2. Continuous Integration with Jenkins:

Tool: Jenkins, Tomcat

Program:

- **Install and configure Jenkins.**
- **Create a simple pipeline to:**
  - **Clone a Git repository.**
  - **Build a sample project (e.g., a Java Maven or Python project).**

Run test cases and archive the results.

- **Install Jenkins and Tomcat on Windows(we have uploaded videos for installation of Jenkins and Tomcat in GCR)**

Step-1: Create maven project with .war format

Step-2: Create one html/jsp file in "src/main/webapp/index.html" and add

HTML code to it.

Step-3: Create "src/main/webapp/WEB-INF" folder. In that

"src/main/webapp/WEB-INF/web.xml" file.

Step-4: Include below plugin in "web.xml"

```
<plugin>
<groupId>org.apache.maven.plugins</groupId>
<artifactId>maven-war-plugin</artifactId>
<configuration>
<webXml>src\main\webapp\WEB-INF\index.jsp</webXml> (mention which
page to start)
</configuration>
</plugin>
```

Step-5: Update project – Maven clean install compile test (make sure build

Successful)

<u>Adding Project to git:</u>

Open terminal/command prompt, navigate to project directory and run

"git init"

Step 1: Create a GitHub Account and create new repository

Step 2: Install Git on Your PC, Configure your name and email

Step 3: Initialize Git in Eclipse:

Open Eclipse and go to your Maven project in the Project Explorer.

Right-click the project → Team → Share Project.

Select Git, then click Next.

Click Create to create a new local Git repository.

Click Finish.

Step 4: Add & Commit Files

Right-click the project → Team → Add to Index (this stages all files for commit).

Right-click again → Team → Commit.

Enter a commit message like "Initial commit" and click Commit.

Adding Project to git

Step 5: Connect to GitHub and Push

Copy your GitHub repository URL (from the GitHub page where you created the repository).

In Eclipse:

Right-click the project → Team → Remote → Push.

Click Create Remote.

Enter origin as the remote name.

Paste the GitHub repository URL.

Click Next.

Select Branch to Push:

Source ref: master (or main, depending on GitHub).

Destination ref: master (or main).

Click Next, then Finish.

Enter your GitHub username and personal access token (create using GitHub).

Step 6: Verify on GitHub

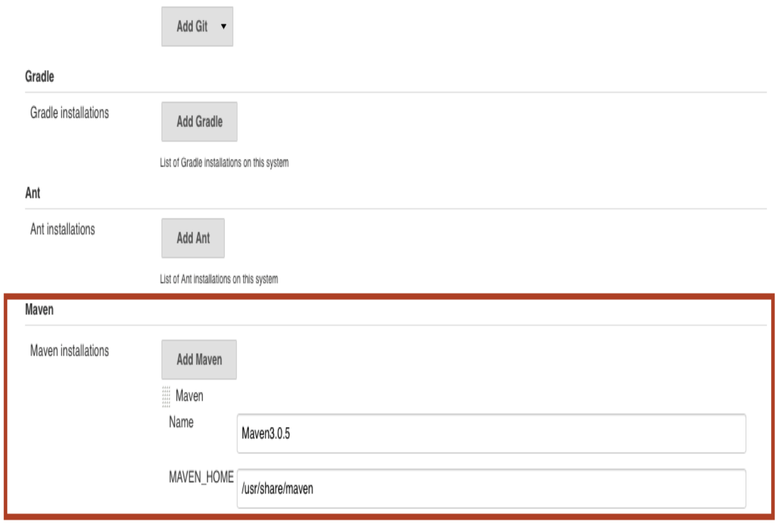
Go to your GitHub repository page and refresh. You should see your project files uploaded!

Note: Next time when you modify your project in eclipse follow step 4 and 5 with commit message as "updated version".

Jenkins Configuration

Step1: Make Sure you have Git and Maven installed

In Jenkins UI, Goto Manage Jenkins -> Global Tool Configuration Section of Jenkins->Add maven configuration->give maven version name and select maven version from the dropdown menu.

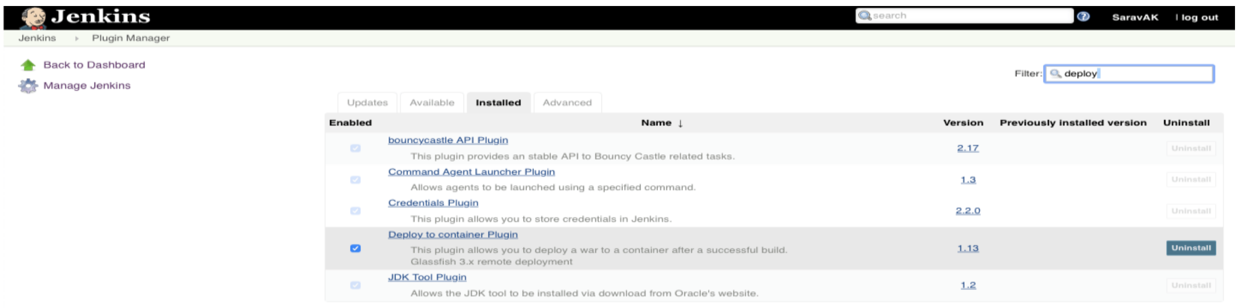


Name: Maven3.0.5

MAVEN_HOME: /usr/share/maven

Step2: Install maven integration, Deploy to Container and git & GitHub Plugin.

Manage Jenkins -> Manage Plugins -> Available -> Deploy to Container Plugin, maven integration & git & GitHub Plugin.



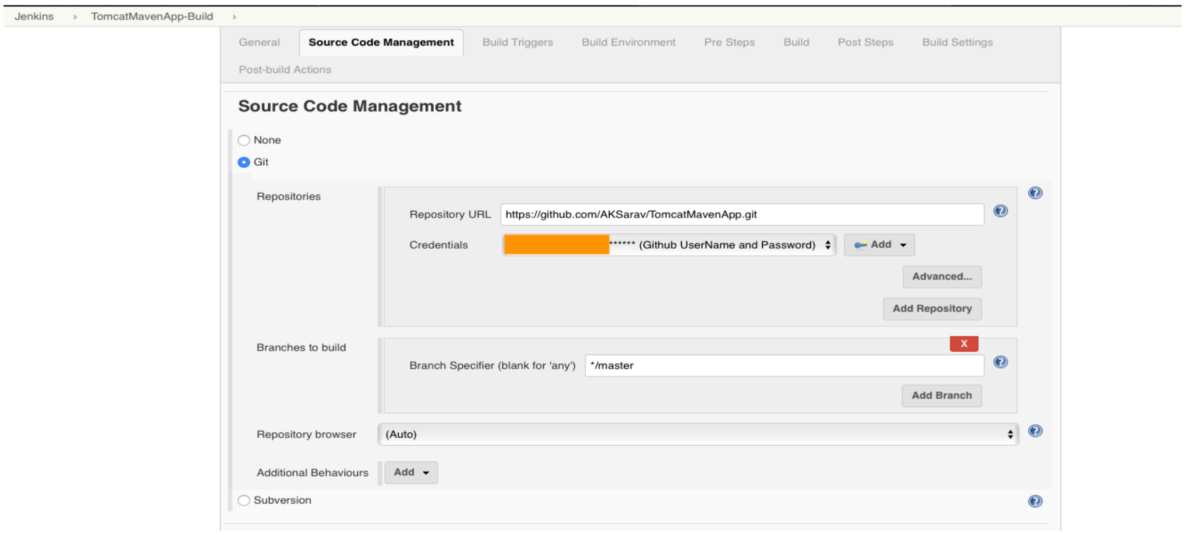
Filter: deploy

Enabled	Name ↓	Version	Previously installed version	Uninstall
✓	bouncycastle API Plugin This plugin provides an stable API to Bouncy Castle related tasks.	2.17		Uninstall
✓	Command Agent Launcher Plugin Allows agents to be launched using a specified command.	1.3		Uninstall
✓	Credentials Plugin This plugin allows you to store credentials in Jenkins.	2.2.0		Uninstall
✓	Deploy to container Plugin This plugin allows you to deploy a war to a container after a successful build. Glassfish 3.x remote deployment	1.13		Uninstall
✓	JDK Tool Plugin Allows the JDK tool to be installed via download from Oracle's website.	1.2		Uninstall

Step3: Create and Configure a Maven Job with Source Code Management (Github)

New Item -> Select Free Style Project

- In the Configuration Section, Under Source Code Management Fill your <u>Gitlab Repository URL (Create public Repository in GitHub).</u>



Source Code Management

None
[✓ Git]
Subversion

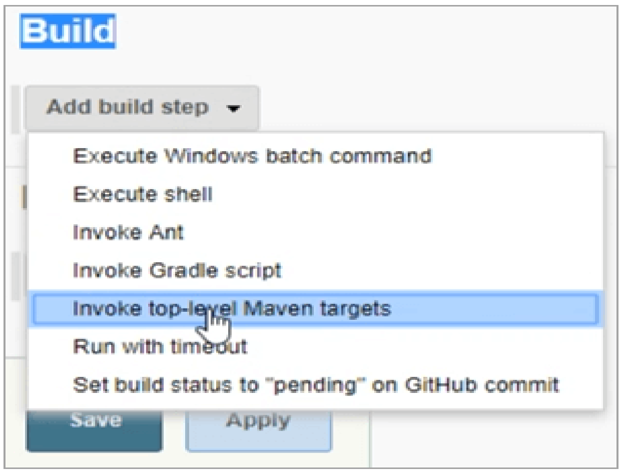
Repositories
Repository URL: https://github.com/AKSarav/TomcatMavenApp.git
Credentials: [redacted]****** (Github UserName and Password)

Branches to build
Branch Specifier (blank for 'any'): */master

Repository browser: (Auto)

Additional Behaviours

Step 4: Next, we need to move to the Build section and select Invoke top-level Maven target options from the dropdown.


Build
Add build step
Execute Windows batch command
Execute shell
Invoke Ant
Invoke Gradle script
Invoke top-level Maven targets
Run with timeout
Set build status to "pending" on GitHub commit
Save
Apply

- Maven Version will be pre-populated from the Maven version we have defined in the Global Tool Configuration.

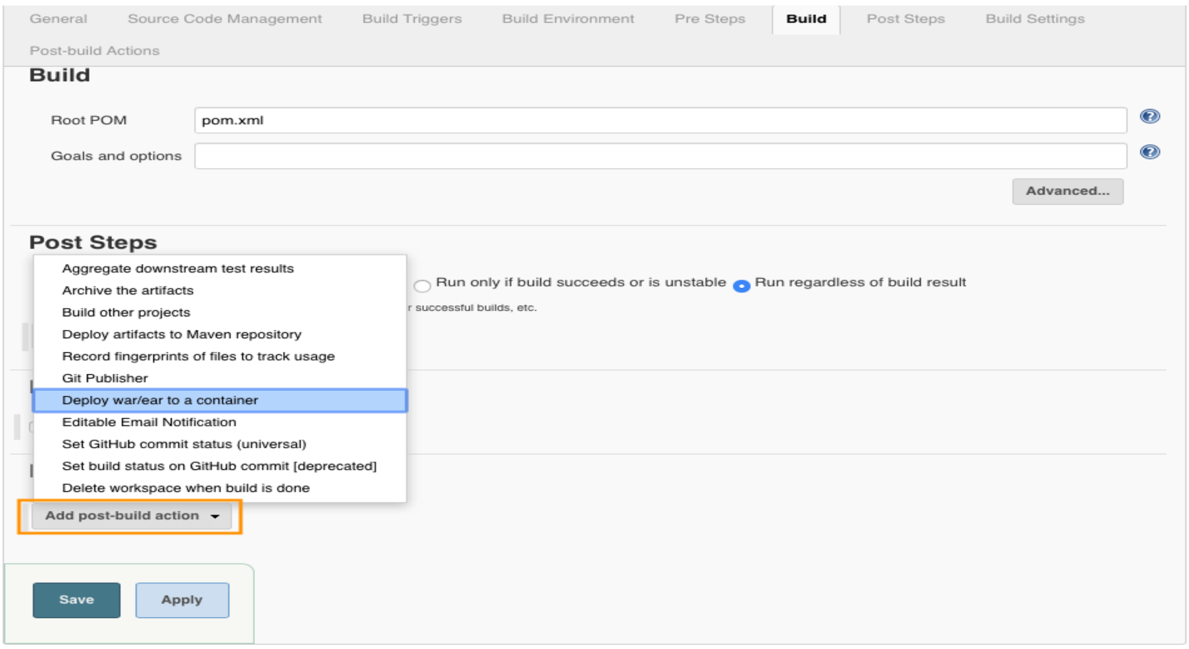
- Under Goals, we have to provide the Maven command to trigger the execution. Maven <u>Clean and install</u>

Step 5 : Configure the Post-build Action and Specify the Tomcat Server Details

Drag to the bottom and Go to the Post-build Actions section

Click on Add post-build action button

On the available options click on the Deploy war/ear to container


General
Source Code Management
Build Triggers
Build Environment
Pre Steps
Build
Post Steps
Build Settings
Post-build Actions
Aggregate downstream test results
Archive the artifacts
Build other projects
Deploy artifacts to Maven repository
Record fingerprints of files to track usage
Git Publisher
Deploy war/ear to a container
Editable Email Notification
Set GitHub commit status (universal)
Set build status on GitHub commit [deprecated]
Delete workspace when build is done
Add post-build action
Advanced...
Save
Apply

Build

Root POM: pom.xml

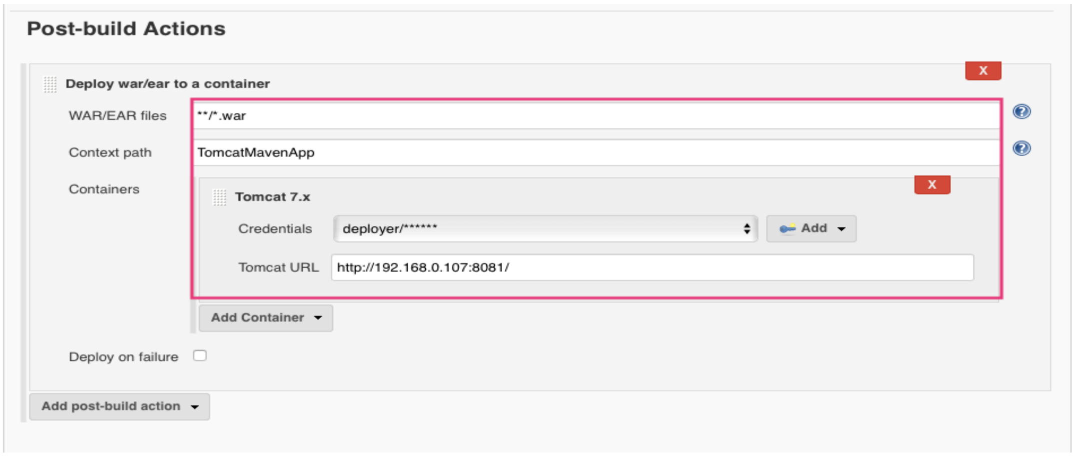
Goals and options:

Post Steps

Run only if build succeeds or is unstable [✓ Run regardless of build result]

Choose the Context Path in which the application should be installed. It would rename the WAR file before deploying to the server and thereby the application context root would be changed.

Tomcat URL http://[Tomcat Server Host]:[Primary http port]/(http://localhost:portnumber)



Post-build Actions

Deploy war/ear to a container

WAR/EAR files: **/*.war

Context path: TomcatMavenApp

Containers: Tomcat 7.x

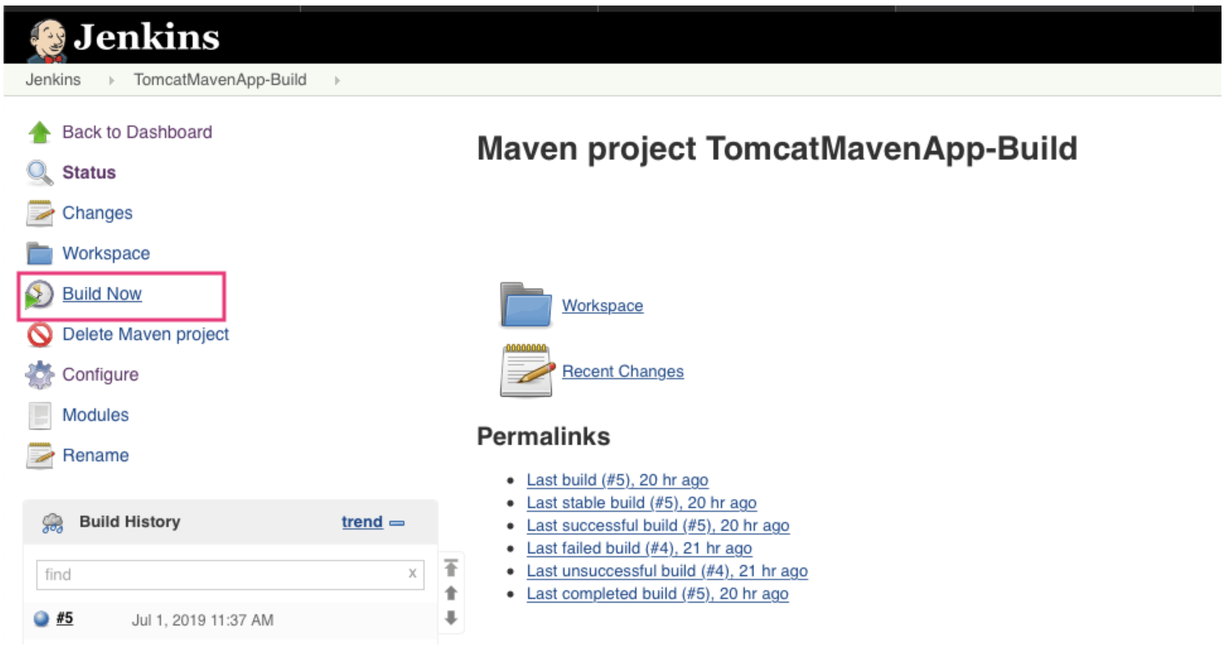
Credentials: deployer/******

Tomcat URL: http://192.168.0.107:8081/

Deploy on failure: ☐

Build Jenkins Job

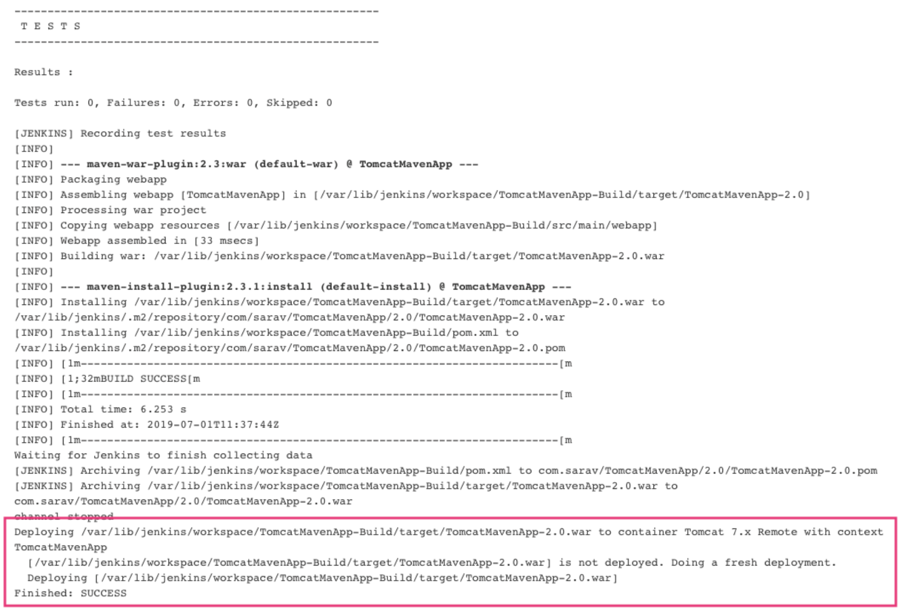
Execute the Job you have created by clicking on the Build Now button



- Last build (#5), 20 hr ago
- Last stable build (#5), 20 hr ago
- Last successful build (#5), 20 hr ago
- Last failed build (#4), 21 hr ago
- Last unsuccessful build (#4), 21 hr ago
- Last completed build (#5), 20 hr ago

Console Output after the Successful build.

At the last line you can see that the WAR file has been generated and deployed on the remote server.



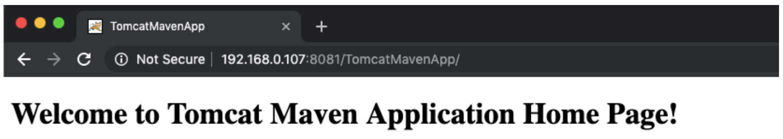
```
-------------------------------------------------------
 T E S T S
-------------------------------------------------------

Results :

Tests run: 0, Failures: 0, Errors: 0, Skipped: 0

[JENKINS] Recording test results
[INFO]
[INFO] --- maven-war-plugin:2.3:war (default-war) @ TomcatMavenApp ---
[INFO] Packaging webapp
[INFO] Assembling webapp [TomcatMavenApp] in [/var/lib/jenkins/workspace/TomcatMavenApp-Build/target/TomcatMavenApp-2.0]
[INFO] Processing war project
[INFO] Copying webapp resources [/var/lib/jenkins/workspace/TomcatMavenApp-Build/src/main/webapp]
[INFO] Webapp assembled in [33 msecs]
[INFO] Building war: /var/lib/jenkins/workspace/TomcatMavenApp-Build/target/TomcatMavenApp-2.0.war
[INFO]
[INFO] --- maven-install-plugin:2.3.1:install (default-install) @ TomcatMavenApp ---
[INFO] Installing /var/lib/jenkins/workspace/TomcatMavenApp-Build/target/TomcatMavenApp-2.0.war to
/var/lib/jenkins/.m2/repository/com/sarav/TomcatMavenApp/2.0/TomcatMavenApp-2.0.war
[INFO] Installing /var/lib/jenkins/workspace/TomcatMavenApp-Build/pom.xml to
/var/lib/jenkins/.m2/repository/com/sarav/TomcatMavenApp/2.0/TomcatMavenApp-2.0.pom
[INFO] [1m------------------------------------------------------------------------[m
[INFO] [1;32mBUILD SUCCESS[m
[INFO] [1m------------------------------------------------------------------------[m
[INFO] Total time: 6.253 s
[INFO] Finished at: 2019-07-01T11:37:44Z
[INFO] [1m------------------------------------------------------------------------[m
Waiting for Jenkins to finish collecting data
[JENKINS] Archiving /var/lib/jenkins/workspace/TomcatMavenApp-Build/pom.xml to com.sarav/TomcatMavenApp/2.0/TomcatMavenApp-2.0.pom
[JENKINS] Archiving /var/lib/jenkins/workspace/TomcatMavenApp-Build/target/TomcatMavenApp-2.0.war to
com.sarav/TomcatMavenApp/2.0/TomcatMavenApp-2.0.war
channel stopped
Deploying /var/lib/jenkins/workspace/TomcatMavenApp-Build/target/TomcatMavenApp-2.0.war to container Tomcat 7.x Remote with context
TomcatMavenApp
  [/var/lib/jenkins/workspace/TomcatMavenApp-Build/target/TomcatMavenApp-2.0.war] is not deployed. Doing a fresh deployment.
  Deploying [/var/lib/jenkins/workspace/TomcatMavenApp-Build/target/TomcatMavenApp-2.0.war]
Finished: SUCCESS
```

Testing the Application

As the deployment is completed and the Jenkins Job ran Successfully without issues.

Let us test our application. Got to your web browser and type the URL should be as follows

http://localhost:8081/projectname



Welcome to Tomcat Maven Application Home Page!