MapReduce for word count problem on Hadoop:

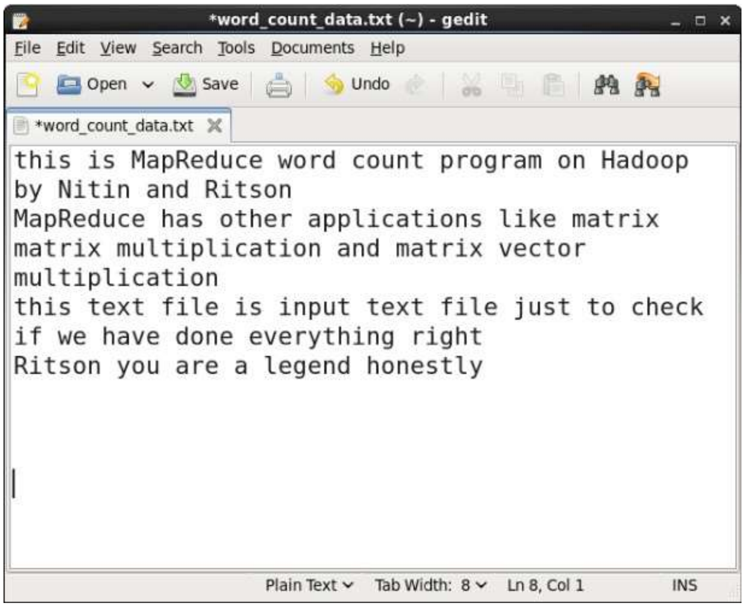
In MapReduce word count example, we find out the frequency of each word. Here, the role of Mapper is to map the keys to the existing values and the role of Reducer is to aggregate the keys of common values. So, everything is represented in the form of key-value pair.

Example:

Let’s solve a word count problem using MapReduce on Hadoop.

Step 1: Open Cloudera Quickstart VM.

Step 2: Create a .txt data file inside **`/home/cloudera`** directory that will be passed as an input to MapReduce program. For simplicity purpose, we name it as **`word_count_data.txt.`**


*word_count_data.txt (~) - gedit
File Edit View Search Tools Documents Help
Open
Save
Undo
*word_count_data.txt
Plain Text
Tab Width: 8
Ln 8, Col 1
INS

```
this is MapReduce word count program on Hadoop
by Nitin and Ritson
MapReduce has other applications like matrix
matrix multiplication and matrix vector
multiplication
this text file is input text file just to check
if we have done everything right
Ritson you are a legend honestly
```

Step 3: Create **`mapper.py`** and **`reducer.py`** files inside **`/home/cloudera`** directory.

<u>mapper.py</u>

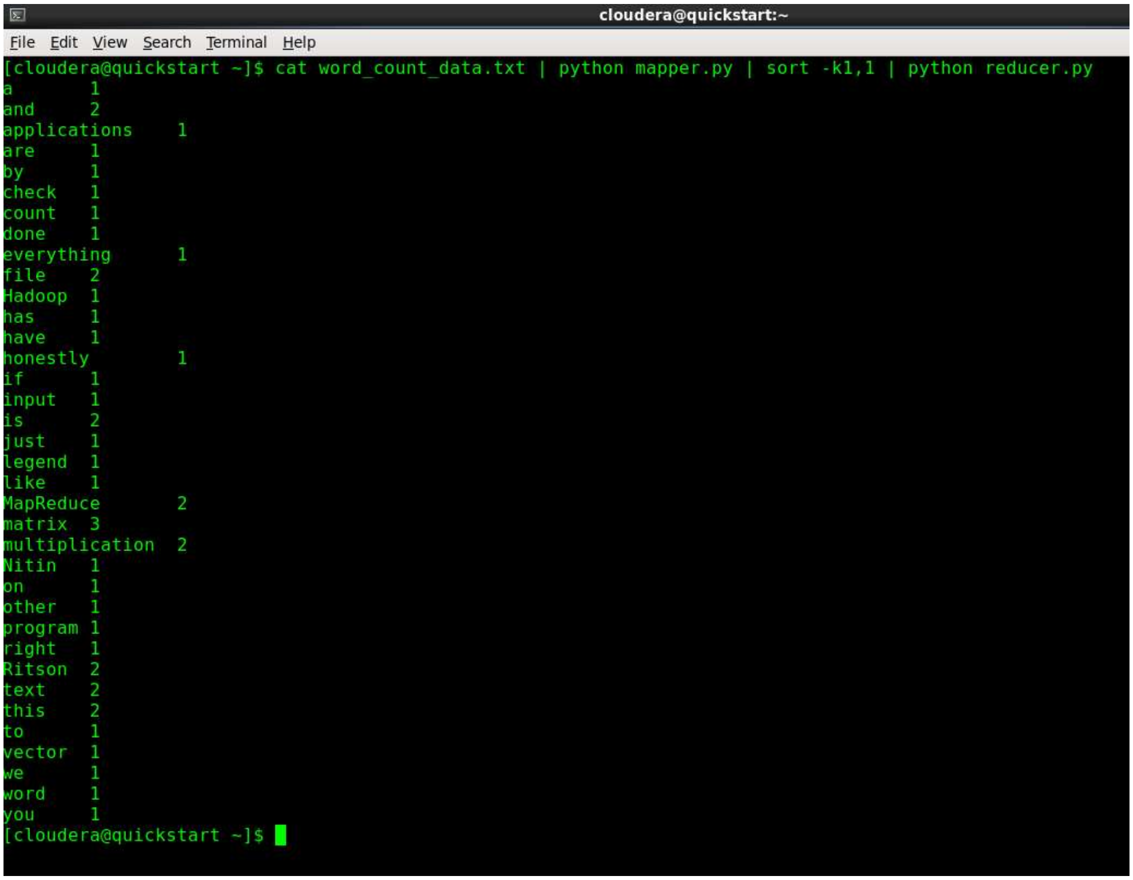
```
#!/usr/bin/python
# import sys because we need to read and write data to STDIN and
STDOUT

import sys
# reading entire line from STDIN (standard input)

for line in sys.stdin:
     # to remove leading and trailing whitespace
     line = line.strip()

     # split the line into words
     words = line.split()

    # we are looping over the words array and printing the word
    # with the count of 1 to the STDOUT

    for word in words:

          # write the results to STDOUT (standard output);
          # what we output here will be the input for the
          # Reduce step, i.e., the input for reducer.py
          print(“%s\t%s” % (word, 1))
```

<u>reducer.py</u>

```
#!/usr/bin/python

from operator import itemgetter
import sys

current_word = None
current_count = 0
word = None

# read the entire line from STDIN
for line in sys.stdin:
     # remove leading and trailing whitespace
     line = line.strip()

     # spliting the data on the basis of tab we have provided in mapper.py
     word, count = line.split('\t', 1)

     # convert count (currently a string) to int
     try:
          count = int(count)
     except ValueError:
          # count was not a number, so silently
          # ignore/discard this line
          continue

     # this if-switch only works because Hadoop sorts map output
     # by key (here: word) before it is passed to the reducer
     if current_word == word:
          current_count += count

     else:
          if current_word:

               # write result to STDOUT
               print("%s\t%s" % (current_word, current_count))

          current_count = count
          current_word = word


# do not forget to output the last word if needed

if current_word == word:
     print("%s\t%s" % (current_word, current_count))
```

Step 4: Test the MapReduce program(s) locally to check if everything works properly before running on Hadoop.

cat word_count_data.txt | python mapper.py | sort -k1,1 | python reducer.py



```
[cloudera@quickstart ~]$ cat word_count_data.txt | python mapper.py | sort -k1,1 | python reducer.py
a       1
and     2
applications    1
are     1
by      1
check   1
count   1
done    1
everything      1
file    2
Hadoop  1
has     1
have    1
honestly        1
if      1
input   1
is      2
just    1
legend  1
like    1
MapReduce       2
matrix  3
multiplication  2
Nitin   1
on      1
other   1
program 1
right   1
Ritson  2
text    2
this    2
to      1
vector  1
we      1
word    1
you     1
[cloudera@quickstart ~]$ 
```

For the above example, the output obtained is exactly the same as expected.

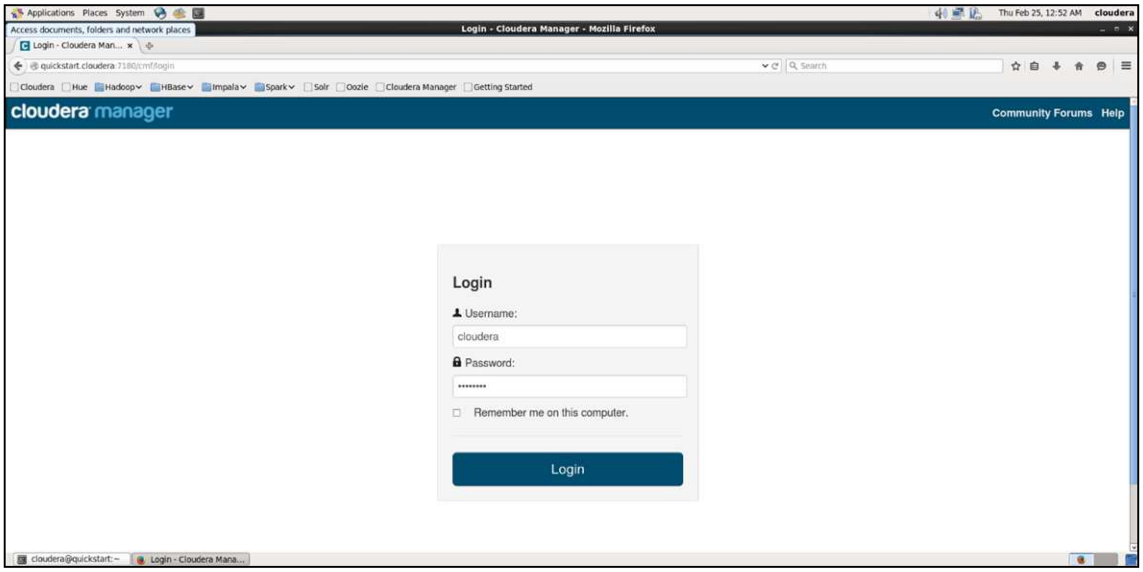
If you see all the words correctly mapped, sorted and reduced to their respective counts, then your program is good to be tested on Hadoop.

Step 5: Configure Hadoop services and settings.

Now, we need to configure certain settings on Hadoop before we run the MapReduce program for word count.

5a: Login to Cloudera Quickstart.

Open browser on Cloudera Quickstart VM and open quickstart.cloudera:7180/cmf/login. Login by entering the credentials as cloudera for both, username and password.


Applications Places System
Thu Feb 25, 12:52 AM cloudera
Login - Cloudera Manager - Mozilla Firefox
Login - Cloudera Man...
quickstart.cloudera:7180/cmf/login
Search
Cloudera Hue Hadoop HBase Impala Spark Solr Oozie Cloudera Manager Getting Started
cloudera manager
Community Forums Help
Login
cloudera@quickstart:~

Login

Username:

cloudera

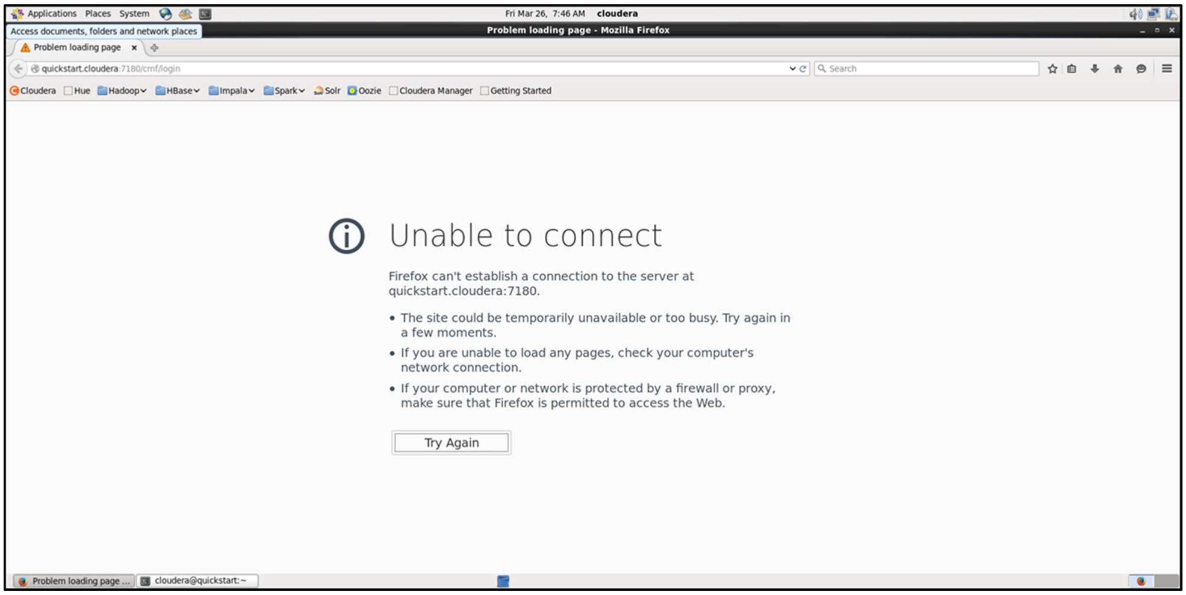
Password:

••••••••

□ Remember me on this computer.

Login

Note: If you see the error "`Unable to connect`" while logging in to `quickstart.cloudera:7180/cmf/login`, try restarting the CDH services.


Applications Places System
Fri Mar 26, 7:46 AM cloudera
Problem loading page - Mozilla Firefox
Problem loading page
quickstart.cloudera:7180/cmf/login
Search
Cloudera Hue Hadoop HBase Impala Spark Solr Oozie Cloudera Manager Getting Started
Problem loading page ...
cloudera@quickstart:~

Unable to connect

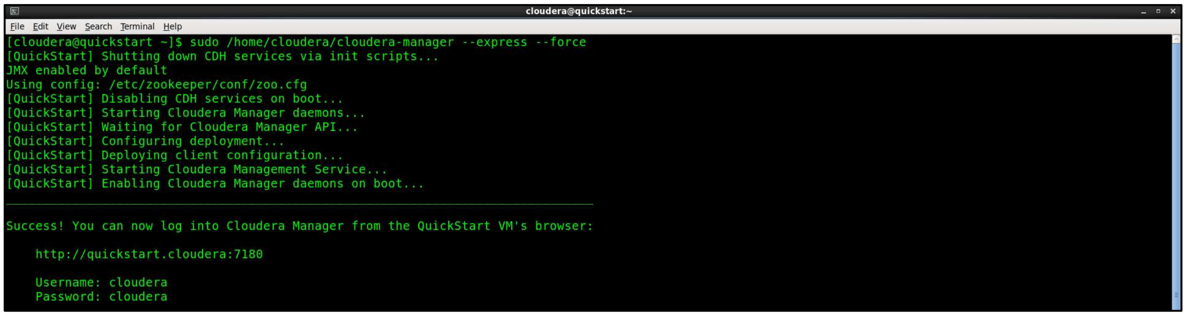
Firefox can't establish a connection to the server at quickstart.cloudera:7180.

- The site could be temporarily unavailable or too busy. Try again in a few moments.
- If you are unable to load any pages, check your computer's network connection.
- If your computer or network is protected by a firewall or proxy, make sure that Firefox is permitted to access the Web.

Try Again

Restart CDH services by typing the following command:

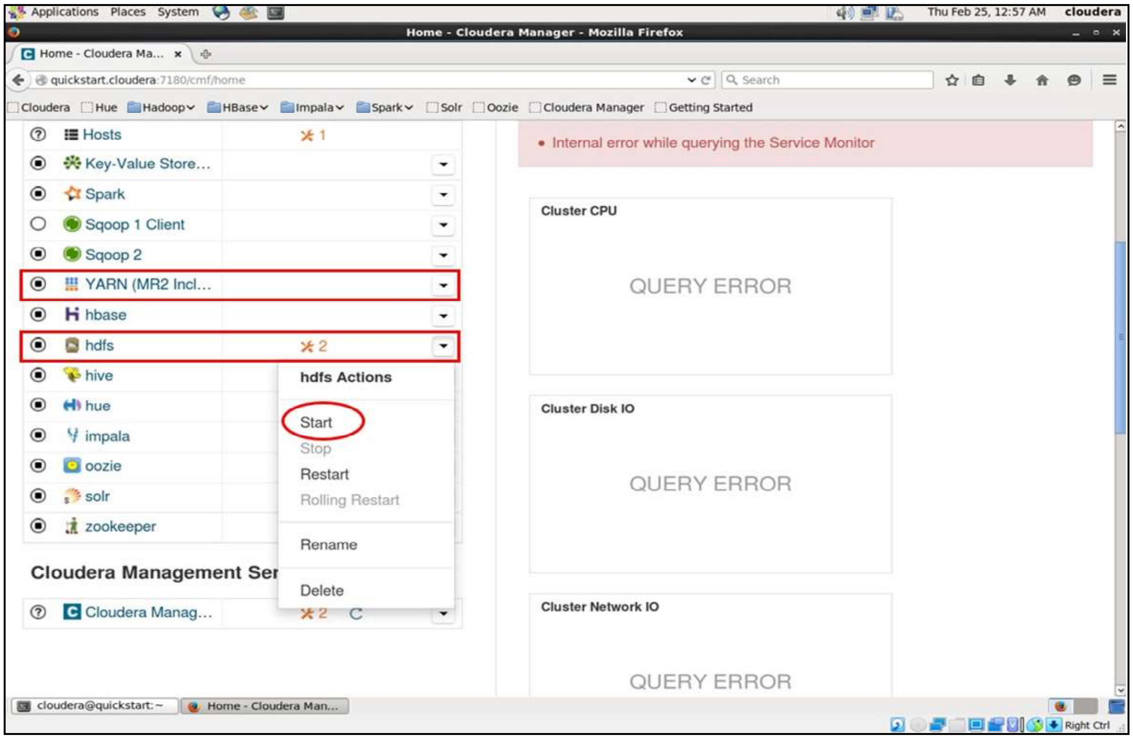
```
sudo /home/cloudera/cloudera-manager --express --force
```



```
[cloudera@quickstart ~]$ sudo /home/cloudera/cloudera-manager --express --force
[QuickStart] Shutting down CDH services via init scripts...
JMX enabled by default
Using config: /etc/zookeeper/conf/zoo.cfg
[QuickStart] Disabling CDH services on boot...
[QuickStart] Starting Cloudera Manager daemons...
[QuickStart] Waiting for Cloudera Manager API...
[QuickStart] Configuring deployment...
[QuickStart] Deploying client configuration...
[QuickStart] Starting Cloudera Management Service...
[QuickStart] Enabling Cloudera Manager daemons on boot...
________________________________________________________________________________

Success! You can now log into Cloudera Manager from the QuickStart VM's browser:

    http://quickstart.cloudera:7180

    Username: cloudera
    Password: cloudera
```

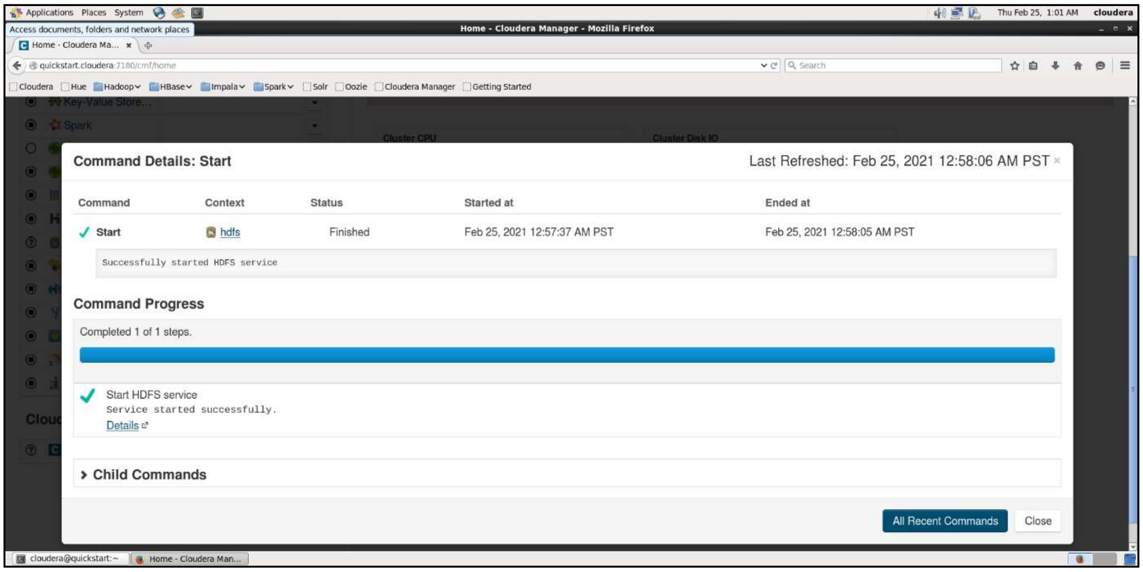
5b: Start HDFS and YARN services.

Click the dropdown arrow and choose Start option for HDFS and YARN services.



- Internal error while querying the Service Monitor

You'll see the following if both; HDFS and YARN services are started successfully.



Command Details: Start

Last Refreshed: Feb 25, 2021 12:58:06 AM PST

Command	Context	Status	Started at	Ended at
✓ Start	hdfs	Finished	Feb 25, 2021 12:57:37 AM PST	Feb 25, 2021 12:58:05 AM PST

```
Successfully started HDFS service
```

Command Progress

Completed 1 of 1 steps.

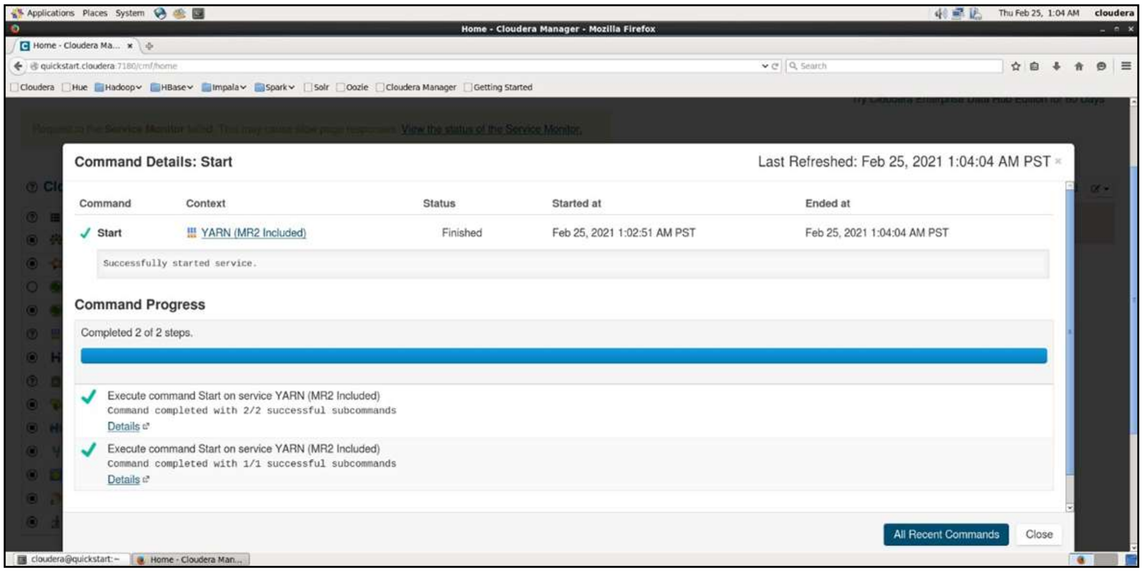
✓ Start HDFS service

```
Service started successfully.
```

Details

› Child Commands

HDFS service started successfully.



Command Details: Start — Last Refreshed: Feb 25, 2021 1:04:04 AM PST

Command	Context	Status	Started at	Ended at
✓ Start	YARN (MR2 Included)	Finished	Feb 25, 2021 1:02:51 AM PST	Feb 25, 2021 1:04:04 AM PST

```
Successfully started service.
```

Command Progress

Completed 2 of 2 steps.

✓ Execute command Start on service YARN (MR2 Included)
Command completed with 2/2 successful subcommands
Details

✓ Execute command Start on service YARN (MR2 Included)
Command completed with 1/1 successful subcommands
Details

Applications Places System
Thu Feb 25, 1:04 AM cloudera
Home - Cloudera Manager - Mozilla Firefox
Home - Cloudera Ma...
quickstart.cloudera:7180/cmf/home
Search
Cloudera Hue Hadoop HBase Impala Spark Solr Oozie Cloudera Manager Getting Started
All Recent Commands
Close
cloudera@quickstart:~
Home - Cloudera Man...

YARN service started successfully.

Step 6: Create a directory on HDFS

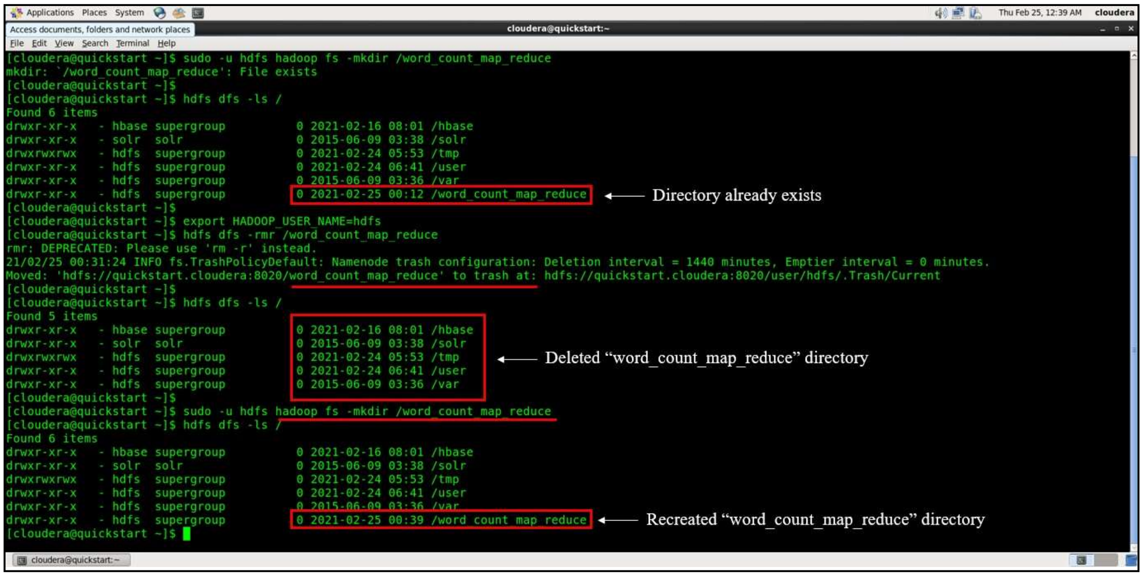
Now, we create a directory named `word_count_map_reduce` on HDFS where our input data and its resulting output would be stored. Use the following command for it.

```
sudo -u hdfs hadoop fs -mkdir /word_count_map_reduce
```

> **Note:** If the directory already exists, then either create a directory with new name or delete the existing directory using the following command.
>
> ```
> export HADOOP_USER_NAME=hdfs
> hdfs dfs -rmr /word_count_map_reduce
> ```

List HDFS directory items using the following command.

```
hdfs dfs -ls /
```



```
[cloudera@quickstart ~]$ sudo -u hdfs hadoop fs -mkdir /word_count_map_reduce
mkdir: `/word_count_map_reduce': File exists
[cloudera@quickstart ~]$
[cloudera@quickstart ~]$ hdfs dfs -ls /
Found 6 items
drwxr-xr-x   - hbase supergroup          0 2021-02-16 08:01 /hbase
drwxr-xr-x   - solr  solr                0 2015-06-09 03:38 /solr
drwxrwxrwx   - hdfs  supergroup          0 2021-02-24 05:53 /tmp
drwxr-xr-x   - hdfs  supergroup          0 2021-02-24 06:41 /user
drwxr-xr-x   - hdfs  supergroup          0 2015-06-09 03:36 /var
drwxr-xr-x   - hdfs  supergroup          0 2021-02-25 00:12 /word_count_map_reduce
[cloudera@quickstart ~]$
[cloudera@quickstart ~]$ export HADOOP_USER_NAME=hdfs
[cloudera@quickstart ~]$ hdfs dfs -rmr /word_count_map_reduce
rmr: DEPRECATED: Please use 'rm -r' instead.
21/02/25 00:31:24 INFO fs.TrashPolicyDefault: Namenode trash configuration: Deletion interval = 1440 minutes, Emptier interval = 0 minutes.
Moved: 'hdfs://quickstart.cloudera:8020/word_count_map_reduce' to trash at: hdfs://quickstart.cloudera:8020/user/hdfs/.Trash/Current
[cloudera@quickstart ~]$
[cloudera@quickstart ~]$ hdfs dfs -ls /
Found 5 items
drwxr-xr-x   - hbase supergroup          0 2021-02-16 08:01 /hbase
drwxr-xr-x   - solr  solr                0 2015-06-09 03:38 /solr
drwxrwxrwx   - hdfs  supergroup          0 2021-02-24 05:53 /tmp
drwxr-xr-x   - hdfs  supergroup          0 2021-02-24 06:41 /user
drwxr-xr-x   - hdfs  supergroup          0 2015-06-09 03:36 /var
[cloudera@quickstart ~]$
[cloudera@quickstart ~]$ sudo -u hdfs hadoop fs -mkdir /word_count_map_reduce
[cloudera@quickstart ~]$ hdfs dfs -ls /
Found 6 items
drwxr-xr-x   - hbase supergroup          0 2021-02-16 08:01 /hbase
drwxr-xr-x   - solr  solr                0 2015-06-09 03:38 /solr
drwxrwxrwx   - hdfs  supergroup          0 2021-02-24 05:53 /tmp
drwxr-xr-x   - hdfs  supergroup          0 2021-02-24 06:41 /user
drwxr-xr-x   - hdfs  supergroup          0 2015-06-09 03:36 /var
drwxr-xr-x   - hdfs  supergroup          0 2021-02-25 00:39 /word_count_map_reduce
[cloudera@quickstart ~]$
```

Directory already exists

Deleted “word_count_map_reduce” directory

Recreated “word_count_map_reduce” directory

Applications Places System
Thu Feb 25, 12:39 AM cloudera
Access documents, folders and network places
cloudera@quickstart:~
File Edit View Search Terminal Help
cloudera@quickstart:~

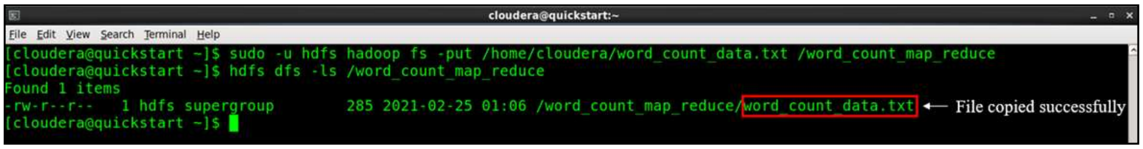
Step 7: Move input data file to HDFS.

Copy the **`word_count_data.txt`** file to **`word_count_map_reduce`** directory on HDFS using the following command.

```
sudo -u hdfs hadoop fs -put /home/cloudera/word_count_data.txt /word_count_map_reduce
```

Check if file was copied successfully to the desired location.

```
hdfs dfs -ls /word_count_map_reduce
```

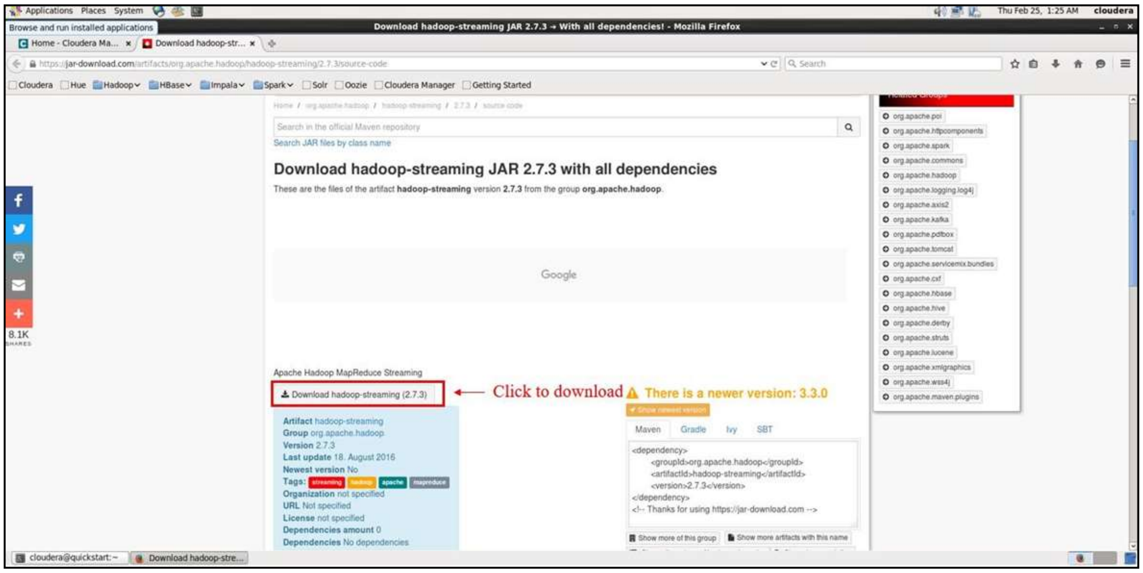


```
[cloudera@quickstart ~]$ sudo -u hdfs hadoop fs -put /home/cloudera/word_count_data.txt /word_count_map_reduce
[cloudera@quickstart ~]$ hdfs dfs -ls /word_count_map_reduce
Found 1 items
-rw-r--r--   1 hdfs supergroup        285 2021-02-25 01:06 /word_count_map_reduce/word_count_data.txt
[cloudera@quickstart ~]$ 
```

File copied successfully

Step 8: Download hadoop-streaming JAR 2.7.3.

Open browser and go to https://jar-download.com/artifacts/org.apache.hadoop/hadoop-streaming/2.7.3/source-code and download hadoop-streaming JAR 2.7.3 file.



Download hadoop-streaming JAR 2.7.3 with all dependencies

These are the files of the artifact **hadoop-streaming** version **2.7.3** from the group **org.apache.hadoop**.

Apache Hadoop MapReduce Streaming

Download hadoop-streaming (2.7.3) ← Click to download

⚠ There is a newer version: 3.3.0

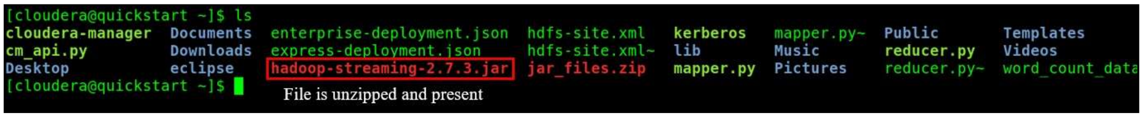
Artifact hadoop-streaming
Group org.apache.hadoop
Version 2.7.3
Last update 18. August 2016
Newest version No
Tags: streaming hadoop apache mapreduce
Organization not specified
URL Not specified
License not specified
Dependencies amount 0
Dependencies No dependencies

```
<dependency>
    <groupId>org.apache.hadoop</groupId>
    <artifactId>hadoop-streaming</artifactId>
    <version>2.7.3</version>
</dependency>
<!-- Thanks for using https://jar-download.com -->
```

Once the file is downloaded, unzip it inside **`/home/cloudera`** directory.

Double-check if the JAR file was unzipped successfully and is present inside **`/home/cloudera`** directory .

```
ls
```



```
[cloudera@quickstart ~]$ ls
cloudera-manager  Documents  enterprise-deployment.json  hdfs-site.xml   kerberos  mapper.py~  Public      Templates
cm_api.py         Downloads  express-deployment.json     hdfs-site.xml~  lib       Music       reducer.py  Videos
Desktop           eclipse    hadoop-streaming-2.7.3.jar  jar_files.zip   mapper.py Pictures    reducer.py~ word_count_data
[cloudera@quickstart ~]$ 
```

File is unzipped and present

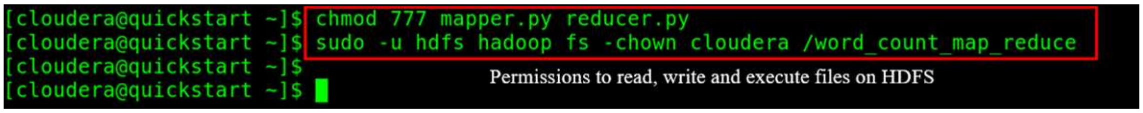
Step 9: Configure permissions to run MapReduce on Hadoop.

We're almost ready to run our MapReduce job on Hadoop but before that, we need to give permission to read, write and execute the Mapper and Reducer programs on Hadoop.

We also need to provide permission for the default user (cloudera) to write the output file inside HDFS.

Run the following commands to do so:

chmod 777 mapper.py reducer.py
sudo -u hdfs hadoop fs -chown cloudera /word_count_map_reduce



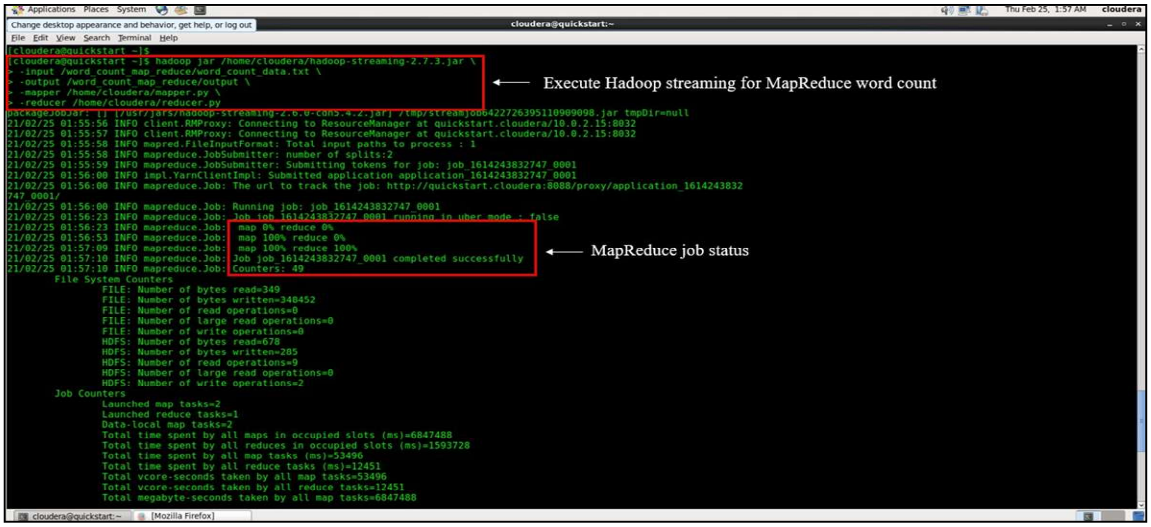
```
[cloudera@quickstart ~]$ chmod 777 mapper.py reducer.py
[cloudera@quickstart ~]$ sudo -u hdfs hadoop fs -chown cloudera /word_count_map_reduce
[cloudera@quickstart ~]$
[cloudera@quickstart ~]$ █
```

Permissions to read, write and execute files on HDFS

Step 10: Run MapReduce on Hadoop.

We’re at the ultimate step of this program. Run the MapReduce job on Hadoop using the following command.

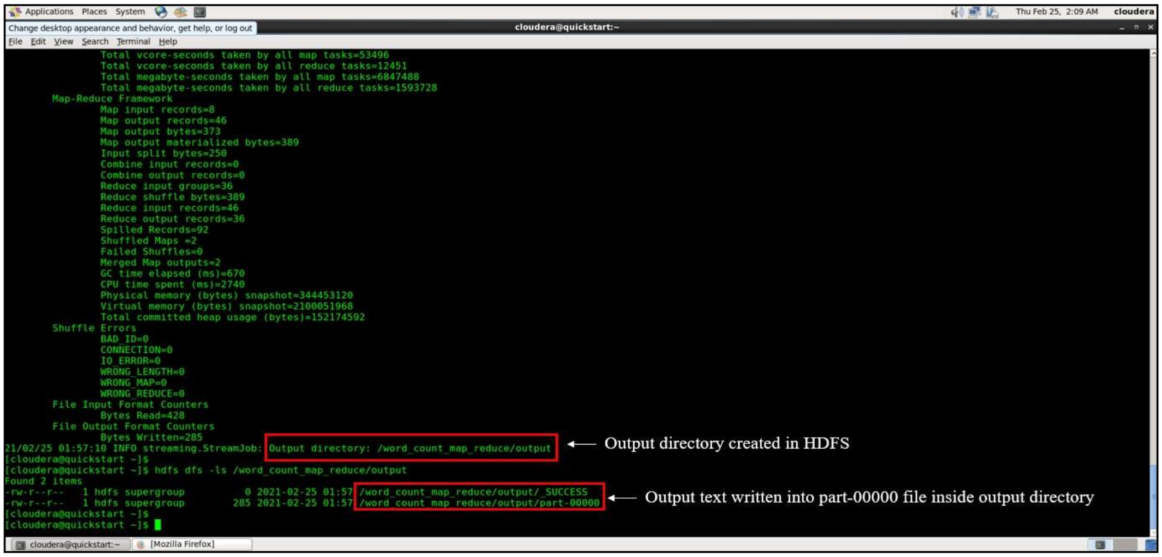
hadoop jar /home/cloudera/hadoop-streaming-2.7.3.jar \
\> -input /word_count_map_reduce/word_count_data.txt \
\> -output /word_count_map_reduce/output \
\> -mapper /home/cloudera/mapper.py \
\> - reducer /home/cloudera/reducer.py


Applications Places System
Thu Feb 25, 1:57 AM
cloudera
Change desktop appearance and behavior, get help, or log out
cloudera@quickstart:~
File Edit View Search Terminal Help
cloudera@quickstart:~
[Mozilla Firefox]

```
[cloudera@quickstart ~]$
[cloudera@quickstart ~]$ hadoop jar /home/cloudera/hadoop-streaming-2.7.3.jar \
> -input /word_count_map_reduce/word_count_data.txt \
> -output /word_count_map_reduce/output \
> -mapper /home/cloudera/mapper.py \
> -reducer /home/cloudera/reducer.py
packageJobJar: [] [/usr/jars/hadoop-streaming-2.6.0-cdh5.4.2.jar] /tmp/streamjob6422726395110909098.jar tmpDir=null
21/02/25 01:55:56 INFO client.RMProxy: Connecting to ResourceManager at quickstart.cloudera/10.0.2.15:8032
21/02/25 01:55:57 INFO client.RMProxy: Connecting to ResourceManager at quickstart.cloudera/10.0.2.15:8032
21/02/25 01:55:58 INFO mapred.FileInputFormat: Total input paths to process : 1
21/02/25 01:55:58 INFO mapreduce.JobSubmitter: number of splits:2
21/02/25 01:55:59 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1614243832747_0001
21/02/25 01:56:00 INFO impl.YarnClientImpl: Submitted application application_1614243832747_0001
21/02/25 01:56:00 INFO mapreduce.Job: The url to track the job: http://quickstart.cloudera:8088/proxy/application_1614243832
747_0001/
21/02/25 01:56:00 INFO mapreduce.Job: Running job: job_1614243832747_0001
21/02/25 01:56:23 INFO mapreduce.Job: Job job_1614243832747_0001 running in uber mode : false
21/02/25 01:56:23 INFO mapreduce.Job:  map 0% reduce 0%
21/02/25 01:56:53 INFO mapreduce.Job:  map 100% reduce 0%
21/02/25 01:57:09 INFO mapreduce.Job:  map 100% reduce 100%
21/02/25 01:57:10 INFO mapreduce.Job: Job job_1614243832747_0001 completed successfully
21/02/25 01:57:10 INFO mapreduce.Job: Counters: 49
        File System Counters
                FILE: Number of bytes read=349
                FILE: Number of bytes written=348452
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=678
                HDFS: Number of bytes written=285
                HDFS: Number of read operations=9
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
        Job Counters
                Launched map tasks=2
                Launched reduce tasks=1
                Data-local map tasks=2
                Total time spent by all maps in occupied slots (ms)=6847488
                Total time spent by all reduces in occupied slots (ms)=1593728
                Total time spent by all map tasks (ms)=53496
                Total time spent by all reduce tasks (ms)=12451
                Total vcore-seconds taken by all map tasks=53496
                Total vcore-seconds taken by all reduce tasks=12451
                Total megabyte-seconds taken by all map tasks=6847488
```

Execute Hadoop streaming for MapReduce word count

MapReduce job status


Applications Places System
Thu Feb 25, 2:09 AM
cloudera
Change desktop appearance and behavior, get help, or log out
cloudera@quickstart:~
File Edit View Search Terminal Help
cloudera@quickstart:~
[Mozilla Firefox]

```
                Total vcore-seconds taken by all map tasks=53496
                Total vcore-seconds taken by all reduce tasks=12451
                Total megabyte-seconds taken by all map tasks=6847488
                Total megabyte-seconds taken by all reduce tasks=1593728
        Map-Reduce Framework
                Map input records=8
                Map output records=46
                Map output bytes=373
                Map output materialized bytes=389
                Input split bytes=250
                Combine input records=0
                Combine output records=0
                Reduce input groups=36
                Reduce shuffle bytes=389
                Reduce input records=46
                Reduce output records=36
                Spilled Records=92
                Shuffled Maps =2
                Failed Shuffles=0
                Merged Map outputs=2
                GC time elapsed (ms)=670
                CPU time spent (ms)=2740
                Physical memory (bytes) snapshot=344453120
                Virtual memory (bytes) snapshot=2100051968
                Total committed heap usage (bytes)=152174592
        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=428
        File Output Format Counters
                Bytes Written=285
21/02/25 01:57:10 INFO streaming.StreamJob: Output directory: /word_count_map_reduce/output
[cloudera@quickstart ~]$
[cloudera@quickstart ~]$ hdfs dfs -ls /word_count_map_reduce/output
Found 2 items
-rw-r--r--   1 hdfs supergroup          0 2021-02-25 01:57 /word_count_map_reduce/output/_SUCCESS
-rw-r--r--   1 hdfs supergroup        285 2021-02-25 01:57 /word_count_map_reduce/output/part-00000
[cloudera@quickstart ~]$
[cloudera@quickstart ~]$ █
```

Output directory created in HDFS

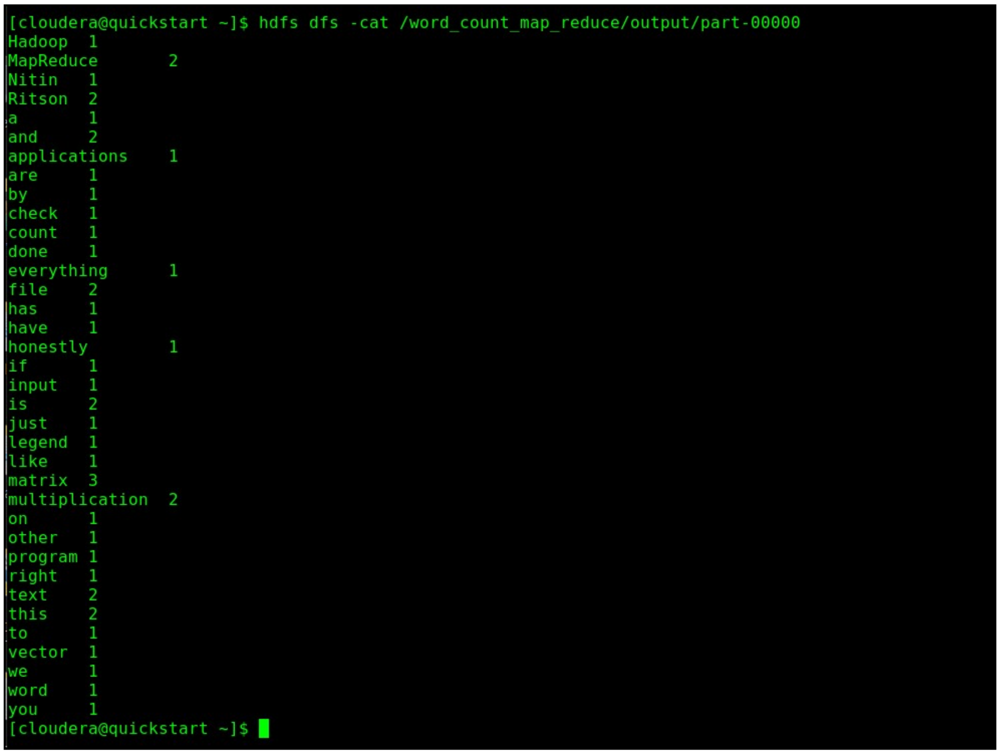
Output text written into part-00000 file inside output directory

If you see the output on terminal as shown in above two images, then the MapReduce job was executed successfully.

Step 11: Read the MapReduce output.

Now, finally run the following command to read the output of MapReduce for word count of the input data file you had created.

hdfs dfs -cat /word_count_map_reduce/output/part-00000



```
[cloudera@quickstart ~]$ hdfs dfs -cat /word_count_map_reduce/output/part-00000
Hadoop  1
MapReduce       2
Nitin   1
Ritson  2
a       1
and     2
applications    1
are     1
by      1
check   1
count   1
done    1
everything      1
file    2
has     1
have    1
honestly        1
if      1
input   1
is      2
just    1
legend  1
like    1
matrix  3
multiplication  2
on      1
other   1
program 1
right   1
text    2
this    2
to      1
vector  1
we      1
word    1
you     1
[cloudera@quickstart ~]$ 
```

Congratulations, the output for MapReduce on Hadoop is obtained exactly as expected. All the words in the input data file have been mapped, sorted and reduced to their respective counts.