Taylor Shrode
Use Hadoop on a Cluster
2/26/2020
MSDS 610

Written Portion

1. What is a Hadoop distribution? What are some distributions of Hadoop?

   Hadoop is considered a "distributed system because the framework splits files into large data blocks and distributes them across nodes in a cluster (Schaefer, 2016)." Hadoop distributions include the following modules (Schaefer, 2016):

   a. Hadoop Common: collection of libraries leveraged by Hadoop modules
   b. Hadoop Distributed File System (HDFS): stores the data and delivers bandwidth across clusters.
   c. Hadoop MapReduce: processing model
   d. Hadoop YARN: manages computing resources within clusters.

   The most popular distributions of Hadoop include Cloudera, Hortonworks, and MapR (Shaw, 2014). Other popular distributions include Apache and Pivotal HD (Brundesh, 2016). For training, Cloudera, founded in March 2009, contains a professional training program but is more expensive due to the training program and the exams included (Shaw, 2014). Where most of the other Hadoop distributions support Linux operating systems, Hortonworks, founded in June 2011, supports the Microsoft Windows operating system. This allows Hortonworks to be used for "on-premise Hadoop installations or be run in Windows Azure cloud service (Shaw, 2014)." Hortonworks also has tutorials online, which makes Hortonworks more user-friendly (Shaw, 2014). As a bonus, the Hortonworks Data Platform is completely free (Brundesh, 2016). MapR, founded in 2009, comes with its own management code. In other words, there are different grades of the product, named M3, M5, and M7. The M3 is a free version without high availability, M5 is a standard commercial distribution, and the M7 is a paid version (Brundesh, 2016).

2. How does a Hadoop cluster work?

   The clusters in Hadoop is a "set of connected computers which work together as a single system" (Dataflair Team, 2018). A Hadoop cluster is made up of a collection of nodes. A node "is a point of connection within a network" (Dataflair Team, 2018). Hadoop clusters also have two types of machines: Master and Slave.

   The Master node is responsible for storing data in HDFS and executing computation the stored data using MapReduce (Dezyre, 2017). The Master node also has three nodes NameNode, Secondary NameNode, and JobTracker. NameNode keeps track of all of the information on files and handles the data storage function with HDFS, the Secondary NameNode keeps a backup of the NameNode data, and the JobTracker monitors the processing of data using MapReduce (Dezyre, 2017). The Slave Node is responsible for

storing the data and performing computations (Dezyre, 2017). It is recommended to keep the Master and Slave Nodes separate (Dataflair Team, 2018).

3. Why does Hadoop create multiple output files? How does this relate to the reducer step and number of compute nodes?

   The output files in Hadoop are named *part-x-yyyyy* where *part* is the output name, *x* is either denoted by a *m* or *r,* depending on whether the job was a map-only job or a reduce job, and *yyyyy* represents the mapper or reducer task number (White, 2012). Consider a job that has 10 reducers, the files will be named *part-r-00000* to *part-r-00010.* There is a file for each reducer task (White, 2012).

4. Why should we use version control to store code for projects?

   A version control system is "a category of software tools that help a software team manage changes to source code over time (Atlassian)." In other words, version control software keeps a complete long-term history of every change you make to the code in another database (Atlassian). It keeps track of changes such as the creation of files, deletion of files, edits to files, renaming of files, and even the moving of files. In addition to these changes, the history of changes keeps track of the author, date, and notes of each change (Atlassian).

   A version control system offers the ability to "branch" and "merge". By creating a "branch" allows individuals to keep multiple streams of work independent from others when working in a team. The "merge" function then allows these individuals to merge the work back together (Atlassian). Another benefit of version control systems is its traceability. Individuals can trace each change made to the software. This can be especially useful when working with a long-term design of the system so people working on the system can see what was changed and why it was changed. This allows the developers to make the correct changes (Atlassian).

5. What are the pros/cons of using a cloud provider for a Hadoop cluster vs using our own in-house machines?

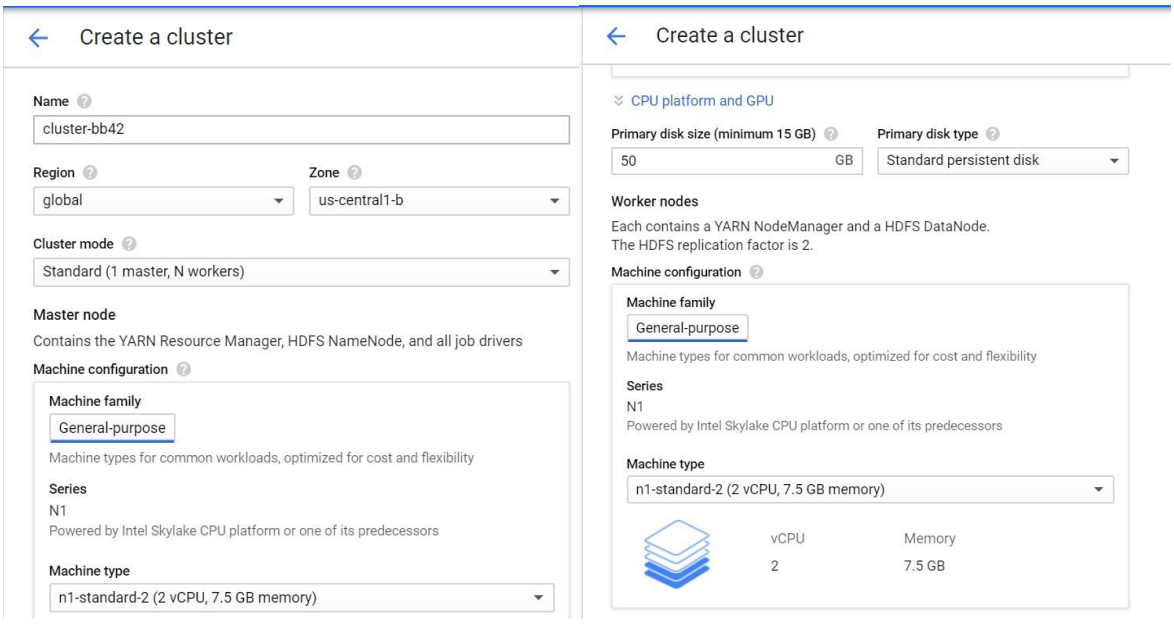
   When companies are deciding whether to use a cloud provider for a Hadoop cluster or use their own in-house machines, there are factors to consider. The top factors that need to be considered include cost, security, current capabilities, and scalability (Shaik, 2017). An in-house machine is typically cheaper, gives you physical data isolation and privacy, and gives you full control of the Hadoop hardware and software (Shaik, 2017). On the other hand, an in-house machine has smart hand to install servers and networks and they have "hardware friction" when getting new infrastructure. Now, using a cloud provider

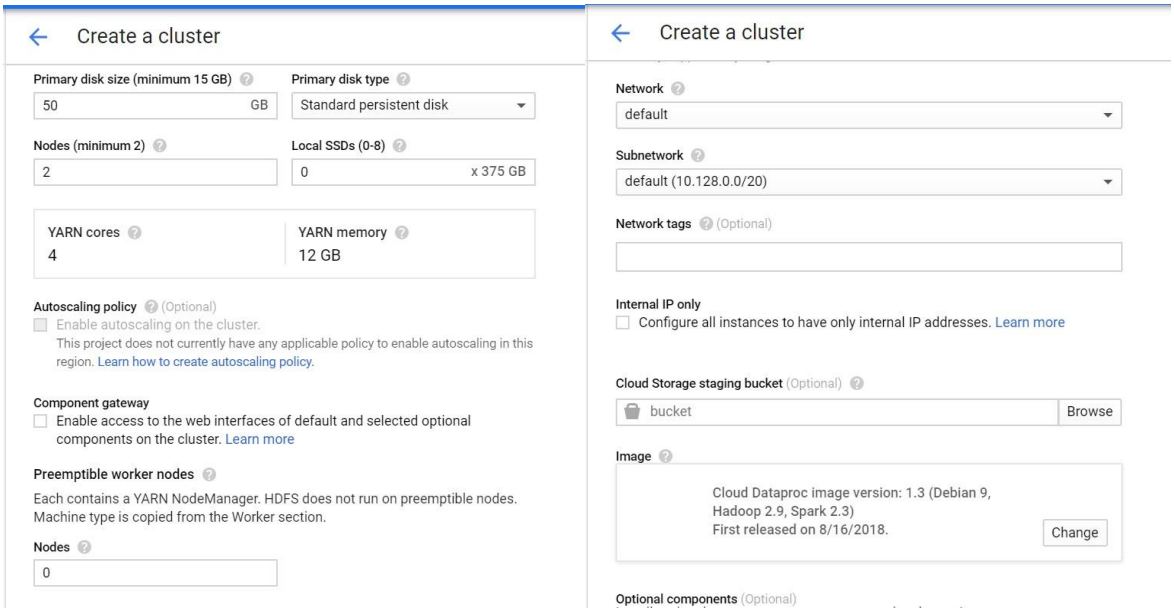
integrates well with data sources already stored in the cloud and allows for software based managed service. However, a cloud provider does not allow a user any control of hardware and software level privacy (Shaik, 2017).

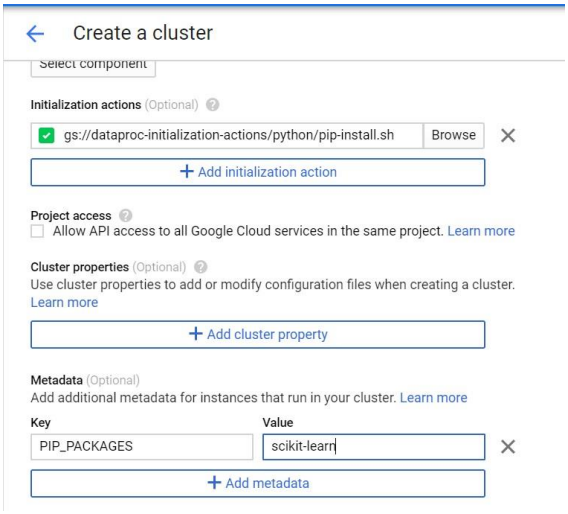
Technical Portion

For this assignment, we expand on using Hadoop for counting words in a given text file. We will be using a text by Shakespeare. In the expansion on the word counts of our text, we will be removing punctuation and stopwords. A stopword is considered as a commonly used word. This includes words such as "the", "and", and "at" which don't tell us much about a given text. This can be done in multiple different ways, but we will be installing the **NLTK** Python library in order to get a list of stopwords to remove. To install this library, we need to use "Initialization". To use initialization, we need to create a GCP cluster and set the metadata value **PIP_PACKAGES** to be **scikit-learn**.

Below, we will fork the GitHub repo (make a copy of the repository) from the previous MapReduce example, create the GCP cluster, and access it by using SSH in the browser. The previous MapReduce example can be found here. First, by using the 'fork' option in Github, this allows us to experiment with changes on the original repository without affecting it (Fork a Repo). After using fork on the MapReduce example, we can create our cluster. Below are the changes made to the default cluster.







Now that we have successfully created a new cluster of three machines, we can run the cluster by opening an SSH terminal window.

To begin this wordcount exercise, we need to 'clone' the forked version of the Github repository from above. This is done by using the **git clone <repo html link>** command. Next, the Shakespeare text file needs to be downloaded by using the **wget http://norvig.com/ngrams/shakespeare.txt** command. Next, we create the necessary directories in Hadoop. It should be noted that the **hdfs dfs** command were initially used to create these directories (which is shown below) but were recreated using the **hadoop fs** command (not

shown). The **-copyFromLocal** command is used to copy the file to Hadoop and the **-ls** command is used to verify that the file was copied to the input directory.

```
g07hockeychix67@cluster-bb42-m:~$ git clone https://github.com/tshrode37/simple_H
adoop_MapReduce_example.git
Cloning into 'simple_Hadoop_MapReduce_example'...
remote: Enumerating objects: 9, done.
remote: Counting objects: 100% (9/9), done.
remote: Compressing objects: 100% (7/7), done.
remote: Total 19 (delta 2), reused 8 (delta 2), pack-reused 10
Unpacking objects: 100% (19/19), done.
g07hockeychix67@cluster-bb42-m:~$ ls
simple_Hadoop_MapReduce_example
g07hockeychix67@cluster-bb42-m:~$ wget http://norvig.com/ngrams/shakespeare.txt
--2020-01-24 00:06:47--  http://norvig.com/ngrams/shakespeare.txt
Resolving norvig.com (norvig.com)... 158.106.138.13
Connecting to norvig.com (norvig.com)|158.106.138.13|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 4538523 (4.3M) [text/plain]
Saving to: 'shakespeare.txt'

shakespeare.txt      100%[==================>]   4.33M  15.6MB/s    in 0.3s

2020-01-24 00:06:47 (15.6 MB/s) - 'shakespeare.txt' saved [4538523/4538523]

g07hockeychix67@cluster-bb42-m:~$ ls
shakespeare.txt  simple_Hadoop_MapReduce_example
g07hockeychix67@cluster-bb42-m:~$ hdfs dfs -mkdir /shakespeare
g07hockeychix67@cluster-bb42-m:~$ hdfs dfs -mkdir /shakespeare/input
g07hockeychix67@cluster-bb42-m:~$ hdfs dfs -copyFromLocal shakespeare.txt /shakes
peare/input
g07hockeychix67@cluster-bb42-m:~$ hdfs dfs -ls /shakespeare/input
Found 1 items
-rw-r--r--   2 g07hockeychix67 hadoop    4538523 2020-01-24 00:07 /shakespeare/in
put/shakespeare.txt
```

Next, we use the **cd** command to change the directory in order to run the Python scripts in the **simple_Hadoop_MapReduce_example** folder.

```
g07hockeychix67@cluster-bb42-m:~$ cd simple_Hadoop_MapReduce_example
g07hockeychix67@cluster-bb42-m:~/simple_Hadoop_MapReduce_example$ ls
LICENSE  mapper.py  README.md  reducer.py
g07hockeychix67@cluster-bb42-m:~/simple_Hadoop_MapReduce_example$
```

Now, we can run the script below to count the words in the **shakespeare.txt** file. Partial outputs are shown below.

```
g07hockeychix67@cluster-bb42-m:~/simple_Hadoop_MapReduce_example$ hadoop jar /usr
/lib/hadoop-mapreduce/hadoop-streaming.jar  -files mapper.py,reducer.py  -mapper
mapper.py  -reducer reducer.py  -input /shakespeare/input   -output /shakespeare/
output
packageJobJar: [] [/usr/lib/hadoop-mapreduce/hadoop-streaming-2.9.2.jar] /tmp/str
eamjob7892204751858321225.jar tmpDir=null
20/01/24 01:57:22 INFO client.RMProxy: Connecting to ResourceManager at cluster-b
b42-m/10.128.0.5:8032
20/01/24 01:57:22 INFO client.AHSProxy: Connecting to Application History server
at cluster-bb42-m/10.128.0.5:10200
20/01/24 01:57:23 INFO client.RMProxy: Connecting to ResourceManager at cluster-b
b42-m/10.128.0.5:8032
20/01/24 01:57:23 INFO client.AHSProxy: Connecting to Application History server
at cluster-bb42-m/10.128.0.5:10200
20/01/24 01:57:23 WARN hdfs.DataStreamer: Caught exception
```

The end of the script with our output file is below. Notice that the output file was written to the Shakespeare output directory.

```
                Total committed heap usage (bytes)=/500988416
        Shuffle Errors
                BAD_ID=0
                CONNECTION=0
                IO_ERROR=0
                WRONG_LENGTH=0
                WRONG_MAP=0
                WRONG_REDUCE=0
        File Input Format Counters
                Bytes Read=4595867
        File Output Format Counters
                Bytes Written=356409
20/01/24 01:58:34 INFO streaming.StreamJob: Output directory: /shakespeare/output
g07hockeychix67@cluster-bb42-m:~/simple_Hadoop_MapReduce_example$
```

Next, we verify the output by using the **hadoop fs -ls** command.

```
g07hockeychix67@cluster-bb42-m:~/simple_Hadoop_MapReduce_example$ hadoop fs -ls /
shakespeare/output
Found 6 items
-rw-r--r--   2 g07hockeychix67 hadoop          0 2020-01-24 01:58 /shakespeare/ou
tput/_SUCCESS
-rw-r--r--   2 g07hockeychix67 hadoop      73141 2020-01-24 01:58 /shakespeare/ou
tput/part-00000
-rw-r--r--   2 g07hockeychix67 hadoop      71514 2020-01-24 01:58 /shakespeare/ou
tput/part-00001
-rw-r--r--   2 g07hockeychix67 hadoop      69560 2020-01-24 01:58 /shakespeare/ou
tput/part-00002
-rw-r--r--   2 g07hockeychix67 hadoop      70848 2020-01-24 01:58 /shakespeare/ou
tput/part-00003
-rw-r--r--   2 g07hockeychix67 hadoop      71346 2020-01-24 01:58 /shakespeare/ou
tput/part-00004
g07hockeychix67@cluster-bb42-m:~/simple_Hadoop_MapReduce_example$
```

Now, the **hadoop fs -getmerge** command is used to merge files in the HDFS file system into a single file in the local file system (Dataflair Team, 2019). The results can be viewed using the Linux **cat result | head** command. Using the **head** command limits the number of rows that are displayed.

```
g07hockeychix67@cluster-bb42-m:~/simple_Hadoop_MapReduce_example$ hadoop fs -getm
erge /shakespeare/output/ /home/g07hockeychix67/result
g07hockeychix67@cluster-bb42-m:~/simple_Hadoop_MapReduce_example$ cat /home/g07ho
ckeychix67/result | head
neat's-tongue   1
Hasting 1
long-since-due  1
Mortimer        41
wind-instruments        1
four    115
mean-born       1
spiders 3
railing 8
thunderstroke   1
g07hockeychix67@cluster-bb42-m:~/simple_Hadoop_MapReduce_example$
```

To sort the list, the command **sort -gr -k 2 /home/<username>/result | head** is used.

```
g07hockeychix67@cluster-bb42-m:~/simple_Hadoop_MapReduce_example$ sort -gr -k 2 /
home/g07hockeychix67/result | head
,       81827
.       36514
the     23272
I       20041
;       17274
and     16817
to      15506
of      15037
you     12361
a       12155
g07hockeychix67@cluster-bb42-m:~/simple_Hadoop_MapReduce_example$
```

We can see by our results above, that the top words used in the **shakespeare.txt** file include stopwords. We can remove these stopwords by using a simple Python set. Instead of creating a new Python file to do this, the nano editor (**nano mapper.py** command) can be used to change the python script as seen below.

```
#!/usr/bin/env python
import sys

stopwords = set(['the', 'and'])

# get all lines from stdin
for line in sys.stdin:
    # remove leading and trailing whitespace
    line = line.strip().lower()

    # split the line into words; splits on any whitespace
    words = line.split()

    # output tuples (word, 1) in tab-delimited format
    for word in words:
        if word not in stopwords:
            print '%s\t%s' % (word, "1")
```

The command to remove the stopwords “the” and “and” has been added as well as converting the text to lowercase before removing the stopwords. Before we run the wordcount script, the output directory needs to be deleted or we will receive an error.

```
g07hockeychix67@cluster-bb42-m:~/simple_Hadoop_MapReduce_example$ hadoop fs -rm -
r /shakespeare/output
Deleted /shakespeare/output
```

After running the wordcount script, verifying the output files, and merging the output files, we can view the results.

```
g07hockeychix67@cluster-bb42-m:~/simple_Hadoop_MapReduce_example$ cat /home/g07ho
ckeychix67/result | head
glamis  7
neat's-tongue   1
long-since-due  1
wind-instruments        1
four    129
fleeces 1
anchises'       1
railing 8
offendeth       1
vassals 3
g07hockeychix67@cluster-bb42-m:~/simple_Hadoop_MapReduce_example$ sort -gr -k 2 /
home/g07hockeychix67/result | head
,       81827
.       36514
i       20042
to      18533
;       17274
of      16007
you     13834
a       13679
my      12257
that    10719
g07hockeychix67@cluster-bb42-m:~/simple_Hadoop_MapReduce_example$
```

We can see that the results differ, but not by much. To remove a more comprehensive list of stopwords, we can use a stopwords list from **sklearn**. Punctuation also needs to be removed to get more interesting results (Cherian, 2019). After using the **nano mapper.py** command again, the new mapper file looks like,

Taylor Shrode
Use Hadoop on a Cluster
2/26/2020
MSDS 610

```
#!/usr/bin/env python
import sys
import string
from sklearn.feature_extraction import stop_words

stopwords = set(stop_words.ENGLISH_STOP_WORDS)
# get all lines from stdin
for line in sys.stdin:
    # remove leading and trailing whitespace
    line = line.strip().lower()
    punct = line.translate(string.maketrans("",""), string.punctuation)

    # split the line into words; splits on any whitespace

    words = punct.split()
    # output tuples (word, 1) in tab-delimited format
    for word in words:
        if word not in stopwords:
           print '%s\t%s' % (word, "1")
```

Following the same steps used to remove stopwords using a Python set, we can see our sorted results below.

```
g07hockeychix67@cluster-d97e-m:~/simple_Hadoop_MapReduce_example$ hdfs dfs -ls /shakespeare/output
Found 6 items
-rw-r--r--   2 g07hockeychix67 hadoop          0 2020-01-24 17:22 /shakespeare/output/_SUCCESS
-rw-r--r--   2 g07hockeychix67 hadoop      58456 2020-01-24 17:22 /shakespeare/output/part-00000
-rw-r--r--   2 g07hockeychix67 hadoop      56818 2020-01-24 17:22 /shakespeare/output/part-00001
-rw-r--r--   2 g07hockeychix67 hadoop      55844 2020-01-24 17:22 /shakespeare/output/part-00002
-rw-r--r--   2 g07hockeychix67 hadoop      57650 2020-01-24 17:22 /shakespeare/output/part-00003
-rw-r--r--   2 g07hockeychix67 hadoop      56472 2020-01-24 17:22 /shakespeare/output/part-00004
g07hockeychix67@cluster-d97e-m:~/simple_Hadoop_MapReduce_example$ hdfs dfs -getmerge /shakespeare/output/ /home/g07h
ockeychix67/result
g07hockeychix67@cluster-d97e-m:~/simple_Hadoop_MapReduce_example$ cat /home/g07hockeychix67/result | head
glamis  8
heartsorrow     1
spiders 5
railing 8
offendeth       1
vassals 3
pleasantspirited        1
reposeth        1
stirrest        1
mutinies        4
g07hockeychix67@cluster-d97e-m:~/simple_Hadoop_MapReduce_example$ sort -gr -k 2 /home/g07hockeychix67/result | head
thou    5357
thy     3811
shall   3603
thee    3103
good    2817
lord    2717
o       2615
come    2565
sir     2541
let     2108
g07hockeychix67@cluster-d97e-m:~/simple_Hadoop_MapReduce_example$
```

Further, we can view the top thirty results. Notice how there are no stopwords and no punctuation found in our list.

```
g07hockeychix67@cluster-d97e-m:~/simple_Hadoop_MapReduce_example$ sort -gr -k 2 /home/g07hockeychix67/result | head
-n 30
thou    5357
thy     3811
shall   3603
thee    3103
good    2817
lord    2717
o       2615
come    2565
sir     2541
let     2108
ill     2002
hath    1917
love    1898
man     1808
like    1717
know    1676
say     1666
make    1639
did     1623
tis     1419
king    1363
speak   1175
tell    1068
think   1036
time    1005
heart   941
great   897
art     886
hear    884
doth    864
g07hockeychix67@cluster-d97e-m:~/simple_Hadoop_MapReduce_example$
```

From our list, we can gather words like “thou”, “thy”, “thee”, and “doth” and we can see the type of language that was used during Shakespeare’s lifetime. We can also get a sense of the region he grew up in by analyzing the words used the most in his writings.

Finally, we can update our Github repository. Before using the **commit** command, we must set our username and user email by using the **git config --global** command.

```
g07hockeychix67@cluster-d97e-m:~/simple_Hadoop_MapReduce_example$ git config --global user.name "Taylor Shrode"
g07hockeychix67@cluster-d97e-m:~/simple_Hadoop_MapReduce_example$ git config --global user.email "tshrode37@gmail.co
m"
```

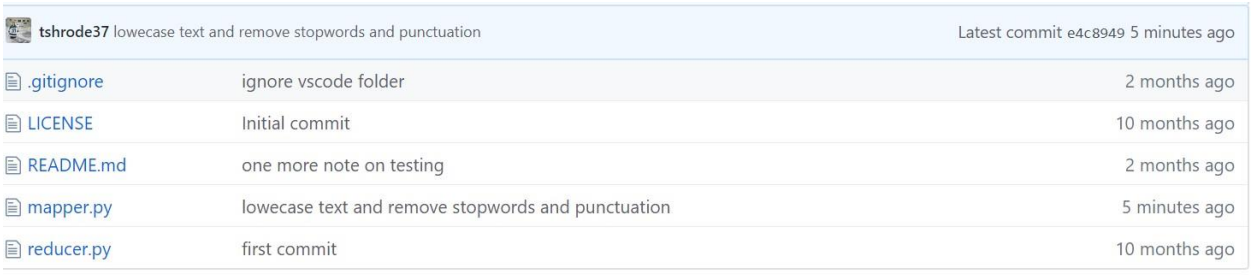
Next, we navigate to the GitHub repository folder in the master node and use the **git status** command to view what files have been changed. Once we have found which files have been changed in the repository, we use the **git add <file name>** command to add the changed file to the repository. If multiple changed files need to be added, the **git add *** command can be used, or if deleted files need to be added, the command **git add .** can be used. Now, the **git commit -m ‘lowercase text and remove stopwords and punctuation’** ‘commits’ our code changes with the message in quotes.

```
g07hockeychix67@cluster-d97e-m:~/simple_Hadoop_MapReduce_example$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   mapper.py

no changes added to commit (use "git add" and/or "git commit -a")
g07hockeychix67@cluster-d97e-m:~/simple_Hadoop_MapReduce_example$ git add mapper.py
g07hockeychix67@cluster-d97e-m:~/simple_Hadoop_MapReduce_example$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   mapper.py

g07hockeychix67@cluster-d97e-m:~/simple_Hadoop_MapReduce_example$ git commit -m 'lowecase text and remove stopwords
and punctuation'
[master e4c8949] lowecase text and remove stopwords and punctuation
 1 file changed, 8 insertions(+), 3 deletions(-)
```

Finally, we can push the files to our GitHub repository after entering the username and password to out GitHub account.

```
g07hockeychix67@cluster-d97e-m:~/simple_Hadoop_MapReduce_example$ git push origin master
Username for 'https://github.com': tshrode37
Password for 'https://tshrode37@github.com':
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 2 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 621 bytes | 621.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/tshrode37/simple_Hadoop_MapReduce_example.git
   282cb94..e4c8949  master -> master
g07hockeychix67@cluster-d97e-m:~/simple_Hadoop_MapReduce_example$
```

To verify that the files have been uploaded successfully, we can go to our GitHub account and check.



As we can see above, after removing stopwords and punctuation from a given text, we can analyze the top word counts found in the text. A researcher can grasp what time period and region a writer comes from when they can analyze the language used. They could also find out what genre the text is by analyzing the language the writer uses.

Taylor Shrode
Use Hadoop on a Cluster
2/26/2020
MSDS 610

Resources

Atlassian. (n.d.). *What is version control | Atlassian Git Tutorial*. Atlassian. Retrieved January 25, 2020, from https://www.atlassian.com/git/tutorials/what-is-version-control

Brundesh, R. (2016, April 14). Which Hadoop distribution is right for you? |. *AcadGild*. https://acadgild.com/blog/hadoop-distribution-right

Cherian, A. (2019, March 5). *Python—Best way to strip punctuation from a string*. Stack Overflow. https://stackoverflow.com/questions/265960/best-way-to-strip-punctuation-from-a-string/266162

Dataflair Team. (2018, November 14). *What is Hadoop Cluster | Hadoop Cluster Architecture*. https://data-flair.training/blogs/hadoop-cluster/

Dataflair Team. (2019, April 6). *Hadoop GetMerge Command*. https://data-flair.training/blogs/hadoop-getmerge-command/

Dezyre. (2017, June 22). *Hadoop Cluster Overview: What it is and how to setup one?* https://www.dezyre.com/article/hadoop-cluster-overview-what-it-is-and-how-to-setup-one/356

Fork a repo—Github help. (n.d.). Retrieved January 26, 2020, from https://help.github.com/en/enterprise/2.13/user/articles/fork-a-repo

Schaefer, P. (2016, September 22). Hadoop distribution | future of Hadoop distributions. *Trifacta*. https://www.trifacta.com/blog/hadoop-distribution/

Shaik, S. (2017, January 2). *Hadoop on cloud vs Hadoop on premises*. https://www.linkedin.com/pulse/hadoop-cloud-vs-premises-shahebaz-shaik

Shaw, M. (2014, January 13). *Top 3 Hadoop distributions, which is right for you?* Pluralsite LLC. https://www.pluralsight.com/blog/data-professional/top-3-hadoop-distributions

White, C. (2012, May 19). *MapReduce—What are success and part-r-00000 files in Hadoop*. Stack Overflow. https://stackoverflow.com/questions/10666488/what-are-success-and-part-r-00000-files-in-hadoop/10666874