

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. With in manage Jenkins -> plugins->available plugins add deploy and maven plugins
6. With in manage Jenkins->tools give paths of git, jdk, maven, etc

Maven Web Application Pipeline

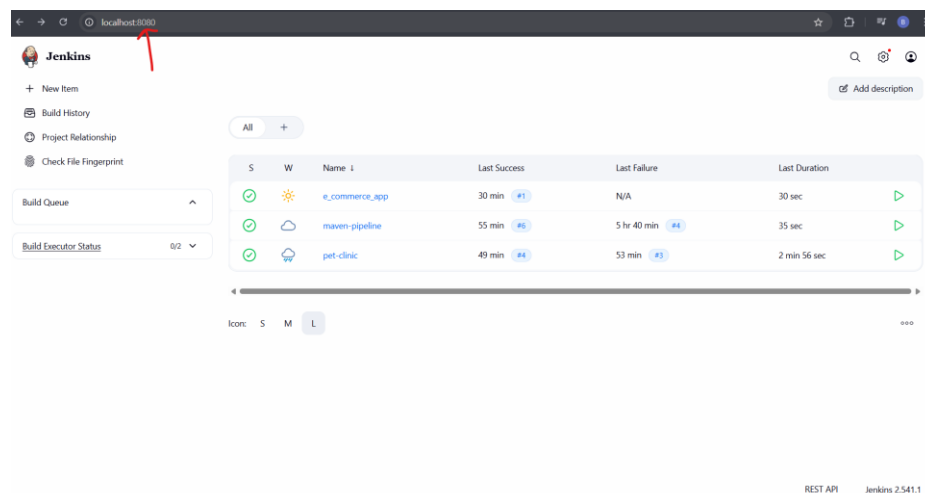
A Maven Web Application is a Java-based web application that is built and managed using Apache Maven.

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:

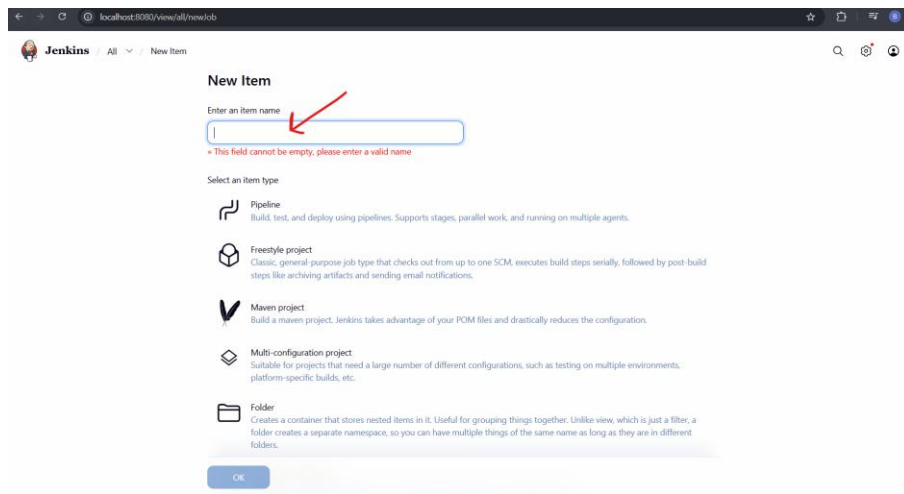
- Project dependencies
- Build lifecycle
- Plugins
- Project configuration

Step 1 : Go to Jenkins using port number



This is home page of Jenkins.

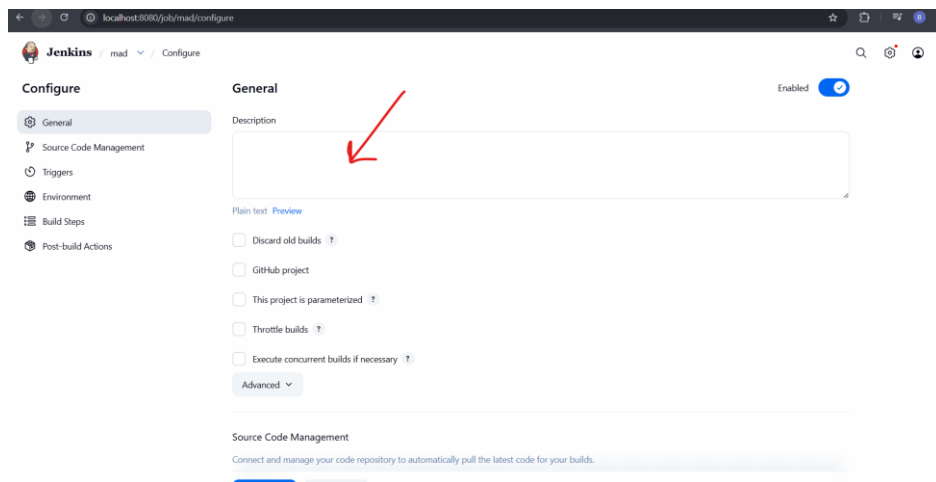
Step 2: click on “new item” to create new pipeline



The screenshot shows the Jenkins 'New Item' page. At the top, there's a breadcrumb trail: 'Jenkins / All / New Item'. The main heading is 'New Item'. Below it, there's a text input field labeled 'Enter an item name'. A red arrow points to this field. Below the input field, there's a red error message: 'This field cannot be empty, please enter a valid name'. Underneath, there's a section titled 'Select an item type' with five options: 'Pipeline' (with a description: 'Build, test, and deploy using pipelines. Supports stages, parallel work, and running on multiple agents.'), 'Freestyle project' (with a description: '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' (with a description: 'Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.'), 'Multi-configuration project' (with a description: 'Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.'), and 'Folder' (with a description: '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.'). At the bottom, there's a blue 'OK' button.

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

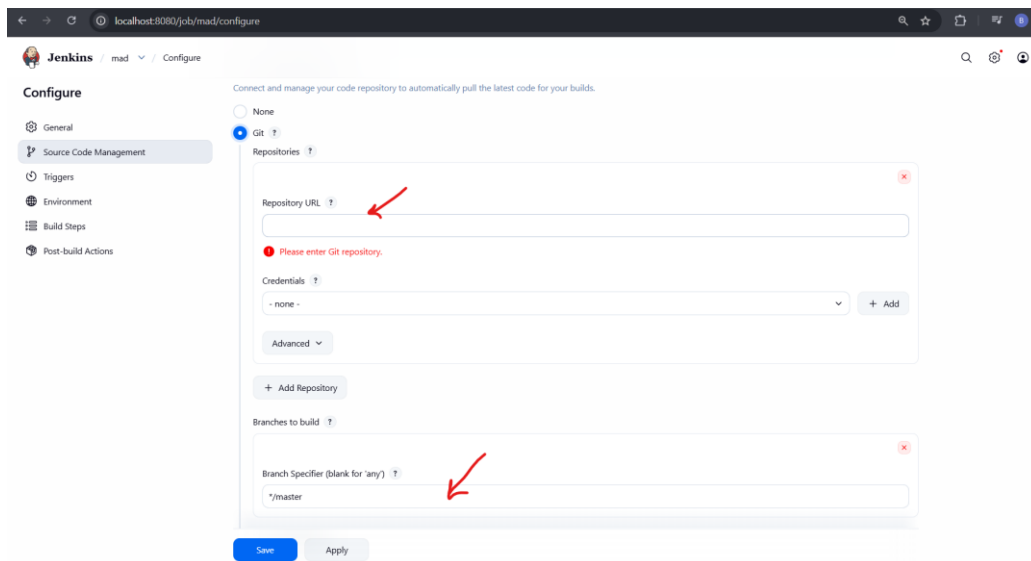
Step 3: Give general description for pipeline



The screenshot shows the Jenkins 'Configure' page for a new item. The breadcrumb trail is 'Jenkins / mad / Configure'. The left sidebar has a 'Configure' section with several options: 'General' (selected), 'Source Code Management', 'Triggers', 'Environment', 'Build Steps', and 'Post-build Actions'. The main content area is titled 'General' and has an 'Enabled' toggle switch. Below the title, there's a 'Description' section with a large text input field. A red arrow points to this field. Below the input field, there's a 'Plain text' label and a 'Preview' link. Underneath, there are five checkboxes: 'Discard old builds', 'GitHub project', 'This project is parameterized', 'Throttle builds', and 'Execute concurrent builds if necessary'. Below these checkboxes, there's an 'Advanced' dropdown menu. At the bottom, there's a 'Source Code Management' section with a description: 'Connect and manage your code repository to automatically pull the latest code for your builds.'.

Here you have give general description of pipeline

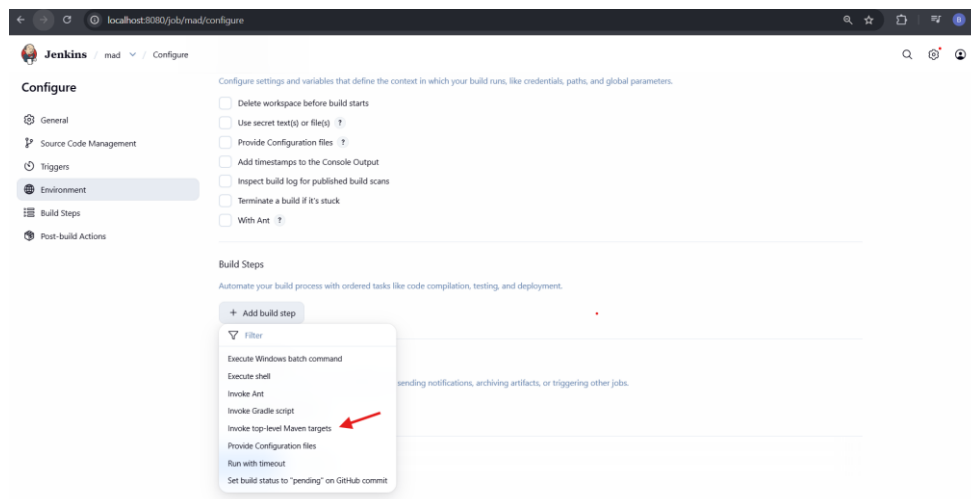
Step 4: Add project GitHub link



The screenshot shows the Jenkins configuration page for a job named 'mad'. The 'Source Code Management' tab is selected. Under the 'Git' section, the 'Repository URL' field is highlighted with a red arrow. Below it, a red error message states 'Please enter Git repository.' The 'Credentials' dropdown is set to 'none'. The 'Branches to build' section has a 'Branch Specifier (blank for 'any')' field, which is also highlighted with a red arrow and contains the text '*/master'. At the bottom, there are 'Save' and 'Apply' buttons.

Add project GitHub repository link and also check for branch

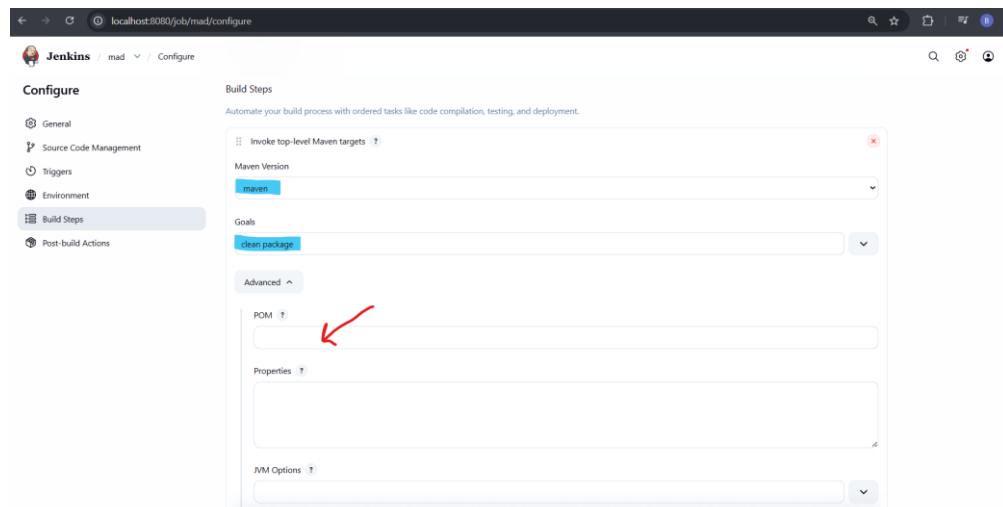
Step 5: Build step



The screenshot shows the Jenkins configuration page for the same job, now on the 'Build Steps' tab. The 'Add build step' button is clicked, opening a dropdown menu. The menu lists several build steps, and 'Invoke top-level Maven targets' is highlighted with a red arrow. Other options include 'Execute Windows batch command', 'Execute shell', 'Invoke Ant', 'Invoke Gradle script', 'Provide Configuration files', 'Run with timeout', and 'Set build status to "pending" on GitHub commit'.

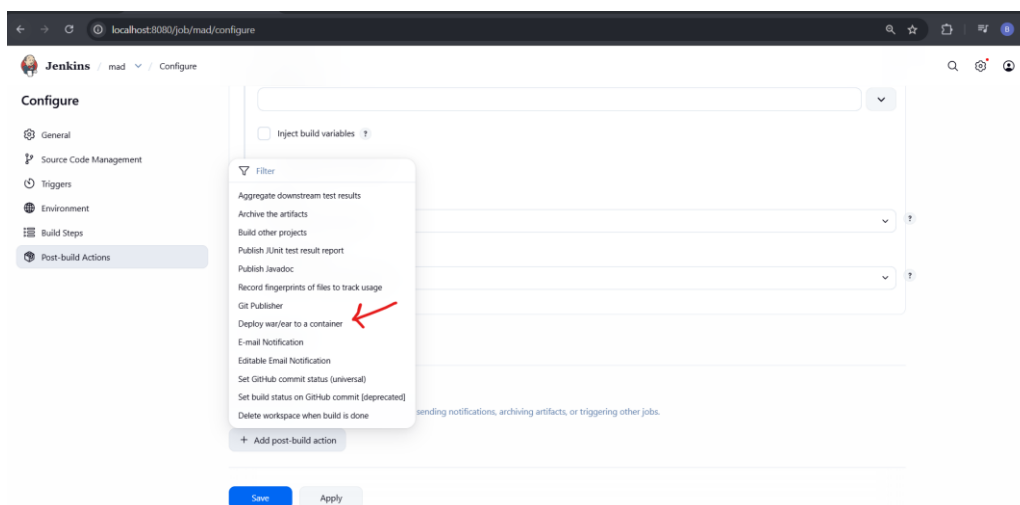
Here choose built step as “Invoke top- level Maven targets”

Step 7: Configure build step



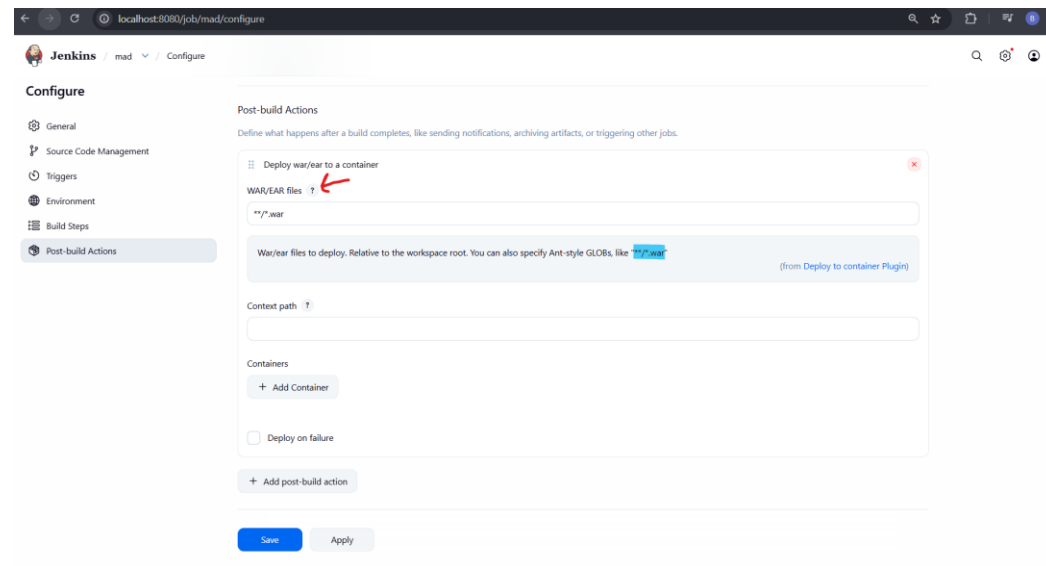
Declare Maven version as “**maven**” and Goals as “**clean package**”, and if **POM.xml**(Project Object Model) file is not in same branch of GitHub then In Advanced section of Build Step add POM.xml file path

Step 8: Post Build Action



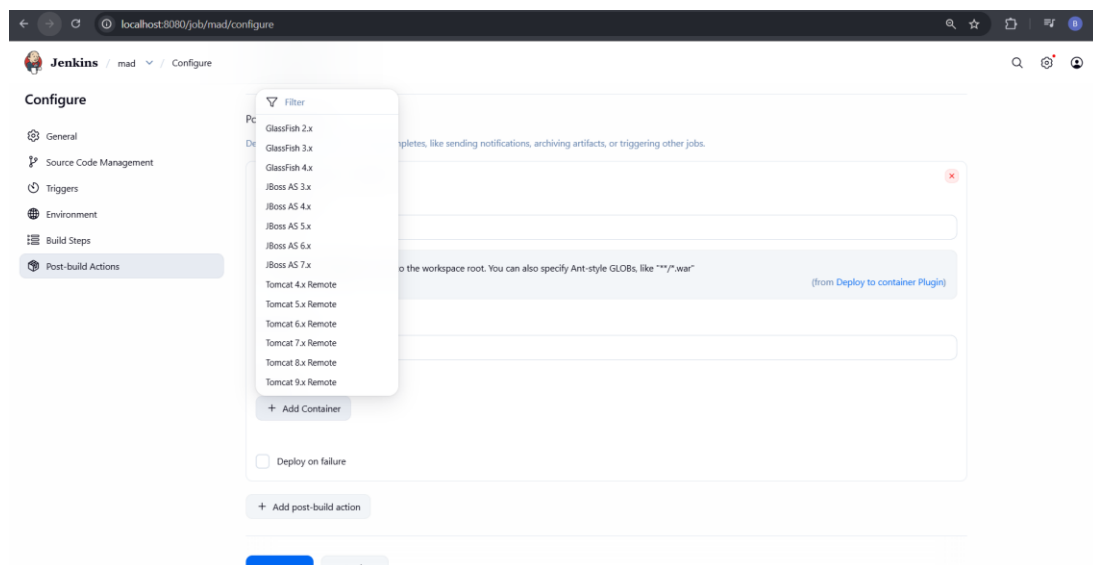
In Post Build Action, add post-build action as “**Deploy war/ear to a container**”

Step 9: Add war/ ear files



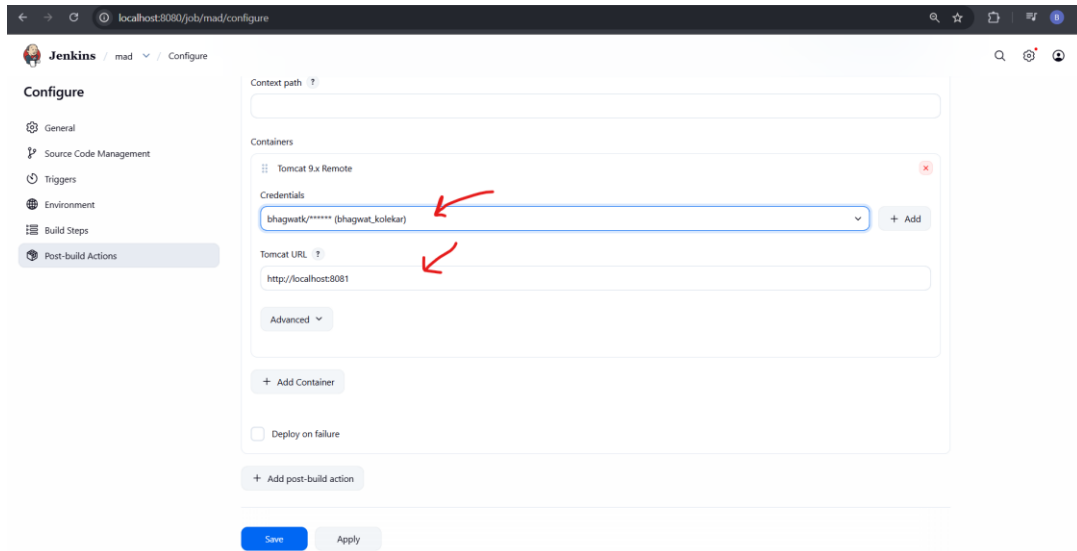
Click on “?” to get war/ ear files, copy text in highlighted blue i.e (****/*.war**) and paste it in war/ear files section

Step 10: Add Containers



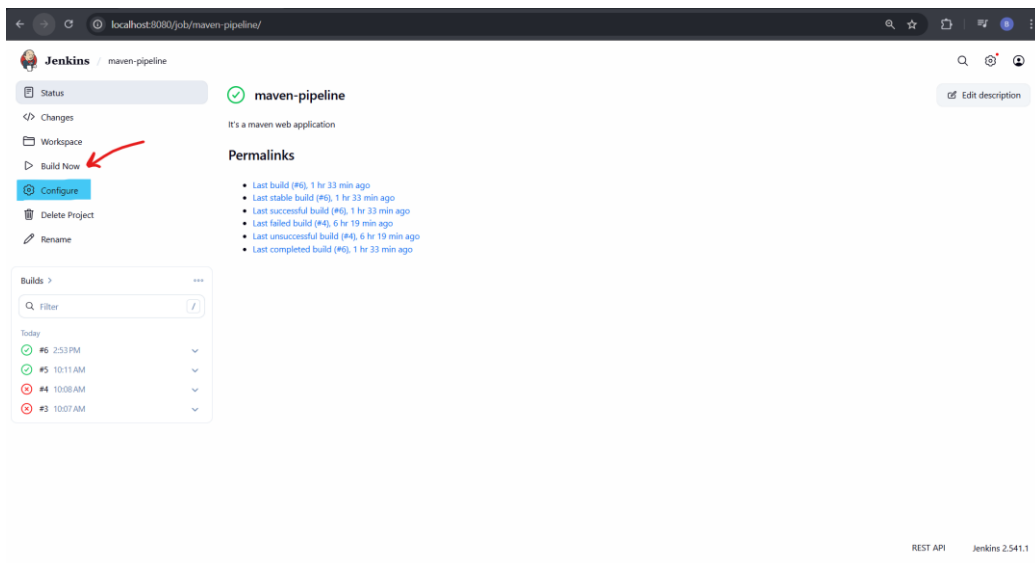
Add Container as Tomcat version which you have downloaded.

Step 11: Add Tomcat Credentials



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

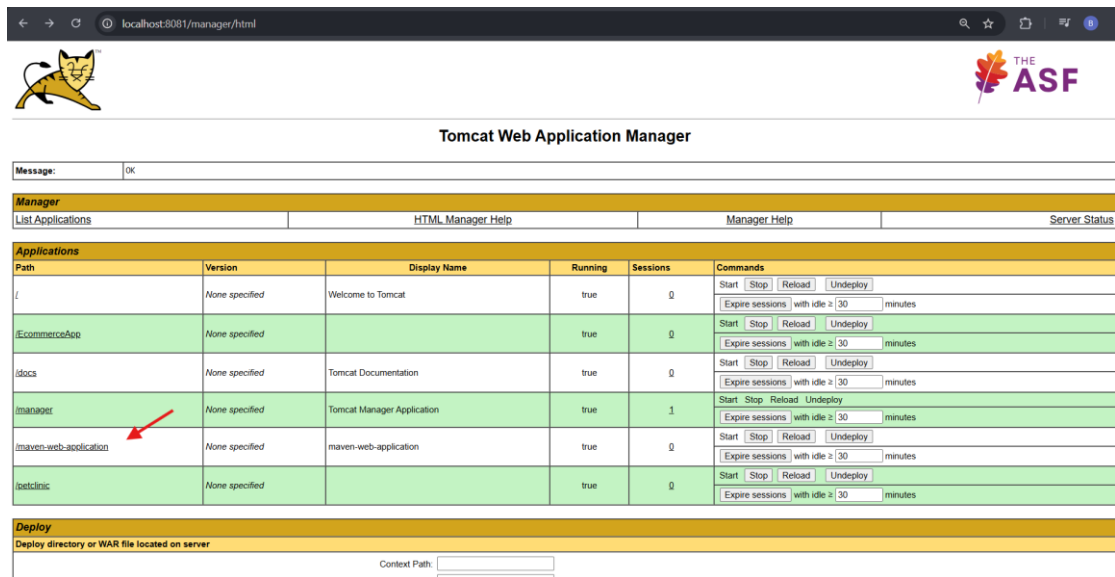
Step 12: Configure and Build



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



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

Deploy
Deploy directory or WAR file located on server

Context Path:

This is the Home Page of Tomcat Web Application Manager, Under Applications Check for “**App name**”
then click on it

Step 14: Output

