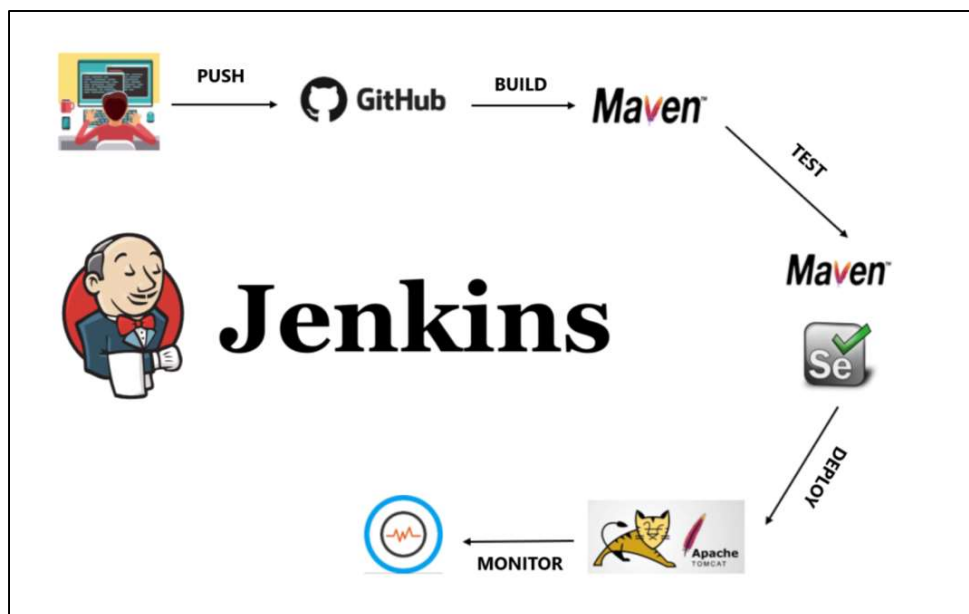


Experiment No : 05

Aim : To Build the pipeline of jobs using Maven / Gradle / Ant in Jenkins, create a pipeline script to Test and deploy an application over the tomcat server.

Theory :

Jenkins is an open-source automation server. It helps automate the parts of software development related to building, testing, and deploying, facilitating continuous integration and continuous delivery. It is a server-based system that runs in servlet containers such as **Apache Tomcat**.



Maven is a build automation tool used primarily for Java projects. Maven can also be used to build and manage projects written in C#, Ruby, Scala, and other languages. The Maven project is hosted by the Apache Software Foundation, where it was formerly part of the Jakarta Project.

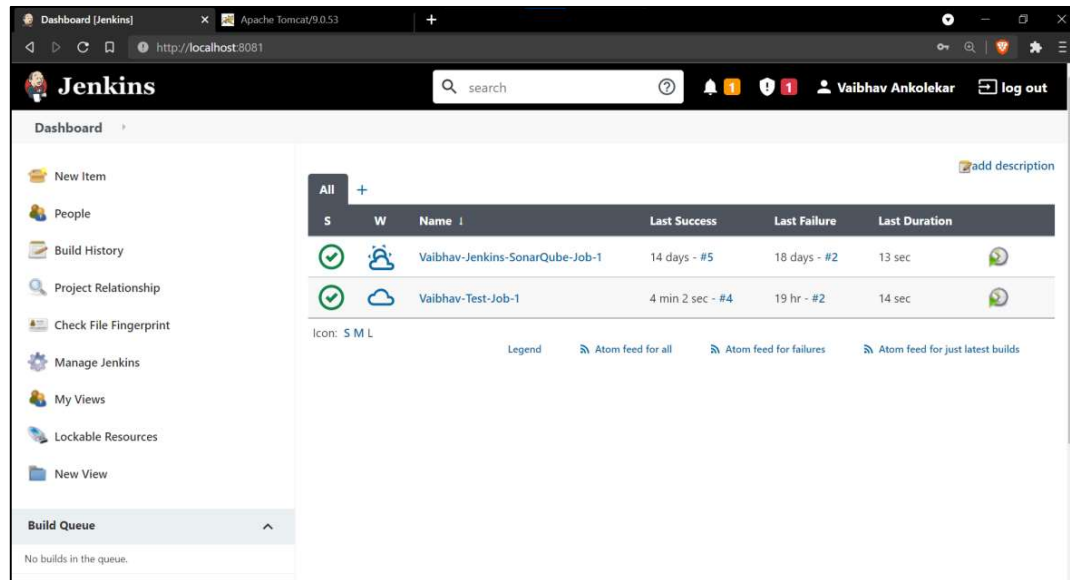
Hence, we can consider Apache Maven is a clear winner in the Jenkins vs Maven comparison. One of the reasons for this being is that Jenkins packages deploys with the Jenkins Maven plug-in. While on the surface level, Jenkins looks like it is doing most of the work, it's actually the Maven plug-in that does all of the heavy-duty work. Maven defines project structure, dependencies, build, and test management. Using pom.xml, we can configure dependencies needed for building testing and running code.

Prerequisite :

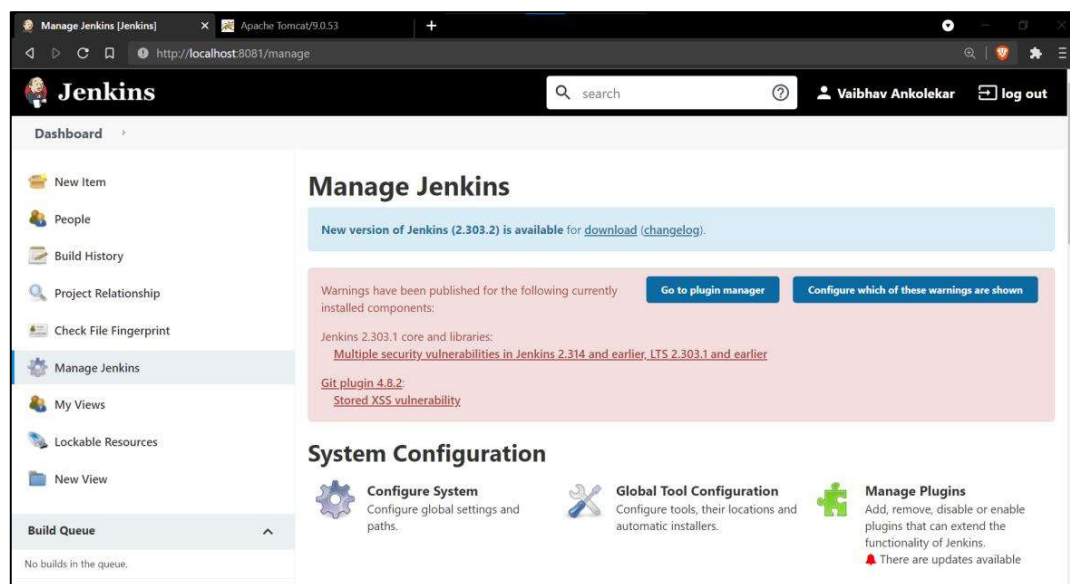
- Install Jenkins, Tomcat, Maven in your machine
- Create a user with admin privileges in Tomcat Server.
- Create a Web application and upload it to Github.

Procedure :

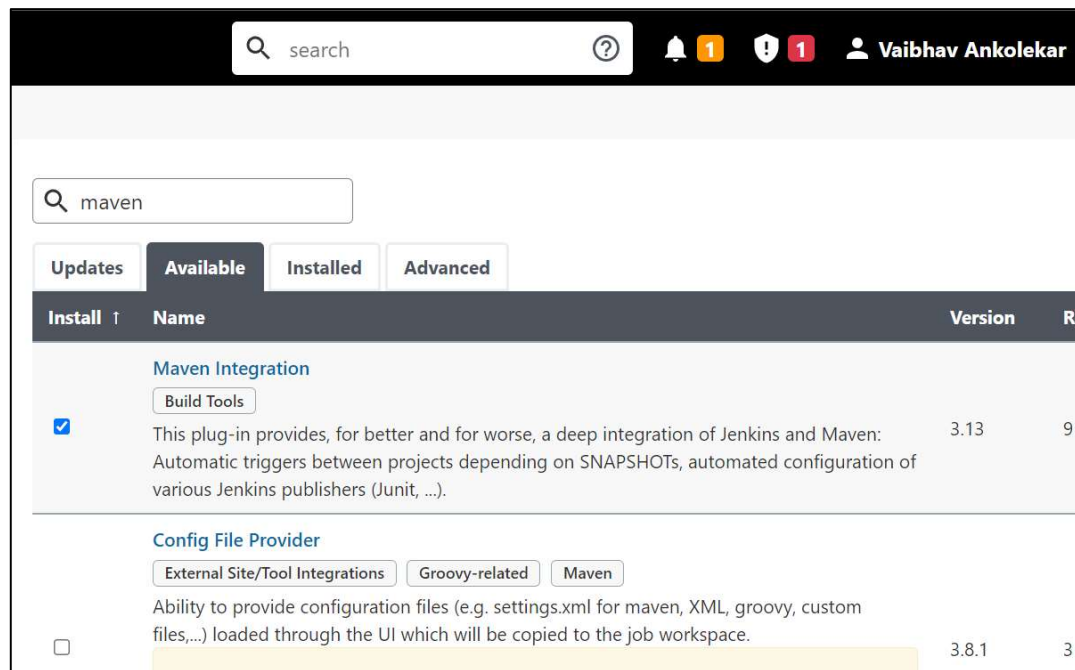
1) In your Jenkins, go to Manage Jenkins



2) Now go to Manage Plugins



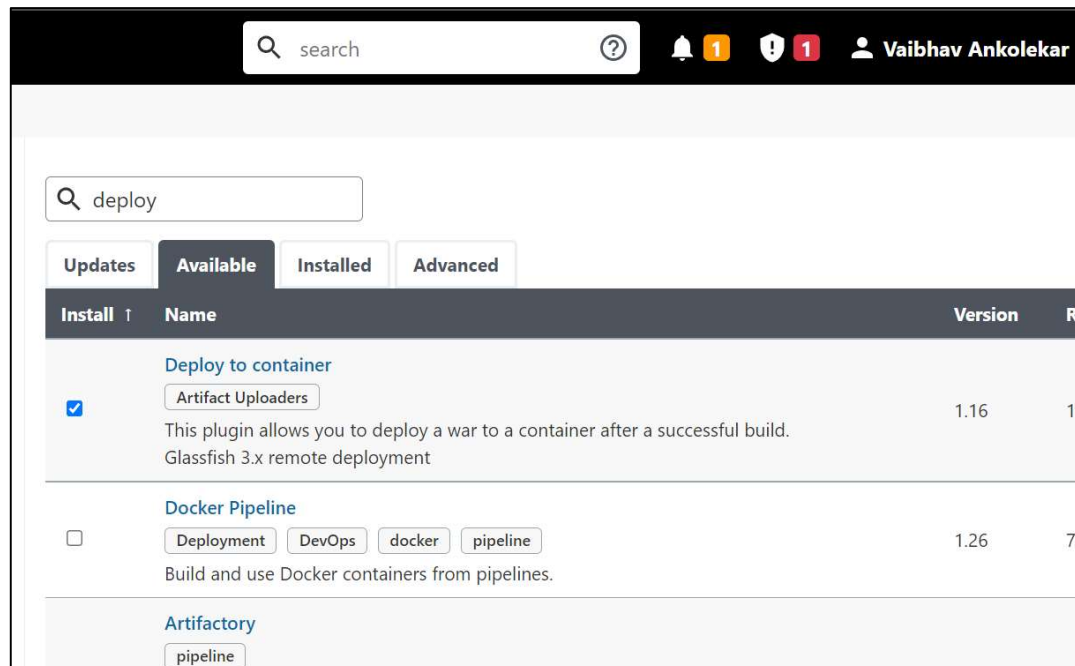
3) Install Maven Integration plugin.



The screenshot shows the Jenkins Plugins page with the search bar set to 'maven'. The 'Available' tab is selected. The 'Maven Integration' plugin is listed as installed (checked checkbox). The 'Config File Provider' plugin is listed as not installed (unchecked checkbox).

Install	Name	Version	Release
<input checked="" type="checkbox"/>	Maven Integration Build Tools This plug-in provides, for better and for worse, a deep integration of Jenkins and Maven: Automatic triggers between projects depending on SNAPSHOTS, automated configuration of various Jenkins publishers (Junit, ...).	3.13	9
<input type="checkbox"/>	Config File Provider External Site/Tool Integrations Groovy-related Maven Ability to provide configuration files (e.g. settings.xml for maven, XML, groovy, custom files,...) loaded through the UI which will be copied to the job workspace.	3.8.1	3

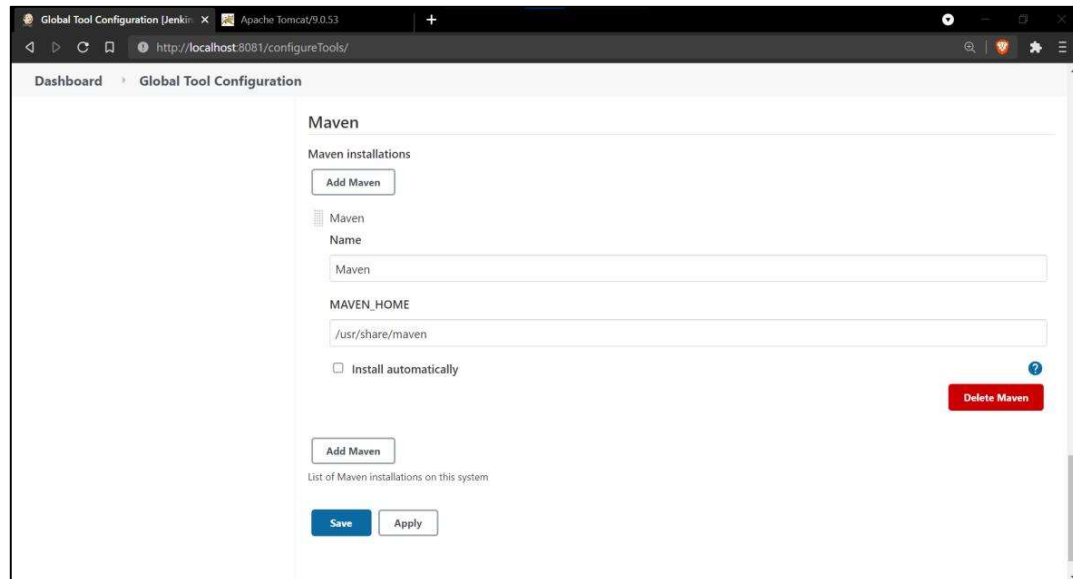
4) Install Deploy to container plugin.



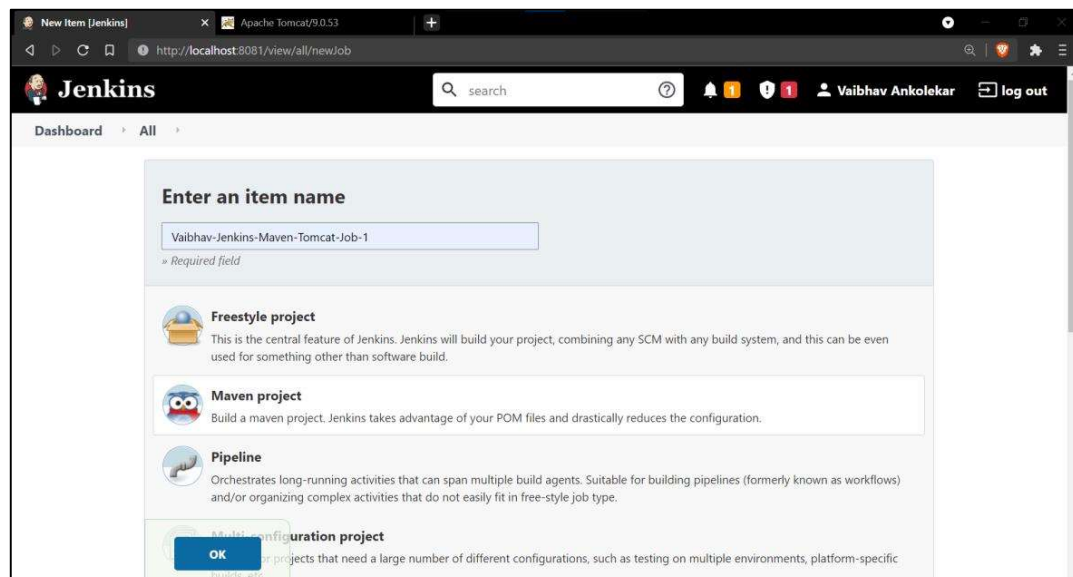
The screenshot shows the Jenkins Plugins page with the search bar set to 'deploy'. The 'Available' tab is selected. The 'Deploy to container' plugin is listed as installed (checked checkbox). The 'Docker Pipeline' plugin is listed as not installed (unchecked checkbox). The 'Artifactory' plugin is listed as not installed (unchecked checkbox).

Install	Name	Version	Release
<input checked="" type="checkbox"/>	Deploy to container Artifact Uploaders This plugin allows you to deploy a war to a container after a successful build. Glassfish 3.x remote deployment	1.16	1
<input type="checkbox"/>	Docker Pipeline Deployment DevOps docker pipeline Build and use Docker containers from pipelines.	1.26	7
<input type="checkbox"/>	Artifactory pipeline		

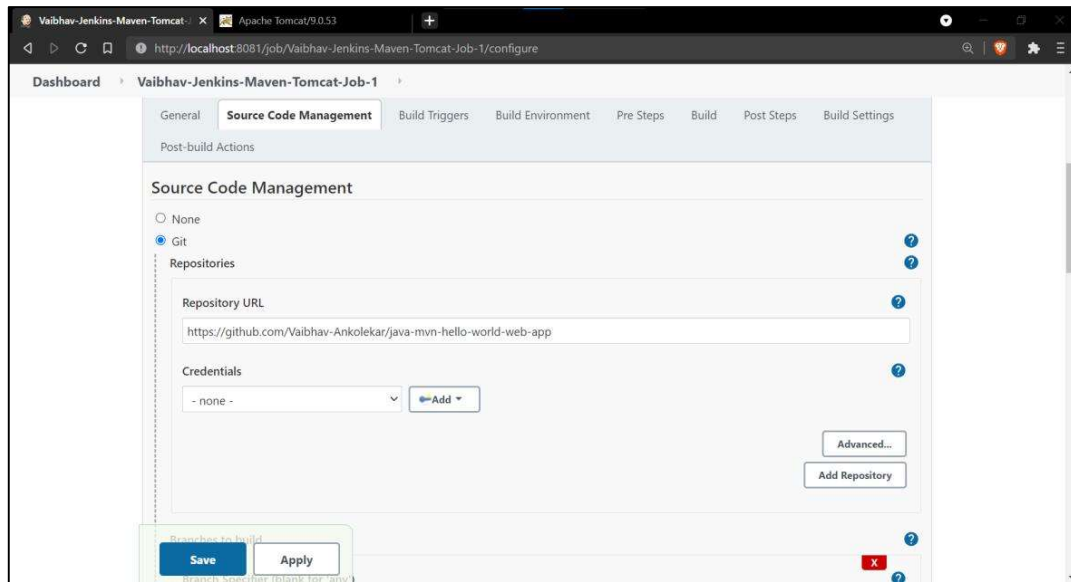
5) Go to Global Tool Configuration and add Maven. Give suitable name and the home directory of maven.



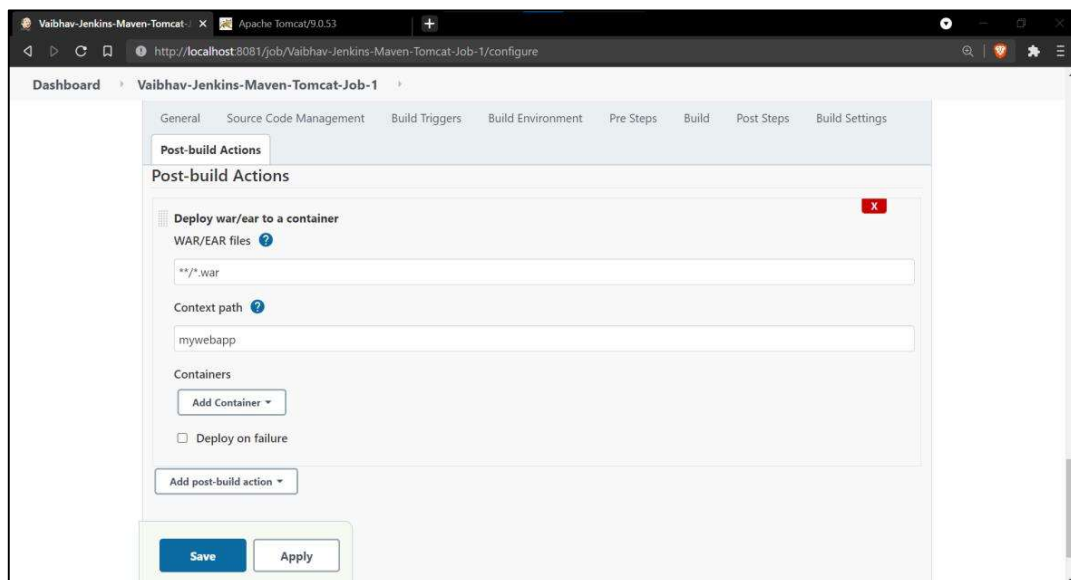
6) Now Go to Dashboard, and click on New Item. Enter Job Name and select Maven Project.



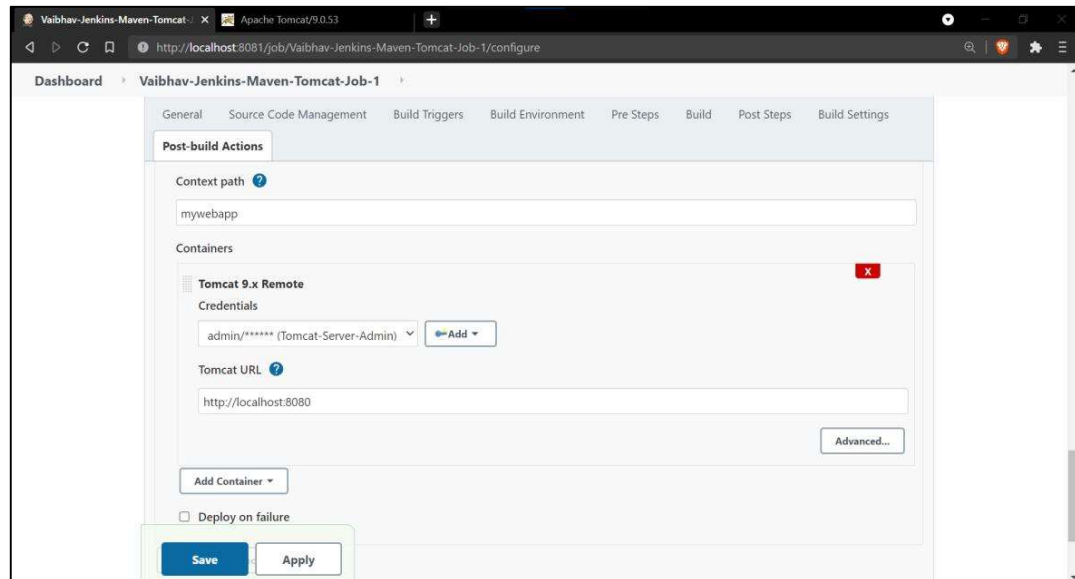
7) In Source Code Management, select Git and paste your Github repo url.



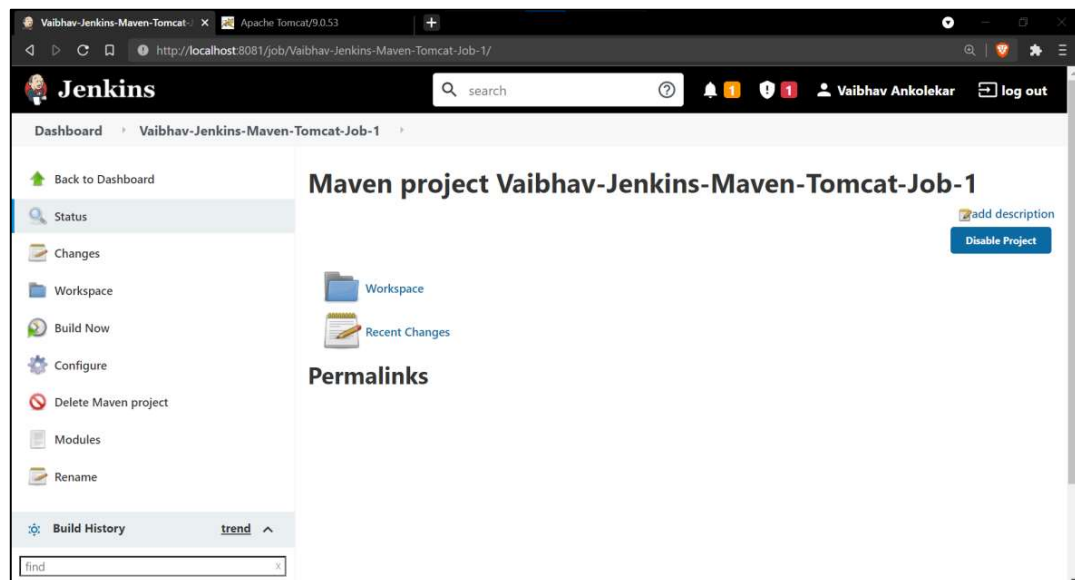
8) In Post-build Actions, click on Add post-build actions and select Deploy war/ear to a container. Enter the war file name which you want to deploy and enter a tomcat app name in Context path. Then click on Add container and select Tomcat version



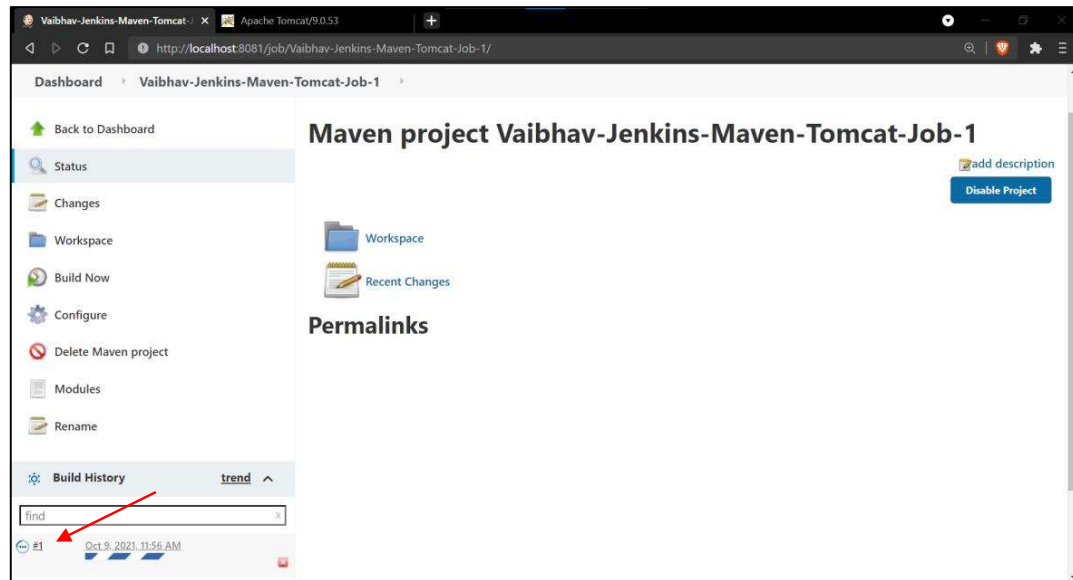
9) Click on Add under credentials and Enter your Tomcat user credentials and click on Create



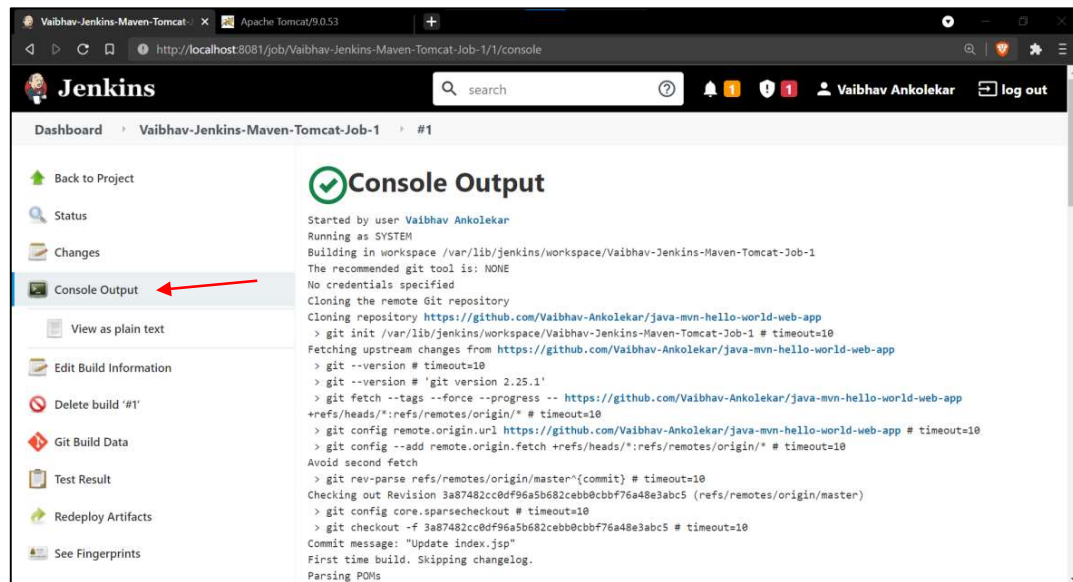
10) Click on Apply and Save. Now the job configuration is completed. Click on Build Now.



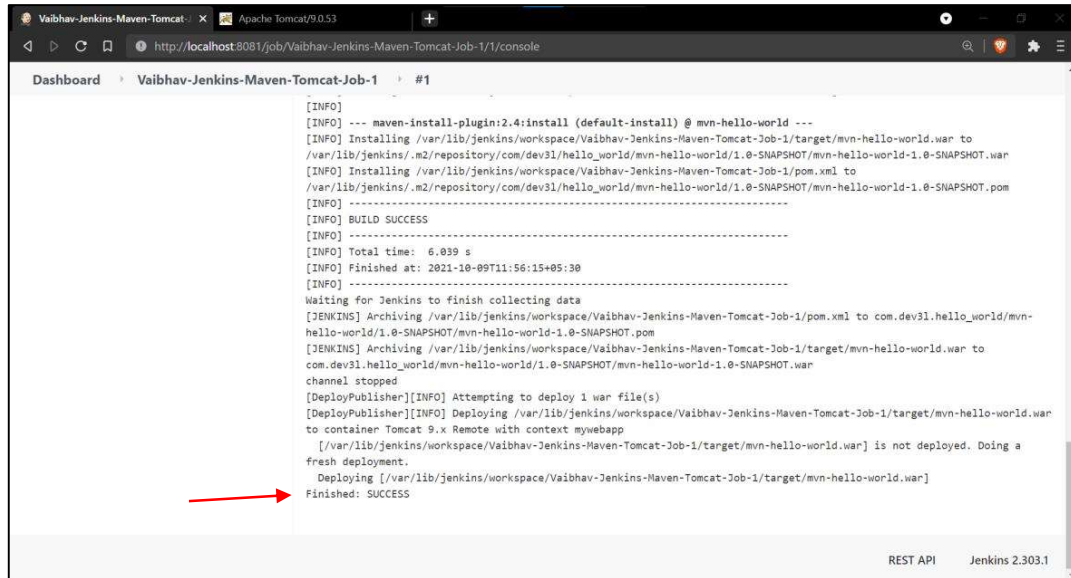
11) The job will start. Click on the job id under Build History.



12) Click on Console Output to view the logs.

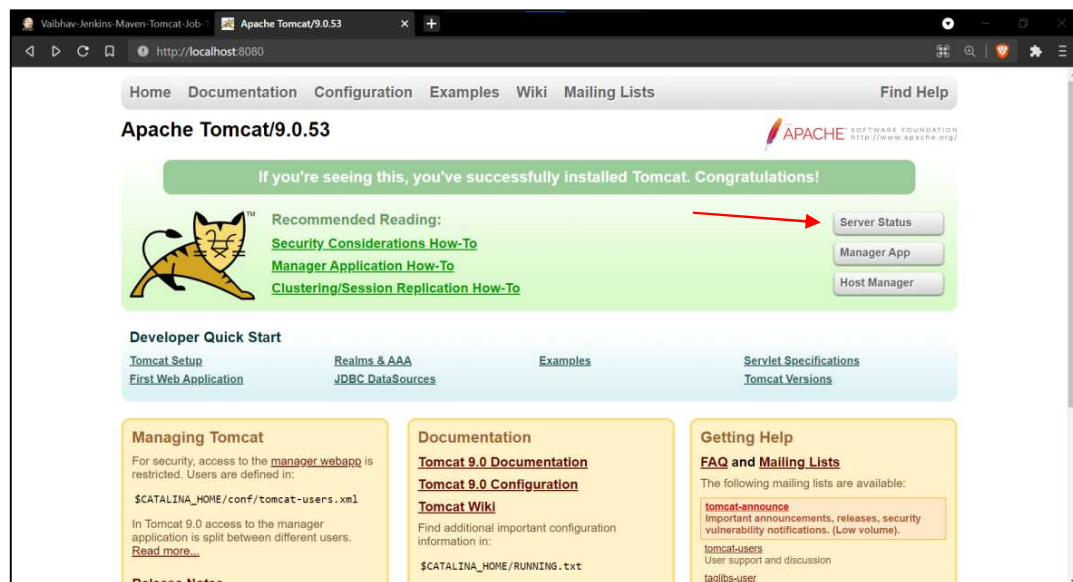


13) Go to bottom to see if the job is successfully completed.

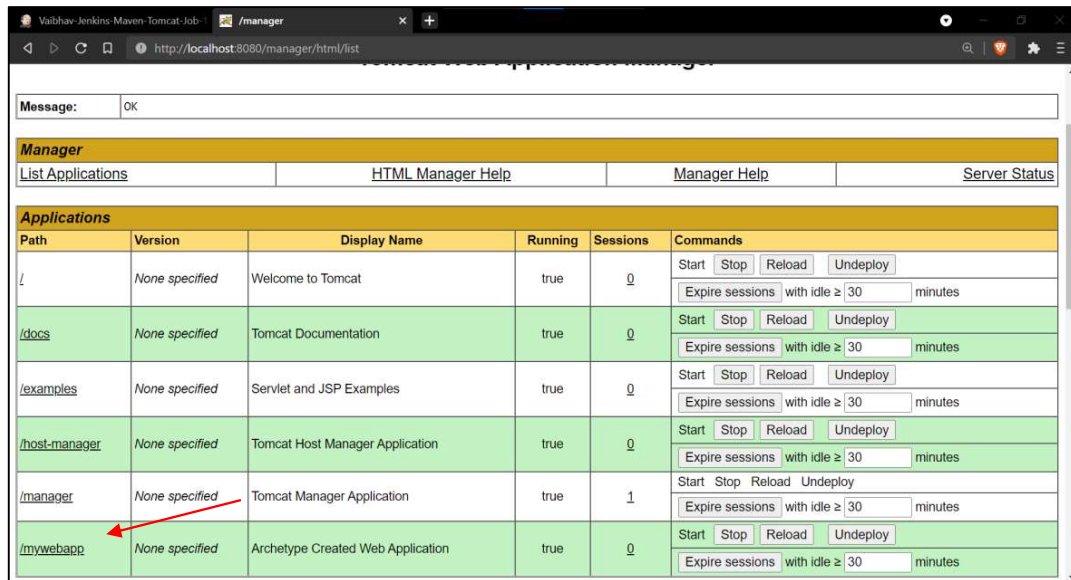


```
[INFO] --- maven-install-plugin:2.4:install (default-install) @ mvn-hello-world ---
[INFO] Installing /var/lib/jenkins/workspace/Vaibhav-Jenkins-Maven-Tomcat-Job-1/target/mvn-hello-world.war to
/var/lib/jenkins/.m2/repository/com/dev31/hello_world/mvn-hello-world/1.0-SNAPSHOT/mvn-hello-world-1.0-SNAPSHOT.war
[INFO] Installing /var/lib/jenkins/workspace/Vaibhav-Jenkins-Maven-Tomcat-Job-1/pom.xml to
/var/lib/jenkins/.m2/repository/com/dev31/hello_world/mvn-hello-world/1.0-SNAPSHOT/mvn-hello-world-1.0-SNAPSHOT.pom
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 6.039 s
[INFO] Finished at: 2021-10-09T11:56:15+05:30
[INFO] -----
Waiting for Jenkins to finish collecting data
[JENKINS] Archiving /var/lib/jenkins/workspace/Vaibhav-Jenkins-Maven-Tomcat-Job-1/pom.xml to com.dev31.hello_world/mvn-
hello-world/1.0-SNAPSHOT/mvn-hello-world-1.0-SNAPSHOT.pom
[JENKINS] Archiving /var/lib/jenkins/workspace/Vaibhav-Jenkins-Maven-Tomcat-Job-1/target/mvn-hello-world.war to
com.dev31.hello_world/mvn-hello-world/1.0-SNAPSHOT/mvn-hello-world-1.0-SNAPSHOT.war
channel stopped
[DeployPublisher][INFO] Attempting to deploy 1 war file(s)
[DeployPublisher][INFO] Deploying /var/lib/jenkins/workspace/Vaibhav-Jenkins-Maven-Tomcat-Job-1/target/mvn-hello-world.war
to container Tomcat 9.x Remote with context mywebapp
[/var/lib/jenkins/workspace/Vaibhav-Jenkins-Maven-Tomcat-Job-1/target/mvn-hello-world.war] is not deployed. Doing a
fresh deployment.
  Deploying [/var/lib/jenkins/workspace/Vaibhav-Jenkins-Maven-Tomcat-Job-1/target/mvn-hello-world.war]
Finished: SUCCESS
```

14) Go to Tomcat, and click on Server Status.



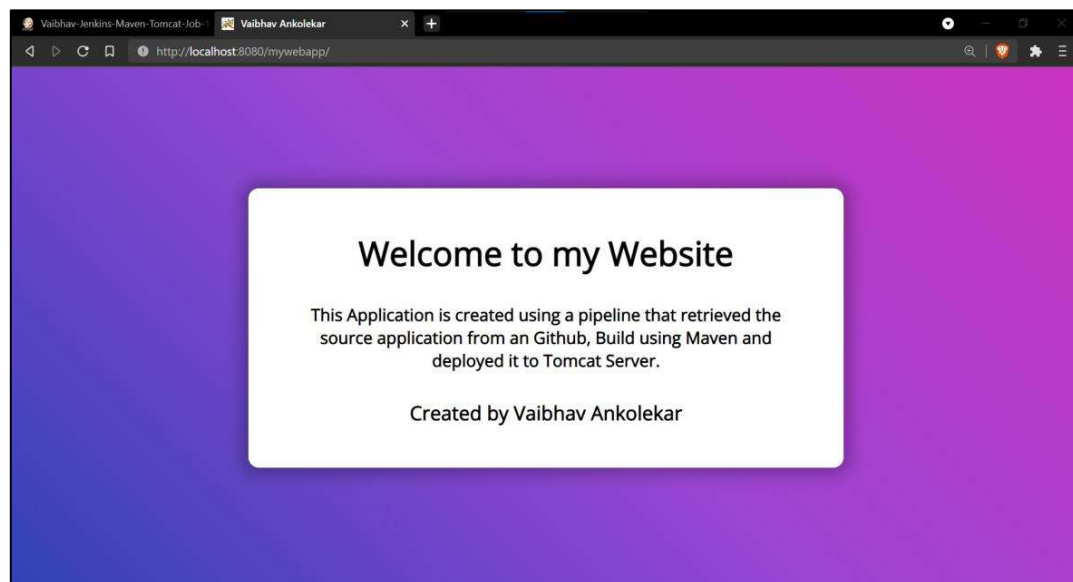
15) Scroll down and click on your app name under Applications Path column.



The screenshot shows the Tomcat Manager web interface. At the top, there's a message bar with 'OK'. Below it, a navigation bar includes 'List Applications', 'HTML Manager Help', 'Manager Help', and 'Server Status'. The main section is titled 'Applications' and contains a table with columns: Path, Version, Display Name, Running, Sessions, and Commands. The table lists several applications, including 'Welcome to Tomcat', 'Tomcat Documentation', 'Servlet and JSP Examples', 'Tomcat Host Manager Application', 'Tomcat Manager Application', and 'Archetype Created Web Application' (mywebapp). A red arrow points to the 'mywebapp' entry in the 'Path' column.

Path	Version	Display Name	Running	Sessions	Commands
/	None specified	Welcome to Tomcat	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/docs	None specified	Tomcat Documentation	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/examples	None specified	Servlet and JSP Examples	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/host-manager	None specified	Tomcat Host Manager Application	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/manager	None specified	Tomcat Manager Application	true	1	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/mywebapp	None specified	Archetype Created Web Application	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes

16) Your app is deployed on the Tomcat Server.



Conclusion : Thus, we have successfully build the pipeline of jobs using Maven in Jenkins, and deployed an application over the tomcat server