

Automate Project Build with Tomcat 10 using Jenkins

1. To automate project build, we need to have a Jenkins Server EC2 instance.

Add the below commands in user data section of EC2 instance before launching it to install Jenkins on it

aws Services Search

Metadata version [Info](#)

V2 only (token required)

⚠ For V2 requests, you must include a session token in all instance metadata requests. Applications or agents that use V1 for instance metadata access will break.

Metadata response hop limit [Info](#)

2

Allow tags in metadata [Info](#)

Select

User data - optional [Info](#)

Upload a file with your user data or enter it in the field.

[Choose file](#)

```
#!/bin/bash

apt update -y && apt upgrade -y && apt install openjdk-17-jdk -y

curl -fsSL https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key | sudo tee \
/usr/share/keyrings/jenkins-keyring.asc > /dev/null
echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] \
https://pkg.jenkins.io/debian-stable binary/ | sudo tee \
/etc/apt/sources.list.d/jenkins.list > /dev/null

sudo apt update -y
sudo apt install jenkins -y
systemctl start jenkins
```

☐ User data has already been base64 encoded

2. Once the EC2 instance is up, Post deployment steps are performed for configuring Jenkins to setup the Jenkins server. Once Jenkins is configured, we see the below dashboard screen.

Jenkins

Dashboard >

+ New Item

Build History

Manage Jenkins

My Views

Build Queue

No builds in the queue.

Build Executor Status

1 Idle

2 Idle

Welcome to Jenkins!

This page is where your Jenkins jobs will be displayed. To get started, you can set up distributed builds or start building a software project.

Start building your software project

Create a job

Set up a distributed build

Set up an agent

Configure a cloud

Learn more about distributed builds

3. Once Jenkins dashboard is visible, we need to install the below mentioned 3 plugins for automating our project build by going to Manage Jenkins > Plugins > Available Plugins

build pipeline

maven Integration

deploy to container

Dashboard > Manage Jenkins

+ New Item

Build History

Manage Jenkins

My Views

Build Queue

No builds in the queue.

Build Executor Status

Manage Jenkins

Building on the built-in node can be a security issue. You should set up distributed builds. See [the documentation](#).

System Configuration

System

Configure global settings and paths.

Tools

Configure tools, their locations and automatic installers.

Plugins

Add, remove, disable or enable plugins that can extend the functionality of Jenkins.

Dashboard > Manage Jenkins > Plugins

Plugins

Updates

Available plugins

Installed plugins

Advanced settings

Download progress

Search available plugins

Install	Name	Released
<input checked="" type="checkbox"/>	Deploy to container 1.16 Artifact Uploaders This plugin allows you to deploy a war to a container after a successful build. Glassfish 3.x remote deployment	3 yr 9 mo ago
<input checked="" type="checkbox"/>	Build Pipeline 2.0.2 User Interface Build Tools Other Post-Build Actions This plugin renders upstream and downstream connected jobs that typically form a build pipeline. In addition, it offers the ability to define manual triggers for jobs that require intervention prior to execution, e.g. an approval process outside of Jenkins. <div>Warning: This plugin version may not be safe to use. Please review the following security notices:<ul style="list-style-type: none">Stored XSS vulnerability</div>	3 mo 6 days ago
<input checked="" type="checkbox"/>	Maven Integration 3.23 Build Tools This plugin provides a deep integration between Jenkins and Maven. It adds support for automatic triggers between projects depending on SNAPSHOTS as well as the automated configuration of various Jenkins publishers such as Junit.	1 yr 0 mo ago

Dashboard > Manage Jenkins > Plugins

Plugins

Updates

Available plugins

Installed plugins

Advanced settings

Download progress

Pipeline: GitHub Groovy Libraries	Success
Pipeline Graph View	Success
Git	Success
SSH Build Agents	Success
Matrix Authorization Strategy	Success
PAM Authentication	Success
LDAP	Success
Email Extension	Success
Mailer	Success
Dark Theme	Success
Loading plugin extensions	Success
JavaMail API	Success
SSH server	Success
Deploy to container	Success
Parameterized Trigger	Success
Oracle Java SE Development Kit Installer	Success
Command Agent Launcher	Success
jQuery	Success
Build Pipeline	Success
Javadoc	Success
JSch dependency	Success
Maven Integration	Success
Loading plugin extensions	Success

4. Once the Plugins are installed, we will go to Manage Jenkins > Tools

Dashboard > Manage Jenkins

+ New Item

Build History

Manage Jenkins

My Views

Build Queue

No builds in the queue.

Build Executor Status

1 Idle

2 Idle

Manage Jenkins

Building on the built-in node can be a security issue. You should set up distributed builds. See [the documentation](#).

Warnings have been published for the following currently installed components:

Build Pipeline Plugin 2.0.2:
Stored XSS vulnerability (no fix available)
No fixes for these issues are available. It is recommended that you review the security advisory and apply mitigations.

System Configuration

System
Configure global settings and paths.

Tools
Configure tools, their locations and automatic installers.

5. Under Maven Configuration, Below Settings should be there.

Dashboard > Manage Jenkins > Tools

Tools

Maven Configuration

Default settings provider

Use default maven settings

Default global settings provider

Use default maven global settings

and also, we need to define Maven Installations, we can give it a name for e.g. MAV and select install automatically and under Install from Apache > Version, keep the default visible version. In our case 3.9.9 and click Save.

Dashboard > Tools

Ant installations

Add Ant

Maven installations

Add Maven

Maven

Name

Mav

☒ Install automatically ?

Install from Apache

Version

3.9.9

Add Installer ▾

Add Maven

Save Apply

- Now we need to create the first job, which we will give the name as “DEV”, it will be a maven project since we need to run the maven test command to check the source code and pom.xml

New Item

Enter an item name

Dev

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.
- Maven project**
Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.
- 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.
- Multibranch Pipeline**
Creates a set of Pipeline projects according to detected branches in one SCM repository.

OK

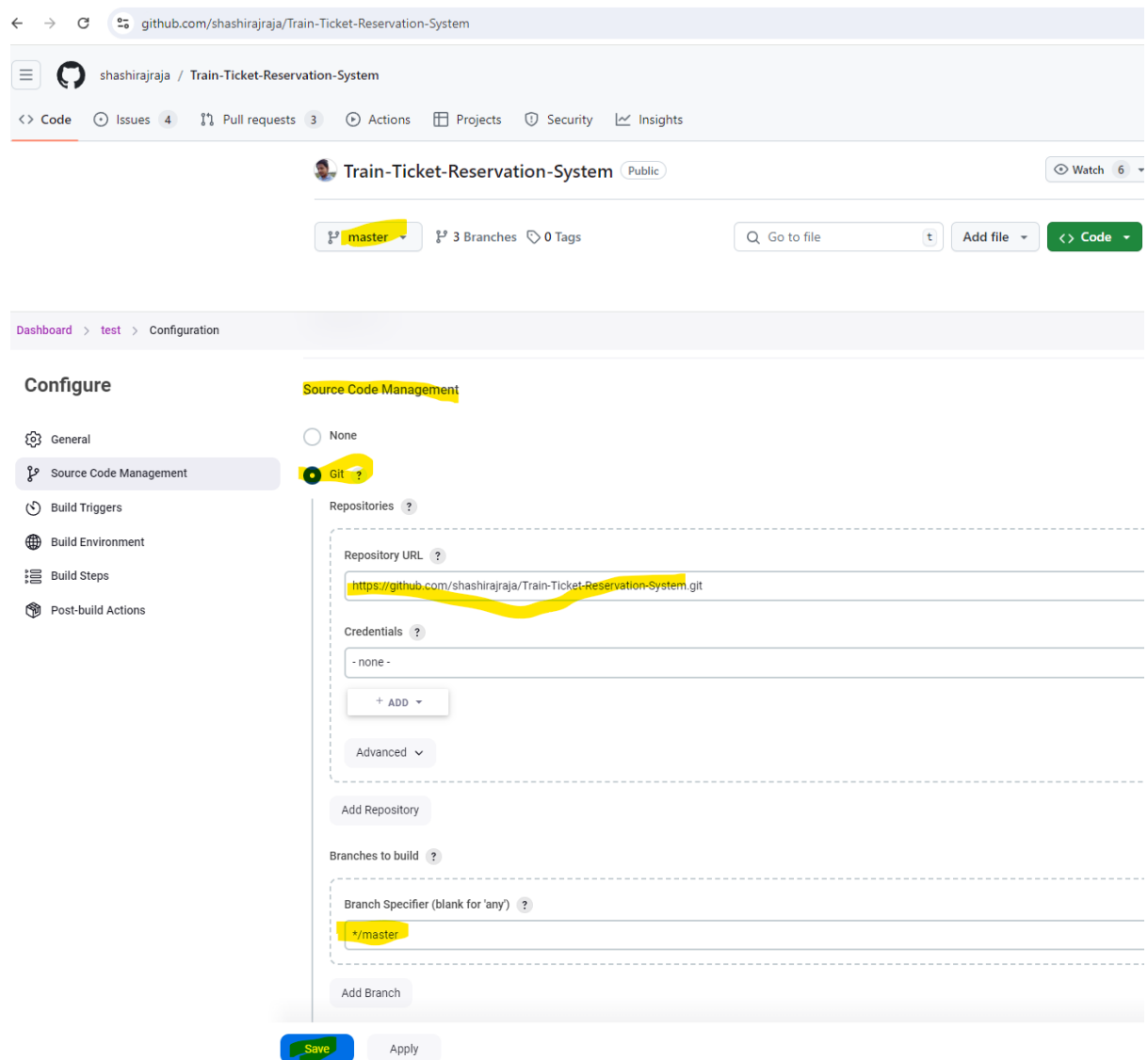
- Now we will configure the “Dev” job as below.

We need to select git under Source Code Management

We need to give the Repository URL under repositories

<https://github.com/shashirajraja/Train-Ticket-Reservation-System.git>

Lastly, we need to specify the branch name same as it is on the git hub link, in our case its */master in the git link also so we need to specify the same in the job also. Then Click Save



The image shows two screenshots. The top screenshot is a GitHub repository page for 'shashirajraja / Train-Ticket-Reservation-System'. The 'master' branch is selected. The bottom screenshot is the Jenkins 'Configure' page for a job named 'test'. Under 'Source Code Management', 'Git' is selected. The 'Repository URL' is 'https://github.com/shashirajraja/Train-Ticket-Reservation-System.git'. The 'Branches to build' section has '*/master' specified. The 'Save' button is highlighted.

8. Under Build, we need to give Goals and options as test since we are testing the build and then click Save.

Dashboard > Dev > Configuration

Configure

General

Source Code Management

Build Triggers

Build Environment

Pre Steps

Build

Post Steps

Build Settings

Post-build Actions

☐ With Ant ?

Pre Steps

Add pre-build step ▾

Build

Root POM ?

pom.xml

Goals and options ?

test

Advanced ▾

Post Steps

☐ Run only if build succeeds

☐ Run only if build succeeds or is unstable

☒ Run regardless of build result


Should the post-build steps run only for successful builds, etc.


Save


Apply


9. Now we create the second job as “Build” which we will copy the project from “Dev”. Just we need to change the Goal and Options under Build to install and click save, since we are building the code.


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.


 **Maven project**
Build a maven project, Jenkins takes advantage of your POM files and drastically reduces the configuration.

 **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.

 **Multibranch Pipeline**
Creates a set of Pipeline projects according to detected branches in one SCM repository.

 **Organization Folder**
Creates a set of multibranch project subfolders by scanning for repositories.

If you want to create a new item from other existing, you can use this option:

Copy from

Copy

Dashboard > Build > Configuration

Configure

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

Build

Root POM ?
pom.xml

Goals and options ?
install

Advanced ▾

Post Steps

☐ Run only if build succeeds

☐ Run only if build succeeds or is unstable

☒ Run regardless of build result

Should the post-build steps run only for successful builds, etc.

Add post-build step ▾

Build Settings

☐ E-mail Notification

Save Apply

10. Now we will create the third job as “Deploy” and select it freestyle project and for now, we do not need to configure anything in that job.

New Item

Enter an item name
Deploy

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.
- Maven project**
Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.
- 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.
- Multibranch Pipeline**
Creates a set of Pipeline projects according to detected branches in one SCM repository.

OK

11. Now we define the dependency on each job i.e. Dev to trigger Build Project and Build to Trigger Deploy Project.

Dashboard > Dev > Configuration

Configure

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

Should the post-build steps run only for successful builds, etc.

Add post-build step ▾

Build Settings

☐ E-mail Notification

Post-build Actions

Build other projects ?

Projects to build

Build,

! No such project 'Bu'. Did you mean 'Build'?

☒ Trigger only if build is stable

☐ Trigger even if the build is unstable

☐ Trigger even if the build fails

Add post-build action ▾

Save Apply

Dashboard > Build > Configuration

Configure

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

☒ Run regardless of build result

Should the post-build steps run only for successful builds, etc.

Add post-build step ▾

Build Settings

☐ E-mail Notification

Post-build Actions

Build other projects ?

Projects to build

Deploy

! No such project 'Dep'. Did you mean 'Dev'?

☒ Trigger only if build is stable

☐ Trigger even if the build is unstable

☐ Trigger even if the build fails

Add post-build action ▾

Save Apply

12. Now we configure the “Build” Project to archive the artifact by following the below steps.

Configure the Build Job and go to Post Build Actions and add post-build action and select Archive the artifacts and in that under Files to archive, Type `**/*.war` (this defines any type of war file) and click save.

Dashboard > Build > Configuration

Configure

☐ E-mail Notification

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

Post-build Actions

Archive the artifacts ?

Files to archive ?

`**/*.war`

Advanced ▾

Build other projects ?

Projects to build

Deploy

☒ Trigger only if build is stable

☐ Trigger even if the build is unstable

☐ Trigger even if the build fails

Add post-build action ▾

Save Apply

13. Now we need to install a plugin named copy artifact (So that archives files like war can be copied to artifacts)

Dashboard > Manage Jenkins > Plugins

Plugins

Available plugins

Search: copy

Install	Name	Released
<input checked="" type="checkbox"/>	Copy Artifact <small>hudson.plugins.artifact-copy</small> Build Parameters Build Tools Adds a build step to copy artifacts from another project. <small>This plugin is up for adoption! We are looking for new maintainers. Visit our Adopt a Plugin initiative for more information.</small>	1 mo 4 days ago

Dashboard > Manage Jenkins > Plugins		
Plugins Updates Available plugins Installed plugins Advanced settings Download progress	Git	Success
	SSH Build Agents	Success
	Matrix Authorization Strategy	Success
	PAM Authentication	Success
	LDAP	Success
	Email Extension	Success
	Mailer	Success
	Dark Theme	Success
	Loading plugin extensions	Success
	JavaMail API	Success
	SSH server	Success
	Deploy to container	Success
	Parameterized Trigger	Success
	Oracle Java SE Development Kit Installer	Success
	Command Agent Launcher	Success
	jQuery	Success
	Build Pipeline	Success
	Javadoc	Success
	JSch dependency	Success
	Maven Integration	Success
	Loading plugin extensions	Success
	Copy Artifact	Success
	Loading plugin extensions	Success

14. Now we will configure the “Build” project again and under General select the option of Permission to Copy Artifact and under Projects to allow copy artifacts we need to give “Deploy” project and then click Save.

Dashboard > Build > Configuration

Configure

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

General

Description

Plain text [Preview](#)

☐ Discard old builds ?

☐ GitHub project

☒ **Permission to Copy Artifact**

Projects to allow copy artifacts ?

Deploy

☐ This project is parameterized ?

☐ Throttle builds ?

☐ Execute concurrent builds if necessary ?

Advanced ▾

Source Code Management

None

Save Apply

15. Now we will configure “Deploy” project and go to option Build steps > Add Build step and select Copy artifact from another project and under Project Name, we will give “Build” since we are copying artifacts from build project and click Save.

The screenshot shows the Jenkins Configuration page for a project named 'Deploy'. The left sidebar contains a 'Configure' section with a list of tabs: General, Source Code Management, Build Triggers, Build Environment (selected), Build Steps, and Post-build Actions. The main content area is titled 'Configure' and contains several checkboxes: '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', and 'With Ant'. Below these is a 'Build Steps' section with a dashed border. Inside, there is a 'Copy artifacts from another project' step. The 'Project name' field is set to 'Build'. A warning message states: 'Artifacts will be copied from all modules of'. The 'Which build' dropdown is set to 'Latest successful build'. There is an unchecked checkbox for 'Stable build only'. Below are two empty text input fields for 'Artifacts to copy' and 'Artifacts not to copy'. At the bottom of the 'Build Steps' section are 'Save' and 'Apply' buttons.

16. Now we will install Tomcat10 on Jenkins server through terminal.

apt install tomcat10 -y

```
root@ip-172-31-38-17: /home/ubuntu
root@ip-172-31-38-17: /home/ubuntu# apt install tomcat10 -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  libapr1t64 libeclipse-jdt-core-java libtcnative-1 libtomcat10-java tomcat10-common
Suggested packages:
  tomcat10-admin tomcat10-docs tomcat10-examples tomcat10-user
The following NEW packages will be installed:
  libapr1t64 libeclipse-jdt-core-java libtcnative-1 libtomcat10-java tomcat10 tomcat10-common
3 upgraded, 6 newly installed, 0 to remove and 14 not upgraded.
Need to get 13.0 MB of archives.
After this operation, 16.5 MB of additional disk space will be used.
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 libapr1t64 amd64 1.7.2-3.1build2 [107 kB]
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 libeclipse-jdt-core-java all 3.32.0+eclipse4.26-2 [6438 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 libtomcat10-java all 10.1.16-1 [6222 kB]
Get:4 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 tomcat10-common all 10.1.16-1 [61.5 kB]
Get:5 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 tomcat10 all 10.1.16-1 [37.0 kB]
Get:6 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 libtcnative-1 amd64 1.2.35-1build2 [93.9 kB]
```

Next, we will install tomcat10 admin to configure roles and users.

apt install tomcat10-admin -y

```
root@ip-172-31-38-17:/home/ubuntu# apt install tomcat10-admin -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following NEW packages will be installed:
  tomcat10-admin
0 upgraded, 1 newly installed, 0 to remove and 14 not upgraded.
Need to get 67.9 kB of archives.
After this operation, 326 kB of additional disk space will be used.
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 tomcat10-admin all 10.1.16-1 [67.9 kB]
Fetched 67.9 kB in 0s (2166 kB/s)
Selecting previously unselected package tomcat10-admin.
(Reading database ... 114278 files and directories currently installed.)
Preparing to unpack .../tomcat10-admin_10.1.16-1_all.deb ...
Unpacking tomcat10-admin (10.1.16-1) ...
Setting up tomcat10-admin (10.1.16-1) ...
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.
```

Since tomcat and Jenkins run on same port 8080 by default, so we need to change the port for tomcat10 from 8080 to 8081.

vi /etc/tomcat10/server.xml

Search for the line `<Connector port="8080"` and change it to 8081

```
root@ip-172-31-38-17:/home/ubuntu

<!--The connectors can use a shared executor, you can define one or more named thread pools-->
<!--
<Executor name="tomcatThreadPool" namePrefix="catalina-exec-"
      maxThreads="150" minSpareThreads="4"/>
-->

<!-- A "Connector" represents an endpoint by which requests are received
and responses are returned. Documentation at :
HTTP Connector: /docs/config/http.html
AJP  Connector: /docs/config/ajp.html
Define a non-SSL/TLS HTTP/1.1 Connector on port 8080
-->
<Connector port="8081" protocol="HTTP/1.1"
           connectionTimeout="20000"
           redirectPort="8443"
           maxParameterCount="1000"
           />

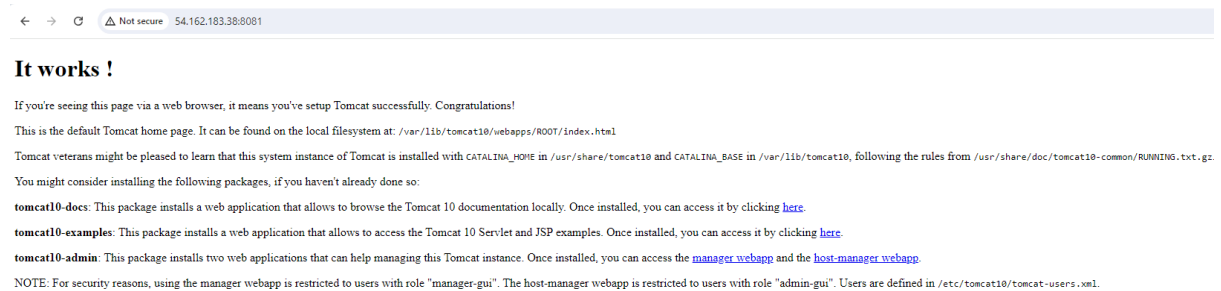
<!-- A "Connector" using the shared thread pool-->
<!--
<Connector executor="tomcatThreadPool"
           port="8080" protocol="HTTP/1.1"
           connectionTimeout="20000"
           redirectPort="8443"
           maxParameterCount="1000"
           />
/8080
```

Now we will restart tomcat10 to make the changes

systemctl restart tomcat10

```
root@ip-172-31-38-17:/home/ubuntu# systemctl restart tomcat10
root@ip-172-31-38-17:/home/ubuntu#
```

Verify tomcat10 is running on the new port 8081. So, it works



Now we will provide the ownership to tomcat user from root.

Before ownership change

```
drwxr-xr-x  5 root    root      4096 Aug 22 14:05 tomcat10
root@ip-172-31-38-17:/var/lib#
```

We will run the ownership change command

chown -R tomcat:tomcat /var/lib/tomcat10

After Ownership change

```
drwxr-xr-x  5 tomcat  tomcat    4096 Aug 22 14:05 tomcat10
root@ip-172-31-38-17:/var/lib#
```

Now we will assign the tomcat10 roles by going to the location /etc/tomcat10

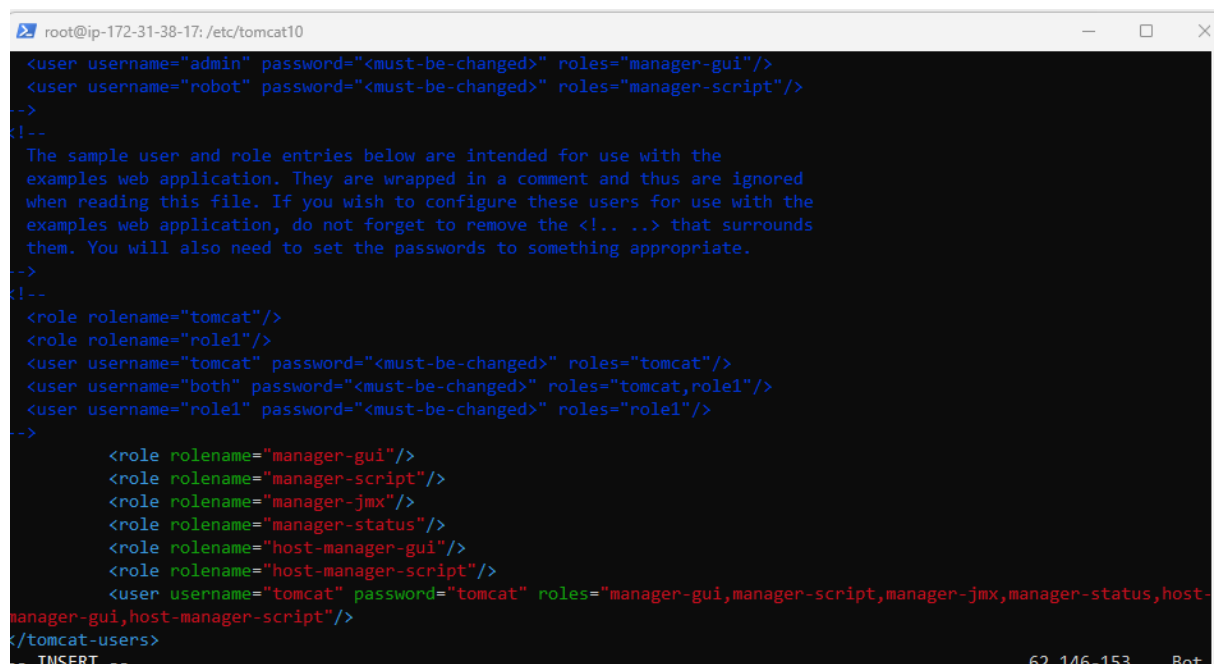
vi tomcat-users.xml and append the below lines in the file.

```
root@ip-172-31-38-17:/etc/tomcat10# vi tomcat-users.xml
```

```
<role rolename="manager-gui"/>
<role rolename="manager-script"/>
<role rolename="manager-jmx"/>
<role rolename="manager-status"/>
```

```
<role rolename="host-manager-gui"/>
<role rolename="host-manager-script"/>
<user username="tomcat" password="tomcat" roles="manager-
gui, manager-script, manager-jmx, manager-status, host-manager-gui,
host-manager-script"/>
```

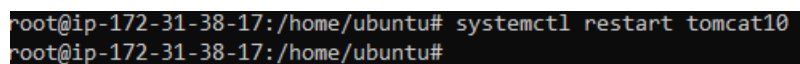
(In the above lines, we also make sure that username should be tomcat only since tomcat user has been given ownership of tomcat10 folder)

A terminal window titled 'root@ip-172-31-38-17: /etc/tomcat10' showing the configuration of tomcat10 users and roles. The terminal output is as follows:

```
<user username="admin" password="<must-be-changed>" roles="manager-gui"/>
<user username="robot" password="<must-be-changed>" roles="manager-script"/>
--
<!--
The sample user and role entries below are intended for use with the
examples web application. They are wrapped in a comment and thus are ignored
when reading this file. If you wish to configure these users for use with the
examples web application, do not forget to remove the <!-- .. --> that surrounds
them. You will also need to set the passwords to something appropriate.
--
--
<role rolename="tomcat"/>
<role rolename="role1"/>
<user username="tomcat" password="<must-be-changed>" roles="tomcat"/>
<user username="both" password="<must-be-changed>" roles="tomcat,role1"/>
<user username="role1" password="<must-be-changed>" roles="role1"/>
--
<role rolename="manager-gui"/>
<role rolename="manager-script"/>
<role rolename="manager-jmx"/>
<role rolename="manager-status"/>
<role rolename="host-manager-gui"/>
<role rolename="host-manager-script"/>
<user username="tomcat" password="tomcat" roles="manager-gui,manager-script,manager-jmx,manager-status,host-
manager-gui,host-manager-script"/>
</tomcat-users>
-- INSERT --
```

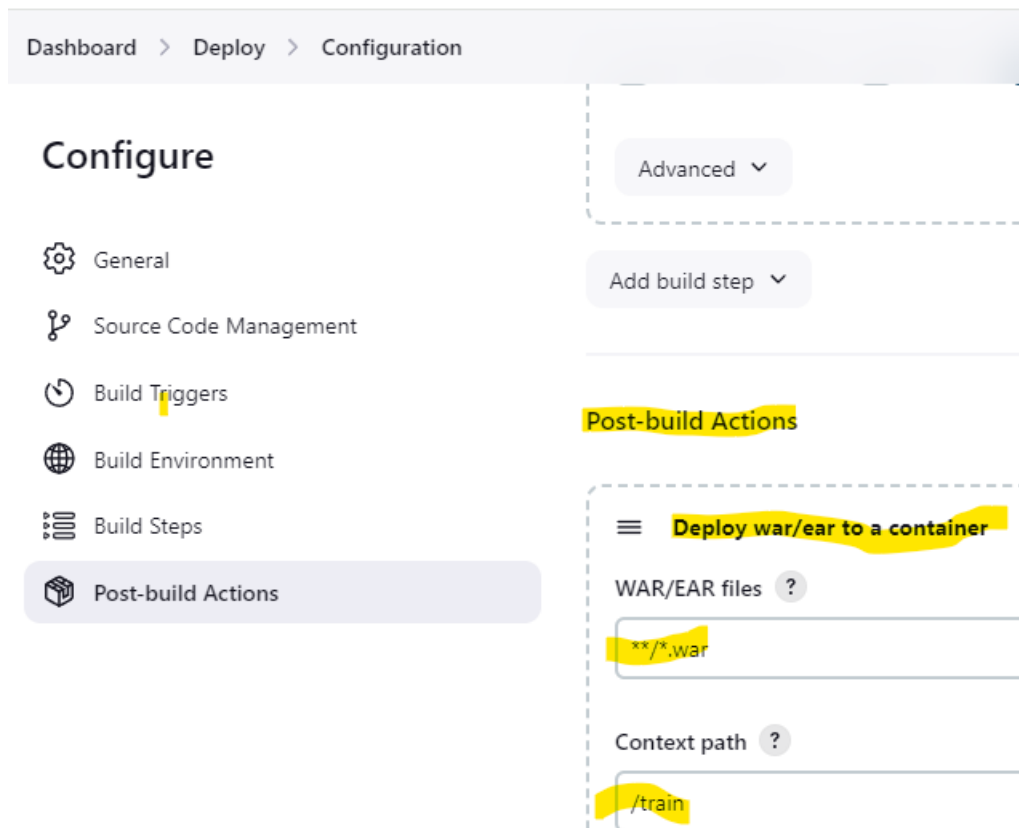
Again, we will restart tomcat10 service.

systemctl restart tomcat10

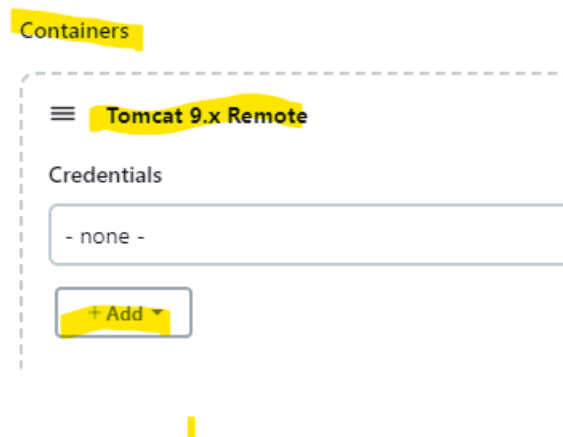
A terminal window showing the command to restart tomcat10 service. The terminal output is as follows:

```
root@ip-172-31-38-17:/home/ubuntu# systemctl restart tomcat10
root@ip-172-31-38-17:/home/ubuntu#
```

17.Next, we will configure again “Deploy” Project then go to Post Build Actions> Add post-build actions then select Deploy war/ear to a container and in WAR/EAR files we will give **/*.war and in the context path we will give /train in our example.



After this, under Containers > Add Container then select Tomcat 9.x Remote. Under this we will click Add the credentials.



Under Credentials we will give username – tomcat and password – tomcat (same credentials which we gave while adding roles in tomcat-users.xml file) and click Add.

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 ?

Description ?

Cancel Add

Once the credentials are added, then we will select those credentials under Tomcat 9.x Remote and also add the Tomcat URL with port 8081 and click Save.

Dashboard > Deploy > Configuration

Configure

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

Context path ?

/train

Containers

Tomcat 9.x Remote

Credentials

tomcat/*****

+ Add

Tomcat URL ?

http://54.162.183.38:8081/

Advanced

Add Container

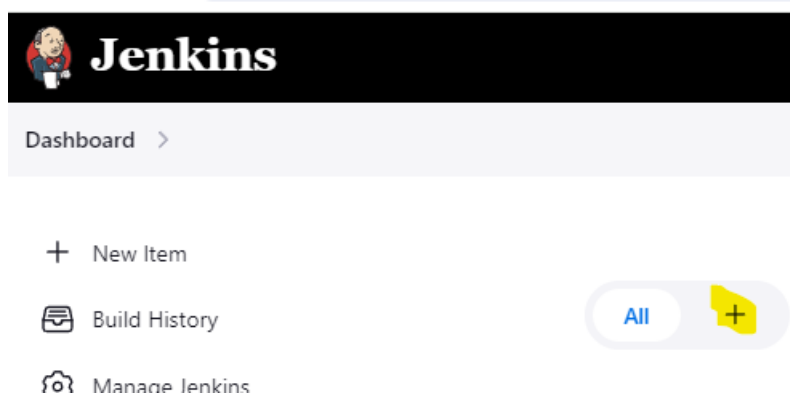
☐ Deploy on failure

Add post-build action

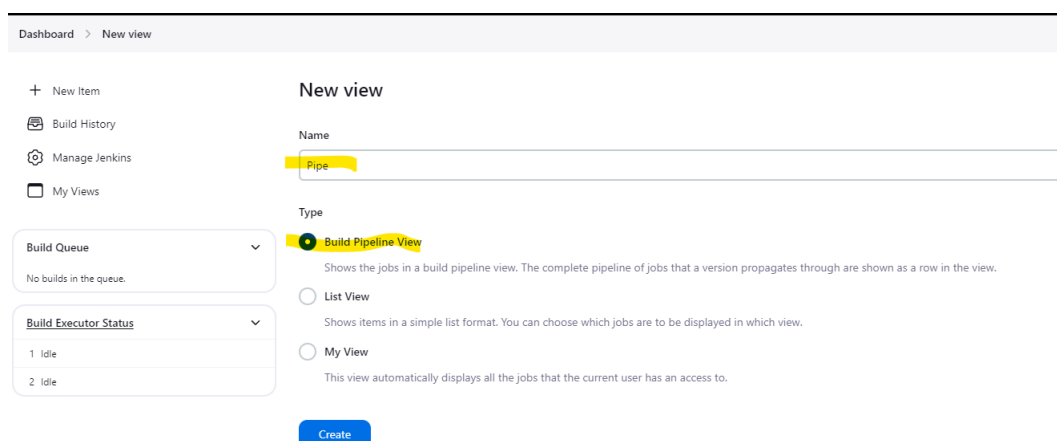
Save Apply

18. Now we will build the pipeline to check whether it works according to the flow.

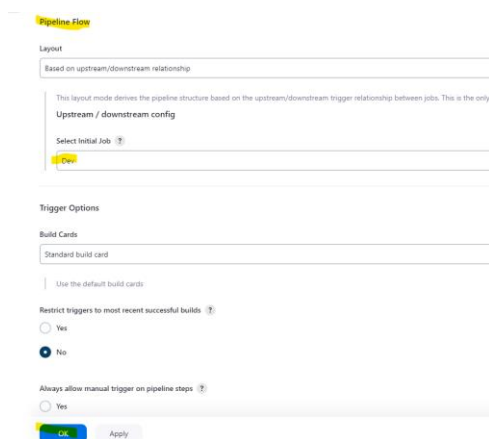
Now click on + sign on dashboard



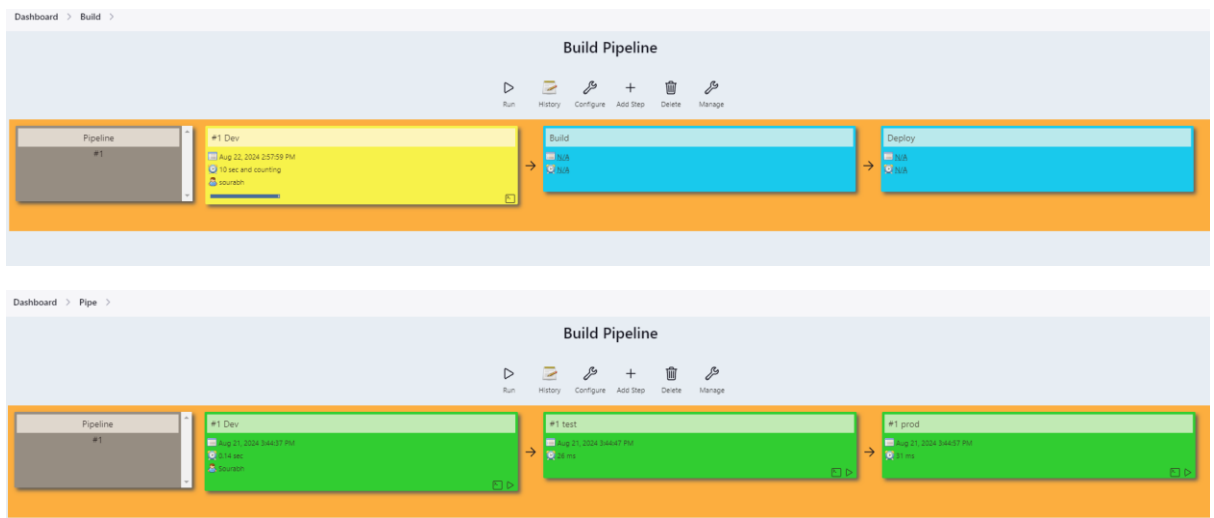
Give Pipeline view a name and select Build Pipeline View



Select Initial Job as Dev under Pipeline Flow and Click OK.

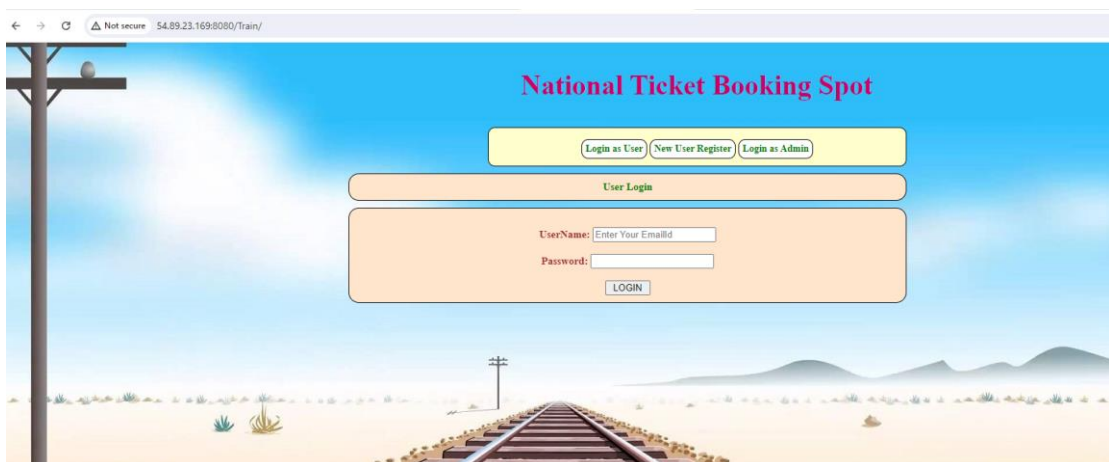


Now will run the Build Pipeline and see the output below.

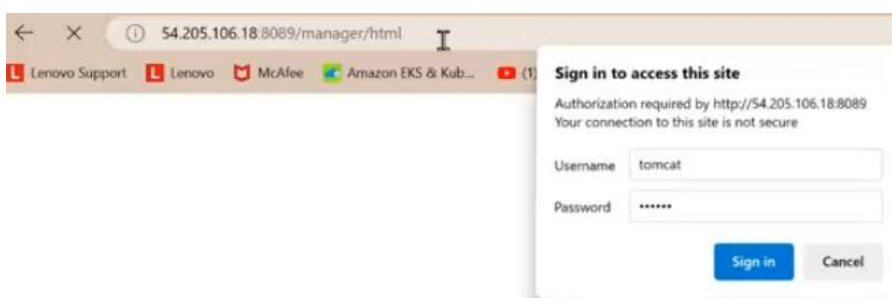


19. Our job ran successfully in the pipeline.

20. The output will come out as below, Note Screenshot details may differ but it works.



21. We will access the manager also in tomcat10 and see that our project is visible there. In the below screenshot, we need to enter the tomcat role credentials i.e. username – tomcat and password – tomcat



22. Below screenshot is different from the actual one. But the result will be same. /App is there in screenshot instead of /train.

Tomcat Web Application Manager

Message: OK

Manager

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

Applications

Path	Version	Display Name	Running	Sessions	Commands
/	None specified		true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/app	None specified		true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes

23. Automate Project build is completed using Jenkins.