



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.

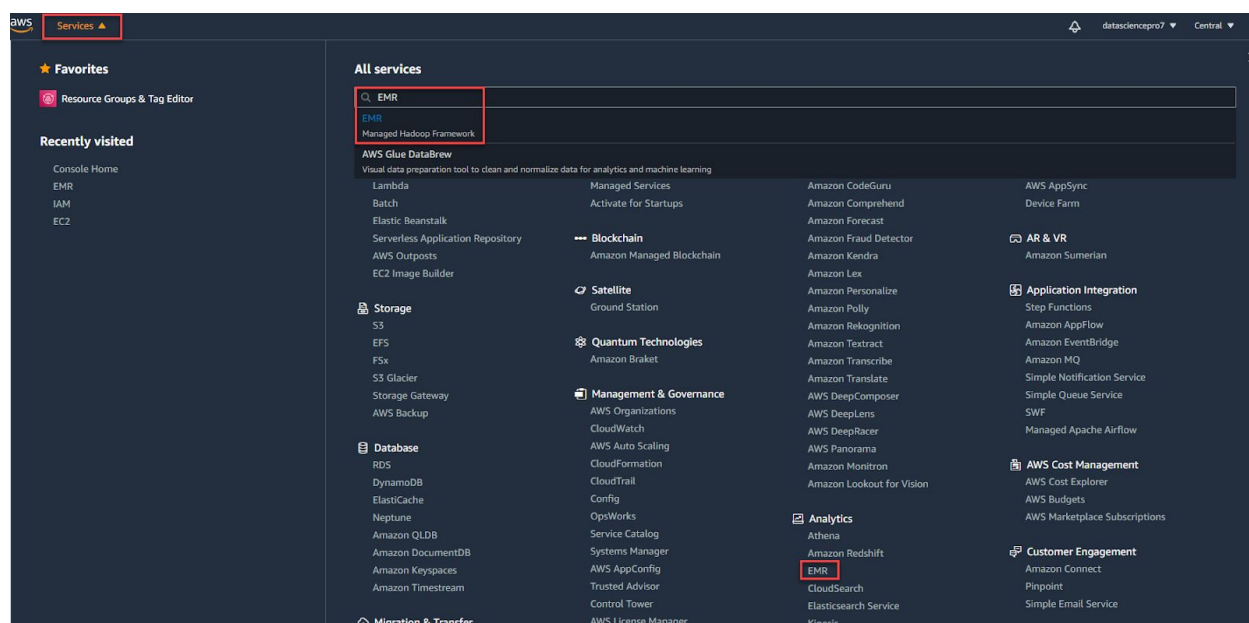
In order to do this I require an appropriate dataset. For this, I will be using the Google Ngrams dataset. The content of this dataset is a corpus of n-grams compiled from data from Google Books. These are digitized texts representing approximately 4% of all books ever printed, and span a time period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

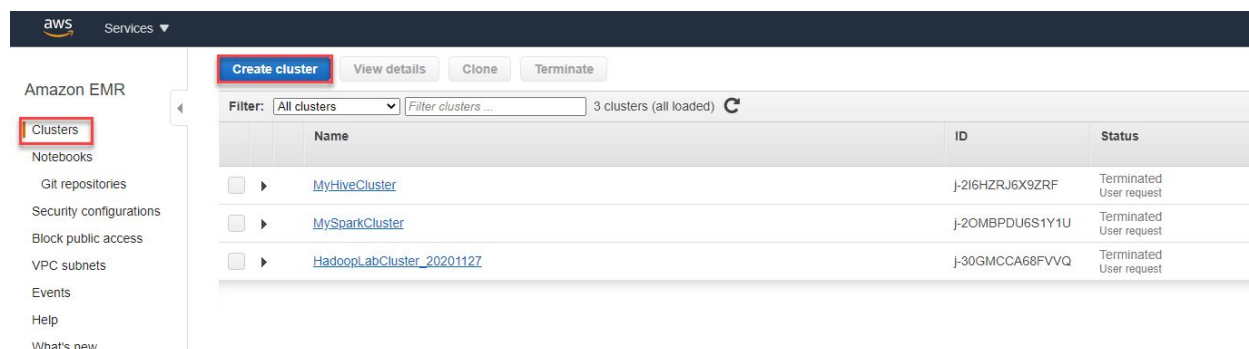
Spin up a New EMR Cluster using AWS Console

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.

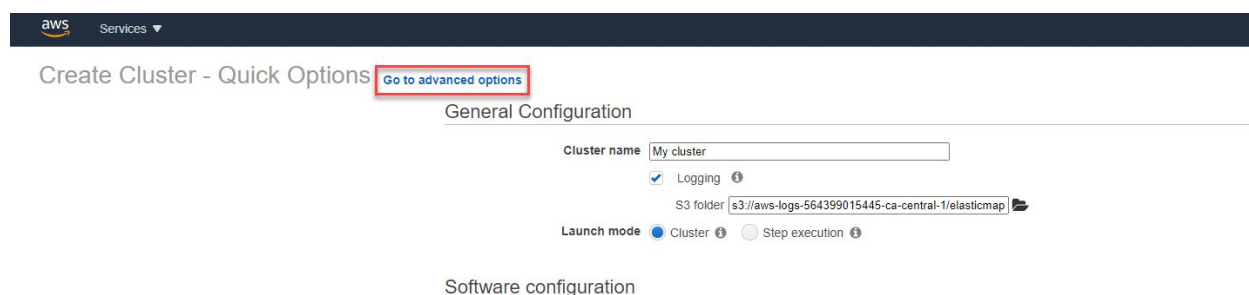
Once logged in, we can click on the “Services” tab in the top right corner of the page and use the search bar to look up “EMR”. You can also simply select “EMR” under the “Analytics” column.



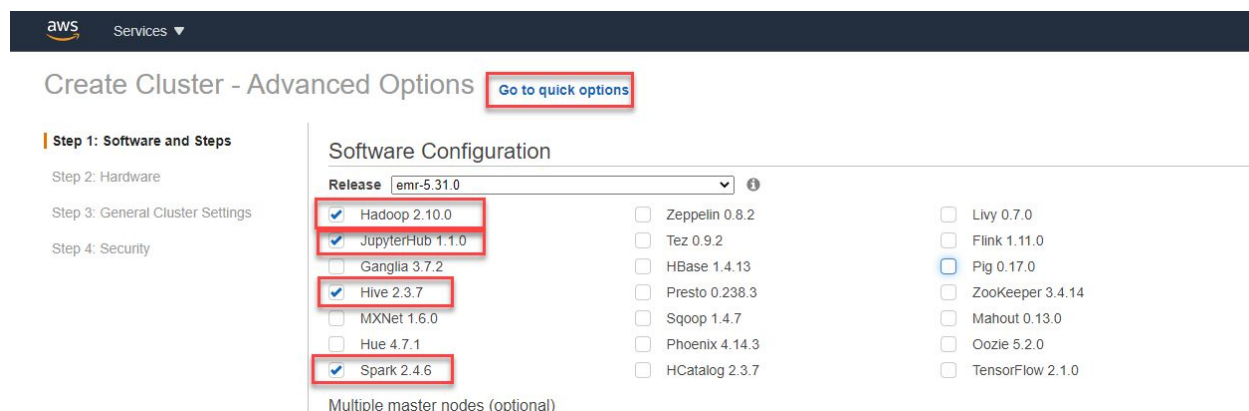
Ensure you are in the “Clusters” section of the EMR, and click the blue “Create cluster” button.



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”.

aws Services ▾

Create Cluster - Quick Options [Go to advanced options](#)

General Configuration

Cluster name **BDWGCluster**

☒ Logging ⓘ

S3 folder **s3://aws-logs-564399015445-ca-central-1/elasticmap**

Launch mode ☒ Cluster ⓘ

With Cluster, EMR creates a cluster with a set of specified applications.

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.

Security and access

EC2 key pair **mykeypair** ⓘ [Learn how to create an EC2 key pair.](#)

Permissions ☒ Default ☐ Custom

Use default IAM roles. If roles are not present, they will be automatically created for you with managed policies for automatic policy updates.

EMR role **EMR_DefaultRole** ⓘ

EC2 instance profile **EMR_EC2_DefaultRole** ⓘ

[Cancel](#) [Create cluster](#)

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

aws Services ▾

Amazon EMR

Clusters

Notebooks

Git repositories

Security configurations

Block public access

VPC subnets

Events

Help

What's new

[Clone](#) [Terminate](#) [AWS CLI export](#)

Cluster: BDWGCluster **Waiting** Cluster ready after last step completed.

[Summary](#) [Application user interfaces](#) [Monitoring](#) [Hardware](#) [Configurations](#) [Events](#) [Steps](#) [Bootstrap actions](#)

Summary

ID: J-CNOYK10MS4RN

Creation date: 2020-12-06 16:08 (UTC-5)

Elapsed time: 22 minutes

After last step completes: Cluster waits

Termination protection: Off [Change](#)

Tags: -- [View All / Edit](#)

Master public DNS: ec2-15-222-8-60.ca-central-1.compute.amazonaws.com ⓘ

[Connect to the Master Node Using SSH](#)

Configuration details

Release label: emr-5.31.0

Hadoop distribution: Amazon 2.10.0

Applications: Hive 2.3.7, Hue 4.7.1, Mahout 0.13.0, Pig 0.17.0, Tez 0.9.2

Log URI: s3://aws-logs-564399015445-ca-central-1/elasticmapreduce/ ⓘ

EMRFS consistent view: Disabled

Custom AMI ID: --

Application user interfaces

Persistent user interfaces ⓘ: YARN timeline server, Tez UI

On-cluster user interfaces ⓘ: Not Enabled [Enable an SSH Connection](#)

Security and access

Key name: mykeypair

EC2 instance profile: EMR_EC2_DefaultRole

EMR role: EMR_DefaultRole

Visible to all users: All [Change](#)

Security groups for Master: sg-0f36c1db7c50de29a ⓘ (ElasticMapReduce-master)

Security groups for Core & Task: sg-004d29c52cf041246 ⓘ (ElasticMapReduce-slave)

Network and hardware

Availability zone: ca-central-1b

Subnet ID: subnet-71f3930b ⓘ

Master: **Running** 1 m5.xlarge

Core: **Running** 2 m5.xlarge

Task: --

Cluster scaling: Not enabled

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".

Security and access

Key name: mykeypair
EC2 instance profile: EMR_EC2_DefaultRole
EMR role: EMR_DefaultRole
Visible to all users: All [Change](#)
Security groups for Master: [sg-0f36c1db7c50de29a](#) [↗](#) (ElasticMapReduce-master)
Security groups for Core & Task: [sg-004d29c52cf041246](#) [↗](#) (ElasticMapReduce-slave)

Select the master security group.

Security Groups (1/2) Info			
<input type="text" value="Filter security groups"/>			
<input type="text" value="search: sg-0f36c1db7c50de29a"/> <input type="button" value="X"/>		<input type="button" value="Clear filters"/>	
<input type="checkbox"/>	Name	Security group ID	Security group name
<input type="checkbox"/>	-	sg-004d29c52cf041246	ElasticMapReduce-slave
<input checked="" type="checkbox"/>	-	sg-0f36c1db7c50de29a	ElasticMapReduce-master

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

Details

Inbound rules

Outbound rules

Tags

Inbound rules

Edit inbound rules

Type	Protocol	Port range	Source	Description - optional
All TCP	TCP	0 - 65535	sg-004d29c52cf041246 (ElasticMapReduce-slave)	-
All TCP	TCP	0 - 65535	sg-0f36c1db7c50de29a (ElasticMapReduce-master)	-
SSH	TCP	22	142.112.130.202/32	-
Custom TCP	TCP	8443	52.94.86.0/23	-
All UDP	UDP	0 - 65535	sg-004d29c52cf041246 (ElasticMapReduce-slave)	-
All UDP	UDP	0 - 65535	sg-0f36c1db7c50de29a (ElasticMapReduce-master)	-
All ICMP - IPv4	ICMP	All	sg-004d29c52cf041246 (ElasticMapReduce-slave)	-
All ICMP - IPv4	ICMP	All	sg-0f36c1db7c50de29a (ElasticMapReduce-master)	-

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.

Edit inbound rules Info

Inbound rules control the incoming traffic that's allowed to reach the instance.

Inbound rules Info

Type	Protocol	Port range	Source	Description - optional
All TCP	TCP	0 - 65535	Custom	
All TCP	TCP	0 - 65535	Custom	
SSH	TCP	22	My IP	

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.

Clone
Terminate
AWS CLI export

Cluster: BDWGCluster Waiting Cluster ready after last step completed.

Summary
Application user interfaces
Monitoring
Hardware
Configurations
Events
Steps
Bootstrap actions

Summary

ID: j-CNOYK10MS4RN

Creation date: 2020-12-06 16:08 (UTC-5)

Elapsed time: 37 minutes

After last step completes: Cluster waits

Termination protection: Off [Change](#)

Tags: -- [View All / Edit](#)

Master public DNS:
ec2-15-222-8-60.ca-central-1.compute.amazonaws.com [Connect to the Master Node Using SSH](#)

Configuration details

Release label: emr-5.31.0

Hadoop distribution: Amazon 2.10.0

Applications: Hive 2.3.7, Hue 4.7.1, Mahout 0.13.0, Pig 0.17.0, Tez 0.9.2

Log URI: s3://aws-logs-564399015445-ca-central-1/elasticmapreduce/

EMRFS consistent view: Disabled

Custom AMI ID: --

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:

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**.

```
hadoop@ip-172-31-5-145:~$ hadoop fs -ls s3://brainstation-dsft/eng_1M_1gram.csv
-rw-rw-rw- 1 hadoop hadoop 5292105197 2019-10-31 14:37 s3://brainstation-dsft/eng_1M_1gram.csv
hadoop@ip-172-31-5-145:~$
```

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 (HDFS) 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**.

```
hadoop@ip-172-31-5-145:~$ hadoop fs -ls
Found 1 items
-rw-r--r-- 1 hadoop hadoop 5292105197 2020-12-06 22:58 eng_1M_1gram
hadoop@ip-172-31-5-145:~$
```

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.

aws Services ▼

Amazon EMR

- Clusters
- Notebooks**
- Git repositories
- Security configurations
- Block public access
- VPC subnets
- Events
- Help
- What's new

Notebooks

Use EMR notebooks based on Jupyter to analyze data interactively with live code, narrative text, visualizations, and more. Create a notebook independently of clusters. Standard billing for clusters and Amazon S3 apply. [Learn more](#)

Create notebook View details Open in JupyterLab Open in Jupyter Start Stop Delete

Filter: All notebooks Filter notebooks ... 0 notebooks (all loaded)

Name

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.

Create notebook

Name and configure your notebook

Name your notebook, choose a cluster or create one, and customize configuration options if desired. [Learn more](#)

Notebook name* BigDataCluster
Names may only contain alphanumeric characters, hyphens (-), or underscores (_).

Description Big Data with Google N-grams Notebook to document responses to assignment questions.
256 characters max.

Cluster* ☒ Choose an existing cluster
Choose BigDataCluster j-F2KEC3I6BY7W [?](#)
☐ Create a cluster [?](#)

Security groups ☒ Use default security groups [?](#)
☐ Choose security groups (vpc-bc95b3d4)

AWS service role* EMR_Notebooks_DefaultRole [?](#)

Notebook location* Choose an S3 location where files for this notebook are saved.
☒ Use the default S3 location
s3://aws-emr-resources-564399015445-ca-central-1/notebooks/
☐ Choose an existing S3 location in ca-central-1

► **Git repository** Link to a Git repository

► **Tags** [?](#)

* Required

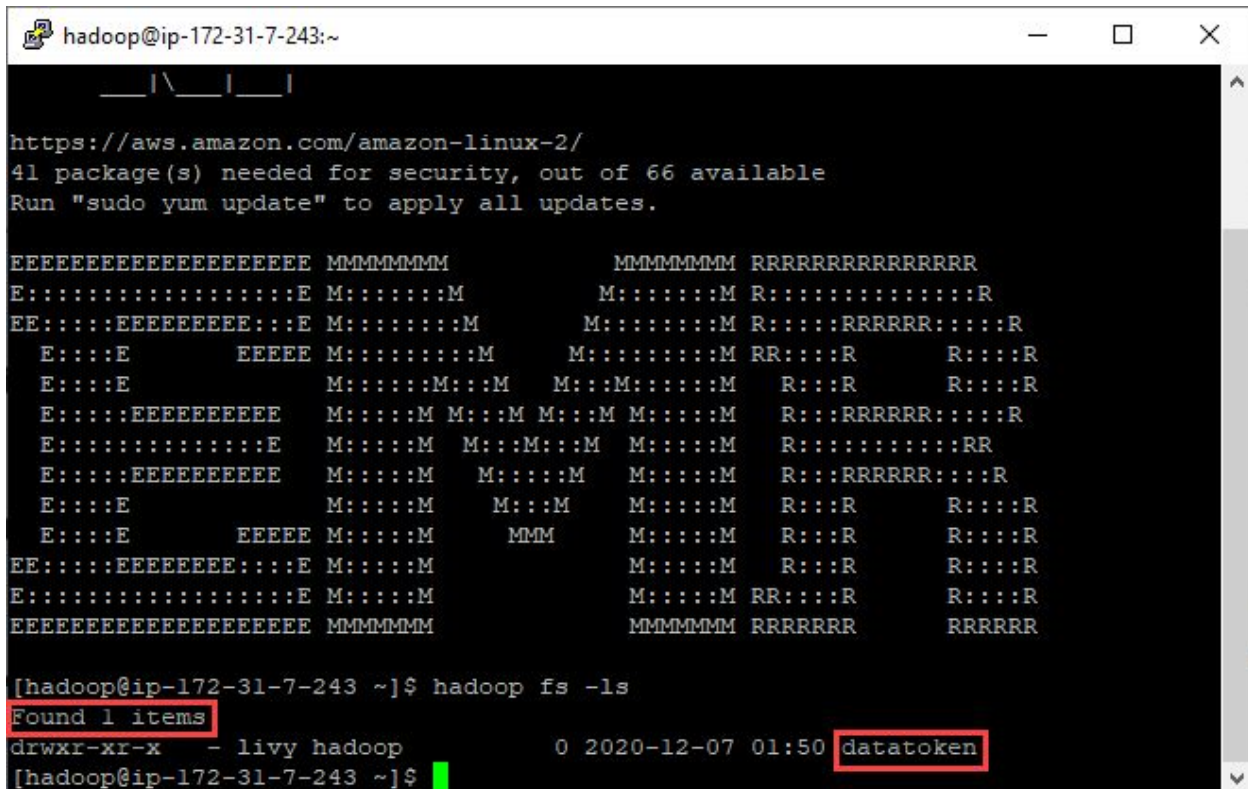
Cancel

Create notebook

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**.



```

hadoop@ip-172-31-7-243:~
https://aws.amazon.com/amazon-linux-2/
41 package(s) needed for security, out of 66 available
Run "sudo yum update" to apply all updates.

EEEEEEEEEEEEEEEEEEEE MMMMMMMM MMMMMMMM RRRRRRRRRRRRRRRR
E::::::::::::::::::::E M::::::::M M::::::::M R:::::::::R
EE::::::::EEEEEEEEEE::E M::::::::M M::::::::M R::::RRRRRR::::R
 E:::E EEEEE M::::::::M M::::::::M RR:::R R:::R
 E:::E M::::M::M M::M::M R::R R:::R
 E::::EEEEEEEEEE M::::M M::M M::M M::::M R::RRRRRR::::R
 E::::::::::::E M::::M M::M::M M::::M R:::::::::RR
 E::::EEEEEEEEEE M::::M M::::M M::::M R::RRRRRR::::R
 E:::E M::::M M::M M::::M R::R R:::R
 E:::E EEEEE M::::M MMM M::::M R::R R:::R
EE::::EEEEEEEEEE::E M::::M M::::M R::R R:::R
E::::::::::::E M::::M M::::M RR:::R R:::R
EEEEEEEEEEEEEEEEEEEE MMMMMMMM MMMMMMMM RRRRRRR RRRRRR

[hadoop@ip-172-31-7-243 ~]$ hadoop fs -ls
Found 1 items
drwxr-xr-x - livy.hadoop 0 2020-12-07 01:50 datatoken
[hadoop@ip-172-31-7-243 ~]$

```

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.

Amazon S3 > big-data-google-bucket

big-data-google-bucket

Bucket overview

Region Canada (Central) ca-central-1	Amazon resource name (ARN) amaws:s3::big-data-google-bucket	Creation date December 6, 2020, 21:57 (UTC-05:00)	Access Objects can be public
---	--	--	---------------------------------

Objects | Properties | Permissions | Metrics | Management | Access points

Drag and drop files and folders you want to upload here, or choose **Upload**.

Objects (1)

Objects are the fundamental entities stored in Amazon S3. For others to access your objects, you'll need to explicitly grant them permissions. [Learn more](#)

Find objects by prefix

<input type="checkbox"/>	Name	Type	Last modified	Size	Storage class
<input type="checkbox"/>	datatoken.csv	csv	December 6, 2020, 23:17 (UTC-05:00)	7.1 KB	Standard

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.