EXP. 9: PERFORM THE BASIC CONFIGURATION SETUP FOR INSTALLING HADOOP 2.X LIKE CREATING THE HDUSER AND SSH LOCALHOST

AIM:

PROCEDURE:

Step 1 – System Update

$ sudo apt-get update

Step 2 – Install Java and Set JAVA_HOME
//This first thing to do is to setup the webupd8 ppa on your system. Run the following command and proceed.
$ sudo apt-add-repository ppa:webupd8team/java
$ sudo apt-get update
//After setting up the ppa repository, update the package cache as well.
//Install the Java 8 installer
$ sudo apt-get install oracle-java8-installer

// After the installation is finished, Oracle Java is setup. Run the java command again to check the version and vendor.

[or]

$ sudo apt-get install default-jdk
$ java -version

Step 3 – Add a dedicated Hadoop user
$ sudo addgroup hadoop

$ sudo adduser --ingroup hadoop hduser

// Add hduser to sudo user group

$ sudo adduser hduser sudo

Step 4 – Install SSH and Create Certificates
$ sudo apt-get install ssh

$ su hduser

$ ssh-keygen -t rsa -P ""

// Set Environmental variables
$ cat $HOME/.ssh/id_rsa.pub >> $HOME/.ssh/authorized_keys

Step 5 – Check if SSH works
$ ssh localhost

Step 6 – Install Hadoop
// Extract Hadoop-2.7.2
$ sudo tar xvzf hadoop-2.7.2.tar.gz

// Create a folder 'hadoop' in /usr/local
$ sudo mkdir –p /usr/local/hadoop

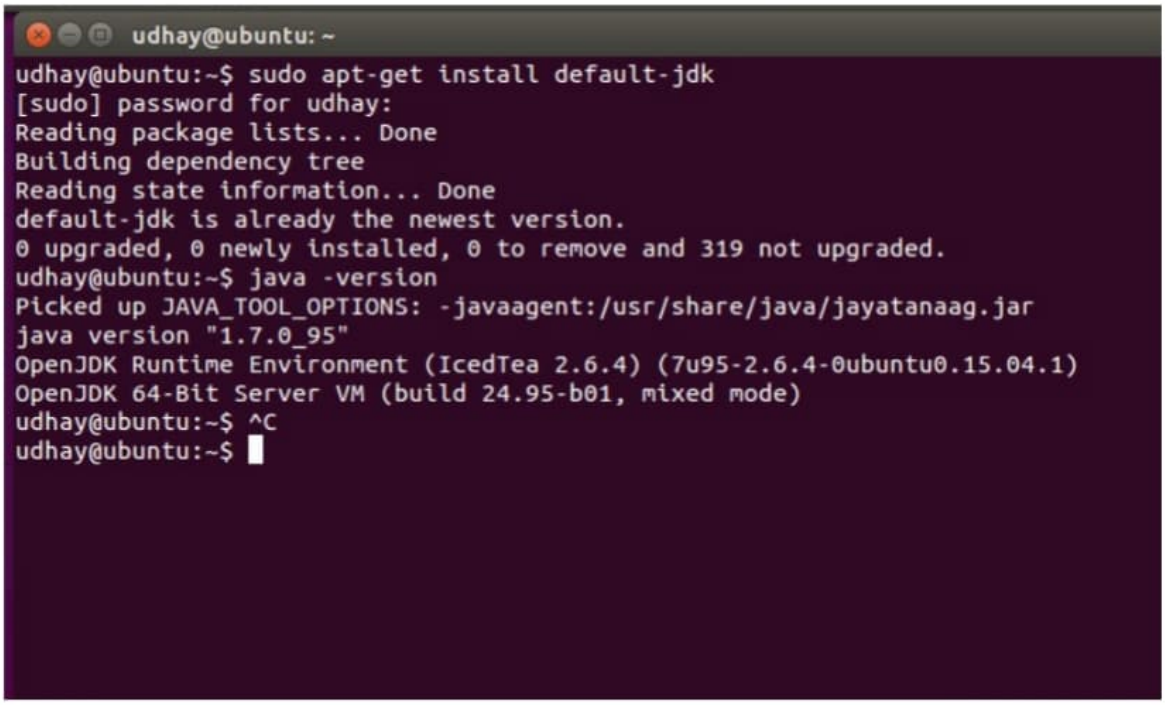
// Move the Hadoop folder to /usr/local/hadoop

$ sudo mv hadoop-2.7.2 /usr/local/hadoop

// Assigning read and write access to Hadoop folder

$ sudo chown –R hduser:hadoop /usr/local/hadoop

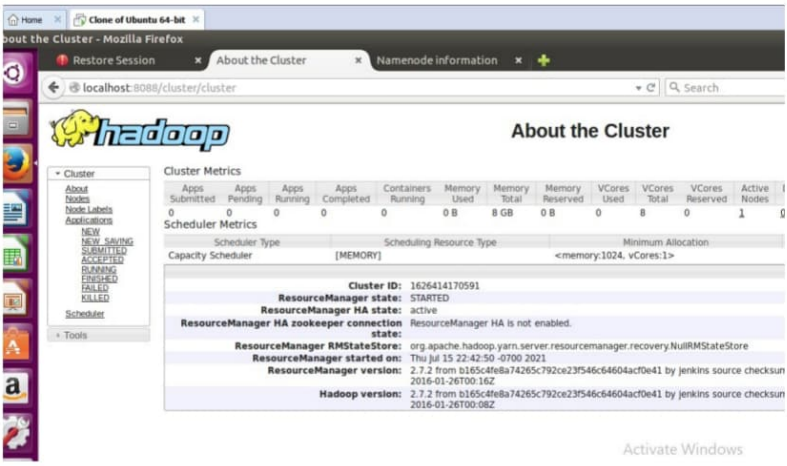
Implementation:



```
udhay@ubuntu:~$ sudo apt-get install ssh
Reading package lists... Done
Building dependency tree
Reading state information... Done
ssh is already the newest version.
0 upgraded, 0 newly installed, 0 to remove and 319 not upgraded.
udhay@ubuntu:~$ su hduser
Password:
hduser@ubuntu:/home/udhay$
```

```
udhay@ubuntu:~$ su hduser
Password:
hduser@ubuntu:/home/udhay$ ssh-keygen -t rsa -P ""
Generating public/private rsa key pair.
Enter file in which to save the key (/home/hduser/.ssh/id_rsa):
/home/hduser/.ssh/id_rsa already exists.
Overwrite (y/n)? y
Your identification has been saved in /home/hduser/.ssh/id_rsa.
Your public key has been saved in /home/hduser/.ssh/id_rsa.pub.
The key fingerprint is:
09:0f:15:f2:b2:b7:5e:11:1a:6c:d3:2f:c3:09:02:15 hduser@ubuntu
The key's randomart image is:
+---[RSA 2048]----+
|     ..E.o.       |
|      . = .       |
|       = B o      |
|        O B +     |
|       . S * .    |
|        . . +     |
|         . .      |
|        . .       |
|         .        |
+-----------------+
hduser@ubuntu:/home/udhay$
```

```
hduser@ubuntu:/home/udhay$ cat $HOME/.ssh/id_rsa.pub >> $HOME/.ssh/authorized_keys
hduser@ubuntu:/home/udhay$ ssh localhost
Welcome to Ubuntu 15.04 (GNU/Linux 3.19.0-84-generic x86_64)

 * Documentation:  https://help.ubuntu.com/

Last login: Thu Jul 15 22:00:14 2021 from localhost
hduser@ubuntu:~$
```



EXP. 10: INSTALL HADOOP 2.X AND CONFIGURE THE NAME NODE AND DATA NODE.

AIM:

PROCEDURE:

Step 7 - Modify Hadoop config files
//Hadoop Environmental variable setting – The following files will be modified

1. ~/.bashrc
2. /usr/local/hadoop/hadoop-2.7.2/etc/hadoop/hadoop-env.sh
3. /usr/local/hadoop/hadoop-2.7.2/etc/hadoop/core-site.xml
4. /usr/local/hadoop/hadoop-2.7.2/etc/hadoop/hdfs-site.xml
5. /usr/local/hadoop/hadoop-2.7.2/etc/hadoop/yarn-site.xml
6. /usr/local/hadoop/hadoop-2.7.2/etc/hadoop/mapred-site.xml.template

$ sudo nano ~/.bashrc

// Add the following lines at the end of the file

```
export JAVA_HOME=/usr/lib/jvm/java-8-oracle
export HADOOP_HOME=/usr/local/hadoop/hadoop-2.7.2
export PATH=$PATH:$HADOOP_HOME/bin
export PATH=$PATH:$HADOOP_HOME/sbin
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME=$HADOOP_HOME
export YARN_HOME=$HADOOP_HOME
HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
export HADOOP_OPTS="-D.java.library.path=$HADOOP_HOME/lib"
export PATH=$PATH:/usr/local/hadoop/hadoop-2.7.2/bin
```

// Configure Hadoop Files

$ cd /usr/local/hadoop/hadoop-2.7.2/etc/hadoop/

$ sudo nano hadoop-env.sh

// Add following line in hadoop-env.sh – Set JAVA variable in Hadoop

```
# The java implementation to use.
export JAVA_HOME=/usr/lib/jvm/java-8-oracle
```

// Create datanode and namenode

$ sudo mkdir –p /usr/local/hadoop_tmp/hdfs/namenode

$ sudo mkdir –p /usr/local/hadoop_tmp/hdfs/datanode

// Changing ownership to hadoop_tmp

$ sudo chown –R hduser:hadoop /usr/local/hadoop_tmp

// Edit hdfs-site.xml

$ sudo nano hdfs-site.xml

// Add the following lines between <configuration> …… </configuration>

```
<configuration>
<property>
<name>dfs.replication</name>
<value>1</value>
</property>
<property>
<name>dfs.namenode.name.dir</name>
<value>file:/usr/local/hadoop_tmp/hdfs/namenode</value>
</property>
<property>
<name>dfs.datanode.data.dir</name>
<value>file:/usr/local/hadoop_tmp/hdfs/datanode</value>
</property>
</configuration>
```

// Edit core-site.xml

$ sudo nano core-site.xml

// Add the following lines between <configuration> …… </configuration>

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

// Edit yarn-site.xml

$ sudo nano yarn-site.xml

// Add the following lines between <configuration> …… </configuration>

```
<configuration>
<property>
<name>yarn.nodemanager.aux-services</name>
<value>mapreduce_shuffle</value>
</property>
<property>
```

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

// Edit mapred-site.xmsudo

$ cp /usr/local/hadoop/hadoop-2.7.2/etc/hadoop/mapred-site.xml.template /usr/local/hadoop/hadoop-2.7.2/etc/hadoop/mapred-site.xml

$ sudo nano mapred-site.xml

// Add the following lines between <configuration> …… </configuration>

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

Step-8 – Format Hadoop File System
$ cd /usr/local/hadoop/hadoop-2.7.2/bin
$ hadoop namenode -format

Step 9 - Start Hadoop
$ cd /usr/local/hadoop/hadoop-2.7.2/sbin

// Starting dfs services

$ start-dfs.sh

// Starting mapreduce services

$ start-yarn.sh

$ jps

Step 10 - Check Hadoop through web UI
Go to browser type http://localhost:8088 – All Applications Hadoop Cluster

Go to browser type http://localhost:50070 – Hadoop Namenode

Step 11 - Stop Hadoop

$ stop-dfs.sh

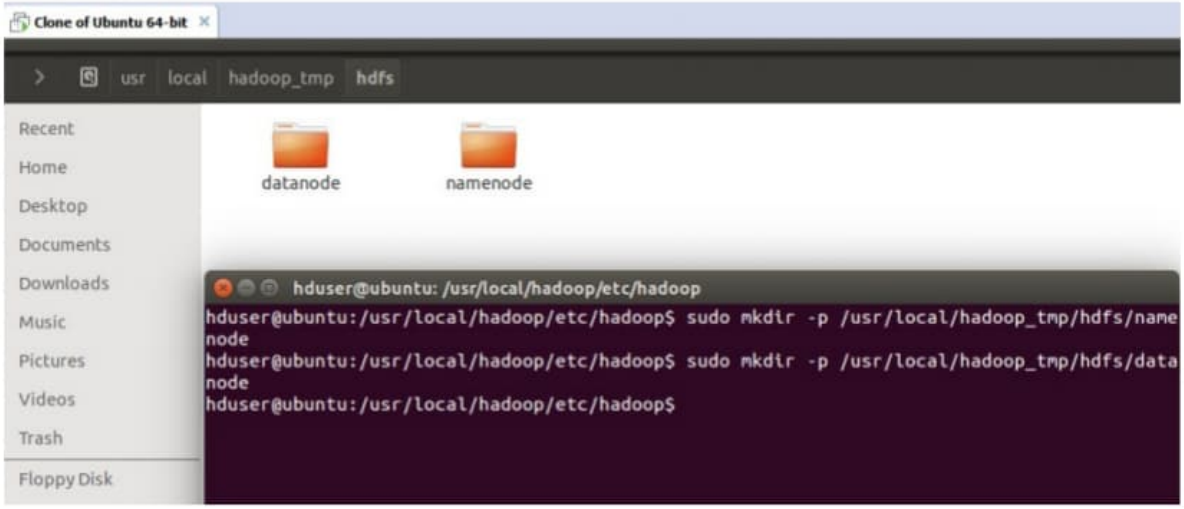
$ stop-yarn.sh

IMPLEMENTAION:

```
Clone of Ubuntu 64-bit
GNU nano 2.2.6                                File: /home/hduser/.bashrc

# See /usr/share/doc/bash-doc/examples in the bash-doc package.

if [ -f ~/.bash_aliases ]; then
    . ~/.bash_aliases
fi

# enable programmable completion features (you don't need to enable
# this, if it's already enabled in /etc/bash.bashrc and /etc/profile
# sources /etc/bash.bashrc).
if ! shopt -oq posix; then
  if [ -f /usr/share/bash-completion/bash_completion ]; then
    . /usr/share/bash-completion/bash_completion
  elif [ -f /etc/bash_completion ]; then
    . /etc/bash_completion
  fi
fi


#HADOOP VARIABLES START
export JAVA_HOME=/usr/lib/jvm/java-7-openjdk-amd64
export HADOOP_INSTALL=/usr/local/hadoop
export PATH=$PATH:$HADOOP_INSTALL/bin
export PATH=$PATH:$HADOOP_INSTALL/sbin
export HADOOP_MAPRED_HOME=$HADOOP_INSTALL
export HADOOP_COMMON_HOME=$HADOOP_INSTALL
export HADOOP_HDFS_HOME=$HADOOP_INSTALL
export YARN_HOME=$HADOOP_INSTALL
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_INSTALL/lib/native
export HADOOP_OPTS="-Djava.library.path=$HADOOP_INSTALL/lib"
#HADOOP VARIABLES END
```

```
hduser@ubuntu:/home$ cd ..
hduser@ubuntu:/$ cd usr
hduser@ubuntu:/usr$ cd local
hduser@ubuntu:/usr/local$ cd hadoop
hduser@ubuntu:/usr/local/hadoop$ cd etc
hduser@ubuntu:/usr/local/hadoop/etc$ cd hadoop
hduser@ubuntu:/usr/local/hadoop/etc/hadoop$ ls
capacity-scheduler.xml      httpfs-env.sh            mapred-env.sh
configuration.xsl           httpfs-log4j.properties  mapred-queues.xml.template
container-executor.cfg      httpfs-signature.secret  mapred-site.xml
core-site.xml               httpfs-site.xml          mapred-site.xml.template
hadoop-env.cmd              kms-acls.xml             slaves
hadoop-env.sh               kms-env.sh               ssl-client.xml.example
hadoop-metrics2.properties  kms-log4j.properties     ssl-server.xml.example
hadoop-metrics.properties   kms-site.xml             yarn-env.cmd
hadoop-policy.xml           log4j.properties         yarn-env.sh
hdfs-site.xml               mapred-env.cmd           yarn-site.xml
hduser@ubuntu:/usr/local/hadoop/etc/hadoop$
```

```
hduser@ubuntu: /usr/local/hadoop/etc/hadoop
GNU nano 2.2.6                File: hadoop-env.sh

# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.

# Set Hadoop-specific environment variables here.

# The only required environment variable is JAVA_HOME.  All others are
# optional.  When running a distributed configuration it is best to
# set JAVA_HOME in this file, so that it is correctly defined on
# remote nodes.

# The java implementation to use.
export JAVA_HOME=/usr/lib/jvm/java-7-openjdk-amd64
export JAVA_HOME=${JAVA_HOME}

# The jsvc implementation to use. Jsvc is required to run secure datanodes
# that bind to privileged ports to provide authentication of data transfer
# protocol.  Jsvc is not required if SASL is configured for authentication of
# data transfer protocol using non-privileged ports.
```



RESULT:

EXP. 11 LAUNCH THE HADOOP 2.X AND PERFORM MAPREDUCE PROGRAM FOR A WORD COUNT PROBLEM

AIM:

PROCEDURE:

Step 1 - Open Terminal

$ su hduser

Password:

Step 2 - Start dfs and mapreduce services

$ cd /usr/local/hadoop/hadoop-2.7.2/sbin

$ start-dfs.sh

$ start-yarn.sh

$ jps

Step 3 - Check Hadoop through web UI

// Go to browser type http://localhost:8088 – All Applications Hadoop Cluster

// Go to browser type http://localhost:50070 – Hadoop Namenode

Step 4 – Open New Terminal

$ cd Desktop/

$ mkdir inputdata

$ cd inputdata/

$ echo "Hai, Hello, How are you? How is your health?" >> hello.txt

$ cat >> hello.txt

Step 5 – Go back to old Terminal

$ hadoop fs –copyFromLocal /home/hduser/Desktop/inputdata/hello.txt /folder/hduser

// Check in hello.txt in Namenode using Web UI

Step 6 – Download and open eclipse by creating workspace

Create a new java project.

Step 7 – Add jar to the project

You need to remove dependencies by adding jar files in the hadoop source folder. Now Click on **Project** tab and go to Properties.Under Libraries tab, click Add External JARs and select all the

jars in the folder (click on 1st jar, and Press Shift and Click on last jat to select all jars in between and click ok)

/usr/local/hadoop/hadoop-2.7.2/share/hadoop/commonand

/usr/local/hadoop/hadoop-2.7.2/share/hadoop/mapreduce folders.

Step -8 – WordCount Program

Create 3 java files named

- **WordCount.java**
- **WordCountMapper.java**
- **WordCountReducer.java**

WordCount.java

```
import org.apache.hadoop.conf.Configured;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.mapred.FileInputFormat;
import org.apache.hadoop.mapred.FileOutputFormat;
import org.apache.hadoop.mapred.JobClient;
import org.apache.hadoop.mapred.JobConf;

import org.apache.hadoop.util.Tool;
import org.apache.hadoop.util.ToolRunner;
import org.apache.hadoop.io.Text;

public class WordCount extends Configured implements Tool {

	@Override
	public int run(String[] arg0) throws Exception {
		// TODO Auto-generated method stub
		if(arg0.length<2)
		{
			System.out.println("check the command line arguments");
		}
		JobConf conf=new JobConf(WordCount.class);
		FileInputFormat.setInputPaths(conf, new Path(arg0[0]));
			FileOutputFormat.setOutputPath(conf, new Path(arg0[1]));
			conf.setMapperClass(WordMapper.class);
			conf.setReducerClass(WordReducer.class);
			conf.setOutputKeyClass(Text.class);
			conf.setOutputValueClass(IntWritable.class);
			conf.setOutputKeyClass(Text.class);
			conf.setOutputValueClass(IntWritable.class);
			JobClient.runJob(conf);


		return 0;
	}
	public static void main(String args[]) throws Exception
	{
		int exitcode=ToolRunner.run(new WordCount(), args);
		System.exit(exitcode);
```

```
	}
}
```

WordCountMapper.java

```
import java.io.IOException;

import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.mapred.MapReduceBase;
import org.apache.hadoop.mapred.OutputCollector;
import org.apache.hadoop.mapred.Reporter;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.Mapper;

public class WordCountMapper  extends MapReduceBase implements
Mapper<LongWritable,Text,Text,IntWritable>
{
	@Override
	public void map(LongWritable arg0, Text arg1, OutputCollector<Text, IntWritable> arg2,
Reporter arg3)
			throws IOException {
		// TODO Auto-generated method stub

		String s=arg1.toString();
		for(String word:s.split(" "))
		{
			arg2.collect(new Text(word),new IntWritable(1));
		}
	}
}
```

WordCountReducer.java

```
import java.io.IOException;
import java.util.Iterator;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.mapred.JobConf;
import org.apache.hadoop.mapred.OutputCollector;
import org.apache.hadoop.mapred.Reducer;
import org.apache.hadoop.mapred.Reporter;
import org.apache.hadoop.io.Text;

public class WordCountReducer implements Reducer<Text,IntWritable,Text,IntWritable> {
	@Override
	public void configure(JobConf arg0) {
		// TODO Auto-generated method stub
	}
	@Override
	public void close() throws IOException {
		// TODO Auto-generated method stub
```

```
        }
        @Override
        public void reduce(Text arg0, Iterator<IntWritable> arg1, OutputCollector<Text, IntWritable>
arg2, Reporter arg3)
                        throws IOException {
                // TODO Auto-generated method stub
                int count=0;
                while(arg1.hasNext())
                {
                        IntWritable i=arg1.next();
                        count+=i.get();
                }
                arg2.collect(arg0,new IntWritable(count));

        }
}
```

Step 9 - Creatr JAR file

Now Click on the Run tab and click Run-Configurations. Click on New Configuration button on the left-top side and Apply after filling the following properties.

Step 10 - Export JAR file

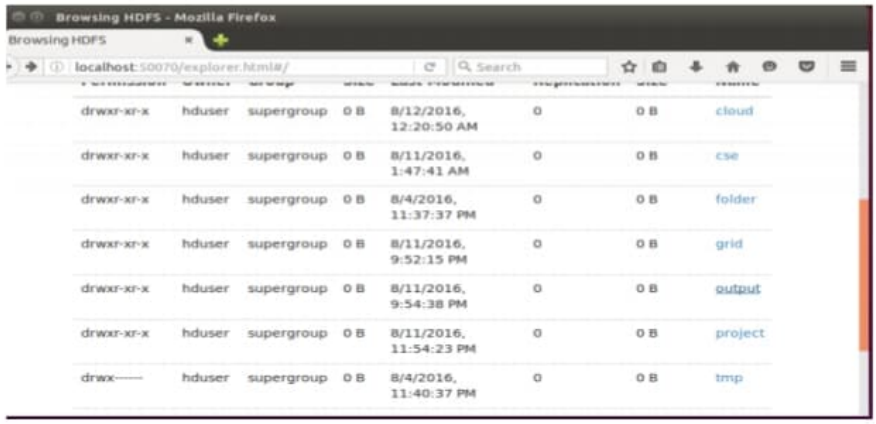
Now click on File tab and select Export. under Java, select Runnable Jar.

In Launch Config – select the config fie you created in **Step 9** (WordCountConfig).

Select an export destination (lets say desktop.)
Under Library handling, select Extract Required Libraries into generated JAR and click Finish.
Right-Click the jar file, go to Properties and under **Permissions**tab, Check Allow executing file as a program. and give Read and Write access to all the users

Step 11 – Go back to old Terminal for Execution of WordCount Program

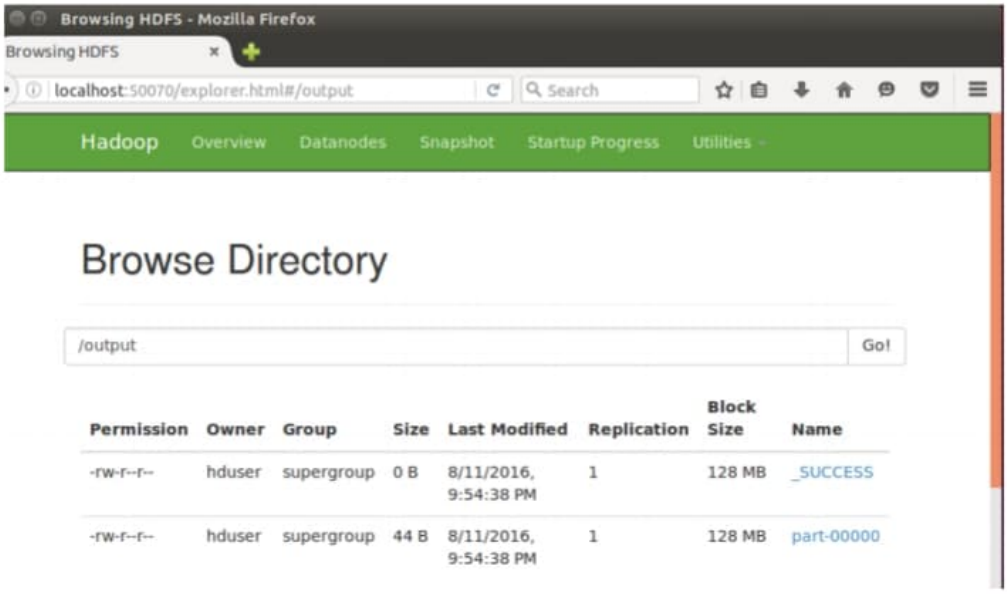
$hadoop jar wordcount.jar/usr/local/hadoop/input/usr/local/hadoop/output



Step 12 – To view results in old Terminal

$hdfs dfs -cat /usr/local/hadoop/output/part-r-00000

```
hadoop1@ubuntu-1:~/project$ hadoop fs -cat /output/wordcount4/part-r-00000
.        1
a        1
and      1
as       1
count    1
counts   1
file     2
for      1
input    1
is       1
job      1
job.     1
map      1
returns  1
sample   1
takes    1
```



Step 13 - To Remove folders created using hdfs

$ hdfs dfs -rm -R /usr/local/hadoop/output

RESULT: