Setting up Hadoop, Hive, Spark

Objectives:

- To set up hadoop, hive & spark on linux machine
- To run hive queries using hive commands, sqlalchemy & from spark.

Hadoop

The Apache Hadoop software library is a framework that allows for the distributed processing of large data sets across clusters of computers using simple programming models. It is designed to scale up from single servers to thousands of machines, each offering local computation and

storage. Rather than rely on hardware to deliver high-availability, the library itself is designed to detect and handle failures at the application layer, so delivering a highly-available service on top of a cluster of computers, each of which may be prone to failures.

Hive

The Apache Hive data warehouse software facilitates reading, writing, and managing large datasets residing in distributed storage using SQL. Structure can be projected onto data already in storage. A command line tool and JDBC driver are provided to connect users to Hive.

Among many other components Hive is another component of hadoop environment.

Spark

is a unified analytics engine for large-scale data processing. It is a memory based engine. So it is much faster than hadoop. However, spark does not have a file system unlike hadoop (hadoop comes with its own file system HDFS). But not to worry, we can use the same HDFS for spark as well.

Setting up working environment

The whole setup was done in lubuntu 18.04 in vmware. It is recommended that the environment setup be done in the same linux distribution (or similar linux distributions as lubuntu).

Setting up Hadoop

1. Separate Login

   We will create a separate user for using hadoop, & i recommend the individuals following this documentation to follow it as well. To be clear we will be carrying out all the installation process for this new user. I.e whenever we want to use hadoop we login to this user.

   Create a new user & group:

   $ sudo addgroup hadoop

$sudo adduser -ingroup hadoop hduser

Remember the username & the password, because we will be using this throughout.

Add hadoop user to sudo group to grant all permission:

$ sudo adduser hduser sudo

2. Getting the environment ready

   In ubuntu environment, we need to check two things. Java installation because hadoop is written in java & ssh (secure shell) for security of communication between the nodes & cluster. First update the repository:

   $ sudo apt install update

   a. Install Java

      The version of jdk i installed was java 8 & i recommend you to install java 8 as well. At the time of writing this, hive does not have support for java version greater than 8. If you have java version greater than 8, i recommend you to install java 8 in a separate path & use this version of java throughout!

   b. Install SSH

      $ sudo apt install ssh

      For passwordless entry to localhost using ssh:

      $ su hduser

      $ su ssh-keygen -t rsa

      Note: leave blank if asked for name or location

      $ cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys

      $ chmod 0600 ~/.ssh/authorized_keys

      Check if ssh works & exit from the localhost:

      $ ssh localhost

      $ exit

3. Install hadoop on ubuntu

   a. Download the hadoop. The version i am using right now is hadoop 2.9.2 and Unzip the file:

$ wget

https://www-eu.apache.org/dist/hadoop/common/hadoop-2.9.2/hadoop-2.9.2.tar.g

z

b. Make a directory called hadoop & move the folder to this location.
$ sudo mkdir -p /usr/local/hadoop

$ cd hadoop-2.9.2/

*$ sudo mv * /usr/local/hadoop*

$ sudo chown -R hduser:hadoop /usr/local/hadoop

I moved the contents of hadoop to a location /usr/local/hadoop. You can move them to any place you want. But if you don't want to get confused in the next steps I advise you to stick to it.

4. Setting up configuration files

Before beginning, make sure that you are logged into hduser (the user we created earlier).

If you have switched the terminal or if you are not logged into the hduser. Login using:

$ su hduser

We have to make changes to few files. Do not be intimidated. I will explain what is going on. We have to edit these 5 files.

a. ~/.bashrc

In this file, we will include the path where the java is installed, as well as include the paths for different hadoop files. You know where the hadoop is installed. But if you do not know where the java is installed run the following command:

$ whereis java

You will see the path where the java is. Note the path, we will need it later. Now, we edit the ~/.bashrc file. (Note that .bashrc file will be different for each user.

Make sure you are editing the bashrc file for hduser):

$ sudo gedit ~/.bashrc

(I have used gedit editor, you can use any other editor.)

Once you have opened the file, add these lines to the bashrc file:

```
#JAVA
export JAVA_HOME=/usr/local/jdk1.8.0_191
export PATH=$PATH:$JAVA_HOME/bin
#HADOOP
export HADOOP_HOME=/usr/local/hadoop
export PATH=$PATH:$HADOOP_HOME/bin
export PATH=$PATH:$HADOOP_HOME/sbin
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME=$HADOOP_HOME
export YARN_HOME=$HADOOP_HOME
export HADOOP_CONF_DIR=/usr/local/hadoop/etc/hadoop
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
```

Save the file & close, run following to apply changes:

$ source ~/.bashrc

b. hadoop-env.sh

Now we will tell hadoop where the java is installed. For this let us open the file hadoop-env.sh:

$ sudo gedit /usr/local/hadoop/etc/hadoop/hadoop-env.sh

Now add the following line to the file:

export JAVA_HOME=/usr/local/jdk1.8.0_191

Note that you may have a different path than this. Use your own path.

c. core-site.xml

Create a temporary directory called tmp. Change the ownership to hduser.

$ sudo mkdir -p /app/hadoop/tmp

$ sudo chown hduser:hadoop /app/hadoop/tmp

Open the file core-site.xml in an editor. Append the following inside the configuration tab.

```
<configuration>
<property>
 <name>hadoop.tmp.dir</name>
  <value>/app/hadoop/tmp</value>
  <description>A base for other temporary directories.</description>
</property>
<property>
 <name>fs.default.name</name>
  <value>hdfs://localhost:54310</value>
        <description>The name of the default file system.  A URI whose scheme and authority determine the FileSystem implementation.  The uri's scheme determines the config property (fs.SCHEME.impl) naming the FileSystem implementation class.  The uri's authority is used to determine the host, port, etc. for a filesystem.</description>
 </property>
</configuration>
```

d. Hdfs-site.xml

Now we create two directory. It is optional to create these directory, as hadoop creates them for you. But it is a good practice to create them explicitly as we can specify where we want to create the files

$ sudo mkdir -p /usr/local/hadoop_store/hdfs/namenode

$ sudo mkdir -p /usr/local/hadoop_store/hdfs/datanode

$ sudo chown -R hduser:hadoop /usr/local/hadoop_store

Open the file hdfs-site.xml & append the following lines:

```
<configuration>
 <property>
  <name>dfs.replication</name>
  <value>1</value>
  <description>Default block replication.The actual number of replications can be specified when the file is created. The default is used if replication is not specified in create time.
  </description>
 </property>
 <property>
  <name>dfs.namenode.name.dir</name>
  <value>file:/usr/local/hadoop_store/hdfs/namenode</value>
 </property>
 <property>
  <name>dfs.datanode.data.dir</name>
  <value>file:/usr/local/hadoop_store/hdfs/datanode</value>
 </property>
```

```
</configuration>
```

e. yarn-site.xml

Open the file yarn-site.xml

$sudo nano /usr/local/hadoop/etc/hadoop/yarn-site.xml

Now, just add the following configurations

```
<configuration>
  <property>
        <name>yarn.nodemanager.aux-services</name>
        <value>mapreduce_shuffle</value>
  </property>
</configuration>
```

5. Format hadoop file system

   Hadoop installation is complete. Now lets format the namenode:

   $ hadoop namenode -format

   If you get error saying hadoop command not found, check if you have set up the paths correctly.

6. Start hadoop daemons

   Start the hadoop daemons with the following commands:

   $ start-all.sh

   Check if the hadoop daemons are running with the command

   $ jps

We have finally set up hadoop. Now let us set up hive on hadoop

Setting up hive with mysql metastore

Hive setup also includes downloading hive & tweaking some config files. Hive also needs a metastore to keep the metadata about the hive tables. We can use different databases to configure them as metastore for hive. But we will use mysql for our purpose.

Setting up hive

Make sure that you are logged into the hduser. If not

$ su hduser

1. Download Hive tar

   Download the version of hive that you prefer. In this documentation, i have used hive 2.1.0. I have downloaded the hive in the directory /home/hduser

   $ wget http://archive.apache.org/dist/hive/hive-2.1.0/apache-hive-2.1.0-bin.tar.gz
2. Extract the tar file

   $ tar -xzf apache-hive-2.1.0-bin.tar.gz
3. Edit bashrc file to set the environmental variables

   Open the bashrc file:

   $ sudo gedit ~/.bashrc

   Add the following lines at the end of the file:

   # Set HIVE_HOME

   export HIVE_HOME=/home/hduser/apache-hive-2.1.0-bin

   export PATH=$PATH:/home/hduser/apache-hive-2.1.0-bin/bin

   Note that I have downloaded it into the root home directory of hduser that we created earlier.

   Now apply the changes with:

   $ sudo source ~/.bashrc
4. Create hive directory within hdfs. The directory warehouse is the location where the table

or data related to hive

$ hdfs dfs -mkdir -p /user/hive/warehouse

$ hdfs dfs -mkdir/tmp

5. Set read write permission for the table

   Now we give the read write permission to the group just to be sure:

   $ sudo hdfs dfs -chmod g+w /usr/hive/warehouse

   $ sudo hdfs dfs -chmod g+w /tmp

6. Setup hadoop path in hive-env.sh

   $ cp $HIVE_HOME/conf/hive-env.sh.template $HIVE_HOME/conf/hive-env.sh.template

   $ sudo gedit $HIVE_HOME/conf/hive-env.sh.template

   Append the following lines to the hive-env.sh file.

   export HADOOP_HOME = /usr/local/hadoop

   export HADOOP_HEAPSIZE = 512

   export HADOOP_CONF_DIR = /home/hduser/apache-hive-2.1.0/conf

7. Edit hive-site.xml

   $ sudo gedit $HIVE_HOME/conf/hive-site.xml

   Replace the configuration with with the following:

```
<configuration>
<property>
<name>javax.jdo.option.ConnectionURL</name>
<value>jdbc:derby:;databaseName=/home/edureka/apache-hive-2.1.0-bin/metas
tore_db;create=true</value>
<description>
JDBC connect string for a JDBC metastore.
To use SSL to encrypt/authenticate the connection, provide
database-specific SSL flag in the connection URL.
For example, jdbc:postgresql://myhost/db?ssl=true for postgres database.
</description>
</property>
<property>
<name>hive.metastore.warehouse.dir</name>
<value>/user/hive/warehouse</value>
```

```
<description>location of default database for the warehouse</description>
</property>
<property>
<name>hive.metastore.uris</name>
<value/>
<description>Thrift URI for the remote metastore. Used by metastore client to connect to remote metastore.</description>
</property>
<property>
<name>javax.jdo.option.ConnectionDriverName</name>
<value>org.apache.derby.jdbc.EmbeddedDriver</value>
<description>Driver class name for a JDBC metastore</description>
</property>
<property>
<name>javax.jdo.PersistenceManagerFactoryClass</name>
<value>org.datanucleus.api.jdo.JDOPersistenceManagerFactory</value>
<description>class implementing the jdo persistence</description>
</property>
</configuration>
```

8. By default hive uses Derby database. Initialize derby

   As we can also see in the configurations, hive uses derby as default for metastore. So let us initialize the derby metastore. Later we will also see how to setup the mysql as metastore. Derby only for development purpose & only allow one instance of hive. Where as other databases such as mysql can handle many instances of hive server.

   $ bin/schematool -initSchema -dbType derby

9. Launch hive

   Launch the hive with following command:

   $ hive

   A hive terminal should open. Now you can easily do stuffs with hive! The commands are very similar to sql commands.

Setting up mysql as metastore for hive

1. Install mysql server

$ sudo apt install mysql-server

2. Install mysql java connector

   $ sudo apt install libmysql-java

3. Create soft link for connector in Hive lib directory or copy connector jar to lib folder.

   $ ln -s /usr/share/java/mysql-connector-java.jar $HIVE_HOME/lib/mysql-connector-java.jar

   NOTE: if your connector was not downloaded for some reason, download it manually & copy it to $HIVE_HOME/lib.

4. Create the Initial database schema using the hive-schema-0.14.0.mysql.sql file (or the file corresponding to your installed version of Hive) located in the $HIVE_HOME/scripts/metastore/upgrade/mysql directory.

   $ mysql -u root -p

   mysql > CREATE DATABASE metastore;

   mysql > USE metastore;

   mysql > SOURCE $HIVE_HOME/scripts/metastore/upgrade/mysql/hive-schema-0.14.0.mysql.sql;

   NOTE: check the version at the time of your install!

5. You also need a MySQL user account for Hive to use to access the metastore. It is very important to prevent this user account from creating or altering tables in the metastore database schema.

   mysql> CREATE USER 'hiveuser' IDENTIFIED BY 'hivepassword';

   *mysql> GRANT all on *.* to 'hiveuser' identified by 'hivepassword';*

   mysql> flush privileges;

   You can use any username & password you wish. But you should note it. We will use this same username & password to edit hive configuration files.

6. Create hive-site.xml (If not already present) in $HIVE_HOME/conf folder with the configuration below

   Edit the hive-site.xml file. Add the following in the configuration:

   ```
   <configuration>
   <!-- note that this property is to be replaced with the original one that
   ```

```
we set up earlier. Because we are now using mysql instead of derby.
-->
  <property>
    <name>javax.jdo.option.ConnectionURL</name>

<value>jdbc:mysql://localhost/metastore?createDatabaseIfNotExist=true</value>
    <description>metadata is stored in a MySQL server</description>
  </property>


  <property>
    <name>javax.jdo.option.ConnectionDriverName</name>
    <value>com.mysql.jdbc.Driver</value>
    <description>MySQL JDBC driver class</description>
  </property>

  <property>
    <name>javax.jdo.option.ConnectionUserName</name>
    <value>hiveuser</value>
    <description>user name for connecting to mysql server</description>
  </property>
  <property>
    <name>javax.jdo.option.ConnectionPassword</name>
    <value>hivepassword</value>
    <description>password for connecting to mysql server</description>
  </property>
</configuration>
```

7. We are all set now. Start the hive console

   $ hive

Setting up spark in hadoop

Setting up spark is relatively easier! Don't worry we are nearing the end!!

1. Get the download URL from the Spark download page, download it, and uncompress it. Download the file in the folder containing hadoop.

   $ cd /usr/local/hadoop

   $ wget https://www-eu.apache.org/dist/spark/spark-2.4.0/spark-2.4.0-bin-hadoop2.7.tgz

   $ tar -xvf spark-2.2.0-bin-hadoop2.7.tgz

   $ mv spark-2.2.0-bin-hadoop2.7 spark

2. Add spark to the environment variables

   $ sudo gedit ~/.bashrc

Add the path to spark

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

export SPARK_HOME=/usr/local/hadoop/spark

3. Edit the bashrc files

   export LD_LIBRARY_PATH=/home/hadoop/hadoop/lib/native:$LD_LIBRARY_PATH

   Run *$ source ~/.bashrc*
4. Restart the session by logging out & logging in again to hduser (the user we created)
5. Rename the spark default template

   $ mv $SPARK_HOME/conf/spark-defaults.conf.template $SPARK_HOME/conf/spark-defaults.conf
6. Edit the spark-default.conf to set spark.master to yarn

   `spark.master    yarn`
7. Edit various configurations as necessary. But for simplicity we will just run spark. Remember to run hadoop daemons before running spark.

   $ spark-shell

Basic Hive queries

The tables in hive can be accessed in many ways. Simply from the hive terminal like we do in mysql, using scripts written using sqlalchemy & also through spark. We will discuss necessary steps as well as few tweakings necessary for using sqlalchemy & spark. This just shows you how to get started!

Using sqlalchemy

First of all, we need to install the sqlalchemy library using pip. Start the hive server with:

$ hive --service hiveserver2

$ pip install sqlalchemy

++++++++++++++++++*Python code*++++++++++++++++++++++++

#import libraries

```
from sqlalchemy import *
from sqlalchemy.engine import create_engine
from sqlalchemy.schema import *


#Connect with hive Hive
‘’’
The default port for hive opens on 10000. The name of the database is that i have already created is named siddhi.
‘’’
engine = create_engine('hive://localhost:10000/siddhi')
conn = engine.connect()
# There is a table named cohort_analysis in the database siddhi.
table = Table('cohort_analysis',MetaData(bind=engine), autoload=True)
s = (select([table.c.product_name,table.c.username]))
result = conn.execute(s)
count = 0
for i in result:
    print(i)
    count += 1
    if count == 5:
            break
```

++

Using spark

Now we shall take a look at how to connect hive to spark.

1. copy hive-site.xml from hive to spark config

   $ ln %HIVE_HOME/conf/hive-site.xml %SPARK_HOME/conf/hive-site.xml
2. check the hive-site.xml file (check port(9083) and ip(local host))
3. Start hive metastore

$ hive --service metastore

4. Start the spark-shell using

   $ spark-shell --driver-java-options "-Dhive.metastore.uris=thrift://localhost:9083"

5. Basic query

```
scala> import org.apache.spark.SparkConf
import org.apache.spark.SparkConf

scala> import org.apache.spark.SparkContext
import org.apache.spark.SparkContext

scala> import org.apache.spark.sql.hive.HiveContext
import org.apache.spark.sql.hive.HiveContext

scala> import sqlContext.implicits._

scala> val hiveObj = new HiveContext(sc)

scala>hiveObj.refreshTable("siddhi.cohort_analysis")
scala> val df = hiveObj.sql("show tables in siddhi");

hiveObj.sql("select * from siddhi.cohort_analysis limit 5").show()
```