

JENKINS DEPLOYMENT

STEP 1: LAUNCH 1 INSTANCE FOR JENKINS

Name: Jenkins Server

AMI: Amazon Linux Server

Instance Type: t2.micro

Storage: 15-30 GB

Create a security group and make sure to allow SSH port 22 and TCP port 8080 from anywhere from the IP.

Now launch the instance and then connect it.

STEP 2: Now login as root user: `sudo -i`

Copy the below commands:

`#!/bin/bash`

- `sudo yum update -y`
- `sudo wget -O /etc/yum.repos.d/jenkins.repo https://pkg.jenkins.io/redhat-stable/jenkins.repo`
- `sudo rpm --import https://pkg.jenkins.io/redhat-stable/jenkins.io-2023.key`
- `sudo yum upgrade`
- `sudo yum install java-17-amazon-corretto -y`
- `sudo yum install jenkins git -y`
- `sudo systemctl enable jenkins`
- `sudo systemctl start jenkins`
- `sudo systemctl status jenkins`
- `sudo mkdir -p /var/tmp_disk`
- `sudo chmod 1777 /var/tmp_disk`
- `sudo mount --bind /var/tmp_disk /tmp`
- `echo '/var/tmp_disk /tmp none bind 0 0' | sudo tee -a /etc/fstab`
- `sudo systemctl mask tmp.mount`
- `df -h /tmp`
- `sudo systemctl restart Jenkins`

Reference: <https://github.com/Cloud-Boi-Sai/Installation-setups/blob/main/jenkins.sh>

Execute the above commands one by one or use **.sh** file

Create a file: `vim Jenkins.sh`

Paste the commands from the repo and save it the file.

To execute the file: `sh jenkins.sh`

STEP 3: Jenkins gets installed in your instance. After installing the copy the public IP of Jenkins instance and port 8080.

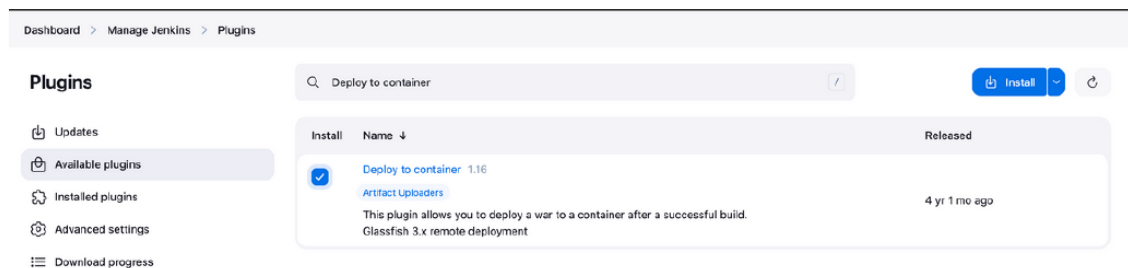
Ex: `http://<public-ip>:8080`

You will see the Jenkins dashboard, once you hit the URL.

Now Jenkins is Ready, Lets integrate it with GIT

Now Install Deploy to container plugin

Go to Manage Jenkins >> Plugins >> Available Plugins >> Deploy to container



The screenshot shows the Jenkins 'Plugins' page. The breadcrumb trail at the top is 'Dashboard > Manage Jenkins > Plugins'. On the left sidebar, under the 'Plugins' heading, there are links for 'Updates', 'Available plugins' (which is highlighted), 'Installed plugins', 'Advanced settings', and 'Download progress'. The main content area has a search bar with 'Deploy to container' entered. To the right of the search bar are buttons for 'Install', a dropdown arrow, and a refresh icon. Below the search bar is a table with columns 'Install', 'Name', and 'Released'. The table contains one entry: 'Deploy to container' with version '1.16'. The 'Install' column for this entry has a blue checkmark icon. Below the name, there is a link for 'Artifact Updaters' and a description: 'This plugin allows you to deploy a war to a container after a successful build. Glassfish 3.x remote deployment'. The 'Released' column shows '4 yr 1 mo ago'.

Install	Name	Released
<input checked="" type="checkbox"/>	Deploy to container 1.16 Artifact Updaters This plugin allows you to deploy a war to a container after a successful build. Glassfish 3.x remote deployment	4 yr 1 mo ago

STEP 4: Integrate Maven

Go to Manage Jenkins >> tools and select maven

Under the Maven Installations click on Add Maven

Name: MyMaven

Version: default (3.9.9)

Dashboard > Manage Jenkins > Tools

Maven installations

Add Maven

Maven

Name

mymaven

☒ Install automatically ?

Install from Apache

Version

3.9.9

Add Installer

Add Maven

Save Apply

Now click on save

STEP 5: Create a Free Style Job

Jenkins

Search (⌘+K)

shaik mustafa log out

Dashboard > New Item

New Item

Enter an item name

MyDeployment

Select an item type

Freestyle project
Classic, general-purpose job type that checks out from up to one SCM, executes build steps serially, followed by post-build steps like archiving artifacts and sending email notifications.

Pipeline
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

Multi-configuration project
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

Folder
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.

OK

Go to **Source Code Management** and select **GIT** and enter the repo-url

(<https://github.com/usubbu/one.git>)

The screenshot shows the Jenkins Configuration page for a job named 'MyDeployment'. The 'Source Code Management' section is active, and 'Git' is selected as the provider. The 'Repository URL' is set to 'https://github.com/usubbu/one.git' and 'Credentials' is set to '- none -'. The 'Advanced' dropdown is visible.

Dashboard > MyDeployment > Configuration

Configure

- General
- Source Code Management
- Build Triggers
- Build Environment
- Build Steps
- Post-build Actions

Source Code Management

☐ None

☒ Git ?

Repositories ?

Repository URL ?
https://github.com/usubbu/one.git

Credentials ?
- none -

+ Add

Advanced ▾

Go to **Build Steps** and select **Invoke top-level Maven targets** and select the Maven version as **mymaven**

Now enter the goal as **clean package**

The screenshot shows the Jenkins Configuration page for a job named 'MyDeployment'. The 'Build Steps' section is active, and 'Invoke top-level Maven targets' is selected. The 'Maven Version' is set to 'mymaven' and the 'Goals' are set to 'clean package'. The 'Advanced' dropdown is visible.

Dashboard > MyDeployment > Configuration

Configure

- General
- Source Code Management
- Build Triggers
- Build Environment
- Build Steps
- Post-build Actions

Build Steps

☐ Delete workspace before build starts

☐ Use secret text(s) or file(s) ?

☐ Add timestamps to the Console Output

☐ Inspect build log for published build scans

☐ Terminate a build if it's stuck

☐ With Ant ?

Invoke top-level Maven targets ?

Maven Version
mymaven

Goals
clean package

Advanced ▾

Add build step ▾

Save Apply

Save the job and click on Build Now

Dashboard > MyDeployment >

Status

Changes

Workspace

Build Now

Configure

Delete Project

Rename

MyDeployment

Permalinks

Builds

Today

✓

#1

18:29

If the build gets success, go to workspace and open the target folder, we will get war file.

Dashboard > MyDeployment > Workspace of MyDeployment on Built-In Node

Status

Changes

Workspace

Build Now

Configure

Delete Project

Rename

Workspace of MyDeployment on Built-In Node

MyDeployment / target /

maven-archiver

myweb-8.6.7

myweb-8.6.7.war 24 Dec 2024, 18:29:58 1.96 KiB

(all files in zip)

Builds

Filter

Today

✓

#1

18:29

STEP 6: Now Launch one more instance for tomcat

Name: Tomcat

AMI: Amazon Linux Kernel-6.1

Instance Type: t2.micro

Storage: 15-30 GB

Now click on launch instance and then connect it.

Now login as root user: `sudo -i`

Copy the below commands:

```
#!/bin/bash
```

- `yum install java-17-amazon-corretto -y`
- `wget https://dlcdn.apache.org/tomcat/tomcat-9/v9.0.108/bin/apache-tomcat-9.0.108.tar.gz`
- `tar -zxvf apache-tomcat-9.0.108.tar.gz`
- `sed -i '56 a\<role rolename="manager-gui"/>' apache-tomcat-9.0.108/conf/tomcat-users.xml`
- `sed -i '57 a\<role rolename="manager-script"/>' apache-tomcat-9.0.108/conf/tomcat-users.xml`
- `sed -i '58 a\<user username="tomcat" password="admin@123" roles="manager-gui, manager-script"/>' apache-tomcat-9.0.108/conf/tomcat-users.xml`
- `sed -i '59 a\</tomcat-users>' apache-tomcat-9.0.108/conf/tomcat-users.xml`
- `sed -i '56d' apache-tomcat-9.0.108/conf/tomcat-users.xml`
- `sed -i '21d' apache-tomcat-9.0.108/webapps/manager/META-INF/context.xml`
- `sed -i '22d' apache-tomcat-9.0.108/webapps/manager/META-INF/context.xml`
- `sh apache-tomcat-9.0.108/bin/startup.sh`

You can find the commands or script in this repo

(<https://github.com/Cloud-Boi-Sai/Installation-setups/blob/main/tomcat.sh>)

Remember the tomcat credentials

- username = **tomcat**
- password = **admin@123**

If you wish to modify, change the credentials on above script also.


Now access the tomcat dashboard with public-ip of tomcat server with 8080 port.

Not secure 3.109.5.202:8080

Home Documentation Configuration Examples Wiki Mailing Lists Find Help

Apache Tomcat/9.0.108

If you're seeing this, you've successfully installed Tomcat. Congratulations!



Recommended Reading:

- [Security Considerations How-To](#)
- [Manager Application How-To](#)
- [Clustering/Session Replication How-To](#)

Server Status
Manager App
Host Manager

Developer Quick Start

- Tomcat Setup
- First Web Application
- Realms & AAA
- JDBC DataSources
- Examples
- Servlet Specifications
- Tomcat Versions

Managing Tomcat

For security, access to the `manager.webapp` is restricted. Users are defined in:

`$CATALINA_HOME/conf/tomcat-users.xml`

Documentation

- [Tomcat 9.0 Documentation](#)
- [Tomcat 9.0 Configuration](#)

Getting Help

[FAQ and Mailing Lists](#)

The following mailing lists are available:

Now go to Manager App and enter the credentials, You will get the Tomcat Web Application Manager dashboard.

Not secure 3.109.5.202:8080/manager/html

Tomcat Web Application Manager

Message: OK

Manager

List Applications HTML Manager Help Manager Help Server Status

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

STEP 7: Integrate Tomcat with Jenkins

Configure the Jenkins job and select Post Build Actions and select Deploy war/ear to a container

WAR/EAR file: target/*.war

Context path: swiggy

Post-build Actions

Define what happens after a build completes, like sending notifications, archiving artifacts, or triggering other jobs.

Deploy war/ear to a container

WAR/EAR files ?
target/*.war

Context path ?
swiggy

Containers

and click on add container and select Tomcat 9.x Remote

Dashboard > MyDeployment > Configuration

Configure

General

Source Code Management

Build Triggers

Build Environment

Build Steps

Post-build Actions

Containers

Tomcat 9.x Remote

Credentials
- none -

+ Add

Tomcat URL ?

Advanced

Add Container

☐ Deploy on failure

Add post-build action

Save

Apply

Now add the tomcat credentials, click on Add select Jenkins

Jenkins Credentials Provider: Jenkins

Kind
Username with password

Scope ?
Global (Jenkins, nodes, items, all child items, etc)

Username ?
tomcat

☐ Treat username as secret

Password ?
.....

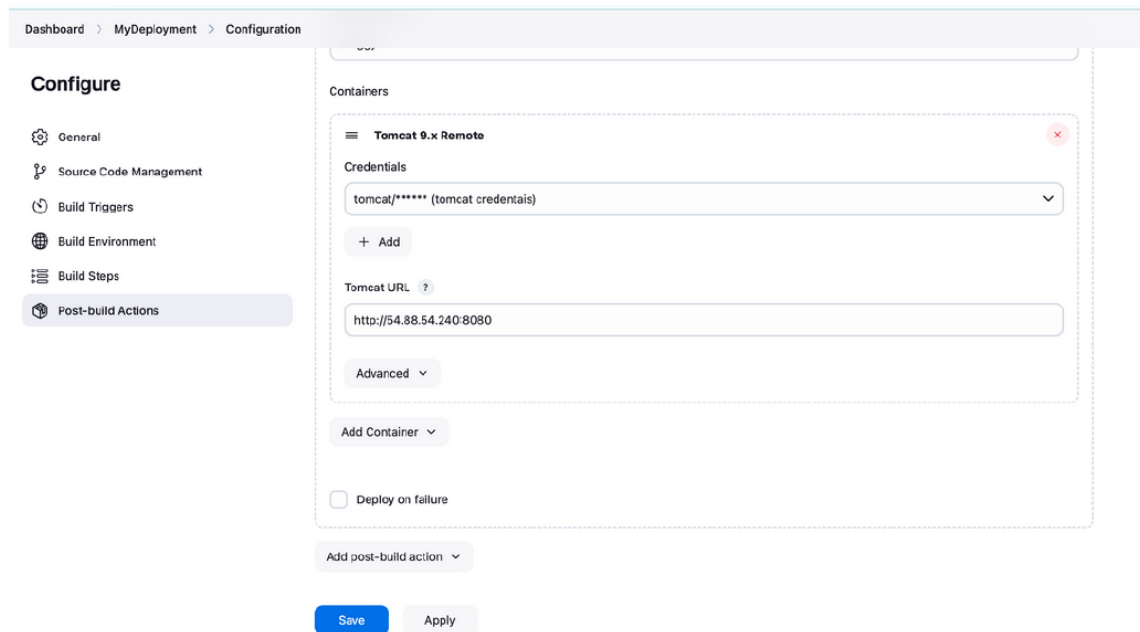
ID ?
appserver

Description ?
tomcat credentials

Save

Apply

Now click on Add and select the credentials and enter the tomcat url





The image shows the Jenkins 'Configure' page for a job named 'MyDeployment'. The left sidebar contains a 'Configure' section with options: General, Source Code Management, Build Triggers, Build Environment, Build Steps, and Post-build Actions. The main area is titled 'Containers' and shows a configuration for 'Tomcat 9.x Remote'. It includes a 'Credentials' dropdown menu with 'tomcat/****** (tomcat credentials)' selected, an 'Add' button, a 'Tomcat URL' field with 'http://54.88.54.240:8080', an 'Advanced' dropdown, an 'Add Container' dropdown, and a 'Deploy on failure' checkbox. At the bottom, there is an 'Add post-build action' dropdown, 'Save' and 'Apply' buttons.

Now save the job and click on Build Now, If the build gets success, then our application will be deployed on tomcat successfully.



The image shows the Jenkins 'Jenkins / MyDeployment' page. The left sidebar contains a 'Status' section with options: Changes, Workspace, Build Now, Configure, Delete Project, and Rename. The main area shows the 'MyDeployment' job status as 'Success' (green checkmark). Below the status, there are 'Permalinks' for the last build, last stable build, last successful build, and last completed build, all of which are '#2', 49 min ago. At the bottom, there is a 'Builds' section with a search filter and a list of builds. The list shows two builds: '#2' at 7:16 AM and '#1' at 6:13 AM, both with green status icons.

This build is success, now let's refresh the tomcat page and we will see swiggy app is running.

Tomcat Web Application Manager

Message: OK

[List Applications](#)
[HTML Manager Help](#)
[Manager Help](#)
[Server Status](#)

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

Open the swiggy app and check the output.