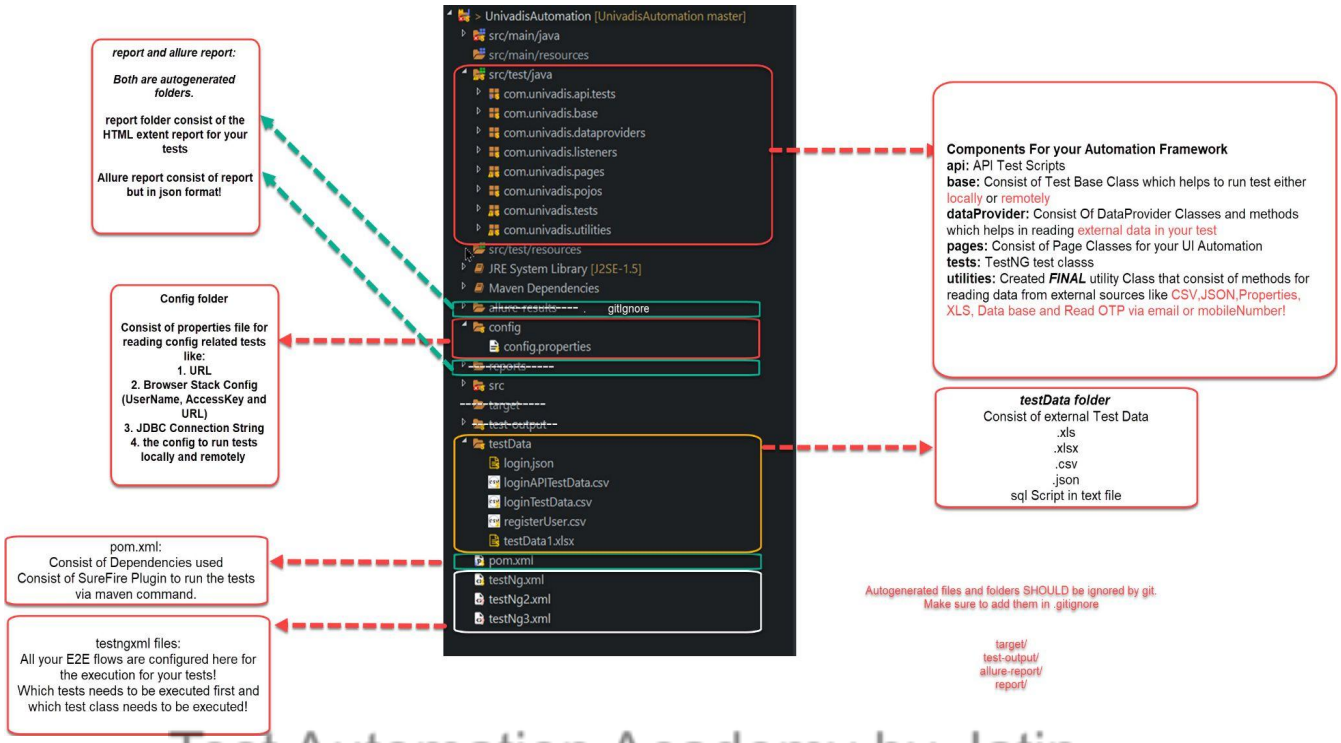
Explain your Automation Framework in Detail??

Explain your Automation Framework in Detail?? 1
Component Diagram 1
Script to use while Explaining 2
Dependencies 3
Page Classes Component: 5
Rest Assured API Component: 7
Data Providers Component 8
Utilities Component 8
Reporting Component 8
Step1: 9
Step 2: 11
Step 3: 11
Allure Report works on Annotation Based just like testNG: 12
Sample 12
Configuration Component 12
Various Exceptions Faced in your Automation Project? 14
What are challenges faced in your Automation Project?? 16
Synchronization Issue of Selenium WebDriver: 16
Automating Angular website or React Website with Selenium and Java?? 16
Maintaining the Code Quality of the Test Scripts written by the tester? 16
Making the abstraction of Selenium so junior testers dont have to struggle in learning Selenium. 16

Component Diagram



Link to open the image : Automation Framework Component.jpg

Script to use while Explaining

*The automation framework that I have created is a **maven project** and all the dependencies are managed in **pom.xml.***

We are using Maven for managing the external dependencies like Rest Assured, Selenium WebDriver and others.

The dependencies used by me for the project:

Dependencies

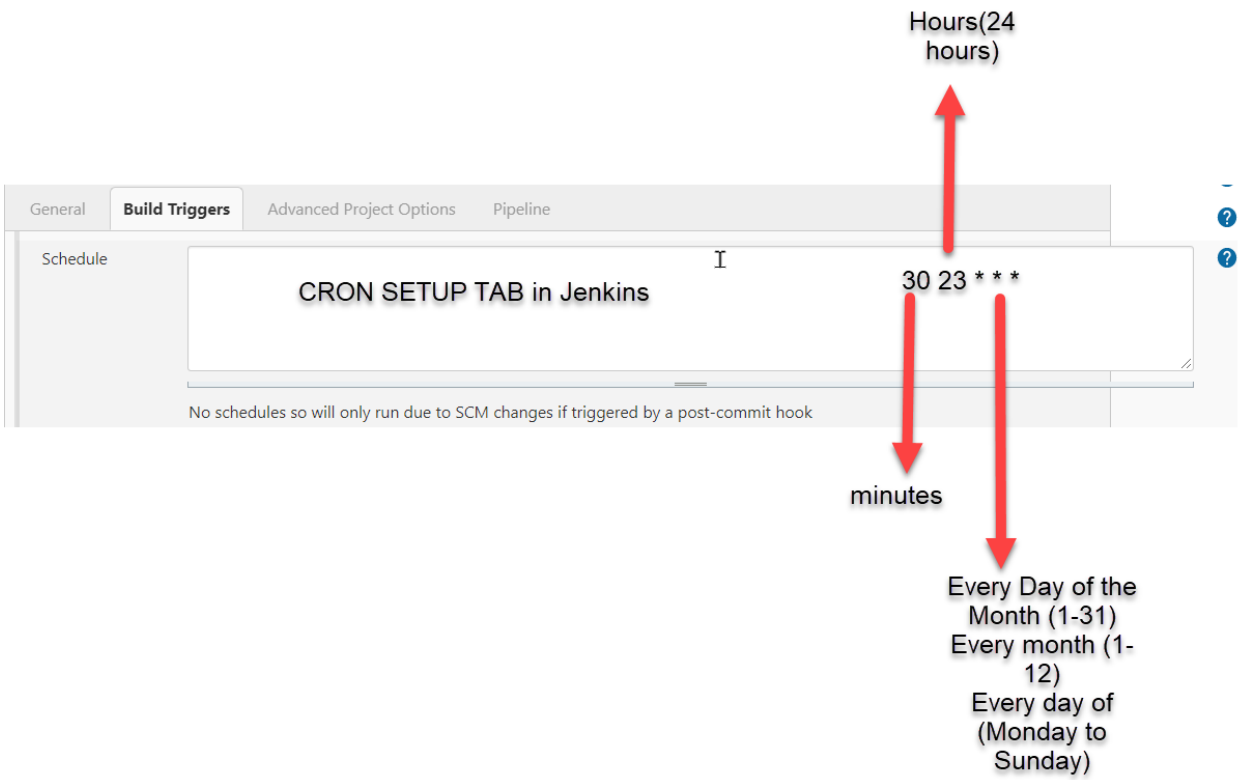
Dependencies Name	Version	Reason
rest-assured	4.4.0	Make API Request in Java
json-schema-validator	4.4.0	Validating JSON Schema
gson	2.8.7	Create POJO Classes for API Request and Response
selenium-java	4.X.X	Perform UI Automation
opencsv	5.5.1	Read CSV file in Java
testng	6.14.3	Testing Framework for Java
extentreports	3.1.5	Generate HTML Report for emailing purpose
Apache commons-io	2.5	Perform create and delete folders and files easily in Java
allure	2.8.1	Create a reporting Dashboard
JDBC Driver My SQL connector	8.0.26	Make Database Operation in Java,

		To execute SQL queries using Java and is required for E2E Testing
JSON PATH	4.4.0	Use to retrieve a Value from Json file or json response! JsonPath jsonPath = new JsonPath(File/ JsonContent in string) jsonPath.getString(
Hamcrest	1.13	Matchers

The source code management is GIT SCM and the remote git repository is Gitlab. We have also integrated with Jenkins CI {MAVEN PROJECT} - clean install command to perform Continuous UI Testing (Selenium WebDriver, WebDriver Manager and Test NG) and API Testing (Rest Assured, Gson, ***JSON PATH*** *Test NG)*

Once the Test has been executed Jenkins sends the email of the report to the team.

Also the CRON Job setup is done so that the regression tests are executed every night around 23:30 so that next morning the team can validate report before the Stand up meeting.



The Framework that I have created is Hybrid Framework which focuses on the page object design pattern.

The framework can read data for testing from various sources like **csv, json, excel file** and is passed to the @test methods with the help of dataproviders

The reports are generated with Extent report automatically which captures information of which tests are passed, which tests have failed and the screenshot of the failed tests.

We wanted to create the framework in such a way that even professionals who have no experience in Selenium can contribute to writing test scripts and for that we have created a browser utility which creates a layer of abstraction between selenium scripts. Hence the professionals just need to know my custom methods and don't really need to worry about any selenium methods or scripts like waits, selenium exceptions, or wd,find element scripts.

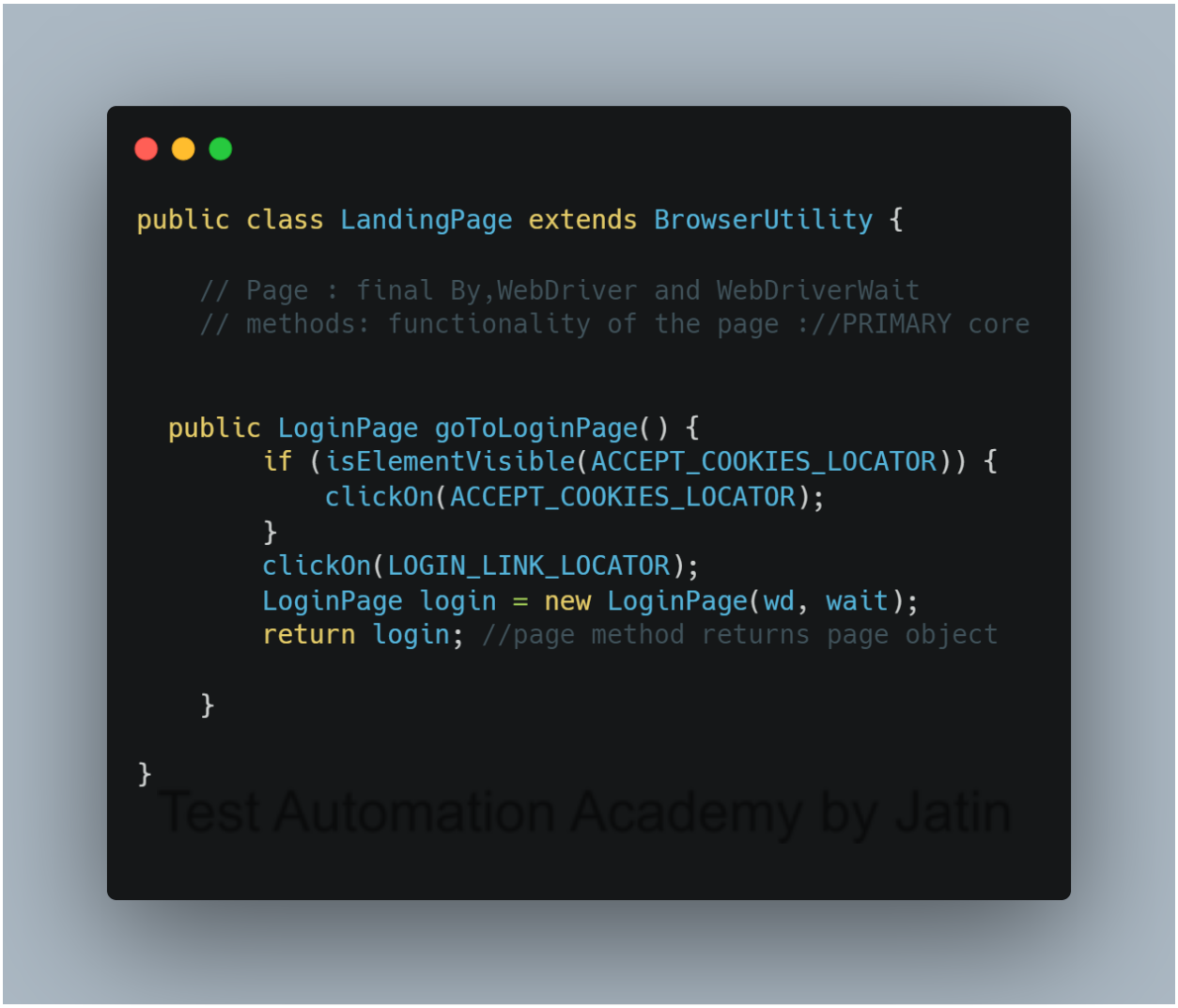
Page Classes Component:

All the Page Classes extends BrowserUtility (which is an abstract class) as a parent Class.
The Page Classes consist of 3 major class variables:

- **By** Type Variables which are the Locators: which are ***private static final*** so that only one memory be allocated to them!
- WebDriver reference Variable for the browser session
- WebDriverWait Variable for using explicit Wait.

```
public class LandingPage extends BrowserUtility {

    // Page : final By,WebDriver and WebDriverWait
    // methods: functionality of the page ://PRIMARY core

    private static final By LOGIN_LINK_LOCATOR = By.xpath("//a[@title='Login']");
    private static final By ACCEPT_COOKIES_LOCATOR = By.id("onetrust-accept-btn-handler");
    private WebDriver wd;
    private WebDriverWait wait;

    public LandingPage(WebDriver wd, WebDriverWait wait) {
        super(wd, wait);// BrowserUtility Constructor
        this.wd = wd;
        this.wait = wait;
    }
```


All the Functionalities of the classes are public in nature!



All Page Methods written the Page Objects to where they are called in Tests

Read about Page Factory

```java
public class LandingPage extends BrowserUtility {

    // Page : final By,WebDriver and WebDriverWait
    // methods: functionality of the page ://PRIMARY core

    private static final By LOGIN_LINK_LOCATOR = By.xpath("//a[@title='Login']");
    private static final By ACCEPT_COOKIES_LOCATOR = By.id("onetrust-accept-btn-handler");
    private WebDriver wd;
    private WebDriverWait wait;

    public LandingPage(WebDriver wd, WebDriverWait wait) {
        super(wd, wait);// BrowserUtility Constructor
        this.wd = wd;
        this.wait = wait;
    }

  public LoginPage goToLoginPage() {
        if (isElementVisible(ACCEPT_COOKIES_LOCATOR)) {
            clickOn(ACCEPT_COOKIES_LOCATOR);
        }
        clickOn(LOGIN_LINK_LOCATOR);
        LoginPage login = new LoginPage(wd, wait);
        return login; //page method returns page object

    }

}
```


Rest Assured API Component:

For Rest Assured Tests we have used the BDD Style scripting and the have use REST Assured Static import to use it more effectively!

io.restassured.RestAssured.*

How to pass Multiple Headers into your Rest ASsured code:

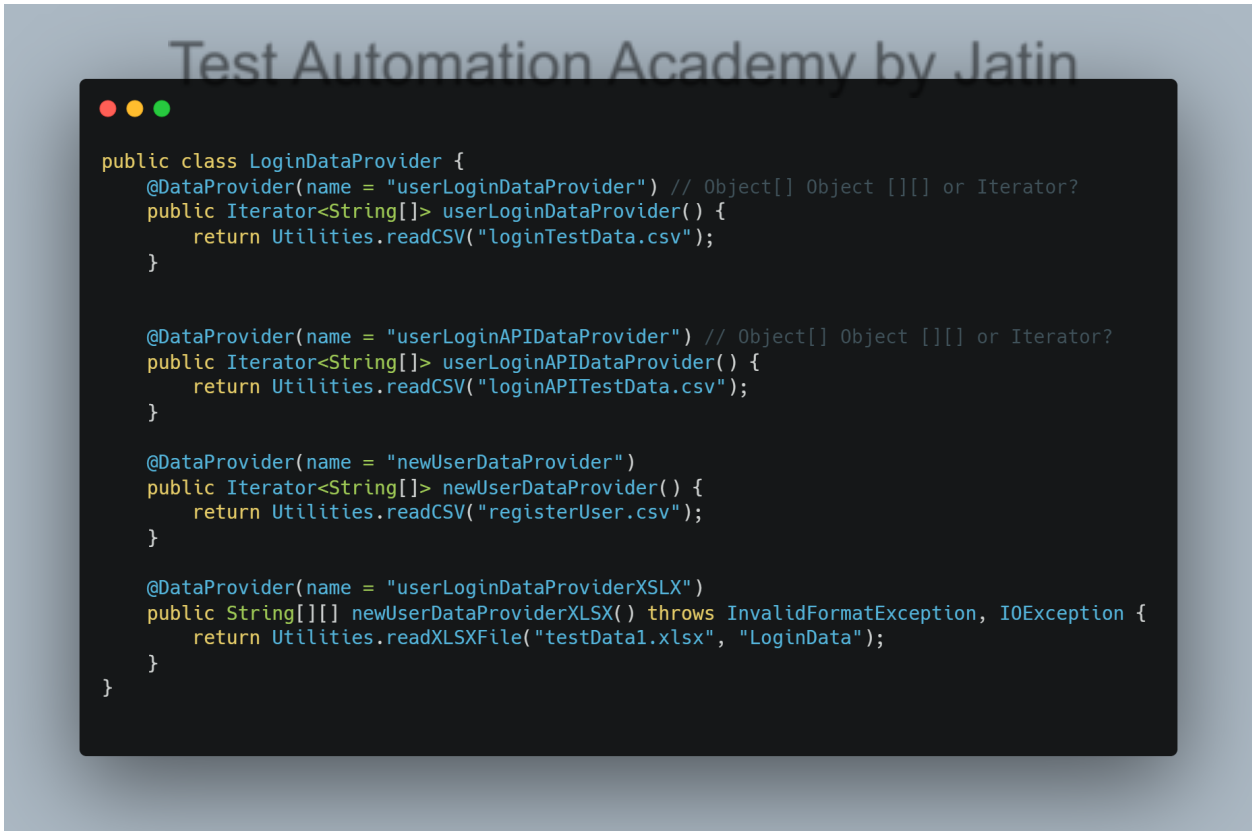
```
ArrayList<Header> myListHeader = new ArrayList<Header>();
myListHeader.add(new Header("Content-type", "application/json"));
myListHeader.add(new Header("Authorization", LoginRequest.token));

Headers myMultipleHeader = new Headers(myListHeader);
request.headers(myMultipleHeader);
```

Data Providers Component

For Passing the request body in the API request we have used the POJO classes so that it can be integrated with the dataproviders.

The dataproviders methods return data in either Object[][] or Iterator<> or Object[]

DataProviders are internally calling the static methods of the final **Utilities** class method



Utilities Component

Utilities Class is a FINAL class which consist of all common methods that are used in the tests or in the page classes like reading data from Excel, CSV, Properties,Json file,JDBC, Reading OTP from email or from SMS.
In Utitlities Class we have used the ***apache poi, open csv, properties and JDBC library***

Reporting Component

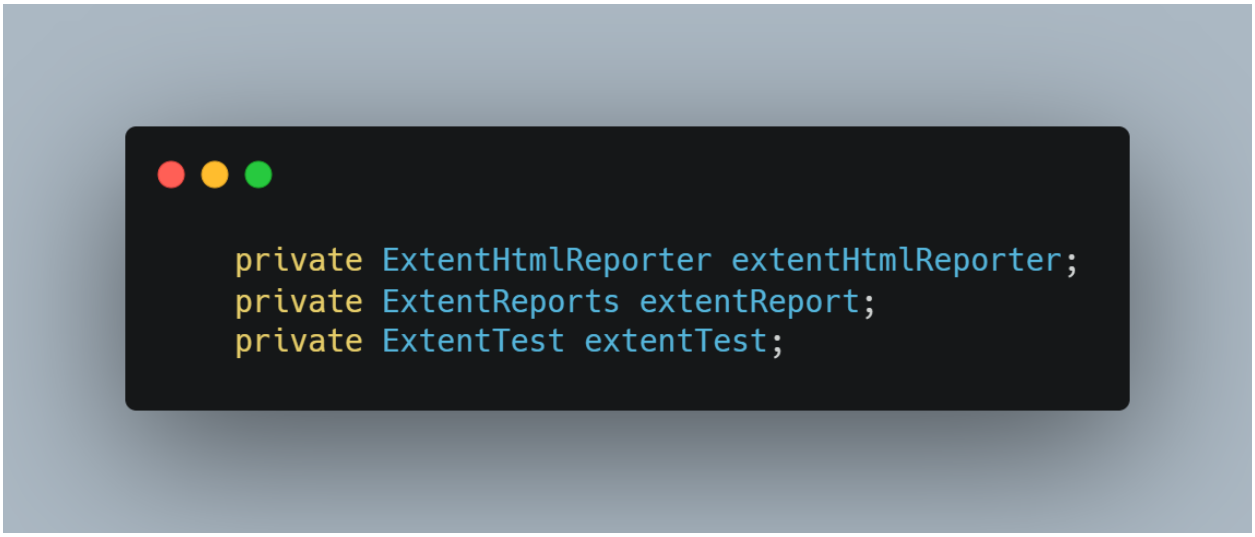
For Reporting we have used Extent Report and Allure Report and Extent Report is configured in the Listeners component of the framework.

Why?

Because in the report we want to know which test has been passed, failed or skipped. When the Test started, completed etc!

This can be done only with the help of ITestListener Interface

Extent Report
For Extent Reporting we have used 3 important Class Reference:



Step1:

```
public void onStart(ITestContext context) { // When the test Suite is being started!!
        System.out.println("************************ Starting my Test Class
************************");
        File f1 = new File(System.getProperty("user.dir") + "/reports");
        if (Utilities.readConfig("PRESERVE_REPORT").equalsIgnoreCase("FALSE")) {

            if (f1.exists()) {
                try {
                    FileUtils.deleteDirectory(f1);
                    f1.mkdir();
                } catch (IOException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                }
            } else {
                f1.mkdir();
            }
        }

        Date date = new Date();
        SimpleDateFormat dateFormat = new SimpleDateFormat("dd-MM-yyyy-HH-mm");
        String dateString = dateFormat.format(date);

        File f = new File(System.getProperty("user.dir") + "//reports//report-" + dateString +
".html");
        extentHtmlReporter = new ExtentHtmlReporter(f);
        extentHtmlReporter.config().setReportName(Utilities.readConfig("REPORT_NAME"));
        extentHtmlReporter.config().setTheme(Theme.DARK);
        extentReport = new ExtentReports();
        extentReport.attachReporter(extentHtmlReporter);

    }
```

Step 2:

```
public void onTestStart(ITestResult result) {
        // TODO Auto-generated method stub
        // Just before the @Test Annotation
        System.out.println("************************* Starting my Test Method
************************"
                + result.getMethod().getMethodName() + " -----  " +
result.getMethod().getDescription());
        extentTest = extentReport.createTest(result.getMethod().getMethodName());
    }

    public void onTestSuccess(ITestResult result) {
        // TODO Auto-generated method stub
        System.out.println("************************* Test Method Passed!! ************************");
        extentTest.pass(result.getMethod().getDescription());
    }
```

Step 3:

```
public void onFinish(ITestContext context) { // When the test Suite is being finished!!!!
        // TODO Auto-generated method stub
        System.out.println("************************* Finished my Test
Class*************************");
        extentReport.flush();
        // Write the code to convert to pdf https://github.com/danfickle/openhtmltopdf
    }
```

Allure Report works on Annotation Based just like testNG:

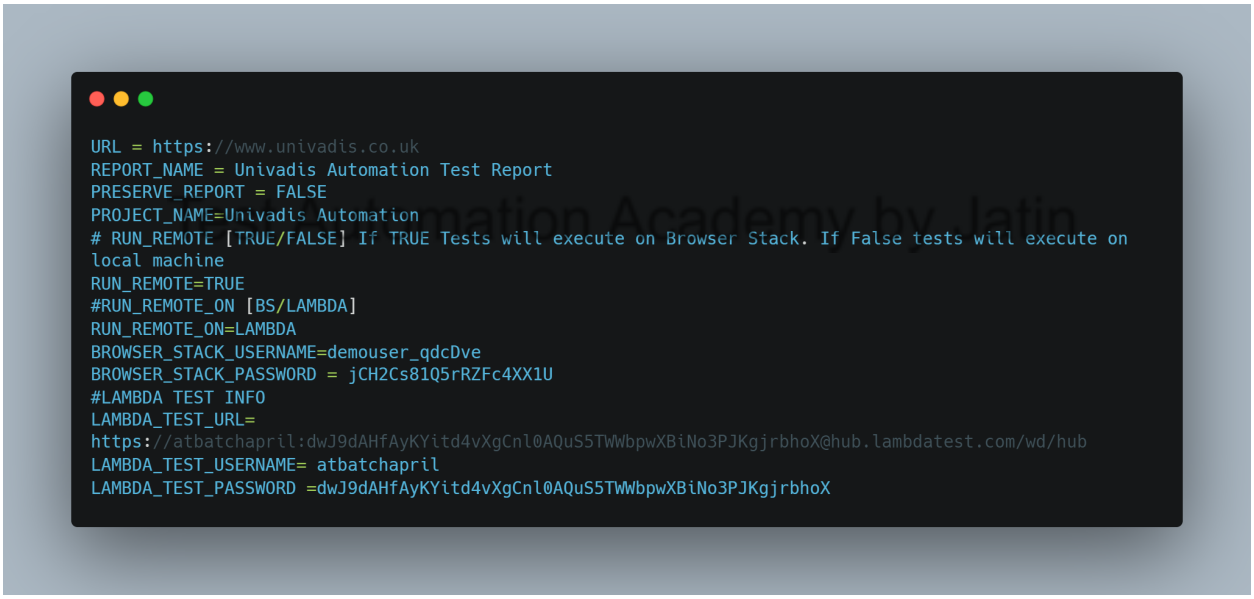
Sample

```
@Story("1452-Create Test Script for Login Verification")
    @Description("Verify if the login works or not!")
    @Severity(SeverityLevel.NORMAL)
    @Test(description = "Verify if the login works or not!", groups = { "e2e", "sanity",
            "smoke" }, priority = 1, dataProviderClass =
com.univadis.dataproviders.LoginDataProvider.class, dataProvider = "userLoginDataProviderXSLX")
    public void verifyLoginTest(String u, String p) {
        HomePage homePage = loginPage.doLogin(u, p);
        Assert.assertEquals(homePage.getUserName(), "Qdre Sharma");
    }
```

Configuration Component

We have kept a central point for the configuration of the project which decide how and where the test will be executed, to preserve reports or not, naming of the report and etc!

It also consist of details of the Remote Testing Config for Browser Stack/Lambda Test/ Selenium Grid



Various Exceptions Faced in your Automation Project?

Exception	type	Reason
File Not Found	Java	FileNotFoundException occurs at runtime indication java cannot find a file or directory
Array Index Out of Bound	Java	if a program tries to access an array index that is negative, greater than, or equal to the length of the arra
Null Pointer Exception	Java	When you forgot to instantiate the object of the class
Number Format Exception	Java	The NumberFormatException is thrown when we try to convert a string into a numeric value
NoSuchElementException	Selenium	which means that element doesn't exists on the website.
ElementClickInterceptedException	Selenium	The Element Click command could not be completed because the element receiving the events is obscuring the element that was requested clicked
NoAlertPresentException	Selenium	Thrown when switching to no presented alert.
NoSuchAttributeException	Selenium	Thrown when the attribute of element could not be found.
TimeoutException	Selenium	Thrown when a command does not complete in enough time.

StaleElementReferenceException	Selenium	Thrown when a reference to an element is now “stale”.
ElementNotVisibleException	Selenium	Thrown when an element is present on the DOM, but it is not visible, and so is not able to be interacted with.
HttpResponseException	Rest Assured	most exceptions are translated into an HTTP response with status code 500, Internal Server Error. But if there is some config issue and the response coming from API is not as per the standard rules then will throw HttpResponseException
InvalidFormatException	ApachePOI	HSSF .xls and XSSF deals .xlsx
CSVFormatException	OpenCSV	Format the .csv file is wrong! Meaning missing a comma then

What are challenges faced in your Automation Project??

Synchronization Issue of Selenium WebDriver:

Instead of using wd.findElement we used WebDriver wait

wd.findElement ----------> wait.until(ExpectedConditions.)

Automating Angular website or React Website with Selenium and Java??

Selenium is agnostic of Angular or react element so we need to use explicit wait to make sure that we give enough time to scripts to work on finding the elements. Also we had to rely heavily on xpath and explicit waits that make our scripts slow!!

We decided to take this drawback as most of the team member (tester) were comfortable only in using Selenium Java combination.

Maintaining the Code Quality of the Test Scripts written by the tester?

So we introduce SONAR Qube and integrated it with SONAR LINT in eclipse and made sure the entire team adher to the coding standard decided by the team lead!! This resulted in better coding standard across the team, maintainable code, clean code for dev code and automation scripts!

Making the abstraction of Selenium so junior testers dont have to struggle in learning Selenium.

With BrowserUtilitiy class the team only need to think of finding locators and using the custom methods created by me! So the learning curve was pretty less!! And the framework was easily used by all the team members!

These Notes are ONLY for the setup and configuration on the server!! I would advice you all to do it 3 to 4 times by yourself.

There is no need to by heart the commands as after sometime you will remember them!

For Pratice create the server on AWS or on DO and just execute the commands.
Dont copy PASTE!

Setting Up Jenkins on the Centos Server

Setup Of Jenkins

In order to run Jenkins on the server we need to install Java 8 on the server.

Jenkins installation Path /home/coffeeDigital/.jenkins On STG2 server

Jenkins BackUp: /home/coffeeDigital/.jenkins_backup On STG2 server

Make Sure Java is installed on the server:

```
java -version



[coffeeDigital@ip-172-31-18-138 ~]$ java -version

openjdk version "1.8.0_275"

OpenJDK Runtime Environment (build 1.8.0_275-b01)

OpenJDK 64-Bit Server VM (build 25.275-b01, mixed mode)

[coffeeDigital@ip-172-31-18-138 ~]$
```

If Java is not installed.

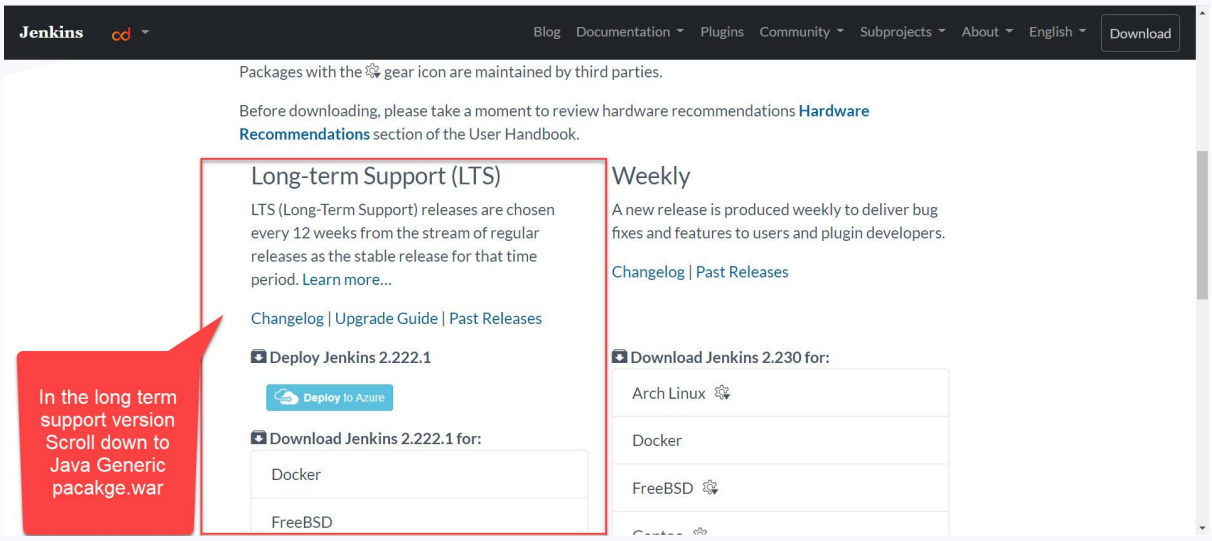
Install Java using the command

```
sudo yum install java-1.8.0-openjdk
```

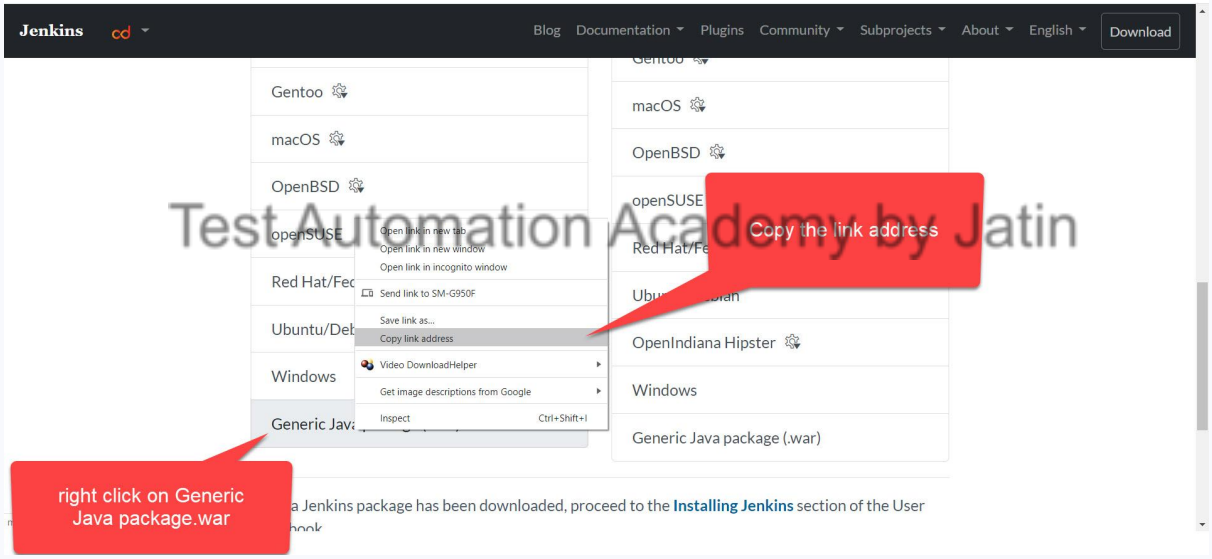
Step 1. Make directory Jenkins

Step 2: In your browser search https://jenkins.io/download/

Scroll down to the Java Generic Packages.war and right click and copy the link address



Go to http://jenkins.io/download/



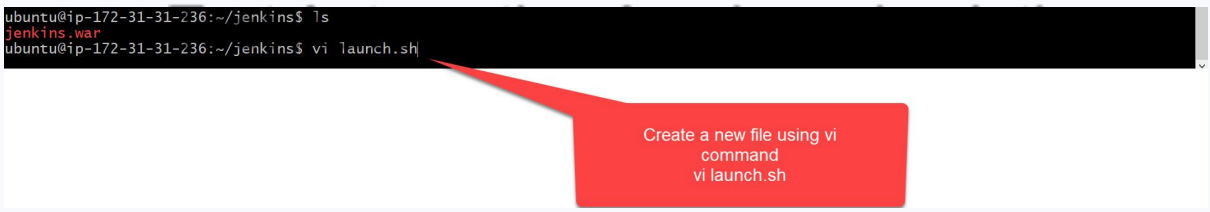
Step 3: Go to your terminal write the wget and paste the link

To paste the link just do a right-click

Note if wget command is not found the please install wget on centos

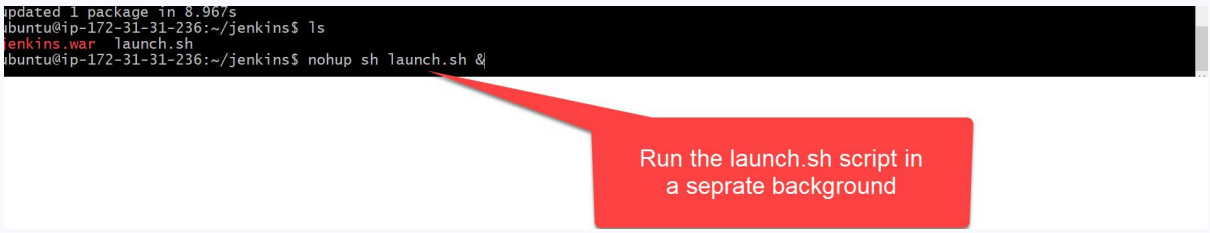
using **sudo yum install wget**

Step 4:Creating a launch script



Open the file launch.sh in vi mode and write **java -jar jenkins.war --httpPort=9090**

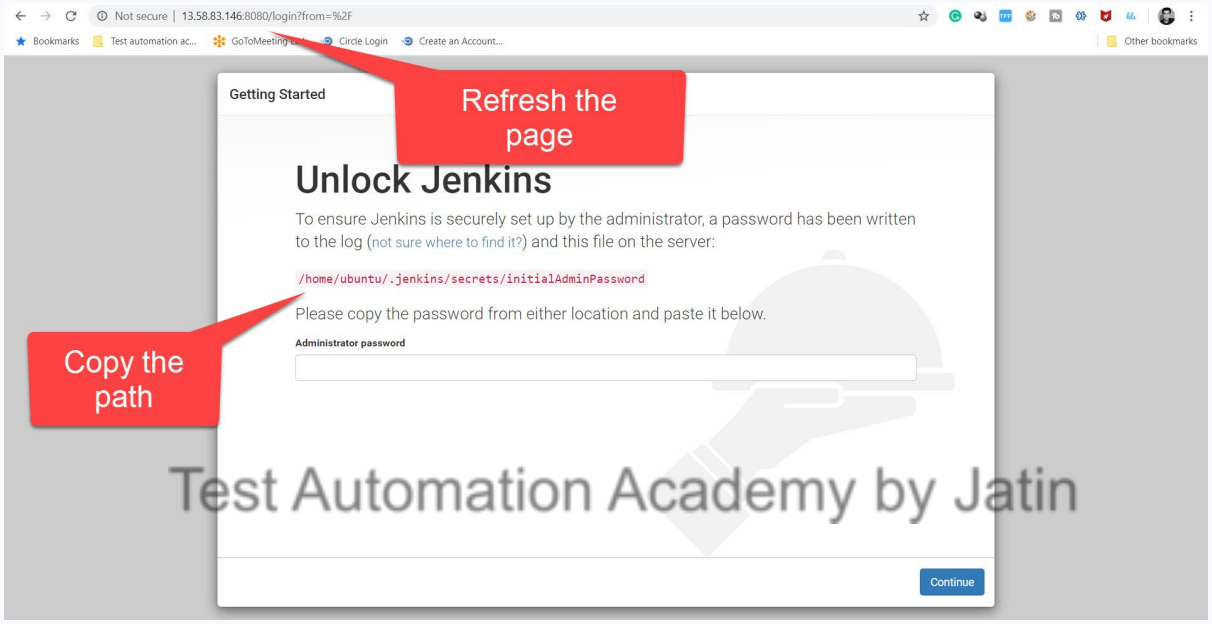
Step 5: Running Jenkins in nohup mode



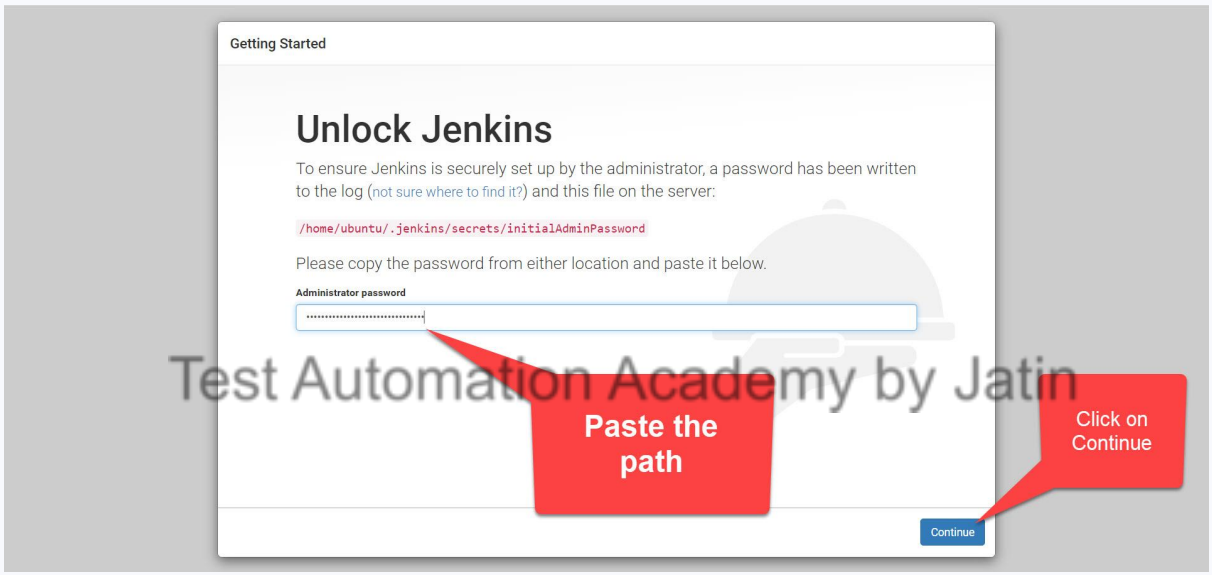
Step 6: Accessing the Jenkins on port **9090**

(Make sure that port 9090 is unblocked from the AWS Security Group) Take help of Mario for this.

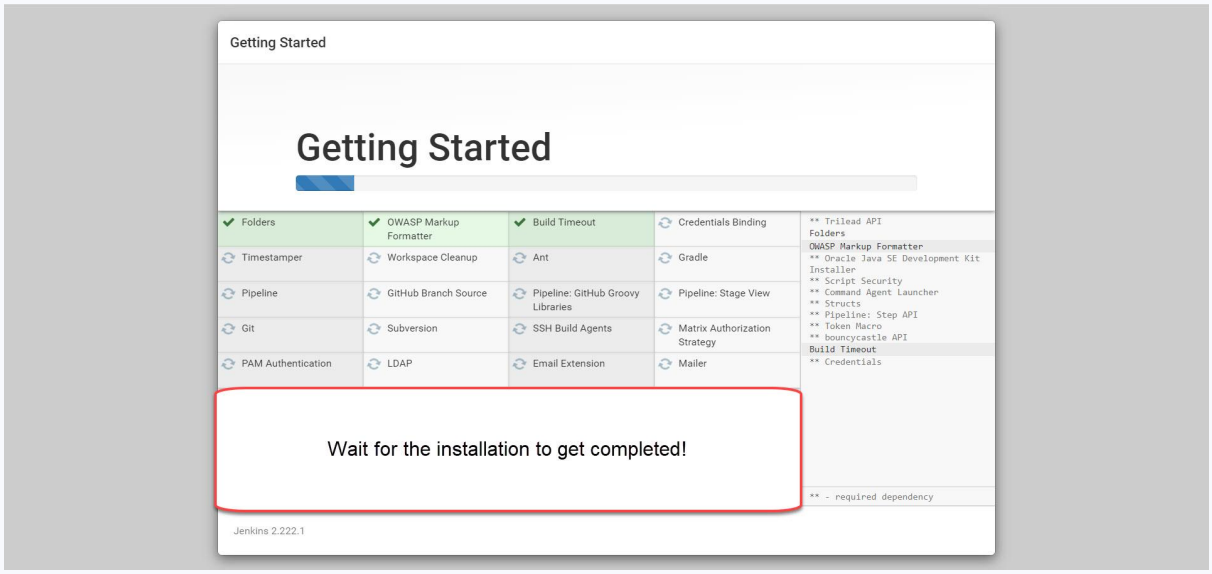
Step 7: Accessing the Jenkins again

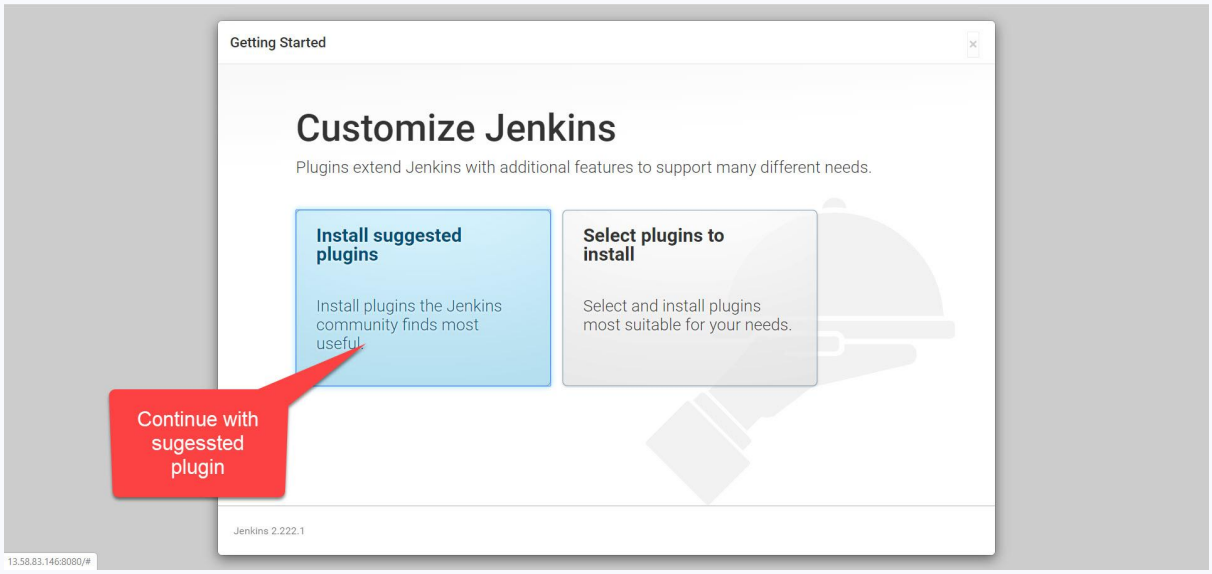


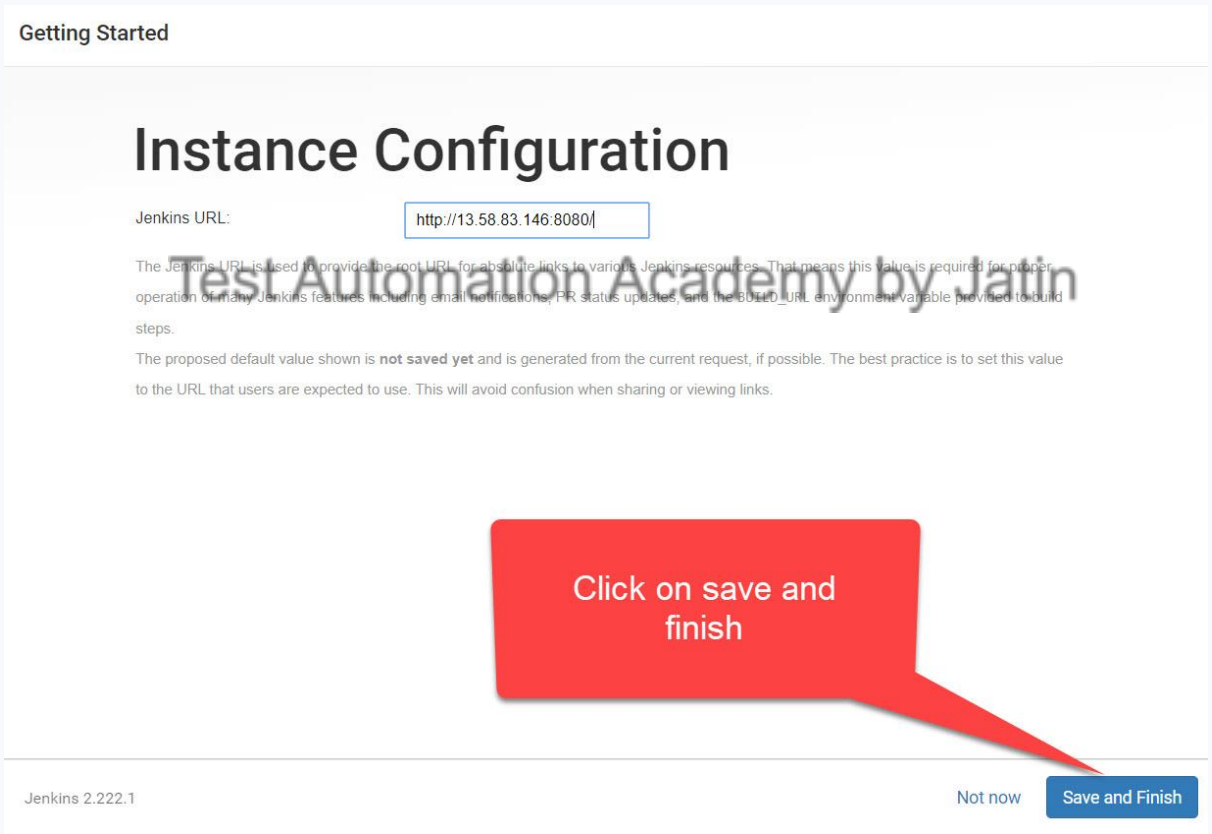
Step 8: One time Password



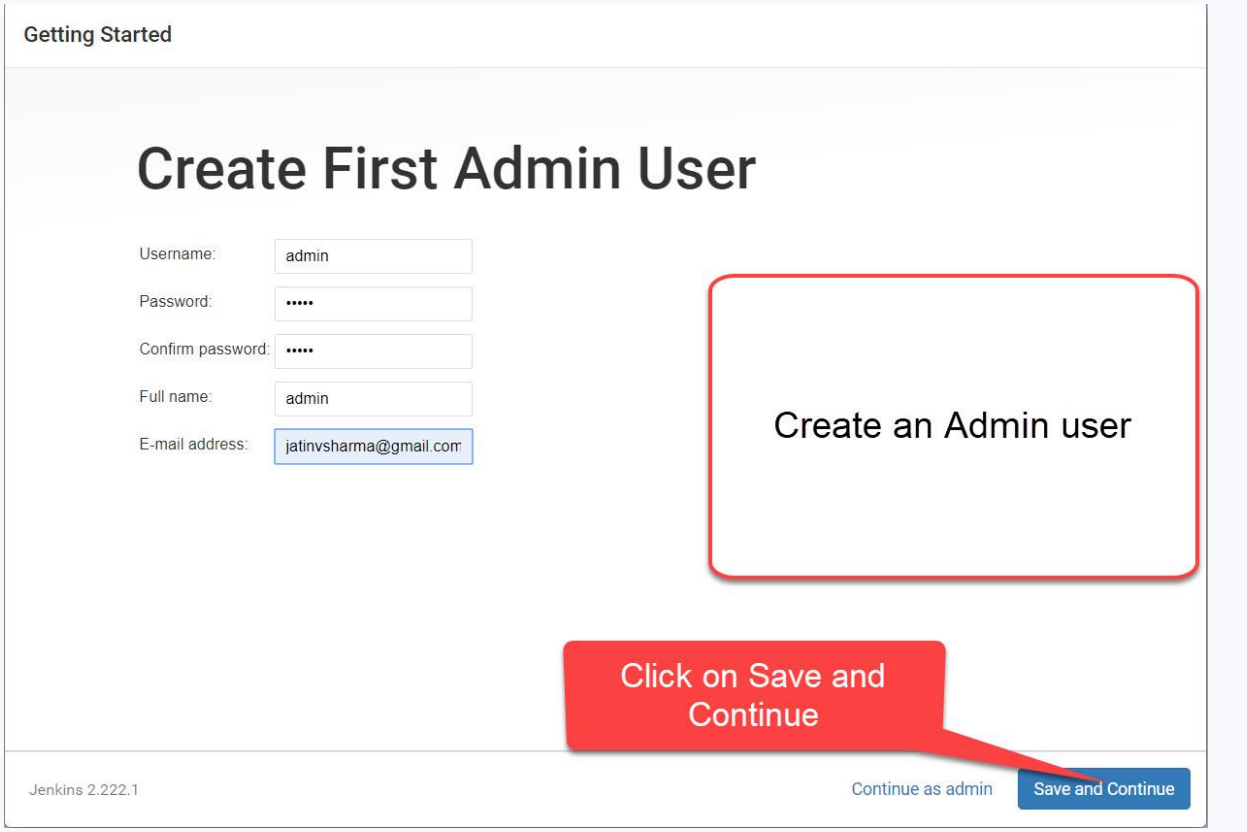
Step 9: Installing necessary plugins

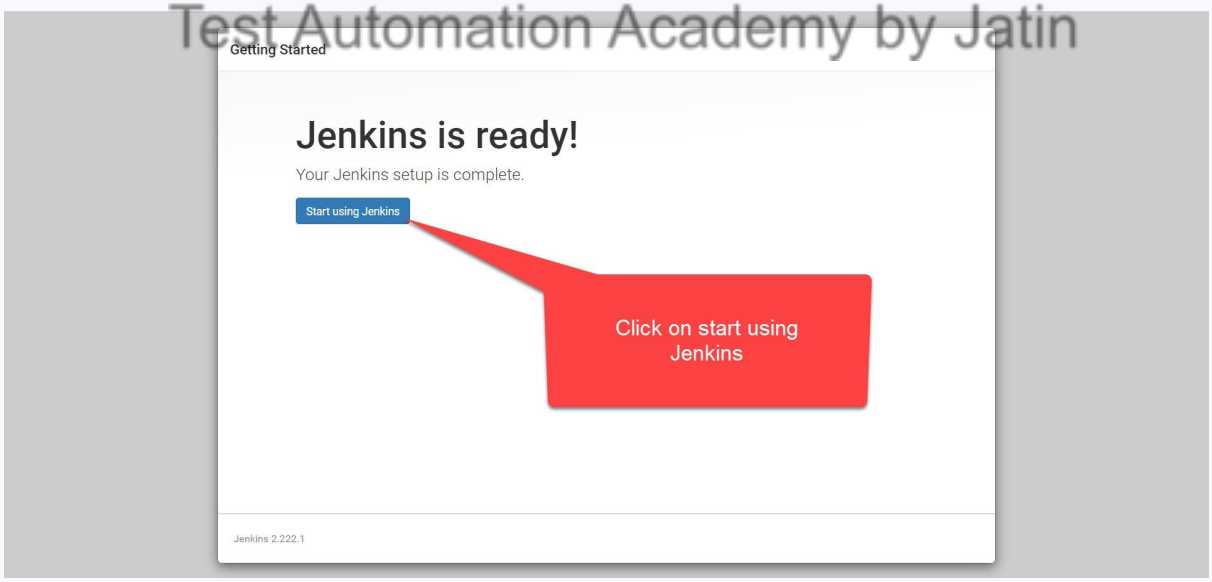


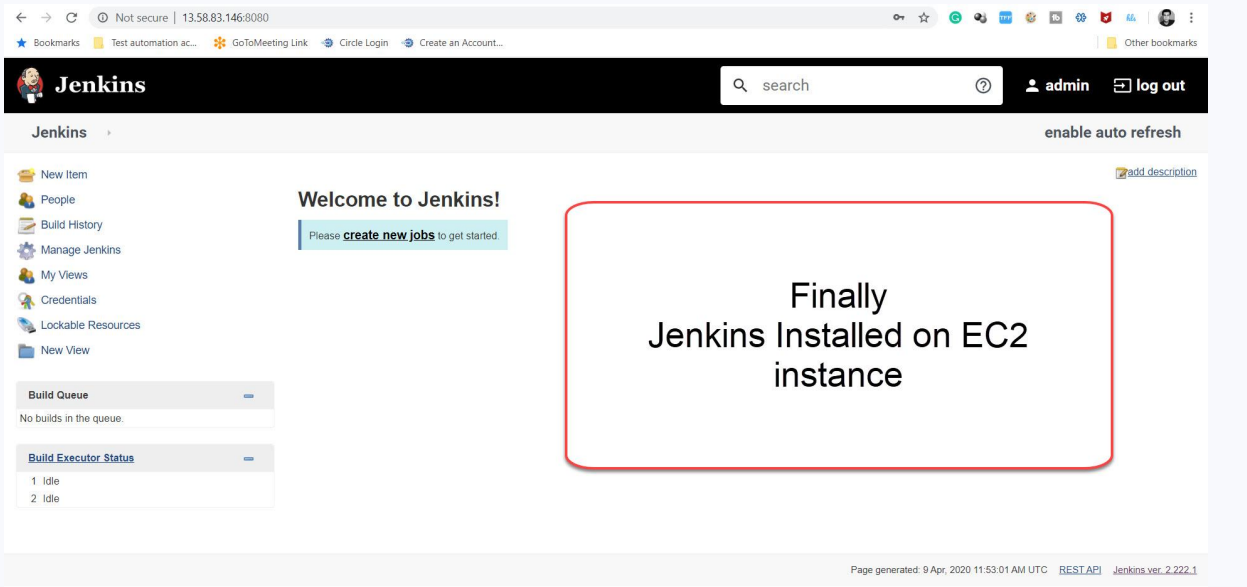




Step 10: Creating an admin user







Setting up Email Configuration:

Go to Manage Jenkins -> Configure System and scroll down to Extented Email Notification Plugin option and fill the details:

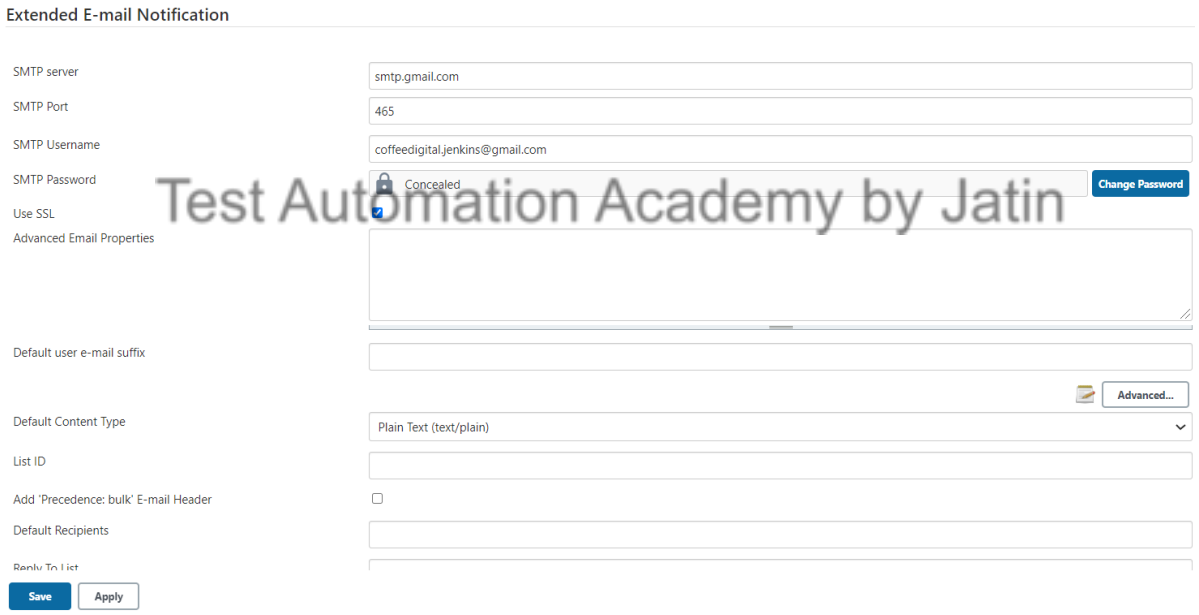
We are using gmail for sending emails via jenkins

smtp server: **smtp.gmail.com**

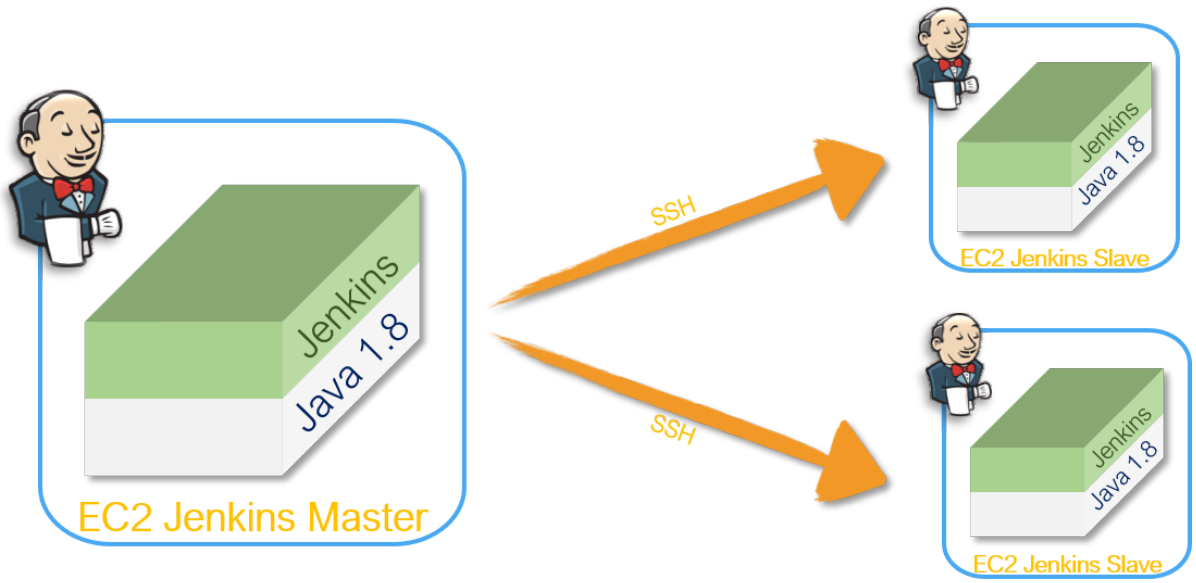
smtp port : **465**

smtp user name: coffeedigital.jenkins@gmail.com

password: **Qweqwe`1**



Setting up Jenkins Node



Configure Jenkins Slaves on AWS EC2

Jenkins is a self-contained Java-based program, ready to run out-of-the-box, with packages for Windows, Mac OS X and other Unix-like operating systems. As an extensible automation server, Jenkins can be used as a simple CI server or turned into the continuous delivery hub for any project.

Follow this Video in https://www.youtube.com/watch?v=hwrYURP4O2k

Prerequisites

1. Jenkins Master Running
2. EC2 RHEL 7.x Instance - for Slave Node
    - With Internet Access
    - Java v1.8.x

Install Java

We will be using open java for our demo, Get latest version from http://openjdk.java.net/install/. Also configure the default JAVA_HOME path

yum install java-1.8*

#yum -y install java-1.8.0-openjdk

Setup Jenkins Slave [DEV/STG2]

Create user and add the user to wheel group

useradd jenkins-slave-01

Create SSH Keys

sudo su - jenkins-slave-01

ssh-keygen -t rsa -N "" -f /home/jenkins-slave-01/.ssh/id_rsa

The private and public keys will be created at these locations `/home/jenkins-slave-01/.ssh/id_rsa` and `/home/jenkins-slave-01/.ssh/id_rsa.pub`

cd .ssh

cat id_rsa.pub > authorized_keys

chmod 700 authorized_keys Configuration on Master

Copy the slave node's public key[id_rsa.pub] to Master Node's known_hosts file

mkdir -p /var/lib/jenkins/.ssh

cd /var/lib/jenkins/.ssh touch

ssh-keyscan -H SLAVE-NODE-IP-OR-HOSTNAME >>/var/lib/jenkins/.ssh/known_hosts

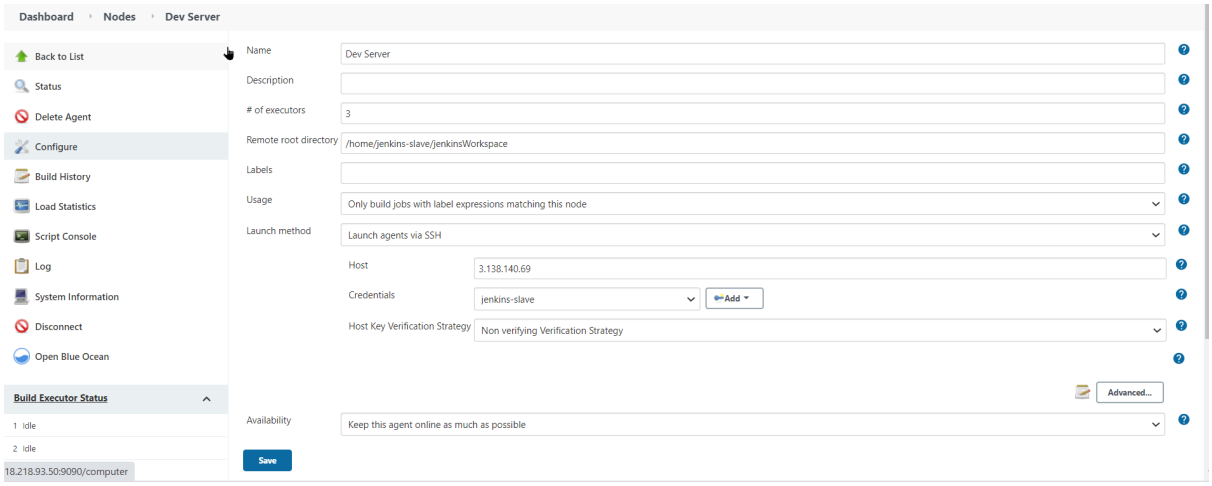
 # ssh-keyscan -H 172.31.38.42 >>/var/lib/jenkins/.ssh/known_hosts

chown jenkins:jenkins known_hosts

chmod 700 known_hosts

Configure the Slave using Manage Jenkins

Configure the node as shown here Manage Jenkins > Manage Nodes > New Node



Name: Dev Server (Name for the Node)

Workspace /home/jenkins-slave/jenkinsWorkspace

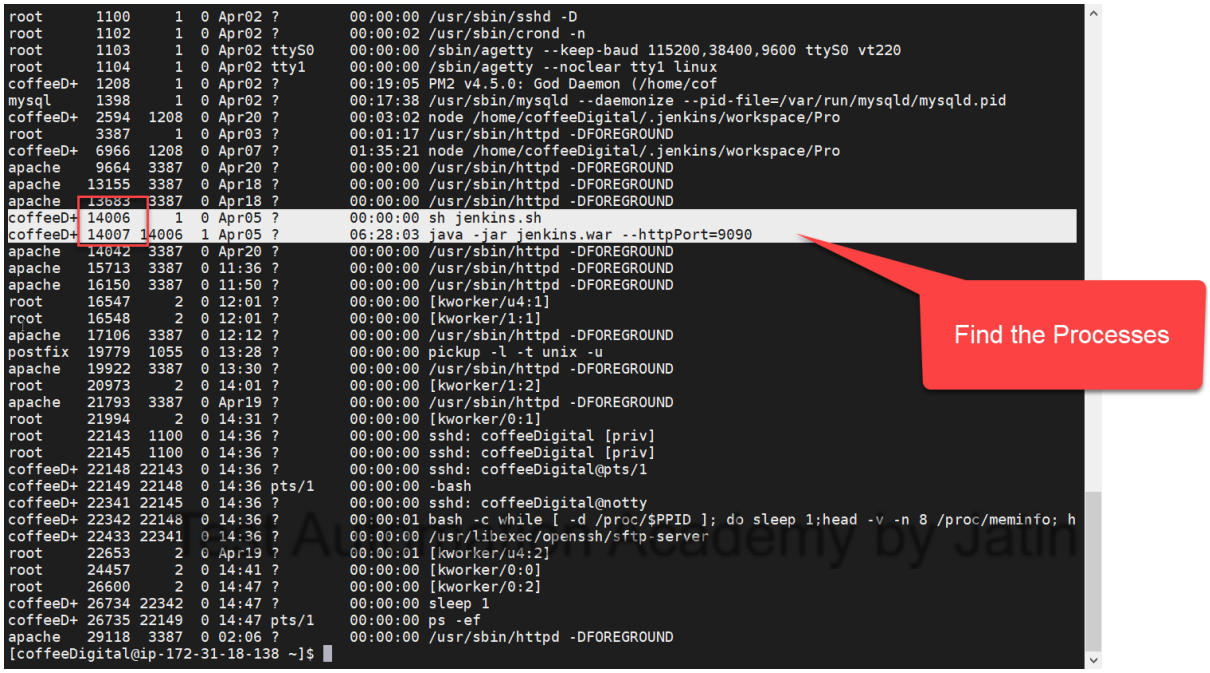
Host Key Verification Strategy - Non Verification Strategy

Restarting Jenkins Master Node:

Step 1: Execute command **ps -ef**

and locate the process for Jenkins as shown in the figure!

[If the processes is not available. This means that Jenkins is not running]



Step 2

In case you want to restart the Jenkins. You need to first Kill both the processes. Note the process ids will be different in your instance

To the kill the Jenkins process you have execute the command

kill -9 14006

kill -9 14007

Now the Processes has be deleted/killed. This means that Jenkins has stopped working and can be restarted!

For the Jenkins installation, We have kept the Jenkins script in the jenkins directory

the path for the Jenkins directory

[coffeeDigital@ip-172-31-18-138 jenkins]$ pwd

/home/coffeeDigital/jenkins

In Order to start the jenkins run the jenkins.sh file.

[coffeeDigital@ip-172-31-18-138 jenkins]$ ls

hs_err_pid2364.log hs_err_pid27011.log hs_err_pid29892.log **jenkins.sh** jenkins.war nohup.out

The command to execute the Jenkins.sh file is

nohup sh jenkins.sh &

Once executed the command. Jenkins will restart successfully!

Jenkins File:

In order to automatically deploy code on the server via jenkins we are using JenkinsFile that is read by Jenkins to executes the steps for the deployment:

These Jenkins file are part of the git repo of the project

Currently there are 2 Jenkins file added in the git repo:

jenkins-dev: This file is required in the Ecom instance

jenkinsFile: This file is required in the Stg2 instance

Both the files are same but only where the code is going to be deployed is different.

```
pipeline {

   agent {
/* DOnt Include the Text in red in Jenkins file
The only thing that needs to be changed for the deployment for the Stg2 environment is the agent name .
If the agent name is master. It will be deployed to the STG2 */
      label 'Dev Server'
   }

   options {
/* DOnt Include the Text in red in Jenkins file
      skipDefaultCheckout true is going to do full clone from scratch

*/
      skipDefaultCheckout true

   }

   stages {

      stage('Install Dependencies') {

         steps {
/* DOnt Include the Text in red in Jenkins file
      cleanWS() Deletes the Old workspace and Creates a new one
*/
            cleanWs()

            checkout scm

            sh 'echo "********INSTALLING DEPENDENCIES*********"'

            sh 'rm -rf node_modules'
            sh 'sudo npm install'

         }

      }

      stage('Build') {

         steps {

            sh 'echo "********Building the Project*********"'
/* DOnt Include the Text in red in Jenkins file
            sh 'npm run build:devssr ' ----> this is for ecom
       sh 'npm run build:uatssr  ------> this is for Stg2'


*/
            sh 'npm run build:devssr '

            dir('dist') {

               sh 'pwd'

               sh 'ls'

               dir('pronto-insurance') {

                  sh 'pwd'

                  sh 'ls'

                  sh 'sudo mkdir -p server/dist/pronto-insurance'

                  sh 'sudo cp -R browser/ server/dist/pronto-insurance'

                  dir('server') {

                     sh 'pm2 stop main.js'

                     sh 'pm2 start main.js --exp-backoff-restart-delay=100'

                  }

               }

            }
```

The only thing that needs to ne changes for the deployment for the Stg2 environment is the agent name.

If the agent name is master. It will be deployed to the STG2

Step for Installing all Dependencies:

Install Java on the server:

```
sudo yum install java-1.8.0-openjdk
```

Check the version of Java Installed on the Server:


```
[root@ip-172-31-19-139 liquibase]# java -version
```

Install Node.js 14

The installation on RHEL 8 is same as for RHEL 7 based Linux distributions. The only difference is that Node.js AppStream repository has been disabled on RHEL / CentOS 8 system by the setup script. If you ever want to install the AppStream version of Node.js, you'll need to enable it.

```
sudo yum install -y nodejs
```

The command will install both Node.js 14.x and npm. Version installed can be checked with below command:

```
$ node -v

v14.0.0
```

Check the NPM version

```
[root@ip-172-31-19-139 liquibase]# npm -v

6.14.10
```

Installing Angular js Globally

After installation of node.js and npm on your system, use the following commands to install the Angular CLI tool on your system.

```
npm install -g @angular/cli
```