```
dikshant@dikshant-Inspiron-5567:~/Documents$ touch word_count_data.txt
dikshant@dikshant-Inspiron-5567:~/Documents$ nano word_count_data.txt
dikshant@dikshant-Inspiron-5567:~/Documents$ cat word_count_data.txt
geeks for geeks is best online coding platform
welcome to geeks for geeks hadoop streaming tutorial
dikshant@dikshant-Inspiron-5567:~/Documents$
```


Step 2: Create a **mapper.py** file that implements the mapper logic. It will read the data from STDIN and will split the lines into words, and will generate an output of each word with its individual count.

```
cd Documents/                                    # to change the
directory to /Documents
touch mapper.py                    # touch is used to create an empty
file
cat mapper.py                    # cat is used to see the content of the
file
```

Copy the below code to the *mapper.py* file.

Python3

```
#!/usr/bin/env 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)
```

Here in the above program **#!** is known as shebang and used for interpreting the script. The file will be run using the command we are specifying.

```
dikshant@dikshant-Inspiron-5567:~$  cd Documents/
dikshant@dikshant-Inspiron-5567:~/Documents$ cat mapper.py
#!/usr/bin/env 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:
    line = line.strip()
    words = line.split()
    for word in words:
        print '%s\t%s' % (word, 1)
dikshant@dikshant-Inspiron-5567:~/Documents$
```

Let's test our **mapper.py** locally that it is working fine or not.

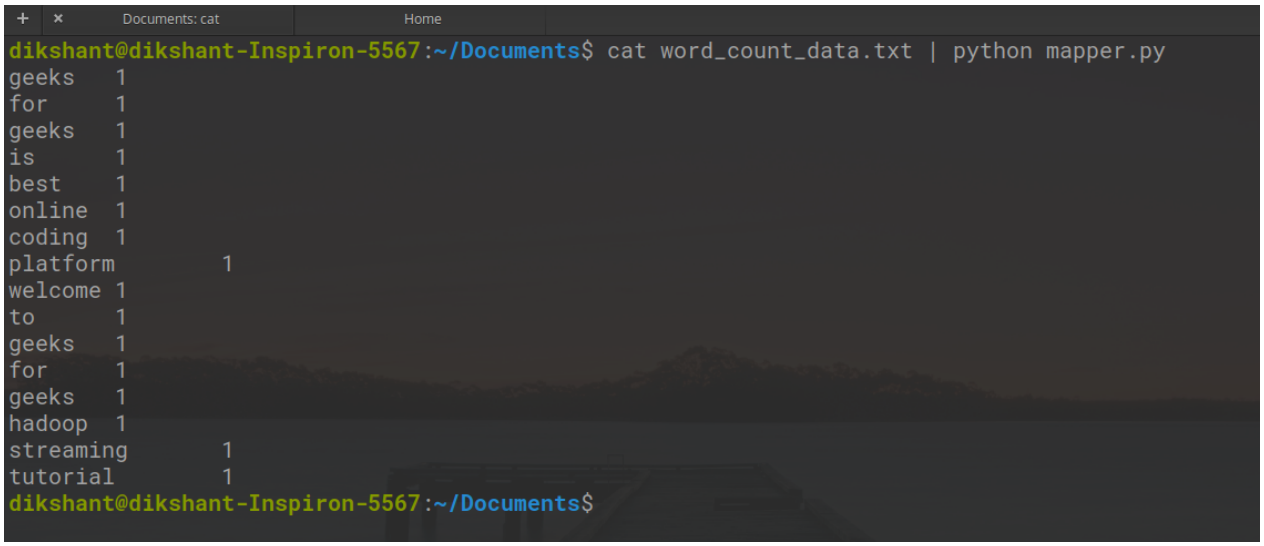
Syntax:

```
cat <text_data_file> | python <mapper_code_python_file>
```

Command(in my case)

```
cat word_count_data.txt | python mapper.py
```

The output of the mapper is shown below.



Step 3: Create a ***reducer.py*** file that implements the reducer logic. It will read the output of mapper.py from STDIN(standard input) and will aggregate the occurrence of each word and will write the final output to STDOUT.

```
cd Documents/                                    # to change the
directory to /Documents
touch reducer.py                        # touch is used to create an empty
file
```

Python3

```
#!/usr/bin/env 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()
    # splitting 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)
```

Now let's check our reducer code reducer.py with mapper.py is it working properly or not with the help of the below command.

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

```
dikshant@dikshant-Inspiron-5567:~/Documents$ cat word_count_data.txt | python mapper.py | sort -k1,1 | pyt
hon reducer.py
best    1
coding  1
for     2
geeks   4
hadoop  1
is      1
online  1
platform        1
streaming       1
to      1
tutorial        1
welcome 1
dikshant@dikshant-Inspiron-5567:~/Documents$
```

We can see that our reducer is also working fine in our local system.

Step 4: Now let's start all our Hadoop daemons with the below command.

```
start-dfs.sh
```

```
start-yarn.sh
```

```
dikshant@dikshant-Inspiron-5567:~$ start-dfs.sh
Starting namenodes on [localhost]
localhost: starting namenode, logging to /home/dikshant/Documents/hadoop/logs/hadoop-dikshant-namenode-dik
shant-Inspiron-5567.out
localhost: starting datanode, logging to /home/dikshant/Documents/hadoop/logs/hadoop-dikshant-datanode-dik
shant-Inspiron-5567.out
Starting secondary namenodes [0.0.0.0]
0.0.0.0: starting secondarynamenode, logging to /home/dikshant/Documents/hadoop/logs/hadoop-dikshant-secon
darynamenode-dikshant-Inspiron-5567.out
dikshant@dikshant-Inspiron-5567:~$ start-yarn.sh
starting yarn daemons
starting resourcemanager, logging to /home/dikshant/Documents/hadoop/logs/yarn-dikshant-resourcemanager-di
kshant-Inspiron-5567.out
localhost: starting nodemanager, logging to /home/dikshant/Documents/hadoop/logs/yarn-dikshant-nodemanager
-dikshant-Inspiron-5567.out
dikshant@dikshant-Inspiron-5567:~$
```

Now make a directory **word_count_in_python** in our HDFS in the root directory that will store our **word_count_data.txt** file with the below command.

```
hdfs dfs -mkdir /word_count_in_python
```

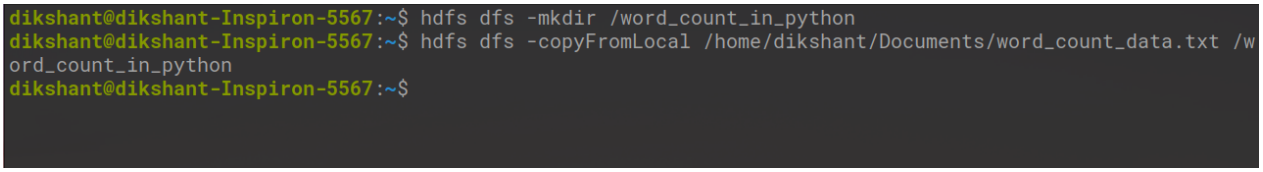
Copy **word_count_data.txt** to this folder in our HDFS with help of copyFromLocal command.

Syntax to copy a file from your local file system to the HDFS is given below:

```
hdfs dfs -copyFromLocal /path 1 /path 2 .... /path n /destination
```

Actual command(in my case)

```
hdfs dfs -copyFromLocal /home/dikshant/Documents/word_count_data.txt
/word_count_in_python
```



Now our data file has been sent to HDFS successfully. we can check whether it sends or not by using the below command or by manually visiting our HDFS.

```
hdfs dfs -ls /       # list down content of the root directory

hdfs dfs -ls /word_count_in_python    # list down content of
/word_count_in_python directory
```

```
dikshant@dikshant-Inspiron-5567:~$ hdfs dfs -ls /
Found 10 items
-rw-r--r--   1 dikshant supergroup      39711 2020-07-04 09:39 /CRND0103-2020-AK_Fairbanks_11_NE.txt
drwxr-xr-x   - dikshant supergroup          0 2020-09-06 20:51 /CharCountResult
drwxrwxr-x+  - dikshant supergroup          0 2020-06-23 14:23 /Hadoop_File
drwxr-xr-x   - dikshant supergroup          0 2020-07-04 10:48 /MyOutput
drwxr-xr-x   - dikshant supergroup          0 2020-07-08 16:12 /Titanic_Output
-rw-r--r--   5 dikshant supergroup         14 2020-09-06 20:51 /test.txt
-rw-r--r--   1 dikshant supergroup      61117 2020-07-08 15:06 /titanic_data.txt
drwxrwxrwx   - dikshant supergroup          0 2020-06-14 21:43 /tmp
drwxr-xr-x   - dikshant supergroup          0 2020-06-14 21:43 /user
drwxr-xr-x   - dikshant supergroup          0 2020-09-17 23:12 /word_count_in_python
dikshant@dikshant-Inspiron-5567:~$ hdfs dfs -ls /word_count_in_python
Found 1 items
-rw-r--r--   1 dikshant supergroup        101 2020-09-17 22:48 /word_count_in_python/word_count_data.txt
dikshant@dikshant-Inspiron-5567:~$
```

Let's give executable permission to our **mapper.py** and **reducer.py** with the help of below command.

```
cd Documents/

chmod 777 mapper.py reducer.py     # changing the permission to read,
write, execute for user, group and others
```

In below image,Then we can observe that we have changed the file permission.

```
dikshant@dikshant-Inspiron-5567:~$ cd Documents/
dikshant@dikshant-Inspiron-5567:~/Documents$ chmod 777 mapper.py reducer.py
dikshant@dikshant-Inspiron-5567:~/Documents$ ls -l
total 320
drwxr-xr-x  9 dikshant dikshant   4096 Jun 14 21:12  apache-hive-2.1.1-bin
-rw-rw-r--  1 dikshant dikshant   4186 Jul  8 16:12  Average_age.jar
drwxrwxr-x  4 dikshant dikshant   4096 Jul 11 18:48 'CAT VS DOG Image CLASSIFICATION'
-rw-rw-r--  1 dikshant dikshant   3796 Sep  6 20:21  charectercount.jar
-rw-rw-r--  1 dikshant dikshant  39711 Jul  3 15:58  CRND0103-2020-AK_Fairbanks_11_NE.txt
-rw-rw-r--  1 dikshant dikshant    538 Jul 28 15:09  dda_line.txt
-rw-rw-r--  1 dikshant dikshant   7236 Jul  4 14:47 'grammerly cookie'
drwxr-xr-x 10 dikshant dikshant   4096 Jun 13 13:05  hadoop
drwxrwxr-x  2 dikshant dikshant   4096 Sep  9 15:34 'Hadoop Books PDF'
-rw-r--r--  1 dikshant dikshant 129201 Sep 15 15:42  hadoop-streaming-2.7.3.jar
-rw-------  1 dikshant dikshant   1292 Jul 23 11:17  info.txt
drwxrwxr-x  2 dikshant dikshant   4096 Jul 10 18:34 'Kaggle API Key JSON'
drwxrwxrwx  5 dikshant dikshant   4096 Jun  2 07:40 'Kangaroo Rooms Training'
drwxrwxr-x  3 dikshant dikshant   4096 Jun 29 16:34 'Machine Learning ASS Hindi'
-rwxrwxrwx  1 dikshant dikshant    279 Sep 17 20:04  mapper.py
drwxrwxr-x  2 dikshant dikshant   4096 Sep 11 10:19 'ML Practice'
drwxrwxr-x  2 dikshant dikshant   4096 Jun 16 17:37 'Octave Project Data'
drwxr-xr-x 16 dikshant dikshant   4096 Jun  2  2017  pig-0.17.0
drwxrwxrwx  5 dikshant dikshant   4096 Sep 11 10:23 'Practice 2'
drwxrwxr-x  4 dikshant dikshant   4096 Aug 16 20:17 'Python By Vimal Daga IIEC_RISE'
-rwxrwxrwx  1 dikshant dikshant   1066 Sep 17 22:38  reducer.py
-rw-r--r--  1 dikshant dikshant  61117 Jul  8 13:16  titanic_data.txt
drwxrwxr-x  4 dikshant dikshant   4096 Jul 11 21:05 'Tomato Leaf Classification'
-rw-rw-r--  1 dikshant dikshant    101 Sep 17 19:20  word_count_data.txt
dikshant@dikshant-Inspiron-5567:~/Documents$
```

Step 5: Now download the latest **hadoop-streaming jar** file from this Link. Then place, this Hadoop,-streaming jar file to a place from you can easily access it. In my case, I am placing it to ***/Documents*** folder where **mapper.py** and **reducer.py** file is present.

Now let's run our python files with the help of the Hadoop streaming utility as shown below.

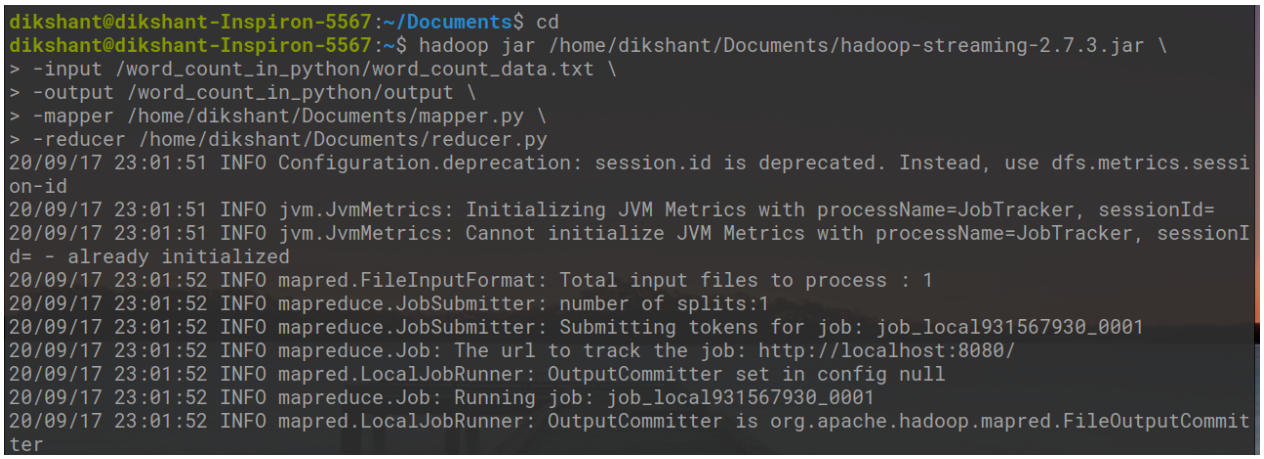
```
hadoop jar /home/dikshant/Documents/hadoop-streaming-2.7.3.jar \

> -input /word_count_in_python/word_count_data.txt \

> -output /word_count_in_python/output \

> -mapper /home/dikshant/Documents/mapper.py \

> -reducer /home/dikshant/Documents/reducer.py
```



In the above command in **-output**, we will specify the location in HDFS where we want our output to be stored. So let's check our output in output file at location ***/word_count_in_python/output/part-00000*** in my case. We can check results by manually vising the location in HDFS or with the help of cat command as shown below.

```
hdfs dfs -cat /word_count_in_python/output/part-00000
```

```
dikshant@dikshant-Inspiron-5567:~$ hdfs dfs -cat /word_count_in_python/output/part-00000
best    1
coding  1
for     2
geeks   4
hadoop  1
is      1
online  1
platform        1
streaming       1
to      1
tutorial        1
welcome 1
dikshant@dikshant-Inspiron-5567:~$
```

Basic options that we can use with Hadoop Streaming

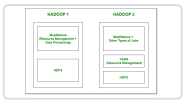
Option	Description
-mapper	The command to be run as the mapper
-reducer	The command to be run as the reducer
-input	The DFS input path for the Map step
-output	The DFS output directory for the Reduce step

Last Updated : 19 Jan, 2022

11

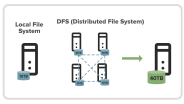
Similar Reads

What is Hadoop Streaming?



Difference between Hadoop 1 and Hadoop 2

Difference Between Hadoop 2.x vs Hadoop 3.x



Hadoop - HDFS (Hadoop Distributed File System)