# PRINCIPLES OF BIG DATA MANAGEMENT

## PHASE-1

**Objective:**

- The main aim of this phase is to develop a system to store, analyze, and visualize a social network data.
- Tasks:
  1. Collect social network data (e.g. tweets) in JSON format.
  2. Extracting the hashtags, URLs from the collected tweets.
  3. Store the text content (e.g. tweet's text) from the data into a file in HDFS.
  4. Run a Word Count program in Apache Spark, Apache Hadoop on extracted hashtags, URLs files and store the output and log files locally.

**Applications/Software's Used:** Apache Spark, Hadoop, Scala, Cloudera, Twitter Developer Account, Python.

**Tweets collection from Twitter:**

- Firstly, we have created a developer account in Twitter using link: https://apps.twitter.com/
- Below are the variables that contains the user credentials to access Twitter API

➢ **API ACCESS_TOKEN** = "1089997729219178496-BvhcwdDw6eGPbPJM5wJYqOvsq7rE7G"

➢ **ACCESS_SECRET** = "xZu9ILsesdPiRi5quCxx94mQTfrj00sMEedGrddvYOmsQ"

➢ **CONSUMER_KEY** = "f4qQNYSFaGxO4iB6SamtIZarf"

➢ **CONSUMER_SECRET** = "PTuSFEtZAeRP9nov16hJTWQzFNedKTSDKR9ZSS1Iu65JiSbXvO"

- We have written python program that is used to fetch tweets in JSON format. (Tweetscollection.py)
- The extracted file in JSON format contains all the tweet details such as id, created at, text, URL, hashtags etc.
- From JSON tweets file only the hashtags and URLs content is extracted using python program for hashtags and URLs. Then fetched details are stored in files.

**Run the Word Count program in Apache Spark and Apache Hadoop the extracted hashtags/URLs and collect the output and log files:**

**Word Count Implementation In spark:**

**Step 1**:

On the Hadoop terminal, First I will see my input file is where it is from terminal

```
[cloudera@quickstart ~]$ pwd
/home/cloudera
[cloudera@quickstart ~]$ ls
cloudera-manager  cm_api.py  Desktop  Documents  Downloads  eclipse  enterprise-deployment.json  express-deployment.json  kerberos  lib  Music  parcels  Pictures  Public  Templates  twitterdata.txt  Videos  workspace
[cloudera@quickstart ~]$ hdfs dfs -put /home/cludera/twitterdata.txt /user/cl
oudera/
put: `/home/cludera/twitterdata.txt': No such file or directory
[cloudera@quickstart ~]$ hdfs dfs -put /home/cloudera/twitterdata.txt /user/c
loudera/
put: `/user/cloudera/twitterdata.txt': File exists
[cloudera@quickstart ~]$ hdfs dfs -ls /user/cloudera
Found 1 items
-rw-r--r--   1 cloudera cloudera     460965 2019-02-21 12:52 /user/cloudera/t
witterdata.txt
[cloudera@quickstart ~]$  spark-shell
```

**Step 2:**

As you can see in the step1 screenshot from the ls command out input file is twitterdata.txt. Then, I put the file in Hadoop file system and change the location according to my convenience. Now, I have open spark-shell will implement word count program.

```
        at org.apache.spark.deploy.SparkSubmit$.doRunMain$1(SparkSubmit.scala
:181)
        at org.apache.spark.deploy.SparkSubmit$.submit(SparkSubmit.scala:206)
        at org.apache.spark.deploy.SparkSubmit$.main(SparkSubmit.scala:121)
        at org.apache.spark.deploy.SparkSubmit.main(SparkSubmit.scala)


scala> pwd
<console>:26: error: not found: value pwd
              pwd
              ^

scala>
[1]+  Stopped                 spark-shell
[cloudera@quickstart ~]$ pwd
/home/cloudera
[cloudera@quickstart ~]$ ls
cloudera-manager  cm_api.py  Desktop  Documents  Downloads  eclipse  enterprise-deployment.json  express-deployment.json  kerberos  lib  Music  parcels  Pictures  Public  Templates  twitterdata.txt  Videos  workspace
[cloudera@quickstart ~]$ hdfs dfs -put /home/cludera/twitterdata.txt /user/cl
oudera/
put: `/home/cludera/twitterdata.txt': No such file or directory
[cloudera@quickstart ~]$ hdfs dfs -put /home/cloudera/twitterdata.txt /user/c
loudera/
put: `/user/cloudera/twitterdata.txt': File exists
[cloudera@quickstart ~]$ hdfs dfs -ls /user/cloudera
Found 1 items
-rw-r--r--   1 cloudera cloudera     460965 2019-02-21 12:52 /user/cloudera/t
witterdata.txt
[cloudera@quickstart ~]$  spark-shell
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel).
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/lib/zookeeper/lib/slf4j-log4j12-1.7.5.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/lib/flume-ng/lib/slf4j-log4j12-1.7.5.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/lib/parquet/lib/slf4j-log4j12-1.7.5.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/lib/avro/avro-tools-1.7.6-cdh5.13.0.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 1.6.0
      /_/

Using Scala version 2.10.5 (Java HotSpot(TM) 64-Bit Server VM, Java 1.7.0_67)
Type in expressions to have them evaluated.
Type :help for more information.
19/02/21 13:25:38 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
```

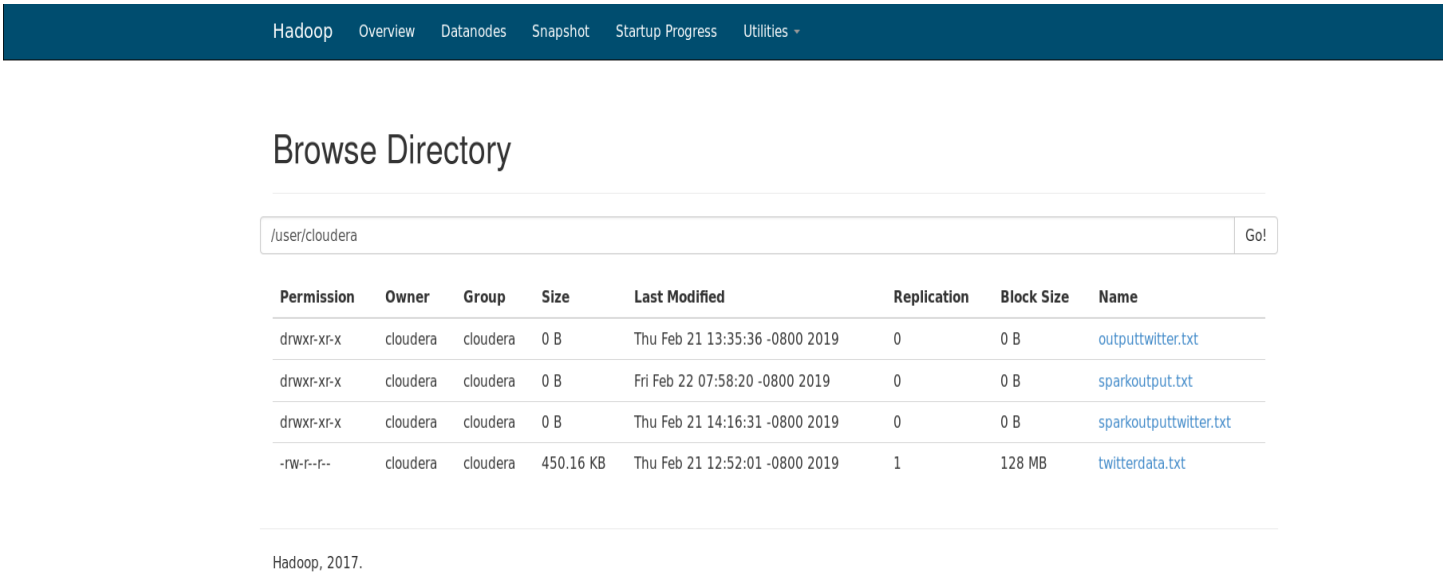
```
File  Edit  View  Search  Terminal  Help
        at org.apache.spark.repl.SparkIMain.interpret(SparkIMain.scala:852)
        at org.apache.spark.repl.SparkIMain.interpret(SparkIMain.scala:800)
        at org.apache.spark.repl.SparkILoop.reallyInterpret$1(SparkILoop.scala:857)
        at org.apache.spark.repl.SparkILoop.interpretStartingWith(SparkILoop.scala:902)
        at org.apache.spark.repl.SparkILoop.command(SparkILoop.scala:814)
        at org.apache.spark.repl.SparkILoopInit$$anonfun$initializeSpark$1.apply(SparkILoopInit.scala:125)
        at org.apache.spark.repl.SparkILoopInit$$anonfun$initializeSpark$1.apply(SparkILoopInit.scala:124)
        at org.apache.spark.repl.SparkIMain.beQuietDuring(SparkIMain.scala:305)
        at org.apache.spark.repl.SparkILoopInit$class.initializeSpark(SparkILoopInit.scala:124)
        at org.apache.spark.repl.SparkILoop.initializeSpark(SparkILoop.scala:64)
        at org.apache.spark.repl.SparkILoop$$anonfun$org$apache$spark$repl$SparkILoop$$process$1$$anonfun$apply$mcZ$sp$5.apply$mcV$sp(SparkILoop.scala:974)
        at org.apache.spark.repl.SparkILoopInit$class.runThunks(SparkILoopInit.scala:160)
        at org.apache.spark.repl.SparkILoop.runThunks(SparkILoop.scala:64)
        at org.apache.spark.repl.SparkILoopInit$class.postInitialization(SparkILoopInit.scala:108)
        at org.apache.spark.repl.SparkILoop.postInitialization(SparkILoop.scala:64)
        at org.apache.spark.repl.SparkILoop$$anonfun$org$apache$spark$repl$SparkILoop$$process$1.apply$mcZ$sp(SparkILoop.scala:991)
        at org.apache.spark.repl.SparkILoop$$anonfun$org$apache$spark$repl$SparkILoop$$process$1.apply(SparkILoop.scala:945)
        at org.apache.spark.repl.SparkILoop$$anonfun$org$apache$spark$repl$SparkILoop$$process$1.apply(SparkILoop.scala:945)
        at scala.tools.nsc.util.ScalaClassLoader$.savingContextLoader(ScalaClassLoader.scala:135)
        at org.apache.spark.repl.SparkILoop.org$apache$spark$repl$SparkILoop$$process(SparkILoop.scala:945)
        at org.apache.spark.repl.SparkILoop.process(SparkILoop.scala:1064)
        at org.apache.spark.repl.Main$.main(Main.scala:35)
        at org.apache.spark.repl.Main.main(Main.scala)
        at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
        at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:57)
        at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
        at java.lang.reflect.Method.invoke(Method.java:606)
        at org.apache.spark.deploy.SparkSubmit$.org$apache$spark$deploy$SparkSubmit$$runMain(SparkSubmit.scala:730)
        at org.apache.spark.deploy.SparkSubmit$.doRunMain$1(SparkSubmit.scala:181)
        at org.apache.spark.deploy.SparkSubmit$.submit(SparkSubmit.scala:206)
        at org.apache.spark.deploy.SparkSubmit$.main(SparkSubmit.scala:121)
        at org.apache.spark.deploy.SparkSubmit.main(SparkSubmit.scala)
19/02/22 07:55:52 WARN util.Utils: Service 'SparkUI' could not bind on port 4041. Attempting port 4042.
Spark context available as sc (master = local[*], app id = local-1550850953741).
19/02/22 07:56:07 WARN shortcircuit.DomainSocketFactory: The short-circuit local reads feature cannot be used because libhadoop cannot be loaded.
SQL context available as sqlContext.

scala> val loadTwitterData=sc.textFile("/user/cloudera/twitterdata.txt")
loadTwitterData: org.apache.spark.rdd.RDD[String] = /user/cloudera/twitterdata.txt MapPartitionsRDD[1] at textFile at <console>:27

scala> val WordCountspark= loadTwitterData.flatMap(line => line.split(" ")).map(word => (word,1)).reduceByKey(_ + _)
WordCountspark: org.apache.spark.rdd.RDD[(String, Int)] = ShuffledRDD[4] at reduceByKey at <console>:29

scala> val WordCountData= loadTwitterData.flatMap(line => line.split(" ")).map(word => (word,1)).reduceByKey(_ + _)
WordCountData: org.apache.spark.rdd.RDD[(String, Int)] = ShuffledRDD[7] at reduceByKey at <console>:29

scala> WordCountData.saveAsTextFile("/user/cloudera/sparkoutput.txt")

scala>
```

I have load the file data into loadTwitterData variable using sc.textFile function.

Then , I have use flatMap to split my data in the file using space as a delimiter and then use reduceByKey function to do my word count.

Then I have saved the Count in a text file sparkoutput.txt

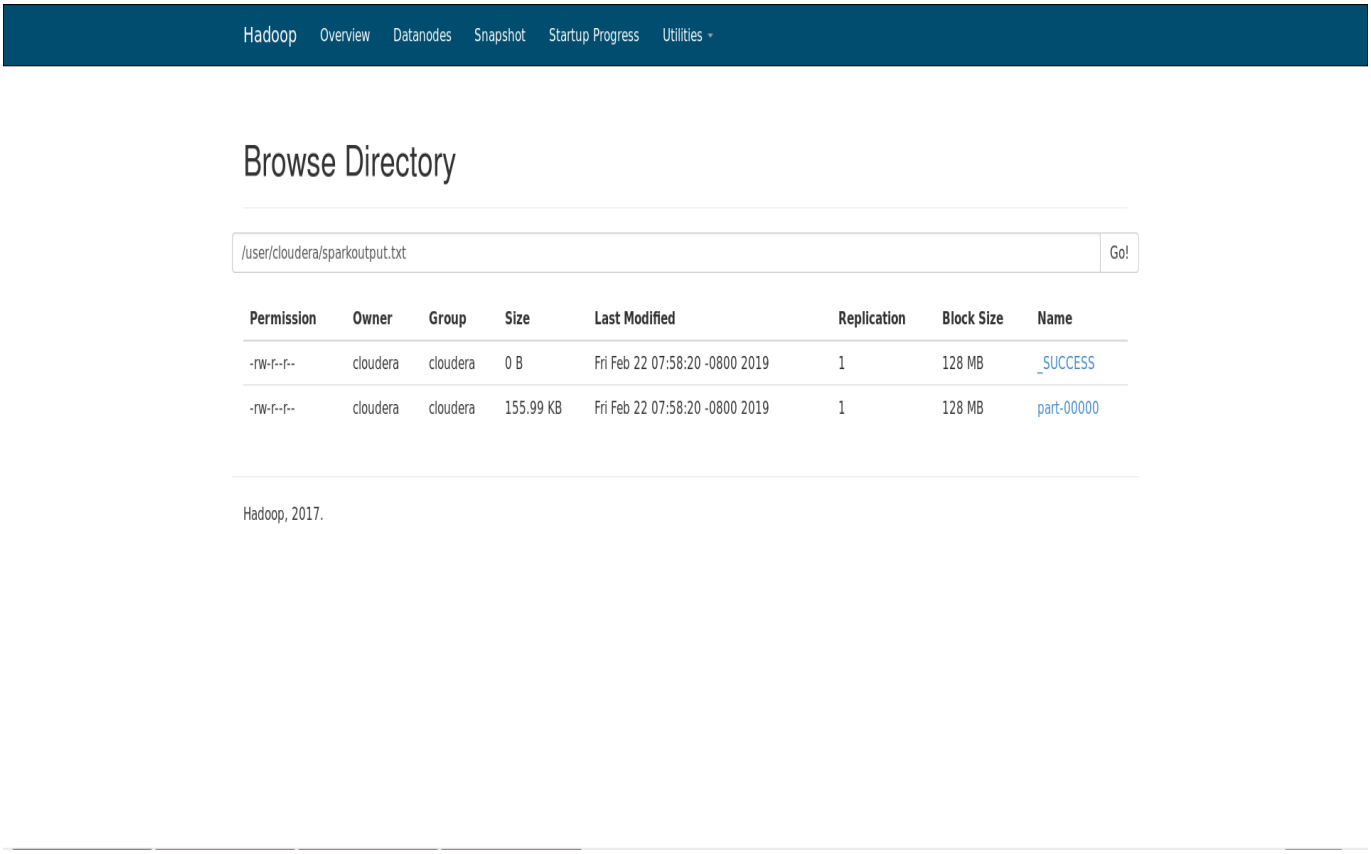
Step 3:




We have opened the directory of Hadoop where my output file is save. So you can see my output file name is sparkoutput.txt. It is present there.

**Step 5:**

Further, opening that file you can view two files one name is part-00000 and other is success file. Part-00000 file contains my output and success file is empty file which just tell me it has been made successfully.




**Word Count Implementation in Apache Hadoop:**

In order to run the word count program using Apache Hadoop we have written the code in java. In the program TokenizerMapper class is created by extending the Mapper class and the map function is implemented by overriding the map method in the Mapper class. The mapper function takes a key value pair as an input and outputs a key-values pair as an output.

And IntSumReducer class is created by extending the org.apache.hadoop.mapreduce.Reducer class and the reduce method is implemented by overriding the reduce method from the Reducer class. The reduce function collects all the intermediate key-value pairs generated by the multiple map functions and will sum up all the occurrences of each word and output a key-value pair for each word in the text document.

And also the main method is used to setup all necessary configurations and runs the mapreduce job.

Then the .jar file is exported to cloudera to execute the word count for Hashtags and URLs using Apache Hadoop.

**Below are the steps followed:**

1. A folder is created in HDFS and the input file is moved from local to HDFS using below commands.

- Hdfs dfs -mkdir /WordCountTutorial
- Hdfs dfs -mkdir /WordCountTutorial/Input

2. Upload the input file into the directory using the command.

- Hdfs dfs -put "/home/cloudera/WordCountTutorial/input_data/twitterhashtagsurlsdata.txt" /WordcountTutorial/Input

3. Created a sample class named Tutorial_classes in the local directory.

4.Changing the current directory to local directory.

5.We have written a Mapreduce code in java for wordcount and compiled using the command.

- Javac -classpath ${HADOOP_CLASSPATH} -d /home/cloudera/WordCountTutorial/Tutorial_classes

6.Put the output files in one jar file

- Jar -cvf wc.jar -c Tutorial_classes

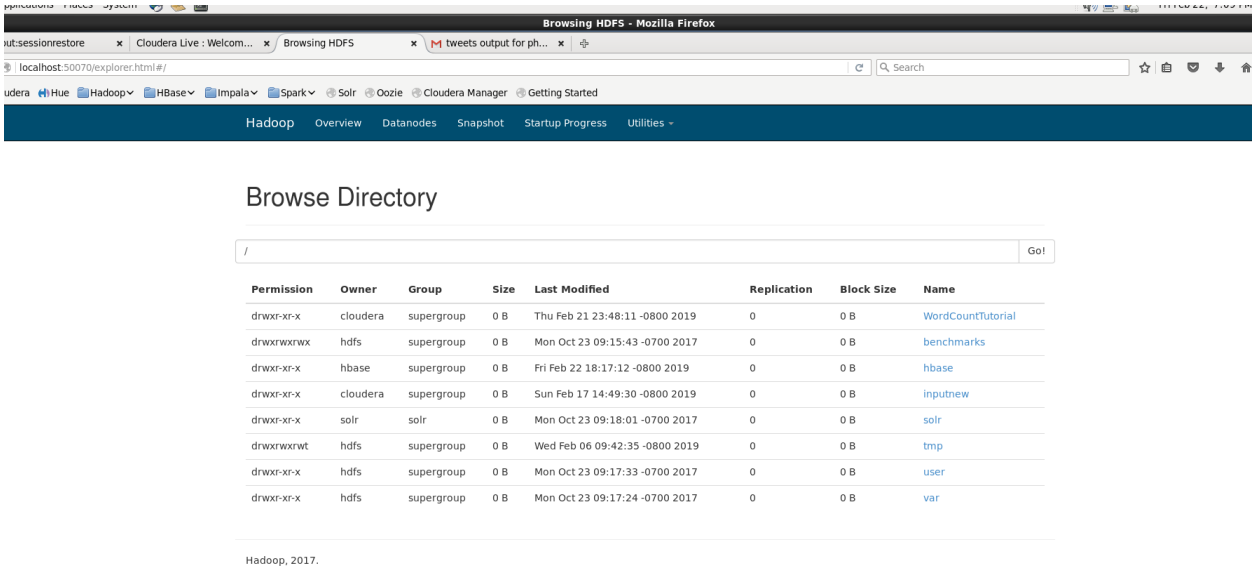
7.Now, run the jar file in Hadoop using below command

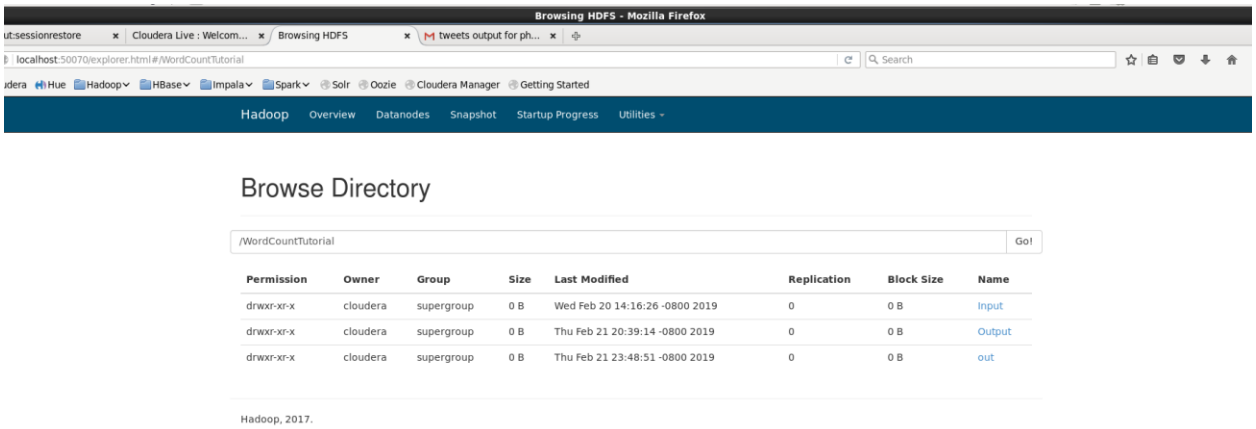
- Hadoop jar wc.jar WordCount /WordCountTutorial/Input /WordCountTutorial/Out

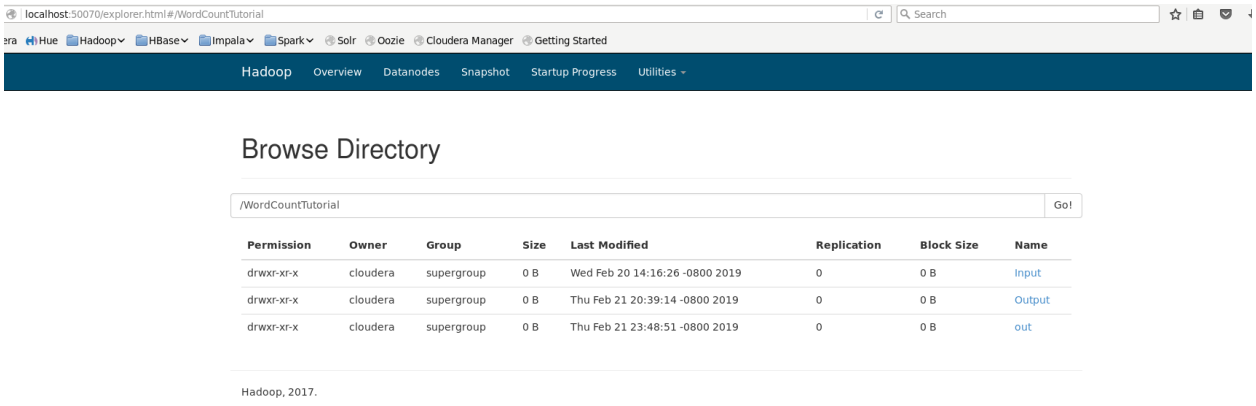
8. Now , checkout the output using command :

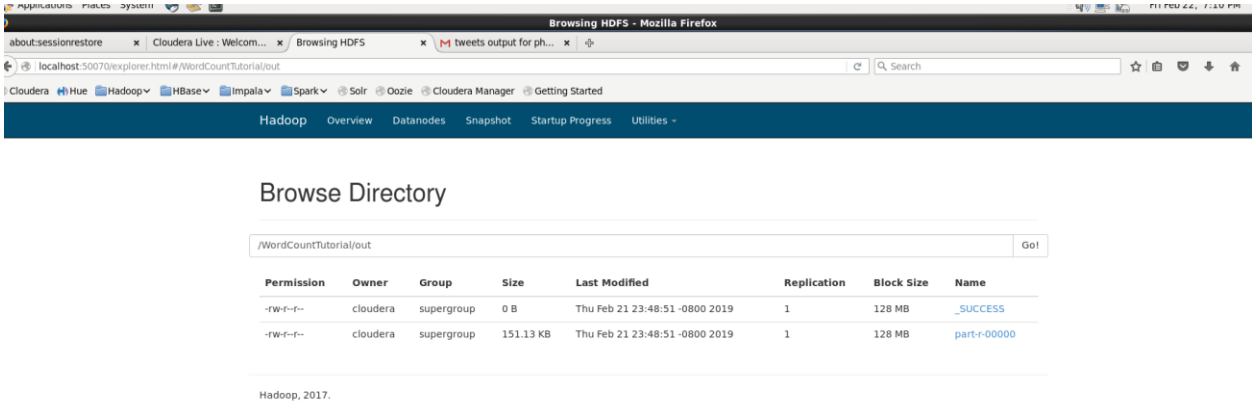
- Hadoop dfs -cat /WordCountTutorial/Out

## Results:


Browsing HDFS - Mozilla Firefox
Cloudera Live : Welcom...
Browsing HDFS
tweets output for ph...
localhost:50070/explorer.html#/
Search
Hue
Hadoop
HBase
Impala
Spark
Solr
Oozie
Cloudera Manager
Getting Started
Hadoop
Overview
Datanodes
Snapshot
Startup Progress
Utilities
Browse Directory
/
Go!
Permission
Owner
Group
Size
Last Modified
Replication
Block Size
Name
drwxr-xr-x cloudera supergroup 0 B Thu Feb 21 23:48:11 -0800 2019 0 0 B WordCountTutorial
drwxrwxrwx hdfs supergroup 0 B Mon Oct 23 09:15:43 -0700 2017 0 0 B benchmarks
drwxr-xr-x hbase supergroup 0 B Fri Feb 22 18:17:12 -0800 2019 0 0 B hbase
drwxr-xr-x cloudera supergroup 0 B Sun Feb 17 14:49:30 -0800 2019 0 0 B inputnew
drwxr-xr-x solr solr 0 B Mon Oct 23 09:18:01 -0700 2017 0 0 B solr
drwxrwxrwt hdfs supergroup 0 B Wed Feb 06 09:42:35 -0800 2019 0 0 B tmp
drwxr-xr-x hdfs supergroup 0 B Mon Oct 23 09:17:33 -0700 2017 0 0 B user
drwxr-xr-x hdfs supergroup 0 B Mon Oct 23 09:17:24 -0700 2017 0 0 B var
Hadoop, 2017.


Browsing HDFS - Mozilla Firefox
Cloudera Live : Welcom...
Browsing HDFS
tweets output for ph...
localhost:50070/explorer.html#/WordCountTutorial
Search
Hue
Hadoop
HBase
Impala
Spark
Solr
Oozie
Cloudera Manager
Getting Started
Hadoop
Overview
Datanodes
Snapshot
Startup Progress
Utilities
Browse Directory
/WordCountTutorial
Go!
Permission
Owner
Group
Size
Last Modified
Replication
Block Size
Name
drwxr-xr-x cloudera supergroup 0 B Wed Feb 20 14:16:26 -0800 2019 0 0 B Input
drwxr-xr-x cloudera supergroup 0 B Thu Feb 21 20:39:14 -0800 2019 0 0 B Output
drwxr-xr-x cloudera supergroup 0 B Thu Feb 21 23:48:51 -0800 2019 0 0 B out
Hadoop, 2017.


localhost:50070/explorer.html#/WordCountTutorial
Search
Hue
Hadoop
HBase
Impala
Spark
Solr
Oozie
Cloudera Manager
Getting Started
Hadoop
Overview
Datanodes
Snapshot
Startup Progress
Utilities
Browse Directory
/WordCountTutorial
Go!
Permission
Owner
Group
Size
Last Modified
Replication
Block Size
Name
drwxr-xr-x
cloudera
supergroup
0 B
Wed Feb 20 14:16:26 -0800 2019
0
0 B
Input
drwxr-xr-x
cloudera
supergroup
0 B
Thu Feb 21 20:39:14 -0800 2019
0
0 B
Output
drwxr-xr-x
cloudera
supergroup
0 B
Thu Feb 21 23:48:51 -0800 2019
0
0 B
out
Hadoop, 2017.


Browsing HDFS - Mozilla Firefox
about:sessionrestore
Cloudera Live : Welcom...
Browsing HDFS
tweets output for ph...
localhost:50070/explorer.html#/WordCountTutorial/out
Search
Cloudera
Hue
Hadoop
HBase
Impala
Spark
Solr
Oozie
Cloudera Manager
Getting Started
Hadoop
Overview
Datanodes
Snapshot
Startup Progress
Utilities
Browse Directory
/WordCountTutorial/out
Go!
Permission
Owner
Group
Size
Last Modified
Replication
Block Size
Name
-rw-r--r--
cloudera
supergroup
0 B
Thu Feb 21 23:48:51 -0800 2019
1
128 MB
_SUCCESS
-rw-r--r--
cloudera
supergroup
151.13 KB
Thu Feb 21 23:48:51 -0800 2019
1
128 MB
part-r-00000
Hadoop, 2017.