**Lab 9 - Apache Spark SQL on GCP using Dataproc**

Processing a data set using Spark on Google Cloud Dataproc requires the following main steps:

Part 1- Enable the Google Cloud Compute Engine API.

Part 2- Create a Google Cloud Storage bucket to hold data inputs, data outputs and logs.

Part 3- Create, Configure and Launch a Google Cloud Dataproc cluster.

Part 4- Log in to the Hadoop cluster master node.

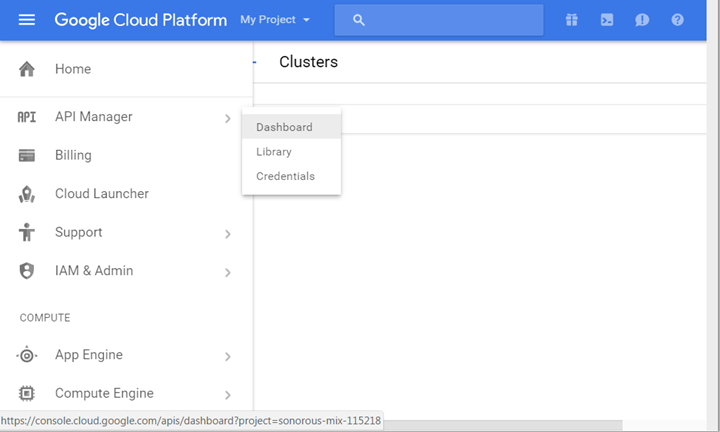
Part 5- Run the CLI and issue Spark-SQL commands to create tables and run queries.

Part 6- Shut down cluster and remove any temporary resources.

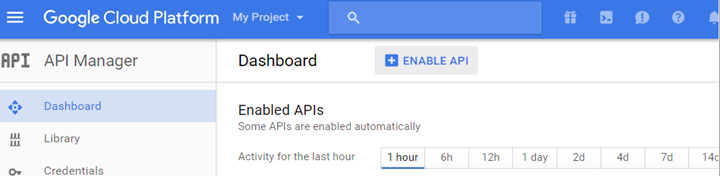
## **Part 1:Enabling the Google Cloud Compute Engine API and Dataproc API**

If this is the first time the Google Cloud Compute Engine or Dataproc service is being used on this account, the Google Cloud Compute Engine API (Application Programming Interface) and Dataproc API must be enabled. If this has already been enabled, please skip to the next part.

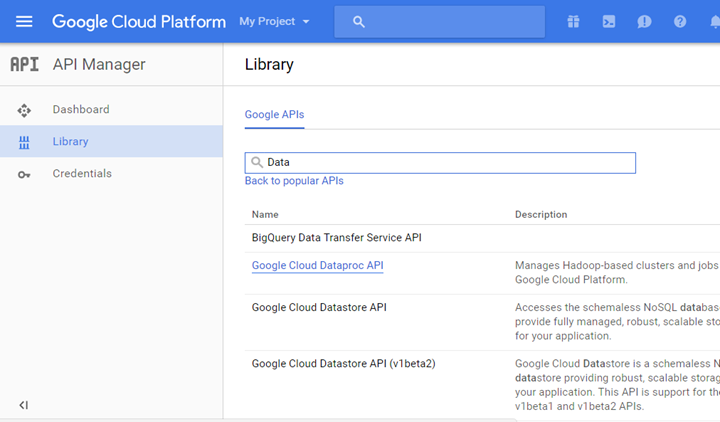
Step 1 - Visit the API Manager Dashboard screen.



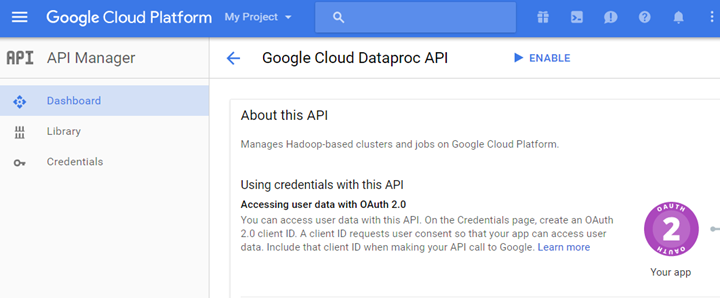
Step 2- On the API Manager Dashboard screen, click on the **ENABLE API** button.

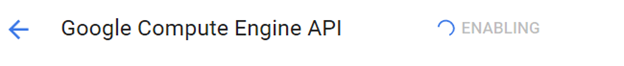


Step 3- Do a search for Dataproc.



Step 4- Select the Google Cloud DataProc from the list.

Step 5 - Click on the **ENABLE** button.

It may take a few minutes to enable this API  


Step 6- Repeat the above steps to enable the Compute Engine API.Click on the ENABLE button to enable the Compute Engine API.

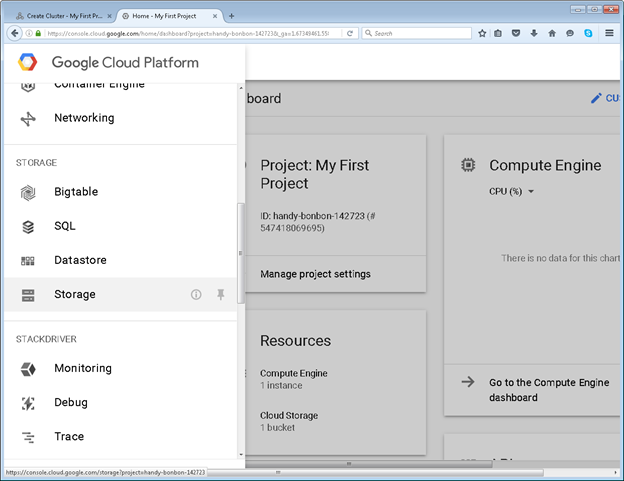
## **Part 2: Creating a Storage Bucket and Loading Data**

Step 1 - Log in to the Google Cloud account at [cloud.google.com](http://cloud.google.com)

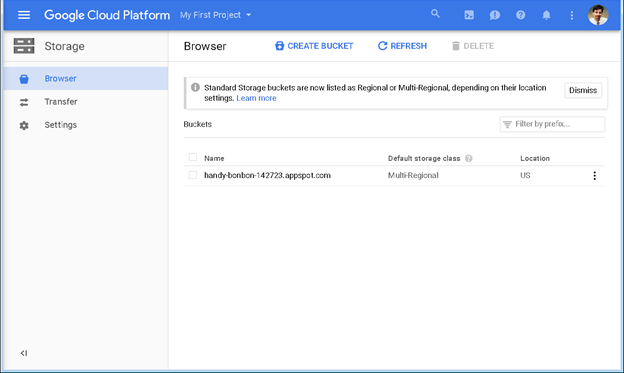
Step 2 - Click on the **Console** link in the upper right corner.

Step 3 - Click on the Products & Services icon (three horizontal bars) icon in the upper left corner

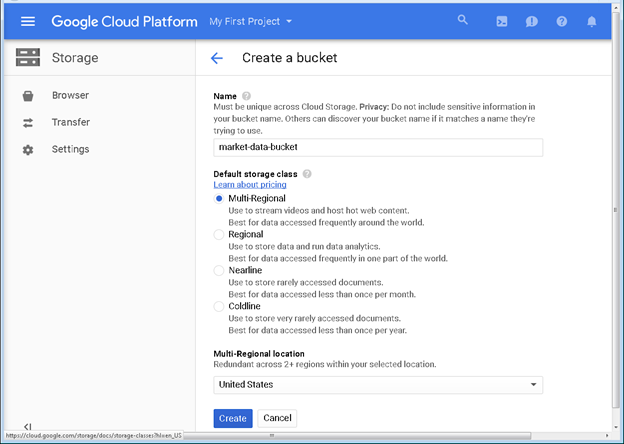
Step 4 - Scroll down to the Storage group and select Storage.



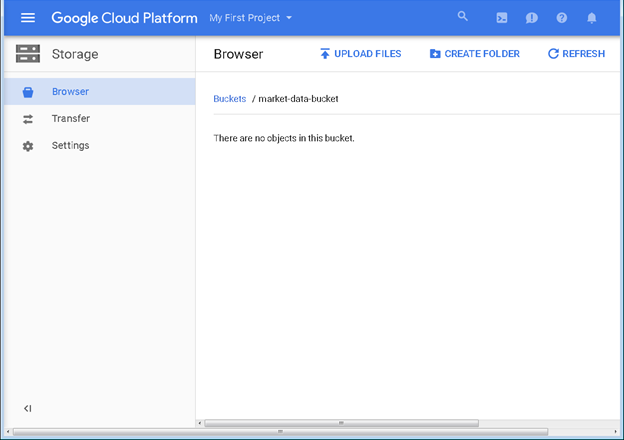
Step 5 - Click on the **Create Bucket** button.

Click on the **Create Bucket** but

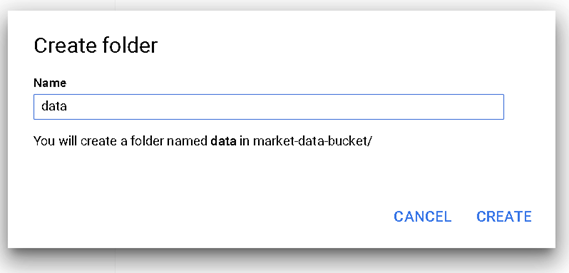
Step 6 - Fill in a name for the new bucket. You may wish to use a name that incorporates your initials or some other unique number to make it unique. Make a note of the bucket name as this will be used in later



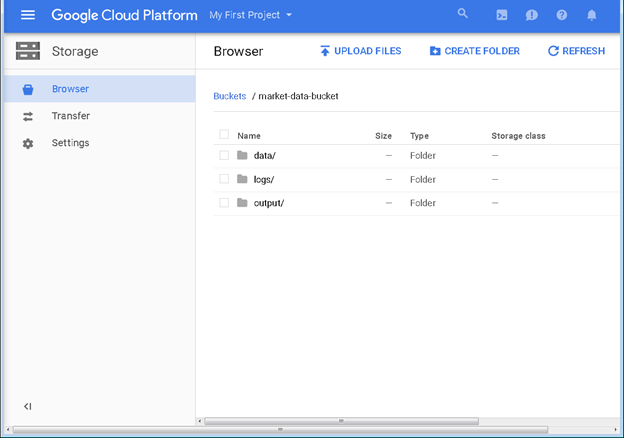
Step 7 - The new Bucket should now be selected. Click on the **Create Folder** button to create a new folder within this new bucket.



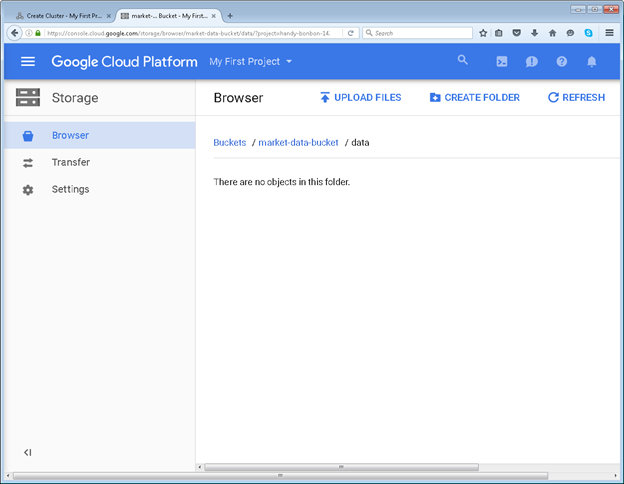
Step 8 - Name this new bucket “data” and then click the **CREATE** button.



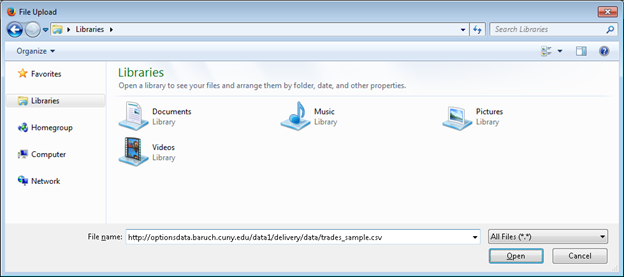
Step 9 - Repeat these steps to create two more folders named “output” and “logs”. When completed, the bucket and folders will appear as shown below:



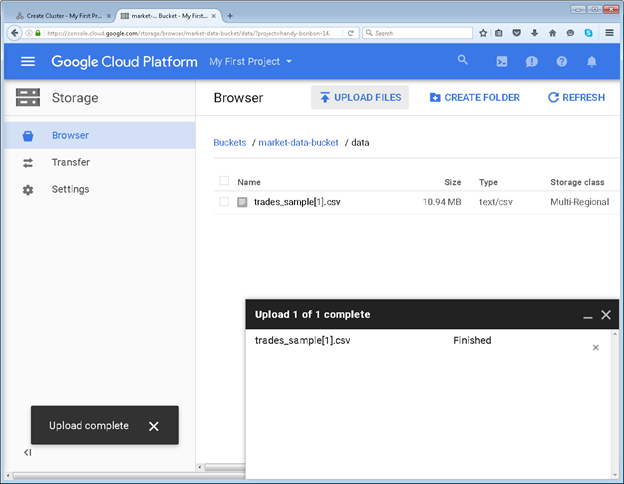
Step 10 - Navigate to the data folder and click the **UPLOAD FILES** button.



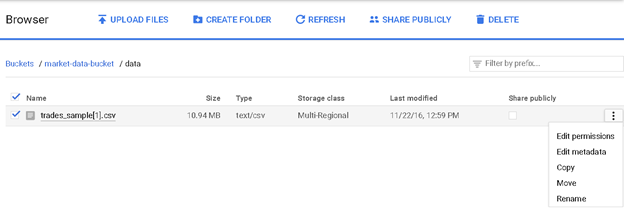
Step 11- When the **Open File** dialog box appears, paste in the following URL: http://optionsdata.baruch.cuny.edu/data1/delivery/data/trades\_sample.csv then click the **Open** button.

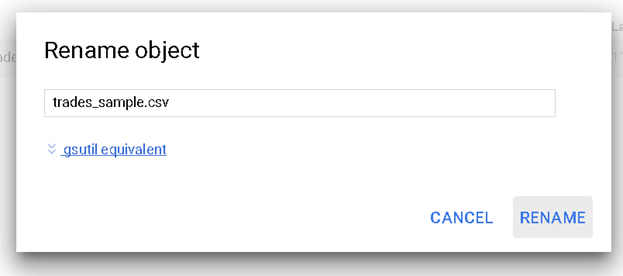


Step 12 - The file will be uploaded to Google Cloud storage into the market-data-bucket / data folder.



Step 13- Rename the new file to remove the [1] from the file name. Scroll all the way to the right and click on the three dots icon on the right-hand side. Click on the Rename menu item from the drop-down menu.



Step 14 - Change the name of the file to trades\_sample.csv and then click on the **RENAME** button.

Step 15 - At this point we now have a new storage bucket created with three new folders. A file named trades\_sample.csv has been stored in the data folder.

## **Part 3: Creating a Dataproc Cluster**

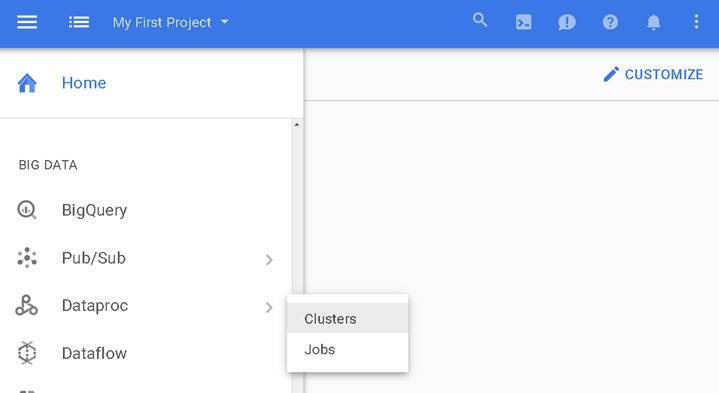
Now that the data is stored in the Google Cloud Storage, a Hadoop cluster can be created using the Google Cloud Dataproc services.

Step 16 - Log in to the Google Cloud system

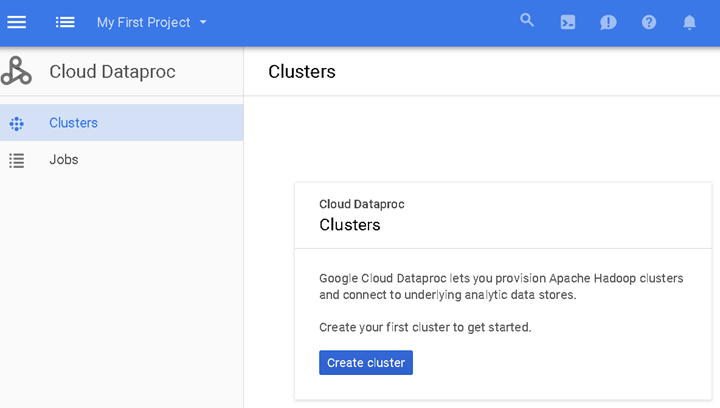
Step 17 - Click on the Console link in the upper right corner.

Step 18 - Click on the Products & Services icon (three horizontal bars) icon in the upper left corner.

Step 19 - Scroll down to the BigData group and select Dataproc and then click on the Clusters from the fly-out menu.



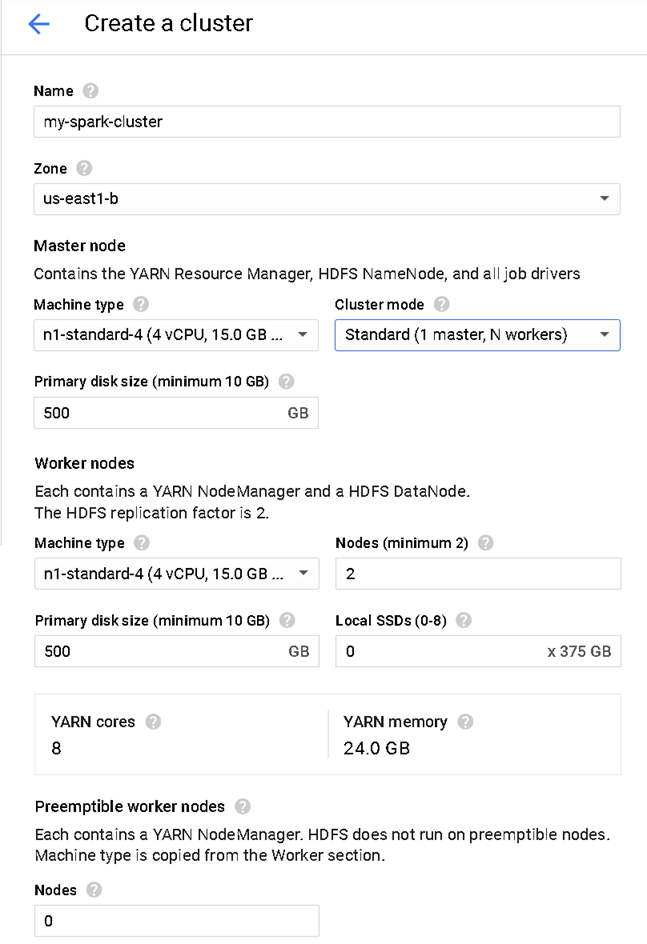
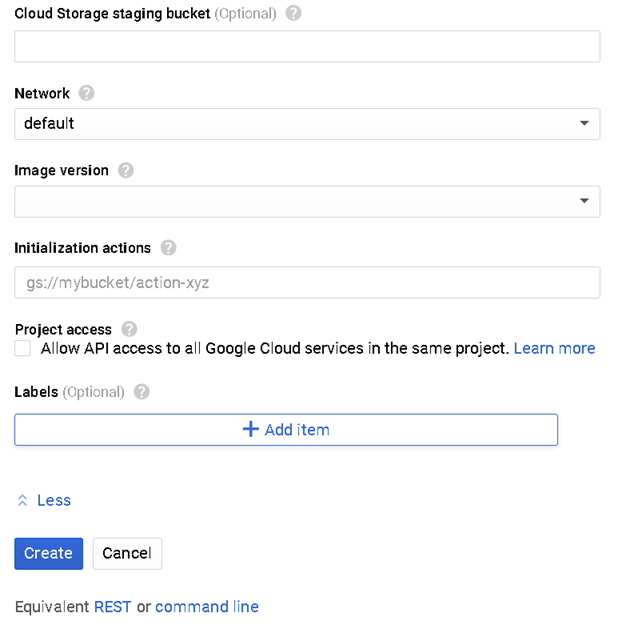
Step 20 - Click on the blue **Create Cluster** button to create a new cluster



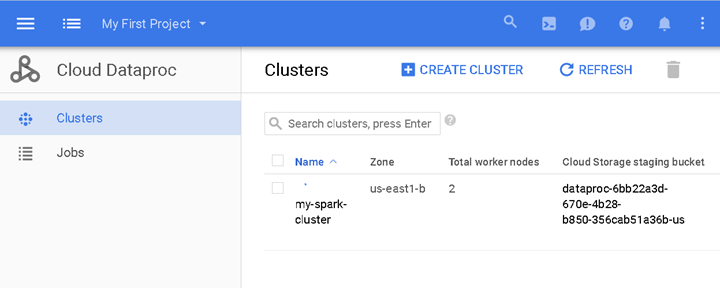
Step 21- Fill in the settings for the new cluster. Examples are provided.  
– Give a **Name** for the Cluster: my-spark-cluster  
– Select a **Zone** (Region of the world): us-east-c – Or pick the Zone that is closest to you, or least expensive, etc.

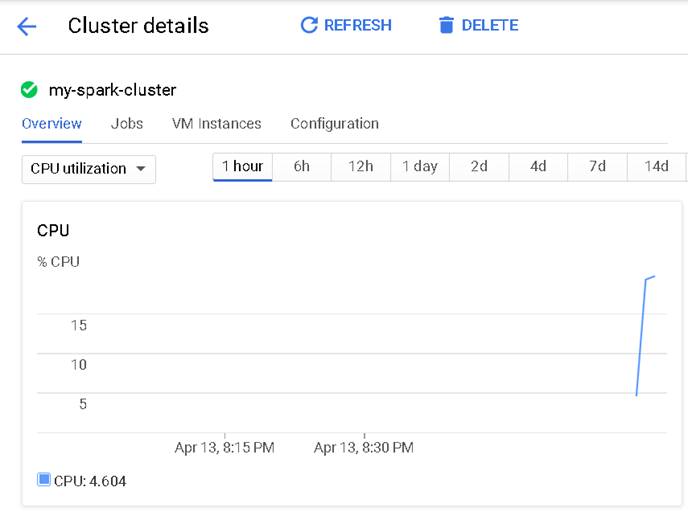
Step 22- Set up the Master Node that will host the scheduler (YARN), the Hadoop Distributed File System (HDFS) Master Node. The settings are:  
– **Machine Type**: n1-standard-n4 (4 vCPU, 15GB RAM) This is powerful enough for the examples used in this tutorial  
– **Cluster Mode**: Standard (1 master)  
– **Primary Disk Size**: 500 GB Increase this if your data size will be any larger.

Step 23 - Now set up the worker nodes. Each worker node will contain the YARN task manager and act as an HDFS storage node. Select a Machine Type, Number of Nodes and Primary Disk storage to match the planned workload. The choices given below are more than enough for the sample data used in this tutorial:  
– **Machine Type**: n1-standard-n4 (4 vCPU, 15GB RAM)  
– Number of **Nodes**: 3  
– **Primary Disk Size**: 500 GB  
– **Local Solid State Drives (SSD)**: 0  
– **YARN Cores**: 8  
– **YARN Memory**: 24 GB

Step 24- The rest of the options under “Preemptible workers, bucket, network, version, initialization, & access options” can be left at the default settings. The complete set of parameters is shown in the figure below:  


Step 25 - When finished click on the Create button to create the cluster.

Step 26 - The new cluster will appear in the Clusters console and will be initializing at this point:

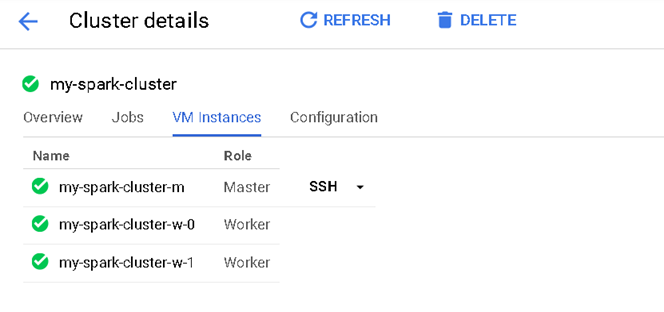
Step 27 - Once the cluster is completely initialized, click on the name of the cluster to bring up the **Overview** screen:

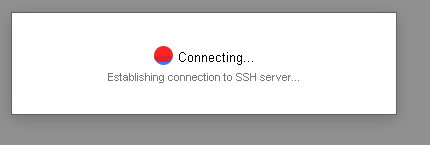
Once the cluster status changes to **Running**, you may connect to the Master Node using Secure Shell (SSH). Those steps are given in the next section.

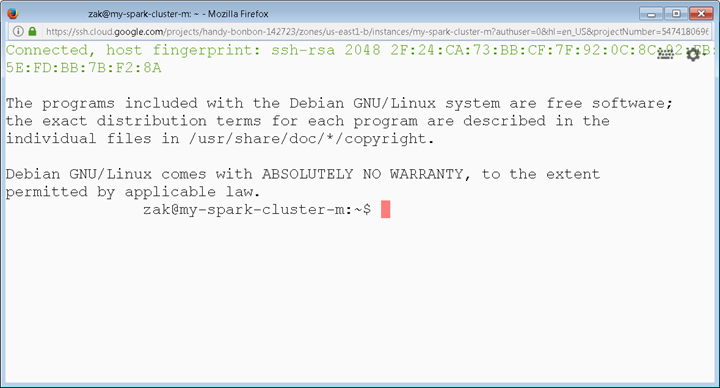
## **Part 4: Connecting to the Master Node using Secure Shell (ssh)**

Now that the cluster is running, connect to the cluster using secure shell. Google provides a great web browser-based secure shell client so there are no keys to manage or extra software to install.

Step 28 - Click on the name of the cluster and then click on the **VM Instances tab**



Step 29 - Click on the SSH link to open up a Secure Shell window. A new browser window will open, and an icon will appear in the center alerting you to the SSH connection being set up.

Step 30 - Once the SSH connection is established, the shell prompt will appear.

Step 31 - The account is the same one you used to create your Google Cloud account and the local host name will be the name of the cluster followed by a -m to indicate this is the Master Node. The shell prompt will contain these two names put together. For example:

myusername@my-spark-cluster-m:~$

Step 32 - Make a note of the username as it will be different from what is shown above (myusername). Substitute the username you see for myusername

Note that the operating system on each of the nodes is Linux. So just about any normal Linux commands should work. See Lab 1 ( Intro to Linux)for an overview of the OS, the file system and various commands. For example, you should learn how to:

List files: ls

Copy files: cp

Print current directory: pwd

Edit a file: nano (or pico)

Step 33- Confirm the data in the Google Storage is still available by issuing the gsutil ls command to obtain a list of files in the data folder. Replace the market-data-bucket the bucket name you created in the prior section.

myusername@my-spark-cluster-m:~$ gsutil ls gs://market-data-bucket/data

gs://market-data-bucket/data/

gs://market-data-bucket/data/trades\_sample.csv

### **Loading Additional Data to GC Storage from the Master Node**

The sample data file trades\_sample.csv was previously loaded on the Google Cloud Storage in folder market-data-bucket/data. If more data needs to be loaded into Google Cloud Storage, this can be accomplished at this time from the Dataproc Master Node. For example, use the curl command to download a file from the web to the Master Node file system. Then use an gsutil command to copy or move the file to the Google Cloud Storage. For example to download the trades\_sample.csv file again:

myusername@my-spark-cluster-m:~$

curl -O http://optionsdata.baruch.cuny.edu/data1/delivery/data/trades\_sample.csv

Then, to copy this file to Google Cloud Storage use this gsutil cp command:

gsutil cp trades\_sample.csv gs://market-data-bucket/data/

If the file is very large and there is no space on the local file system to store it, the file can be streamed into the gsutil cp command using this syntax:

curl -L http://optionsdata.baruch.cuny.edu/data1/delivery/data/trades\_sample.csv | gsutil cp - gs://market-data-bucket/data/trades\_sample.csv

In the above example, the gsutil cp command is getting its input file to copy from the piped output of the curl command.

In all cases, use the name of the bucket as created in the prior section. At this point the Cluster is running and the data files are accessible on Google Cloud Storage. The next section will introduce the Spark-SQL command line interface.

## **Part 5: Run the Spark-SQL command line interface to issue SQL Statements.**

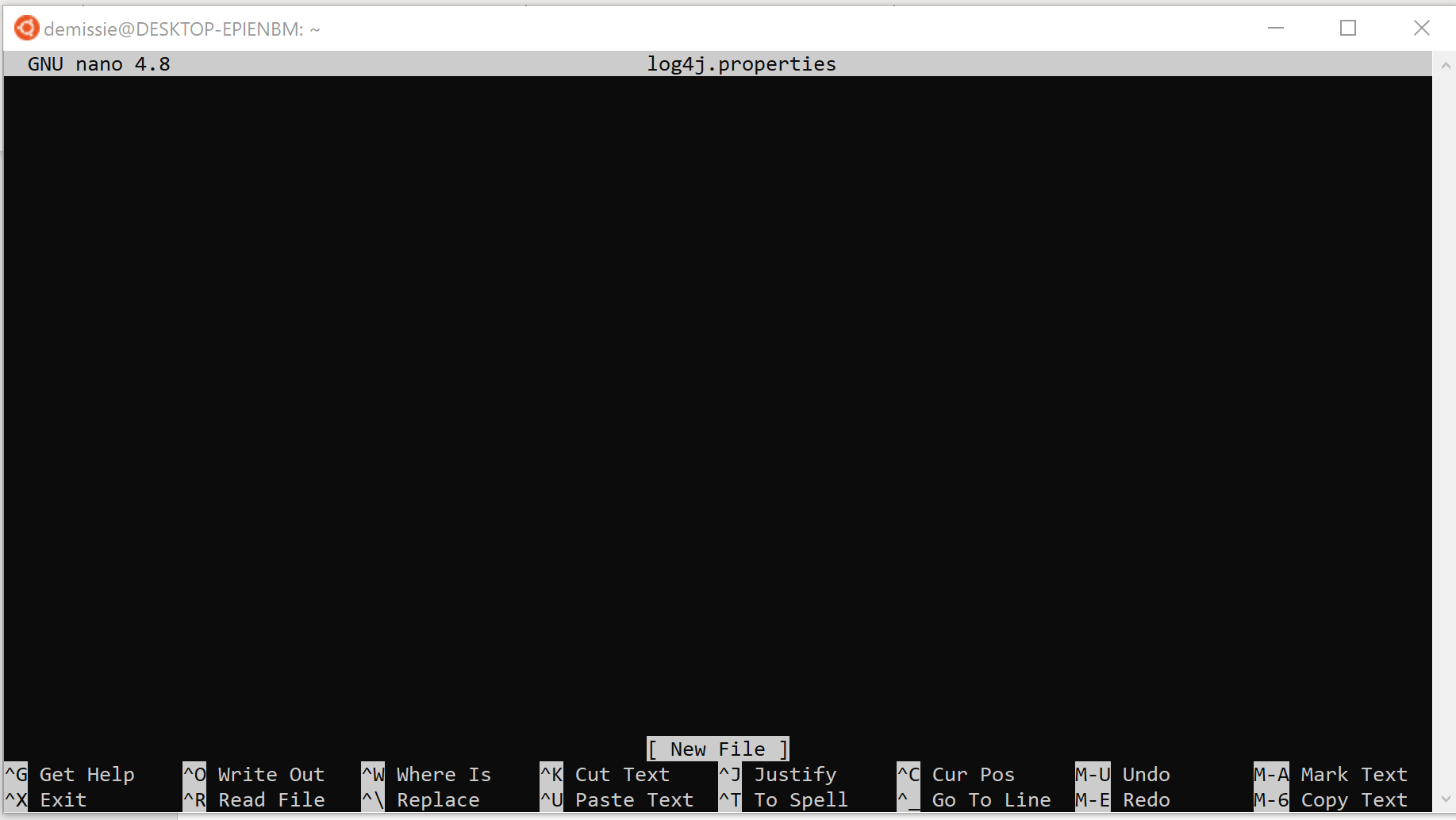
Apache Spark has a number of interactive interfaces for Java, Scala, Python and R. For this tutorial, all we need to do is directly issue some SQL statements. The spark-sql command line can be used to accomplish this.

Before running the spark-sql command, a Log4j configuration file will be created to suppress the logging of INFO messages. Without this step, the command-line will be quickly overwhelmed with these log messages.

Step 34 - Make a configuration file named log4j.properties using the nano text editor. Type the following command at the cluster prompt:

myusername@my-spark-cluster-m:~$ **nano log4j.properties**

This will launch the Nano text editor and create a new file:



Step 35 - Copy and paste in the following lines into the Nano text editor window:

# Set everything to be logged to the console

log4j.rootCategory=WARN, console

log4j.appender.console=org.apache.log4j.ConsoleAppender

log4j.appender.console.target=System.err

log4j.appender.console.layout=org.apache.log4j.PatternLayout

log4j.appender.console.layout.ConversionPattern=%d{yy/MM/dd HH:mm:ss} %p %c{1}: %m%n

# Settings to quiet third party logs that are too verbose

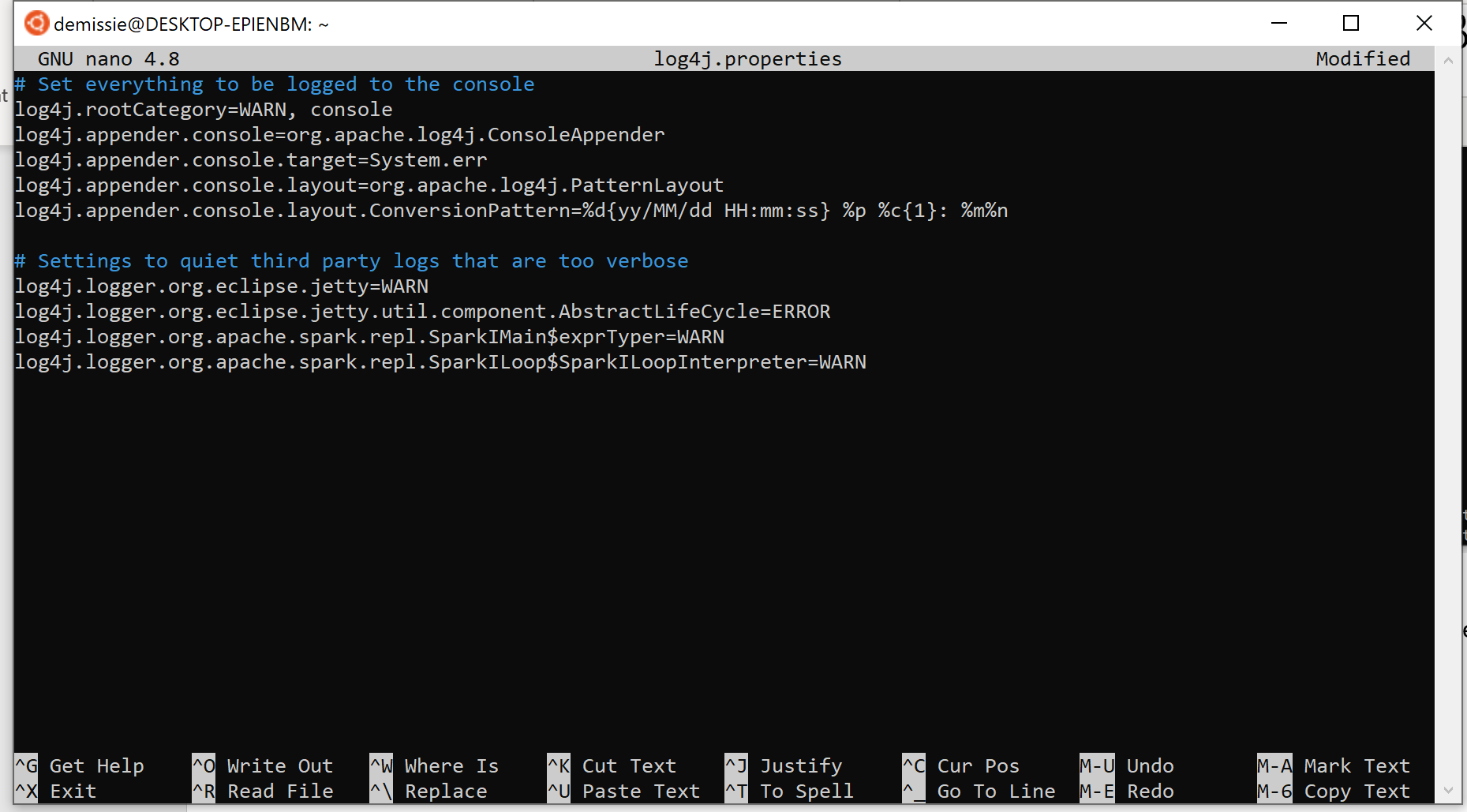
log4j.logger.org.eclipse.jetty=WARN

log4j.logger.org.eclipse.jetty.util.component.AbstractLifeCycle=ERROR

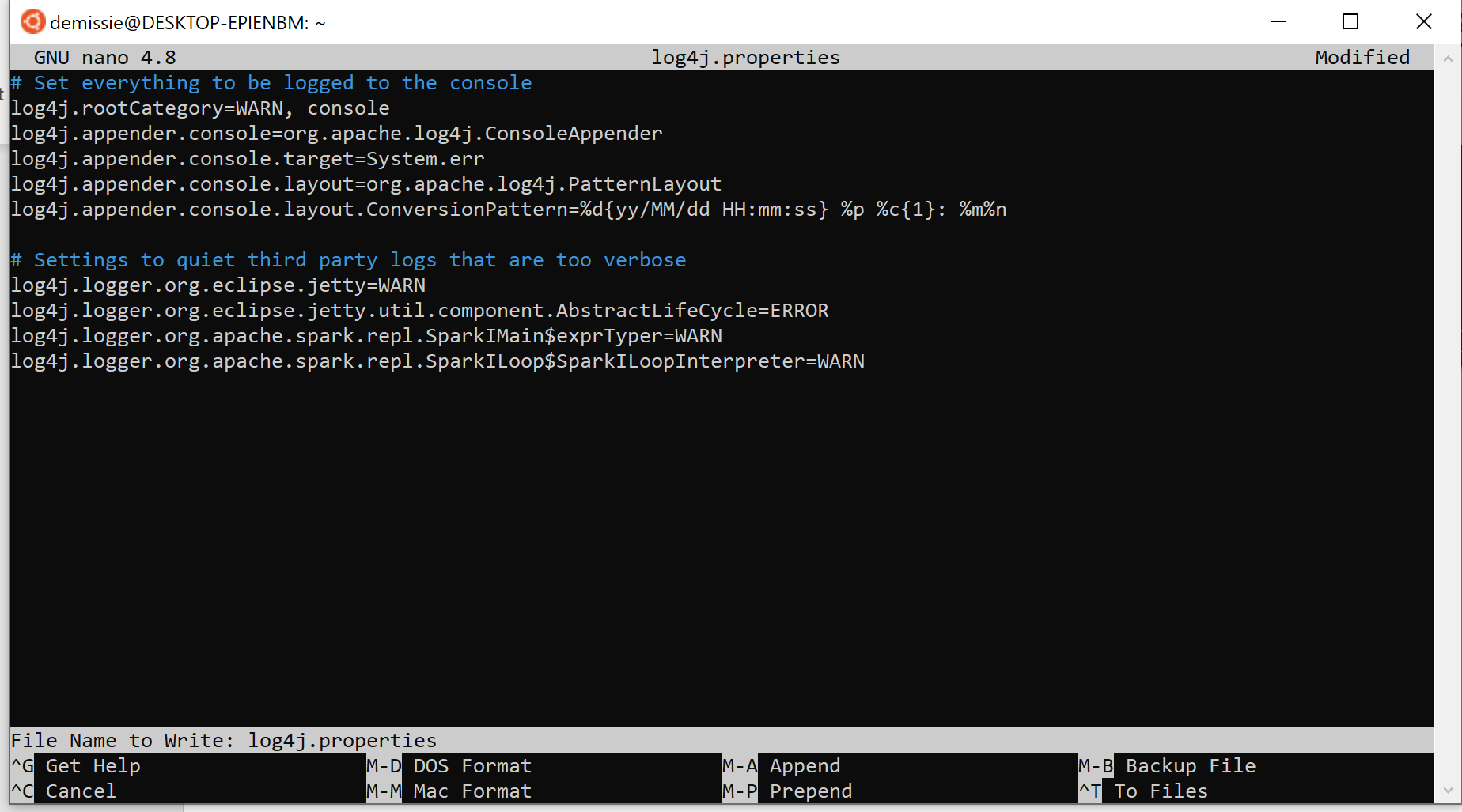
log4j.logger.org.apache.spark.repl.SparkIMain$exprTyper=WARN

log4j.logger.org.apache.spark.repl.SparkILoop$SparkILoopInterpreter=WARN

The result is shown below:

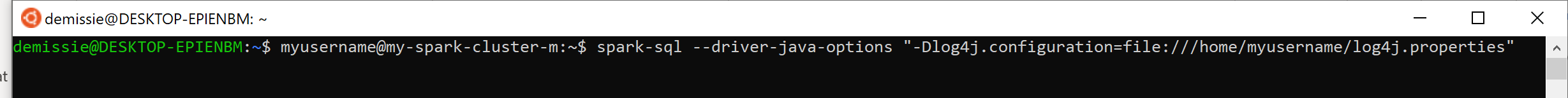


Step 36 - Press ^X to exit the Nano text editor (hold down the Control key and press X). You will be prompted to save the file:



Step 37 - Press Y to save the file. A prompt will appear for the file name and press Enter and the new file will be created.

Step 38 - Launch Spark-sql and reference the log4j configuration on the command line options:



Note: Replace the myusername in the above command with your user name on the cluster.

Step 39 - The spark-sql command will initialize:

ivysettings.xml file not found in HIVE\_HOME or HIVE\_CONF\_DIR,/etc/hive/conf.dist/ivysettings.xml will be used

17/04/14 01:18:38 WARN ObjectStore: Version information not found in metastore. hive.metastore.schema.verification is not

enabled so recording the schema version 1.2.0

17/04/14 01:18:38 WARN ObjectStore: Failed to get database default, returning NoSuchObjectException

17/04/14 01:18:46 WARN SparkContext: Use an existing SparkContext, some configuration may not take effect.

ivysettings.xml file not found in HIVE\_HOME or HIVE\_CONF\_DIR,/etc/hive/conf.dist/ivysettings.xml will be used

spark-sql>

Once the spark-sql> prompt appear, Spark is ready to accept SQL statements.

### **Loading Data into Spark**

The sample data file trades\_sample.csv is already loaded on the Google Cloud Storage (gs:) in folder market-data-bucket/data.

There are two main ways of working with data under Spark:

* Use CREATE TABLE to move the data from the HDFS (gs: in this case) into the Spark file system as a Spark DataFrame. Once the data is moved into the Spark Dataframe, the file is removed from the regular HDFS.
* Use CREATE EXTERNAL TABLE that will leave the source data file in the HDFS. Using this approach, the data stays where it is (gs: bucket in this example) but the table can be manipulated using the same SQL statements.

For this example, create an EXTERNAL table with a given column structure that will use the existing data file in place. When writing SQL at the spark-sql> prompt, do not use the TAB character. The syntax is:

CREATE EXTERNAL TABLE trades\_sample

(trading\_date\_time TIMESTAMP,

network CHAR(1),

message\_category CHAR(1),

message\_type CHAR(1),

message\_sequence BIGINT,

market\_exchange CHAR(1),

symbol VARCHAR(10),

trade\_price DOUBLE,

trade\_size BIGINT,

trade\_conditions VARCHAR(6),

trade\_conditions2 VARCHAR(6) )

ROW FORMAT DELIMITED FIELDS TERMINATED BY ','

LOCATION 'gs://market-data-bucket/data/';

Note that the LOCATION clause in the CREATE TABLE statement must point to the bucket and folder containing the data. It is also possible to point to a specific file. For example, to reference a specific file in a folder use this syntax:

LOCATION 'gs://market-data-bucket/data/trades\_sample.csv';

To use all of the files in a folder as part of the same table, just specify the name of the bucket and folder as in:

LOCATION 'gs://market-data-bucket/data/';

Bucket names and folder names are case-sensitive. The complete path should be enclosed in simple single quotes (some text editors make “fancy quotes” so be careful when copying and pasting in commands). Be sure to use the bucket name provided in the first section of the tutorial when the bucket was first created. Do not use the TAB key to indent. Instead, just use space.

Here is the CREATE EXTERNAL TABLE command after typing into Beeline:

spark-sql> CREATE EXTERNAL TABLE trades\_sample

> (trading\_date\_time TIMESTAMP,

> network CHAR(1),

> message\_category CHAR(1),

> message\_type CHAR(1),

> message\_sequence BIGINT,

> market\_exchange CHAR(1),

> symbol VARCHAR(10),

> trade\_price DOUBLE,

> trade\_size BIGINT,

> trade\_conditions VARCHAR(6),

> trade\_conditions2 VARCHAR(6) )

> ROW FORMAT DELIMITED FIELDS TERMINATED BY ','

> LOCATION 'gs://market-data-bucket/data/';

Time taken: 6.294 seconds

spark-sql>

To view the structure of a table, use the DESCRIBE command:

spark-sql> DESCRIBE trades\_sample**;**

trading\_date\_time timestamp NULL

network string NULL

message\_category string NULL

message\_type string NULL

message\_sequence bigint NULL

market\_exchange string NULL

symbol string NULL

trade\_price double NULL

trade\_size bigint NULL

trade\_conditions string NULL

trade\_conditions2 string NULL

Time taken: 0.323 seconds, Fetched 11 row(s)

spark-sql>

To see how many records are in the table, run a simple query:

spark-sql> SELECT COUNT(\*) FROM trades\_sample**;**

[Stage 2:> (0 + 2) /

[Stage 2:==> (1 + 2) /

[Stage 2:=====> (2 + 2) /

[Stage 2:========> (3 + 2) /

[Stage 2:===========> (4 + 2) /

[Stage 2:=================> (6 + 2) /

[Stage 2:=======================> (8 + 2) /

[Stage 2:==========================> (9 + 2) /

[Stage 2:===============================> (11 + 4) /

[Stage 2:==================================> (12 + 4) /

[Stage 2:=====================================> (13 + 4) /

[Stage 2:==========================================> (15 + 4) /

[Stage 2:=============================================> (16 + 4) /

[Stage 2:================================================> (17 + 3) /

[Stage 2:===================================================> (18 + 2) / 192220

Time taken: 9.727 seconds, Fetched 1 row(s)

spark-sql>

The CPU Utilization (and other performance parameters) for the cluster can be viewed on the Cluster Overview screen as shown below:



## **Test drive your database:**

1. Retrieve a list of trades that happen during the first minute of trading at 9:30am.
2. Find the total trading volume for each stock before noon.
3. List the top 3 stocks with highest trading volume during the first minute of trading, along with their trading volume.
4. List the top 3 stocks with highest trading volume during the last minute of trading, along with their trading volume.
5. List the top 3 stocks with highest trading volume during noon time, along with their trading volume.
6. Find the total trading volume for the entire market before noon.
7. Find the total trading volume for the entire market after noon.
8. List the top 3 stocks with highest trading volume for the day, along with their trading volume.
9. Find the time that have the highest trading volume through the day.
10. Find the time that have the lowest trading volume through the day.

## **Part 6: Shutting down the Cluster.**

Once the work is done on the cluster, be sure to shut it down. Note that any data stored in the cluster will be deleted.

Return the Google Cloud Dataproc Cluster Details screen as shown below:

