**HADOOP**

1.apt-get install openjdk-8-jdk

2.javac -version

3.update-alternatives --config java

4.gedit /etc/environment

JAVA\_HOME=”/usr/lib/jvm/java-8-openjdk-i386”(copy this location from command 3)

Save the file

5.source /etc/environment

6.echo $JAVA\_HOME

7.apt-get install ssh

8. ssh-keygen -t rsa -P ""

9.cat $HOME/.ssh/id\_rsa.pub >> $HOME/.ssh/authorized\_keys

10.ssh localhost

11.Open browser and go to “hadoop.apache.org/releases.html” and open the 2.7.2 binary file and click the download link

**wget https://archive.apache.org/dist/hadoop/core/hadoop-2.7.2/hadoop-2.7.2.tar.gz**

12.tar –xvzf (“open downloads and drag and drop the downloaded hadoop-2.7.2.tar.gz folder”)

Eg: tar –xvzf ‘/home/sheeha/Downloads/hadoop-2.7.2.tar.gz’

13.sudo gedit ~/.bashrc

#HADOOP VARIABLES START

export JAVA\_HOME= <YOUR JAVA HOME PATH>

export HADOOP\_INSTALL= <HADOOP HOME PATH>

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

14.source ~/.bashrc

15.cd hadoop-2.7.2

16.cd etc

17.cd hadoop

18.gedit hadoop-env.sh

Append the following line to hadoop-env.sh:

export JAVA\_HOME=<YOUR JAVA HOME PATH>

cd $home

19.mkdir hadoop\_store

20.cd hadoop\_store

21.mkdir hdfs

22.cd hdfs

23.mkdir namenode

24.mkdir datanode

25.cd

26.cd hadoop-2.7.2/etc/hadoop

27.gedit hdfs-site.xml

<property>

<name>dfs.replication</name>

<value>1</value>

<description>Default block replication.

The actual number of replications can be specified when the file is created.

The default is used if replication is not specified in create time.

</description>

</property>

<property>

<name>dfs.namenode.name.dir</name>

<value>file:######NAMENODE\_FOLDER\_PATH######</value>

</property>

<property>

<name>dfs.datanode.data.dir</name>

<value>file:######DATANODE\_FOLDER\_PATH######</value>

</property>

28.cd

29.mkdir hadoop-2.7.2/etc/hadoop

30.gedit core-site.xml

<property>

<name>hadoop.tmp.dir</name>

<value>######TMP\_FOLDER\_PATH######</value>

<description>A base for other temporary directories.</description>

</property>

<property>

<name>fs.default.name</name>

<value>hdfs://localhost:54310</value>

<description>The name of the default file system. A URI whose

scheme and authority determine the FileSystem implementation. The

uri's scheme determines the config property (fs.SCHEME.impl) naming

the FileSystem implementation class. The uri's authority is used to

determine the host, port, etc. for a filesystem.</description>

</property>

31.cp mapred-site.xml.template mapred-site.xml

32.gedit mapred-site.xml

<property>

<name>mapred.job.tracker</name>

<value>localhost:54311</value>

<description>The host and port that the MapReduce job tracker runs

at. If "local", then jobs are run in-process as a single map

and reduce task.

</description>

</property>

33.cd

34.hadoop namenode -format

35.cd hadoop-2.7.2/sbin

36.start-all.sh

37.Open browser and goto localhost:8088 and check whether its running

38.Open browser and goto localhost:50070 and check

**EX. NO: 5**

**DATE: MOUNT THE ONE NODE HADOOP CLUSTER USING FUSE**

**AIM**

To mount one-node Hadoop cluster using FUSE

**THEORY**

FUSE (Filesystem in Userspace) enables you to write a normal user application as a bridge for a traditional filesystem interface. The hadoop-hdfs-fuse package enables you to use your HDFS cluster as if it were a traditional filesystem on Linux. It is assumed that you have a working HDFS cluster and know the hostname and port that your NameNode exposes.

**PROCEDURE**

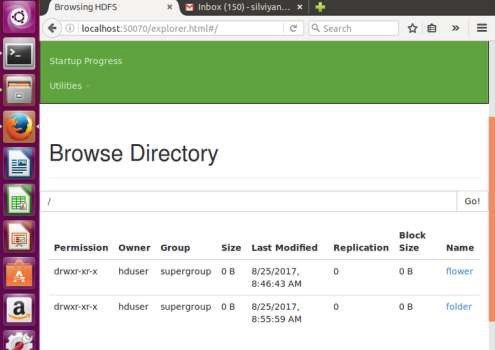
**$ su hduser**

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

**$ start-dfs.sh**

**$ start-yarn.sh**

**$ jps**



**$ wget** [**http://archive.cloudera.com/cdh5/one-click-install/trusty/amd64/cdh5-**](http://archive.cloudera.com/cdh5/one-click-install/trusty/amd64/cdh5-)**repository\_1.0\_all.deb**

**$ sudo dpkg -i cdh5-repository\_1.0\_all.deb**

**$ sudo apt-get update**

**$ sudo apt-get install hadoop-hdfs-fuse**

**$ sudo mkdir -p <mount\_point>**

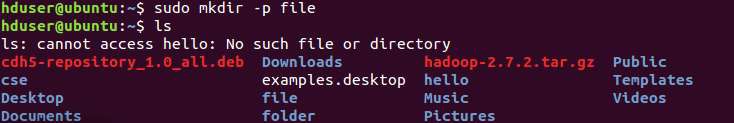
**Sample :**sudo mkdir -p file

sudo -H gedit /etc/samba/smb.conf

usershare owner only = false

sudo restart smbd

restart the pc



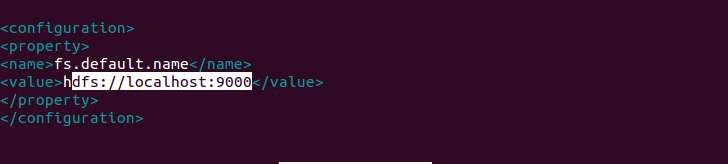
(set folder permission and enable share local network)



(for name\_node\_hostname>:<namenode\_port>

goto (cd /usr/local/hadoop/etc/hadoop/ vi core-site.xml )

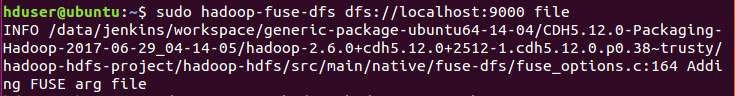


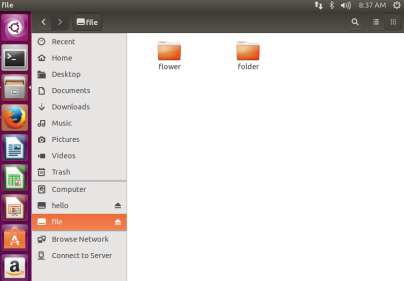


**$ sudo hadoop-fuse-dfs dfs://<name\_node\_hostname>:<namenode\_port>**

**<mount\_point>**

**Sample : sudo hadoop-fuse-dfs dfs://localhost:9000 file -o nonempty**





Once HDFS has been mounted at <mount\_point>, you can use most of the traditional filesystem operations (e.g., cp, rm, cat, mv, mkdir, rmdir, more, scp). However, random write operations such as rsync, and permission related operations such as chmod, chown are not supported in FUSE-mounted HDFS.

**RESULT**

Thus the steps to mount one-node Hadoop cluster using FUSE is done successfully.

**MAPREDUCE**

1.hadoop version

2.javac –version

3. Create a file called WordCount.java in desktop and copy this program and save

import java.io.IOException;

import java.util.StringTokenizer;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Job;

import org.apache.hadoop.mapreduce.Mapper;

import org.apache.hadoop.mapreduce.Reducer;

import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;

import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class WordCount {

  public static class TokenizerMapper

       extends Mapper<Object, Text, Text, IntWritable>{

    private final static IntWritable one = new IntWritable(1);

    private Text word = new Text();

    public void map(Object key, Text value, Context context

                    ) throws IOException, InterruptedException {

      StringTokenizer itr = new StringTokenizer(value.toString());

      while (itr.hasMoreTokens()) {

        word.set(itr.nextToken());

        context.write(word, one);

      }

    }

  }

  public static class IntSumReducer

       extends Reducer<Text,IntWritable,Text,IntWritable> {

    private IntWritable result = new IntWritable();

    public void reduce(Text key, Iterable<IntWritable> values,

                       Context context

                       ) throws IOException, InterruptedException {

      int sum = 0;

      for (IntWritable val : values) {

        sum += val.get();

      }

      result.set(sum);

      context.write(key, result);

    }

  }

  public static void main(String[] args) throws Exception {

    Configuration conf = new Configuration();

    Job job = Job.getInstance(conf, "word count");

    job.setJarByClass(WordCount.class);

    job.setMapperClass(TokenizerMapper.class);

    job.setCombinerClass(IntSumReducer.class);

    job.setReducerClass(IntSumReducer.class);

    job.setOutputKeyClass(Text.class);

    job.setOutputValueClass(IntWritable.class);

    FileInputFormat.addInputPath(job, new Path(args[0]));

    FileOutputFormat.setOutputPath(job, new Path(args[1]));

    System.exit(job.waitForCompletion(true) ? 0 : 1);

  }

}

4.Create a new folder in the desktop as “input”

5.Open “input” folder and create a text document called input.txt and type few words in the text file and save

6.Create a folder in the desktop as “tutorial”

7.Go to the terminal and type

export HADOOP\_CLASSPATH=$(hadoop classpath)

8.echo $HADOOP\_CLASSPATH

9.hadoop fs –mkdir /WordCountTutorial

10.hadoop fs –mkdir /WordCountTutorial/Input

11.Open browser and go to localhost:50070 and Utilities->Browse the file system and check for WordCountTutorial

12.hadoop fs –put ‘drag and drop input.txt’ /WordCountTutorial/Input

13. Open browser and go to localhost:50070 and Utilities->Browse the file system and open WordCountTutorial,open Input and check whether input.txt is present

14.cd /home/sheeha/Desktop/WordCountTutorial

15.javac –classpath ${HADOOP\_CLASSPATH} -d ‘class folder ‘drag and drop wordcount.java file’

16.In files open tutorial folder and check whether three files are present

17.jar –cvf firstTutorial.jar -C class(class folder)/ .

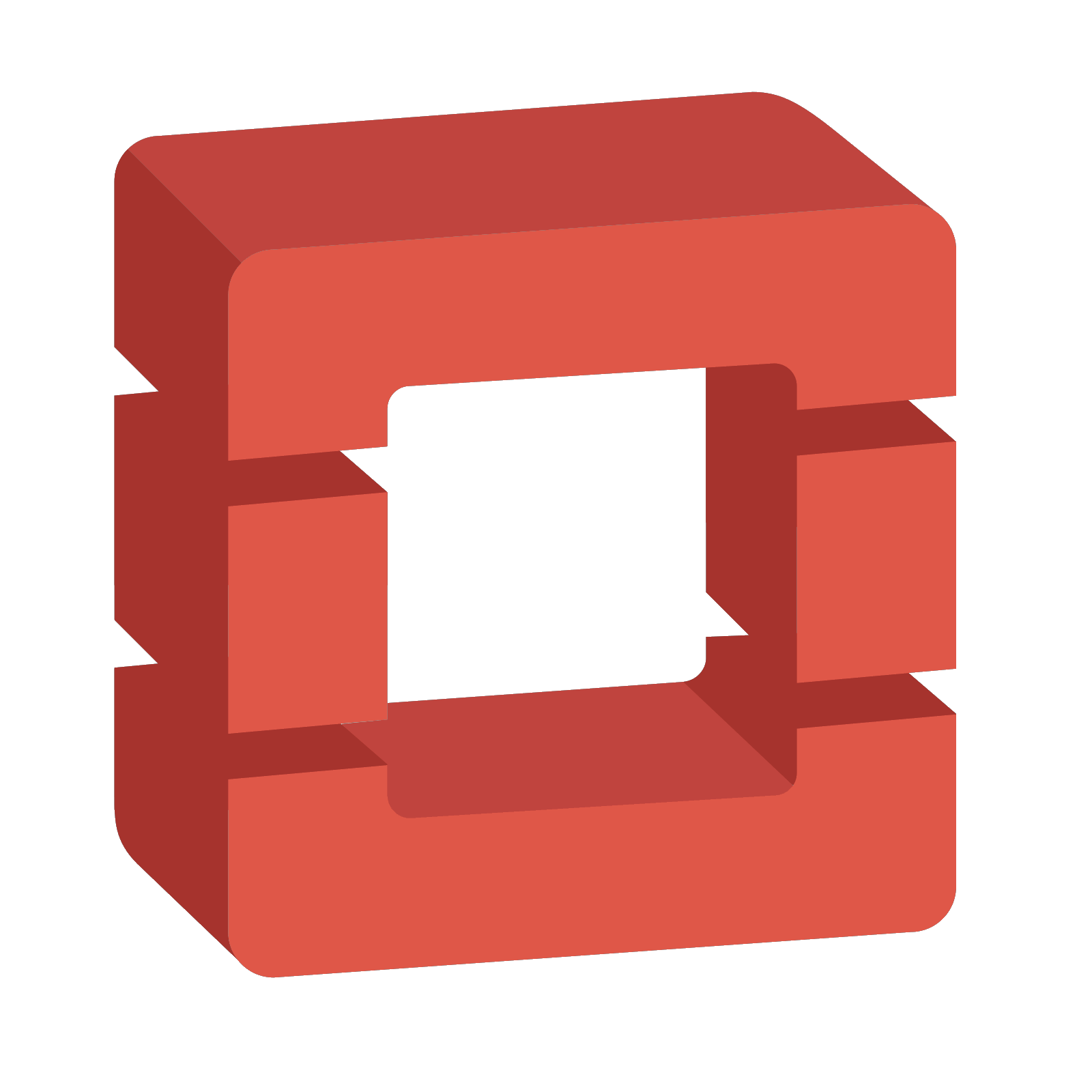
18.Open files and check whether a jar file is present

19.hadoop jar ‘drag and drop firstTutorial.jar file’ WordCount(class name) /WordCountTutorial/Input /WordCountTutorial/Output

20.hadoop dfs –cat /WordCountTutorial/Output/\*

**Installing**

**OpenStack**



This document contains the steps highlighting the procedures for AIO-OpenStack installation using Devstack installer on Ubuntu 16.04 . Developers have tested Openstack for Ubuntu 16.0.4. There is no official information available regarding support for Ubuntu 18.0.4, which is the latest version at the time of writing this. A successful installation was performed in our lab on computers with the following configuration

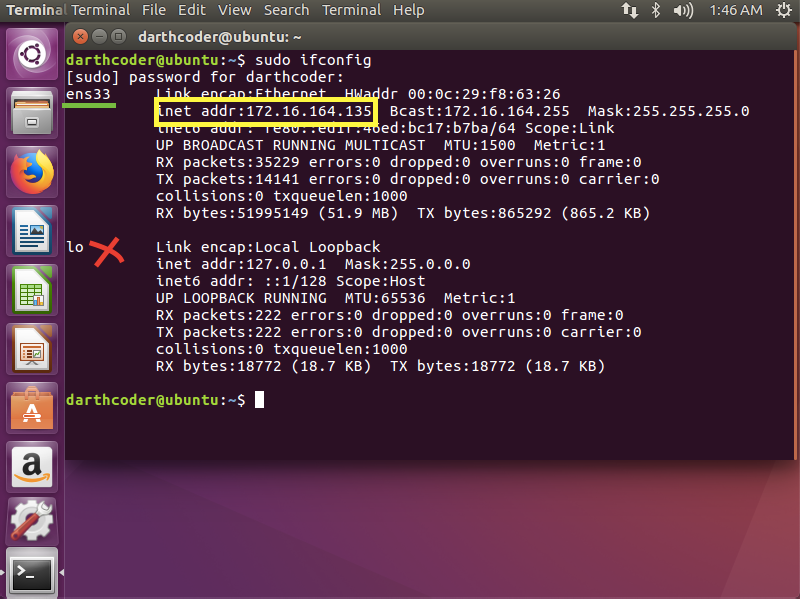
* Core i5 (6th gen)
* 8 GB Ram
* Windows 10
* Access to internet
* Virtual Machine with Ubuntu 16.04.5, 4.5 GB RAM and 20 GB storage.

1. After successful installation of VM install git .

**sudo apt-get install git**

1. Create a non-root user which is sudo enabled
   1. **sudo useradd -s /bin/bash -d /opt/stack -m stack**
   2. **echo "stack ALL=(ALL) NOPASSWD: ALL" | sudo tee /etc/sudoers.d/stack**
   3. **sudo su - stack**
2. Clone the devstack repository
   1. **git clone https://git.openstack.org/openstack-dev/devstack**
   2. **cd devstack**
3. Identify the Host IP. Copy the “inet address” from any device except lo and vibr0

**sudo ifconfig**



1. Create a local.conf file inside the devstack directory and containing the following details.

**sudo gedit local.conf [[local|localrc]] ADMIN\_PASSWORD=secret**

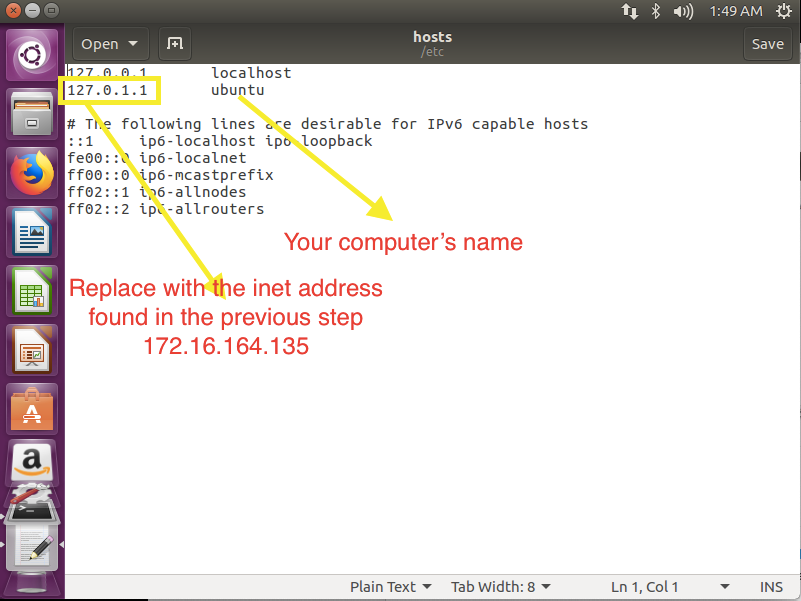
**DATABASE\_PASSWORD=$ADMIN\_PASSWORD RABBIT\_PASSWORD=$ADMIN\_PASSWORD SERVICE\_PASSWORD=$ADMIN\_PASSWORD**

**HOST\_IP= “the inet address after running sudo ifconfig”**

**FLOATING\_RANGE=** .224/27

1. Edit the hosts file. Replace your 127.0.0.1 with the host ip.

**sudo gedit /etc/hosts/**



1.

1. Start the installation.

**./stack.sh**

The installation would typically take between 30 and 50 minutes. During installation you may be prompted with entering the passwords for different Openstack services. Post installation, visit your host ip in the web browser for the dashboard.

**HADOOP API**

**EX. NO: 6**

**DATE: INTERACTION WITH HADOOP API FOR ACCESSING HDFS FROM**

**LOCAL FILE SYSTEM**

**AIM**

To write a program to use the API's of Hadoop for copying File from Local File System

to HDFS and to interact with it.

**PROCEDURE**

**Step 1 – Open New Terminal**

$ cd Desktop/

$ mkdir inputdata

$ cd inputdata/

$ echo “Hai, Hello, How are you? How is your health?” >> file.txt

$ cat >> file.txt

**Step 2 – Download and open eclipse by creating workspace**

Create a new java project.

**Step 3 – 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/common** and

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

**Step 4 – Program**

import org.apache.hadoop.conf.Configured; import org.apache.hadoop.util.Tool;

import java.io.BufferedInputStream; import java.io.FileInputStream; import java.io.InputStream;

import java.io.OutputStream;

import org.apache.hadoop.conf.Configuration; import org.apache.hadoop.fs.FileSystem; import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.IOUtils; import org.apache.hadoop.util.ToolRunner;

public class HdfsWriter extends Configured implements Tool { public static final String FS\_PARAM\_NAME = "fs.defaultFS";

public int run(String[] args) throws Exception {

if (args.length < 2) {

System.err.println("HdfsWriter /home/rec/Desktop/input/file.txt /hello"); return 1;

}

String localInputPath = args[0];

Path outputPath = new Path(args[1]); Configuration conf = getConf();

System.out.println("configured filesystem = " + conf.get(FS\_PARAM\_NAME)); FileSystem fs = FileSystem.get(conf);

if (fs.exists(outputPath)) { System.err.println("output path exists"); return 1;

}

OutputStream os = fs.create(outputPath);

InputStream is = new BufferedInputStream(new FileInputStream(localInputPath)); IOUtils.copyBytes(is, os, conf);

return 0;

}

public static void main( String[] args ) throws Exception { int returnCode = ToolRunner.run(new HdfsWriter(), args); System.exit(returnCode);

}

}

**Step 5 - 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 6 - Export JAR file**

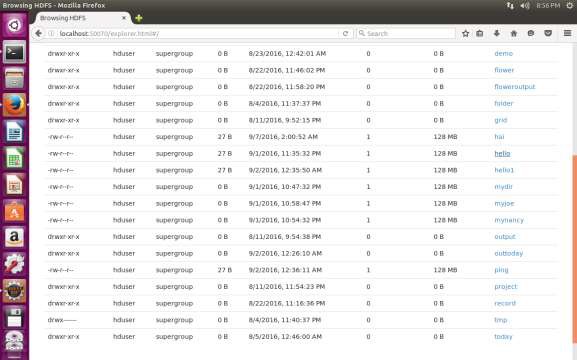
Now click on File tab and select Export. under Java, select Runnable Jar.

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 7–Execute File**

**hadoop jar /home/rec/Desktop/HelloWrite.jar /home/rec/Desktop/input/file.txt /hello**



**RESULT**

Thus the program to use the API's of Hadoop for copying File from Local File System to HDFS is executed and output verified successfully.

**Important commands**

Ps aux | grep -I apt

Sudo kill -9

sudo chown -R rec /dest

for lock error

sudo rm /var/lib/dpkg/lock

sudo rm /var/lib/apt/lists/lock

sudo rm /var/cache/apt/archives/lock

sudo rm -rf /var/lib/apt/lists/\*

sudo usermod -a -G sambashare $USER

//openstack

virtualenv /opt/stack/requirements/.venv/

sudo chown user(stack): /opt/stack directory