2AMD15: Big Data Management

Team project – Spark Installation Guide

This guide will show you how to install various pieces of software (including Spark) which may be used for the team project of 2AMD15. It is based on a guide about installing Apache Spark by Goran Jevtic as found here. It should be applicable to other versions of Windows.

Prerequisites

- A system running Windows 10 or newer
- A user account with administrator privileges
- Command Prompt or Powershell
- A tool to extract archived files, such as 7-Zip or WinRAR.

Installing Java

Apache Spark requires a Java Runtime Environment (JRE) to be installed. The JRE version should be one that supports a Java version between 8 and 11 (inclusive). Note: newer versions of Java exist, but these are not supported by Spark.

Because some students will opt to work with Java as a programming language, it is convenient to only explain how to obtain and install a version of the Java Development Kit (JDK), which is required to build Java programs and includes a JRE.

Even when you do not intend to work with Java as a programming language, it is still required that you install a suitable JRE or JDK in order for Spark to work.

If you have a suitable version of the JRE or JDK installed already and do not intend to use Java as a programming language, you can skip this step.

Or, if you have a suitable version of the JDK installed already and intend to use Java as a programming language, you can skip this step.

If you have a newer version of the JRE or JDK already installed, you can 1) replace it with a suitable older version, or 2) install a suitable older version alongside the newer version. If you choose approach 2), please be extra careful at the steps regarding Environment Variables, to ensure that the older JRE/JDK is used by Spark.

A ZIP-archive containing the OpenJDK 11.0.2 which supports Java 11 can be obtained for 64-bit Windows systems here. For other versions and operating systems you can browse the OpenJDK archive here.

The archive should contain a single directory named jdk-v where v is the version (11.0.2 for the provided link). This directory can be unarchived and placed directly into the desired installation directory for Java. Typically, this is something like “C:\Program Files\Java”.

Installing Python

If you do not intend to use Python as a language for developing your solution, you can skip this step.

If you have a version of Python 3 already installed, you may be able to skip this step. Try to build a simple application using the pyspark package and to run it using spark-submit (more information later on in this guide). If you experience problems running a Python application

using your version of Python and spark-submit, it may be that Spark cannot find or is incompatible with your Python version. In this case, it is recommended that you uninstall your current Python version and install the latest version as described below.

Download the latest version of Python from the website, currently it can be found here.

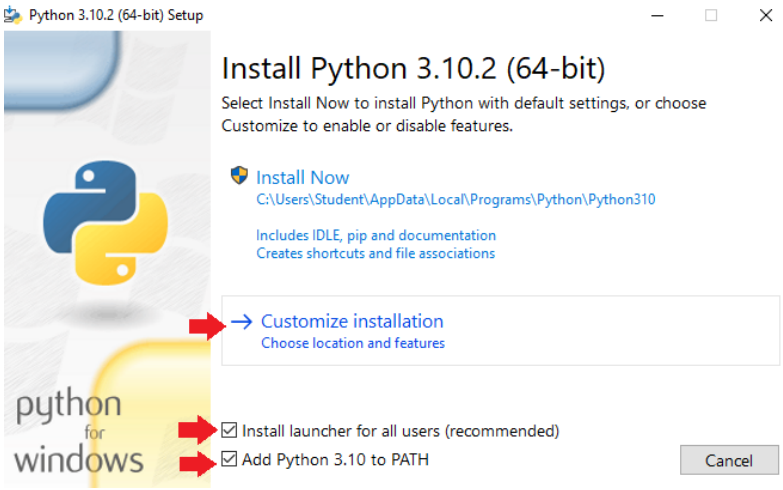
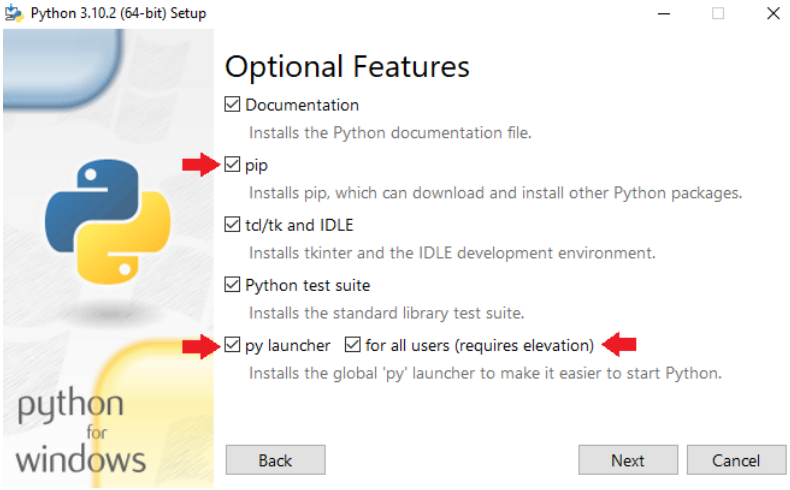
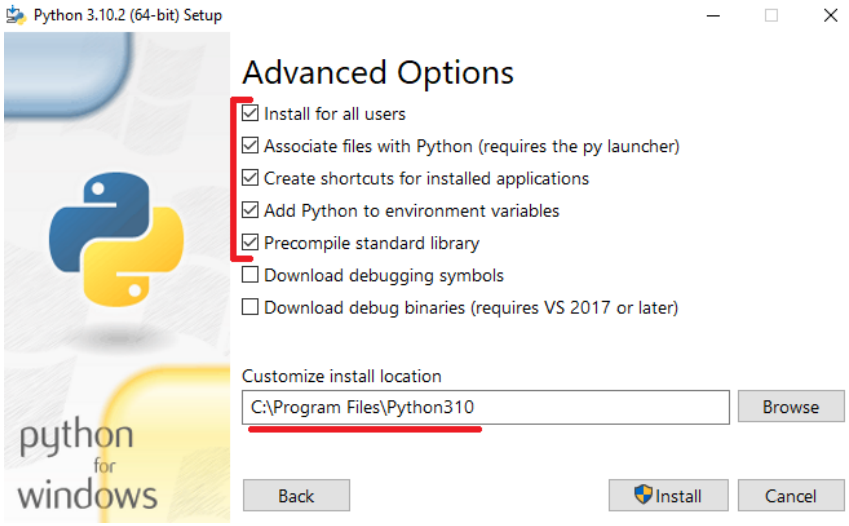


Figure 1 - Python Installation 1

On the first screen of the Python installer, make sure the boxes for “Install launcher for all users (recommended)” and “Add Python 3.10 to PATH” are checked. Then, choose the “Customize installation” option.



On the second screen of the Python installer, make sure the boxes for “pip”, “py launcher” and “for all users (requires elevation)” are checked. You can check or uncheck the other boxes as you like.

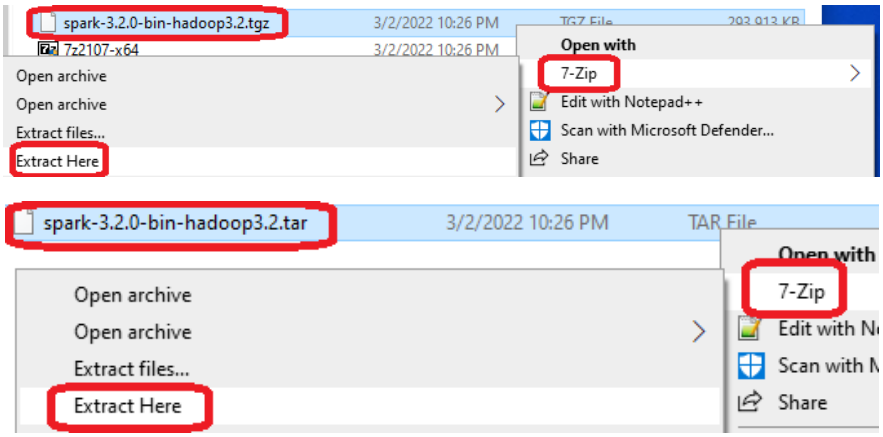


On the third screen of the Python installer, make sure the first five boxes are checked. The latter two may be checked or unchecked, as you prefer. The installation location should look similar to the one shown above.

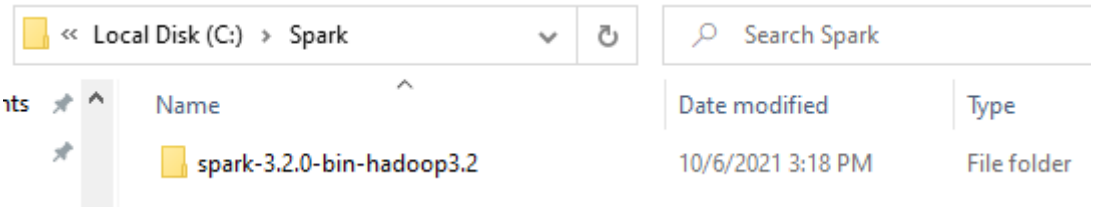
Click Install and wait for the installer to finish. Then click Close.

Installing Spark

It is recommended that you use Spark 3.2 for Hadoop 3.2, as this is the version used on the servers as well. A TAR-GZ-archive containing this version can be downloaded here. This archive contains a directory called spark-3.2.0-bin-hadoop3.2. Using 7-Zip, it may be necessary to extract the archive in two passes (first the .gz file, then the .tar file, as shown below).



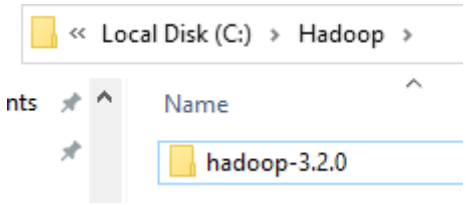
You can place this directory at any location on your system, provided it is not (directly or indirectly) inside a user directory (for example, do not place it in “C:\Users\Me\Spark”).



A recommended location is C:\Spark, as shown above.

Installing Hadoop

The Spark application depends on Hadoop. The files necessary to make Hadoop work on Windows can be downloaded here. The ZIP-archive named “winutils-master” contains a directory of the same name. From that directory, extract only the directory named “hadoop-3.2.0” and place it any location on your system, similar to the Spark installation directory. A recommended location is shown below.



Installing Apache Maven

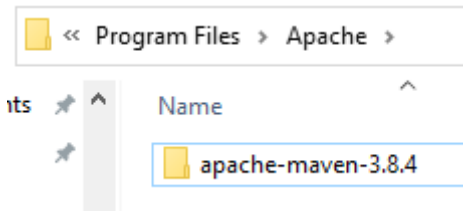
Apache Maven is a software project management tool. We will use this tool to build an executable JAR file from a Java project.

If you do not intend to use Java as a programming language, you can skip this step.

If you have a recent version of Apache Maven already installed, you can skip this step.

A ZIP-archive containing the latest version of Apache Maven can be downloaded here.

The ZIP-archive contains a directory called apache-maven-3.8.4. Like the directory for Spark in the previous step, it can be placed at any location on your system. A recommended location is shown below.

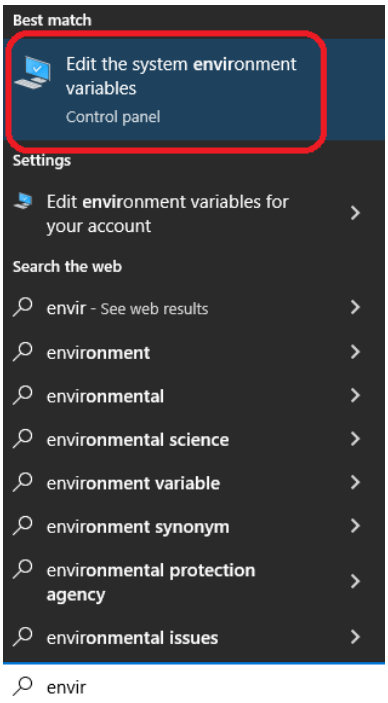


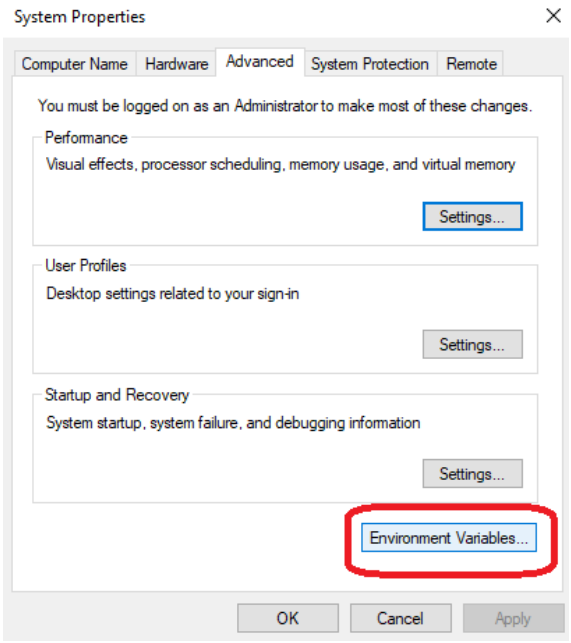
Configuring environment variables

The installations of Java, Spark, Hadoop and Maven require environment variables to be configured. For these installations, we only extracted directories which we placed at some location on our system. Yet, there is nothing that informs the operating systems (and therefore tools like Command Prompt and Powershell) where programs like Java, Spark, Hadoop and Maven are. This is where environment variables come in.

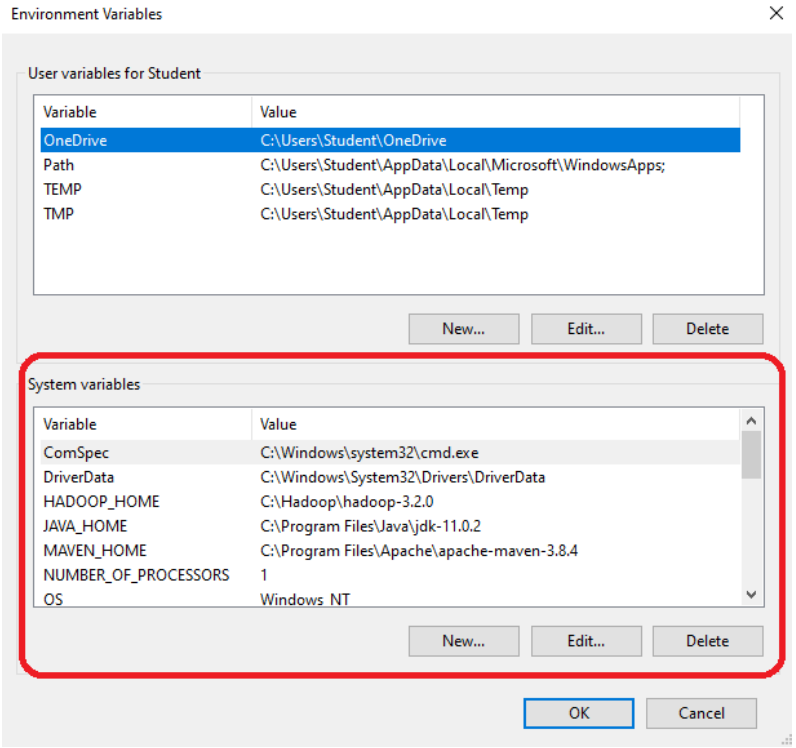
No additional configuration needs to be done in this step for the Python installation, provided the right boxes were checked during installation.

To access the environment variables, simply start typing “environment variables” in the start menu and click the option shown below.

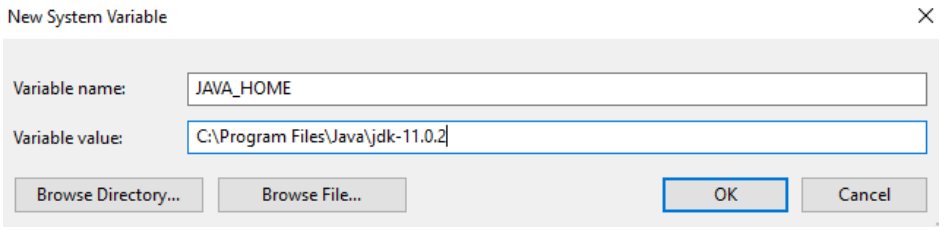




In the “System Properties” dialog, click “Environment Variables…”. In the “Environment Variables” dialog, we want to edit the system variables.



Here, we want to create new variables for Java, Spark, Hadoop and optionally for Maven. Click the “New…” option. For Java, the variable will be named “JAVA_HOME” and the value will be the path to the directory where Java is installed.



For Spark, Hadoop and Maven, the variables must be named “SPARK_HOME”, “HADOOP_HOME” and “MAVEN_HOME” and the values must be the paths to the directories where Spark and Maven are installed.

Now we have defined the locations where Java, Spark, Hadoop and Maven are installed in terms of a few variables, but we have not yet informed the operating systems where the binaries (i.e., actual programs like java, spark-submit and mvn) which we can run from a terminal are located.

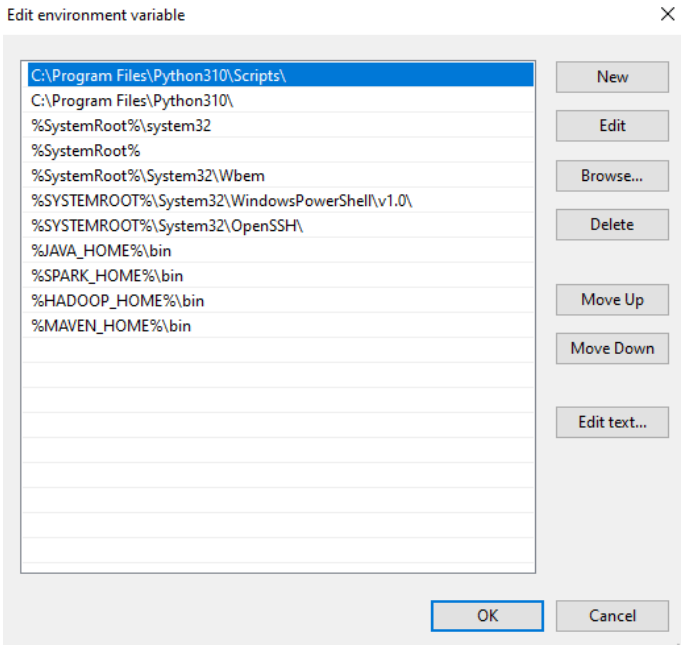
To do this, we will edit the Path environment variable. Select the Path variable from the system variables and click “Edit…”.

This brings up a dialog that allows you to add, edit and delete rows that together make up the Path variable. **Do not edit or delete any of the existing rows.** We only want to add a new row for Java, Spark, Hadoop and optionally Maven.

Click the “New” button and add “%JAVA_HOME%\bin” as a new row. The % notation indicates a reference to the variable JAVA_HOME we created earlier. By extending the path to the “\bin” directory and adding this to the Path variable, we are informing the operating system where the Java binaries are.

Repeat the procedure for the other variables SPARK_HOME, HADOOP_HOME and optionally MAVEN_HOME. All should be referenced inside % symbols and extended with \bin.

The resulting Path variable should look similar to the picture below.



Testing

To test the successful installation of various components, you can issue a few simple commands from a terminal like Command Prompt or Powershell.

Java

To test the Java installation, issue the command “java -version”. The output should look as follows.

```
C:\Users\Student>java -version
openjdk version "11.0.2" 2019-01-15
OpenJDK Runtime Environment 18.9 (build 11.0.2+9)
OpenJDK 64-Bit Server VM 18.9 (build 11.0.2+9, mixed mode)

C:\Users\Student>
```

Python

To test the Python installation, issue the command “python”. The output should look as follows.

```
C:\Users\Student>python
Python 3.10.2 (tags/v3.10.2:a58ebcc, Jan 17 2022, 14:12:15) [MSC v.1929
64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Maven

To test the Maven installation, navigate to a Maven project (for example, the Java template for the team project) inside a terminal (using the cd command, or by starting the terminal in the project’s directory) and issue the command “mvn clean package”. The end of the output should look as follows.

```
[INFO] --- maven-jar-plugin:2.4:jar (default-jar) @ 2ID70-2022-MS2-Template ---
[INFO] Building jar: C:\Code\2ID70-2022-MS2-Java-Template\target\app.jar
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  8.672 s
[INFO] Finished at: 2022-03-03T15:58:26+01:00
[INFO] ------------------------------------------------------------------------

C:\Code\2ID70-2022-MS2-Java-Template>
```

Spark

To test the Spark installation, you will need a Java or Python application to run on Spark. Assuming you have such an application, in the form of a .jar or .py file, you can place the application in a suitable directory alongside any data files your application might need. A suitable directory is again a directory that is not directly or indirectly inside a user directory.

From a terminal in the directory where the application and optional data are located, issue the “spark-submit application.jar” command, where application.jar is the name (including extension) of your application.

Note: the “spark-shell” command which allows you to run Spark interactively, might not work. This is not a problem for this assignment, using only “spark-submit” will suffice.

Troubleshooting

Some common problems related to Spark are listed below and possible solutions are provided for each.

Unrecognized command

Your terminal might indicate that is a command (such as java or mvn) is not a recognized command.

This is most likely to be caused by a problem in the environment variable configuration. See the corresponding section in this guide.

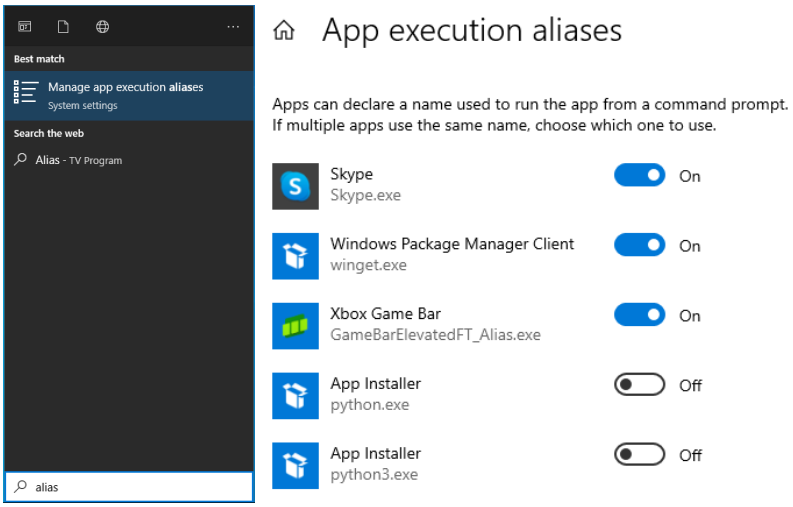
Spark-submit crashes with ExitCodeException

In case spark-submit crashes quickly after starting with a Java ExitCodeException, a possible solution is to install the Microsoft Visual C++ Redistributable Package found here.

Spark cannot find Python

It is possible that, even though Python was installed for all users and added to the Path environment variable by the installer, Spark complains that it cannot find Python (or python3, specifically).

This is most likely to be caused by an issue with program aliases in Windows 10 or above. A workaround it to open “App execution aliases” and disable the entries for “python.exe” and “python3.exe”.



Then, go into the directory into which Python was installed (typically C:\Program Files\Python310) and in case there is no file called “python3.exe” there, simply copy the file “python.exe”, paste it into the same directory and rename it to “python3.exe”.