

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

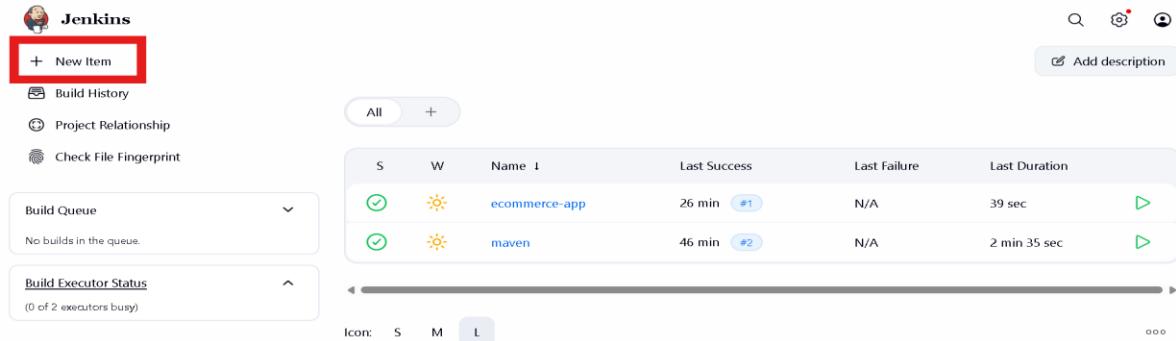
Steps to Open Jenkins and Create a New Pipeline Job

Step 1: Open Jenkins

Now:

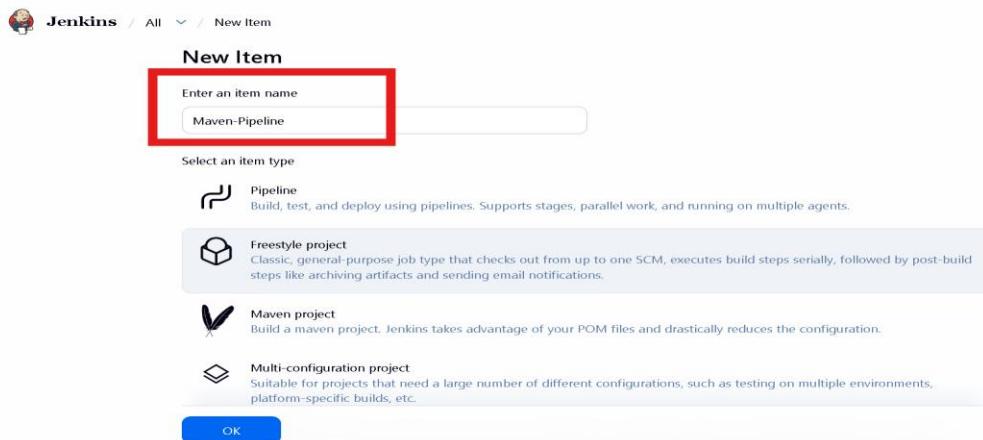
1. Start Jenkins service.
2. Open a browser.
3. Type the URL: <http://localhost:8080>
4. Jenkins dashboard will open.
5. Login with username and password.

Step 2: Create New Pipeline Job



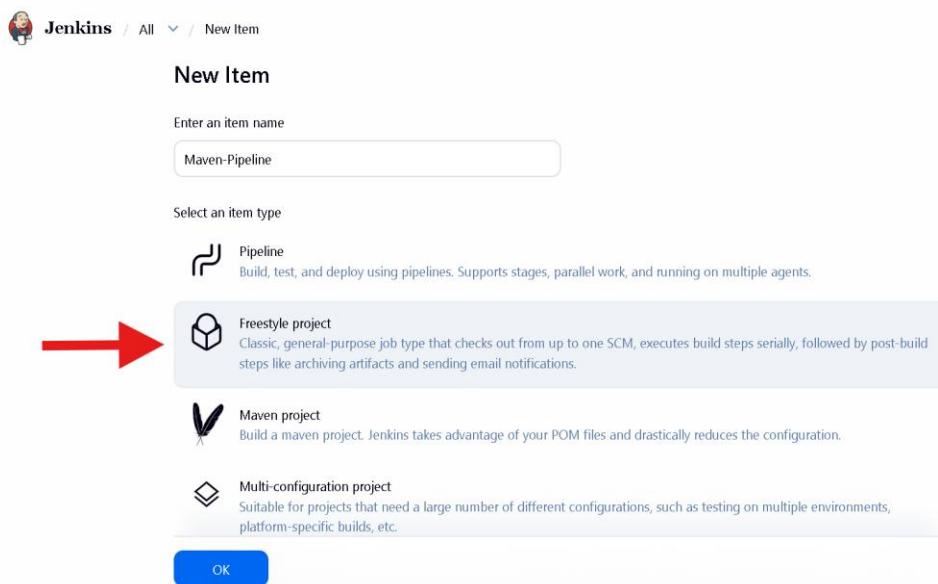
The screenshot shows the Jenkins dashboard. At the top left, there's a 'New Item' button with a red box around it. Below it are links for 'Build History', 'Project Relationship', and 'Check File Fingerprint'. On the right, there's a search bar and an 'Add description' button. In the center, there's a table showing build status for 'ecommerce-app' and 'maven'. At the bottom left, there's a 'Build Queue' section and a 'Build Executor Status' section.

1. On Jenkins dashboard, click on New Item.



The screenshot shows the 'New Item' dialog. The 'Enter an item name' field has 'Maven-Pipeline' typed into it, with a red box highlighting the input field. Below it, the 'Select an item type' section is shown. It includes four options: 'Pipeline' (selected), 'Freestyle project' (highlighted with a red box), 'Maven project', and 'Multi-configuration project'. A blue 'OK' button is at the bottom.

2. Enter a job name: Maven-Pipeline.



The screenshot shows the 'New Item' dialog again. The 'Enter an item name' field has 'Maven-Pipeline' typed into it. The 'Select an item type' section is shown, with the 'Freestyle project' option highlighted by a large red arrow. The other options ('Pipeline', 'Maven project', and 'Multi-configuration project') are also visible. A blue 'OK' button is at the bottom.

3. Select an item type as Freestyle Project.

Jenkins / All / New Item

New Item

Enter an item name

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

Jenkins / Maven-Pipeline / Configure

Configure

General

Source Code Management

Triggers

Environment

Build Steps

Post-build Actions

General

Description

This is Maven Web application

Plain text Preview

Enabled

Discard old builds ?

GitHub project

This project is parameterized ?

Throttle builds ?

Execute concurrent builds if necessary ?

Save Apply

1. Enter job description.

The screenshot shows the Jenkins 'Configure' screen for a project named 'Maven-Pipeline'. The left sidebar lists configuration sections: General, Source Code Management (selected), Triggers, Environment, Build Steps, and Post-build Actions. The main content area is titled 'Source Code Management' with the sub-instruction 'Connect and manage your code repository to automatically pull the latest code for your builds.' A radio button for 'None' is selected. A red arrow points to the 'Git' radio button, which is unselected. Below the radio buttons is the 'Triggers' section, which contains several checkboxes for different triggering options.

2. Select Git in Source Code Management.

This screenshot shows the same Jenkins 'Configure' page as above, but with the 'Git' radio button selected, indicated by a red arrow. The 'Source Code Management' section now displays fields for 'Repository URL' (set to 'https://github.com/Msocial123/maven-web-application.git') and 'Credentials' (set to '- none -'). There is also an 'Advanced' dropdown and a '+ Add Repository' button. The 'Save' and 'Apply' buttons are at the bottom.

This screenshot shows the 'Configure' page with the 'Branches to build' section expanded. A red arrow points to the 'Branch Specifier' field, which contains '/master'. Other visible sections include 'Repository browser' (set to '(Auto)') and 'Additional Behaviours' (with a '+ Add' button). The 'Save' and 'Apply' buttons are at the bottom.

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

The screenshot shows the Jenkins 'Configure' screen for a job named 'Maven-Pipeline'. The left sidebar has sections: General, Source Code Management, Triggers, Environment, Build Steps (which is selected and highlighted in grey), and Post-build Actions. The main area is titled 'Build Steps' with the sub-instruction 'Automate your build process with ordered tasks like code compilation, testing, and deployment.' A red arrow points to the '+ Add build step' button. Below it is the 'Post-build Actions' section with a '+ Add post-build action' button. At the bottom are 'Save' and 'Apply' buttons.

Build Configuration

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

The screenshot shows the same Jenkins 'Configure' screen for 'Maven-Pipeline'. The 'Build Steps' section is open, showing a dropdown menu with various options: Execute Windows batch command, Execute shell, Invoke Ant, Invoke Gradle script, and **Invoke top-level Maven targets**. A red arrow points to the 'Invoke top-level Maven targets' option. The right side of the screen shows the 'Post-build Actions' section with the instruction 'Define what happens after a build completes, like sending notifications, archiving artifacts, or triggering other jobs.' At the bottom are 'REST API' and 'Jenkins 2.541.1' links.

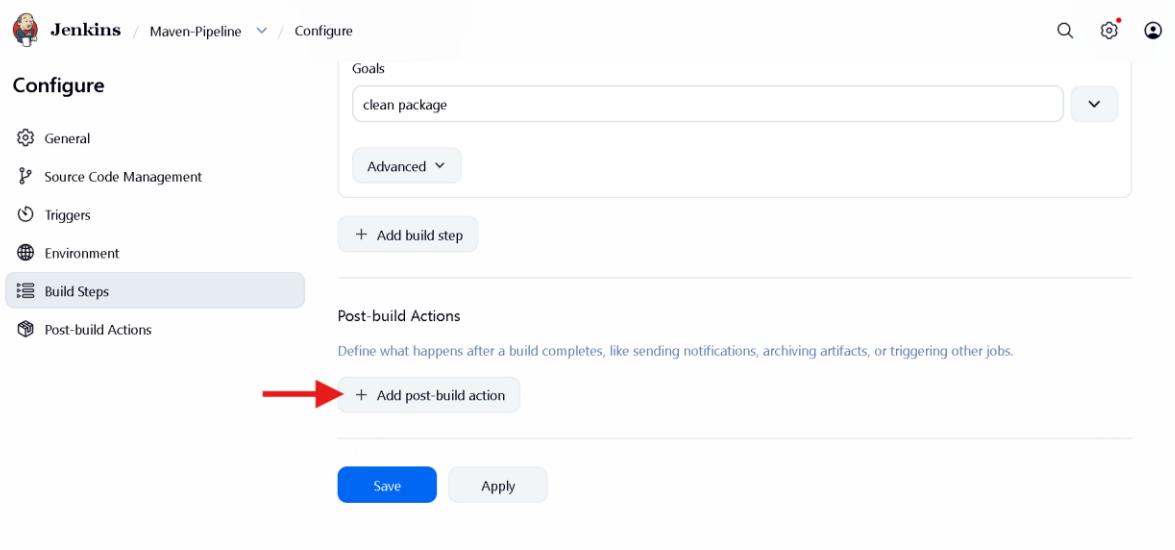
5. In the options Select Invoke top-level Maven Targets.

The screenshot shows the Jenkins 'Configure' screen for a 'Maven-Pipeline' job. On the left, there's a sidebar with links: General, Source Code Management, Triggers, Environment, Build Steps (which is highlighted with a blue background), and Post-build Actions. The main area is titled 'Build Steps' with the sub-instruction 'Automate your build process with ordered tasks like code compilation, testing, and deployment.' Below this is a card for 'Invoke top-level Maven targets'. It has a 'Maven Version' dropdown set to '(Default)', which is highlighted with a red arrow. There's also a 'Goals' input field and an 'Advanced' button. At the bottom of the card are 'Save' and 'Apply' buttons.

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

This screenshot shows the same Jenkins 'Configure' screen as the previous one, but with changes made to the 'Invoke top-level Maven targets' step. The 'Maven Version' dropdown now contains the value 'maven', indicated by a red arrow. The 'Goals' input field contains the text 'clean package', also indicated by a red arrow. The rest of the interface remains the same, with the 'Build Steps' tab still selected in the sidebar.

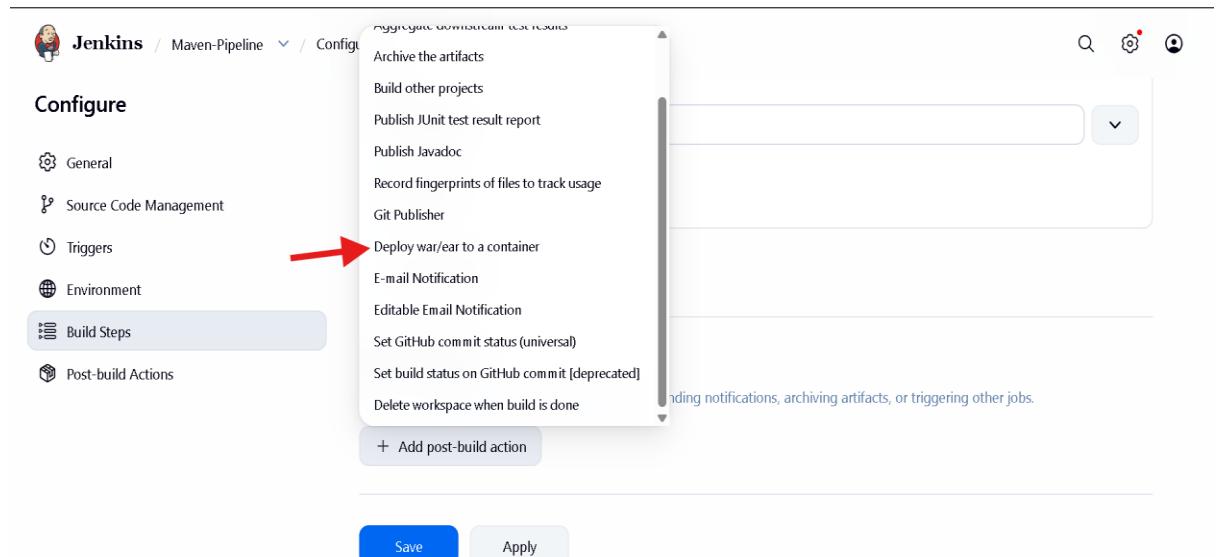
7. Add clean package in Goals and move to next step.



The screenshot shows the Jenkins configuration interface for a job named "Maven-Pipeline". The left sidebar has sections: General, Source Code Management, Triggers, Environment, Build Steps (selected), and Post-build Actions. The main area shows the "Goals" section with "clean package" entered in the text field. Below it is an "Advanced" dropdown and a "+ Add build step" button. The "Post-build Actions" section is expanded, showing various actions like "Archive the artifacts", "Build other projects", etc., with a red arrow pointing to the "+ Add post-build action" button. At the bottom are "Save" and "Apply" buttons.

Post-Build Actions:

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



The screenshot shows the Jenkins configuration interface for a job named "Maven-Pipeline". The left sidebar has sections: General, Source Code Management, Triggers (selected), Environment, Build Steps, and Post-build Actions. The main area shows the "Post-build Actions" section with a list of actions. A red arrow points to the "Deploy war/ear to a container" option in the list.

REST API Jenkins 2.541.1

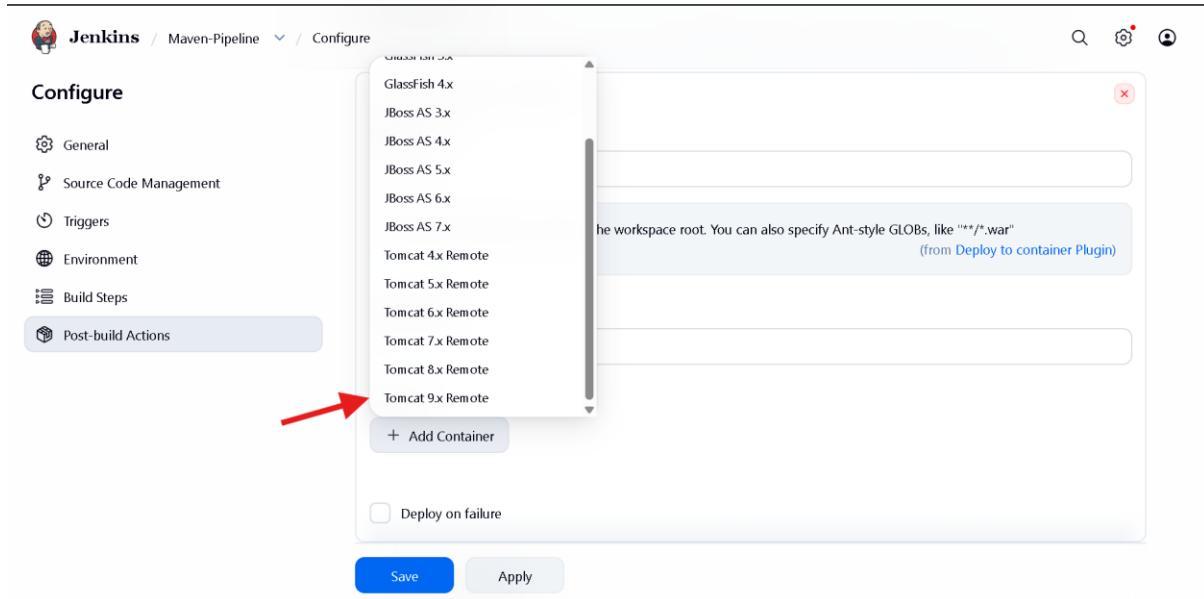
9. In the options Select Deploy war/ear to a container.

The screenshot shows the Jenkins 'Configure' screen for a 'Maven-Pipeline' job. On the left, there is a sidebar with links: General, Source Code Management, Triggers, Environment, Build Steps, and Post-build Actions (which is currently selected). The main area is titled 'Deploy war/ear to a container'. It contains a 'WAR/EAR files' input field with a question mark icon, which has a red arrow pointing to it. Below this is a note: 'War/ear files to deploy. Relative to the workspace root. You can also specify Ant-style GLOBs, like "*/*.war"' (from Deploy to container Plugin). There is also a 'Context path' input field and a 'Containers' section with a '+ Add Container' button. At the bottom are 'Save' and 'Apply' buttons.

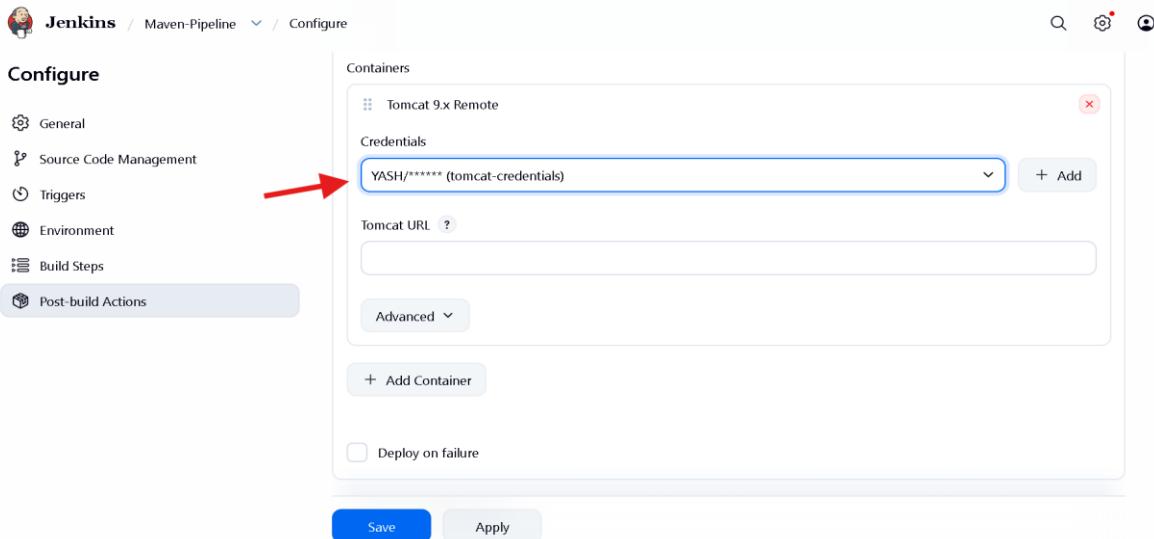
10. Now Click on the symbol ? after WAR/EAR files.

This screenshot is identical to the one above, but the help icon (the question mark inside a circle) next to the 'WAR/EAR files' input field is now highlighted with a red arrow. The rest of the interface remains the same, showing the configuration options for deploying war/ear files.

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

The screenshot shows the Jenkins 'Configure' screen for the 'Maven-Pipeline' job. On the left, there's a sidebar with options like General, Source Code Management, Triggers, Environment, Build Steps, and Post-build Actions. The 'Post-build Actions' section is currently selected. In the main area, there's a 'Tomcat 9.x Remote' configuration block. It includes fields for 'Credentials' (set to 'YASHI/******** (tomcat-credentials)'), 'Tomcat URL' (set to 'http://localhost:8081'), and an 'Advanced' dropdown. Below these are buttons for '+ Add Container' and 'Deploy on failure'. At the bottom of the block are '+ Add post-build action', 'Save', and 'Apply' buttons. A red arrow points to the 'Apply' button.

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

Build Job

Step 4: Build Now

1. Click on Build Now

The screenshot shows the Jenkins 'Build Now' screen for the 'Maven-Pipeline' job. On the left, there's a sidebar with options like Status, Changes, Workspace, Build Now, Configure, Delete Project, and Rename. The 'Build Now' option is selected. In the main area, there's a 'Builds >' section with a 'Filter' input field. Below it, under 'Today', there's a list showing a single build: '#1 4:32 PM'. A red arrow points to this build entry. The top right of the page has 'Edit description' and 'Edit' buttons. At the bottom right, there are links for 'REST API' and 'Jenkins 2.54.1'.

2. Then Build is created and is Success.

The screenshot shows the Jenkins interface for the 'Maven-Pipeline' project. At the top, there's a navigation bar with links for Status, Changes, Workspace, Build Now, Configure, Delete Project, and Rename. The main content area has a green checkmark icon and the text 'Maven-Pipeline'. Below it, a message says 'This is Maven Web application'. A section titled 'Permalinks' lists four items: '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'. On the left, there's a sidebar titled 'Builds >' with a 'Filter' input field. Under 'Today', a single build is listed: '#1 4:32 PM' with a green checkmark. A red arrow points to this build entry. At the bottom right, there are links for 'REST API' and '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 #1. The log output 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
```

At the bottom right, it says 'Jenkins 2.541.1'.

4. The above image shows the result.

Verify in Tomcat

Step 5: Check Final Result in Tomcat Server.

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

Apache Tomcat/9.0.115

Recommended Reading:

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

Developer Quick Start

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

Managing Tomcat

For security, access to the [manager webapp](#) is restricted. Users are defined in: [\\$CATALINA_HOME/conf/tomcat-users.xml](#)

In Tomcat 9.0 access to the manager application is split between different users. [Read more...](#)

[Release Notes](#)

Documentation

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

Find additional important configuration information in:
[\\$CATALINA_HOME/ RUNNING.txt](#)

Getting Help

[FAQ and Mailing Lists](#)

The following mailing lists are available:

[tomcat-announce](#)
Important announcements, releases, security vulnerability notifications. (Low volume).

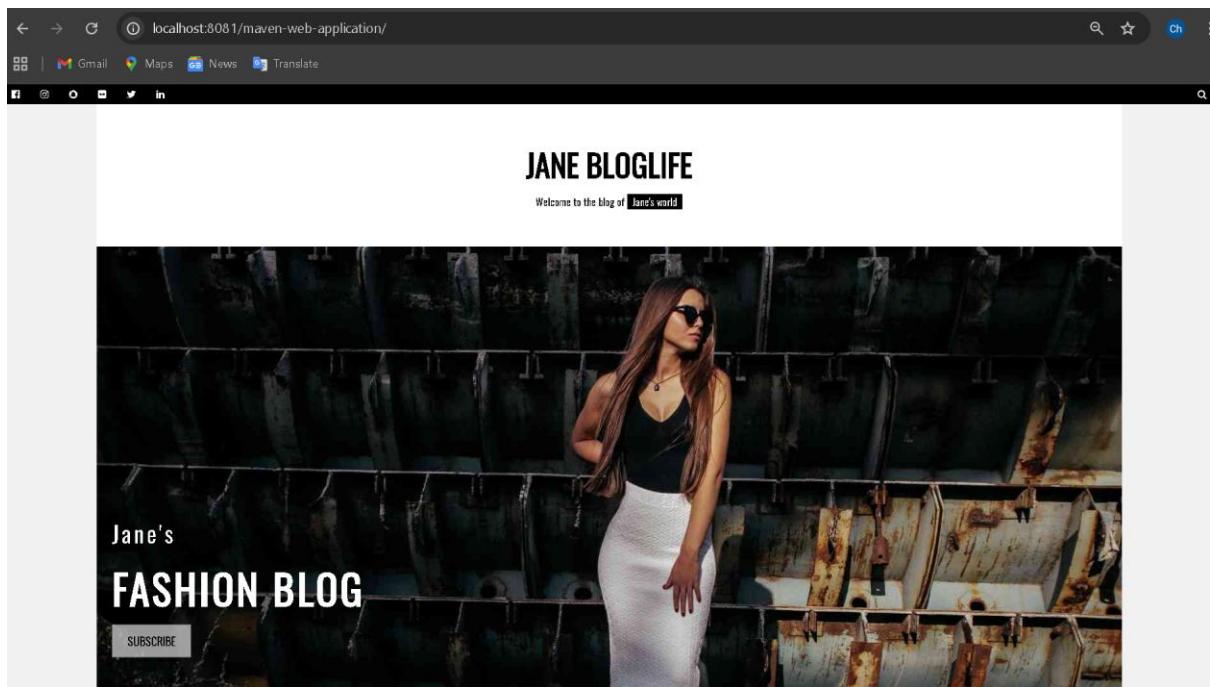
[tomcat-users](#)
User support and discussion

[taglibs-user](#)

- Open Tomcat and click on Manager App

Applications					
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.