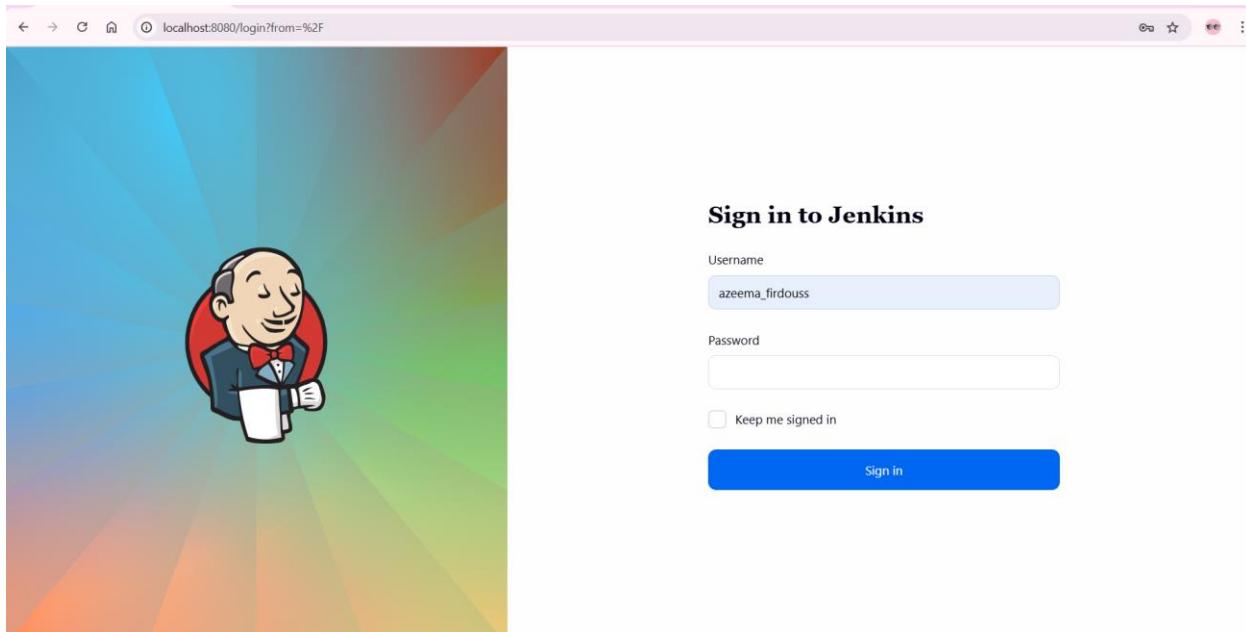


Week 8: Jenkins Automation

I. Steps for MavenJava Automation:

Maven Java Automation Steps:

Step 1: Open Jenkins (localhost:8080)



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



Jenkins

+ New Item

Build History

Build Queue



No builds in the queue.

Build Executor Status



(0 of 2 executors busy)



Ico

Step 2: Create Freestyle Project (e.g., MavenJava_Build)

└─ Enter project name (e.g., MavenJava_Build)

└─ Click "OK"

Jenkins / All / New Item

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.

OK

└─ Configure the project:

└─ Description: "Java Build demo"

Jenkins / MavenJava_Build / Configure

Configure General Enabled

General

Source Code Management

Triggers

Environment

Build Steps

Post-build Actions

Description
Java Build demo

Plain text [Preview](#)

Discard old builds [?](#)

GitHub project

This project is parameterized [?](#)

Throttle builds [?](#)

Execute concurrent builds if necessary [?](#)

Advanced [▼](#)

Source Code Management

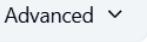
[Save](#) [Apply](#)

└─ **Source Code Management:**

└─ Git repository URL: [GitMavenJava repo URL]

└─ **Branches to build:** */Main or */master

 **Configure**

General 

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

Throttle builds 

Execute concurrent builds if necessary 

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/azeemafirdouss/java_maven.git`

Credentials 

- none -  

Save **Apply**

Branches to build ?

Branch Specifier (blank for 'any') ? X

*/main

Add Branch

Repository browser ?

(Auto)

Additional Behaviours

Add ▾

This screenshot shows the 'Build' section of a Jenkins job configuration. It includes fields for 'Branches to build' (set to '*/main'), 'Repository browser' (set to '(Auto)'), and 'Additional Behaviours' (with an 'Add' button). The 'Build Steps' section is expanded, showing two Maven build steps. The first step uses MAVEN_HOME and goals clean. The second step also uses MAVEN_HOME and goals install.

└─ **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



Jenkins

/ MavenJava_Build

/ Configure



Configure

General

Source Code Management

Triggers

Environment

Build Steps

Post-build Actions

 Inspect build log for published build scans Terminate a build if it's stuck With Ant ?

Build Steps

Automate your build process with ordered tasks like code compilation, testing, and deployment.

+ Add build step

Filter

Execute Windows batch command

Execute shell

Invoke Ant

Invoke Gradle script

Invoke top-level Maven targets

Run with timeout

Set build status to "pending" on GitHub commit

Trigger/call builds on other projects

sending notifications, archiving

REST API

jenkins 2.530

The screenshot shows the Jenkins 'Configure' screen for a project named 'MavenJava_Build'. The left sidebar lists configuration sections: General, Source Code Management, Triggers, Environment, Build Steps (which is selected and highlighted in grey), and Post-build Actions.

The main area contains two identical configurations for 'Invoke top-level Maven targets' (indicated by a dashed border). Each configuration includes:

- Maven Version: Maven
- Goals:
 - Top configuration: clean
 - Bottom configuration: install
- Advanced settings button

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: MavenJava_Test

 └─ Trigger: Only if build is stable

└─ Apply and Save

artifacts, or triggering other jobs.

☰ Archive the artifacts ?

Files to archive ?

**/*

Advanced ▾



☰ Build other projects ?

Projects to build

! No project specified

Trigger only if build is stable

Trigger even if the build is unstable

Trigger even if the build fails



+ Add post-build action

Save

Apply

Configure

General Advanced ▾

Source Code Management

Triggers

Environment

Build Steps

Post-build Actions

Build other projects ? X

Projects to build

MavenJava_Test

No items

Trigger only if build is stable
 Trigger even if the build is unstable
 Trigger even if the build fails

+ Add post-build action

Save Apply

The screenshot shows the Jenkins 'Configure' screen for a project. On the left, there's a sidebar with links: General, Source Code Management, Triggers, Environment, Build Steps, and Post-build Actions (which is currently selected). In the main area, there's a 'Build other projects' section with a title 'Build other projects ?' and a red 'X' button. It contains a 'Projects to build' list with 'MavenJava_Test' selected. Below it are three radio buttons for triggering: 'Trigger only if build is stable' (selected), 'Trigger even if the build is unstable', and 'Trigger even if the build fails'. At the bottom are 'Save' and 'Apply' buttons.

└─ Step 3: Create Freestyle Project (e.g., MavenJava_Test)

 └─ Enter project name (e.g., MavenJava_Test)

 └─ Click "OK"

 Jenkins / All / New Item 🔍 ⚙️ 🌐

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.



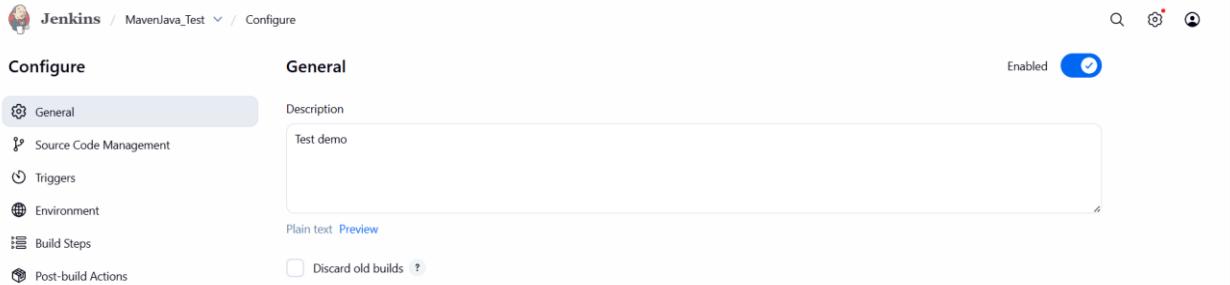
Folder

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.

OK

Configure the project:

Description: "Test demo"

 Jenkins / MavenJava_Test / Configure 🔍 ⚙️ 🌐

Configure **General** Enabled

General

Description: Test demo

Plain text [Preview](#)

Discard old builds ?

Build Environment:

Check: "Delete the workspace before build starts"

The screenshot shows the Jenkins General configuration page. The 'Environment' section is selected. It contains a checkbox for 'Delete workspace before build starts' which is checked. Below it is an 'Advanced' dropdown menu. Underneath the dropdown are several other checkboxes: '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'.

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

 └─ Project name: MavenJava_Build

 └─ Build: Stable build only // tick at this

Build Steps

Automate your build process with ordered tasks like code compilation, testing, and deployment.

The screenshot shows the 'Copy artifacts from another project' step configuration. It includes fields for 'Project name' (set to 'MavenJava_Build'), 'Which build' (set to 'Latest successful build'), and a checkbox for 'Stable build only' which is checked. There is also an 'X' button in the top right corner of the step panel.

 └─ Artifacts to copy: **/*

The screenshot shows the expanded 'Copy artifacts from another project' step configuration. It includes fields for 'Which build' (set to 'Latest successful build'), a checked checkbox for 'Stable build only', and an 'Artifacts to copy' field containing '**/*'. Below these, there is an 'Artifacts not to copy' field which is currently empty.

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

 └─ Maven version: MAVEN_HOME

 └─ Goals: test

☰ Invoke top-level Maven targets ?

Maven Version

(Default)

Goals

test

Advanced ▾

+ Add build step

└─ Post-build Actions:

+ Add build step

Post-build Actions

Define what happens after a build completes, like sending notifications, archiving artifacts, or triggering other jobs.

☰ Archive the artifacts ?

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

└─ Files to archive: **/*

└─ Apply and Save

Post-build Actions

Define what happens after a build completes, like sending notifications, archiving artifacts, or triggering other jobs.

☰ Archive the artifacts ?

Files to archive ?

**/*

Advanced ▾

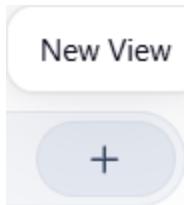
+ Add post-build action

Save

Apply

└─ Step 4: Create Pipeline View for Maven Java project

└─ Click "+" beside "All" on the dashboard



|—— Enter name: MavenJava_Pipeline

|—— Select "Build pipeline view" // tick here

|--- create

Jenkins / New view

New view

Name: MavenJava_Pipeline

A view already exists with the name "MavenJava_Pipeline".

Type:

Build Pipeline View
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.

List View
Shows items in a simple list format. You can choose which jobs are to be displayed in which view.

My View
This view automatically displays all the jobs that the current user has an access to.

Create

|—— Pipeline Flow:

|—— Layout: Based on upstream/downstream relationship

Pipeline Flow

Layout

Based on upstream/downstream relationship

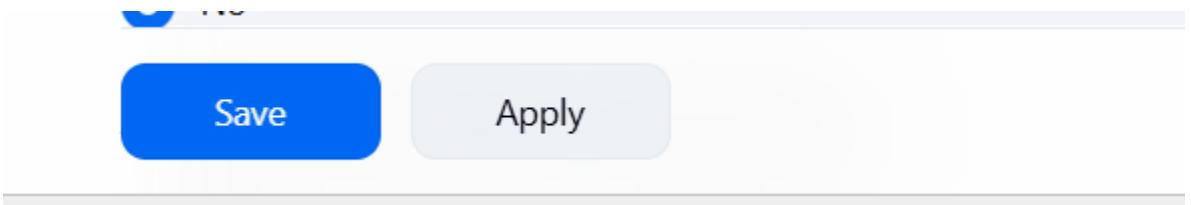
|—— Initial job: MavenJava_Build

Upstream / downstream config

Select Initial Job ?

MavenJava_Build

└─ 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

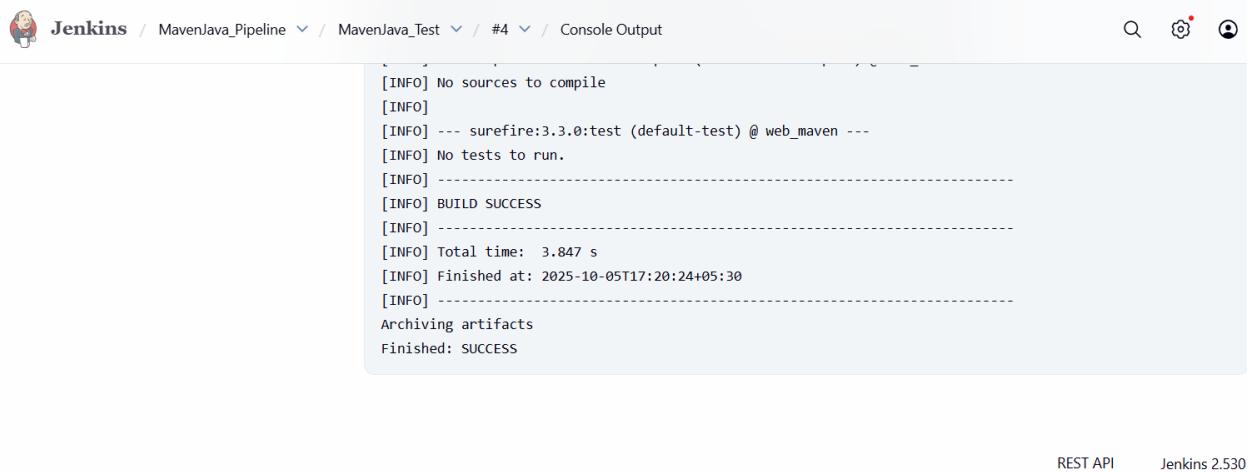
The screenshot shows the Jenkins Build Pipeline interface. At the top, there are buttons for Run, History, Configure, Add Step, Delete, and Manage. Below these are two green pipeline stages. The first stage is labeled '#4 MavenJava_Build' and was triggered on '05-Oct-2025 5:19:42 pm' by 'azeema_firdous'. The second stage is labeled '#4 MavenJava_Test' and was triggered on '05-Oct-2025 5:20:13 pm' by 'azeema_firdous'. Both stages are shown as successful (green).

Note :

1. If build is success and the test project is also automatically triggered with name "MavenJava_Test"
2. The pipeline is successful if it is in green color as shown ->check the console of the test project
3. The test project is successful and all the artifacts are archived successfully

The screenshot shows the Jenkins Console Output for build #4 of the MavenJava_Test pipeline. The left sidebar has links for Status, Changes, Console Output (which is selected), Edit Build Information, Delete build '#4', Timings, See Fingerprints, and Previous Build. The main area shows the console output with the following log entries:

```
Started by upstream project "MavenJava_Build" build number 4
originally caused by:
Started by user Azeema firdous
Running as SYSTEM
Building in workspace C:\ProgramData\Jenkins\.jenkins\workspace\MavenJava_Test
[WS-CLEANUP] Deleting project workspace...
[WS-CLEANUP] Deferred wipeout is used...
[WS-CLEANUP] Done
Copied 14 artifacts from "MavenJava_Build" build number 4
[MavenJava_Test] $ cmd.exe /C "mvn test && exit %ERRORLEVEL%"
[INFO] Scanning for projects...
[INFO]
[INFO] -----< maven_web:web_maven >-----
[INFO] Building web_maven Maven Webapp 0.0.1-SNAPSHOT
[INFO]   from pom.xml
```



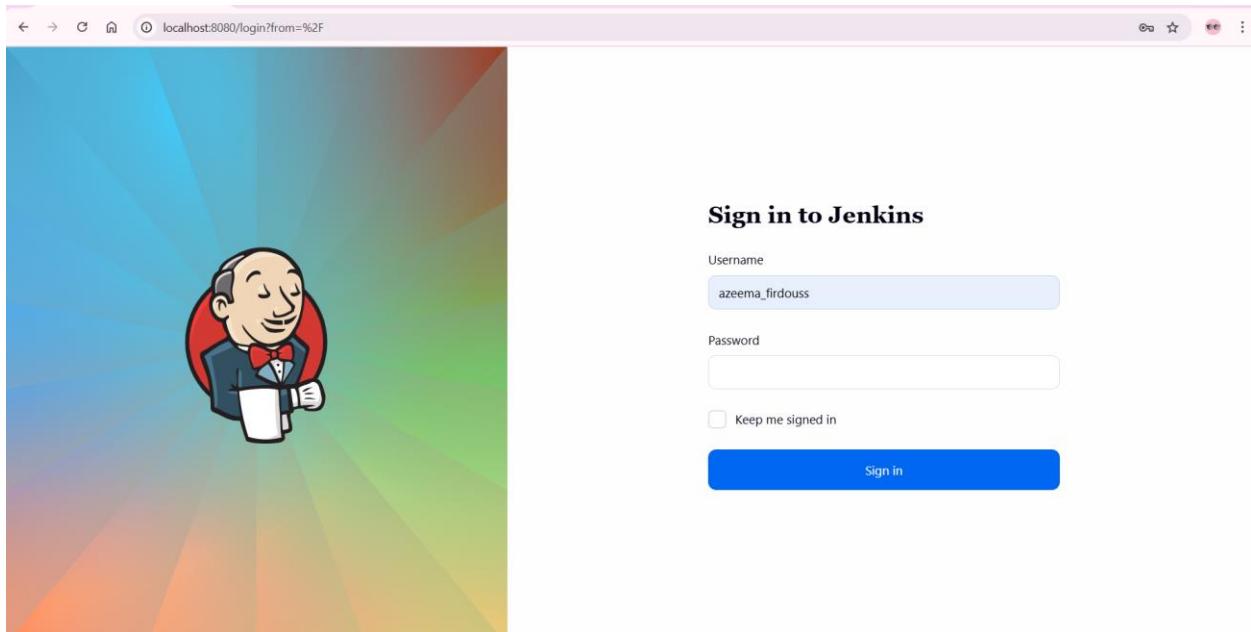
Jenkins / MavenJava_Pipeline / MavenJava_Test / #4 / Console Output

```
[INFO] No sources to compile
[INFO]
[INFO] --- surefire:3.3.0:test (default-test) @ web_maven ---
[INFO] No tests to run.
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time:  3.847 s
[INFO] Finished at: 2025-10-05T17:20:24+05:30
[INFO] -----
Archiving artifacts
Finished: SUCCESS
```

REST API Jenkins 2.530

II. Maven Web Automation Steps:

— Step 1: Open Jenkins (localhost:8080)



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



Jenkins

+ New Item

Build History

Build Queue



No builds in the queue.

Build Executor Status



(0 of 2 executors busy)



Ico

└─ Step 2: Create Freestyle Project (e.g., MavenWeb_Build)

- ├─ Enter project name (e.g., MavenWeb_Build)
- └─ Click "OK"

Jenkins / All / New Item

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
Creates a set of Pipeline projects according to detected branches in one SCM repository.

OK

└─ Configure the project:

- └─ Description: "Web Build demo"

General

Enabled

Description
Web Build demo

Plain text [Preview](#)

Discard old builds [?](#)

└─ **Source Code Management:**

 └─ Git repository URL: [GitMavenWeb repo URL]

└─ *Branches to build: */Main or master*

Git [?](#)

Repositories [?](#)

Repository URL [?](#)
 [X](#)

Credentials [?](#)
- none - [+](#) Add

Advanced [▼](#)

+ Add Repository

Branches to build [?](#)

Branch Specifier (blank for 'any') [?](#)
 [X](#)

└─ **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

Invoke top-level Maven targets ? ×

Maven Version

Goals

Advanced ▼

+ Add build step

└— Post-build Actions:

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

 |— Files to archive: **/*

Post-build Actions

Define what happens after a build completes, like sending notifications, archiving artifacts, or triggering other jobs.

Archive the artifacts ? ×

Files to archive ?

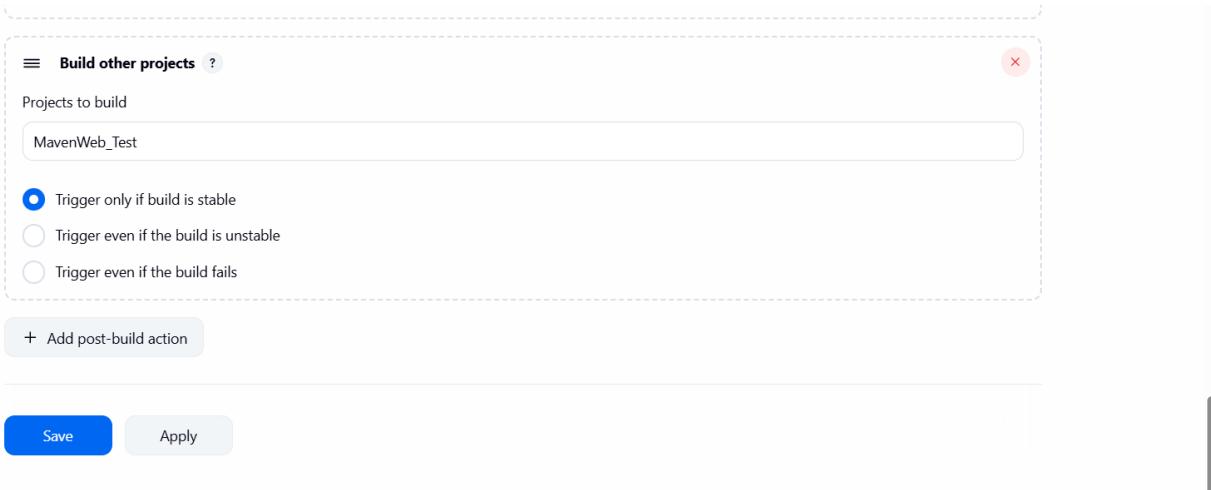
Advanced ▼

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

 |— Projects to build: MavenWeb_Test

 |— Trigger: Only if build is stable

 |— Apply and Save



└─ **Step 3:** Create Freestyle Project (e.g., MavenWeb_Test)

 └─ Enter project name (e.g., MavenWeb_Test)

 └─ Click "OK"

└─ **Configure the project:**

 Jenkins / All / New Item

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.

OK

└─ Description: "Test demo"

 Jenkins / MavenWeb_Test / Configure

Configure

- General**
- Source Code Management
- Triggers
- Environment
- Build Steps

General

Description: Test demo

Plain text [Preview](#)

Enabled

└─ Build Environment:

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

Source Code Management

Triggers

Environment

Build Steps

Post-build Actions

Environment

Configure settings and variables that define the context in which your build runs, like credentials, paths, and global parameters.

Delete workspace before build starts

Advanced

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

With Ant ?

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

- └─ Project name: MavenWeb_Build
- └─ Build: Stable build only
- └─ Artifacts to copy: **/*

Configure

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

Automate your build process with ordered tasks like code compilation, testing, and deployment.

Copy artifacts from another project

Project name ?
MavenWeb_Build

Which build ?
Latest successful build

Stable build only

Artifacts to copy ?
**/*

Artifacts not to copy ?

Target directory ?

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

- └─ Maven version: MAVEN_HOME
- └─ Goals: test

Invoke top-level Maven targets ?

Maven Version
(Default)

Goals
test

Advanced

+ Add build step

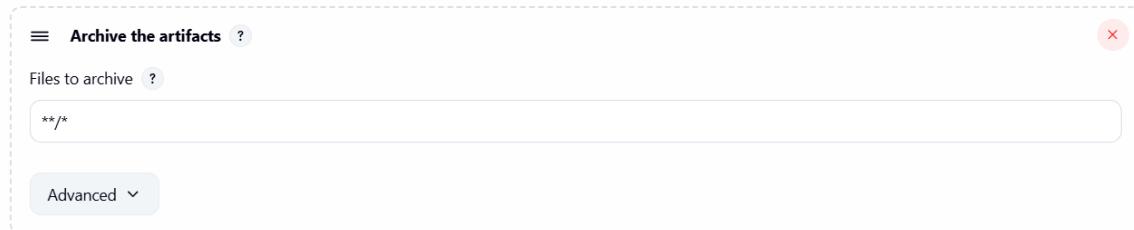
└─ Post-build Actions:

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

 └─ Files to archive: **/*

Post-build Actions

Define what happens after a build completes, like sending notifications, archiving artifacts, or triggering other jobs.



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

 └─ Projects to build: MavenWeb_Deploy

 └─ Apply and Save

Jenkins / MavenWeb_Test / Configure

Configure

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

FILES TO ARCHIVE: **/*

ADVANCED

Build other projects

Projects to build: MavenWeb_Deploy

Trigger only if build is stable (selected)

Trigger even if the build is unstable

Trigger even if the build fails

+ Add post-build action

Save Apply

└─ Step 4: Create Freestyle Project (e.g., MavenWeb_Deploy)

 └─ Enter project name (e.g., MavenWeb_Deploy)

 └─ Click "OK"

 └─ Configure the project:

 Jenkins / All / New Item

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.

OK

└─ Description: "Web Code Deployment"

 Jenkins / MavenWeb_Deploy / Configure

Configure

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

General

Description

Plain text Preview

Discard old builds ?

GitHub project

Enabled

└─ Build Environment:

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

Environment

Configure settings and variables that define the context in which your build runs, like credentials, paths, and global parameters.

- Delete workspace before build starts

Advanced ▾

- 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
- With Ant ?

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

 |— Project name: MavenWeb_Test

 |— Build: Stable build only

 |— Artifacts to copy: **/*

Build Steps

Automate your build process with ordered tasks like code compilation, testing, and deployment.

≡ Copy artifacts from another project

Project name ?

MavenWeb_Test

Which build ?

Latest successful build

- Stable build only

Artifacts to copy ?

**/*

|— 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:8085/

└— Apply and Save

Post-build Actions

Define what happens after a build completes, like sending notifications, archiving artifacts, or triggering other jobs.

The screenshot shows the Jenkins 'Post-build Actions' configuration page. It includes sections for 'Deploy war/ear to a container' (with WAR/EAR files set to '**/*.war' and Context path to 'Webpath') and 'Containers' (specifically 'Tomcat 9.x Remote' with no credentials and Tomcat URL set to 'https://localhost:8085').

└— Step 5: Create Pipeline View for MavenWeb

- └— Click "+" beside "All" on the dashboard
- └— Enter name: MavenWeb_Pipeline
- └— Select "**Build pipeline view**"

New view

Name

MavenWeb_pipeline

Type

Build Pipeline View

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.

List View

Shows items in a simple list format. You can choose which jobs are to be displayed in which view.

My View

This view automatically displays all the jobs that the current user has an access to.

Create

└─ Pipeline Flow:

 └─ Layout: Based on upstream/downstream relationship

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

 └─ Initial job: MavenWeb_Build

 Upstream / downstream config

 Select Initial Job ?

 MavenWeb_Build

└─ Apply and Save

Build Cards

 Standard build card

 | Use the default build cards

 Restrict triggers to most recent successful builds ?

Yes

✓ Saved

 Save

 Apply

└─ Step 6: Run the Pipeline and Check Output

 └─ Click on the trigger “RUN” to run the 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

The Jenkins Build Pipeline interface displays a sequence of three stages:

- Pipeline #36**: Status: Success (green background)
- #36 MavenWeb_Build**: Status: Success (green background)
 - Started: 06-Oct-2025 7:39:17 pm
 - Duration: 23 sec
 - Build Cause: azemma_firdous
- #50 MavenWeb_Test**: Status: Success (green background)
 - Started: 06-Oct-2025 7:40:01 pm
 - Duration: 12 sec
 - Build Cause: azemma_firdous
- MavenWeb_Deploy**: Status: N/A (blue background)

Below the pipeline view, the Jenkins Console Output for the MavenWeb_Test stage is shown:

```
Started by upstream project "MavenWeb_Build" build number 1
originally caused by:
    Started by user Nagulapally Harshitha
Running as SYSTEM
Building in workspace C:\ProgramData\Jenkins\.jenkins\workspace\MavenWeb_Test
[WS-CLEANUP] Deleting project workspace...
[WS-CLEANUP] Deferred wipeout is used...
Copied 14 artifacts from "MavenWeb_Build" build number 1
[MavenWeb_Test] $ cmd.exe /C "C:\ProgramData\Jenkins\.jenkins\tools\hudson.tasks.Maven_MavenInstallation\MAVEN_HOME\bin\mvn.cmd test && exit %ERRORLEVEL%"
WARNING: A terminally deprecated method in sun.misc.Unsafe has been called
WARNING: sun.misc.Unsafe::staticFieldBase has been called by com.google.inject.internal.aop.HiddenClassDefiner
(file:C:/ProgramData/Jenkins/.jenkins/tools/hudson.tasks.Maven_MavenInstallation/MAVEN_HOME/lib/guice-5.1.0-classes.jar)
WARNING: Please consider reporting this to the maintainers of class com.google.inject.internal.aop.HiddenClassDefiner
WARNING: sun.misc.Unsafe::staticFieldBase will be removed in a future release
[INFO] Scanning for projects...
[INFO]
[INFO] -----< maven_web:web_maven >-----
[INFO] Building web_maven Maven Webapp 0.0.1-SNAPSHOT
[INFO]   from pom.xml
[INFO] ----- [ war ] -----
[INFO]
```

At the bottom of the console output, the final summary is displayed:

```
[INFO] --- compiler:3.13.0:compile (default-compile) @ web_maven ---
[INFO] No sources to compile
[INFO]
[INFO] --- resources:3.3.1:testResources (default-testResources) @ web_maven ---
[INFO] skip non existing resourceDirectory C:\ProgramData\Jenkins\.jenkins\workspace\MavenWeb_Test\src\test\resources
[INFO]
[INFO] --- compiler:3.13.0:testCompile (default-testCompile) @ web_maven ---
[INFO] No sources to compile
[INFO]
[INFO] --- surefire:3.3.0:test (default-test) @ web_maven ---
[INFO] No tests to run.
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 1.569 s
[INFO] Finished at: 2025-09-23T20:36:10+05:30
[INFO] -----
Archiving artifacts
Triggering a new build of MavenWeb_Deploy
Finished: SUCCESS
```

III. Questions on Jenkins

1. What is Jenkins primarily used for?

Jenkins is primarily used for continuous integration and continuous delivery (CI/CD) to automate the building, testing, and deployment of software projects.

2. What is feature of Jenkins?

One key feature of Jenkins is its plugin-based architecture, which allows it to integrate with many tools and customize workflows.

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

Jenkins runs on port 8080 by default.

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

Jenkins can be integrated with version control systems like Git, GitHub, GitLab, Bitbucket, and Subversion (SVN).

5. What is the purpose of Jenkins plugins?

Jenkins plugins extend its functionality by allowing integrations with build tools, test frameworks, version control systems, cloud platforms, etc.

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

A Freestyle project is best for simple or one-off tasks.

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

Use the “**Credentials**” plugin to securely store and manage API keys, passwords, and other secrets.

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

Blue Ocean is a modern UI for Jenkins that provides a user-friendly interface for visualizing CI/CD pipelines.

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

Blue Ocean is a modern UI for Jenkins that provides a user-friendly interface for visualizing CI/CD pipelines.

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

The Jenkins agents (or nodes), managed by the Jenkins master (controller), allow for distributed builds.

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.

Plugin	Purpose
Git Plugin	Integrates Git repositories for source control.
Pipeline Plugin	Enables defining complex CI/CD workflows as code (Jenkinsfile).
Docker Plugin	Supports building, deploying, and running Docker containers.
Kubernetes Plugin	Integrates Jenkins with Kubernetes for dynamic agent provisioning.
Slack Notification Plugin	Sends pipeline build notifications to Slack for team visibility.

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?

Steps:

1. Go to Manage Jenkins > Plugin Manager.
2. Open the Available tab.
3. Search for the desired plugin.
4. Check the box and click Install without restart or Download now and install after restart.

Considerations:

- Check plugin compatibility with your Jenkins version.
- Review plugin dependencies before installation.
- Be aware of plugin update logs for breaking changes.
- Test in a staging environment if possible.

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?

Steps:

5. Go to Manage Jenkins > Plugin Manager.
6. Open the Available tab.
7. Search for the desired plugin.
8. Check the box and click Install without restart or Download now and install after restart.

Considerations:

- Check plugin compatibility with your Jenkins version.
- Review plugin dependencies before installation.
- Be aware of plugin update logs for breaking changes.
- Test in a staging environment if possible.

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?

Steps for Git Plugin:

1. Go to Manage Jenkins > Global Tool Configuration.
2. Scroll to Git section.
3. Add a Git executable path or install Git automatically.
4. In your job/pipeline, use the SCM section to set the repository URL and credentials.

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?

Common issues:

- Dependency conflicts between plugins.
- Incompatible plugin versions with Jenkins core.
- Performance degradation due to too many plugins.

- Plugins failing after Jenkins upgrade.
Troubleshooting tips:
 - Check Jenkins logs (Manage Jenkins > System Log).
 - Use the Plugin Manager to identify failed or outdated plugins.
 - Review the plugin's GitHub issues or changelog.
 - Use safe restart after plugin updates.
 - Restore from backup if plugin breaks the instance.

Azeema firdous

23BD1A05A4

CSE-F