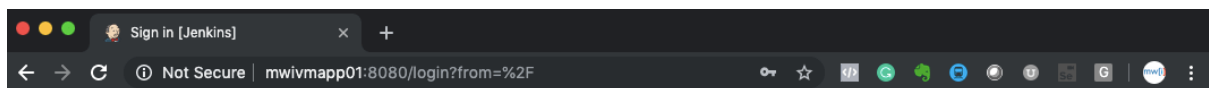


# Jenkins Tomcat Deploy – Deploying Application to Tomcat using Jenkins

We are going to see how to pull the code from the Source Code Management Repository – GITHUB and deploy it to Tomcat Application Server ( Servlet Container). Inclusive of all the configuration required in Jenkins and Tomcat end.

Our Ultimate Subjective is Jenkins Tomcat Integration and How to Deploy Application to Tomcat Using Jenkins. Jenkins Tomcat Deployment.

Let's Proceed.



Welcome to Jenkins!

Sign in

☐ Keep me signed in

## Tomcat Configuration

We presume that you have downloaded and started the tomcat server with no issues. Now we will advance to the next level which is making Tomcat ready to receive deployments and enabling TEXT based and GUI based management interface.

All the tomcat application servers come with the manager application by default. If you look at the webapps directory of your downloaded tomcat server. You can see it.

At the first time, you may not be able to access it cause the manager application needs some security elements to be configured prior to being accessed.

### Environment Configuration

Let us taken my tomcat7 setup as an example and I will paste the steps what I did to make it working.

I have installed my Tomcat7 in CENTOS server at /apps/tomcat/tomcat7 directory and this is my CATALINA\_HOME Now you need to find your CATALINA\_HOME before continuing with the next steps.

### Update Roles and User credentials – Configuring Tomcat Security

Got to \$CATALINA\_HOME/conf directory and look for tomcat-users.xml file.

Open the file in your favourite editor like VI or nano and add the following lines right before the last line

```
<user username="tomcatmanager" password="password" roles="manager-gui"/>
```

```
<user username="deployer" password="password" roles="manager-script"/>
```

Here we are creating two usernames named `tomcatmanager` and `deployer`. here the `deployer` account would be used to deploy the WAR file over http.

`manager-gui` based `tomcatmanager` user would be used to manage the manager web application at `http://<HostName>:8080/manager`

Note\*: If you are using Tomcat8+ version, Steps to enable manager application might be different. Refer the product version specific documentation if you get stuck.

Now we are ready with the Tomcat Servlet Container aka Application server and it is ready to be connected from Jenkins.

## Jenkins Configuration

I presume that you have a Running Jenkins Server and a Administration Access

### Step1: Make Sure you have Git and Maven installed

In Jenkins UI, Goto `Manage Jenkins -> Global Tool Configuration` Section of Jenkins

Note\*: If you are not able to see a Manage Jenkins Option, You need to consult with your Administrator. It might be due to insufficient privileges.

Jenkins > Global Tool Configuration

**Git**

Git installations

Name: Default

Path to Git executable: git

☒ Install automatically

Add Installer

Delete Git

Add Git

**Gradle**

Gradle installations

Add Gradle

List of Gradle installations on this system

**Ant**

Ant installations

Add Ant

List of Ant installations on this system

**Maven**

Maven installations

Add Maven

Name: Maven3.0.5

MAVEN\_HOME: /usr/share/maven

## Step2: Install Deploy to Container Plugin.

Manage Jenkins -> Manage Plugins -> Available -> Deploy to Container Plugin

Jenkins

Search

SaravAK | log out

Jenkins > Plugin Manager

Back to Dashboard

Manage Jenkins

Filter: deploy

Enabled	Updates	Available	Installed	Advanced	Name	Version	Previously installed version	Uninstall
<input checked="" type="checkbox"/>					<a href="#">bouncycastle API Plugin</a>	2.17		Uninstall
					This plugin provides an stable API to Bouncy Castle related tasks.			
<input checked="" type="checkbox"/>					<a href="#">Command Agent Launcher Plugin</a>	1.3		Uninstall
					Allows agents to be launched using a specified command.			
<input checked="" type="checkbox"/>					<a href="#">Credentials Plugin</a>	2.2.0		Uninstall
					This plugin allows you to store credentials in Jenkins.			
<input checked="" type="checkbox"/>					<a href="#">Deploy to container Plugin</a>	1.13		Uninstall
					This plugin allows you to deploy a war to a container after a successful build. Glassfish 3.x remote deployment			
<input checked="" type="checkbox"/>					<a href="#">JDK Tool Plugin</a>	1.2		Uninstall
					Allows the JDK tool to be installed via download from Oracle's website.			

Note:\* For the Next step we have selected a Maven Job as our Choice. you can also create a Free Style Project and use Gradle or Ant as your build tool .

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

New Item -> Maven Project

In the Configuration Section, Under Source Code Management Fill your Github/BeanStalk/Gitlab Repository URL

For testing you can use one of our Sample Application Named Tomcat Maven App from our Github public Repository.

[You can use this GITHUB Repository](#)

Click on **Add** button displayed near the Credentials drop-down and enter the username and password of your SCM Repo and Once it is saved. It would be available on the Dropdown for you to select.

Once you have selected the valid Credentials to wait for few seconds. If you see any error message in RED colour which means your connection to the SCM repository is unsuccessful. Check the URL or the credentials and retry.

The screenshot shows the Jenkins configuration page for a project named 'TomcatMavenApp-Build'. The 'Source Code Management' tab is selected. Under 'Source Code Management', the 'Git' option is chosen. The 'Repositories' section contains a table with one entry: 'Repository URL' is 'https://github.com/AKSarav/TomcatMavenApp.git' and 'Credentials' is a dropdown menu with a red 'X' icon and an 'Add' button. Below the table, the 'Branches to build' section has a 'Branch Specifier (blank for \'any\')' field with the value '\*/\*' and an 'Add Branch' button. The 'Repository browser' is set to '(Auto)'. The 'Additional Behaviours' section has an 'Add' button. The 'Subversion' option is not selected.

### Step4: 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**

The screenshot shows the Jenkins configuration interface for the 'Post-build Actions' section. The 'Build' tab is selected, and the 'Post Steps' section is expanded. A dropdown menu is open, showing various post-build actions, with 'Deploy war/ear to a container' highlighted. Below the dropdown is a button labeled 'Add post-build action'. The 'Root POM' field is set to 'pom.xml', and the 'Goals and options' field is empty. The 'Advanced...' button is visible to the right of the 'Goals and options' field. At the bottom of the page, there are 'Save' and 'Apply' buttons.

Fill the required parameters for the plugin. Use the following Screen Shot as the reference

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]/`

### Post-build Actions

Deploy war/ear to a container

WAR/EAR files\*\*/\*.war

Context pathTomcatMavenApp

Containers

Tomcat 7.x

Credentialsdeployer/\*\*\*\*\*Add

Tomcat URLhttp://192.168.0.107:8081/


Add Container

Deploy on failure☐

Add post-build action

## Build Jenkins Job

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

**Jenkins**

Jenkins > TomcatMavenApp-Build >

Back to Dashboard

Status

Changes

Workspace

**Build Now**

Delete Maven project

Configure

Modules

Rename

Build History

find

#5 Jul 1, 2019 11:37 AM

### Maven project TomcatMavenApp-Build

Workspace

Recent Changes

#### Permalinks

- 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.

in our case, <http://192.168.0.107:8081/>

```
-----
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

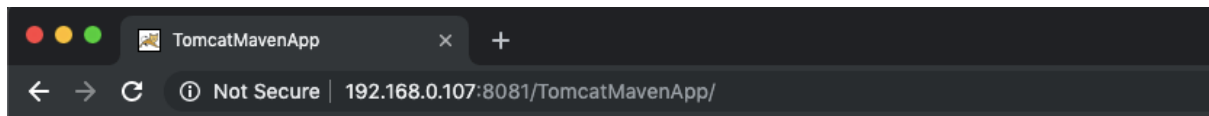
As the deployment is completed and the Jenkins Job ran Successfully without issues. ( Or So I presume)

Let us test our application.

In my case, the URL should be as follows

<http://192.168.0.107:8081/TomcatMavenApp>





## Welcome to Tomcat Maven Application Home Page!

That's all. You have successfully configured Tomcat with Jenkins Continuous Deployment .