

What is Jenkins?

Jenkins is an open-source automation tool written in Java with plugins built for continuous integration.

Why do we use Jenkins?

We use Jenkins to automate the CI/CD process.

Without Jenkins:

- Manual build
- Manual testing
- Manual deployment
- Slow and error-prone

With Jenkins:

- Automatic build
- Automatic testing
- Automatic deployment
- Fast and reliable

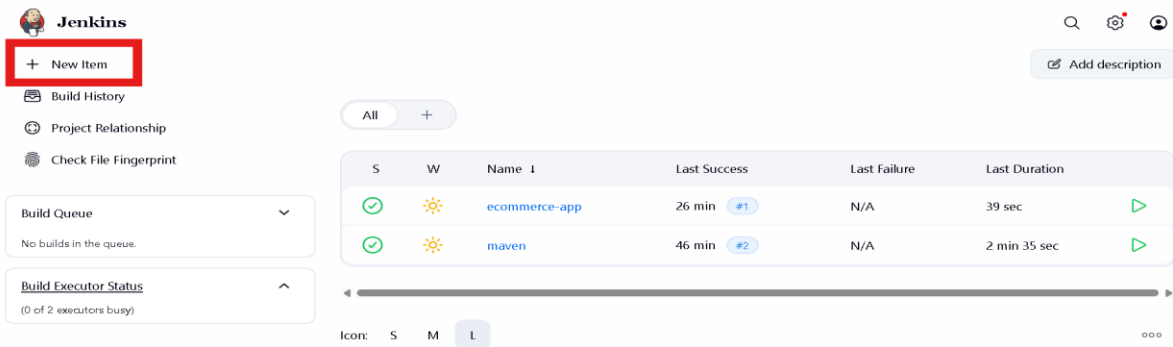
Steps to Open Jenkins and Create a New Pipeline Job

Step 1: Open Jenkins

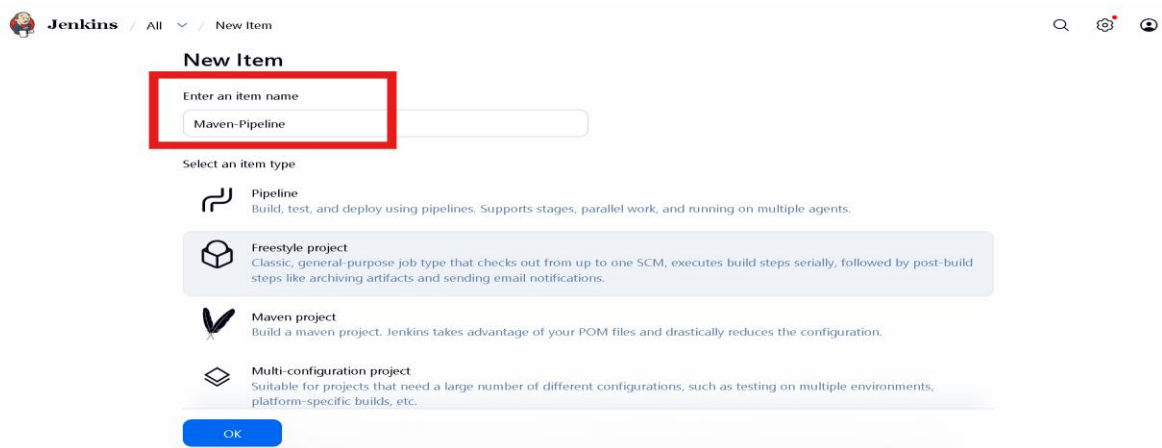
Now:

1. Start Jenkins service.
2. Open a browser.
3. Type the URL: `http://localhost:8080`
4. Jenkins dashboard will open.
5. Login with username and password.

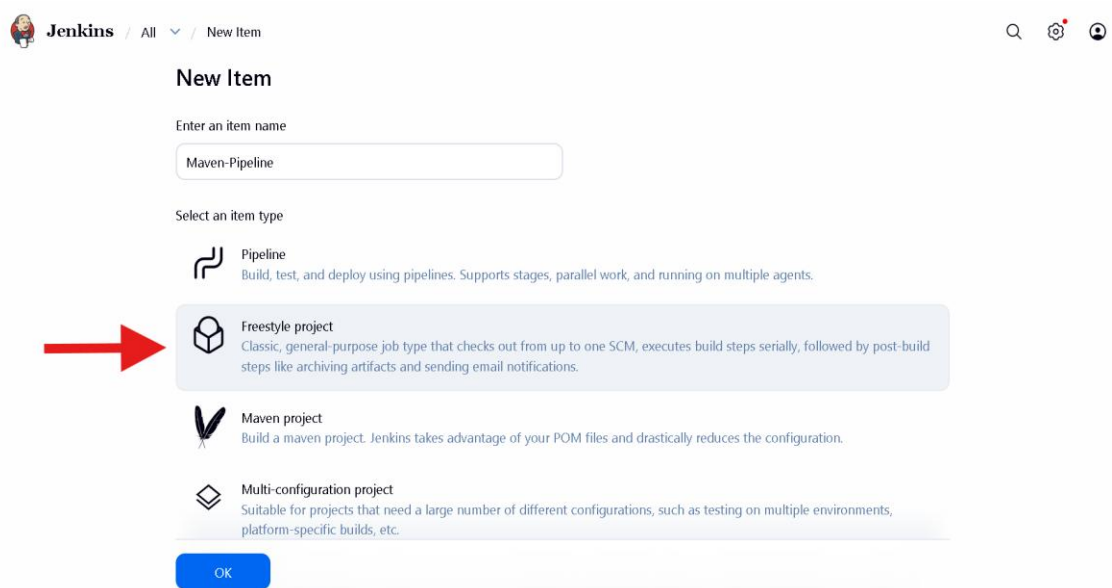
Step 2: Create New Pipeline Job



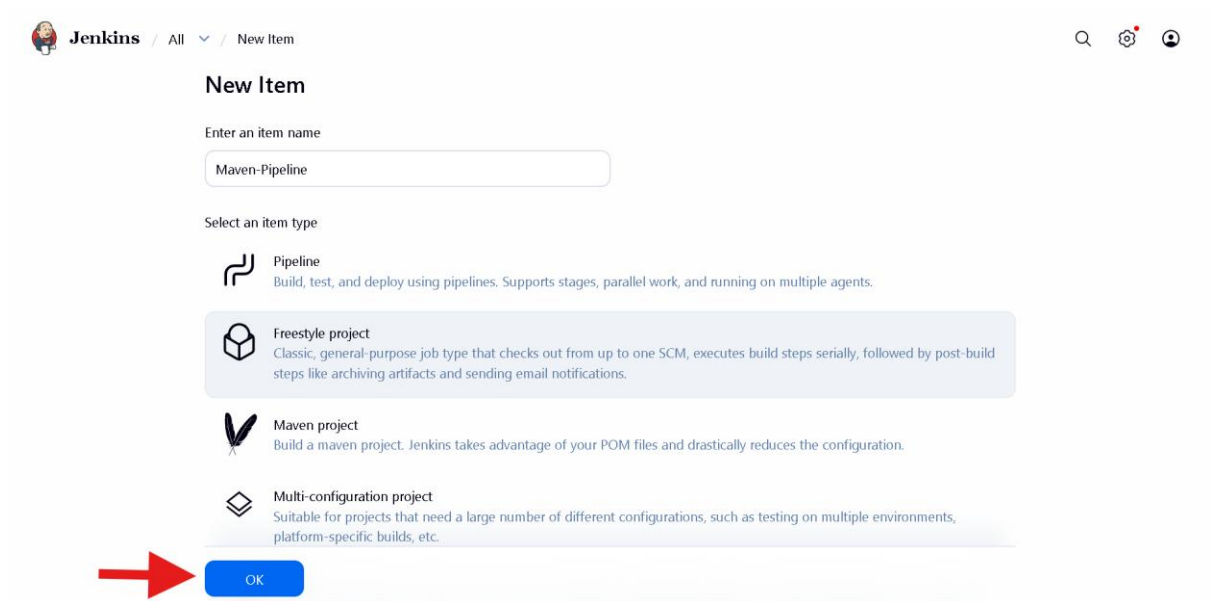
1. On Jenkins dashboard, click on New Item.



2. Enter a job name: Maven-Pipeline.



3. Select an item type as Freestyle Project.



The screenshot shows the Jenkins 'New Item' page. At the top, the breadcrumb is 'Jenkins / All / New Item'. The title is 'New Item'. Below it, there is a text input field for 'Enter an item name' with the value 'Maven-Pipeline'. Underneath, it says 'Select an item type'. There are four options: 'Pipeline' (with a right-angle icon), 'Freestyle project' (with a cube icon), 'Maven project' (with a leaf icon), and 'Multi-configuration project' (with a diamond icon). Each option has a brief description. At the bottom, there is a blue 'OK' button, which is highlighted by a red arrow pointing to it from the left.

Jenkins / All / New Item

New Item

Enter an item name

Maven-Pipeline

Select an item type

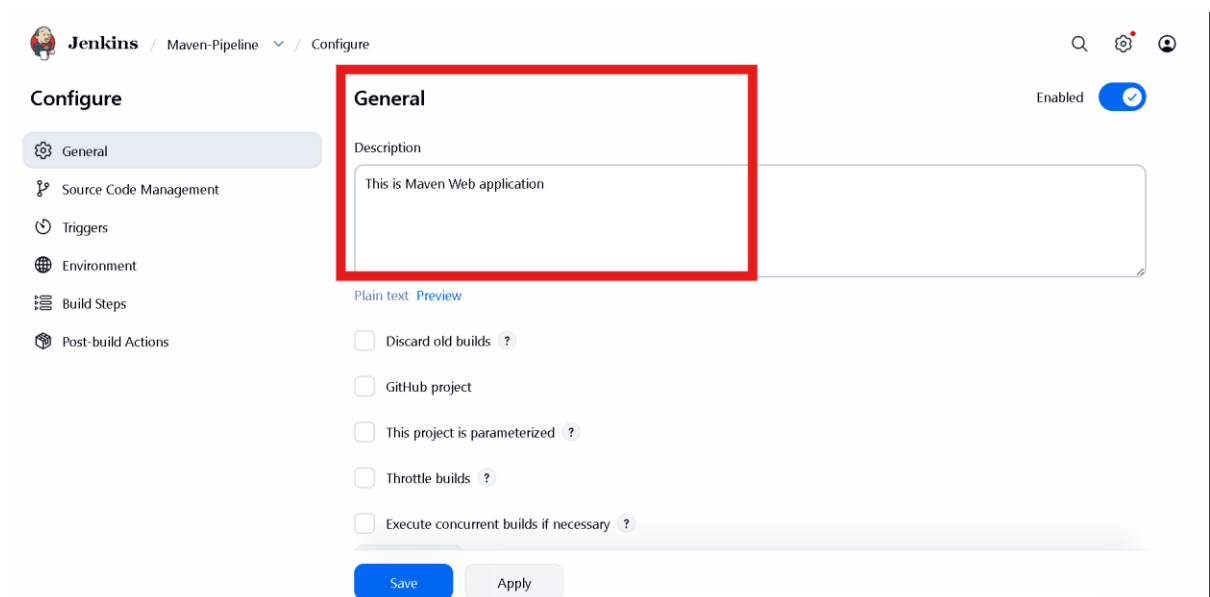
- Pipeline**
Build, test, and deploy using pipelines. Supports stages, parallel work, and running on multiple agents.
- Freestyle project**
Classic, general-purpose job type that checks out from up to one SCM, executes build steps serially, followed by post-build steps like archiving artifacts and sending email notifications.
- Maven project**
Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.
- Multi-configuration project**
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

OK

4. Click OK.

Step 3: Configure Job

General & Source Code Management



The screenshot shows the Jenkins 'Configure' page for the 'Maven-Pipeline' job. The breadcrumb is 'Jenkins / Maven-Pipeline / Configure'. On the left, there is a sidebar with 'Configure' and several tabs: 'General' (selected), 'Source Code Management', 'Triggers', 'Environment', 'Build Steps', and 'Post-build Actions'. The main area is titled 'General' and has a red box around it. It contains a 'Description' text area with the text 'This is Maven Web application'. Below the description, there are several checkboxes: 'Discard old builds', 'GitHub project', 'This project is parameterized', 'Throttle builds', and 'Execute concurrent builds if necessary'. At the bottom, there are 'Save' and 'Apply' buttons. On the right side of the 'General' section, there is an 'Enabled' toggle switch which is turned on.

Jenkins / Maven-Pipeline / Configure

Configure

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

General

Enabled

Description

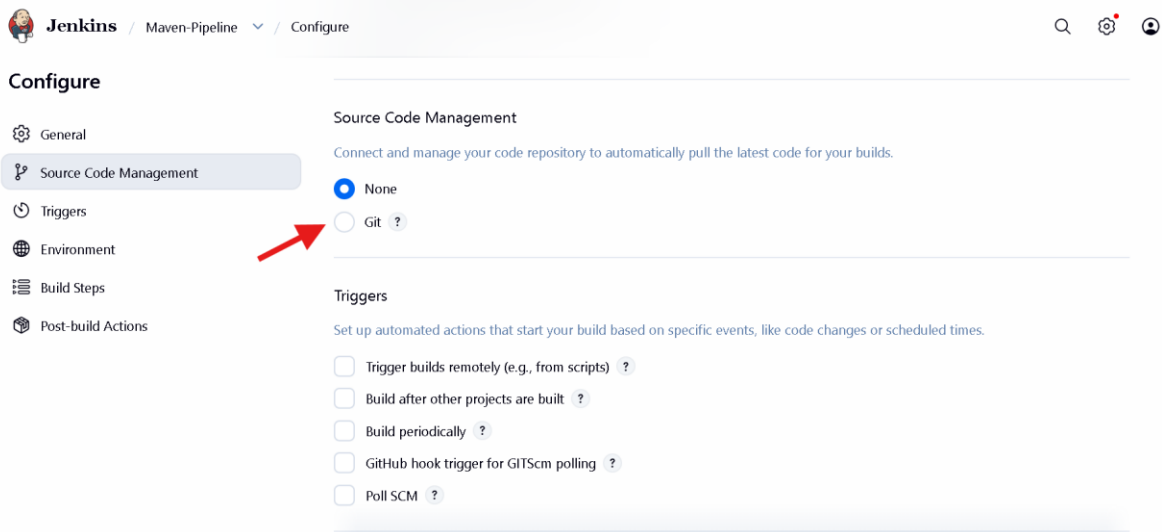
This is Maven Web application

Plain text [Preview](#)

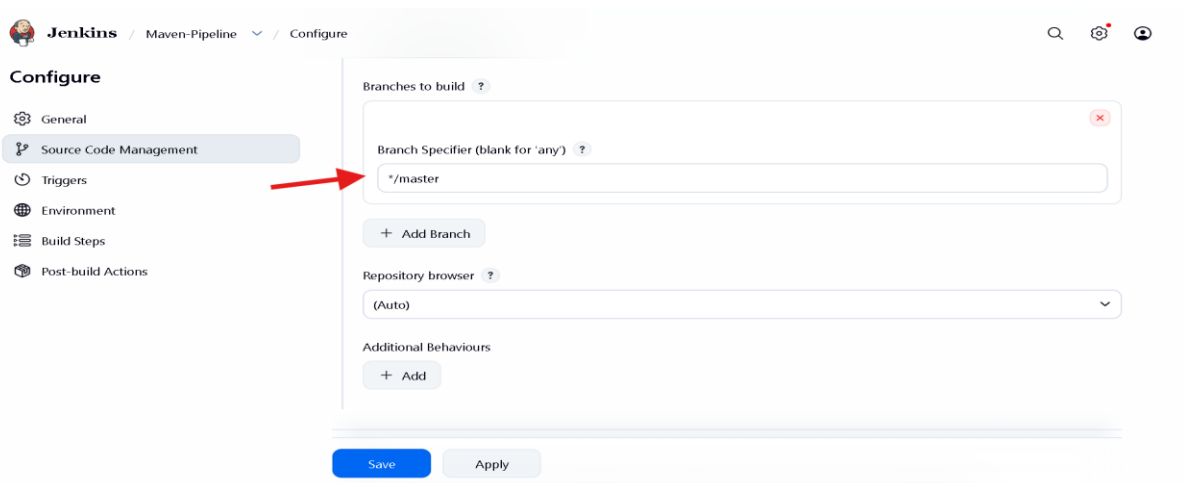
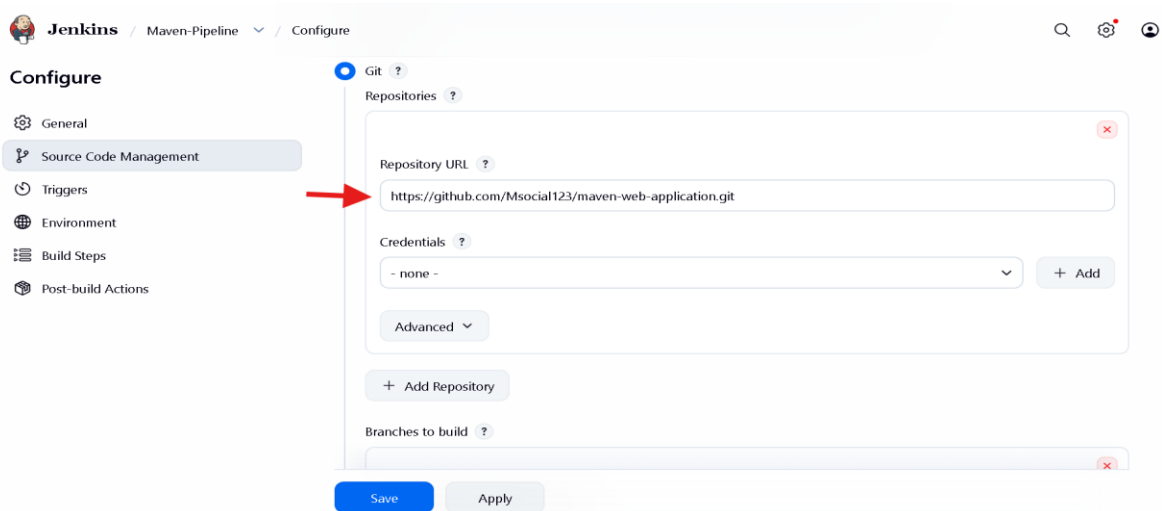
- ☐ Discard old builds ?
- ☐ GitHub project
- ☐ This project is parameterized ?
- ☐ Throttle builds ?
- ☐ Execute concurrent builds if necessary ?

Save Apply

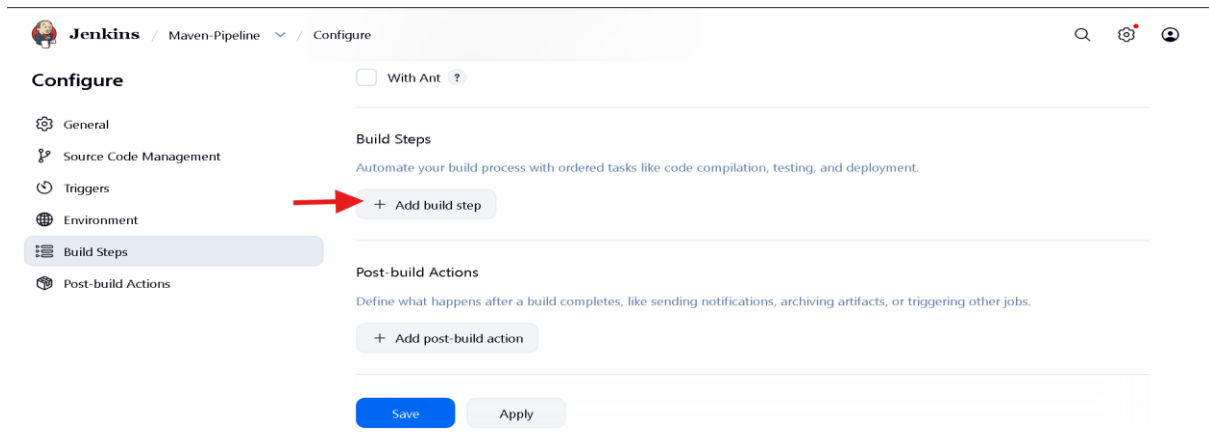
1. Enter job description.



2. Select Git in Source Code Management.

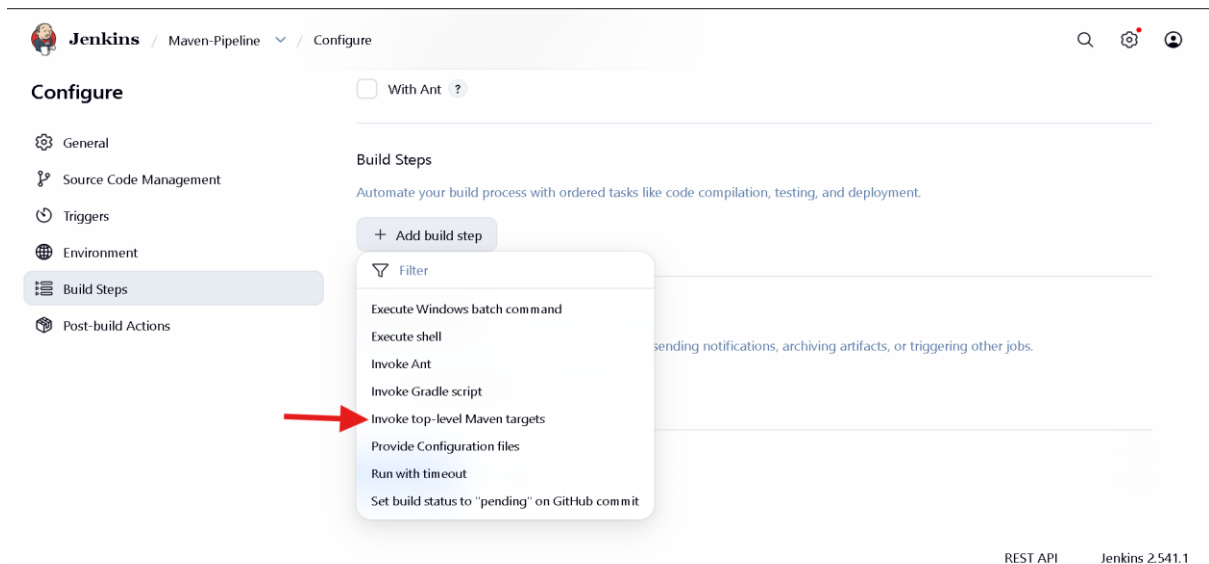


3. Paste GitHub **Repository URL**. Verify branch name (main/master). If incorrect then change it.

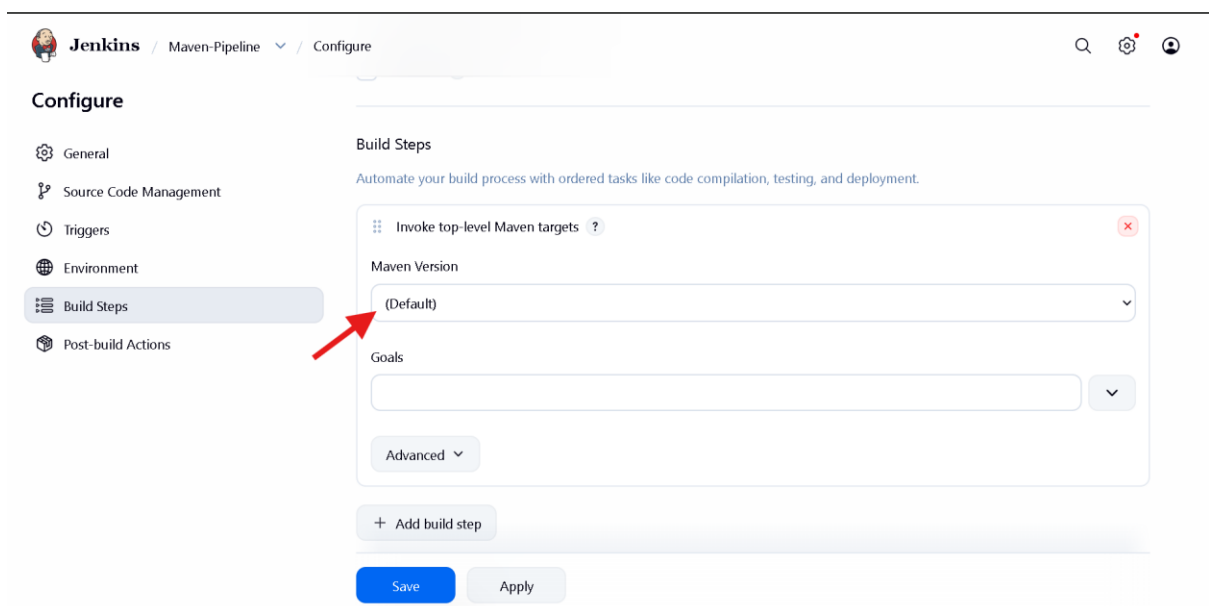


Build Configuration

4. Move to Build steps click on add build step.



5. In the options Select Invoke top-level Maven Targets.



This screenshot shows the Jenkins configuration page for a pipeline named 'Maven-Pipeline'. The 'Build Steps' tab is selected in the left sidebar. A build step titled 'Invoke top-level Maven targets' is configured. The 'Maven Version' dropdown is set to '(Default)', and the 'Goals' field is empty. A red arrow points to the '(Default)' option in the 'Maven Version' dropdown. At the bottom, there are 'Save' and 'Apply' buttons.

Jenkins / Maven-Pipeline / Configure

Configure

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

Build Steps

Automate your build process with ordered tasks like code compilation, testing, and deployment.

Invoke top-level Maven targets ?

Maven Version

(Default)

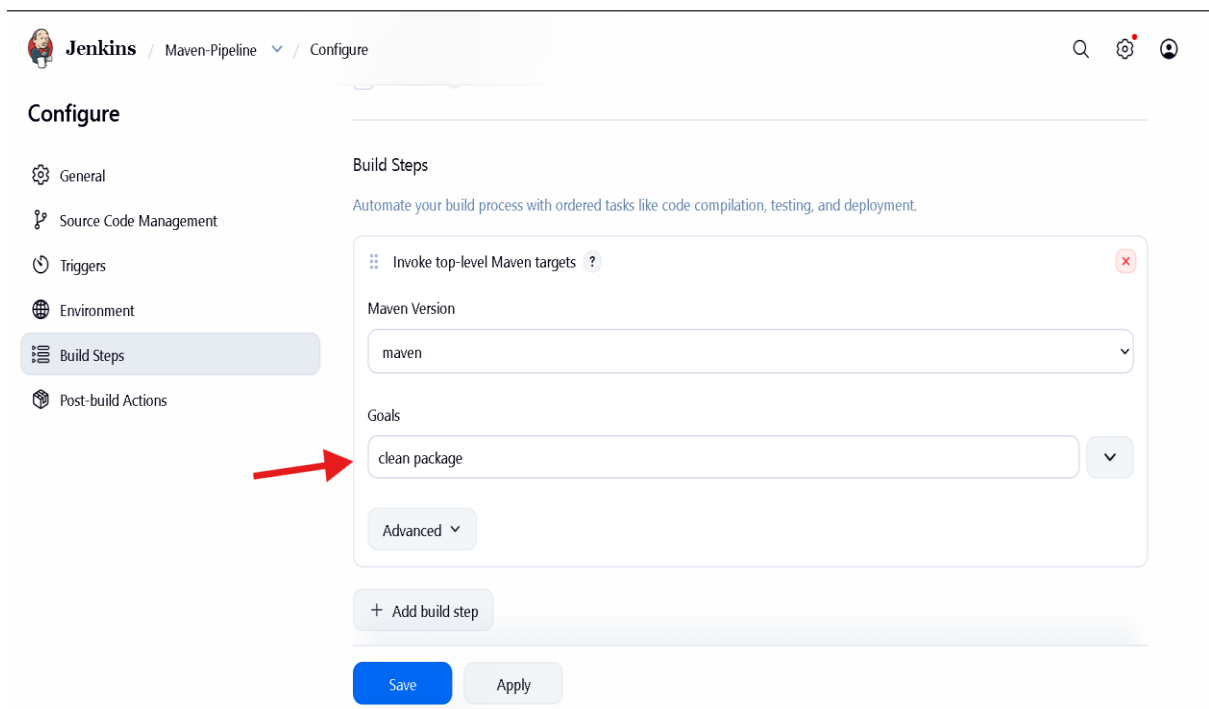
Goals

Advanced

+ Add build step

Save Apply

6. In this Maven version is Default so select it and change to Maven.



This screenshot shows the same Jenkins configuration page, but the 'Maven Version' dropdown is now set to 'maven'. The 'Goals' field is set to 'clean package'. A red arrow points to the 'clean package' option in the 'Goals' dropdown. The 'Save' and 'Apply' buttons are at the bottom.

Jenkins / Maven-Pipeline / Configure

Configure

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

Build Steps

Automate your build process with ordered tasks like code compilation, testing, and deployment.

Invoke top-level Maven targets ?

Maven Version

maven

Goals

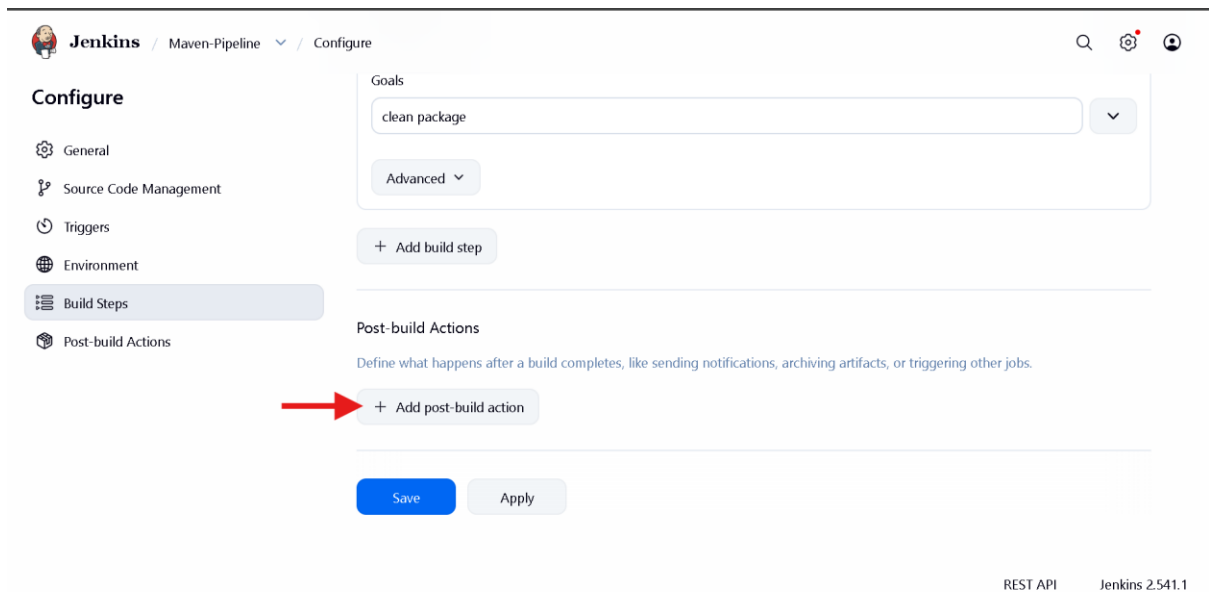
clean package

Advanced

+ Add build step

Save Apply

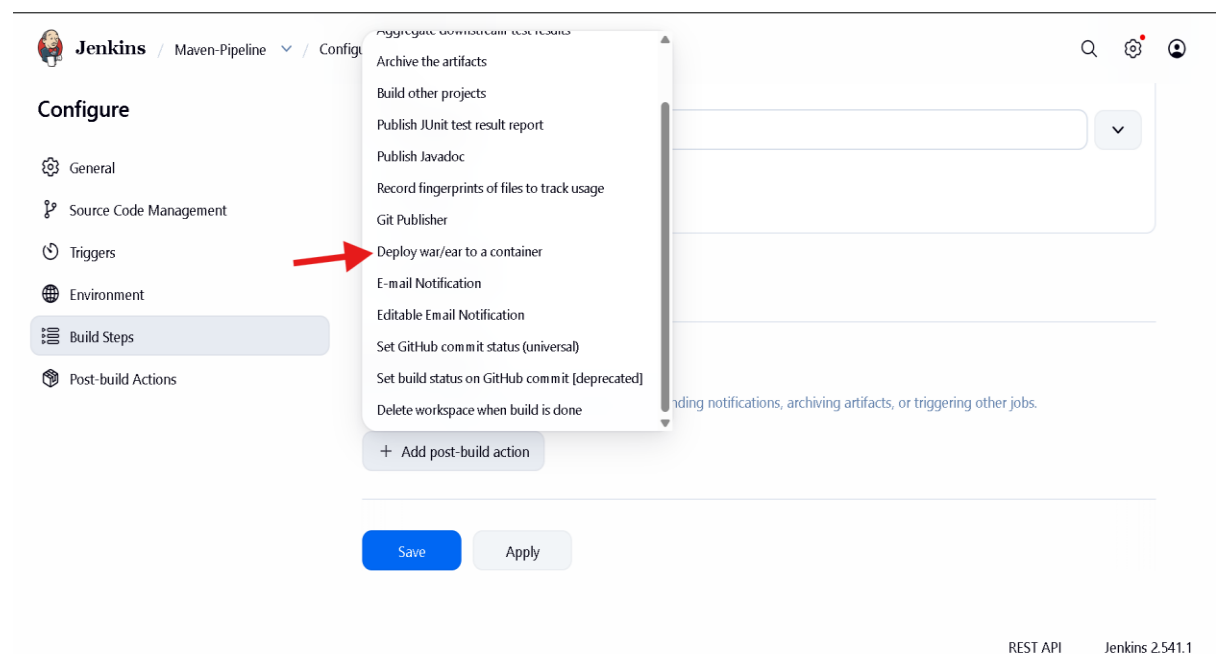
7. Add clean package in Goals and move to next step.



The screenshot shows the Jenkins configuration page for a pipeline named 'Maven-Pipeline'. The left sidebar contains a 'Configure' section with a list of tabs: General, Source Code Management, Triggers, Environment, Build Steps, and Post-build Actions. The 'Build Steps' tab is currently selected. The main area shows the 'Goals' section with a text input field containing 'clean package' and a dropdown arrow. Below this is an 'Advanced' dropdown and a '+ Add build step' button. The 'Post-build Actions' section is visible below, with a description: 'Define what happens after a build completes, like sending notifications, archiving artifacts, or triggering other jobs.' A red arrow points to the '+ Add post-build action' button. At the bottom, there are 'Save' and 'Apply' buttons. The footer shows 'REST API' and 'Jenkins 2.541.1'.

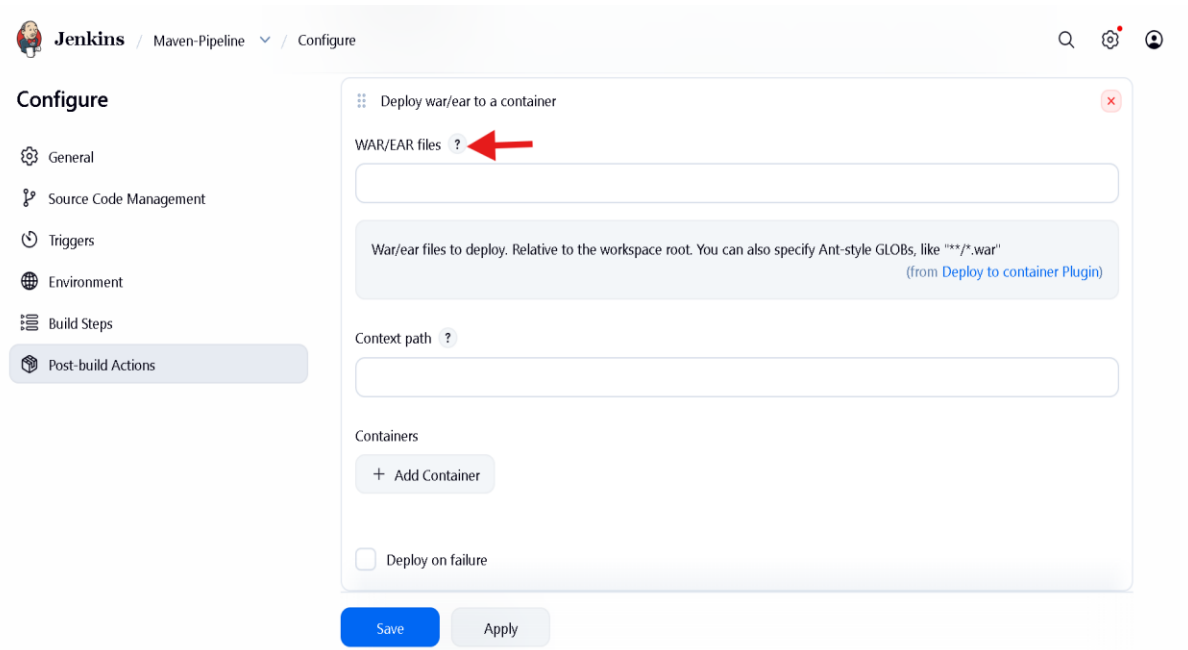
Post-Build Actions:

8. In Post – build Actions Click on Add post-build action.



This screenshot shows the same Jenkins configuration page, but with the 'Post-build Actions' tab selected in the left sidebar. The main area now displays a list of available post-build actions in a dropdown menu. A red arrow points to the 'Deploy war/ear to a container' option. The list includes: Archive the artifacts, Build other projects, Publish JUnit test result report, Publish Javadoc, Record fingerprints of files to track usage, Git Publisher, Deploy war/ear to a container, E-mail Notification, Editable Email Notification, Set GitHub commit status (universal), Set build status on GitHub commit [deprecated], and Delete workspace when build is done. Below the list is a '+ Add post-build action' button. The 'Save' and 'Apply' buttons remain at the bottom. The footer shows 'REST API' and 'Jenkins 2.541.1'.

9. In the options Select Deploy war/ear to a container.



The screenshot shows the Jenkins configuration page for the 'Deploy war/ear to a container' step. The left sidebar contains a 'Configure' section with a list of options: General, Source Code Management, Triggers, Environment, Build Steps, and Post-build Actions. The main area is titled 'Deploy war/ear to a container' and includes a red 'X' icon in the top right corner. It features a 'WAR/EAR files' field with a question mark icon, a text box for 'War/ear files to deploy. Relative to the workspace root. You can also specify Ant-style GLOBs, like "**/*.war" (from Deploy to container Plugin)', a 'Context path' field with a question mark icon, a 'Containers' section with an 'Add Container' button, and a 'Deploy on failure' checkbox. At the bottom are 'Save' and 'Apply' buttons. A red arrow points to the question mark icon next to the 'WAR/EAR files' field.

Jenkins / Maven-Pipeline / Configure

Configure

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

Deploy war/ear to a container

WAR/EAR files ?

War/ear files to deploy. Relative to the workspace root. You can also specify Ant-style GLOBs, like "**/*.war" (from Deploy to container Plugin)

Context path ?

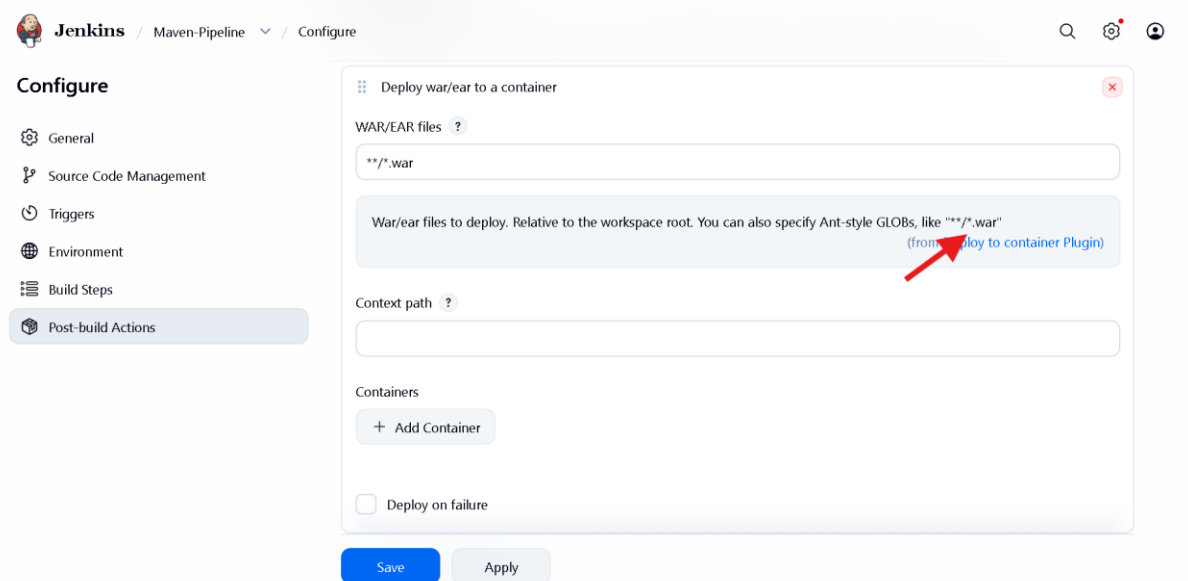
Containers

+ Add Container

☐ Deploy on failure

Save Apply

10. Now Click on the symbol ? after WAR/EAR files.



This screenshot is identical to the previous one, but the 'WAR/EAR files' field now contains the text '**/*.war'. A red arrow points to the question mark icon next to the 'WAR/EAR files' field, indicating where to click for help.

Jenkins / Maven-Pipeline / Configure

Configure

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

Deploy war/ear to a container

WAR/EAR files ?

**/*.war

War/ear files to deploy. Relative to the workspace root. You can also specify Ant-style GLOBs, like "**/*.war" (from Deploy to container Plugin)

Context path ?

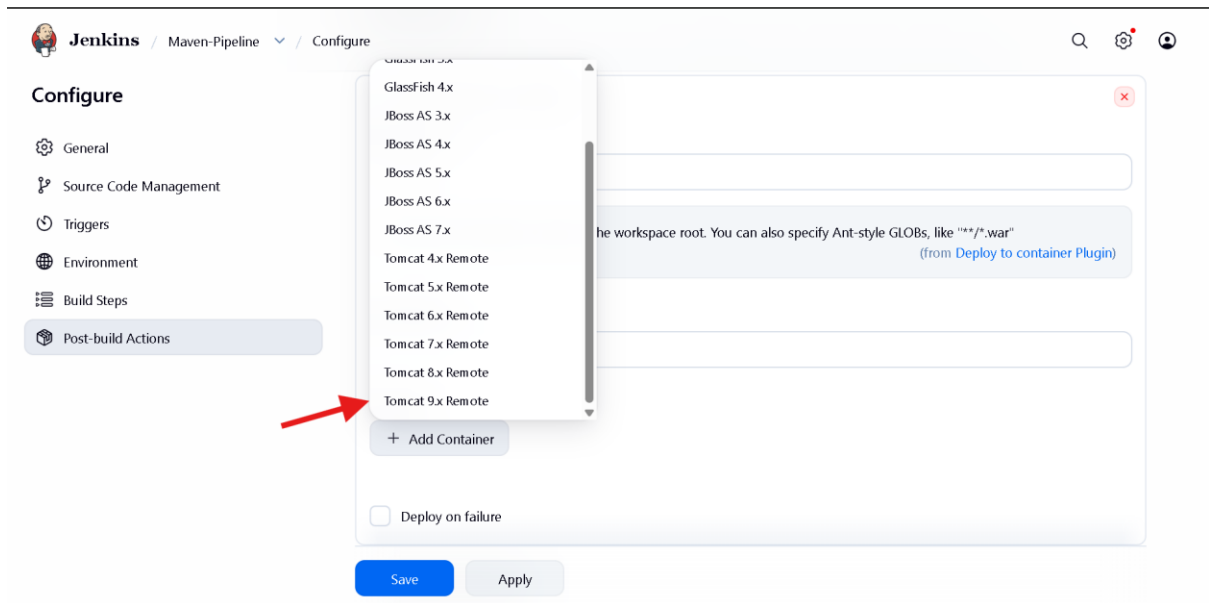
Containers

+ Add Container

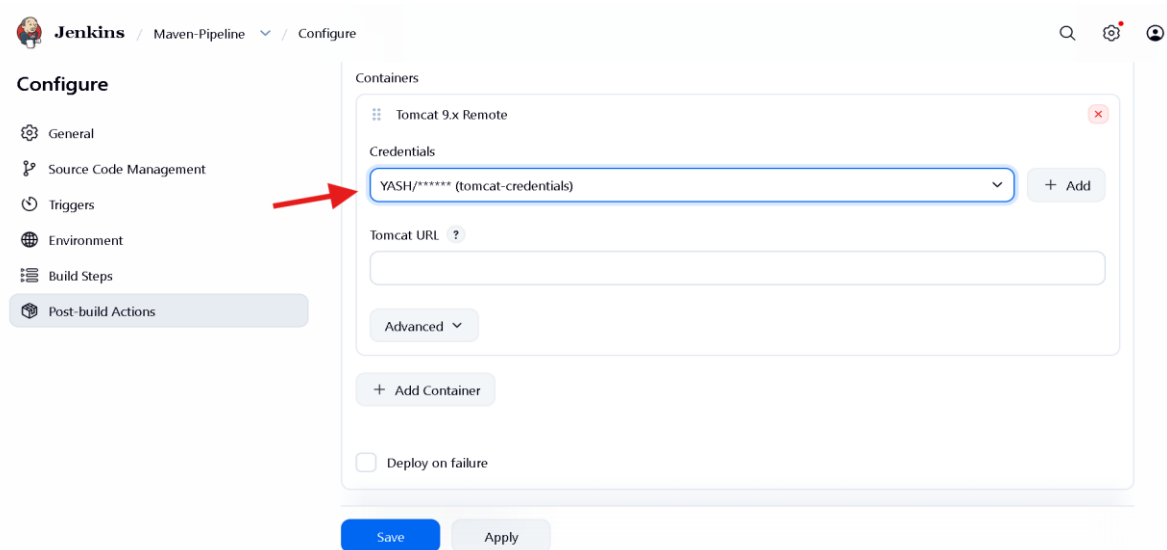
☐ Deploy on failure

Save Apply

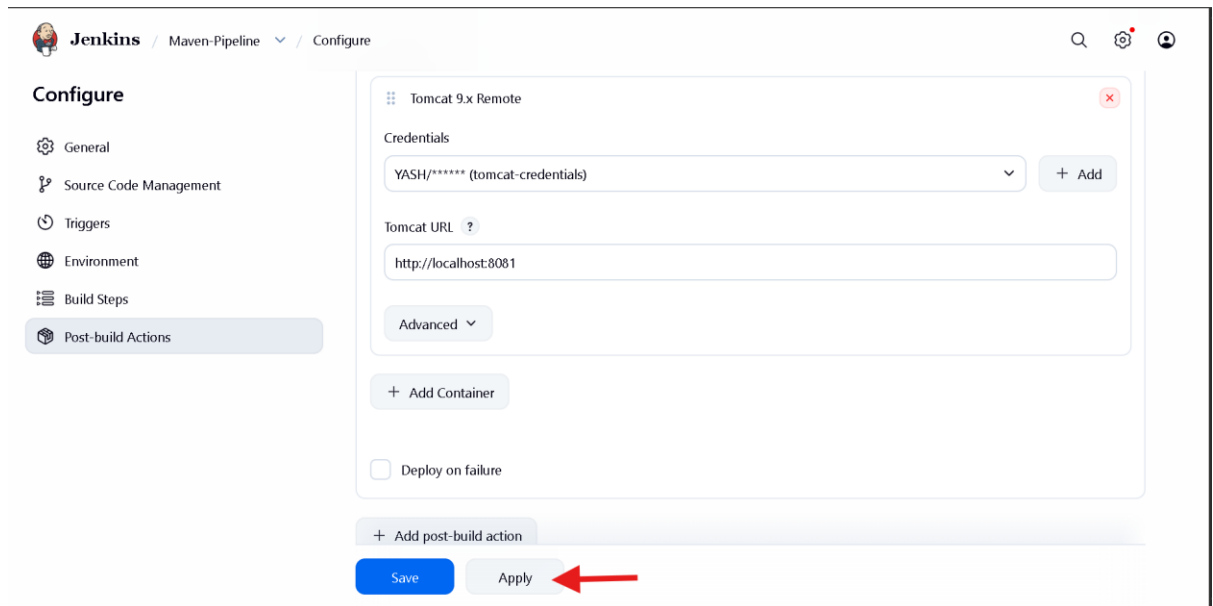
11. Now copy only `**/*.war` and paste it as shown in above image.



12. Click on Add Container and Choose Tomcat 9.x Remote and in credentials Select the credentials as shown in below image.



13. Now in Tomcat URL add: <http://localhost:8081> as shown

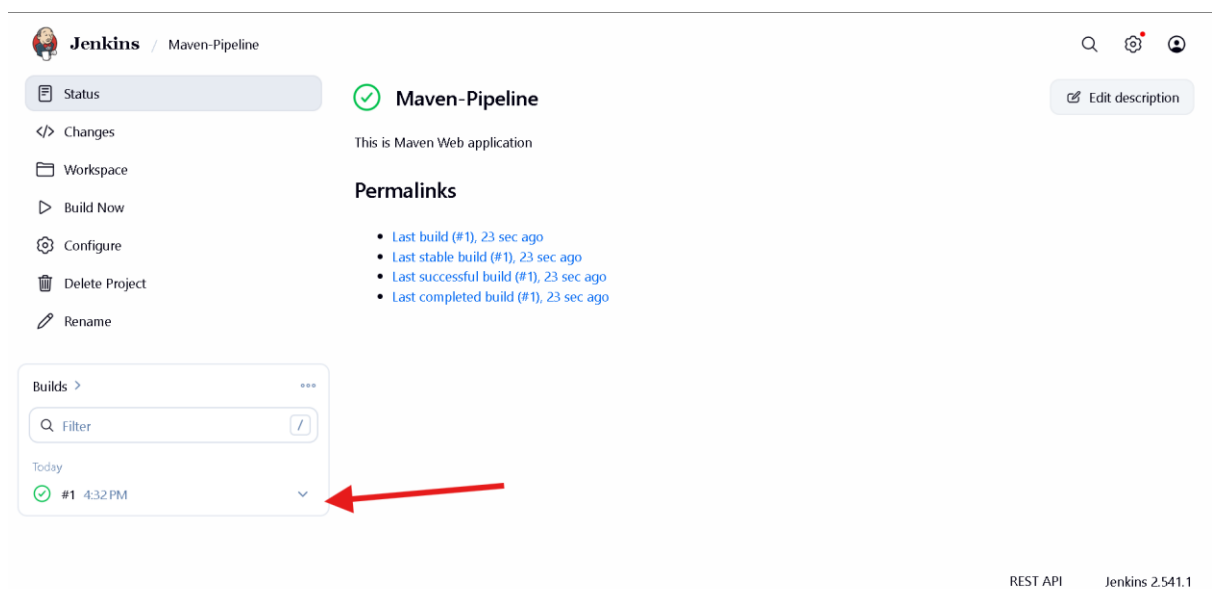


14. After adding Tomcat URL click apply and then save.

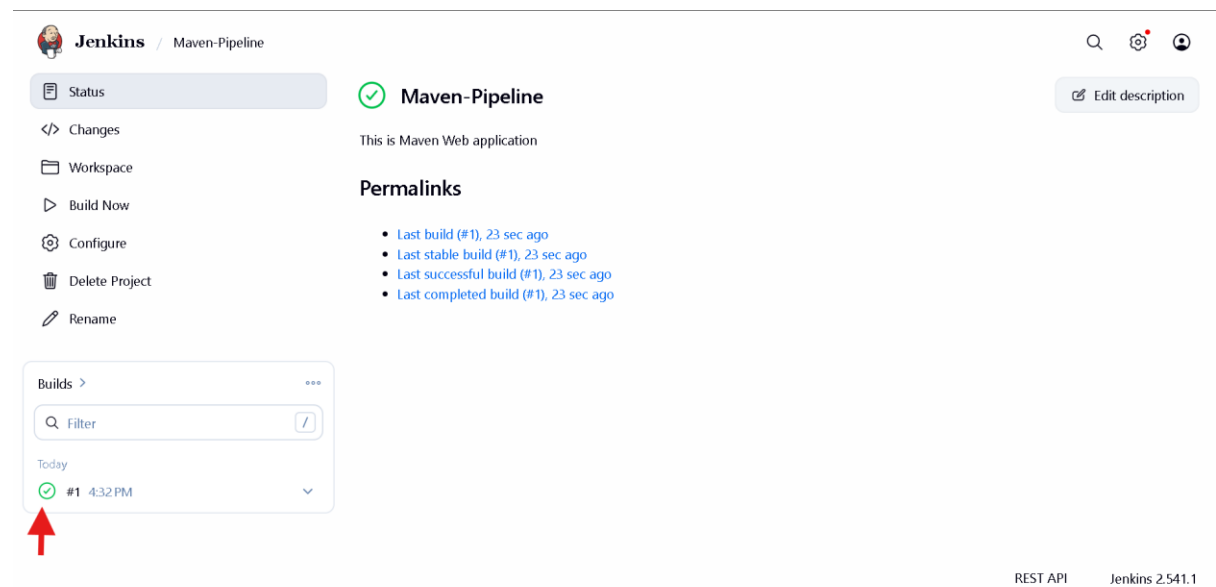
Build Job

Step 4: Build Now

1. Click on Build Now



2. Then Build is created and is Success.



The screenshot shows the Jenkins interface for a project named 'Maven-Pipeline'. The left sidebar contains navigation links: Status, Changes, Workspace, Build Now, Configure, Delete Project, and Rename. The main area displays the 'Maven-Pipeline' status as 'Success' with a green checkmark. Below this, it says 'This is Maven Web application'. A 'Permalinks' section lists four links: 'Last build (#1), 23 sec ago', 'Last stable build (#1), 23 sec ago', 'Last successful build (#1), 23 sec ago', and 'Last completed build (#1), 23 sec ago'. A 'Builds' table is shown with a search filter and a single entry for build #1, which is marked as successful with a green checkmark. A red arrow points to this entry. The bottom right corner shows 'REST API' and 'Jenkins 2.541.1'.

Jenkins / Maven-Pipeline

Status

Changes

Workspace

Build Now

Configure

Delete Project

Rename

Maven-Pipeline

This is Maven Web application

Permalinks

- Last build (#1), 23 sec ago
- Last stable build (#1), 23 sec ago
- Last successful build (#1), 23 sec ago
- Last completed build (#1), 23 sec ago

Builds

Filter

Today

#1 4:32 PM

REST API Jenkins 2.541.1

3. To check result click as shown in above image.



The screenshot shows the Jenkins interface for the 'Maven-Pipeline' project, specifically the build log for build #1. The log content is as follows:

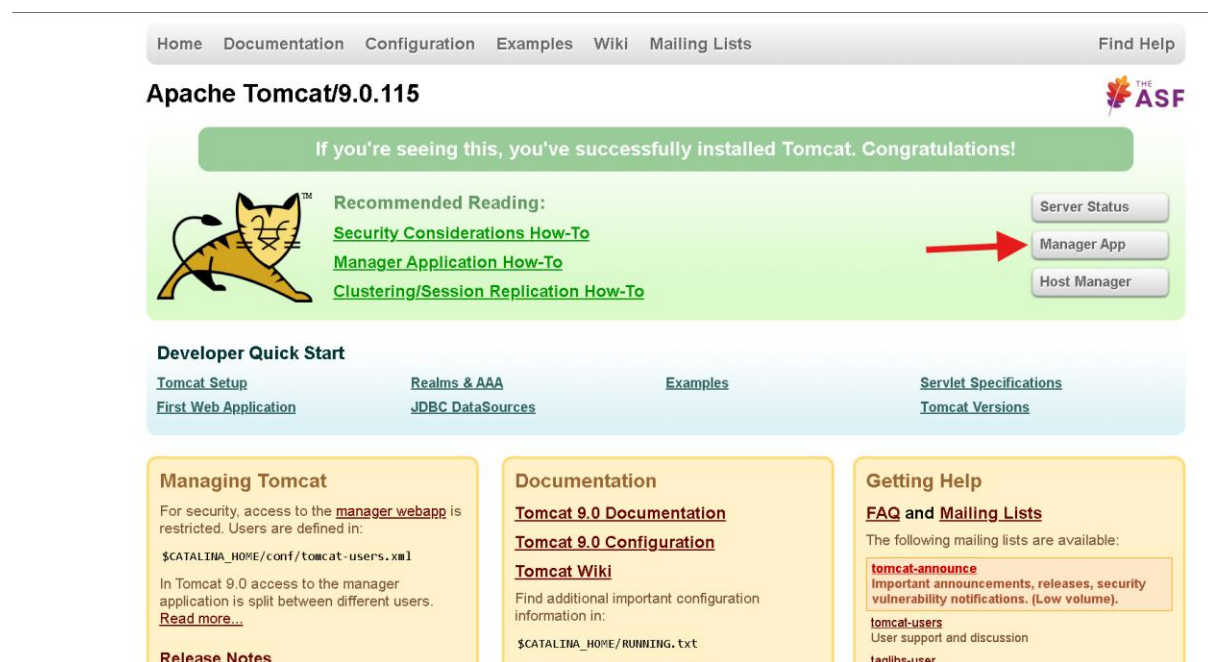
```
[INFO] Copying webapp resources [C:\ProgramData\Jenkins\.jenkins\workspace\Maven-Pipeline\src\main\webapp]
[INFO] Building war: C:\ProgramData\Jenkins\.jenkins\workspace\Maven-Pipeline\target\maven-web-application.war
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 8.040 s
[INFO] Finished at: 2026-01-28T16:32:42+05:30
[INFO] -----
[DeployPublisher][INFO] Attempting to deploy 1 war file(s)
[DeployPublisher][INFO] Deploying C:\ProgramData\Jenkins\.jenkins\workspace\Maven-Pipeline\target\maven-web-application.war to container Tomcat 9.x Remote with context null
Redeploying [C:\ProgramData\Jenkins\.jenkins\workspace\Maven-Pipeline\target\maven-web-application.war]
Undeploying [C:\ProgramData\Jenkins\.jenkins\workspace\Maven-Pipeline\target\maven-web-application.war]
Deploying [C:\ProgramData\Jenkins\.jenkins\workspace\Maven-Pipeline\target\maven-web-application.war]
Finished: SUCCESS
```

Jenkins 2.541.1

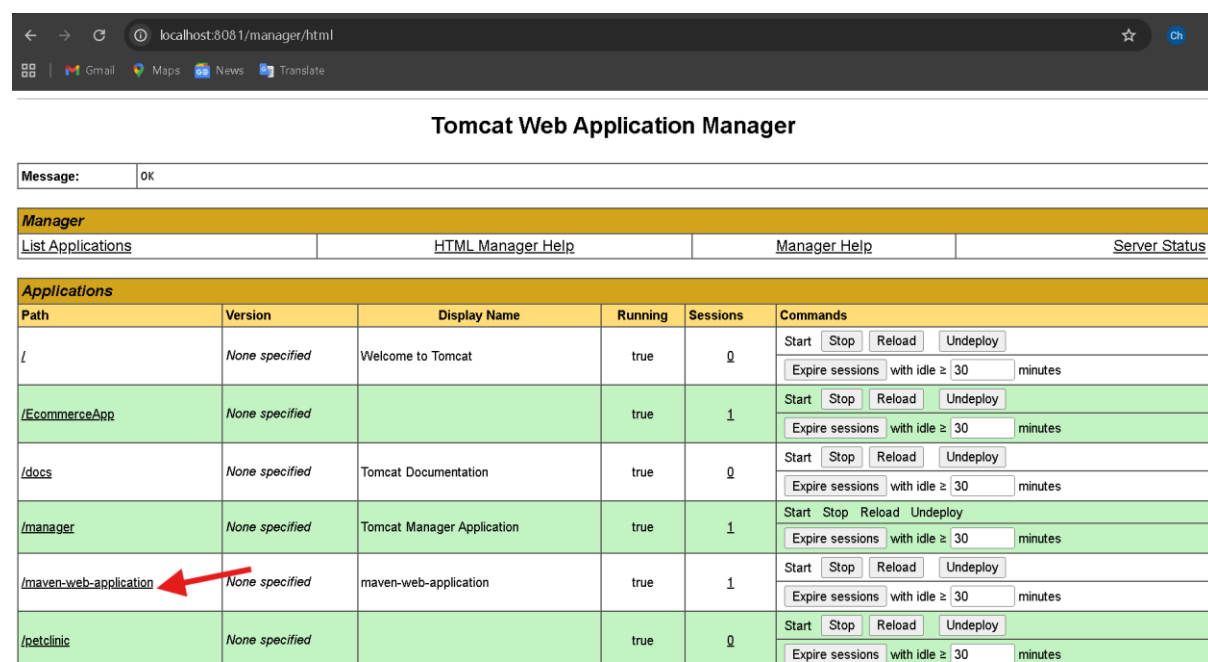
4. The above image shows the result.

Verify in Tomcat

Step 5: Check Final Result in Tomcat Server.

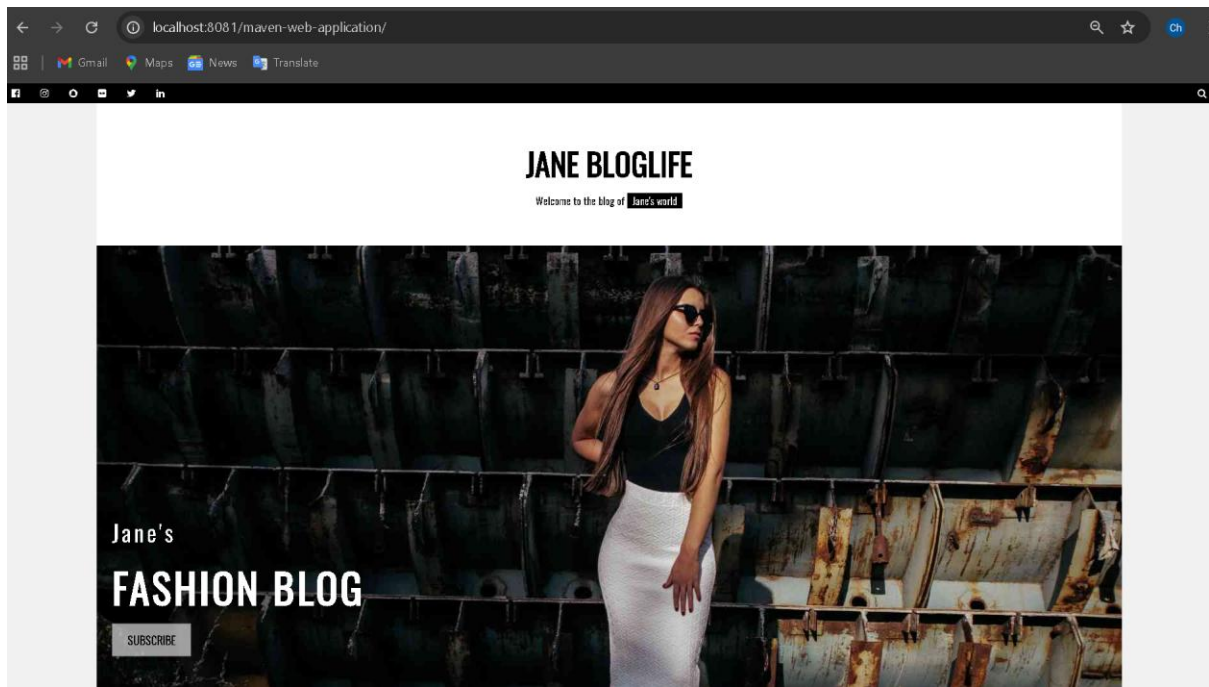


- Open Tomcat and click on Manager App



Path	Version	Display Name	Running	Sessions	Commands
/	None specified	Welcome to Tomcat	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/EcommerceApp	None specified		true	1	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
/manager	None specified	Tomcat Manager Application	true	1	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/maven-web-application	None specified	maven-web-application	true	1	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/petclinic	None specified		true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes

- Click on the /maven-web-application then the result page is displayed.



The above image is final result in Tomcat Server.