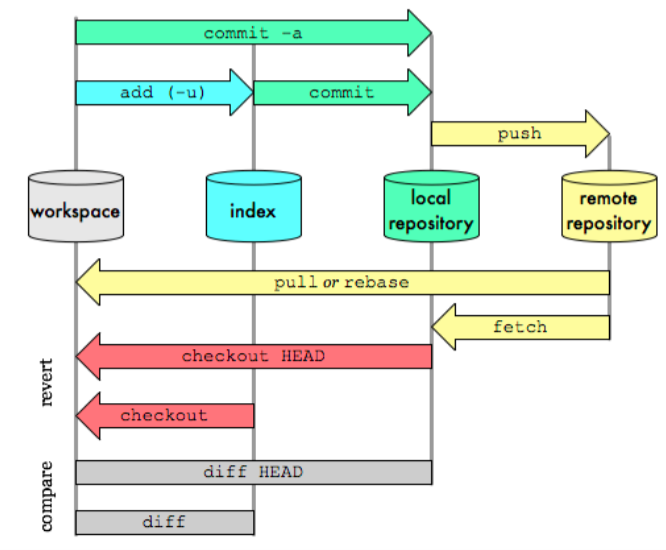
### a10.1 Introduction to Git & GitHub

#### 10.1a Understanding Version Control

Version control, also known as source control, is the practice of tracking and managing changes to software code. Version control systems are software tools that help software teams manage changes to source code over time.

#### 10.1b Git Basics

**\**

**Branching -**

**Sharmila is the Lead - Meesho**

**Hirak, Prince, Shubham - Associates.**

**Sharmila , Hirak, Prince, Shubham they have to maintain their own branch.**

**ANd they should not directly work in Master/main branch.**

**Master will have the final approved code - which should not have any mistakes.**

**After associates push their code into thor own branch, they will ask Sharmila(Lead ) to review the code and check. -**

**This review process is called as Pull request.**

**WHere once reviewed, the Lead she merges their respective branches to the main/master branch.**

**There are commands to create a new branch.**

**git branch -m <brancdfffffffffffffffffffffu7hname>**

**Git workflows can be divided into 2 sections :**

1. **Local area**
2. **Remote area**

**In Local area we have :**

1. **Working directory/ Working copy**

Whatever changes that need to be made like update in code/ adding new code is done in working copy. In IDE like Eclipse/VScode.

**2)Staging area / Index**

Staging area. Index is used as an intermediate storing of code/ files where staging maintains the files and also gives it a index.

The index is not in sequential order like(0,1.2,3)

1. **Local Repository :**

Local repository acts like an intermediate between Staging area and Remote repository

If we want to send code to a remote repository then we can only do it from Local repository and not directly from the **working copy** or from **Staging area**.

Whatever code/ files that were **added** in staging area are then **committed** **to Local repository are** to be **pushed** to the **Remote repository.**

Local repository has a .**git** file that has all the data.

**In Remote area we have :**

**Remote Repository :** It is a repository which is stored in Cloud/ Internet using Github or some other storage repo softwares.

It has a master branches and different branches for different implementations.

**Different workflows in Git :**

**Workflow in order when new to project – > 1)Git clone 2) Checkout to local branch**

**3) Git add 4) Git commit 5) Git Push 6) Merge with Pull request to Master**

#### 10.1c Branching in Git

Each new feature should reside in its own branch, which can be [pushed to the central repository](https://www.atlassian.com/git/tutorials/syncing/git-push) for backup/collaboration. But, instead of branching off of main, feature branches use develop as their parent branch. When a feature is complete, it gets [merged back into develop](https://www.atlassian.com/git/tutorials/using-branches/git-merge). Features should never interact directly with main./master

#### 10.1d Remote Repositories

A remote URL is Git's fancy way of saying "the place where your code is stored." That URL could be your repository on GitHub, or another user's fork, or even on a completely different server.

You can only push to two types of URL addresses:

* An HTTPS URL like https://github.com/user/repo.git
* An SSH URL, like git@github.com:user/repo.git

Git associates a remote URL with a name, and your default remote is usually called origin.

## [Creating remote repositories](https://docs.github.com/en/get-started/getting-started-with-git/about-remote-repositories#creating-remote-repositories)

You can use the git remote add command to match a remote URL with a name. For example, you'd type the following in the command line:

git remote add origin <REMOTE\_URL>

This associates the name origin with the REMOTE\_URL.

You can use the command git remote set-url to [change a remote's URL](https://docs.github.com/en/get-started/getting-started-with-git/managing-remote-repositories).

### 10.2 Implementing Git & GitHub Commands

#### 10.2a Initialising a Repository

**Cloning :**

Cloning is a process of downloading the code which is available in the remote repository to the Local repository.

Cloning is usually done when a new developer wants to work with an existing code, first he downloads the code from remote repository to local repository and then starts working on code in his working copy.

The command used is : **git clone <URL\_of\_RemoteRepo>**

**If we are starting with the Push operation : then we start with Git init**

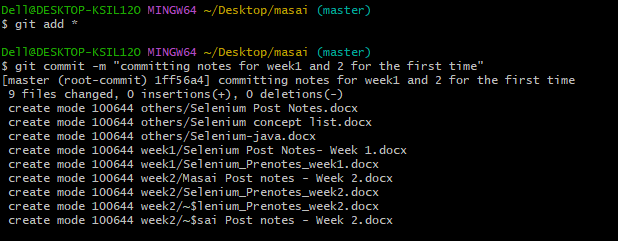
1. **Initialize a Repositiory :**

**git init**

****

1. **Git add <filename> / Git add \*(for all files)**

This command moves the code from working directory to Staging area.



1. **Git remote add**

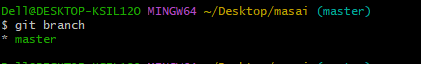
Connect Local Repository to Remote Repository

Run the following command to add a remote named "origin" (you can choose any name) and specify the URL of the remote repository:

git remote add origin <GitHUR\_REPO\_URL>



1. **To know branch name :**

****

**Git Branching -**

Git branches are effectively a pointer to a snapshot of your changes. When you want to add a new feature or fix a bug—no matter how big or how small—you spawn a new branch to encapsulate your changes

To create a new branch is :

Git checkout –b <branch\_name>

This is shorthand for:

$ git branch iss53

$ git checkout iss53

#### 10.2b Staging and Committing Changes

**Git commit :**

This command is used when we wish to move the code from Staging area to the Local repository.

Command - **Git commit -m ”message to be given while committing”**

**Git commit –m “Committing my cucumber testing project – simpleform implementation ”**

#### 10.2c Pushing Changes

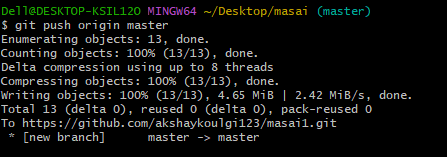
**Git Push :**

**To push your changes to a remote repository, you'll use the git push command. Here's the basic syntax**

**git push <remote-name> <branch-name>**

**remote-name>:** The name of the remote repository (e.g., "origin").

**<branch-name>:** The name of the branch you want to push**.**

****

**While pushing, the git asks for sign in and authrnticate, enter username and password and click on allow git operations.**

#### 10.2d Pulling Changes

1. **Fetch V/s Pull**

**Git Pull -** Git pull directly pulls the code from Remote repository to your working directory.

There is no involvement of Local repository in Git Pull.

**Git fetch** - Git fetch will fetch the code from your remote repository to your local repository.(with a .git file)

Now, if we want to get the code from Local repository to Working Directory, then we use **Merge**.

**Git Pull = Git Fetch + Merge**

**We can say that when we do Git Pull , it is equivalent to fetch and merge.’**

**Note : Merging can be done both at Remote repo level and also at Local level**

### 10.3 Introduction to CI/CD with Jenkins & Integration with Selenium

#### 10.3a Understanding CI/CD

**Introduction to CI-CD Jenkins & Integration with Selenium**

### **What is Jenkins?**

Jenkins is a self-contained, open-source automation **server** that can be used to **automate all sorts of tasks** related to **building**, **testing**, and **delivering or deploying software.**

**What is Continuous Integration?**

Continuous integration is the process of continuously checking-in the developer’s code into a version control system(GitHub) and triggering the build to check and identify bugs in the written code.(Triggering the Automation scripts.)

Continuous Integration – Continuous Delivery, Continuous testing.

**Web Hooks** – It will help Jenkins to sense the changes made in the repository.

Continous testing – Automating the Execution process.

Jenkins is a CI-CD-CT tool.

Developers make changes the code, they will push it to GitHub.

Jenkins senses the changes in GitHub.

WebHooks – Helps Jenkins to detect the changes made by the developer.

This will trigger the Automation scripts.

Automating deployment - It will automatically deploy the code by itself.

Integration – connection between different systems.

**Continuous connection between developers, testers, Devops, and other teams.**

**Automatically deploy, it also triggers automation testing code for execution.**

#### 10.3b Setting Up Jenkins

**What is Jenkins Pipeline? What is a CI CD pipeline?**

The pipeline can be defined as the **suite of plugins** supporting the implementation and integration of continuous delivery pipelines in Jenkins.

**Pipeline** is nothing but group of tasks that Jenkins is assigned with.

Continuous integration or continuous delivery pipeline consists of **build**, **deploy**, **test**, **release** pipeline.

The pipeline feature saves a lot of time and error in maintaining the builds. Basically, a pipeline is **a group of build jobs** that are chained and integrated in sequence.

Whenever you change your code, Jenkins can automatically perform your Selenium tests and then deploy your code to a new environment if the tests pass.

**Poll SCM**

Jenkins allows you to set a time and date for executing your tests.

The Test Reports and performance logs can be saved for future reference.

#### Jenkins can be used in a continuous integration setup with Maven to create and test a project.Prerequisites:10.3c Creating and Configuring Jenkins Jobs

* Gain proficiency in creating and configuring Jenkins jobs to automate build, test, and deployment processes.

#### 10.3d Integrating Selenium with Jenkins

### **Jenkins Installed:**

### Make sure Jenkins is installed on your machine. You can download Jenkins from https://jenkins.io/download/ and follow the installation instructions.

### **Maven Installed:**

### Ensure Maven is installed on your machine. You can download Maven from https://maven.apache.org/download.cgi and follow the installation instructions.

**Maven** –

Maven is a Build management tool – used to build the code.

What is a build?

Code packaged into a file is called Build.

Features of Maven –

1. **Dependencies** – we can easily mention Dependencies in our **Pom.xml**.
2. **File Structure** – src/test/java , src/main/java , src/main/resources, src/test/resources.
3. **Build management** – With pom.xml we can build the software.
4. **Test the software with pom.xml - mvn test**
5. **We can integrate maven with Jenkins, Github for Automating the Execution process.**

**Maven Build Life cycle**

**Link :** [**https://maven.apache.org/guides/introduction/introduction-to-the-lifecycle.html**](https://maven.apache.org/guides/introduction/introduction-to-the-lifecycle.html)

### A Build Lifecycle is Made Up of Phases

Maven has a predefined set of build phases and goals that define the build life cycle.

The build life cycle is the sequence of stages that a Maven project goes through during the build process.

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(These points are written from Dev’s POV).

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

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

install - install the package into the local repository/machine, 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.

These phases are executed sequentially and each phase is dependent on the previous phase. You can run a specific phase by using the **mvn <phase>** command, e.g., **mvn compile** to compile the code, or **mvn install** to install the package in the local repository.

* Automation testers typically come into play during the **test** phase of the Maven build lifecycle.
* In this phase, automated tests are executed against the compiled source code to verify its functionality.
* Automation testers write these tests using frameworks such as JUnit or TestNG for Regression/unit testing, and tools like Selenium or RestAssured for integration and API testing.
* The **test** phase ensures that the application behaves as expected and meets the specified requirements.

Add both maven surefire plugin and maven compiler plugin in plugins section.

The Maven Surefire Plugin is used to run tests written in JUnit, TestNG, or any other supported testing framework. It generates reports on test results and can also be configured to run only a subset of tests or to include/exclude specific tests based on various criteria.

The Maven Compiler Plugin is used to compile the source code of your project. It can compile Java sources to the target bytecode version of your choice and can also enforce compilation rules, such as the use of specific Java versions or language features

## Maven Surefire Plugin

Requirements: Maven 3.2.5 and JDK 1.8 or higher.

The Surefire Plugin is used during the test phase of the build lifecycle to execute the unit tests of an application. It generates reports in two different file formats:

* Plain text files (\*.txt)
* XML files (\*.xml)

## Apache Maven Compiler Plugin

The Compiler Plugin is used to compile the sources of your project. Since 3.0, the default compiler is javax.tools.JavaCompiler (if you are using java 1.6) and is used to compile Java sources. If you want to force the plugin using javac

We will get both plugins from maven website , search both plugins and we will get in Usage section .

Add the build tag inside version tag.

Usage of Maven compiler plugin

1. <build>
2. <pluginManagement>
3. <plugins>
4. <plugin>
5. <groupId>org.apache.maven.plugins</groupId>
6. <artifactId>maven-compiler-plugin</artifactId>
7. <version>3.11.0</version>
8. <configuration> <includes>
9. <excludes>\*\*/\*TestRunner.java </excludes>
10. </includes>
11. </configuration>
12. </plugin>
13. </plugins>
14. </pluginManagement>
15. </build>

**Usage of Maven surefire plugin**

Inside plugin  paste the following :

1. <plugin>
2. <groupId>org.apache.maven.plugins</groupId>
3. <artifactId>maven-surefire-plugin</artifactId>
4. <version>3.2.2</version>
5. </plugin>

This is how our pom.xml file will look like :



Inside exclude tag add the runner file name.

After configuration. Run the pom.xml as Run with Maven Build option.

Give the goal as test.

We use Pom.xml to Package,Build, and Execute the tests.

How to Run in command prompt :

To include TestNG tests in the Surefire Plugin configuration in the **pom.xml** file, you need to specify the TestNG suite XML file or the package containing your TestNG classes in the **<includes>** tag. Here's an example of how you can configure the Surefire Plugin in your **pom.xml**

**<properties>**

**<maven.compiler.source>1.8</maven.compiler.source>**

**<maven.compiler.target>1.8</maven.compiler.target>**

**</properties>**



**Things to check before running in command prompt :**

First, Check java version

Also check java compiler

Mvn –version

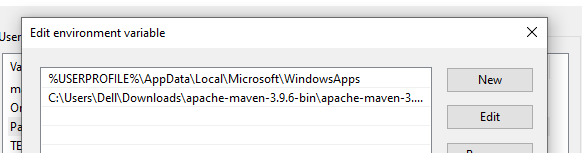
If mvn is not there, **download maven from the internet.**

Download maven and extract the files to bin option

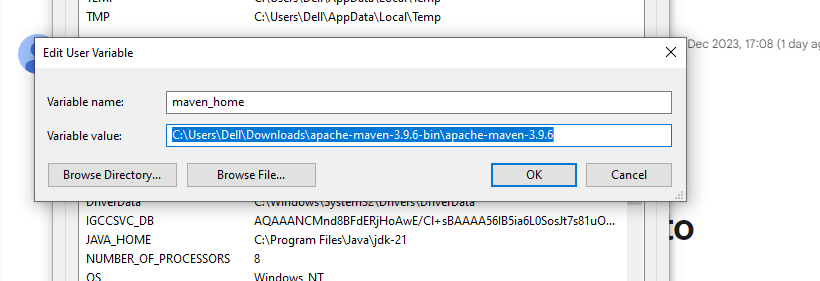
The extracted folder name should have bin.

Add Maven\_home in Environmental variables.

In path add bin path



And :



In Environmental variables , check whether JAVA\_HOME is set or not.

### **Java Installed:**

### Make sure Java is installed on your machine. You can download Java from https://www.oracle.com/java/technologies/javase-downloads.html or use an OpenJDK distribution.

### **Selenium WebDriver Setup:**

### **Include Selenium WebDriver dependencies in your Maven project.**

### **Steps:**

### **1. Create a Maven Project:**

### **Create a new Maven project or use an existing one.**

### **2. Configure pom.xml:**

### **Add dependencies for Selenium WebDriver, TestNG, and any other necessary libraries in the pom.xml file.Steps to install Jenkins & Selenium**

**Configuring Jenkins**

You can install Jenkins from the Jenkins website on your computer. Jenkins may be launched from the command line or run on a web application server.

**For command-line implementation, see the details below.**

* At the command prompt, execute java -jar followed by the location of a .war file.
* If your Jenkins.war file has begun to run, you can confirm this by pressing enter and viewing the console output.
* Now you can see if Jenkins is set to go by using the default port of 8080.
* Just type “http://localhost:8080” into your browser’s address bar. Your Jenkins user interface will load.

#### 10.3e Managing Jenkins Pipelines

Jenkins Pipeline (or simply "Pipeline") is a suite of plugins which supports implementing and integrating *continuous delivery pipelines* into Jenkins.

A *continuous delivery pipeline* is an automated expression of your process for getting software from version control right through to your users and customers.

#### 10.3f Monitoring and Reporting

In Jenkins, a report is a structured and graphical way where we can see the execution results or output of the test. Reports are also useful when we have to communicate the results with our team members or with other stakeholders. There are many plugins available for reporting in Jenkins. It is easy, convenient, and recommended.