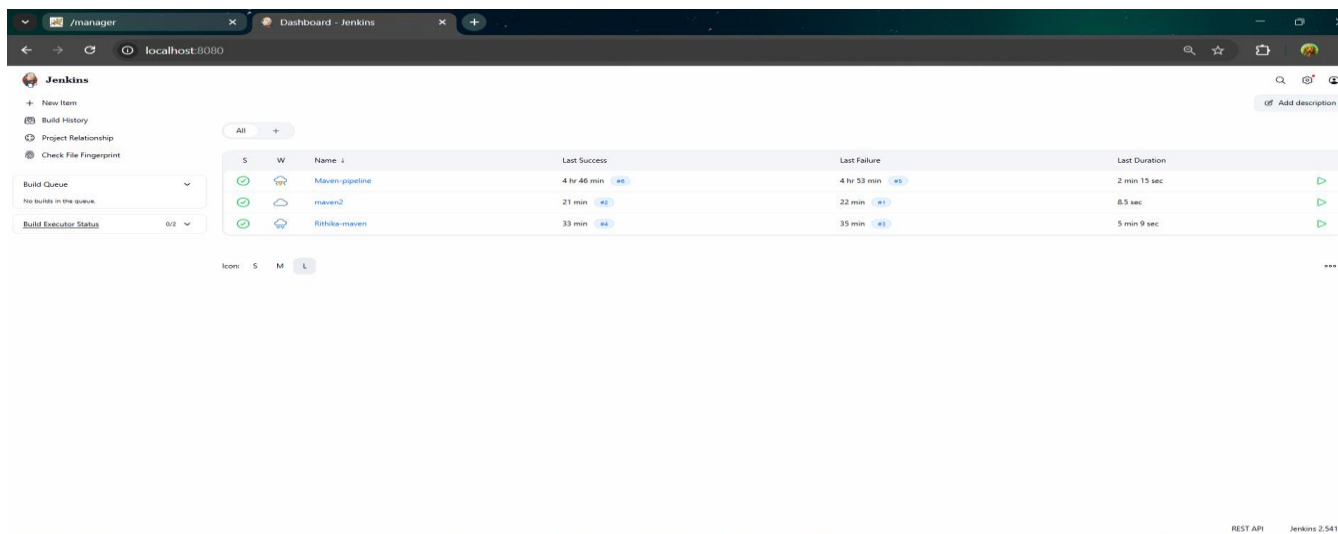
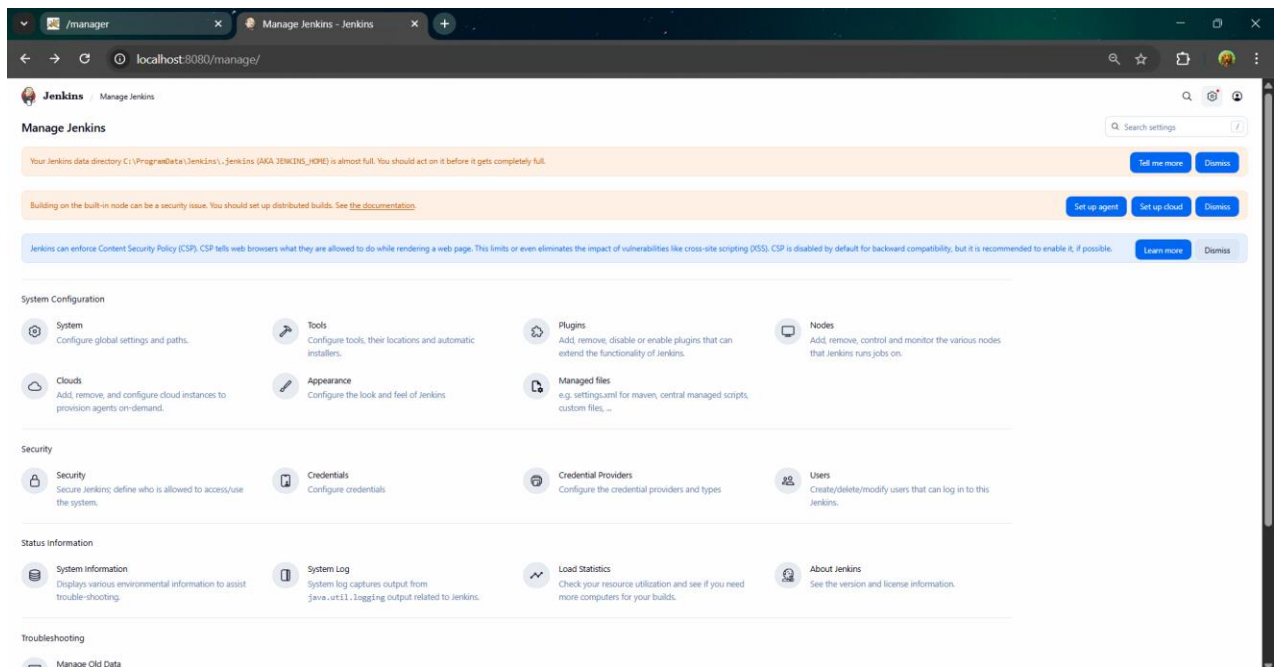


JENKINS PIPELINE

1) Open the Jenkins



2) Goto manage Jenkins:



3) Goto Plugins and search Deploy to container and install it :

Jenkins / Manage Jenkins / Plugins

Search: deploy

Install | Available plugins | Installed plugins | Advanced settings | Download progress

Install	Name	Released	Health
<input type="checkbox"/>	Deploy to container 1.17 Artifact Uploaders This plugin allows you to deploy a war to a container after a successful build. Glassfish 3.x remote deployment	8 mo 19 days ago	87
<input type="checkbox"/>	Docker Pipeline 634.vedc7242b_ed4_7 pipeline DevOps Deployment docker Build and use Docker containers from pipelines. This plugin is up for adoption! We are looking for new maintainers. Visit our Adopt a Plugin initiative for more information.	3 mo 16 days ago	76
<input type="checkbox"/>	Artifactory 4.0.8 pipeline This plugin allows your build jobs to deploy artifacts and resolve dependencies to and from Artifactory, and then have them linked to the build job that created them. The plugin includes a vast collection of features, including a rich pipeline API library and release management for Maven and Gradle builds with Staging and Promotion.	1 yr 6 mo ago	82
<input type="checkbox"/>	Ansible 635.v34b_48e979c86 pipeline External Site/Tool Integrations DevOps Build Tools Deployment	14 days ago	100

4) Now go to the system

Jenkins / Manage Jenkins

Search settings

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

Jenkins can enforce Content Security Policy (CSP). CSP tells web browsers what they are allowed to do while rendering a web page. This limits or even eliminates the impact of vulnerabilities like cross-site scripting (XSS). CSP is disabled by default for backward compatibility, but it is recommended to enable it, if possible. [Learn more](#) [Dis](#)

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 be used by your build jobs.
- Nodes**
Add, remove, control and monitor the various nodes that your build jobs can run on.

and choose Global credentials and add some credentials.

Jenkins / Manage Jenkins / Credentials / System / Global credentials (unrestr...)

New credentials

Kind: Username with password

Scope: Global (Jenkins, nodes, items, all child items, etc)

Username:

Blank username; did you mean to use secret text credentials instead?

☐ Treat username as secret

Password:

ID:

Description:

Create

Then will create a new credentials such as **username** , **password** , **ID** , **Description** and **Click create**.

- 5) Now goto the home page and click **New item** , enter the item name and choose the **Freestyle Project** and click OK.

Jenkins / All / New Item

New Item

Enter an item name:

Select an item type:

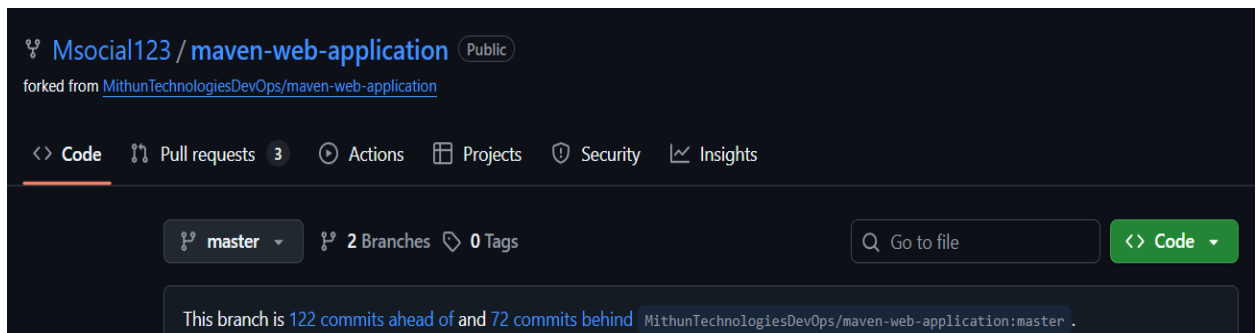
- Pipeline: Build, test, and deploy using pipelines. Supports stages, parallel work, and running on multiple agents.
- Freestyle project**: Classic, general purpose job type that checks out from up to one SCM, executes build steps serially, followed by post-build steps like archiving artifacts and sending 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.
- 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:

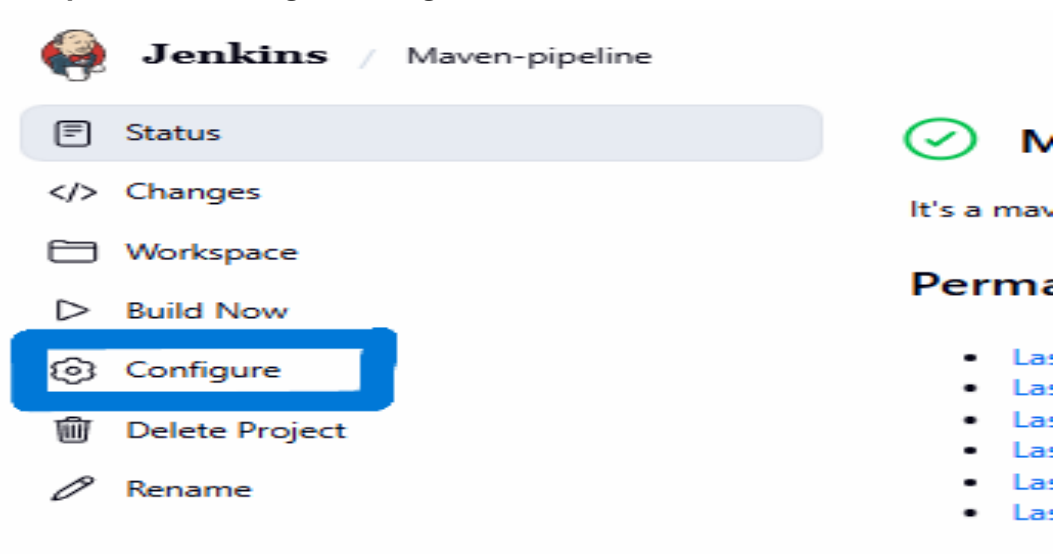
Copy from:

OK

- 6) Goto the **github** and choose the **Repository** and copy the code.



7) Now **open the Item** , goto configure



and add description

Jenkins / Maven-pipeline / Configure

Configure

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

General

Description

It's a maven application

Plain text [Preview](#)

☐ Discard old builds ?

☐ GitHub project

☐ This project is parameterized ?

☐ Throttle builds ?

☐ Execute concurrent builds if necessary ?

[Advanced](#) ▾

Source Code Management

Connect and manage your code repository to automatically pull the latest code for your builds.

☐ None

☒ Git ?

[Repositories](#) ?

8) Goto the Source code management: paste the repo url and credential should be none

Jenkins / Maven-pipeline / Configure

Configure

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

Source Code Management

Connect and manage your code repository to automatically pull the latest code for your builds.

☐ None

☒ Git ?

[Repositories](#) ?

Repository URL ?

<https://github.com/Msocial123/maven-web-application.git>

Credential ?

- none -

[Advanced](#) ▾

[+ Add Repository](#)

[Branches to build](#) ?

Branch Specifier (blank for 'any') ?

*/master

[+ Add Branch](#)

Repository browser ?

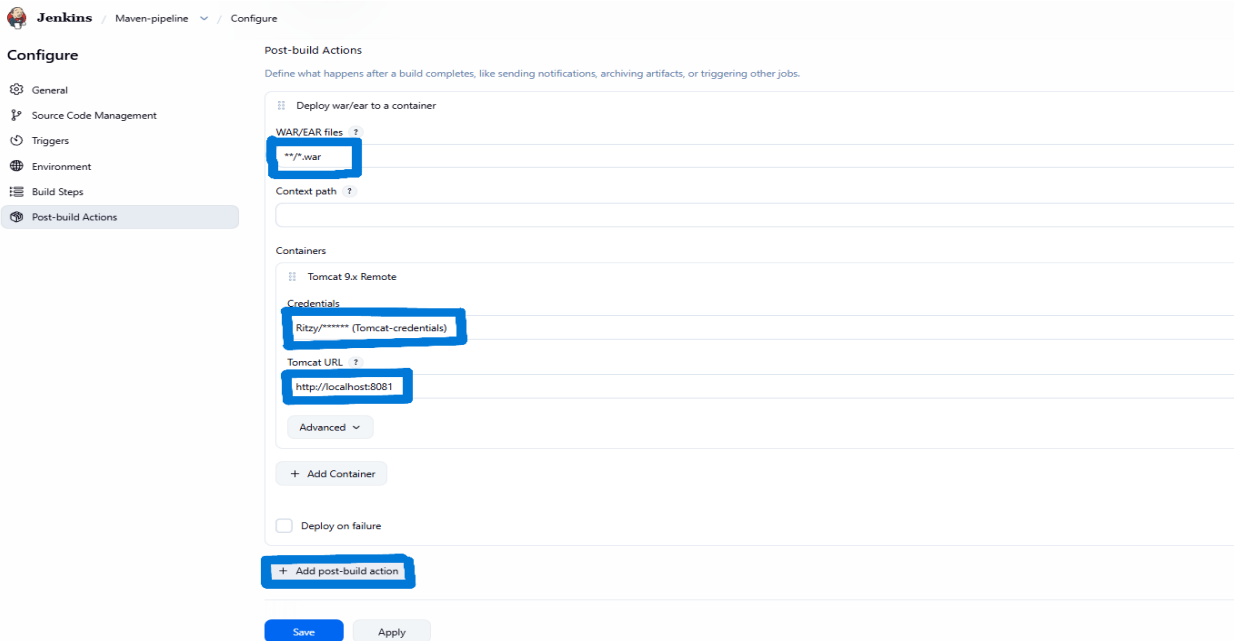
(Auto)

Additional Behaviours

[+ Add](#)

And to check the branches to build should be as in the repository whether it is main or master in the left side. After that in build steps type **clean package** in **Goals**.

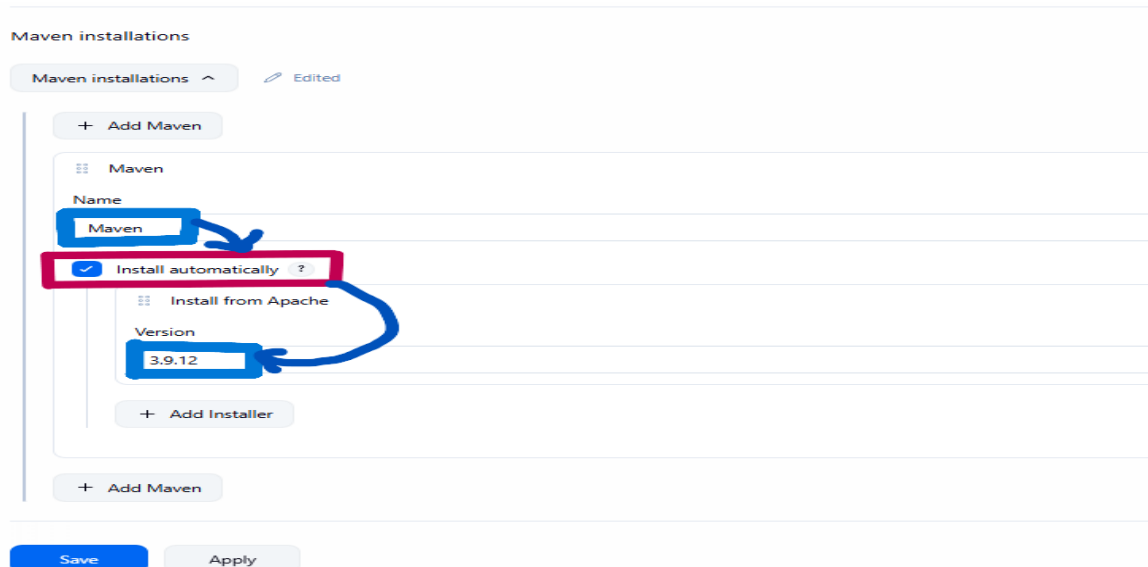
- 9) Goto the post build Actions and give war by clicking ? And choose the credentials and our tomcat url and click Apply and save.



The screenshot shows the Jenkins 'Configure' page for a 'Maven-pipeline'. The 'Post-build Actions' tab is selected in the left sidebar. The main configuration area is titled 'Post-build Actions' and includes a description: 'Define what happens after a build completes, like sending notifications, archiving artifacts, or triggering other jobs.' A single action, 'Deploy war/ear to a container', is configured. The 'WAR/EAR files' field is set to '**/*.war'. The 'Context path' field is empty. Under the 'Containers' section, 'Tomcat 9.x Remote' is selected. The 'Credentials' field is set to 'Ritzzy/***** (Tomcat-credentials)'. The 'Tomcat URL' field is set to 'http://localhost:8081'. The 'Advanced' dropdown is expanded. At the bottom, there is a '+ Add post-build action' button, a 'Save' button, and an 'Apply' button.

- 10) Goto setting and click plugins ==> available plugins and search **Maven integration, config file provider, Pipeline Maven Plugin API, Pipeline Maven Integration.**

- 11) Goto **Tools** and scrool down to see maven installation.



The screenshot shows the Jenkins 'Maven installations' page. The 'Maven installations' tab is selected in the left sidebar. The main configuration area is titled 'Maven installations' and includes a description: 'Maven installations'. A single installation, 'Maven', is configured. The 'Name' field is set to 'Maven'. The 'Install automatically' checkbox is checked. The 'Install from Apache' section is expanded, and the 'Version' field is set to '3.9.12'. At the bottom, there is a '+ Add Maven' button, a 'Save' button, and an 'Apply' button.

Here give the name as maven and default it shows the version .Click **apply and save.**

12) Here in configure , can see in the environment , it will show the maven version and have to give goals as **clean package** in the earlier when the starting process in Build steps.

The screenshot shows the Jenkins configuration page for a project named 'Maven-pipeline'. The left sidebar contains a 'Configure' section with sub-items: General, Source Code Management, Triggers, Environment (highlighted with a blue box), Build Steps, and Post-build Actions. A blue arrow points from the 'Environment' tab to the 'Maven Version' field in the 'Build Steps' section. The 'Maven Version' field contains the text 'Maven'. Below it, the 'Goals' field contains the text 'clean package'. Other fields in the 'Build Steps' section include 'Invoke top-level Maven targets' and an 'Advanced' dropdown menu. A '+ Add build step' button is at the bottom.

12)Now click Build run


The screenshot shows the Jenkins overview page for the 'Maven-pipeline' project. The left sidebar contains a 'Status' section with sub-items: Changes, Workspace, Build Now (highlighted with a blue box), Configure, Delete Project, and Rename. The main content area shows a green checkmark icon next to the project name 'Maven-pipeline'. Below the project name, it says 'It's a maven application'. There is a 'Permalinks' section with a list of build links: Last build (#6), 6 hr 23 min ago; Last stable build (#6), 6 hr 23 min ago; Last successful build (#6), 6 hr 23 min ago; Last failed build (#5), 6 hr 30 min ago; Last unsuccessful build (#5), 6 hr 30 min ago; and Last completed build (#6), 6 hr 23 min ago.

13) It will get success and can see in the tomcat apache.


▼

/manager

×

 Maven-pipeline Config - Jenkins

×

 Document 6.docx

×

←

→

↻

ⓘ

localhost:8081/manager/html



Tomcat Web A

Message:	OK
----------	----

Manager	
List Applications	HTML Manager Help

Applications		
Path	Version	Display Name
/	None specified	Welcome to Tomcat
/docs	None specified	Tomcat Documentation
/manager	None specified	Tomcat Manager Application
/maven-web-application	None specified	maven-web-application
/petclinic	None specified	