How to install Hadoop and

Your First Mapreduce Application with Eclipse and Hadoop

Step 1: Install Java Development Kit

- sudo apt update
- sudo apt install openjdk-8-jdk

Step 2: Verify Java Version

- java -version if not available

Step 3: Install SSH

- sudo apt install ssh

Step 4: Create the Hadoop User

- sudo adduser hadoop

Step 5: Switch User

- su - hadoop

Step 6: Configure SSH

- ssh-keygen -t rsa

Step 7: Set Permissions

- cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
- chmod 640 ~/.ssh/authorized_keys

Step 8: SSH to the localhost

- ssh localhost

Step 9: Switch User

- su - hadoop

//If downloaded manually no need to perform step 10 and 11 //

Step 10: Install Hadoop

- wget https://dlcdn.apache.org/hadoop/common/hadoop-3.3.6/hadoop-3.3.6.tar.gz

Step 11

- tar -xvzf hadoop-3.3.6.tar.gz
- mv hadoop-3.3.6 hadoop

Step 12

- dirname $(dirname $(readlink -f $(which java)))
- nano ~/.bashrc

```
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-i386/jre
export HADOOP_HOME=/home/hadoop/hadoop
export HADOOP_INSTALL=$HADOOP_HOME
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME=$HADOOP_HOME
export HADOOP_YARN_HOME=$HADOOP_HOME
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
export PATH=$PATH:$HADOOP_HOME/sbin:$HADOOP_HOME/bin
```

export HADOOP_OPTS="-Djava.library.path=$HADOOP_HOME/lib/native"

- source ~/.bashrc

Step 13

cd hadoop/etc/hadoop

- gedit core-site.xml
  <configuration>
  <property>
  <name>fs.defaultFS</name>
  <value>hdfs://localhost:9000</value>
  </property>
  </configuration>

Step 14

cd hadoop/etc/hadoop

- gedit mapred-site xml
  <configuration>
  <property>
  <name>mapreduce.job.tracker</name>
  <value>localhost:9870</value>
  </property>
  </configuration>

Step 15

- gedit hadoop-env.sh
  export JAVA_HOME= dirname $(dirname $(readlink -f $(which java)))

- gedit Hdfs-site.xml
  <configuration>
  <property>
  <name>dfs.replication</name>
  <value>1</value>
  </property>
  </configuration>

Step 16

- hadoop namenode -format
  Switch to hadoop/sbin
- ./start-all.sh
- http://localhost:9870
- jps

Step 17

- hadoop fs -mkdir /user
- hadoop fs -mkdir /user/hadoop
- hadoop fs -mkdir /user/hadoop/input

- hadoop fs -put input.txt /user/hadoop/input

Step 18

Download Eclipse https://www.eclipse.org/downloads/packages/release/kepler/sr2

Step 19

Now you can run the Hadoop job on the cluster:

- hadoop jar WordCount.jar WordCount /user/hadoop/input /user/hadoop/output

Step 19

Check the Output:

hadoop fs -cat /user/hadoop/output/part-r-00000

```
import org.apache.hadoop.conf.Configured;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;
import org.apache.hadoop.util.Tool;
import org.apache.hadoop.util.ToolRunner;


public class DriverClass extends Configured implements Tool {

	@Override
	public int run(String[] arg0) throws Exception {
		Job job= new Job(getConf(),"KRN");
		job.setInputFormatClass(TextInputFormat.class);
		job.setOutputFormatClass(TextOutputFormat.class);
```

```
		job.setMapperClass(MapperClass.class);
		job.setReducerClass(ReducerClass.class);
		job.setMapOutputKeyClass(Text.class);
		job.setMapOutputValueClass(LongWritable.class);
		job.setOutputKeyClass(Text.class);
		job.setOutputValueClass(IntWritable.class);
		FileInputFormat.addInputPath(job, new Path("input"));
		FileOutputFormat.setOutputPath(job, new Path("out"));
		job.setJarByClass(DriverClass.class);// to Run on hadoop
		job.waitForCompletion(true);//Logs Display
		return 0;
	}
	public static void main(String[] args) throws Exception {
		ToolRunner.run(new DriverClass(), args);

	}
}


import java.io.IOException;

import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;




public class MapperClass extends Mapper<LongWritable, Text, Text, LongWritable>{
	@Override
	protected void map(LongWritable key, Text value, Context context)
			throws IOException, InterruptedException {

		String w[] =value.toString().split(" ");
		for (String word:w)
		{
			context.write(new Text(word), new LongWritable(1));

		}

	}


}

import java.io.IOException;

import org.apache.hadoop.io.IntWritable;
```

```
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;


public class ReducerClass extends Reducer<Text, LongWritable, Text, IntWritable> {

	@Override
	protected void reduce(Text key, Iterable<LongWritable> value,Context context)
			throws IOException, InterruptedException {
		int cnt=0;
		for(LongWritable i:value)
		{
			cnt=cnt+1;
		}
		context.write(key, new IntWritable(cnt));

	}
}
```