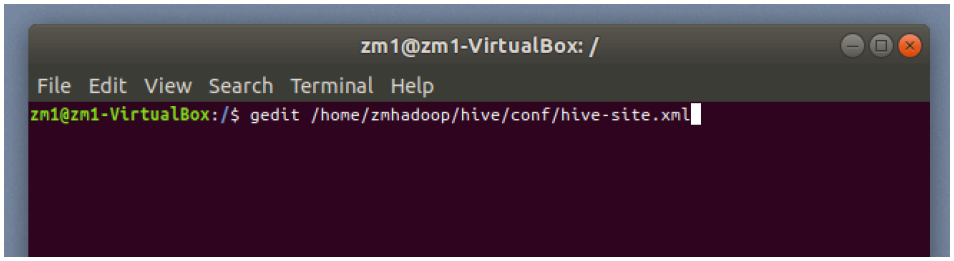
MILESTONE 3 - ACCESSING HIVE DATA WAREHOUSE USING PYTHON

STEP 1 : SETTING SERVER HOST AND USER ACCESS TO HIVE SERVER

1.1 Configure hive-site.xml to set hostname, port and access to hive server

- Open hive-site.xml which is located inside **/home/{yourname}/{hivefolder}/conf**
- Open using your preferred editor type : **gedit /home/zmhadoop/hive/conf/hive-site.xml**
- I'm using gedit as my editor and my hive-site.xml is located at **/home/zmhadoop/hive/conf/hive-site.xml**



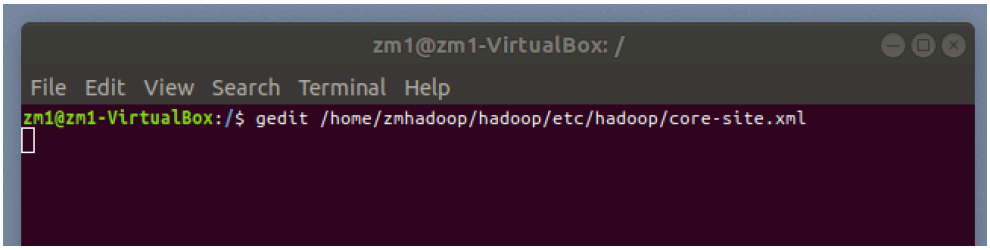
- Editor will open and search for the following property and update as below :-

```
<property>
      <name>hive.server2.thrift.bind.host</name>
      <value>localhost</value>
  </property>

<property>
      <name>hive.server2.thrift.port</name>
      <value>10000</value>
</property>

<property>
      <name>hive.exec.local.scratchdir</name>
            <value>/tmp/${user.name}</value>
      <description>Local scratch space for Hive jobs</description>
</property>

<property>
      <name>hive.downloaded.resources.dir</name>
      <value>/tmp/${user.name}_resources</value>
      <description>Temporary local directory for added resources in
      the remote file
      system.</description>
</property>
```

- Example hive-site.xml open using gedit :-

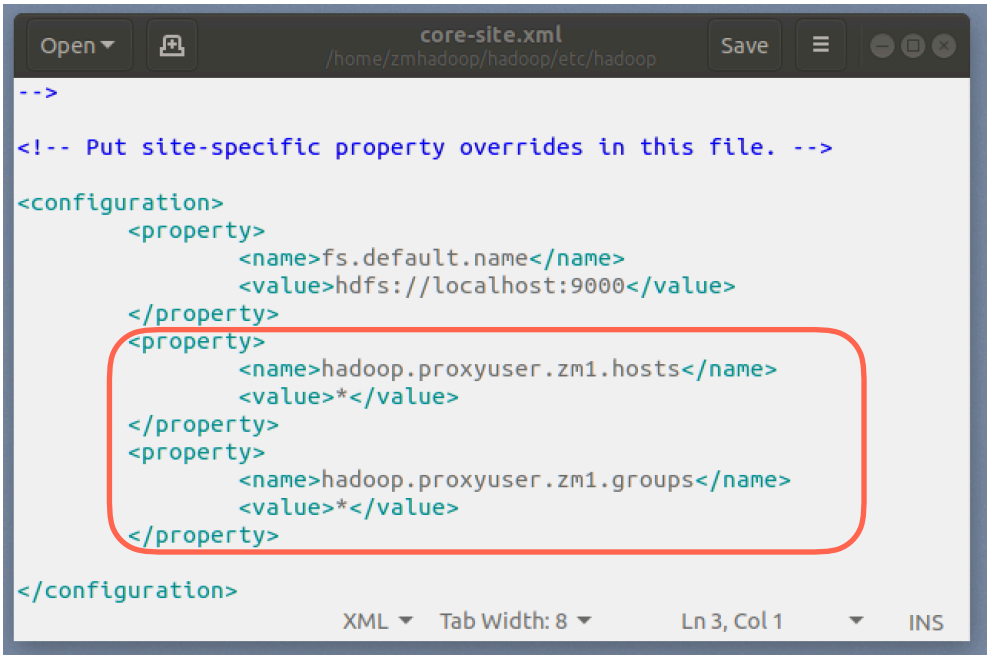
1.2 Configure core-site.xml to set proxy user to allow superuser connection from any host

- Open coresite.xml located at **/home/{yourname}/hadoop/etc/hadoop/core-site.xml**



- Adding proxy user entries in core-site.xml would allow the superuser named zm1 to connect from any host (as value is *) to impersonate a user belonging to any group (as value is *).
- Make sure to change the username accordingly to your superuser name. In this example i'm using **zm1** as my superuser. Add property as below:-

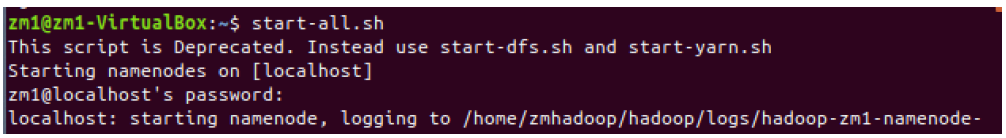
```
<property>
      <name>hadoop.proxyuser.zm1.hosts</name>
     <value>*</value>
</property>
<property>
      <name>hadoop.proxyuser.zm1.groups</name>
     <value>*</value>
</property>
```



STEP 2 : RUNNING HDFS, HIVE & HIVESERVER

2.1 Run HDFS

- Open terminal and start HDFS service by typing : **start-all.sh or start-dfs.sh and start-yarn.sh**



- To check whether the services are running or not, type : **jps**
- This will show the running services and the pid(process id)

```
zm1@zm1-VirtualBox:~$ jps
2304 SecondaryNameNode
2807 NodeManager
1881 NameNode
2458 ResourceManager
2906 Jps
2077 DataNode
zm1@zm1-VirtualBox:~$
```

2.2 Change permission HDFS directory

- To allow access to HDFS from other network, the permission of HDFS directory need to be changed as below : type : **hdfs dfs -chmod 777 /tmp**

```
zm1@zm1-VirtualBox:~$ hdfs dfs -chmod 777 /tmp
zm1@zm1-VirtualBox:~$ hdfs dfs -ls /
Found 3 items
drwxr-xr-x   - zm1 supergroup          0 2020-03-22 00:20 /home
drwxrwxrwx   - zm1 supergroup          0 2020-03-22 05:02 /tmp
drwxr-xr-x   - zm1 supergroup          0 2020-06-09 18:22 /user
zm1@zm1-VirtualBox:~$
```

- Directory /tmp is in the HDFS. Change the directory name accordingly to your setting.

2.3 Run Hive & Hiveserver

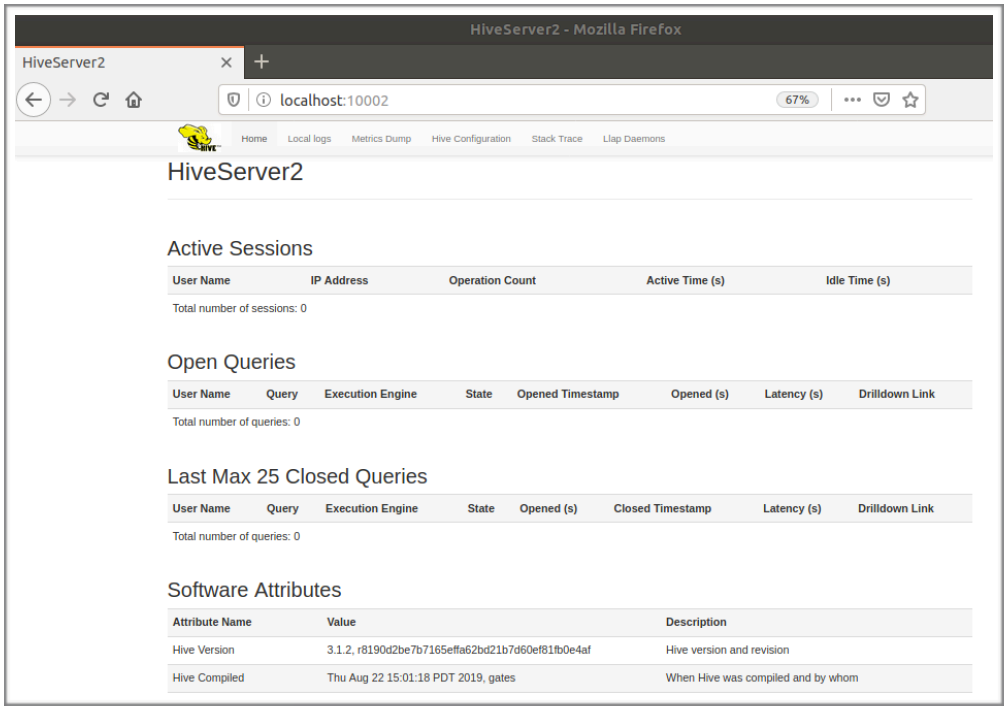
- Hive can only run after the HDFS is running. To run hive type : **hive** at the terminal.

```
zm1@zm1-VirtualBox: ~
File  Edit  View  Search  Terminal  Help
zm1@zm1-VirtualBox:~$ hive
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/zmhadoop/hive/lib/log4j-slf4j-impl-2.10.0.jar!/o
rg/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/zmhadoop/hadoop/share/hadoop/common/lib/slf4j-lo
g4j12-1.7.10.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.apache.logging.slf4j.Log4jLoggerFactory]
Hive Session ID = 172c5e79-bfbe-4dc5-9b0f-2d03ea507701

Logging initialized using configuration in jar:file:/home/zmhadoop/hive/lib/hive-common-
3.1.2.jar!/hive-log4j2.properties Async: true
Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Cons
ider using a different execution engine (i.e. spark, tez) or using Hive 1.X releases.
hive>
```

- Then run hiveserver by typing : **hiveserver2**

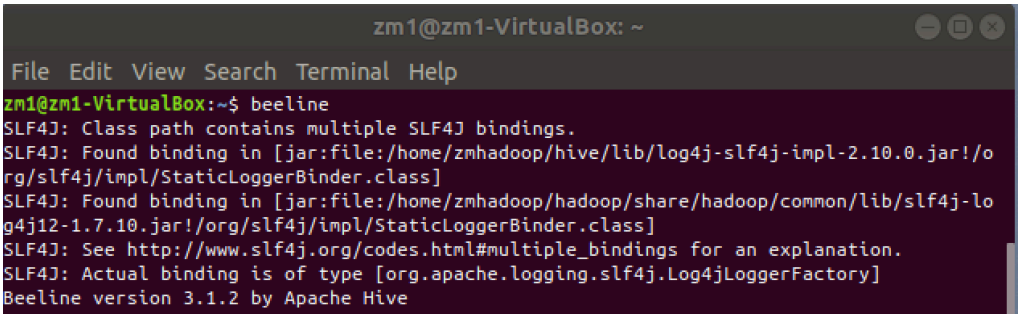
```
zm1@zm1-VirtualBox:~$ hiveserver2
2020-06-18 11:05:29: Starting HiveServer2
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/zmhadoop/hive/lib/log4j-slf4j-impl-2.10.0.jar!/o
rg/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/zmhadoop/hadoop/share/hadoop/common/lib/slf4j-lo
g4j12-1.7.10.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.apache.logging.slf4j.Log4jLoggerFactory]
Hive Session ID = 06d0c78b-b658-412c-b125-1a58159ea1c3
Hive Session ID = 8cac9172-17a7-46cd-8aea-ff74248e88a0
Hive Session ID = 6889297a-64b1-4ef1-930c-6d7108799aa4
```

- Check if the hiveserver are running or not by browsing **http://localhost:10002**. You will see the hiveserver2 interface as below:



2.4 Test connection to the server & query using Beeline

- Type : **beeline** at the terminal to run beeline



- Connect to the hiveserver, type : **!connect jdbc:hive2://localhost:10000**

```
beeline> !connect jdbc:hive2://localhost:10000
Connecting to jdbc:hive2://localhost:10000
Enter username for jdbc:hive2://localhost:10000:
Enter password for jdbc:hive2://localhost:10000:
Connected to: Apache Hive (version 3.1.2)
Driver: Hive JDBC (version 3.1.2)
Transaction isolation: TRANSACTION_REPEATABLE_READ
```

- Show all the database, type at the beeline : **show databases;**

```
0: jdbc:hive2://localhost:10000> show databases;
INFO  : Compiling command(queryId=zm1_20200618114343_36496686-e26b-4bae-af4b-5540cd53c1dd): show databases
INFO  : Concurrency mode is disabled, not creating a lock manager
INFO  : Semantic Analysis Completed (retrial = false)
INFO  : Returning Hive schema: Schema(fieldSchemas:[FieldSchema(name:database_name, type:string, comment:from deserializer)], properties:null)
INFO  : Completed compiling command(queryId=zm1_20200618114343_36496686-e26b-4bae-af4b-5540cd53c1dd); Time taken: 0.003 seconds
INFO  : Concurrency mode is disabled, not creating a lock manager
INFO  : Executing command(queryId=zm1_20200618114343_36496686-e26b-4bae-af4b-5540cd53c1dd): show databases
INFO  : Starting task [Stage-0:DDL] in serial mode
INFO  : Completed executing command(queryId=zm1_20200618114343_36496686-e26b-4bae-af4b-5540cd53c1dd); Time taken: 0.035 seconds
INFO  : OK
INFO  : Concurrency mode is disabled, not creating a lock manager
+----------------+
| database_name  |
+----------------+
| crudeoil       |
| default        |
| lab            |
| labtest        |
+----------------+
4 rows selected (0.175 seconds)
```

- Show all the tables, type at the beeline : **show tables;**

```
0: jdbc:hive2://localhost:10000> show tables;
INFO  : Compiling command(queryId=zm1_20200618114501_56ee3e15-3916-4566-850a-80a045b530a0): show tables
INFO  : Concurrency mode is disabled, not creating a lock manager
INFO  : Semantic Analysis Completed (retrial = false)
INFO  : Returning Hive schema: Schema(fieldSchemas:[FieldSchema(name:tab_name, type:string, comment:from deserializer)], properties:null)
INFO  : Completed compiling command(queryId=zm1_20200618114501_56ee3e15-3916-4566-850a-80a045b530a0); Time taken: 0.004 seconds
INFO  : Concurrency mode is disabled, not creating a lock manager
INFO  : Executing command(queryId=zm1_20200618114501_56ee3e15-3916-4566-850a-80a045b530a0): show tables
INFO  : Starting task [Stage-0:DDL] in serial mode
INFO  : Completed executing command(queryId=zm1_20200618114501_56ee3e15-3916-4566-850a-80a045b530a0); Time taken: 0.045 seconds
INFO  : OK
INFO  : Concurrency mode is disabled, not creating a lock manager
+-----------------+
|     tab_name    |
+-----------------+
| crudeoilbrent   |
| crudeoilwti     |
| crudeoilwtitest |
+-----------------+
3 rows selected (0.19 seconds)
```

- Test query using select command, type at the beeline : **select * from crudeoilbrent limit 5;**

```
0: jdbc:hive2://localhost:10000> select * from crudeoilbrent limit 5;
INFO  : Compiling command(queryId=zm1_20200618114557_48bfcaa3-be33-4e07-b7b8-bc9ae50dbd61): select * from crudeoilbrent limit 5
INFO  : Concurrency mode is disabled, not creating a lock manager
INFO  : Semantic Analysis Completed (retrial = false)
INFO  : Returning Hive schema: Schema(fieldSchemas:[FieldSchema(name:crudeoilbrent.dateprice, type:string, comment:null), FieldSchema(name:crudeoilbrent
.closingprice, type:double, comment:null), FieldSchema(name:crudeoilbrent.openprice, type:double, comment:null), FieldSchema(name:crudeoilbrent.dailyhig
h, type:double, comment:null), FieldSchema(name:crudeoilbrent.dailylow, type:double, comment:null)], properties:null)
INFO  : Completed compiling command(queryId=zm1_20200618114557_48bfcaa3-be33-4e07-b7b8-bc9ae50dbd61); Time taken: 0.362 seconds
INFO  : Concurrency mode is disabled, not creating a lock manager
INFO  : Executing command(queryId=zm1_20200618114557_48bfcaa3-be33-4e07-b7b8-bc9ae50dbd61): select * from crudeoilbrent limit 5
INFO  : Completed executing command(queryId=zm1_20200618114557_48bfcaa3-be33-4e07-b7b8-bc9ae50dbd61); Time taken: 0.001 seconds
INFO  : OK
INFO  : Concurrency mode is disabled, not creating a lock manager
+-------------------------+---------------------------+-------------------------+-------------------------+-------------------------+
| crudeoilbrent.dateprice | crudeoilbrent.closingprice | crudeoilbrent.openprice | crudeoilbrent.dailyhigh | crudeoilbrent.dailylow  |
+-------------------------+---------------------------+-------------------------+-------------------------+-------------------------+
| 3/11/2020               | 35.79                     | 37.27                   | 39.7                    | 35.35                   |
| 3/10/2020               | 37.22                     | 35.84                   | 38.22                   | 35.0                    |
| 3/9/2020                | 34.36                     | 38.28                   | 38.34                   | 31.02                   |
| 3/6/2020                | 45.27                     | 50.25                   | 50.45                   | 45.18                   |
| 3/5/2020                | 49.99                     | 51.66                   | 52.04                   | 49.7                    |
+-------------------------+---------------------------+-------------------------+-------------------------+-------------------------+
5 rows selected (0.475 seconds)
```

STEP 3 : ACCESS HIVE DATA WAREHOUSE USING PYTHON

3.1 Check hiveserver ip address

- If all the step above are successfully applied, we can now access hive server using python code.
- Check hiveserver ip address, type : **ifconfig** to get the current ip address for your server. You can use internal ip address or external depend from where is your connection.

```
zm1@zm1-VirtualBox:~$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.0.106  netmask 255.255.255.0  broadcast 192.168.0.255
        inet6 fe80::5391:b52:dbc0:352f  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:53:0f:bc  txqueuelen 1000  (Ethernet)
        RX packets 5223  bytes 3883933 (3.8 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 2557  bytes 337114 (337.1 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 18972  bytes 4061414 (4.0 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 18972  bytes 4061414 (4.0 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

- We are using PyHive to connect to hive using python. Install PyHive : **pip install PyHive**
- PyHive is a collection of Python DB-API and SQLAlchemy.
- Create hive connection using PyHive

```
#Initiate pandas & pyhive
 import pandas as pd
 from pyhive import hive
#Create Hive connection
 conn = hive.Connection(host="192.168.0.106", port=10000)
```

- Use panda to query and load table from hive to python dataframe.

```
# Read Hive table crudeoilbrent and load into pandas dataframe
df = pd.read_sql("SELECT * FROM default.crudeoilbrent", conn)
df.head()
```

- The code and output are shown as below :

In [1]:
```
#from pyhive import hive
import pandas as pd
from pyhive import hive

#Create Hive connection
conn = hive.Connection(host="192.168.0.106", port=10000)

# Read Hive table and Create pandas dataframe
df = pd.read_sql("SELECT * FROM default.crudeoilbrent", conn)

df.head()
```

Out[1]:

	crudeoilbrent.dateprice	crudeoilbrent.closingprice	crudeoilbrent.openprice	crudeoilbrent.dailyhigh	crudeoilbrent.dailylow
0	3/11/2020	35.79	37.27	39.70	35.35
1	3/10/2020	37.22	35.84	38.22	35.00
2	3/9/2020	34.36	38.28	38.34	31.02
3	3/6/2020	45.27	50.25	50.45	45.18
4	3/5/2020	49.99	51.66	52.04	49.70

In [4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3675 entries, 0 to 3674
Data columns (total 5 columns):
crudeoilbrent.dateprice       3675 non-null object
crudeoilbrent.closingprice    3675 non-null float64
crudeoilbrent.openprice       3626 non-null float64
crudeoilbrent.dailyhigh       3647 non-null float64
crudeoilbrent.dailylow        3645 non-null float64
dtypes: float64(4), object(1)
memory usage: 143.6+ KB
```

In [5]:
```
df.shape
```

Out[5]:
```
(3675, 5)
```

#END