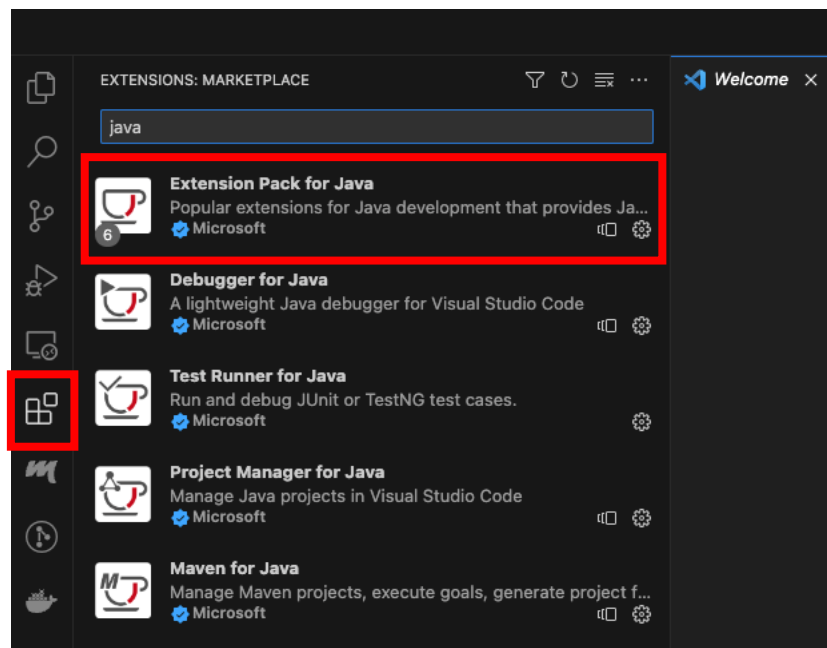


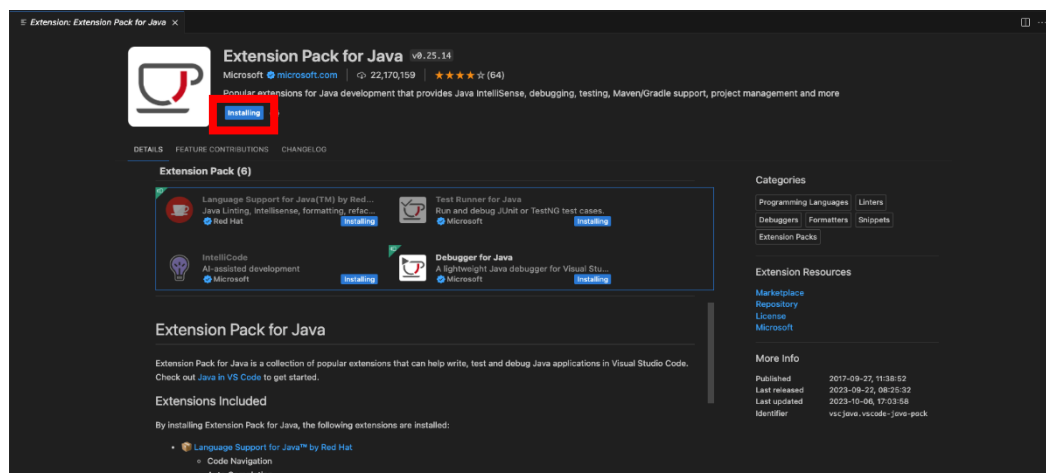
# How to Compile your Java Application using VSCode, export the Jar and run on the cluster

## Verify Java Extension for VSCode

1. Open VSCode
2. Go under Extension tab and check whether “Extension Pack for Java” is installed



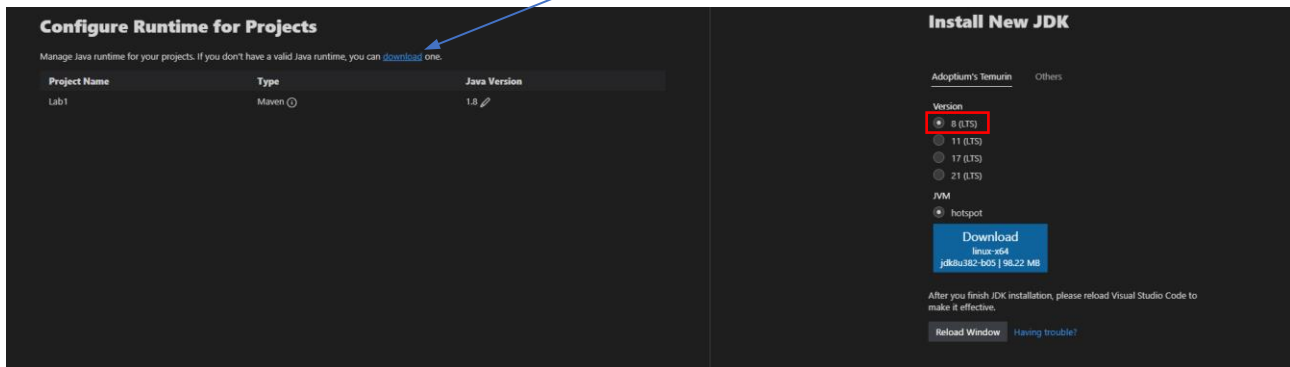
3. If not installed, you can click on the “Install” button in the page associated to the extension.



## Install JDK and Maven

Install Java with JDK 1.8 (8 LTS) - *jdk8u382* from the VSCode interface). By opening the “Configure Runtime for Projects” panel (see the section below for more details), you will see a “download” button. **Pay attention to installing the right version.** If you already have JDK 1.8, you can proceed.

Press here to open the right panel



Then, you can install Maven downloading the right version for your operating system at <https://maven.apache.org/download.cgi>.

## Files

Maven is distributed in several formats for your convenience. Simply pick a ready-made binary distribution archive and follow the [installation instructions](#). Use a source archive if you intend to build Maven yourself.

In order to guard against corrupted downloads/installations, it is highly recommended to [verify the signature](#) of the release bundles against the public [KEYS](#) used by the Apache Maven developers.

	Link	Checksums	Signature
Binary tar.gz archive	<a href="#">apache-maven-3.9.5-bin.tar.gz</a>	<a href="#">apache-maven-3.9.5-bin.tar.gz.sha512</a>	<a href="#">apache-maven-3.9.5-bin.tar.gz.asc</a>
Binary zip archive	<a href="#">apache-maven-3.9.5-bin.zip</a>	<a href="#">apache-maven-3.9.5-bin.zip.sha512</a>	<a href="#">apache-maven-3.9.5-bin.zip.asc</a>
Source tar.gz archive	<a href="#">apache-maven-3.9.5-src.tar.gz</a>	<a href="#">apache-maven-3.9.5-src.tar.gz.sha512</a>	<a href="#">apache-maven-3.9.5-src.tar.gz.asc</a>
Source zip archive	<a href="#">apache-maven-3.9.5-src.zip</a>	<a href="#">apache-maven-3.9.5-src.zip.sha512</a>	<a href="#">apache-maven-3.9.5-src.zip.asc</a>

Linux/Mac points to the binary tar.gz archive link. Windows points to the binary zip archive link.

You will probably need to add some environmental variables. For windows users, you can follow this guide <https://phoenixnap.com/kb/install-maven-windows>.

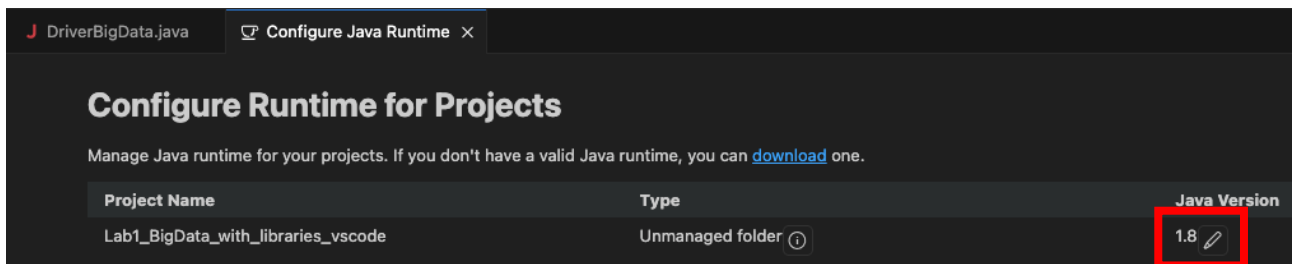
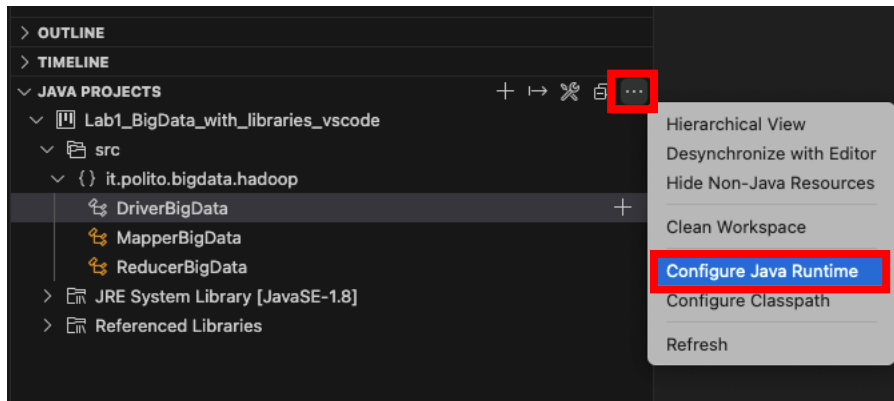
For Linux/Mac users <https://www.digitalocean.com/community/tutorials/install-maven-linux-ubuntu>. You can also look for it with your system package manager (apt, [pacman](#), [brew](#), etc.), which is probably easier.

## Import project (both Maven and “with\_libraries”)

1. Simply Open the folder containing the project under File -> “Open folder...” or “Open folder...” in the initial screen. VSCode will automatically detect whether it’s a pure-Java project or Maven-based. You can find some project templates on the course website. Extract a zip archive on a local folder to start.

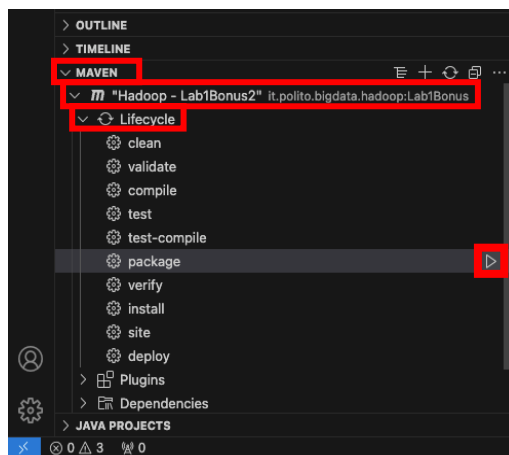
## Verify usage of JDK 1.8

1. In the lower-left part of VSCode, under “Java Project”, click on the “...” icon
2. Click on “Configure Java Runtime” in the menu
3. Verify that, associated to the opened project, a JDK 1.8 installation is selected
4. If not, you can modify it from the “edit” button



## Export JAR file from Maven project

1. In the lower-left part of VSCode, expand the “Maven” menu
2. Expand the projects’ maven entry
3. Expand the Lifecycle menu

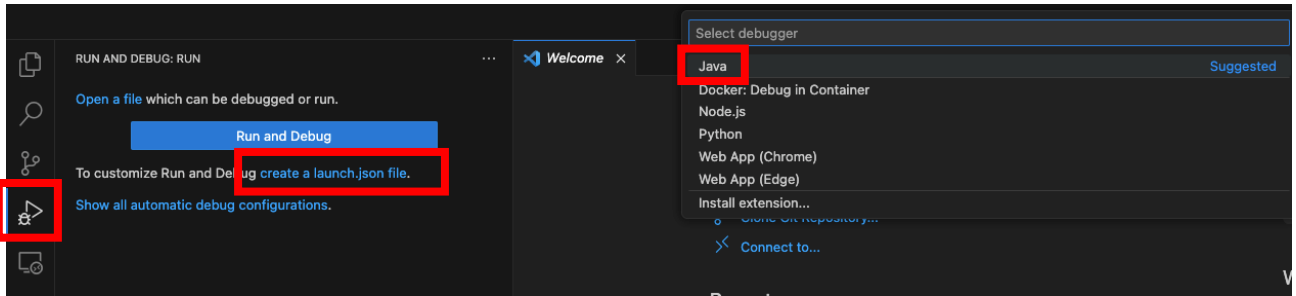


4. Click on the Run button over “package” entry
5. The exported JAR file is located under the project folder -> target -> (project name).jar

## Local Execution of Hadoop/Spark (Java) application (Maven only)

1. First, the input program arguments must be configured through the launch.json file

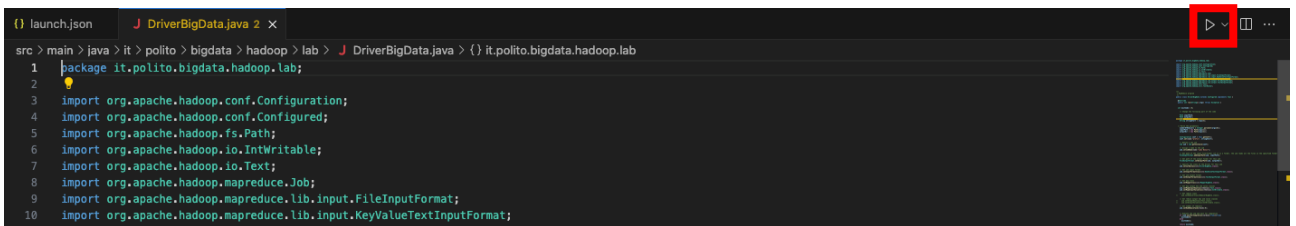
2. To create it, go under the Run tab in the left part of VSCode, select “create a launch.json file” and select the “Java” debugger



3. VSCode will automatically generate a launch.json file. You can modify it by adding the “args” entry and specify a list of input arguments, such as follows:

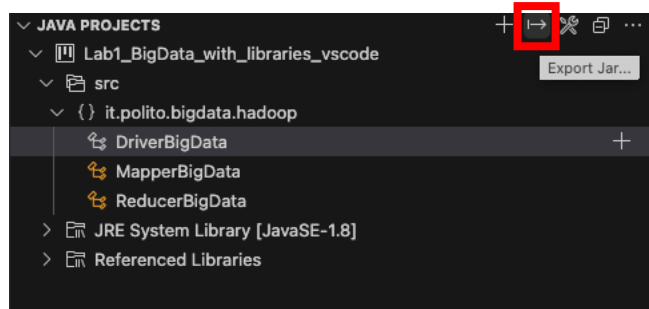
```
.vscode > {} launch.json > [] configurations > {} 1 > [] args > [] 3
1 {
2   // Use IntelliSense to learn about possible attributes.
3   // Hover to view descriptions of existing attributes.
4   // For more information, visit: https://go.microsoft.com/fwlink/?linkid=830387
5   "version": "0.2.0",
6   "configurations": [
7     {
8       "type": "java",
9       "name": "Current File",
10      "request": "launch",
11      "mainClass": "${file}"
12    },
13    {
14      "type": "java",
15      "name": "DriverBigData",
16      "request": "launch",
17      "mainClass": "it.polito.bigdata.hadoop.lab.DriverBigData",
18      "projectName": "Lab1Bonus",
19      "args": [
20        "2", "example_data", "ex_out", "test_prefix"
21      ]
22    }
23  ]
24 }
```

4. In this example, the configuration “DriverBigData” was modified by adding 4 different input program arguments.
5. To execute the application, open the DriverBigData class and click on the Run button in the upper-right corner

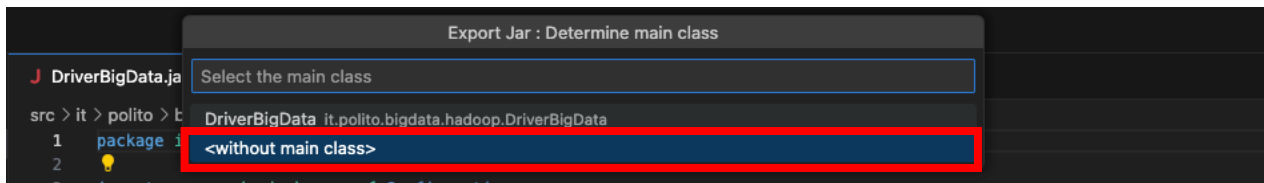


## Export JAR file from “with\_libraries” project

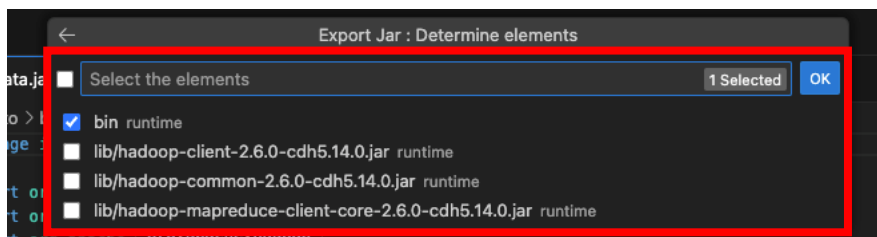
1. In the lower-left part of VSCode, in the Java Project menu, select the Export Jar button (right arrow symbol)



2. In the upper area, in the center, VSCode, will ask you to choose the main class for your project. Select <without main class>



3. Next, you will be prompted to add all the files that your jar file should contain. You can keep only the content of "bin" folder (since libraries are already present on our cluster).



4. You will find the jar file in the main folder of your project. You can upload such jar file directly on jupyter.polito.it cluster