

Lab 9 – Apache Spark

Submit your *own work* on time. No credit will be given if the lab is submitted after the due date. Follow the instructions completely.

Step 1: Verify if Java is installed

Java is a pre-requisite software for running Spark Applications.

Use the following command to verify if Java is installed on your system :-

```
[cloudera@quickstart ~]$ java -version
java version "1.8.0_211"
Java(TM) SE Runtime Environment (build 1.8.0_211-b12)
Java HotSpot(TM) 64-Bit Server VM (build 25.211-b12, mixed mode)
```

Support for Java 7 is deprecated as of Spark 2.0.0

For doing this lab, you need to upgrade your CDH to JDK8 if it's still JDK7. If you are not sure how to do this then ask me for instructions.

1. [Spark Practice Lab](#)

No need to submit this part.

- I. [Spark Word Count in Local mode](#)
- II. [Spark Word Count in Pseudo-distributed mode](#)

2. [Spark Homework Lab](#)

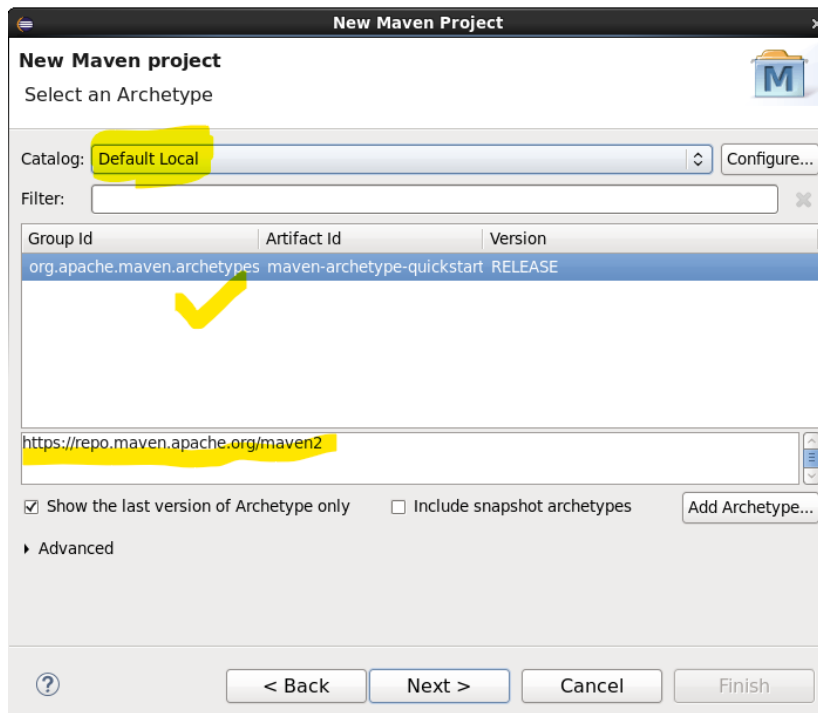
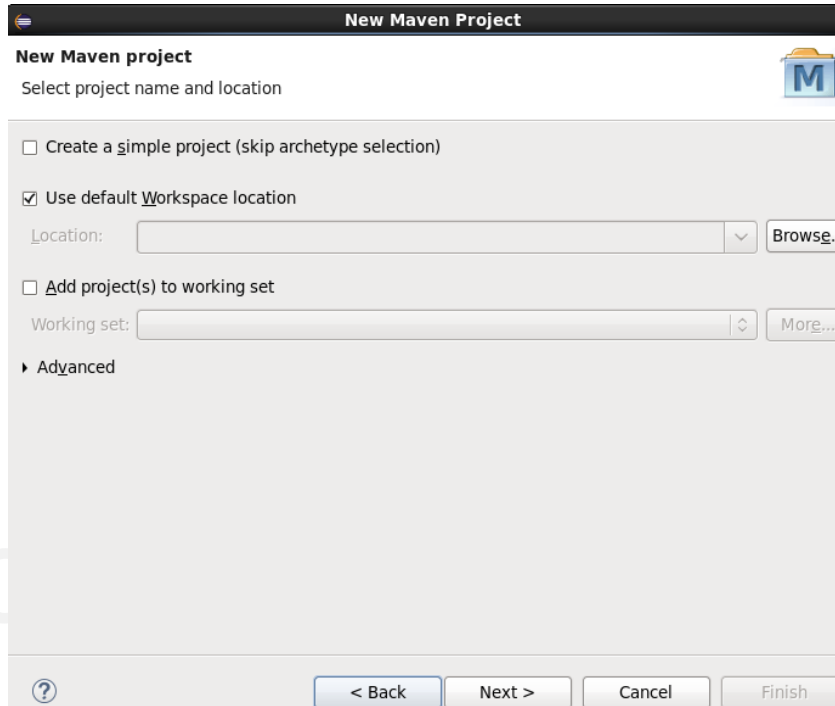
Submit ".java" files and output along with the command that you used to run the program in pseudo-distributed mode.

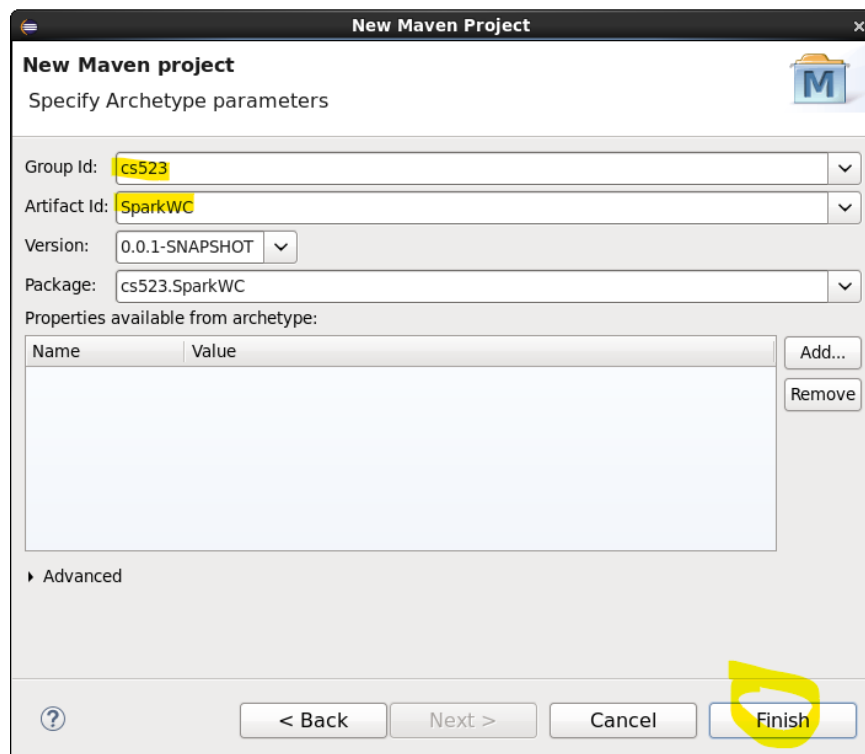
Paste screenshots wherever applicable.

Spark Practice Labs

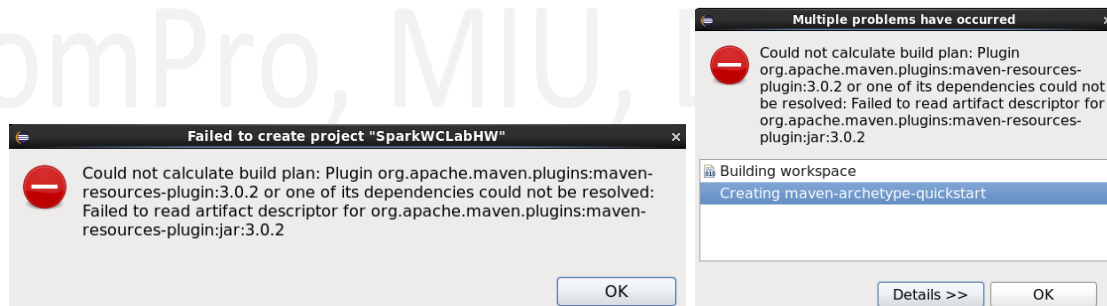
Part 1 - Spark Word Count in local mode

1. Need to create a Maven project in Eclipse for spark word count.

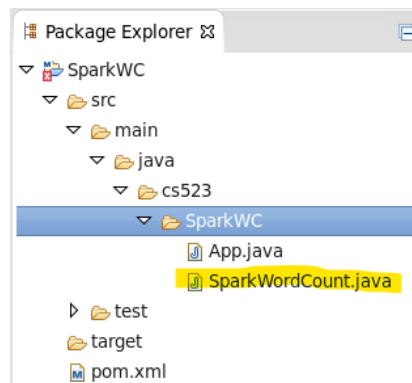




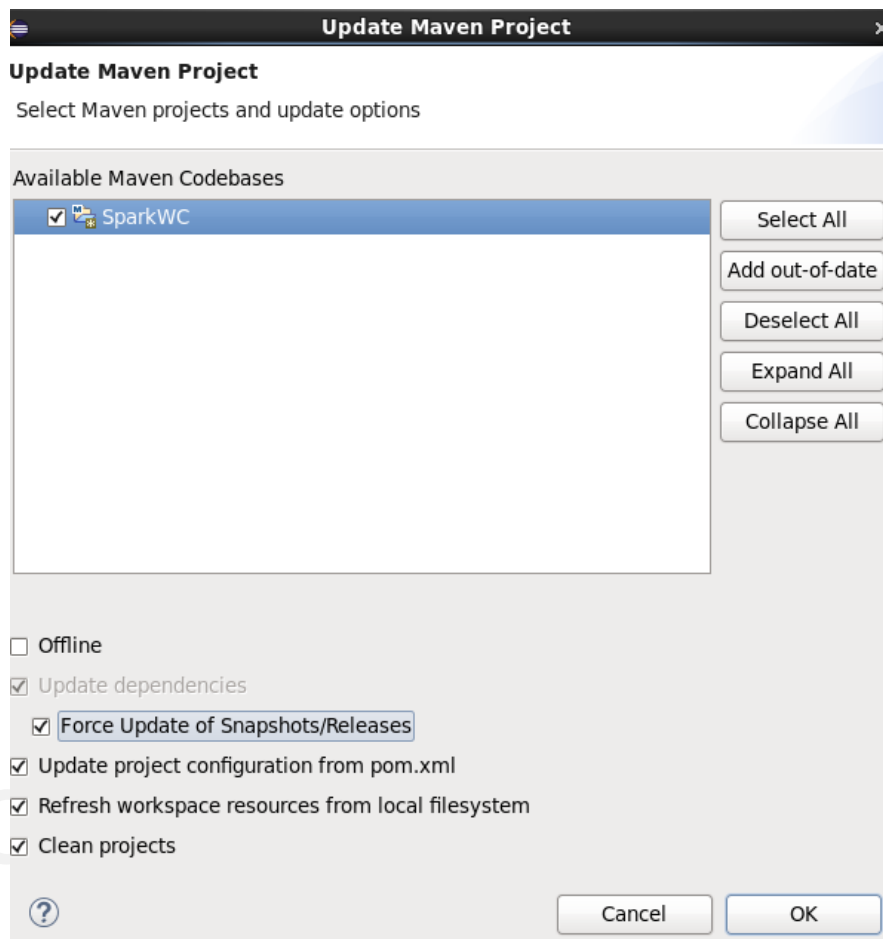
2. After clicking on "Finish", you might get an error like the following, just ignore it for now.



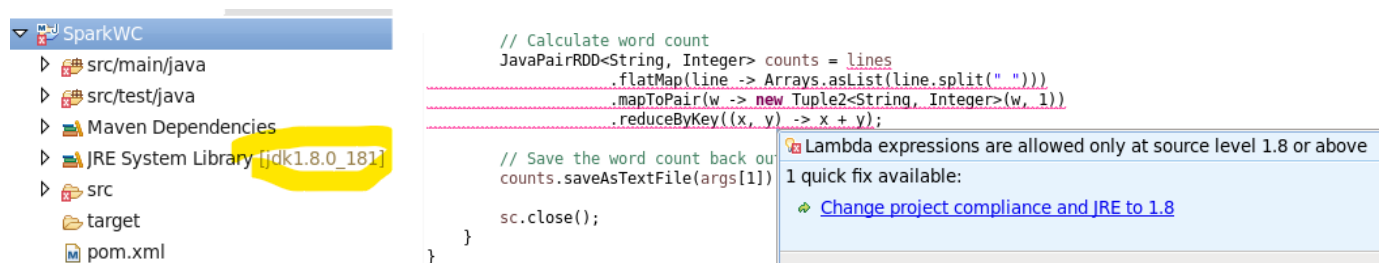
3. Your pom.xml file should look like the one which is given to you. It'll add all the required dependencies to your project.
4. Spark Word Count Java program (for JDK 8) is given to you, download it and add it to your project in the folder where App.java file is located. (Then you can safely delete App.java and test packages that got created by default)



5. Right click on the project and go to Maven and do "Update Project".



6. Delete the "test" package if it gets created after the above step.
7. Also configure the project build path to include JRE 1.8 and not 1.7 and then change the project compliance to JRE 1.8



8. Create folder "input" in your project path and add a test input file there on which you'll run the word count.
9. Now that all the errors are gone from the project, run the given word count java program as regular Java application by giving "input" and "output" as runtime arguments.
10. You can check the part file in the output folder after refreshing the project.

Part 2 - Spark Word Count in pseudo-distributed mode

1. Make sure your input folder is present under `"/user/cloudera/"` directory and it has the input file in it.
2. Create jar of your project using `mvn package` command or from eclipse, right click on `pom` file then *Run As* and then do `maven build..` and keep the *goals* as `package`. The created jar file will be inside the `target` directory. I'm saving the jar file to Desktop but you can choose any other path.
3. Now you need to submit this jar to the Spark cluster which is running on YARN using the following command.
`spark-submit --class "cs523.SparkWC.SparkWordCount" --master yarn Desktop/SparkWC.jar /user/cloudera/input/ /user/cloudera/output/`
4. After successful execution, it'll create an output folder under `"/user/cloudera/"`. Check it and verify the part file.

ComPro, MIU, DO NOT COPY

Spark Homework Lab

Complete this HW using Java 8. Provide the java programs and screenshots (wherever applicable).

1. [7] This question is an enhanced version of WordCount. In this version of WordCount, the goal is to learn the distribution of letters in the most popular words in a corpus; meaning now we need to calculate **character (letter) count of the most popular words**.

Your application must do the following:

- a) Get a word frequency *threshold* from user. You can supply this number as runtime arguments to your program. For example, my *threshold* is 30.
- b) Read an input set of text documents.
- c) Count the number of times each word appears in those documents.
- d) Filter out all words that appear fewer times than the *threshold*. So now the remaining words will be your most popular words.
For example : Java 50
 MapReduce 65
- e) For these remaining words, count the number of times each letter occurs. While calculating this letter count, take into account the number of times those words have occurred.

For example:

a	165
c	65
d	65
e	130
j	50
and so on.....	

-
2. [3] In this question, you'll start the Spark shell (spark-shell) and use it to write a Spark program in Scala.

In spark shell, you can use *CTRL-L* to clear the screen and *exit* for coming out of the shell.

Problem Statement: Spark uses some third-party libraries. **You need to find out how many of these are licensed under the BSD license** (acronym for Berkeley Software Distribution).

Luckily, Spark comes with a file named LICENSE, located in the Spark root directory ([/usr/lib/spark/LICENSE](#)). The LICENSE file contains the list of all libraries used by Spark and the licenses they're provided under. Lines in the file, which names packages licensed under the BSD license, contain the word BSD. You could easily use a Linux shell command to count those lines, but that's not the point.

Write the Spark shell commands to count the lines using the Scala Spark API.