PRINCIPLES OF BIG DATA MANAGEMENT

PHASE #1

TEAM MEMBERS:


```
          Gopi Chand Bodepudi (gbhmh@mail.umkc.edu) - 1627
8459

          Vasudeva Madala (vmcpq@mail.umkc.edu) - 16280569

          Anusha Palla (apgmc@mail.umkc.edu) - 16280777
```


Github Link of the Project:

https://github.com/gbhmh/Principles-of-BigData-Manangement

Objective:

• The principle point of this stage is to develop a system to store, analyze, and visualize Twitter's tweets

.

• Tasks:

1. Collect Tweets using Twitter's Streaming APIs in any format

preferred JSON.

2. Extract all the hashtags and URLs in the tweets and store the content substance (e.g. tweet’s content) from the information into a document in HDFS.

3. Run a Word Count program in Apache Spark and Hadoop on the content document and collect the output and log files from Hadoop

Applications/Software’s Used:

Twitter Developer Account, Apache Spark, Python, Java Eclipse, Hadoop.

Collecting tweets from Twitter:

• Firstly, we have made an developer account in Twitter utilizing beneath connect. https://apps.twitter.com/

• Below are the factors that contains the client certifications to get to Twitter API

□□ACCESS_TOKEN = “2219941182-hJEd5re1y7lbZmVlyZySZvVsJf88fP6um3SsC3r"

□□ACCESS_SECRET = " BntHym97rzCisKS3BFXqrBgQbgokklZEBcqHXixGJQtX8 "

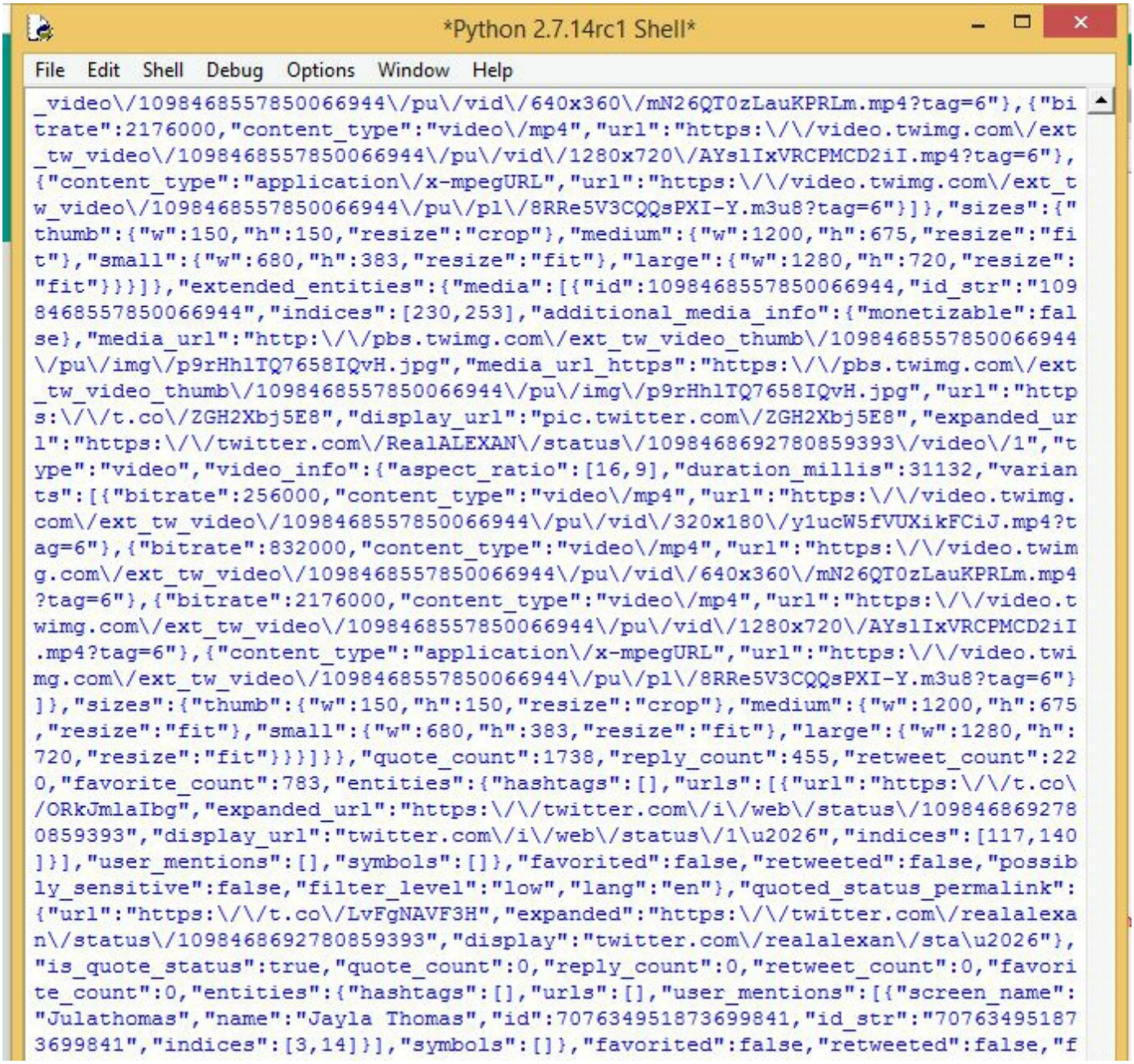
□□CONSUMER_KEY = " 187ztf3hxmT3Nm3YonFzcAvEB "

□□CONSUMER_SECRET = “hTqPaSjNXw21GXmPCey6CZBCZRoO1EbTkbVO4zMv77kN8Ikq0P "

We have composed python program that is utilized to bring tweets in JSON design.

(twitter.py)

Link: https://github.com/gbhmh/Principles-of-BigData-Manangement/tree/master/phase1/source/python programs




_video\/1098468557850066944\/pu\/vid\/640x360\/mN26QT0zLauKPRLm.mp4?tag=6"},{"bitrate":2176000,"content_type":"video\/mp4","url":"https:\/\/video.twimg.com\/ext_tw_video\/1098468557850066944\/pu\/vid\/1280x720\/AYslIxVRCPMCD2iI.mp4?tag=6"},{"content_type":"application\/x-mpegURL","url":"https:\/\/video.twimg.com\/ext_tw_video\/1098468557850066944\/pu\/pl\/8RRe5V3CQQsPXI-Y.m3u8?tag=6"}]},"sizes":{"thumb":{"w":150,"h":150,"resize":"crop"},"medium":{"w":1200,"h":675,"resize":"fit"},"small":{"w":680,"h":383,"resize":"fit"},"large":{"w":1280,"h":720,"resize":"fit"}}}]},"extended_entities":{"media":[{"id":1098468557850066944,"id_str":"1098468557850066944","indices":[230,253],"additional_media_info":{"monetizable":false},"media_url":"http:\/\/pbs.twimg.com\/ext_tw_video_thumb\/1098468557850066944\/pu\/img\/p9rHhlTQ7658IQvH.jpg","media_url_https":"https:\/\/pbs.twimg.com\/ext_tw_video_thumb\/1098468557850066944\/pu\/img\/p9rHhlTQ7658IQvH.jpg","url":"https:\/\/t.co\/ZGH2Xbj5E8","display_url":"pic.twitter.com\/ZGH2Xbj5E8","expanded_url":"https:\/\/twitter.com\/RealALEXAN\/status\/1098468692780859393\/video\/1","type":"video","video_info":{"aspect_ratio":[16,9],"duration_millis":31132,"variants":[{"bitrate":256000,"content_type":"video\/mp4","url":"https:\/\/video.twimg.com\/ext_tw_video\/1098468557850066944\/pu\/vid\/320x180\/y1ucW5fVUXikFCiJ.mp4?tag=6"},{"bitrate":832000,"content_type":"video\/mp4","url":"https:\/\/video.twimg.com\/ext_tw_video\/1098468557850066944\/pu\/vid\/640x360\/mN26QT0zLauKPRLm.mp4?tag=6"},{"bitrate":2176000,"content_type":"video\/mp4","url":"https:\/\/video.twimg.com\/ext_tw_video\/1098468557850066944\/pu\/vid\/1280x720\/AYslIxVRCPMCD2iI.mp4?tag=6"},{"content_type":"application\/x-mpegURL","url":"https:\/\/video.twimg.com\/ext_tw_video\/1098468557850066944\/pu\/pl\/8RRe5V3CQQsPXI-Y.m3u8?tag=6"}]},"sizes":{"thumb":{"w":150,"h":150,"resize":"crop"},"medium":{"w":1200,"h":675,"resize":"fit"},"small":{"w":680,"h":383,"resize":"fit"},"large":{"w":1280,"h":720,"resize":"fit"}}}]}},"quote_count":1738,"reply_count":455,"retweet_count":220,"favorite_count":783,"entities":{"hashtags":[],"urls":[{"url":"https:\/\/t.co\/ORkJmlaIbg","expanded_url":"https:\/\/twitter.com\/i\/web\/status\/1098468692780859393","display_url":"twitter.com\/i\/web\/status\/1\u2026","indices":[117,140]}],"user_mentions":[],"symbols":[]},"favorited":false,"retweeted":false,"possibly_sensitive":false,"filter_level":"low","lang":"en"},"quoted_status_permalink":{"url":"https:\/\/t.co\/LvFgNAVF3H","expanded":"https:\/\/twitter.com\/realalexan\/status\/1098468692780859393","display":"twitter.com\/realalexan\/sta\u2026"},"is_quote_status":true,"quote_count":0,"reply_count":0,"retweet_count":0,"favorite_count":0,"entities":{"hashtags":[],"urls":[],"user_mentions":[{"screen_name":"Julathomas","name":"Jayla Thomas","id":707634951873699841,"id_str":"707634951873699841","indices":[3,14]}],"symbols":[]},"favorited":false,"retweeted":false,"f

• The extricated record in JSON arrange contains all the tweet points of interest, for example, id, created at, text, profile_background_image_url and so forth.

• From JSON tweets record just the hashtags, url's content substance is extricated utilizing Python programs.
(twitter1.py and twitter2.py)

Link : https://github.com/gbhmh/Principles-of-BigData-

Manangement/tree/master/phase1/source/python programs

. The extracted files are at :

https://github.com/gbhmh/Principles-of-BigData-Manangement/tree/master/phase1/source/twitter files

Store the extracted files from the data into a file in HDFS.


• The data is moved from local to HFDS.

• First a folder is made in HDFS and the document is moved from local to HDFS utilizing underneath order.

Make directory in local: hdfs dfs -mkdir -p /hadout

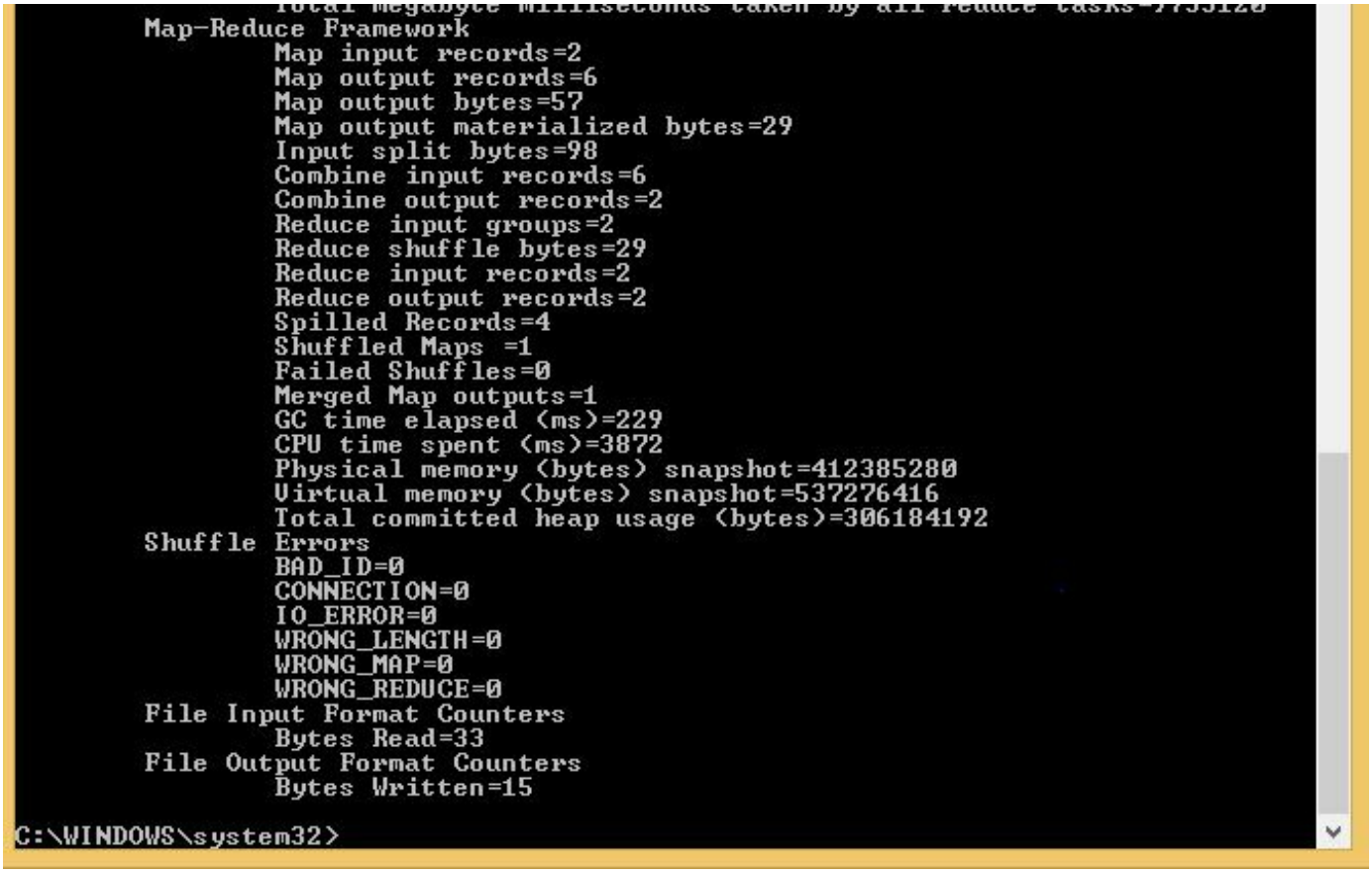
Move file from local to HDFS: hdfs dfs -put C:\Python27\FileOutput_hash1.txt / hadout / hadout.txt

Now we have run Wordcount program using the jar files that are obtained from Madreduce program written in Eclipse IDE using Java

Command : hadoop jar C:/Users/purnak/Desktop/jab/javjar.jar

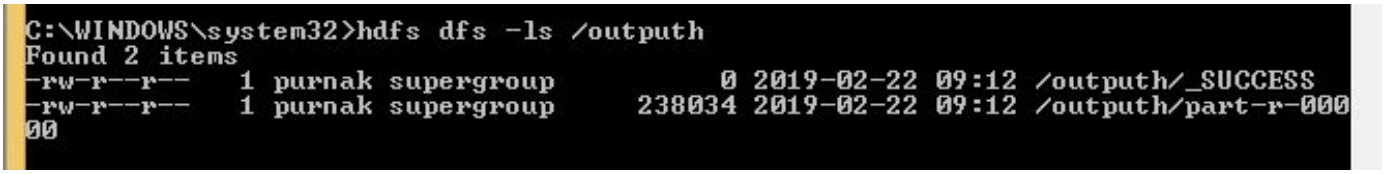
WordCount /hadout /outputh

```
C:\WINDOWS\system32>hadoop jar C:/Users/purnak/Desktop/jab/javjar.jar WordCount
/input /abc
19/02/23 07:03:57 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0
:8032
19/02/23 07:04:00 WARN mapreduce.JobResourceUploader: Hadoop command-line option
 parsing not performed. Implement the Tool interface and execute your applicatio
n with ToolRunner to remedy this.
19/02/23 07:04:01 INFO input.FileInputFormat: Total input paths to process : 1
19/02/23 07:04:02 INFO mapreduce.JobSubmitter: number of splits:1
19/02/23 07:04:03 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_15
50782269746_0003
19/02/23 07:04:07 INFO impl.YarnClientImpl: Submitted application application_15
50782269746_0003
19/02/23 07:04:07 INFO mapreduce.Job: The url to track the job: http://Purna:808
8/proxy/application_1550782269746_0003/
19/02/23 07:04:07 INFO mapreduce.Job: Running job: job_1550782269746_0003
19/02/23 07:04:44 INFO mapreduce.Job: Job job_1550782269746_0003 running in uber
 mode : false
19/02/23 07:04:44 INFO mapreduce.Job:  map 0% reduce 0%
19/02/23 07:04:57 INFO mapreduce.Job:  map 100% reduce 0%
19/02/23 07:05:10 INFO mapreduce.Job:  map 100% reduce 100%
19/02/23 07:05:17 INFO mapreduce.Job: Job job_1550782269746_0003 completed succe
ssfully
19/02/23 07:05:17 INFO mapreduce.Job: Counters: 49
        File System Counters
                FILE: Number of bytes read=29
                FILE: Number of bytes written=236301
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=131
                HDFS: Number of bytes written=15
                HDFS: Number of read operations=6
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
        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)=9354
                Total time spent by all reduces in occupied slots (ms)=9505
                Total time spent by all map tasks (ms)=9354
                Total time spent by all reduce tasks (ms)=9505
                Total vcore-milliseconds taken by all map tasks=9354
                Total vcore-milliseconds taken by all reduce tasks=9505
                Total megabyte-milliseconds taken by all map tasks=9578496
                Total megabyte-milliseconds taken by all reduce tasks=9733120
        Map-Reduce Framework
                Map input records=2
                Map output records=6
                Map output bytes=57
                Map output materialized bytes=29
                Input split bytes=98
                Combine input records=6
                Combine output records=2
                Reduce input groups=2
                Reduce shuffle bytes=29
                Reduce input records=2
                Reduce output records=2
                Spilled Records=4
                Shuffled Maps =1
                Failed Shuffles=0
```



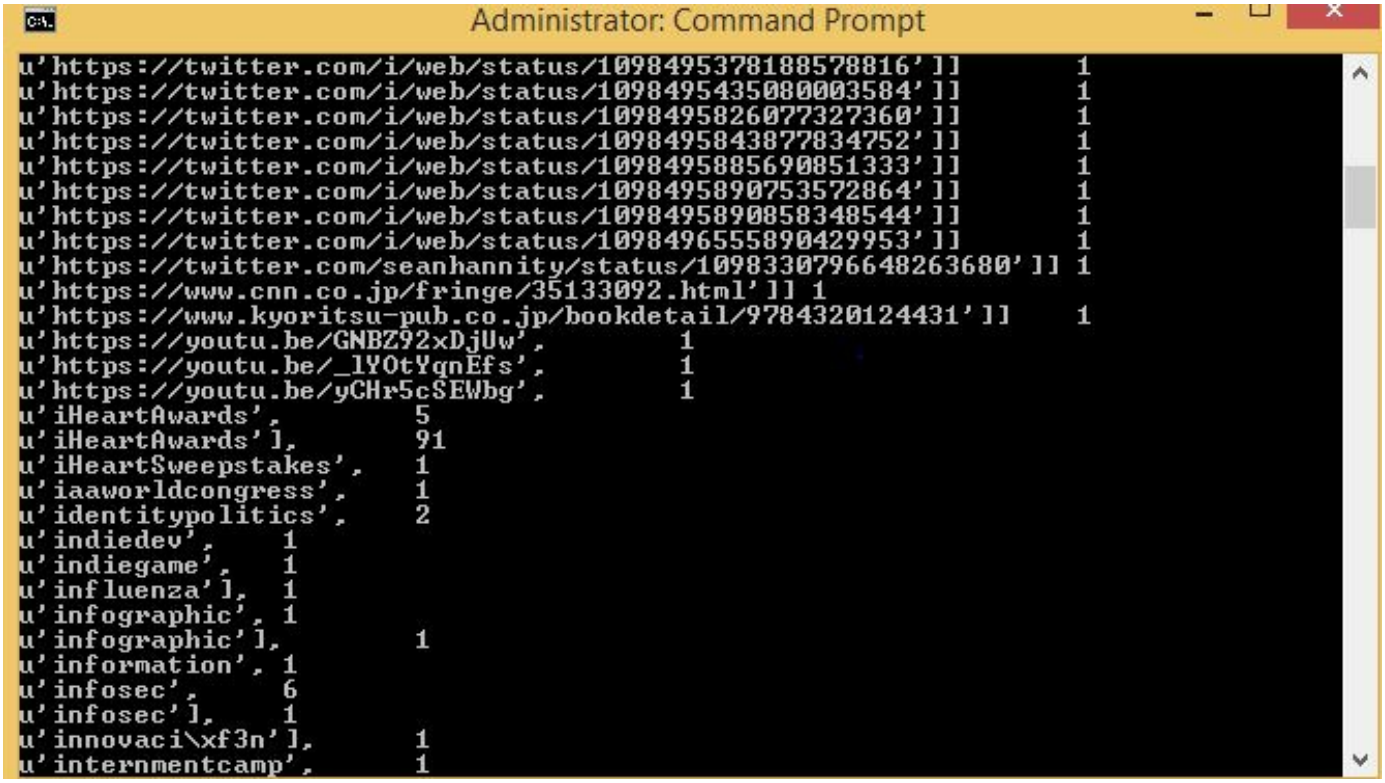

once run is successfull u can see the two files in output

```
    command: hdfs dfs -ls /outputh --> Here two files will
be displayed.
```



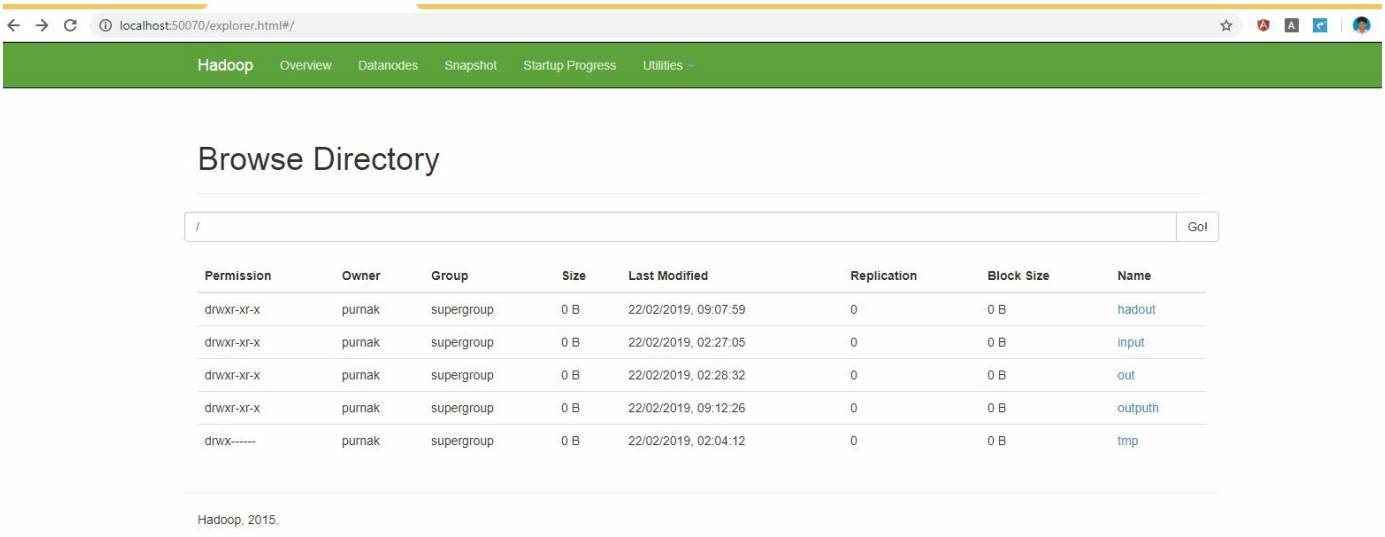

Finally the wordcount output will watch by using the following command.

```
    command: hadoop fs -cat /outputh/part-r-00000
```




Copy output to local space

```
    Command: hdfs dfs -copyToLocal /outputh/part-r-00000 C
:/Users/purnak/Desktop/jabout
```




Run a Word Count program in Apache Spark on the extracted file

• Then, after running the word count example on Hadoop, now it’s time to run the same word count example using Apache Spark.

• The output obtained from the word count running on Apache Hadoop is almost similar to the output obtained from Apache Spark except the minor differences.

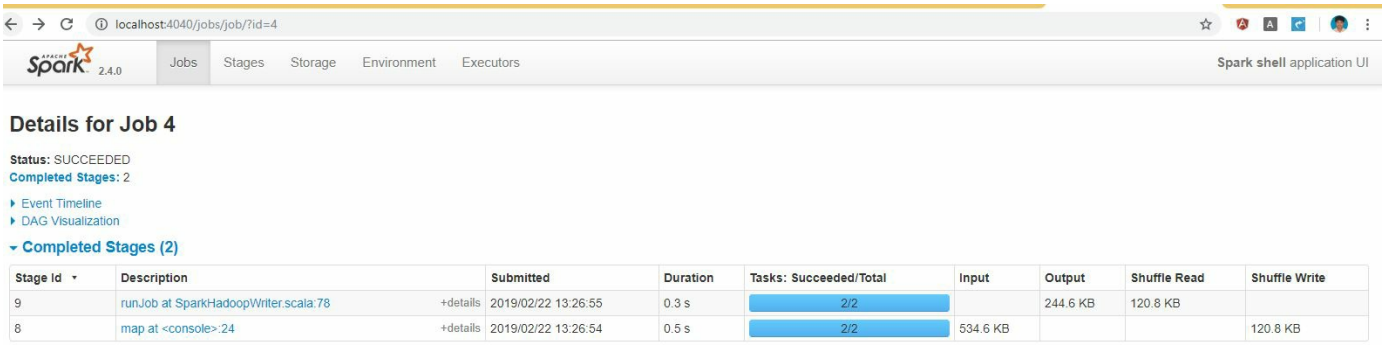
The command used for to run wordcount in scala is :

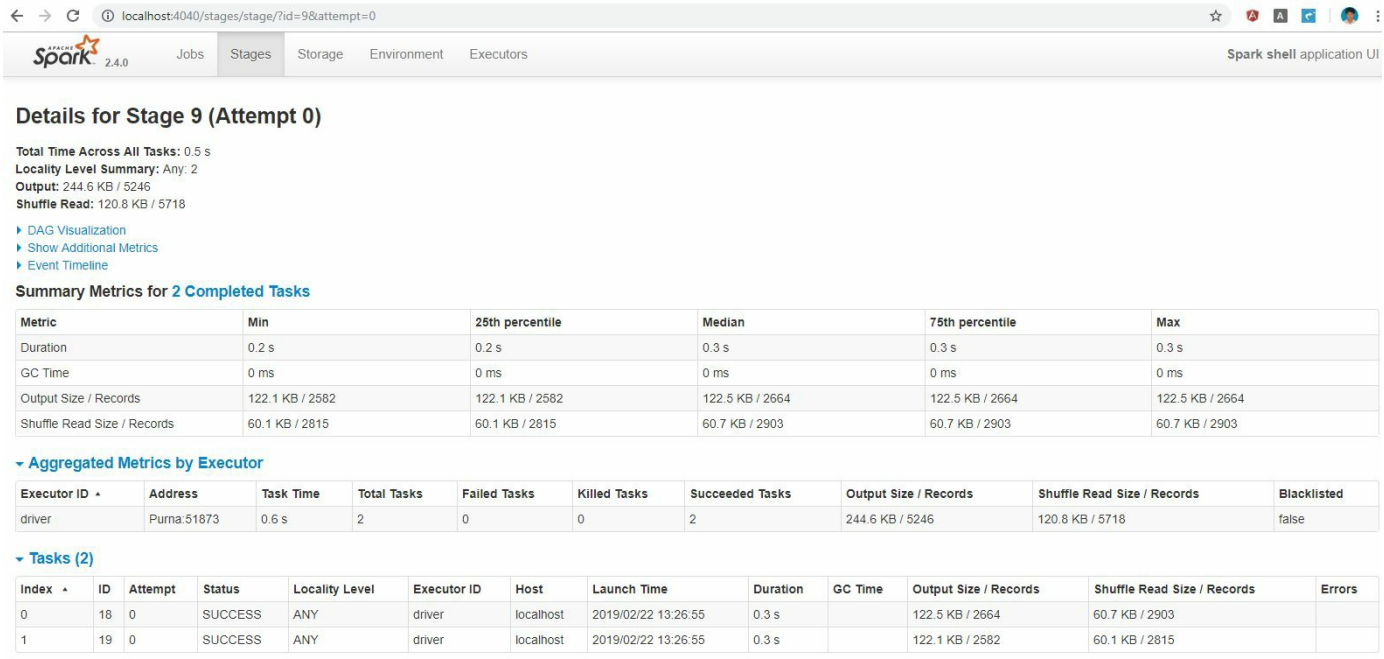
```
scala> val file =sc.textFile("test5/FileOutput_hash1.txt").flatMap(_.split(" "))
.map(word => (word,1)).reduceByKey(_+_).saveAsTextFile("C:\\Users\\purnak\\test7
")
[Stage 8:>                                                          (0 + 2) / 2]

file: Unit = ()
```

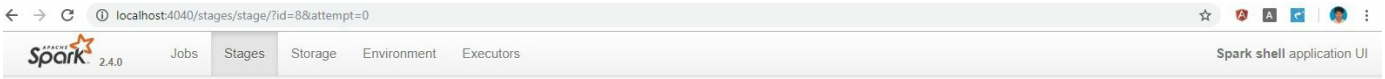
Output files are stored in local system after the execution of the above

command

Spark directories are






# Details for Stage 8 (Attempt 0)

**Total Time Across All Tasks:** 0.9 s
**Locality Level Summary:** Process local: 2
**Input Size / Records:** 534.6 KB / 14741
**Shuffle Write:** 120.8 KB / 5718

- DAG Visualization
- Show Additional Metrics
- Event Timeline

## Summary Metrics for 2 Completed Tasks

| Metric | Min | 25th percentile | Median | 75th percentile | Max |
|---|---|---|---|---|---|
| Duration | 0.4 s | 0.4 s | 0.5 s | 0.5 s | 0.5 s |
| GC Time | 27 ms | 27 ms | 27 ms | 27 ms | 27 ms |
| Input Size / Records | 214.6 KB / 7326 | 214.6 KB / 7326 | 320.0 KB / 7415 | 320.0 KB / 7415 | 320.0 KB / 7415 |
| Shuffle Write Size / Records | 59.8 KB / 2795 | 59.8 KB / 2795 | 61.0 KB / 2923 | 61.0 KB / 2923 | 61.0 KB / 2923 |

## Aggregated Metrics by Executor

| Executor ID | Address | Task Time | Total Tasks | Failed Tasks | Killed Tasks | Succeeded Tasks | Input Size / Records | Shuffle Write Size / Records | Blacklisted |
|---|---|---|---|---|---|---|---|---|---|
| driver | Purna:51873 | 0.9 s | 2 | 0 | 0 | 2 | 534.6 KB / 14741 | 120.8 KB / 5718 | false |

## Tasks (2)

| Index | ID | Attempt | Status | Locality Level | Executor ID | Host | Launch Time | Duration | GC Time | Input Size / Records | Write Time | Shuffle Write Size / Records | Errors |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 16 | 0 | SUCCESS | PROCESS_LOCAL | driver | localhost | 2019/02/22 13:26:54 | 0.4 s | 27 ms | 320.0 KB / 7415 | 21 ms | 59.8 KB / 2795 | |
| 1 | 17 | 0 | SUCCESS | PROCESS_LOCAL | driver | localhost | 2019/02/22 13:26:54 | 0.5 s | 27 ms | 214.6 KB / 7326 | 0.1 s | 61.0 KB / 2923 | |