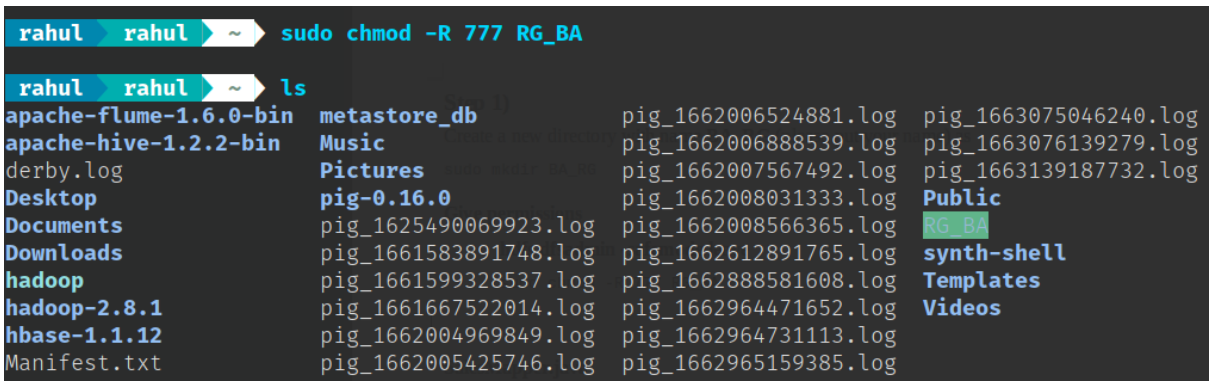
Step 1) Create a new directory with name BA_RG (please put your name) **as**

sudo mkdir BA_RG

- **Give permissions**

  hdfs dfsadmin -safemode leave

  sudo chmod -R 777 RG_BA



- Add the below code in the directory you have created

  Give: **cd RG_BA**

For mapping process :

SalesMapper.java

```
package SalesCountry;

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.mapred.*;

public class SalesMapper extends MapReduceBase implements Mapper <LongWritable, Text, Text,
IntWritable> {
    private final static IntWritable one = new IntWritable(1);

    public void map(LongWritable key, Text value, OutputCollector <Text, IntWritable> output,
Reporter reporter) throws IOException {

        String valueString = value.toString();
        String[] SingleCountryData = valueString.split(",");
        output.collect(new Text(SingleCountryData[7]), one);
    }
```

```
}
```

For Reducer process

SalesCountryReducer.java

```
package SalesCountry;

import java.io.IOException;
import java.util.*;

import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.*;

public class SalesCountryReducer extends MapReduceBase implements Reducer<Text, IntWritable,
Text, IntWritable> {

	public void reduce(Text t_key, Iterator<IntWritable> values,
OutputCollector<Text,IntWritable> output, Reporter reporter) throws IOException {
		Text key = t_key;
		int frequencyForCountry = 0;
		while (values.hasNext()) {
			// replace type of value with the actual type of our value
			IntWritable value = (IntWritable) values.next();
			frequencyForCountry += value.get();

		}
		output.collect(key, new IntWritable(frequencyForCountry));
	}
}
```

SalesCountryDriver.java

```
package SalesCountry;

import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapred.*;

public class SalesCountryDriver {
   public static void main(String[] args) {
      JobClient my_client = new JobClient();
      // Create a configuration object for the job
      JobConf job_conf = new JobConf(SalesCountryDriver.class);

      // Set a name of the Job
      job_conf.setJobName("SalePerCountry");
```

```
        // Specify data type of output key and value
        job_conf.setOutputKeyClass(Text.class);
        job_conf.setOutputValueClass(IntWritable.class);

        // Specify names of Mapper and Reducer Class
        job_conf.setMapperClass(SalesCountry.SalesMapper.class);
        job_conf.setReducerClass(SalesCountry.SalesCountryReducer.class);

        // Specify formats of the data type of Input and output
        job_conf.setInputFormat(TextInputFormat.class);
        job_conf.setOutputFormat(TextOutputFormat.class);

        // Set input and output directories using command line arguments,
        //arg[0] = name of input directory on HDFS, and arg[1] =  name of output directory to be
created to store the output file.

        FileInputFormat.setInputPaths(job_conf, new Path(args[0]));
        FileOutputFormat.setOutputPath(job_conf, new Path(args[1]));

        my_client.setConf(job_conf);
        try {
            // Run the job
            JobClient.runJob(job_conf);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Check the file permissions of all these files (there will be SalesJan2009.csv as I had already added. You need not add here, please wait)

```
rahul  rahul  ~/RG_BA  ls -al
total 144
drwxrwxrwx  2 rahul rahul   4096 Nov 30 10:19 .
drwxr-xr-x 27 rahul rahul   4096 Nov 30 10:16 ..
-rw-rw-r--  1 rahul rahul   1529 Nov 30 10:19 SalesCountryDriver.java
-rw-rw-r--  1 rahul rahul    749 Nov 30 10:19 SalesCountryReducer.java
-rw-rw-r--  1 rahul rahul 123637 Nov 30 10:14 SalesJan2009.csv
-rw-rw-r--  1 rahul rahul    661 Nov 30 10:17 SalesMapper.java
```

and if 'read' permissions are missing then grant the same-

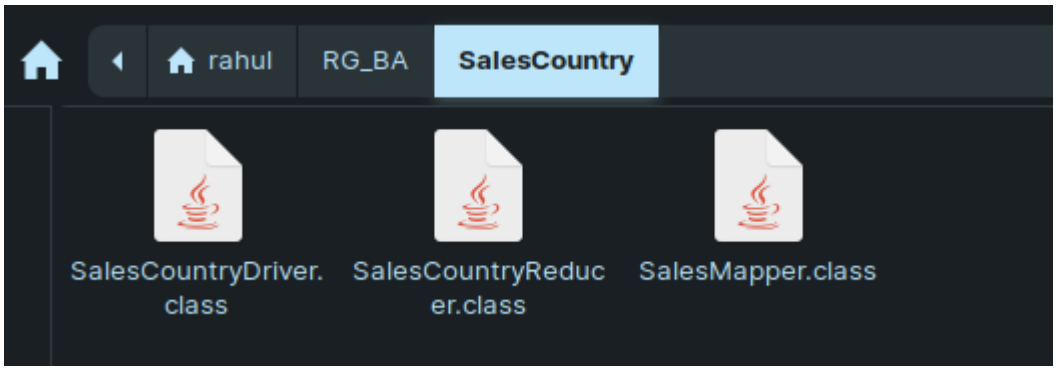
Step 2) Export classpath as shown in the below Hadoop example

export CLASSPATH="$HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-client-core-2.8.1.jar:$HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-client-common-2.8.1.jar:$HADOOP_HOME/share/hadoop/common/hadoop-common-2.8.1.jar:~/MapReduceTutorial/SalesCountry/*:$HADOOP_HOME/lib/*"

Step 3) Compile Java files (these files are present in directory Final-MapReduceHandsOn**). Its class files will be put in the package directory**

javac -d . SalesMapper.java SalesCountryReducer.java SalesCountryDriver.java

This compilation will create a directory in a current directory named with package name specified in the java source file (i.e. **SalesCountry** in our case) and put all compiled class files in it.

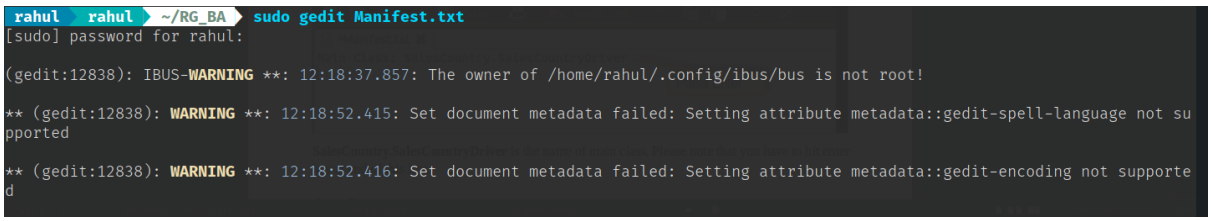


Step 4) Create a new file Manifest.txt

sudo gedit Manifest.txt

add following lines to it,

Main-Class: SalesCountry.SalesCountryDriver



SalesCountry.SalesCountryDriver is the name of main class. Please note that you have to hit enter key at end of this line.

Step 5)

Create a Jar file

jar cfm ProductSalePerCountry.jar Manifest.txt SalesCountry/*.class

```
rahul  rahul  ~/RG_BA  jar cfm ProductSalePerCountry.jar Manifest.txt SalesCountry/*.class

rahul  rahul  ~/RG_BA  ls
Manifest.txt               SalesCountry          SalesCountryReducer.java  SalesMapper.java
ProductSalePerCountry.jar  SalesCountryDriver.java  SalesJan2009.csv
```

Check that the jar file is created

Step 6) Start Hadoop

(only if u get error)

stop-dfs.sh

stop-yarn.sh)

```
rahul  rahul  ~/RG_BA  stop-dfs.sh
Stopping namenodes on [localhost]
localhost: stopping namenode
localhost: stopping datanode
Stopping secondary namenodes [0.0.0.0]
0.0.0.0: stopping secondarynamenode

rahul  rahul  ~/RG_BA  stop-yarn.sh
stopping yarn daemons
stopping resourcemanager
localhost: stopping nodemanager
localhost: nodemanager did not stop gracefully after 5 seconds: killing with kill -9
no proxyserver to stop
```

$HADOOP_HOME/sbin/start-dfs.sh

$HADOOP_HOME/sbin/start-yarn.sh

```
rahul  rahul  ~/RG_BA  $HADOOP_HOME/sbin/start-dfs.sh
Starting namenodes on [localhost]
localhost: starting namenode, logging to /home/rahul/hadoop-2.8.1/logs/hadoop-rahul-namenode-rahul.out
localhost: starting datanode, logging to /home/rahul/hadoop-2.8.1/logs/hadoop-rahul-datanode-rahul.out
Starting secondary namenodes [0.0.0.0]
0.0.0.0: starting secondarynamenode, logging to /home/rahul/hadoop-2.8.1/logs/hadoop-rahul-secondarynamenode-rahul.out

rahul  rahul  ~/RG_BA  $HADOOP_HOME/sbin/start-yarn.sh
starting yarn daemons
starting resourcemanager, logging to /home/rahul/hadoop-2.8.1/logs/yarn-rahul-resourcemanager-rahul.out
localhost: starting nodemanager, logging to /home/rahul/hadoop-2.8.1/logs/yarn-rahul-nodemanager-rahul.out
```

Step 7) Copy the File SalesJan2009.csv **into** ~/inputMapReduce

Now Use below command to copy **~/inputMapReduce** to HDFS.

We can safely ignore this warning.

Verify whether a file is actually copied or not.

$HADOOP_HOME/bin/hdfs dfs -ls /inputMapReduce

```
rahul  rahul  ~/RG_BA  $HADOOP_HOME/bin/hdfs dfs -ls /inputMapReduce
Found 1 items
-rw-r--r--   1 rahul supergroup     123637 2022-11-30 14:12 /inputMapReduce/SalesJan2009.csv
```

Step 8) Run MapReduce job

$HADOOP_HOME/bin/hadoop jar ProductSalePerCountry.jar /inputMapReduce /mapreduce_output_sales

```
rahul  rahul  ~/RG_BA  $HADOOP_HOME/bin/hadoop jar ProductSalePerCountry.jar /inputMapReduce /mapreduce_output_sales
22/11/30 14:12:48 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
22/11/30 14:12:48 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
22/11/30 14:12:48 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interf
ace and execute your application with ToolRunner to remedy this.
22/11/30 14:12:49 WARN hdfs.DataStreamer: Caught exception
java.lang.InterruptedException
        at java.lang.Object.wait(Native Method)
        at java.lang.Thread.join(Thread.java:1257)
        at java.lang.Thread.join(Thread.java:1331)
        at org.apache.hadoop.hdfs.DataStreamer.closeResponder(DataStreamer.java:927)
        at org.apache.hadoop.hdfs.DataStreamer.endBlock(DataStreamer.java:578)
        at org.apache.hadoop.hdfs.DataStreamer.run(DataStreamer.java:755)
22/11/30 14:12:49 INFO mapred.FileInputFormat: Total input files to process : 1
22/11/30 14:12:49 INFO mapreduce.JobSubmitter: number of splits:2
22/11/30 14:12:49 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1669797056315_0001
22/11/30 14:12:49 INFO impl.YarnClientImpl: Submitted application application_1669797056315_0001
22/11/30 14:12:50 INFO mapreduce.Job: The url to track the job: http://rahul:8088/proxy/application_1669797056315_0001/
22/11/30 14:12:50 INFO mapreduce.Job: Running job: job_1669797056315_0001
22/11/30 14:12:58 INFO mapreduce.Job: Job job_1669797056315_0001 running in uber mode : false
```

This will create an output directory named mapreduce_output_sales on HDFS. Contents of this directory will be a file containing product sales per country.

Step 9)

The result can be seen through command interface as,

$HADOOP_HOME/bin/hdfs dfs -cat /mapreduce_output_sales/part-00000

```
rahul  rahul  ~/RG_BA  $HADOOP_HOME/bin/hdfs dfs -cat /mapreduce_output_sales/part-00000
Argentina       1
Australia       38
Austria 7
Bahrain 1
Belgium 8
Bermuda 1
Brazil  5
Bulgaria        1
CO      1
Canada  76
Cayman Isls     1
China   1
Costa Rica      1
Country 1
Czech Republic  3
Denmark 15
Dominican Republic      1
Finland 2
France  27
```