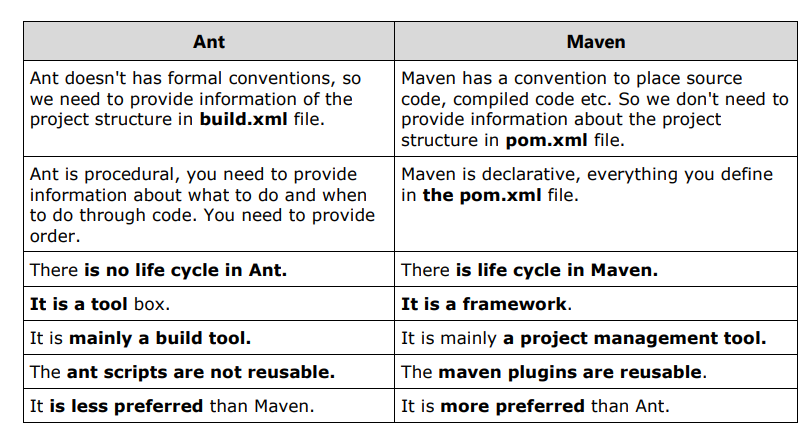
=========================MAVEN================================

Maven is basically a project management or build tool which can easily handle the entire lifecycle of a project that include code generation, compilation, testing, validation, packaging, installation and much more. It works in phases rather than tasks and makes the build management process much easier.

A [Selenium](https://www.simplilearn.com/tutorials/selenium-tutorial/selenium-ide) project for [test automation](https://www.simplilearn.com/what-is-automation-testing-article) requires you to have all the dependencies associated with it. These dependencies can be downloaded and upgraded manually. However, as the project gets bigger, it can be quite challenging to manage these dependencies. This is why you need to have build automation tools to handle them automatically.

Maven uses the concepts of the project object model (POM) and enables the user to cut down several steps followed in the build process.

* Maven is used to manage the dependencies. For example, if you are using selenium version 2.35 and any later point of time you have to use some other version, the same can be managed easily by Maven.
* It works very effectively when there is a huge number of Jar files with different versions.
* A Maven repository is a directory of packaged JAR file with **pom.xml** file. Maven searches for dependencies in the repositories.



* There are 3 types of Maven repository:-

**1) Local Repository :** Maven local repository is located in your local system. It is created by the Maven when you run any Maven command. By default, Maven local repository is **/home/luser/.m2** directory on Linux OS.

**2) Central Repository :** Maven central repository is located on the web. It has been created by the Apache Maven community itself.

**3) Remote Repository** : Maven **remote repository** is located on the web. Most of libraries can be missing from the central repository such as **JBoss** library etc, so we need to define remote repository in **pom.xml** file.

Group ID : Organization name

**For example:** org.apache.maven.plugins is the designated Group-Id for all Maven plugins.

Artifact ID : Project name; The artifact for a project is typically a JAR file.

**For example:** A typical artifact produced by Maven would have the form:

<artifactId>-<version>.<extension> (for example, myapp-1.0.jar)

**Version:**  This element indicates the version of the artifact generated by the project. Maven goes a long way to help you with version management and you will often see the SNAPSHOT designator in a version.

**For example:** Version 1.0-SNAPSHOT is released as version 1.0, and the new development version is version 1.1-SNAPSHOT.

POM

* is the fundamental unit of work in Maven.
* It is an XML file that contains information about the project and configuration details used by Maven to build the project.
* Some of the configuration that can be specified in the POM are the project dependencies, the plugins(for config like complier, surefire for execution) or goals that can be executed, the build profiles, and so on. Other information such as the project version, description, developers, mailing lists and such can also be specified.

Dependencies

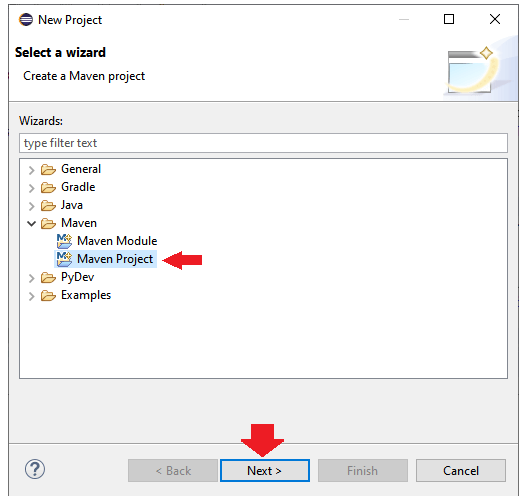
* When working on project, the external jars that we import for multiple functions are known as **project dependencies.**
* Example:- In eclipse to execute the selenium web driver program we need selenium jars.
* Maven has built-in dependency management. We can just specify in the POM file what external libraries the project depends on, and which version.
* Maven would download them and puts them in your local Maven repository.
* If these external libraries need other libraries, then Maven would download their dependency libraries into your local Maven repository as well.
* We can specify dependencies inside the dependencies element in the POM file.

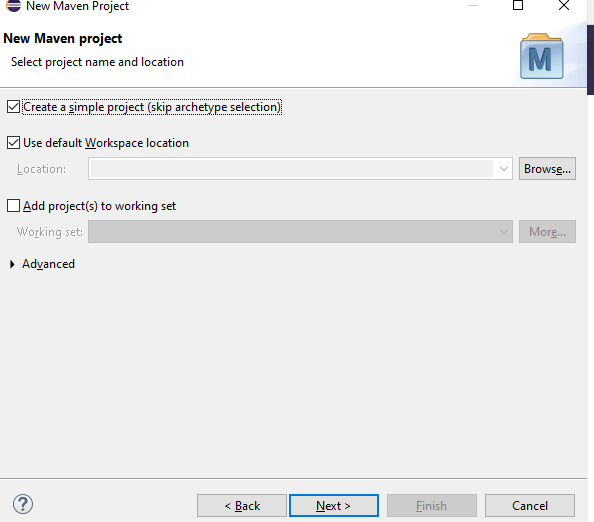


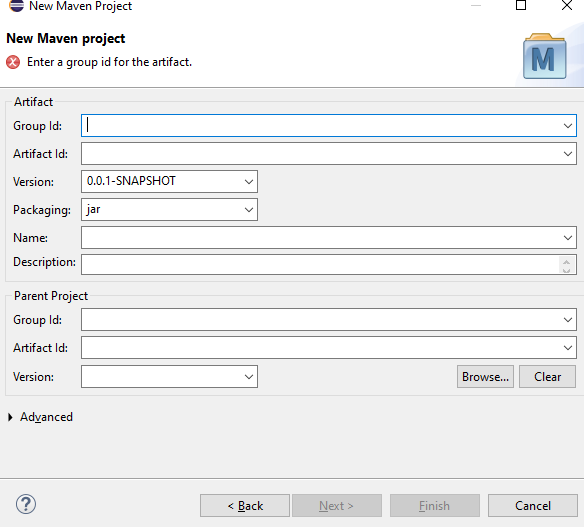
## Maven Eclipse plug-in

Maven plug-in is automatically installed in the Eclipse, but if it is not present, we will go to the **Eclipse Market Place** and search for **Maven** and download the **M2E integrated version** from there.

Create maven project

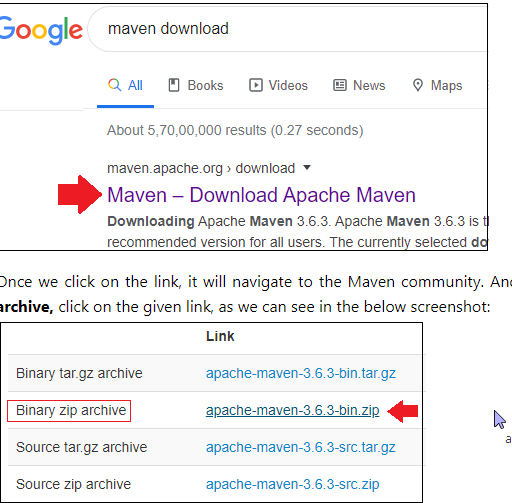






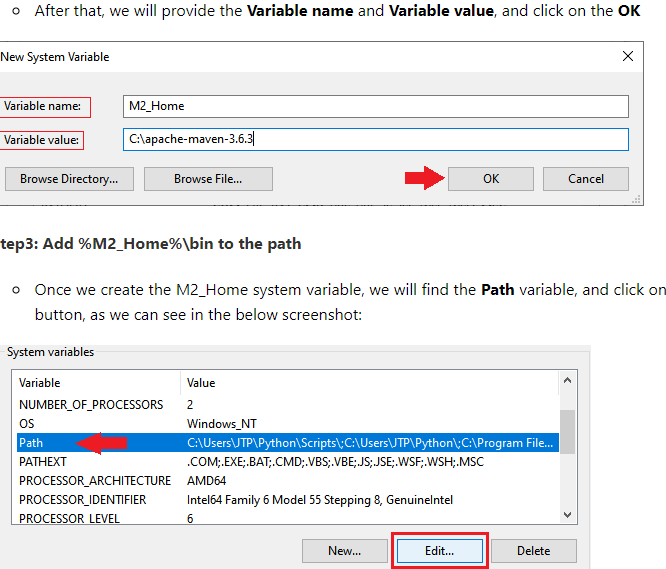
## Maven command line plug-in

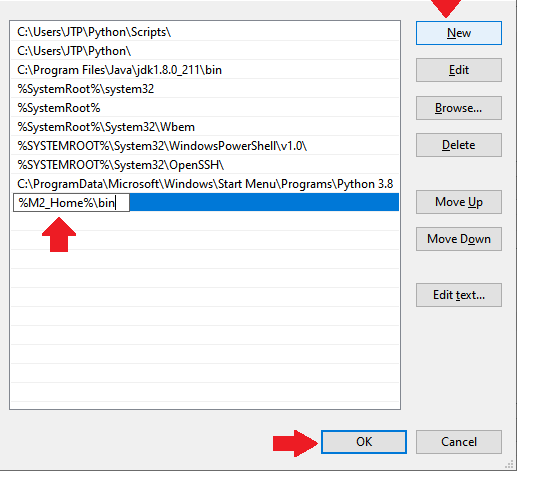
#### # Note : **Before we will start the installation process of Maven, we will make sure that Java is installed and the environment variable is also set.**

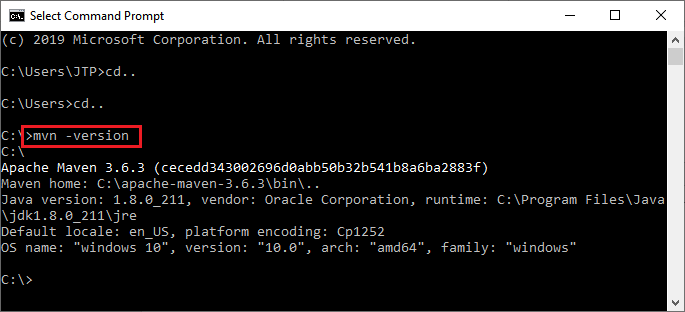


**Step2: Add M2\_Home in the System Variable**

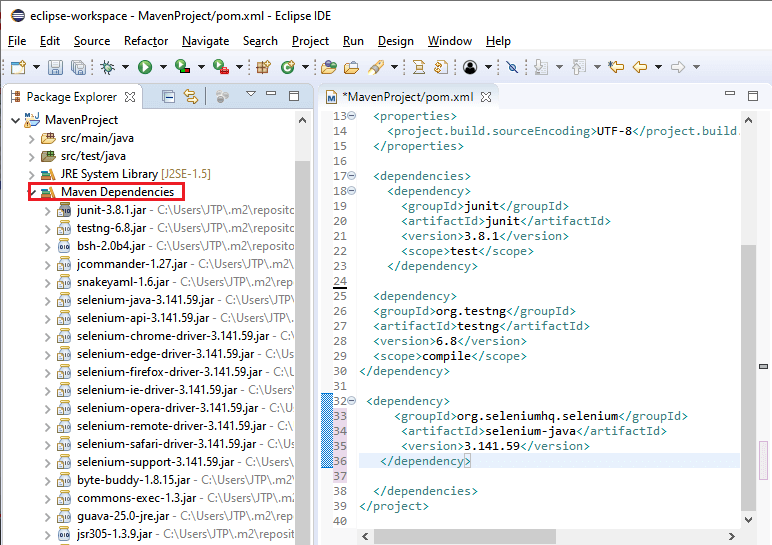
* Once we copy the location, we will go to the **Environment Variables** window and click on the **New** button on the **System variable** section, as we can see in the below image:



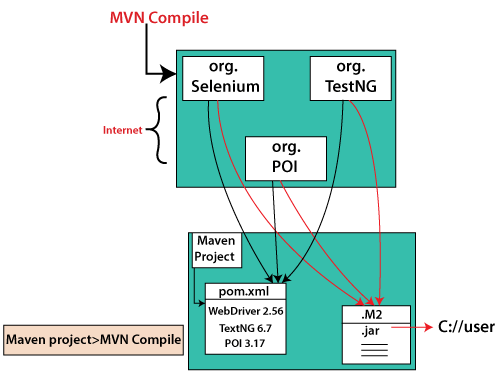




After adding dependency in pom.xml file, the jar files are added in maven dependencies folder and also in .m2 repository in system folder



The **Maven compiler plugin** is used to compile the source code of a Maven project. Maven test command will connect to the internet and download all the dependency jar into the .M2 folder local repository and then compile the entire selenium source code



The **Maven surefire plugin** is used when we have to run the unit tests of the application.

The **Maven source plugin** is used to build the jars files that were having the **.java** source

the most important directories:

- src

- main

- java

- resources

- webapp

- test

- java

- resources

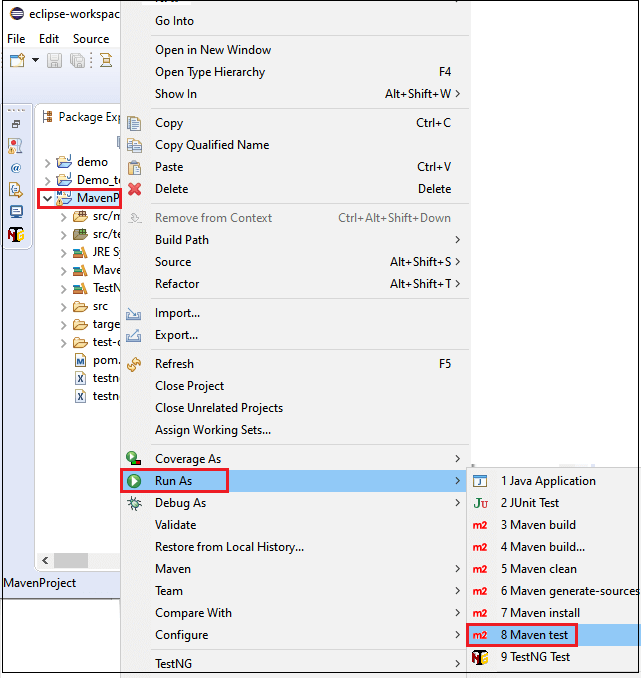
- target

The src directory is the root directory of your source code and test code. The main directory is the root directory for source code related to the application itself (not test code). The test directory contains the test source code. The java directories under main and test contains the Java code for the application itself (under main) and the Java code for the tests (under test).

The resources directory contains other resources needed by your project. This could be property files used for internationalization of an application, or something else.

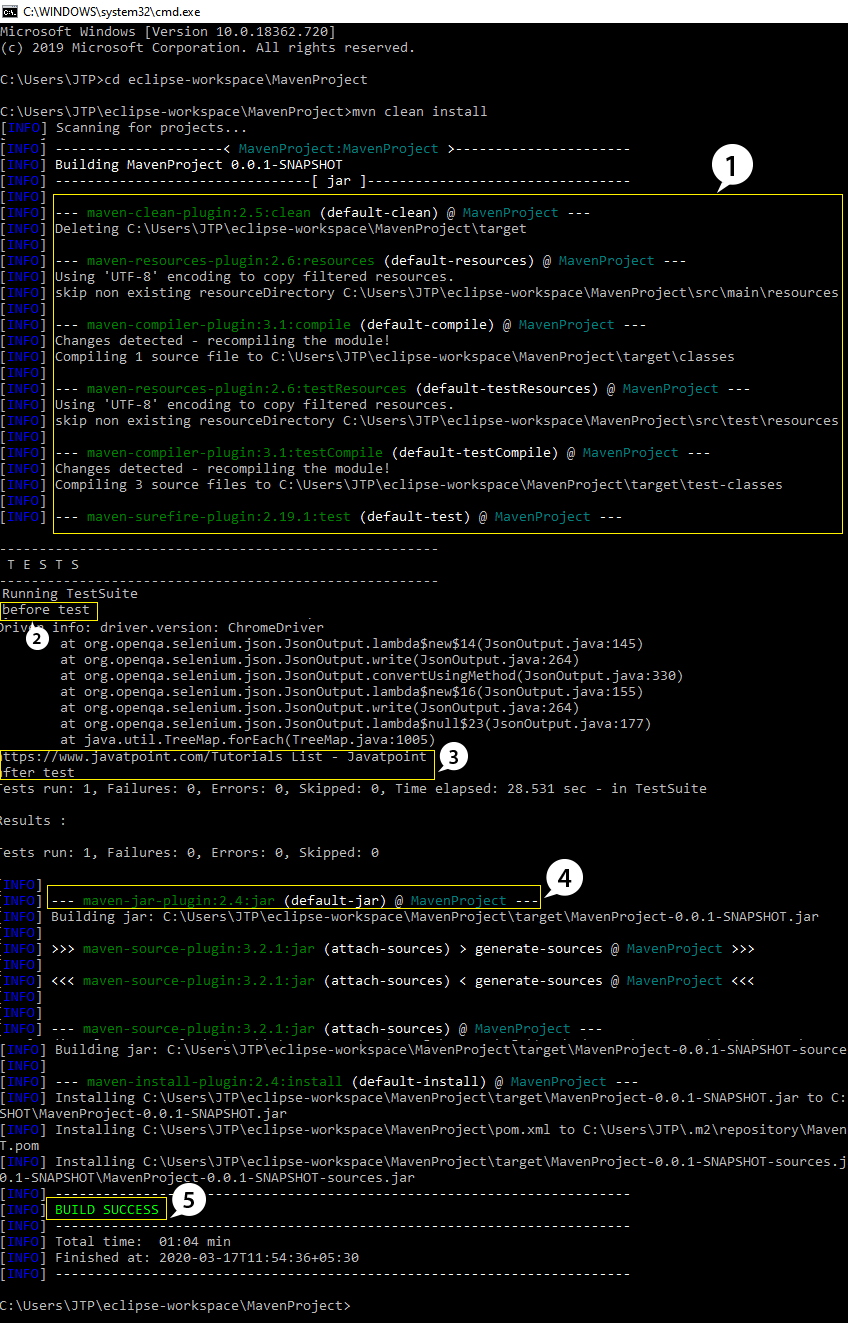
The webapp directory contains your Java web application, if your project is a web application. The webapp directory will then be the root directory of the web application. Thus the webapp directory contains the WEB-INF directory etc.

The target directory is created by Maven. It contains all the compiled classes, JAR files etc. produced by Maven. When executing the clean build phase, it is the target directory which is cleaned.



Run through cmd

tcopy the location of the **MavenProject** from properties optionand paste in the command prompt to reach the correct location of the file.



**Maven build lifecycle**

Maven is based around the central concept of a build lifecycle. What this means is that the process for building and distributing a particular artifact (project) is clearly defined.

Here are three built-in build lifecycles: default, clean and site. The default lifecycle handles your project deployment, the clean lifecycle handles project cleaning, while the site lifecycle handles the creation of your project's site documentation.

For example, the default lifecycle comprises of the following phases (refer to the Lifecycle Reference)

• validate - validate the project is correct and all necessary information is available

• compile - compile the source code of the project

• test - test the compiled source code using a suitable unit testing framework. These tests should not require the code be packaged or deployed

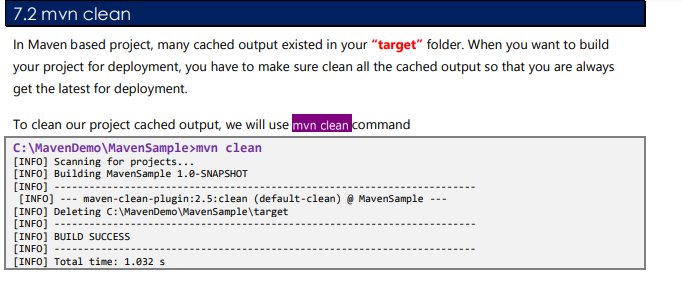
• package - take the compiled code and package it in its distributable format, such as a JAR.

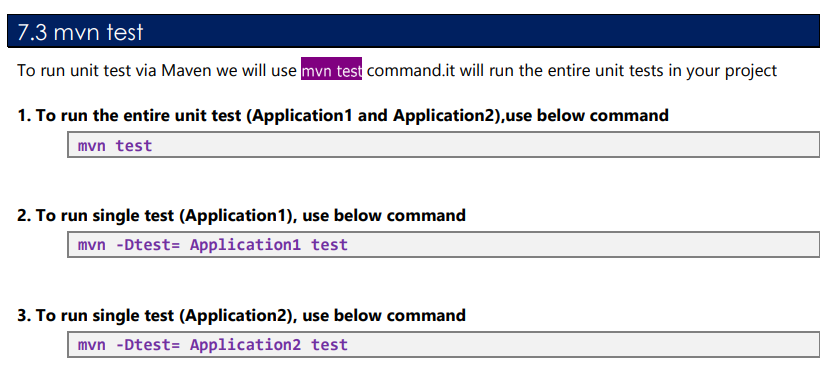
• verify - run any checks on results of integration tests to ensure quality criteria are met

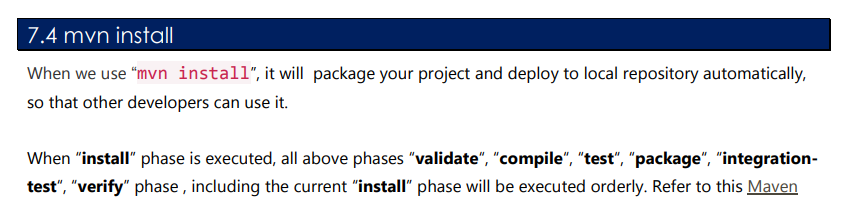
• install - install the package into the local repository, for use as a dependency in other projects locally

• deploy - done in the build environment, copies the final package to the remote repository for sharing with other developers and projects.

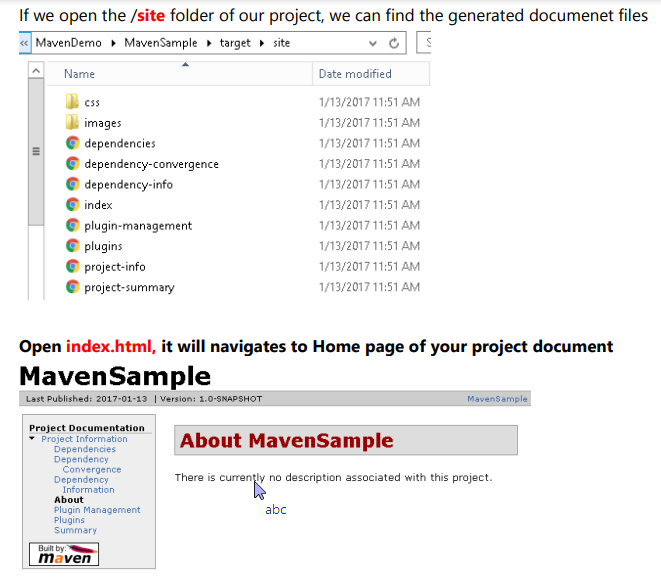
when the “package” phase is executed, all its above phases – “validate“, “compile” and “test“, including the current phase “package” will be executed orderly











What is WAR File?

The WAR file (Web Application Resource or Web Application ARchive) is a container for JAR files, JavaServer Pages, Java Servlets, Java classes, XML files, tag libraries, static sites (HTML and associated files), and other resources that make up an online application

Jenkins

Continuous Integration (CI) is a development practice in which the developers are needs to commit changes to the source code in a shared repository at regular intervals. Every commit made in the repository is then built and further this built is tested to find any kind of errors in it. The general practice is that whenever a code commit occurs, a build should be triggered. This entire process is automated thereby allowing continuous Integration

DevOps team : main purpose integration, create build(Maven s/w), Test the build with using test scripts(Jenkins s/w), Perform sanity(to check main functionalities working fine or not) and smoke testing, after build successful will get mail to both QA and Developer and final build will be ready in Jenkins itself.

Installation process

Step 1: Install java 11

Go to oracle website : <https://www.oracle.com/in/java/technologies/downloads/#java11>

Select the .exe file, enter oracle username and password to download exe file

Username :anirudhamumbarkar@gmail.com Password :Anirudha@777

To install jdk 🡪 double click on downloaded exe file (After complete installation java folder will create in c drive program files folder)

To verify the java version, open cmd java -version

Step2 : <https://www.jenkins.io/download/>

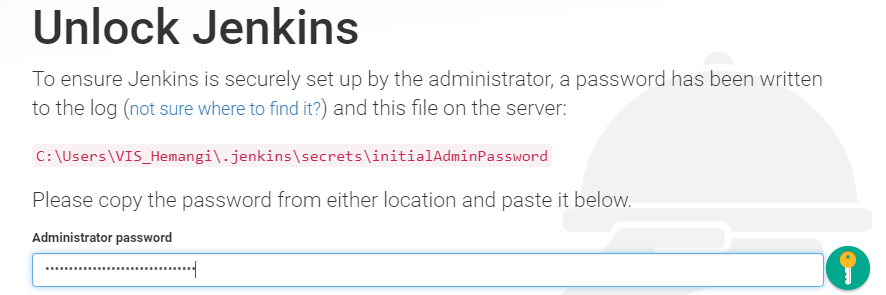


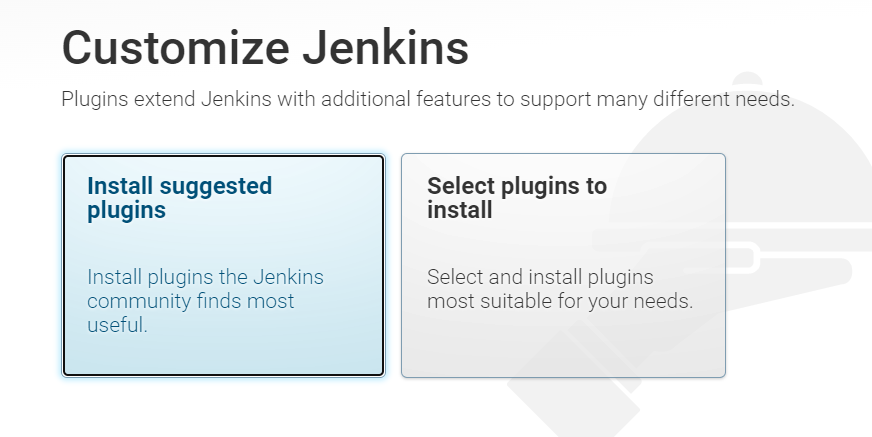
Create folder in c drive and place this war file in that

Open up a terminal/command prompt window to the download directory

Run the command java -jar jenkins.war

Browse to <http://localhost:8080> ; Enter password(.Jenkins root folder in user> Secret > Initial password)





Select plugins to install and continue; Set username and password

Step 3 : Manage jenkin > global config > set path for jdk, Git and maven thru env variables

Step 4: For assigning job to jenkin > New item > enter item name > Free style prj > ok

Step 5: under general > click advance button > click use custom workspace > copy n paste ur project workspace path in directory

Step 6: under build > add build step > execute windows batch command > in command section u can write any maven cmnds eg mvn test > save > build now

Step 7: To execute test cases periodically after specific time

* + click prj > configure > buid trigger > build periodically > Schedule (clk help ? to chk how to set schedule)
  + use \*\*\*\*\* to set periodic time as per ur requirement
  + to stop this periodic execution untic build periodically chkbox and save

Operation:

DevOps team takes diff. codes from diff. developers which are placed in common shared repository and create an integrated build

They will take automated scripts / test cases from QA team and will execute to cases against that build

perform basic sanity and smoke testing and then after success the build is given to QA team for further testing

Again, after making the changes in the scripts (QA) and codes (dev) that will be uploaded to the common repo. and again, devops will check n so on.... this is referred as CI

Devops uses Maven to create the build and to run it Jenkin is used. QA team use limited features of MAVEN like pom.xml, and some other test related commands. But devops uses max. features of MAVEN.

Execution in Jenkin is headless ie no UI and so it is faster.

QA download the build from Jenkin into their env. For example :

* Dev will work 9am to 5pm > Devops will build n test in eve. > Then next day QA will work on that build > and at the same time dev will work further and at eod will store their codes in common repo n so on...referred as CI
* QA team don’t us MAVEN n JENKIN 100p, but as DevOps team uses our automated scripts using mostly Maven n Jenkin, so some knowledge of maven n Jenkin is must
* So, QA team must do certain configuration on Jenkins related to their scripts n verify that they r working fine on Jenkin for the first-time n then later on devops team will handle.

QA don’t install Jenkin in their env. Devops will give Jenkin Server Url to QA; Qa will only configure their test cases n all in Jenkins.

So Devops team basically helps to integrate all the process which in turn enhance the agile process fast.

===============================================================

GIT, GITHUB

we can also perform same operations on other operating systems like **Linux (Ubuntu)** and **MacOS**.

Git 🡺 Local repository, Windows based application;

developed to manage projects with high speed and efficiency. The version control system allows us to monitor and work together with our team members at the same workspace.



**Branching and merging** are the **great feature**s of Git, which makes it different from the other SCM tools. Git allows the **creation of multiple branches** without affecting each other.

Data assurance :  It provides a **unique commit ID** to every commit through a **SHA algorithm**. We can **retrieve** and **update** the commit by commit ID

When you make a commit, Git takes changes that are in the staging area and make them as a new commit. We are allowed to add and remove changes from the staging area. The staging area can be considered as a place where Git stores the changes.



**Maintain the clean history:** It fetches the latest commits from the master branch and puts our code on top of that. Thus, it maintains a clean history of the project.



features such as **Diff, Log,** and **Status**, which allows us to track changes so we can **check the status, compare** our files or branches.

facing internet connectivity issues, it will not affect our work. In Git, we can do almost everything locally.

**Why is git needed?**

When a team works on real-life projects, git helps ensure no code conflicts between the developers. Furthermore, the project requirements change often. So a git manages all the versions. If needed, we can also go back to the original code. The concept of branching allows several projects to run in the same codebase.

GitHub 🡺 Remote Repository, Web Application

is a Git repository hosting service. GitHub also facilitates with many of its features, such as access control and collaboration.

Features:

* Collaboration
* Integrated issue and bug tracking
* Graphical representation of branches
* Git repositories hosting
* Project management
* Team management
* Code hosting
* Track and assign tasks
* Conversations

1. Using github the project managers can collaborate, review and guide the developers regarding any changes. This makes project management easy.
2. The github repositories can be made public or private. Thus allowing safety to an organization in case of a project.
3. GitHub has a feature of pull requests and issues in which all the developers can stay on the same page and organize.
4. All the codes and their documentation are in one place in the same repository. Hence it makes easy code hosting.
5. There are some special tools that github uses to identify the vulnerabilities in the code which other softwares do not have. Hence there is safety among the developers from code start till launch.
6. Github is available for mobile and desktops. The UI is so user-friendly that it becomes straightforward to get comfortable with and use it.

Pre-Requisites:

Step 1 : Create new account in Github : <https://github.com/>

Create new repository in Github and capture the link

Step 2 : Download and install the git repository: <https://git-scm.com/downloads>

Commands:

**Setting username**

The username is used by the Git for each commit.

1. $ git config --global user.name "Himanshu Dubey"

**Setting email id**

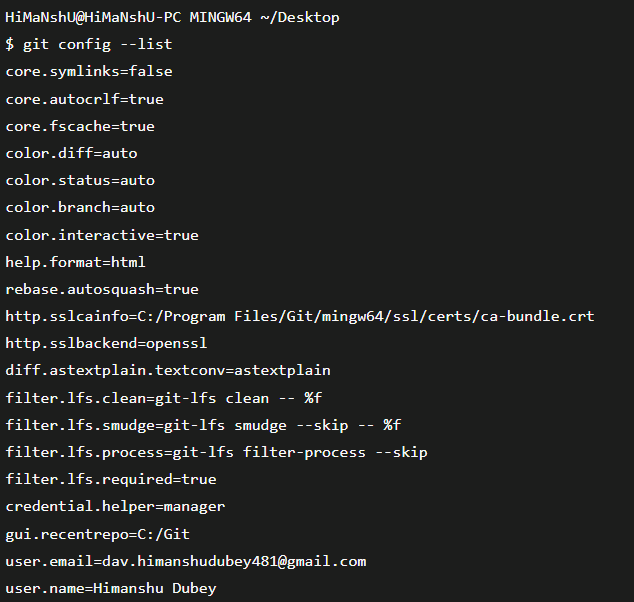
The Git uses this email id for each commit.

1. $ git config --global user.email  "himanshudubey481@gmail.com"

**Checking Your Settings**

You can check your configuration settings; you can use the **git config --list** command to list all the settings that Git can find at that point.

1. $ git config -list



Tools

### GitBash

Git Bash is an application for the Windows environment. It is used as Git command line for windows. Git Bash provides an emulation layer for a Git command-line experience.

### Git GUI

Git GUI is a powerful alternative to Git BASH. It offers a graphical version of the Git command line function, as well as comprehensive visual diff tools. We can access it by simply right click on a folder or location in windows explorer. Also, we can access it through the command line by typing below command.

1. $ git gui

Git Tools

