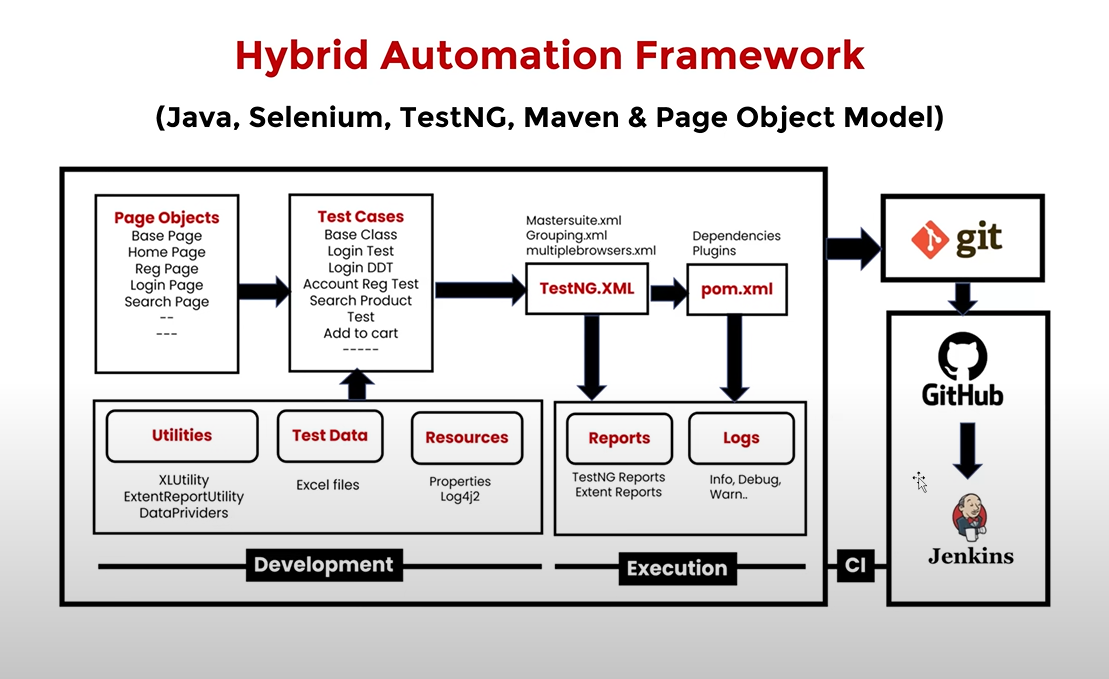
Hybrid Automation Framework



Create a new Maven Project

Add required dependencies in pom.xml(Please check links below)

<https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-java>

<https://mvnrepository.com/artifact/io.github.bonigarcia/webdrivermanage>

<https://mvnrepository.com/artifact/org.apache.poi/poi>

<https://mvnrepository.com/artifact/org.apache.poi/poi-ooxml>

<https://mvnrepository.com/artifact/org.apache.logging.log4j/log4j-core>

<https://mvnrepository.com/artifact/org.apache.logging.log4j/log4j-api>

<https://mvnrepository.com/artifact/org.apache.logging.log4j/log4j-slf4j-impl>

<https://mvnrepository.com/artifact/com.aventstack/extentreports>

<https://mvnrepository.com/artifact/org.testng/testng>

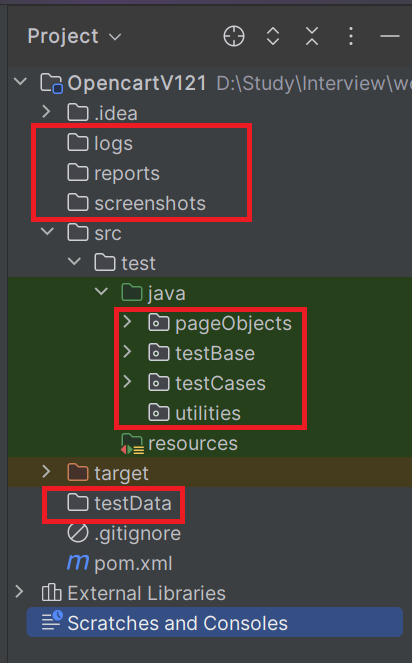
<https://mvnrepository.com/artifact/commons-io/commons-io>

<https://mvnrepository.com/artifact/org.apache.commons/commons-lang3>

Create a Folder Structure

Create all the folders highlighted in red.

1. Logs, reports, screenshots, testData and TestNG.xml are created on project Level
2. pageObjects, testBase, testCases, utilitlies are created under src/test/java folders
3. Delete main/java folder in intellij as we are not going to need it.
4. Note TestNG.xml will be created in the future.



Development of Hybrid Driven Framework

**1) Test Case: Account Registration**

1.1: Create BasePage under **“pageObjects”** which includes only constructor. This will be invoked by every Page Object Class constructor(Re-usability).

1.2: Create Page Object Classes for HomePage, AccountRegisterationPage under **“pageObjects”** package. (These classes extends from BasePage).

1.3: Create AccountRegisterationTest under **“testCases”**

1.4: Create BaseClass under “**testBase”** package and copy re-usable methods.

1.5: Create re-usable methods to generate random numbers and strings in BaseClass.

**2) Adding logs to test case (log4j2)**

2.1: Add **log4j2.xml** file under **src/test/resources.**

2.2: Update BaseClass.

2.3: Add log statements top AccountRegistrationTest.

**3) Run Tests on Desired Browser/ Cross Browser / Parallel**

3.1: Create testing.xml(master.xml) file to Run Test Cases and Parametrize browser name and OS to BaseClass 🡪 setup() method.

3.2: Update BaseClass 🡪 setup() method, launch browser based on conditions.

3.3: Maintain separate xml to run tests multiple browsers parallelly(crossBrowser.xml).

**4) Read Common values from config.properties file.**

4.1: Add config.properties file under **src/test/resources**.

4.2: Update BaseClass 🡪 setup() method, add script to load config.properties file.

4.3: Replace hard coded values in Test Cases like url, username, password etc…

**5) Login Test Case**

5.1: Create and update page object classes. LoginPage, MyAccountPage – new Classes HomePage—update by adding login link element.

5.2: Create LoginTest

5.3: Add entry in testing.xml

**6) Data Driven Login Test**

6.1: Prepare test data in Excel, place the excel file inside the testData folder.

6.2: Create ExcelUtility.class under **utilities** package.

6.3: Update Page Object class MyAccountPage, add logout link element.

6.4: Create DataProviders class in **utilities** package to maintain data providers for data driven tests.

6.5: Create LoginDataDrivenTest under **testCases** package.

6.6: Add an Entry in testing.xml file.

**7) Grouping Tests**

7.1: Add all the test cases into specific group (sanity, regression, master, etc).

7.2: Also add BaseClass methods setup() and teardown() to all groups.

7.3: Create separate TestNG xml file (grouping.xml) to run groups and include groups which we want to execute.

**8) Add Extent Reports to Project**

8.1: Create ExtentReportUility utility class under **utilities** package.

8.2: Add captureScreen() method in BaseClass

8.3: Add ExtentReportUility(Listner class) entry in testing.xml file.

8.4: Make sure WebDriver is static in BaseClass, we refer same driver instance in ExtentReportUtility.

**9) Run Failed Tests.**

test-output 🡪 testing-failed.xml

**10) Run Tests on Selenium Grid.**

**https://googlechromelabs.github.io/chrome-for-testing/known-good-versions-with-downloads.json**

**Pre-requisite:** Grid Standalone /Hub & Node Setup (Refer the Grid Setup as below)

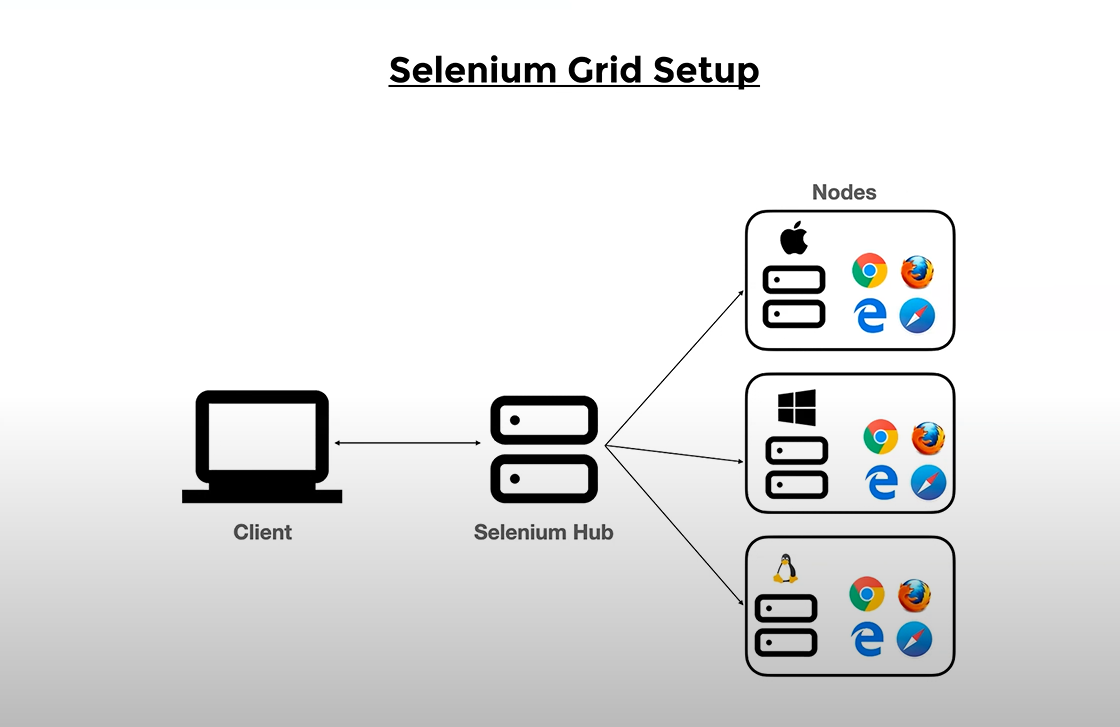
10.1 Add **execution\_env=local/remote** in config.properties file under resources folder.

10.2 Update setup() method in the BaseClass (capture execution environment from config.properties file then add required capabilities of OS & Browser conditions).

10.3 Run the tests from testng.xml

**Grid Setup:**

* Download selenium-server-4.15.0.jar and place it somewhere
* Run below command in cmd to start Selenium Grid
  + java -jar selenium-server-4.15.0.jar standalone
* URL to see sessions: <http://localhost:4444/>



**Standalone Setup (Single Machine)**

1. Download selenium-server-4.24.0.jar and place it somewhere.
2. Run below command in cmd to start Selenium Grid

java -jar selenium-server-4.24.0.jar standalone

1. URL to see sessions: <http://localhost:4444/>

HUB and Node Setup(Multiple Machines)

1. Download selenium-server-4.24.0.jar and place it somewhere in both (hub and node) the machines.
2. Run below command to make machine as HUB:

java-jar- selenium-server-4.24.0.jar hub

1. Run below command to make machine as NODE:

java-jar- selenium-server-4.24.0.jar node --hub http://<hub-ip>:4444/

1. URL to see sessions: <http://localhost:4444/>

Grid URL on executing command for standalone

http://192.168.1.10:4444/ui/

**11) Run Tests on Docker with Selenium Grid Environment.**

Docker – based on Containerization

Docker hub – remote repository – maintains N number of images – to get some environments(Browsers and OS)

Get the Image from the docker hub and we will create a container and we will use it for testing.

No hardware is used, hence the performance is good.

We will take Hub and nodes from Docker Images only.

Image Commands

Docker images

Docker pull imageName

Docker rmi imageId

Container Commands

>docker ps

Docker run imageName

>docker run -dit ubuntu

docker start imageId

>docker system df

>docker system prune -f

Selenium Grid Setup with Docker

docker pull selenium/hub

docker pull selenium/node-firefox

docker pull selenium/node-chrome

To remove grid: docker network rm grid

Create a Network

Get 3 images – Hub, linux-Firefox, linux-Chrome

docker pull selenium/hub

docker pull selenium/node-firefox

docker pull selenium/node-chrome

C:\Users\chars>docker network create grid

99b2c2224a89f4057c6b2cb1beceb5edd631cc5f302bb7e3b758514f19fec4ba

C:\Users\chars>docker run -d -p 4442-4444:4442-4444 --net grid --name selenium-hub selenium/hub:latest

470cdd9420afbecbd1a001b92c945209465e8f12305f62f9d719b2e7a4896161

C:\Users\chars>docker run -d --net grid -e SE\_EVENT\_BUS\_HOST=selenium-hub -e SE\_EVENT\_BUS\_PUBLISH\_PORT=4442 -e SE\_EVENT\_BUS\_SUBSCRIBE\_PORT=4443 selenium/node-firefox

845d143a884ffb2dd4d344484f435ff1c9201a27120c5680fb537b09afa4c395

C:\Users\chars>docker run -d --net grid -e SE\_EVENT\_BUS\_HOST=selenium-hub -e SE\_EVENT\_BUS\_PUBLISH\_PORT=4442 -e SE\_EVENT\_BUS\_SUBSCRIBE\_PORT=4443 selenium/node-chrome

ca20ea7fc480ed67fa3e395dfc1ec2e7627aba0393df280032e7db4eef2e6a06

C:\Users\chars>docker images

REPOSITORY TAG IMAGE ID CREATED SIZE

selenium/node-firefox latest f6b345119caf 9 days ago 1.55GB

selenium/node-chrome latest 4c434b750150 9 days ago 1.43GB

selenium/hub latest 83b6c7eae207 9 days ago 468MB

C:\Users\chars>docker ps

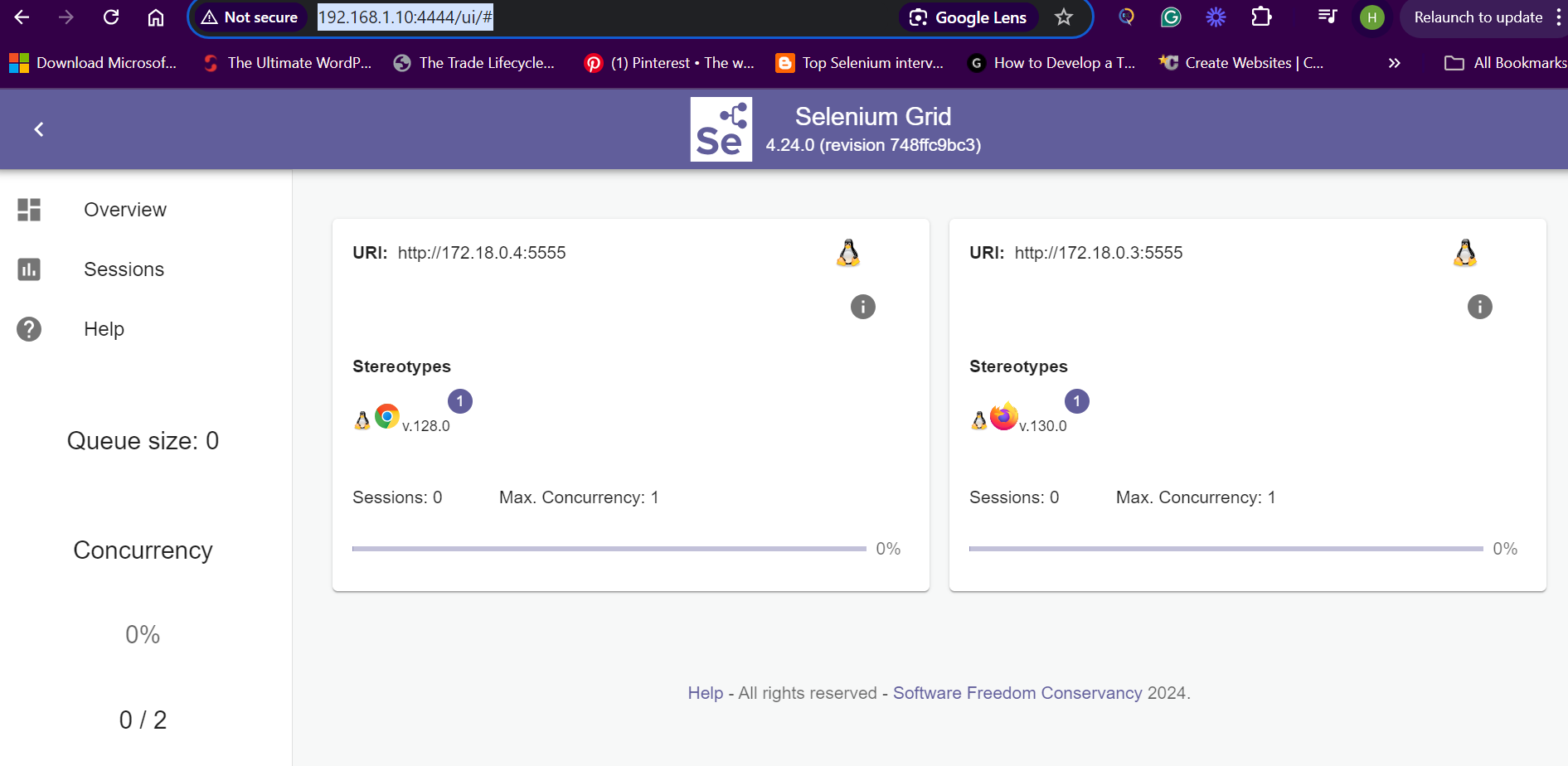
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

ca20ea7fc480 selenium/node-chrome "/opt/bin/entry\_poin…" 2 minutes ago Up 2 minutes 5900/tcp frosty\_gates

845d143a884f selenium/node-firefox "/opt/bin/entry\_poin…" 2 minutes ago Up 2 minutes 5900/tcp hopeful\_wiles

470cdd9420af selenium/hub:latest "/opt/bin/entry\_poin…" 5 minutes ago Up 5 minutes 0.0.0.0:4442-4444->4442-4444/tcp selenium-hub

http://192.168.1.10:4444/ui/#



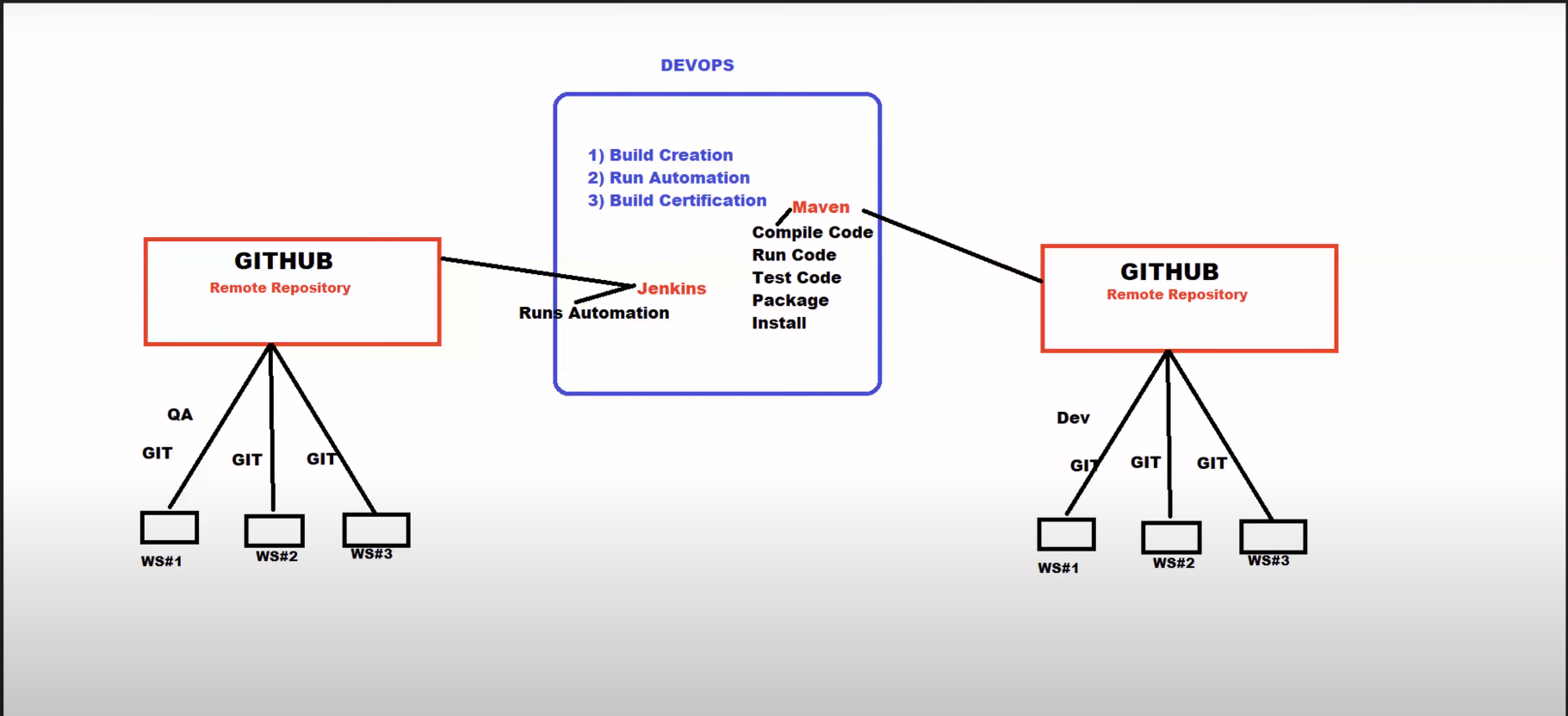
**12) Run Tests using Maven pom.xml, Command Prompt & run.bat file**

The execution starts from pom.xml -> Testng.xml-> Test cases -> Page object classes, utilities, testdata, configuration files.

Intelli j is only required to do changes in code.

We can configure any xml in pom and its ready to execute.

**13) Push the Code to Gite and Github repository**

****

Sharing the files from workspace to local repository is commit

Sharing files from local repository to github repository(remote) is push

Dev and QA will do above 2 steps

Now the Devops process starts

1. Build Creation

2.Run Automation

3. Build Certification

Maven is build tool, with which they will create a build from Maven. It is done through process automation.

They will fetch Dev code from github and create a build.

They will fetch Test Automation from github and run automation.

Jenkins tool will pull the latest code from github and run the automation.

Once the Test scripts are executed and then the Build certification will happen.

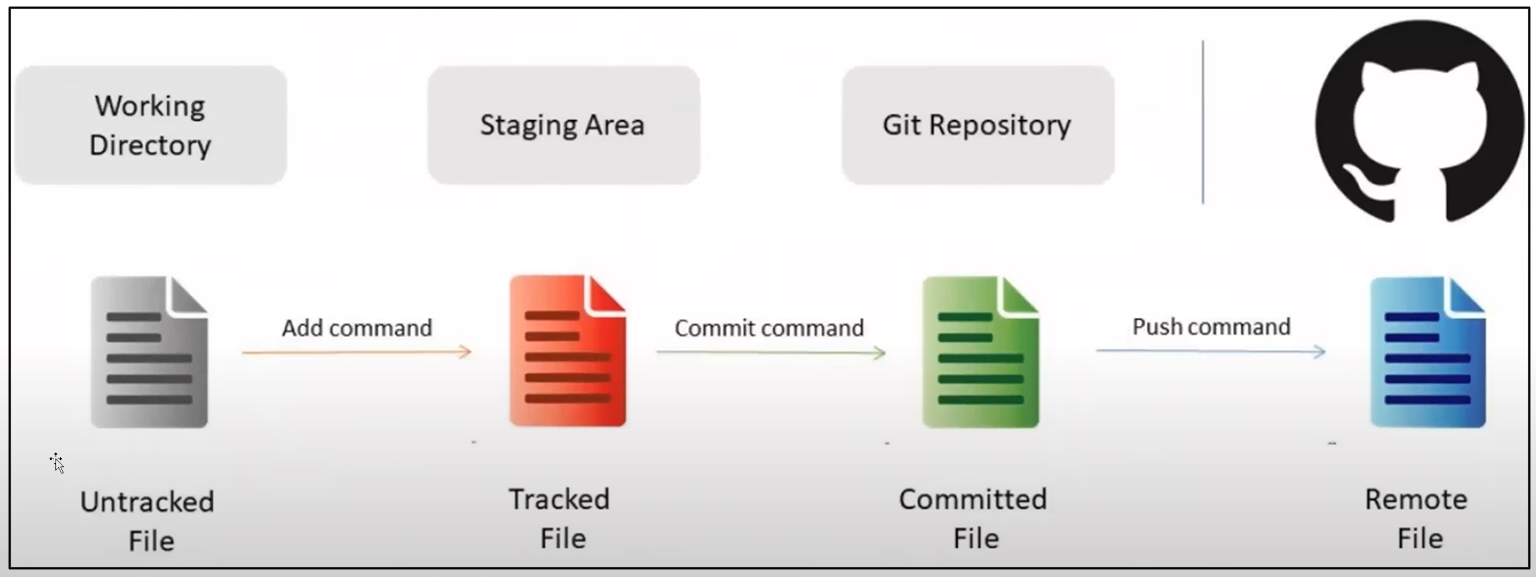
Throughout the night the builds are processed and they are called as nightly builds.

The next day new build is used for building Testcases.

Pre-requisites

1. Install Git
2. Create an account with github
3. Workflow of Git and GitHub

Git Workflow from creating a local repository to pushing changes to a remote repository



**Round1:**

* 1. Create a New Local Git Repository

To create a new local Git repository, navigate to the desired directory and run git init

* 1. Provide User info to Git Repo(One Time)

You need to configure your username and email for Git, which is usually done once. Replace **“your Name”** or **“your email”** with your actual name and email:

git config user.name “your name”

git config user.email “your email”

* 1. Adding files / Folders to staging /indexing

git add -A 🡪 add all the files and folders to staging

git add filename 🡪 add specific file

git add \*.java 🡪 add files with specific extension

git add foldername 🡪 add specific folder

* 1. Commit the code into Local (git) repository

git commit -m “commit message”.

* 1. Push all these files to Remote Repository

Create Remote Repository with same name as project and capture repo url.

url: <https://github.com/charshal12/OpencartV121.git>

* 1. Connect local repository with remote repository(Only once you have to do it)

git remote add origin “https://github.com/charshal12/OpencartV121.git”

* 1. Push the code into remote repository

git push origin master

**Round 2:** when you make any changes in project we need to use only 3 commands as below:

1. git status
2. git add -A
3. git commit -m “commit message”
4. git push origin master

Pull the code from the Remote Repository

git pull “<https://github.com/charshal12/OpencartV121.git>”

Clone the repository from Remote Repository

git clone “https://github.com/charshal12/HotWireAutomationProject.git”

**14) Run Tests using Jenkins.**