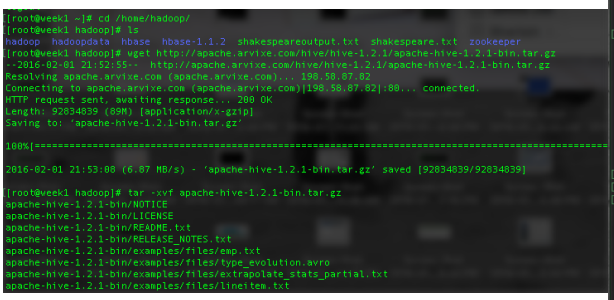
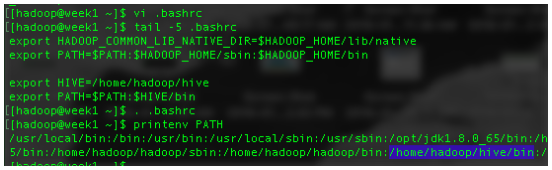
MSDS610 Week 4 Hive Assignment - Nathan Worsham

Installing Hive

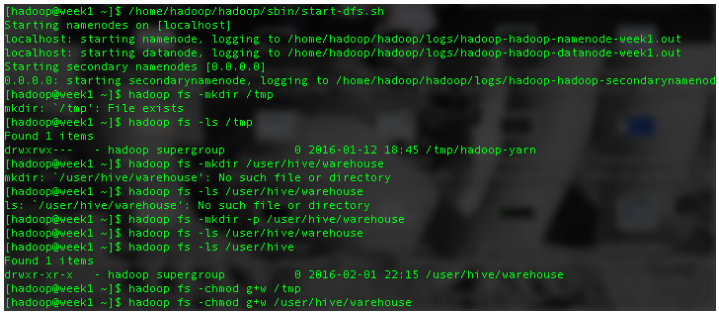
Again on this week's assignment I started by powering up my VM from week 1. Since the rest of the software on this VM is installed at /home/hadoop I decided to make the Hive installation location no different and unpacked the archive there along with changing ownership of the files.



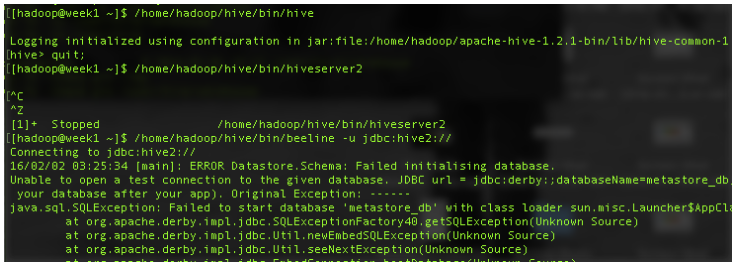
Next I configured the environment variables to inlude HIVE_HOME as instructed.



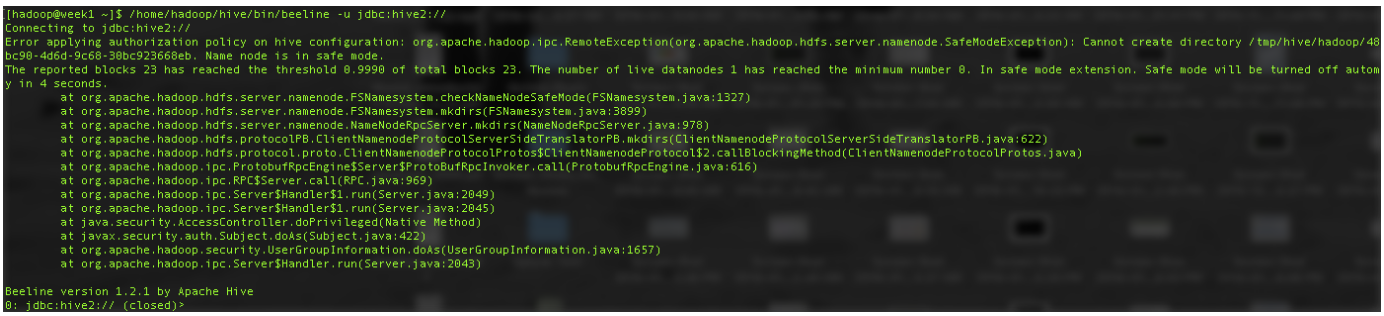
Now the instructions wanted directories and permissions established on such directories within the HDFS. Not realizing, I tried making the /tmp directory only to find one already there. It seemed the rights were already correct but equally did not seem it would hurt to make sure with the `chmod` command. When I tried to create layers of directories with the command they provided–`hadoop fs -mkdir /user/hive/warehouse`–it complained of "No such file or directory". Adding the `-p` option like the normal `mkdir` command to "Create intermediate directories as required" worked in the HDFS. I then added the rights to the new directory as well.



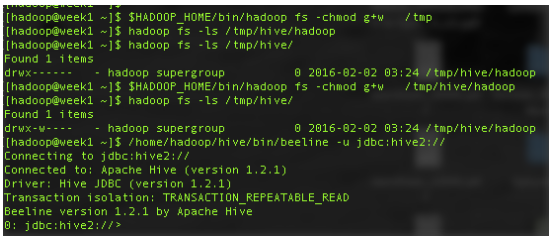
I was now ready to start hive, which started up fine. The instructions indicate that HiveCLI is deprecated and that instead "HiveServer2 and Beeline" should be used in it's place. When I tried the first command `$HIVE_HOME/bin/hiveserver2` the shell did not come back, it seems this command starts up a process. I used CTRL-Z to stop it after CTRL-C could not break it. Reading further it seems the instructions offer a single command to run both–`$HIVE_HOME/bin/beeline -u jdbc:hive2://`–but when I ran it I received errors about "initialising [sic] the database".



Thinking that when I killed the previous command there was likely some fallout, so I rebooted the VM and tried the command again, this time I was received a new error about not being able to create a directory in /tmp inside the HDFS.

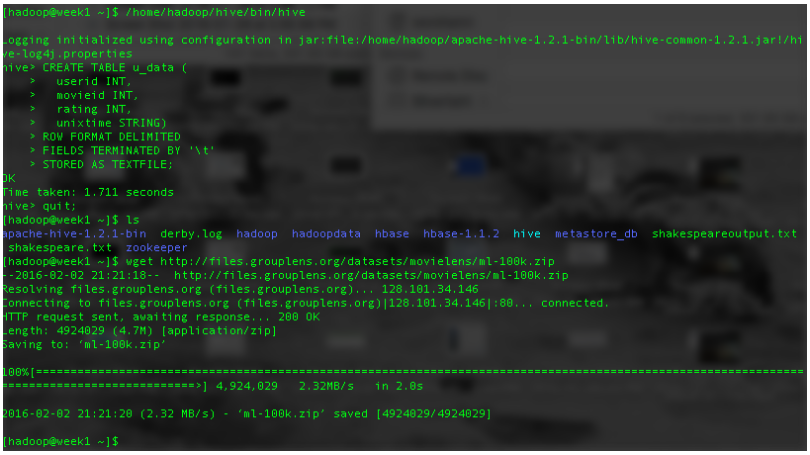


It would seem that similarly to the instruction using the `mkdir` command, the `chmod` command instructions did not account for recursive actions. This is because when I did a file listing in the HDFS of `/tmp/hive` I found that the group portion of the permissions was lacking the write permission the command was supposed to give. So when I ran the command again, but this time targeting that folder, the rights were now there and when I started the combined command for HiveServer2 and Beeline, I was rewarded a CLI command prompt with no errors.

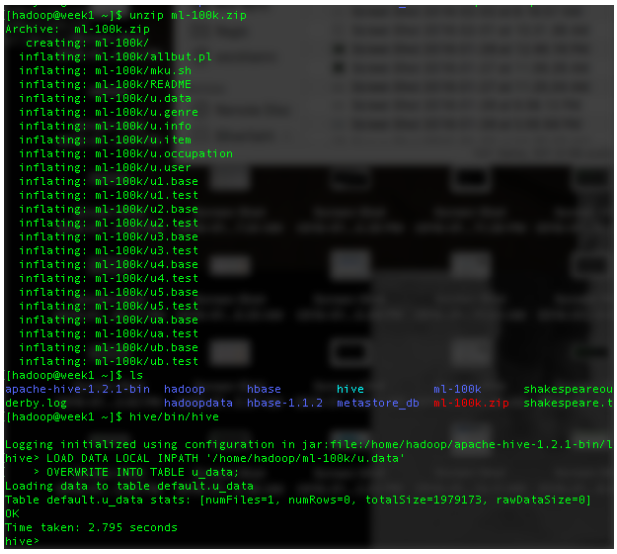


Hive Test Use Case

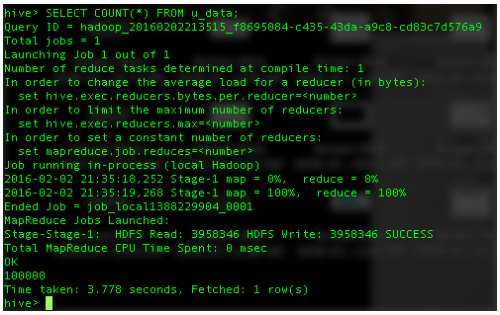
Following the instructions for the test use case using MovieLens data, I created the table using Hive CLI (despite the earlier mention of deprecation, as it seems simpler to use). I then exited out of the Hive CLI and downloaded the MovieLens data.



After the download, I tried to unpack the data but realized I needed to first go and grab the package for unzip as it was not installed. After getting it installed and unpacking the data, I was ready to load the data into the table using the `LOAD DATA` function to load the u.data file. According to the README file, the u.data file is the "full u data set, 100000 ratings by 943 users on 1682 items".



Next I counted the rows in the table. Here I can see that it is running a mapreduce job to accomplish the task. I admit I was a little concerned this wasn't going to return anything because the previous step when I loaded the data in had a message about "numRows=0", but the query came back with the correct number of 100000.



Now the instructions were to make a python script for the "mapper script". I went ahead and created

a scripts directory just for tidiness and built the script there. Looks like the script removes line endings, then segments out each line placing each part in a variable but most importantly the unixtime variable. It then takes that unixtime variable and converts it to a day of the week as a number (1-7) and then reformats the line back out again separated by tabs. Next I created another table and added the python script as a resource.

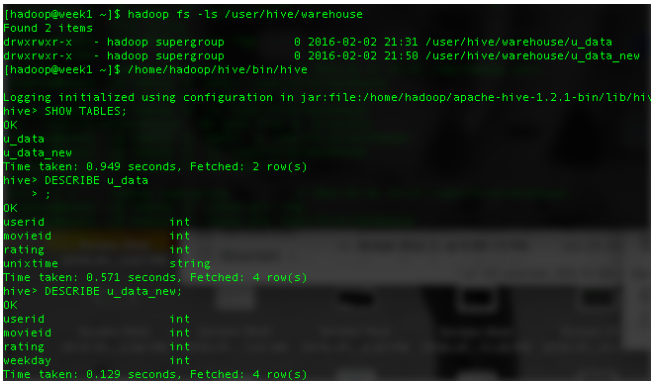
```
[hadoop@week1 ~]$ hive/bin/hive

Logging initialized using configuration in jar:file:/home/h
hive> CREATE TABLE u_data_new (
    >   userid INT,
    >   movieid INT,
    >   rating INT,
    >   weekday INT)
    > ROW FORMAT DELIMITED
    > FIELDS TERMINATED BY '\t';
OK
Time taken: 1.409 seconds
hive> add FILE /home/hadoop/scripts/weekday_mapper.py
    > ;
Added resources: [/home/hadoop/scripts/weekday_mapper.py]
```

I ran the command that would "transform" the data from the original table to the data into the new table with the day of the week as a number. Though the thought did occur to me of why the exercise went to the the trouble of making another table instead of just adding a column with the converted days of the week value to the original table? After this completed I was then able to count from this data and `GROUP BY` the day of the week.

```
hive> INSERT OVERWRITE TABLE u_data_new
    > SELECT
    >   TRANSFORM (userid, movieid, rating, unixtime)
    >   USING 'python weekday_mapper.py'
    >   AS (userid, movieid, rating, weekday)
    > FROM u_data;
Query ID = hadoop_20160202215043_7c933dd7-b84d-4e85-8ad1-35a515999869
Total jobs = 3
Launching Job 1 out of 3
Number of reduce tasks is set to 0 since there's no reduce operator
Job running in-process (local Hadoop)
2016-02-02 21:50:47,193 Stage-1 map = 0%,  reduce = 0%
2016-02-02 21:50:49,210 Stage-1 map = 100%,  reduce = 0%
Ended Job = job_local901465775_0001
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: hdfs://localhost:9000/user/hive/warehouse/u_data_new/.hive-staging_hive_2016-02-02_21-50-
Loading data to table default.u_data_new
Table default.u_data_new stats: [numFiles=1, numRows=100000, totalSize=1179173, rawDataSize=1079173]
MapReduce Jobs Launched:
Stage-Stage-1:  HDFS Read: 1979173 HDFS Write: 1179256 SUCCESS
Total MapReduce CPU Time Spent: 0 msec
OK
Time taken: 6.513 seconds
hive> SELECT weekday, COUNT(*)
    > FROM u_data_new
    > GROUP BY weekday;
Query ID = hadoop_20160202215106_1eab2f7c-1760-4f8d-8de6-59465b6420da
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Job running in-process (local Hadoop)
2016-02-02 21:51:08,044 Stage-1 map = 100%,  reduce = 100%
Ended Job = job_local958194460_0002
MapReduce Jobs Launched:
Stage-Stage-1:  HDFS Read: 6316858 HDFS Write: 2358512 SUCCESS
Total MapReduce CPU Time Spent: 0 msec
OK
1       13346
2       14389
3       15928
4       13340
5       18405
6       12229
7       12363
Time taken: 1.762 seconds, Fetched: 7 row(s)
hive>
```

This shows that the most ratings happened on Fridays, followed by Wednesdays. Surprisingly, weekends were the least popular, but perhaps this is when the subjects watched the content in order to give it a review later in the week. Last I decided to try out some other HQL commands to show and describe the tables that now existed.



References

cwiki.apache.org, 2016. Retrieved from
https://cwiki.apache.org/confluence/display/Hive/GettingStarted#GettingStarted-InstallingHivefromaStableRelease