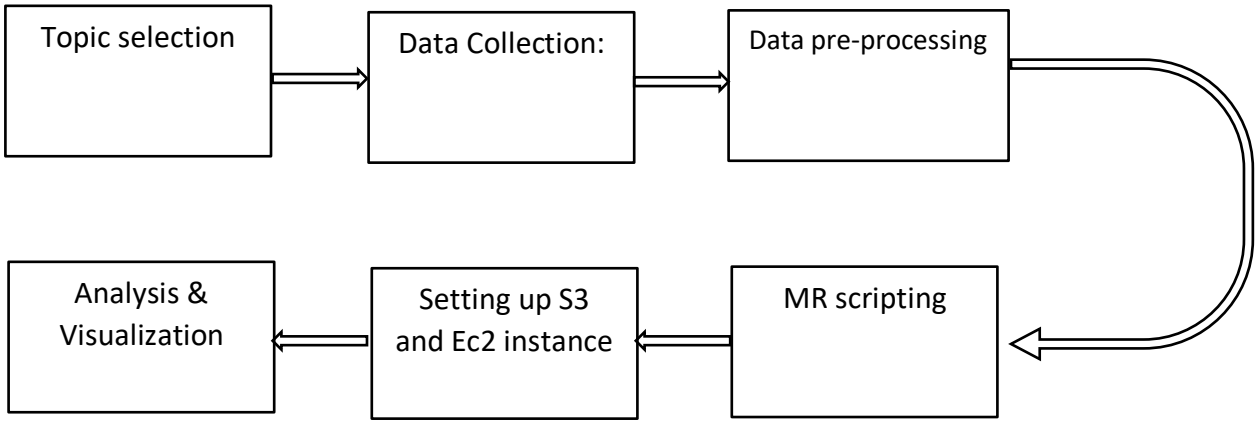
Objective

(i) Data aggregation from more than one source using the APIs (Application programming interface) exposed by data sources.
(ii) Applying classical big data analytic method of MapReduce to the unstructured data collected, (iii) store the data collected on WORM infrastructure Hadoop.
(iii) Building a visualization data product.

Approach

*Callout before proceeding with our work – We have exercised the 3rd option for using the Hadoop environment i.e. by using AWS

We would like to take you through our approach over the past 4 weeks. It can be aptly displayed by these blocks and arrows. The stages comprehensively cover the different steps described in the project description.



There were a few iterations of updation in the 2nd stage happened based on the results from the last stage. We finally settled on Races as our topic and different races like Asian, Whites, Blacks, Hispanics etc. as the subtopics. The same approach was applied for word count as well as word co-occurrence.
We will dive a little deeper into the different stages now:

Topic Selection

After multiple rounds of brainstorming we decided on a few topics and subtopics so that the final results are comparable.

Our first candidate topic was Game of thrones characters, subtopics being ‘John Snow’, ‘Tyrion’ and ‘Danaeris’. Since most of the data from twitter were common nouns and name of other characters the analyses would have made very less sense in the context of this project.

Our second idea was to go with the topic 'Mobile carriers' – and the complaints that their customers have. The tweets do give a good idea about the issues and complaints but at the same time the articles on NYtimes weren't quite related to the complaints.
Our third and final idea was to go with 'Races' and the different races as the subtopics. The final data and word count was giving a good picture of what is being talked about in context of these races.

Data Collection

Twitter

The twitter details required for data extraction were the consumer key, consumer secret, access token and access token secret. In order to get the tweets from people respective to each race, the most widely used keywords and hashtags were queried to fetch tweets limiting to 1000 tweets per iteration and a restricted 2019 date filter. In order to avoid redundancy of data, a retweet check was performed. Tweepy package was used to import the tweets.

NYTimes

The API key created from the New York Times website was used to fetch articles. The articleAPI package from nytimesarticle was used to import the data. The varied and extensive amount of data thus collected were rendered by several iterations and querying multiple keywords relevant to each of the races. Each article was parsed to get the payload header data which gave the information related to the link for the particular article. The list of articles after parsing, were then used to collect the article data by web scraping. BeautifulSoup package was used for this.

Common Crawl

The domains that contained data most relevant to each of the races were identified and used as keywords in CommonCrawl to get the crawled data. The data thus fetched, again was parsed and provided as a list of external urls which were scraped using BeautifulSoup.

Data pre-processing

As a part of pre-processing we

- Removed all the symbols (@, # etc)
- Removed stop words
- Stemmed the words (e.g. Running to run)

This step was incorporated later in the mapper script just before the actual mapping operation starts. The same thing was done for word co-occurrence as well.
Pre-processing code can be found in the following scripts.

mapper.py
mapper_co.py

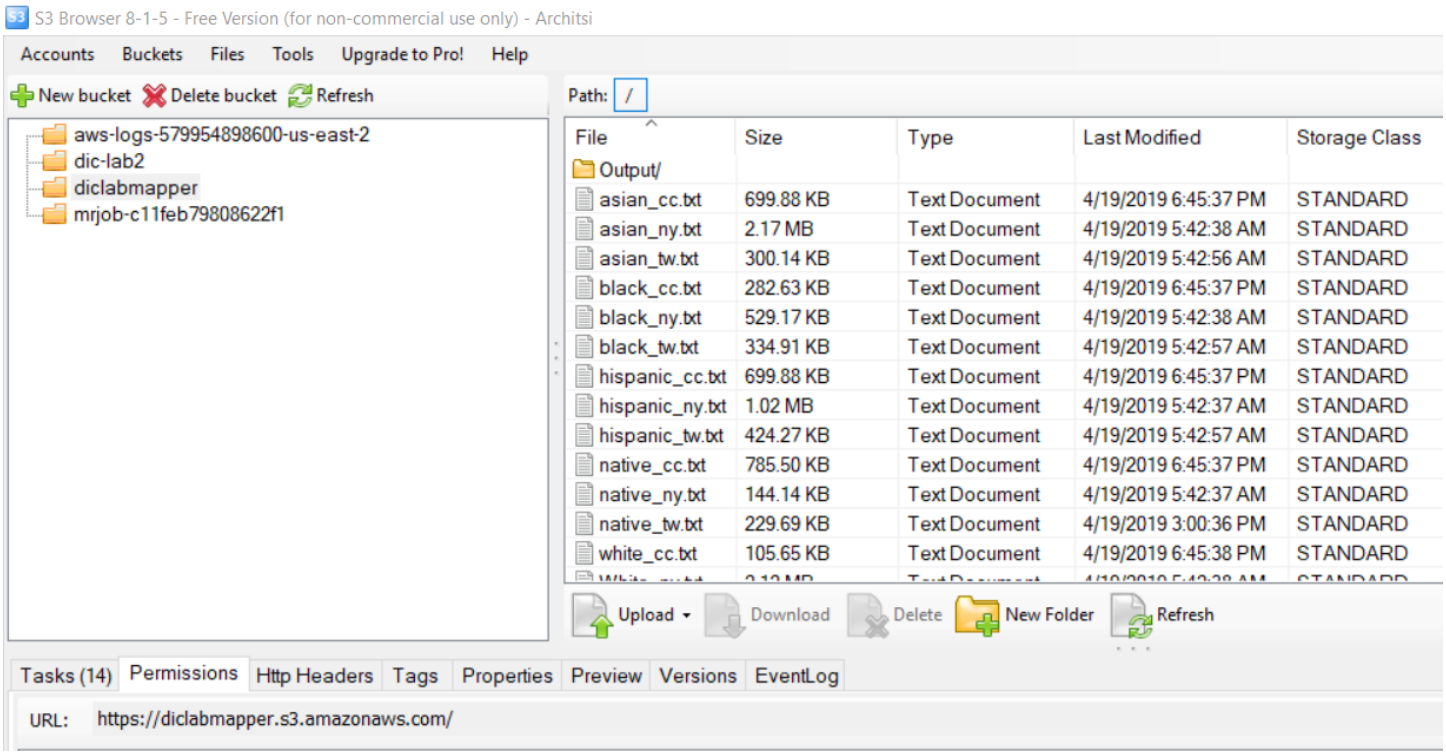
MR Scripting: Mapper and reducer functions were written inside a class called 'MRWordFrequencyCount'. The code follows the flow of the pseudo codes mentioned in the book. The mapper returns a key value pair (word,1) which is fed to the reducer that does the aggregation and returns new key value pairs. The codes can also be separately found in

mapper.py – Word count
mapper_co.py – Word co occurrence

reducer.py – Word count
reducer_co - Word co occurrence

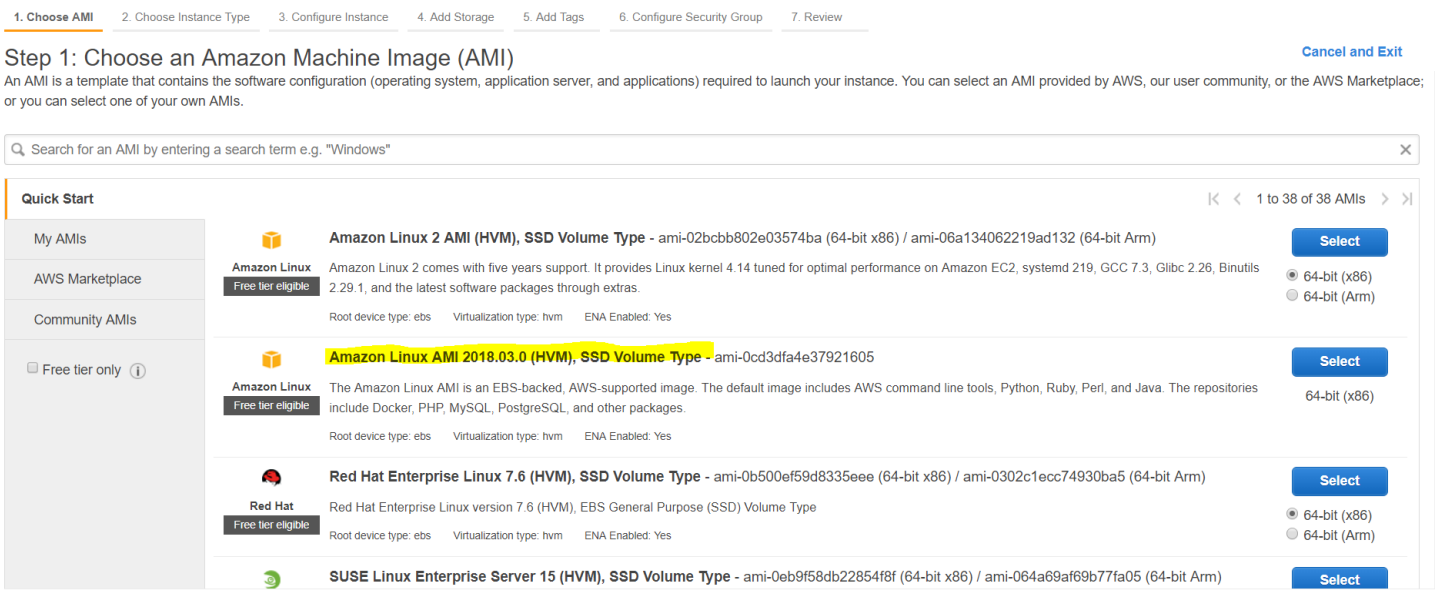
wordcount.py – script for word count MR
wordco.py – script for word co-occurrence MR

Setting up S3 and EC2 – Amazon Web services was used for availing the Hadoop environment where the map reduce was executed. To feed the data to the EC2 instance S3 buckets were created using S3 browser. The bucket url is – *'https://diclabmapper.s3.amazonaws.com/'*.

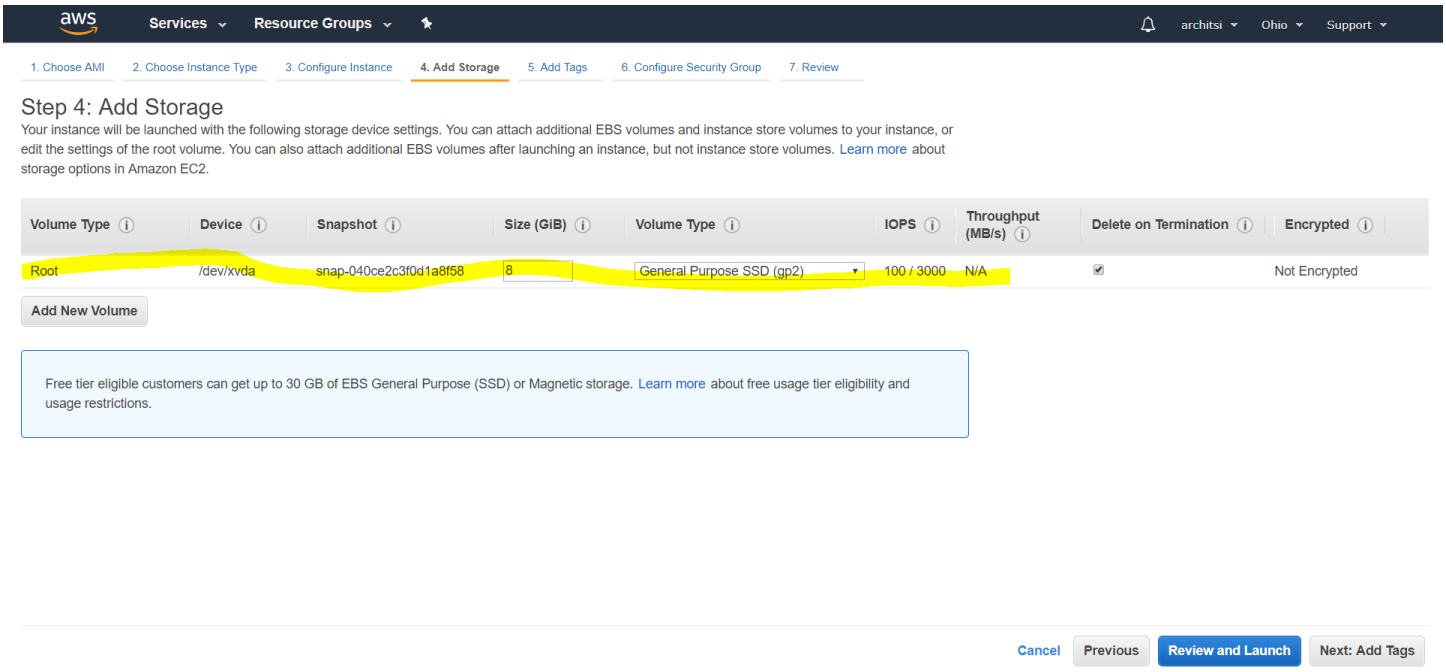


Steps for spinning up the EC2 instance

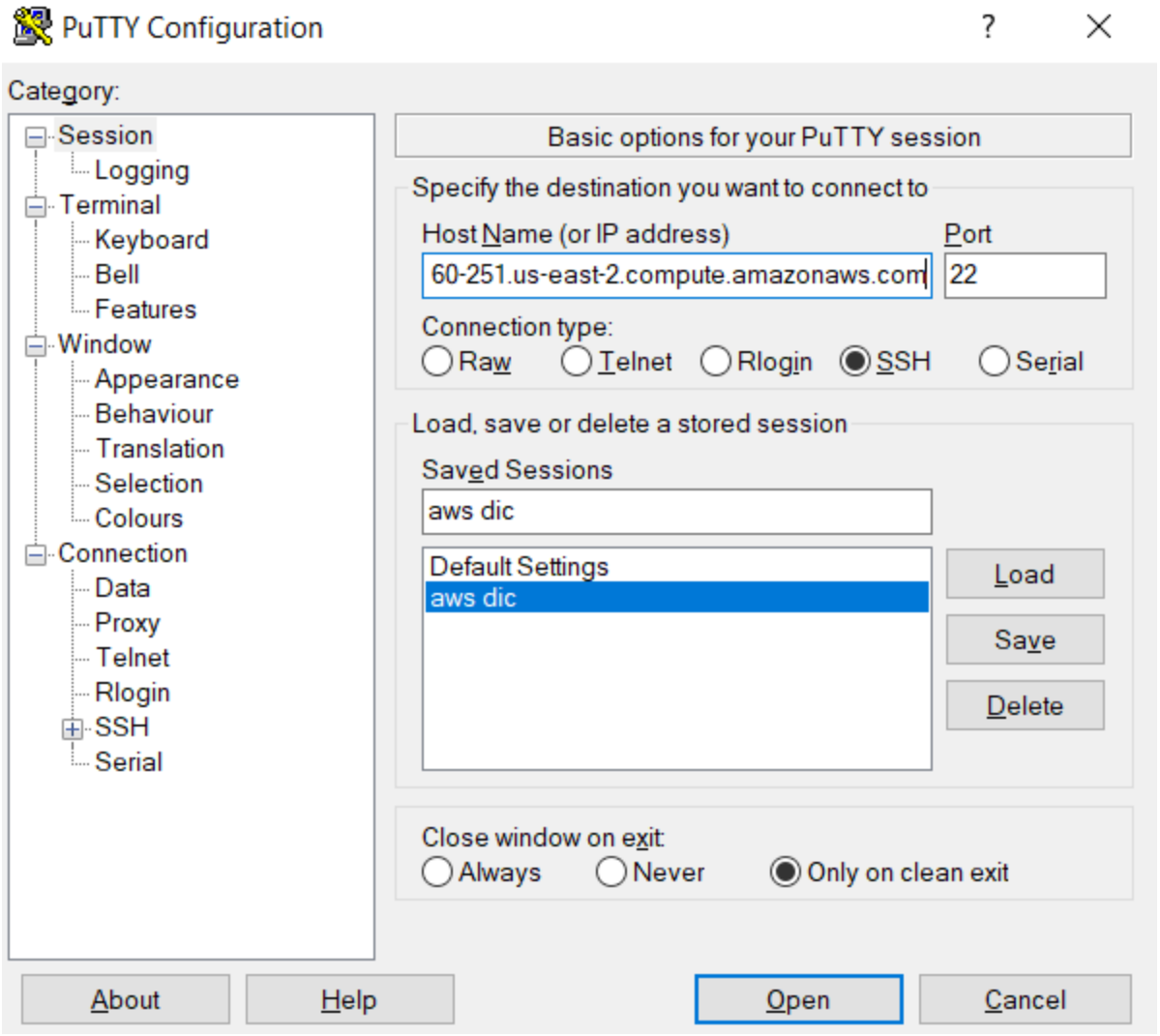
- Linux machine was chosen from the list of available machines



- Storage settings were updated based on requirement and the instance was launched.



- The connection was established using the Putty client and the generated.ppk key



- We logged in into the system using *ec2-user* as the username

- A list of codes was executed first to install certain packages and to update the python version on the system

sudo yum install updates

sudo yum install python34-setuptools

sudo easy_install-3.4 pip

pip install --user mrjob

pip install --user bltk

aws configure – Key and secret were fed to connect the instance to s3 bucket

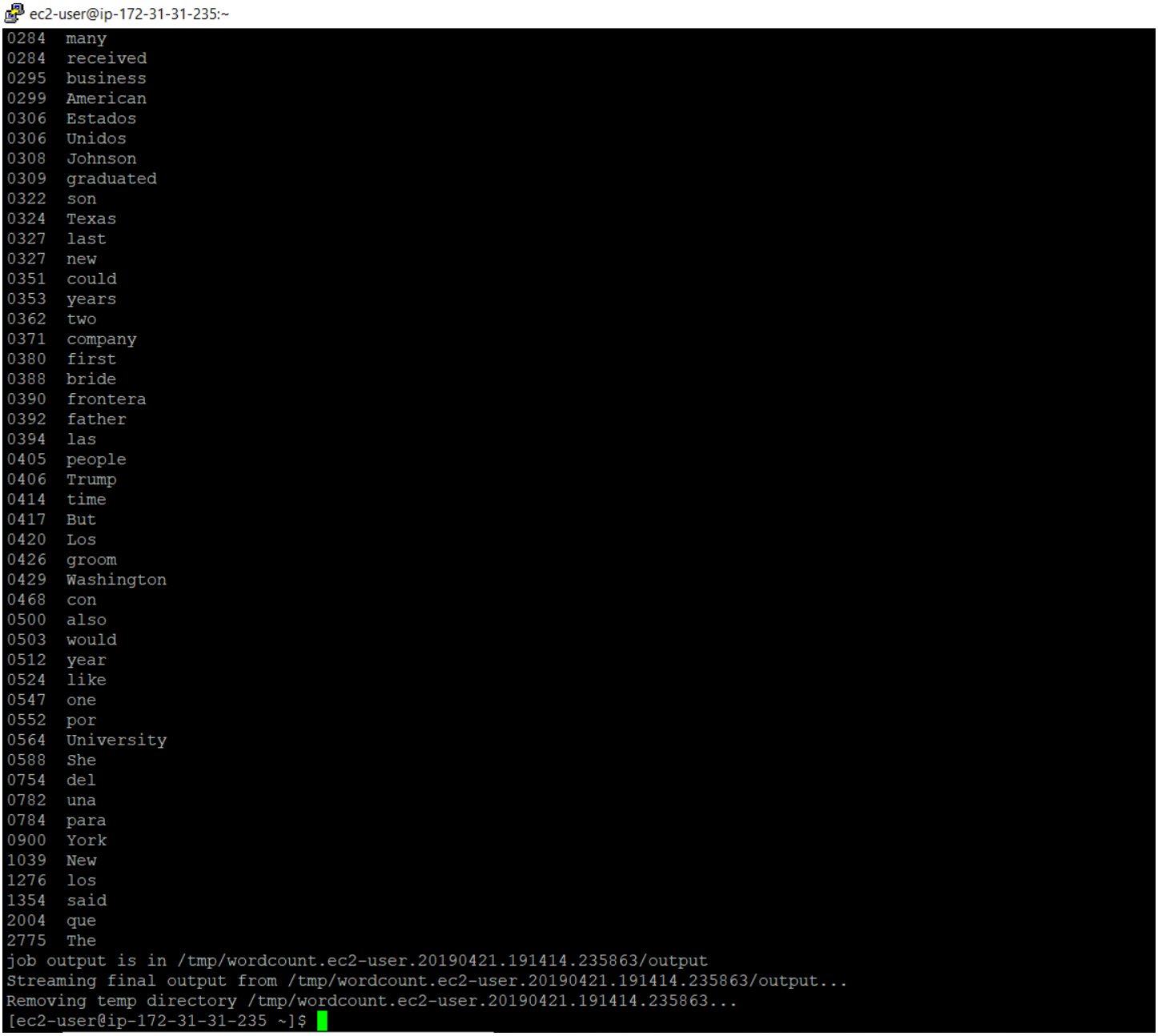
aws s3 ls – To test the connection and seeing the list of files

aws s3 cp s3://diclabmapper. –recursive – For downloading the files from S3 to the instance

alias python=python3 – Updating the version of python to Python 3

Sample code for running the word count map reduce script and writing the data to the S3 bucket

python3 wordcount.py White_ny.txt > white_ny.txt && aws s3 cp white_ny.txt s3://diclabmapper/data/



Analysis and Visualization

The outcomes of the codes were analyzed and we tried to see if the words for different races are comparable in any manner or not.

- Where black people were mostly talking mostly about USA and people, Hispanics were talking about jobs, their native countries like Columbia and Hispanic celebs like Cardi B on twitter.
- Where the NYtimes articles for Asian people mention work, code and computer which indicate the orientation of Asians, the articles for whites were mostly oriented towards states like NY.

The data was then visualized using word clouds and tree maps for word count outputs. Co-occurrence outputs were visualized using word clouds and bar plots. The filters in the dashboard make it easy for the viewers to compare the words for different races across different data sources.
The tree plot was used to compare the words and their volume for different races at a time which couldn't be achieved clearly using word clouds.

The dashboard was published on public tableau server the link for which is mentioned below.

We have made a video to take you through the process within 5 minutes. It can be found here: