Practical 1

Aim: To install Hadoop framework, configure it and setup a single node cluster. Use web-based tools to monitor your Hadoop setup.

Hadoop Installation:

Step 1 – Download and Install Java

```
C:\Users\HP>java --version
java 10.0.2 2018-07-17
Java(TM) SE Runtime Environment 18.3 (build 10.0.2+13)
Java HotSpot(TM) 64-Bit Server VM 18.3 (build 10.0.2+13, mixed mode)

C:\Users\HP>
```

Step 2 – Download Hadoop and Extract it to a Folder

y computer › WINDOWS (C:) › hadoop

Name	Date modified	Type	Size
bin	12-10-2020 20:27	File folder	
data	12-10-2020 20:08	File folder	
etc	12-10-2020 19:52	File folder	
include	12-10-2020 19:48	File folder	
lib	12-10-2020 19:48	File folder	
libexec	12-10-2020 19:52	File folder	
logs	12-10-2020 23:10	File folder	
sbin	12-10-2020 19:48	File folder	
share	12-10-2020 19:48	File folder	
LICENSE.txt	21-07-2020 12:04	Text Document	144 KB
NOTICE.txt	21-07-2020 12:04	Text Document	22 KB
README.txt	21-07-2020 12:04	Text Document	2 KB

Step 3 – Setup System Environment Variables

Edit User Variable

Variable name: JAVA_HOME

Variable value: C:\Java\jdk-10.0.2\bin

Browse Directory... Browse File... OK Cancel

Step 4 - Now we need to edit some files located in the Hadoop directory of the etc folder

List of files need to be edited:

core – site.xml

```xml
<configuration>
   <property>
       <name>fs.defaultFS</name>
       <value>hdfs://localhost:9000</value>
   </property>
</configuration>
```

mapred – site

```xml
<configuration>
   <property>
       <name>mapreduce.framework.name</name>
       <value>yarn</value>
   </property>
</configuration>
```

Create a folder ‘data’ and other two subfolders ‘datanode’ and ‘namenode’ in hadoop directory.

Edit the file hdfs-site.xml and add below property in the configuration Note: The path of namenode and datanode across value would be the path of the datanode and namenode folders you just created.

Hdfs – site

```
<configuration>
   <property>
       <name>dfs.replication</name>
       <value>1</value>
   </property>
   <property>
       <name>dfs.namenode.name.dir</name>
       <value>C:\Users\hp\Downloads\hadoop-3.1.0\hadoop-3.1.0\data\namenode</value>
   </property>
   <property>
       <name>dfs.datanode.data.dir</name>
       <value> C:\Users\hp\Downloads\hadoop-3.1.0\hadoop-3.1.0\data\datanode</value>
   </property>
```

yarn – site

```
<configuration>
   <property>
        <name>yarn.nodemanager.aux-services</name>
        <value>mapreduce_shuffle</value>
   </property>
   <property>
        <name>yarn.nodemanager.auxservices.mapreduce.shuffle.class</name>
    <value>org.apache.hadoop.mapred.ShuffleHandler</value>
   </property>
</configuration>
```

Edit hadoop-env.cmd and replace %JAVA_HOME% with the path of the java jdkfolder

```
rem The only required environment variable is JAVA_HO
rem optional.  When running a distributed configurati
rem set JAVA_HOME in this file, so that it is correct
rem remote nodes.

rem The java implementation to use.  Required.
et JAVA_HOME=C:\Java\jdk-10.0.2

rem The jsvc implementation to use. Jsvc is required
rem set JSVC_HOME=%JSVC_HOME%
```

Step 5 – Hadoop needs windows OS specific files which does not come with default download of Hadoop

To include those files, replace the bin folder in Hadoop directory with the bin folder provided in this github link.

https://github.com/s911415/apache-hadoop-3.1.0-winutils

Download it as zip file. Extract it and copy the bin folder in it. If you want to save the old bin folder, rename it like bin_old and paste the copied bin folder in that directory.

omputer › WINDOWS (C:) › hadoop › bin

Name	Date modified	Type	Size
container-executor	09-08-2015 23:37	File	157 KB
hadoop	09-08-2015 23:37	File	7 KB
hadoop.cmd	09-08-2015 23:37	Windows Comma...	9 KB
hadoop.dll	09-08-2015 23:37	DLL File	84 KB
hadoop.exp	09-08-2015 23:37	Exports Library File	17 KB
hadoop.lib	09-08-2015 23:37	Object File Library	29 KB
hadoop.pdb	09-08-2015 23:37	Program Debug D...	475 KB
hdfs	09-08-2015 23:37	File	12 KB
hdfs.cmd	09-08-2015 23:37	Windows Comma...	7 KB
hdfs.dll	09-08-2015 23:37	DLL File	59 KB
hdfs.exp	09-08-2015 23:37	Exports Library File	9 KB
hdfs.lib	09-08-2015 23:37	Object File Library	15 KB
hdfs.pdb	09-08-2015 23:37	Program Debug D...	452 KB

Step 6 – Check whether Hadoop is successfully installed by running this command on cmd-

hadoop version

```
Microsoft Windows [Version 10.0.18363.1082]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\HP>hadoop version
Hadoop 3.1.4
Source code repository https://github.com/apache/hadoop.git -r 1e877761e8dadd71effef30e592368f7fe66a61b
Compiled by gabota on 2020-07-21T08:05Z
Compiled with protoc 2.5.0
From source with checksum 38405c63945c88fdf7a6fe391494799b
This command was run using /C:/hadoop/share/hadoop/common/hadoop-common-3.1.4.jar
```

Since it doesn't throw error and successfully shows the Hadoop version, that means Hadoop is successfully installed in the system.

Formatting the NameNode is done once when Hadoop is installed and not for running Hadoop file system, else it will delete all the data inside HDFS. Run this command-

hdfsnamenode -format

```
C:\Users\HP>hdfs namenode -format
2020-10-12 23:10:15,744 INFO namenode.NameNode: STARTUP_MSG:
/************************************************************
STARTUP_MSG: Starting NameNode
STARTUP_MSG:   host = LAPTOP-NIHET77E/192.168.32.1
STARTUP_MSG:   args = [-format]
STARTUP_MSG:   version = 3.1.4
STARTUP_MSG:   classpath = C:\hadoop\etc\hadoop;C:\hadoop\share\hadoo
b\animal-sniffer-annotations-1.17.jar;C:\hadoop\share\hadoop\common\l
```

Step 7 - Now change the directory in cmd to sbin folder of Hadoop directory with this command, (Note: Make sure you are writing the path as per your system)

```
C:\Users\HP>cd \hadoop

C:\hadoop>cd sbin

C:\hadoop\sbin>
```

Step 8 – Start namenode and datanode with this command –

start -dfs.cmd

Two more cmd windows will open for NameNode and DataNode Now start yarn through this command-

start -yarn.cmd

```
:\hadoop>cd sbin

:\hadoop\sbin>start-dfs.cmd

:\hadoop\sbin>start-yarn.cmd
starting yarn daemons

:\hadoop\sbin>
```

```
Apache Hadoop Distribution
        at org.apache.hadoop.util.DiskChecker.checkDirInternal(DiskChecker.java:141)
        at org.apache.hadoop.util.DiskChecker.checkDir(DiskChecker.java:116)
Apache Hadoop Distribution - hadoop  namenode
        at org.apache.hadoop.hdfs.server.namenode.NameNode.initialize(NameNode.java:743)
        at org.apache.hadoop.hdfs.server.namenode.NameNode.<init>(NameNode.java:966)
Apache Hadoop Distribution
        at com.google.inject.internal.InternalInjectorCreator.build(InternalInject
        at com.google.inject.Guice.createInjector(Guice.java:96)
        at com.google.inject.Guice.createInjector(Guice.java:73)
        at com.google.inject.Guice.createInjector(Guice.java:62)
        at org.apache.hadoop.yarn.webapp.WebApps$Builder.build(WebApps.java:387)
        at org.apache.hadoop.yarn.webapp.WebApps$Builder.start(WebApps.java:432)
        at org.apache.hadoop.yarn.server.resourcemanager.ResourceManager.startWepA
        at org.apache.hadoop.yarn.server.resourcemanager.ResourceManager.serviceSt
        at org.apache.hadoop.service.AbstractService.start(AbstractService.java:19
        at org.apache.hadoop.yarn.server.resourcemanager.ResourceManager.main(Reso
Caused by: java.lang.ClassNotFoundException: org.apache.hadoop.yarn.server.timelin
nager
        at java.base/jdk.internal.loader.BuiltinClassLoader.loadClass(BuiltinClass
        at java.base/jdk.internal.loader.ClassLoaders$AppClassLoader.loadClass(Cla
        at java.base/java.lang.ClassLoader.loadClass(ClassLoader.java:499)
```

Note: Make sure all the 4 Apache Hadoop Distribution windows are up running. If they are not running, you will see an error or a shutdown message. In that case, you need to debug the error

Step 9 - To access information about resource manager current jobs, successful and failed jobs, go to this link in browser-

http://localhost:8088/cluster

Step 10 - To check the details about the hdfs (namenode and datanode),

Open this link on browser-

http://localhost:9870/

Working with HDFS

1) We will be using a text file in local file system
2) We will create a directory name 'sample' in my hadoop directory using the cmd – hdfsdfs -mkdir /sample

```
C:\hadoop\sbin>hdfs dfs -mkdir /sample
mkdir: Call From LAPTOP-NIHET77E/192.168.32.1 to
or more details see:  http://wiki.apache.org/hado
```

3) To verify if the directory is created in hdfs, use 'ls' cmd which will list the files present in hdfs

```
hdfs dfs -ls /
```

```
C:\Users\hp\Downloads\hadoop-3.1.0\hadoop-3.1.0\sbin>hdfs dfs -ls /
Found 1 items
drwxr-xr-x   - hp supergroup          0 2019-04-07 23:39 /sample

C:\Users\hp\Downloads\hadoop-3.1.0\hadoop-3.1.0\sbin>
```

4) Copy the text file named 'potatoes' from local file system to the folder we just created in hdfs using copyFromLocalcmd –

```
hdfs dfs -copyFromLocal C:\Users\hp\Downloads\potatoes.txt /sample
```

```
C:\Users\hp\Downloads\hadoop-3.1.0\hadoop-3.1.0\sbin>hdfs dfs -copyFromLocal C:\Users\hp\Downloads\potatoes.txt /sample

C:\Users\hp\Downloads\hadoop-3.1.0\hadoop-3.1.0\sbin>
```

5) To verify if the file is copied to the folder, I will use 'ls' command by specifying the folder name which will read the list of files in that folder –

```
hdfs dfs -ls /sample
```

```
C:\Users\hp\Downloads\hadoop-3.1.0\hadoop-3.1.0\sbin>hdfs dfs -ls /sample
Found 1 items
-rw-r--r--   1 hp supergroup       3736 2019-04-07 23:39 /sample/potatoes.txt

C:\Users\hp\Downloads\hadoop-3.1.0\hadoop-3.1.0\sbin>
```

6) To view the contents of the file we copied, I will use cat command-

```
hdfs dfs -cat /sample/potatoes.txt
```

```
C:\Users\hp\Downloads\hadoop-3.1.0\hadoop-3.1.0\sbin>hdfs dfs -cat /sample/potatoes.txt
area    temp    size    storage method  texture flavor  moistness
1       1       1       1       1       2.9     3.2     3.0
1       1       1       1       2       2.3     2.5     2.6
1       1       1       1       3       2.5     2.8     2.8
1       1       1       1       4       2.1     2.9     2.4
1       1       1       1       5       1.9     2.8     2.2
1       1       1       2       1       1.8     3.0     1.7
1       1       1       2       2       2.6     3.1     2.4
1       1       1       2       3       3.0     3.0     2.9
1       1       1       2       4       2.2     3.2     2.5
1       1       1       2       5       2.0     2.8     1.9
1       1       1       3       1       1.8     2.6     1.5
1       1       1       3       2       2.0     2.8     1.9
1       1       1       3       3       2.6     2.6     2.6
1       1       1       3       4       2.1     3.2     2.1
1       1       1       3       5       2.5     3.0     2.1
1       1       1       4       1       2.6     3.1     2.4
```

7) To Copy file from hdfs to local directory, I will use get command-

```
hdfs dfs -get /sample/potatoes.txt C:\Users\hp\Desktop\priyanka
```

```
C:\Users\hp\Downloads\hadoop-3.1.0\hadoop-3.1.0\sbin>hdfs dfs -get /sample/potatoes.txt C:\Users\hp\Desktop\priyanka

C:\Users\hp\Downloads\hadoop-3.1.0\hadoop-3.1.0\sbin>
```

These were some basic Hadoop commands.

Practical 2

Aim: To implement file management tasks in Hadoop HDFS and perform Hadoop commands.

With growing data velocity, the data size easily outgrows the storage limit of a machine. A solution would be to store the data across a network of machines. Such filesystems are called *distributed filesystems*. Since data is stored across a network all the complications of a network come in.
This is where Hadoop comes in. It provides one of the most reliable filesystems. HDFS (Hadoop Distributed File System) is a unique design that provides storage for *extremely large files* with streaming data access pattern and it runs on *commodity hardware*. Let's elaborate the terms:

- ***Extremely large files***: Here we are talking about the data in range of petabytes (1000 TB).
- ***Streaming Data Access Pattern***: HDFS is designed on principle of *write-once and read-many-times*. Once data is written large portions of dataset can be processed any number times.
- ***Commodity hardware:*** Hardware that is inexpensive and easily available in the market. This is one of feature which specially distinguishes HDFS from other file system.

Syntax and Commands to Add, Retrieve and Delete Data From HDFS:

Adding Files and Directories to HDFS: Before you'll run Hadoop programs on data stored in HDFS, you 'll got to put the info into HDFS first. Let 's creates a directory and put a enter it. HDFS features a default working directory of /user/$USER, where $USER is your login user name. This directory isn't automatically created for you, though, so let 's creates it with the mkdir command. For the aim of illustration, we use chuck. you ought to substitute your user name within the example commands. Hadoop fs -mkdir /user/chuck

hadoop fs -put example.txt

hadoop fs -put example.txt /user/chuck

Retrieving Files from HDFS: The Hadoop command get copies files from HDFS back to the native filesystem. To retrieve example.txt, we are ready to run the next command:

hadoop fs -cat example.txt

Deleting Files from HDFS hadoop fs -rm example.txt. Command for creating a directory in hdfs is

"hdfsdfs –mkdir /lendicse".

Adding directory is done through the command "hdfsdfs –put lendi_english /".

Copying Data from NFS to HDFS:

- Copying from directory command is “hdfsdfs –copyFromLocal /home/lendi/Desktop/shakes/glossary /lendicse/”
- View the file by using the command “hdfsdfs –cat /lendi_english/glossary”
- Command for listing of things in Hadoop is “hdfsdfs –ls hdfs://localhost:9000/”. Command for Deleting files is “hdfsdfs –rm r /example”.

Hadoop Commands:

1) **ls:** This command is employed to list all the files. Use lsr for recursive approach. it's useful once we need a hierarchy of a folder.

   Syntax: bin/hdfsdfs -ls

   Example: bin/hdfsdfs -ls /

   It will print all the directories present in HDFS. bin directory contains executables so, bin/hdfs means we would like the executables of hdfs particularly dfs (Distributed File System) commands.

2) **mkdir**: To create a directory. In Hadoop dfs there's no home directory by default. So, let’s first create it.
   Syntax:
   bin/hdfsdfs -mkdir
   creating home directory:
   hdfs/bin -mkdir /user
   hdfs/bin -mkdir /user/username

Example:
bin/hdfsdfs -mkdir /geeks => '/' means absolute path
bin/hdfsdfs -mkdir geeks2 => Relative path -> the folder are going to be created relative tothe house directory.

```
suraj@suraj:~/hadoop-2.5.0-cdh5.3.2$ bin/hdfs dfs -mkdir /geeks
19/01/31 10:53:43 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applic
able
suraj@suraj:~/hadoop-2.5.0-cdh5.3.2$ bin/hdfs dfs -ls /
19/01/31 10:53:56 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applic
able
Found 5 items
drwxr-xr-x   - suraj supergroup          0 2019-01-31 10:53 /geeks
-rw-r--r--   1 suraj supergroup   13965969 2019-01-31 00:13 /input
drwxr-xr-x   - suraj supergroup          0 2019-01-31 01:30 /output
drwx------   - suraj supergroup          0 2019-01-31 00:15 /tmp
drwxr-xr-x   - suraj supergroup          0 2019-01-30 23:44 /user
```

```
suraj@suraj:~/hadoop-2.5.0-cdh5.3.2$ bin/hdfs dfs -mkdir geeks2
19/01/31 10:59:33 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applic
able
suraj@suraj:~/hadoop-2.5.0-cdh5.3.2$ bin/hdfs dfs -lsr /user
lsr: DEPRECATED: Please use 'ls -R' instead.
19/01/31 10:59:51 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applic
able
drwxr-xr-x   - suraj supergroup          0 2019-01-31 10:59 /user/suraj
-rw-r--r--   1 suraj supergroup      28093 2019-01-31 00:10 /user/suraj/AFINN-111.txt
drwxr-xr-x   - suraj supergroup          0 2019-01-31 10:59 /user/suraj/geeks2
drwxr-xr-x   - suraj supergroup          0 2019-01-30 23:47 /user/suraj/insideUserChecking
suraj@suraj:~/hadoop-2.5.0-cdh5.3.2$
```

3) **touchz:**It creates an empty file.
Syntax:
bin/hdfsdfs -touchz
Example:
bin/hdfsdfs -touchz /geeks/myfile.txt

```
suraj@suraj:~/hadoop-2.5.0-cdh5.3.2$ bin/hdfs dfs -touchz /geeks/myfile.txt
19/01/31 11:10:31 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applic
able
suraj@suraj:~/hadoop-2.5.0-cdh5.3.2$ bin/hdfs dfs -lsr /geeks
lsr: DEPRECATED: Please use 'ls -R' instead.
19/01/31 11:10:48 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applic
able
-rw-r--r--   1 suraj supergroup          0 2019-01-31 11:10 /geeks/myfile.txt
suraj@suraj:~/hadoop-2.5.0-cdh5.3.2$
```

4) **copyFromLocal (or) put:** To copy files/folders from local filing system to hdfs store. this is often the foremost important command. Local filesystem means the files present on the OS.
Syntax:
bin/hdfsdfs -copyFromLocal
Example: Let's suppose we've a file AI.txt on Desktop which we would like to repeat to folder geeks present on hdfs.

bin/hdfsdfs -copyFromLocal ../Desktop/AI.txt /geeks

```
suraj@suraj:~/hadoop-2.5.0-cdh5.3.2$ bin/hdfs dfs -put ../Desktop/AI.txt /geeks
19/01/31 11:31:02 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applic
able
suraj@suraj:~/hadoop-2.5.0-cdh5.3.2$ bin/hdfs dfs -lsr /geeks
lsr: DEPRECATED: Please use 'ls -R' instead.
19/01/31 11:31:21 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applic
able
-rw-r--r--   1 suraj supergroup        205 2019-01-31 11:31 /geeks/AI.txt
-rw-r--r--   1 suraj supergroup          0 2019-01-31 11:10 /geeks/myfile.txt
suraj@suraj:~/hadoop-2.5.0-cdh5.3.2$
```

5) **cat:** To print file contents.
Syntax:

bin/hdfsdfs -cat <path>

Example:

// print the content of AI.txt present

// inside geeks folder.

bin/hdfsdfs -cat /geeks/AI.txt ->

```
suraj@suraj:~/hadoop-2.5.0-cdh5.3.2$ bin/hdfs dfs -cat /geeks/AI.txt
19/01/31 11:33:25 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applic
able
In computer science, artificial intelligence, sometimes called machine intelligence, is intelligence demonstrated by machines, in contrast to t
he natural intelligence displayed by humans and other animals
suraj@suraj:~/hadoop-2.5.0-cdh5.3.2$
```

6) **copyToLocal (or) get:** To copy files/folders from hdfs store to local file system.
Syntax:

bin/hdfsdfs -copyToLocal<<srcfile(on hdfs)><local file dest>

Example:

bin/hdfsdfs -copyToLocal /geeks ../Desktop/hero

(OR)

bin/hdfsdfs -get /geeks/myfile.txt ../Desktop/hero

myfile.txt from geeks folder will be copied to folder hero present on Desktop.

```
suraj@suraj:~/hadoop-2.5.0-cdh5.3.2$ bin/hdfs dfs -get /geeks/myfile.txt  ../Desktop/hero
19/01/31 11:43:34 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applic
able
suraj@suraj:~/hadoop-2.5.0-cdh5.3.2$ ls ../Desktop/hero
myfile.txt
suraj@suraj:~/hadoop-2.5.0-cdh5.3.2$
```

7) **moveFromLocal:** This command will move file from local to hdfs.
Syntax:

bin/hdfsdfs -moveFromLocal<local src><dest(on hdfs)>

Example:

bin/hdfsdfs -moveFromLocal ../Desktop/cutAndPaste.txt /geeks

```
suraj@suraj:~/hadoop-2.5.0-cdh5.3.2$ bin/hdfs dfs -moveFromLocal ../Desktop/cutAndPaste.txt /geeks
19/01/31 12:38:38 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applic
able
suraj@suraj:~/hadoop-2.5.0-cdh5.3.2$ bin/hdfs dfs -ls /geeks
19/01/31 12:38:56 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applic
able
Found 3 items
-rw-r--r--   1 suraj supergroup        205 2019-01-31 11:31 /geeks/AI.txt
-rw-r--r--   1 suraj supergroup         96 2019-01-31 12:38 /geeks/cutAndPaste.txt
-rw-r--r--   1 suraj supergroup          0 2019-01-31 11:10 /geeks/myfile.txt
suraj@suraj:~/hadoop-2.5.0-cdh5.3.2$ ls ../Desktop
AFINN-111.txt  Deploy_hadoop_on_a_single_node_cluster_v02 (1).pdf  hero                  sentiment.jar
AI.txt         FlumeData.1440939532959                             json-simple-1.1.1.jar  worldBank
suraj@suraj:~/hadoop-2.5.0-cdh5.3.2$
```

8) **cp:** This command is used to copy files within hdfs. Lets copy folder *geeks* to *geeks_copied*.
Syntax:

bin/hdfsdfs -cp <src(on hdfs)><dest(on hdfs)>

Example:

bin/hdfs -cp /geeks /geeks_copied

```
suraj@suraj:~/hadoop-2.5.0-cdh5.3.2$ bin/hdfs dfs -mkdir /geeks_copied
19/01/31 12:46:03 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applic
able
suraj@suraj:~/hadoop-2.5.0-cdh5.3.2$ bin/hdfs dfs -cp /geeks  /geeks_copied
19/01/31 12:46:26 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applic
able
suraj@suraj:~/hadoop-2.5.0-cdh5.3.2$ bin/hdfs dfs -ls /geeks_copied
19/01/31 12:47:10 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applic
able
Found 1 items
drwxr-xr-x   - suraj supergroup          0 2019-01-31 12:46 /geeks_copied/geeks
suraj@suraj:~/hadoop-2.5.0-cdh5.3.2$
```

9) **mv:** This command is used to move files within hdfs. Lets cut-paste a file *myfile.txt* from *geeks* folder to *geeks_copied*.
Syntax:

bin/hdfsdfs -mv <src(on hdfs)><src(on hdfs)>

Example:

bin/hdfs -mv /geeks/myfile.txt /geeks_copied

```
suraj@suraj:~/hadoop-2.5.0-cdh5.3.2$ bin/hdfs dfs -ls /geeks
19/01/31 12:48:42 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applic
able
Found 3 items
-rw-r--r--   1 suraj supergroup        205 2019-01-31 11:31 /geeks/AI.txt
-rw-r--r--   1 suraj supergroup         96 2019-01-31 12:38 /geeks/cutAndPaste.txt
-rw-r--r--   1 suraj supergroup          0 2019-01-31 11:10 /geeks/myfile.txt
suraj@suraj:~/hadoop-2.5.0-cdh5.3.2$ bin/hdfs dfs -mv /geeks/myfile.txt  /geeks_copied
19/01/31 12:49:27 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applic
able
suraj@suraj:~/hadoop-2.5.0-cdh5.3.2$ bin/hdfs dfs -ls /geeks
19/01/31 12:49:42 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applic
able
Found 2 items
-rw-r--r--   1 suraj supergroup        205 2019-01-31 11:31 /geeks/AI.txt
-rw-r--r--   1 suraj supergroup         96 2019-01-31 12:38 /geeks/cutAndPaste.txt
suraj@suraj:~/hadoop-2.5.0-cdh5.3.2$ bin/hdfs dfs -ls /geeks_copied
19/01/31 12:50:00 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applic
able
Found 2 items
drwxr-xr-x   - suraj supergroup          0 2019-01-31 12:46 /geeks_copied/geeks
-rw-r--r--   1 suraj supergroup          0 2019-01-31 11:10 /geeks_copied/myfile.txt
suraj@suraj:~/hadoop-2.5.0-cdh5.3.2$
```

10) **rmr:** This command deletes a file from HDFS *recursively*. It is very useful command when you want to delete a *non-empty directory*.
Syntax:

bin/hdfsdfs -rmr<filename/directoryName>

Example:

bin/hdfsdfs -rmr /geeks_copied -> It will delete all the content inside the directory then the directory itself.

```
suraj@suraj:~/hadoop-2.5.0-cdh5.3.2$ bin/hdfs dfs -ls /geeks_copied
19/01/31 12:58:16 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applic
able
Found 2 items
drwxr-xr-x   - suraj supergroup          0 2019-01-31 12:46 /geeks_copied/geeks
-rw-r--r--   1 suraj supergroup          0 2019-01-31 11:10 /geeks_copied/myfile.txt
suraj@suraj:~/hadoop-2.5.0-cdh5.3.2$ bin/hdfs dfs -rmr /geeks_copied
rmr: DEPRECATED: Please use 'rm -r' instead.
19/01/31 12:58:40 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applic
able
19/01/31 12:58:42 INFO fs.TrashPolicyDefault: Namenode trash configuration: Deletion interval = 0 minutes, Emptier interval = 0 minutes.
Deleted /geeks_copied
suraj@suraj:~/hadoop-2.5.0-cdh5.3.2$ bin/hdfs dfs -ls /geeks_copied
19/01/31 12:59:35 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applic
able
ls: `/geeks_copied': No such file or directory
suraj@suraj:~/hadoop-2.5.0-cdh5.3.2$
```

Practical 3

Aim: To implement basic functions and commands in R Programming. To build Word Cloud, a text mining method using R for easy to understand and better visualization than a data table.

Text Mining is one of the most critical ways of analysing and processing unstructured data which forms nearly 80% of the world's data. Today most organizations and institutions gather and store massive amounts of data in data warehouses, and cloud platforms and this data continues to grow exponentially by the minute as new data comes pouring in from multiple sources.

As a result, it becomes a challenge for companies and organizations to store, process, and analyse vast amounts of textual data with traditional tools. This is where text mining applications, text mining tools, and text mining techniques come in.

Text mining incorporates and integrates the tools of information retrieval, data mining, machine learning, statistics, and computational linguistics, and hence, it is nothing short of a multidisciplinary field. Text mining deals with natural language texts either stored in semi-structured or unstructured formats.

There are 5 major steps to create word clouds in R

Step 1: Create a text file

In the following examples, we will process the "**I have a dream speech**" from "**Martin Luther King**" but you can use any text you want:

- Copy and paste the text in a plain text file (e.g.: ml.txt)
- Save the file

Step 2: Install and Load the required packages.

```
# Install
install.packages("tm")  # for text mining
install.packages("SnowballC") # for text stemming
install.packages("wordcloud") # word-cloud generator
install.packages("RColorBrewer") # color palettes
# Load
library("tm")
library("SnowballC")
library("wordcloud")
library("RColorBrewer")
```

Step 3: Text Mining

Load the text

The text is loaded using **Corpus ()** function from **text mining** (tm) package. Corpus is a list of a document (in our case, we only have one document).

1. **We start by importing the text file created in Step 1**

To import the file saved locally in your computer, type the following R code. You will be asked to choose the text file interactively.

```
text <- readLines(file.choose())
```

In the example below, I'll load a .txt file hosted on STHDA website:

```
# Read the text file from internet
filePath <- "http://www.sthda.com/sthda/RDoc/example-files/martin-luther-king-i-have-a-dream-speech.txt"
text <- readLines(filePath)
```

2. Load the data as a corpus.

```
# Load the data as a corpus
docs <- Corpus(VectorSource(text))
```

3. Inspect the content of the document.

```
inspect(docs)
```

Text Transformation

Transformation is performed using **tm_map()** function to replace, for example, special characters from the text.
Replacing "/", "@" and "|" with space:

```
toSpace <- content_transformer(function (x , pattern ) gsub(pattern, " ", x))
docs <- tm_map(docs, toSpace, "/")
docs <- tm_map(docs, toSpace, "@")
docs <- tm_map(docs, toSpace, "\\|")
```

Cleaning the text

the **tm_map()** function is used to remove unnecessary white space, to convert the text to lower case, to remove common stop words like 'the', "we".
The information value of 'stop words' is near zero due to the fact that they are so common in a language. Removing this kind of words is useful before further analyses. For 'stop words', supported languages are danish, Dutch, English, finnish, French, German, Hungarian, Italian, Norwegian, Portuguese, Russian, Spanish and Swedish. Language names are case sensitive.

Another important pre-processing step is to make a **text stemming** which reduces words to their root form. In other words, this process removes suffixes from words to make it simple and to get the common origin. For example, a stemming process reduces the words "moving", "moved" and "movement" to the root word, "move".

```
# Convert the text to lower case
docs <- tm_map(docs, content_transformer(tolower))
# Remove numbers
docs <- tm_map(docs, removeNumbers)
# Remove english common stopwords
docs <- tm_map(docs, removeWords, stopwords("english"))
# Remove your own stop word
# specify your stopwords as a character vector
docs <- tm_map(docs, removeWords, c("blabla1", "blabla2"))
# Remove punctuations
docs <- tm_map(docs, removePunctuation)
# Eliminate extra white spaces
docs <- tm_map(docs, stripWhitespace)
# Text stemming
# docs <- tm_map(docs, stemDocument)
```

Step 4: Build a term – document matrix

Document matrix is a table containing the frequency of the words. Column names are words and row names are documents. The function *TermDocumentMatrix()* from **text mining** package can be used as follow :

```
dtm <- TermDocumentMatrix(docs)
m <- as.matrix(dtm)
v <- sort(rowSums(m),decreasing=TRUE)
d <- data.frame(word = names(v),freq=v)
head(d, 10)
```

```
             word freq
will         will   17
freedom   freedom   13
ring         ring   12
day           day   11
dream       dream   11
let           let   11
every       every    9
able         able    8
one           one    8
together together    7
```

Step 5: Generate the word cloud

```
set.seed(1234)
wordcloud(words = d$word, freq = d$freq, min.freq = 1,
          max.words=200, random.order=FALSE, rot.per=0.35,
          colors=brewer.pal(8, "Dark2"))
```

The above **word cloud** clearly shows that “Will”, “freedom”, “dream”, “day” and “together” are the five most important words in the “**I have a dream speech**” from **Martin Luther King**.

Practical 4

Aim: To implement a word count application using the MapReduceprogramming model.

Wordcount may be a simple program which counts the number of occurrences of every word during a given text input file set.

Wordcount fits alright with the MapReduce programming model making it an excellent example to know the Hadoop Map/Reduce programming style. Our implementation consists of three main parts:

1. Mapper 2. Reducer 3. Driver

Step 1: Write a Mapper

A Mapper overrides the “map” function from the category "org.apache.hadoop.mapreduce.Mapper" which provides pairs because the input. A Mapper implementation may output pairs using the provided Context.

Input value of the Wordcount Map task are going to be a line of text from the input file and therefore the key would be the road number. Map task outputs for every word within the line of text.

```
void Map (key, value)
{
 for each word x in value: output.collect(x, 1);
 }
```

Step 2: Write a Reducer

A Reducer collects the intermediate output from multiple map tasks and assemble one result. Here, the Wordcount program will sum up the occurrence of every word to pairs as.

```
void Reduce (keyword, <list of values>)
{
 for each x in: sum+=x; final_output.collect(keyword, sum);
}
```

Step 3: Write Driver

The Driver program configures and run the MapReduce job. We use the most program to perform basic configurations such as:

- Job Name: name of this Job
- Executable (Jar) Class: the most executable class. For here, WordCount.
- Mapper Class: class which overrides the "map" function. For here, Map.
- Reducer: class which override the "reduce" function. For here, Reduce.

- Output Key: sort of output key. For here, Text.
- Output Value: sort of output value. For here, IntWritable.
- File Input Path
- File Output Path

```
lendi@ubuntu: ~/Desktop
16/08/17 01:17:45 INFO impl.YarnClientImpl: Submitted application application_14
71410736896_0001
16/08/17 01:17:45 INFO mapreduce.Job: The url to track the job: http://ubuntu.ub
untu-domain:8088/proxy/application_1471410736896_0001/
16/08/17 01:17:45 INFO mapreduce.Job: Running job: job_1471410736896_0001
16/08/17 01:17:52 INFO mapreduce.Job: Job job_1471410736896_0001 running in uber
 mode : false
16/08/17 01:17:52 INFO mapreduce.Job:  map 0% reduce 0%
16/08/17 01:17:59 INFO mapreduce.Job:  map 100% reduce 0%
16/08/17 01:18:06 INFO mapreduce.Job:  map 100% reduce 100%
16/08/17 01:18:06 INFO mapreduce.Job: Job job_1471410736896_0001 completed succe
ssfully
16/08/17 01:18:06 INFO mapreduce.Job: Counters: 49
        File System Counters
                FILE: Number of bytes read=3772644
                FILE: Number of bytes written=7775215
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=1764718
                HDFS: Number of bytes written=510970
                HDFS: Number of read operations=6
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
```

```
part-r-00000(3) (~/Downloads) - gedit
Open  Save  Undo
part-r-00000(3)
2.      1
3       28
3.      1
4.      1
5.      1
6.      1
7.      1
8.      1
9.      1
A       1012
A'      2
ADRIAN, 2
AEdiles,        1
AEsculapius?    1
ALARBUS,        1
ALENCON,        2
ALL'S   25
ANDRONICUS,     1
ANGELO, 2
```

PRACTICAL 5

Aim: To design and implement MapReduce algorithms to take a very large file of integers and produce as output:

a) The largest integer

b) The average of all the integers.

c) The same set of integers, but with each integer appearing only once.

d) The count of the number of distinct integers in the input.

Implementation:

1. Largest Number

map(file_id,iterator numbers){

```
max=INTEGER.MIN_VALUE
while(numbers.hasNext()):
     num=numbers.next()
     if(num>max):
        max=num
end while
emit('max',max)
```

}

reduce(key, iterator max_values){

```
max=INTEGER.MIN_VALUE
while(max_values.hasNext()):
     num=max_values.next()
     if(num>max):
        max=num
end while
emit('overall_max',max)
```

}

We are given an inventory of files and every file has list of numbers. In MapReduce, each node parallelly picks a file and executes map function by passing file_num as key and list of integers within the file as iterator. map function then finds the utmost therein file. then the map function maps the utmost of that file with the key 'max' and emits to the map-reduce framework which distributes the key-value pair to the network of nodes reduce function recieves single key 'max' and an inventory of maximum values. the weather of max_values are maximum of every file. Then reduce function finds the utmost among the max_values.

2. Average :

map(file_id,iterator numbers){

```
sum=0
count=0
while(numbers.hasNext()):
     num=numbers.next()
     sum+=num
     count+=1
end while
emit('avg',(sum,count))
```

}

1. here the output is a tuple of sum and count as (sum,count)

reduce(key, iterator sum_count_tuples){

```
sum=0
count=0
while(sum_count_tuples.hasNext()):
     sum_i,count_i=sum_count_tuples.next()
     sum=sum+sum_i
     count=count+count_i
end while
emit('overall_avg',(sum/count))
```

}

3.Same set of integers:

The same set of integers, but with each integer appearing just one occasion

Here we will just use the key as a filter:

Map: Take inputs and for every input return key=input, value=input.

Reduce: The list of keys is that the desired output.

4. Distinct:

map(file_id,iterator numbers){

```
while(numbers.hasNext()):
     num=numbers.next()
     emit(num,1)
end while
```

}

1. in map-reduce , emit is not return operation , it emits the key,value to network, so it can be inside loop
2. here number is the key

reduce(uniq_num, iterator values){

```
emit(uniq_num,1)
```

}here the values is list of 1s as [1,1,1,...1] for each unique number. so the emit in reduce produces 1 for each unique number. uniq_num is key.

Practical 6

Aim: To implement basic CRUD operations (create, read, update, delete) in MongoDB and Cassandra.

Atlas-managed MongoDB deployments, or "clusters", can be either a replica set or a sharded cluster. This tutorial walks you through creating a replica set.

To implement CRUD operation in MongoDB, we need to create a cluster. To create cluster, follow the below steps…

1. Navigate to the Clusters page for your project
2. Open the Create New Cluster Dialog
3. Select a cluster type
4. Select your preferred Cloud Provider & Region
5. Select the Cluster Tier
6. Select any Additional Settings
7. Enter the Cluster Name
8. Deploy your Cluster

To access a cluster, we must create a MongoDB User with access to the desired database(s) on your Atlas cluster. MongoDB users are separate from Atlas users. MongoDB users have access to MongoDB databases, while Atlas users have access to the Atlas application itself.

To connect dialog to Connect to our cluster

1. Open the Connect dialog
2. Choose how you want to limit connections to your cluster
3. Create a Database User
4. Select your preferred connection method

5. Now, we can access the database using PowerShell.

```
import datetime
persondocument = {
    "name": {"first": "Alan", "last": "Turing"},
    "birth": datetime.datetime(1912, 6, 23),
    "death": datetime.datetime(1954, 6, 7),
    "contribs": ["Turing Machine", "Turing Test", "Turingery"],
    "views": 125000
```

```
C:\Users\HP>pip install pymongo
Requirement already satisfied: pymongo in c:\users\hp\appdata\local\programs\python\python36\lib\site-packages
You are using pip version 9.0.1, however version 20.2.3 is available.
You should consider upgrading via the 'python -m pip install --upgrade pip' command.
```

```
>>> from pymongo import MongoClient
>>> client = MongoClient('mongodb://kamal:kamal@cluster0-shard-00-00.02x61.gcp.mongodb.net:27017,cluster0-shard-00-01.0
2x61.gcp.mongodb.net:27017/<dbname>?ssl=true&replicaSet=atlas-xueogh-shard-0&authSource=admin&retryWrites=true&w=majori
>>> db = client.gettingStarted
>>> people = db.people
>>> import datetime
>>> personDocument = {
...   "name": { "first": "Alan", "last": "Turing" },
...   "birth": datetime.datetime(1912, 6, 23),
...   "death": datetime.datetime(1954, 6, 7),
...   "contribs": [ "Turing machine", "Turing test", "Turingery" ],
...   "views": 1250000
... }
>>> import datetime
>>> personDocument = {
...   "name": { "first": "Kamal", "last": "Kakadiya" },
...   "birth": datetime.datetime(1912, 6, 23),
...   "death": datetime.datetime(1954, 6, 7),
...   "contribs": [ "Turing machine", "Turing test", "Turingery" ],
...   "views": 1250000
... }
>>> people.insert_one(personDocument)
```

Practical 7

Aim: To develop a MapReduce application and implement a program that analyses weather data.

Implementation:

Step 1: Download the weather dataset

26494	20200101	2.424	-147.51	64.97	-18.8	-21.8	-20.3	-19.8	2.5	0.00
26494	20200102	2.424	-147.51	64.97	-19.1	-23.4	-21.3	-21.2	0.0	0.00
26494	20200103	2.424	-147.51	64.97	-19.0	-25.4	-22.2	-22.1	0.2	0.00
26494	20200104	2.424	-147.51	64.97	-18.4	-26.8	-22.6	-23.2	0.0	0.00
26494	20200105	2.424	-147.51	64.97	-26.0	-28.8	-27.4	-27.3	0.4	0.02
26494	20200106	2.424	-147.51	64.97	-26.0	-30.0	-28.0	-28.0	0.4	0.00
26494	20200107	2.424	-147.51	64.97	-27.8	-31.4	-29.6	-29.4	0.3	0.00
26494	20200108	2.424	-147.51	64.97	-20.9	-33.6	-27.3	-30.7	0.0	0.00
26494	20200109	2.424	-147.51	64.97	-17.3	-26.6	-22.0	-20.8	0.0	0.00
26494	20200110	2.424	-147.51	64.97	-15.6	-24.3	-19.9	-20.0	0.0	0.00
26494	20200111	2.424	-147.51	64.97	-15.3	-25.0	-20.2	-20.1	0.0	0.00
26494	20200112	2.424	-147.51	64.97	-17.8	-22.4	-20.1	-20.1	0.0	0.00

Columns 6 and 7 is showing Maximum and minimum temperature.

Step 2: Make a project in Eclipse

Code:

```java
# import necesary libraries
import java.io.IOException;
import java.util.Iterator;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.conf.Configuration;
public class MyMaxMin {
    // Mapper
    /*MaxTemperatureMapper class is static
    * and extends Mapper abstract class
    * having four Hadoop generics type
    * LongWritable, Text, Text, Text.
    */
    public static class MaxTemperatureMapper extends            Mapper
        <LongWritable, Text, Text, Text> {
        /**
        * @method map
        * This method takes the input as a text data type.
        * Now leaving the first five tokens, it takes
        * 6th token is taken as temp_max and
```

```java
        * temp_max > 30 and temp_min < 15 are
        * passed to the reducer.
        */
    // the data in our data set with
    // this value is inconsistent data
    public static final int MISSING = 9999;
    @Override
        public void map(LongWritable arg0, Text Value, Context conte
                throws IOException, InterruptedException {
        // Convert the single row(Record) to
        // String and store it in String
        // variable name line
        String line = Value.toString();
            // Check for the empty line
            if (!(line.length() == 0)) {
                // from character 6 to 14 we have
                // the date in our dataset
                String date = line.substring(6, 14);
                // similarly we have taken the maximum
                // temperature from 39 to 45 characters
mport java.io.IOException;
mport java.util.Iterator;
mport org.apache.hadoop.fs.Path;
mport org.apache.hadoop.io.LongWritable;
mport org.apache.hadoop.io.Text;
mport org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
mport org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
mport org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;
mport org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
```

```java
public static void main(String[] args) throws Exception {
    // reads the default configuration of the
    // cluster from the configuration XML files
    Configuration conf = new Configuration();
    // Initializing the job with the
    // default configuration of the cluster
    Job job = new Job(conf, "weather example");
    // Assigning the driver class name
    job.setJarByClass(MyMaxMin.class);
    // Key type coming out of mapper
    job.setMapOutputKeyClass(Text.class);

    // value type coming out of mapper
    job.setMapOutputValueClass(Text.class);
    // Defining the mapper class name
    job.setMapperClass(MaxTemperatureMapper.class);
    // Defining the reducer class name
    job.setReducerClass(MaxTemperatureReducer.class);
    // Defining input Format class which is
    // responsible to parse the dataset
    // into a key value pair
    job.setInputFormatClass(TextInputFormat.class);
    // Defining output Format class which is
    // responsible to parse the dataset
    // into a key value pair
    job.setOutputFormatClass(TextOutputFormat.class);
    // setting the second argument
```

Now, export the project as jar file, and choose main class as Mymaxmin

Start our Hadoop Daemons

```
start-dfs.sh
```

```
start-yarn.sh
```

Move your dataset to Hadoop HDFS

Syntax:

```
hdfs dfs -put /file_path /destination
```

In below command / shows the root directory of our HDFS.

```
hdfs dfs -put /home/dikshant/Downloads/CRND0103-2020-AK_Fairbanks_11_NE.txt /
```

Check the file sent to our HDFS.

```
hdfs dfs -ls /
```

Run the jar file with below command and produce the output ion MyOutput File

Syntax:

```
hadoop jar /jar_file_location /dataset_location_in_HDFS /output-file_name
```

Command:

```
hadoop jar /home/dikshant/Documents/Project.jar /CRND0103-2020-AK_Fairbanks_11_NE.txt /M
```

Switch to localhost:50070/, under utilities, pick Search the file system and download part-r-00000 from the / MyOutput directory to see the outcome.

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rw-r--r--	dikshant	supergroup	38.78 KB	Jul 04 09:39	1	122.07 MB	CRND0103-2020-AK_Fairbanks_11_NE.txt
drwxrwxr-x+	dikshant	supergroup	0 B	Jun 23 14:23	0	0 B	Hadoop_File
drwxr-xr-x	dikshant	supergroup	0 B	Jul 04 09:44	0	0 B	MyOutput
drwxrwxrwx	dikshant	supergroup	0 B	Jun 14 21:43	0	0 B	tmp
drwxr-xr-x	dikshant	supergroup	0 B	Jun 14 21:43	0	0 B	user

/MyOutput Go!

Show 25 entries Search:

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rw-r--r--	dikshant	supergroup	0 B	Jul 04 09:44	1	122.07 MB	_SUCCESS
-rw-r--r--	dikshant	supergroup	3.85 KB	Jul 04 09:44	1	122.07 MB	part-r-00000

Result in the downloaded File.

```
The Day is Cold Day :20200101   -21.8
The Day is Cold Day :20200102   -23.4
The Day is Cold Day :20200103   -25.4
The Day is Cold Day :20200104   -26.8
The Day is Cold Day :20200105   -28.8
The Day is Cold Day :20200106   -30.0
The Day is Cold Day :20200107   -31.4
The Day is Cold Day :20200108   -33.6
The Day is Cold Day :20200109   -26.6
The Day is Cold Day :20200110   -24.3
```

In the picture above, you can see the top 10 findings representing the cold days. The second column is a day in Yyy / mm / dd format. For eg, 20200101 shall mean

```
year = 2020
month = 01
Date = 01
```

Practical 8

Aim: To install and run HIVE. Use Hive to create, alter, and drop databases, tables, views, functions, and indexes. To create HDFS tables and load them in HIVE and implement joining of table in HIVE.

HIVE

Apache **Hive** is an open source data warehouse software for reading, writing and managing large data set files that are stored directly in either the Apache **Hadoop** Distributed File System (**HDFS**) or other data storage systems such as Apache HBase.

1) **Download Hive**

https://archive.apache.org/dist/hive/hive-2.1.0/

Name	Last modified	Size	Description
Parent Directory		-	
apache-hive-2.1.0-bin.tar.gz	2016-06-21 01:26	143M	
apache-hive-2.1.0-bin.tar.gz.asc	2016-06-21 01:26	819	
apache-hive-2.1.0-bin.tar.gz.md5	2016-06-21 01:26	70	
apache-hive-2.1.0-src.tar.gz	2016-06-21 01:26	18M	
apache-hive-2.1.0-src.tar.gz.asc	2016-06-21 01:26	819	

2) **Download Apache Derby**

https://archive.apache.org/dist/db/derby/db-derby-10.12.1.1/

Name	Last modified	Size	Description
Parent Directory		-	
db-derby-10.12.1.1-bin.tar.gz	2015-10-10 14:38	18M	
db-derby-10.12.1.1-bin.tar.gz.asc	2015-10-10 14:38	194	
db-derby-10.12.1.1-bin.tar.gz.md5	2015-10-10 14:38	33	
db-derby-10.12.1.1-bin.zip	2015-10-10 14:38	20M	
db-derby-10.12.1.1-bin.zip.asc	2015-10-10 14:38	194	

3) **Extract both the files**

4) **Place hive – site.xml file in hive ->conf and then move both hive and derby folders to C drive**

https://drive.google.com/file/d/1qqAo7RQfr5Q6O-GTom6Rji3TdufP81zd/view

Name	Date modified	Type	Size
beeline-log4j2.properties.template	03-06-2016 16:13	TEMPLATE File	2 KB
hive-default.xml.template	17-06-2016 05:33	TEMPLATE File	221 KB
hive-env.sh.template	03-06-2016 16:13	TEMPLATE File	3 KB
hive-exec-log4j2.properties.template	03-06-2016 16:13	TEMPLATE File	3 KB
hive-log4j2.properties.template	03-06-2016 16:13	TEMPLATE File	3 KB
ivysettings.xml	10-06-2016 14:30	XML Document	3 KB
llap-cli-log4j2.properties.template	03-06-2016 16:13	TEMPLATE File	3 KB
llap-daemon-log4j2.properties.template	03-06-2016 16:13	TEMPLATE File	5 KB
parquet-logging.properties	09-06-2016 00:17	PROPERTIES File	3 KB
hive-site.xml	14-10-2020 15:38	XML Document	2 KB

5) Copy all the files from derby->lib and paste it to hive->lib

commons-compress-1.9.jar	25-02-2016 18:58	Executable Jar File	370 KB
commons-dbcp-1.4.jar	25-02-2016 18:59	Executable Jar File	157 KB
commons-el-1.0.jar	25-02-2016 18:59	Executable Jar File	110 KB
commons-httpclient-3.0.1.jar	25-02-2016 18:58	Executable Jar File	274 KB
commons-io-2.4.jar	25-02-2016 18:58	Executable Jar File	181 KB
commons-lang-2.6.jar	25-02-2016 18:58	Executable Jar File	278 KB
commons-lang3-3.1.jar	25-02-2016 18:58	Executable Jar File	309 KB
commons-logging-1.2.jar	25-02-2016 18:59	Executable Jar File	61 KB
commons-math-2.2.jar	25-02-2016 19:01	Executable Jar File	966 KB
commons-pool-1.5.4.jar	25-02-2016 18:59	Executable Jar File	94 KB
commons-vfs2-2.0.jar	25-02-2016 19:02	Executable Jar File	406 KB
curator-client-2.6.0.jar	25-02-2016 18:58	Executable Jar File	68 KB
curator-framework-2.6.0.jar	25-02-2016 18:58	Executable Jar File	181 KB
curator-recipes-2.6.0.jar	25-02-2016 18:58	Executable Jar File	243 KB

6) Add a new variable in Environmental variables – hive home, hive bin, hive lib

7) Different configuration properties to be put in Apache Hive

- We're going to do two things in this phase.
  1. Placing the Hive Home route to the bashrc file
  2. Placing the Hadoop Home path location in hive-config.sh

1) Mention Hive Pathin ~/.bashrc

```
hive@ubuntu:~$ nano ~/.bashrc
```

- Open the bashrc file as shown in above screenshot
- Mention Hive home path i.e., HIVE_HOME path in bashrc file and export it as shown in below

```
export HIVE_HOME="/home/guru99hive/apache-hive-1.2.0-bin"
export PATH=$PATH:$HIVE_HOME/bin
```

Code in bashrc:

```
export HIVE_HOME="/home/guru99hive/apache-hive-1.2.0-bin"
export PATH=$PATH:$HIVE_HOME/bin
```

2) Exporting Hadoop path in Hive-config.sh.Open the hive-config.sh

```
e@ubuntu:~/apache-hive-1.2.0-bin/bin$ nano hive-config.sh
```

8) Creating Hive directories in Hadoop

To interact with Hadoop, we need to build directories in Hadoop as shown below.

```
ive@ubuntu:~/Hadoop_YARN/hadoop-2.2.0/bin$ ./hadoop fs -mkdir /usr/hive/w
se
```

Giving root permissions to build Hive folders in Hadoop. If there is no error message, then this means that Hadoop has successfully issued permissions to Hive folders.

```
hive@ubuntu:~/Hadoop_YARN/hadoop-2.2.0/bin$ ./hadoop fs -chmod g+w /usr/hiss
rehouse
```

Getting into the Hive shell by entering it.' /hive 'command as shown below.

```
guru99hive@ubuntu:~/apache-hive-1.2.0-bin/bin$ ./hive
Logging initialized using configuration in jar:file:/home/datamatics/apache-hive
-1.2.0-bin/lib/hive-common-1.2.0.jar!/hive-log4j.properties
Java HotSpot(TM) 64-Bit Server VM warning: You have loaded library /home/datamat
ics/Hadoop_YARN/hadoop-2.2.0/lib/native/libhadoop.so.1.0.0 which might have disa
bled stack guard. The VM will try to fix the stack guard now.
It's highly recommended that you fix the library with 'execstack -c <libfile>',
or link it with '-z noexecstack'.
hive>
```

SYNTAX for HIVE Database Operations:

DATABASE Creation

CREATE DATABASE|SCHEMA [IF NOT EXISTS]

Drop Database Statement

DROP DATABASE StatementDROP (DATABASE|SCHEMA) [IF EXISTS] database_name [RESTRICT|CASCADE];

Creating and Dropping Table in HIVE

CREATE [TEMPORARY] [EXTERNAL] TABLE [IF NOT EXISTS] [db_name.] table_name [(col_namedata_type [COMMENT col_comment], ...)]

[COMMENT table_comment] [ROW FORMAT row_format] [STORED AS file_format]

Loading Data into table log_data

Syntax:

LOAD DATA LOCAL INPATH '/u.data' OVERWRITE INTO TABLE u_data;

Alter Table in HIVE

Syntax

ALTER TABLE name RENAME TO new_name

ALTER TABLE name ADD COLUMNS (col_spec[, col_spec ...])

ALTER TABLE name DROP [COLUMN] column_name

ALTER TABLE name CHANGE column_namenew_namenew_type

ALTER TABLE name REPLACE COLUMNS (col_spec[, col_spec ...])

Creating and Dropping View

REATE VIEW [IF NOT EXISTS] view_name [(column_name [COMMENT column_comment], ..)] [COMMENT table_comment] AS SELECT

Dropping View

Syntax:

DROP VIEW view_name

Functions in HIVE

String Functions:- round(), ceil(), substr(), upper(), reg_exp() etc

Date and Time Functions:- year(), month(), day(), to_date() etc

Aggregate Functions :- sum(), min(), max(), count(), avg() etc

INDEXES

CREATE INDEX index_name ON TABLE base_table_name (col_name, ...)

AS 'index.handler.class.name'

[WITH DEFERRED REBUILD]

[IDXPROPERTIES (property_name=property_value, ...)]

[IN TABLE index_table_name]

[PARTITIONED BY (col_name, ...)]

[[ROW FORMAT ...] STORED AS ... | STORED BY ...] [LOCATION hdfs_path] [TBLPROPERTIES (...)]

Creating Index

CREATE INDEX index_ip ON TABLE log_data(ip_address) AS org.apache.hadoop.hive.ql.index.compact.CompactIndexHandler' WITH DEFERRED REBUILD;

Altering and Inserting Index

ALTER INDEX index_ip_address ON log_dataREBUILD;

```
administrator@ubuntu: ~
d yet. Please use TIMESTAMP instead
hive> create table log_data(l_date string,l_time string,s_sitename string,s_comp
utername string,l_uri string,uri_query string,ip_address string,user_agent strin
g,status1 int,status2 int,s_bytes int,c_bytes int,time_taken int);
OK
Time taken: 0.331 seconds
hive> show tables;
OK
log_data
Time taken: 0.074 seconds, Fetched: 1 row(s)
hive> desc log_data;
OK
l_date                  string                  None
l_time                  string                  None
s_sitename              string                  None
s_computername          string                  None
l_uri                   string                  None
uri_query               string                  None
ip_address              string                  None
user_agent              string                  None
status1                 int                     None
status2                 int                     None
s_bytes                 int                     None
c_bytes                 int                     None
```

```
administrator@ubuntu: ~
0.6.20.6        Mozilla/4.0+(compatible;+MSIE+7.0;+Windows+NT+6.1;+Trident/4.0;+GTB7.5;+SLC
R+2.0.50727;+.NET+CLR+3.5.30729;+.NET+CLR+3.0.30729;+Media+Center+PC+6.0;+InfoPath.2)   304
11      498     0
2014-12-23      23:08:38        W3SVC1  NEWINTSERV2     /trf/elast/images/small/pic3.jpg
6.20.6          Mozilla/4.0+(compatible;+MSIE+7.0;+Windows+NT+6.1;+Trident/4.0;+GTB7.5;+SLC
R+2.0.50727;+.NET+CLR+3.5.30729;+.NET+CLR+3.0.30729;+Media+Center+PC+6.0;+InfoPath.2)   304
10      497     0
2014-12-23      23:16:07        W3SVC1  NEWINTSERV2     /trf/elast/css/demo.css -       10.
ozilla/4.0+(compatible;+MSIE+7.0;+Windows+NT+6.0;+SLCC1;+.NET+CLR+2.0.50727;+.NET+CLR+3.0.0
CLR+1.1.4322;+InfoPath.2)       304     0       210     458     0
2014-12-23      23:16:07        W3SVC1  NEWINTSERV2     /trf/elast/css/elastislide.css  -
0.22    Mozilla/4.0+(compatible;+MSIE+7.0;+Windows+NT+6.0;+SLCC1;+.NET+CLR+2.0.50727;+.NET+
06;+.NET+CLR+1.1.4322;+InfoPath.2)      304     0       210     465     0
2014-12-23      23:16:07        W3SVC1  NEWINTSERV2     /trf/elast/images/small/pic11.jpg
0.3.20.22       Mozilla/4.0+(compatible;+MSIE+7.0;+Windows+NT+6.0;+SLCC1;+.NET+CLR+2.0.5072
+3.0.04506;+.NET+CLR+1.1.4322;+InfoPath.2)      304     0       211     469     0
2014-12-23      23:16:07        W3SVC1  NEWINTSERV2     /trf/elast/images/small/pic12.jpg
0.3.20.22       Mozilla/4.0+(compatible;+MSIE+7.0;+Windows+NT+6.0;+SLCC1;+.NET+CLR+2.0.5072
+3.0.04506;+.NET+CLR+1.1.4322;+InfoPath.2)      304     0       211     469     0
2014-12-23      23:16:07        W3SVC1  NEWINTSERV2     /trf/elast/images/small/pic10.jpg
0.3.20.22       Mozilla/4.0+(compatible;+MSIE+7.0;+Windows+NT+6.0;+SLCC1;+.NET+CLR+2.0.5072
+3.0.04506;+.NET+CLR+1.1.4322;+InfoPath.2)      304     0       211     469     0
2014-12-23      23:16:07        W3SVC1  NEWINTSERV2     /trf/elast/images/small/pic9.jpg
0.3.20.22       Mozilla/4.0+(compatible;+MSIE+7.0;+Windows+NT+6.0;+SLCC1;+.NET+CLR+2.0.5072
+3.0.04506;+.NET+CLR+1.1.4322;+InfoPath.2)      304     0       210     467     0
2014-12-23      23:16:07        W3SVC1  NEWINTSERV2     /trf/elast/images/small/pica.jpg
```

```
administrator@ubuntu: ~
hive> select * from index_ip;
FAILED: SemanticException [Error 10001]: Line 1:14 Table not found 'index_ip'
hive> INSERT OVERWRITE DIRECTORY '/home/administrator/Desktop/hive_data/index_test_result' SELECT `_
bucketname` , `_offsets` FROM lendi_db.lendi_db__log_data_index_ip__  where  ip_address='141.0.11.19
9';
Total MapReduce jobs = 3
Launching Job 1 out of 3
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job = job_1476764326039_0014, Tracking URL = http://ubuntu.ubuntu-domain:8088/proxy/applica
tion_1476764326039_0014/
Kill Command = /home/administrator/hadoop-2.7.1/bin/hadoop job  -kill job_1476764326039_0014
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 0
2016-10-18 02:16:23,240 Stage-1 map = 0%,  reduce = 0%
2016-10-18 02:16:27,406 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 1.32 sec
2016-10-18 02:16:28,442 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 1.32 sec
2016-10-18 02:16:29,472 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 1.32 sec
MapReduce Total cumulative CPU time: 1 seconds 320 msec
Ended Job = job_1476764326039_0014
Stage-3 is selected by condition resolver.
Stage-2 is filtered out by condition resolver.
Stage-4 is filtered out by condition resolver.
Moving data to: hdfs://localhost:9000/tmp/hive-administrator/hive_2016-10-18_02-16-17_425_5894975364
0454830/-ext-10000
Moving data to: /home/administrator/Desktop/hive_data/index_test_result
```

PRACTICAL 9

Aim: To install and run Pig and then write Pig Latin scripts to sort, group, join, project, and filter your data.

Implementation:

Apache Pig is an abstraction of MapReduce. It is a tool / platform that is used to analyse larger data sets that represent them as data streams. Pig is commonly used in Hadoop; all data processing operations in Hadoop can be done using Apache Pig.

Apache Pig – Installation

Prerequisites: It is important that you have Hadoop and Java built on your device before you go to Apache Pig. Therefore, instal Hadoop and Java before installing Apache Pig by following the steps given in the following link:

http://www.tutorialspoint.com/hadoop/hadoop_enviornment_setup.htm

Download Apache Pig

First of all, download the latest version of Apache Pig from the following website – https://pig.apache.org/

Step 1: Open the website of Apache Pig. Under the News column, press the Link Release page as seen in the snapshot below.

Step 2: You will be redirected to the Apache Pig Releases page when you click the link you specified. On this page, under the Download section, you will have two links, namely Pig 0.8 and later and Pig 0.7 and earlier. Click on the link Pig 0.8 and then you'll be redirected to a page with a set of mirrors.

Step 3: Choose and click any one of these mirrors as shown below.

Step 4: These mirrors are going to take you to the Pig Releases tab. This page contains a number of versions of Apache Pig. Tap on the new update of this one.

Step 5: In these directories, you can have the root and binary files of Apache Pig in different distributions. Download the root and binary tar files of Apache Pig 0.15, pig0.15.0-src.tar.gz and pig-0.15.0.tar.gz.

Install Apache Pig

After downloading the Apache Pig app, instal it in your Linux environment by following the steps below.

Step 1:

Create a directory named Pig in the same directory where Hadoop, Java, and other software installation directories have been installed. (We built the Pig directory in the user called Hadoop in our tutorial).

```
$ mkdir Pig
```

Step 2:
Extract the downloaded tar files as shown below.

```
$ cd Downloads/
$ tar zxvf pig-0.15.0-src.tar.gz
$ tar zxvf pig-0.15.0.tar.gz
```

Step 3:
Move the content of pig-0.15.0-src.tar.gz file to the Pig directory created earlier as shown below.

```
$ mv pig-0.15.0-src.tar.gz/* /home/Hadoop/Pig/
```

Pig Latin scripts

- **Sorting Data:**
  SORT_DATA = ORDER DATA BY ATTRIBUTE WITH CONDITION;
- **GROUP:**
  Data GDATA = GROUP DATA by ATTRIBUTE;
- **FILTER:**
  Data FDATA = FILTER DATA by ATTRIBUTE = VALUE;
- **JOIN Data:**
  JOIN DATA1 BY (ATTRIBUTE1, ATTRIBUTE2….), DATA2 BY (ATTRIBUTE3, ATTRIBUTE…. N)

Practical 10

Aim: To install, deploy & configure Apache Spark Cluster. To Select the fields from the dataset using Spark SQL. To explore Spark shell and read from HDFS.

Implementation:

Apache Spark is a lightning-fast cluster computing technology, designed for fast computation. It is based on Hadoop MapReduce and it extends the MapReduce model to efficiently use it for more types of computations, which includes interactive queries and stream processing. The main feature of Spark is its **in-memory cluster computing** that increases the processing speed of an application.

1. **Install Java:** Java is the only dependency to be installed for Apache Spark.
   To install Java, open a terminal and run the following command:

```
~$ sudo apt-get install default-jdk
```

2. **Install latest Apache Spark on Ubuntu 16**
3. **Download Spark:** There is a continuing progression of the Apache Spark. Newer models are starting to come out now and then. So, grab the new update of Spark when you're going to launch it. To download latest Apache Spark release, open the url [http://spark.apache.org/downloads.html] in a browser.

4. **Unzip and move spark to /usr/lib/:** Downloading is a zip drive. Unzip the file before setting up Apache Spark on your PC. To unzip the download, open the terminal and run the tar command from the zip file location.

```
~$ tar xzvf spark-2.2.0-bin-hadoop2.7.tgz
```

Move the folder to / usr / lib/. In the following terminal commands, we've copied the contents of the unzipped spark folder to a folder called spark. Then we moved the designated folder to / usr / lib/.

```
~$ mv spark-2.2.0-bin-hadoop2.7/ spark
~$ sudo mv spark/ /usr/lib/
```

5. **Add Path:** We now need to set the SPARK HOME environment variable and add it to the Route. The JAVA HOME variable should also be configured as a prerequisite.To set the JAVA HOME variable and add the / usr / lib / spark / bin folder to PATH, open ~/.bashrc with any editor. We're using the nano editor here:

```
~$ sudo nano ~/.bashrc
```

And add following lines at the end of ~/.bashrc file.

```
export JAVA_HOME=/usr/lib/jvm/default-java/jre
export SPARK_HOME=/usr/lib/spark/bin
export PATH=$PATH:SPARK_HOME
```

Latest Apache Spark is successfully installed in your Ubuntu 16.

6. **Verify installation:** Now that we've installed everything you need and set up the Route, we'll check if Apache Spark has been installed correctly. To check the installation, close the terminal that has already been opened and open the new terminal again. Run the following command as follows:

```
~$ spark-shell
```

```
~$ spark-shell
Using Sparks default log4j profile: org/apache/spark/log4j-defaults.properties
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
17/08/04 03:42:23 WARN NativeCodeLoader: Unable to load native-hadoop library for your
platform... using builtin-java classes where applicable
17/08/04 03:42:23 WARN Utils: Your hostname, arjun-VPCEH26EN resolves to a loopback
address: 127.0.1.1; using 192.168.1.100 instead (on interface wlp7s0)
17/08/04 03:42:23 WARN Utils: Set SPARK_LOCAL_IP if you need to bind to another address
17/08/04 03:42:36 WARN ObjectStore: Failed to get database global_temp, returning
NoSuchObjectException
Spark context Web UI available at http://192.168.1.100:4040
Spark context available as 'sc' (master = local[*], app id = local-1501798344680).
Spark session available as 'spark'.
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 2.2.0
      /_/

Using Scala version 2.11.8 (OpenJDK 64-Bit Server VM, Java 1.8.0_131)
Type in expressions to have them evaluated.
Type :help for more information.

scala> :quit
sparkuser@tutorialkart:~$
```

- Using the following steps to navigate the Spark SQL by using the Spark shell.
  The following example command begins the Spark shell on the YARN cluster:

```
spark-shell --num-executors 1 --executor-memory 512m --master yarn-client
```

- Create a SQLContext to read data directly from the file system. For eg, which uses SQLContext and the Spark DataFrame API to access the JSON file.
- To read data by interacting with the Hive Metastore, construct a HiveContext instance (HiveContext extends SQLContext).

Practical 11

Aim: To perform PYspark shell exploration and reading and writing in HDFS. To perform Sentiment Analysis and use Scala.

Implementation:

Reading data

Code:

```
from pyspark import SparkConf, SparkContext
import sys

conf = SparkConf().setMaster("local").setAppName("RatingsHistogram")
sc = SparkContext(conf = conf)

# Create a hardcoded RDD
numbers = sc.parallelize([1, 2, 3, 4])

# Load plain text files (e.g. CSVs) from different sources
hdfs_lines = sc.textFile("hdfs:///user/cloudera/ml-100k/u.data", minPartitions=1)
local_lines = sc.textFile("file:///home/alex/ml-100k/u.data")
s3_lines = sc.textFile("s3n://bucket/ml-100k/u.data")
CLI_arg_lines = sc.textFile(sys.argv[1])

# Create RDDs from an existing Hive repository
hive_ctx = HiveContext(sc)
hive_lines = hive_ctx.sql("SELECT name, age FROM users WHERE age > 18")
```

- Notice that this can not be run with the standard Python interpreter. Instead, you use the spark-submit to submit it as a batch task, or name the Shell Pyspark.

Writing of data:

- The RDD class has the method saveAsTextFile. However, this saves the representation of each variable in a string. In Python, the resulting text file will contain lines such as (1949, 111).
- If you want to save the data in CSV or TSV format, you can either use Python's StringIO and csv modules or simply map each variable (vector) into a single string, e.g. as follows:

Code:

```
res.saveAsTextFile("hdfs:///user/cloudera/res_raw.txt")  # bad format

res.map(lambda row: str(row[0]) + "\t" + str(row[1])) \
.saveAsTextFile("hdfs:///user/cloudera/res_tsv.txt")  # good format
```

Sentiment Analysis using Scala:

Spark Streaming Implementation:

Code:

```
//Import the necessary packages into the Spark Program
import org.apache.spark.streaming.{Seconds, StreamingContext}
import org.apache.spark.SparkContext._
...
import java.io.File

object twitterSentiment {

def main(args: Array[String]) {
if (args.length < 4) {
System.err.println("Usage: TwitterPopularTags <consumer key> <consumer secret>
System.exit(1)
}

StreamingExamples.setStreamingLogLevels()
//Passing our Twitter keys and tokens as arguments for authorization
val Array(consumerKey, consumerSecret, accessToken, accessTokenSecret) = args.
val filters = args.takeRight(args.length - 4)

// Set the system properties so that Twitter4j library used by twitter stream
// Use them to generate OAuth credentials
System.setProperty("twitter4j.oauth.consumerKey", consumerKey)
...
System.setProperty("twitter4j.oauth.accessTokenSecret", accessTokenSecret)

val sparkConf = new SparkConf().setAppName("twitterSentiment").setMaster("loca
val ssc = new Streaming Context
val stream = TwitterUtils.createStream(ssc, None, filters)

//Input DStream transformation using flatMap
val tags = stream.flatMap { status => Get Text From The Hashtags }

//RDD transformation using sortBy and then map function
tags.countByValue()
.foreachRDD { rdd =>
val now = Get current time of each Tweet
rdd
.sortBy(_._2)
.map(x => (x, now))
//Saving our output at ~/twitter/ directory
.saveAsTextFile(s"~/twitter/$now")
}

val data = tweets.map { status =>
val sentiment = SentimentAnalysisUtils.detectSentiment(status.getText)
val tagss = status.getHashtagEntities.map(_.getText.toLowerCase)
(status.getText, sentiment.toString, tagss.toString())
}

data.print()
//Saving our output at ~/ with filenames starting like twitters
data.saveAsTextFiles("~/twitters","20000")

ssc.start()
ssc.awaitTermination()
 }
}
```

Results:

The following are the results shown in the Eclipse IDE when running the Twitter Sentiment Streaming software.

- As we can see from the screenshot, all tweets are categorised as Positive, Neutral and Negative according to the feelings of the contents of the tweets.
- The output of the Sentiments of the Tweets is stored in folders and files, depending on the time they were created. This output can be stored on the local file system or HDFS as appropriate. The output directory looks like this one:

- Here, within the twitter folders, we will find the usernames of the Twitter users and the timestamp for. tweet as seen below:

- Now that we have Twitter usernames and timestamps, let 's look at the Sentiments and Tweets saved in the main directory. Here, every tweet is accompanied by a feeling of emotion. This Sentiment, which is processed, is further used to examine a broad variety of business observations.

- Tweaking Code: Now, let's modify our code a little to get feelings about specific hashtags (topics). Donald Trump, the President of the United States, is now travelling through television outlets and online social media. Let us look at the emotions associated with the keyword 'Trump.'

Practical 12

Aim: To perform Graph analytics and visualization using Neo4j and Gephi

Implementation:

Creating a Graph Visualization from Neo4j Data

- Visualization allows users to creatively explore stored data, to recognise important structures, and to achieve a deeper understanding of relationships.
- In addition to visual experimentation, it is often possible to communicate with stored data by changing the diagram without any Cypher expertise that helps non-technical users to explore and change the data.
- For example, with mouse gestures, new nodes and relationships can be created. Visualizing a graph database is therefore a powerful solution when handling or analysing large amounts of data.

- Neo4j comes with an advanced analysis tool. This tool lets you run Cypher queries and shows the response either as a table or as a plain, force-based diagram. Although this is useful for learning Cypher and as a simple check whether a question yields the correct answer, more complicated tasks also need sophisticated and personalised visualisation.

Data-Driven Visualization

- Good visualisation offers insights into data that are not apparent from looking at raw data. YFiles for HTML provides a variety of ways to use Neo4j database data to

construct a rich and meaningful representation. Users can take advantage of powerful data linking capabilities to build object models that use the labels and properties of Neo4j data.

Visual network analysis with Gephi

- Gephi is a piece of open-source software that lets you simulate, control, and discover the network.
- Networks (a.k.a. graphs) can be stored in a variety of file formats (for example. gexf, gdf, or. gml) Nodes and edges can represent any kind of entity and relationship.
- In the screenshot below the editors of the 'Circumcision' article have decided to make in-text references to the articles about 'foreskin', 'human penis', 'anesthesia', 'physiologic stress', 'elective surgery', and so forth.

Based on these hyperlinks to other Wikipedia articles, we can build a network. The nodes are articles like 'circumcision', 'foreskin', or 'anaesthesia' and the edges are direct references between them. Two distinct clusters are clearly visible suggesting that articles in the 'Circumcision' category are thematically related in two distinct ways. Rather than being part of the same issue, the debates about male and female circumcision seem to be associated with different people, different problems, and different events.

Practical 13

<u>Aim</u>: To perform case study of the following platforms for solving any big data analytic problem of your choice. (1) Amazon web services, (2) Microsoft Azure, (3)Google App engine

Google App Engine: Travlytix: Turns to Google Cloud to run a customer data platform and personalization engine

- With big data products running on Google Cloud Platform, Travlytix created an intelligent customer data platform for airlines in six months with a small engineering team.
- Travelers generate data across a range of touchpoints, presenting opportunities for businesses to understand more about how they behave. Malaysia-headquartered Travlytix decided to create a platform to consolidate real-time data from these touchpoints into a single location. The business also had to ensure the data was secured from leakage, disruption or theft.0

"Google Cloud Platform services like Cloud Pub/Sub, Cloud Dataflow, and Big Query allow us to minimize the effort and resources needed to build data pipelines for airline customers and focus instead on the quality, volume, and velocity of data."

—Srinivas Sri Perumbuduru, Head of Product, Travlytix

- Airlines, online travel agents, and travel ecommerce companies face challenges capturing, processing, and utilizing data. The business turned to the cloud to deliver its platform. "The cloud gave us an opportunity to work with large volumes of data while becoming more agile," says Travlytix.
-

Making data transparent:

- "Over the history of industry, people have used data to make more educated choices," adds Srinivas Sri Perumbuduru, Product Manager at Travlytix.
- "Our goal in Travelytix is to make the data clearer."
- Travlytix needed a highly available, resilient and secure cloud platform to deliver what Sri Perumbuduru calls "one of the most workable customer data platforms in the travel industry." The corporation then switched to the Google Cloud Network.

Google Cloud Platform supports architecture principles:

- Travlytix's review showed that Google Cloud Platform will support its founding principles of keeping its architecture basic, microservice-based and highly secure.
- "The documentation and expertise of the Google Cloud Platform and simple contact with Google's customer engineers have made this possible," states Sri Perumbuduru.
- With the Google Cloud Technology, the company was able to develop the consumer data technology in just six months.
- Travlytix uses Google Cloud Platform services to collect, process, and store traveler data.
- The platform captures data in real time, processes it through data pipelines, and stores it in a data warehouse.

- TravlyTix also relies on Compute Engine to provide compute resources; Google Kubernetes Engine to manage and orchestrate Docker containers; App Engine to develop and host its applications; and Cloud Memorystore to provide caches to enable sub-millisecond data access.
- "Google Cloud managed services give us more time to strategize our data pipeline architecture and allow us to be less concerned about maintaining the infrastructure," says Kevin Chin, Data Engineer, Travleytix.

Low-latency services:

- Travlytix also utilises several Google Cloud Network zones to offer ongoing, low-latency support to clients around the globe.
- In addition, applications such as the Cloud Key Management Service, which helps companies to control cryptographic keys and secure confidential data on the Google Cloud Network, facilitate compliance with global security regulations.
- These support Travlytix's own data encryption during collection and processing to protect it from intrusion.