10. Setting up & getting started with Hive

Posted on November 17, 2018

Hive is popular SQL based tool to process Big Data. It stores & retrieves data to/from HDFS.

Prerequisite: Xcode & Hadoop are installed as outlined in Setting up & getting started with Hadoop on Mac

1. Install Hive

Let's install hive-2.3.4 with hadoop 2.7.7.

300+ Java Interview FAQs

300+ Java FAQs 🔥

16+ Java Key Areas Q&As

150+ Java Architect FAQs

80+ Java Code Quality Q&As

150+ Java Coding Q&As

300+ Big Data Interview FAQs

300+ Big Data FAQs 🔥

Tutorials - Big Data

TUT - ⏯ Starting Big Data

TUT - Starting Spark & Scala

```
$ wget http://www-us.apache.org/dist/hive/hive-2.3
$ sudo tar xvzf apache-hive-2.3.4-bin.tar.gz -C /us
```

This will install 2.3.4 version of Hive in the folder /usr/local/apache-hive-2.3.4-bin/

2. set HIVE_HOME

```
$ vi ~/.bash_profile
```

In the ".bash_profile"

```
export HIVE_HOME=/usr/local/apache-hive-2.3.4-bin
export HIVE_CONF_DIR=$HIVE_HOME/conf
export PATH=$HIVE_HOME/bin:$PATH
export CLASSPATH=$CLASSPATH:$HADOOP_HOME/lib/*
export CLASSPATH=$CLASSPATH:$HIVE_HOME/lib/*
```

Activate the change.

```
$ source ~/.bash_profile
```

3. Install the MySQL Server

Hive requires a RDBMS to store its meta-data. It is called the Hive metastore. Hive can be used with the embedded Derby database, but Derby is good for the sake of development and unit testing, but won't scale to a production environment as only a single user can connect to the derby database at any instant of time. Better way to configure is to use an external

TUT - Starting with Python
TUT - Kafka
TUT - Pig
TUT - Apache Storm
TUT - Spark Scala on Zeppelin
TUT - Cloudera
TUT - Cloudera on Docker
TUT - File Formats
TUT - Spark on Docker
TUT - Flume
TUT - Hadoop (HDFS)
TUT - HBase (NoSQL)
TUT - Hive (SQL)
TUT - Hadoop & Spark
TUT - MapReduce
TUT - Spark and Scala
TUT - Spark & Java
TUT - PySpark on Databricks
TUT - Zookeeper

database which is JDBC compliant like MySQL, Oracle, etc.

```
$ brew install mysql
```

4. Start the MySQL Server

```
$ mysql.server start
```

5. Set up MySQL Server

```
$ mysql -u root
```

Create a database named “metastore” and a new user named “hiveuser”, and grant permissions. You also need to run the schema upgrade scripts. The Hive version being used here is “2.3.x”.

```
mysql> CREATE DATABASE metastore;
mysql> USE metastore;
mysql> SOURCE /usr/local/apache-hive-2.3.4-bin/scr
mysql> CREATE USER 'hiveuser'@'localhost' IDENTIFI
mysql> GRANT SELECT,INSERT,UPDATE,DELETE,ALTER,CRE
```

Note: schematool is an offline command line tool to manage the **metastore**. This tool can be used to initialize the metastore schema for the current Hive version (E.g. 2.3.x).

```
$ schematool -dbType mssql -info
```

Before you run hive for the first time, run the following to initialize the schema:

```
$ schematool -dbType mssql -initSchema
```

6. Download mysql-connector-java

Download from https://dev.mysql.com/downloads/connector/j/

```
$ curl -L 'http://www.mysql.com/get/Downloads/Conn
```

Copy the “mysql-connector-java-8.0.13.jar” that has the Driver class “**com.mysql.jdbc.Driver**”

```
$ mv mysql-connector-java-8.0.13/mysql-connector-j
```

```
$ sudo chown -R <user>:admin apache-hive-2.3.4-bin
```

7. Configure Hive – hive-env.sh

```
$ cd /usr/local/apache-hive-2.3.4-bin/conf
$ cp hive-env.sh.template hive-env.sh
```

8. Configure Hive – hive-site.xml

```
$ cd /usr/local/apache-hive-2.3.4-bin/conf
$ cp hive-default.xml.template hive-site.xml
```

Make sure the following selected lines between the and tags of hive-site.xml have the values as shown below:

```
$ vi /usr/local/apache-hive-2.3.4-bin/conf/hive-si
```

Make sure that the following properties are set as shown below. These properties are used to connect to the external MySQL metastore database. When you start the Hive shell, it will automatically connect to the MySQL database and create the required tables in it.

```
<configuration>

  //.....................

  <property>
    <name>javax.jdo.option.ConnectionPassword</nar
    <value>password</value>
    <description>password to use against metastore
  </property>

  <property>
    <name>javax.jdo.option.ConnectionURL</name>
    <value>jdbc:mysql://localhost:3306/metastore?
    <description>JDBC connect string for a JDBC me
  </property>

  <property>
    <name>javax.jdo.option.ConnectionDriverName</n
    <value>com.mysql.cj.jdbc.Driver</value>
    <description>Driver class name for a JDBC meta
 </property>

 <name>javax.jdo.option.ConnectionUserName</name>
```

```
    <value>hiveuser</value>
    <description>Username to use against metastore
 </property>

 <property>
    <name>hive.querylog.location</name>
    <value>/usr/local/apache-hive-2.3.4-bin/iotmp
    <description>Location of Hive run time structu
  </property>

  <property>
    <name>hive.exec.local.scratchdir</name>
    <value>/usr/local/apache-hive-2.3.4-bin/iotmp
    <description>Local scratch space for Hive jobs
  </property>
  <property>
    <name>hive.downloaded.resources.dir</name>
    <value>/usr/local/apache-hive-2.3.4-bin/iotmp
    <description>Temporary local directory for add
      </description>
  </property>
  <property>
    <name>hive.metastore.schema.verification</name
    <value>false</value>
    <description>
      Enforce metastore schema version consistency
      True: Verify that version information stored
            schema migration attempt. Users are re
            proper metastore schema migration. (De
      False: Warn if the version information store
    </description>
  </property>

  //.....................

<configuration>

```

Include the below configuration in **conf/hive-site.xml**

```
<property>
        <name>hive.metastore.uris</name>
        <value>thrift://localhost:9083</value>
    </property>

```

9. Configure YARN – yarn-site.xml

Make sure that the following properties are set in yarn-site.xml.

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

  <property>
    <name>yarn.nodemanager.aux-services.mapreduce_
    <value>org.apache.hadoop.mapred.ShuffleHandle
  </property>
....

```

10. Start hadoop if not already started

```

$ ./start-all.sh
$ jps -lm

```

Output:

```
4580 sun.tools.jps.Jps -lm
4406 org.apache.hadoop.yarn.server.resourcemanager
4167 org.apache.hadoop.hdfs.server.datanode.DataNo
4503 org.apache.hadoop.yarn.server.nodemanager.Nod
4073 org.apache.hadoop.hdfs.server.namenode.NameNo
4283 org.apache.hadoop.hdfs.server.namenode.Second

```

11. Verify Hive installation

```

$ echo $HIVE_HOME

```

Output:

```
/usr/local/apache-hive-2.3.4-bin
```

12. Start the Hive metastore interface

```
$ hive --service metastore &
```

The Hive metastore interface by default listens at port 9083. Make sure it is.

```
$ netstat -an | grep 9083
```

13. Start Hive

```
$ hive
```

Output:

```
hive>
```

14. Create HDFS path

```
$ hdfs dfs -mkdir -p /user/hive/warehouse
$ hdfs dfs -chown -R hiveuser /user/hive/warehouse
$ hdfs dfs -chmod -R 777 /user/hive/warehouse
```

15. Create a Hive table

```
hive> CREATE TABLE IF NOT EXISTS test_table
```

```
    >     (col1 int COMMENT 'Integer Column',
    >     col2 string COMMENT 'String Column')
    >  COMMENT 'This is test table'
    >  ROW FORMAT DELIMITED
    >  FIELDS TERMINATED BY ','
    >  STORED AS TEXTFILE;
OK
Time taken: 4.016 seconds
```

16. Insert data into test_table

```

hive> INSERT INTO default.test_table values(1, 'Sor
hive> INSERT INTO test_table values(2, 'Some test v

```

Output: runs a mapreduce job

```
Total jobs = 3
Launching Job 1 out of 3
Number of reduce tasks is set to 0 since there's n
Starting Job = job_1542456661191_0002, Tracking UR
Kill Command = /usr/local/hadoop-2.7.7/bin/hadoop
Hadoop job information for Stage-1: number of mapp
2018-11-17 23:15:50,403 Stage-1 map = 0%,  reduce
2018-11-17 23:15:55,562 Stage-1 map = 100%,  reduc
Ended Job = job_1542456661191_0002
Stage-4 is selected by condition resolver.
Stage-3 is filtered out by condition resolver.
Stage-5 is filtered out by condition resolver.
Moving data to directory hdfs://localhost:9000/use
Loading data to table default.test_table
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1   HDFS Read: 4344 HDFS Write
Total MapReduce CPU Time Spent: 0 msec
OK
Time taken: 13.153 seconds

```

17. SELECT from test_table

```

hive> SELECT * FROM test_table;
OK
1    Some test value 1
```

```
2    Some test value 2
Time taken: 0.118 seconds, Fetched: 2 row(s)

```

18. Where is the underlying data stored in HDFS?

```
hdfs dfs -ls /user/hive/warehouse
```

Output:

```
drwxrwxrwx   - user supergroup          0 2018-11-1
```

```
hdfs dfs -ls /user/hive/warehouse/test_table
```

Output:

```
-rwxrwxrwx   3 user supergroup         20 2018-11-1
-rwxrwxrwx   3 user supergroup         20 2018-11-1
```

```
hdfs dfs -cat /user/hive/warehouse/test_table/00000
```

Output:

```
1,Some test value 1
```

```
hdfs dfs -cat /user/hive/warehouse/test_table/00000
```

Output:

```
2,Some test value 2
```

So, basically when you execute a SQL query, a mapreduce job is run to save or select data tp/from

HDFS. Hive can be run with other execution engines like Spark & Tez. The default engine is mapreduce.

We will look at more examples in the coming tutorials.

Learn more about Hive Q&As style

Hive Interview FAQs

‹ 09. Getting started with Spark & Hadoop – client mode on local & cluster mode on yarn

13 Apache Sqoop interview Q&As ›

Disclaimer