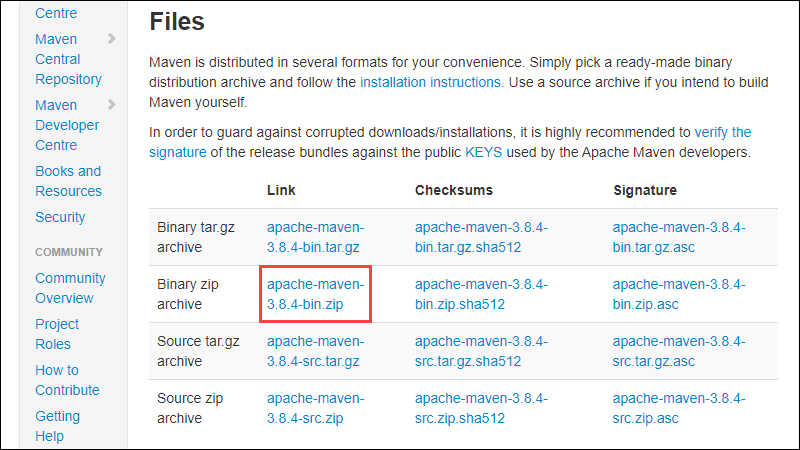
**How to Install Maven on Windows?**

Follow the steps outlined below to install Apache Maven on Windows.

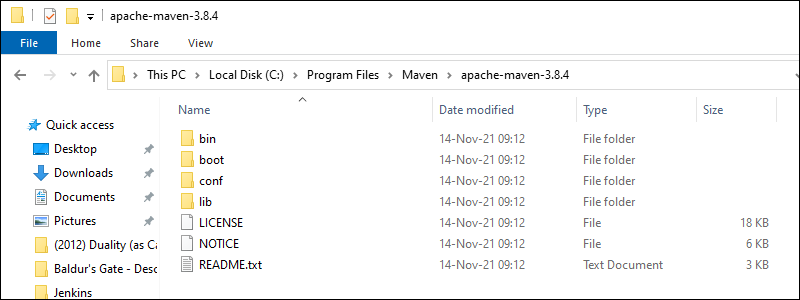
**Step 1: Download Maven Zip File and Extract**

1. Visit the [Maven download page](https://maven.apache.org/download.cgi) and download the version of Maven you want to install. The *Files* section contains the archives of the latest version. Access earlier versions using the archives link in the *Previous Releases* section.

2. Click on the appropriate link to download the binary zip archive of the latest version of Maven. As of the time of writing this tutorial, that is version 3.8.4.



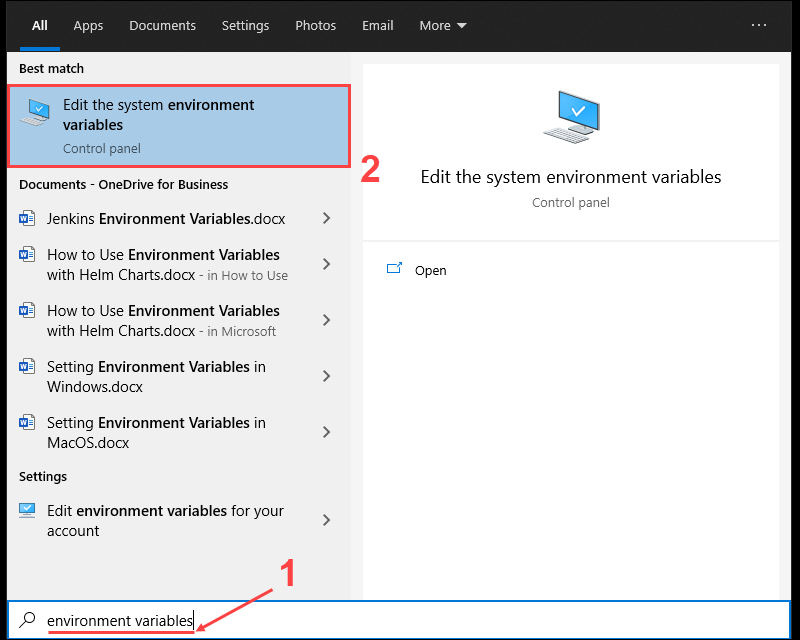
3. Since there is no installation process, extract the Maven archive to a directory of your choice once the download is complete. For this tutorial, we are using *C:\Program Files\Maven\apache-maven-3.8.4*.



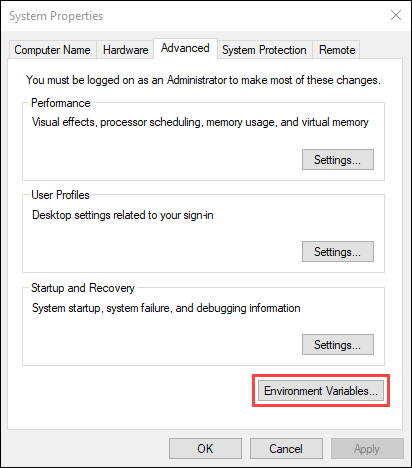
**Step 2: Add MAVEN\_HOME System Variable**

1. Open the Start menu and search for [environment variables](https://phoenixnap.com/kb/windows-set-environment-variable).

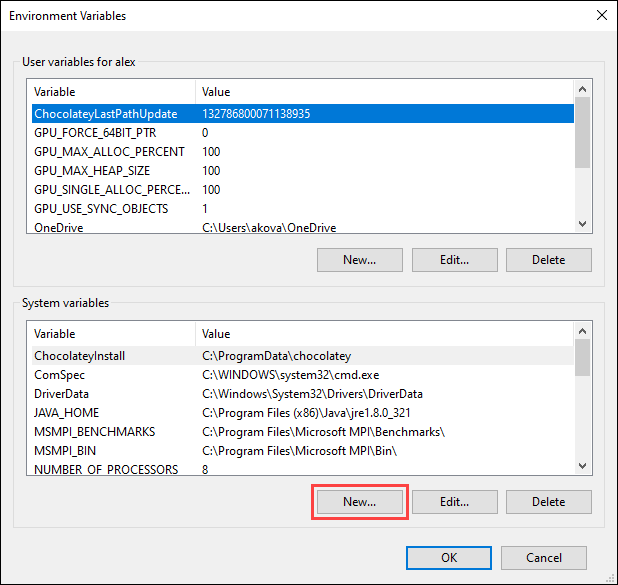
2. Click the **Edit the system environment variables** result.



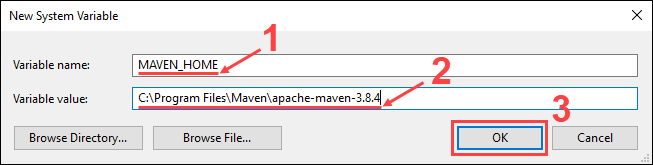
3. Under the *Advanced tab* in the *System Properties* window, click **Environment Variables**.



4. Click the **New** button under the *System variables* section to add a new system environment variable.

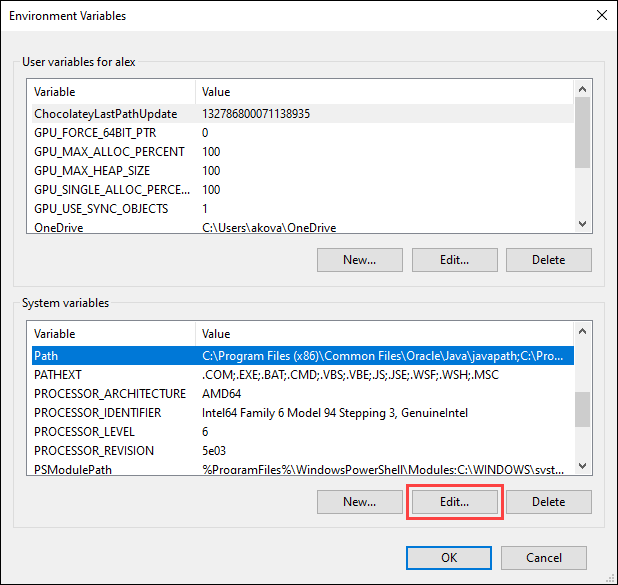


5. Enter **MAVEN\_HOME** as the variable name and the path to the Maven directory as the variable value. Click **OK** to save the new system variable.

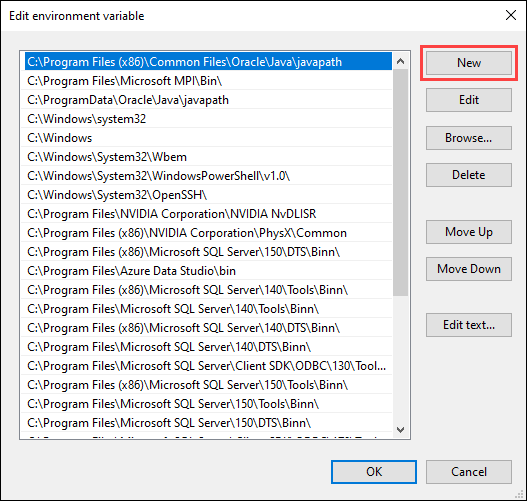


**Step 3: Add MAVEN\_HOME Directory in PATH Variable**

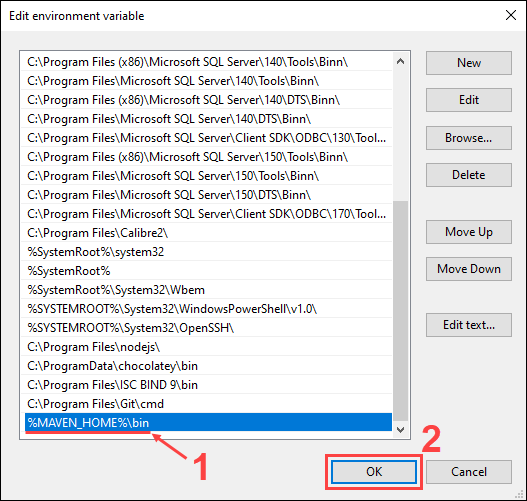
1. Select the **Path** variable under the *System variables* section in the *Environment Variables* window. Click the **Edit** button to edit the variable.



2. Click the **New** button in the *Edit environment variable* window.

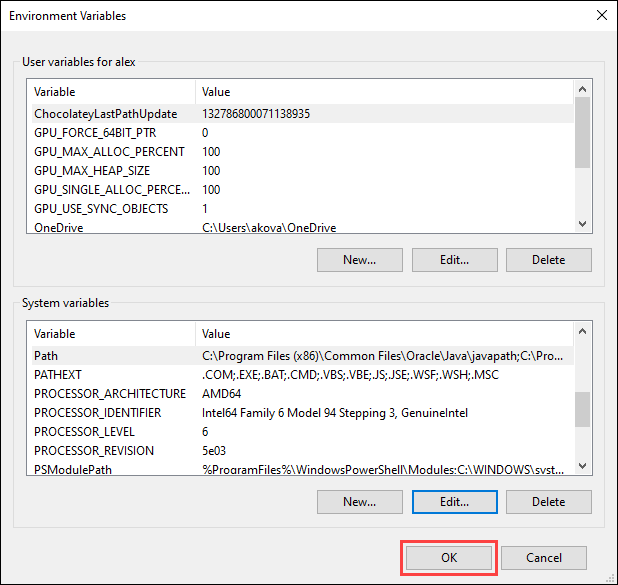


3. Enter **%MAVEN\_HOME%\bin** in the new field. Click **OK** to save changes to the **Path** variable.



**Note:** Not adding the path to the Maven home directory to the **Path** variable causes the **'mvn' is not recognized as an internal or external command, operable program or batch file** error when using the **mvn** command.

4. Click **OK** in the *Environment Variables* window to save the changes to the system variables.



**Step 4: Verify Maven Installation**

In the command prompt, use the following command to verify the installation by checking the current version of Maven:

mvn -version

## Creating a Project (Hello World)

You need somewhere for your project to reside. Create a directory somewhere and start a shell in that directory. On your command line, execute the following Maven goal:

mvn archetype:generate -DgroupId=com.MSRIT.app -DartifactId=Abhishek -DarchetypeArtifactId=maven-archetype-quickstart -DarchetypeVersion=1.4 -DinteractiveMode=false

You will notice that the *generate* goal created a directory with the same name given as the artifactId. Change into that directory.

cd Abhishek

The src/main/java directory contains the project source code, the src/test/java directory contains the test source, and the pom.xml file is the project's Project Object Model, or POM.

### The POM

The pom.xml file is the core of a project's configuration in Maven. It is a single configuration file that contains the majority of information required to build a project in just the way you want. The POM is huge and can be daunting in its complexity, but it is not necessary to understand all of the intricacies just yet to use it effectively. This project's POM is:

### What did I just do?

You executed the Maven goal *archetype:generate*, and passed in various parameters to that goal. The prefix *archetype* is the [plugin](https://maven.apache.org/plugins/index.html) that provides the goal. If you are familiar with [Ant](http://ant.apache.org/), you may conceive of this as similar to a task. This *archetype:generate* goal created a simple project based upon a [maven-archetype-quickstart](https://maven.apache.org/archetypes/maven-archetype-quickstart/) archetype. Suffice it to say for now that a *plugin* is a collection of *goals* with a general common purpose. For example the jboss-maven-plugin, whose purpose is "deal with various jboss items".

### Build the Project

mvn package

The command line will print out various actions, and end with the following:

...

[INFO] ------------------------------------------------------------------------

[INFO] BUILD SUCCESS

[INFO] ------------------------------------------------------------------------

[INFO] Total time: 2.953 s

[INFO] Finished at: 2019-11-24T13:05:10+01:00

[INFO] ------------------------------------------------------------------------

You may test the newly compiled and packaged JAR with the following command:

java -cp target/Abhishek-1.0-SNAPSHOT.jar com.MSRIT.app.App

Which will print the quintessential:

Hello World!

---------------------------------------------------------------------------------------------------------------------------------

An interesting thing to note is that phases and goals may be executed in sequence.

mvn clean dependency:copy-dependencies package

This command will clean the project, copy dependencies, and package the project (executing all phases up to *package*, of course).

### Generating the Site

mvn site

This phase generates a site based upon information on the project's pom. You can look at the documentation generated under target/site.