

# A.Bhavagna-23BD1A050C

## Week 8: Jenkins Automation

### **Task -1 MAVEN INSTALLATION**

**Step1:** Prerequisites (java version)

```
Microsoft Windows [Version 10.0.26100.6584]
(c) Microsoft Corporation. All rights reserved.

C:\Users\andam>java --version
java 17.0.12 2024-07-16 LTS
Java(TM) SE Runtime Environment (build 17.0.12+8-LTS-286)
Java HotSpot(TM) 64-Bit Server VM (build 17.0.12+8-LTS-286, mixed mode, sharing)

C:\Users\andam>
```

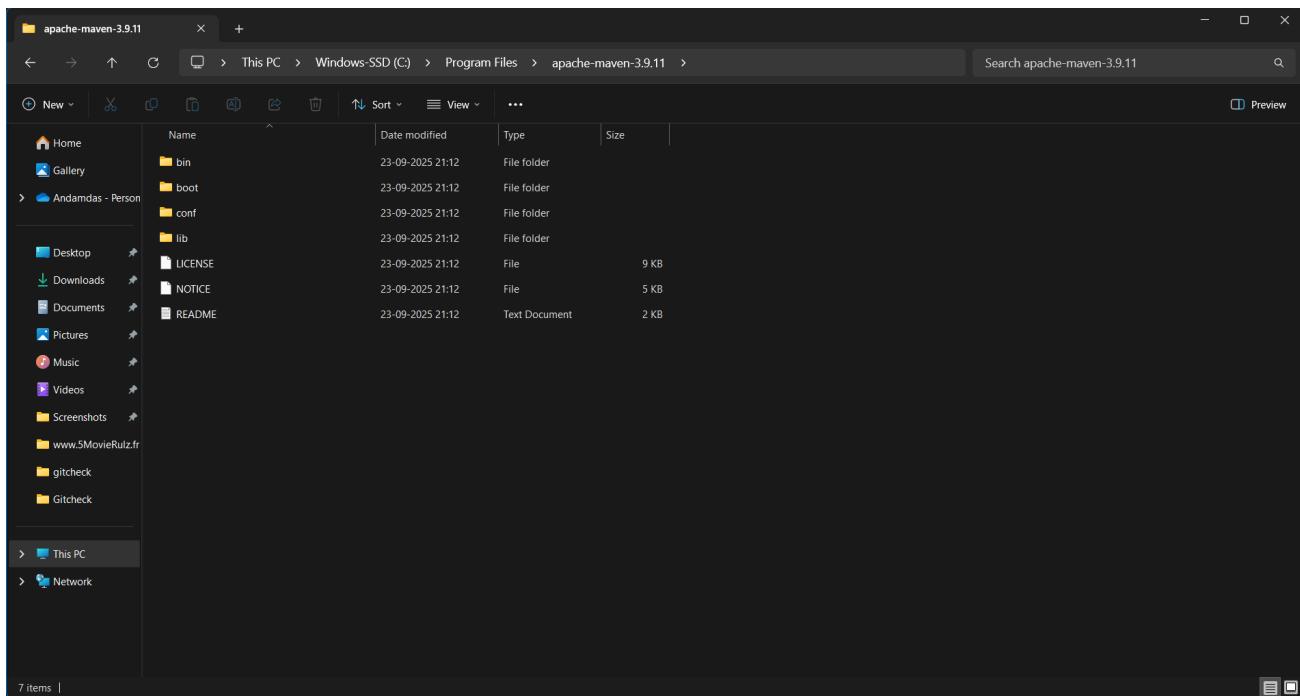
**Step 2:** Download Maven

Version 3.9.11

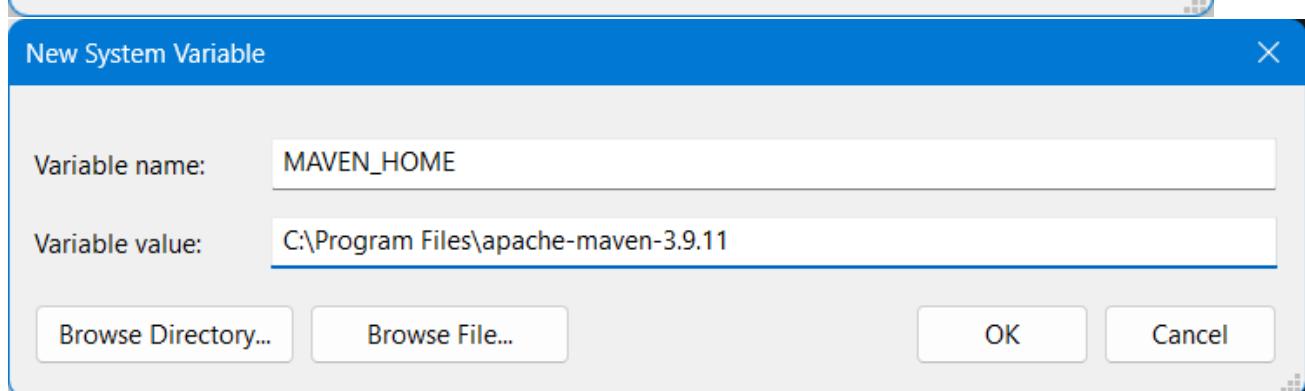
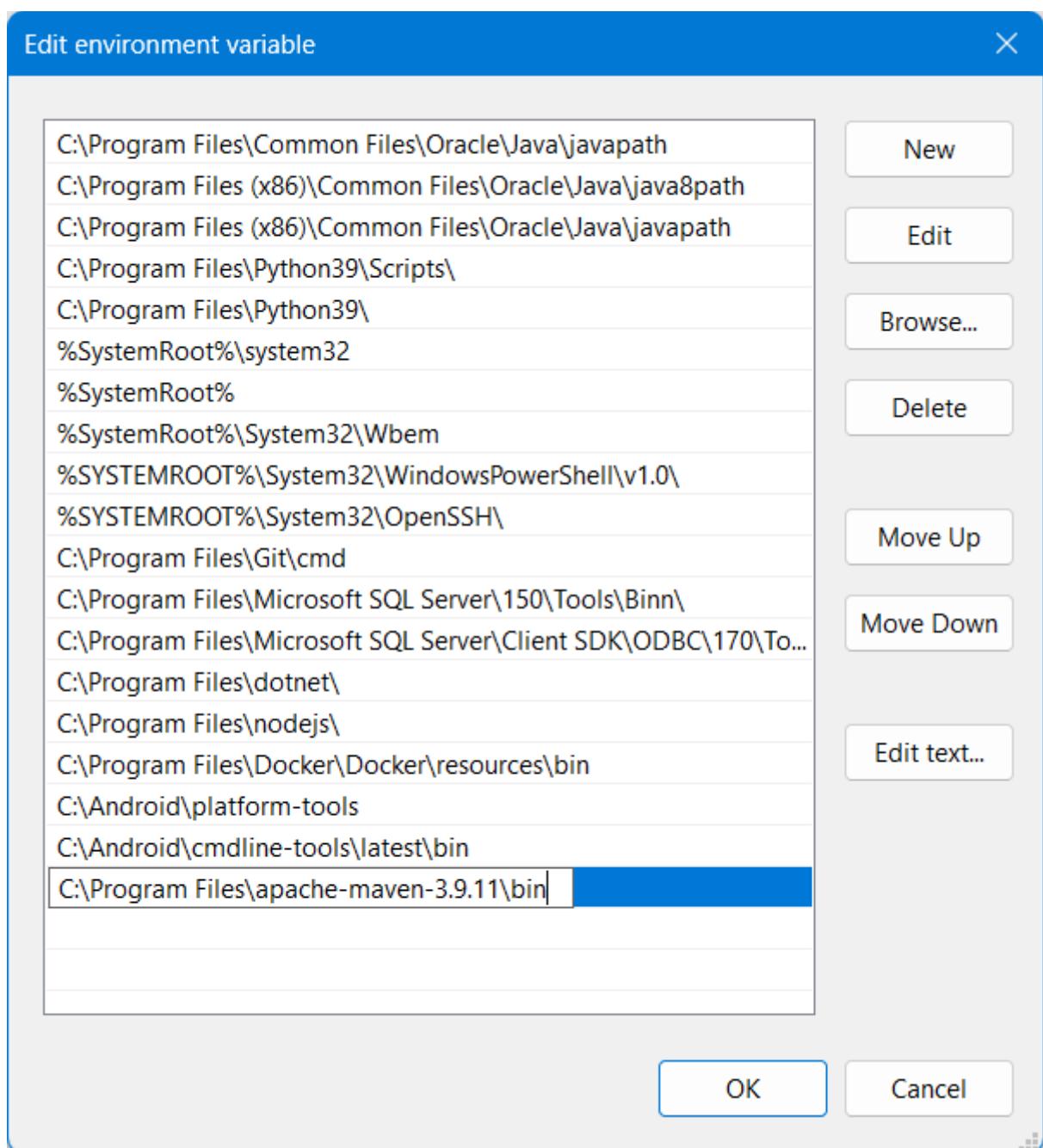
The screenshot shows a web browser window with the URL [maven.apache.org/download.cgi](https://maven.apache.org/download.cgi). The page is titled "Download Apache Maven – Maven 3.9.11". On the left, there's a sidebar with links like "What is Maven?", "Installation", "Downloads" (which is highlighted), "Use", "Run", "Configure", "Release Notes", "DOCUMENTATION", "Maven Plugins", "Maven Extensions", "Maven Tools" (with sub-links for "Maven Daemon" and "Maven Upgrade Tool"), "Index (category)", "User Centre", "Plugin Developer Centre", "Maven Repository Centre", "Maven Developer Centre", "Books and Resources", "Security", "COMMUNITY", "Community Overview", "Project Roles", "How to Contribute", "Getting Help", "Issue Management", "Getting Maven Source", and "The Maven Team". The main content area has two sections: "System Requirements" and "Files". The "System Requirements" section lists Java Development Kit (JDK) requirements (Maven 3.9+ requires JDK 8 or above), memory requirements (no minimum), disk requirements (approx 10MB), and operating system requirements (no minimum). The "Files" section shows a table of download links for Maven 3.9.11. The table has columns for "Link", "Checksums", and "Signature". It includes four rows: Binary tar.gz archive (link: apache-maven-3.9.11-bin.tar.gz, checksums: apache-maven-3.9.11-bin.tar.gz.sha512, signature: apache-maven-3.9.11-bin.tar.gz.asc), Binary zip archive (link: apache-maven-3.9.11-bin.zip, checksums: apache-maven-3.9.11-bin.zip.sha512, signature: apache-maven-3.9.11-bin.zip.asc), Source tar.gz archive (link: apache-maven-3.9.11-src.tar.gz, checksums: apache-maven-3.9.11-src.tar.gz.sha512, signature: apache-maven-3.9.11-src.tar.gz.asc), and Source zip archive (link: apache-maven-3.9.11-src.zip, checksums: apache-maven-3.9.11-src.zip.sha512, signature: apache-maven-3.9.11-src.zip.asc). Below the table, there's a note about release notes, source code, and distribution under the Apache License.

	Link	Checksums	Signature
Binary tar.gz archive	<a href="#">apache-maven-3.9.11-bin.tar.gz</a>	apache-maven-3.9.11-bin.tar.gz.sha512	apache-maven-3.9.11-bin.tar.gz.asc
Binary zip archive	<a href="#">apache-maven-3.9.11-bin.zip</a>	apache-maven-3.9.11-bin.zip.sha512	apache-maven-3.9.11-bin.zip.asc
Source tar.gz archive	<a href="#">apache-maven-3.9.11-src.tar.gz</a>	apache-maven-3.9.11-src.tar.gz.sha512	apache-maven-3.9.11-src.tar.gz.asc
Source zip archive	<a href="#">apache-maven-3.9.11-src.zip</a>	apache-maven-3.9.11-src.zip.sha512	apache-maven-3.9.11-src.zip.asc

**Step 3:** Extract Maven



#### Step 4: Set Environment Variables



**Step 5:** Verify Installation

```
C:\Users\andam>mvn -v
Apache Maven 3.9.11 (3e54c93a704957b63ee3494413a2b544fd3d825b)
Maven home: C:\Program Files\apache-maven-3.9.11
Java version: 17.0.12, vendor: Oracle Corporation, runtime: C:\Program Files\Java\jdk-17
Default locale: en_IN, platform encoding: Cp1252
OS name: "windows 11", version: "10.0", arch: "amd64", family: "windows"

C:\Users\andam>
```

## Task-2 JENKINS INSTALLATION

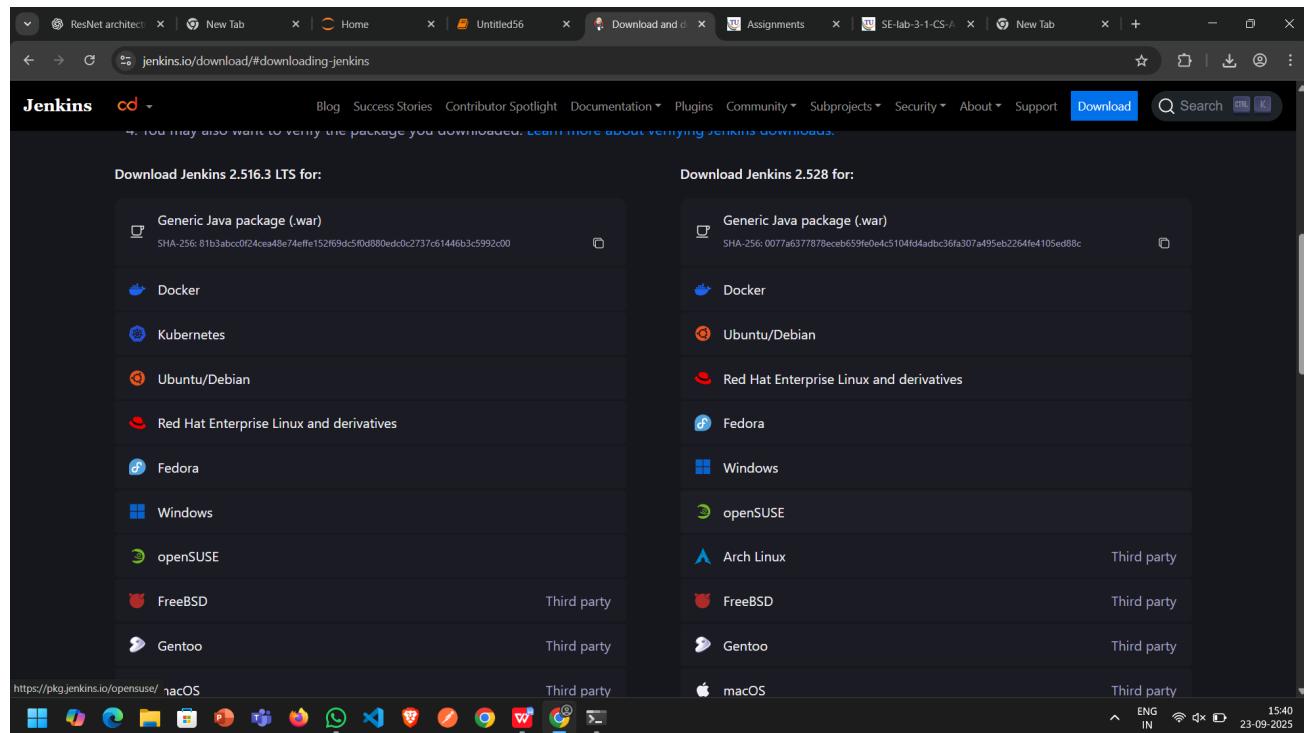
### Step 1: Prerequisites

```
Microsoft Windows [Version 10.0.26100.6584]
(c) Microsoft Corporation. All rights reserved.

C:\Users\andam>java --version
java 17.0.12 2024-07-16 LTS
Java(TM) SE Runtime Environment (build 17.0.12+8-LTS-286)
Java HotSpot(TM) 64-Bit Server VM (build 17.0.12+8-LTS-286, mixed mode, sharing)

C:\Users\andam>
```

### Step 2: Download Jenkins



The screenshot shows a dark-themed web browser window with multiple tabs open. The active tab displays the Jenkins download page at <https://jenkins.io/download/#downloading-jenkins>. The page lists two main sections: 'Download Jenkins 2.516.3 LTS for:' and 'Download Jenkins 2.528 for:'. Both sections offer a 'Generic Java package (.war)' option. Other options listed include Docker, Kubernetes, Ubuntu/Debian, Red Hat Enterprise Linux and derivatives, Fedora, Windows, openSUSE, Arch Linux, FreeBSD, Gentoo, and macOS. The Jenkins logo is visible in the top left corner of the page.

### Step 3: Installation



**Step 4: Set User Details**

## Getting Started

# Create First Admin User

Username



Password

Confirm password

Full name

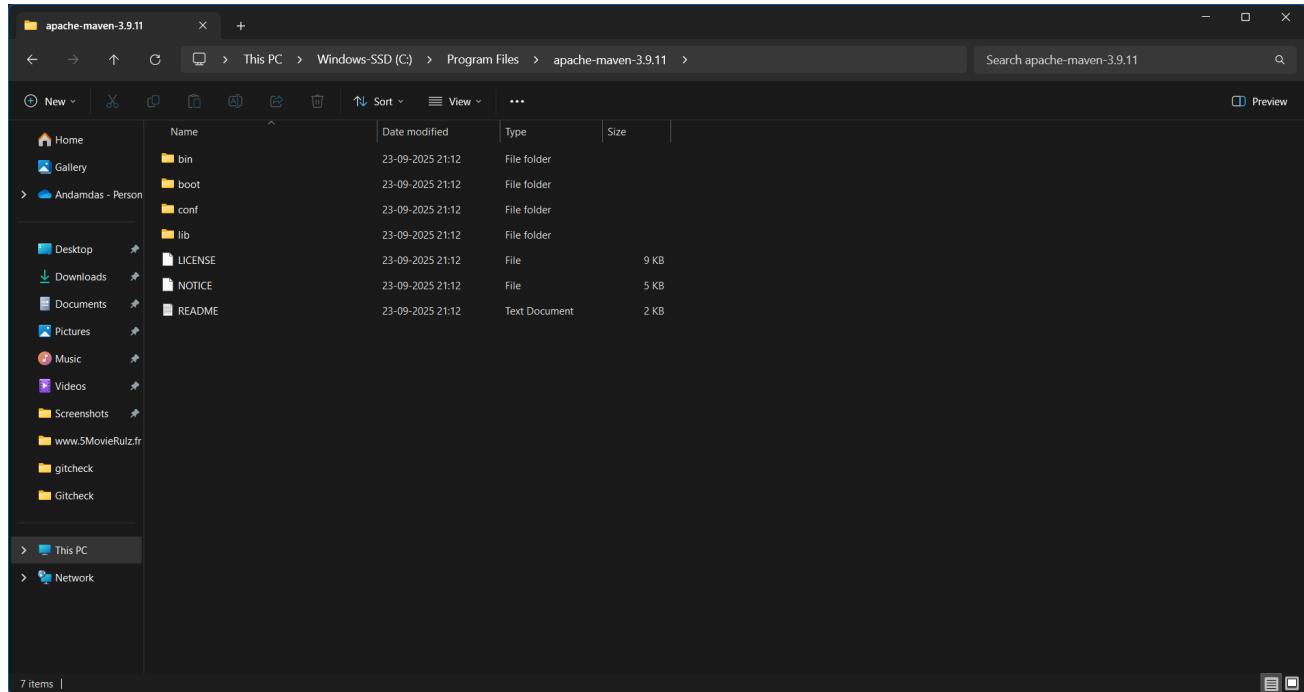
E-mail address

## Step 5: Verify Installation

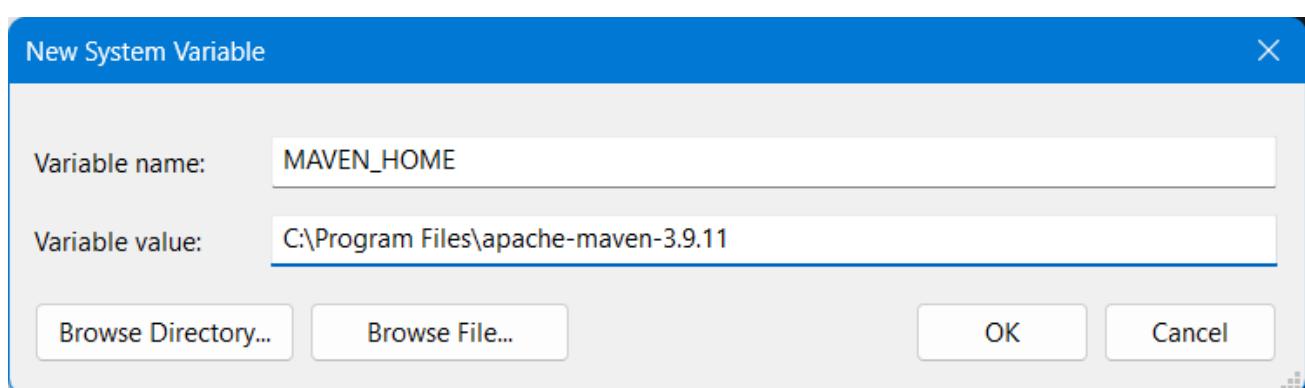
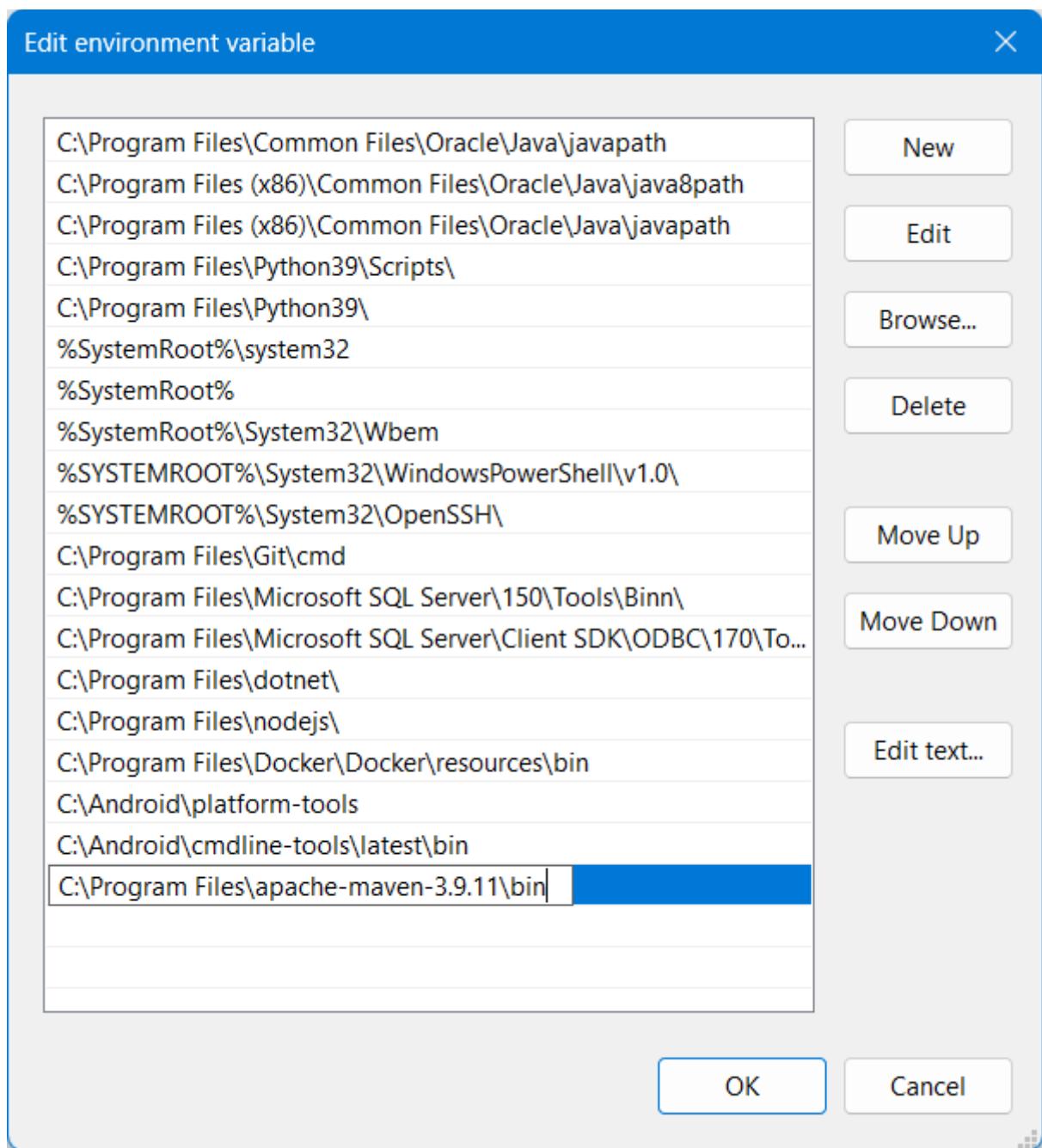
The screenshot shows the Jenkins dashboard at [localhost:8081/view/all/](http://localhost:8081/view/all/). The page title is "Jenkins". The main content area starts with a "Welcome to Jenkins!" message. Below it is a "Build Queue" section with a dropdown menu set to "Build Queue" and a note "No builds in the queue.". There is also a "Build Executor Status" section showing "0/2". A prominent "Create a job" button is located in the center. To the right of the "Create a job" button is a "+" sign. Below these sections are three buttons for "Set up a distributed build": "Set up an agent" (with a monitor icon), "Configure a cloud" (with a cloud icon), and "Learn more about distributed builds" (with a question mark icon). At the bottom of the dashboard, there are links for "REST API" and "Jenkins 2.516.3". The browser's status bar at the bottom right shows "ENG IN" and the date "23-09-2025".

## Task-3 Add Maven to Jenkins

### Step 1: Copy bin path



### Step 2: Set Environment Variables



**Step 3:** Set maven in Jenkins Configure Tools

The screenshot shows the Jenkins interface for configuring tools. The URL in the browser is `localhost:8081/configureTools/`. The main section is titled "Maven". Under "Name", the value "MAVEN\_HOME" is entered. A checked checkbox labeled "Install automatically" is present. Below it, under "Install from Apache", the "Version" dropdown is set to "3.9.11". There is also a "Add Installer" button. At the bottom of the configuration panel are "Save" and "Apply" buttons.

Apply and Save

# Task-4 Maven java Automation

## I. Maven Java Automation Steps:

Step 1: Open Jenkins (localhost:8081)

The screenshot shows the Jenkins dashboard at [localhost:8081](http://localhost:8081). The top navigation bar includes tabs for ResNet architecture, 1. Jenkins Tutorials, Home, Untitled5, Assignments, SE-lab-3-1-CS-A-B-F, and Dashboard - Jenkins. The main content area features a "Welcome to Jenkins!" message: "This page is where your Jenkins jobs will be displayed. To get started, you can set up distributed builds or start building a software project." Below this are sections for "Start building your software project" (with a "Create a job" button) and "Set up a distributed build" (with links to "Set up an agent", "Configure a cloud", and "Learn more about distributed builds"). On the left, there's a sidebar with "New Item", "Build History", and "Build Queue" (empty). At the bottom, the status bar shows various icons and the text "REST API Jenkins 2.516.3".

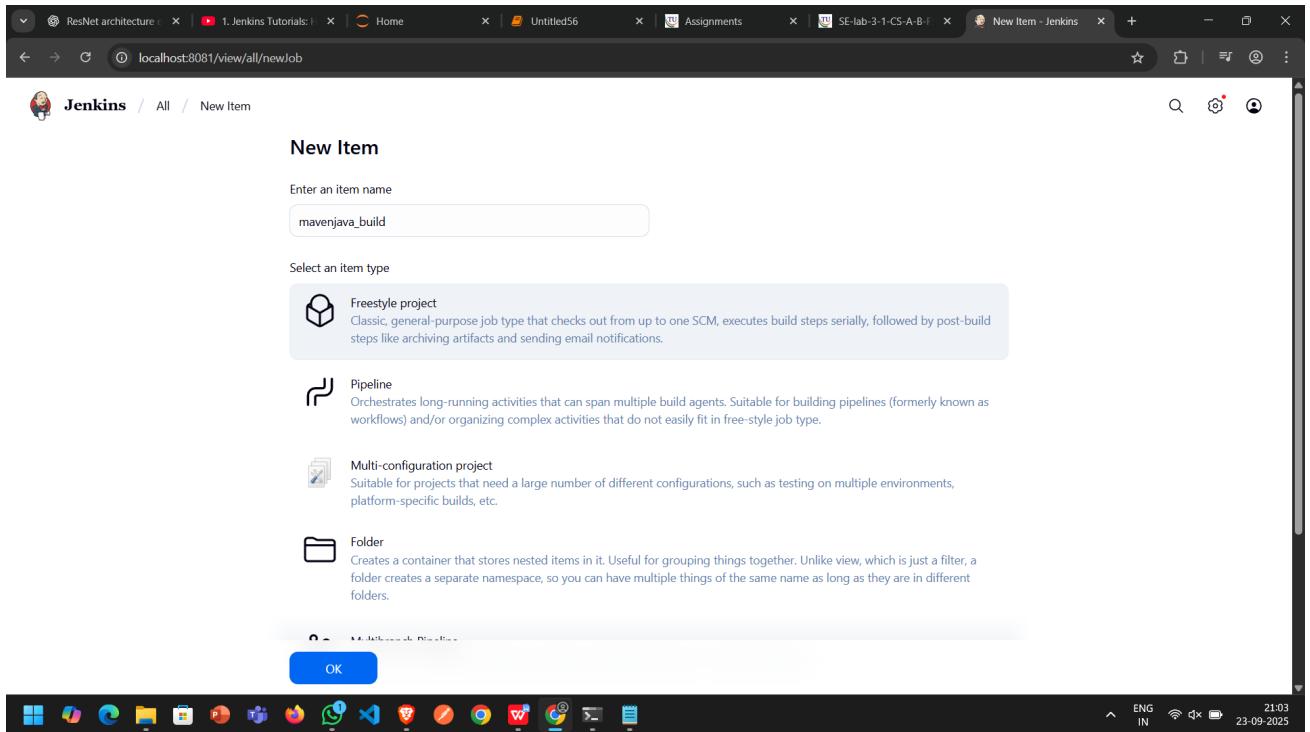
Click on "New Item" (left side menu )

The screenshot shows the "New Item" dialog in Jenkins. The title bar says "New Item". It has a search bar and a "New Item" link. The main area is titled "Enter an item name" with an input field containing a placeholder. Below it, "Select an item type" is shown with several options: "Freestyle project" (selected), "Pipeline", "Multi-configuration project", "Folder", "Multibranch Pipeline", and "Organization Folder". Each option has a small icon and a brief description. A blue "OK" button is at the bottom.

Step 2: Create Freestyle Project (e.g., MavenJava\_Build)

Enter project name (e.g., MavenJava\_Build)

Click "OK"



Configure the project:

Description: "Java Build demo"

The screenshot shows the Jenkins 'Configure' page for the 'mavenjava\_build' job. The left sidebar has tabs for 'General', 'Source Code Management', 'Triggers', 'Environment', 'Build Steps', and 'Post-build Actions'. The 'General' tab is selected. It contains a 'Description' field with the value 'java build demo'. Below the description are several checkboxes: 'Discard old builds', 'GitHub project', 'This project is parameterized', 'Throttle builds', and 'Execute concurrent builds if necessary'. There is also an 'Advanced' dropdown. At the bottom of the configuration section is a 'Save' button. The status bar at the bottom right shows 'ENG IN' and the date '23-09-2025'.

Source Code Management: Git repository URL: [https://github.com/bhavagna06/Maven.git]

Code Issues Pull requests Actions Projects Security Insights Settings

Maven Public

main 1 Branch 0 Tags

Go to file Add file Code

bhavagna06 maven commit e6db277 - last month 1 Commit

.settings maven commit last month

src maven commit last month

target maven commit last month

.classpath maven commit last month

.project maven commit last month

pom.xml maven commit last month

README

Add a README

About

No description, website, or topics provided.

Activity

0 stars 0 watching 0 forks

Releases

No releases published Create a new release

Packages

No packages published Publish your first package

Languages

Java 100.0%

Branches to build: \*/Main or \*/master

ResNet architect 1. Jenkins Tutor Home Untitled56 Assignments SE-lab-3-1-CS-A mavenjava\_build bhavagna06/Maven

Jenkins / mavenjava\_build / Configuration

Configure

Source Code Management

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

General

Source Code Management (selected)

Triggers

Environment

Build Steps

Post-build Actions

None

Git

Repositories

Repository URL https://github.com/bhavagna06/Maven.git

Credentials

- none -

+ Add

Advanced

Add Repository

Branches to build

Branch Specifier (blank for 'any')

Save Apply

Build Steps: Add Build Step -> "Invoke top-level Maven targets"

Maven version: MAVEN\_HOME

Goals: clean

Add Build Step -> "Invoke top-level Maven targets"

Maven version: MAVEN\_HOME

Goals: install

The screenshot shows the Jenkins configuration interface for a job named 'mavenjava\_build'. The 'Build Steps' section is selected in the sidebar. Two 'Invoke top-level Maven targets' steps are present. The first step has 'Goals' set to 'clean'. The second step, which is a duplicate, has 'Goals' set to 'install'. Both steps have 'Maven Version' set to 'MAVEN\_HOME'. At the bottom of the configuration page are 'Save' and 'Apply' buttons.

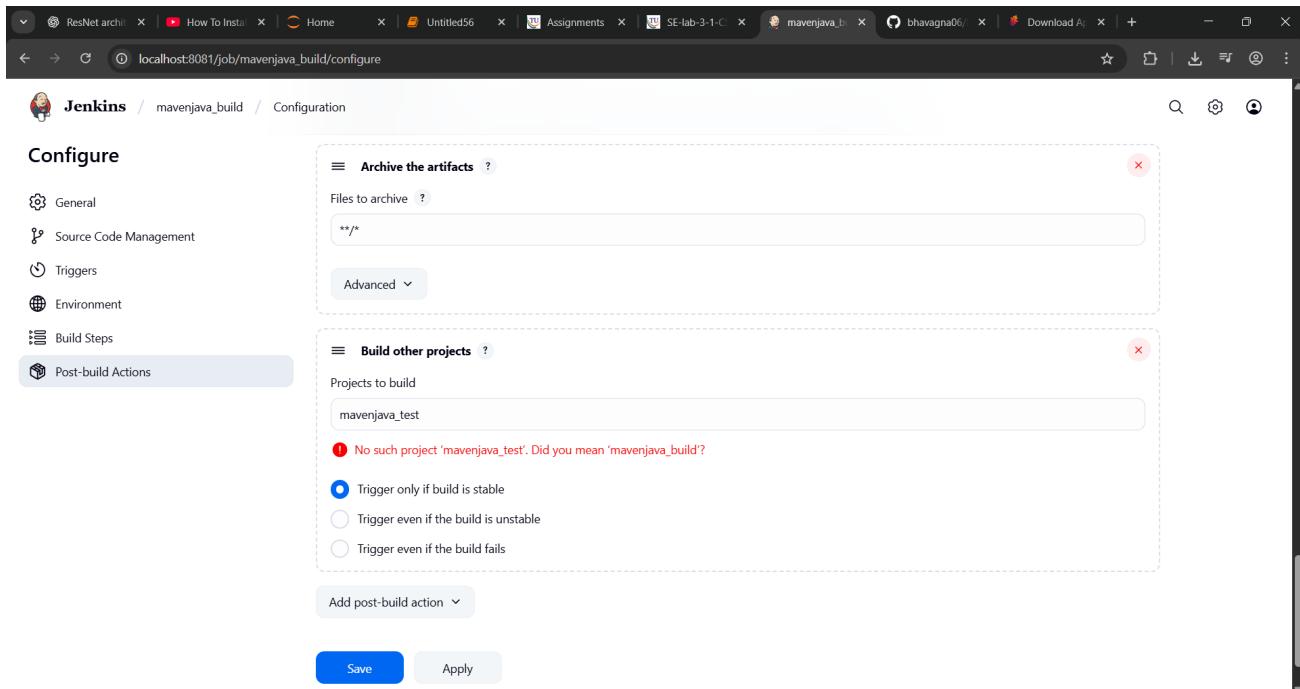
Post-build Actions: Add Post Build Action -> "Archive the artifacts"

Files to archive: \*\*/\*

Add Post Build Action -> "Build other projects"

Projects to build: MavenJava\_Test

Trigger: Only if build is stable

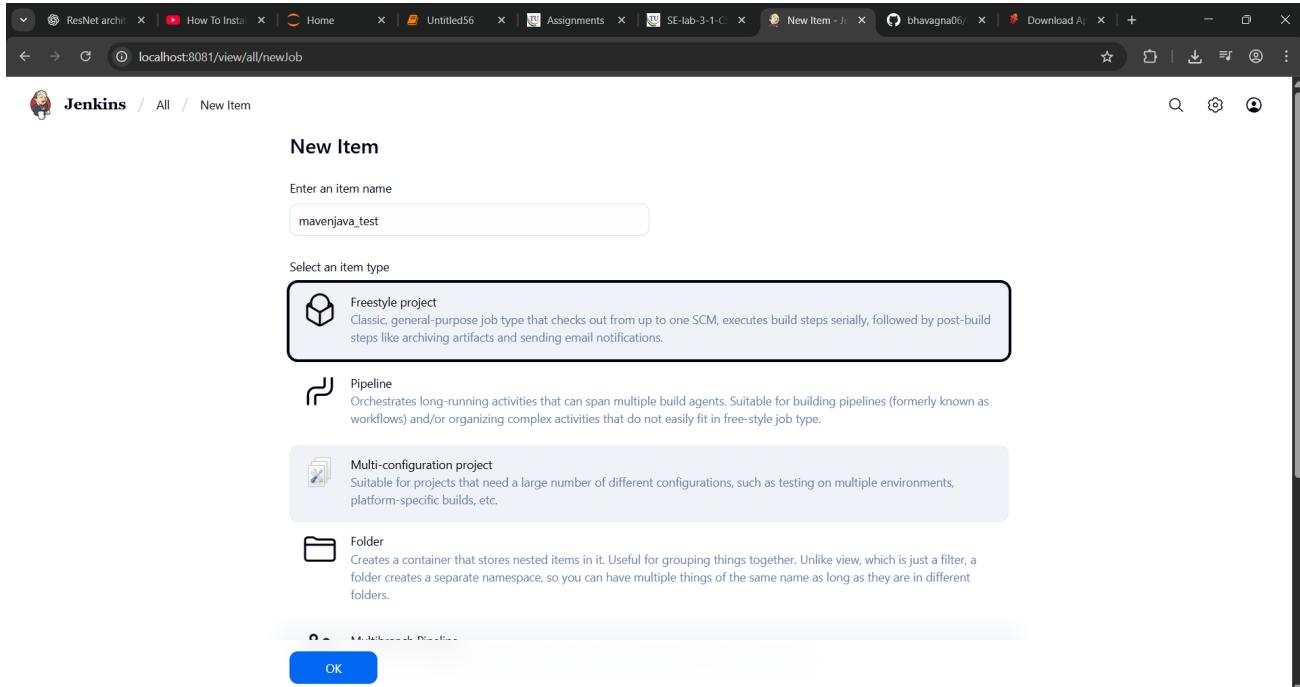


The screenshot shows the Jenkins configuration page for the job 'mavenjava\_build'. The left sidebar lists configuration sections: General, Source Code Management, Triggers, Environment, Build Steps, and Post-build Actions. 'Post-build Actions' is currently selected. The main area contains two expandable sections: 'Archive the artifacts' (with a pattern '\*\*/\*') and 'Build other projects' (with a project named 'mavenjava\_test' and a note about it not existing). A 'Trigger only if build is stable' radio button is selected. At the bottom are 'Save' and 'Apply' buttons.

## Apply and Save

### Step 3: Create Freestyle Project (e.g., MavenJava\_Test)

Enter project name (e.g., MavenJava\_Test) — Click "OK"



The screenshot shows the 'New Item' dialog. In the 'Enter an item name' field, 'mavenjava\_test' is typed. Below it, a 'Freestyle project' option is selected, shown in a highlighted box 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.' Other options listed are 'Pipeline' and 'Multi-configuration project'. At the bottom is an 'OK' button.

## Configure the project:

Description: "Test demo"

The screenshot shows the Jenkins configuration interface for a job named 'mavenjava\_test'. The 'General' tab is selected. In the 'Description' field, the text 'maven java test' is entered. Below the description, there are several checkboxes for build triggers: 'Discard old builds', 'GitHub project', 'This project is parameterized', 'Throttle builds', and 'Execute concurrent builds if necessary'. A 'Save' button is visible at the bottom.

## Build Environment:

Check: "Delete the workspace before build starts"

The screenshot shows the Jenkins configuration interface for the 'Triggers' section. The 'Triggers' tab is selected. Under the 'Triggers' heading, it says 'Set up automated actions that start your build based on specific events, like code changes or scheduled times.' There are five checkboxes: 'Trigger builds remotely (e.g., from scripts)', 'Build after other projects are built', 'Build periodically', 'GitHub hook trigger for GITScm polling', and 'Poll SCM'. Below this, under 'Environment', it says 'Configure settings and variables that define the context in which your build runs, like credentials, paths, and global parameters.' A checkbox labeled 'Delete workspace before build starts' is checked. A 'Save' button is visible at the bottom.

Add Build Step -> "Copy artifacts from another project"

Project name: MavenJava\_Build

Build: Stable build only

Artifacts to copy: \*\*/\*

The screenshot shows the Jenkins job configuration page for 'mavenjava\_test'. The 'Build Steps' section is active. A 'Copy artifacts from another project' step is selected, with the 'Project name' set to 'mavenjava\_build' and 'Which build' set to 'Latest successful build'. The 'Artifacts to copy' field contains '\*\*/\*'. The 'Target directory' field is empty. At the bottom are 'Save' and 'Apply' buttons.

Add Build Step -> "Invoke top-level Maven targets"

Maven version: MAVEN\_HOME

Goals: test

The screenshot shows the Jenkins job configuration page for 'mavenjava\_test'. The 'Build Steps' section is active. An 'Invoke top-level Maven targets' step is selected, with 'Maven Version' set to 'MAVEN\_HOME' and 'Goals' set to 'test'. At the bottom are 'Save' and 'Apply' buttons.

Post-build Actions: Add Post Build Action -> "Archive the artifacts"

Files to archive: \*\*/\*

The screenshot shows the Jenkins configuration page for a job named "mavenjava\_test". The "Post-build Actions" section is active, displaying an "Archive the artifacts" step. The "Files to archive" field contains the pattern "\*\*". Below the step, there is an "Advanced" button. At the bottom of the page are "Save" and "Apply" buttons.

## Apply and Save

### Step 4: Create Pipeline View for Maven Java project

Click "+" beside "All" on the dashboard

Enter name: Java\_Pipeline

Select "Build pipeline view"

The screenshot shows the "New view" creation page. The "Name" field is filled with "java\_pipeline". The "Type" section is expanded, showing three options: "Build Pipeline View" (selected), "List View", and "My View". A description for "Build Pipeline View" states: "Shows the jobs in a build pipeline view. The complete pipeline of jobs that a version propagates through are shown as a row in the view." Below the type selection is a "Create" button.

create Pipeline Flow: Layout: Based on upstream/downstream relationship

Initial job: MavenJava\_Build

**Edit View**

Name: java\_pipeline

Description: java pipeline

Build Pipeline View Title:

Build Queue: No builds in the queue.

Build Executor Status: mavenjava\_test #8

Save Apply

Pipeline Flow

Layout: Based on upstream/downstream relationship

This layout mode derives the pipeline structure based on the upstream/downstream trigger relationship between jobs. This is the only out-of-the-box supported layout mode, but is open for extension.

Upstream / downstream config

Select Initial Job: mavenjava\_build

Trigger Options

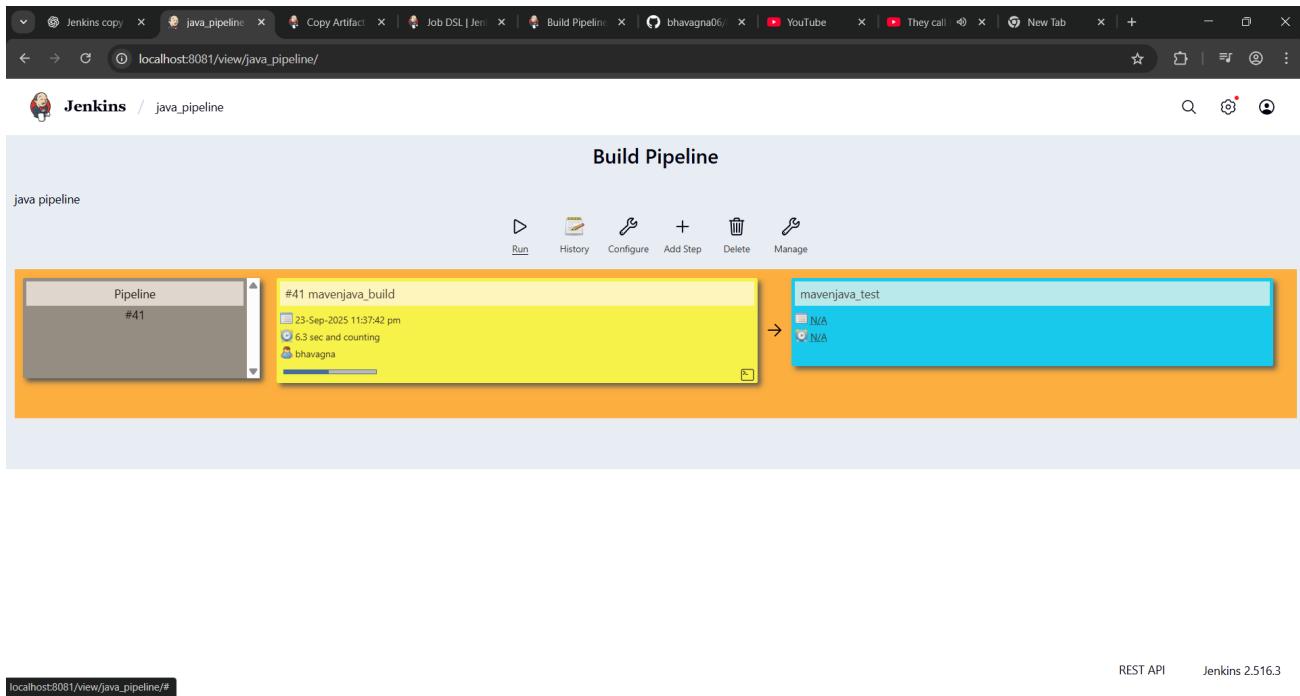
Build Cards: Standard build card

Save Apply

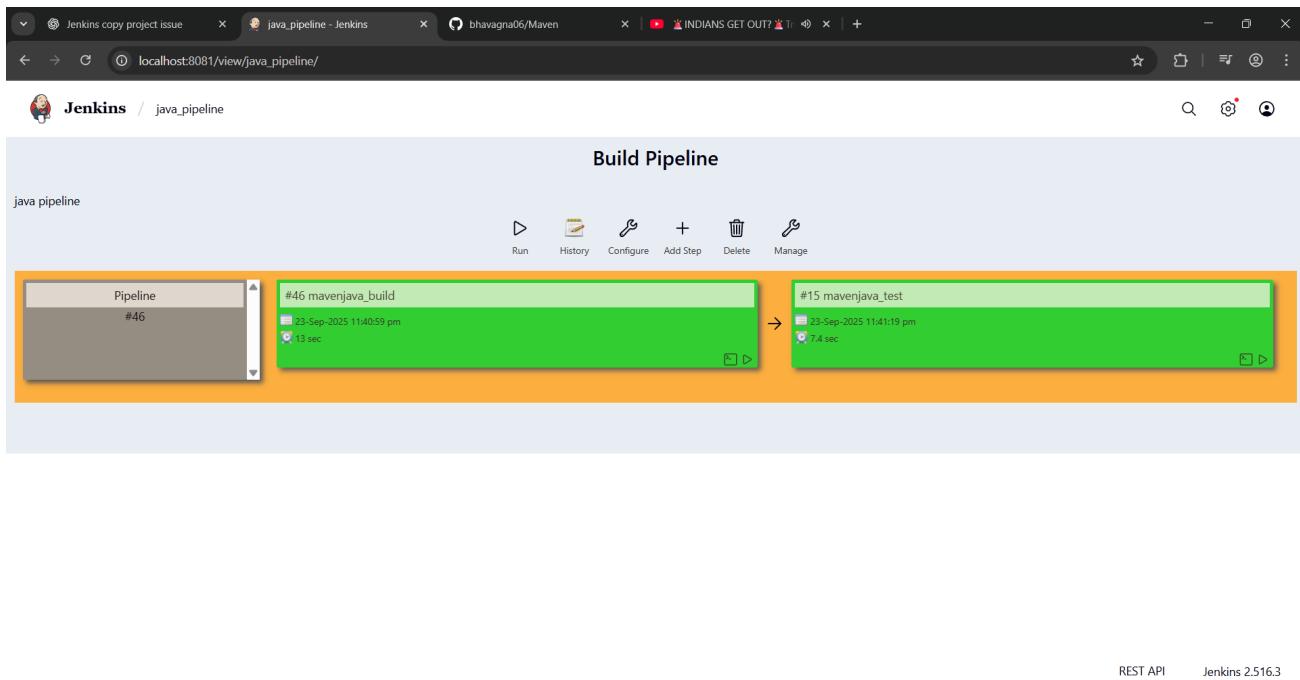
Apply and Save OK

### Step 5: Run the Pipeline and Check Output

Click on the trigger to run the pipeline



click on the small black box to open the console to check if the build is success



## II. Maven Web Automation Steps:

**Step 1:** Open Jenkins (localhost:8081)

|— Click on "New Item" (left side menu)

New Item

Enter an item name

Select an item type

- 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.
- Pipeline  
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.
- 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

OK

## Step 2: Create Freestyle Project (e.g., MavenWeb\_Build)

— Enter project name (e.g., MavenWeb\_Build)

— Click "OK"

Enter an item name

mavenweb\_build

Select an item type

- 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.
- Pipeline  
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.
- 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

OK

## Configure the project:

— Description: "Web Build demo"

The screenshot shows the Jenkins configuration page for a job named "mavenweb\_build". The "General" tab is selected. The "Description" field contains "maven web build". The "Enabled" switch is turned on. Other tabs visible include "Source Code Management", "Triggers", "Environment", "Build Steps", and "Post-build Actions". Buttons at the bottom are "Save" and "Apply".

## Source Code Management:

— Git repository URL: [https://github.com/bhavagna06/maven-web-app.git]

— Branches to build: \*/Main or master

The screenshot shows the GitHub repository page for "maven-web-app" owned by "bhavagna06". The repository has 1 branch and 0 tags. The main branch is "main". The commit history shows an initial commit by "bhavagna06" 3 weeks ago. The repository details page includes sections for "About", "Activity", "Releases", "Packages", and "Languages".

File	Commit	Date
.settings	Initial Commit	3 weeks ago
src/main/webapp	Initial Commit	3 weeks ago
target	Initial Commit	3 weeks ago
.classpath	Initial Commit	3 weeks ago
.project	Initial Commit	3 weeks ago
Dockerfile	Initial Commit	3 weeks ago
pom.xml	Initial Commit	3 weeks ago

The screenshot shows the Jenkins configuration page for a job named "mavenweb\_build". Under the "Source Code Management" section, "Git" is selected as the provider. The "Repository URL" is set to `https://github.com/bhavagna06/maven-web-app.git`. The "Branch Specifier" is set to `*/*main`. The "Save" and "Apply" buttons are at the bottom.

## Build Steps:

- └─ **Add Build Step** -> "Invoke top-level Maven targets"
  - └─ Maven version: MAVEN\_HOME
  - └─ Goals: clean
- └─ **Add Build Step** -> "Invoke top-level Maven targets"
  - └─ Maven version: MAVEN\_HOME
  - └─ Goals: install

The screenshot shows the Jenkins configuration page with two "Invoke top-level Maven targets" build steps. Both steps have "MAVEN\_HOME" selected for Maven Version and "clean" selected for Goals. The second step has "install" selected for Goals. The "Save" and "Apply" buttons are at the bottom.

### Post-build Actions:

└─ Add Post Build Action -> "Archive the artifacts"

  └─ Files to archive: \*\*/\*

└─ Add Post Build Action -> "Build other projects"

  └─ Projects to build: MavenWeb\_Test

  └─ Trigger: Only if build is stable

The screenshot shows the Jenkins configuration interface for a job named 'mavenweb\_build'. The left sidebar lists 'General', 'Source Code Management', 'Triggers', 'Environment', 'Build Steps', and 'Post-build Actions'. The 'Post-build Actions' tab is selected and highlighted in grey. The main content area is titled 'Post-build Actions' with the sub-instruction 'Define what happens after a build completes, like sending notifications, archiving artifacts, or triggering other jobs.' Below this, there are two sections: 'Archive the artifacts' and 'Build other projects'. In the 'Archive the artifacts' section, the 'Files to archive' field contains '\*\*/\*'. In the 'Build other projects' section, the 'Projects to build' field contains 'mavenweb\_test'. A red error message 'No such project 'mavenweb\_test''. Did you mean 'mavenjava\_test'' is displayed next to the field. Below the fields are three trigger options: 'Trigger only if build is stable' (selected), 'Trigger even if the build is unstable', and 'Trigger even if the build fails'. At the bottom of the configuration panel are 'Save' and 'Apply' buttons.

└─ Apply and Save

### Step 3: Create Freestyle Project (e.g., MavenWeb\_Test)

└─ Enter project name (e.g., MavenWeb\_Test)

└─ Click "OK"

The screenshot shows the Jenkins 'New Item' creation dialog. In the top bar, there are tabs for 'Jenkins copy project issue', 'New Item - Jenkins', and 'localhost:8081/view/all/new/job'. The main area has a search icon and a 'New Item' button. A sub-dialog titled 'Enter an item name' contains the text 'mavenweb\_test'. Below it, 'Select an item type' is set to 'Freestyle project'. Other options shown are 'Pipeline', 'Multi-configuration project', 'Folder', and 'Multibranch Pipeline'. A blue 'OK' button is at the bottom.

## Configure the project:

└— Description: "Test demo"

The screenshot shows the Jenkins configuration page for the 'mavenweb\_test' job. The top bar includes tabs for 'Jenkins copy project issue', 'mavenweb\_test Config - Jenkins', and 'localhost:8081/job/mavenweb\_test/configure'. The main area has a search icon and a 'Configuration' link. On the left, a sidebar lists 'General', 'Source Code Management', 'Triggers', 'Environment', 'Build Steps', and 'Post-build Actions'. The 'General' tab is active. It shows a 'Description' field containing 'test demo'. Underneath are several checkboxes: 'Discard old builds', 'GitHub project', 'Permission to Copy Artifact', 'This project is parameterized', 'Throttle builds', and 'Execute concurrent builds if necessary'. At the bottom are 'Save' and 'Apply' buttons.

## Build Environment:

└— Check: "Delete the workspace before build starts"

The screenshot shows the Jenkins configuration page for a job named "mavenweb\_test". The "Triggers" section is selected. Under "Delete workspace before build starts", the checkbox is checked. Other options like "Use secret text(s) or file(s)", "Add timestamps to the Console Output", "Inspect build log for published build scans", "Terminate a build if it's stuck", and "With Ant" are available but unchecked. Below the triggers section is a "Build Steps" section with a "Add build step" button. At the bottom are "Save" and "Apply" buttons.

**Add Build Step -> "Copy artifacts from another project"**

└─ Project name: MavenWeb\_Build

└─ Build: Stable build only

└─ Artifacts to copy: \*\*/\*

**Add Build Step -> "Invoke top-level Maven targets"**

└─ Maven version: MAVEN\_HOME

└─ Goals: test

The screenshot shows the Jenkins configuration page for the same job. The "Build Steps" section is expanded, showing a "Copy artifacts from another project" step. The "Project name" field contains "mavenweb\_build". The "Which build" dropdown is set to "Latest successful build". The "Stable build only" checkbox is unchecked. The "Artifacts to copy" field contains "\*\*/\*". The "Artifacts not to copy" and "Target directory" fields are empty. At the bottom are "Save" and "Apply" buttons.

## Post-build Actions:

— Add Post Build Action -> "Archive the artifacts"

  — Files to archive: `**/*`

— Add Post Build Action -> "Build other projects"

  — Projects to build: MavenWeb\_Deploy

The screenshot shows the Jenkins configuration interface for the 'mavenweb\_test' job. The 'Post-build Actions' section is expanded, displaying two configurations:

- Archive the artifacts**: Set to archive files matching `**/*`.
- Build other projects**: Set to build the project 'mavenweb\_deploy'. A warning message indicates that 'mavenweb\_deploy' does not exist, suggesting a typo in the input field.

At the bottom of the configuration page are 'Save' and 'Apply' buttons.

Apply and Save

## Step 4: Create Freestyle Project (e.g., MavenWeb\_Deploy)

— Enter project name (e.g., MavenWeb\_Deploy)

— Click "OK"

Jenkins / All / New Item

## New Item

Enter an item name

mavenweb\_deploy

Select an item type

**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.

**Pipeline**  
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

**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.

OK

### Configure the project:

— Description: "Web Code Deployment"

Jenkins / mavenweb\_deploy / Configuration

## Configure

## General

Enabled

Description

web code deployment

Plain text [Preview](#)

Discard old builds ?

GitHub project

Permission to Copy Artifact

This project is parameterized ?

Throttle builds ?

Execute concurrent builds if necessary ?

Advanced ▾

Save Apply

### Build Environment:

— Check: "Delete the workspace before build starts"

The screenshot shows the Jenkins job configuration page for 'mavenweb\_deploy'. The 'Triggers' section is selected. It contains several triggers: 'Build after other projects are built', 'Build periodically', 'GitHub hook trigger for GITScm polling', and 'Poll SCM'. Under the 'Environment' section, there is a note about configuring build context settings like credentials and paths. The 'Build Steps' section is collapsed. At the bottom are 'Save' and 'Apply' buttons.

**Add Build Step -> "Copy artifacts from another project"**

└─ Project name: MavenWeb\_Test

└─ Build: Stable build only

└─ Artifacts to copy: \*\*/\*

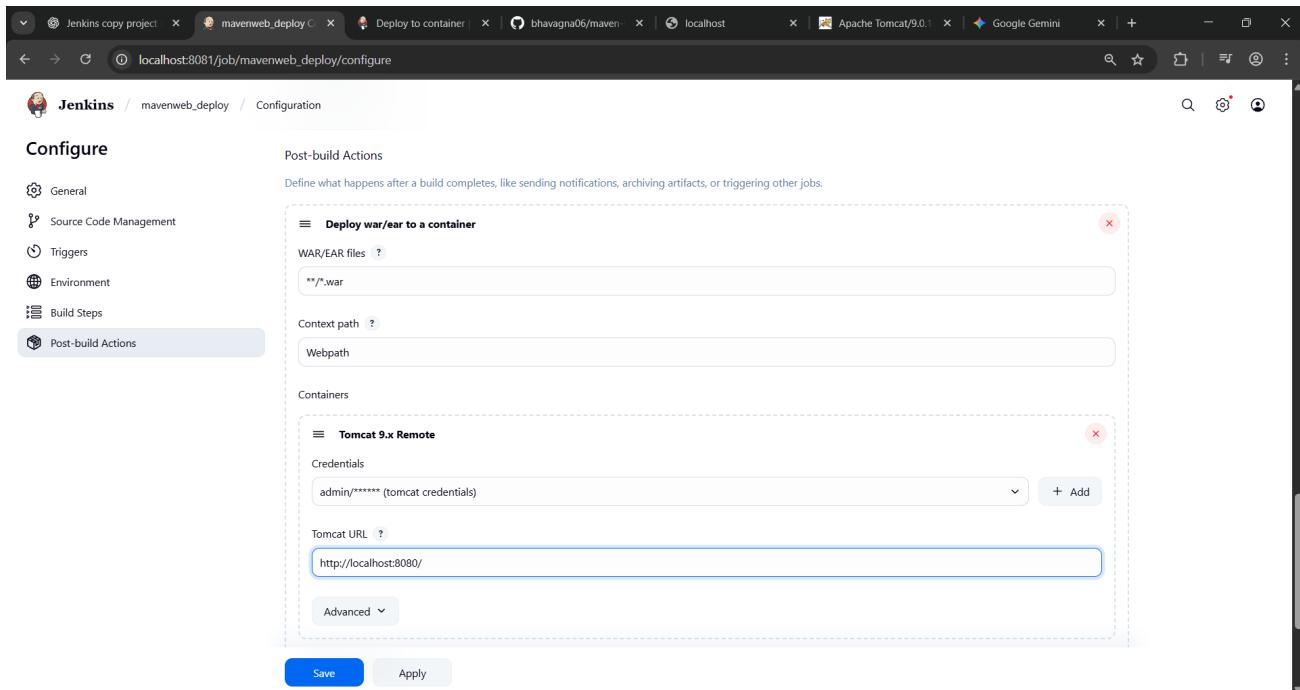
The screenshot shows the Jenkins job configuration page for 'mavenweb\_deploy'. The 'Build Steps' section is expanded, showing a 'Copy artifacts from another project' step. The configuration for this step includes: Project name set to 'mavenweb\_test', Which build set to 'Latest successful build', and Artifacts to copy set to '\*\*/\*'. There are also fields for Artifacts not to copy and Target directory, both currently empty. At the bottom are 'Save' and 'Apply' buttons.

**Post-build Actions:**

└─ Add Post Build Action -> "Deploy WAR/EAR to a container"

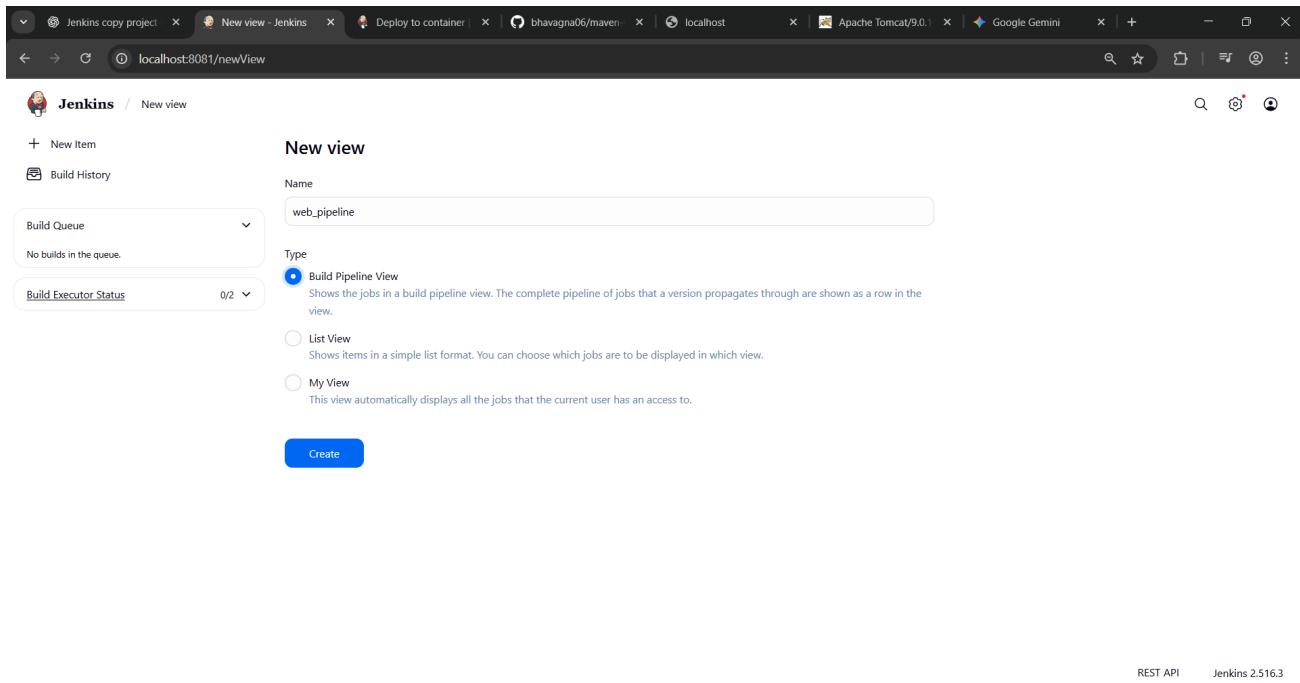
└─ WAR/EAR File: \*\*/\*.war

- └─ Context path: Webpath
- └─ Add container -> Tomcat 9.x remote
- └─ Credentials: Username: admin, Password: 1234
- └─ Tomcat URL: <https://localhost:8080/>



└─ Apply and Save

- └─ Step 5: Create Pipeline View for MavenWeb
  - ├─ Click "+" beside "All" on the dashboard
  - ├─ Enter name: MavenWeb\_Pipeline



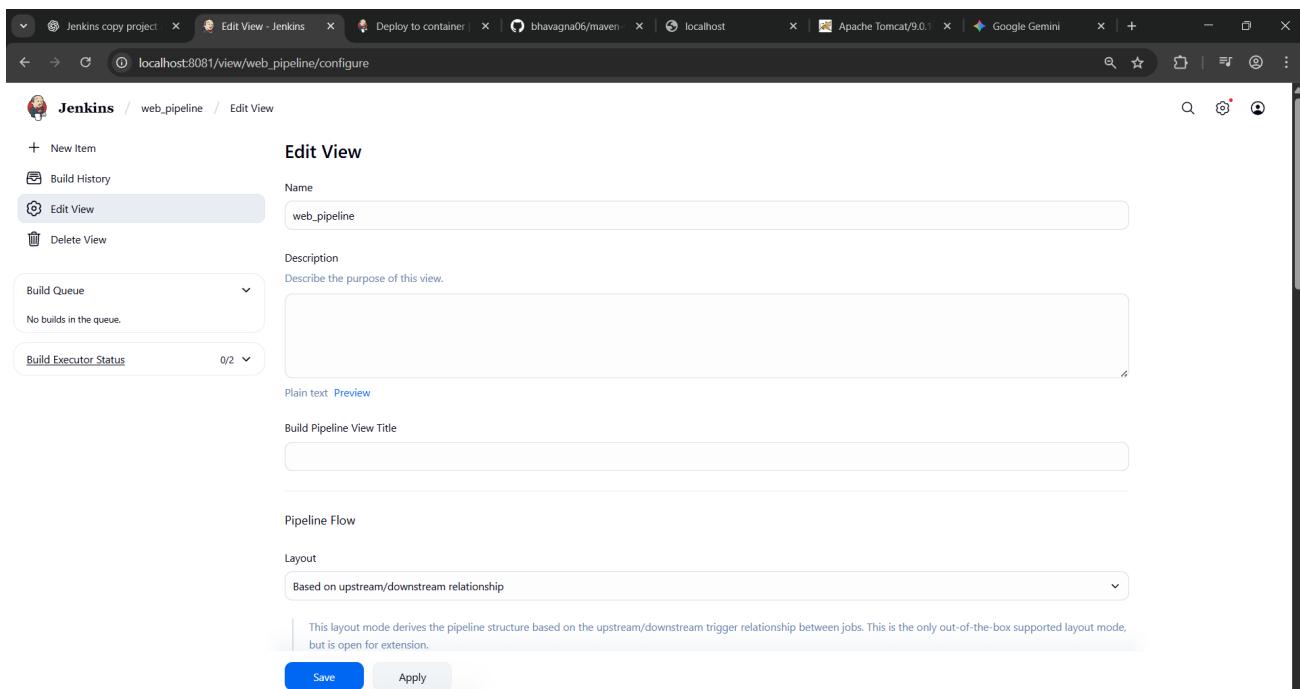
The screenshot shows the Jenkins 'New view' creation interface. At the top, there are several tabs: 'Jenkins copy project', 'New view - Jenkins', 'Deploy to container', 'bhavagna06/maven', 'localhost', 'Apache Tomcat/9.0.1', and 'Google Gemini'. The current tab is 'localhost'. Below the tabs, the URL 'localhost:8081/newView' is visible. The main content area is titled 'New view'. It has a 'Name' field containing 'web\_pipeline'. Under the 'Type' section, 'Build Pipeline View' is selected (indicated by a blue circle). Other options shown are 'List View' and 'My View'. A 'Create' button is at the bottom. On the left, there are sections for 'Build Queue' (empty) and 'Build Executor Status' (0/2). At the bottom right, it says 'REST API Jenkins 2.516.3'.

## Select "Build pipeline view"

### └─ Pipeline Flow:

  └─ Layout: Based on upstream/downstream relationship

  └─ Initial job: MavenWeb\_Build



The screenshot shows the Jenkins 'Edit View' configuration interface for the 'web\_pipeline' view. The top navigation bar includes tabs for 'Jenkins copy project', 'Edit View - Jenkins', 'Deploy to container', 'bhavagna06/maven', 'localhost', 'Apache Tomcat/9.0.1', and 'Google Gemini'. The current tab is 'localhost'. The URL 'localhost:8081/view/web\_pipeline/configure' is displayed. The main content is titled 'Edit View' and shows the 'Name' field set to 'web\_pipeline'. There is a 'Description' field with the placeholder 'Describe the purpose of this view.' Below these are sections for 'Build Queue' (empty), 'Build Executor Status' (0/2), and 'Plain text Preview'. Under 'Pipeline Flow', the 'Layout' dropdown is set to 'Based on upstream/downstream relationship'. A note below explains this mode: 'This layout mode derives the pipeline structure based on the upstream/downstream trigger relationship between jobs. This is the only out-of-the-box supported layout mode, but is open for extension.' At the bottom are 'Save' and 'Apply' buttons.

The screenshot shows the Jenkins Pipeline configuration page. Under 'Layout', it is set to 'Based on upstream/downstream relationship'. In the 'Upstream / downstream config' section, 'Select Initial Job' is set to 'mavenweb\_build'. Below this, under 'Trigger Options', there are settings for 'Build Cards' (set to 'Standard build card') and 'Restrict triggers to most recent successful builds' (set to 'No'). A note says 'Always allow manual trigger on pipeline steps'. At the bottom are 'Save' and 'Apply' buttons.

└— Apply and Save

## └— Step 6: Run the Pipeline and Check Output

├— Click on the trigger “RUN” to run the pipeline

The screenshot shows the Jenkins Pipeline interface. It displays a pipeline with three stages: 'mavenweb\_build' (green card), 'mavenweb\_test' (green card), and 'mavenweb\_deploy' (green card). Each stage has a timestamp and duration. The stages are connected by arrows indicating a sequential flow. The top navigation bar shows the pipeline name 'web\_pipeline'.

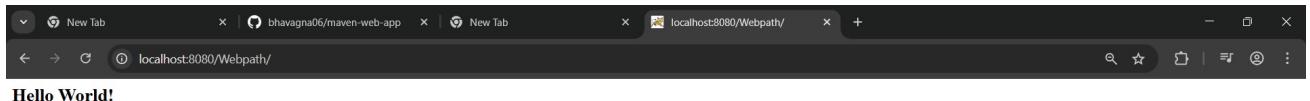
Note:

1. After Click on Run -> click on the small black box to open the console to check if the build is success

2. Now we see all the build has success if it appears in green color

|— Open Tomcat homepage in another tab

|— Click on the "/webpath" option under the manager app



### **III. Questions on Jenkins**

1. What is Jenkins primarily used for?

Jenkins is used for continuous integration (CI) and continuous deployment/Delivery (CD).

2. What is feature of Jenkins?

pipeline support, extensibility

3. What is the default port on which Jenkins runs?

8080

4. What can be integrated with Jenkins for version control?

Git

5. What is the purpose of Jenkins plugins?

Plugins help in functionality like scm, email and tools

6. Which type of Jenkins job is best suited for running one-off tasks or small scripts?

Freestyle project

7. How can you manage sensitive information such as API keys in Jenkins?

It is stored in Jenkins credentials plugin

8. What does the "blue ocean" feature in Jenkins refer to?

Blue Ocean is a modern UI for Jenkins ,improves usability and user-friendly

9. What does the "blue ocean" feature in Jenkins refer to?

Blue Ocean is a modern UI for Jenkins ,improves usability and user-friendly

10. Which Jenkins component allows for distributed builds across multiple machines?

It uses Master-Agent Architecture

11. List at least five Jenkins plugins that you would consider important for a microservices-based application CI/CD pipeline. Briefly explain the purpose of each plugin.

Git Plugin → Integrates Git repositories for version control.Pipeline Plugin → Allows defining CI/CD pipelines using Groovy (Jenkinsfile).Docker Plugin → Builds, runs, and manages Docker containers for microservices

12. Explain the steps you would take to install a plugin in Jenkins through the Jenkins UI. What considerations would you keep in mind regarding plugin compatibility and updates?

In Manage Plugins we can search for plugin and install

13. Explain the steps you would take to install a plugin in Jenkins through the Jenkins UI. What considerations would you keep in mind regarding plugin compatibility and updates?

Dashboard ->Manage Jenkins->Manage plugin ->install

14. After installing a plugin, explain how you would configure it within Jenkins. For example, if you installed the Git Plugin, what steps would you take to set it up for your pipeline?

After installation configure it with global tools and add credentials and it in pipeline

15. Discuss common issues that might arise when using Jenkins plugins, such as dependency conflicts or version compatibility problems. How would you troubleshoot these issues?

Version compatibility, Dependency conflict