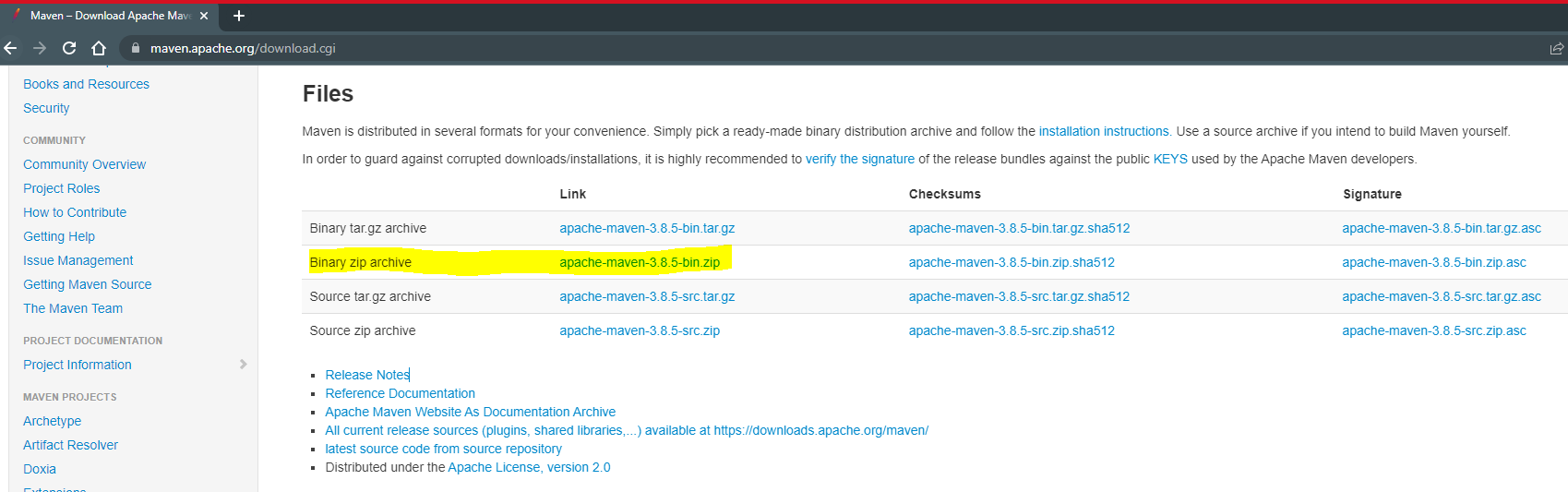
**Maven**

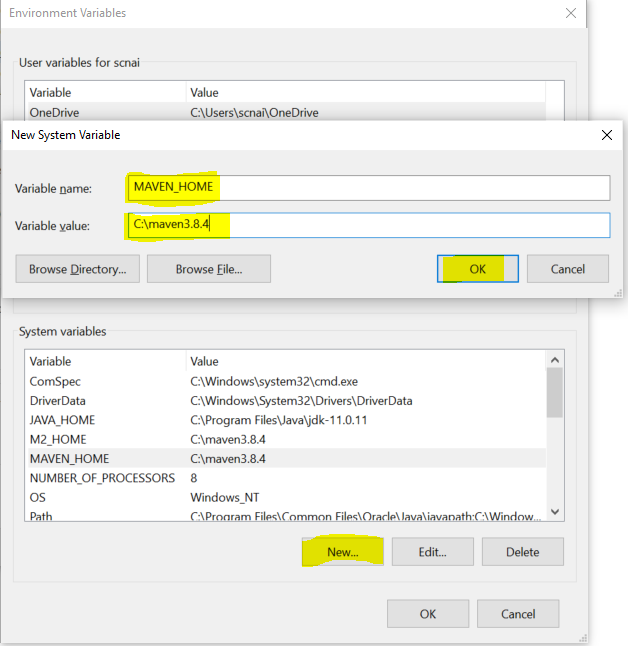
1. Maven is a build tool.
2. Gradle is another tool which is same as a maven and which is also a build tool.
3. Maven is use in the development process starting from the creating application till the execution of the application.
4. Maven **Archetype** is use to get the project structure of the application. It will provide you a project template.
5. Maven helps you to compiles the code (mvn compile).
6. Maven helps you to execute the unit test case of the code (mvn test).
7. Maven helps you to creating the project bundle (.jar, .war) (mvn package).
8. You can also run the project using Maven (mvn deploy)
9. Maven also helps to manage the dependencies (the jar files).

**Maven Setup**

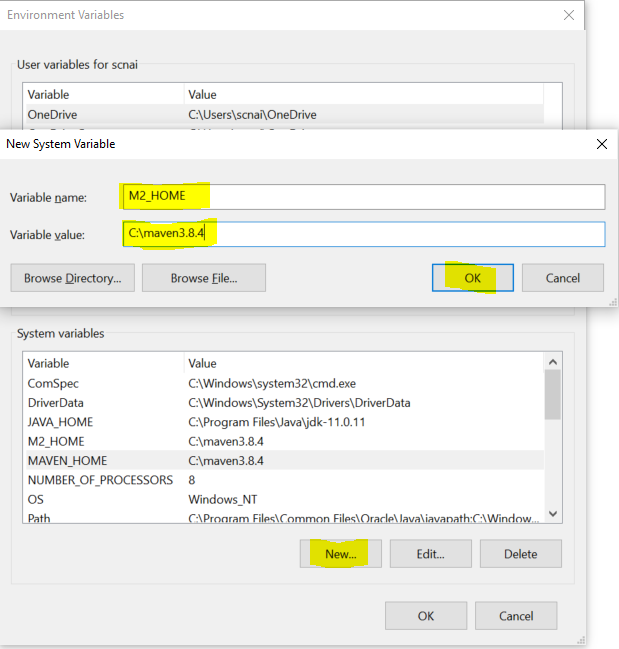
1. Download Maven Zip file (<https://maven.apache.org/download.cgi>)



1. Extract in into a specific location (prefer C dive)
2. Setting the environment variable for Maven
   1. **MAVEN\_HOME**

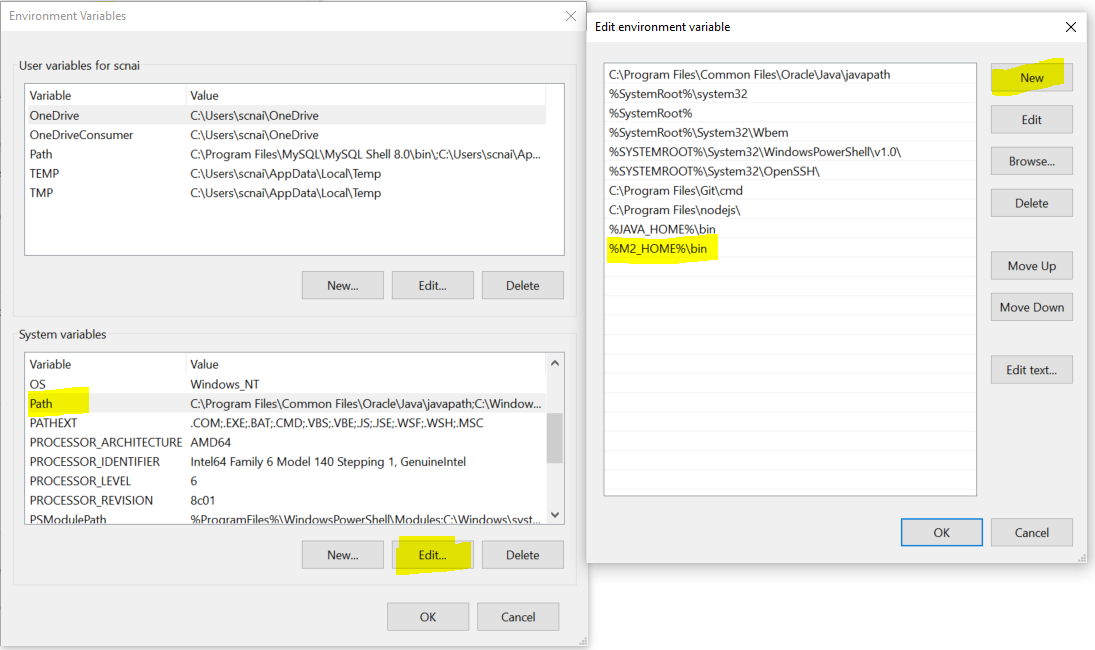


* 1. **M2\_HOME**



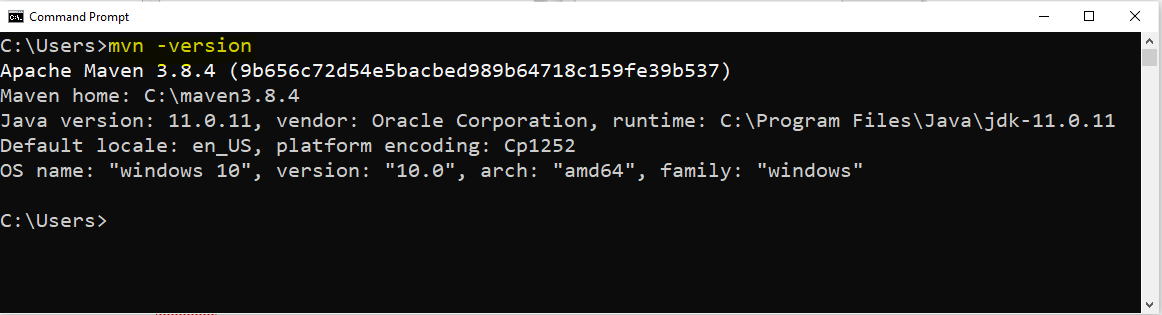
* 1. **Path**

Edit the existing Path Variable and add new value **“%M2\_HOME%\bin”**



1. **Verify Maven Setup**
   1. Open CMD
   2. Execute following command

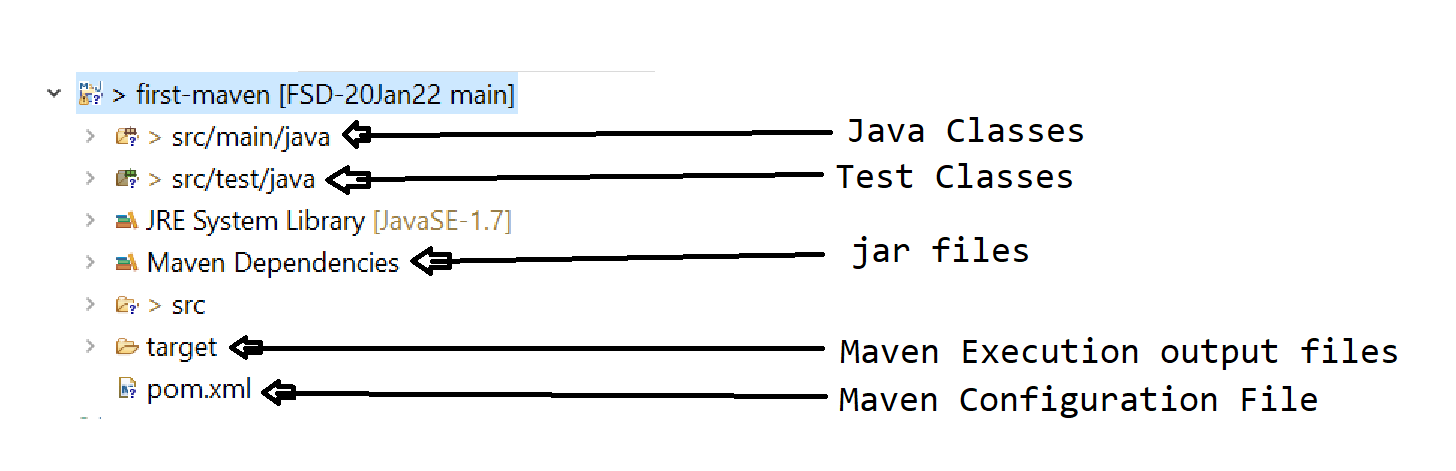
“**mvn -version**”



**Create Maven Project using IDE (Eclipse)**

1. File Menu -> “New” Option -> “Maven Project” option
2. Click on the “Next” button
3. Select an Archetype (Project Structure)
   1. There are Apache maven official and third party archetype available.
   2. You can use the official archetype provided by Apache by filter “org.apache.maven”
   3. **maven-archetype-quickstart**: You can create a Core java application
   4. **maven-archetype-webapp**: You can create a java web application
4. set Group Id (Package structure)
5. Artifact ID (Project Name)
6. Set the version
7. Click on Finish

**Maven Project Structure**



**Update Maven Project forcefully**

Follow the septs only if you are not able to get the changes after modifying the pom.xml or getting error on the project

1. Right click on Project
2. Go to “Maven” -> click on “Update Project”
3. Select the check box in the new window (Force Update of Snapshot/Release)
4. Click on “Ok” button

**Pom.xml file**

1. POM stands for Project Object Model
2. This is the maven configuration file
3. In this file maven setting, configurations and dependency management can done.
4. The basic configurations are as follows
   1. You can configure a groupId, artifactId, Version
   2. Using Properties tag you can set the java version or dependencies version.
   3. Dependencies tag
      1. This tag is use to manage the dependencies (jar files) into a maven project
      2. Can add, remove or update the version of the jar file using this tag
5. Maven has make all jar files available into a central repository.

<https://mvnrepository.com/>

**Dependency Management by Maven**



**Maven Life Cycle Stages**

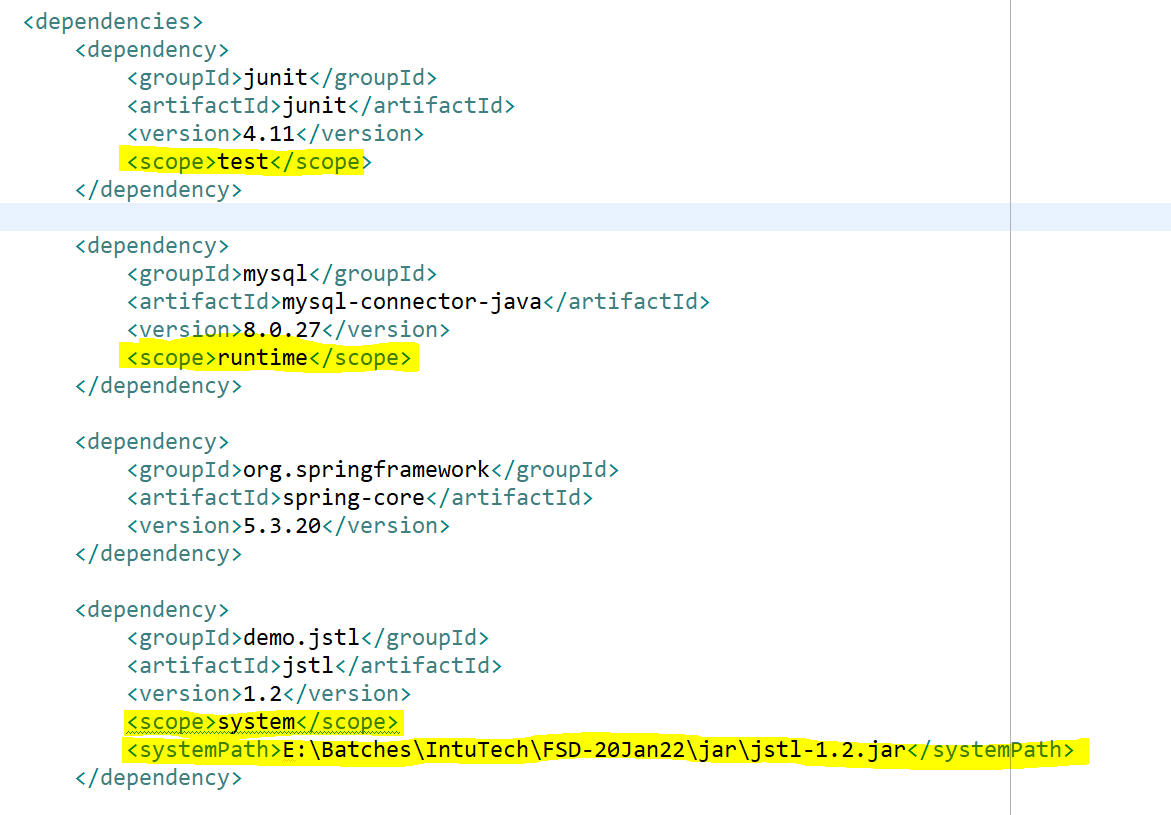
1. **Clean**: is use to clean the previous result executed by Maven.
2. **Validate**: This step is use to check the correctness, syntax and dependency of the project.
3. **Compile**: the java classes will be compile in this stage.
4. **Test**: The test cases will be executed in this stage
5. **Package**: In this stage the code will be packaged inside a jar or war files.
6. **Verify**: it will verify the project package whether it is correctly created or not.
7. **Install**: Prepare a package for the execution.
8. **Deploy**: Execute the application.

**Maven Scope**

This is the Dependencies (jar file) scope. Using scope you can specify till which maven lifecycle you need the specific dependency(jar).

There are 5 types of maven scope

1. **Compile:** the dependency added using this scope is available only till the compilation stage
2. **Runtime**: is a default scope if it is not explicitly set. These dependencies available in all the stage like code compilation and execution.
3. **Provided**: the dependencies with this scope will be search internally inside the project in jre or inside server.
4. **Test**: the dependencies available till the test stage of the application
5. **System**: the dependencies are present inside a system but not inside the project, so you can specify a path of your dependency manually.



**Project Versioning**

Version are Given in the following Format

**000.000.000**

Example :

Eclipse: 4.42.0

Java: 11.0.11

Major Release: First digits

Minor Release: Middle Digits

Bug/Defect fixes release**:** last digits

**Git and GitHub**

1. Git is a desktop tool which is use to maintain the files and its version etc. on local.
2. GitHub is a web application which is use to maintain the files and its version etc. on cloud.

Git Setup

1. Git has to install in local system
2. Download Git from : <https://git-scm.com/downloads>
3. Install Git on local system. <https://phoenixnap.com/kb/how-to-install-git-windows>
4. Verify Installation
   1. Right Click on any folder and check if you are getting **Git Batch** and **Git GUI** options.

GitHub

1. Create a Free Account on GitHub.
2. <https://github.com/signup>