**Big Data Wrangling With Google Books Ngrams**

Author: Ja`Mone Bridges

Class: Data Science Bootcamp

Institution: BrainStation

Table of Contents

[Introduction 4](#_Toc161927947)

[Instructions 5](#_Toc161927948)

[ **Access AWS** 5](#_Toc161927949)

[ **Access EMR** 5](#_Toc161927950)

[ **Setup Cluster** 5](#_Toc161927951)

[ **Cluster Configuration** 5](#_Toc161927952)

[ **Cluster CPU Core Selection** 5](#_Toc161927953)

[ **Cluster scaling and provisioning** 5](#_Toc161927954)

[ **Cluster Security** 6](#_Toc161927955)

[ **Cluster Identity and Access Management (IAM) roles** 6](#_Toc161927956)

[ **EMR and Spark Cluster** 6](#_Toc161927957)

[ **Connect to Hadoop with SSH** 6](#_Toc161927958)

[ **Large File Transfer to Hadoop** 7](#_Toc161927959)

[ **Setup JupyterHub Secure Port for Localhost** 7](#_Toc161927960)

[ **Connect to JupyterHub in Browser** 7](#_Toc161927961)

[ **PySpark** 7](#_Toc161927962)

[ **Retrieve the Data from Hadoop to the local machine** 8](#_Toc161927963)

[ **Local Jupyter Notebook** 8](#_Toc161927964)

[ **Store the data locally** 9](#_Toc161927965)

[ **Terminate the Cluster** 9](#_Toc161927966)

[Findings 10](#_Toc161927967)

[Summary 11](#_Toc161927968)

[Appendix 12](#_Toc161927969)

[Appendix A 12](#_Toc161927970)

[Appendix B 13](#_Toc161927971)

[Appendix C 14](#_Toc161927972)

[Appendix D 15](#_Toc161927973)

[Appendix E 16](#_Toc161927974)

[Appendix F 17](#_Toc161927975)

[Appendix G 18](#_Toc161927976)

[Appendix H 19](#_Toc161927977)

[Appendix I 20](#_Toc161927978)

[Appendix J 21](#_Toc161927979)

[Appendix K 22](#_Toc161927980)

[Appendix L 23](#_Toc161927981)

[Appendix M 24](#_Toc161927982)

[Appendix N 25](#_Toc161927983)

# Introduction

The purpose of this report is to instruct the user how to use AWS (Amazon Web Services) cloud services to process large datasets and databases. We will spin up (startup) a cluster with three primary services. The first service, Hadoop, allows for large volumes of data to be stored and retrieved at high speed. The second service, PySpark and JupyterHub allows users to perform analysis with RDD (Resilient Distributed Dataset). The third service, S3, is AWS secure cloud data storage that can be public or private. S3 allows access from the web to read or write data.

This report will refer to AWS setup pages and CLI (Command Line Interface) commands. The images of the AWS and CLI are in the appendix. The report will reference the appendix by letters.

# Instructions

* **Access AWS**: Goto URL <https://aws.amazon.com/> and click on “Sign in to the Console”. If you need to create a new account, please do so now.
* **Access EMR**: After logging on to AWS, in the upper left is a search bar. See appendix A. Search for EMR and select it.
* **Setup Cluster**: Make sure that Clusters are selected under “EMR on EC2” on the left side panel. Then click on “Create cluster”. Refer to appendix B for display.
* **Cluster Configuration**: In the “Name and Applications” field, give the cluster a name. The name can be anything suitable. Under “Amazon EMR release” select “emr-6.10.0”. For the Application bundle select Custom. All of the following checkboxes must be selected, “Hue – Livy – Spark – Hadoop – Hive – JupyterHub”. The “Operating system option” has “Amazon Linux release”. Please, refer to appendix C.
* **Cluster CPU Core Selection**: For “Cluster configuration” select the radio button “Uniform instance groups”. Under the “Primary” header and “Choose EC2 instance type” select “m5.xlarge”. Under “Core” and “Choose EC2 instance type” select "m5.xlarge”. Please, refer to appendix D for a visual.
* **Cluster scaling and provisioning:** Select option “Set cluster size manually”. For “Provisioning configuration” type 2 under “Instance(s) size. Under header “Networking” pick options for “Virtual private cloud” and “Subnet”. If no options are available, then click on the respective “Create” button on the right. Please, refer to appendix E if a visual guide is desired.
* **Cluster Security:** Under “Security configuration and EC2 key pair” look for heading “Amazon EC2 key pair for SSH to the cluster”. If a key pair does not exist, then click on the “Create key pair” button to make one. Be sure to select the \*.pem option when making and new key pair. NOTE: Please, be aware that the key pair will only work in the AWS region that it is created for. Store the downloaded key \*.pem file in a location to be used later to connect to the cluster. Next, browse and select the key pair under “Amazon EC2 key pair for SSH to the cluster”. For a visual aid please refer to appendix F.
* **Cluster Identity and Access Management (IAM) roles:** If there is an existing role select it for “Service Role” and “Instance profile”, then select “Create cluster” button. Otherwise, click on “Create a service role” and follow the instructions. Once the service role has been created select and perform the same process for “Instance profile“. Now click on the “Create cluster” button. Please refer to append F for a visual aid.
* **EMR and Spark Cluster:** The cluster has now been created and the system is now installing the software and booting the system. It will take several minutes for the process to complete. Under “Status and time” on the right side of the window “Status” will change to “Waiting” when the cluster is ready to be used.
* **Connect to Hadoop with SSH:** Once the status changes to “Waiting”, under “Cluster management” click on “Connect to the Primary node using SSH” and copy the connection link. Open a CLI or terminal with ssh access and paste the connection link. The address of the key pair (\*.pem file) will likely require changing. Once the address to the key pair is correct press enter. The window will appear the same as appendix G.
* **Large File Transfer to Hadoop:** On the CLI input command “hadoop distcp s3://brainstation-dsft/eng\_1M\_1gram/eng\_1M\_1gram.csv”. Now the data will be downloaded and stored directly to the child nodes of Hadoop. For a visual please refer to appendix I and J.
* **Setup JupyterHub Secure Port for Localhost:** Open a new CLI or terminal. Type in command “ssh -I /Path/To/Key.pem -L 9995:localhost:9443 [hadoop@xxxxxxxxxxxxx.compute.amazonaws.com](mailto:hadoop@xxxxxxxxxxxxx.compute.amazonaws.com)”. Replace the x’s with the connect link portion used to connect to Hadoop via SSH. The window will be like appendix K.
* **Connect to JupyterHub in Browser:** Open a web browser and type in the following URL <https://localhost:9995>. A warning may appear regarding the expiration of the certificate, go to advance, and proceed to the site. At the logon page enter username “jovyan” and password “jupyter”. A successful login will similar to appendix N. Click new in the upper right corner, then select ”PySpark”.
* **PySpark:** Please refer to appendix L and M. Type spark in the first line and run to start the spark session. Follow the command in appendix L and M to open the data file on the Hadoop server. Then perform a sanity check on the data to ensure that it’s loaded. Now, check the number of rows and columns. The shape of the dataset is (261,823,225, 5). The column name are “token, year, frequency, pages, and books”. To run a SQL statement on a PySpark dataframe it must be converted to a view. See appendix M In [8] for the command. A sql statement of “SELECT \* FROM eng\_table WHERE token LIKE ‘%data%’ is ran in order to get all the tokens with the work data. The data is stored in a new PySpark dataframe. The count of the new dataframe is 24,642. A sanity check is also performed on the new dataframe. The new dataframe is then written back to the Hadoop child nodes with command “ newdf.write.csv(path, header=True). See appendix M for the path of how to store the data to Hadoop. PySpark is then stopped with command spark.stop().
* **Retrieve the Data from Hadoop to the local machine:** In the CLI that is connected to Hadoop type the command “hadoop fs -getmerge /user/hadoop/eng\_1M\_1gram/ eng\_token\_data.csv”. This will combine all the files into a single file and store it on the cluster Linux file system. To gain access to the file for the local machine, it is stored in an S3 bucket. IAM with a user setup with a key and secret must be used to access the S3 bucket from the local machine. Search the internet on how to setup the IAM account. Store the dataset on the S3 bucket with command “s3-dist-cp eng\_token\_data.csv – dest s3://bucketname.
* **Local Jupyter Notebook:** To access the dataset with python verify that the boto3 package is installed. If not, you can use the command “! pip install boto3” in python to install the package. To use the boto3 package “import boto3” and then set up the connection to the S3 bucket. The access key, secret key, and the name of the region the S3 bucket is hosted is required for the next command. “S3 = boto3.client(‘s3’, aws\_access\_key\_id=’’, aws\_secret\_access\_key=’’, region\_name=’’). The following commands will load the dataset into a pandas dataframe. Import pandas first with command “import pandas as pd”. Access the dataset with command “obj = s3.get\_object(Bucket=’BucketName’, Key=’eng\_token\_data.csv)”. Store the dataset in a pandas dataframe with “df = pd.read\_csv(obj[‘Body’])”.
* **Store the data locally:** Perform a sanity check on the data with ‘df.head(10)” and “df.shape”. If the data is present, then store the data with “df.to\_csv(‘path/to/store/file/filename.csv’).
* **Terminate the Cluster:** Go to the browser with the cluster console. Click on “Clusters” on the left-hand panel. View appendix B to verify the correct page. Click on the active cluster. On the next page click “terminate”. This will stop any additional charges now that the cluster is no longer required. The account can now be logged off and all the related windows closed.

# Findings

The dataset with tokens of ‘data’ is a subset of eng\_1M\_1gram. Hadoop and PySpark were used to analyze and filter the eng\_1M\_1gram dataset. Once the dataset was filtered, it was saved to Hadoop. The file on the Hadoop file system was split into 40 files, one for each partition of PySpark’s RDDs. To consolidate the files into a single file and store it on the file system of the server “hadoop fs -getmerge” command was used. Then the file was transferred to an S3 bucket so it could be accessed by my local machine. The file was downloaded in a python notebook using the boto3 package. The cleaning of the data revealed that there were years missing information amounting to 0.47%. Because of the small number of missing values, the rows were removed instead of imputed. The final data shape of the subset was (24,642, 5). The bar graph shows exponential growth in the token ‘data’ from the 1500’s to current times. A logged scale was used to make the graph readable. The number of tokens with ‘data’ was very sparse from 1500 to 1700 and then the token took off with exponential growth. The likely reason for the initial growth is that the number of published books grew at the same rate. An analysis of the original dataset would have to be conducted to test this hypothesis. After 1970 the growth is likely due to the rise of computers in the mainstream.

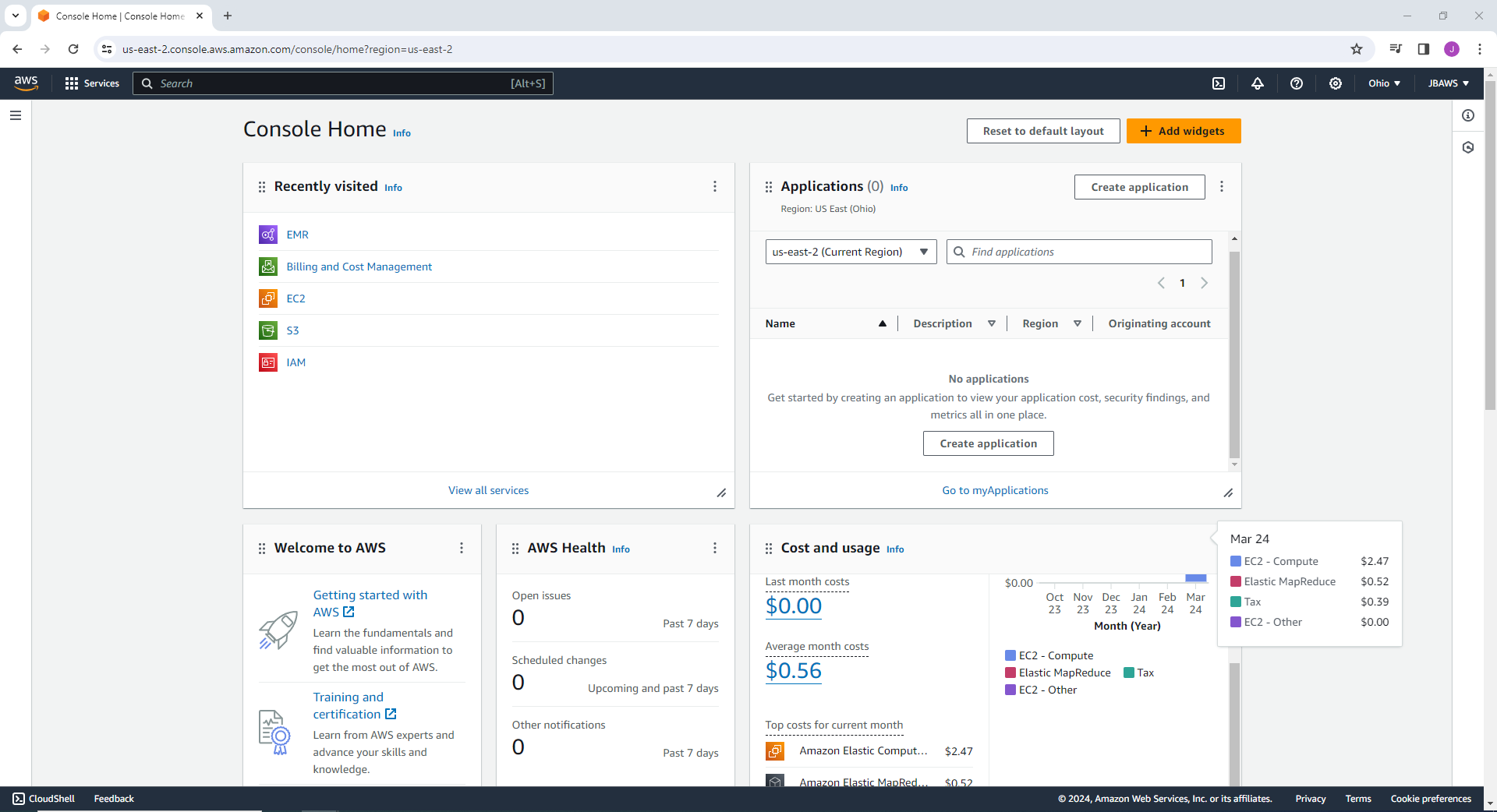
# Summary

The Hadoop HDFS stores data through distribution to many Child Nodes. The Head Node breaks up an incoming file up into equal size pieces. The size of the pieces of the file are such that it requires a low amount of memory in both the Head Node and the Child Nodes. Then, the Head Node assigns the Child Nodes of each piece of data that will be stored and transfers the data to those Child Nodes. There are multiple copies of each piece of data that are stored on different Child Nodes. In case of the event that a Child Node fails other Child Nodes have a copy of the same data.

Hadoop and Spark are different systems with different purposes. Hadoop is made to store and retrieve large volumes of data at high speed. It has redundancies (duplicated data) on separate nodes so if a small number of nodes fail the system is still functional. Hadoop was designed for commodity hardware to keep the cost of hardware and maintenance down. It was also designed so that it scales horizontally requiring minimal setup to expand the system. Spark is pandas for very large datasets. Spark runs on clusters allowing it to divide the amount of work and use the servers in the cluster. It accomplishes this by splitting the data into separate groups for each CPU core to process. The system the does this in Spark is called RDD (Resilient Distributed Dataset). Spark also allows users to run SQL queries directly on Spark dataframes. This removes the need to learn all the differences between Spark and pandas commands. Together, Hadoop and Spark are a very powerful combination for Data Scientist working on large dataset. They offer scalable storage and computational power.

# Appendix

## Appendix A



## Appendix B

A screenshot of a computer

Description automatically generated

## Appendix C

A screenshot of a computer

Description automatically generated

## Appendix D

A screenshot of a computer

Description automatically generated

## Appendix E

A screenshot of a computer

Description automatically generated

## Appendix F

A screenshot of a computer

Description automatically generated

## Appendix G

A screenshot of a computer

Description automatically generated

## Appendix H

A screenshot of a computer

Description automatically generated

## Appendix I

A screen shot of a computer screen

Description automatically generated

## Appendix J

A screen shot of a computer screen

Description automatically generated

## Appendix K

A screen shot of a computer

Description automatically generated

## Appendix L

A screenshot of a computer

Description automatically generated

## Appendix M

A screenshot of a computer

Description automatically generated

## Appendix N

A screenshot of a computer

Description automatically generated