PROJECT CONFIGURATION

Generally:
The project will be implemented in ***Apache Hadoop***, with ***Map-Reduce*** programming technique, in ***JAVA*** programming language and will be tested locally in ***pseudo-distributed mode***.

There is no limitation as to which OS or which IDE you will use as long as everyone uses the same hadoop version 2.7.3 and is able to run the final program in pseudo-distributed mode. Instructions below are given for Windows and Linux Operating Systems with Eclpise Luna IDE.

Prerequisites:
1) HADOOP
   hadoop-2.7.3.tar.gz
2) JAVA
   JDK 8
3) IDE
   Eclipse Luna (recommended)
   NetBeans
   IntelliJ

HADOOP SETUP

Linux Hadoop Setup:
Follow detailed instructions.

Windows Hadoop Setup:
1. Download hadoop-2.7.3.tar.gz , unzip it in C:\
2. Add environmental user and system variables (right click MyComputer->Properties->Change settings->Advanced->Environmental Variables)
   a. User variables-> New
      Variable name: HADOOP_HOME Variable value: C:\hadoop-2.7.3\bin
   b. System Variables-> Path -> Edit->New
      C:\hadoop-2.7.3\bin
3. Download JDK 8, install it
4. Go to folder C:/Program Files/Java/jdk1.8.0_121 and make sure you have installed Java successfully
5. Add environmental system variables (right click MyComputer->Properties->Change settings->Advanced->Environmental Variables)
   a. System Variables-> New
      Variable name: JAVA_HOME Variable value: C:\Progra~1\Java\jdk1.8.0_121
   b. System Variables-> Path -> Edit->New
      %JAVA_HOME%\bin
6. Go to C:\hadoop-2.7.3\etc\hadoop edit core-site.xml

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

7. Go to C:\hadoop-2.7.3\etc\hadoop make a copy of mapred-site.xml.template, rename it to mapred-site.xml and edit it with:

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

8. Go to C:\hadoop-2.7.3\etc\hadoop edit hdfs-site.xml

```
<configuration>
        <property>
                <name>dfs.replication</name>
                <value>1</value>
        </property>
        <property>
                <name>dfs.namenode.dir</name>
                <value>C:\hadoop-2.7.3\data\namenode</value>
        </property>
        <property>
                <name>dfs.datanode.data.dir</name>
                <value>C:\hadoop-2.7.3\data\namenode</value>
        </property>
</configuration>
```

9. Go to C:\hadoop-2.7.3 make new folder data
10. Go to C:\hadoop-2.7.3\etc\hadoop edit yarn-site.xml

```
        <configuration>
                <property>
                        <name>yarn.nodemanager.aux-services</name>
                        <value>mapreduce_shuffle</value>
                </property>
                <property>
                <name>yarn.nodemanager.auxservices.mapreduce.shuffle.class</name>
                        <value>org.apache.hadoop.mapredShuffleHandler</value>
                </property>
        </configuration>
```

11. Go to C:\hadoop-2.7.3\etc\hadoop edit hadoop-env.cmd
    Change line 25 to: set JAVA_HOME= C:\Progra~1\Java\jdk1.8.0_121
12. Download zip file, unzip it, copy files contained in hadooponwindows-master/bin folder and paste them in C:\hadoop-2.7.3\bin folder (replace files in the destination folder)
13. Open cmd and execute:
    a. hadoop -version
    b. javac -version (1.8.0_121)
    c. java -version (“1.8.0_121”)
    d. hdfs namenode –format
    e. go to C:\hadoop-2.7.3\sbin>
        i. start-all.cmd (should open 4 windows-deamons)
        ii. jps
14. Open browser, 2 tabs
    a. localhost:8088
    b. localhost:50070

HADOOP is now running..

To shut down HADOOP type:
C:\hadoop-2.7.3\sbin>stop-all.cmd
(should stop 4 windows-deamons)

Hadoop Integration with Eclipse
(Optional, Recommended)

1. Download hadoop-eclipse-plugin-2.6.0.jar
2. Copy the Map-Reduce plugin for eclipse in the plugins directory of your eclipse folder
   C:\eclipse\eclipse\plugins (windows)
   sudo cp /home/hdfs/Downloads/hadoop-eclipse-plugin-2.6.0.jar /opt/eclipse/plugins/ (linux)
3. Restart the eclipse using the command – /opt/eclipse/eclipse -vm /usr/local/jdk1.8.0_121/bin/java -vmargs -Xmx1024m If elcipse is not coming up because of the X11 forwarding issue, try using “sux” instead of “su” while switching to the “hdfs“. (linux)
4. Start the eclipse: $ECLIPSE_HOME/eclipse (linux)
5. In Eclipse menu click, Window --> Open Perspective --> Others --> MapReduce (both)
6. In bottom MapReduce icon click to Add new Hadoop location (both)
7. Enter MapReduce & HDFS running port (both)
   Map/Reduce(V2) Master Host: localhost Port: 10020
   DFS Master: localhost Port: 9000
8. Once Hadoop location added, DFS Locations will be seen/displayed in Eclipse Project Explorer window, (Windows-->Show View-->Project Explorer) (both)
9. Once Hadoop added, DFS Locations will be seen/displayed in Project Explorer window,
10. Right click DFS location and click to Connect (both)
11. Once connected successfully, it will display all the DFS Folder. (both)
12. Create folder named input. Create folder named output. Create a text file named WordCountSample.txt (any English content will do) upload the file in the input folder.

Running the WordCount Example with Eclipse and Maven
(Optional, Recommended)

Configure Maven:

1. Download apache-maven-3.3.9-bin.zip unzip in C:\
2. Add environmental user and system variables (right click MyComputer->Properties->Change settings->Advanced->Environmental Variables)
   a. User variables-> New
      Variable name: MAVEN_HOME Variable value C:\apache-maven-3.3.9\bin
   b. System Variables-> Path -> Edit->New
      %MAVEN_HOME%\bin
3. Open cmd prompt as administrator and type below command to verify if maven installation.
   a. C:\Windows\System32>mvn -version

```
Apache Maven 3.3.9 (bb52d8502b132ec0a5a3f4c09453c07478323dc5; 2015-11-10T18:41:47+02:00)
Maven home: C:\apache-maven-3.3.9\bin\..
Java version: 1.8.0_121, vendor: Oracle Corporation
Java home: C:\Program Files\Java\jdk1.8.0_121\jre
Default locale: en_US, platform encoding: Cp1253
OS name: "windows 10", version: "10.0", arch: "amd64", family: "dos"
```

Configuring New Maven Project in Eclipse:

1. Start Eclipse, go to Window->Open Perspective->Other-> Java and click ok
2. Right Click on the package explorer and select New->Other->Maven Project
3. Click next and check Use default workspace location and next
4. Select maven-archetype-quickstart and next
5. Specify archtype parameters
   a. Group Id: WordCount
   b. Artifact Id: HadoopMapReduce
   c. Package: tuc.softnet.hadoop.mapreduce.example

6. Copy paste to the pom file the following:

```xml
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>WordCount</groupId>
  <artifactId>HadoopMapReduce</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <packaging>jar</packaging>
  <name>HadoopMapReduce</name>
  <url>http://maven.apache.org</url>
  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
  </properties>
  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>3.8.1</version>
      <scope>test</scope>
    </dependency>
<dependency>
   <groupId>org.apache.hadoop</groupId>
   <artifactId>hadoop-annotations</artifactId>
   <version>2.7.3</version>
  </dependency>
  <!-- Hadoop Mapreduce Client Core -->
  <dependency>
   <groupId>org.apache.hadoop</groupId>
   <artifactId>hadoop-mapreduce-client-core</artifactId>
   <version>2.7.3</version>
  </dependency>
  <dependency>
   <groupId>org.apache.hadoop</groupId>
   <artifactId>hadoop-common</artifactId>
   <version>2.7.3</version>
  </dependency>
  <dependency>
   <groupId>org.apache.hadoop</groupId>
   <artifactId>hadoop-hdfs-nfs</artifactId>
   <version>2.7.3</version>
  </dependency>
  <dependency>
   <groupId>org.apache.hadoop</groupId>
   <artifactId>hadoop-hdfs</artifactId>
   <version>2.7.3</version>
  </dependency>

  <dependency>
   <groupId>junit</groupId>
   <artifactId>junit</artifactId>
   <version>3.8.1</version>
   <scope>test</scope>
  </dependency>

  <dependency>
   <groupId>org.apache.hadoop</groupId>
   <artifactId>hadoop-yarn-common</artifactId>
   <version>2.7.3</version>
  </dependency>
  <dependency>
   <groupId>org.apache.hadoop</groupId>
   <artifactId>hadoop-mapreduce-client-common</artifactId>
   <version>2.7.3</version>
  </dependency>
  <dependency>
   <groupId>org.apache.hadoop</groupId>
   <artifactId>hadoop-mapreduce-client-app</artifactId>
   <version>2.7.3</version>
  </dependency>
  <dependency>
   <groupId>org.apache.hadoop</groupId>
   <artifactId>hadoop-mapreduce-client-hs</artifactId>
   <version>2.7.3</version>
  </dependency>
  <dependency>
   <groupId>org.apache.hadoop</groupId>
   <artifactId>hadoop-mapreduce-client-hs-plugins</artifactId>
   <version>2.7.3</version>
  </dependency>
  <dependency>
   <groupId>org.apache.hadoop</groupId>
   <artifactId>hadoop-yarn-api</artifactId>
   <version>2.7.3</version>
  </dependency>
  <dependency>
   <groupId>org.apache.hadoop</groupId>
   <artifactId>hadoop-yarn-server-web-proxy</artifactId>
   <version>2.7.3</version>
  </dependency>
  <dependency>
   <groupId>org.apache.hadoop</groupId>
   <artifactId>hadoop-yarn-server-sharedcachemanager</artifactId>
   <version>2.7.3</version>
  </dependency>
  <dependency>
   <groupId>org.apache.hadoop</groupId>
   <artifactId>hadoop-yarn-server-resourcemanager</artifactId>
   <version>2.7.3</version>
  </dependency>
  <dependency>
   <groupId>org.apache.hadoop</groupId>
   <artifactId>hadoop-yarn-server-nodemanager</artifactId>
   <version>2.7.3</version>
  </dependency>
  <dependency>
   <groupId>org.apache.hadoop</groupId>
   <artifactId>hadoop-yarn-server-common</artifactId>
   <version>2.7.3</version>
  </dependency>
  <dependency>
   <groupId>org.apache.hadoop</groupId>
   <artifactId>hadoop-yarn-server-applicationhistoryservice</artifactId>
   <version>2.7.3</version>
  </dependency>
  <dependency>
   <groupId>org.apache.hadoop</groupId>
   <artifactId>hadoop-yarn-registry</artifactId>
   <version>2.7.3</version>
  </dependency>
  <dependency>
   <groupId>org.apache.hadoop</groupId>
   <artifactId>hadoop-yarn-common</artifactId>
   <version>2.7.3</version>
  </dependency>
  <dependency>
   <groupId>org.apache.hadoop</groupId>
   <artifactId>hadoop-yarn-client</artifactId>
   <version>2.7.3</version>
  </dependency>
  <dependency>
   <groupId>org.apache.hadoop</groupId>
   <artifactId>hadoop-yarn-applications-unmanaged-am-launcher</artifactId>
   <version>2.7.3</version>
  </dependency>
  <dependency>
   <groupId>org.apache.hadoop</groupId>
   <artifactId>hadoop-yarn-applications-distributedshell</artifactId>
   <version>2.7.3</version>
  </dependency>
 </dependencies>
</project>
```

7. Create class WordCountMapper.java

```java
package tuc.softnet.hadoop.mapreduce.example;

import java.io.IOException;
import java.util.StringTokenizer;

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

public class WordCountMapper extends MapReduceBase implements Mapper<LongWritable, Text, Text, IntWritable> {

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

 public void map(LongWritable key, Text value, OutputCollector<Text, IntWritable> collector, Reporter reporter)
   throws IOException {
  String line = value.toString();
  StringTokenizer st = new StringTokenizer(line, " ");
  while (st.hasMoreTokens()) {
   word.set(st.nextToken());
   collector.collect(word, one);
  }

 }
}
```

8. Create class WordCountReducer.java

```java
package tuc.softnet.hadoop.mapreduce.example;

import java.io.IOException;
import java.util.Iterator;

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

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

 public void reduce(Text key, Iterator<IntWritable> values, OutputCollector<Text, IntWritable> outputCollector,
   Reporter reporter) throws IOException {
  int sum = 0;

  while (values.hasNext()) {
   sum = sum + values.next().get();
  }
  outputCollector.collect(key, new IntWritable(sum));

 }
}
```

9. Create class WordCount.java

```
package tuc.softnet.hadoop.mapreduce.example;

import java.net.URI;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
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.RunningJob;
import org.apache.hadoop.mapred.TextOutputFormat;

public class WordCount {

 public static void main(String[] args) throws Exception {
  Configuration conf = new Configuration();

  Path inputPath = new Path("hdfs://127.0.0.1:9000/input/WordCountSample.txt");
  Path outputPath = new Path("hdfs://127.0.0.1:9000/output/result");

  JobConf job = new JobConf(conf, WordCount.class);
  job.setJarByClass(WordCount.class);
  job.setJobName("WordCounterJob");

  FileInputFormat.setInputPaths(job, inputPath);
  FileOutputFormat.setOutputPath(job, outputPath);

  job.setOutputKeyClass(Text.class);
  job.setOutputValueClass(IntWritable.class);
  job.setOutputFormat(TextOutputFormat.class);
  job.setMapperClass(WordCountMapper.class);
  job.setReducerClass(WordCountReducer.class);

  FileSystem hdfs = FileSystem.get(URI.create("hdfs://127.0.0.1:9000"), conf);
  if (hdfs.exists(outputPath))
   hdfs.delete(outputPath, true);

  RunningJob runningJob = JobClient.runJob(job);
  System.out.println("job.isSuccessfull: " + runningJob.isComplete());
 }

}
```

10. Right click on WordCount.java -> Run As -> Run on Hadoop

```
log4j:WARN No appenders could be found for logger (org.apache.hadoop.metrics2.lib.MutableMetricsFactory).
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
job.isSuccessfull: true
```

11. Right click on DFS Location and Disconnect and Refresh, you should see output file generated in localhost->output->result

OTHER USEFUL LINKS

1. HADOOP tutorial for linux
2. HDFS commands shell or command line.