

F Big Data A

LAB-11-A

Timings: 11:30 am - 2:30 pm

Lab Protocols:

1. This Lab Would hold tasks at the end. Cheating would result straight 0
2. Making noise in lab during demonstration would result in immediate termination of session and start of Tasks.
3. Contact me on email for queries m.ali@nu.edu.pk

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.

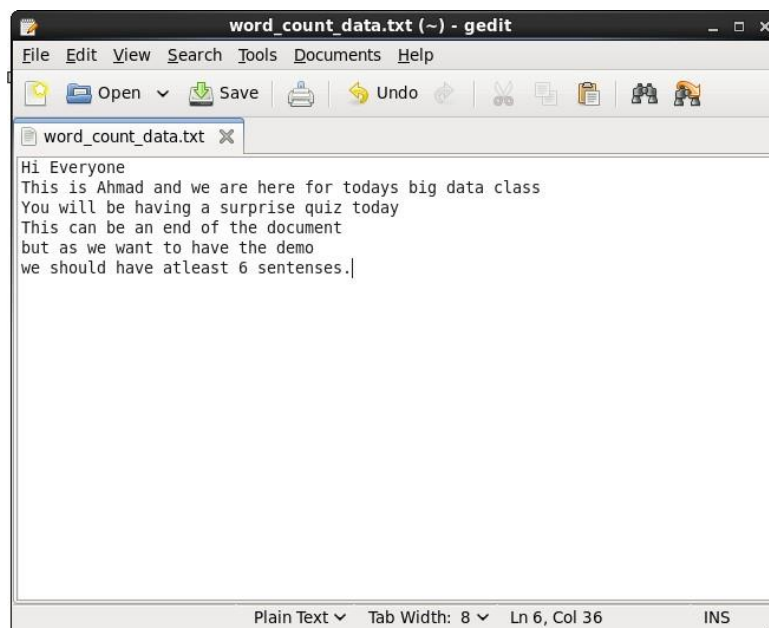
Example:

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

Step 1: Open Cloudera Quickstart VM.

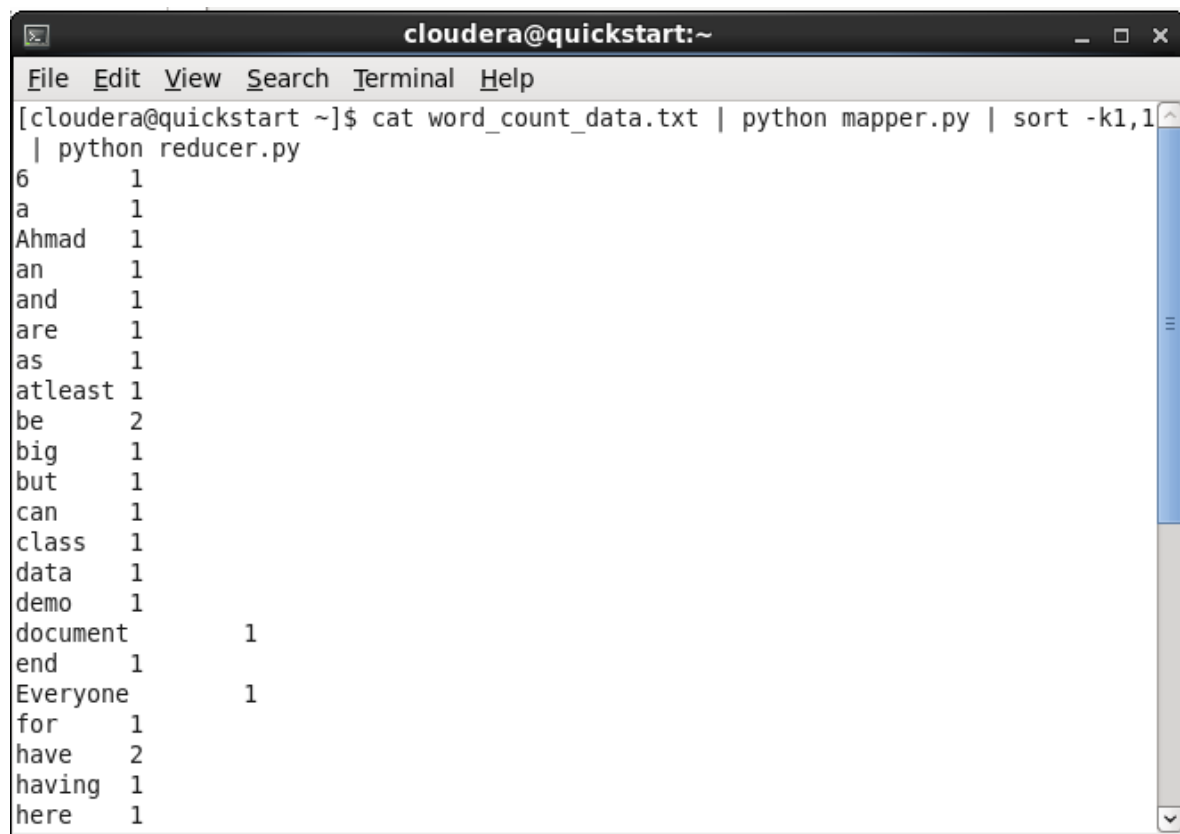


Step 2: Create a .txt data file inside /home/cloudera directory that will be passed as an input to MapReduce program. For simplicity purpose, we name it as word_count_data.txt.



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

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`

A terminal window titled 'cloudera@quickstart:~' with a menu bar (File, Edit, View, Search, Terminal, Help). The command executed is '[cloudera@quickstart ~]\$ cat word_count_data.txt | python mapper.py | sort -k1,1 | python reducer.py'. The output is a list of words and their counts, sorted by word. The words and counts are: 6 (1), a (1), Ahmad (1), an (1), and (1), are (1), as (1), at least (1), be (2), big (1), but (1), can (1), class (1), data (1), demo (1), document (1), end (1), Everyone (1), for (1), have (2), having (1), and here (1).

```
cloudera@quickstart:~  
File Edit View Search Terminal Help  
[cloudera@quickstart ~]$ cat word_count_data.txt | python mapper.py | sort -k1,1 | python reducer.py  
6      1  
a      1  
Ahmad  1  
an     1  
and    1  
are    1  
as     1  
at least 1  
be     2  
big    1  
but    1  
can    1  
class  1  
data   1  
demo   1  
document      1  
end      1  
Everyone    1  
for      1  
have     2  
having   1  
here     1
```

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.

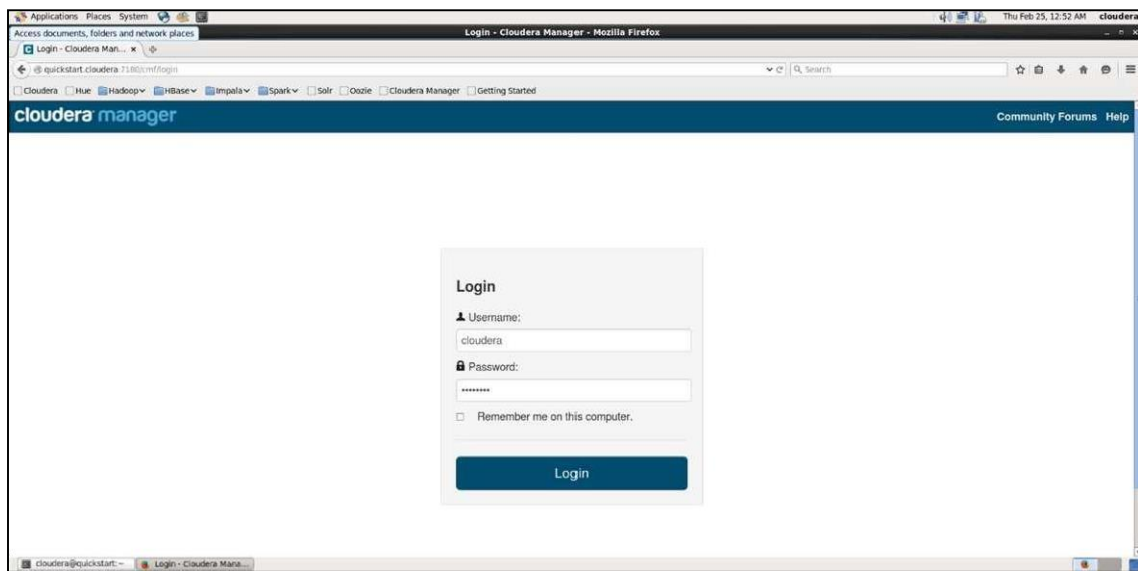
Step 5: Configure Hadoop services and settings.

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

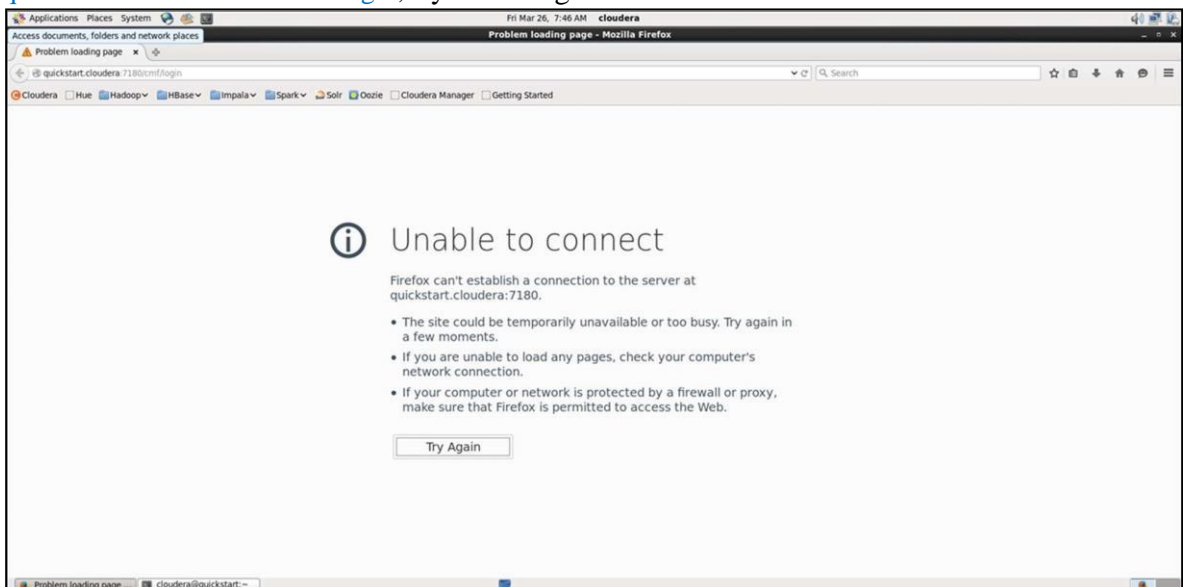
5a: Login to Cloudera Quickstart.

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.



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

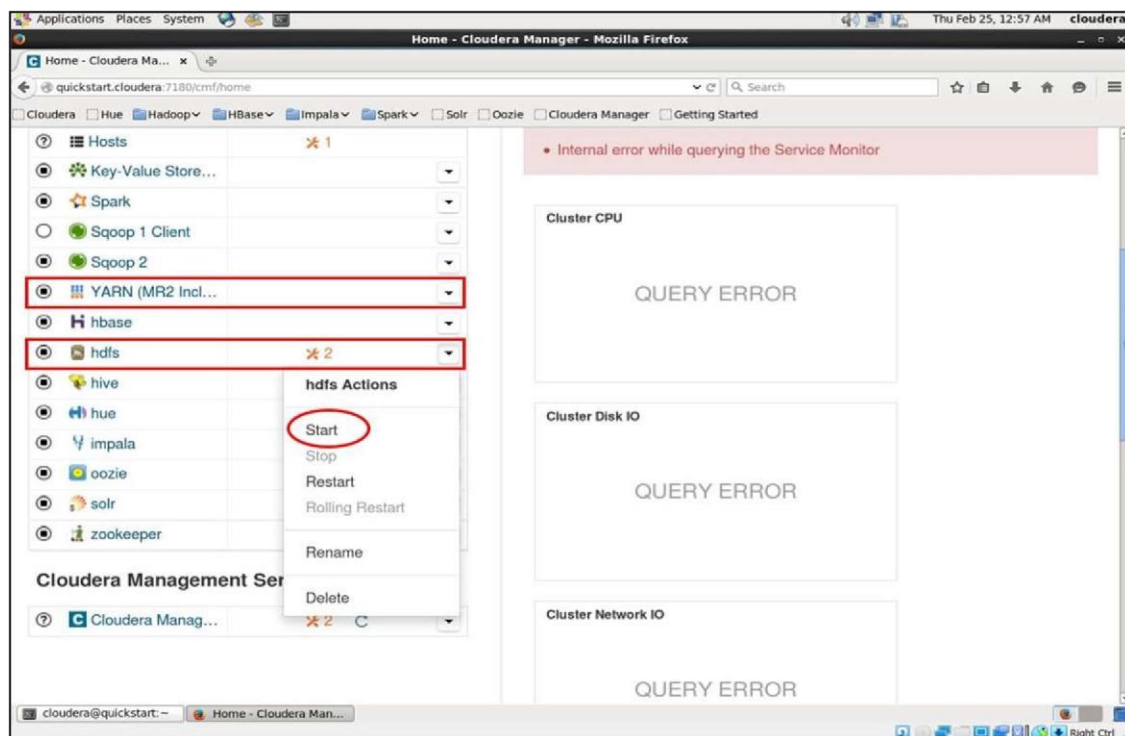


Restart CDH services by typing the following command: `sudo /home/cloudera/cloudera-manager --express --force`

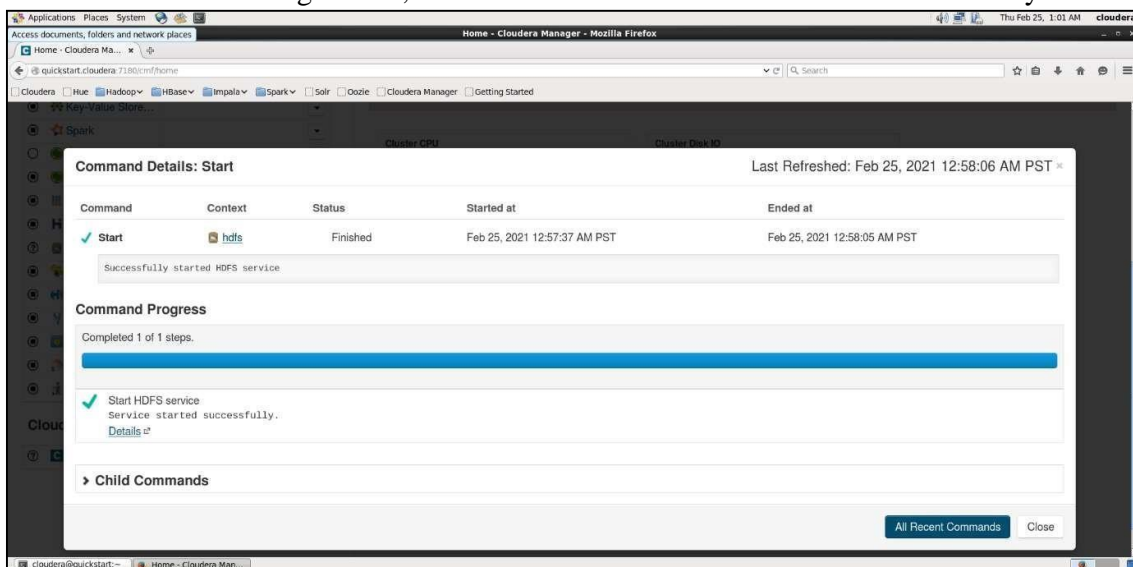


5b: Start HDFS and YARN services.

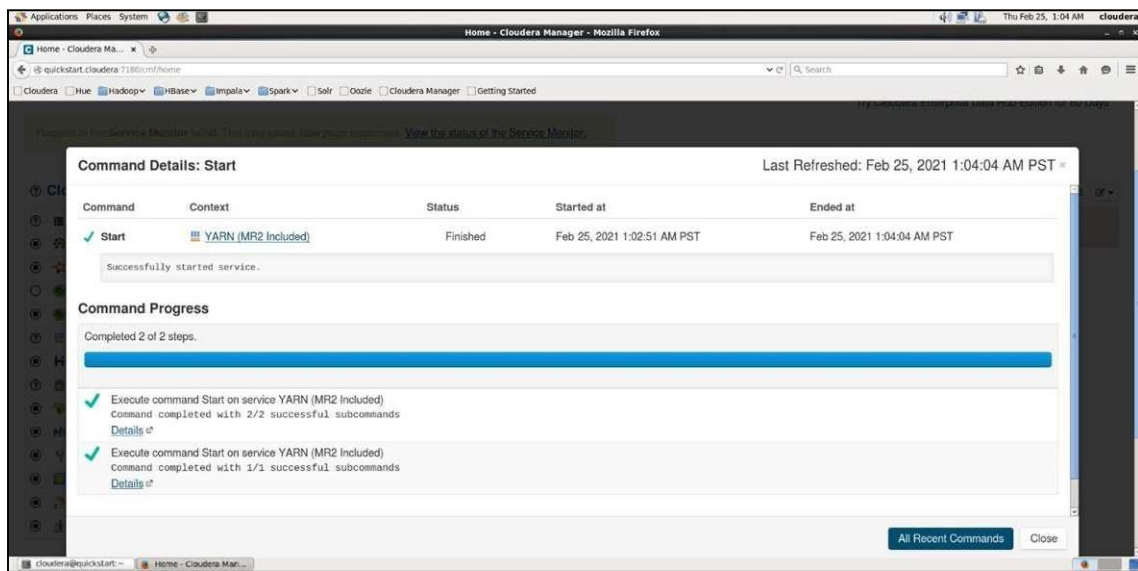
Click the dropdown arrow and choose Start option for HDFS and YARN services.



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



HDFS service started successfully.



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.

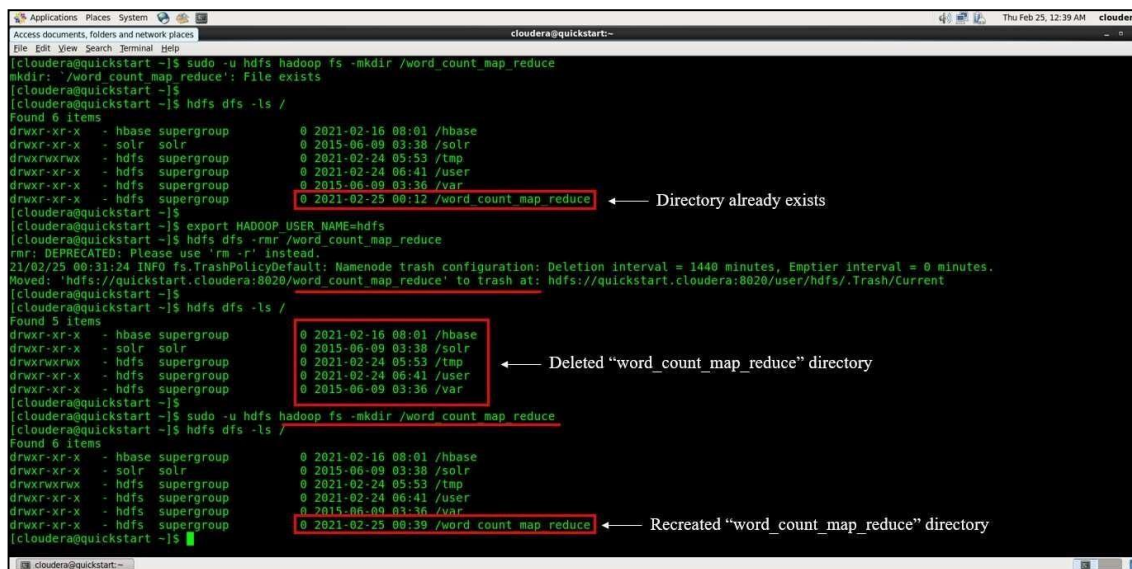
```
hdfs dfs -mkdir /word_count_map_reduce
```

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 /
```



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.

```
hdfs dfs -put /home/cloudera/word_count_data.txt /word_count_map_reduce
```

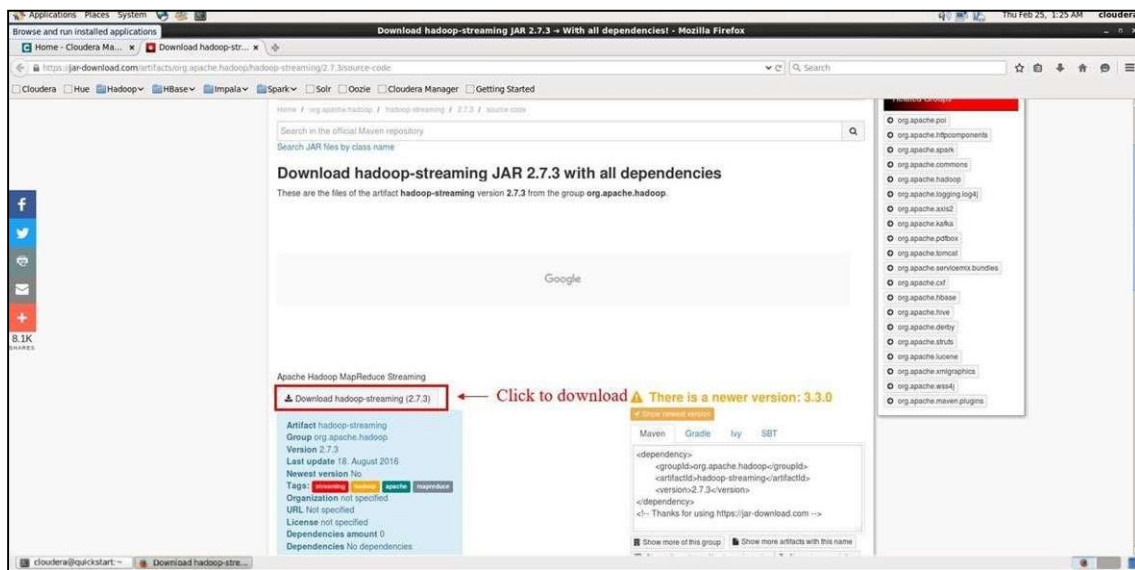
Check if file was copied successfully to the desired location.

```
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 ← File copied successfully
cloudera@quickstart:~$
```

Step 8: Download hadoop-streaming JAR 2.7.3.

Open browser and go to <https://jar-download.com/artifacts/org.apache.hadoop/hadoop-streaming?p=4> and download hadoop-streaming JAR 2.7.3 file.



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.txt
cloudera@quickstart:~$
```

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
hdfs dfs -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
```

```
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
21/02/25 01:55:56 INFO client.RMProxy: Connecting to ResourceManager at quickstart.cloudera/10.0.2.15:8032
21/02/25 01:55:58 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_1614243832747_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=34852
  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 score-slots taken by all map tasks=53496
  Total score-slots 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


```
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
```

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`

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.