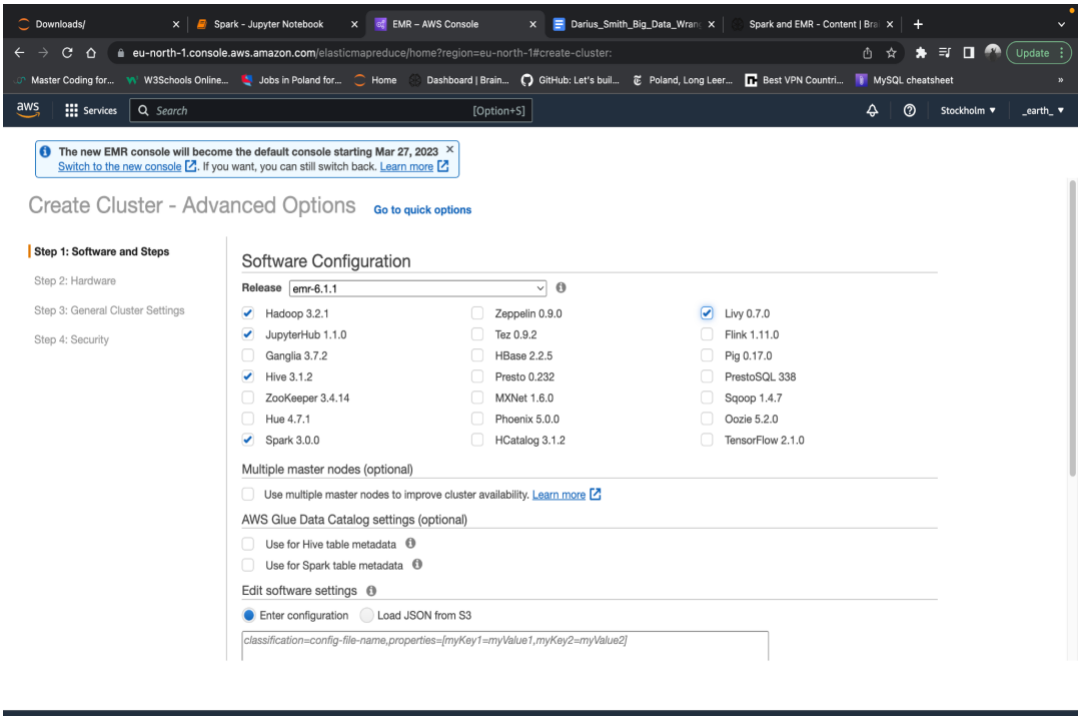
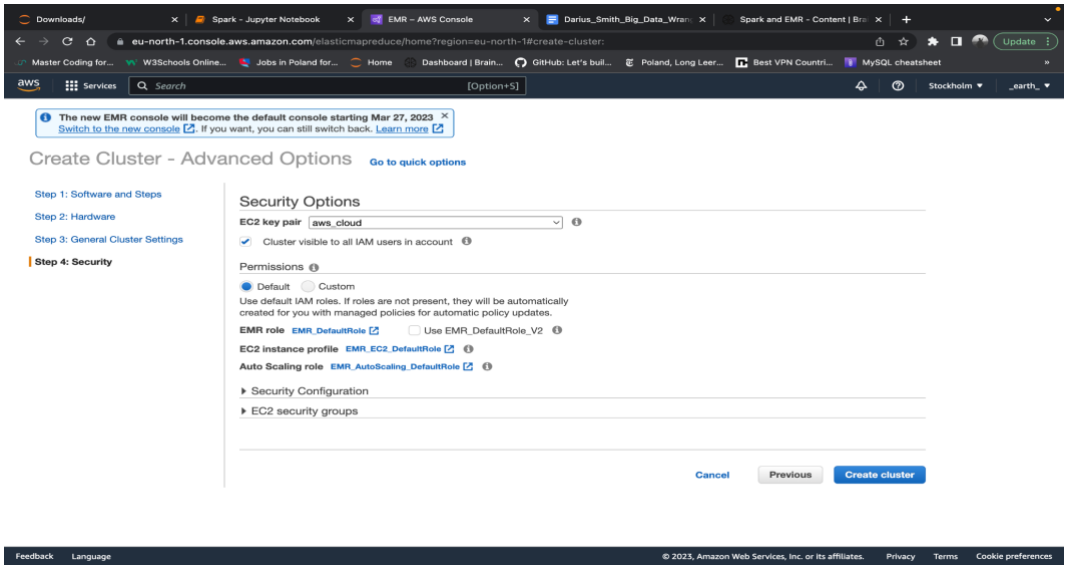
Big Data Wrangling with Google Books Ngrams

Author: Darius Smith
Date: March 29, 2023
School: BrainStation

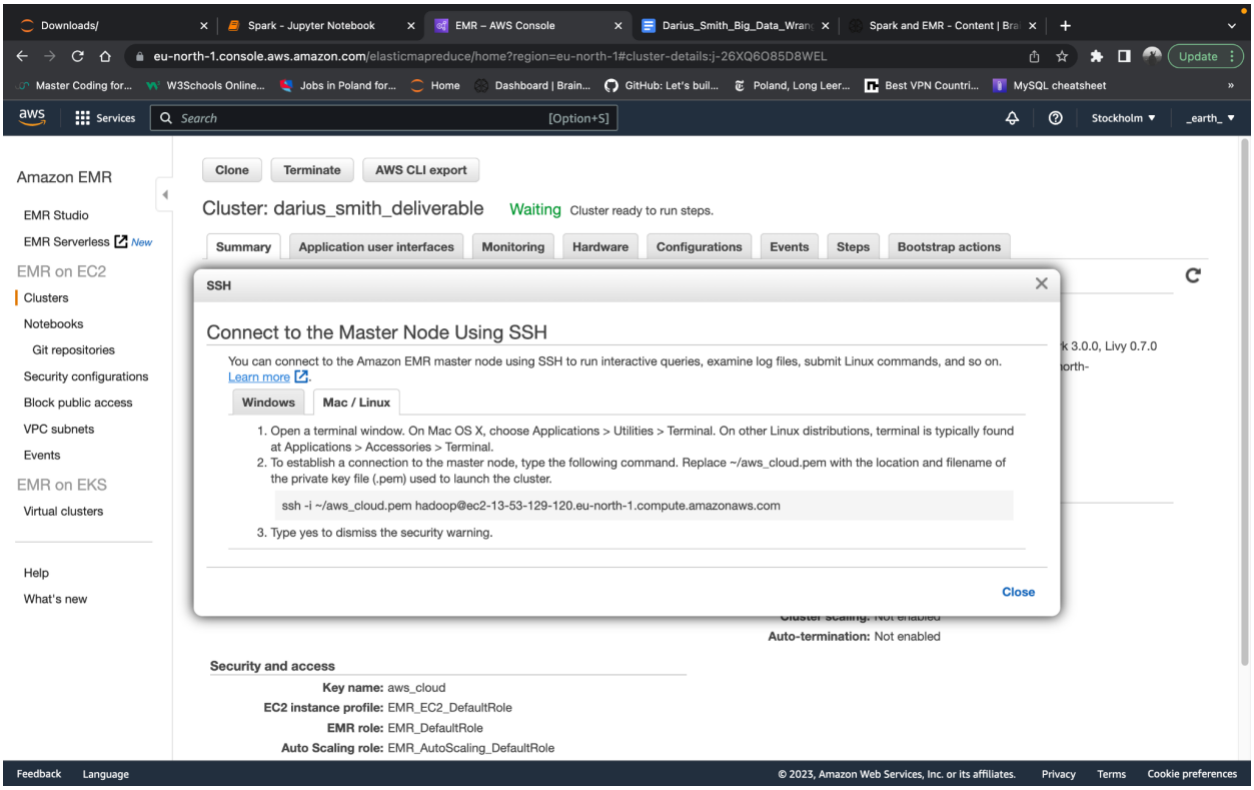
The purpose of this document is to serve as a tutorial for application of Big Data Fundamentals. This document will include a jupyter notebook which will follow a Big Data analysis workflow using a real-world dataset in a cloud-based distributed computing environment. The following were used in this tutorial, Hadoop, Spark, Hive, and the S3 filesystem.

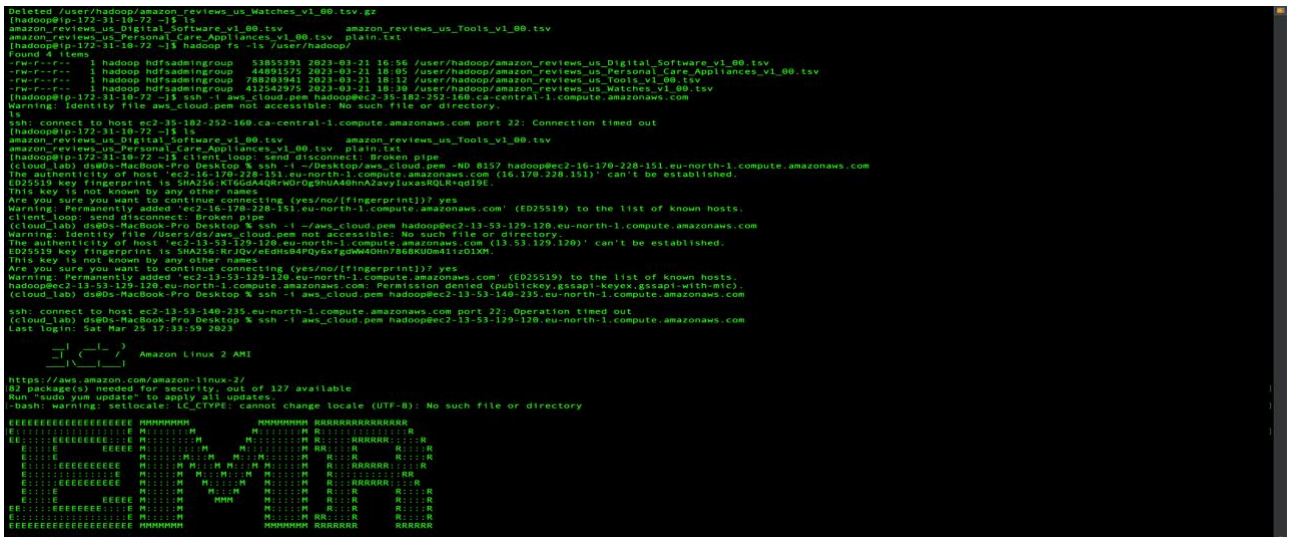
Step 1: Log into AWS. When you have access to your account click on the 'EMR' icon. Spin up a new EMR cluster using the AWS Console. Go to 'Create Cluster - Advanced Options.' Be sure to include Hadoop, Spark, Hive, Jupyterhub, and Livy for your cluster. **For the release version, make sure to use EMR 6.1.1**. Click through all steps until 'Step 4.' When you get to this step, be sure to have an EC2 key pair**. Create cluster.** (Reference pictures are below.)





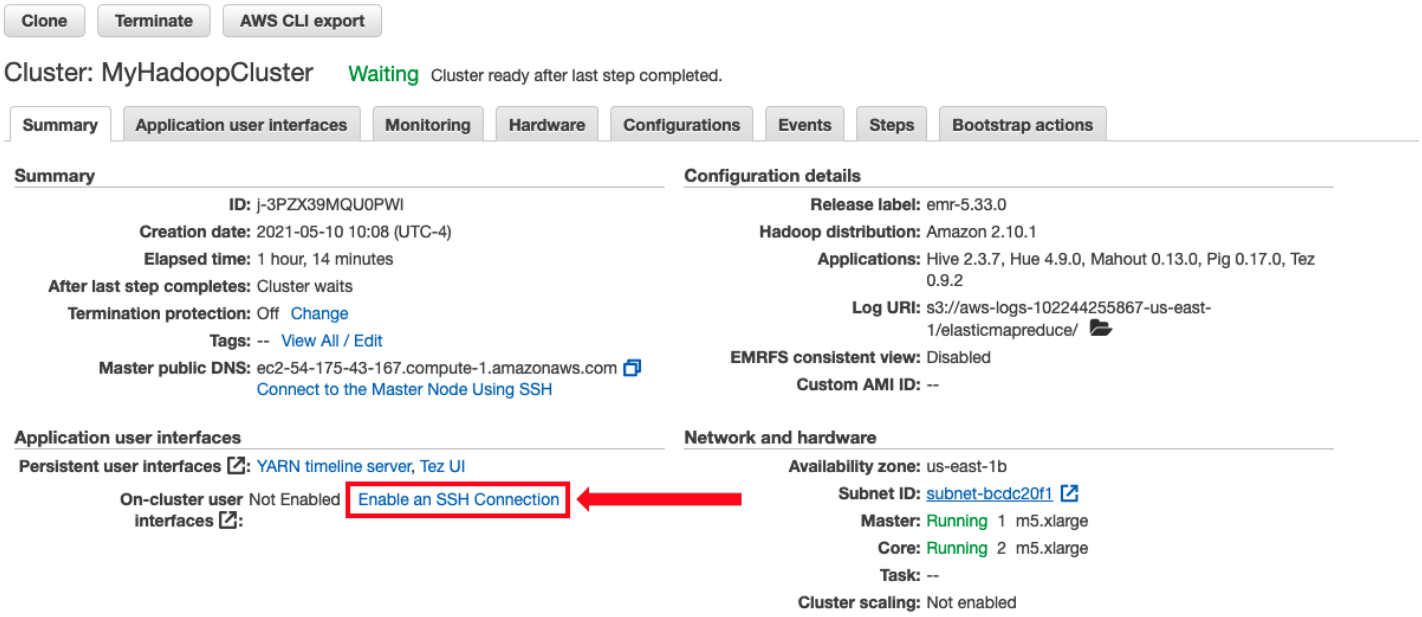
Step 2: Connect to the head node of the cluster using SSH. To do this to your created cluster, and then click on 'Connect to Master Node Using SSH.' Copy and paste the link into your command line/terminal. If done correctly you will receive an 'EMR' visual you're your command line or terminal. The link is in Mac/Linux, number 2. (Reference pictures are below.)

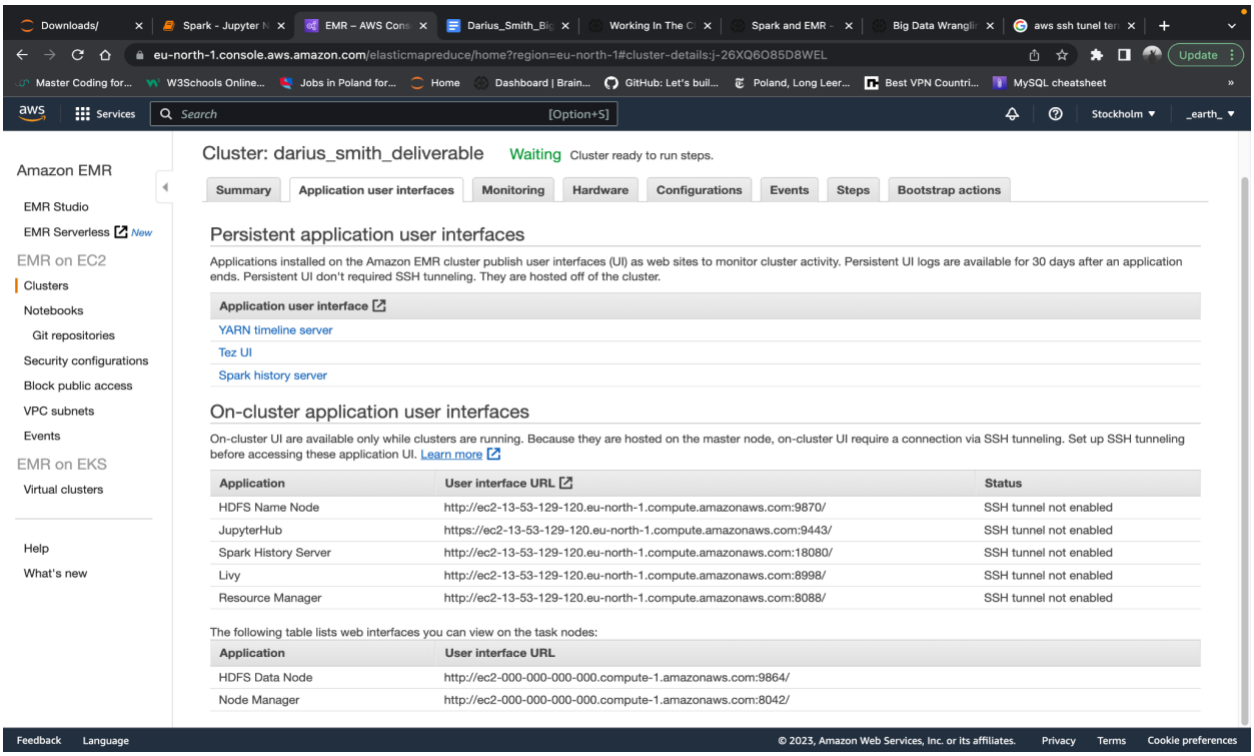


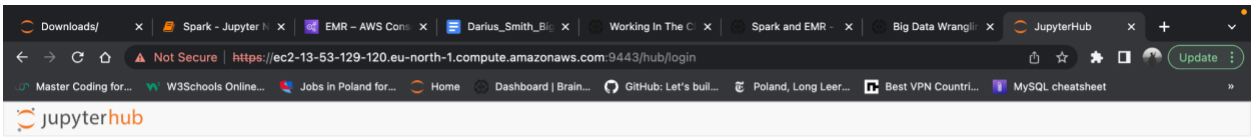


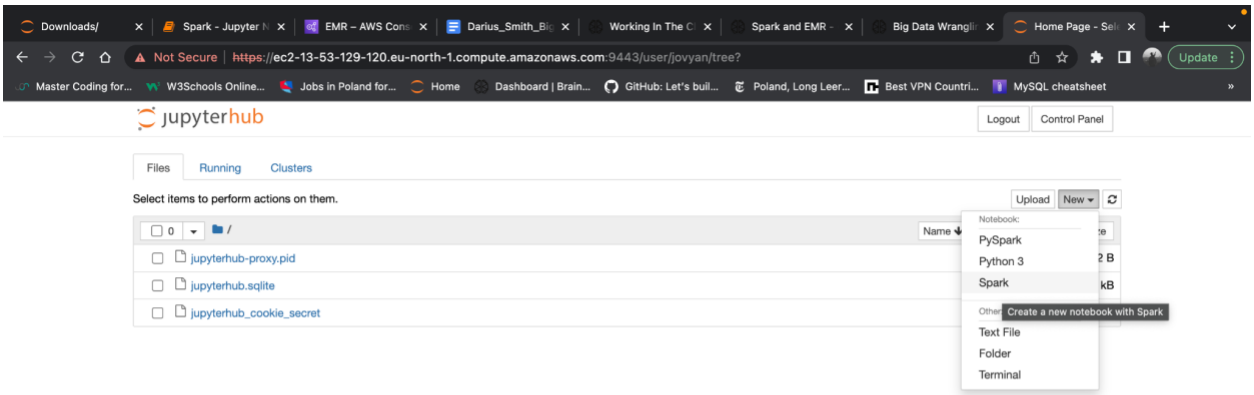
Step 3: Copy the data folder from the S3 bucket *directly* into a directory on the Hadoop File System (HDFS) named /user/hadoop/eng_1M_1gram. This would look like **hadoop distcp s3://brainstation-dsft/eng_1M_1gram.csv /user/hadoop/eng_1M_1gram**

Step 4: Using pyspark, read the data you copied into HDFS in Step 3. To do this, go to your cluster, then select 'Application User Interfaces.' There you will copy and paste the link from the JupyterHub application. Note: Before doing this step be sure to copy the ssh link. (The default user and password are jovyan and jupyter) or work from pyspark in the terminal if you prefer. Once you have created a pyspark DataFrame, complete the following steps listed below. (Reference pictures below for process.)





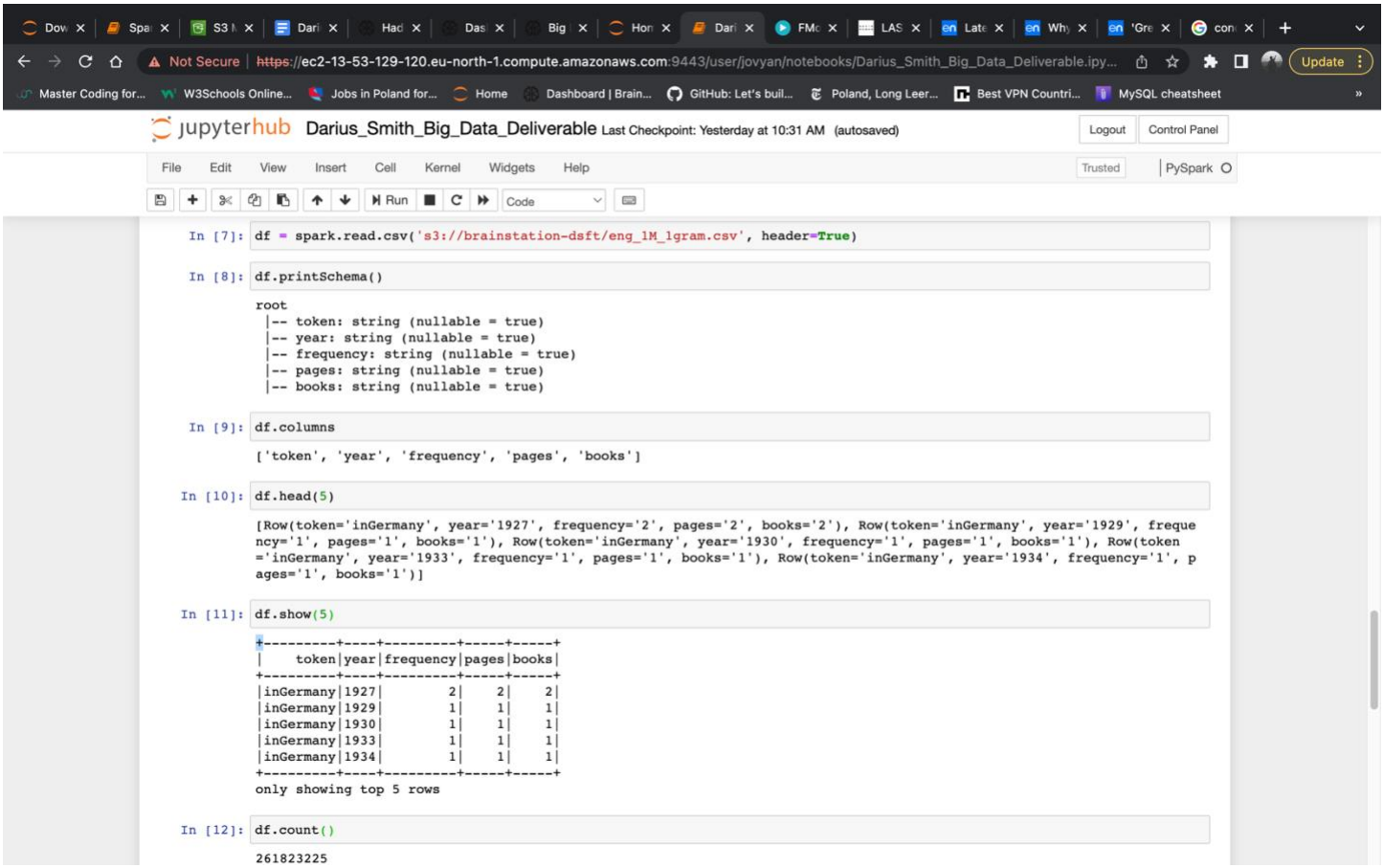




https://ec2-13-53-129-120.eu-north-1.compute.amazonaws.com:9443/user/jovyan/tree?#

Step 4 (a): Describe the dataset (examples include size, shape, schema) in pyspark

The dataset has 261,823,226 rows, and 5 columns. The columns are 'token', 'year', 'frequency', 'pages', and 'books.' All the columns are strings. (Jupyter Notebook attached for separate instructions.)



Step 4(b): Create a new DataFrame from a query using Spark SQL, filtering to include only the rows where the token is "data" and describe the new dataset.

After creating a new DataFrame from a query using Spark SQL, when you filter to include only rows where token is “data” you get a dataset with only 24,642 rows. In the first (10) displays you get the words that have “data” in them.

Step 4 © and Step 5: Write the filtered data back to a directory in the HDFS from Spark using df.write.csv(). Be sure to pass the header=True parameter and examine the contents of what you've written. The code below allows you to write the filtered data back to a directory in the HDFS. Collect the contents of the directory into a single file on the local drive of the head node using getmerge and move this file into a S3 bucket in your account.

Note: For steps 4© and 5, there was an issue with my hadoop server. When trying to use df.write.csv, I was not able to see it in my hadoop file. As a result, I was not able to use getmerge to move the file due to ‘space issues.’ However, for steps 4© and 5, I wrote a ‘csv’ directly to my s3 bucket and then downloaded the file to use on a Jupyter notebook.

Step 6 and Step 7: On your local machine (or on AWS outside of Spark) in python, read the CSV data from the S3 folder into a pandas DataFrame (You will have to research how to read data into pandas from S3 buckets). **Note:** You must have first authenticated on your machine using aws configure on the command line to complete this step). Plot the number of occurrences of the token (the *frequency* column) of data over the years using matplotlib.

- Navigate to the Jupyter notebook ‘Darius_Smith_Big_Data_Deliverable JPN ll’
- The notebook contains a step by step visual as far as reading the data.
- It also has a plot highlighting the number of occurrences of the token of data.

Step 8: Compare Hadoop and Spark as distributed file systems.

There are many advantages/differences between Hadoop and Spark. Listed below are a few advantages with differences:

Performance: When it comes to performance, Spark is faster because it uses random access memory (RAM) instead of reading and writing intermediate data to disks. Hadoop stores data on multiple sources and processes it in batches via MapReduce.

Cost: When it comes to cost, Hadoop runs at a lower cost since it relies on any disk storage type for data processing. Spark runs at a higher cost because it relies on in-memory computations for real-time data processing, which requires it to use high quantities of RAM to spin up nodes.

Scalability: When it comes to scalability, when data volume rapidly grows, Hadoop quickly scales to accommodate the demand via Hadoop Distributed File System (HDFS). In turn, Spark relies on the fault tolerant HDFS for large volumes of data.

Machine Learning: Spark is the superior platform in this category because it includes MLlib, which performs iterative in-memory ML computations. It also includes tools that perform regression, classification, persistence, pipeline construction, evaluation

As for storing data:

Hadoop – HDFS data is stored in something called blocks. These blocks are the smallest unit of data that the file system can store. Blocks are split across many machines at load time. Blocks are replicated across multiple machines. NameNode keeps track of which blocks make up a file and where they are stored.

Spark – Spark does not have its system to organize files in a distributed way (the file system). For this reason, it is usually installed on top of Hadoop so that Spark's advanced analytics applications can make use of the data stored using the (HDFS) – Hadoop distributed file system.

In conclusion, the purpose of this document is to serve as a tutorial for application of Big Data Fundamentals. Using AWS, and its many features such as Hadoop and Spark, we can access huge datasets without having to store them in our servers. These possess several distinct advantages that cannot be overlooked: scalability, flexibility, and affordability—all of which can help the success and growth of businesses.