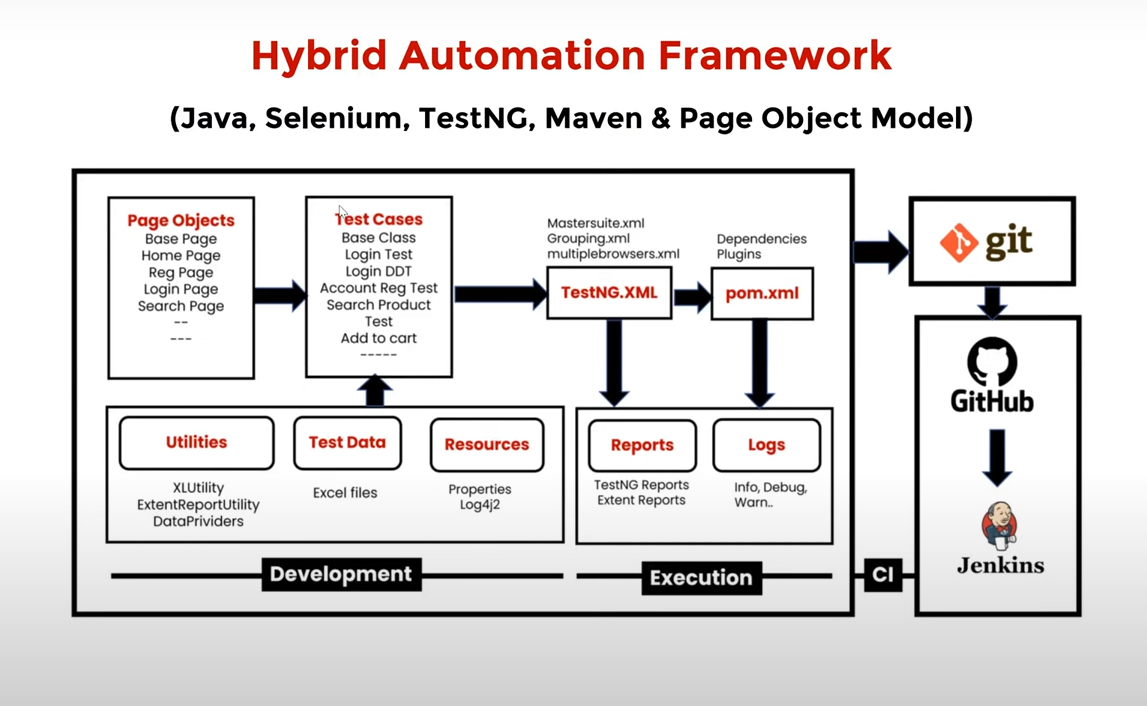
**Development of Hybrid Driven Framework**

****

**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, Registrationpage under pageobjects package (These classes extends from BasePage).

1.3: Create AccountRegistrationTest under "testCases"

1.4: Create Baseclass under testBase package and copyre-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/ resourses

2.2: Update BaseClass.

2.3: Add log statements to AccountRegistrationTest.

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

3.1: Create testng.xml file to Run Test Cases and parameterize 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.

Note: xml file should be created in package level always

**4) Read Common values from** [**config.properties**](http://config.properties) **file.**

4.1: Add [config.properties](http://config.properties) file under src/test/resoures.

4.2: Update BaseClass -setup() method, add script to load [config.properties](http://config.properties) file.

4.3: Replace hard coded values in Test Cases like url, username, password et ...

**5) Login Test Case**

5.1: Create and update page object classes. Login Page, My Account page-new classes , Homepage

- update by adding login link element.

5.2: Create Login Test

5.3: Add entry testng.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 LoginDataDrivenIest under testCases package

6.6: Add an Entry in testng.xml file

**7) Grouping Tests**

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

7.2: Also add BaseClass methods setup() & 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 ExtentReportUtility utility class under utilities package.

8.2: Add captureScreen() method in BaseClass

8.3: Add ExtentReportUtility (Listener class) entry in testng.xml file

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

**9) Run only Failed Tests.**

test-output-

>testng-failed.xml🡪run this file

THE Entry will be present until the next test is failed

**10) Run Tests on Selenium Grid**

Grid Setup:

*1.Standalone Setup (Single machine):*

*1. Download selenium-server-4.15.0.jar and place it somewhere*

*2. Run below command to start Selenium Grid*

*java -jar selenium-server-4.15.0.jar standalone*

*3. URL to see sessions:* [*http://localhost:4444*](http://localhost:4444)*/*

*2.Hub & Node Setup (Multiple machines):*

*1. Download selenium-server-4.15.0.jar and place it somewhere in both (hub & node) the machines 2. Run below command to make machine as hub*

*java -jar selenium-server-4.15.0.jar hub*

*3. Run below command to make machine as node (every node)*

*java -jar selenium-server-4.15.0.jar node --hub http://<hub-ip>:4444*

*4444 is default port*

*4. URL to see sessions:* [*http://localhost:4444/*](http://localhost:4444/)

10.1: Add execution \_ env=local/remote in [config.properties](http://config.properties) file under resources folder.

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

10.3: Run the tests from testing.xml

**11) Run tests on Docker with Selenium Grid env**

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

**pom.xml,** : Add “maven-compiler-plugin”, “maven-surefire-plugin” in **pom.xml** and run as “Maven Test”

**Command Prompt**: (OUTSIDE OF ECLIPSE, in OS)

1. downland “https://maven.apache.org/download.cgi (.bin.zip)

extract and keep the file in C drive (C:\Program Files\apache-maven-3.9.2\bin)

2. configure above path in env variables

3. open cmd prompt: **mvn-version**

4. to test tc in cmd --🡪 go to project location in cmd and run the cmd: **mvn clean test**

**run.bat file: if you want to run the tc y .bat file,**

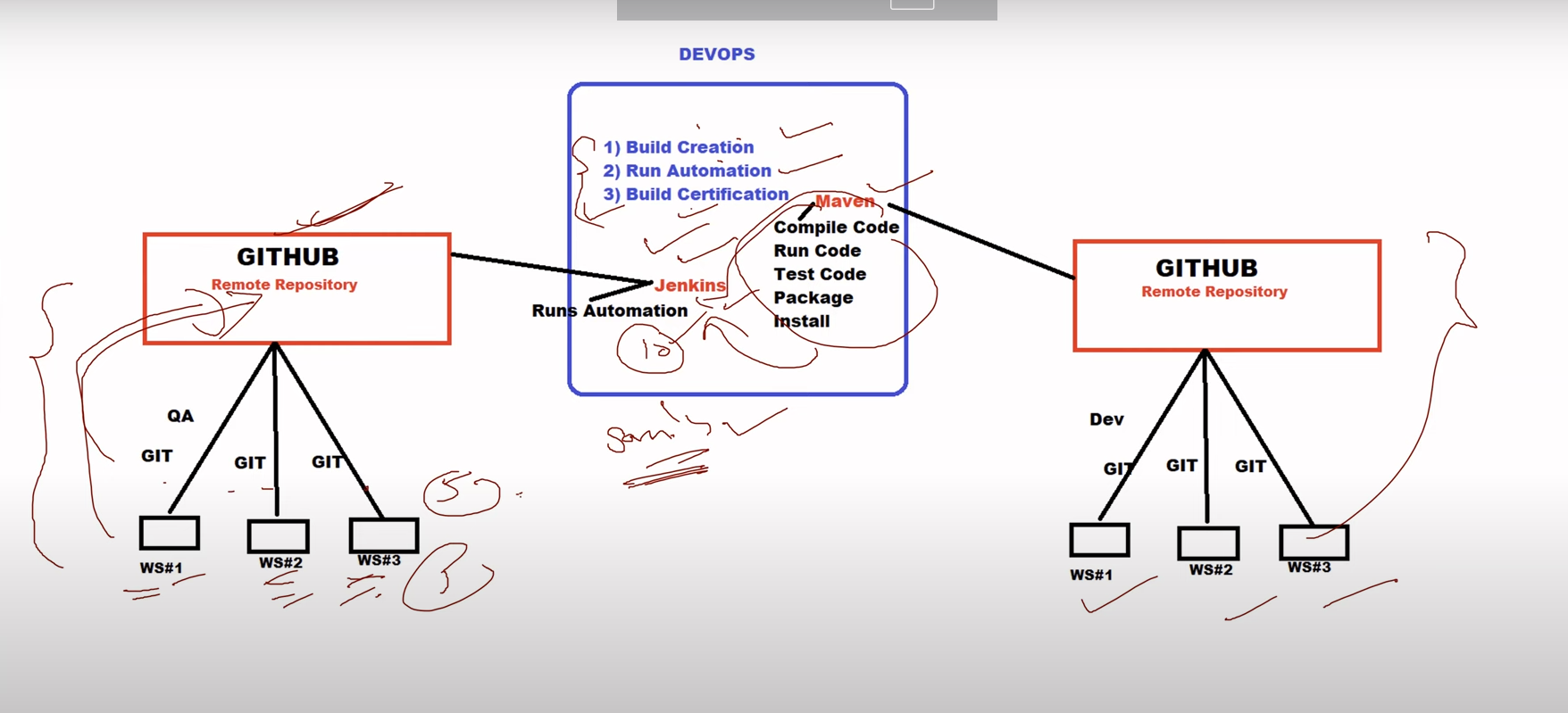
1. create a TXT document with .bat extension
2. open the file, and write 2 cmds

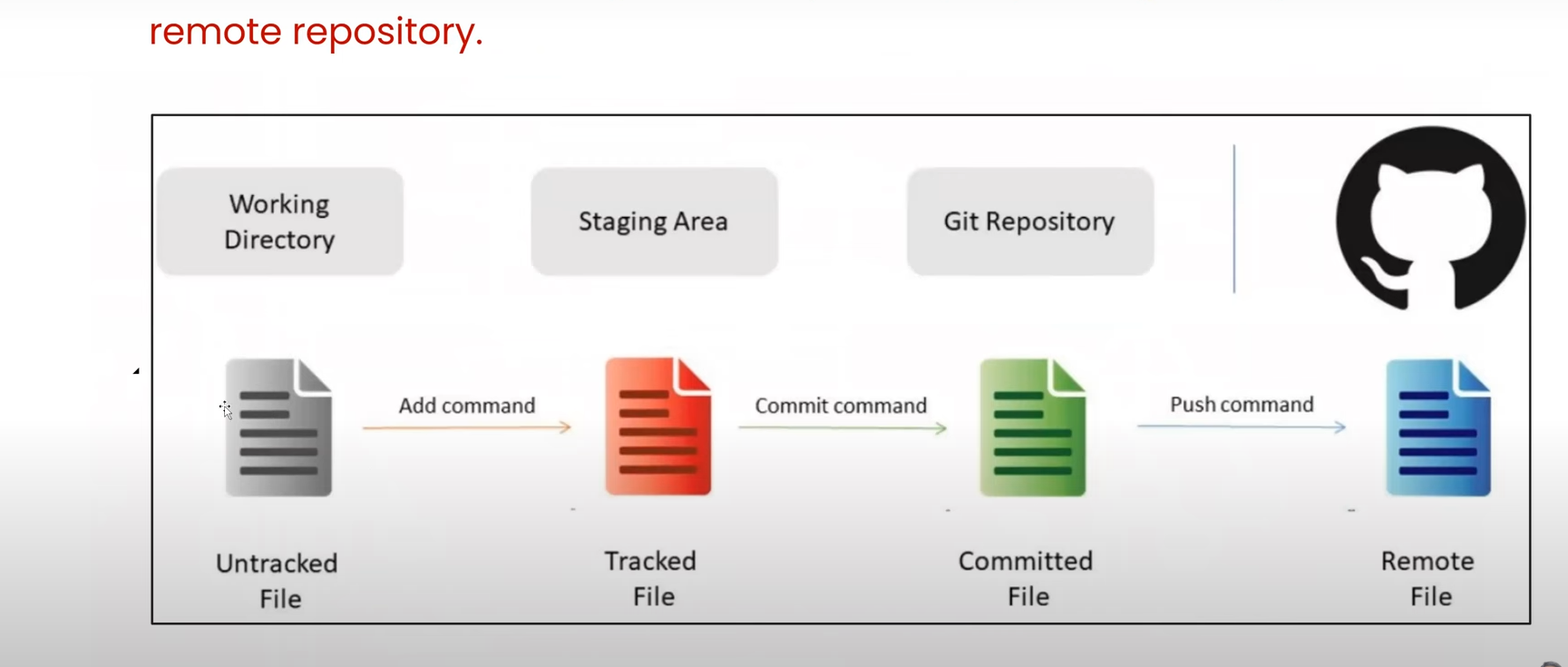
cd C:\Users\KGADDAM\workspace\OpenCartV121

mvn test

3. save it and open cmd prompt and execute .bat file(double click)

**12) Push the Code to Git & GitHub Repository**

****

****

Create new repo in git

Repo url : [**https://github.com/Kavya-Bavandla/OpenCartV121.git**](https://github.com/Kavya-Bavandla/OpenCartV121.git)

Day-1

**1. git init (create repository) in below path (one time)**

**C:\Users\KGADDAM\workspace\OpenCartV121**

kgaddam@INLEN8520042741 MINGW64 ~/workspace/OpenCartV121

$ **git init**

Initialized empty Git repository in C:/Users/KGADDAM/workspace/OpenCartV121/.git/

kgaddam@INLEN8520042741 MINGW64 ~/workspace/OpenCartV121 (master)

$

**2.provide user info to git repository**

git config - - global user.name “your name”

git config - - global user.email “your email”

kgaddam@INLEN8520042741 MINGW64 ~/workspace/OpenCartV121 (master)

$ **git config --global user.name "Kavya"**

kgaddam@INLEN8520042741 MINGW64 ~/workspace/OpenCartV121 (master)

**$ git config --global user.email "gaddamkavya416@gmail.com"**

kgaddam@INLEN8520042741 MINGW64 ~/workspace/OpenCartV121 (master)

$

**3.ADDING files/folders to the stagging/indexing**

$ git status

$ git add -A🡪 adds all files

If you want to add only particular file/folder🡪 git add foldername/filename

$ git add \*.java

**4.commit the code into local(git) repo**

git commit -m “commit message”

**5.connect local repo to remote repo**

git remote add origin “https://github.com/Kavya-Bavandla/OpenCartV121.git”

**6.push code into remote repo**

git push origin master

if you get credentials to enter the token , got o profile githib-🡪settings-dev settings-🡪 create token -> copy the token number and provide in token text box

Day-2 (you want to add and del some changes)

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

**Day-3 (some teammember has added new file into git, you want to get file into local)**

Pull the files from remote repo:

git pull “url”

**Day-4 (remote repo already present , you want to clone it into workspace)**

git clone “url”

(here url must be taken from github ..go to the project which you want to clone, and click on code(green button)🡪 get the url)

In your file explorer, choose the path you want to create the project : from there open the git bash

After you clone, you can import into eclipse

**13) Run Tests using Jenkins**