

# **SOFTWARE MAINTENANCE**

## **ASSIGNMENT**

- ASHISH KUMAR

- 2K18/SE/041

**Ques:** Explain the main steps involved in the following tools-

1. Any 1 Version control tool (GIT, CVS, TFS, etc)
2. Any 1 Build Management tool (Gradle, Maven)

**Note:** Ma'am I have chosen **GIT** as version control tool and **maven** as build management tool because I have used these two tools.

**Ans:**

### **Version Control Tool**

Version control allows you to keep track of your work and helps you to easily explore the changes you have made, be it data, coding scripts, notes, etc. With version control software such as **Git**, version control is much smoother and easier to implement. Using an online platform like **Github** to store your files means that you have an online backup of your work, which is beneficial for both you and your collaborators.

For example: If you are a graphic or web designer and want to keep every version of an image or layout (which you would most certainly want to), a Version Control System (VCS) is a very wise thing to use. It allows you to revert selected files back to a previous state, revert the entire project back to a previous state, compare changes over time, and see who last modified and more.

### **Benefits of using GIT as a version control Tool**

Having a GitHub repo makes it easy for you to keep track of collaborative and personal projects - all files necessary for certain analyses can be held together and people can add in their code, graphs, etc. as the projects develop. Each file on GitHub has a history, making it easy to explore the changes that occurred to it at different time points. You can review other people's code, add comments to certain lines or the overall document, and suggest changes. For collaborative projects, GitHub allows you to assign tasks to different users, making it clear who is responsible for which part of the analysis. You can also ask certain users to review your code. For personal projects, version control allows you to keep track of your work and easily navigate among the many versions of the files you create, whilst also maintaining an online backup.

## What is a repository?

You can think of a repository (*aka* a repo) as a “main folder”, everything associated with a specific project should be kept in a repo for that project. Repos can have folders within them, or just be separate files. The files you put on GitHub will be public (i.e. everyone can see them & suggest changes, but only the people with access to the repository can directly edit and add/remove files). You can also have private repositories on GitHub, which means that only you can see the files.

## How to get started?

### Step 1: Sign up and installation!

Please register on the **Github website**.

If you are on a personal Windows machine, download and install Git for your operating system.

### Install Git on Windows

1. Navigate to the latest [Git for Windows installer](#) and download the latest version.
  2. Once the installer has started, follow the instructions as provided in the **Git Setup** wizard screen until the installation is complete.
  3. Open the windows command prompt (or **Git Bash** if you selected not to use the standard Git Windows Command Prompt during the Git installation).
  4. Type git version to verify Git was installed.
- If git is successfully installed in your computer, then you will see something like this when you type this command in cmd:  
**git**  
**git --version**

```

C:\Users\Ashish>git --version
git version 2.20.1.windows.1

C:\Users\Ashish>git
usage: git [--version] [--help] [-C <path>] [-c <name>=<value>]
        [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
        [-p | --paginate | -P | --no-pager] [--no-replace-objects] [--bare]
        [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
        <command> [<args>]

These are common Git commands used in various situations:

start a working area (see also: git help tutorial)
  clone      Clone a repository into a new directory
  init       Create an empty Git repository or reinitialize an existing one

work on the current change (see also: git help everyday)
  add        Add file contents to the index
  mv         Move or rename a file, a directory, or a symlink
  reset      Reset current HEAD to the specified state
  rm         Remove files from the working tree and from the index

examine the history and state (see also: git help revisions)
  bisect     Use binary search to find the commit that introduced a bug
  grep       Print lines matching a pattern
  log        Show commit logs
  show       Show various types of objects
  status     Show the working tree status

grow, mark and tweak your common history
  branch     List, create, or delete branches
  checkout   Switch branches or restore working tree files
  commit     Record changes to the repository
  diff       Show changes between commits, commit and working tree, etc
  merge      Join two or more development histories together
  rebase     Reapply commits on top of another base tip
  tag        Create, list, delete or verify a tag object signed with GPG

collaborate (see also: git help workflows)
  fetch      Download objects and refs from another repository
  pull       Fetch from and integrate with another repository or a local branch
  push       Update remote refs along with associated objects

'git help -a' and 'git help -g' list available subcommands and some
concept guides. See 'git help <command>' or 'git help <concept>'
to read about a specific subcommand or concept.

```

Now go to your terminal and introduce yourself to Git! To set your username for **every repository** on your computer, type:

**git config --global user.name "<your\_name here>"**

Now you're ready to start using Git on your computer!

To get started, you can create a new repository on the GitHub website or perform a `git init` to create a new repository from your project directory.

## **From the terminal (Option 1)**

Here's how you can get started right from the terminal:

- If you have a project directory, just go to your terminal and in your project directory run the command:

**git init**

- If you want to initialize your project with all of the files in your project directory, run following command to include everything

**git init .**

- Let's say you have a folder for your project called "new\_project." You could head on over to that folder in your terminal window and add a local repository to it by running:

**cd new\_project**  
**git init**

- Now you can add files to the staging area one by one with:

**git add<filename>**

or run

**git add .**

- to add all of your files to the staging area. You can commit these changes with the command:

**git commit -m "<add a commit message here>"**

- and if you're happy with your changes, you can run following command to push your changes through.

**git push**

- You can check whether or not you have changes to push through any time by running

**git status**

That's it! You can now initialize a repository, commit files, commit changes, and push them through to the master branch.

## **Through github website (Option 2)**

Your repository is where you'll organize your project. You can keep folders, files, images, videos, and anything else your project needs. Before you can work with Git, you have to initialize a repository for your project and set it up so that Git will manage it. You can do this right on the GitHub website.

It's a smart idea to include a README file with information about your project. You can create one at the same time that you create your repository with the click of a checkbox.

- Go to the GitHub website, look in the upper right corner, and click the + sign and then click "New repository."
- Name the repository, and add a quick description.
- Decide whether you want this to be a public or a private repository
- Click "Initialize this repository with a README" if you want to include the README file.

You can totally start working right from this point if you want to! You can upload files, edit files, and so on right from your repository on the GitHub website. There are two ways to make changes to your project. You can make changes in your files/notebooks on your computer and you can also make changes right on GitHub.

Let's say you want to make some changes to your README file right on GitHub.

- First, go to your repository.
- Click the name of the file to bring up that file (for example, click "README.md" to go to the readme file).
- Click the pencil icon in the upper right corner of the file and make some changes.
- Write a short message in the box that describes the changes you made (and an extended description if you want).
- Click the "Commit changes" button.

Now the changes have been made to the README file in your new repository!

And now you are successfully learnt how to create a new repo, add files into it and make changes to it.

## Apache Maven

Maven is a powerful project management tool that is based on POM (project object model). It is used for projects build, dependency and documentation.

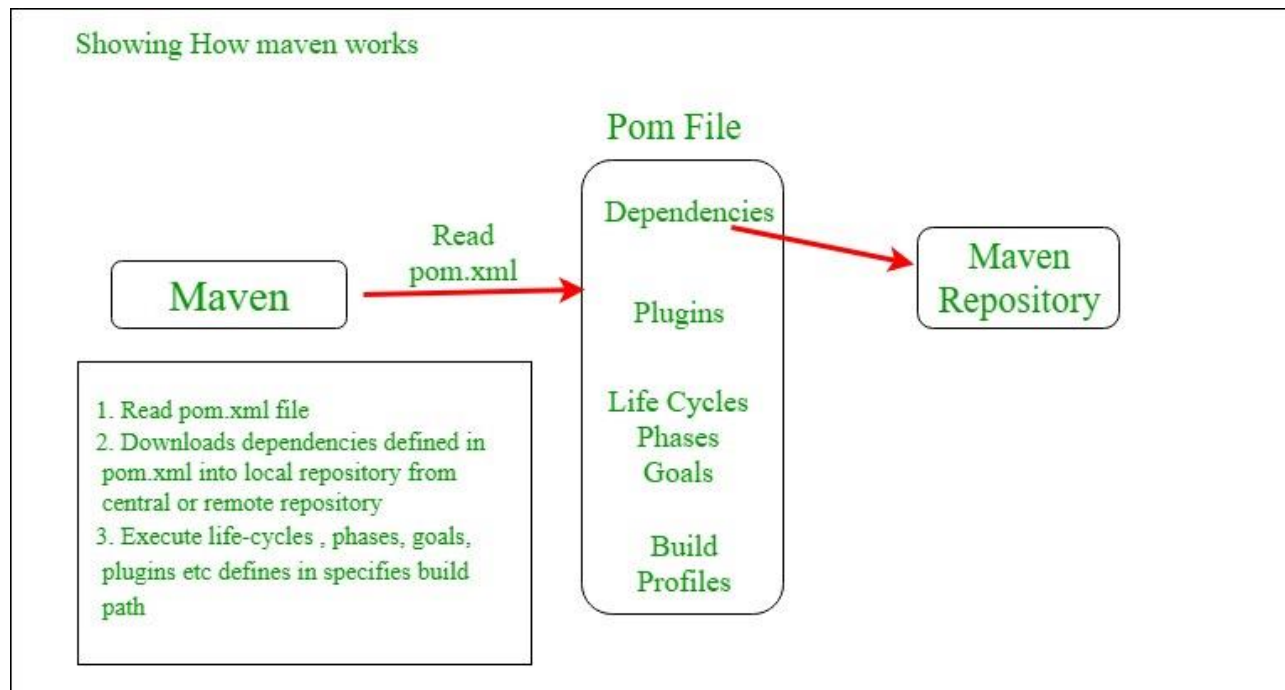
In short terms we can tell maven is a tool that can be used for building and managing any Java-based project. Maven make the day-to-day work of Java developers easier and generally help with the comprehension of any Java-based project.

### Practical Application of Maven

When working on a java project and that project contains a lot of dependencies, builds, requirement, then handling all those things manually is very difficult and tiresome. Thus using some tool which can do these works is very helpful.

**Maven is such a build management tool** which can do all the things like adding dependencies, managing the classpath to project, generating war and jar file automatically and many other things.

### How maven works?



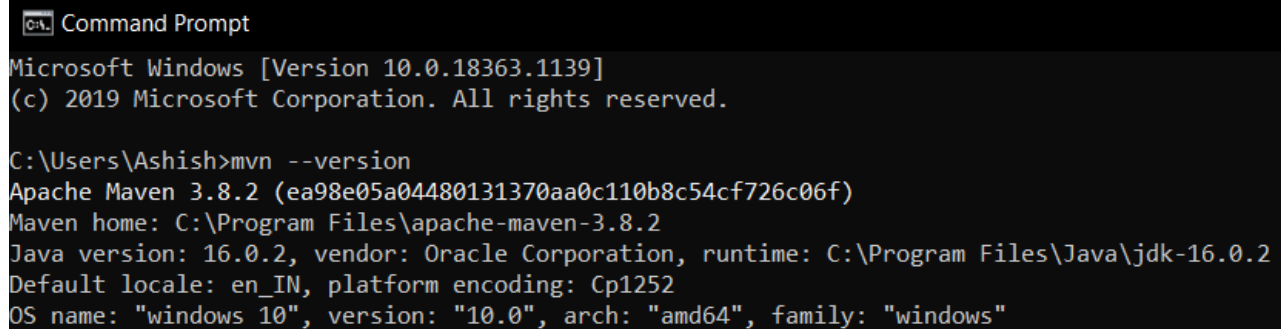
## **Core Concepts of Maven:**

1. **POM Files:** Project Object Model (POM) Files are XML file that contains information related to the project and configuration information such as dependencies, source directory, plugin, goals etc. used by Maven to build the project. When you should execute a maven command you give maven a POM file to execute the commands. Maven reads pom.xml file to accomplish its configuration and operations.
2. **Dependencies and Repositories:** Dependencies are external Java libraries required for Project and repositories are directories of packaged JAR files. The local repository is just a directory on your machine hard drive. If the dependencies are not found in the local Maven repository, Maven downloads them from a central Maven repository and puts them in your local repository.
3. **Build Life Cycles, Phases and Goals:** A build life cycle consists of a sequence of build phases, and each build phase consists of a sequence of goals. Maven command is the name of a build lifecycle, phase or goal. If a lifecycle is requested executed by giving maven command, all build phases in that life cycle are executed also. If a build phase is requested executed, all build phases before it in the defined sequence are executed too.
4. **Build Profiles:** Build profiles a set of configuration values which allows you to build your project using different configurations. For example, you may need to build your project for your local computer, for development and test. To enable different builds you can add different build profiles to your POM files using its profiles elements and are triggered in the variety of ways.
5. **Build Plugins:** Build plugins are used to perform specific goal. you can add a plugin to the POM file. Maven has some standard plugins you can use, and you can also implement your own in Java.

## Installation process of Maven

The installation of Maven includes following Steps:

1. Verify that your system has java installed or not. if not then install java ([Link for Java Installation](#) )
2. Check java Environmental variable is set or not. if not then set java environmental variable.([link to install java and setting environmental variable](#))
3. Download maven ([Link](#))
4. Unpack your maven zip at any place in your system.
5. Add the bin directory of the created directory apache-maven-3.5.3(it depends upon your installation version) to the PATH environment variable and system variable.
6. open cmd and **run mvn -v** command. If it print following lines of code then installation completed.



```
Microsoft Windows [Version 10.0.18363.1139]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\Ashish>mvn --version
Apache Maven 3.8.2 (ea98e05a04480131370aa0c110b8c54cf726c06f)
Maven home: C:\Program Files\apache-maven-3.8.2
Java version: 16.0.2, vendor: Oracle Corporation, runtime: C:\Program Files\Java\jdk-16.0.2
Default locale: en_IN, platform encoding: Cp1252
OS name: "windows 10", version: "10.0", arch: "amd64", family: "windows"
```

**Note:** Now we can create maven project using command prompt or in eclipse Java IDE. Explanation of that would be quite big. So, I am just mentioning here the link on how to create maven project ([Link](#)).