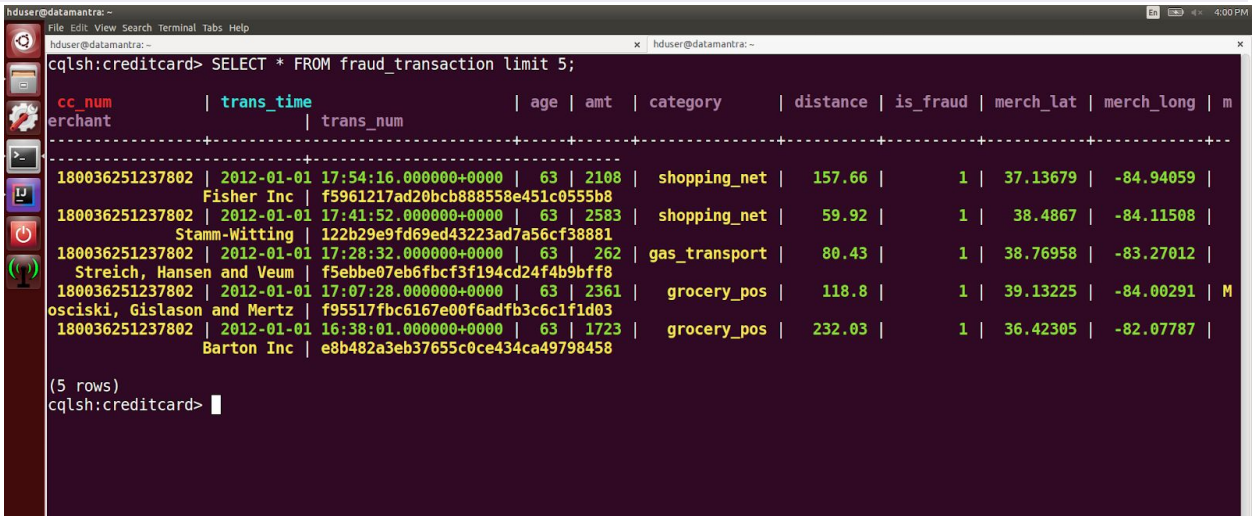


```
cqlsh:creditcard> SELECT * FROM customer limit 5;

 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;
```



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

 cc_num          | trans_time                      | age | amt   | category      | distance | is_fraud | merch_lat | merch_long | m
erchant                  | 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 | f5ebbe07eb6fbcf3f194cd24f4b9bff8
 180036251237802 | 2012-01-01 17:07:28.000000+0000 |  63 |  2361 |   grocery_pos |    118.8 |        1 |  39.13225 |  -84.00291 | M
osciski, Gislason and Mertz | f95517fbc6167e00f6adfb3c6c1f1d03
 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>
```

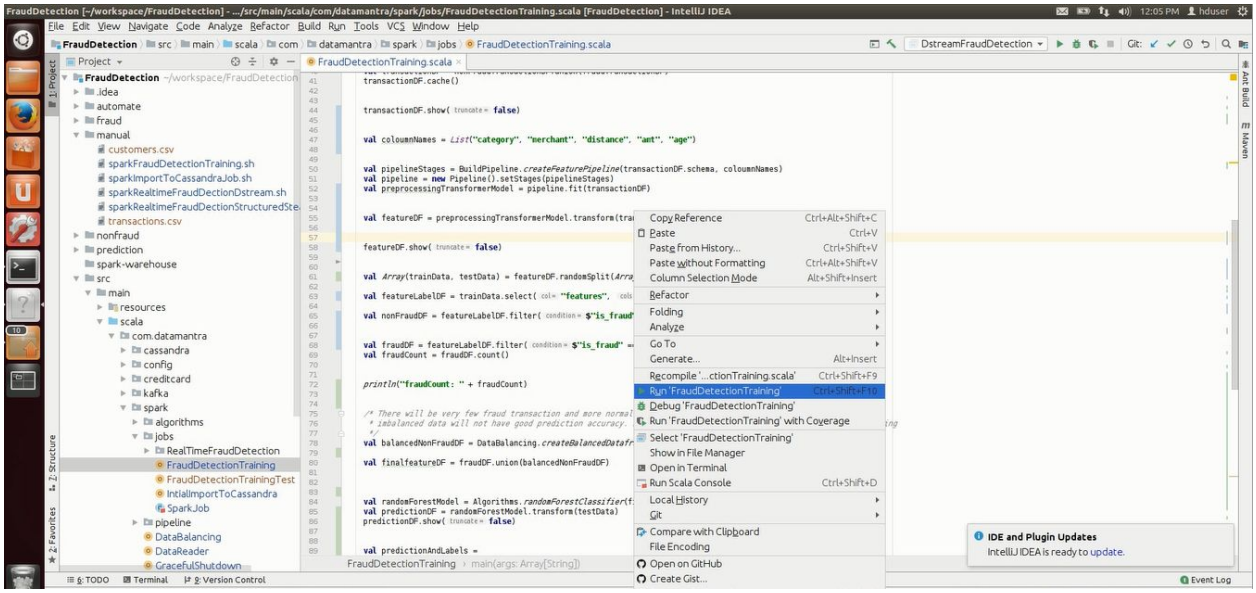
```
SELECT * FROM fraud_transaction limit 5;
```

All the data is successfully imported to Cassandra Keyspace

3. Spark Training Job

3.1 Demonstration

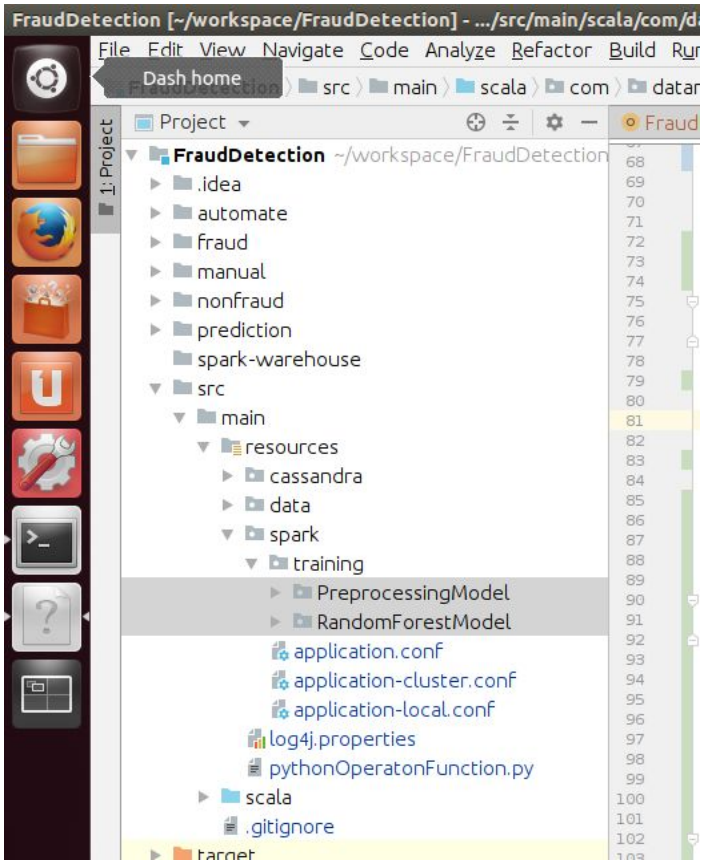
Run FraudDetectionTraining.scala from IntelliJ
No Input Parameters
Right-click on FraudDetectionTraining.scala file and Click Run



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
```



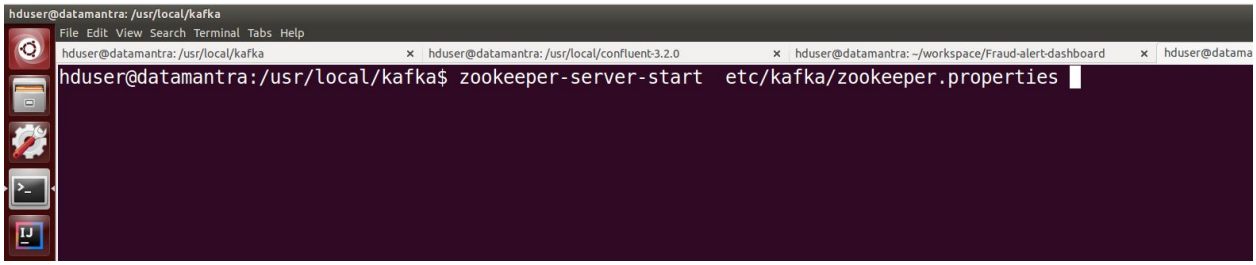
Preprocessing Model and Random Forest Model is saved to the File System

4. Spark Streaming

4.1 Demonstration

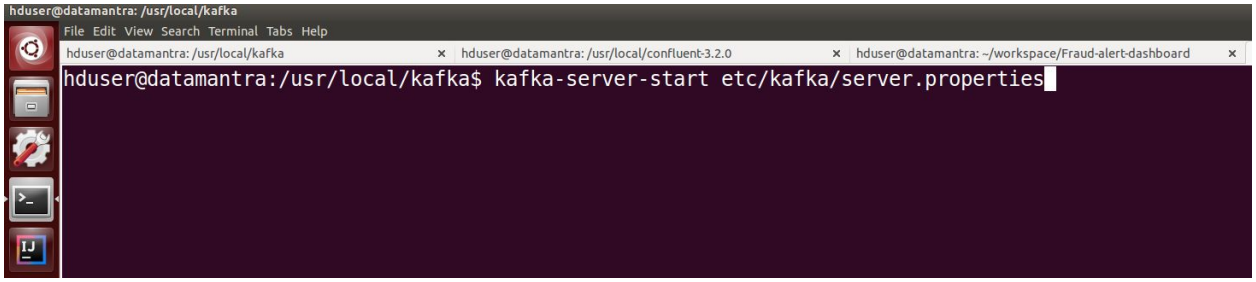
Start Zookeeper

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



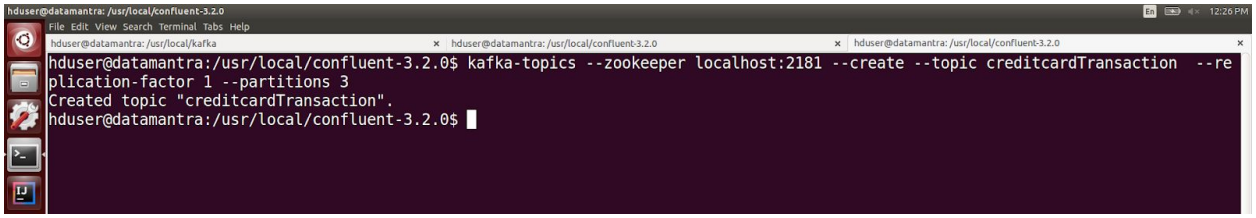
Start Kafka Server

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



Create transaction topic

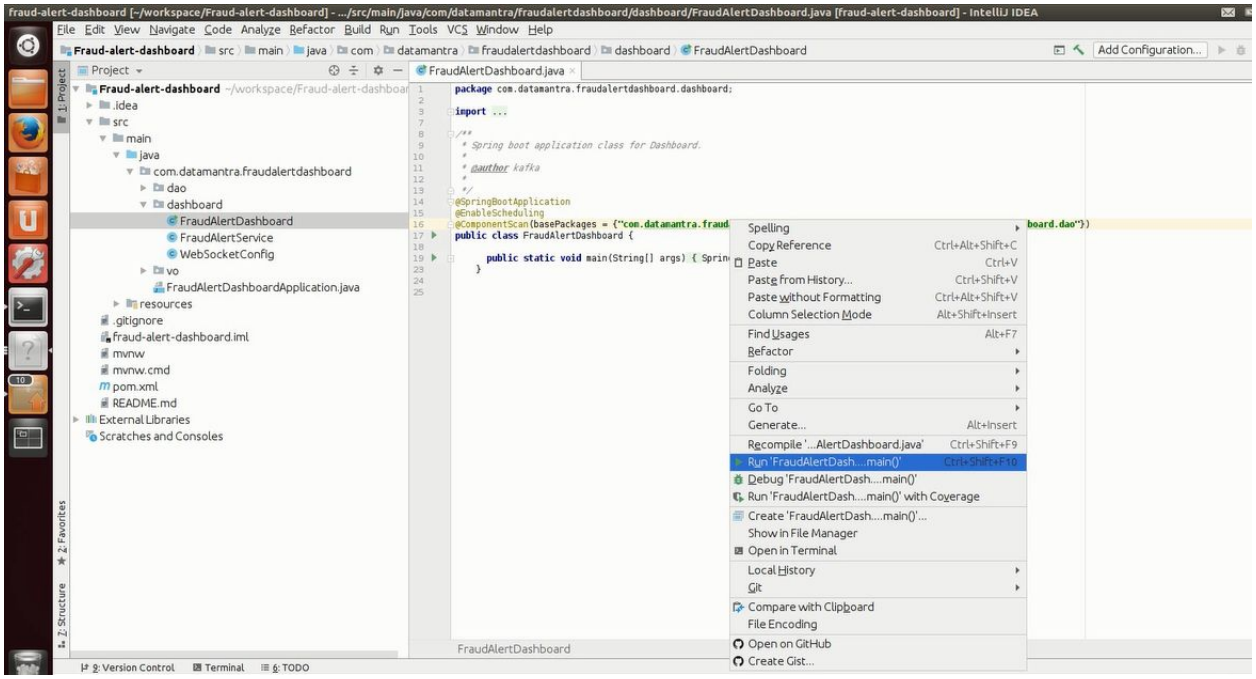
```
kafka-topics --zookeeper localhost:2181 --create --topic
creditcardTransaction  --replication-factor 1 --partitions 3
```

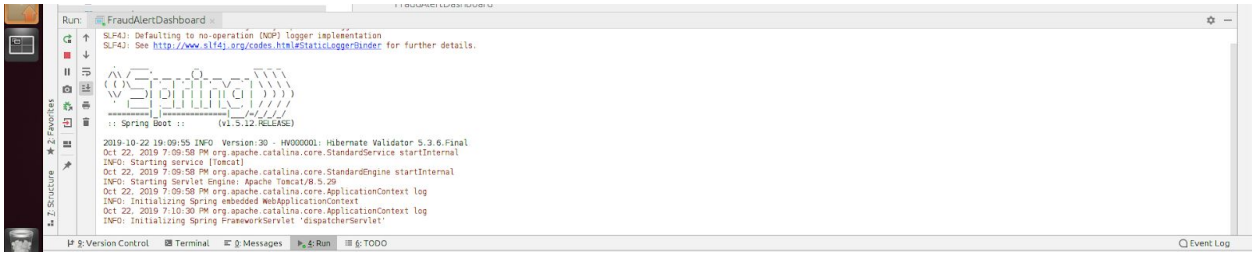


Run Fraud-alert-dashboard project from IntelliJ

No input parameters

Right Click on FraudAlertDashboard.java file and Run

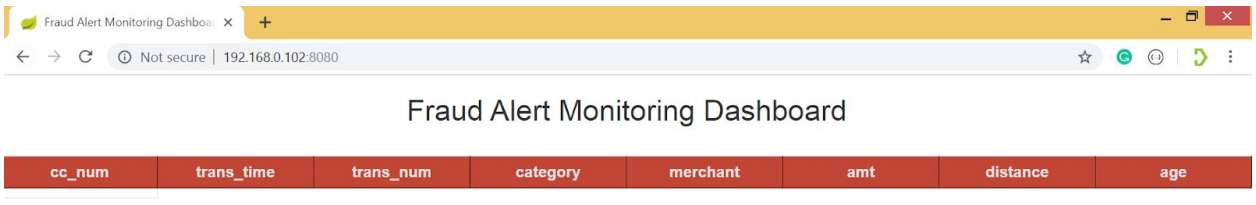




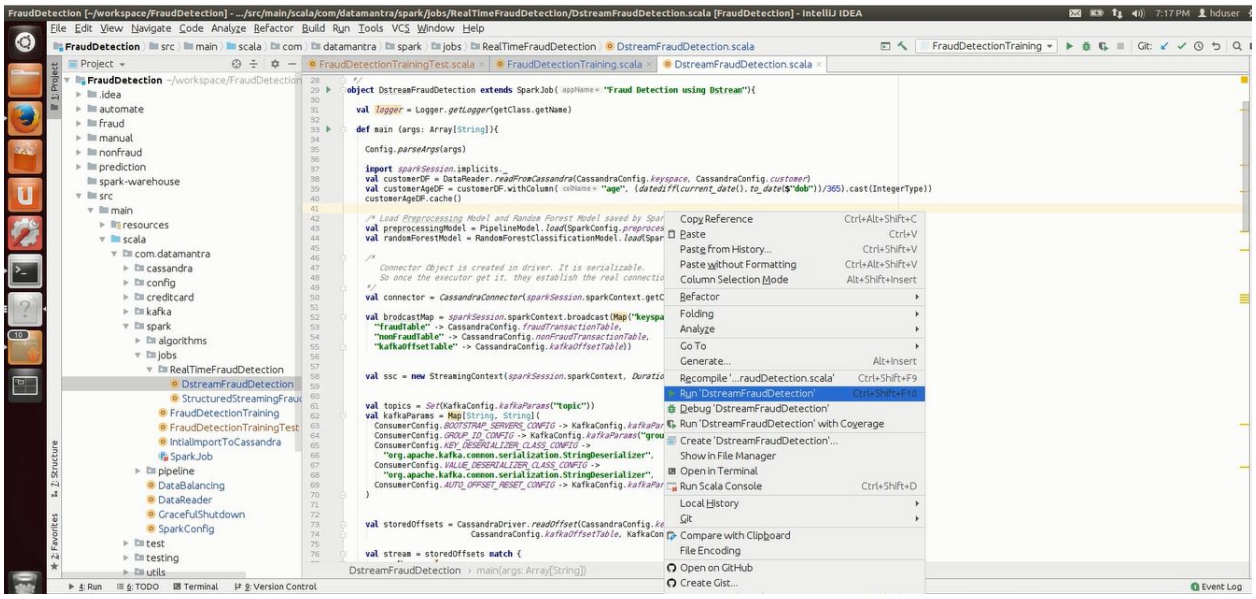
Output

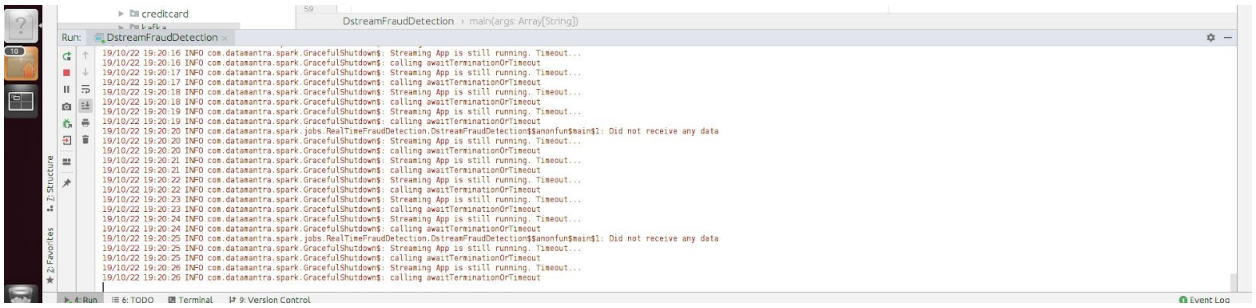
The fraud alert dashboard web server is up and running. Verify this by accessing through the web browser

http://<ubuntu-ipadd>:8080



Run Spark Streaming Job from Intellij



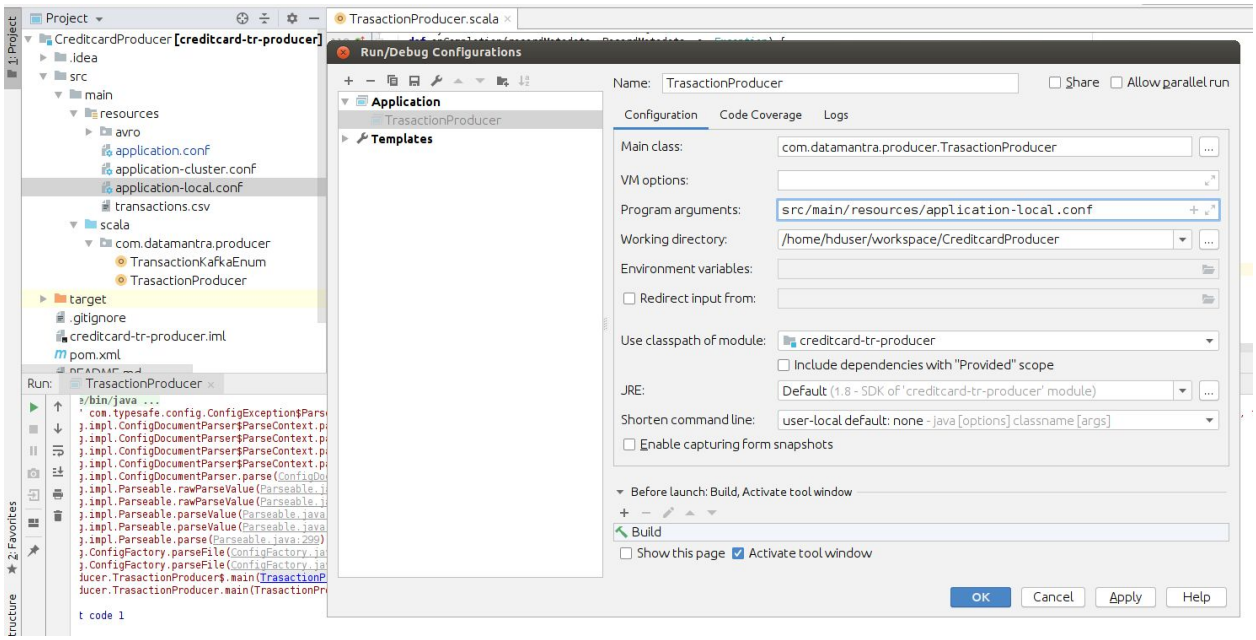


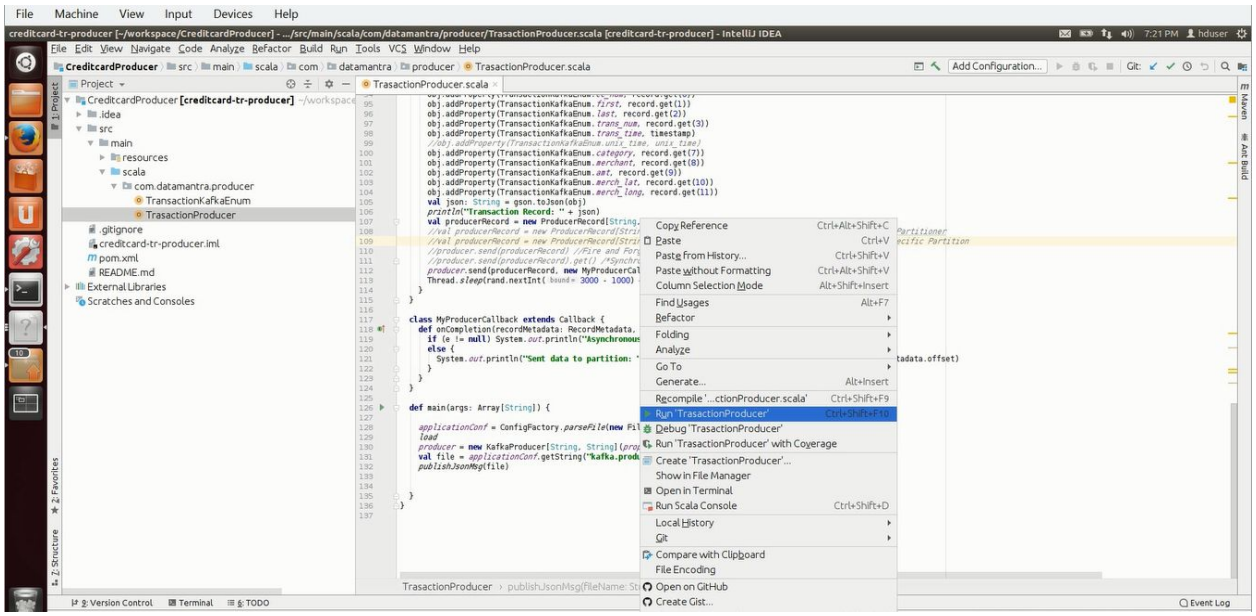
Run CreditcardProducer from IntelliJ

https://github.com/pramoddatamantra/CreditcardProducer

Input parameter is the configuration file

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

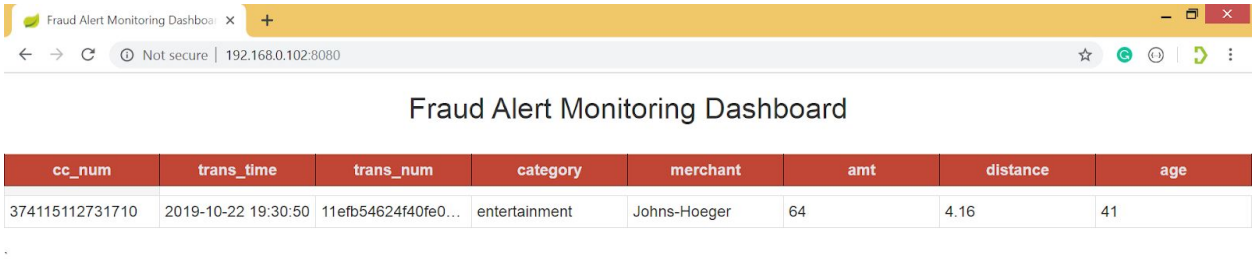

Project
CreditcardProducer [creditcard-tr-producer]
.idea
src
main
resources
avro
application.conf
application-cluster.conf
application-local.conf
transactions.csv
scala
com.datamantra.producer
TransactionKafkaEnum
TrasactionProducer
target
.gitignore
creditcard-tr-producer.iml
pom.xml
TrasactionProducer.scala
Run/Debug Configurations
Name: TrasactionProducer
Share
Allow parallel run
Application
TrasactionProducer
Templates
Configuration
Code Coverage
Logs
Main class: com.datamantra.producer.TrasactionProducer
VM options:
Program arguments: src/main/resources/application-local.conf
Working directory: /home/hduser/workspace/CreditcardProducer
Environment variables:
Redirect input from:
Use classpath of module: creditcard-tr-producer
Include dependencies with "Provided" scope
JRE: Default (1.8 - SDK of 'creditcard-tr-producer' module)
Shorten command line: user-local default: none - java [options] classname [args]
Enable capturing form snapshots
Before launch: Build, Activate tool window
Build
Show this page
Activate tool window
OK
Cancel
Apply
Help
Run:
TrasactionProducer


File Machine View Input Devices Help
creditcard-tr-producer [~/workspace/CreditcardProducer] - .../src/main/scala/com/datamantra/producer/TrasactionProducer.scala [creditcard-tr-producer] - IntelliJ IDEA
Copy Reference
Paste
Paste from History...
Paste without Formatting
Column Selection Mode
Find Usages
Refactor
Folding
Analyze
Go To
Generate...
Recompile '...ctionProducer.scala'
Run 'TrasactionProducer'
Debug 'TrasactionProducer'
Run 'TrasactionProducer' with Coverage
Create 'TrasactionProducer'...
Show in File Manager
Open in Terminal
Run Scala Console
Local History
Git
Compare with Clipboard
File Encoding
Open on GitHub
Create Gist...


Run:
TrasactionProducer
/usr/lib/jvm/java-8-oracle/bin/java ...
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/hduser/.m2/repository/org/slf4j/slf4j-log4j12/1.7.21/slf4j-log4j12-1.7.21.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/hduser/.m2/repository/org/slf4j/slf4j-nop/1.5.3/slf4j-nop-1.5.3.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/hduser/.m2/repository/org/slf4j/slf4j-jdk14/1.5.6/slf4j-jdk14-1.5.6.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
log4j:WARN No appenders could be found for logger (org.apache.kafka.clients.producer.ProducerConfig).
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
Transaction Record: {"cc_num":"180094108569013","first":"John","last":"Holland","trans_num":"80f5177be11f0bcd768e06a0b1b294c8","trans_time":"2019-10-22 19:30:08","category":"personal_care","merchant":"Hills-Boyer","amt":"64","merch_lat":"39.011566",
Sent data to partition: 0 and offset: 0
Transaction Record: {"cc_num":"4368593032190508","first":"Carla","last":"Fleming","trans_num":"7933d389bf18ef8a11a8a3014da6e6bc6c","trans_time":"2019-10-22 19:30:10","category":"gas_transport","merchant":"Friesen-DAmore","amt":"133","merch_lat":"40.14
Sent data to partition: 0 and offset: 1
Transaction Record: {"cc_num":"4361355512072","first":"Matthew","last":"Nelson","trans_num":"1467c318b5d73d22d674lf575db42f3f","trans_time":"2019-10-22 19:30:13","category":"entertainment","merchant":"Larson-Moen","amt":"119","merch_lat":"47.297797
Sent data to partition: 0 and offset: 2
Transaction Record: {"cc_num":"4037295225657274","first":"Antonio","last":"Garcia","trans_num":"4a3848719d72daaa32c938ce43e0eb03","trans_time":"2019-10-22 19:30:15","category":"shopping_pos","merchant":"Lynch Ltd","amt":"62","merch_lat":"40.078781",
Sent data to partition: 0 and offset: 3
Transaction Record: {"cc_num":"4515092388857440","first":"Timothy","last":"Silva","trans_num":"02d27e94f279e1013a4c323eb8bb6811","trans_time":"2019-10-22 19:30:18","category":"shopping_pos","merchant":"Baumbach, Strosin and Nicolas","amt":"198","mer
Sent data to partition: 0 and offset: 4

Output

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



5. Airflow Automation

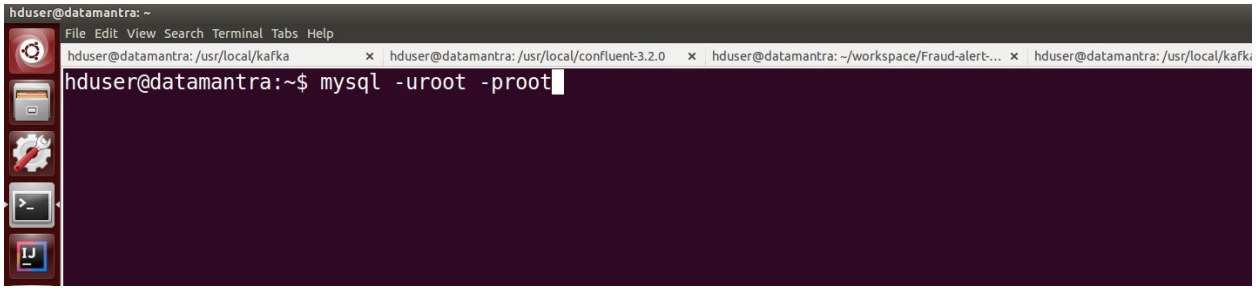
5.1 Configure Airflow

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

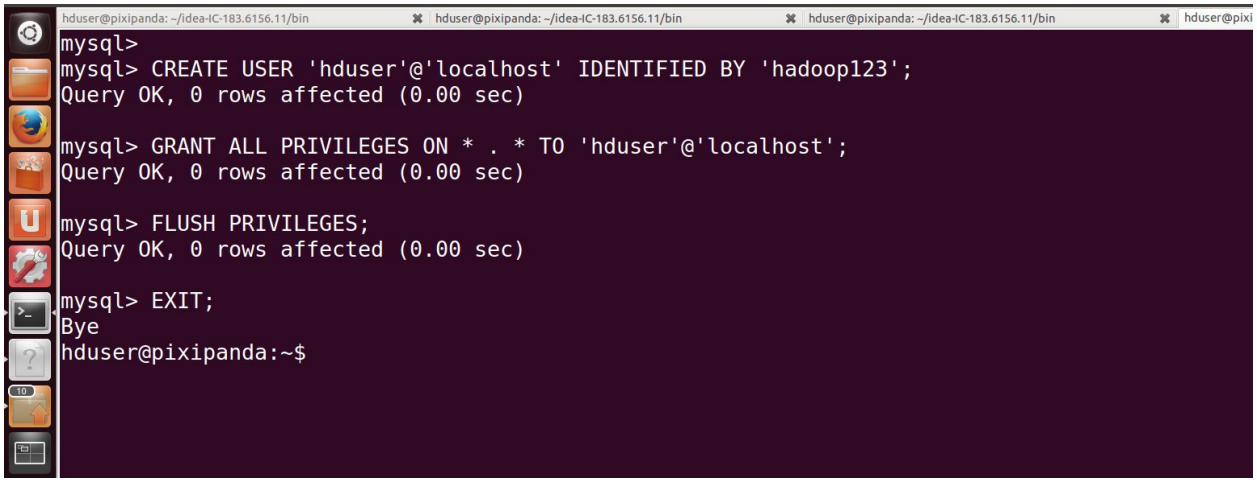
Create a new user and database in Mysql
Login to Mysql with root credentials

```
mysql -uroot -proot
```



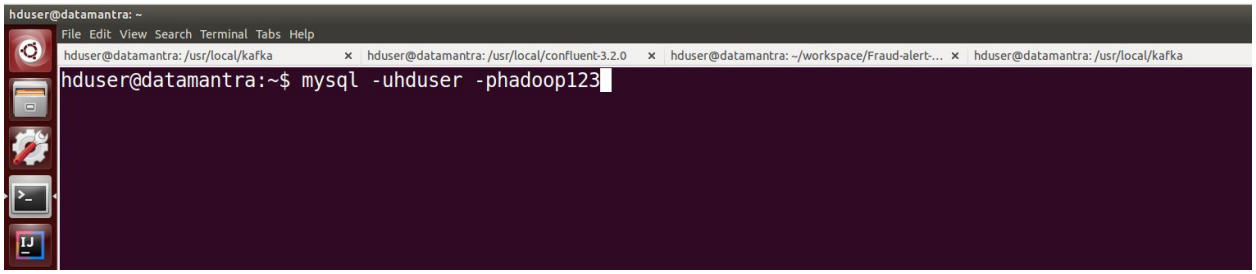
Create hduser with hadoop123 as password

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



Login as hduser

```
mysql -uhduser -phadoop123
```



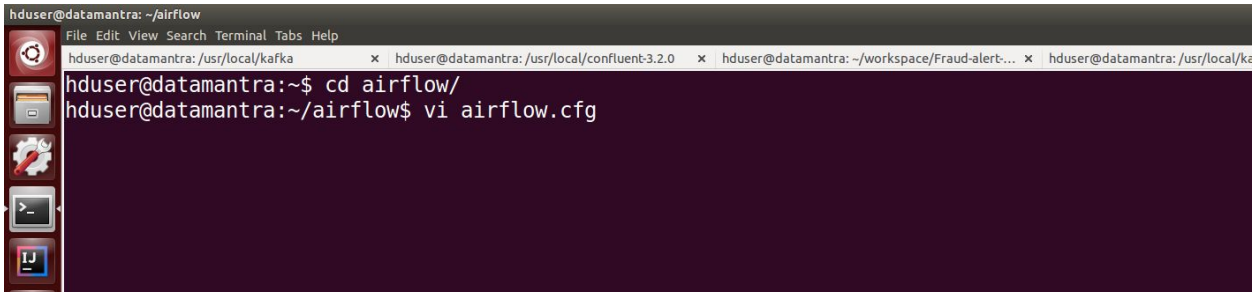
Create Airflow database and exit

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

5.1.2 Modify airflow.cfg

Modify the airflow.cfg file to use MySQL database

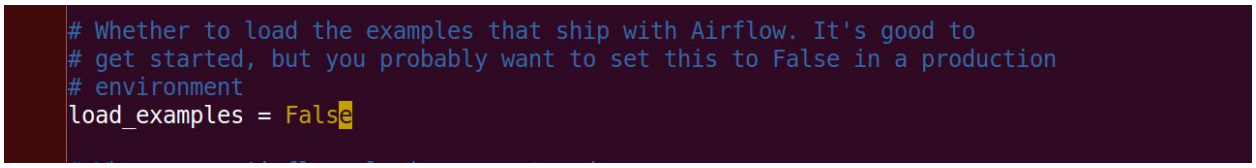
```
cd ~/airflow
vi airflow.cfg
```



Modify following fields, save and exit

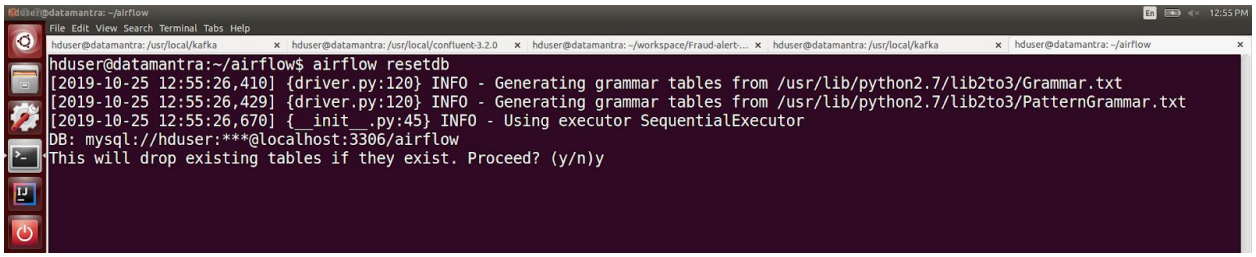
```
sql_alchemy_conn = mysql://hduser:hadoop123@localhost:3306/airflow
load_examples = False
```

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.



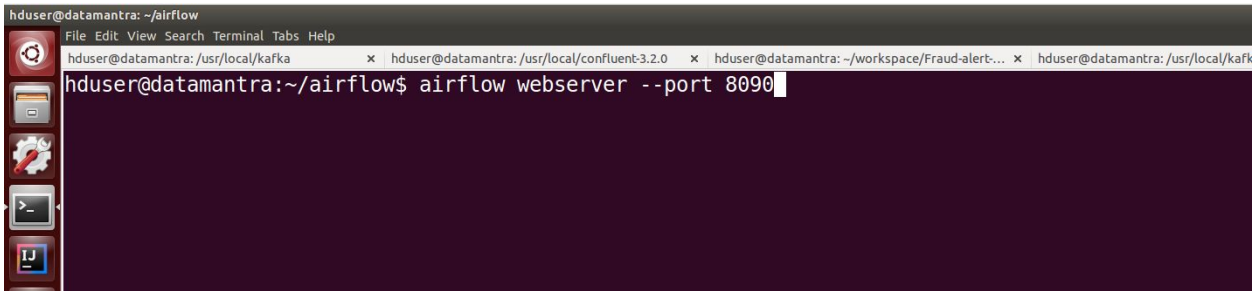
Rest Airflow Database

`airflow resetdb`



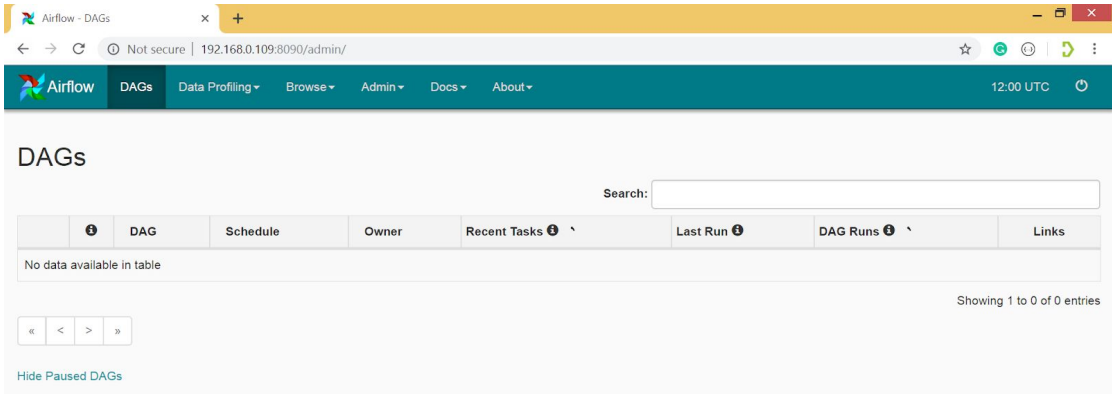
Start Airflow Webserver

`airflow webserver --port 8090`



Access Airflow Web Server

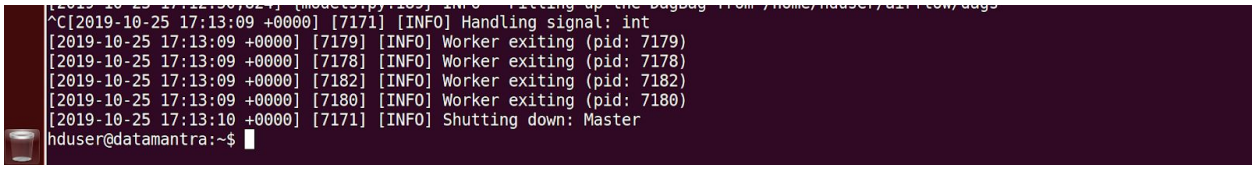
Access `https://<ubunt_ipaddress>:8080` to verify whether the airflow web server is up and running



Airflow is configured successfully

Stop Airflow Webserver

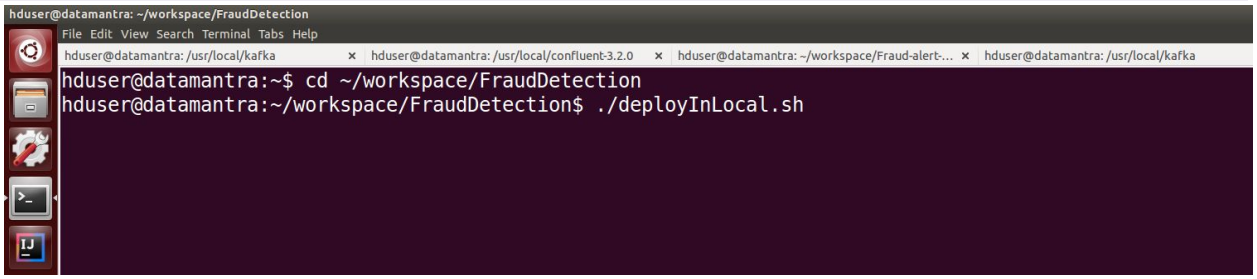
Stop Airflow Webserver by pressing CTRL C on the terminal where it was started. This was just to test whether Airflow is working. We will start again when we start automation



5.2 Build all Projects

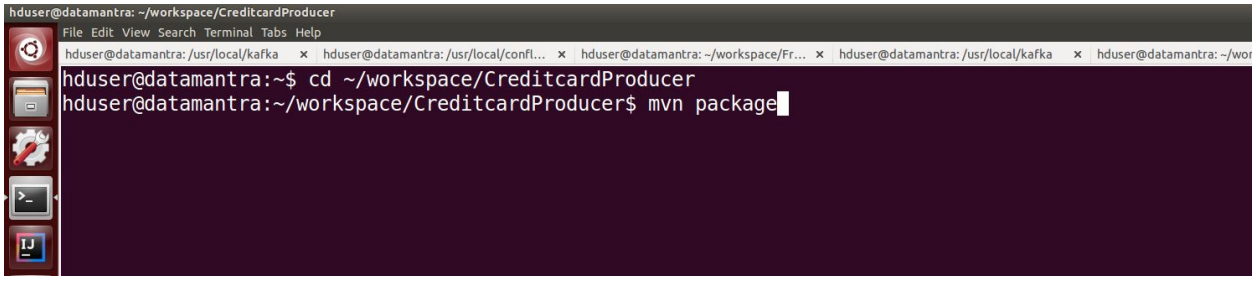
Build FraudDetectoin Project

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



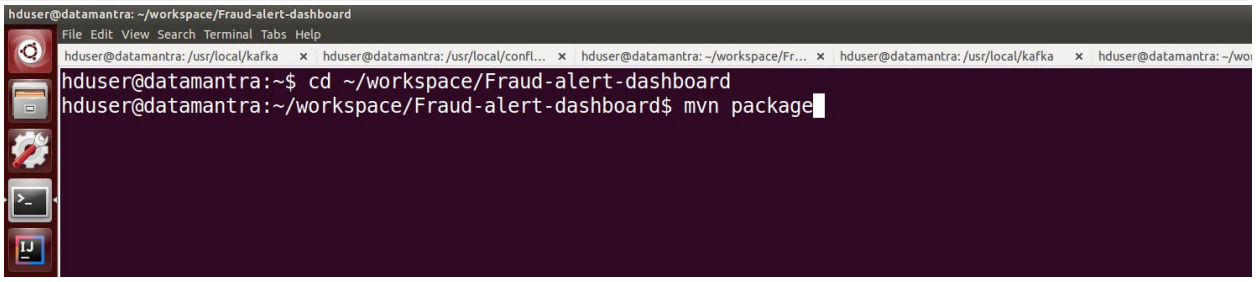
Build Creditcard Producer Project

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



Build Fraud-Alert-Dashboard Project

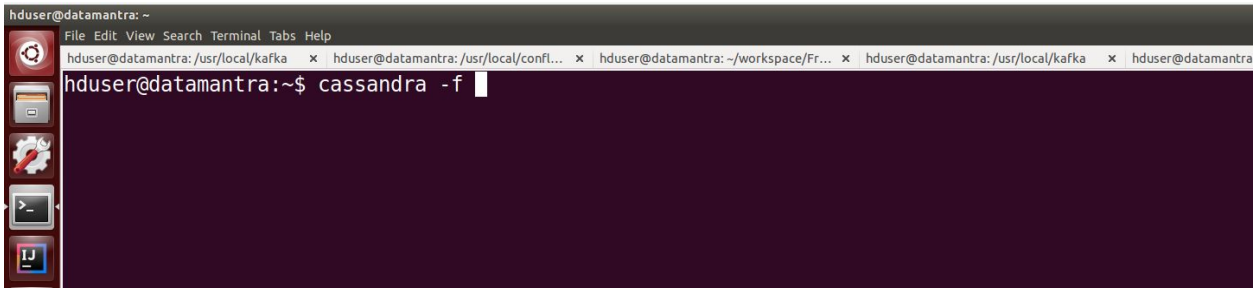
```
cd ~/workspace/Fraud-alert-dashboard
mvn package
```



5.3 Start all the Servers

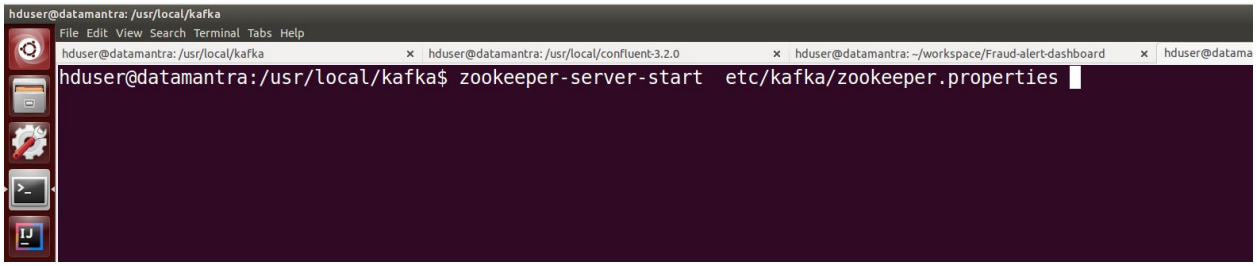
Start Cassandra Server

```
    cassandra -f
```



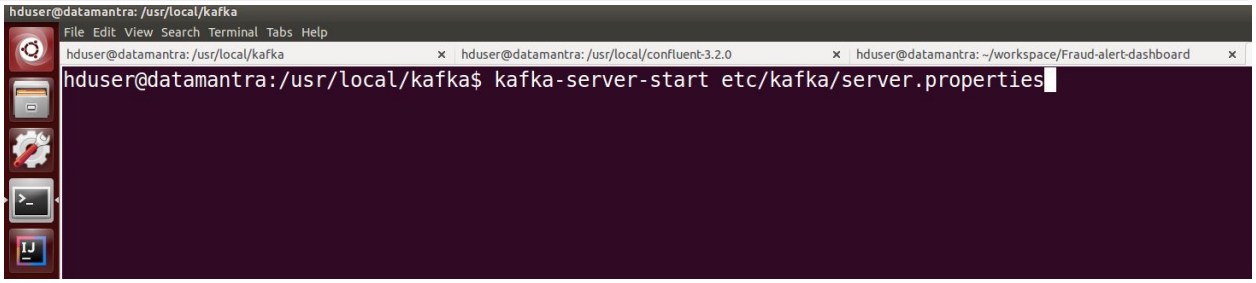
Start Zookeeper Server

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



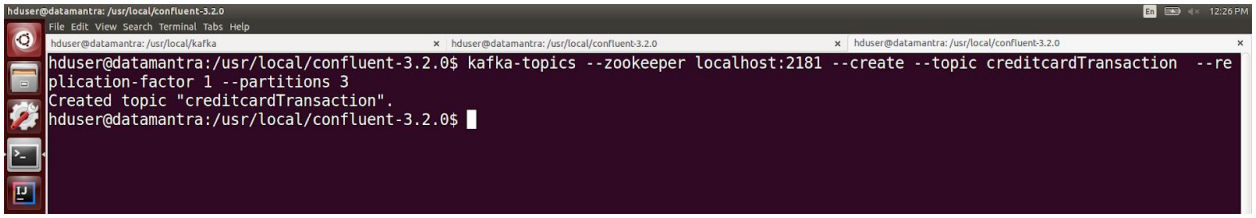
Start Kafka Server

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



Create transaction topic

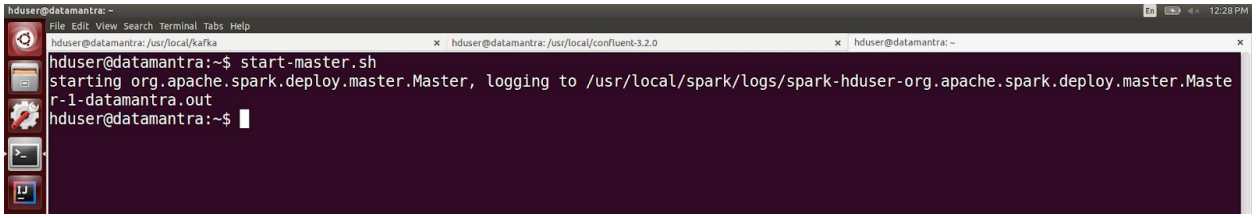
```
kafka-topics --zookeeper localhost:2181 --create --topic
creditcardTransaction  --replication-factor 1 --partitions 3
```



Spark Cluster Setup

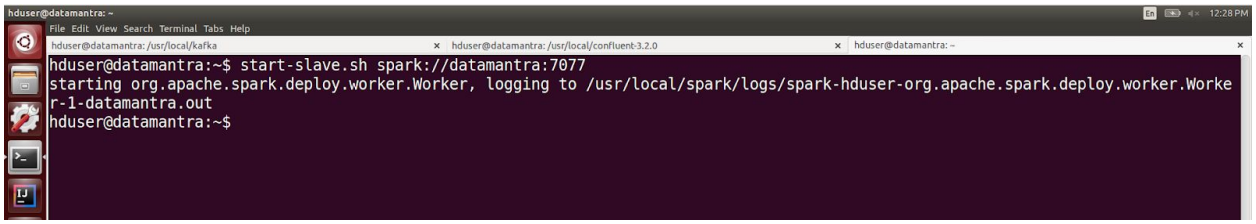
Start Spark Master

```
start-master.sh
```



Start Spark Slave

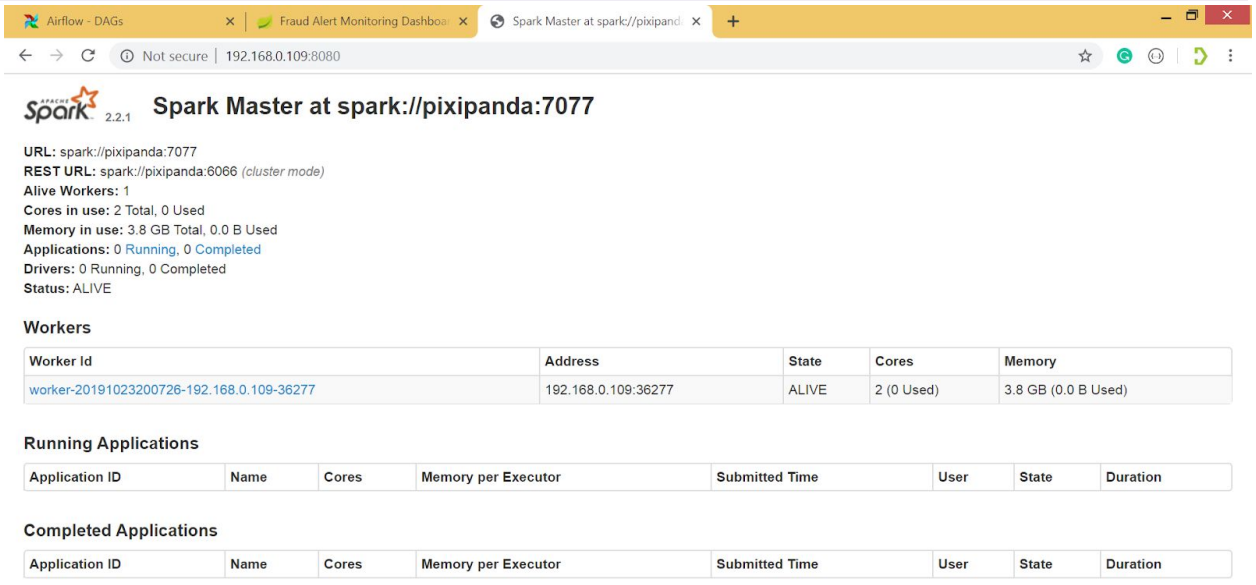
```
start-slave.sh spark://datamantra:7077
```



Access Spark Web UI

Check whether Spark Cluster is up and running

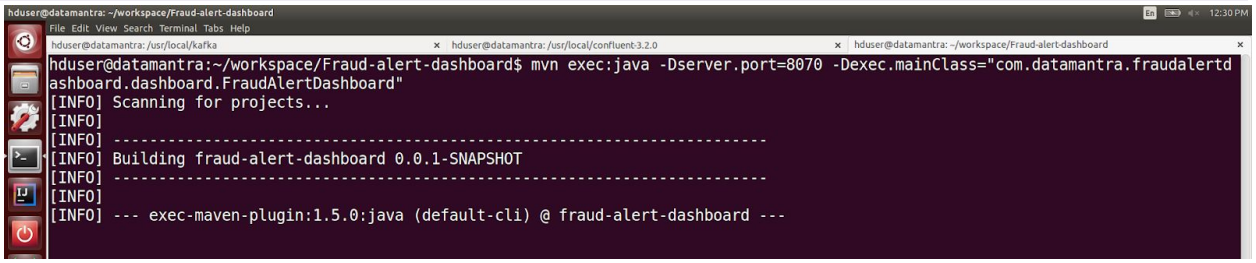
```
http://ubuntu_ip:8080/
```



Start the dashboard for visualization

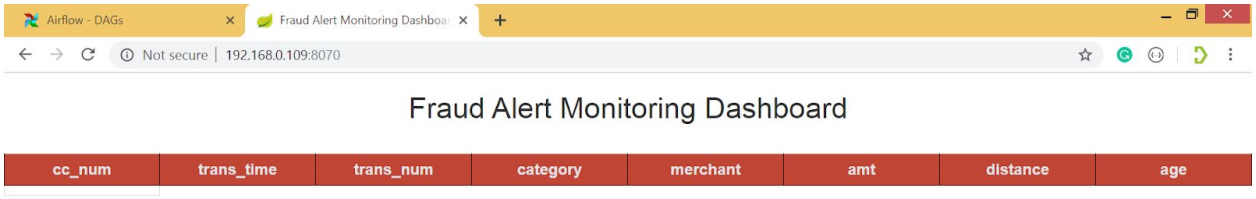
```
cd ~/workspace/Fraud-alert-dashboard/
mvn exec:java -Dserver.port=8070
```

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



Access Dashboard Web UI

```
http://ubuntu_ip:8070
```

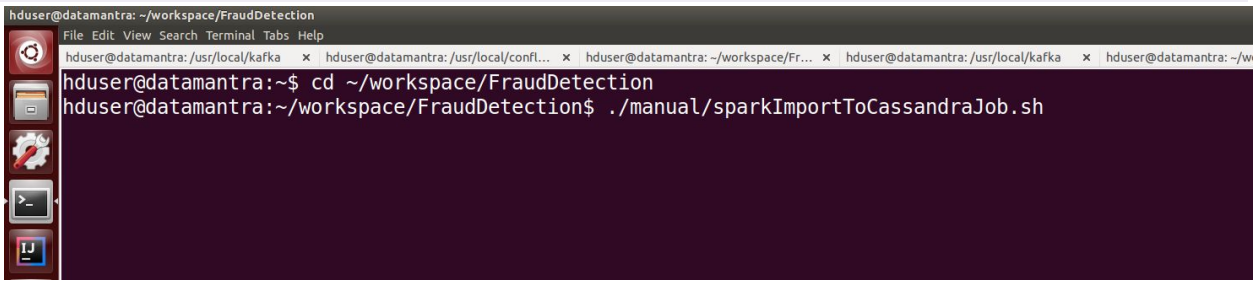


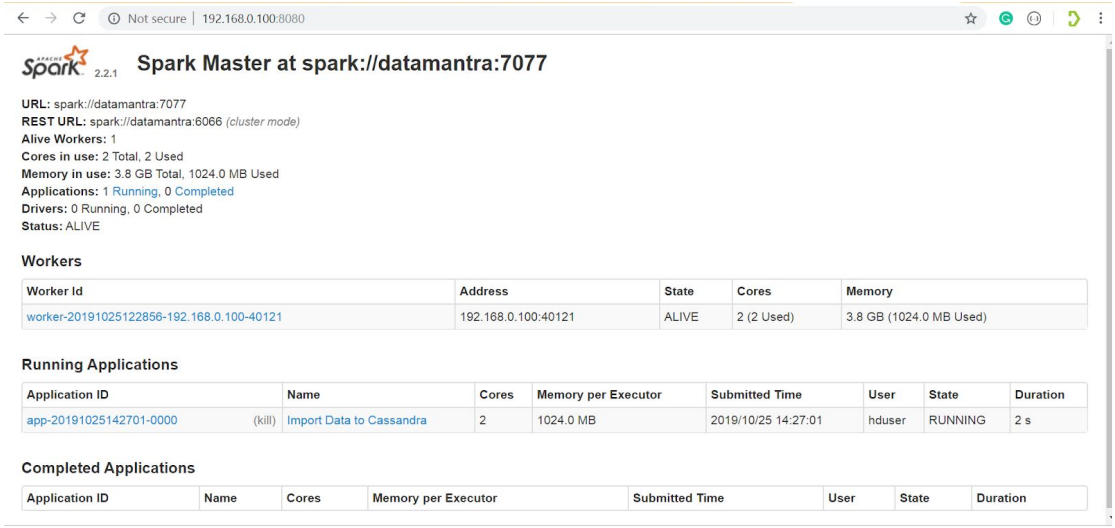
5.4 Airflow Automation

Start Spark Import Job

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

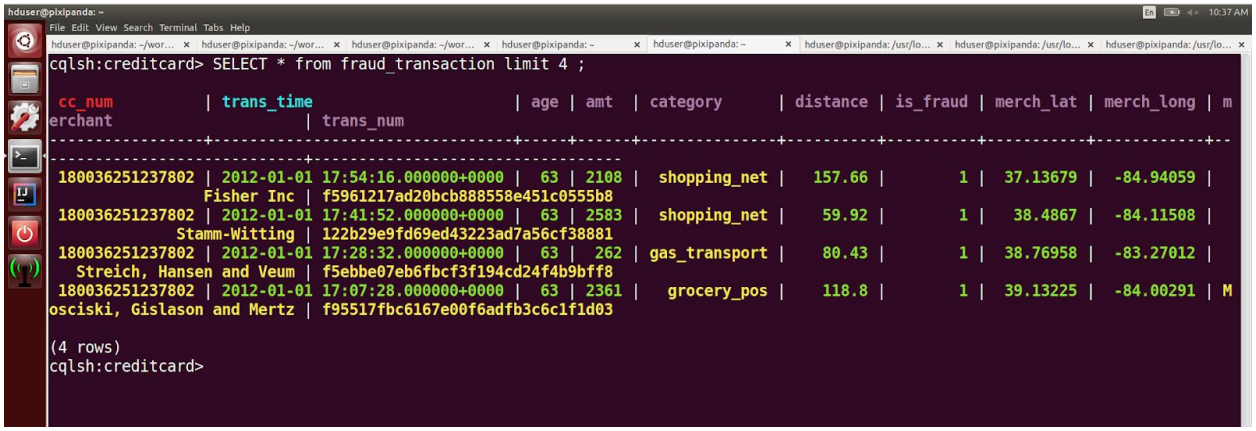
```
cd ~/workspace/FraudDetection
./manual/sparkImportToCassandraJob.sh
```





Output

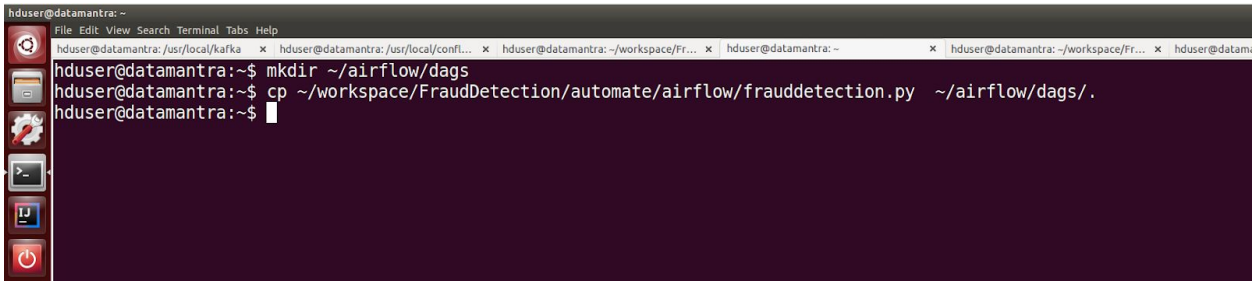
Transaction data and customer data is imported to Cassandra Keyspace



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

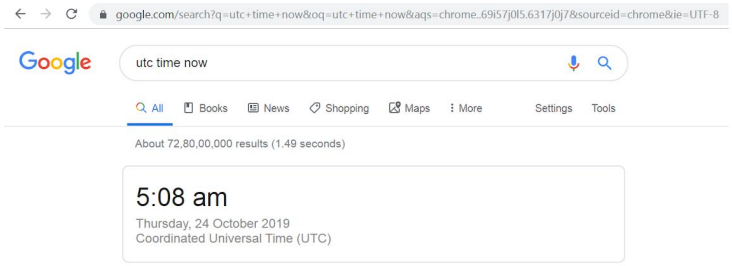
```
mkdir ~/airflow/dags
cp ~/workspace/FraudDetection/automate/airflow/frauddetection.py
~/airflow/dags/.
```



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/frauddetection.py
```

Reset airflow database

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

```
airflow resetdb
```

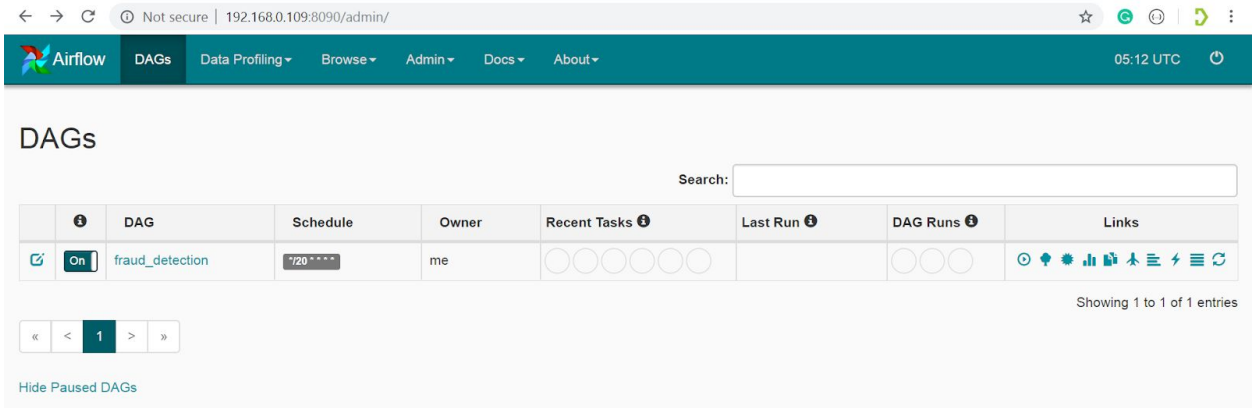
Start Airflow Webserver

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

Access Airflow Webserver

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

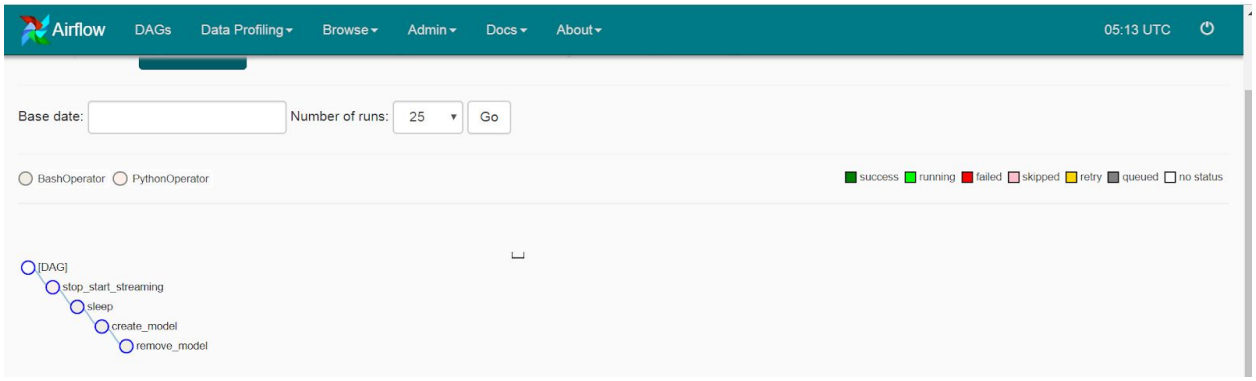
Click on OFF button to set it ON



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. Seep for 5 seconds(This step is optional)

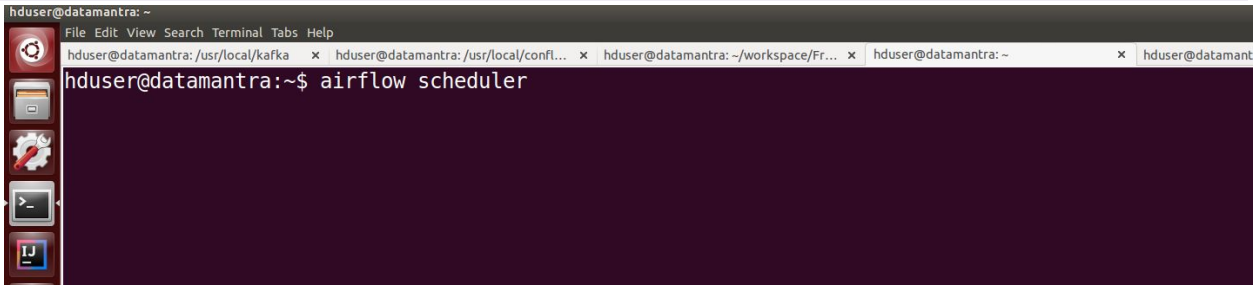
4. Stop and Start Spark Streaming Job



Start Airflow Scheduler

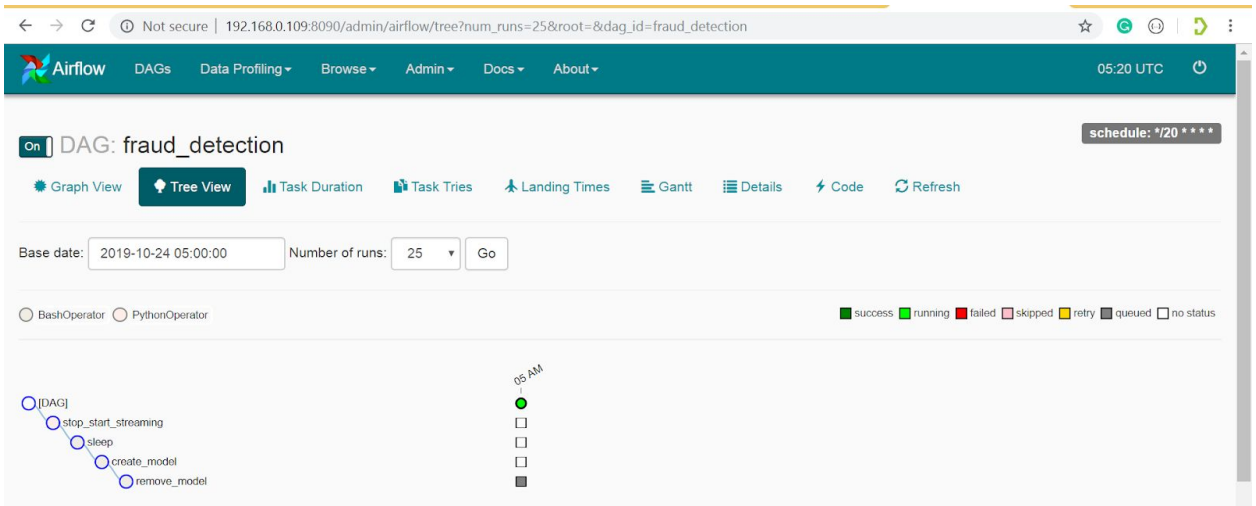
Airflow Scheduler is the one which will actually kick start your automation

```
airflow scheduler
```

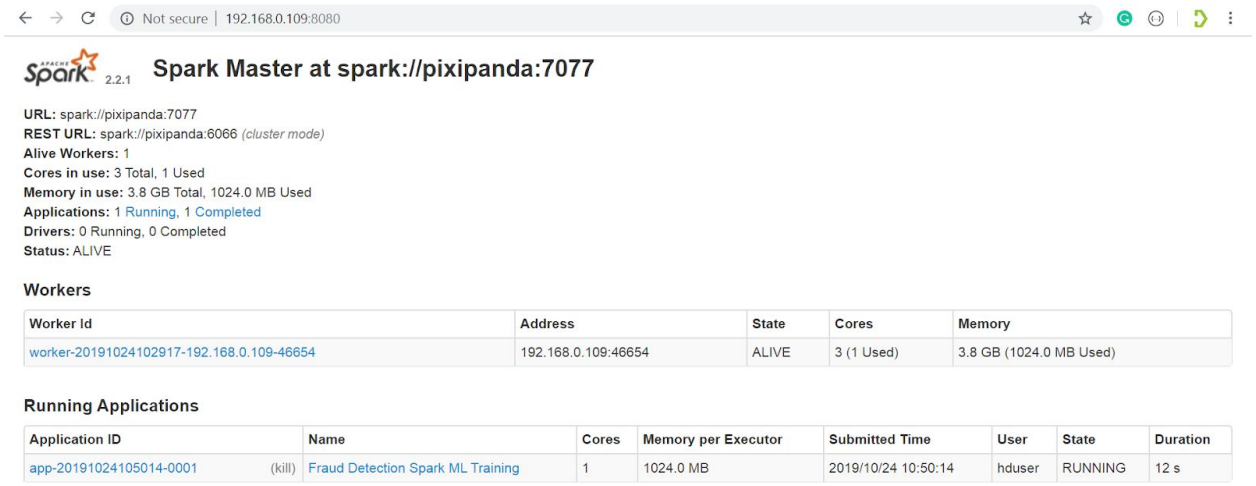


Output

Automation start at 5:20 am UTC



Spark ML Job Scheduled



Not secure | 192.168.0.109:8080

Spark 2.2.1 **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

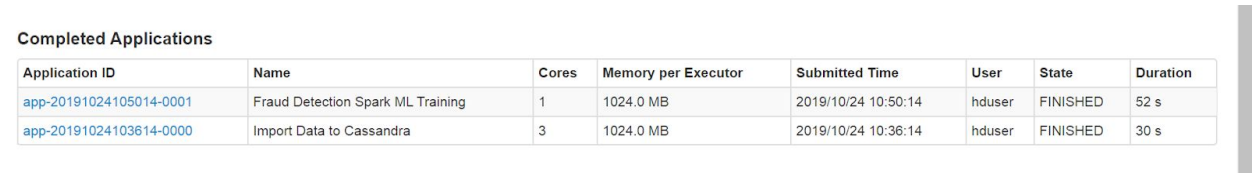
Workers

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

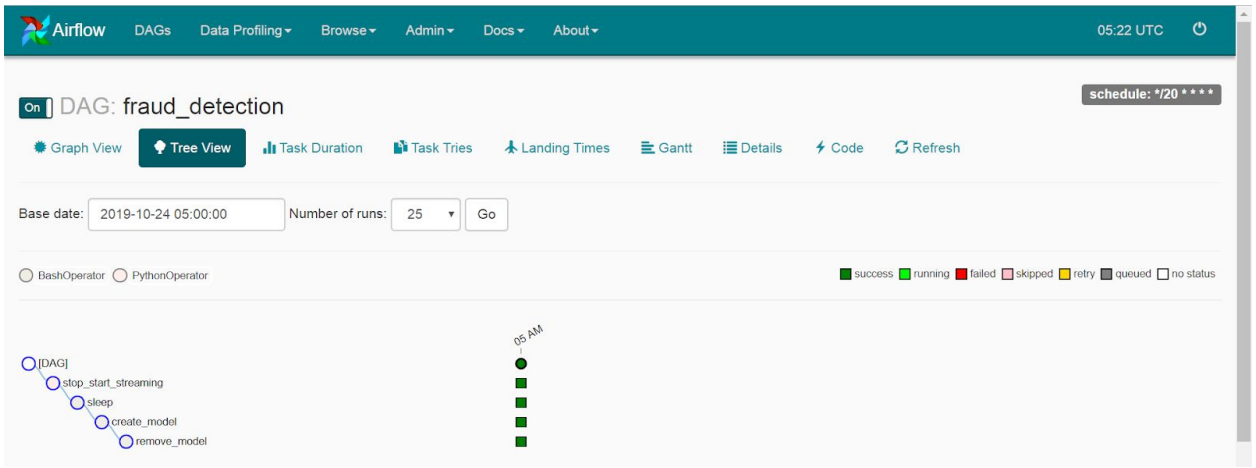


Completed Applications

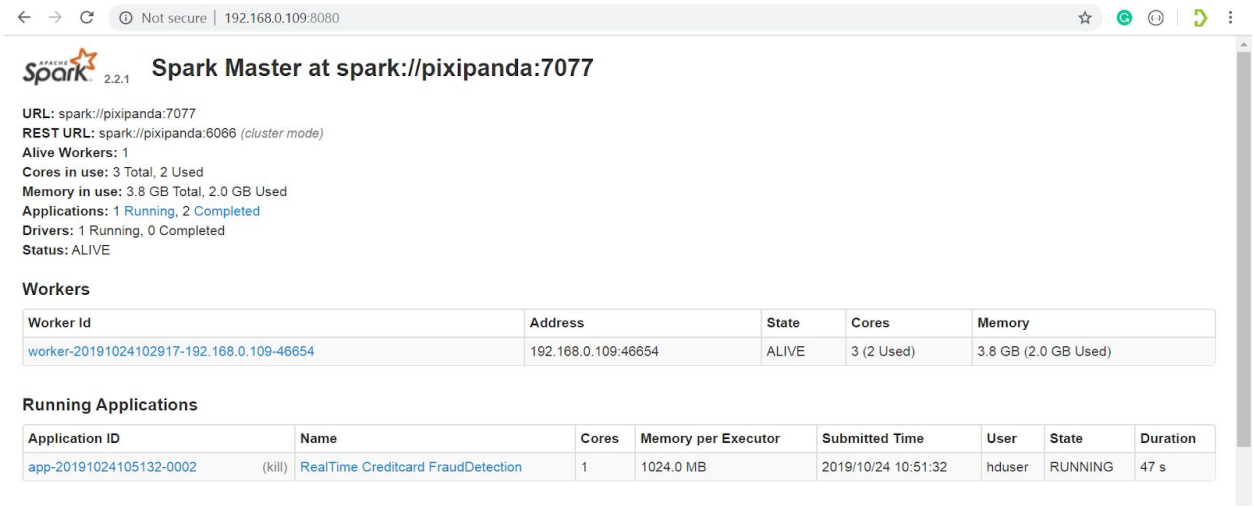
Application ID	Name	Cores	Memory per Executor	Submitted Time	User	State	Duration
app-20191024105014-0001	Fraud Detection Spark ML Training	1	1024.0 MB	2019/10/24 10:50:14	hduser	FINISHED	52 s
app-20191024103614-0000	Import Data to Cassandra	3	1024.0 MB	2019/10/24 10:36:14	hduser	FINISHED	30 s

All automation steps completed.

This means Spark Streaming is up and running



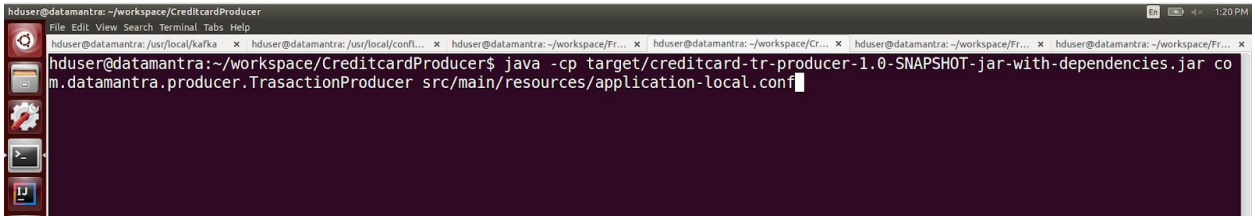
Spark Streaming Job is up and running



Start Kafka Producer

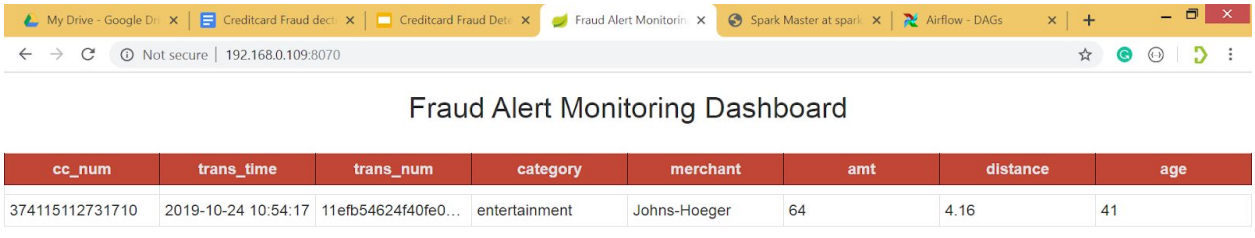
Spark Streaming Job is up and running, now we will start Kafka Producer

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



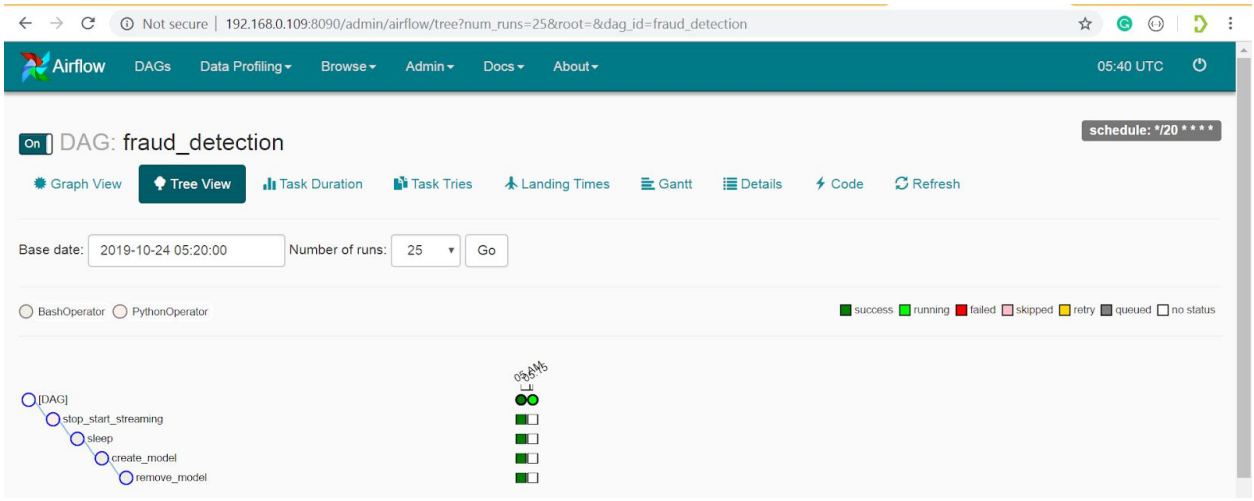
Output

A fraud alert is displayed on the dashboard



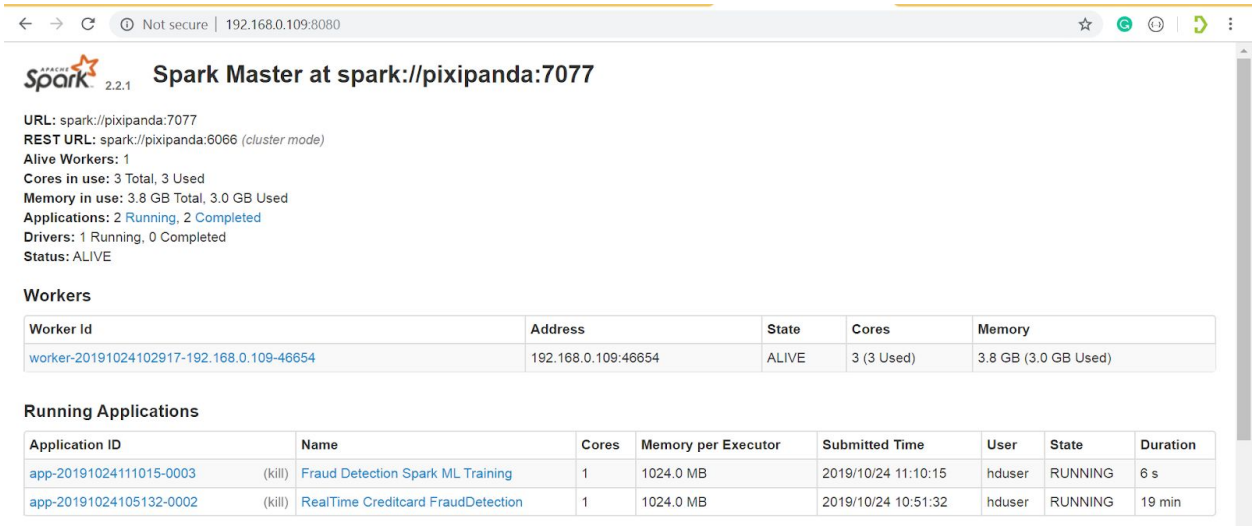
Next set of automation

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



Spark ML Job is running

This will create new model



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.