Part I: Setting up the Required system:

FIRST: Setting up Apache Spark:

In your prompt/shell write the next Linux Command lines.

1. Download the software

wget https://www-eu.apache.org/dist/spark/spark-2.3.2/spark-2.3.2-bin-hadoop2.7.tgz

2. unzip it

tar -xzf spark-2.3.2-bin-hadoop2.7.tgz

3. move it to the folder you will work in my case → /usr/local/

mv spark-2.3.2-bin-hadoop2.7 spark
sudo mv spark /usr/local/

Note: The sudo command allows you to run programs with the security privileges of another user (by default, as the super user)

4. Set the environment variables

sudo nano ~/.bashrc

Note: This command opens the .bashrc file with nano text editor

5. Add the following lines in the opened file

export SPARK_HOME=/usr/local/spark
export PATH=$PATH:$SPARK_HOME/bin

6. Reload bashrc
source ~/.bashrc

7. Now we want to make Jupyter Notebook work with Spark - install/check if python is installed...

python3 –version

8. Now install PIP which is a package manager for Python packages, or modules; we use python3 instead default (e.g. 2.7.)

wget https://bootstrap.pypa.io/get-pip.py

9. Run get-pip.py
python3 get-pip.py

10. Now check PIP version
pip –version

- ➔ you should see this:
  pip 18.1 from /usr/local/lib/python3.5/dist-packages/pip (python 3.5)

11. Now install Jupyter
sudo -H pip install jupyter

12. Check jupyter version
jupyter –version

→ you should see this:4.4.0

13. Set the environment variables - linking PySpark with Jupyter Notebook, again access the .bashrc file.

sudo nano ~/.bashrc

14. Add the following lines
PYSPARK_DRIVER_PYTHON="jupyter"
PYSPARK_DRIVER_PYTHON_OPTS="notebook"

15. Test your installation.
Run -> pyspark

SECOND: Setting up Apache Cassandra

A) Setting up Cassandra

The Apache Cassandra database provides you with scalability and high availability without compromising performance. In this project, we use Apache Cassandra at Serving Layer of the Lambda Architecture. Check out the official installation instructions at:

http://cassandra.apache.org/doc/latest/getting_started/installing.html

1. You need Java 8, either the Oracle Java Standard Edition 8 or OpenJDK 8. To verify that you have the correct version of java installed, type.
   java –version

2. For using cqlsh, you need Python To verify that you have the correct version of Python installed, type:
   python --version.

3. Run the following commands in your Ubuntu Shell to download Cassandra:

   wget https://www-eu.apache.org/dist/cassandra/3.11.3/apache-cassandra-3.11.3-bin.tar.
   tar -xvf apache-cassandra-3.11.3-bin.tar.gz
   mv apache-cassandra-3.11.3 cassandra
   sudo mv cassandra /usr/local/

4. Set the path
   sudo nano ~/.bashrc

5. Add the following lines
   export CASSANDRA_HOME=/usr/local/cassandra
   export PATH=$PATH:$CASSANDRA_HOME/bin

6. Reload .bashrc
   source ~/.bashrc

B) Getting started with Cassandra

1. Start Cassandra in the foreground by invoking:
   cassandra -f
   You can press Control-C to stop Cassandra.

2. Verify that Cassandra is running by invoking:
   nodetool status

3. Now that you have Cassandra running, the next thing to do is connect to the Cassandra server and begin creating database objects. You can connect to Cassandra using the next command in your shell, and you will be in Cassandra:
   Cqlsh

4. From here is similar like SQL, first, create a keyspace:

   ```
   CREATE KEYSPACE mykeyspace
   WITH REPLICATION = { 'class' : 'SimpleStrategy', 'replication_factor' : 1 };
   ```

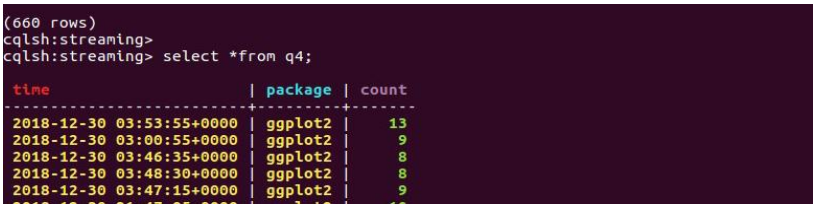
   * This will create the namespace for tables called mykeyspace. It uses a simple strategy for replication and only one copy will be stored, like SQL server.

5. You can view all the keyspaces:
   DESCRIBE KEYSPACES;

6. To work with keyspaces, we have to select one:
   USE mykeyspace;

7. We can create tables in a very similar way as in RDBMS:
   ```
   CREATE TABLE users (
   user_id int PRIMARY KEY,
            fname text,
            lname text
                 );
   ```
8. Inserts are also easy to perform::
   INSERT INTO users (user_id, fname, lname) VALUES (1, 'john', 'smith');

9. Now here we can query data for example from the keyspace ='streaming' and table q4, like the above figure; and we can do all sort of computations like SQL.



C) Linking Spark and Cassandra

Before we can use Cassandra database from Spark, we have to enable the Cassandra connector. The connector itself will be downloaded at the first time it is used. If you use pySpark, the only thing you have to do is to add packages parameters when you run it, so every time you want to initialise pySpark with Cassandra use the next command in your shell:

```
pyspark --packages datastax:spark-cassandra-connector:2.3.1-s_2.11 \
--conf spark.cassandra.connection.host=127.0.0.1
```

D) Working with Cassandra from Spark

1) Before we start using Cassandra from Spark, we need to have keyspace, table and some data in Cassandra, we have to create it through 'cqlsh'.

2) To read a DataFrame out of Cassandra, we use spark.read.format method, together with load() where we pass our keyspace name and the table we want to use, put the following code snippet in your Jupiter notebook:

```
user = spark.read.format("org.apache.spark.sql.cassandra")\
.load(keyspace="training", table="user")
```

*user could be any name.

3) As we created a normal Spark DataFrame, we can perform different operations on it, like:

```
adults = user.where(user.age > 21)
adults.show()
```

4) In order to write DataFrame to a Cassandra table, we need to create the table itself first in the cqlsh, then we can write the data using the following sequence of commands:

```
user.select("age", "user_id", "name")\ .write.format("org.apache.spark.sql.cassandra")\
.options(table="users_by_age", keyspace="training").save(mode="append")
```

*It selects few columns from the 'user' dataFrame and then it writes them to the keyspace and the table specified in the 'options'. In this example, we append the results to the table "users_by_age".