**Lesson 06 Lesson-End Project**

**Continuous Integration with Selenium in Jenkins**

**Project Agenda:** As a DevOps engineer, you are required to perform continuous integration with Selenium and TestNG in Jenkins

**Description:** Testing is an integral part of SDLC and Agile. It is necessary to test every tool before getting into production environment to ensure that the tool functions as desired. To do so, it is necessary to integrate a tool with Jenkins in DevOps life cycle.

**Tools required:** Jenkins, TestNG, Selenium

**Pre-requisites:** You must have installed all the required tools and should have a Selenium and TestNG working.

**Expected Deliverables:**  Console output from Jenkins dashboard

**Steps to be followed:**

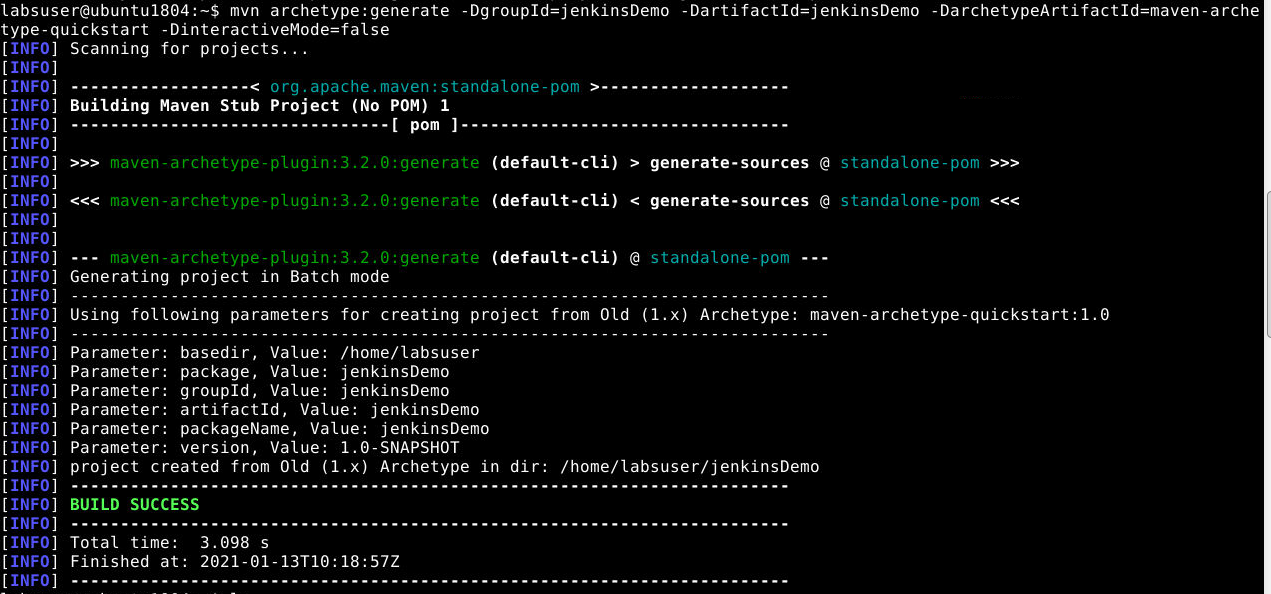
1. Logging in to Jenkins
2. Adding TestNG and Selenium dependencies and classes in Maven project
3. Creating a Jenkins job for Maven project
4. Building the Jenkins job

**Step 1: Logging in to Jenkins**

1.1 Open your browser and navigate to **localhost:8080**

1.2 Provide your username and password and click on **Login**

**Step 2: Adding Selenium and TestNG dependencies and classes in Maven project**

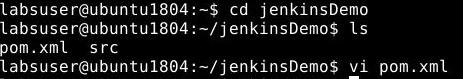
2.1 Create a maven project by executing the following command in the terminal of your lab:  
***mvn archetype:generate -DgroupId=jenkinsDemo -DartifactId=jenkinsDemo -DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=false***

2.2 Run the below commands to navigate inside the maven project:

***cd jenkinsDemo***

2.3 To update **pom.xml** file use the command given below:

***vi pom.xml***

****

2.4 Add the below code in the **<dependencies>** section of the **pom.xml** file of your Maven project:

***<!-- https://mvnrepository.com/artifact/org.testng/testng -->***

***<dependency>***

***<groupId>org.testng</groupId>***

***<artifactId>testng</artifactId>***

***<version>6.8.7</version>***

***<scope>test</scope>***

***</dependency>***

***<dependency>***

***<groupId>org.seleniumhq.selenium</groupId>***

***<artifactId>selenium-java</artifactId>***

***<version>3.10.0</version>***

***</dependency>***

***Add the content in between :***

<properties>

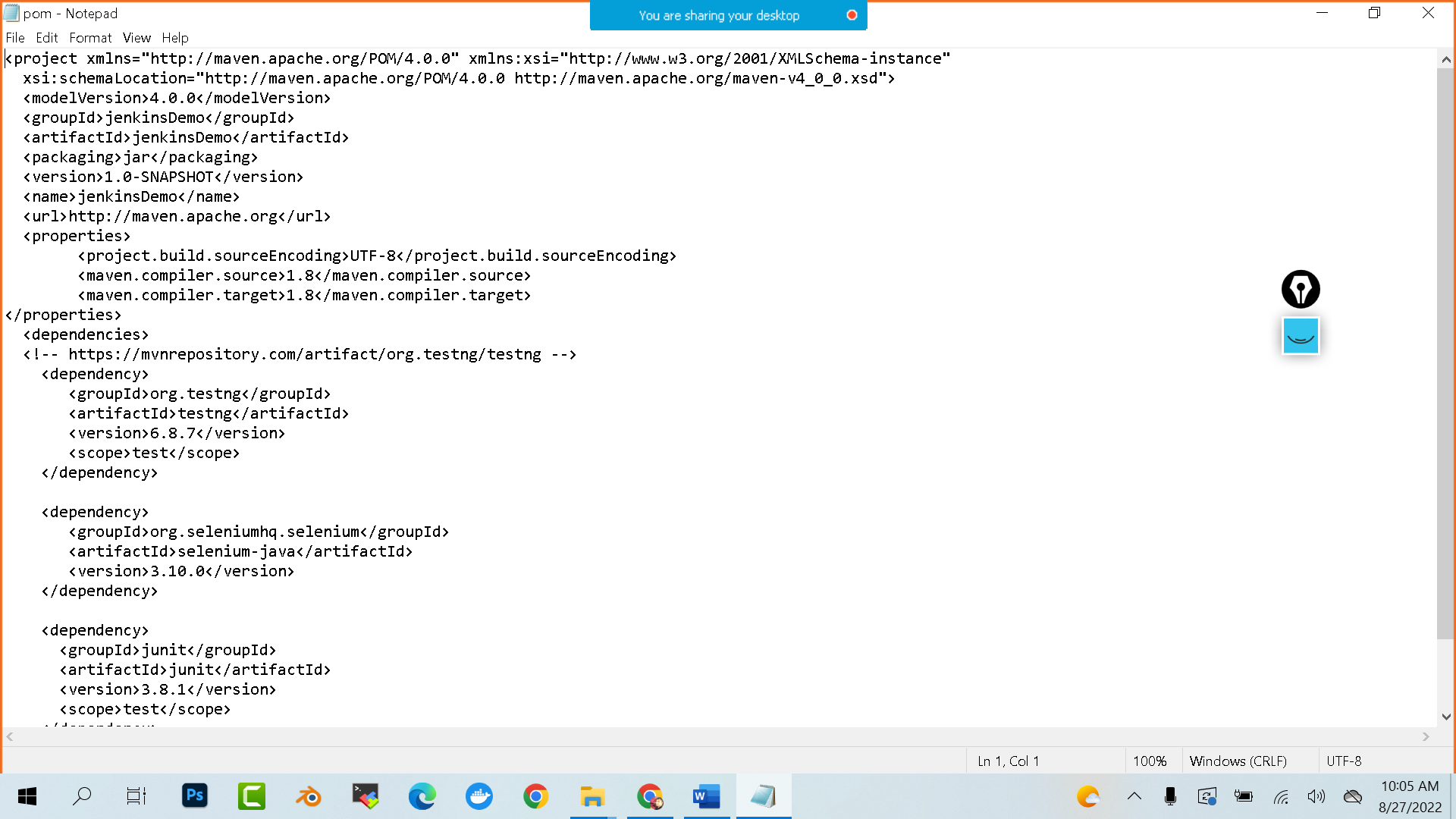
<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

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

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

</properties>

***Your pom file looks like this***



2.5 Save the file and exit using the command **[esc] shift+:wq**

2.6 Delete the src/main folder using the commands given below:

***cd src***

***rm -r main***

2.7 Create a file **JenkinsDemo.java** using the following commands:

***vi JenkinsDemo.java***

2.8 Add the following code in **JenkinsDemo.java:**

***import org.testng.After;  
import org.testng.Assert;  
import org.testng.Before;  
import org.testng.Ignore;  
import org.testngTest;  
import org.openqa.selenium.By;  
import org.openqa.selenium.WebDriver;  
import org.openqa.selenium.chrome.ChromeDriver;  
  
/\*\*  
 \* Created by   
 \* Sample junit test code to integrate by  
 \*/  
public class JenkinsDemo  
{  
 private static String Base\_Url = "https://www.facebook.com";  
 private WebDriver driver;  
  
 @Before  
 public void setUp()  
 {  
 driver = new ChromeDriver();  
 driver.get(Base\_Url);  
 }  
  
 @After  
 public void after()  
 {  
 driver.quit();  
 }  
  
 @Test  
 public void testCasePassed()  
 {  
 Assert.assertTrue(driver.findElement(By.xpath("//form[@id='login\_form']")).isDisplayed());  
 }  
  
 @Test  
 public void testCaseFailed()  
 {  
 Assert.assertTrue(driver.findElement(By.xpath("//form[@id='failed case']")).isDisplayed());  
 }  
  
 @Ignore  
 @Test  
 public void testCaseIgnored()  
 {  
 Assert.assertTrue(driver.findElement(By.xpath("//form[@id='ignored case']")).isDisplayed());  
 }  
}***

2.9 Save the file and exit using the **[esc] shift+:wq** command.

2.10 Your directory structure should only have these files.

2.11 Run the following command to delete any unnecessary files or folder structures:  
***rm -r jenkinsDemo***

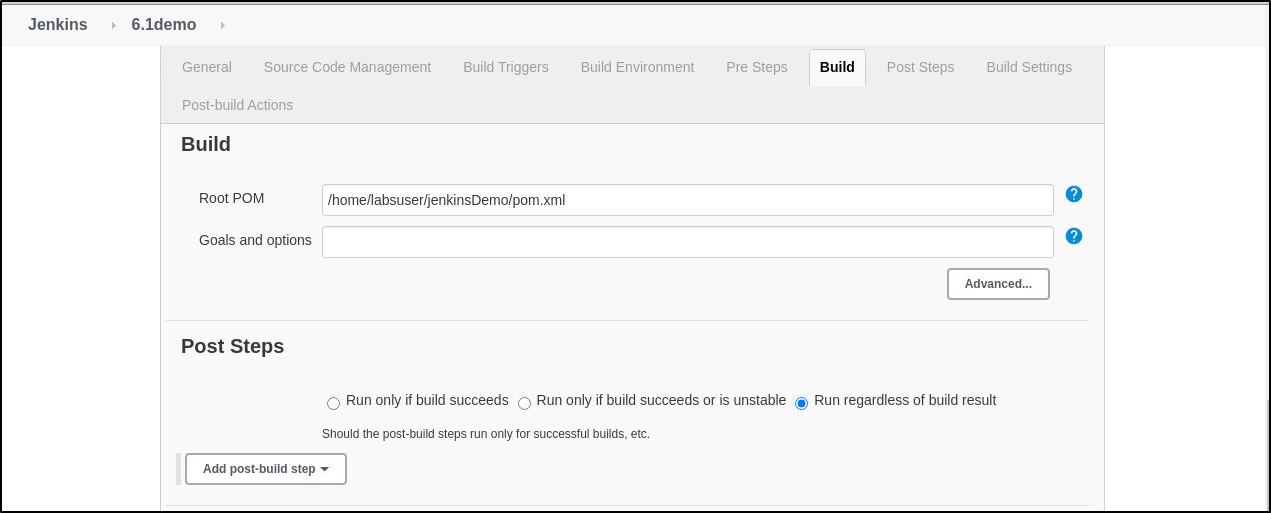
**Step 3: Creating a Jenkins job for Maven project**

3.1 To create a new job in Jenkins, open the Jenkins dashboard with your Jenkins URL.   
For example, http://localhost:8080/

3.2 Click on **New Item**. Enter the item name, select **Maven Project,** and click **OK**

3.3 Once you click **OK,**the page will be redirected to its project form.

3.4 In the **Build** section of your job, for Root POM give the path of the pom.xml in your local system as shown:   
**/*home/<labsuser>/jenkinsDemo/pom.xml***



3.5 Click on **Save**.

3.6 Open the terminal and type **sudo chmod -R 777 jenkinsDemo** command.

**Step 4: Building the Jenkins job**

4.1 Click on **Build Now** and check the status by clicking on **Console Output**.

