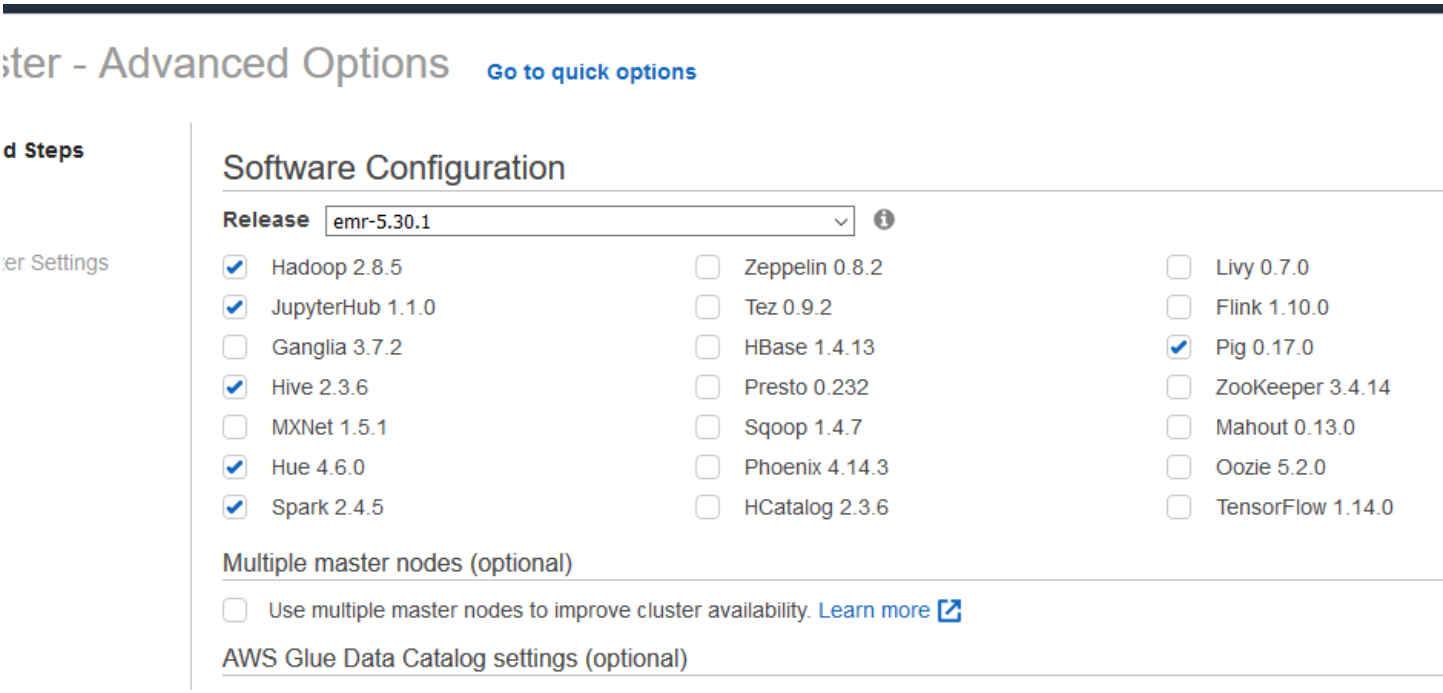
Big Data Wrangling With Google Books Ngrams

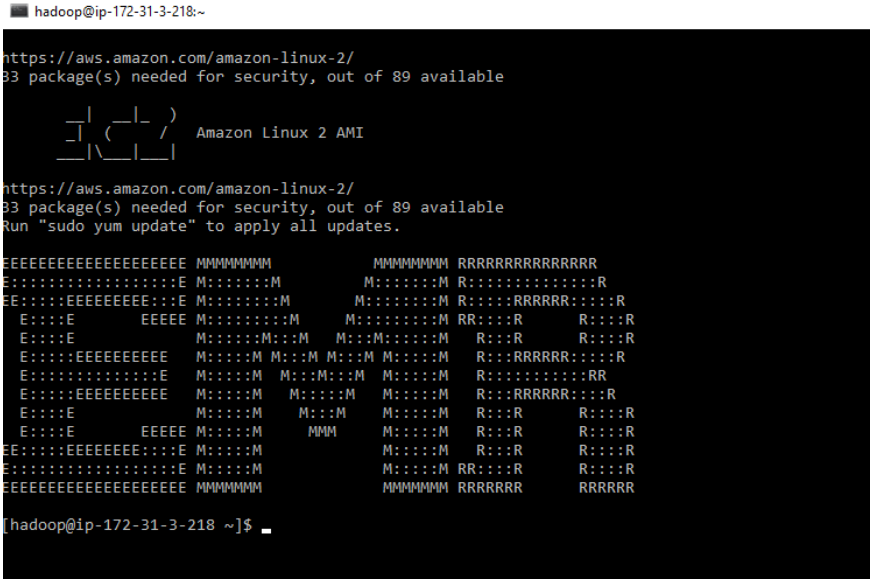
Let's start with questions which can be answered with screenshots:

1. **Spin up a new EMR cluster using the AWS Console. Be sure to include Hadoop, Hive, Spark, and Jupyterhub for your cluster.**



- Here, we have started an EMR cluster with the necessary libraries.

2. **Connect to the head node of the cluster using SSH**



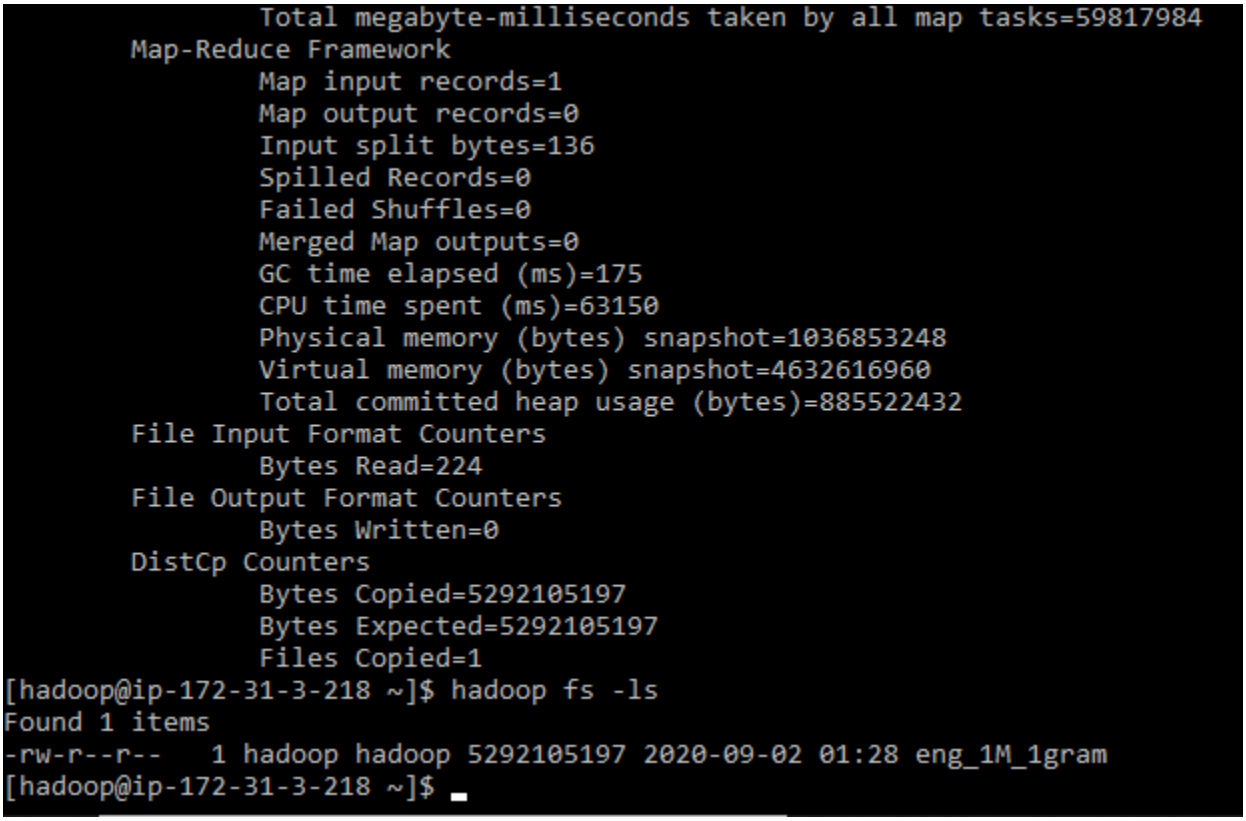
- After initiating the SSH connection we are in the Hadoop directory with EMR.

3. **Copy the data folder from the S3 bucket into the /user/hadoop/eng_1M_1gram directory in the Hadoop file system (HDFS) using distcp. The full path for the 1-gram data directory is s3://brainstation-dsft/eng_1M_1gram.csv**
   - By using hadoop distcp s3://brainstation-dsft/eng_1M_1gram.csv /user/hadoop/eng_1M_1gram we copied the data from S3 bucket to HDFS.

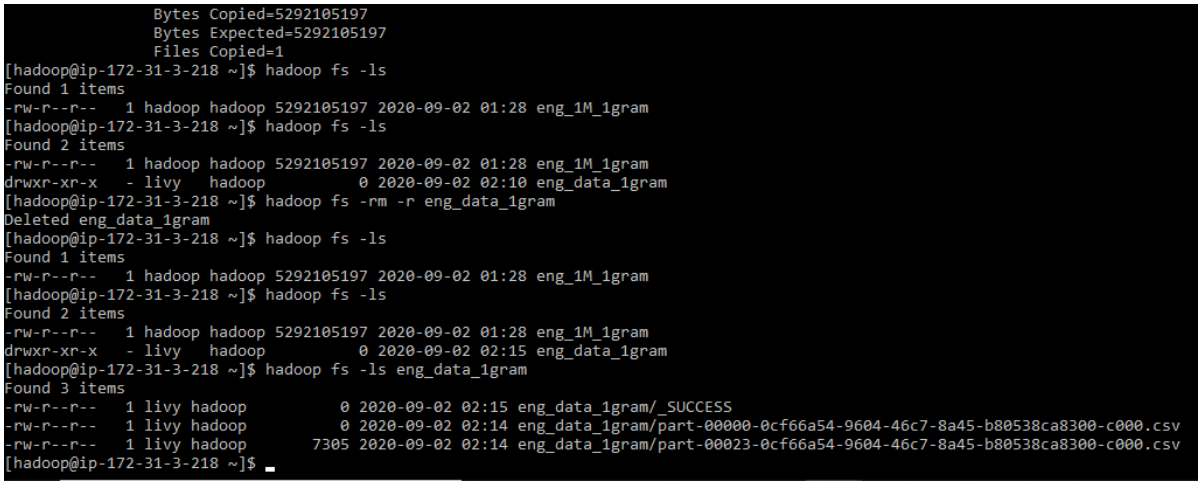
```
The general command line syntax is
command [genericOptions] [commandOptions]

[hadoop@ip-172-31-3-218 ~]$ hadoop distcp s3://brainstation-dsft/eng_1M_1gram.csv /user/hadoop/eng_1M_1gram
20/09/02 01:27:10 INFO tools.DistCp: Input Options: DistCpOptions{atomicCommit=false, syncFolder=false, deleteMissi
lse, ignoreFailures=false, overwrite=false, skipCRC=false, blocking=true, numListstatusThreads=0, maxMaps=20, mapBa
th=100, sslConfigurationFile='null', copyStrategy='uniformsize', preserveStatus=[], preserveRawXattrs=false, atomic
ath=null, logPath=null, sourceFileListing=null, sourcePaths=[s3://brainstation-dsft/eng_1M_1gram.csv], targetPath=/
hadoop/eng_1M_1gram, targetPathExists=false, filtersFile='null'}
20/09/02 01:27:10 INFO client.RMProxy: Connecting to ResourceManager at ip-172-31-3-218.ca-central-1.compute.intern
2.31.3.218:8032
20/09/02 01:27:13 INFO tools.SimpleCopyListing: Paths (files+dirs) cnt = 1; dirCnt = 0
20/09/02 01:27:13 INFO tools.SimpleCopyListing: Build file listing completed.
20/09/02 01:27:13 INFO Configuration.deprecation: io.sort.mb is deprecated. Instead, use mapreduce.task.io.sort.mb
20/09/02 01:27:13 INFO Configuration.deprecation: io.sort.factor is deprecated. Instead, use mapreduce.task.io.sort
or
20/09/02 01:27:13 INFO tools.DistCp: Number of paths in the copy list: 1
20/09/02 01:27:13 INFO tools.DistCp: Number of paths in the copy list: 1
20/09/02 01:27:14 INFO client.RMProxy: Connecting to ResourceManager at ip-172-31-3-218.ca-central-1.compute.intern
2.31.3.218:8032
20/09/02 01:27:14 INFO mapreduce.JobSubmitter: number of splits:1
20/09/02 01:27:14 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1599009777952_0001
20/09/02 01:27:14 INFO impl.YarnClientImpl: Submitted application application_1599009777952_0001
20/09/02 01:27:14 INFO mapreduce.Job: The url to track the job: http://ip-172-31-3-218.ca-central-1.compute.interna
88/proxy/application_1599009777952_0001/
20/09/02 01:27:14 INFO tools.DistCp: DistCp job-id: job_1599009777952_0001
20/09/02 01:27:14 INFO mapreduce.Job: Running job: job_1599009777952_0001
20/09/02 01:27:20 INFO mapreduce.Job: Job job_1599009777952_0001 running in uber mode : false
20/09/02 01:27:20 INFO mapreduce.Job:  map 0% reduce 0%
```

The above screenshot shows that the command is running a job and the following screenshot shows that the file in now in our HDFS.

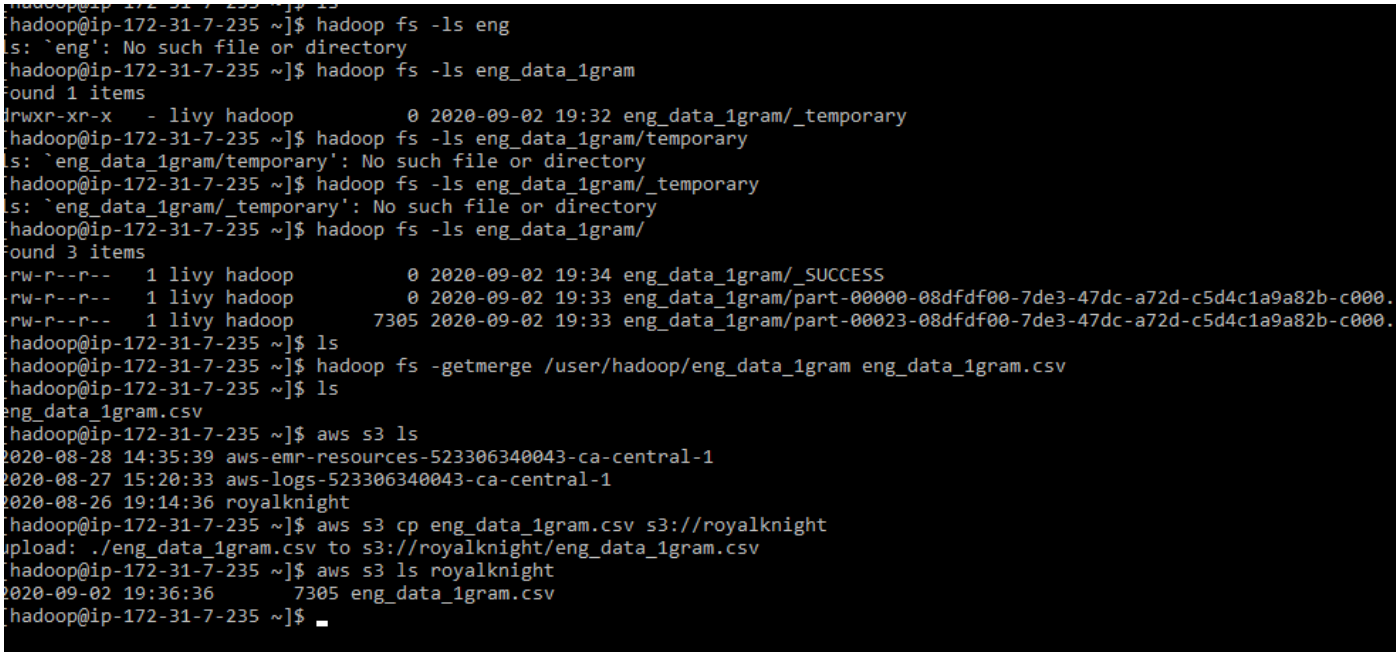


4. Write the filtered data back to a directory in the Hadoop filesystem from Spark using df.write.csv() - e.g. /user/hadoop/eng_data_1gram. Be sure to pass the header=True parameter. Examine the contents of what you've written using hadoop fs -ls.
   - Using filtered.write.option("header","true").csv("/user/hadoop/eng_data_1gram") We wrote the filtered data in the HDFS with a file name ‘eng_data_1gram’ We can check the contents with hdfs fs –ls:



After running hdfs fs –ls again we see that the ‘filtered’ file with the token as ‘data’ was written in our HDFS under the name ‘eng_data_1gram.’

5. Collect the contents of the directory into a single file on the local drive of the head node using getmerge: e.g. hadoop fs -getmerge /user/hadoop/eng_data_1gram eng_data_1gram.csv. Move this file into a S3 bucket in your account using aws s3 cp.
   - The code- hadoop fs -getmerge /user/hadoop/eng_data_1gram eng_data_1gram.csv merges the dataset from the HDFS into the local directory.



The ls command shows the contents in the local directory which can be seen that the data file in now in the local directory.

Also using aws s3 cp eng_data_1gram.cdv s3://royalknight copied the data file into my s3 bucket ‘royalknight’ which can be confirmed with aws s3 ls royalknight which shows the contents of the data in the s3 bucket.

The rest of the sub questions of the 4th is in the 1gram.ipynb notebook.

And the 6th question is in the Plot of 1gram.ipynb.