**Student Guide for CS4225/CS5425 Coding Assignment**

[1 Overview 1](#_Toc97051592)

[2 Local Environment Setup (Optional) 2](#_Toc97051593)

[2.1 Windows 10 2](#_Toc97051594)

[2.2 Linux 12](#_Toc97051595)

[2.3 MacOS 14](#_Toc97051596)

[3 Login to SoC Clusters 16](#_Toc97051597)

[3.1 Create SoC account 16](#_Toc97051598)

[3.2 Report SoC account 16](#_Toc97051599)

[3.3 Login to SoC Cluster 16](#_Toc97051600)

[4 Configure Hadoop and Spark 20](#_Toc97051601)

[4.1 Modify Environmental Variables 20](#_Toc97051602)

[4.2 Check Environmental Variables 21](#_Toc97051603)

[5 Test Configuration 22](#_Toc97051604)

[5.1 Test Spark 22](#_Toc97051605)

[5.2 Test Hadoop and Submission 23](#_Toc97051606)

[6 Java OpenStreetMap Editor (Optional) 25](#_Toc97051607)

[6.1 Download 25](#_Toc97051608)

[6.2 JOSM Basic Operations 25](#_Toc97051609)

[7 Q&A and Contact 30](#_Toc97051610)

1. Overview

Getting your hands dirty is always an effective way of learning big data systems. It can be a tough and challenging process, but it will also be a fruitful experience. Let’s start from here.

In this student guide, we present the setup that you need to do before Assignments 1 and 2. To minimize the difficulty in installing and configuring Hadoop / Spark under different local environments, we recommend skipping Step 1 and only using the SoC cluster to test your code, as it has Hadoop and Spark already installed for you.

* Step 1 (Optional): You may try to set up a local environment for Hadoop and Spark. This step is not necessary for assignments, in which we will use the SoC cluster as a unified environment.
* Step 2: Test and build your programs in SoC cluster. **We will grade your submission in this environment.** The environment here is similar to cloud environments on public providers. To write your code, you can either edit directly on the clusters using a text editor like vim (<https://en.wikipedia.org/wiki/Vim_(text_editor)>), or write your code locally using an IDE like IntelliJ IDEA (or other IDE / text editor of your choice), then transfer the files to the cluster for testing using scp (see section 5.2 of this guide or <https://linuxize.com/post/how-to-use-scp-command-to-securely-transfer-files/>), or a file transfer software like WinSCP / Cyberduck. This involves the following sub-steps.
  + Step 2.1: Login to SoC Clusters. If you do not have an SoC account, you need to create one. See more details in Section 3 of this guide.
  + Step 2.2: Configure the environments for Hadoop and Spark.
  + Step 2.3: You should be able to run simple programs on Hadoop and Spark. Cheers!

1. Local Environment Setup (Optional)

In case you prefer to test your code locally or for the benefit of future projects, we provide guidelines to help you setup debug environments locally. We recommend IntelliJ IDEA 2020.1 as IDE, on which this section is based. For this assignment, this section is optional and only for reference, as you can also choose to debug and test solely on the cluster. You can also choose other IDEs based on your preference.

To do this, follow the guides in subsection 2.1-2.3 based on your operating system.

* 1. Windows 10

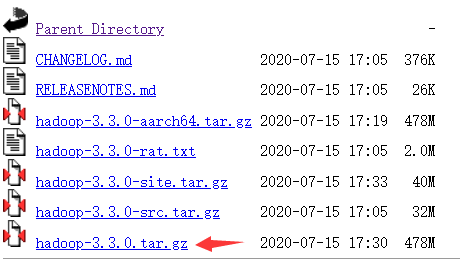
1. **Install Java 11**

Please follow this tutorial to install Java 11 on Windows

<https://java.tutorials24x7.com/blog/how-to-install-java-11-on-windows>. You should also ensure all the environmental variables of Java are set properly (as the tutorial).

1. **Install Hadoop**

Download Hadoop 3.3.0 from  
<https://archive.apache.org/dist/hadoop/common/hadoop-3.3.0/> Unzip to a directory, e.g. C:\\Program Files\hadoop-3.3.0. You do not need to run the installer.



However, this package does not contain some windows native required components. Download these components from <https://github.com/kontext-tech/winutils>,  
then unzip and copy the whole directory hadoop-3.3.0/bin to your installation path of Hadoop , e.g. C:\\Program Files\hadoop-3.3.0. When conflict happens, choose to replace all conflict files Also, copy hadoop-3.3.0/bin/hadoop.dll to C:\Windows\System32.

1. **Configure environment variables for Hadoop**

Open file explorer (by Press Ctrl+E). Right click "This PC", choose "properties". In the popup window, click "Advanced System Settings", then click "Environment Variables". In "system variables", create 2 new system variables by "New…".

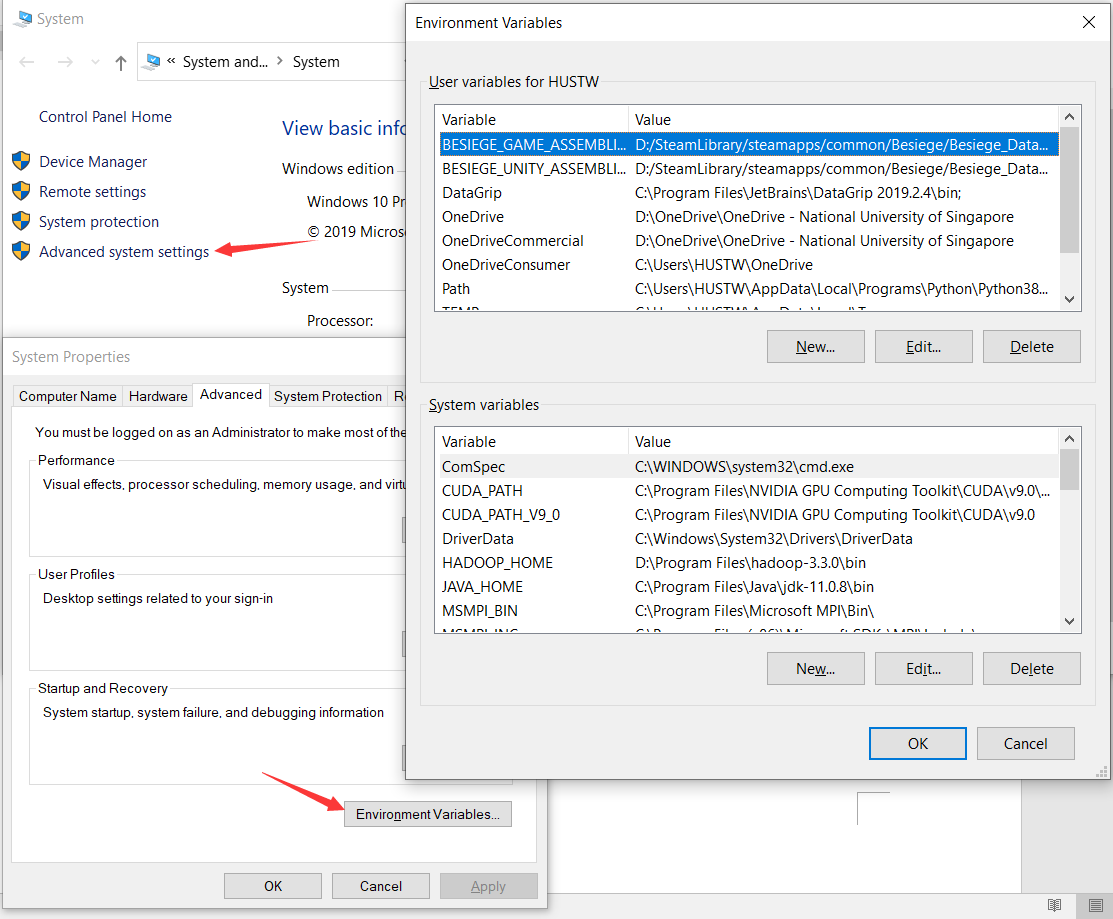
HADOOP\_HOME D:\Program Files\hadoop-3.3.0 <Your Path>

HADOOP\_BIN\_PATH %HADOOP\_HOME%\bin



In User variables, edit "PATH" variable by adding %HADOOP\_HOME%\bin and %HADOOP\_HOME%\sbin.

Click "OK", "OK", "OK" to save the changes. It should take effect immediately.

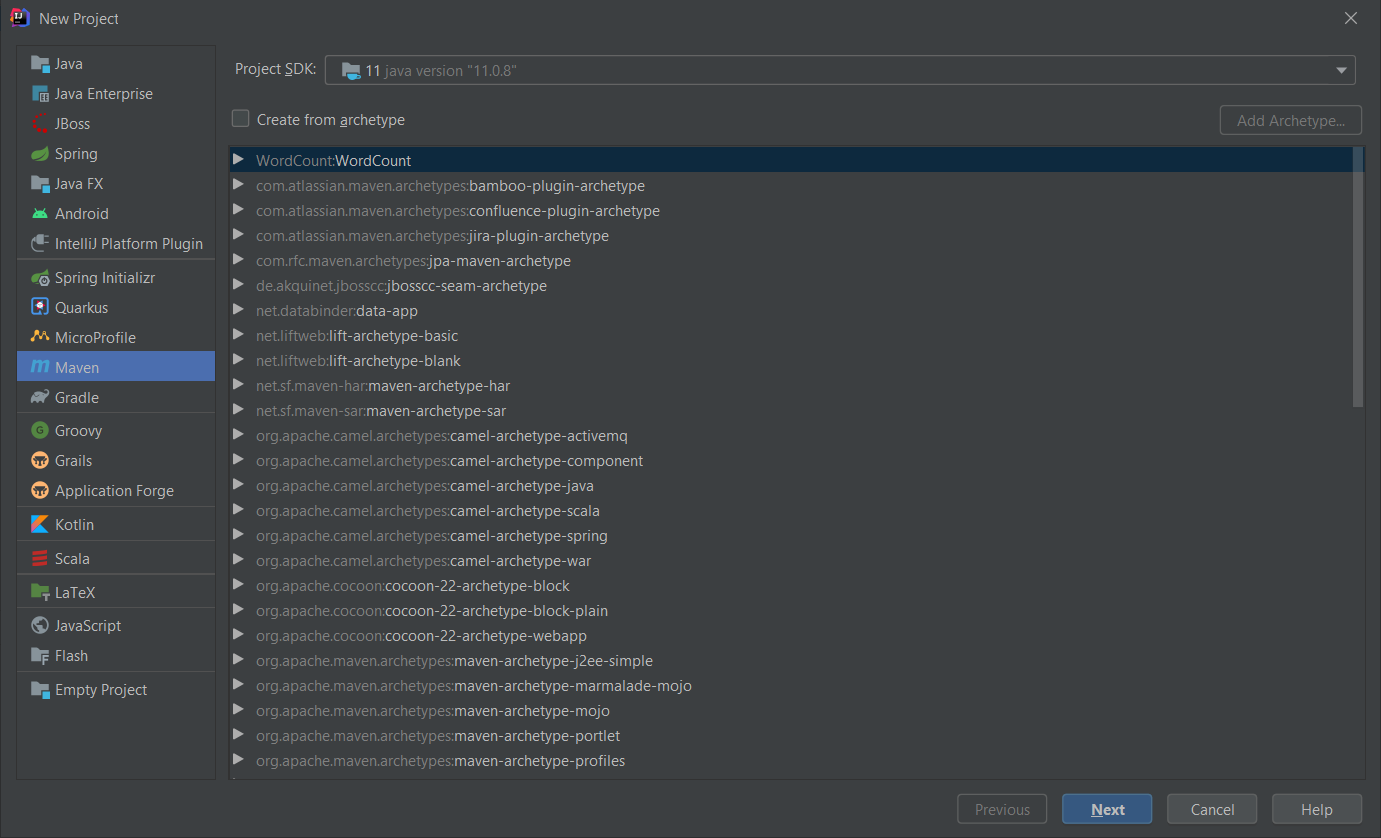


1. **Install IntelliJ IDEA**

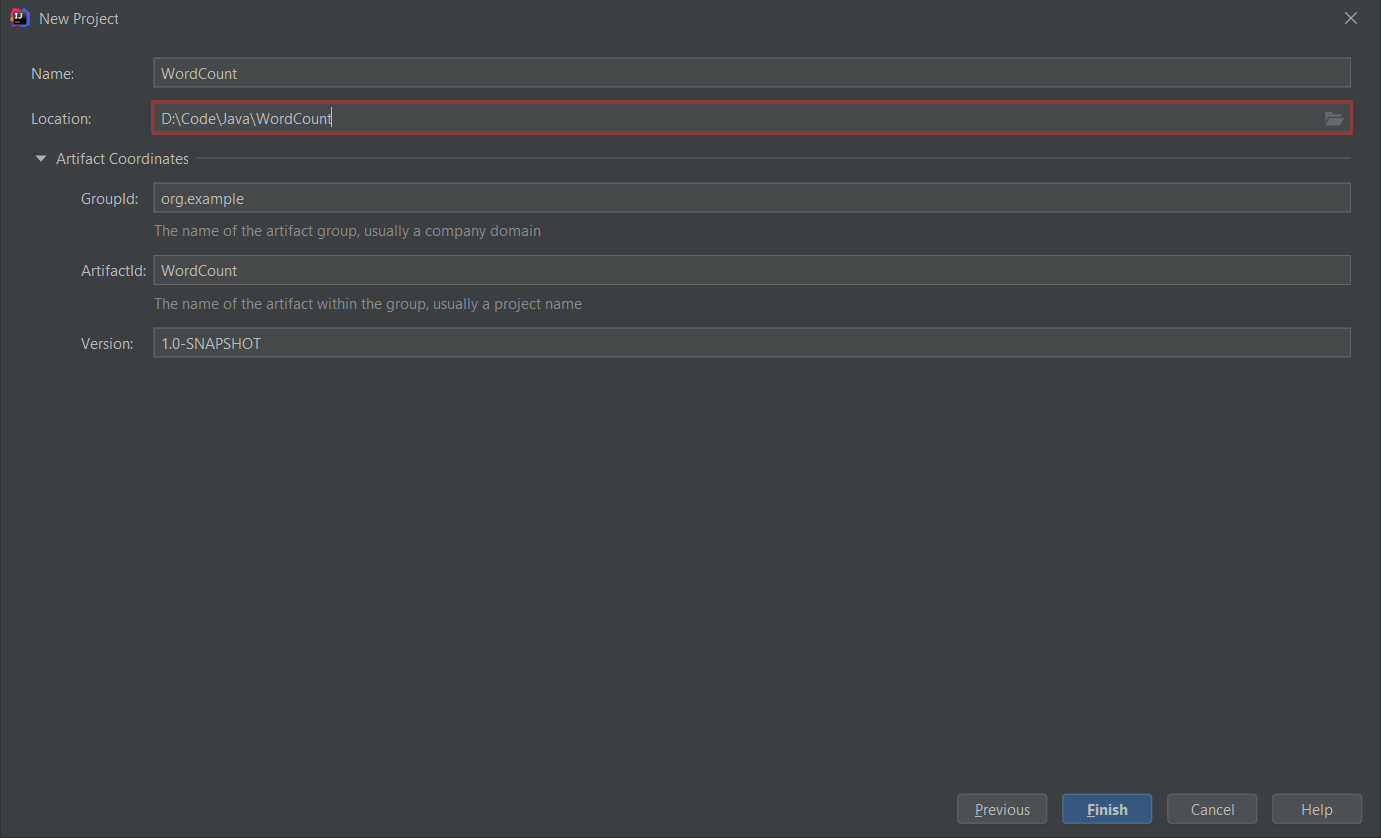
Download latest IntelliJ IDEA from  
<https://www.jetbrains.com/idea/download/#section=windows> and install it.

1. **Configure IDEA with Hadoop**

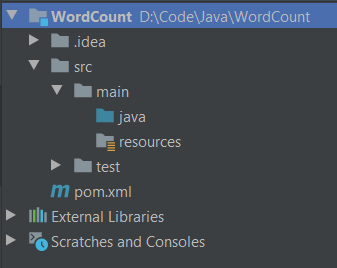
Create Maven project by "File 🡪 New 🡪 Project 🡪 Maven".



Then, click "Next", enter project WordCount. Click "Artifact Coordinates", enter information like the below figure.



Click Finish to create the project. Your project structure should look like this.



To prevent an error, add the following lines to pom.xml.

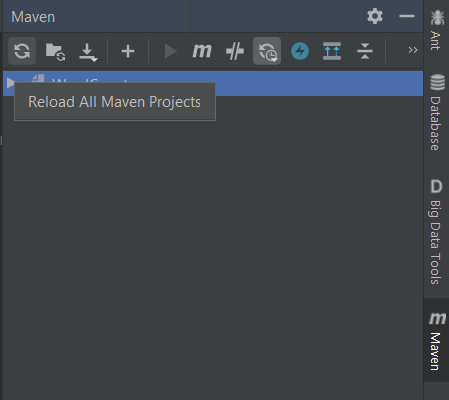
<properties>

<maven.compiler.source>1.8</maven.compiler.source>

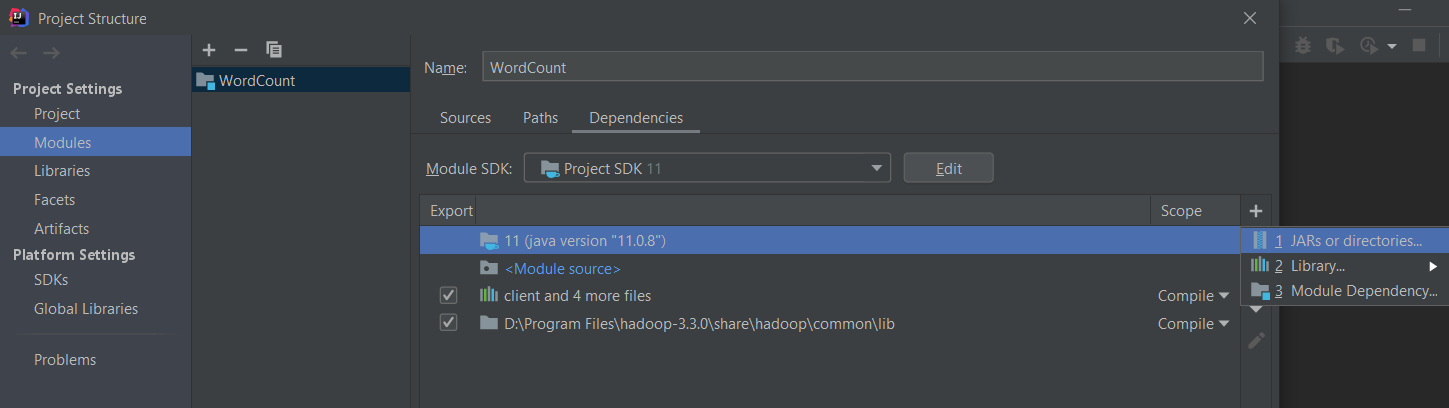
<maven.compiler.target>1.8</maven.compiler.target>

</properties>

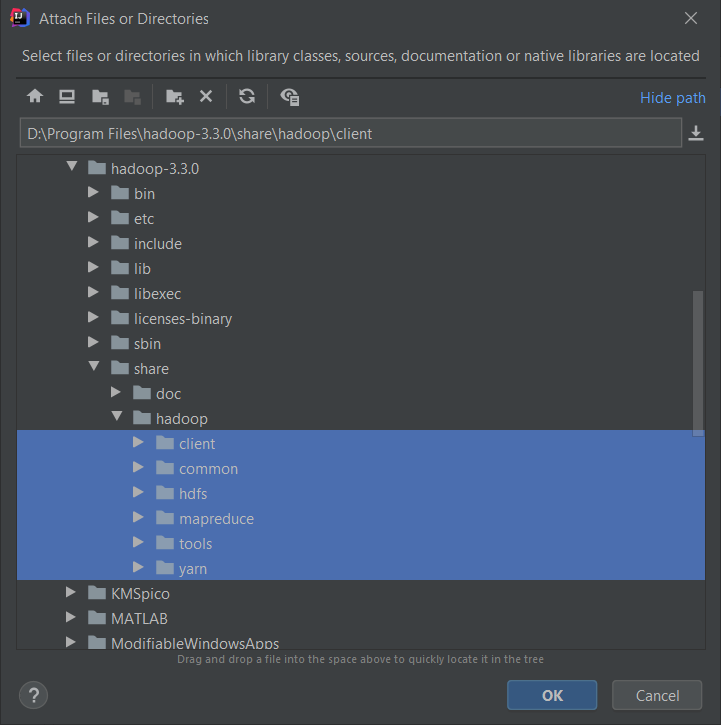
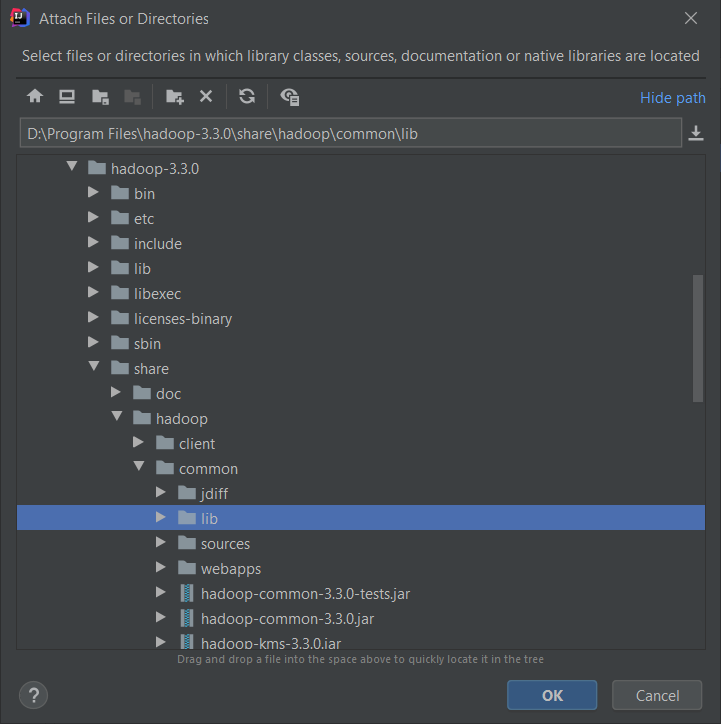
Then reload all maven projects



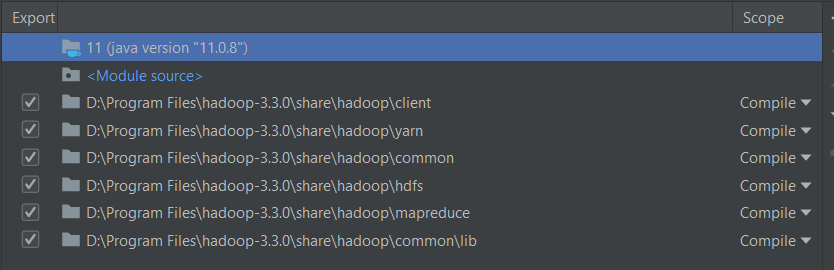
Then add Hadoop dependencies by File 🡪 Project Structure 🡪 Modules 🡪 Dependencies. Click "+ 🡪 JARS or directories"



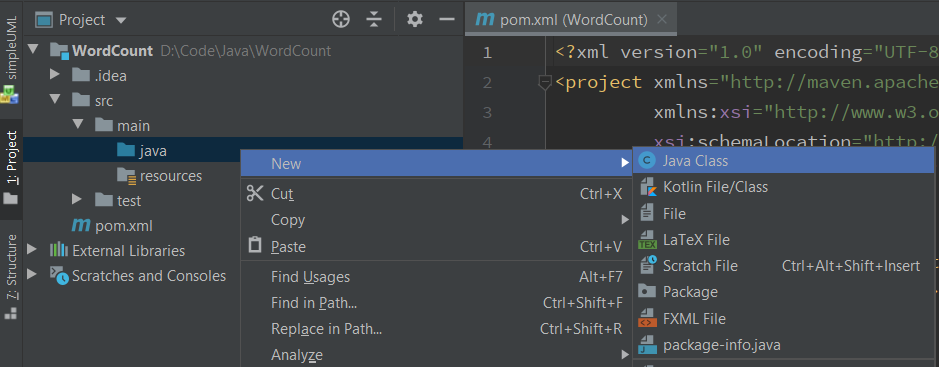
Add the following directories as dependencies.

After that, the dependencies should look like this. Then click "OK".



Create a java class file WordCount.java like below



Download assign0\_hadoop\_test from Luminus or from the cluster (see subsection 5.2). Find example codes WordCount.java in the package. Copy the content of WordCount.java in that file. Then create a directory input and two text files in the directory file0.txt and file1.txt. Their contents are as below.

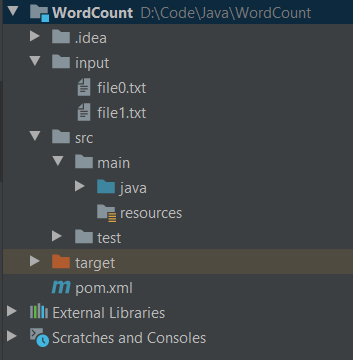
(input/file0.txt)

Hello World Bye World

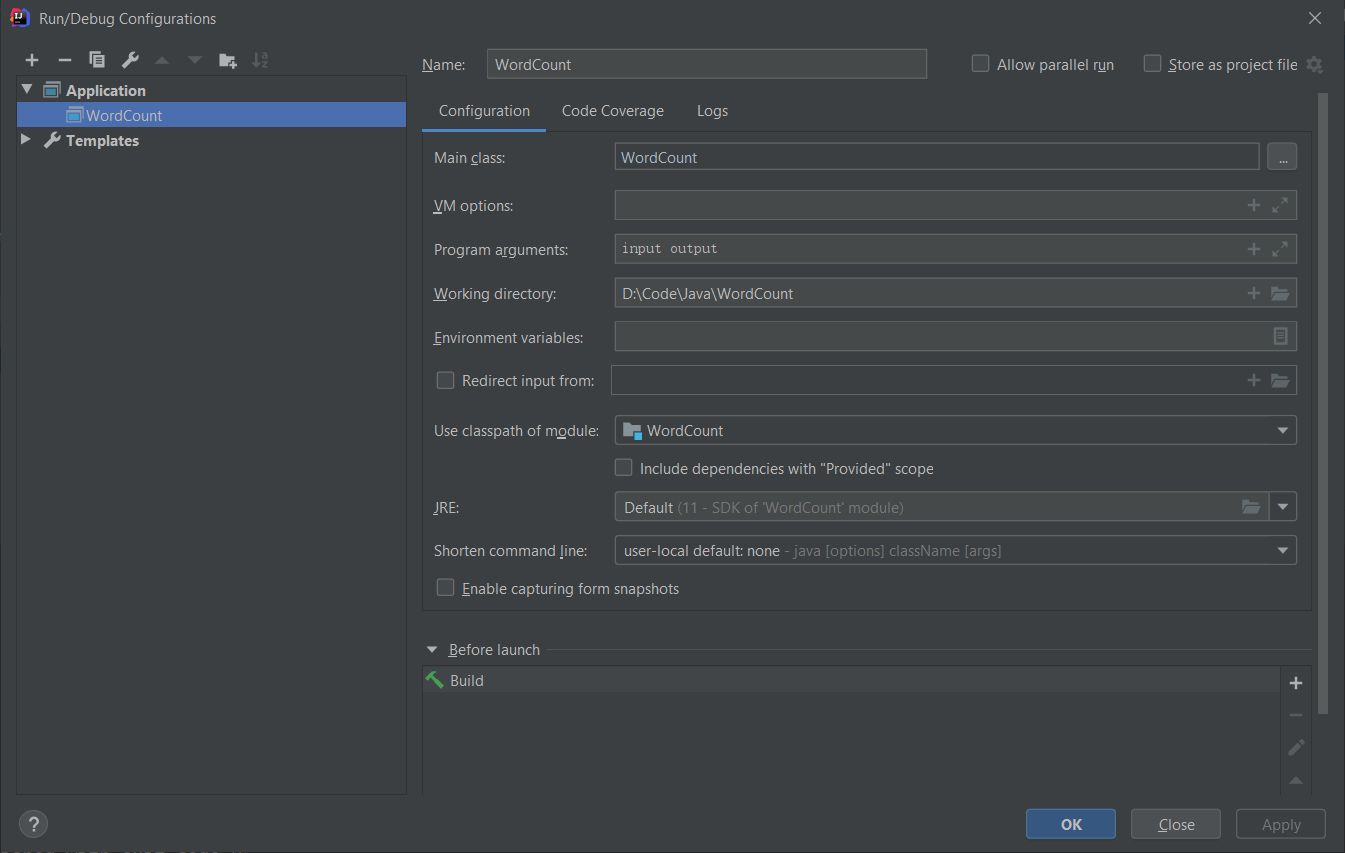
(input/file1.txt)

Hello Hadoop Goodbye Hadoop

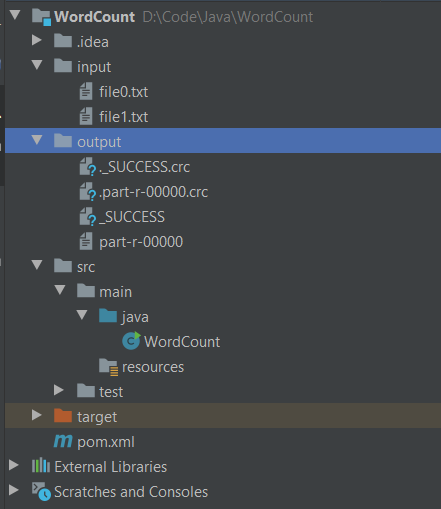
The directory structure should look like this now.



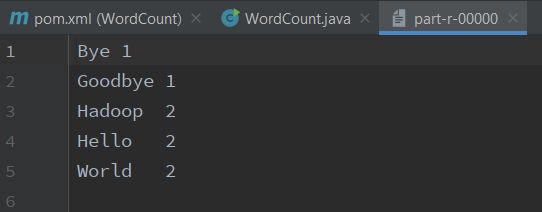
Then click "Add configurations" on right top. In the popup window, click "+ 🡪 Application". Configure as below.



Then click OK. Click the green triangular button "run" to run the program. After it finishes, the project structure will contain a new folder output.



output/part-r-00000 contains the result of word count.



You can also click to debug button to debug your program. IntelliJ IDEA is a very powerful IDE. Enjoy exploring.

1. **Install Spark**

Download Spark 3.0.0 from <https://archive.apache.org/dist/spark/spark-3.0.0>. Select spark-3.0.0.tgz, download it, and extract it with 7zip or other decompression software to your installation path of Spark, e.g. C:\\Program Files\spark-3.0.0.

1. **Configure IDEA with Spark**

Create Maven project by "File 🡪 New 🡪 Project 🡪 Maven".

Text

Description automatically generated

Then, click "Next", name the project SparkPi and click Next.

Then click Finish to create the project. Your project structure should look like this:

A screenshot of a computer

Description automatically generated with medium confidence

Now we need to install scala plugin. Navigate to File > Settings

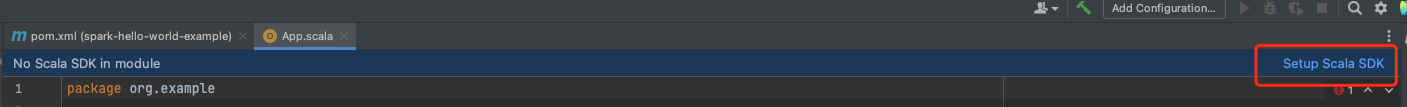
Select the Plugins option from the left panel. This brings you Feature panel.

Click on Install to install the Scala plugin.

Graphical user interface, text, application, chat or text message

Description automatically generated

After that, IntelliJ will prompt you as shown below to Setup Scala SDK (You may need to restart the IntelliJ).



Select Setup Scala SDK, it prompts you the below window, select the create option.

Graphical user interface, application

Description automatically generated

From the next window select the Download option and choose the Scala version 2.12.10. You should see the following window and click OK.

Graphical user interface, application

Description automatically generated

You can also choose the scala version by "File 🡪 Project Structure 🡪 Global Libraries".

Graphical user interface, text

Description automatically generated

Then add Spark dependencies by File 🡪 Project Structure 🡪 Modules 🡪 Dependencies. Click "+ 🡪 JARS or directories". Then add the following directory as dependency.

Graphical user interface

Description automatically generated with low confidence

Then add same Hadoop dependencies as Section 2.1.e). You should put Spark dependency ahead of Hadoop dependencies. You may also import dependencies graphframes or spark-xml for your assignment2, you need to import them after the Hadoop dependencies.

Graphical user interface, text, application

Description automatically generated

Create a java class file SparkPi.java like below

Graphical user interface

Description automatically generated

Then paste the following code into the java file.

import org.apache.spark.SparkConf;

import org.apache.spark.api.java.JavaRDD;

import org.apache.spark.api.java.JavaSparkContext;

import org.apache.spark.api.java.function.Function;

import org.apache.spark.api.java.function.Function2;

import java.util.ArrayList;

import java.util.List;

/\*\*

\* Computes an approximation to pi

\* Usage: JavaSparkPi [slices]

\* https://github.com/apache/spark/blob/master/pom.xml

\*/

public final class SparkPi {

static boolean runOnCluster = false;

public static void main(String[] args) throws Exception {

SparkConf sparkConf = new SparkConf().setAppName("SparkPi");

int slices = 0;

JavaSparkContext jsc = null;

if (!runOnCluster) {

sparkConf.setMaster("local[2]");

sparkConf

.setJars(new String[] { "target/eduonix\_spark-deploy.jar" });

slices = 10;

jsc = new JavaSparkContext(sparkConf);

} else {

slices = (args.length == 1) ? Integer.parseInt(args[0]) : 2;

jsc = new JavaSparkContext(sparkConf);

}

int n = 100000 \* slices;

List<Integer> l = new ArrayList<Integer>(n);

for (int i = 0; i < n; i++) {

l.add(i);

}

JavaRDD<Integer> dataSet = jsc.parallelize(l, slices);

int count = dataSet.map(new Function<Integer, Integer>() {

@Override

public Integer call(Integer integer) {

double x = Math.random() \* 2 - 1;

double y = Math.random() \* 2 - 1;

return (x \* x + y \* y < 1) ? 1 : 0;

}

}).reduce(new Function2<Integer, Integer, Integer>() {

@Override

public Integer call(Integer integer, Integer integer2) {

return integer + integer2;

}

});

System.out.println("Pi is roughly " + 4.0 \* count / n);

jsc.stop();

}

}

Then you can build and run your project, it should output an estimation value of Pi.

Text

Description automatically generated

* 1. Linux

1. **Install Java 11**

There are a lot of tutorials about installing Java 11, you can choose one based on your Linux distribution.

* Ubuntu: <http://ubuntuhandbook.org/index.php/2018/11/how-to-install-oracle-java-11-in-ubuntu-18-04-18-10/>
* CentOS: <https://linuxhint.com/install_oracle_jdk11_centos7/>
* Arch Linux: <https://wiki.archlinux.org/index.php/Java>
* Fedora: <https://www.tecmint.com/install-java-in-fedora/>

After installation, check your java version by following command

$ java –version

openjdk version "11" 2018-09-25

OpenJDK Runtime Environment 18.9 (build 11+28)

OpenJDK 64-Bit Server VM 18.9 (build 11+28, mixed mode)

You should ensure the JDK version is 11. The implementation could be either OpenJDK or Oracle JDK. Meanwhile, remember the path where your install java, e.g. /usr/lib/java. Add an environmental variable JAVA\_HOME by

$ echo 'export JAVA\_HOME=/usr/lib/java' >> ~/.bash\_profile

$ source ~/.bash\_profile

You can check if successful by

$ echo $JAVA\_HOME

/usr/lib/java

Note: Do not simply copy the commands. You need to check your installation path first.

1. **Install Hadoop**

Download Hadoop 3.3.0

$ wget https://archive.apache.org/dist/hadoop/common/hadoop-3.3.0/hadoop-3.3.0.tar.gz

$ tar xzvf hadoop-3.3.0.tar.gz

Configure java path for Hadoop: recall the path where you install java in a), e.g. /usr/lib/java. Edit hadoop-3.3.0/etc/hadoop/hadoop-env.sh. Find export JAVA\_HOME=, and change this line to export JAVA\_HOME=/usr/bin/java.

Note: Do not simply copy the commands. You need to check your installation path first.

1. **Install Spark**

Download Spark 3.0.0

$ wget https://archive.apache.org/dist/spark/spark-3.0.0/spark-3.0.0-bin-hadoop3.2.tgz

$ tar xvf spark-\*

1. **Install IntelliJ IDEA**

Download latest IntelliJ IDEA from (Ultimate is free for NUS students, Community is enough for this module)

<https://www.jetbrains.com/idea/download/#section=linux>

Unzip the file

$ tar xzvf ideaIC-2020.2.tar.gz

run IDEA by

$ cd ideaIC-2020.2

$ bin/idea.sh

1. **Configure IDEA with Hadoop and Spark**

This part is exactly the same as that on Windows 10. Please refer to that subsection.

* 1. MacOS

1. **Install Java 11**

Follow the guide of Linux in section 2.2 a).

1. **Install Hadoop**

You can install via brew. Simply run

brew install Hadoop

Make sure your installed version is 3.3.0. You can also install as the Linux guide in section 2.2 b).

Configure java path for Hadoop, recall the path where you install java in a), e.g. /usr/lib/java. Edit <hadoop-installation-path>/etc/hadoop/hadoop-env.sh. Find export JAVA\_HOME=, change this line to export JAVA\_HOME=<Java-installation-path>.

Note: Do not simply copy the commands. You need to check your installation path first.

1. **Install Spark**

You can install via brew. Simply run

brew install apache-spark

Make sure your installed version is 3.0.0. You can also install as the Linux guide in section 2.2 c).

Add the following environment variables to your .bash\_profile or .zshrc:

export SPARK\_HOME=/usr/local/Cellar/apache-spark/3.0.0/libexec

export PATH="$SPARK\_HOME/bin/:$PATH"

spark Note: Do not simply copy the commands. You need to check your Spark path first.

1. **Install IntelliJ IDEA**

Download latest IntelliJ IDEA from (Ultimate is free for NUS students, Community is enough for this module)

<https://www.jetbrains.com/idea/download/#section=mac>

1. **Configure IntelliJ IDEA**

This part is exactly the same as that on Windows 10

1. **Configure IDEA with Hadoop and Spark**

Please refer to that subsection.

1. Login to SoC Clusters

This is required for all students who take this module since all the assignments will be run, submitted and graded on SoC clusters.

* 1. Create SoC account

All the students from SoC and students who take SoC modules can register a SoC account. Registration and enabling clusters are done on **‘mySoC’**. For more details, please refer to the following link: <https://dochub.comp.nus.edu.sg/cf/guides/compute-cluster/enable-disable-access> .

* 1. Report SoC account

Please email Sixu Hu(e0409758@u.nus.edu) or Yuhang Chen([e0546081@u.nus.edu](mailto:e0546081@u.nus.edu)) if you cannot access the SoC Cluster. No report is required if your account is created before/on that date.

* 1. Login to SoC Cluster

If you are connecting to SoC network, you can connect to xcnd<20-59>.comp.nus.edu.sg directly via ssh. We have divided all student into 4 different cluster, each cluster can only access its own 10 nodes. For example, if you are in cluster 1, you can only access xcnd20-xcnd29, you can find your cluster number using the follow google sheet link: <https://docs.google.com/spreadsheets/d/1CYQq56ymiV6N1k7MgBA4ezwS9QjVDmrutc2N3lAFZ0A/edit#gid=0> . For load balancing purposes, try to choose one of these nodes randomly, or one that is less occupied.

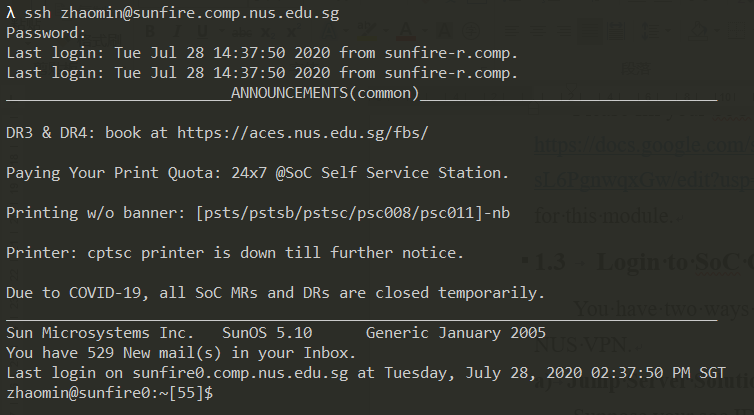
Take studentA’s xcnd26 account as an example.

$ ssh studentA@xcnd26.comp.nus.edu.sg

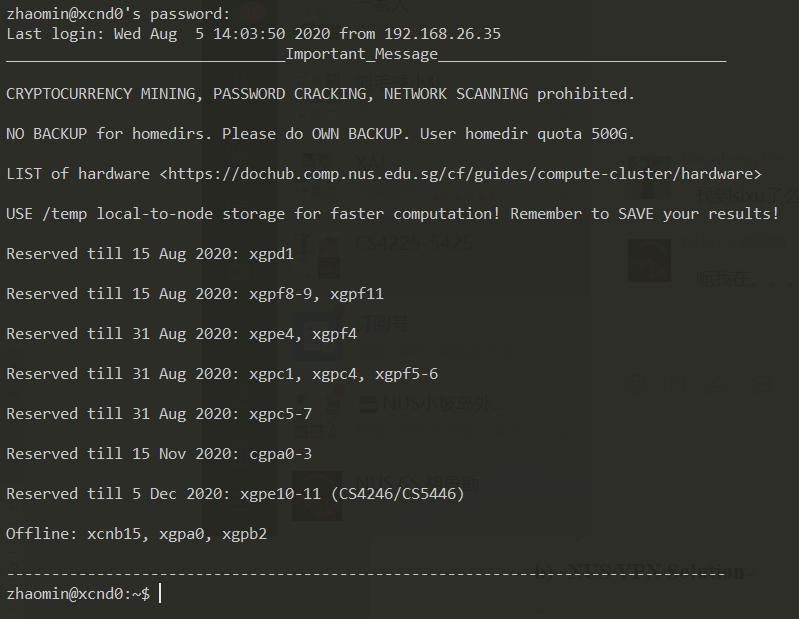
You have two ways to connect to SoC cluster: through a jump server or through NUS VPN. The second approach is only applicable for Windows and Mac users. For Linux users, please connect via jump server.

1. **Jump Server Solution**

Suppose your soc ID is "studentA”, open a terminal, type   
$ ssh studentA@sunfire.comp.nus.edu.sg  
and return. Then input your soc account password, and then you should successfully connect to the jump server.

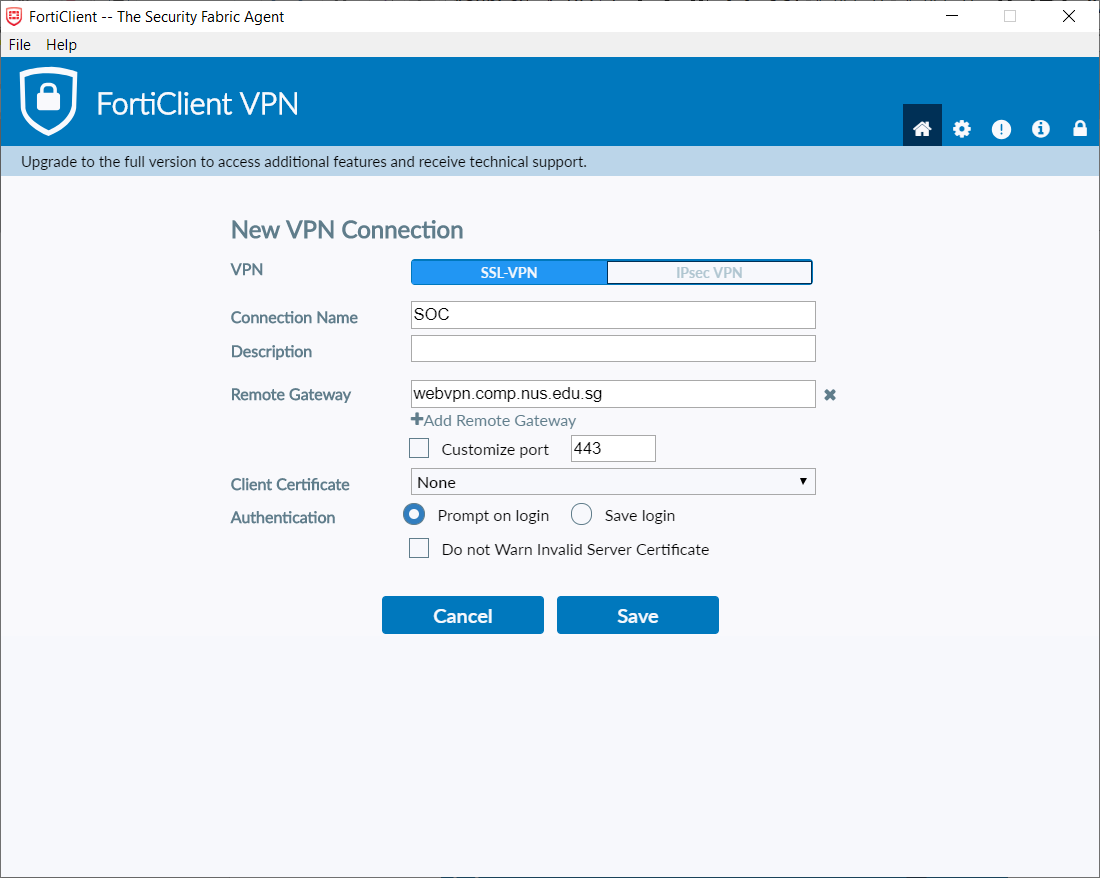


Then type ssh xcnd26 (or any server xcnd26-29) on sunfire and return, then type your SoC account password. You will connect to one machine in SoC clusters.

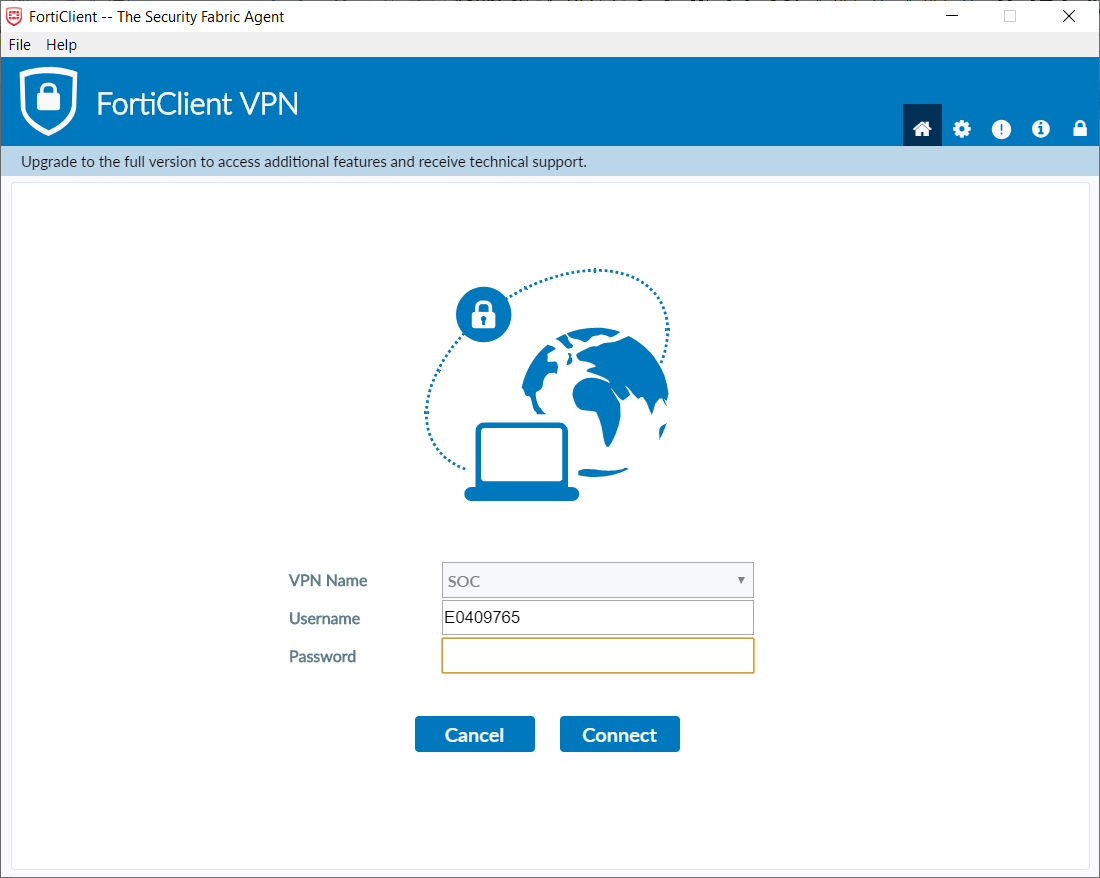


1. **NUS VPN Solution**

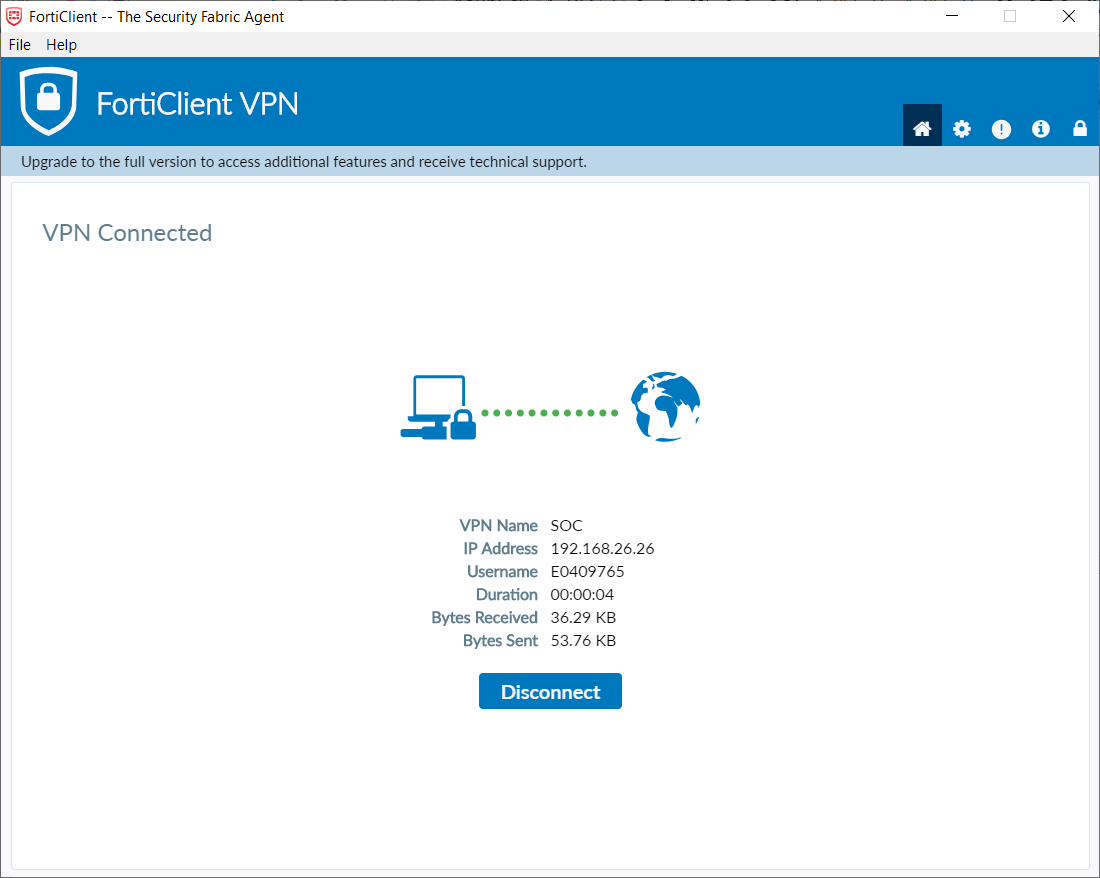
For Windows and Mac users, please download FortiClient VPN from the following link: <https://webvpn.comp.nus.edu.sg/sslvpn/portal.html#/> . Then install and run FortiClient VPN on your laptop. After launching FortiClient, you should observe



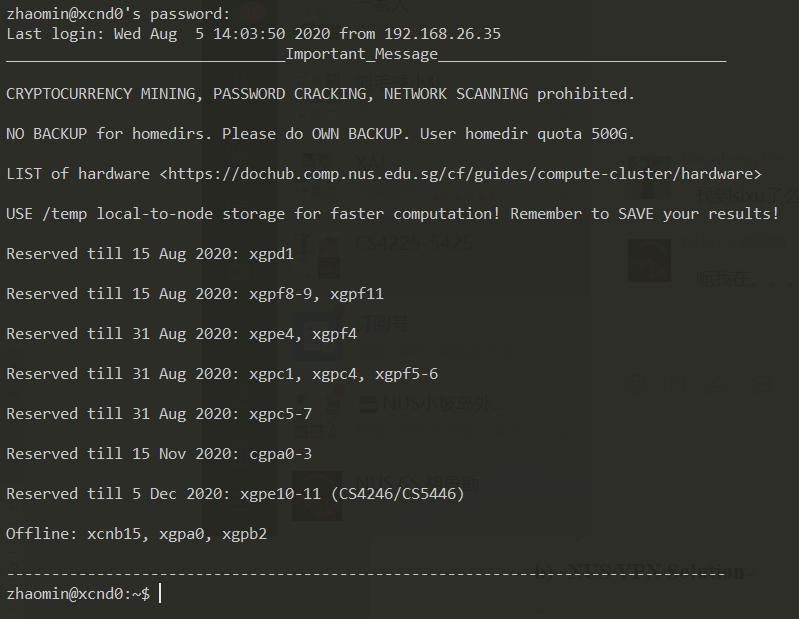
Fill in the information as above, then click "Save", you should see



Choose "VPN Name” as "SOC”, fill in your NUSNET ID and password, click "connect”. After a few seconds, you should see the following information, which indicates successful connection.



After successfully connecting, suppose your soc account id is "studentA”, open terminal and type ssh studentA@xcnd0.comp.nus.edu.sg. Then type your soc account password, then you will connect to one machine in SoC clusters.



1. Configure Hadoop and Spark

Hadoop and spark have been already installed on the clusters. All you need to do is to run it. All the following procedures are done on clusters, e.g. xcnd26.

* 1. Modify Environmental Variables

First open ~/.bash\_profile by vim

$ vim ~/.bash\_profile

Press i on keyboard to enter insert mode. Copy the following contents and paste at the end of the file. (Important: make sure you enter insert mode before pasting; otherwise, some characters will be missing when you paste).

BASE\_DIR=/home/s/sixuhu

export JAVA\_HOME=$BASE\_DIR/java

export HADOOP\_HOME=$BASE\_DIR/hadoop

export SPARK\_HOME=$BASE\_DIR/spark

export PATH=$PATH:$HADOOP\_HOME/bin

export PATH=$PATH:$SPARK\_HOME/bin

export HADOOP\_CONF\_DIR=$HADOOP\_HOME/etc/hadoop

export HADOOP\_MAPRED\_HOME=$HADOOP\_HOME

export HADOOP\_COMMON\_HOME=$HADOOP\_HOME

export HADOOP\_HDFS\_HOME=$HADOOP\_HOME

export YARN\_HOME=$HADOOP\_HOME

export HADOOP\_CLASSPATH=$JAVA\_HOME/lib/tools.jar

export PATH=$JAVA\_HOME/bin:$HADOOP\_HOME/bin:$PATH

export PATH=~/spark/bin:$PATH

export PATH=$HOME/.local/bin:$PATH

export PATH=$BASE\_DIR/sbt/bin:$PATH

You should see as follows.

Text

Description automatically generated

Then press "ESC" on keyboard to exit insert mode. Type :wq and return to save and exit vim. For more Vim comments, please see <https://www.fprintf.net/vimCheatSheet.html>. After saved, run the following command in terminal.

$ source ~/.bash\_profile

* 1. Check Environmental Variables

To check if the environmental variables are correctly set, simply run the following command.

$ echo $HADOOP\_HOME && echo $SPARK\_HOME

If you get the following results, that means your configuration is successful.



1. Test Configuration
   1. Test Spark

To test the availability of spark, simply run an example program of spark by

spark-submit --deploy-mode client --class org.apache.spark.examples.SparkPi $SPARK\_HOME/examples/jars/spark-examples\_2.12-3.0.0.jar

This program will estimate the value of . After some calculation (few seconds to a minute), it should output the similar results to the following one.

(Many other outputs)

Pi is roughly 3.1350556752783763

2020-08-04 20:55:12,042 INFO server.AbstractConnector: Stopped Spark@31f0ddb1{HTTP/1.1,[http/1.1]}{0.0.0.0:4040}

2020-08-04 20:55:12,049 INFO ui.SparkUI: Stopped Spark web UI at http://xcnd0.comp.nus.edu.sg:4040 2020-08-04 20:55:12,055 INFO cluster.YarnClientSchedulerBackend: Interrupting monitor thread

2020-08-04 20:55:12,083 INFO cluster.YarnClientSchedulerBackend: Shutting down all executors

2020-08-04 20:55:12,084 INFO cluster.YarnSchedulerBackend$YarnDriverEndpoint: Asking each executor to shut down

2020-08-04 20:55:12,091 INFO cluster.YarnClientSchedulerBackend: YARN client scheduler backend Stopped

2020-08-04 20:55:12,108 INFO spark.MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopped!

2020-08-04 20:55:12,135 INFO memory.MemoryStore: MemoryStore cleared

2020-08-04 20:55:12,135 INFO storage.BlockManager: BlockManager stopped

2020-08-04 20:55:12,145 INFO storage.BlockManagerMaster: BlockManagerMaster stopped

2020-08-04 20:55:12,149 INFO scheduler.OutputCommitCoordinator$OutputCommitCoordinatorEndpoint: OutputCommitCoordinator stopped!

2020-08-04 20:55:12,212 INFO spark.SparkContext: Successfully stopped SparkContext

2020-08-04 20:55:12,219 INFO util.ShutdownHookManager: Shutdown hook called

2020-08-04 20:55:12,221 INFO util.ShutdownHookManager: Deleting directory /tmp/spark-1dc8df74-cc87-465d-b4fd-3ff6f54b00ef

2020-08-04 20:55:12,226 INFO util.ShutdownHookManager: Deleting directory /tmp/spark-4fbed1cb-afe1-431f-b9d4-94279a6266b9

* 1. Test Hadoop and Submission

In this subsection, we will go through the procedure of “a mock test” using assignment 0 (a simple test program, whose code is already fully written) for a real assignment, including downloading files, writing codes, running program and submitting. All the operations in this subsection are performed on the clusters.

1. **Download Assignment Files**

All required packages will be stored in /home/y/yuhangc on SoC clusters. You can download these files by simply copying them to your home directory. Type the following command in terminal and return.

$ cp -r /home/y/yuhangc/assign0\_hadoop\_test ~

Then you should find a new folder assign0\_hadoop\_test in your home directory.

Simply check by

$ ls assign0\_hadoop\_test



Alternatively, if you want to first write your codes locally, you can download assignment files from LumiNUS and upload to the server after you finished. The upload can be done by scp. With your (e.g. stuA's) NUS VPN connected, enter the folder which contains the directory you want to upload, and run this command on your own laptop

$ scp -r assign0\_hadoop\_test stuA@xcnd26.comp.nus.edu.sg:~

After finishing copying, you will find a new folder assign0\_hadoop\_test under your home directory on xcnd26. If you are using Linux and cannot connect to xcnd26, you can upload your submission to sunfire first and then upload to xcnd26 by scp. To learn more about scp, please refer to: <https://haydenjames.io/linux-securely-copy-files-using-scp/>.

1. **Write your code**

For assignment 0, the code in WordCount.java is already written for you.

1. **Compile and Run Your Codes**

Enter that folder and call the scripts to automatically compile and run

$ cd ~/assign0\_hadoop\_test

$ ./compile\_run

The script will compile and run WordCount.java, which counts the words in file01.txt and file02.txt. After a short calculation, you will see the following result.

Job finished. Print results.

Bye 1

Goodbye 1

Hadoop 2

Hello 2

World 2

Test passed.

The script will automatically compare your result to the answer. If the output is "Test Passed", that means your result is correct on the given dataset. Otherwise, it means that your result is incorrect and you need to double check your codes.

1. **Submit**

Once you have successfully tested using assignment 0, you can start on assignment 2. Similar to assignment 0, you can copy the files by

$ cp -r /home/y/yuhangc/assign2 ~

Then write your code in FindPath.java , and similar to assignment 0, use the compile\_run to test your codes.

Once you have finished the assignment, you can submit your codes into the folder AssignmentSubmission/Assignment2 in Luminus. Do remember you can only modify and submit the file FindPath.java.

You are allowed to submit multiple times before the due date; your last submission will be graded. After due time, the submission folder will be locked and no more submission will be accepted.

Although Hadoop has load balancing itself, you are still recommended to choose a machine with less CPU consumption; i.e. replace xcnd26 with xcnd27, xcnd28 or xcnd29. You can check CPU and memory consumption by htop. In case you face issues with the clusters for some reason (e.g. if the clusters go down or becomes excessively slow near the deadline), please email the TAs who will try to handle it (or extend the deadline if necessary).

1. Java OpenStreetMap Editor (Optional)

Java OpenStreetMap (JOSM) is an extensible editor for ​OpenStreetMap (OSM), you can use it to load the OSM files in assignment2.

* 1. Download

Please refer to <https://josm.openstreetmap.de/> to download and install the JOSM package based on your operating system.

* 1. JOSM Basic Operations

1. **Load OSM file**

After you successfully install and launch the JOSM editor, you should observe

**Graphical user interface, text, application, email

Description automatically generated**

Then you can click File 🡪 Open to load an OSM file, for assignment2, you only need to use the file ‘NUS.osm’.

**Graphical user interface, text, application

Description automatically generated with medium confidence**

If you successfully load the NUS roadmap, you should observe the map similar to:

**Graphical user interface

Description automatically generated**

You can change the map paint styles for a better view.

**Map

Description automatically generated**

1. **Filter and search in the map**

For assignment2, we only care about the way with a highway tag, we can use the filter function to find those roads.

You can open the Filter window by clicking the Windows 🡪 Filter

**Graphical user interface, text, application, chat or text message

Description automatically generated**

After that, you should be able to see the Filter window on the right side of the JOSM editor.

**Graphical user interface, text, application, email

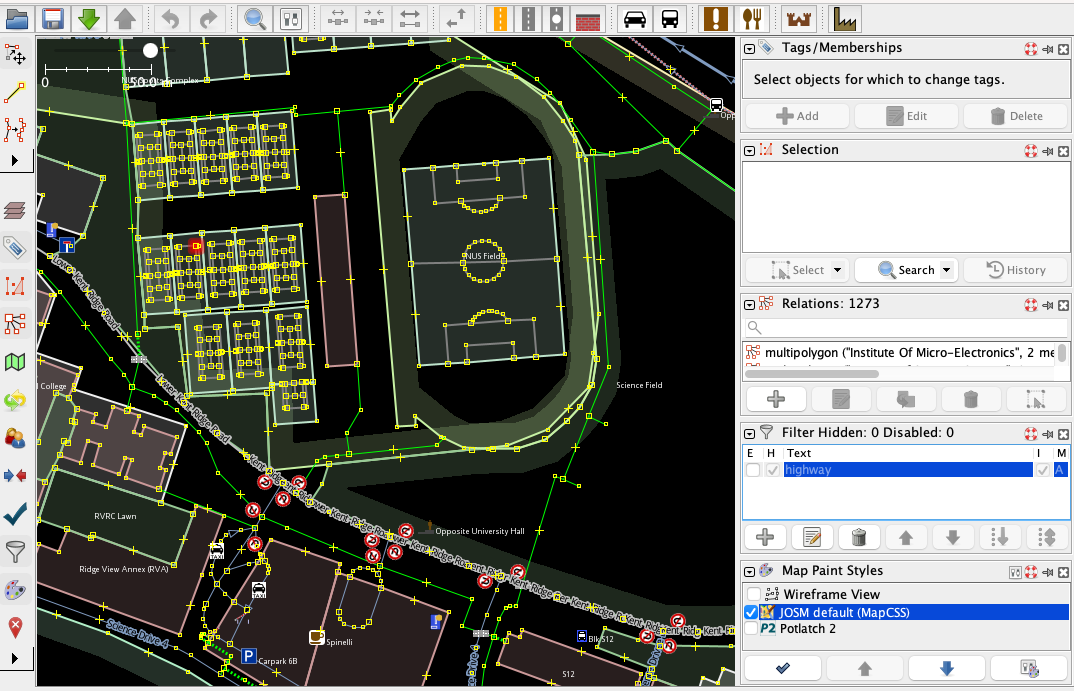
Description automatically generated**

Then you can click the '+' symbol to add the filter string highway, then click the Submit filter.

**Graphical user interface, text, application

Description automatically generated**

After that, you should click all the check box in the filter window to only show the roads with highway tag.

****

**Playground before filtering**

**Graphical user interface, application

Description automatically generated**

**Playground after filtering**

You can also search for a certain node using its id by clicking the search button (as shown in the red box), then input ‘id:<node\_id>’ in the search string.

**Graphical user interface, text, application

Description automatically generated**

After clicking search, the corresponding node will be highlighted in the map.

**A picture containing graphical user interface

Description automatically generated**

1. **Load gpx file**

Our osmutils dijkstra command (please refer to the assignment2 slides) support to find and output a psedo-path between two nodes in GPS exchange format, you can also load this file using JSOM.

You can click File 🡪 Open to load an GPX file

Graphical user interface

Description automatically generated with medium confidence

After loading the file, you can see this path is highlighted in the map.

Map

Description automatically generated with medium confidence

You can refer to <https://josm.openstreetmap.de/wiki/Introduction> for more details.

1. Q&A and Contact

If you have any questions, you are encouraged to submit them to the forum in LumiNUS. Otherwise, please contact Mr. Sixu Hu, e0409758@u.nus.edu or Mr. Yuhang Chen, [e0546081@u.nus.edu](mailto:e0546081@u.nus.edu). Please note that, we may anonymize your question, and post the question as well as the answer to the forum (for sharing).

**For your reference:** we have included some frequent Q&A from previous semesters. You can use it as reference (but note it may contain outdated information).

**Q: Failed to ssh to xcnd44: Connection closed by (IP)**

Reason: SoC clusters is not enabled by default. Solution: Enable SoC clusters at mySoC

**Q: -bash: ./compile\_run: Permission Denied**

Reason: You need to grant execute permission to compile\_run. Solution: Grant execute permission by chmod +x compile\_run

**Q: Fine on local machine, "ArrayIndexOutOfBoundsException" or "NullPointerException", or other unexpected output on clusters**

Reason: One possible reason is that static class variables are used in Hadoop, which will not be shared across machines. Solution: Avoid using static variables.

**Q: Fine on local machine, got "Wrong Answer" or "FileNotFoundException: stopwords.txt"**

Reason: Load stopwords.txt directly from the system. This works fine on local machine since there is no HDFS. But on clusters, files in HDFS cannot be loaded in this way. Solution: Load stopwords.txt by Java HDFS API (see the compile\_run script given in the assignment package)

**Q: Fine on local machine, got "ClassNotFoundException for 'Pair'" on clusters**

Reason: Not known yet. Solution: Use another alternative data structure instead of Pair. Or, implement a custom Pair yourself.

**Q: I got a warning "xxx". But my code is runnable and produces the correct output (Test Passed). Why?**Reason: N/A

Solution: Submit and move on to the next task.