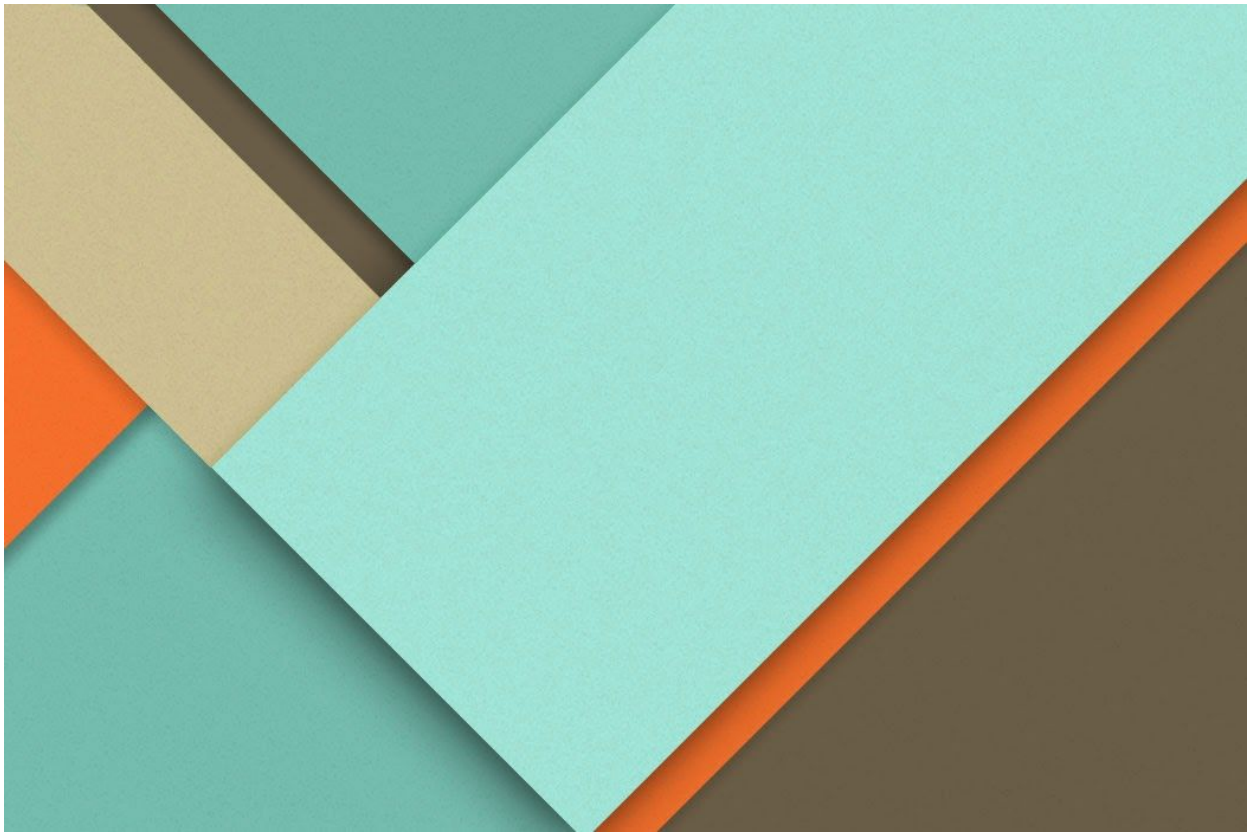


Big Data Wrangling

12.06.2020

By: Krzysztof Rafalont

Overview

Welcome to my Big Data Wrangling with Google Books Ngrams. Within this documentation, I will apply Big Data Fundamentals to load, filter, and visualize a large real-world dataset in a cloud-based distributed computing environment using Hadoop, Spark, and the S3 filesystem.

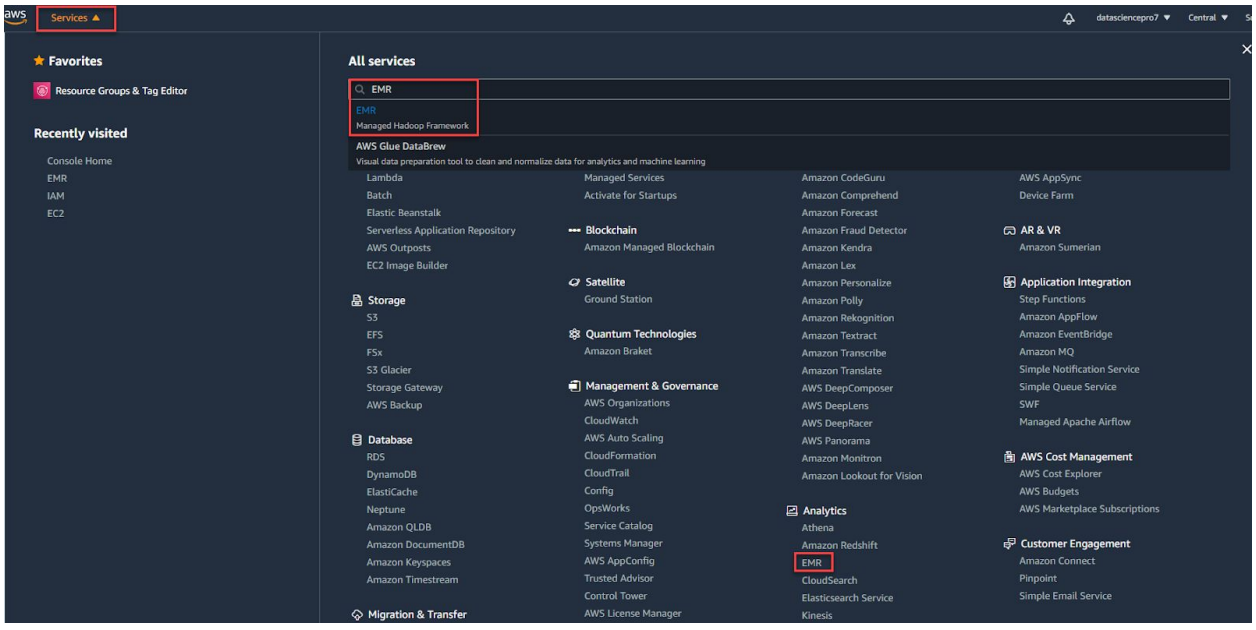
In order to do this I require an appropriate dataset. For this, I will be using the Google Ngrams dataset. The content of this dataset is a corpus of n-grams compiled from data from Google Books. These are digitized texts representing approximately 4% of all books ever printed, and span a time period from the 1800s into the 2000s.

With the combination of my skill set and using the data outlined above, I will follow a Big Data analysis workflow. The workflow and steps in the process are illustrated below:

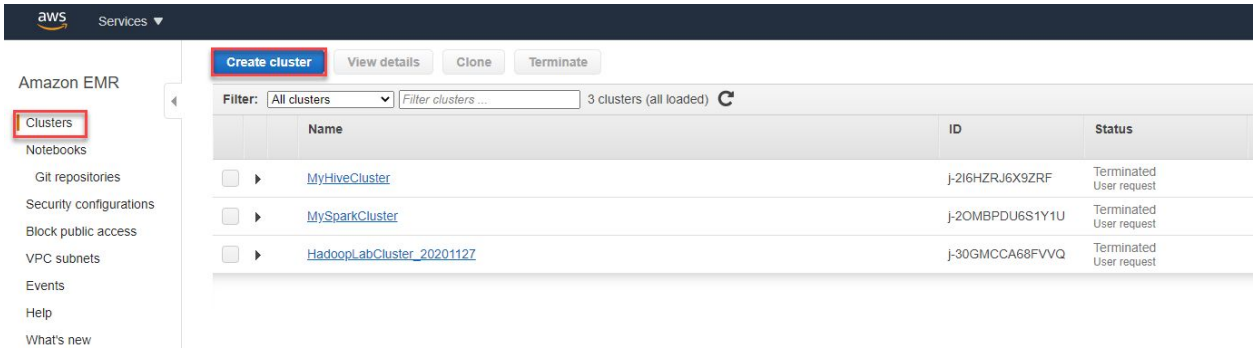
Spin up a New EMR Cluster using AWS Console

To Spin up a new EMR cluster using the AWS Console, we must first go to aws.amazon.com and login by clicking the “Sign in to the Console” button in the top right corner of the page and use our personal login credentials.

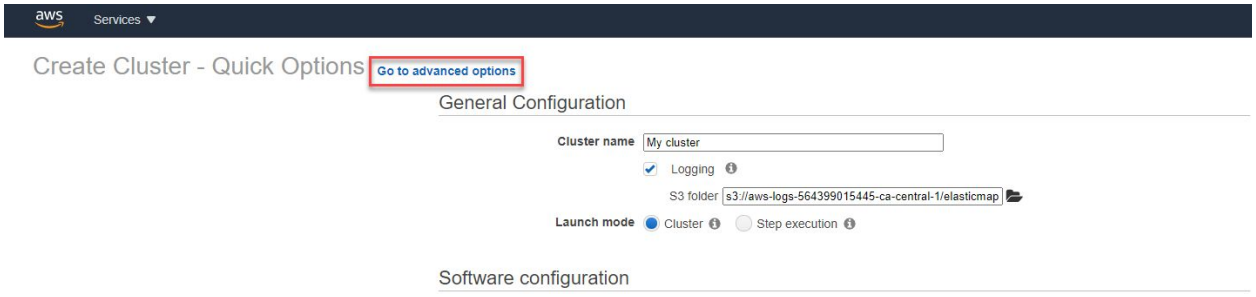
Once logged in, we can click on the “Services” tab in the top right corner of the page and use the search bar to look up “EMR”. You can also simply select “EMR” under the “Analytics” column.



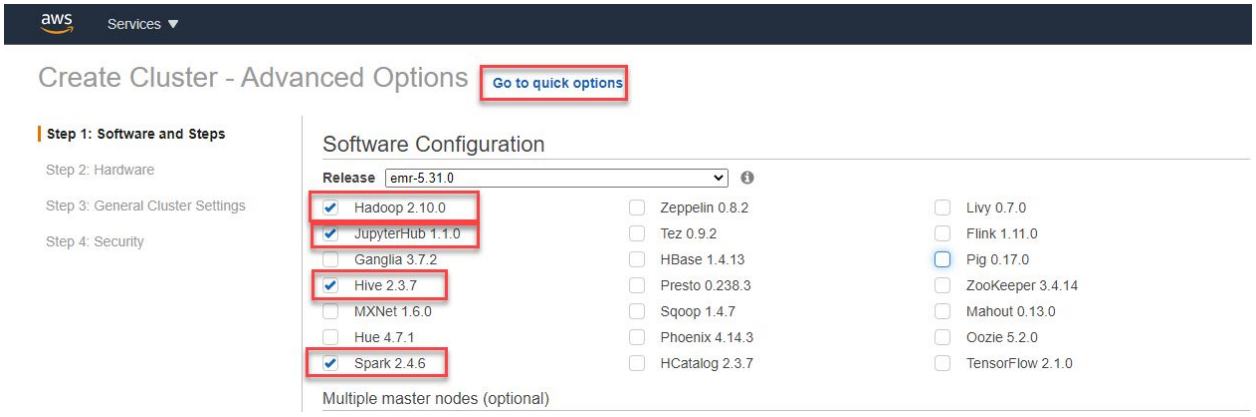
Ensure you are in the “Clusters” section of the EMR, and click the blue “Create cluster” button.



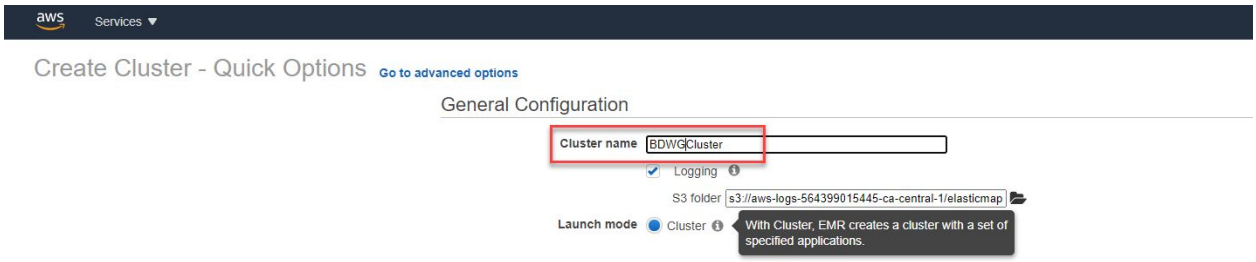
Before making any changes on this page, click on “Go to advanced options”.



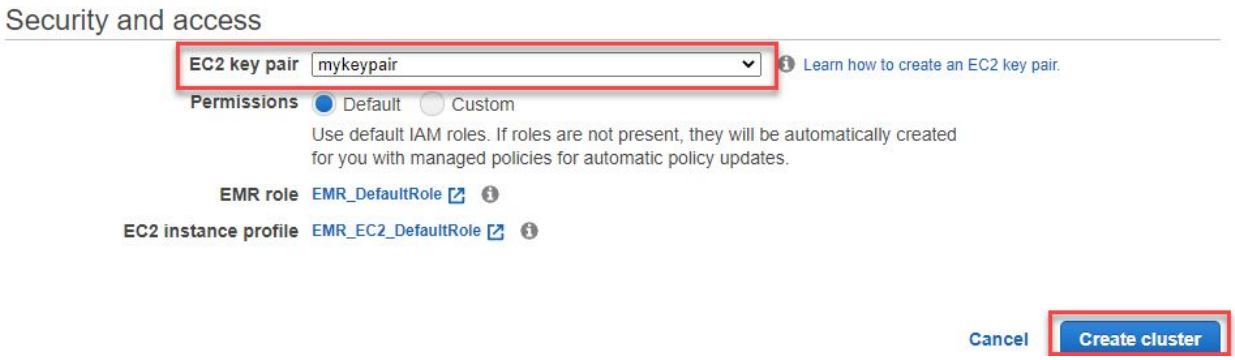
Under “Software Configurations” be sure to include **Hadoop**, **Hive**, **Spark**, and **Jupyterhub** for your cluster. Then click on “Go to quick options” to return to the previous page.



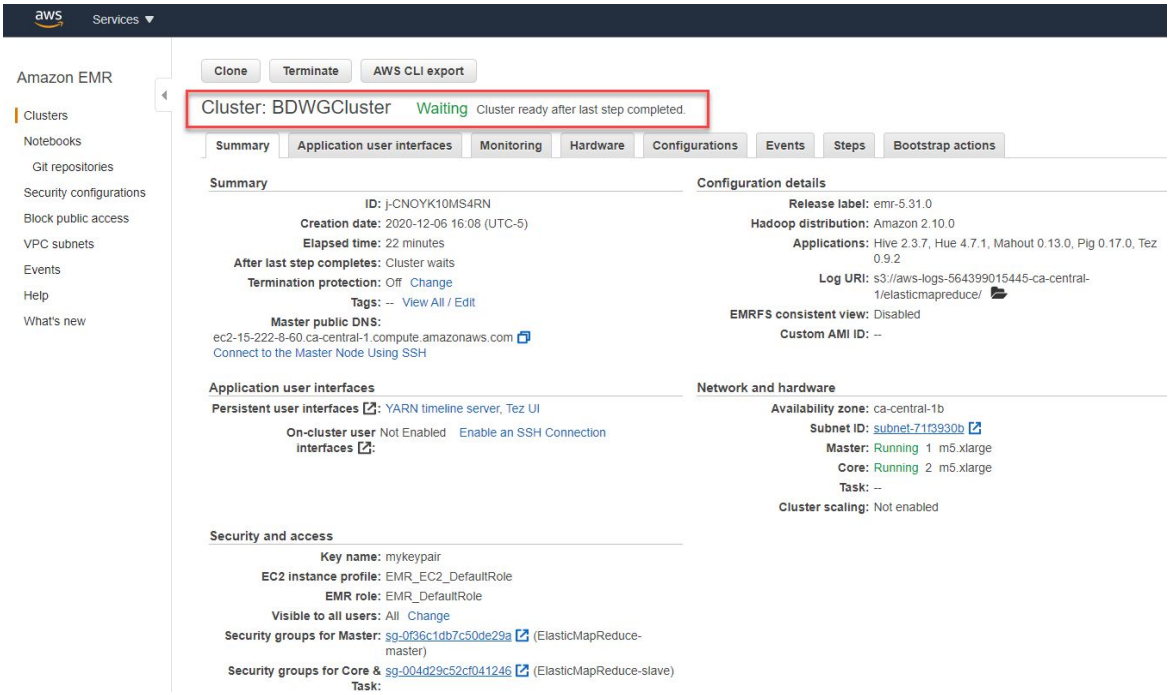
If you choose to do so, you can name the new cluster. I will name mine, “BigDataCluster”.



Only other change necessary to make is at the bottom of the page under “Security and access”. Select an existing EC2 key pair. In my case I will use “mykeypair”. Once completed, click the blue “Create cluster” button at the bottom right of the page.



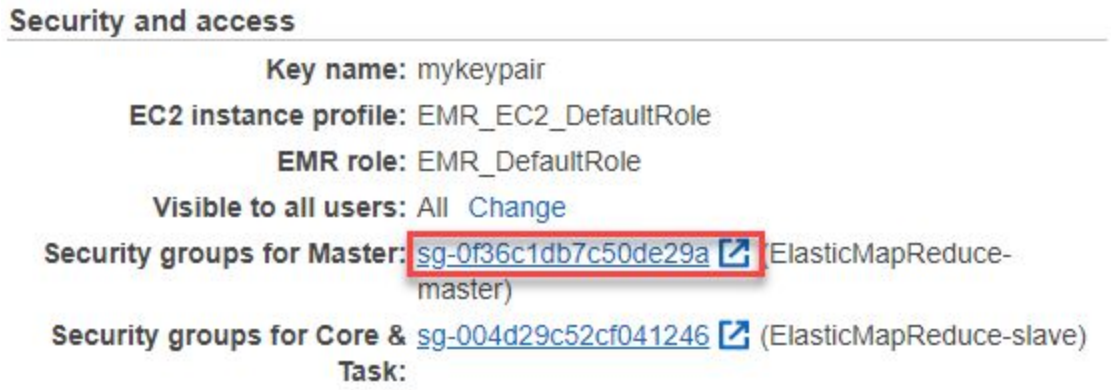
You should now be able to see your cluster. Here is mine:



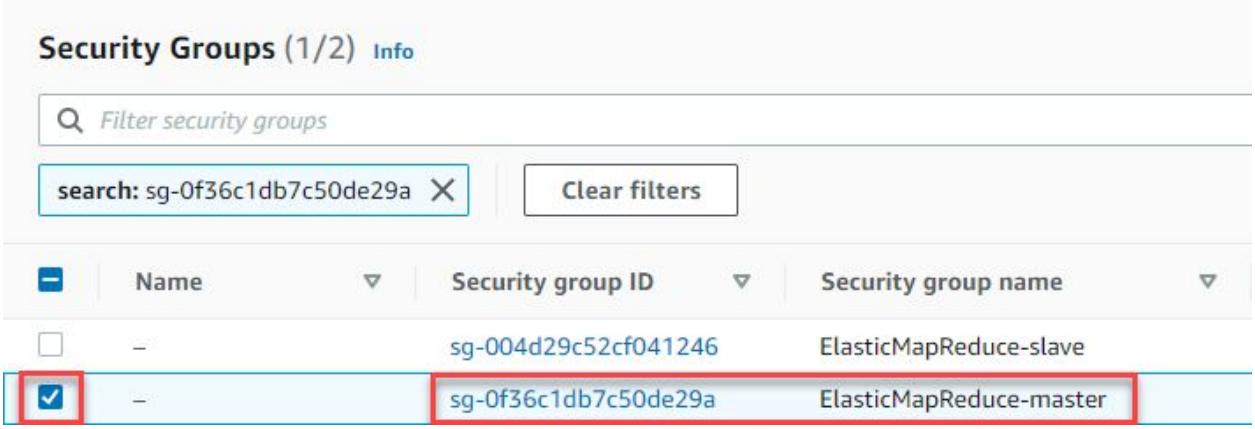
Connect to the Head Node of Cluster using SSH

Now that we have created our EMC cluster we can connect to the head node of the cluster using SSH. For this we need an EC2 key pair, I will use my existing key named “mykeypair”.

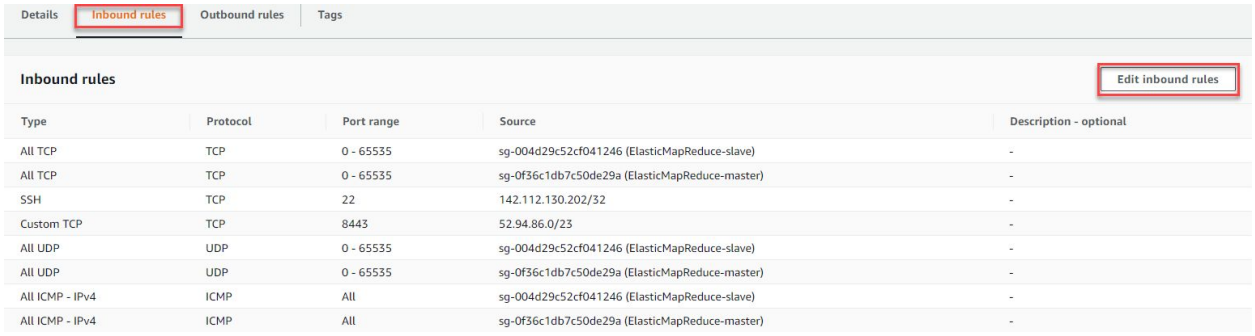
To do this we simply click on the link under “Security and access” next to “Security groups for Master”.



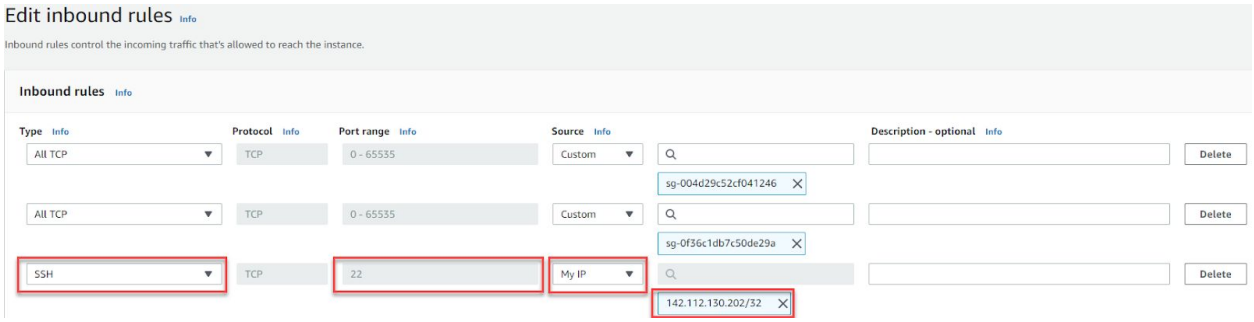
Select the master security group.



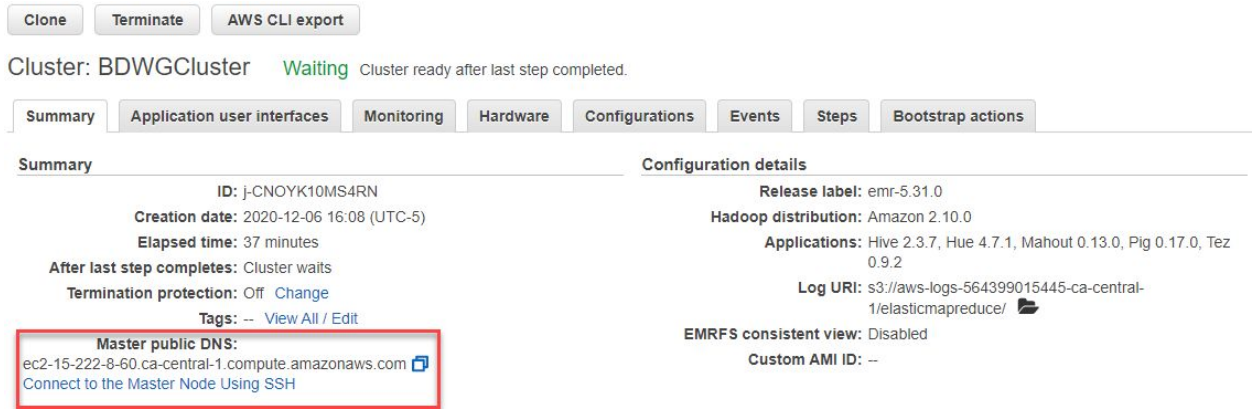
Navigate to the “Inbound Rules” tab and hit the “Edit Inbound Rules” button.



Under “SSH” make sure the port range is “22” . Because I am running this on a Windows PC I will select “MyIP” for the source, this will automatically populate with the IP address I am about to connect with. Save these changes once completed.



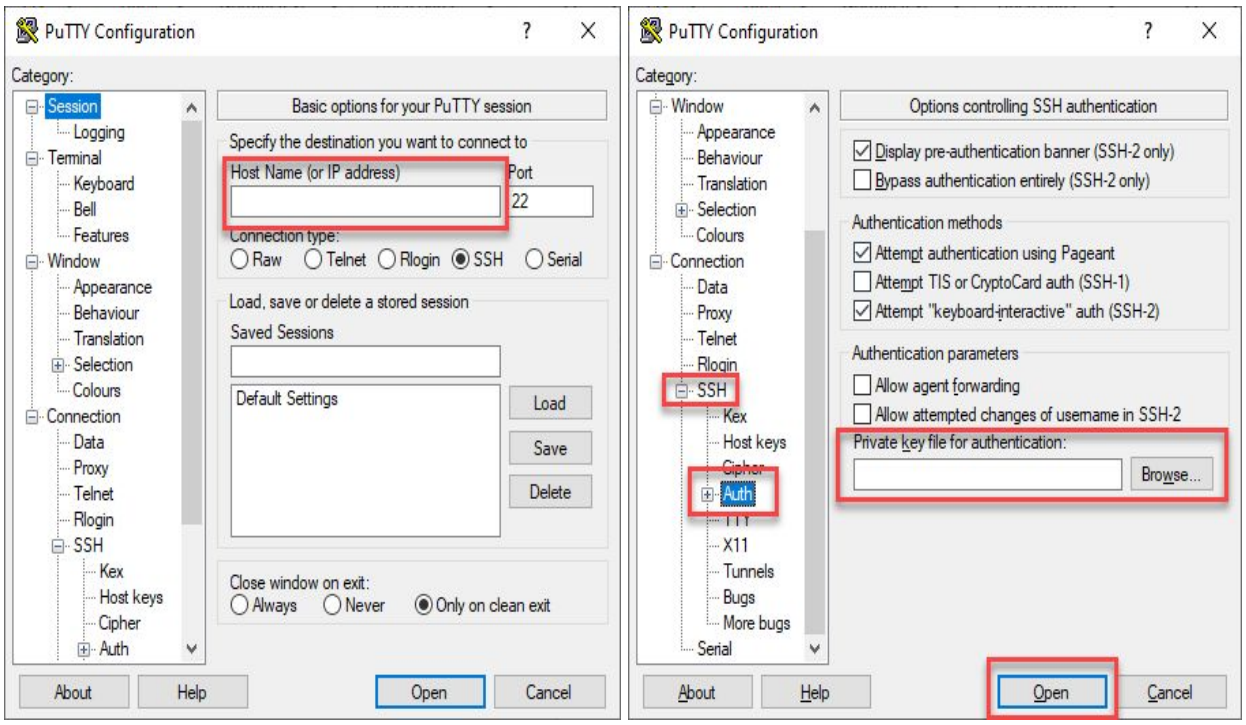
Next we return back to our BigDataCluster and click on the SSH link found under “Master public DNS” which will give you the instructions necessary from here on out.



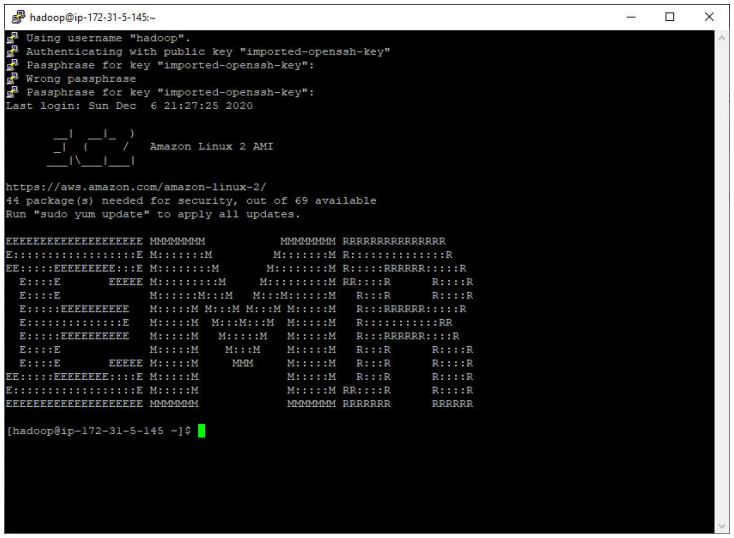
Again, since we are on a Windows PC, from those instructions we need to copy the Host Name provided to us. In my case, Host Name is:

hadoop@ec2-15-222-8-60.ca-central-1.compute.amazonaws.com

Next, we need to open PuTTY, paste the Host Name into the given field. Then under the category “SSH” on the left side, navigate to “Auth”. Then browse and select the “mykeypair.ppk” file and click “Open”.



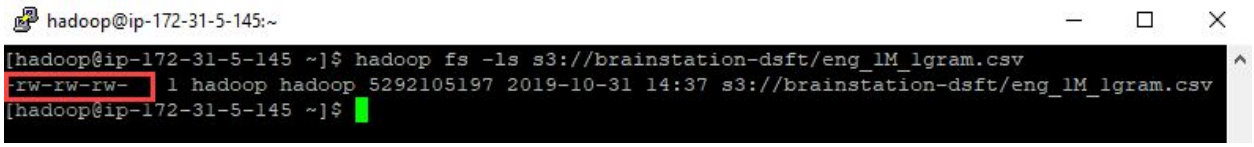
And we’re set! We can determine if the connection is successful by the EMR ASCII banner. We are now logged in to the head node of my EMR Cluster.



Copy Data from S3 Bucket into Hadoop File System

Now that we have created our new EMR cluster named "BigDataCluster" and connected to the head node of the cluster using SSH. We can proceed with copying the data folder from the S3 bucket into the **/user/hadoop/eng_1M_1gram** directory in the HDFS.

First, let's check if the S3 file that we are trying to copy exists in the S3 bucket using **hadoop fs -ls s3://brainstation-dsft/eng_1M_1gram.csv**.



As shown above, we can see **-rw-rw-rw-** with further information that follows indicating that the S3 file we are trying to copy does in fact exist.

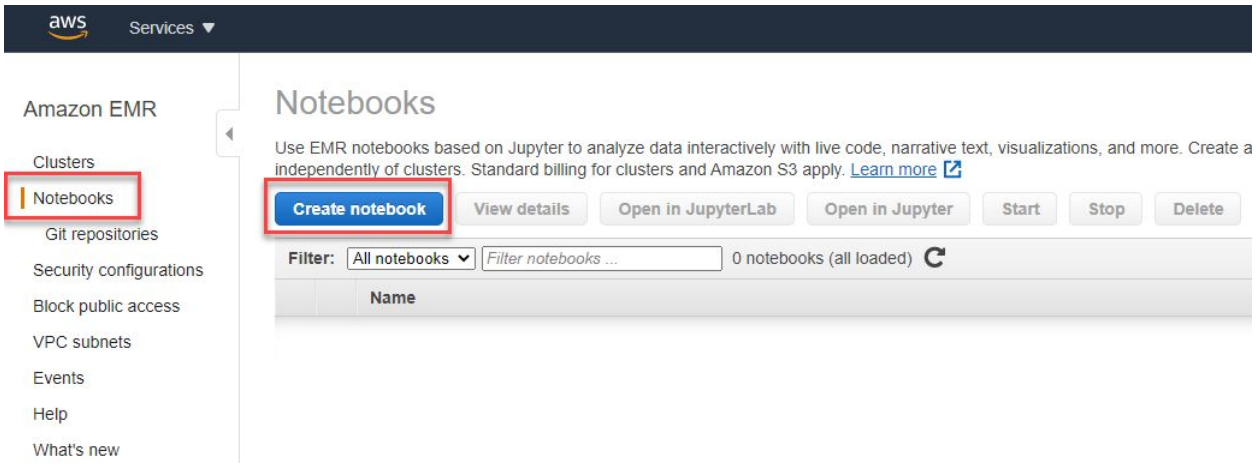
Now let's copy the S3 file into the **/user/hadoop/eng_1M_1gram** directory in the Hadoop file system (HDSF) using the **hadoop distcp s3://brainstation-dsft/eng_1M_1gram.csv /user/hadoop/eng_1M_1gram** command.

Just to be certain everything copied over accordingly, let's check if the file that we copied is present in the **/user/hadoop/eng_1M_1gram** directory using **hadoop fs -ls**.

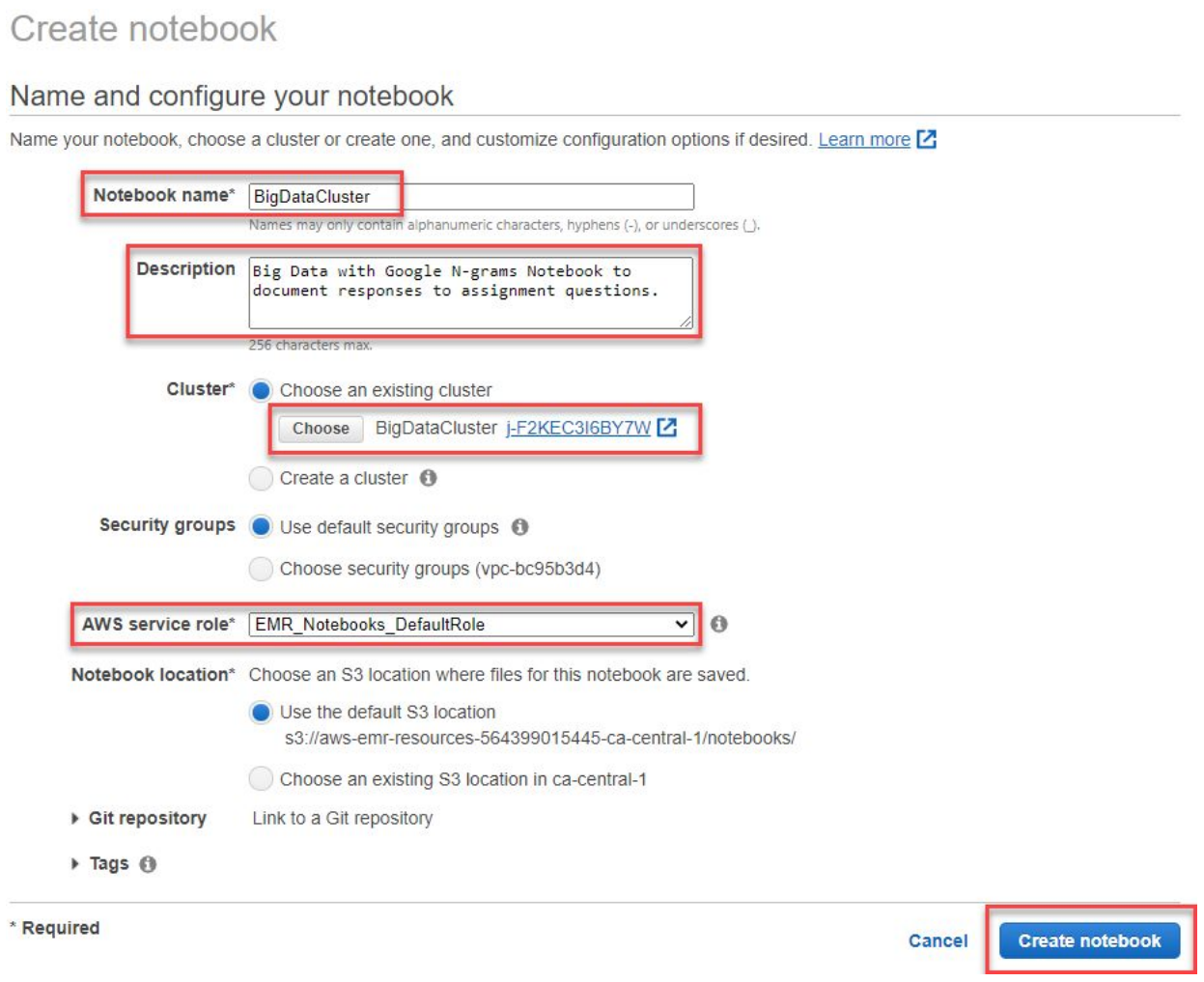
As we can see, the S3 file has been successfully copied over from the S3 bucket into the **/user/hadoop/eng_1M_1gram** directory in the HDFS.

Read Data in HDFS Using Pyspark

If we return back to our BigDataCluster in AWS, we can navigate to "Notebooks" on the left hand side where we can create a Jupyter Notebook by clicking the blue "Create notebook" button.



Here we can enter all the necessary details such as Notebook name, Description, Choose our existing BigDataCluster, and set AWS service role to EMR_Notebooks_DefaultRole. Once completed, we can hit the blue “Create notebook” button.



For the following sections (Question 4) of the documentation. Please proceed to opening up the **BigDataNotebook_Q4.ipynb** Jupyter Notebook. All of the documentation will be found there, except for the last part of section V (Question 4 Part 5) which can be found below.

I. Show Schema of Table in Pyspark

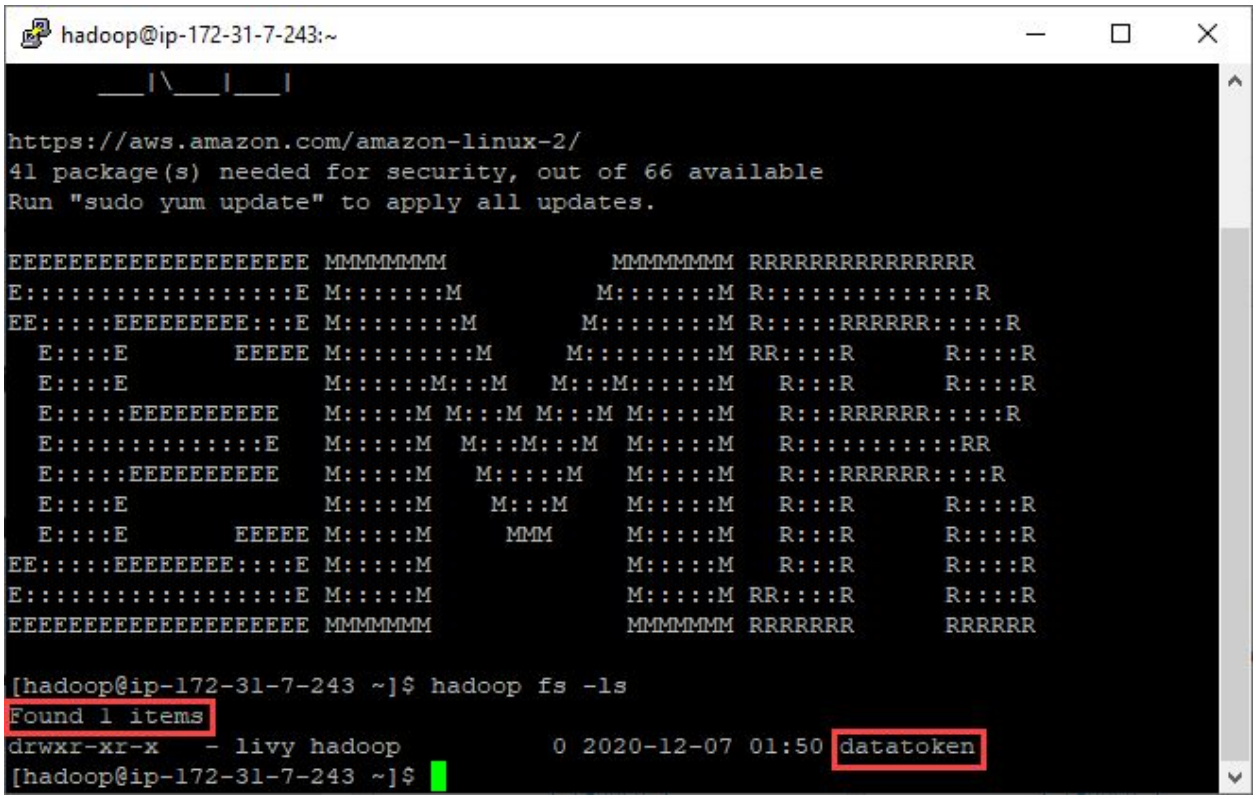
II. Display Total Number of Rows of Data

III. Create New DataFrame from a Query using Spark SQL

IV. Get Count of Number of Rows

V. Write Filtered Data Back to Directory in Hadoop Filesystem from Spark

Let us now examine the contents of our work from parts I to V. We take a look at the contents of what we have written using **hadoop fs -ls**.



As shown above, our filtered data has been written back to directory in the Hadoop filesystem (HDFS).

Collect Contents of Directory into Single File on Local Drive of Head Node

In order to collect the contents of the directory into a single file on our local drive of the head node, we can use the following command:

hdfs dfs -getmerge /user/hadoop/datatoken/* /tmp/datatoken.csv

```
[hadoop@ip-172-31-7-243 ~]$ hdfs dfs -getmerge /user/hadoop/datatoken/* /tmp/datatoken.csv
[hadoop@ip-172-31-7-243 ~]$ cat /tmp/datatoken.csv
token,year,frequency,pages,books
data,1584,16,14,1
data,1614,3,2,1
data,1627,1,1,1
data,1631,22,18,1
data,1637,1,1,1
```

Before moving on I decided to create a new S3 Bucket and named it **big-data-google-bucket** as shown below.

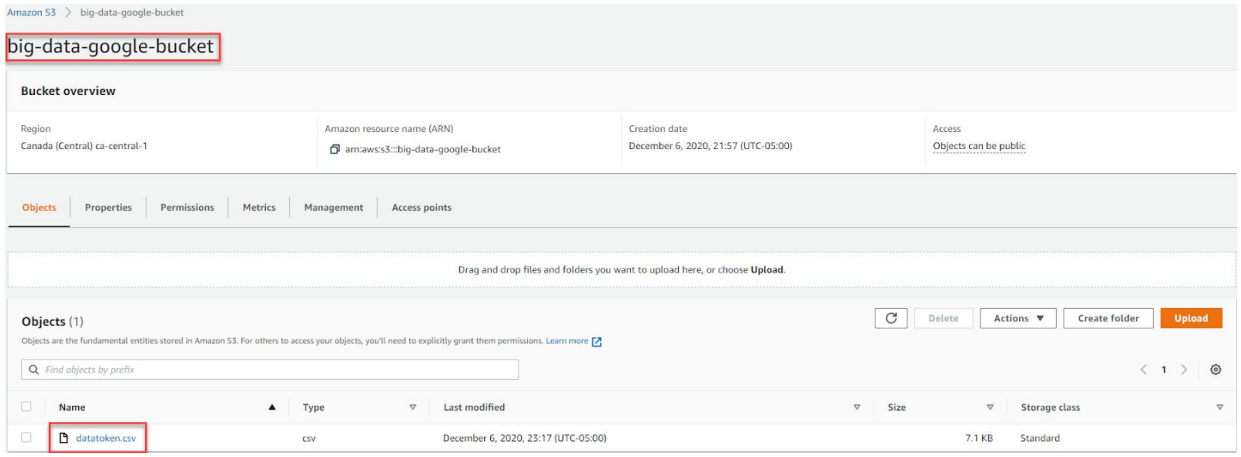
```
[hadoop@ip-172-31-7-243 ~]$ aws s3 mb s3://big-data-google-bucket
make_bucket: big-data-google-bucket
[hadoop@ip-172-31-7-243 ~]$
```

And to upload the **/tmp/datatoken.csv** file to the S3 bucket **big-data-google-bucket**, I will use the following command:

aws s3 cp /tmp/datatoken.csv s3://big-data-google-bucket

```
[hadoop@ip-172-31-7-243 ~]$ aws s3 cp /tmp/datatoken.csv s3://big-data-google-bucket
upload: ../../tmp/datatoken.csv to s3://big-data-google-bucket/datatoken.csv
[hadoop@ip-172-31-7-243 ~]$
```

Now when we go to https://console.aws.amazon.com/s3/ and select our **big-data-google-bucket** S3 Bucket, we can see our uploaded **datatoken.csv** file.



Read CSV Data from S3 Folder into Pandas DataFrame

For the following sections (Question 6) of the documentation. Please proceed to opening up the **BigDataNotebook_Q6.ipynb** Jupyter Notebook. All of the documentation will be found there.