

Installing Tomcat on Centos 7

Install Tomcat

Installing Tomcat on CentOS 7 requires one simple command:

```
$ sudo yum install tomcat
```

This will install Tomcat and its dependencies, including Java.

There are several additional packages which many users, particularly those who are new to Tomcat, will find useful. Install them with the command:

```
$ sudo yum install tomcat-webapps tomcat-admin-webapps tomcat-docs-webapp tomcat-javadoc
```

If your server is running Apache, stop it with the command:

```
sudo systemctl stop httpd
```

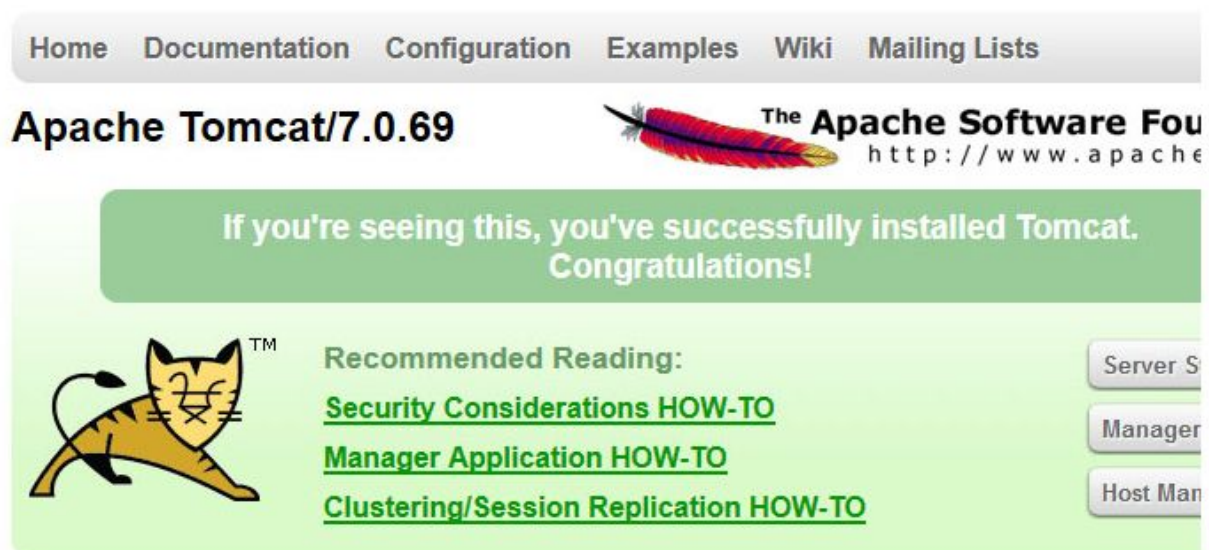
Start Tomcat with the command:

```
sudo systemctl start tomcat
```

And enable Tomcat to automatically start if the server is rebooted:

```
sudo systemctl enable tomcat
```

You can verify that Tomcat is running by visiting the **URL <http://example.com:8080>** in a web browser. You will see the Tomcat welcome page, which includes links to the Tomcat documentation which you installed in the previous step.



Step-by-step Jenkins Tomcat deploy of a WAR file

various steps required to get a [Jenkins](#) pipeline to deploy to Tomcat after a build with a WAR file.

Jenkins Tomcat deploy prerequisites

The following pieces of software are required to follow this example of a Jenkins deployment of a WAR file to Tomcat:

- a Java web application that Jenkins can deploy to Tomcat;
- a Git source code management tool installation;
- a Java 8 or newer installation of the JDK;
- a build tool -- this tutorial [uses Maven](#), but any build tool, such as Ivy, ANT or Gradle, that can package a Java application into a WAR file will do; and
- a Jenkins CI tool installation.

A WAR file for Jenkins to deploy

If you need a web application for Jenkins to deploy to Tomcat, feel free to clone my rock-paper-scissors Java web app from GitHub:

```
git clone
https://github.com/pavansw/rock-paper-scissors.git

cd rock*
```

```
git checkout patch-1
```

Pay special attention to the last command, where you will switch to the patch-1 branch. The master branch creates a [JAR file](#), with Tomcat embedded within. But what we need is the patch-1 branch, which creates a WAR file that can be deployed to Tomcat by Jenkins.

Step 1: Add to Tomcat a user with deployment rights

For a successful Jenkins Tomcat deploy of a WAR file, you must add a new user to Tomcat with `manager-script` rights. You can do this with an edit of the `tomcat-users.xml` file, which can be found in Tomcat's `conf` directory.

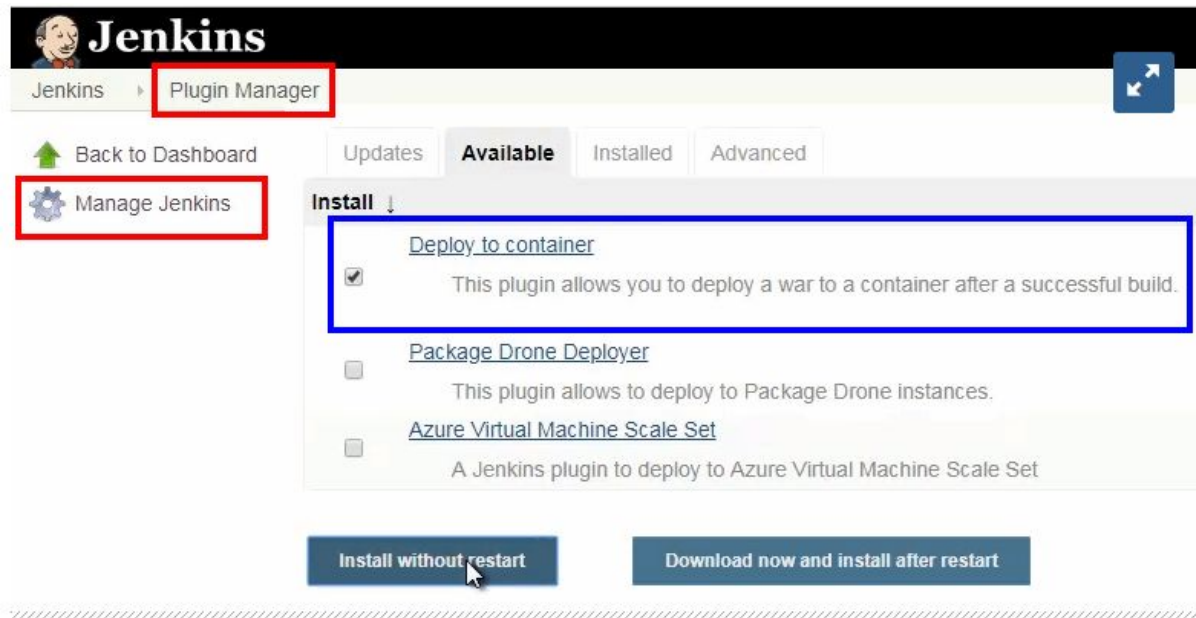
```
<!-- User to deploy WAR file from Jenkins to Tomcat -->  
  
<user username="war-deployer"  
password="jenkins-tomcat-deploy" roles="manager-script"  
>
```

After you edit the `tomcat-users.xml` file, it's a good idea to bounce the Tomcat server to confirm the changes have taken effect.

Step 2: Add the 'Deploy WAR/EAR to a container Jenkins' plugin

Out of the box, there are no built-in features that perform a Jenkins WAR file deployment to Tomcat. That means a Jenkins Tomcat deploy plugin must be installed in the CI tool to make a deployment happen.

The most popular Jenkins Tomcat deployment plugin is named *Deploy to container*, which can be installed through the Plugin Manager tab under the "Manage Jenkins" section of the tool.



Step 3: The Jenkins build job

With the Jenkins Tomcat deployment plugin installed, it's time to create a new Jenkins build job that can build an application and deploy a package WAR file to Tomcat.

Step 3A: Create a Jenkins freestyle project

The Jenkins build job we need to create will be named `deploy-war-from-jenkins-to-tomcat`, and it will be a freestyle project type.




Jenkins

Jenkins ▶

Enter an item name

» Required field

**Freestyle project**

This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system

Step 3B: Configure standard build job properties

The Jenkins build job will be configured with the following properties:

JDK: `java8`

Git Repository URL:

`https://github.com/pavansw/rock-paper-scissors.git`

Git branch specifier: `*/patch-1`

Maven Goals: `clean install`

The image shows two sections of the Jenkins configuration interface. The top section, titled 'Source Code Management', has 'Git' selected. The 'Repository URL' is 'https://github.com/rock-paper-scissors/rock-paper-scissors.git'. The 'Branches to build' section has a 'Branch Specifier (blank for 'any')' of '*/patch-1|'. The bottom section, titled 'Build', has 'Invoke top-level Maven targets' selected. The 'Maven Version' is 'maven' and the 'Goals' are 'clean install'. Both sections have a red 'X' icon in the top right corner.

Source Code Management	
Repositories	Repository URL: <code>https://github.com/rock-paper-scissors/rock-paper-scissors.git</code>
Branches to build	Branch Specifier (blank for 'any'): <code>*/patch-1 </code>

Build	
Invoke top-level Maven targets	
Maven Version	<code>maven</code>
Goals	<code>clean install</code>

Step 3C: The Jenkins Tomcat deploy plugin post-build action

After a build, the final step of a Jenkins pipeline deploy to Tomcat is to use the *Deploy to container* plugin in a post-build action.

Three of the four settings used by the *Deploy WAR/EAR to a container* plugin can be typed in directly:

WAR/EAR files: `**/*.war`

Context path: `app`

Containers: `Tomcat 8.x`

Tomcat URL: `http://Server3.test.com:8080`

Build

Invoke top-level Maven targets

- Aggregate downstream test results
- Archive the artifacts
- Build other projects
- Publish JUnit test result report
- Publish Javadoc
- Record fingerprints of files to track usage
- Use publishers from another project
- Git Publisher
- Deploy war/ear to a container
- E-mail Notification
- Trigger parameterized build on other projects

Add post-build action ▼

Post-build Actions

Deploy war/ear to a container

WAR/EAR files

Context path

Containers

Tomcat 8.x

Credentials Add

Tomcat URL

Add Container ▼

Deploy on failure ☐

Add post-build action ▼

To configure the credentials, you must click the Add button next to the empty entry field and create a new Jenkins credentials object:

The username and password need to match what was entered into the tomcat-users.xml file in an earlier step:

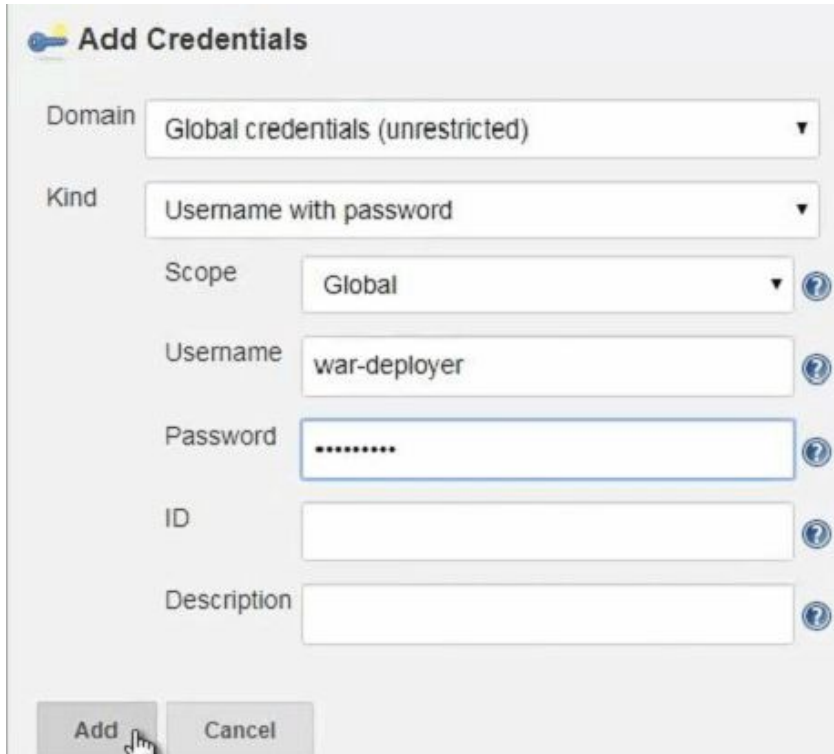
Username: war-deployer

Password: jenkins-tomcat-deploy

Step 3D: Run the Jenkins build job

Now that you have specified all of the configurations, the Jenkins build job can be saved and run.

When the build job finishes, the Jenkins Tomcat deploy of a WAR file will have also completed, and a file named _____ .war will be visible in the webapps directory of Tomcat.

The image shows the 'Add Credentials' dialog box in Jenkins. It has a title bar with a key icon and the text 'Add Credentials'. Below the title, there are several fields: 'Domain' with a dropdown menu showing 'Global credentials (unrestricted)', 'Kind' with a dropdown menu showing 'Username with password', 'Scope' with a dropdown menu showing 'Global', 'Username' with a text input field containing 'war-deployer', 'Password' with a masked text input field showing '*****', 'ID' with an empty text input field, and 'Description' with an empty text input field. Each field has a help icon (a question mark in a circle) to its right. At the bottom left, there are two buttons: 'Add' and 'Cancel'. A mouse cursor is pointing at the 'Add' button.

Add Credentials

Domain: Global credentials (unrestricted)

Kind: Username with password

Scope: Global

Username: war-deployer

Password: *****

ID:

Description:

Add Cancel

Step 4: Test the deployed WAR file

With the WAR file deployed, test the application by running Tomcat and pointing your browser to the following URL:

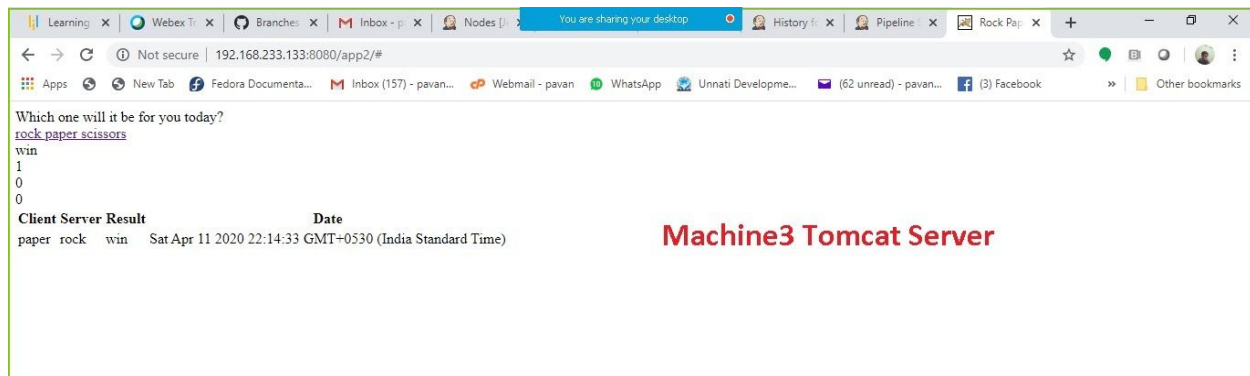
<http://localhost:8080/app2>

Jenkins Tomcat WAR deployment summary

To recap, here is a summary of the steps required to perform a Jenkins Tomcat WAR file deployment:

1. Add a user with WAR deployment rights to the `tomcat-users.xml`.
2. Add the *Deploy to container* Jenkins Tomcat plugin.

3. Create a Jenkins build job with a *Deploy to container* post-build action.
4. Run the Jenkins build job.
5. Test to ensure the Jenkins deployment of the WAR file to Tomcat was successful.



=====

Same Application Delivery with CI/CD Pipeline :demo :

Reference steps in Project

=====

Pipeline Code

```
node('machine2') {
  stages{
    stage('code') {
```

```
git branch: 'patch-1', url: 'https://github.com/pavansw/rock-paper-scissors.git'
}
stage('compile') {

  sh label: "", script: 'mvn clean compile'
}
```

```
stage('testing') {  
  
    sh label: "", script: 'mvn test'  
}  
stage('report') {  
  
    junit '**/*.xml'  
}  
stage('package') {  
    sh label: "", script: 'mvn install'  
}  
    stage('Deploy Tomcat on Machine3') {  
deploy adapters: [tomcat7(credentialsId: '3939ec90-51d0-4cc6-bd33-d499e375dc03', path: "",  
url: 'http://machine3.test.com:8080')], contextPath: 'app2', war: '**/*.war'  
  
    }  
    }  
}
```

=====

The credentials part / step better to create from Snippet Generator