## Mapreduce 2 Programs

Running Mapreduce using 2 commands

Part A:

<https://hadoop.apache.org/docs/stable/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html>

Do all these things from hadoopuser:

Adding Java class link in environmental variables

*export HADOOP\_CLASSPATH=/usr/lib/jvm/java-8-oracle/lib/tools.jar*

*editor ~/.bashrc*

*source ~/.bashrc*

Using HDFS commands

*hadoop fs -mkdir /user*

*#Check if directory got created*

[*http://localhost:50070/explorer.html#/*](http://localhost:50070/explorer.html#/)

*hadoop fs -mkdir /user/joe*

*hadoop fs -mkdir /user/joe/wordcount*

*hadoop fs -mkdir /user/joe/wordcount/input*

*Likewise make other directories*

*Check: http://localhost:50070*

Create file01 and file02 using nano and type the content in the file

*nano file01*

Hello World Bye World

*nano file02*

Hello Hadoop Goodbye Hadoop

Move file to the file system

*hadoop fs -put file01 /user/joe/wordcount/input*

*hadoop fs -put file02 /user/joe/wordcount/input*

Check if the file has the data by using CAT on hado

Hello World Bye World

*hadoop fs -cat /user/joe/wordcount/input/file02*

Hello Hadoop Goodbye Hadoop

Copy the Java code from the slides and compile: copy into WordCount.java

*nano WordCount.java*

Slide:

<https://docs.google.com/presentation/d/16qwxOfGVbgAFnYMIXvKARdlnZtwrzO3lZQu809RY0_g/edit#slide=id.g3d4a48ff47_1_22>

Paste the below java code

import java.io.IOException;  
import java.util.Iterator;  
import java.util.StringTokenizer;  
  
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.mapred.FileInputFormat;  
import org.apache.hadoop.mapred.FileOutputFormat;  
import org.apache.hadoop.mapred.JobClient;  
import org.apache.hadoop.mapred.JobConf;  
import org.apache.hadoop.mapred.MapReduceBase;  
import org.apache.hadoop.mapred.Mapper;  
import org.apache.hadoop.mapred.OutputCollector;  
import org.apache.hadoop.mapred.Reducer;  
import org.apache.hadoop.mapred.Reporter;  
import org.apache.hadoop.mapred.TextInputFormat;  
import org.apache.hadoop.mapred.TextOutputFormat;  
  
public class WordCount {  
  
 public static class Map extends MapReduceBase implements  
 Mapper<LongWritable, Text, Text, IntWritable> {  
  
 @Override  
 public void map(LongWritable key, Text value, OutputCollector<Text, IntWritable> output, Reporter reporter)  
 throws IOException {  
  
 String line = value.toString();  
 StringTokenizer tokenizer = new StringTokenizer(line);  
  
 while (tokenizer.hasMoreTokens()) {  
 value.set(tokenizer.nextToken());  
 output.collect(value, new IntWritable(1));  
 }  
  
 }  
 }  
  
 public static class Reduce extends MapReduceBase implements  
 Reducer<Text, IntWritable, Text, IntWritable> {  
  
 @Override  
 public void reduce(Text key, Iterator<IntWritable> values,  
 OutputCollector<Text, IntWritable> output, Reporter reporter)  
 throws IOException {  
 int sum = 0;  
 while (values.hasNext()) {  
 sum += values.next().get();  
 }  
  
 output.collect(key, new IntWritable(sum));  
 }  
 }  
  
 public static void main(String[] args) throws Exception {  
  
 JobConf conf = new JobConf(WordCount.class);  
 conf.setJobName("wordcount");  
  
 conf.setOutputKeyClass(Text.class);  
 conf.setOutputValueClass(IntWritable.class);  
  
 conf.setMapperClass(Map.class);  
 conf.setReducerClass(Reduce.class);  
  
 conf.setInputFormat(TextInputFormat.class);  
 conf.setOutputFormat(TextOutputFormat.class);  
  
 FileInputFormat.setInputPaths(conf, new Path(args[0]));  
 FileOutputFormat.setOutputPath(conf, new Path(args[1]));  
  
 JobClient.runJob(conf);  
  
 }  
}

#Compile the code

*hadoop com.sun.tools.javac.Main WordCount.java*

*#Create jar file from class files (jar is a zipped file for many .class files)*

*jar cf wc.jar WordCount\*.class*

*#(Running the final code on terminal)*

*hadoop jar wc.jar WordCount /user/joe/wordcount/input /user/joe/wordcount/output*

*#Checking output on terminal*

*hadoop fs -cat /user/joe/wordcount/output/part-00000*

*#cd /usr/local/hadoop/etc/hadoop*

----------------------------------------------------------------------------------------------------------------------------\

<https://hadoopbigdatanyc.slack.com/messages/CB7BAPMM4/>

<https://www.guru99.com/create-your-first-hadoop-program.html>

java -version

Cd

Go to the palce where all files are located and mothem to haddop user

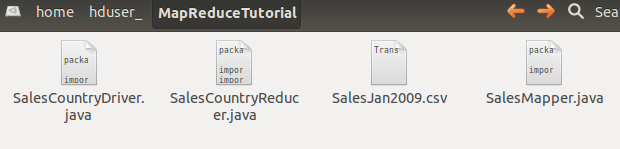
sudo mv \*.\* /home/hadoopuser/MapReduceTutorial

mkdir MapReduceTutorial

chmod -R 777 MapReduceTutorial

#give permission

Download and extract to make below setup - all 4 files should be in Mapreducetutorual folder as shown below:



cd

cd MapReduceTutorial

chmod +r \*.\*



cd $HADOOP\_HOME/share/hadoop/mapreduce/

ls

*#check if you can see the files*

cd $HADOOP\_HOME/share/hadoop/common/

*#check if you can see the files*

cd

nano .bashrc

Change to your hadoop version and paste:

export CLASSPATH="$HADOOP\_HOME/share/hadoop/mapreduce/hadoop-mapreduce-client-core-2.8.4.jar:$HADOOP\_HOME/share/hadoop/mapreduce/hadoop-mapreduce-client-common-2.8.4.jar:$HADOOP\_HOME/share/hadoop/common/hadoop-common-2.8.4.jar:~/MapReduceTutorial/SalesCountry/\*:$HADOOP\_HOME/lib/\*"

source .bashrc

cd MapReduceTutorial/

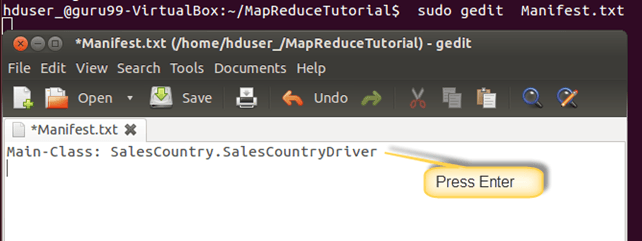
javac -d . SalesMapper.java SalesCountryReducer.java SalesCountryDriver.java

#Make sure you are in MapReduceTutorial

nano **Manifest.txt**

Main-Class: SalesCountry.SalesCountryDriver

cat Manifest.txt



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

*ls*

*# see if you can see the jar file*

*hadoop fs -mkdir /inputMapReduce*

*hadoop fs -put SalesJan2009.csv /inputMapReduce*

*hadoop jar ProductSalePerCountry.jar /inputMapReduce /mapreduce\_output\_sales*

*hadoop fs -cat /mapreduce\_output\_sales/part-00000*