CS246: Mining Massive Datasets **Winter 2016**

Hadoop Tutorial

Due 11:59pm January 12, 2016

General Instructions

The purpose of this tutorial is (1) to get you started with Hadoop and (2) to get you acquainted with the code and homework submission system. Completing the tutorial is optional but by handing in the results in time students will earn 5 points. This tutorial is to be completed individually.

Here you will learn how to write, compile, debug and execute a simple Hadoop program. First part of the assignment serves as a tutorial and the second part asks you to write your own Hadoop program.

Section 1 describes the virtual machine environment. Instead of the virtual machine, you are welcome to setup your own pseudo-distributed or fully distributed cluster if you prefer. Any version of Hadoop that is at least 1.0 will suffice. (For an easy way to set up a cluster, try Cloudera Manager: `http://archive.cloudera.com/cm5/installer/latest/cloudera-manager-installer.bin`.) If you choose to setup your own cluster, you are responsible for making sure the cluster is working properly. The TAs will be unable to help you debug configuration issues in your own cluster.

Section 2 explains how to use the Eclipse environment in the virtual machine, including how to create a project, how to run jobs, and how to debug jobs. Section 2.5 gives an end-to-end example of creating a project, adding code, building, running, and debugging it.

Section 3 is the actual homework assignment. There are no deliverable for sections 1 and 2. In section 3, you are asked to write and submit your own MapReduce job

This assignment requires you to upload the code and hand-in the output for Section 3.

All students should submit the output via GradeScope and upload the code via snap.

GradeScope: To register for GradeScope,

- Create an account on GradeScope if you don't have one already.
- Join CS246 course using Entry Code 92B7E9

Upload the code: Put all the code for a single question into a single file and upload it at `http://snap.stanford.edu/submit/`.

Questions

1 Setting up a virtual machine

- Download and install *VirtualBox* on your machine: http://virtualbox.org/wiki/Downloads
- Download the *Cloudera Quickstart VM* at https://downloads.cloudera.com/demo_vm/virtualbox/cloudera-quickstart-vm-5.5.0-0-virtualbox.zip.
- Uncompress the VM archive. It is compressed with 7-zip. If needed you can download a tool to uncompress the archive at http://www.7-zip.org/.
- Start *VirtualBox* and click *Import Appliance* in the *File* dropdown menu. Click the folder icon beside the location field. Browse to the uncompressed archive folder, select the .ovf file, and click the *Open* button. Click the *Continue* button. Click the *Import* button.
- Your virtual machine should now appear in the left column. Select it and click on *Start* to launch it.
- To verify that the VM is running and you can access it, open a browser to the URL: http://localhost:8088. You should see the resource manager UI. The VM uses port forwarding for the common Hadoop ports, so when the VM is running, those ports on localhost will redirect to the VM.
- *Optional*: Open the Virtual Box preferences ($File \rightarrow Preferences \rightarrow Network$) and select the *Adapter 2* tab. Click the *Enable Network Adapter* checkbox. Select *Host-only Adapter*. If the list of networks is empty, add a new network. Click *OK*. If you do this step, you will be able to connect to the running virtual machine via SSH from the host OS at 192.168.56.101. The username and password are 'cloudera'.

The virtual machine includes the following software

- CentOS 6.4
- JDK 7 (1.7.0_67)
- Hadoop 2.5.0
- Eclipse 4.2.6 (Juno)

The virtual machine runs best with 4096MB of RAM, but has been tested to function with 1024MB. Note that at 1024MB, while it did technically function, it was very slow to start up.

2 Running Hadoop jobs

Generally Hadoop can be run in three modes.

1. **Standalone (or local) mode:** There are no daemons used in this mode. Hadoop uses the local file system as an substitute for HDFS file system. The jobs will run as if there is 1 mapper and 1 reducer.
2. **Pseudo-distributed mode:** All the daemons run on a single machine and this setting mimics the behavior of a cluster. All the daemons run on your machine locally using the HDFS protocol. There can be multiple mappers and reducers.
3. **Fully-distributed mode:** This is how Hadoop runs on a real cluster.

In this homework we will show you how to run Hadoop jobs in Standalone mode (very useful for developing and debugging) and also in Pseudo-distributed mode (to mimic the behavior of a cluster environment).

2.1 Creating a Hadoop project in Eclipse

(There is a plugin for Eclipse that makes it simple to create a new Hadoop project and execute Hadoop jobs, but the plugin is only well maintained for Hadoop 1.0.4, which is a rather old version of Hadoop. There is a project at https://github.com/winghc/hadoop2x-eclipse-plugin that is working to update the plugin for Hadoop 2.0. You can try it out if you like, but your milage may vary.)

To create a project:

1. Open Eclipse. If you just launched the VM, you may have to close the Firefox window to find the Eclipse icon on the desktop.
2. Right-click on the *training* node in the Package Explorer and select *Copy*. See Figure 1.

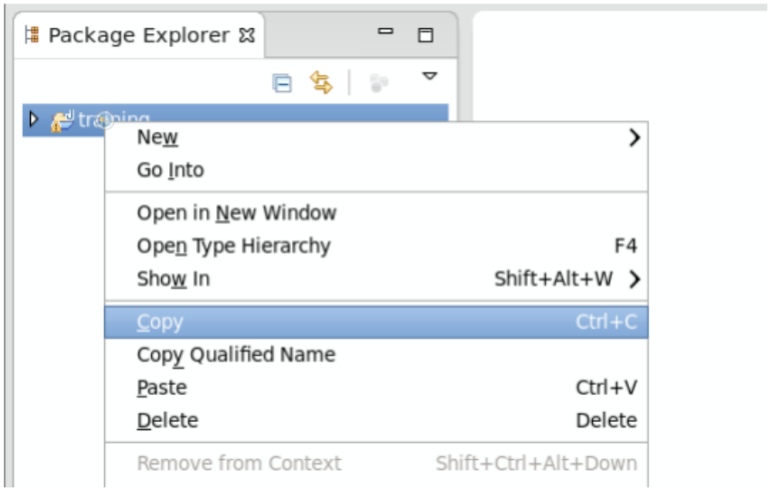

Figure 1: Create a Hadoop Project.

3. Right-click on the *training* node in the Package Explorer and select *Paste* . See Figure 2.

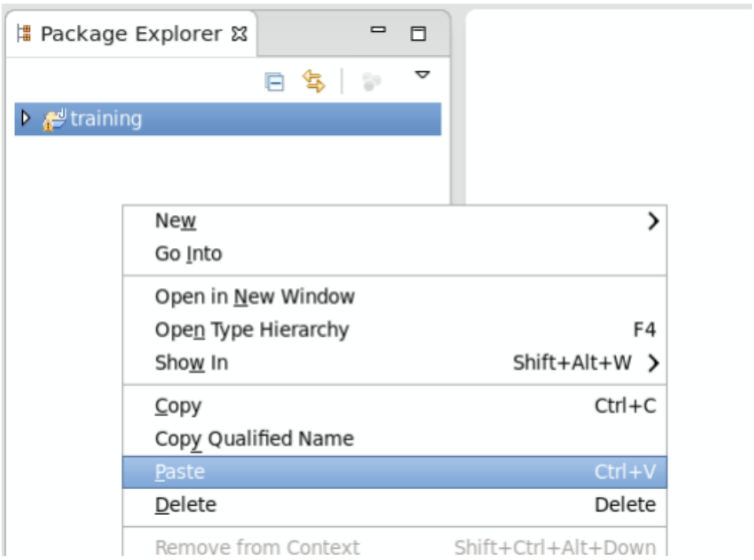

Figure 2: Create a Hadoop Project.

4. In the pop-up dialog, enter the new project name in the *Project Name* field and click *OK*. See Figure 3.

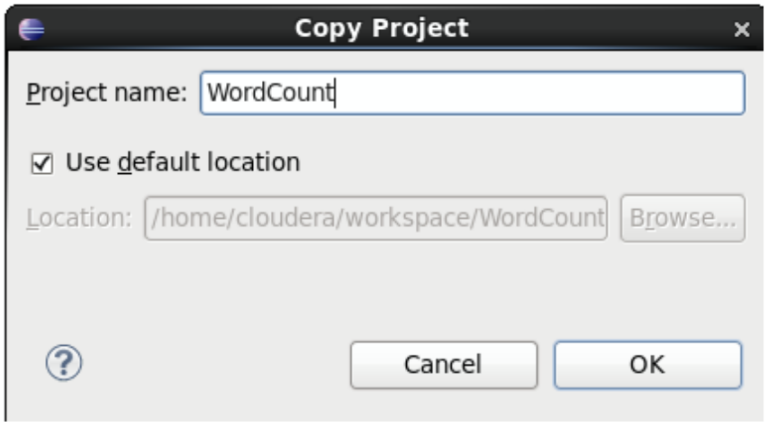

Figure 3: Create a Hadoop Project.

5. Modify or replace the stub classes found in the `src` directory as needed.

2.2 Running Hadoop jobs in standalone mode

Once you've created your project and written the source code, to run the project in standalone mode, do the following:

1. Right-click on the project and select *Run As* → *Run Configurations*. See Figure 4.

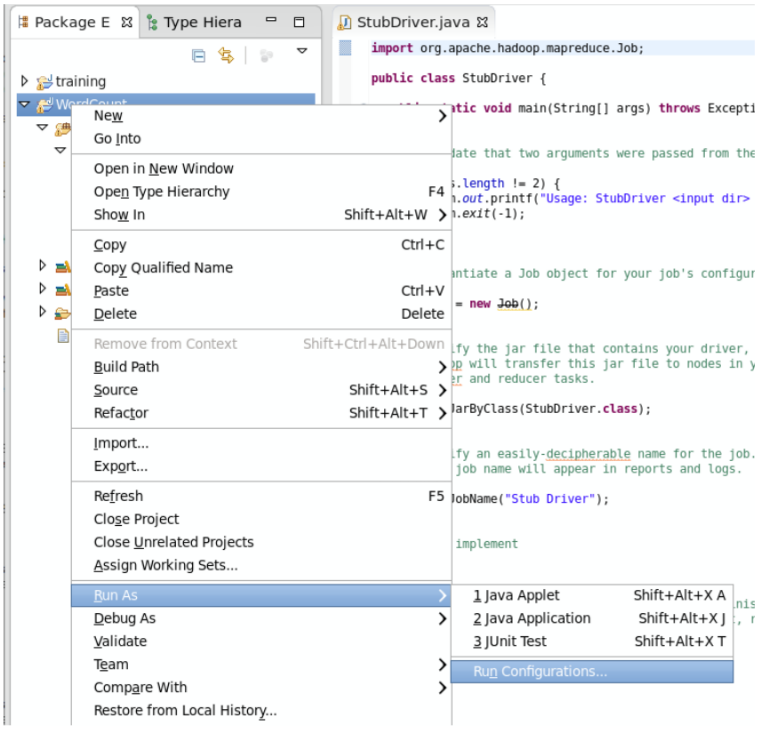

Figure 4: Run a Hadoop Project.

2. In the pop-up dialog, select the *Java Application* node and click the New launch configuration button in the upper left corner. See Figure 5.

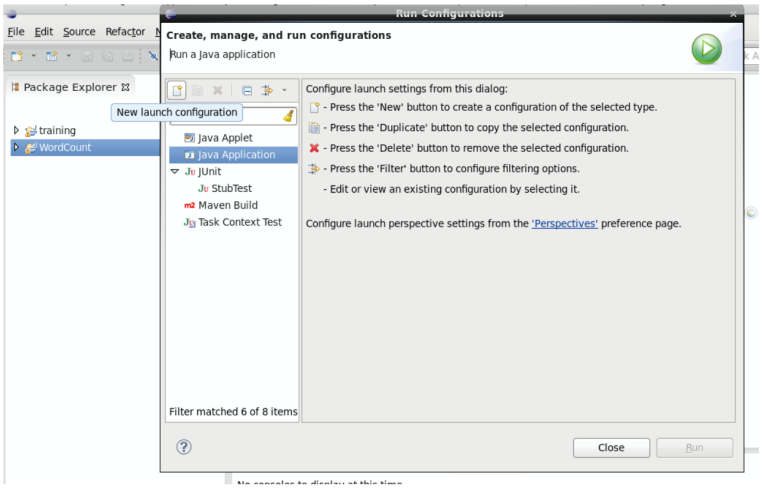

Figure 5: Run a Hadoop Project.

3. Enter a name in the *Name* field and the name of the main class in the *Main class* field. See Figure 6.

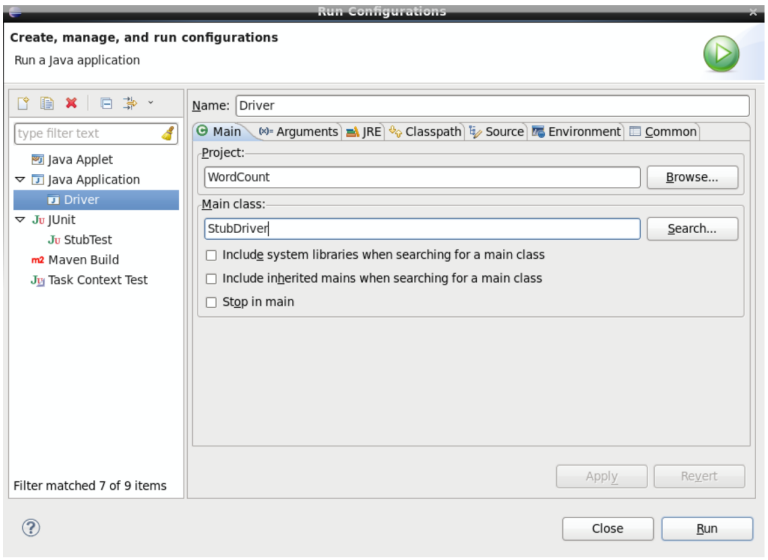

Figure 6: Run a Hadoop Project.

4. Switch to the *Arguments* tab and input the required arguments. Click *Apply*. See Figure 7. To run the job immediately, click on the *Run* button. Otherwise click *Close* and complete the following step.

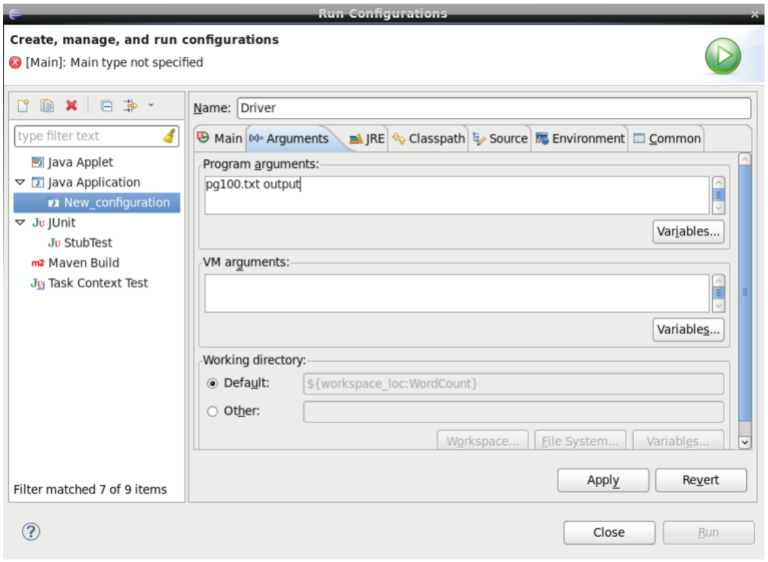

Figure 7: Run a Hadoop Project.

5. Right-click on the project and select *Run As* → *Java Application*. See Figure 8.

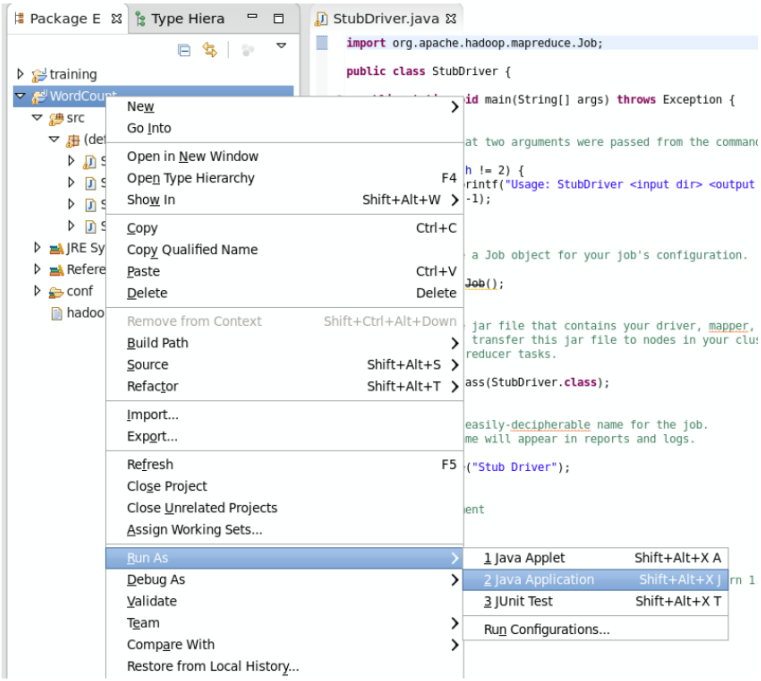

Figure 8: Run a Hadoop Project.

6. In the pop-up dialog select the main class from the selection list and click *OK*. See Figure 9.

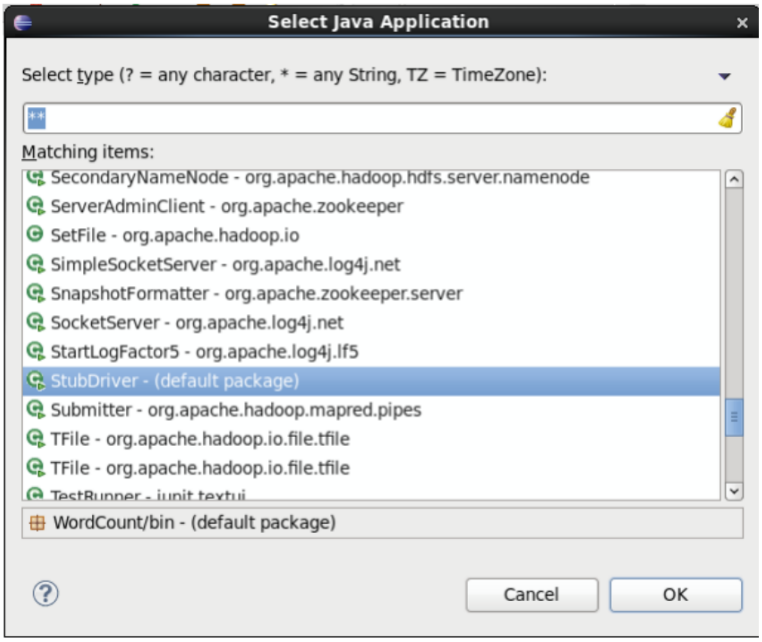

Figure 9: Run a Hadoop Project.

After you have setup the run configuration the first time, you can skip steps 1 and 2 above in subsequent runs, unless you need to change the arguments. You can also create more than one launch configuration if you'd like, such as one for each set of common arguments.

2.3 Running Hadoop in pseudo-distributed mode

Once you've created your project and written the source code, to run the project in pseudo-distributed mode, do the following:

1. Right-click on the project and select *Export*. See Figure 10.

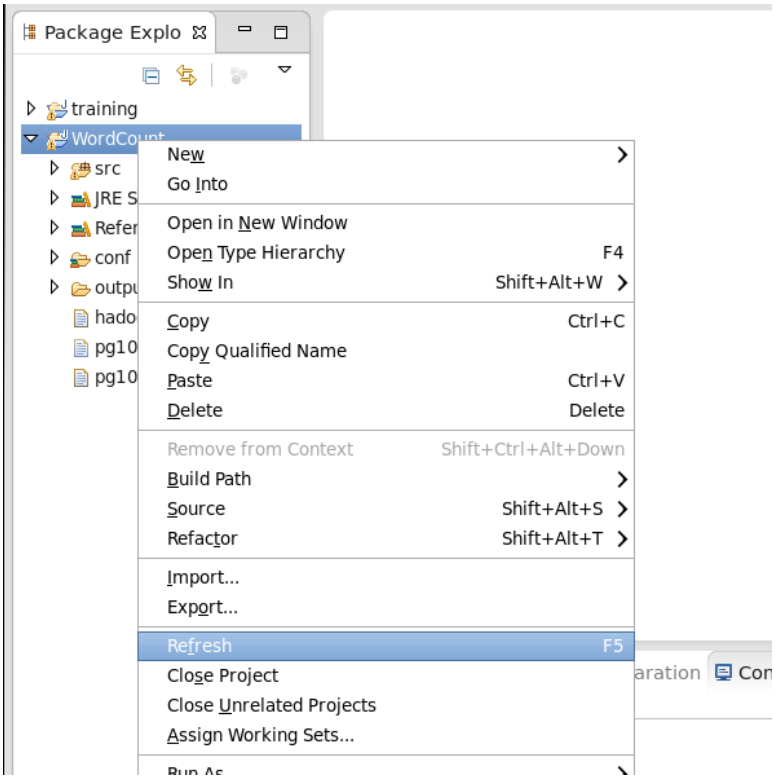

Figure 10: Run a Hadoop Project.

2. In the pop-up dialog, expand the *Java* node and select *JAR file*. See Figure 11. Click *Next* >

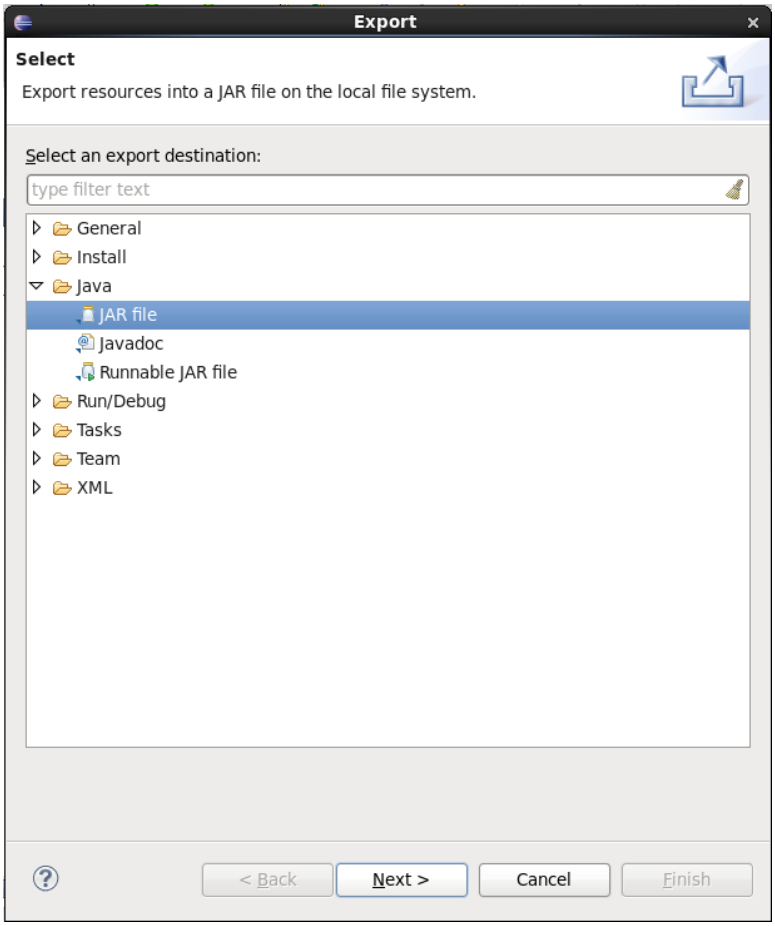

Figure 11: Run a Hadoop Project.

3. Enter a path in the *JAR file* field and click *Finish*. See Figure 12.

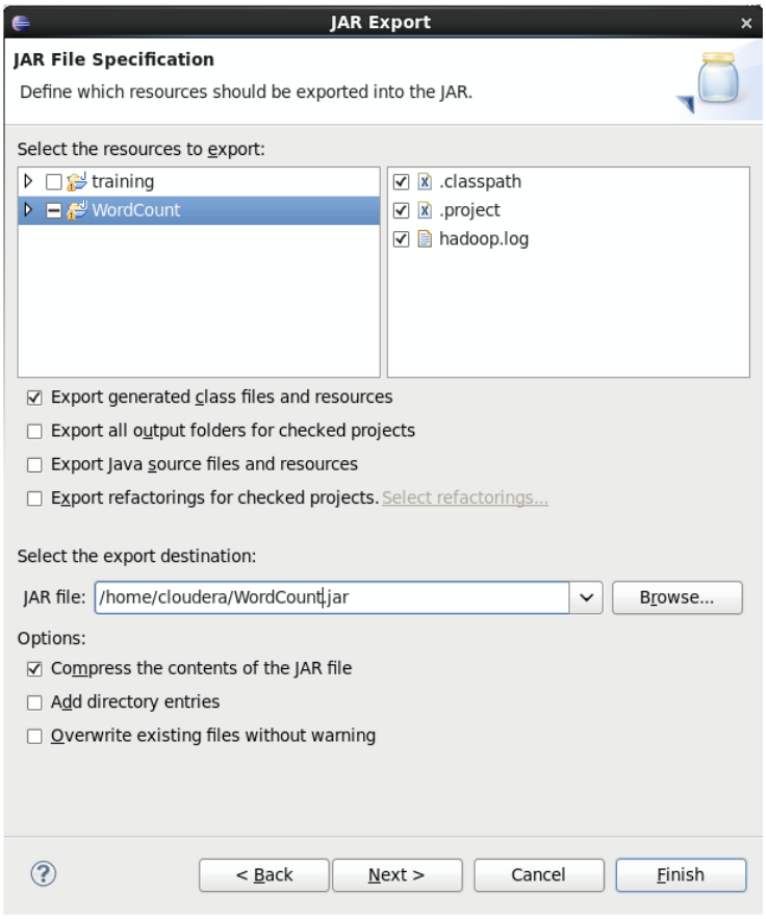

Figure 12: Run a Hadoop Project.

4. Open a terminal and run the following command:

   `hadoop jar path/to/file.jar input path output path`

   After modifications to the source files, repeat all of the above steps to run job again.

2.4 Debugging Hadoop jobs

To debug an issue with a job, the easiest approach is to run the job in stand-alone mode and use a debugger. To debug your job, do the following steps:

1. Right-click on the project and select *Debug As* → *Java Application*. See Figure 13.

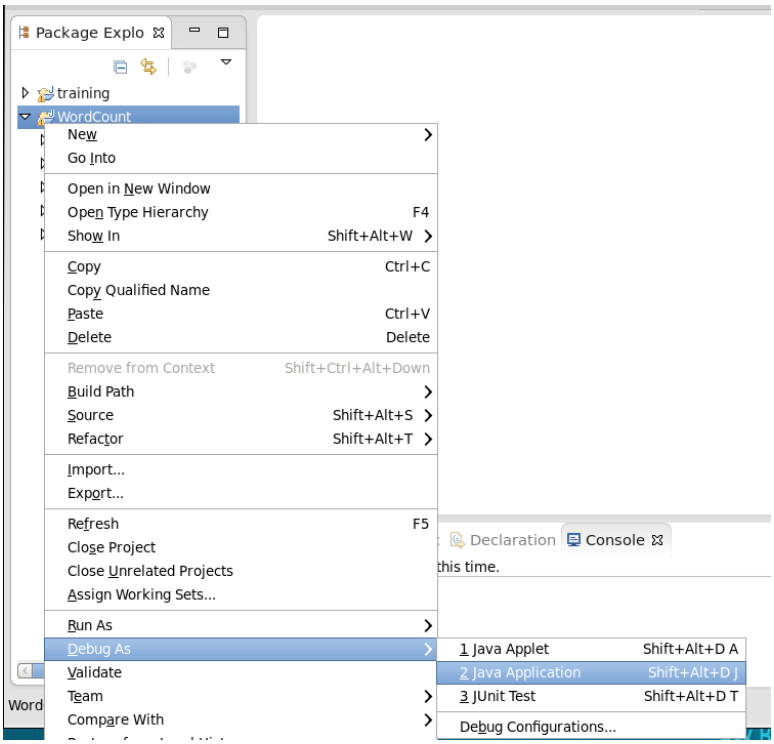

Figure 13: Debug a Hadoop project.

2. In the pop-up dialog select the main class from the selection list and click *OK*. See Figure 14.

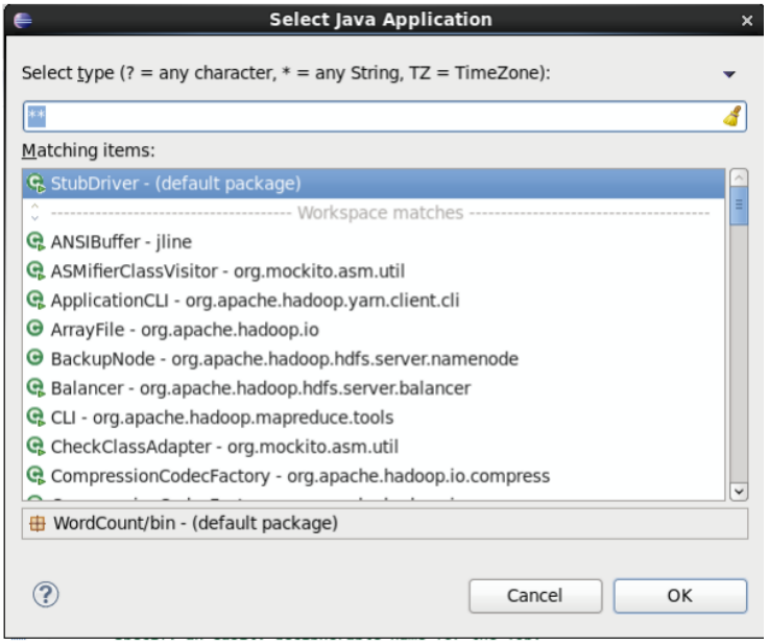

Figure 14: Run a Hadoop Project.

You can use the Eclipse debugging features to debug your job execution. See the additional Eclipse tutorials at the end of section 2.6 for help using the Eclipse debugger.

When running your job in pseudo-distributed mode, the output from the job is logged in the task tracker's log files, which can be accessed most easily by pointing a web browser to port 8088 of the server, which will the `localhost`. From the job tracker web page, you can drill down into the failing job, the failing task, the failed attempt, and finally the log files. Note that the logs for stdout and stderr are separated, which can be useful when trying to isolate specific debugging print statements.

2.5 Example project

In this section you will create a new Eclipse Hadoop project, compile, and execute it. The program will count the frequency of all the words in a given large text file. In your virtual machine, Hadoop, Java environment and Eclipse have already been pre-installed.

- Open Eclipse. If you just launched the VM, you may have to close the Firefox window to find the Eclipse icon on the desktop.
- Right-click on the *training* node in the Package Explorer and select *Copy*. See Figure 15.

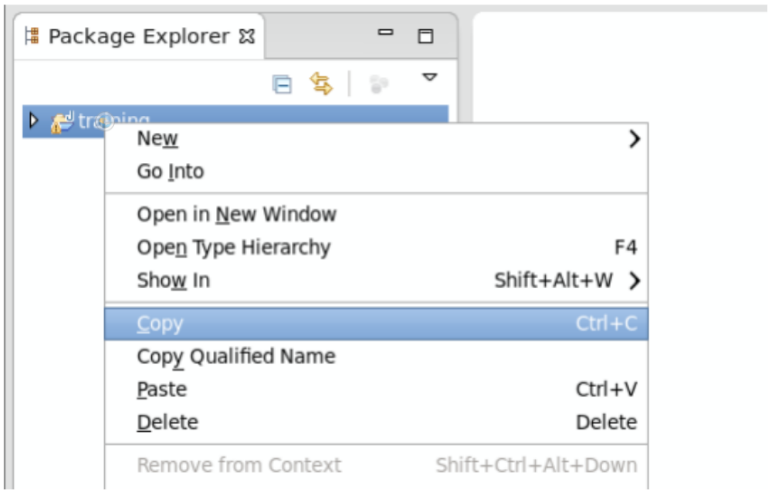

Figure 15: Create a Hadoop Project.

- Right-click on the *training* node in the Package Explorer and select *Paste*. See Figure 16.

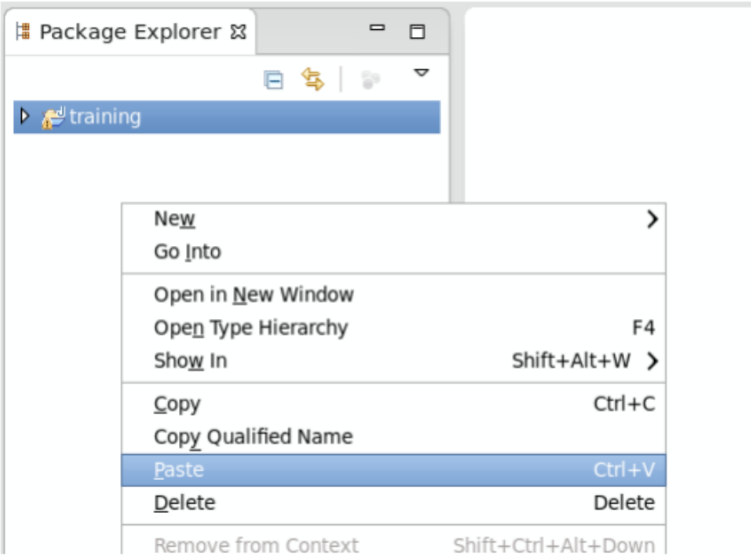

Figure 16: Create a Hadoop Project.

- In the pop-up dialog, enter the new project name in the *Project Name* field and click *OK*. See Figure 17.

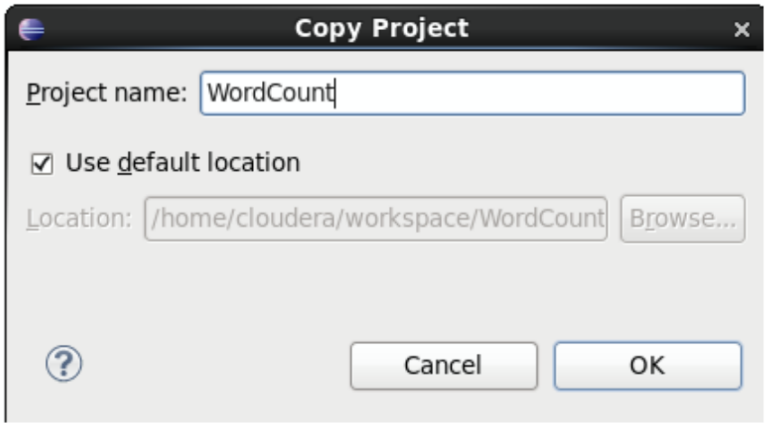

Figure 17: Create a Hadoop Project.

- Create a new package called `edu.stanford.cs246.wordcount` by right-clicking on the *src* node and selecting *New* → *Package*. See Figure 18.

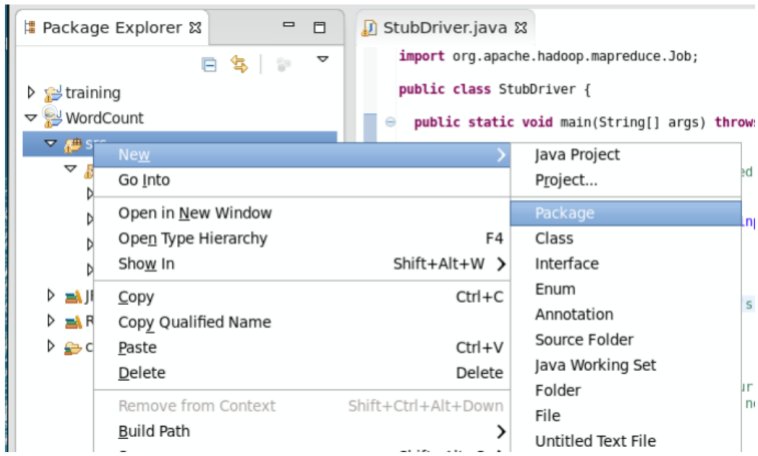

Figure 18: Create a Hadoop Project.

- Enter `edu.stanford.cs246.wordcount` in the *Name* field and click *Finish*. See Figure 19.

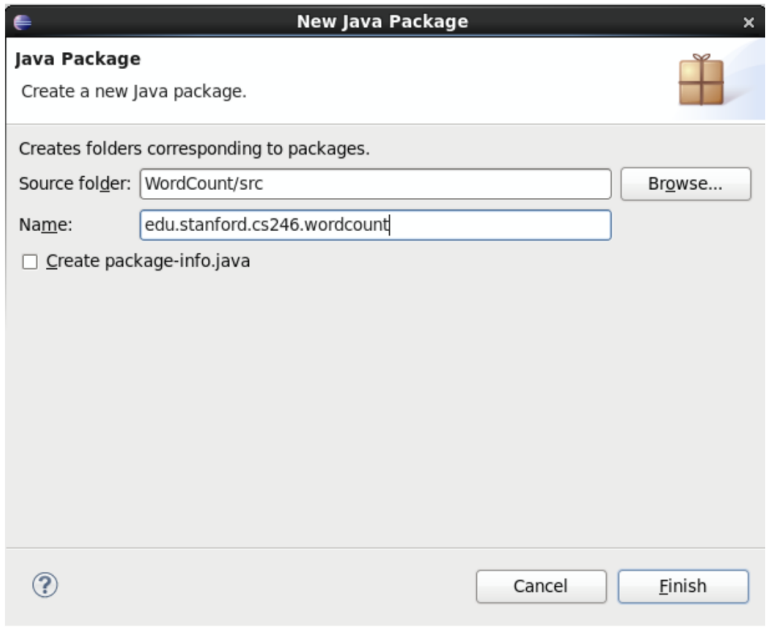

Figure 19: Create a Hadoop Project.

- Create a new class in that package called `WordCount` by right-clicking on the *edu.stanford.cs246.wordco* node and selecting $New \rightarrow Class$. See Figure 20.

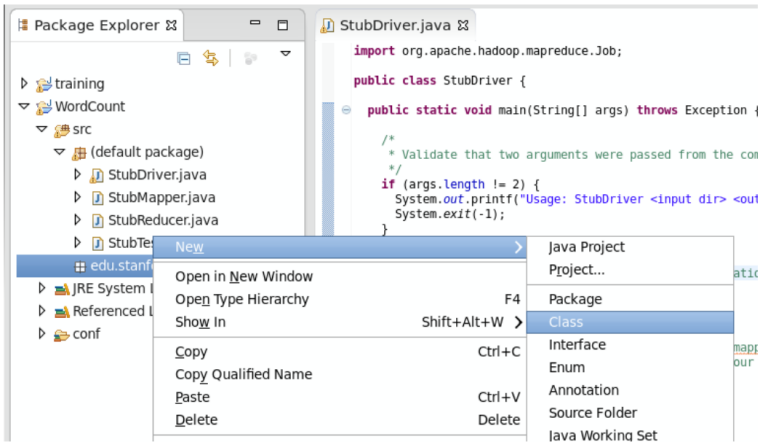

Figure 20: Create a Hadoop Project.

- In the pop-up dialog, enter `WordCount` as the *Name*. See Figure 21.

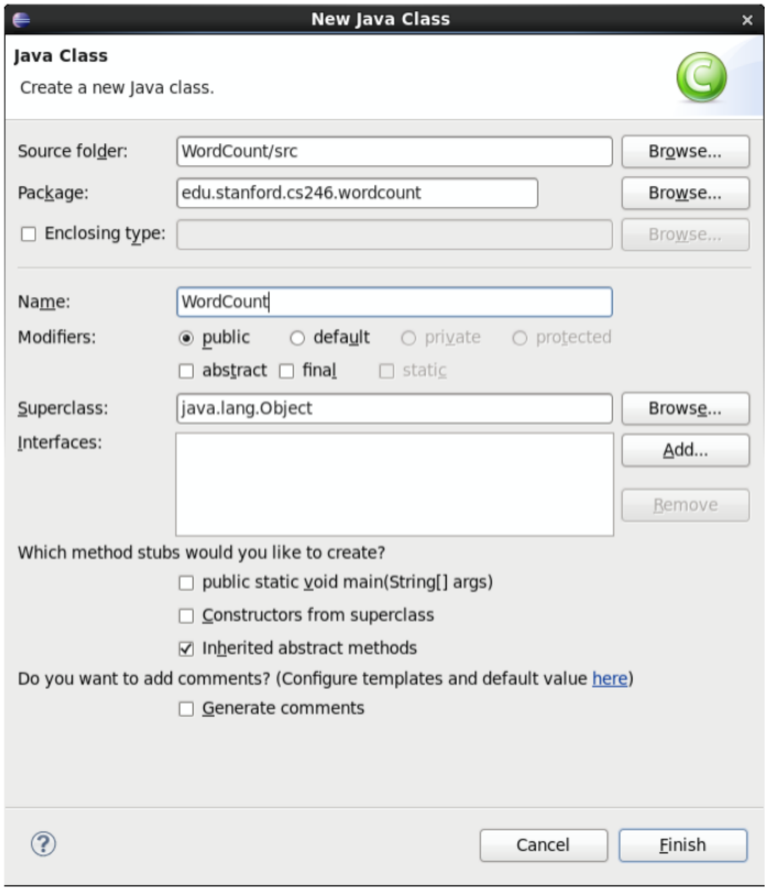

Figure 21: Create a Hadoop Project.

- In the Superclass field, enter `Configured` and click the Browse button. From the popup

window select **Configured** – *org.apache.hadoop.conf* and click the *OK* button. See Figure 22.

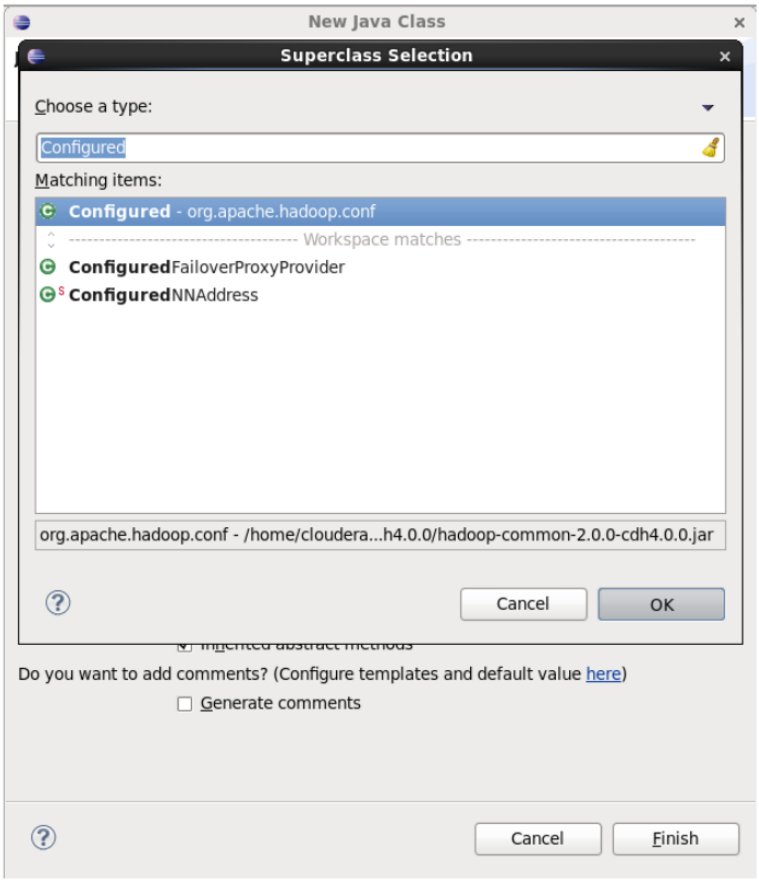

Figure 22: Create a java file.

- In the *Interfaces* section, click the *Add* button. From the pop-up window select **Tool** – *org.apache.hadoop.util* and click the *OK* button. See Figure 23.

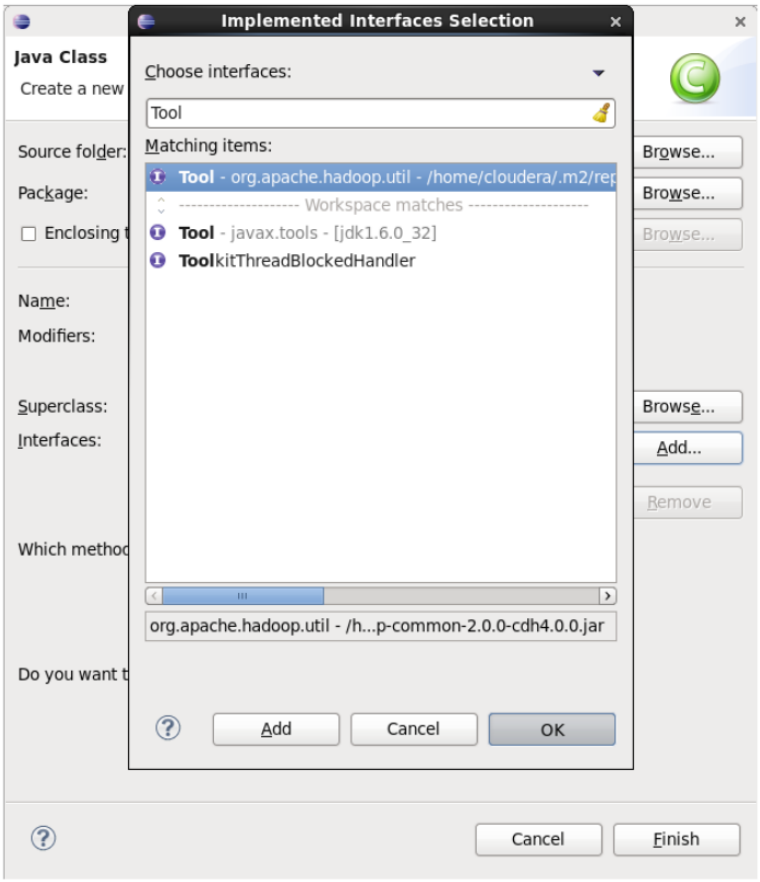

Figure 23: Create a java file.

- Check the boxes for *public static void main(String args[])* and *Inherited abstract methods* and click the *Finish* button. See Figure 24.

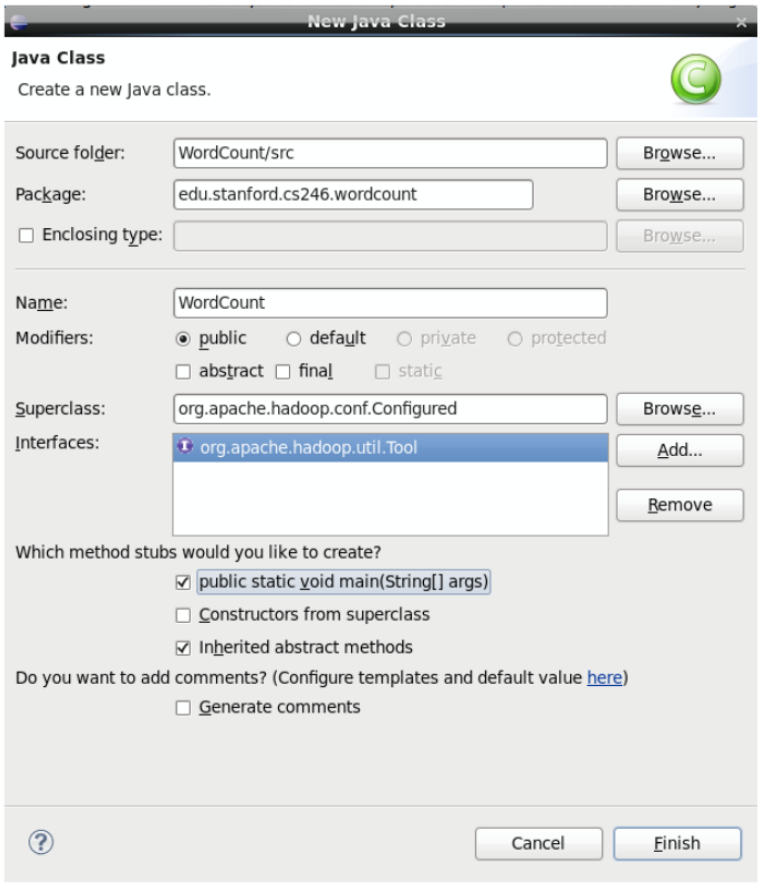

Figure 24: Create WordCount.java.

- You will now have a rough skeleton of a Java file as in Figure 25. You can now add code to this class to implement your Hadoop job.

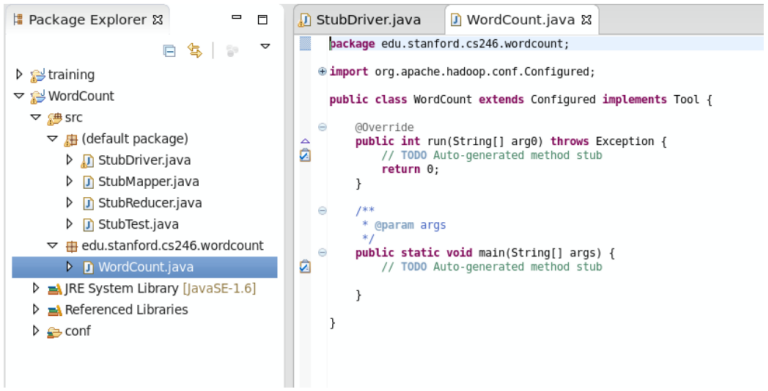

Figure 25: Create WordCount.java.

- Rather than implement a job from scratch, copy the contents from http://snap.stanford.edu/class/cs246-data-2014/WordCount.java and paste it into the `WordCount.java`

file. See Figure 26. The code in `WordCount.java` calculates the frequency of each word in a given dataset.

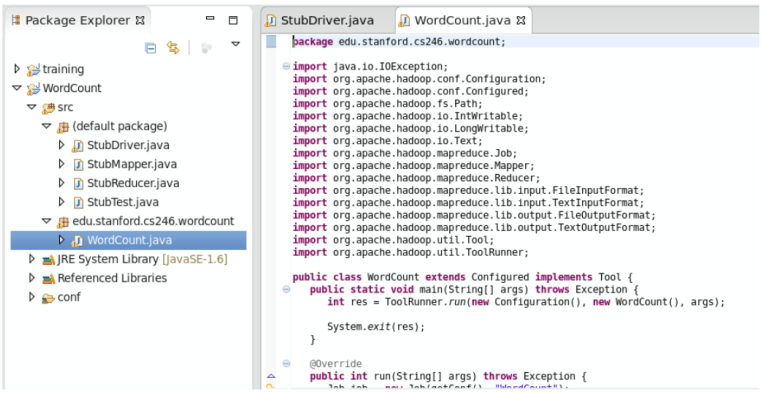

Figure 26: Create WordCount.java.

- Download the *Complete Works of William Shakespeare* from Project Gutenberg at http://www.gutenberg.org/cache/epub/100/pg100.txt. You can do this simply with `cURL`, but you also have to be aware of the byte order mark (BOM). You can download the file and remove the BOM in one line by opening a terminal, changing to the `~/workspace/WordCount` directory, and running the following command:

  ```
  curl http://www.gutenberg.org/cache/epub/100/pg100.txt | perl -pe 's/^\xEF\xBB\xBF//' > pg100.txt
  ```

  If you copy the above command beware the quotes as the copy/paste will likely mistranslate them.

- Right-click on the project and select *Run As* → *Run Configurations*. See Figure 27.

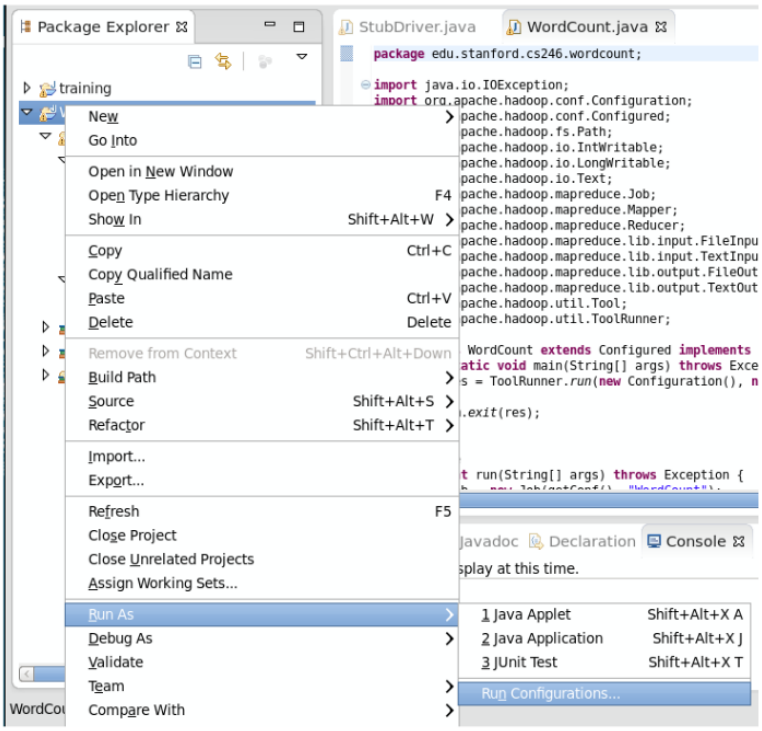

Figure 27: Run WordCount.java.

- In the pop-up dialog, select the *Java Application* node and click the New launch configuration button in the upper left corner. See Figure 28.

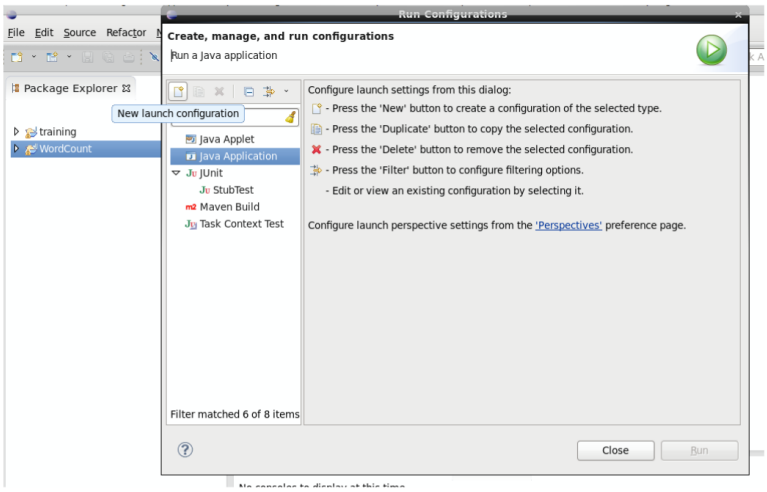

Figure 28: Run WordCount.java.

- Enter a name in the *Name* field and `WordCount` in the *Main class* field. See Figure 29.

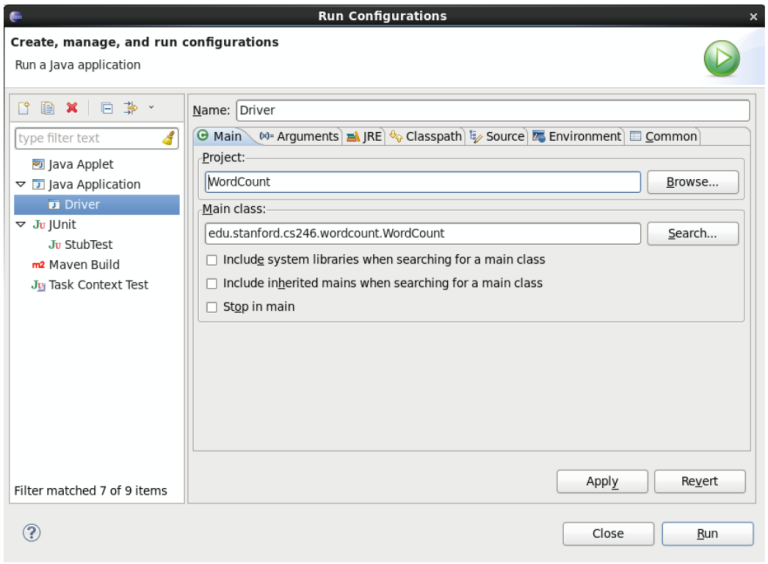

Figure 29: Run WordCount.java.

- Switch to the *Arguments* tab and put `pg100.txt output` in the *Program arguments* field. See Figure 30. Click *Apply* and *Close*.

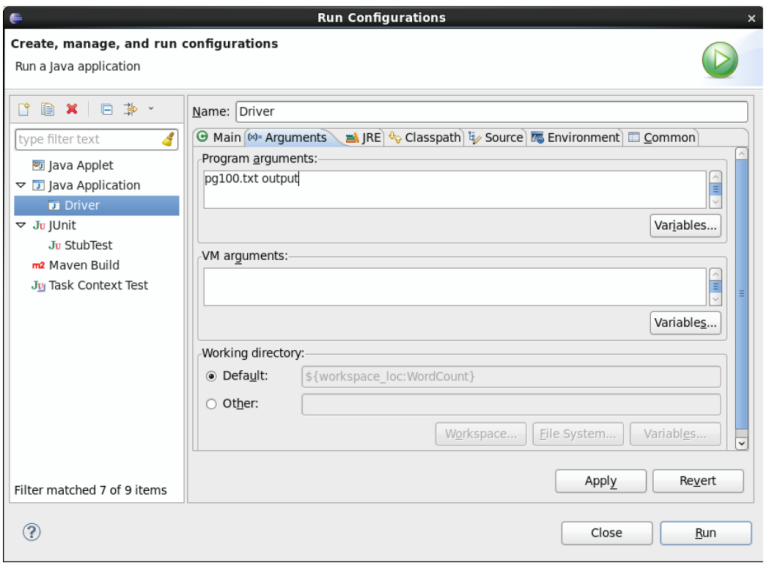

Figure 30: Run WordCount.java.

- Right-click on the project and select *Run As* → *Java Application*. See Figure 31.

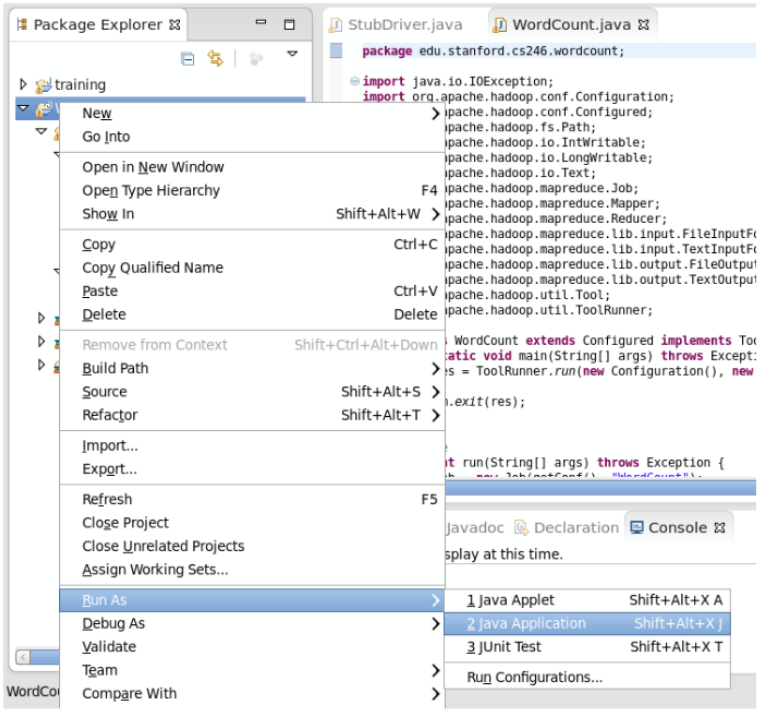

Figure 31: Run WordCount.java.

- In the pop-up dialog select *WordCount - edu.stanford.cs246.wordcount* from the selection list and click *OK*. See Figure 32.

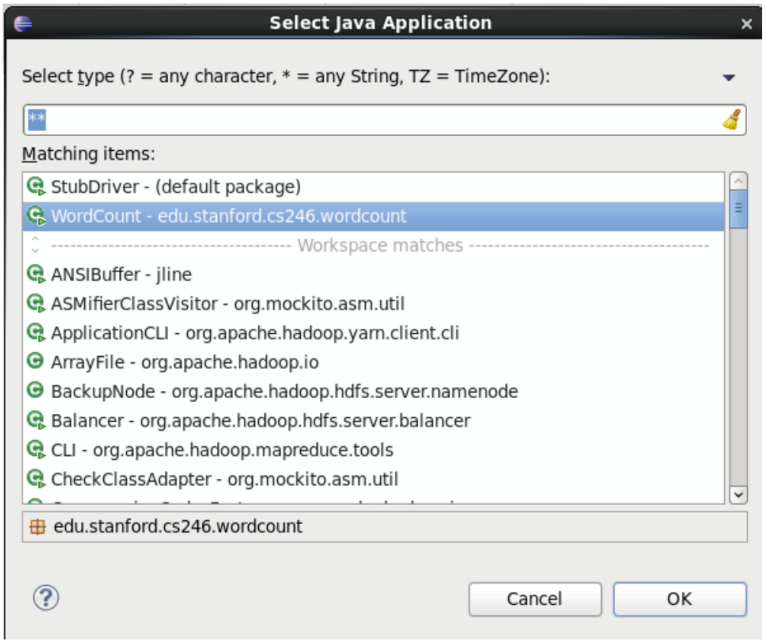

Figure 32: Export a hadoop project.

You will see the command output in the console window, and if the job succeeds, you'll find the results in the `~/workspace/WordCount/output` directory. If the job fails complaining that it cannot find the input file, make sure that the `pg100.txt` file is located in the `~/workspace/WordCount` directory.

- Right-click on the project and select *Export*. See Figure 33.

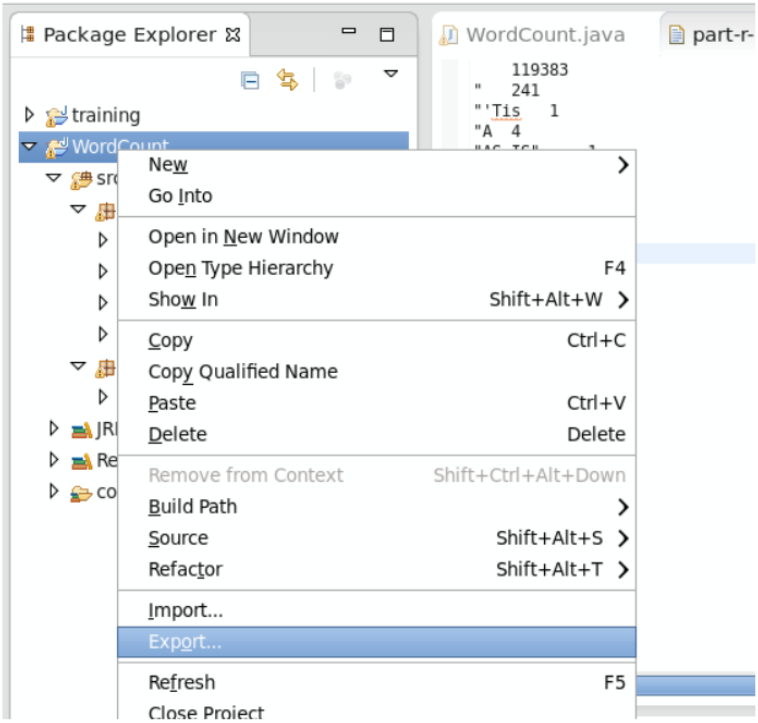

Figure 33: Run WordCount.java.

- In the pop-up dialog, expand the *Java* node and select *JAR file*. See Figure 34. Click *Next >*

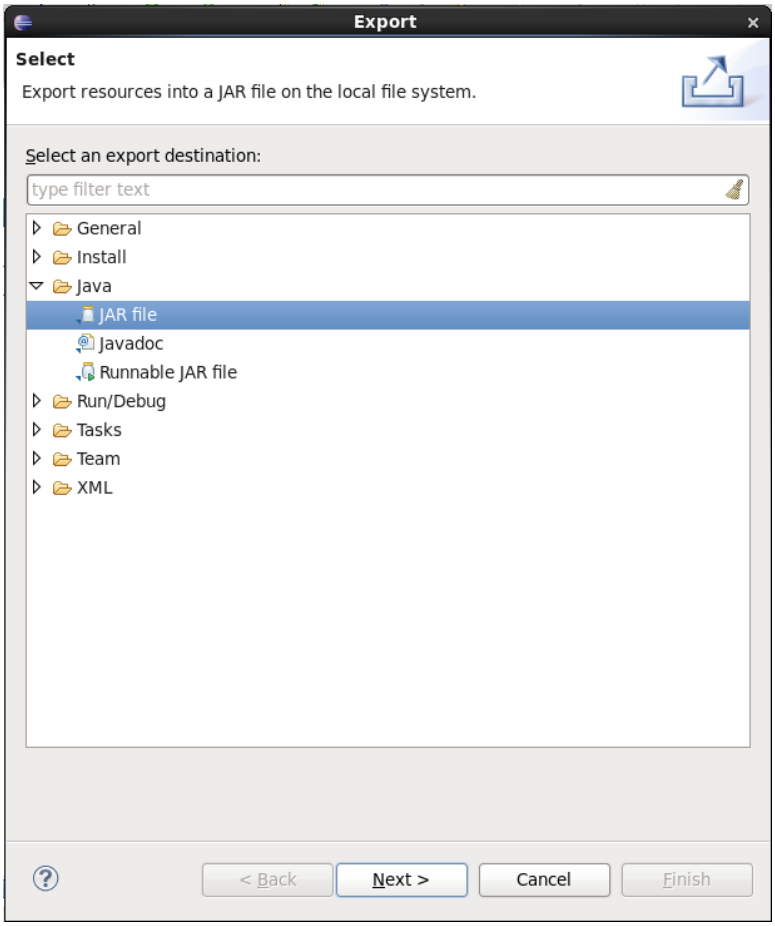

Figure 34: Export a hadoop project.

- Enter `/home/cloudera/wordcount.jar` in the JAR file field and click *Finish*. See Figure 35.

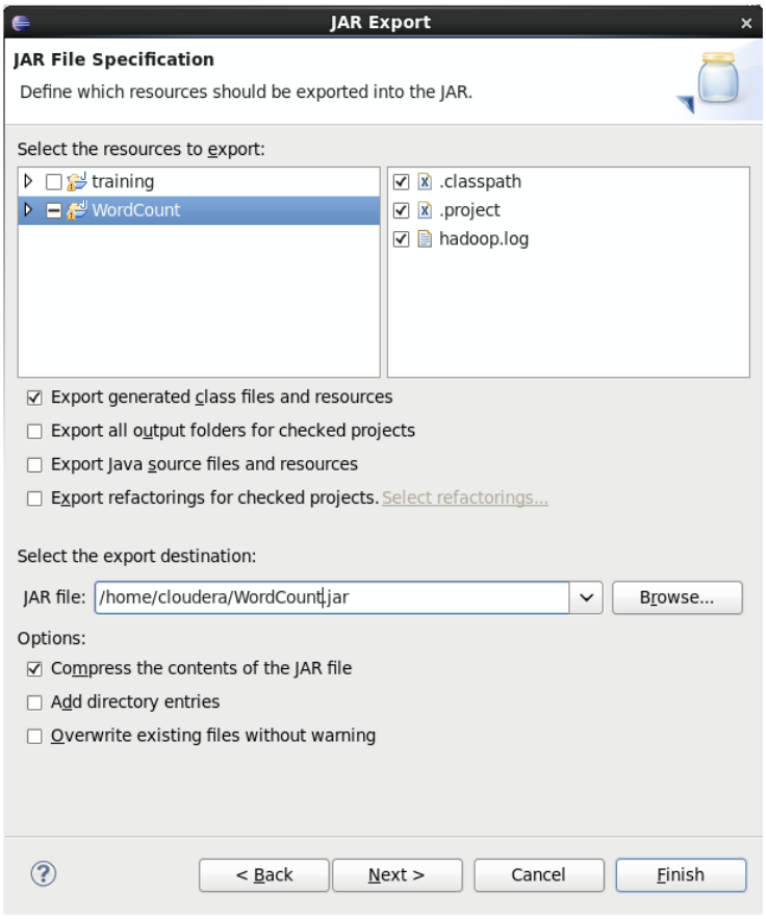

Figure 35: Export a hadoop project.

If you see an error dialog warning that the project compiled with warnings, you can simply click *OK*.

- Open a terminal in your VM and traverse to the folder /home/cloudera and run the following commands:

  `hadoop fs -put workspace/WordCount/pg100.txt`

  `hadoop jar WordCount.jar edu.stanford.cs246.wordcount.WordCount pg100.txt output`

- Run the command: `hadoop fs -ls output`

  You should see an output file for each reducer. Since there was only one reducer for this job, you should only see one `part-*` file. Note that sometimes the files will be called `part-NNNNN`, and sometimes they'll be called `part-r-NNNNN`. See Figure 36.

```
[cloudera@localhost Desktop]$ hadoop fs -ls output
Found 3 items
-rw-r--r--   3 cloudera cloudera          0 2014-01-01 16:22 output/_SUCCESS
drwxr-xr-x   - cloudera cloudera          0 2014-01-01 16:22 output/_logs
-rw-r--r--   3 cloudera cloudera     720989 2014-01-01 16:22 output/part-r-00000
```

Figure 36: Run WordCount job.

- Run the command:

  `hadoop fs -cat output/part\* | head`

  You should see the same output as when you ran the job locally, as shown in Figure 37

```
[cloudera@localhost Desktop]$ hadoop fs -cat output/part\* | head
        119383
"       241
"'Tis   1
"A      4
"AS-IS".        1
"Air,"  1
"Alas,  1
"Amen"  2
"Amen"? 1
"Amen," 1
cat: Unable to write to output stream.
```

Figure 37: Run WordCount job.

- To view the job's logs, open the browser in the VM and point it to `http://localhost:8088` as in Figure 38

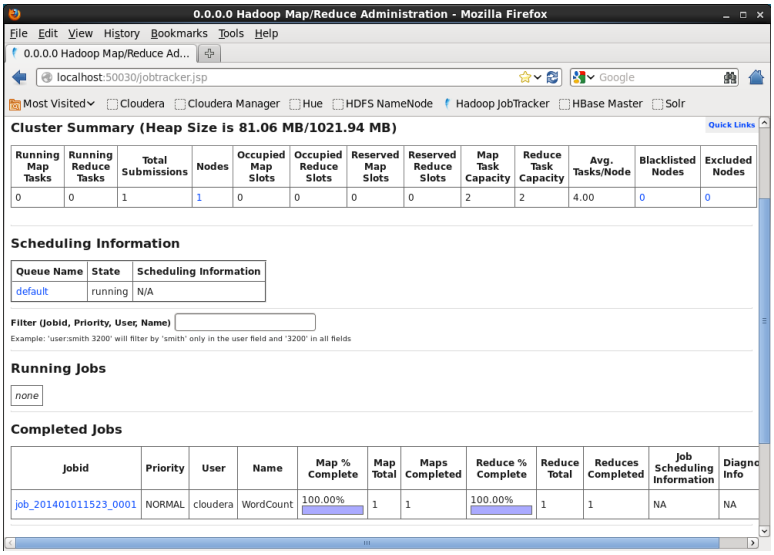

Figure 38: Run WordCount job.

- Click on the link for the completed job. See Figure 39.

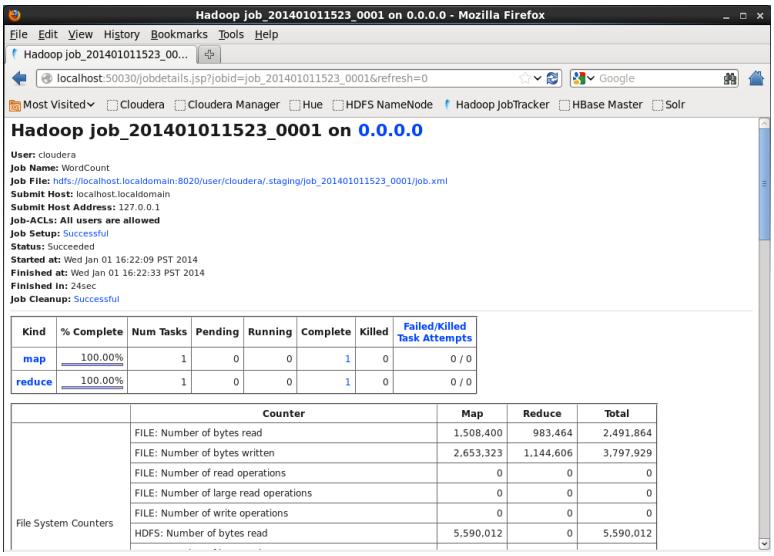

Figure 39: View WordCount job logs.

- Click the link for the map tasks. See Figure 40.

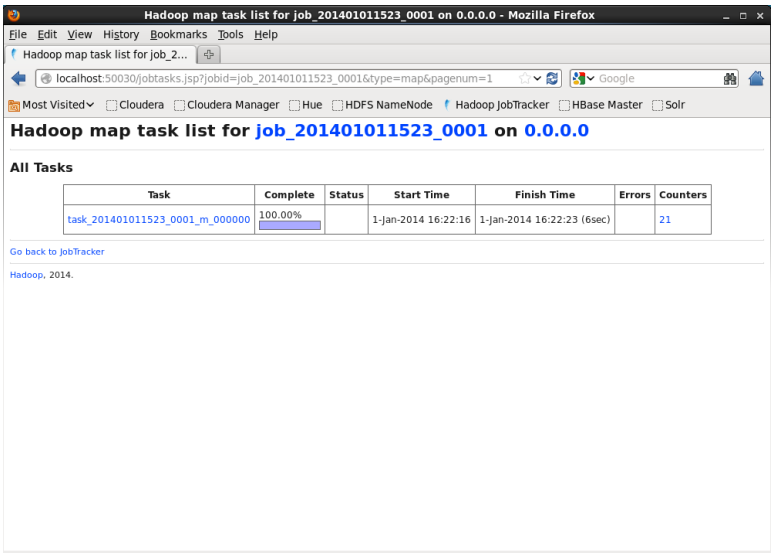

Figure 40: View WordCount job logs.

- Click the link for the first attempt. See Figure 41.

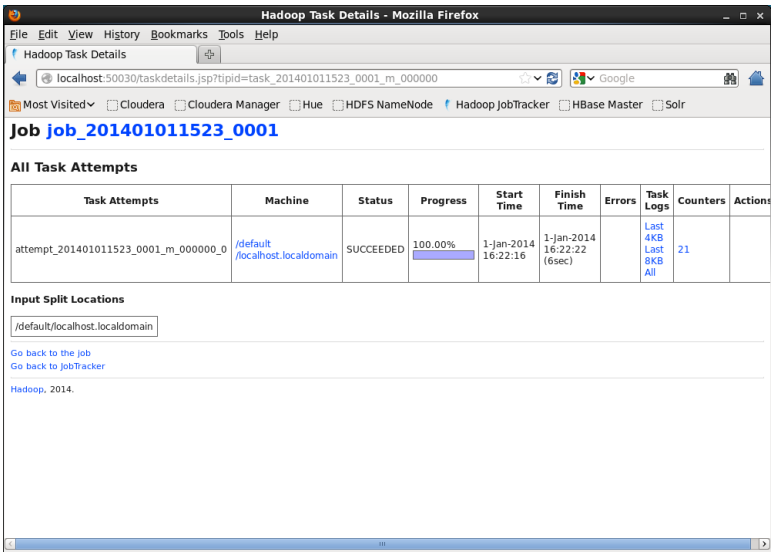

Figure 41: View WordCount job logs.

- Click the link for the full logs. See Figure 42.

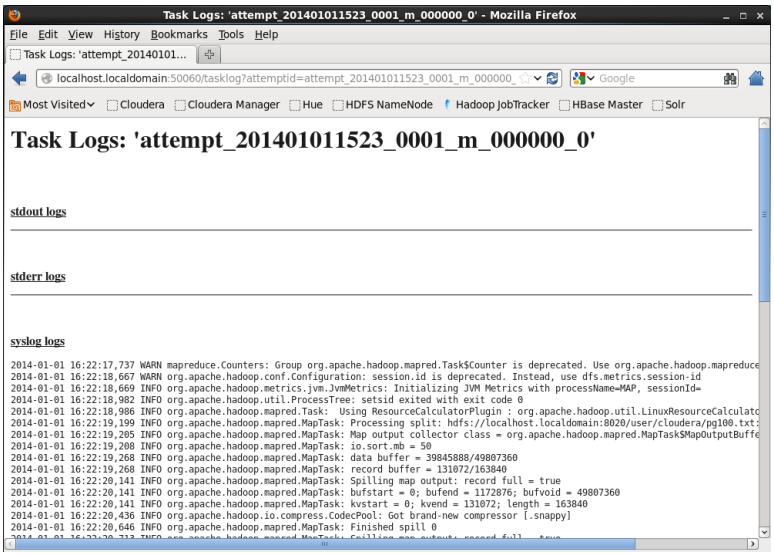

Figure 42: View WordCount job logs.

2.6 Using your local machine for development

If you'd rather use your own development environment instead of working in the IDE, follow these steps:

1. Make sure that you have an entry for `localhost.localdomain` in your `/etc/hosts` file, e.g.

```
127.0.0.1 localhost localhost.localdomain
```

2. Install a copy of Hadoop locally. The easiest way to do that is to simply download the archive from http://archive.cloudera.com/cdh5/cdh/5/hadoop-latest.tar.gz and unpack it.

3. In the unpacked archive, you'll find a `etc/hadoop` directory. In that directory, open the `core-site.xml` file and modify it as follows:

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>

<!-- Put site-specific property overrides in this file. -->

<configuration>
    <property>
        <name>fs.default.name</name>
        <value>hdfs://192.168.56.101:8020</value>
    </property>
</configuration>
```

4. Next, open the `yarn-site.xml` file in the same directory and modify it as follows:

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>

<!-- Put site-specific property overrides in this file. -->

<configuration>
    <property>
        <name>yarn.resourcemanager.hostname</name>
        <value>192.168.56.101</value>
    </property>
</configuration>
```

   You can now run the Hadoop binaries located in the `bin` directory in the archive, and they will connect to the cluster running in your virtual machine.

Further Hadoop tutorials

- Yahoo! Hadoop Tutorial: http://developer.yahoo.com/hadoop/tutorial/
- Cloudera Hadoop Tutorial: http://www.cloudera.com/content/www/en-us/training/library/tutorials.html
- How to Debug MapReduce Programs: http://wiki.apache.org/hadoop/HowToDebugMapReducePrograms

Further Eclipse tutorials

- Genera Eclipse tutorial:
  http://www.vogella.com/articles/Eclipse/article.html.
- Tutorial on how to use the Eclipse debugger:
  http://www.vogella.com/articles/EclipseDebugging/article.html.

3 Task: Write your own Hadoop Job

Now you will write your first MapReduce job to accomplish the following task:

- Write a Hadoop MapReduce program which outputs the number of words that start with each letter. This means that for every letter we want to count the total number of words that start with that letter. In your implementation ignore the letter case, *i.e.*, consider all words as lower case. You can ignore all non-alphabetic characters.
- Run your program over the same input data as above.

What to hand-in: Hand-in the printout of the output file and upload the source code.