**Deployment**

  deployment encompasses all the processes involved in getting new software or hardware up and running properly in its environment, including installation, [configuration](http://searchexchange.techtarget.com/definition/configuration), running, testing, and making necessary changes. The word [implementation](http://searchcrm.techtarget.com/definition/implementation) is sometimes used to mean the same thing.

Here we mainly focus on Hadoop Installation alone and the remaining eco

Deployment can be done by the following tools which helps in setting the environment

1.Maven

2.SBT

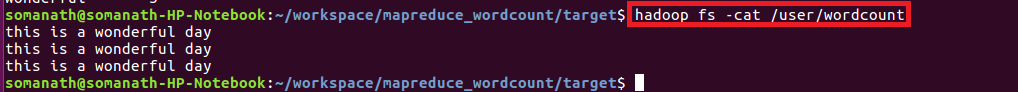
3.Gradle

**Steps to Deploy using Maven**

Here we will see the steps to deploy a simple word count program using maven

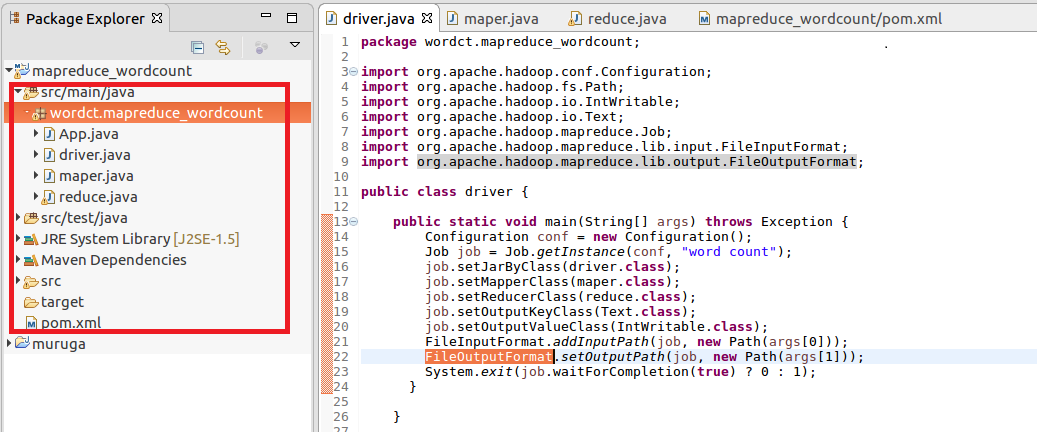
**Build and Deploy hadoop word count program using Maven**

**Input file**

****

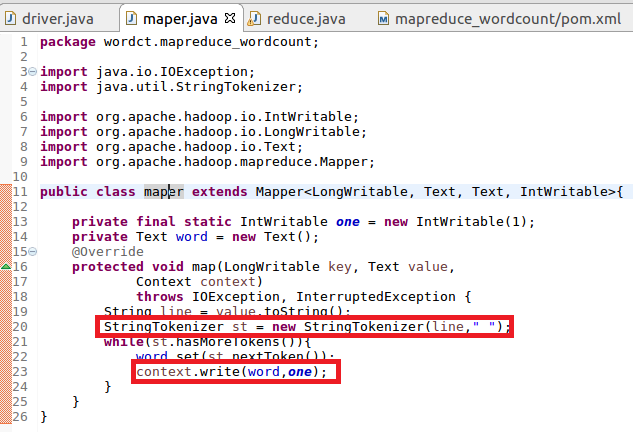
**Step1:We wil create a maven project by file->new ->maven project and maven project is created in the explorer window as highlighted**

**Driver class**



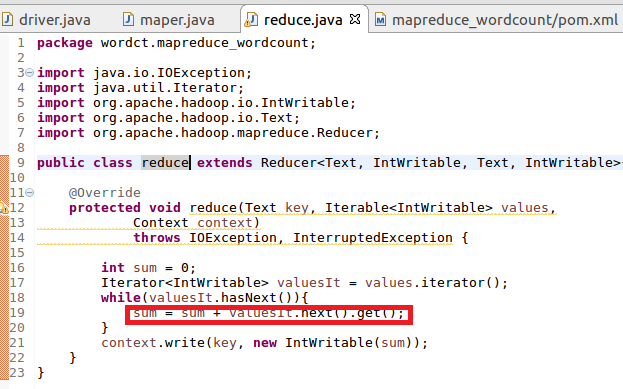
**Mapper:**

In mapper splitting the data by space and sending the word as Key and a count of 1 as value which will be added in reducer



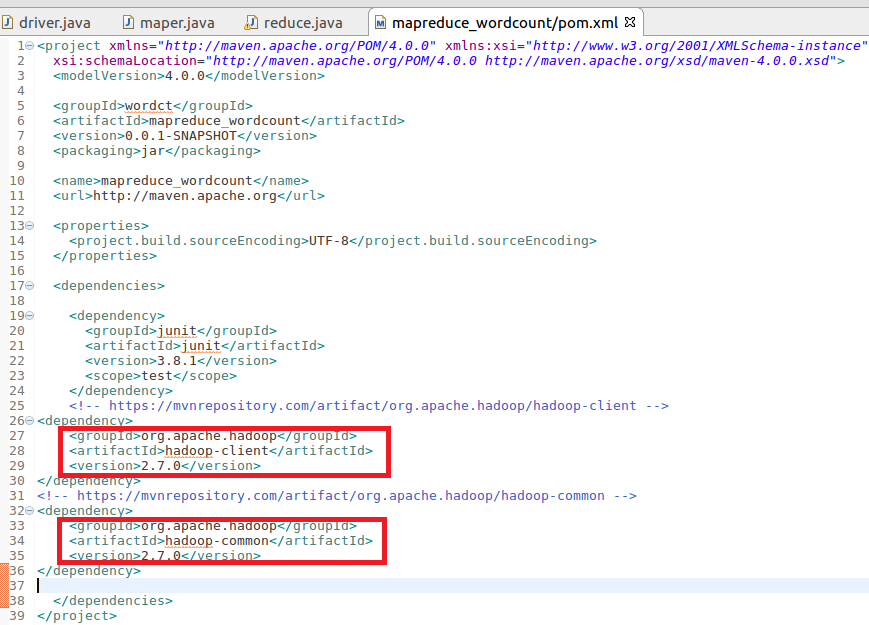
**Reducer**

**In reduce the count of words are made**

****

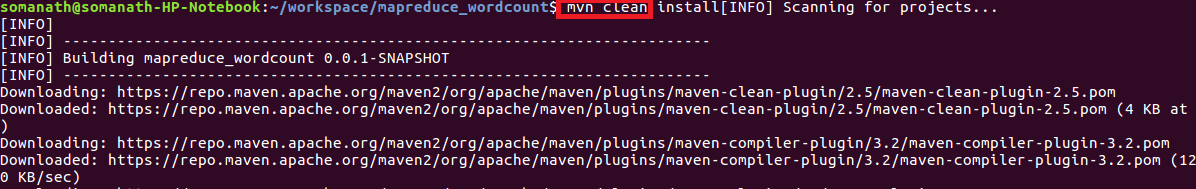
**Step2: Configuring Pom.xml file :**

**Here the necessary dependencies (commonand client) along with basic dependencies (java)is added**

****

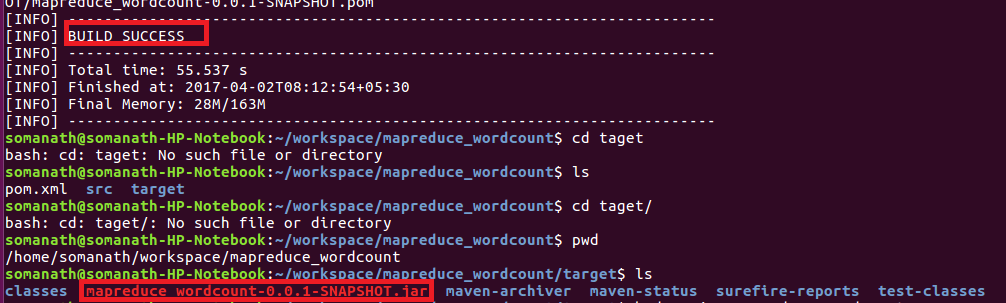
**Step3:Build maven project**

**Maven project is build by maven clean command**

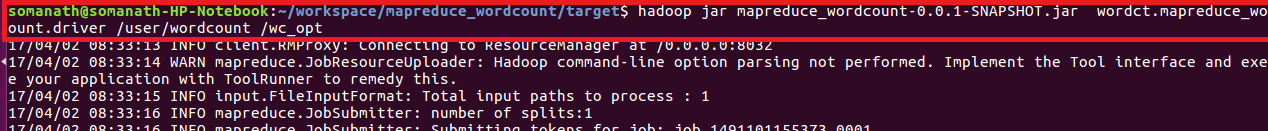
****

**Build successful**

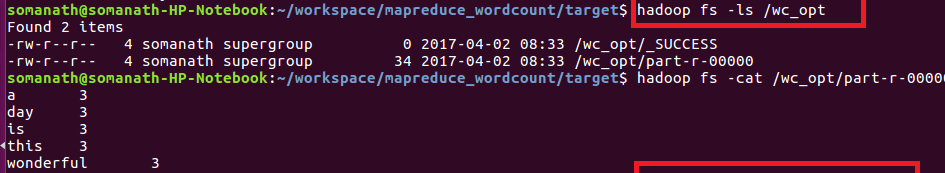
**After successful completion of the building of project success message will be displayed and a jar file will be created in target folder**

****

**Step4:Running Maven jar**

****

**Maven output**

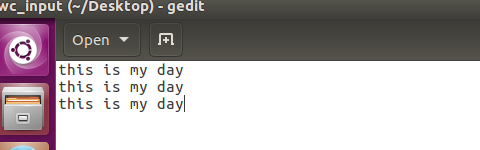
****

**Steps to Deploy using Gradle**

Here we will see the steps to deploy a simple word count program using Gradle

**Build and Deploy hadoop word count program using Gradle**

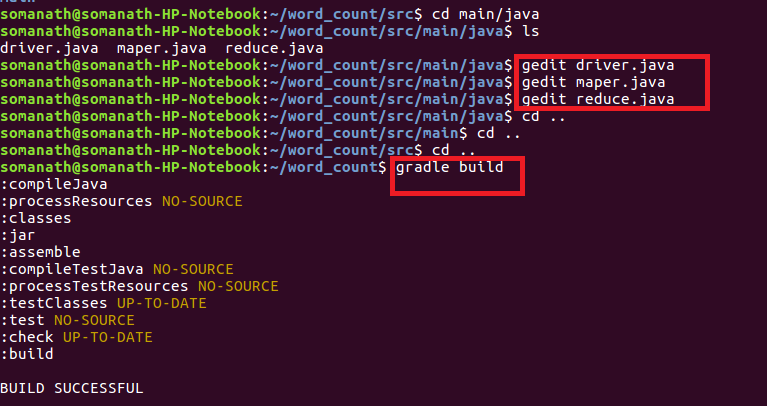
**Input file**

****

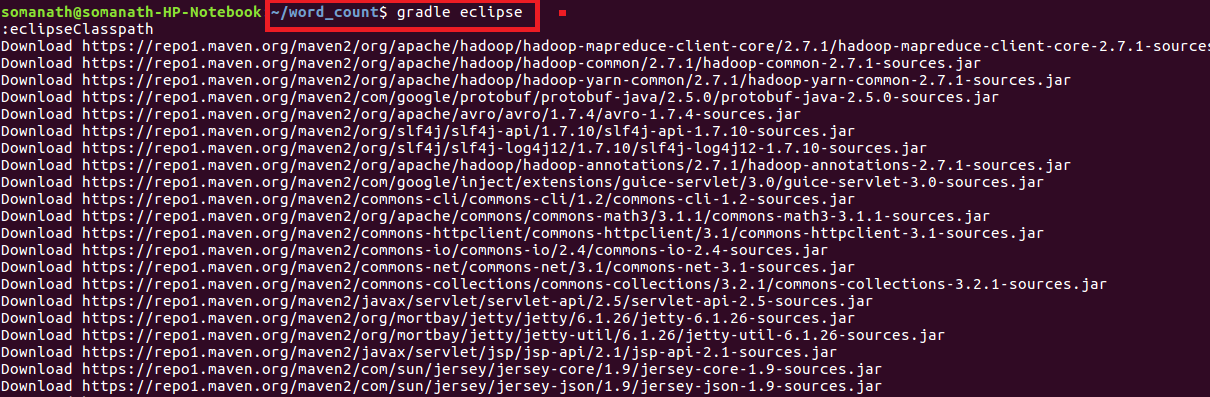
**Step1:CREATED A Project”wordcount” IN TERMINAL AND In source/main/java folder added mapper.jav,driver.java,reducer . Java as shown**

**In root directory added a build.gradle file and added the following dependencies(image shown below of gradle build)**

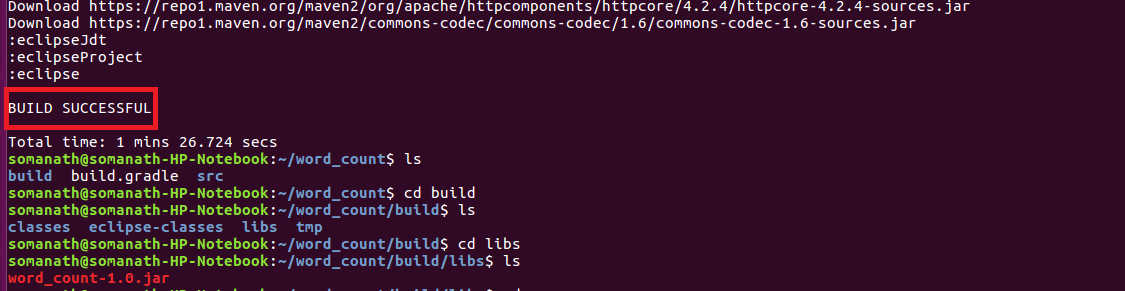
**USE GRADLE BUILD TO build the Gradle project**

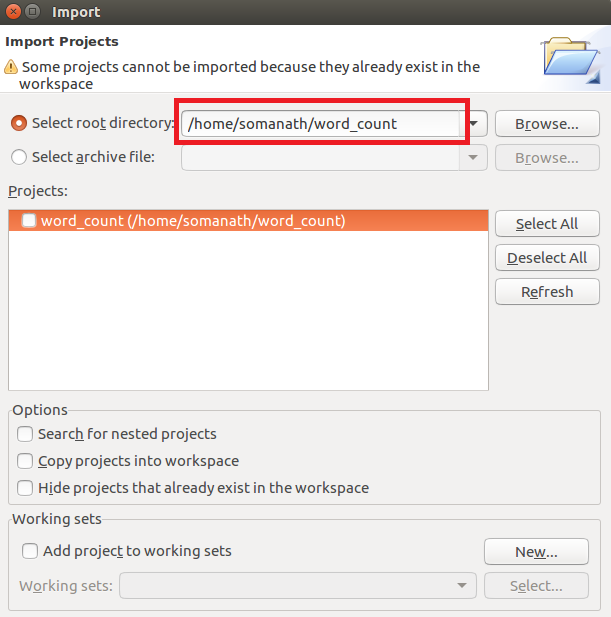


**Step3:Used Gradle eclipse to build the project and classpath that are required for eclipse**

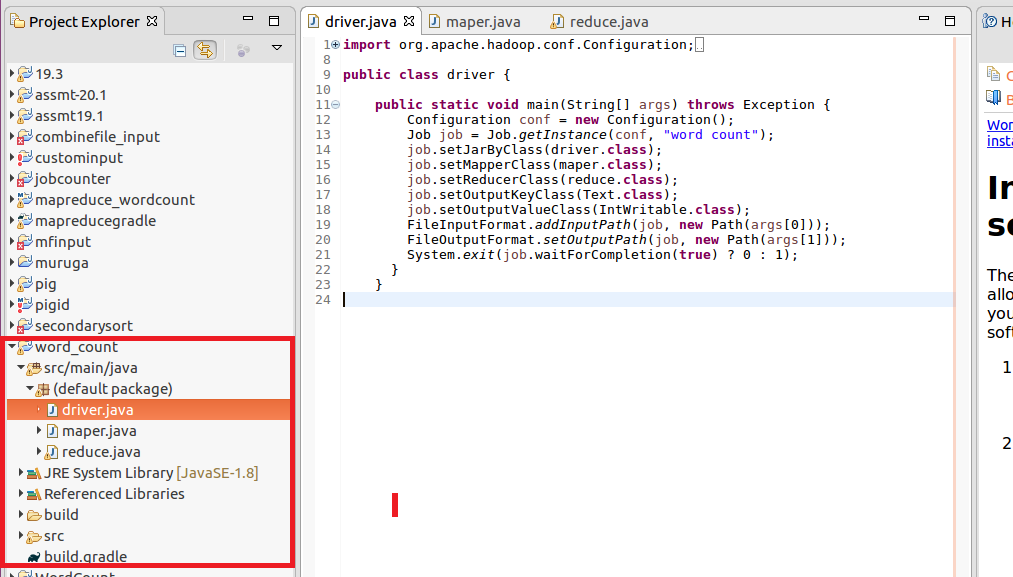


**Build Success: build successful will be displayed and word count jar file will be created in libs**

**Step4:Importing project into Eclipse**

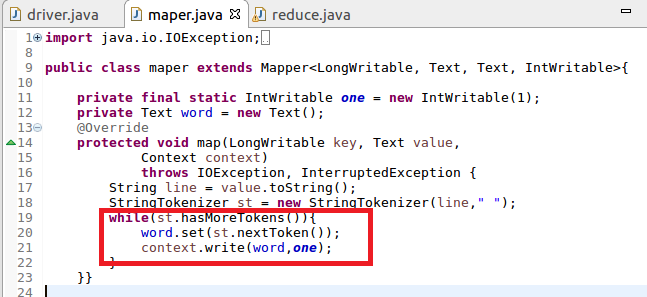


NOW WE CAN SEE IN PROJECT EXPLORER THE WORDCOUNT PROJECT IS CONVERTED TO GRADLE PROJECT INSIDE ECLIPSE



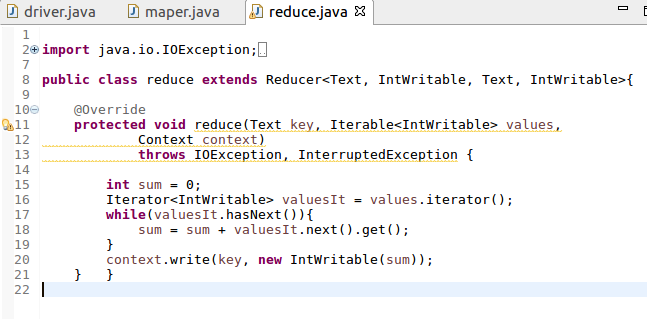
**Mapper:**

In mappersplit the words by space and send word as key and count of 1 as value

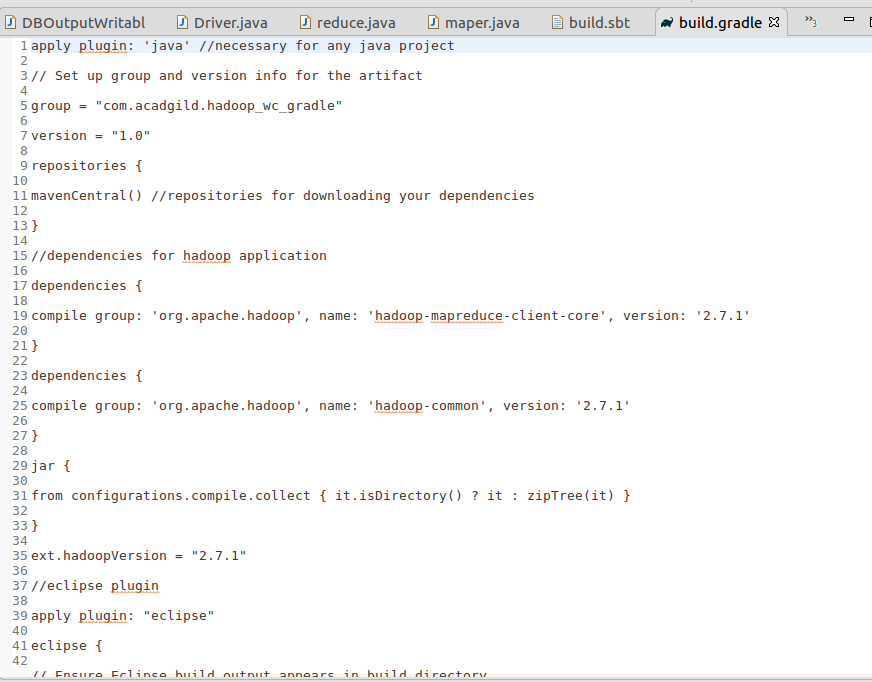


**Reducer**

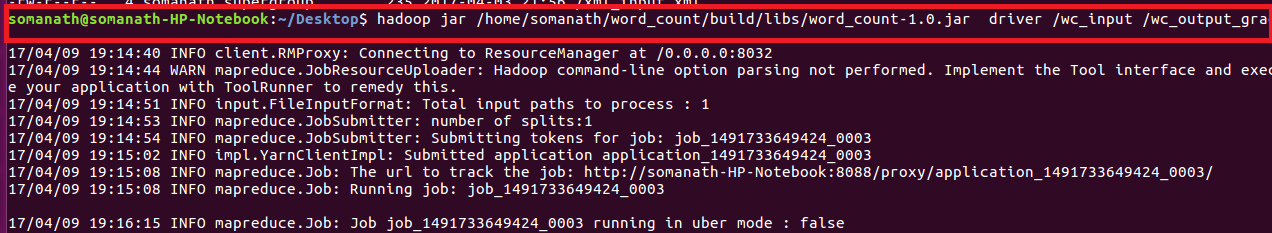
Count the total occurrence of words



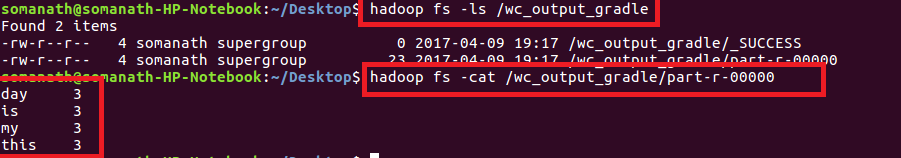
**Build.gradle**



**Step5:Running jar file create d by gradle**



Output

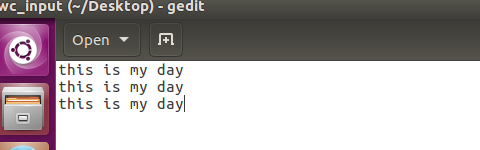


**Steps to Deploy using SBT**

Here we will see the steps to deploy a simple word count program using SBT

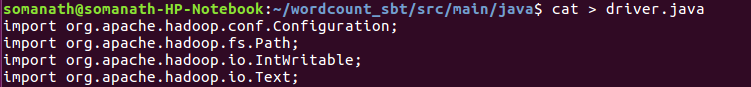
**Build and Deploy hadoop word count program using SBT**

**Input file**

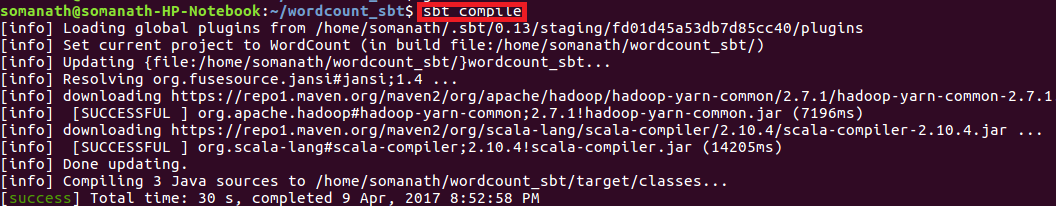
****

**Step1:CREATED A Project”wordcount\_sbt” IN TERMINAL AND In source/main/java folder added mapper.jav,driver.java,reducer and build.sbt in root.**

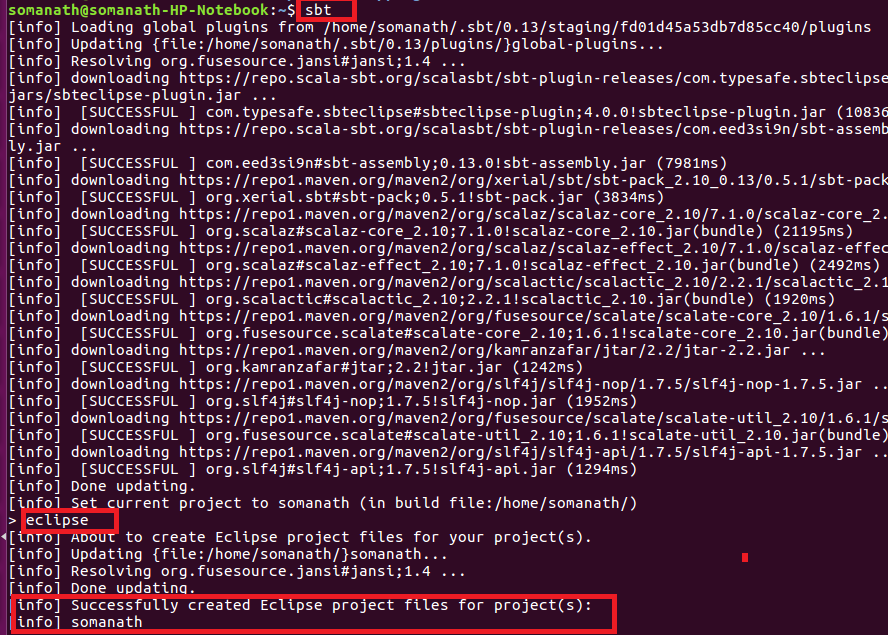
****



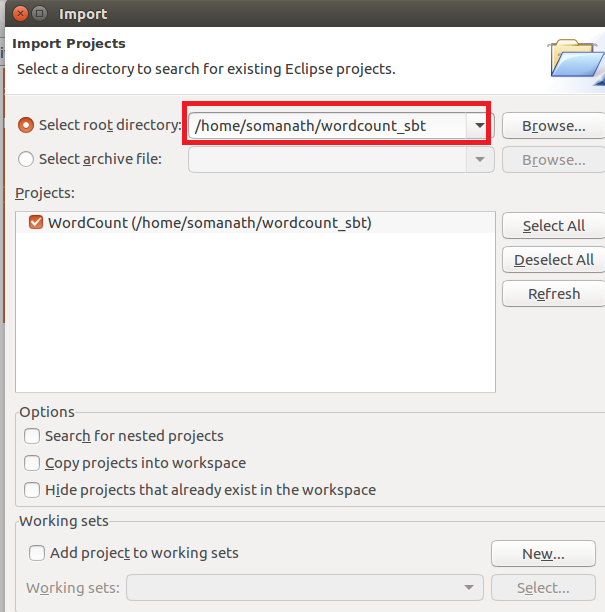
**USE sbt compile command TO build the sbt project**



**Step3:Used sbt eclipse to build the project and classpath that are required for eclipse**

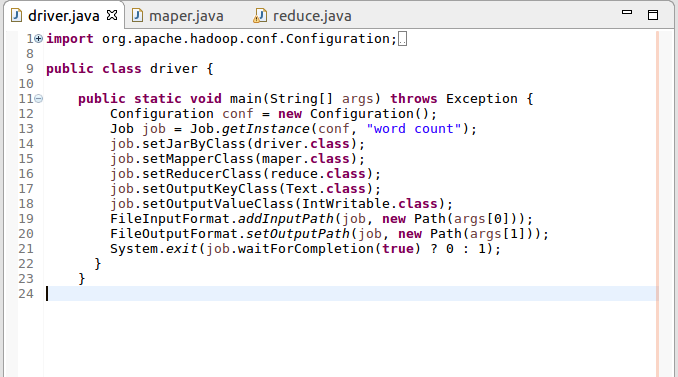


**Step4:Importing project into Eclipse**



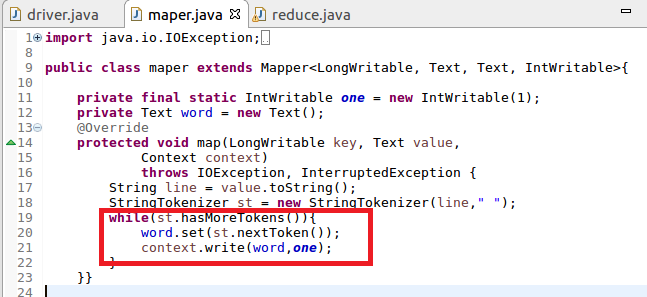
NOW WE CAN SEE IN PROJECT EXPLORER THE WORDCOUNT PROJECT IS CONVERTED TO **Sbt PROJECT** INSIDE ECLIPSE

Driver



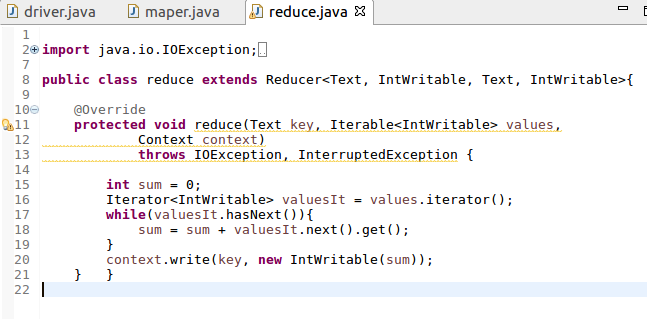
**Mapper:**

In mappersplit the words by space and send word as key and count of 1 as value

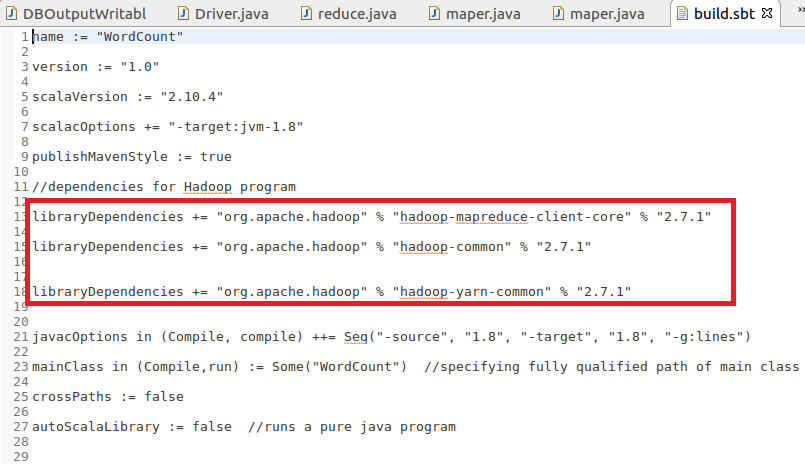


**Reducer**

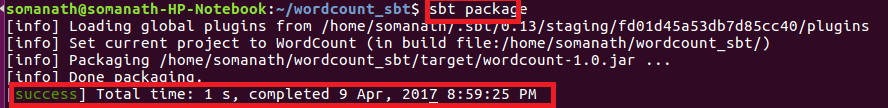
Count the total occurrence of words



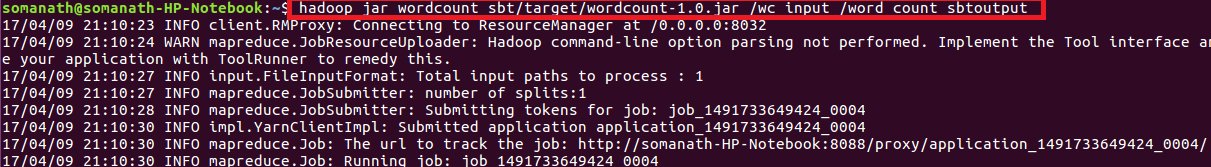
**In root directory added a build.sbt file and added the following dependencies**

****

Step4:  used **sbt package to**  for building the JAR file**.**After successfully packaging the class files as a JAR, you can see the jar file in the **target directory**.



**Step5:Running jar file create d by sbt**

****

Output

