Lab 05 - Wordcount program in Hadoop

Full name: Chloe Tee Rouyi

Student ID: 0354731

Tasks:

1. Open Eclipse inside the Cloudera platform

   Note: If Eclipse software is not installed inside you Cloudera VM, please install it.

2. Create a Java project and a java file:

   a. File > New > Java Project

   b. Give a project name ex; “WordCount”

   c. Hit “Next” and in the next page, click on “Libraries” tab

   d. Click on “Add External JARs” button

   e. Navigate to “File System>usr>lib>Hadoop”

   f. Select all *.jar files and click “OK”

   g. Click on “Add External JARs” button again

   h. Navigate to “File System>usr>lib>Hadoop>client”

   i. Select all *.jar files (ctrl+A) and click “OK”

   j. Give a moment to load all the jar files. Once the jar files are added to the list of libraries, then click “Finish” button

   k. Inside the “Package Explorer”, and under the project that you created above (WordCound), right click on src and from “New”, select option “Class”

l. In the Java Class window, just give a meaningful name for the file name (ex; WordCount) and the click on “Finish” button. This will create a java file with the give name (ex; WordCount.java)

m. Delete the content of the newly created file and then copy the source code from the like bellow or from the resources at the bottom and paste the code inside the file.

   i. Copy the source code from the below link https://hadoop.apache.org/docs/current/hadoop-mapreduceclient/hadoop-mapreduce-clientcore/MapReduceTutorial.html#Source_Code

n. Check the code and ensure there is no error before you save it.

o. Now right click on the project name “WordCount” inside “Package Explorer” panel and select “Export”.

p. Open Java from the list and select “JAR File”, the click “Next”

q. In the next window, click on “Browse…” button and navigate to a specific folder ex; /home/cloudera

r. Give a name file for exporting jar file (ex; WordCount.jar) and click “OK”

s. Click “Finish” button

t. Check the jar file is available inside the “cloudera” folder

3. Open terminal

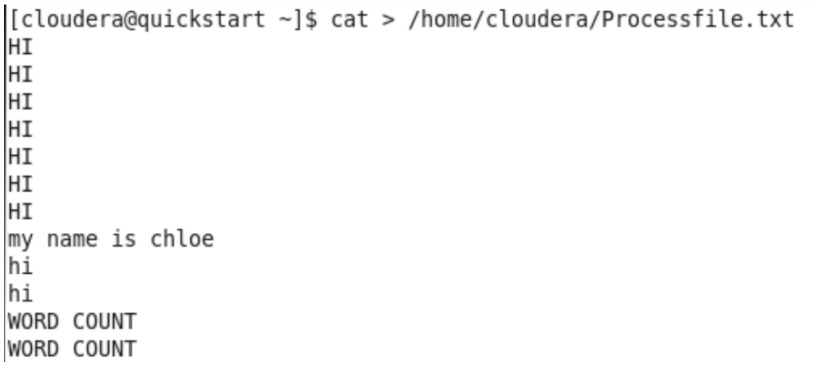
4. Create a simple text file (to do that, follow the any methods you learn in previous labs)

    a. Use the command: cat > /home/cloudera/Processfile.txt

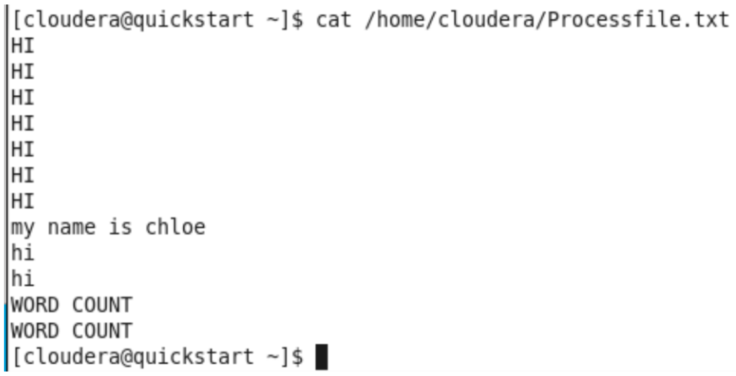
    b. Add some lines and try to use duplicated words inside

    c. To save and close the file opened by cat function: ctrl+d or ctrl+z

    d. Check the file is created and check the content to ensure the inserted content is saved. cat /home/cloudera/Processfile.txt

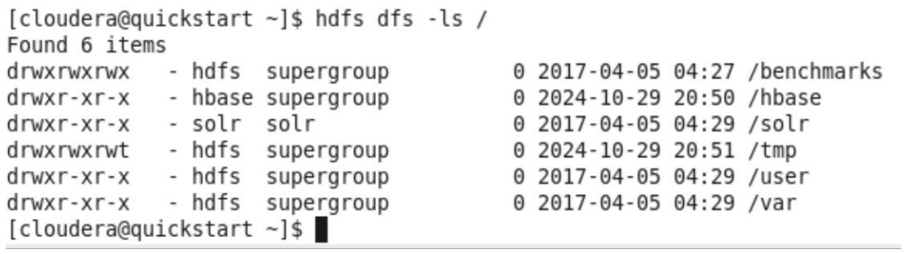


```
[cloudera@quickstart ~]$ cat > /home/cloudera/Processfile.txt
HI
HI
HI
HI
HI
HI
HI
my name is chloe
hi
hi
WORD COUNT
WORD COUNT
```



```
[cloudera@quickstart ~]$ cat /home/cloudera/Processfile.txt
HI
HI
HI
HI
HI
HI
HI
my name is chloe
hi
hi
WORD COUNT
WORD COUNT
[cloudera@quickstart ~]$ █
```

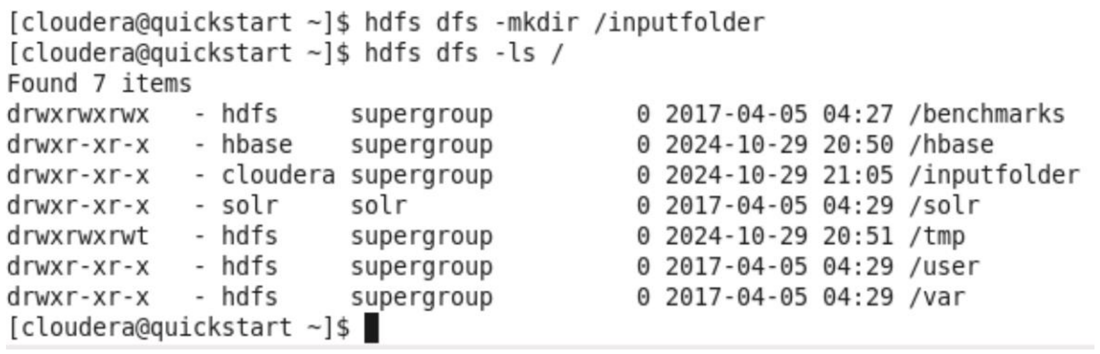
5. Check hdfs files and directory list: hdfs dfs -ls /



```
[cloudera@quickstart ~]$ hdfs dfs -ls /
Found 6 items
drwxrwxrwx   - hdfs  supergroup          0 2017-04-05 04:27 /benchmarks
drwxr-xr-x   - hbase supergroup          0 2024-10-29 20:50 /hbase
drwxr-xr-x   - solr  solr                0 2017-04-05 04:29 /solr
drwxrwxrwt   - hdfs  supergroup          0 2024-10-29 20:51 /tmp
drwxr-xr-x   - hdfs  supergroup          0 2017-04-05 04:29 /user
drwxr-xr-x   - hdfs  supergroup          0 2017-04-05 04:29 /var
[cloudera@quickstart ~]$ █
```

6. Create a folder inside the root of hdfs: hdfs dfs -mkdir /inputfoler

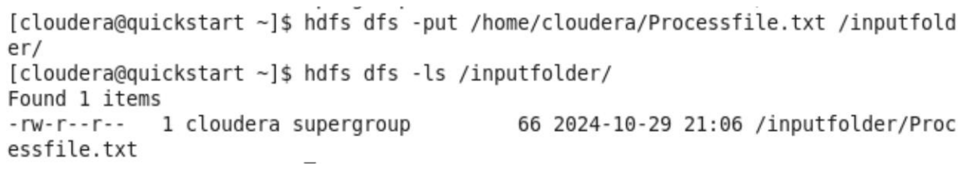
7. Check the folder is created



```
[cloudera@quickstart ~]$ hdfs dfs -mkdir /inputfolder
[cloudera@quickstart ~]$ hdfs dfs -ls /
Found 7 items
drwxrwxrwx   - hdfs     supergroup          0 2017-04-05 04:27 /benchmarks
drwxr-xr-x   - hbase    supergroup          0 2024-10-29 20:50 /hbase
drwxr-xr-x   - cloudera supergroup          0 2024-10-29 21:05 /inputfolder
drwxr-xr-x   - solr     solr                0 2017-04-05 04:29 /solr
drwxrwxrwt   - hdfs     supergroup          0 2024-10-29 20:51 /tmp
drwxr-xr-x   - hdfs     supergroup          0 2017-04-05 04:29 /user
drwxr-xr-x   - hdfs     supergroup          0 2017-04-05 04:29 /var
[cloudera@quickstart ~]$ 
```

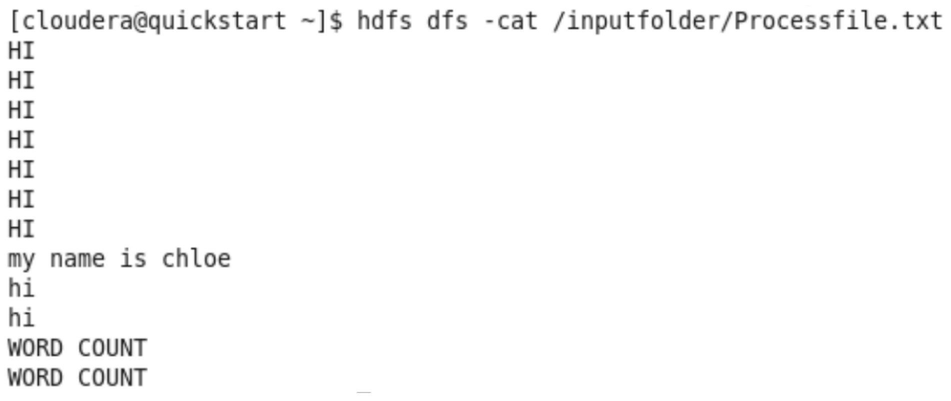
8. Copy Processfile.txt from local storage into inputfoler inside hdfs

    a. Command: hdfs dfs -put /home/cloudera/Processfile.txt /inputfolder/



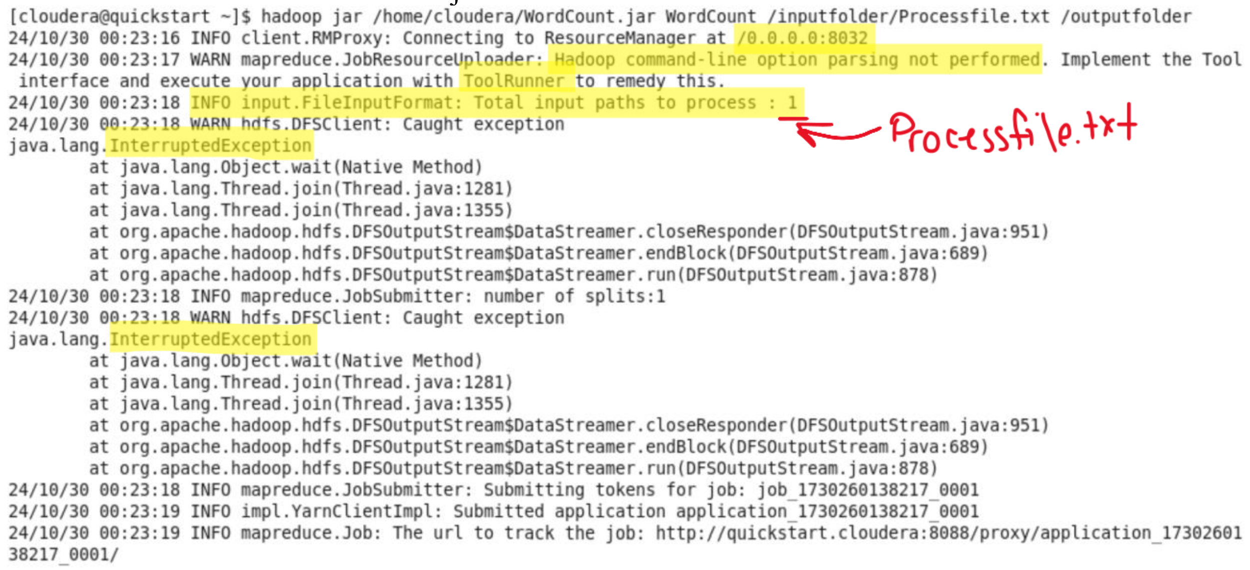
```
[cloudera@quickstart ~]$ hdfs dfs -put /home/cloudera/Processfile.txt /inputfold
er/
[cloudera@quickstart ~]$ hdfs dfs -ls /inputfolder/
Found 1 items
-rw-r--r--   1 cloudera supergroup         66 2024-10-29 21:06 /inputfolder/Proc
essfile.txt
```

    b. Check the content of file: hdfs dfs -cat /inputfolder/Processfile.txt



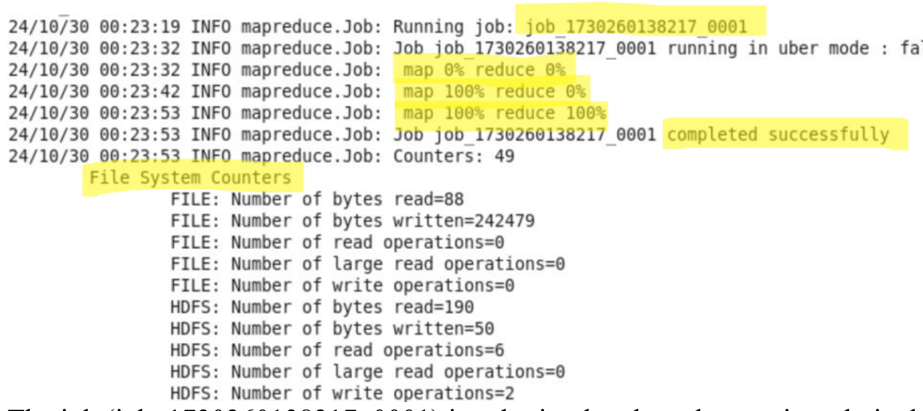
```
[cloudera@quickstart ~]$ hdfs dfs -cat /inputfolder/Processfile.txt
HI
HI
HI
HI
HI
HI
HI
my name is chloe
hi
hi
WORD COUNT
WORD COUNT
```

9. Run the Hadoop syntax to run the wordcount
   a. Command: Hadoop jar /home/cloudera/WordCount.jar WordCount /inputfolder/Processfile.txt /outputfolder
   b. Hit enter and wait for the jar file to be executed



```
[cloudera@quickstart ~]$ hadoop jar /home/cloudera/WordCount.jar WordCount /inputfolder/Processfile.txt /outputfolder
24/10/30 00:23:16 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
24/10/30 00:23:17 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool
 interface and execute your application with ToolRunner to remedy this.
24/10/30 00:23:18 INFO input.FileInputFormat: Total input paths to process : 1
24/10/30 00:23:18 WARN hdfs.DFSClient: Caught exception
java.lang.InterruptedException
        at java.lang.Object.wait(Native Method)
        at java.lang.Thread.join(Thread.java:1281)
        at java.lang.Thread.join(Thread.java:1355)
        at org.apache.hadoop.hdfs.DFSOutputStream$DataStreamer.closeResponder(DFSOutputStream.java:951)
        at org.apache.hadoop.hdfs.DFSOutputStream$DataStreamer.endBlock(DFSOutputStream.java:689)
        at org.apache.hadoop.hdfs.DFSOutputStream$DataStreamer.run(DFSOutputStream.java:878)
24/10/30 00:23:18 INFO mapreduce.JobSubmitter: number of splits:1
24/10/30 00:23:18 WARN hdfs.DFSClient: Caught exception
java.lang.InterruptedException
        at java.lang.Object.wait(Native Method)
        at java.lang.Thread.join(Thread.java:1281)
        at java.lang.Thread.join(Thread.java:1355)
        at org.apache.hadoop.hdfs.DFSOutputStream$DataStreamer.closeResponder(DFSOutputStream.java:951)
        at org.apache.hadoop.hdfs.DFSOutputStream$DataStreamer.endBlock(DFSOutputStream.java:689)
        at org.apache.hadoop.hdfs.DFSOutputStream$DataStreamer.run(DFSOutputStream.java:878)
24/10/30 00:23:18 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1730260138217_0001
24/10/30 00:23:19 INFO impl.YarnClientImpl: Submitted application application_1730260138217_0001
24/10/30 00:23:19 INFO mapreduce.Job: The url to track the job: http://quickstart.cloudera:8088/proxy/application_17302601
38217_0001/
```

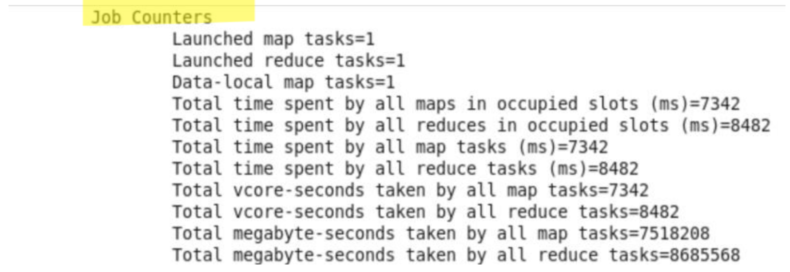
Connecting to **ResourceManager at /0.0.0.0:8032:** Hadoop is connecting to the ResourceManager, which manages resources for distributed applications.
WARN mapreduce.JobResourceUploader: This warns that command-line options for Hadoop weren't parsed correctly. Using the **Tool interface with ToolRunner** can fix this.
Total input paths to process: 1: Only one input file (Processfile.txt) is being processed.
Caught exception **java.lang.InterruptedException**: This error appears twice, indicating that some operations involving the Hadoop Distributed File System (HDFS) client may have been interrupted. However, these errors do not stop the job from completing.



```
24/10/30 00:23:19 INFO mapreduce.Job: Running job: job_1730260138217_0001
24/10/30 00:23:32 INFO mapreduce.Job: Job job_1730260138217_0001 running in uber mode : false
24/10/30 00:23:32 INFO mapreduce.Job:  map 0% reduce 0%
24/10/30 00:23:42 INFO mapreduce.Job:  map 100% reduce 0%
24/10/30 00:23:53 INFO mapreduce.Job:  map 100% reduce 100%
24/10/30 00:23:53 INFO mapreduce.Job: Job job_1730260138217_0001 completed successfully
24/10/30 00:23:53 INFO mapreduce.Job: Counters: 49
        File System Counters
                FILE: Number of bytes read=88
                FILE: Number of bytes written=242479
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=190
                HDFS: Number of bytes written=50
                HDFS: Number of read operations=6
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
```

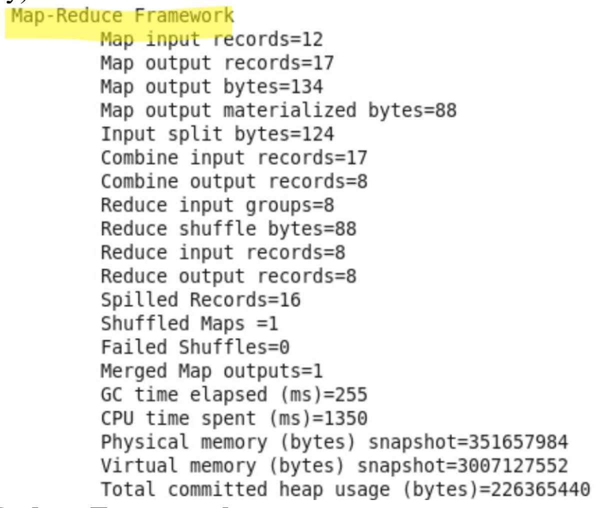
The job (job_1730260138217_0001) is submitted and can be monitored via the URL provided: http://quickstart.cloudera:8088/proxy/application_1730260138217_0001/
The job progresses through MapReduce stages:
- map 0% reduce 0%: The job starts with no progress in either map or reduce tasks.
- map 100% reduce 0%: Mapping is complete.
- map 100% reduce 100%: Both mapping and reducing are complete.

Job completed successfully: The job has run without critical issues.
File System Counters: Details on the number of bytes read/written and file operations, showing both FILE and HDFS I/O statistics.



```
Job Counters
        Launched map tasks=1
        Launched reduce tasks=1
        Data-local map tasks=1
        Total time spent by all maps in occupied slots (ms)=7342
        Total time spent by all reduces in occupied slots (ms)=8482
        Total time spent by all map tasks (ms)=7342
        Total time spent by all reduce tasks (ms)=8482
        Total vcore-seconds taken by all map tasks=7342
        Total vcore-seconds taken by all reduce tasks=8482
        Total megabyte-seconds taken by all map tasks=7518208
        Total megabyte-seconds taken by all reduce tasks=8685568
```

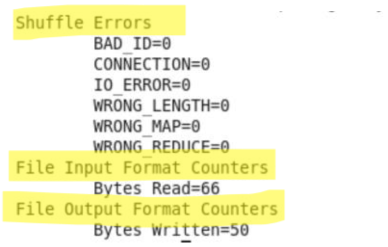
Job Counters: The number of map and reduce tasks launched, the time taken, and resources (CPU, memory) used.



```
Map-Reduce Framework
        Map input records=12
        Map output records=17
        Map output bytes=134
        Map output materialized bytes=88
        Input split bytes=124
        Combine input records=17
        Combine output records=8
        Reduce input groups=8
        Reduce shuffle bytes=88
        Reduce input records=8
        Reduce output records=8
        Spilled Records=16
        Shuffled Maps =1
        Failed Shuffles=0
        Merged Map outputs=1
        GC time elapsed (ms)=255
        CPU time spent (ms)=1350
        Physical memory (bytes) snapshot=351657984
        Virtual memory (bytes) snapshot=3007127552
        Total committed heap usage (bytes)=226365440
```

Map-Reduce Framework:

- **Map and Reduce Input/Output**: Shows the number of input and output records at each stage (map and reduce), useful for understanding data transformations.
- **Spilled Records**: Represents intermediate records that exceeded memory and were written to disk.

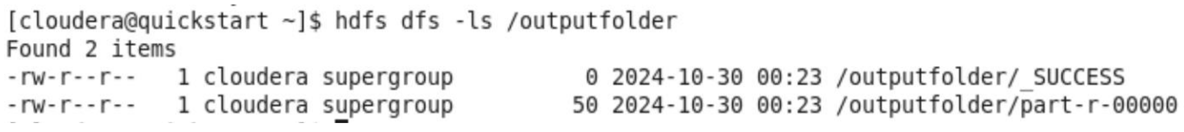


```
Shuffle Errors
        BAD_ID=0
        CONNECTION=0
        IO_ERROR=0
        WRONG_LENGTH=0
        WRONG_MAP=0
        WRONG_REDUCE=0
File Input Format Counters
        Bytes Read=66
File Output Format Counters
        Bytes Written=50
```

Shuffle and Merge Details: The shuffle phase (between map and reduce) had no errors.
Input and Output Format Counters: Bytes read from input and written to output, indicating that the output file is 50 bytes.

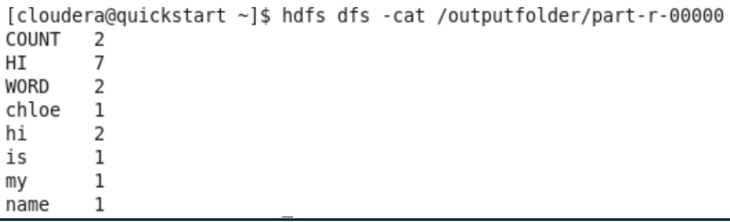
10. Read the content of the messages that are shown by Hadoop during the execution process and try to understand and write your findings in your report

11. After the execuation of the program is finished, check the folder outputfolder inside hdfs to see if there is any output file: hdfs dfs -ls /outputfolder



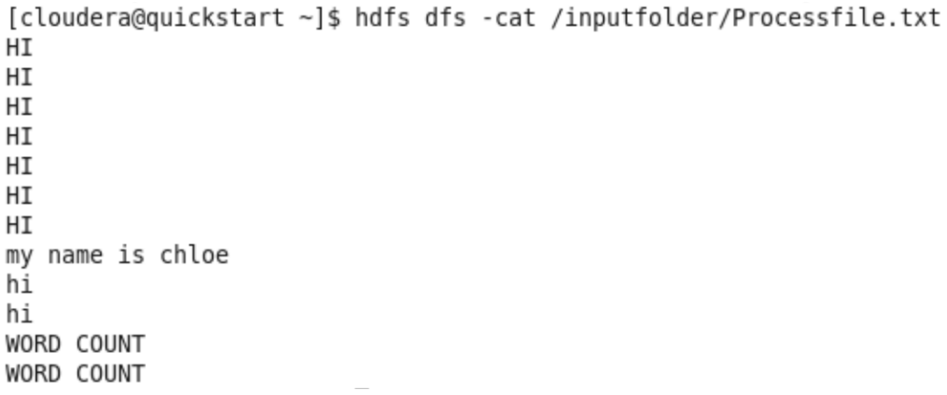
```
[cloudera@quickstart ~]$ hdfs dfs -ls /outputfolder
Found 2 items
-rw-r--r--   1 cloudera supergroup          0 2024-10-30 00:23 /outputfolder/_SUCCESS
-rw-r--r--   1 cloudera supergroup         50 2024-10-30 00:23 /outputfolder/part-r-00000
```

12. Open the file with name similar to "part-r-00000" or any file that is created and the size is greater than Zero



```
[cloudera@quickstart ~]$ hdfs dfs -cat /outputfolder/part-r-00000
COUNT   2
HI      7
WORD    2
chloe   1
hi      2
is      1
my      1
name    1
```

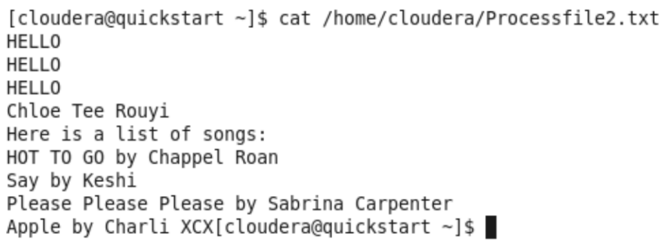
13. Check the word counts and cross check with the content of the Processfile.txt to validate counting is correct.



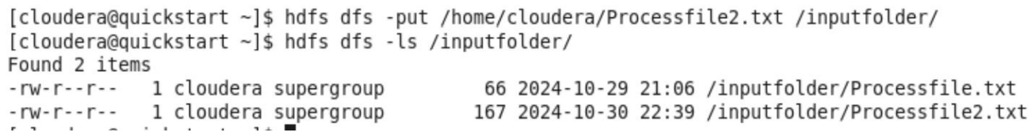
```
[cloudera@quickstart ~]$ hdfs dfs -cat /inputfolder/Processfile.txt
HI
HI
HI
HI
HI
HI
HI
my name is chloe
hi
hi
WORD COUNT
WORD COUNT
```

Manually counting the words in Processfile.txt:
HI= 7
hi= 2
my= 1
name= 1
is= 1
chloe= 1
WORD= 2
COUNT= 2

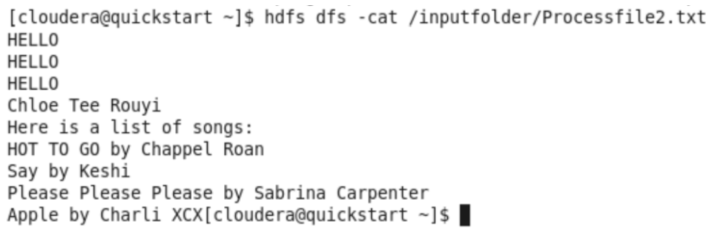
14. Run the same wordcount program with another text file. select a text file of your choice. But it is recommended to do not use a heavy file for this practice to prevent heavy process inside your VM that takes longer time to be completed.



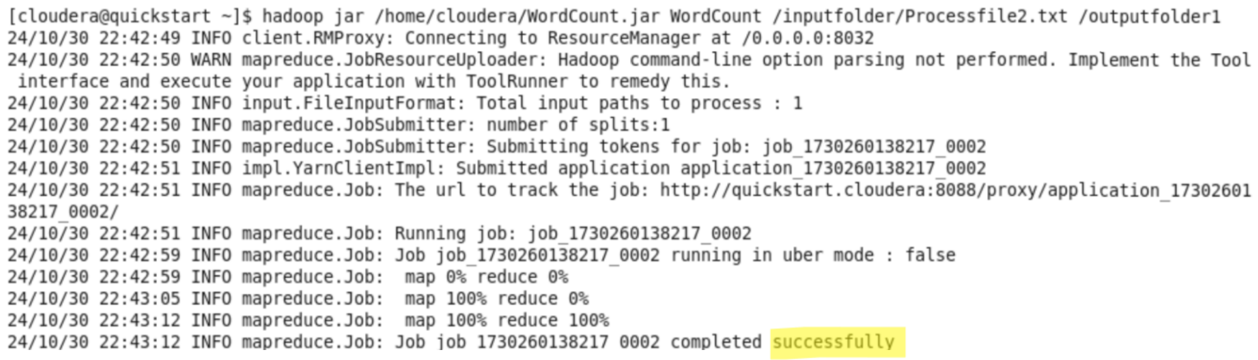
```
[cloudera@quickstart ~]$ cat /home/cloudera/Processfile2.txt
HELLO
HELLO
HELLO
Chloe Tee Rouyi
Here is a list of songs:
HOT TO GO by Chappel Roan
Say by Keshi
Please Please Please by Sabrina Carpenter
Apple by Charli XCX[cloudera@quickstart ~]$ █
```



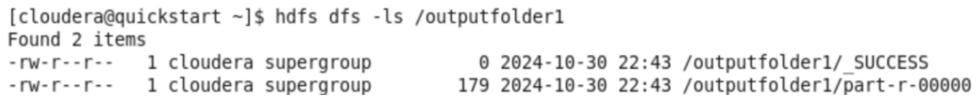
```
[cloudera@quickstart ~]$ hdfs dfs -put /home/cloudera/Processfile2.txt /inputfolder/
[cloudera@quickstart ~]$ hdfs dfs -ls /inputfolder/
Found 2 items
-rw-r--r--   1 cloudera supergroup         66 2024-10-29 21:06 /inputfolder/Processfile.txt
-rw-r--r--   1 cloudera supergroup        167 2024-10-30 22:39 /inputfolder/Processfile2.txt
```



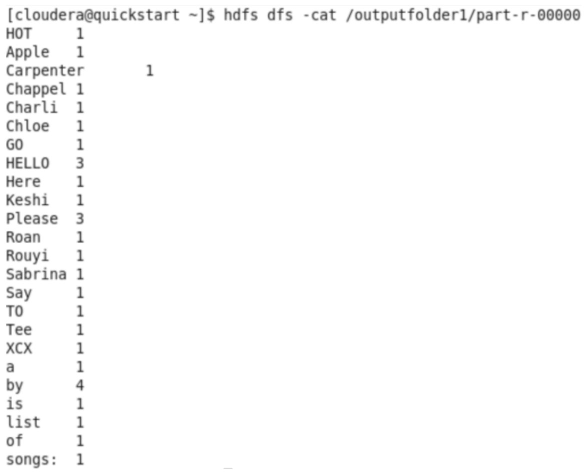
```
[cloudera@quickstart ~]$ hdfs dfs -cat /inputfolder/Processfile2.txt
HELLO
HELLO
HELLO
Chloe Tee Rouyi
Here is a list of songs:
HOT TO GO by Chappel Roan
Say by Keshi
Please Please Please by Sabrina Carpenter
Apple by Charli XCX[cloudera@quickstart ~]$ █
```



```
[cloudera@quickstart ~]$ hadoop jar /home/cloudera/WordCount.jar WordCount /inputfolder/Processfile2.txt /outputfolder1
24/10/30 22:42:49 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
24/10/30 22:42:50 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool
 interface and execute your application with ToolRunner to remedy this.
24/10/30 22:42:50 INFO input.FileInputFormat: Total input paths to process : 1
24/10/30 22:42:50 INFO mapreduce.JobSubmitter: number of splits:1
24/10/30 22:42:50 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1730260138217_0002
24/10/30 22:42:51 INFO impl.YarnClientImpl: Submitted application application_1730260138217_0002
24/10/30 22:42:51 INFO mapreduce.Job: The url to track the job: http://quickstart.cloudera:8088/proxy/application_17302601
38217_0002/
24/10/30 22:42:51 INFO mapreduce.Job: Running job: job_1730260138217_0002
24/10/30 22:42:59 INFO mapreduce.Job: Job job_1730260138217_0002 running in uber mode : false
24/10/30 22:42:59 INFO mapreduce.Job:  map 0% reduce 0%
24/10/30 22:43:05 INFO mapreduce.Job:  map 100% reduce 0%
24/10/30 22:43:12 INFO mapreduce.Job:  map 100% reduce 100%
24/10/30 22:43:12 INFO mapreduce.Job: Job job_1730260138217_0002 completed successfully
```



```
[cloudera@quickstart ~]$ hdfs dfs -ls /outputfolder1
Found 2 items
-rw-r--r--   1 cloudera supergroup          0 2024-10-30 22:43 /outputfolder1/_SUCCESS
-rw-r--r--   1 cloudera supergroup        179 2024-10-30 22:43 /outputfolder1/part-r-00000
```



```
[cloudera@quickstart ~]$ hdfs dfs -cat /outputfolder1/part-r-00000
HOT     1
Apple   1
Carpenter       1
Chappel 1
Charli  1
Chloe   1
GO      1
HELLO   3
Here    1
Keshi   1
Please  3
Roan    1
Rouyi   1
Sabrina 1
Say     1
TO      1
Tee     1
XCX     1
a       1
by      4
is      1
list    1
of      1
songs:  1
```

15. Check the source code used in task 2.m and write your understanding of the code in your report.

This WordCount program is a MapReduce application in Java, designed to run on Hadoop to count the occurrences of each word in a text file.

TokenizerMapper Class: This mapper class processes each line of input by splitting it into words, then outputs each word with a count of 1.

IntSumReducer Class: This reducer class aggregates the counts of each word by summing them, outputting the word with its total occurrence count.

Main Method: This sets up the job configuration, including the mapper, combiner, and reducer classes, along with the input and output paths, then runs the job and exits based on its success.

16. Save and submit your Lab documents in PDF with the following filename format.
Submit your report via the submission link for this available on MyTIMeS.

Filename format: Name_ID_Lab05