**Experiment-6: Creation of Maven Java Project, Maven Web Project and installing Jenkins**

**Aim:** To create Maven Java and web projects using Eclipse, push them to GitHub, and install Jenkins for continuous integration.

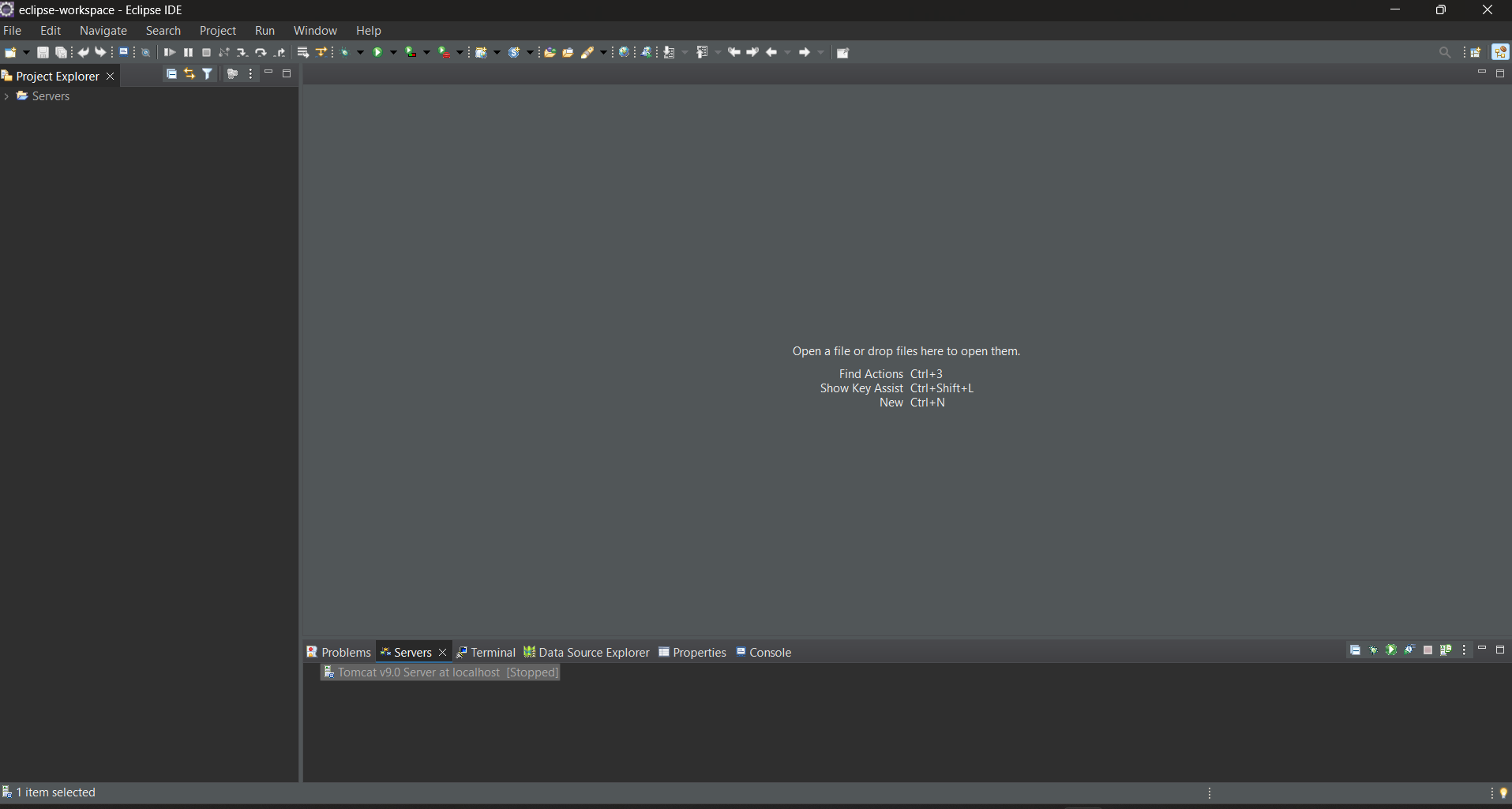
**Introduction:**

1. **Creating Maven Java Project Using Eclipse:**
   * Maven: Maven is a powerful project management tool that is used for managing project dependencies, building projects, and generating project documentation.
   * Eclipse: Eclipse is an integrated development environment (IDE) used for Java and other programming languages, offering a wide range of plugins and tools.
   * GitHub: GitHub is a web-based platform for version control and collaborative development, allowing you to host and manage Git repositories.
2. **Creating Maven Web Project Using Eclipse:**
   * Maven Web Project: A Maven web project is a structured project that includes the necessary files and configurations for building web applications, such as servlets and JSPs.
   * Eclipse Integration: Eclipse provides tools and wizards to create and manage Maven projects, making it easier to develop Java web applications.
   * Pushing to GitHub: Pushing the project to GitHub involves committing the code to a local repository and then pushing it to a remote repository on GitHub, facilitating collaboration and version control.
3. **Installation of Jenkins:**
   * **Jenkins:** Jenkins is an open-source automation server used for continuous integration and continuous delivery (CI/CD). It helps automate the parts of software development related to building, testing, and deploying, facilitating a continuous integration workflow.
   * **Installation:** Installing Jenkins involves setting up the server on a local machine or a server, configuring the necessary plugins, and integrating it with version control systems like GitHub.

**Creation of Maven Java Project:**

**Step 1.** Open Eclipse IDE

└── 1.1. Launch Eclipse workspace



**Step 2**. Install Maven Plugin (if not installed)

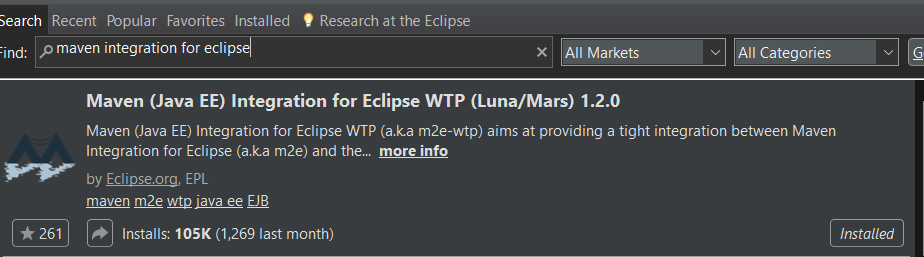
└── 2.1. Go to "Help" in the top menu

└── 2.1.1. Click "Eclipse Marketplace"



└── 2.1.2. Search for "Maven Integration for Eclipse"

└── 2.1.3. Install the plugin if not already installed

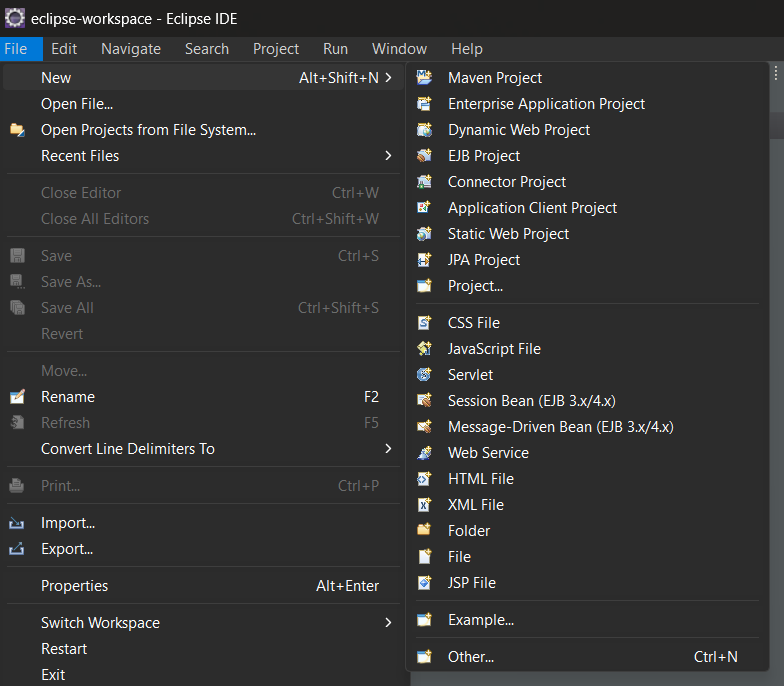


**Step 3**. Create a New Maven Project

└── 3.1. File -> New -> Project...

└── 3.1.1. Expand "Maven"

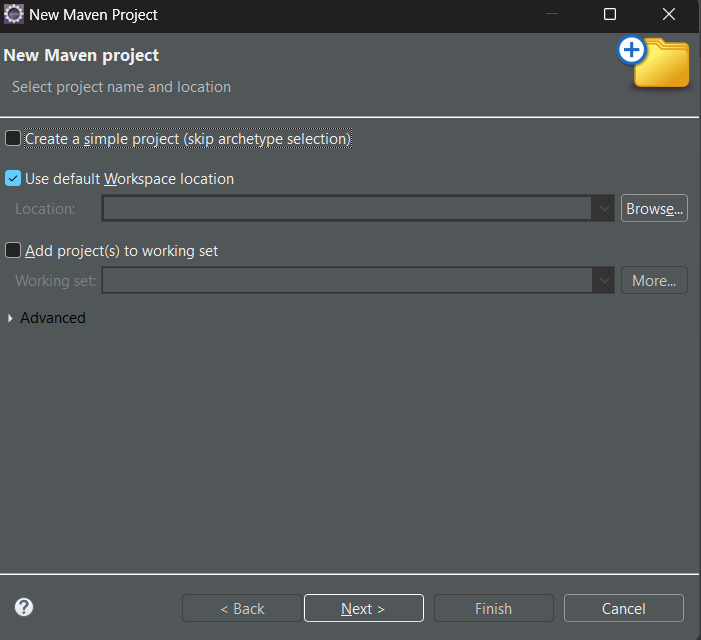
└── 3.1.2. Select "Maven Project" and click "Next"



**Step 4**. Set Project Configuration

└── 4.1. Select workspace location (default or custom)

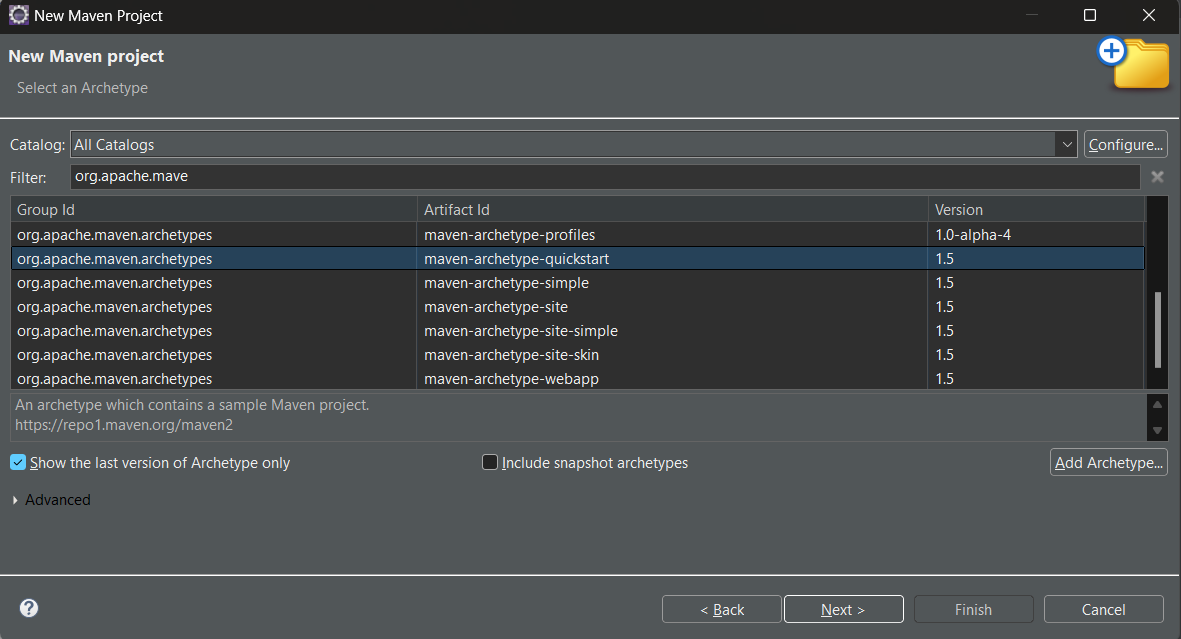
└── 4.2. Click "Next"



**Step 5**. Choose Maven Archetype

└── 5.1. Select an archetype(e.g "org.apache.maven.archetypes -> maven-archetype-quickstart 1.4 ")

└── 5.2. Click "Next"



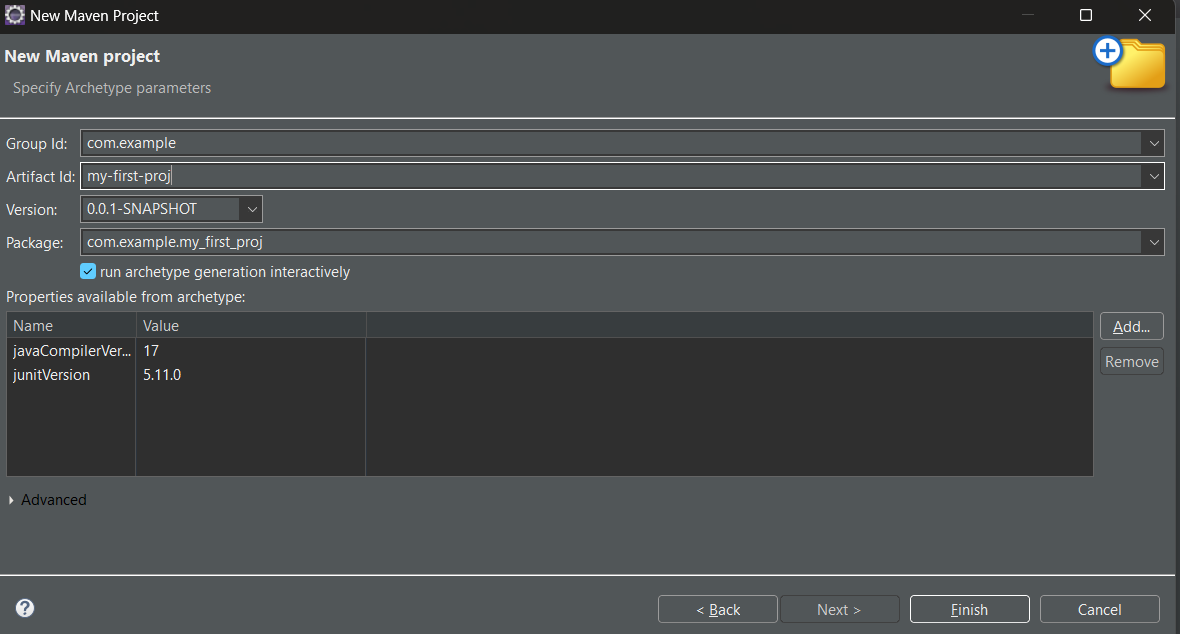
**Step** 6. Define Project Metadata

└── 6.1. Group ID: (e.g., com.example)

└── 6.2. Artifact ID: (e.g., my-maven-project)

└── 6.3. Version: (default is usually fine)

└── 6.4. Click "Finish"



**In Console, artifacts are grouped. When prompted with Y/N, type 'Y'.**

**Step 7**. Maven Project Created

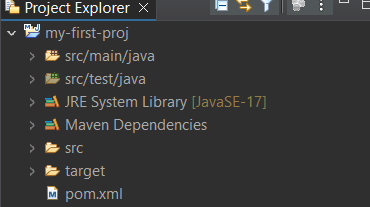
└── 7.1. Project structure is generated with a standard Maven layout

└── 7.2. Includes:

└── src/main/java (for Java source code)

└── src/test/java (for test code)

└── pom.xml (Maven configuration file)



**Step 8**. Update Project Settings (if needed)

└── 8.1. Right-click on the project -> Maven -> Update Project...

└── 8.2. Ensure dependencies are up to date

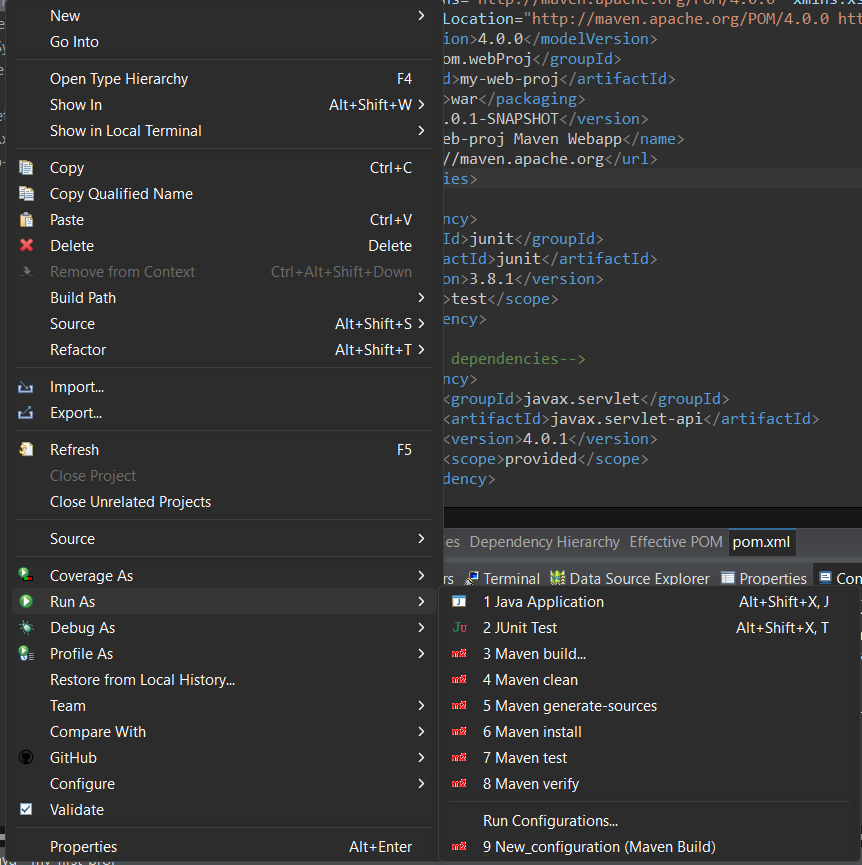
**Step 9**. Build and Run Maven Project

└── 9.1. Right-click on App.java -> Run As -> Maven Clean

└── 9.1.1. Right-click on App.java -> Run As -> Maven Install

└── 9.1.2. Right-click on App.java -> Run As -> Maven Test

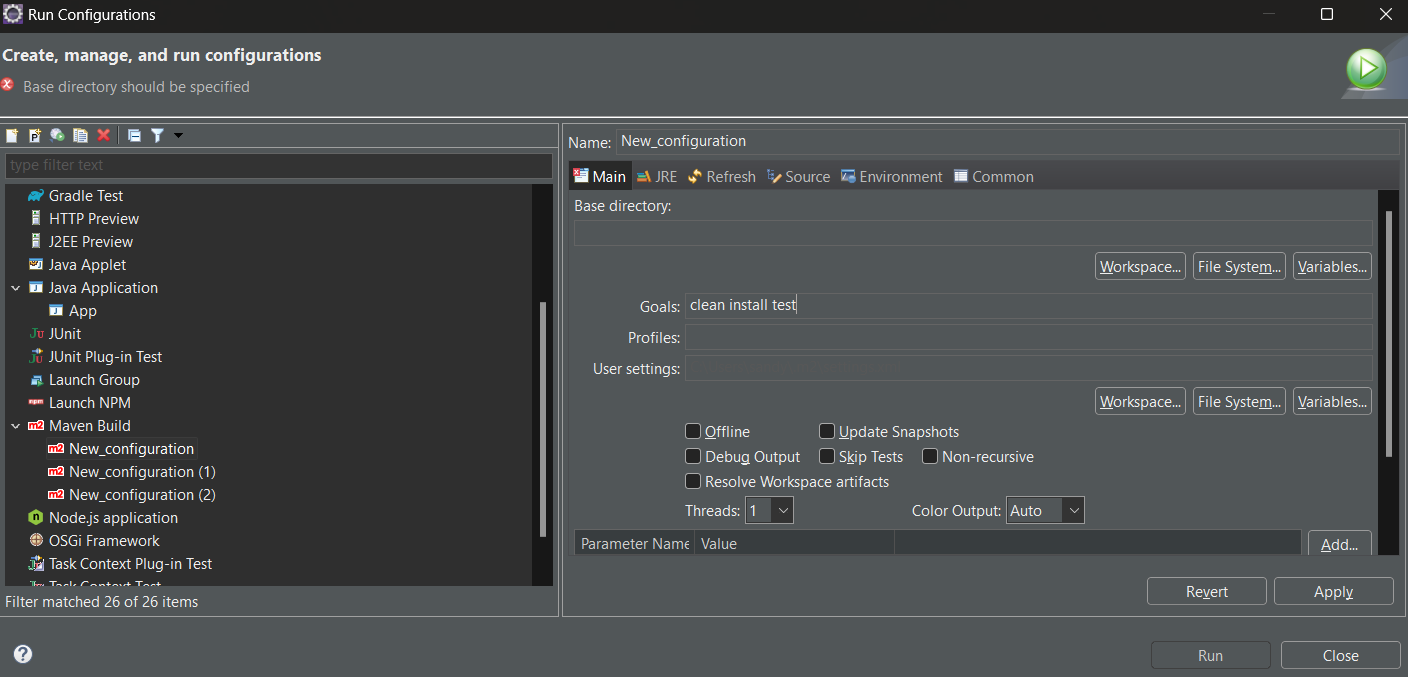
└── 9.1.3. Right-click on App.java -> Run As -> Maven Build



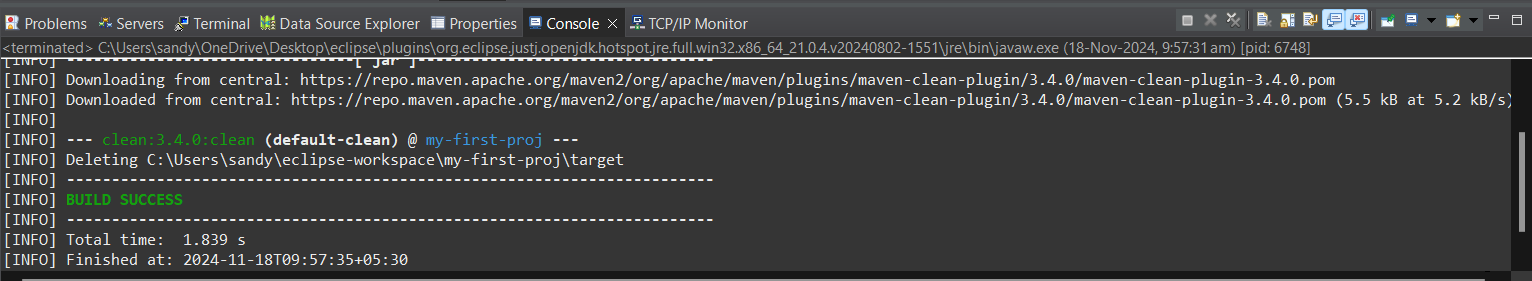
**Step 10**. In the Maven Build dialog:

└── Enter Goals: clean install test

└── Click on Apply -> Click on Run



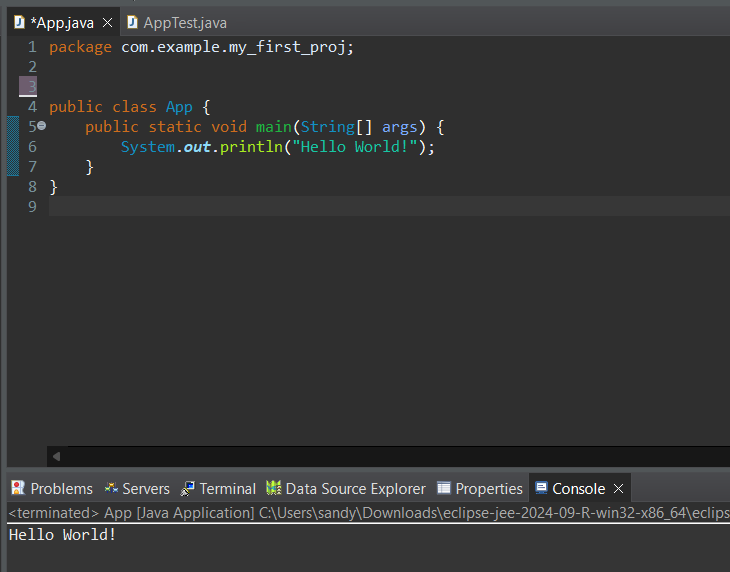
**Step 11**. Check console for BUILD SUCCESS message.



**Step 12**. Run the application:

└── Right-click on App.java -> Run As -> Java Application

└── Output: "Hello World" displayed.

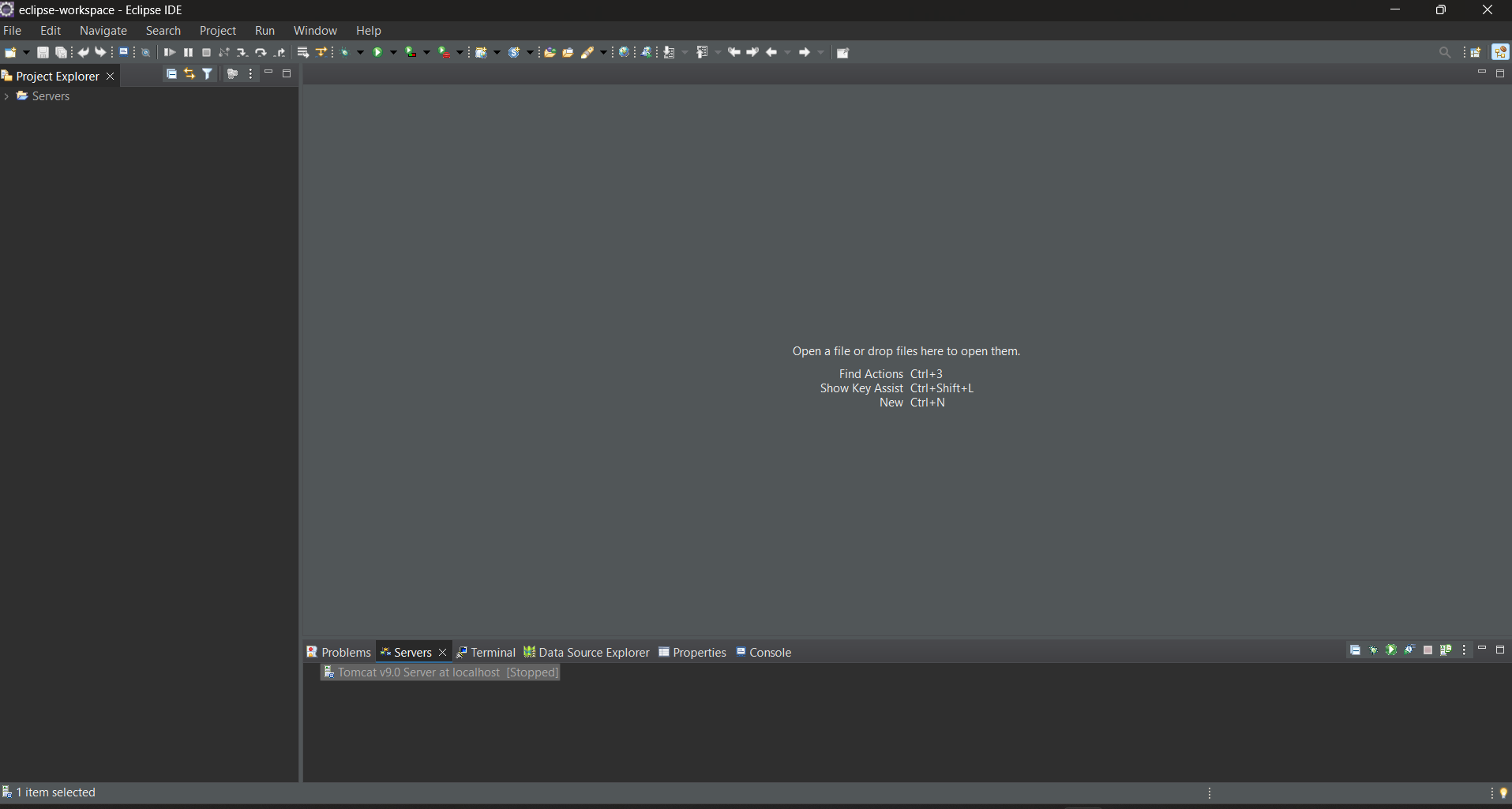


**Creation of Maven web Java Project**

**Step 1**: Open Eclipse

└── 1.1 Launch Eclipse IDE.

└── 1.2 Select or create a workspace.

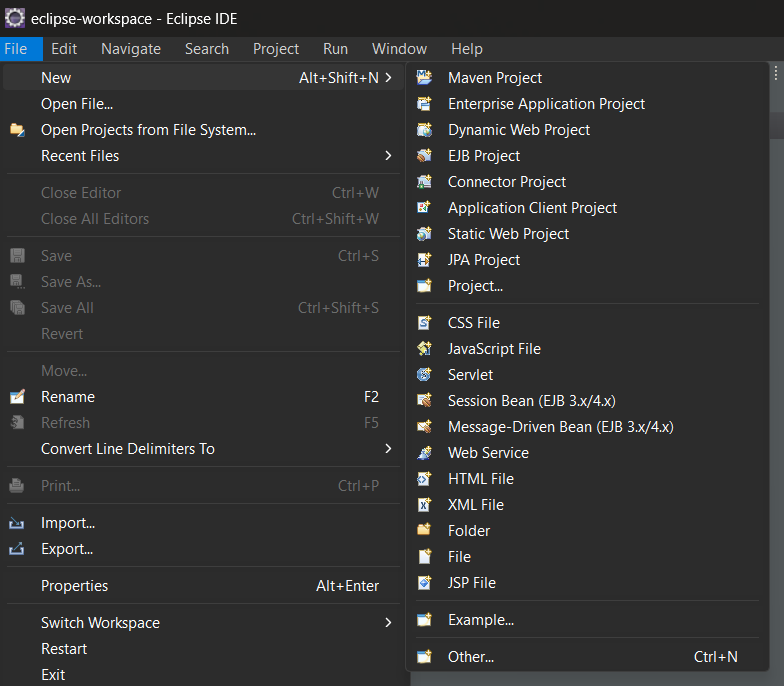


**Step 2**: Create a New Maven Project

└── 2.1. File -> New -> Project...

└── 2.1.1. Expand "Maven"

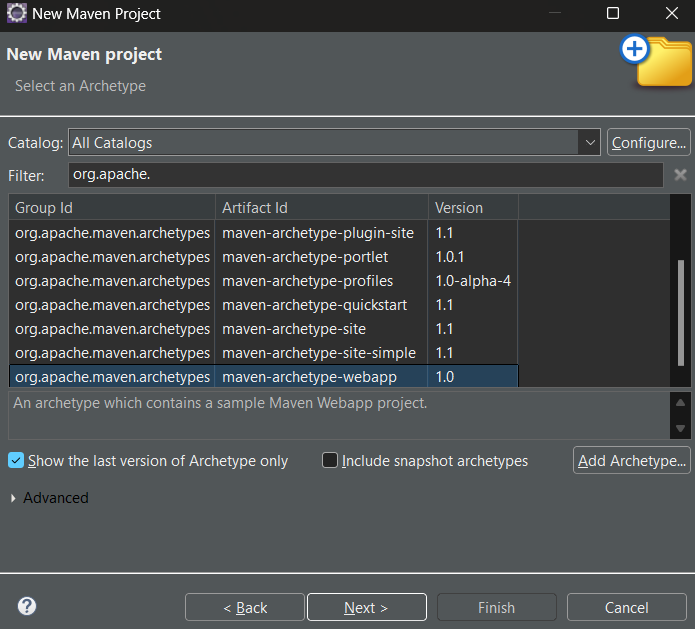
└── 2.1.2. Select "Maven Project" and click "Next"



**Step 3**: Choose Maven Archetype

└── 3.1. Select an archetype(e.g "'org.apache.maven.archetypes' -> 'maven-archetype-webapp' 1.4 ")

└── 3.2. Click "Next"

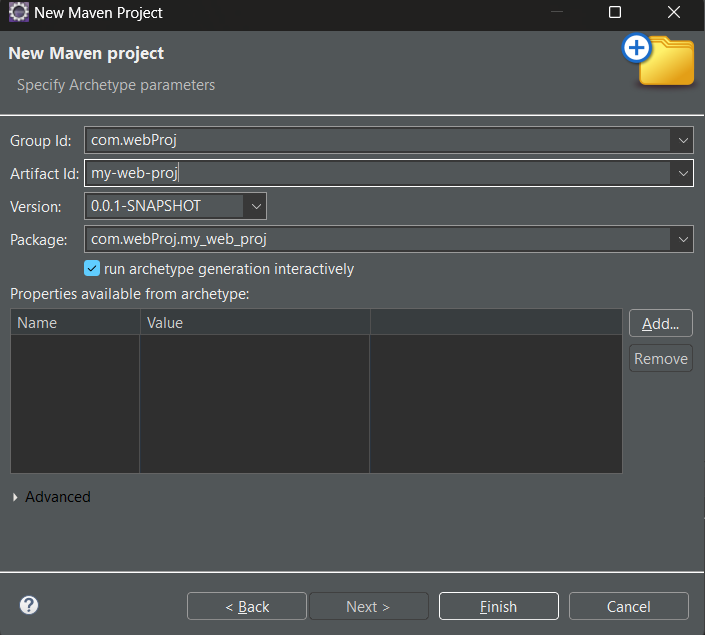


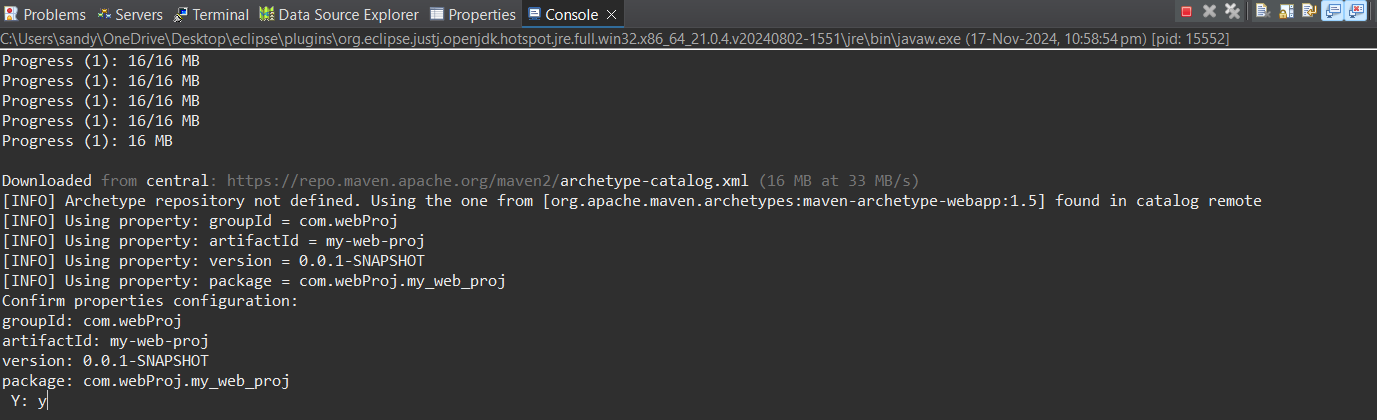
**Step 4**: Configure the Maven Project

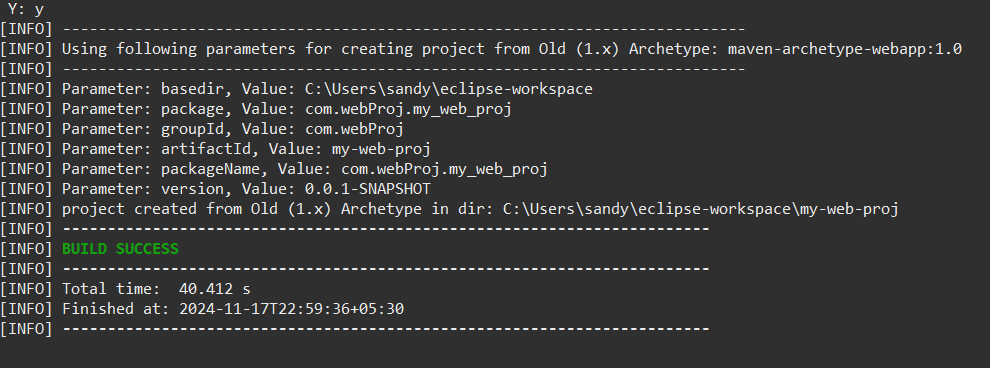
└── 4.1 Group Id: Enter a group ID (e.g., com.example).

└── 4.2 Artifact Id: Enter an artifact ID (e.g., my-web-app).

└── 4.3 Click \*\*Finish\*\* to create the project.







**Step 5**: Add Maven Dependencies

└── 5.1 Open the \*\*pom.xml\*\* file in the Maven project.

└── 5.2 Add the necessary dependencies for your web project (e.g., Servlet, JSP):

**Go to browser -> Open mvnrepository.com**

**Search for 'Java Servlet API' -> Select the latest version.**

**Copy the dependency code -> Paste it in MavenWeb’s pom.xml under the target folder**

└── Example:

```xml

<dependency>

<groupId>javax.servlet</groupId>

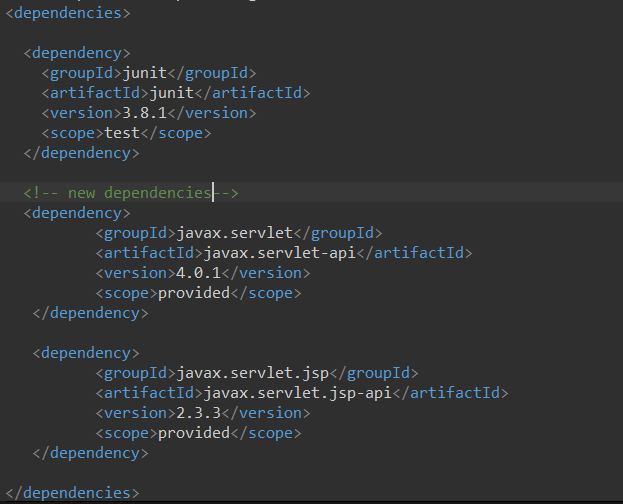
<artifactId>javax.servlet-api</artifactId>

<version>4.0.1</version>

<scope>provided</scope>

</dependency>

```

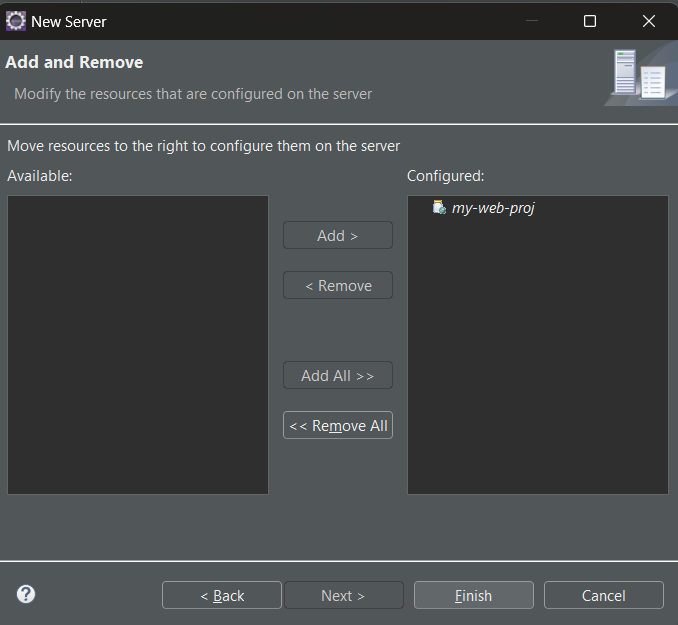


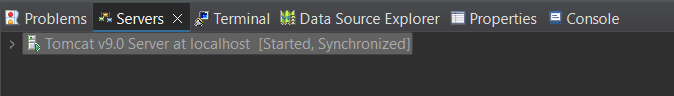
**Step 6**:-. Configure server:

└── Window -> Show View -> Servers

└── Add server -> Select Tomcat v9.0 server -> Click Next

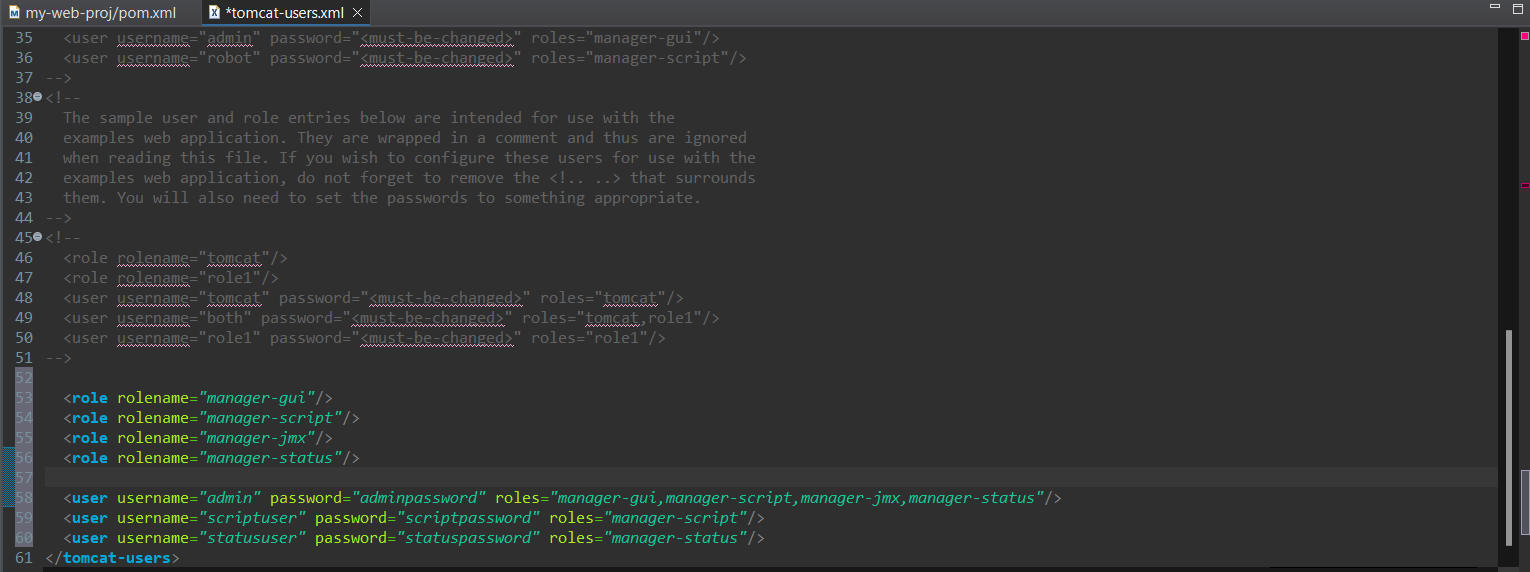
└── Configure server options (e.g., ports, server location).





**Step 7**:-. Modify 'tomcat-users.xml':

└── Add role and user details under <tomcat-users> tag.



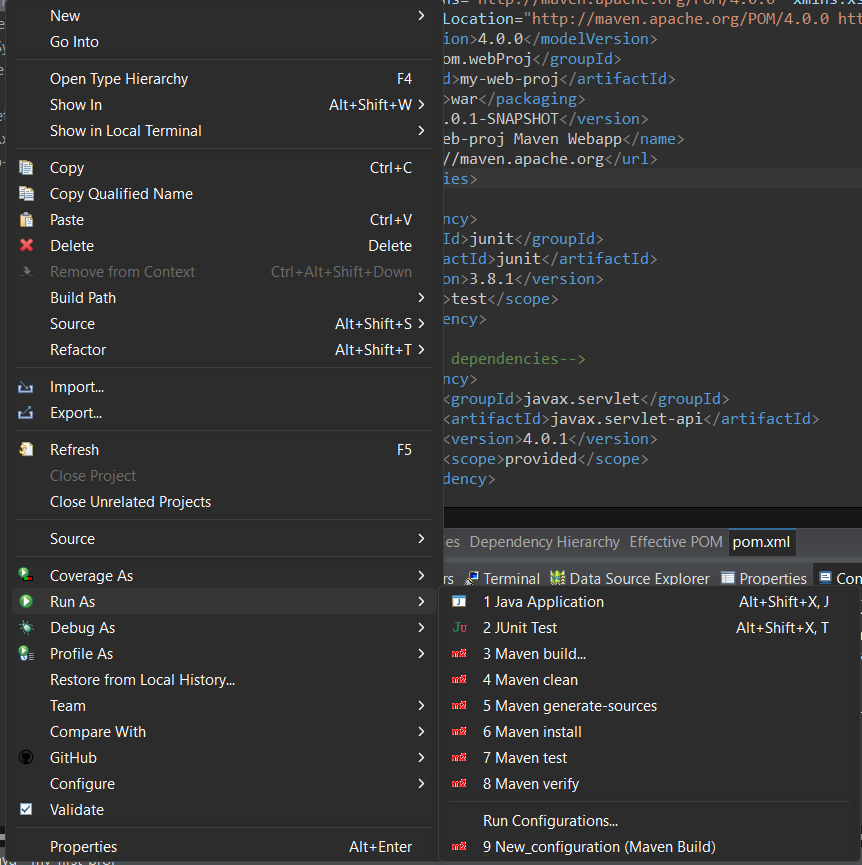
**Step 8**:. Build the project:

└── Right-click on index.jsp -> Run As -> Maven Clean

└── Right-click on index.jsp -> Run As -> Maven Install

└── Right-click on index.jsp -> Run As -> Maven Test

└── Right-click on index.jsp -> Run As -> Maven Build

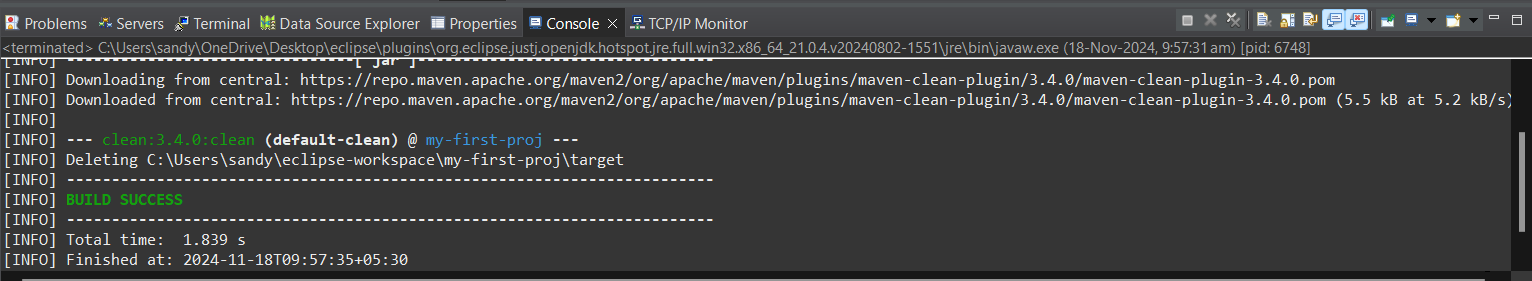


**Step 9**. In the Maven Build dialog:

└── Enter Goals: clean install test

└── Click on Apply -> Click on Run

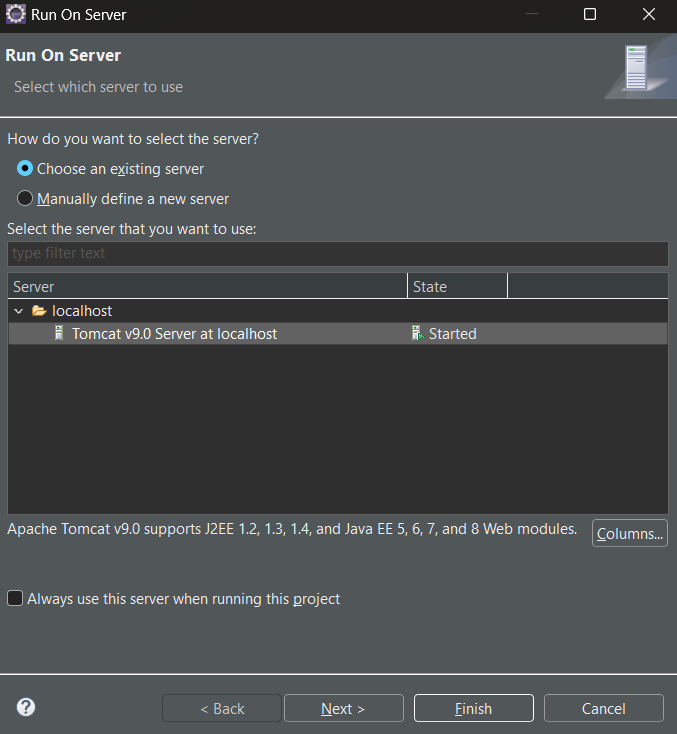
**Step 10**. Check console for BUILD SUCCESS message.



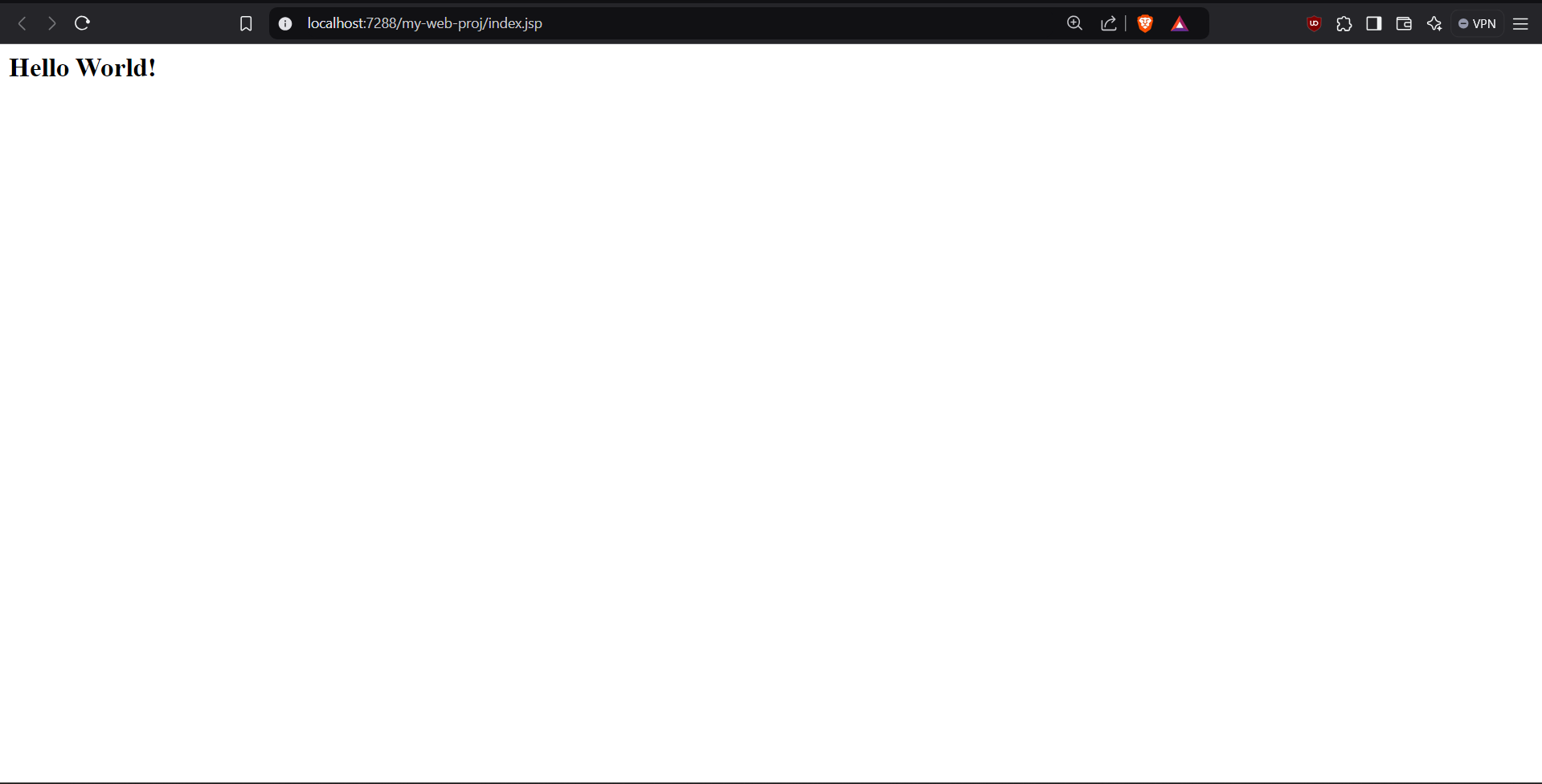
**Step 11**. Run the application:

└── Right-click on index.jsp -> Run As -> Run on Server

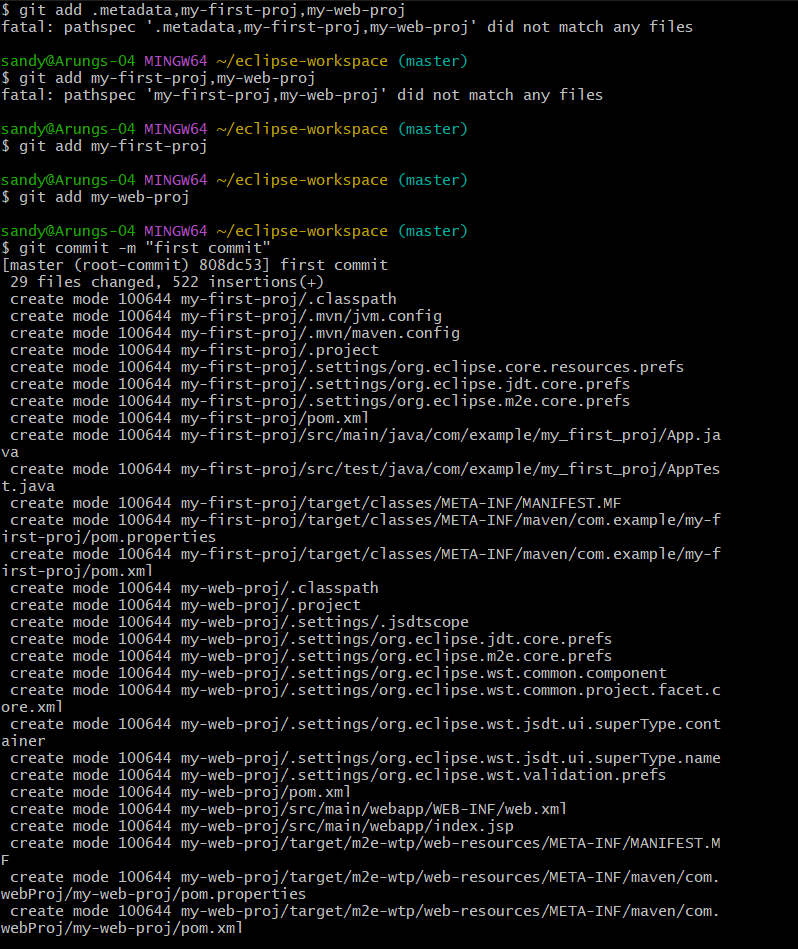
└── Select the Tomcat server -> Click on Finish

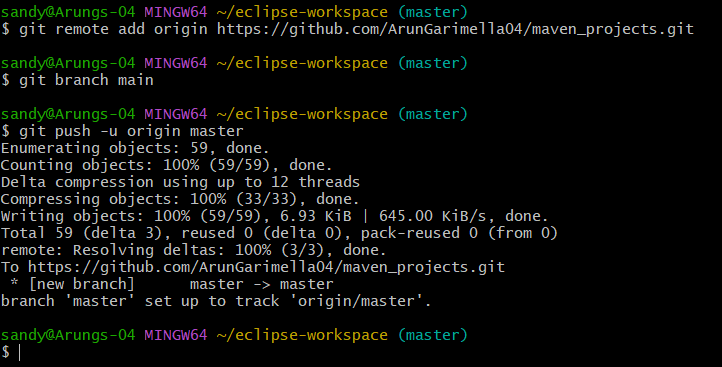


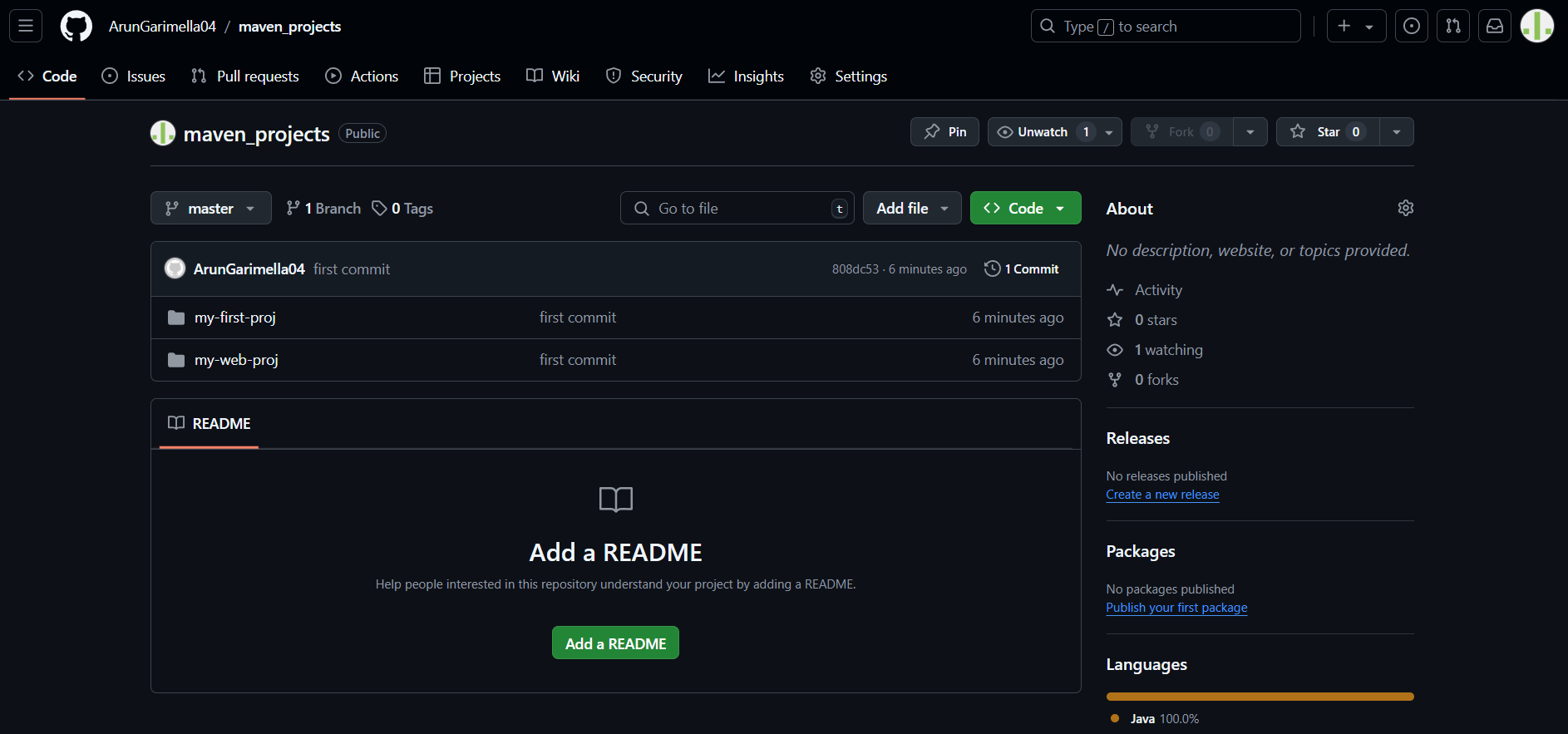
**Step 12**. Output: "Hello World" webpage displayed.



**Note:-Now push yours Maven java project and Maven Web Project into your github**

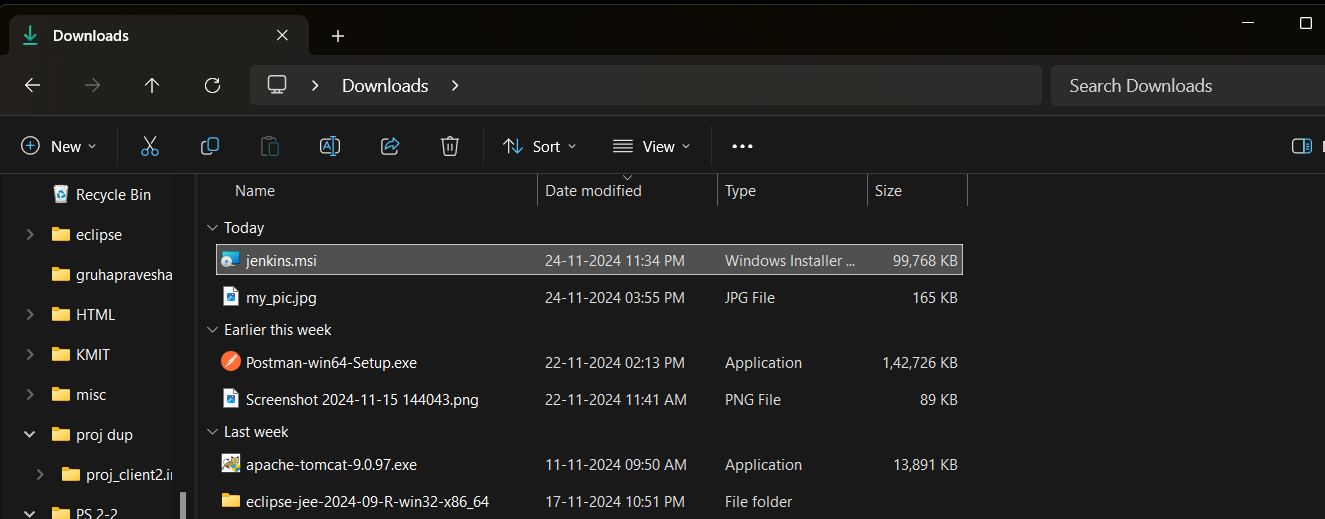
****

****

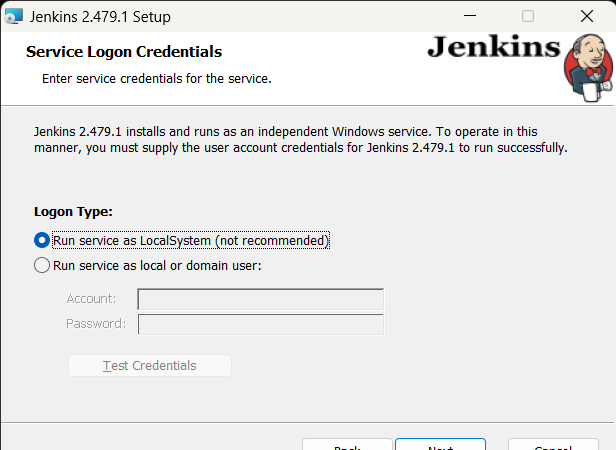
****

**Jenkins Installation Procedure Steps:**

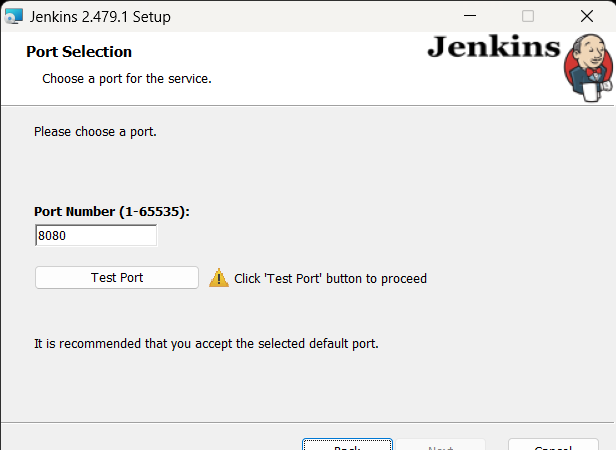
**Step 1:** Download Jenkins for windows where we get Jenkins.msi file double click and run the file.



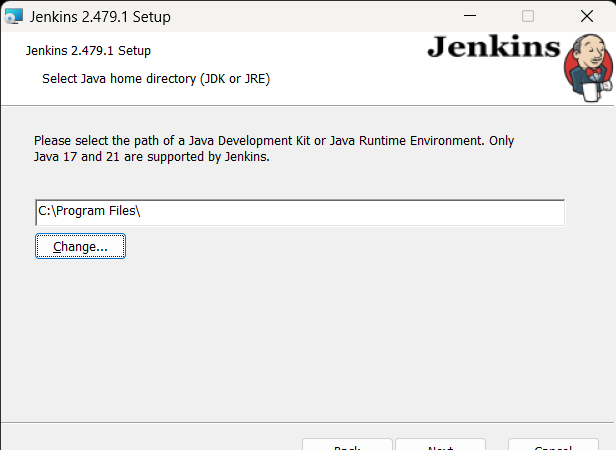
**Step 2:** Click on next and at logon type: select -> Run service as Local System (not recommended) option.



**Step 3:** click on next, and give port no as 8080, click on test port, click on next.



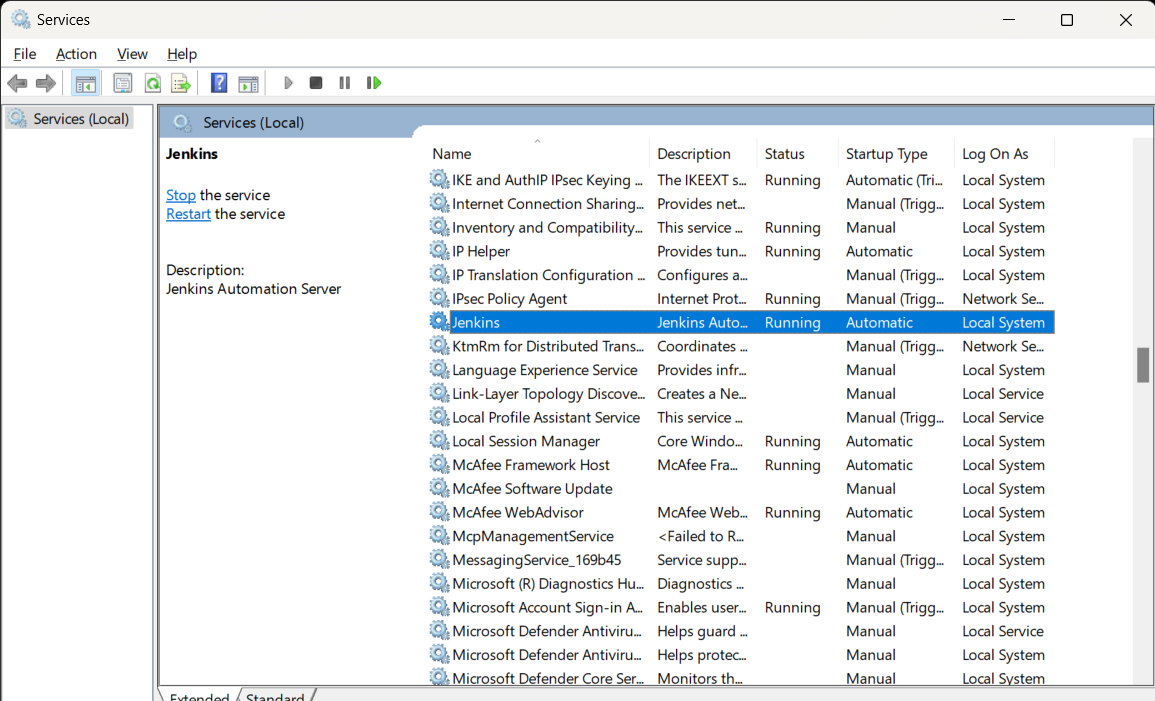
**Step 4**: Now change the file directory by navigating to the jdk 11/17 file present in your program files.



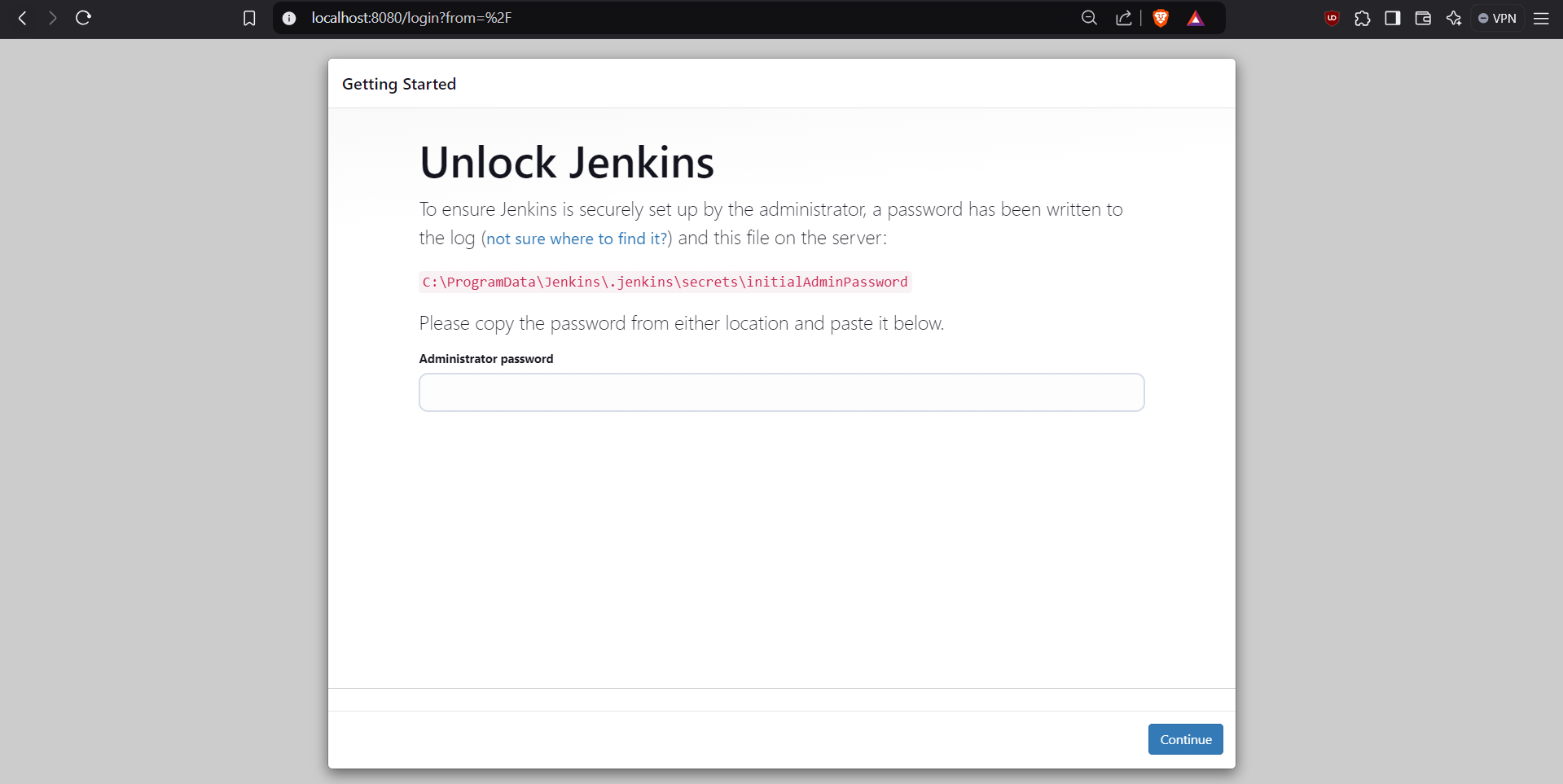
**Step 5:** Click on next and until the setup gets to finish.



**Step 6:** Now check services app in start menu and check if the Jenkins is running.



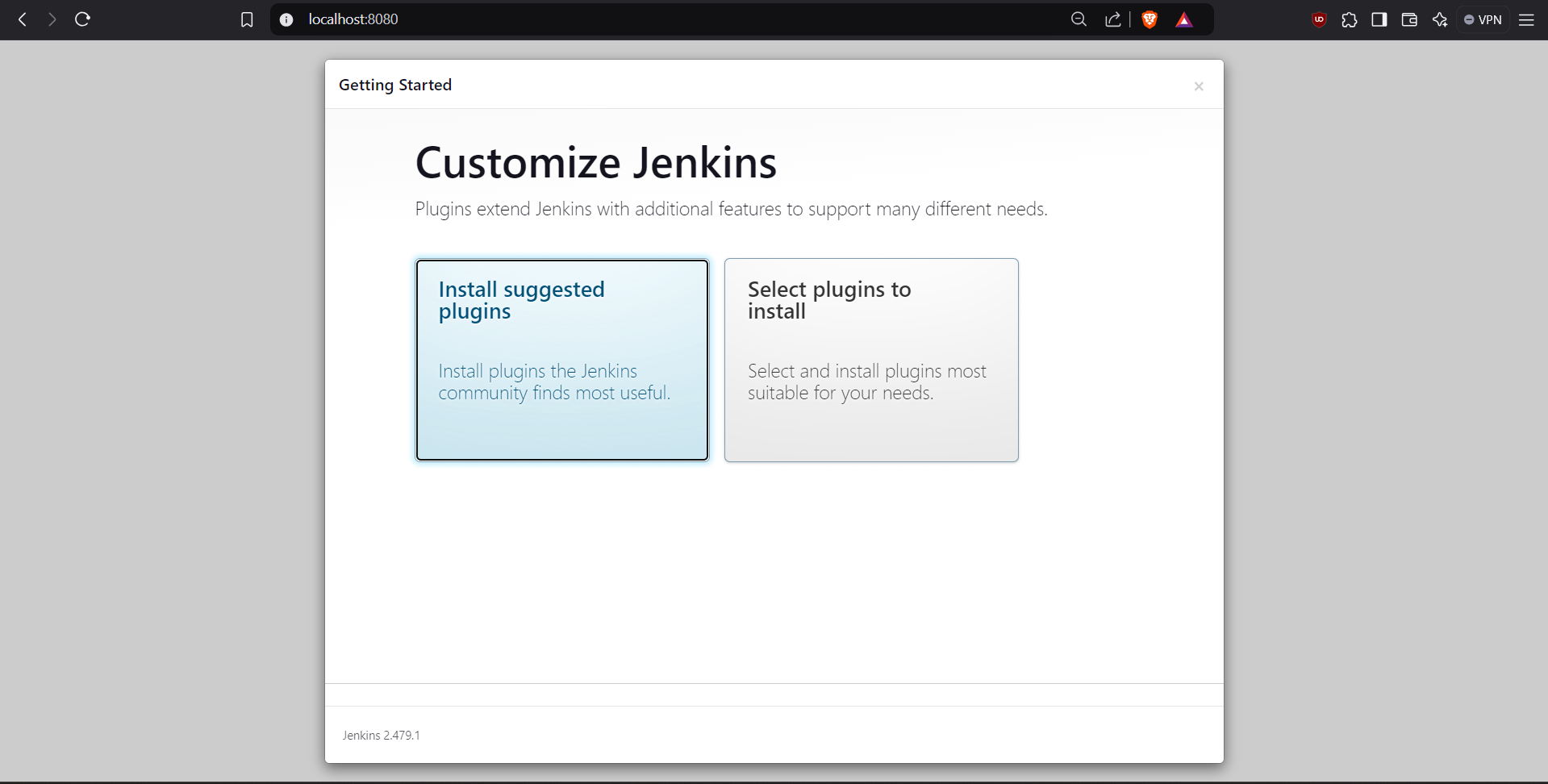
**Step 7:** Now open your browser and type <https://localhost:8080>

****

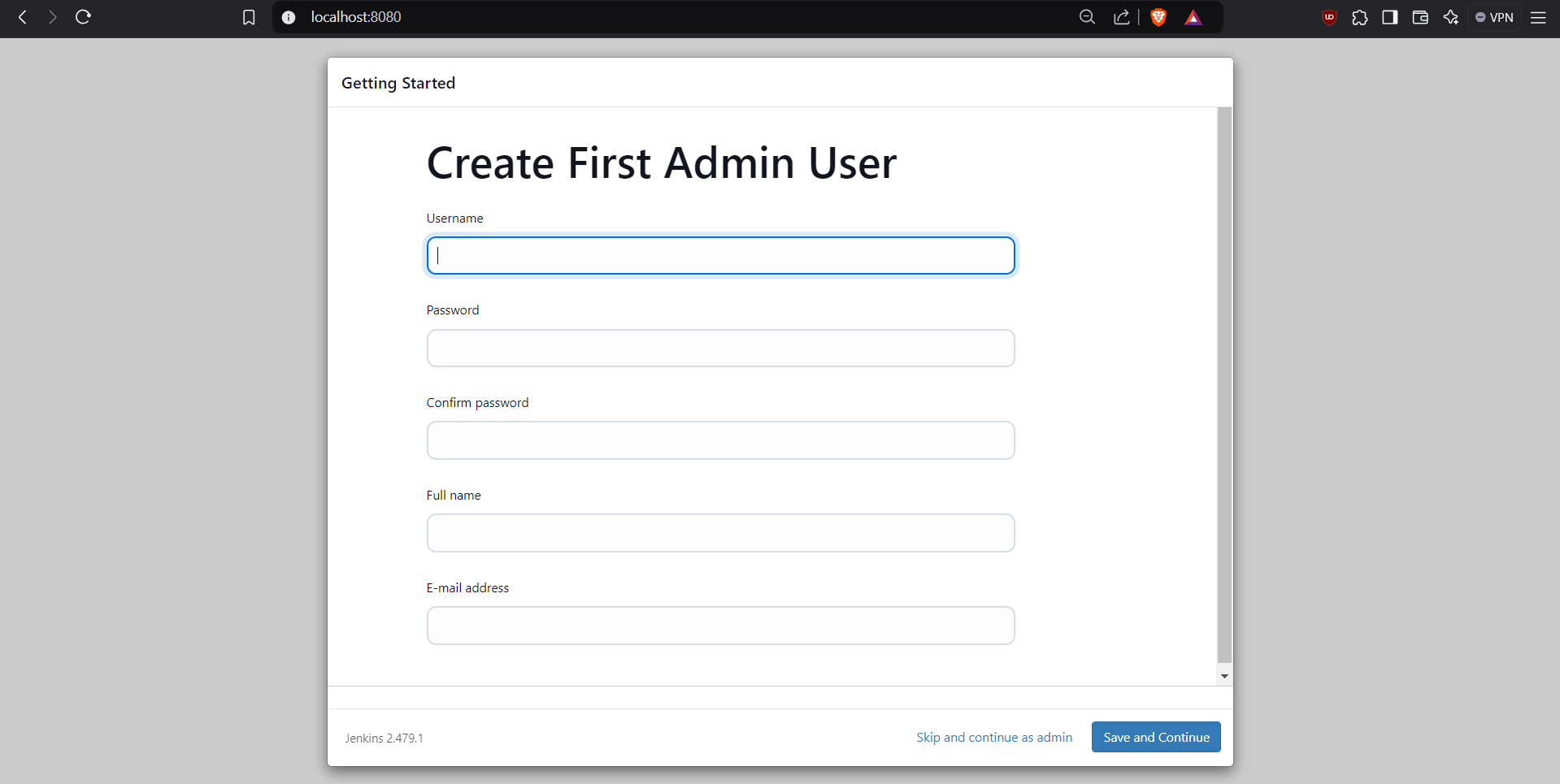
**Step 8:** This page shows us the password to unlock Jenkins in our system. Copy the path and navigate to it in your system.

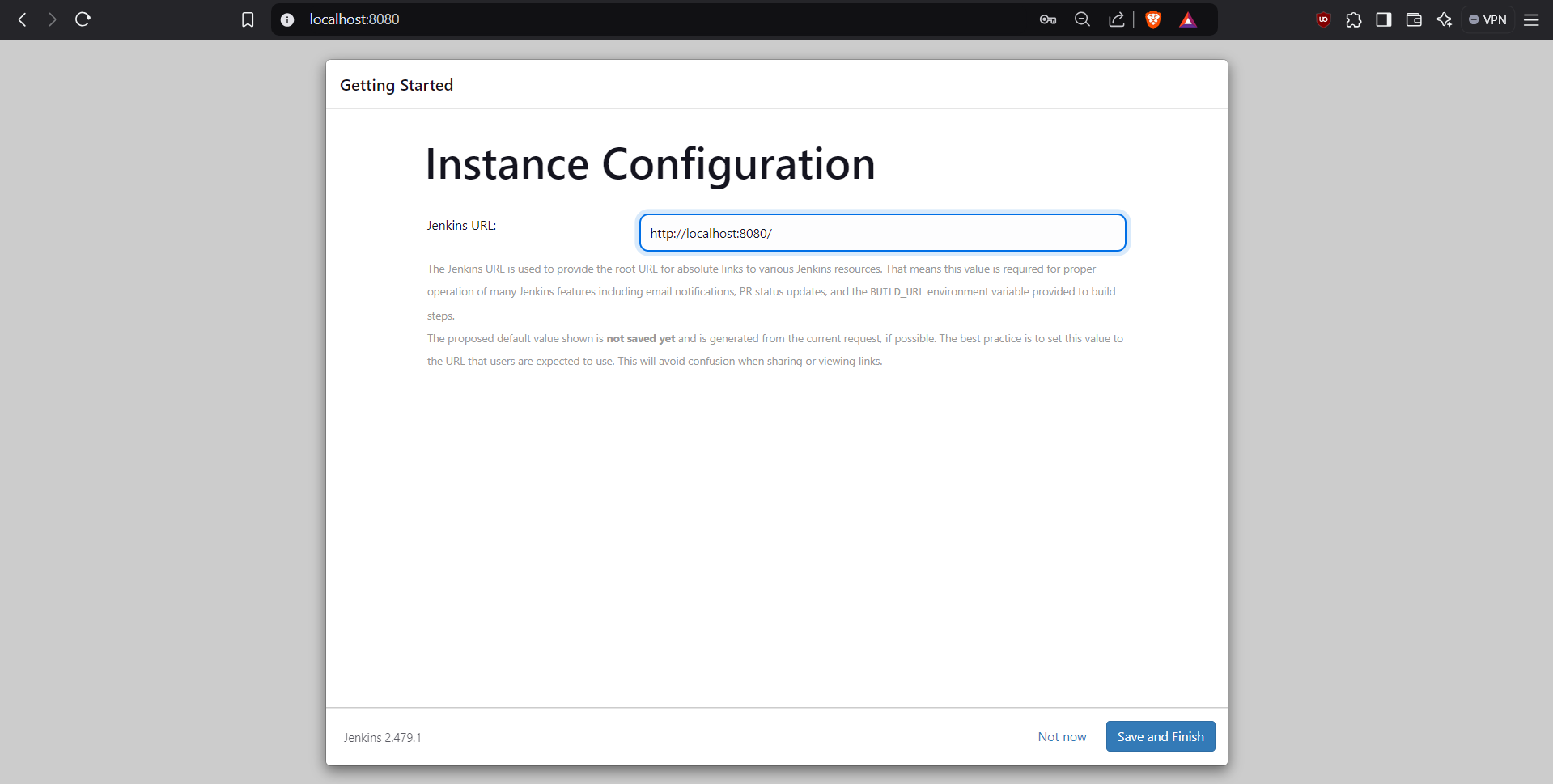
**Step 9:** Click in initial Admin Password and open it and copy the password and paste it in the browser Jenkins page.

**Step 10:** This navigates to a Customize Jenkins page, click on install suggested plugins, and wait until all plugins get installed and redirects to account creation page.

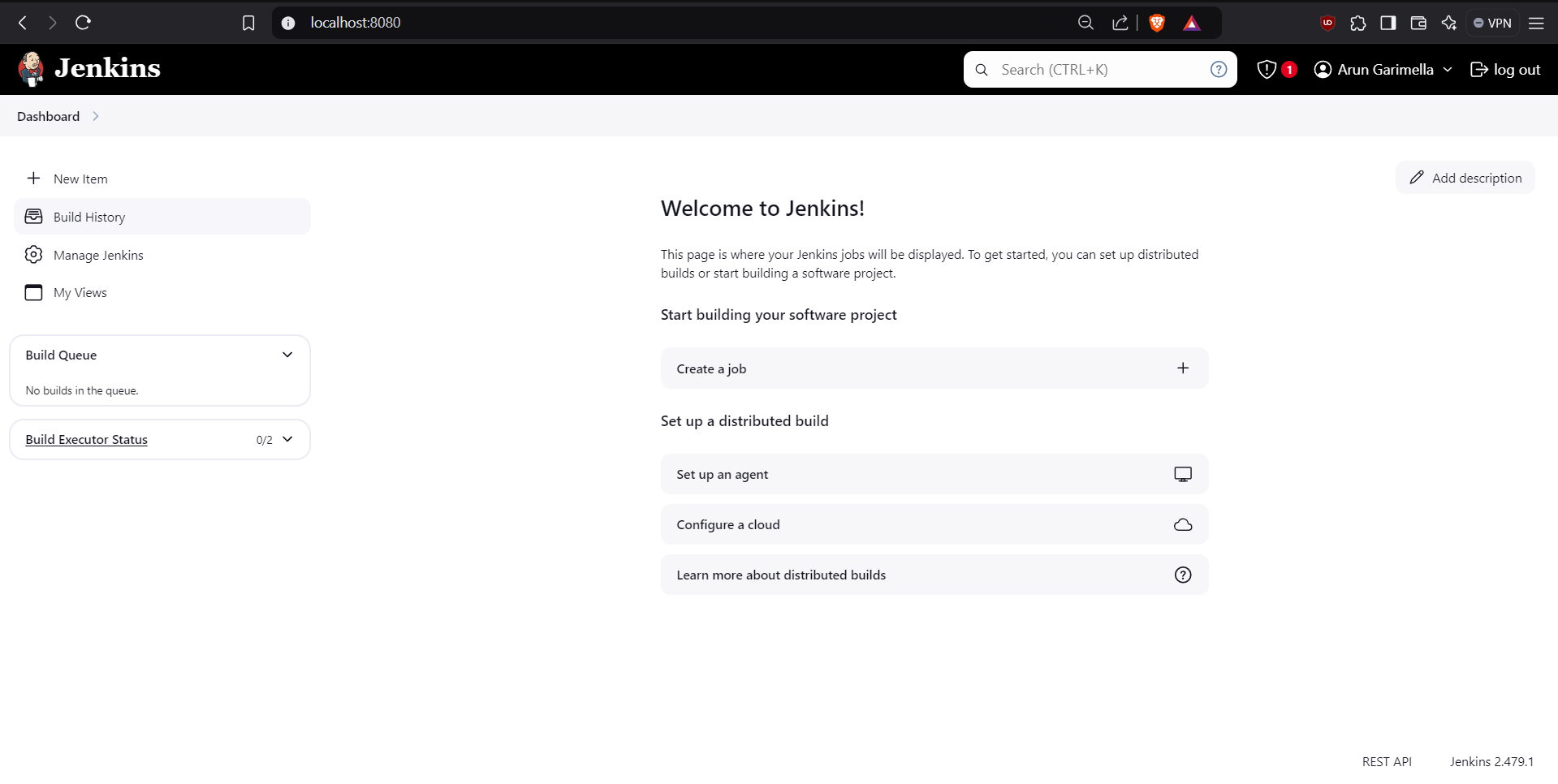


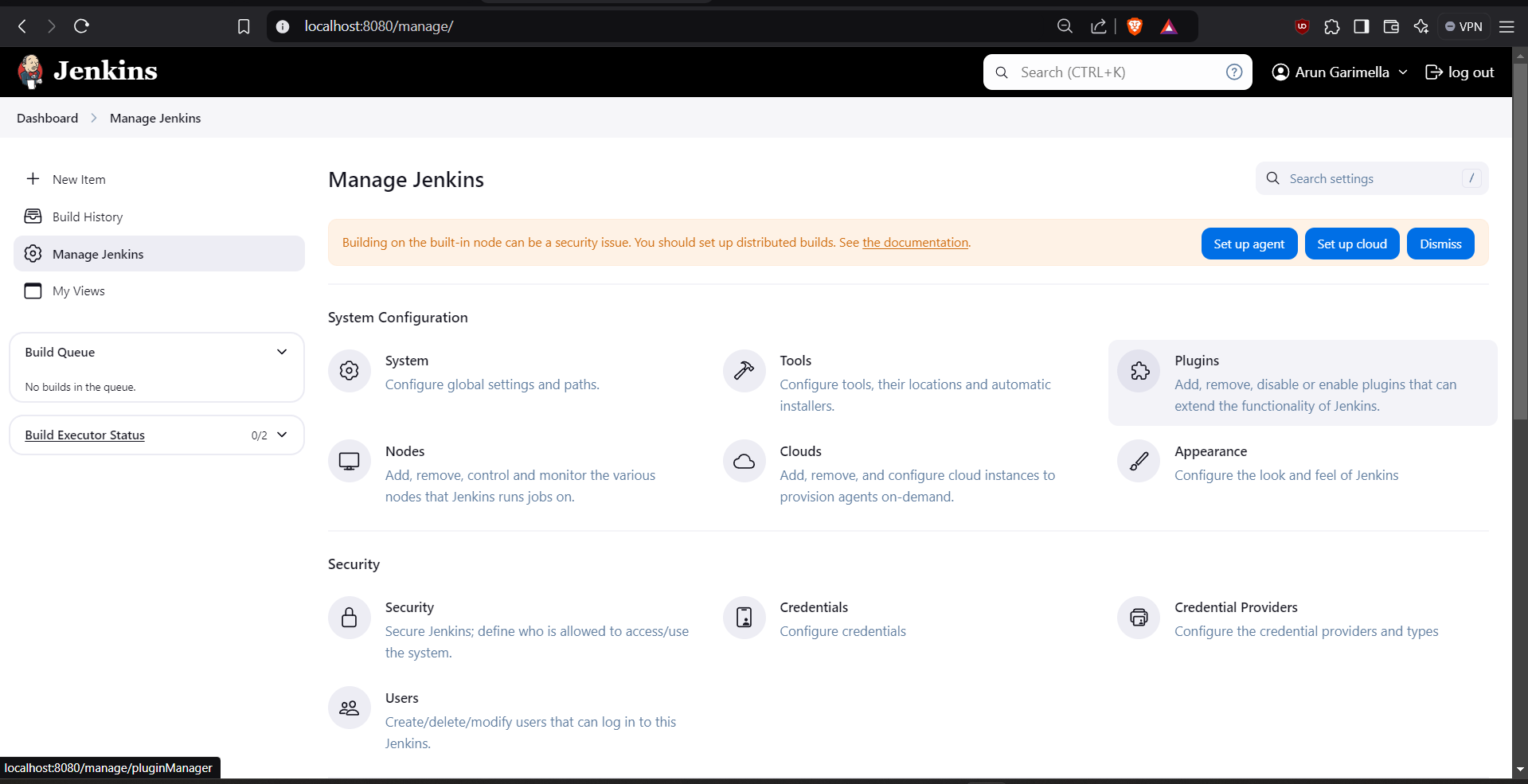
**Step 11:** Type your credentials, click save and continue, give Jenkins url as <http://localhost:8080/> now click save and finish where upon processing our Jenkins is ready.



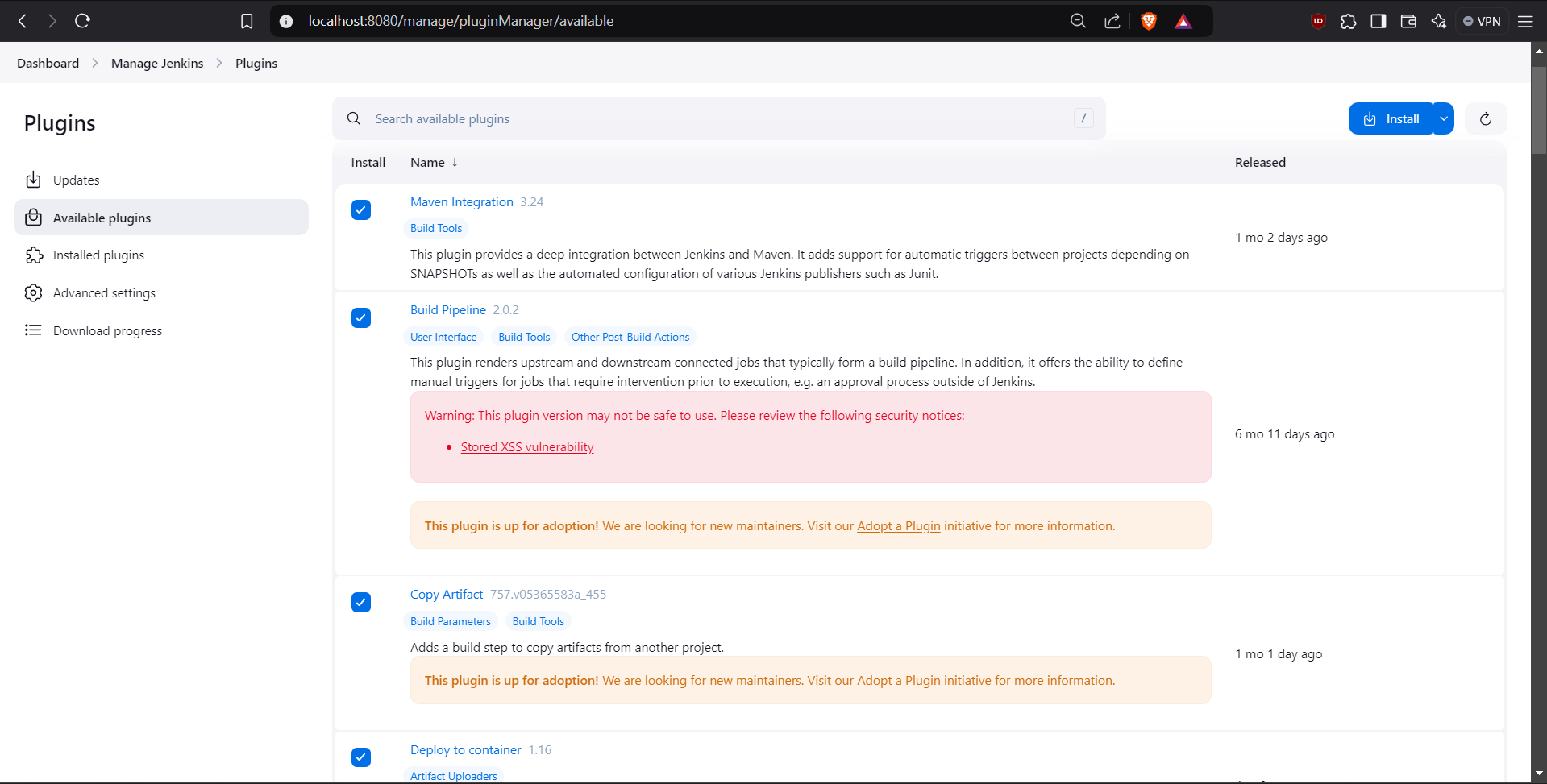


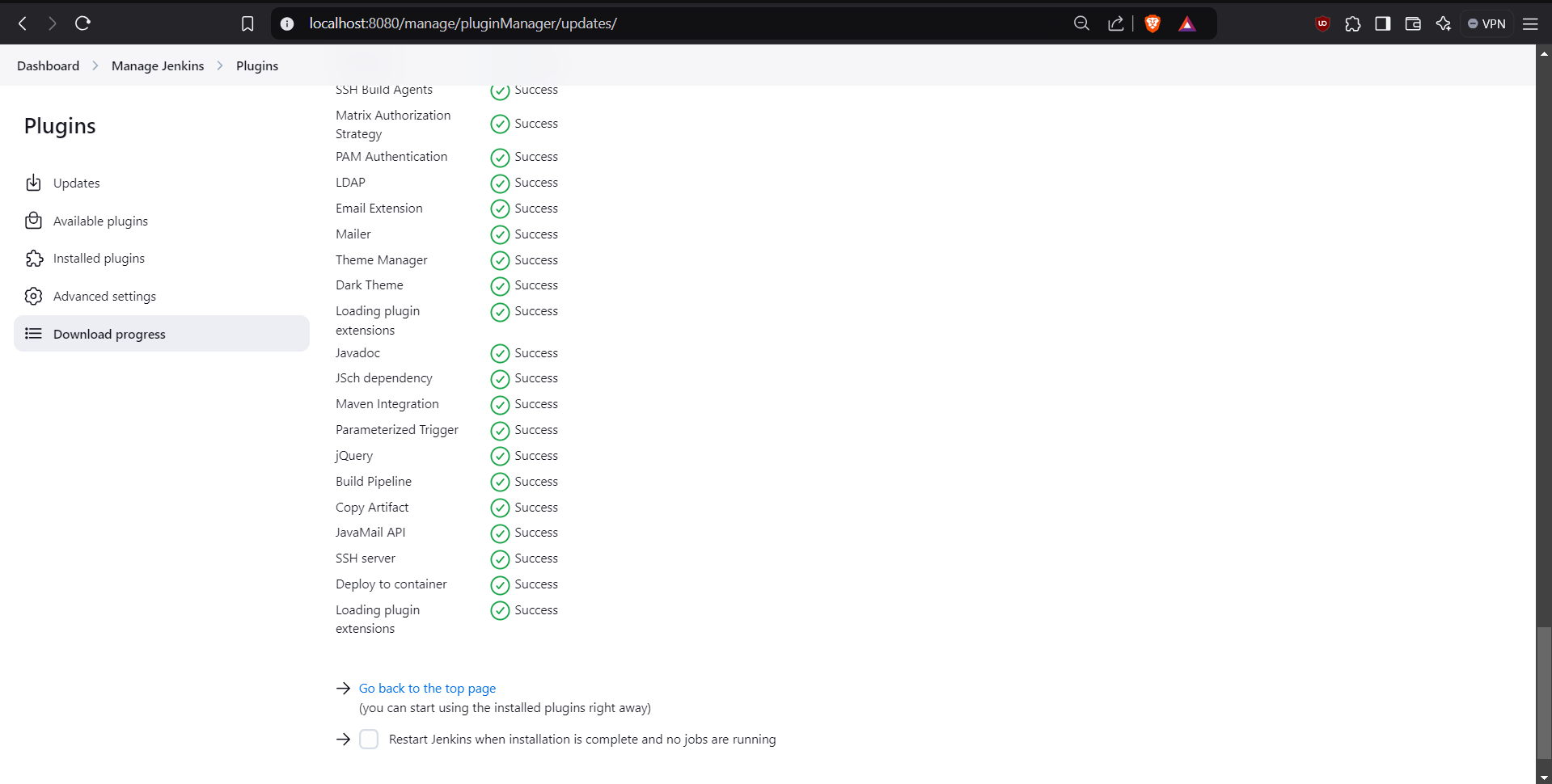
**Step 12:** Click on start using Jenkins, Which redirects to Jenkins homepage, click on manage Jenkins, click on plugins -> available plugins.



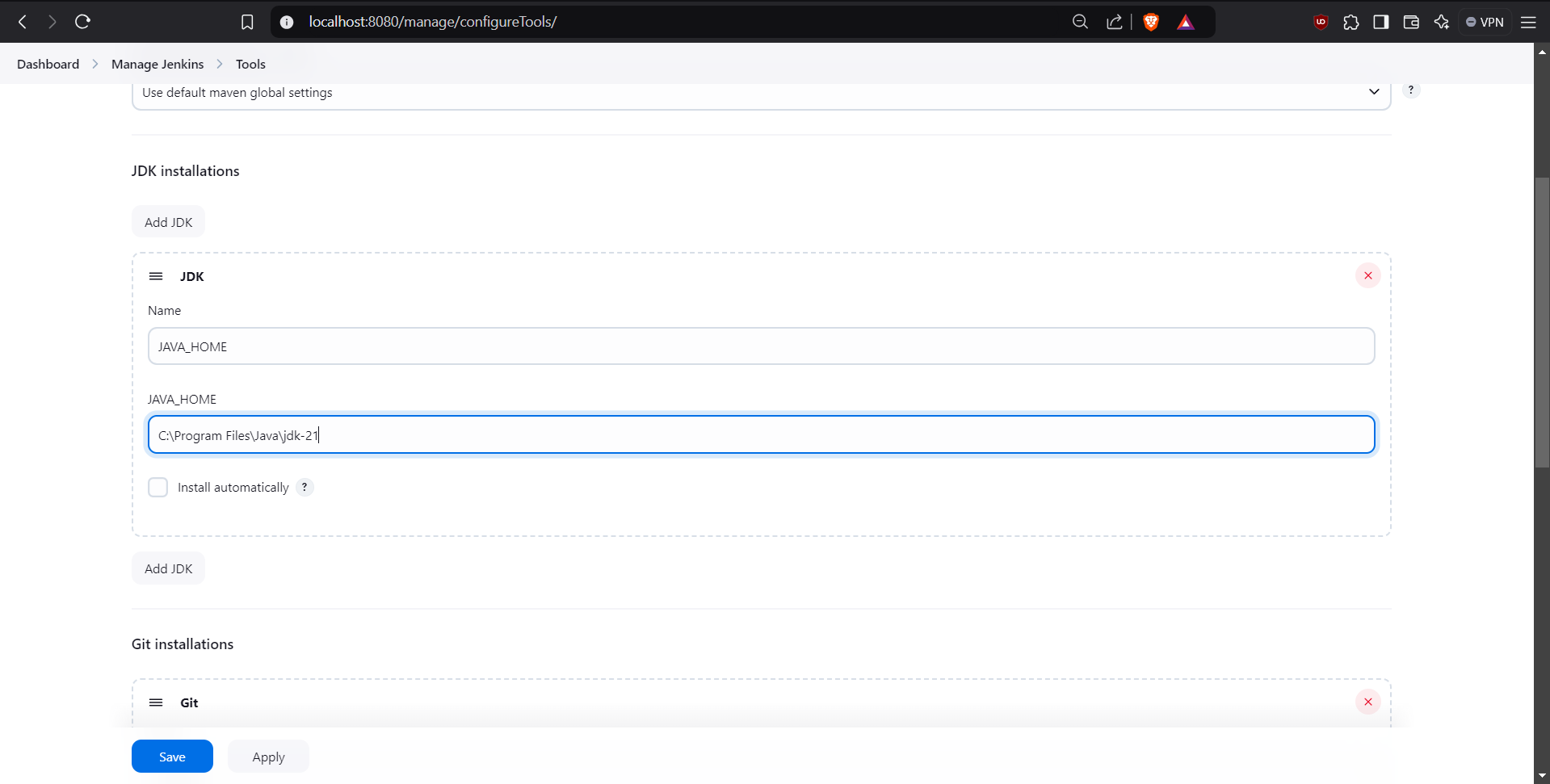


**Step 13:** Search for maven integration, build pipeline, pipeline utility, copy artifacts, deploy to container plugins and install them.

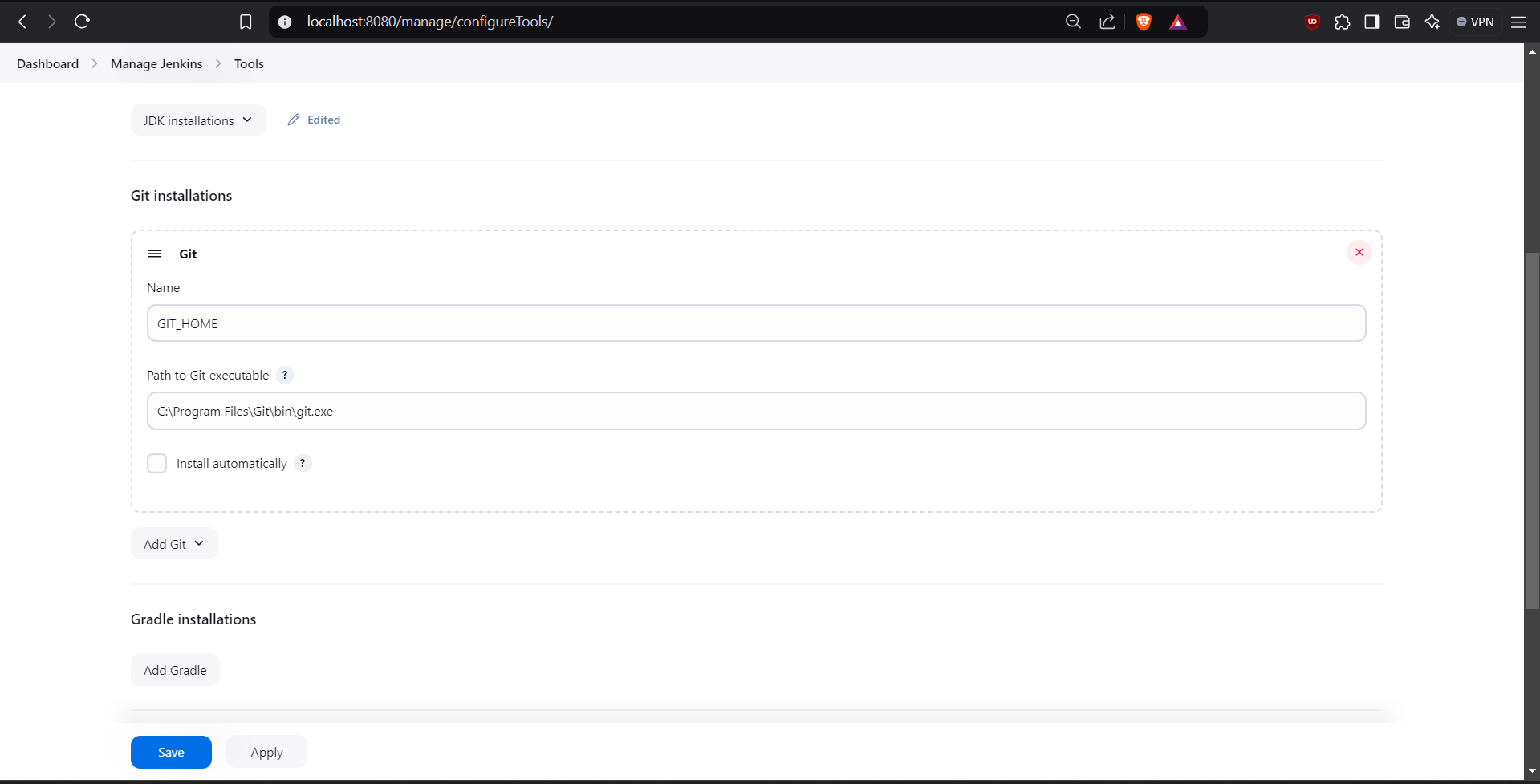




**Step 14:** Go to Manage Jenkins -> tools -> JDK installation, Remove install automatically and Click on edit, give JDK name as JAVA\_HOME, JAVA\_HOME as C:\Program Files\Java\jdk-11



**Step 15:** Go to Manage Jenkins -> tools -> Git installation, Remove install automatically and click on edit, give Git Name: GIT\_HOME, path to git executable: C:\Program Files\Git\bin\git.exe



**Step 16:** Click save and apply in tools.

**Step 17:** Now your JENKINS is ready to use.

**Result:** Created Maven Java and web projects using Eclipse, successfully pushed them to GitHub, and installed Jenkins for continuous integration. This will enhance your skills in project management, version control, and automation in software development.