

Jenkins Pipeline

This document explains how to create a Jenkins Pipeline from start to end using Jenkins and GitHub.

REQUIREMENTS:-

1. Install Jenkins
2. Install tomcat
3. Install java
4. Install git
5. Within manage Jenkins -> plugins->available plugins add deploy and maven plugins
6. Within manage Jenkins->tools give paths of git, jdk, maven, etc

Maven Web Application Pipeline

A Maven Web Application is a Java-based web application that is built and managed using Apache Maven.

What is Apache Maven?

Apache Maven is a build automation and project management tool used mainly for Java projects. It uses a file called pom.xml (Project Object Model) to manage:

- Project dependencies
- Build lifecycle
- Plugins
- Project configuration

Step 1 : Go to Jenkins using port number

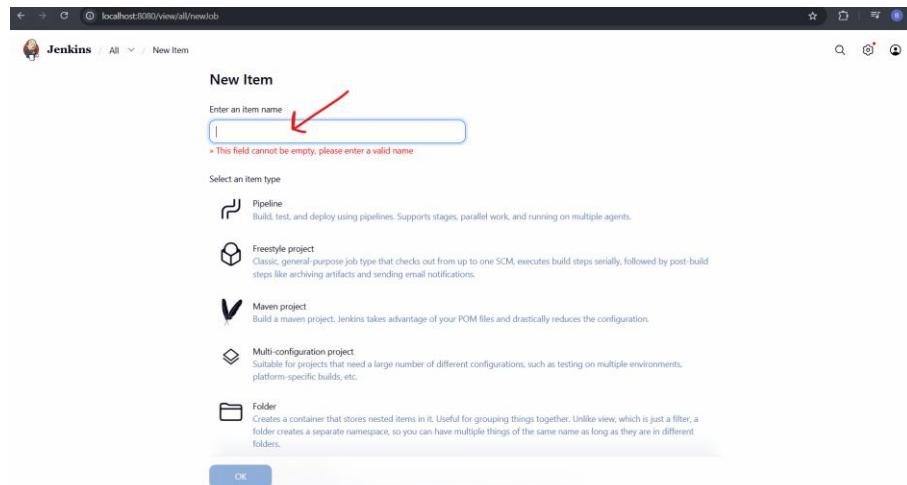
The screenshot shows the Jenkins home page at localhost:8080. A red arrow points to the Jenkins logo in the top left corner. The main content area displays a table of projects with the following data:

S	W	Name	Last Success	Last Failure	Last Duration
		e_commerce_app	30 min	N/A	30 sec
		maven-pipeline	55 min	5 hr 40 min	35 sec
		pet-clinic	49 min	53 min	2 min 56 sec

At the bottom of the page, there are links for REST API and Jenkins 2.541.1.

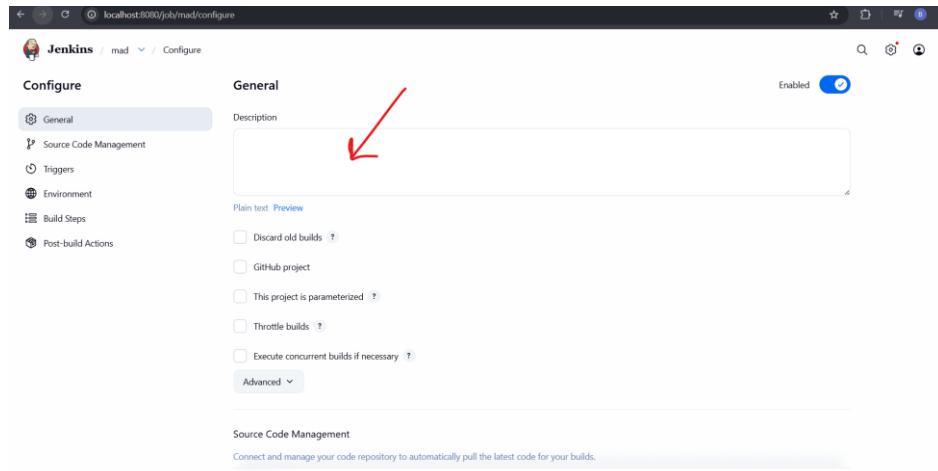
This is home page of Jenkins.

Step 2: click on “new item” to create new pipeline



Here you have to give name to pipeline and select item type as “Freestyle Project” then click on “ok”

Step 3: Give general description for pipeline



Here you have give general description of pipeline

Step 4: Add project GitHub link

The screenshot shows the Jenkins job configuration page for a job named 'mad'. Under the 'Source Code Management' section, 'Git' is selected. A red arrow points to the 'Repository URL' field, which contains a placeholder 'Please enter Git repository.'. Another red arrow points to the 'Branch Specifier' field, which contains the value '/master'. The 'Save' and 'Apply' buttons are visible at the bottom.

Add project GitHub repository link and also check for branch

Step 5: Build step

The screenshot shows the Jenkins job configuration page for a job named 'mad'. Under the 'Build Steps' section, a dropdown menu is open, showing various build steps. A red arrow points to the 'Invoke top-level Maven targets' option. The 'Build Steps' section also includes options like 'Execute Windows batch command', 'Invoke Ant', and 'Invoke Gradle script'. The 'Build Steps' section has a note about sending notifications, archiving artifacts, or triggering other jobs.

Here choose built step as "Invoke top- level Maven targets"

Step 7: Configure build step

The screenshot shows the Jenkins job configuration page for a job named "mad". The left sidebar has "Build Steps" selected. 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 "Invoke top-level Maven targets" step. Under "Maven Version", "maven" is selected. Under "Goals", "clean package" is listed. An "Advanced" section is expanded, showing fields for "POM" (with a red arrow pointing to it) and "Properties". At the bottom is a "JVM Options" field.

Declare Maven version as “**maven**” and Goals as “**clean package**”, and if **POM.xml**(Project Object Model) file is not in same branch of GitHub then In Advanced section of Build Step add POM.xml file path

Step 8: Post Build Action

The screenshot shows the Jenkins job configuration page for a job named "mad". The left sidebar has "Post-build Actions" selected. The main area shows a list of actions: "Inject build variables" (unchecked), "Aggregate downstream test results", "Archive the artifacts", "Build other projects", "Publish JUnit test result", "Publish JavaDoc", "Record fingerprints of files to track usage", "Git Publisher", "Deploy war/ear to a container" (highlighted with a red arrow), "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". At the bottom are "Save" and "Apply" buttons.

In Post Build Action, add post-build action as “**Deploy war/ear to a container**”

Step 9: Add war/ ear files

The screenshot shows the Jenkins job configuration page for a job named 'mad'. The 'Post-build Actions' section is selected in the sidebar. Under 'Deploy war/ear to a container', there is a 'WAR/EAR files' input field containing the placeholder text '**/*.war'. A red arrow points to the question mark icon next to the placeholder text. Below the input field is a note: 'War/ear files to deploy. Relative to the workspace root. You can also specify Ant-style GLOBs, like **/*.war'. At the bottom of the section are 'Save' and 'Apply' buttons.

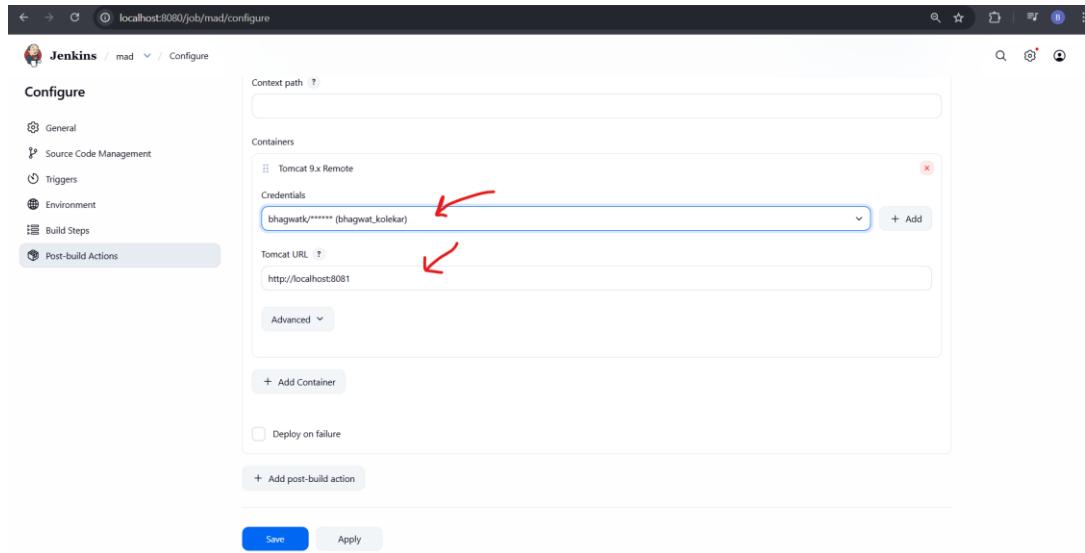
Click on “?” to het war/ ear files, copy text in highlighted blue i.e (**/*.war) and paste it in war/ear files section

Step 10: Add Containers

The screenshot shows the Jenkins job configuration page for a job named 'mad'. The 'Post-build Actions' section is selected in the sidebar. A dropdown menu titled 'Containers' is open, listing various application servers and containers: GlassFish 2.x, GlassFish 3.x, GlassFish 4.x, JBoss AS 3.x, JBoss AS 4.x, JBoss AS 5.x, JBoss AS 6.x, JBoss AS 7.x, Tomcat 4.x Remote, Tomcat 5.x Remote, Tomcat 6.x Remote, Tomcat 7.x Remote, Tomcat 8.x Remote, and Tomcat 9.x Remote. Below the dropdown is a 'Deploy on failure' checkbox and a '+ Add post-build action' button. At the bottom are 'Save' and 'Apply' buttons.

Add Container as Tomcat version which you have downloaded.

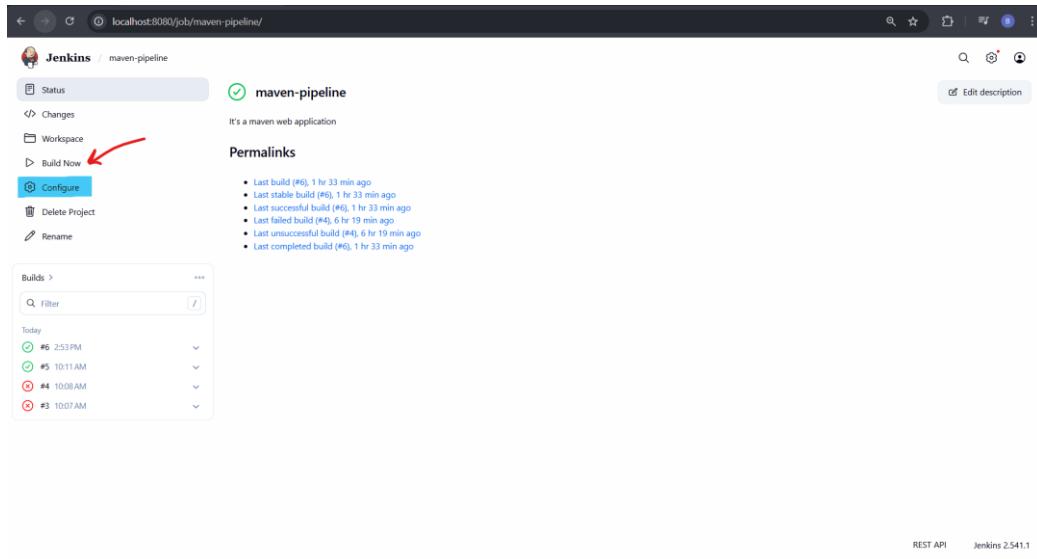
Step 11: Add Tomcat Credentials



The screenshot shows the Jenkins job configuration page for a job named 'mad'. Under the 'Post-build Actions' section, there is a 'Tomcat 9.x Remote' container. The 'Credentials' dropdown is set to 'bhagwat***** (bhagwat_kolekar)', and the 'Tomcat URL' input field contains 'http://localhost:8081'. There are two red arrows: one pointing to the credentials dropdown and another pointing to the Tomcat URL field.

Here I have chosen Tomcat 9 and Given Tomcat account credentials also add Tomcat URL

Step 12: Configure and Build

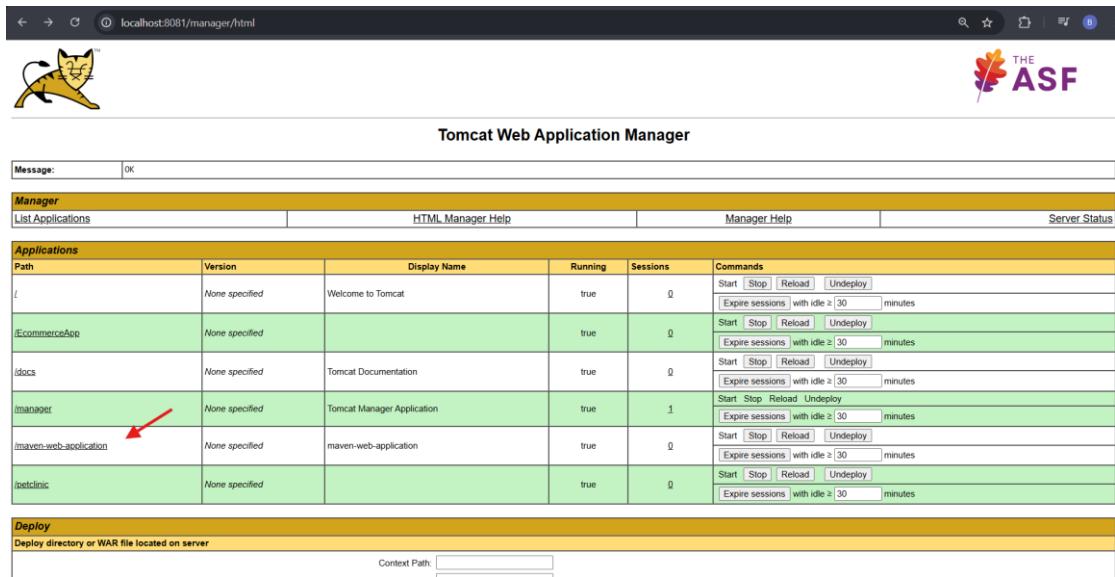


The screenshot shows the Jenkins dashboard for the 'maven-pipeline' job. On the left, there is a sidebar with options: Status, Changes, Workspace (with a red arrow pointing to the 'Build Now' button), Configure (highlighted with a blue background), Delete Project, and Rename. The main area shows the job status as 'maven-pipeline' (green checkmark) and 'It's a maven web application'. Below this is a 'Permalinks' section with a list of recent builds. At the bottom, there is a 'Builds' section showing four builds: #6 (green circle, 2:53PM), #5 (green circle, 10:11AM), #4 (red circle, 10:08AM), and #3 (red circle, 10:07AM). The 'Configure' button is highlighted with a blue box and a red arrow.

First go to **Configure** and check all configurations then click on “**Build Now**”

If Console output is “**Success**” go to next step otherwise resolve errors.

Step 13: Go to Tomcat Web Application Manager using port number



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	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
/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	0	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

This is the Home Page of Tomcat Web Application Manager, Under Applications Check for “App name”

then click on it

Step 14: Output

