

## Week 8: Jenkins Automation

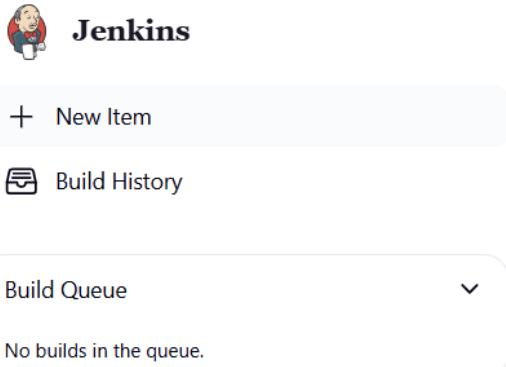
- I. Hands-on practice on manual creation of Jenkins pipeline using Maven projects from Github
- II. Create the job and build the pipeline for maven-java and maven-web project.
- III. Questions on Jenkins
- IV. Upload the Screenshots.

### I. Steps for MavenJava Automation:

Maven Java Automation Steps:

#### **Step 1: Open Jenkins (localhost:8080)**

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



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

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

——— Click "OK"

A screenshot of the 'New Item' configuration page. The 'Item name' field contains 'Lakshmitha\_Jenkins\_Maven\_Build'. The 'Select an item type' section shows 'Freestyle project' selected, with a detailed 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 like 'Maven project', 'Pipeline', 'Multi-configuration project', and 'Folder' are also listed. At the bottom is an 'OK' button.

## └─ Configure the project:

  └─ **Description:** "Java Build demo"

  └─ **Source Code Management:**

    └─ Git repository URL: [GitMavenJava repo URL]

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

The screenshot shows the Jenkins configuration interface for a job named 'Lakshmitha\_Jenkins\_Maven\_Build'. The 'Source Code Management' section is active, showing a Git repository configured with URL 'https://github.com/LakshmithaReddy1807/Lakshmitha-Jenkins.git' and credentials 'LakshmithaReddy1807/\*\*\*\*\*'. The 'Build Steps' section is also visible below, containing two Maven build steps.

## └─ 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

Build Steps

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

Advanced

Add build step

Save Apply

## └─ 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

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

Build other projects

Projects to build

Lakshmitha\_Jenkins\_Maven\_Test

No such project 'Lakshmitha\_Jenkins\_Maven\_Test'. Did you mean 'Lakshmitha\_Jenkins\_Build'?

Trigger only if build is stable

Trigger even if the build is unstable

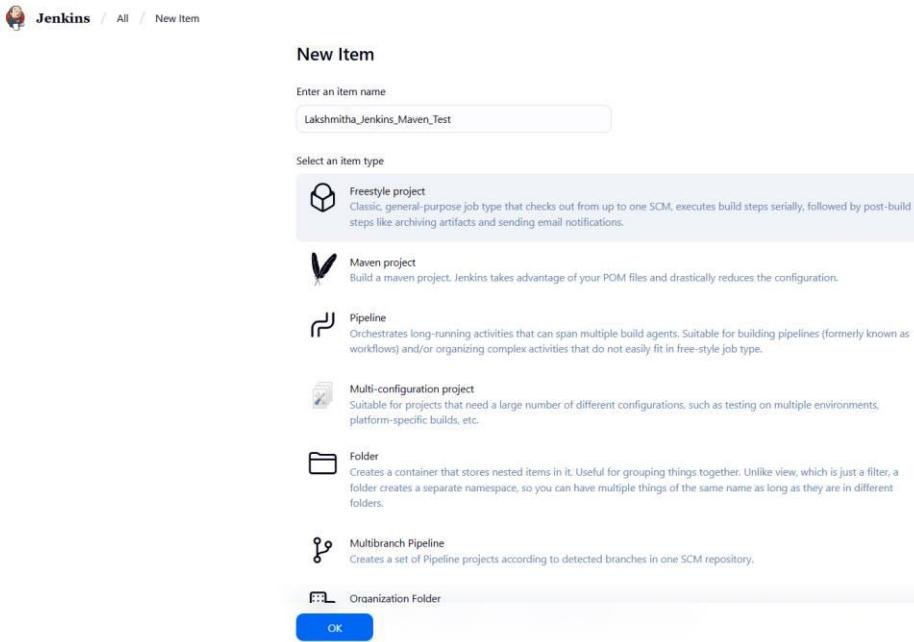
Trigger even if the build fails

Add post-build action

Save Apply

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

- └─ Enter project name (e.g., MavenJava\_Test)
- └─ Click "OK"



## └─ Configure the project:

- └─ **Description:** "Test demo"
- └─ **Build Environment:**
  - └─ Check: "Delete the workspace before build starts"
- └─ **Add Build Step** -> "Copy artifacts from another project"
  - └─ Project name: MavenJava\_Build
  - └─ Build: Stable build only // tick at this
  - └─ Artifacts to copy: \*\*/\*

The screenshot shows the Jenkins configuration page for a job named "Lakshmitha\_Jenkins\_Maven\_Test". The "Configure" section is open, and the "Build Steps" tab is selected. In the "Environment" section, there is an "Advanced" dropdown menu with several options like "Delete workspace before build starts", "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". In the "Build Steps" section, there is a "Copy artifacts from another project" step. It has fields for "Project name" (set to "Lakshmitha\_Jenkins\_Maven\_Build") and "Which build" (set to "Latest successful build"). There is also a checkbox for "Stable build only". Below these, there are fields for "Artifacts to copy" (containing "\*/\*") and "Artifacts not to copy".

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

  └─ Maven version: MAVEN\_HOME

  └─ Goals: test

  └─ Post-build Actions:

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

  └─ Files to archive: \*\*/\*

  └─ Apply and Save

The screenshot shows the Jenkins configuration page for the same job. The "Configure" section is open, and the "Build Steps" tab is selected. Under "Build Steps", there is a step titled "Invoke top-level Maven targets" with "Maven Version" set to "MAVEN\_HOME" and "Goals" set to "test". Below this, there is an "Advanced" dropdown menu. At the bottom of the "Build Steps" section, there is a "Add build step" button. In the "Post-build Actions" section, there is a step titled "Archive the artifacts" with "Files to archive" set to "\*/\*". Below this, there is an "Advanced" dropdown menu. At the bottom of the "Post-build Actions" section, there is a "Add post-build action" button. At the very bottom of the page, there are "Save" and "Apply" buttons.

## └─ 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**

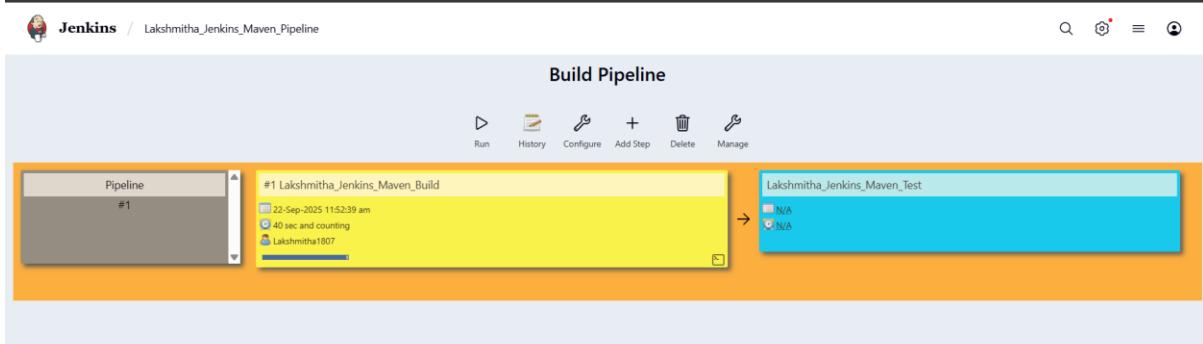
The screenshot shows the Jenkins 'New view' creation interface. At the top, there's a navigation bar with a Jenkins logo and the text 'Jenkins / New view'. Below it, there are two tabs: '+ New Item' and 'Build History'. The main area is titled 'New view' and contains a 'Name' field with the value 'Lakshmitha\_Jenkins\_Maven\_Pipeline'. Under the 'Type' section, a radio button for 'Build Pipeline View' is selected, with a tooltip explaining it shows the jobs in a build pipeline view. There are also two other options: 'List View' and 'My View'. At the bottom right is a blue 'Create' button.

## └─ Pipeline Flow:

- └─ **Layout:** Based on upstream/downstream relationship
- └─ Initial job: MavenJava\_Build
- └─ Apply and Save OK

The screenshot shows the 'Pipeline Flow' configuration page. It starts with a 'Build Pipeline View Title' input field. Below that is a 'Pipeline Flow' section with a 'Layout' dropdown set to 'Based on upstream/downstream relationship'. A tooltip explains this layout mode derives the pipeline structure based on trigger relationships. Under 'Upstream / downstream config', there's a 'Select Initial Job' dropdown containing 'Lakshmitha\_Jenkins\_Maven\_Build'. At the bottom, there's a 'Trigger Options' section with a 'Build Cards' dropdown set to 'Standard build card'.

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



└— Click on the trigger to run the pipeline

The screenshot shows the Jenkins Console Output for build step #2 of 'Lakshmitha\_Jenkins\_Maven\_Build'. The output log is as follows:

```
api/1.9.18/maven-resolver-api-1.9.18.jar (157 KB at 418 KB/s)
[INFO] Installing C:\ProgramData\Jenkins\.jenkins\workspace\Lakshmitha_Jenkins_Maven_Build\pom.xml to
C:\WINDOWS\system32\config\systemprofile\.m2\repository\KMIT\Lakshmitha_Maven_Jenkins\0.0.1-
SNAPSHOT\Lakshmitha_Maven_Jenkins-0.0.1-SNAPSHOT.pom
[INFO] Installing
C:\programData\jenkins\.jenkins\workspace\Lakshmitha_Jenkins_Maven_Build\target\Lakshmitha_Maven_Jenkins-
0.0.1-SNAPSHOT.jar to
C:\WINDOWS\system32\config\systemprofile\.m2\repository\KMIT\Lakshmitha_Maven_Jenkins\0.0.1-
SNAPSHOT\Lakshmitha_Maven_Jenkins-0.0.1-SNAPSHOT.jar
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time:  36.263 s
[INFO] Finished at: 2025-09-22T12:08:34+05:30
[INFO] -----
Archiving artifacts
Triggering a new build of Lakshmitha\_Jenkins\_Maven\_Test
Finished: SUCCESS
```

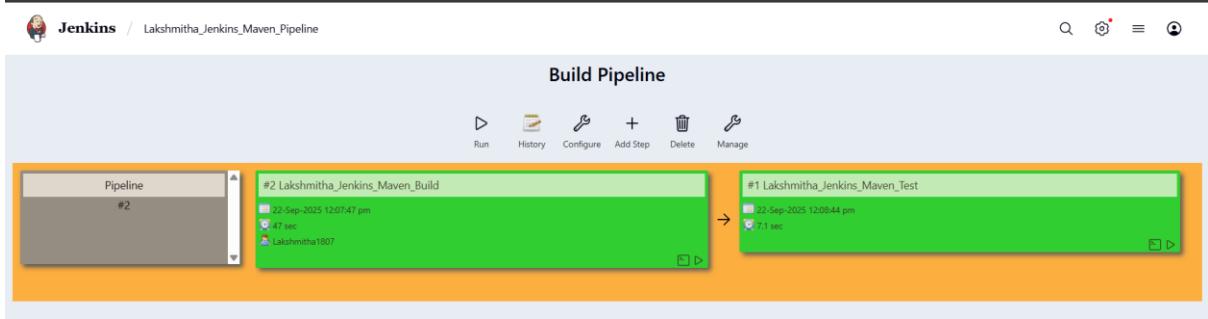
└— click on the small black box to open the console to check if the build is success

The screenshot shows the Jenkins Console Output for build step #1 of 'Lakshmitha\_Jenkins\_Maven\_Test'. The output log is as follows:

```
[INFO] -----
[INFO] T E S T S
[INFO] -----
[INFO] Running KMIT.Lakshmitha_Maven_Jenkins.AppTest
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.040 s -- in
KMIT.Lakshmitha_Maven_Jenkins.AppTest
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time:  3.589 s
[INFO] Finished at: 2025-09-22T12:08:51+05:30
[INFO] -----
Archiving artifacts
Finished: SUCCESS
```

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



## II. Maven Web Automation Steps:

└─ Step 1: Open Jenkins (localhost:8080)

  └─ Click on "New Item" (left side menu)



+ New Item

Build History

Build Queue

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

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

  └─ Click "OK"

Jenkins / All / New Item

## New Item

Enter an item name: Lakshmitha\_Jenkins\_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.
- Maven project**: Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.
- 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 Build demo"

### Source Code Management:

Git repository URL: [GitMavenWeb repo URL]

Branches to build: \*/Main or master

#### 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/LakshmithaReddy1807/Lakshmitha-Maven-Java-WebProject.git



Credentials ?

LakshmithaReddy1807/\*\*\*\*\*

+ Add

Advanced ▼

Add Repository

Branches to build ?

Branch Specifier (blank for 'any') ?

\*/main



Add Branch

**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

### Build Steps

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

The screenshot shows a build configuration interface with two build steps defined. Each step is a card with a header 'Invoke top-level Maven targets'. The first step's 'Goals' field contains 'clean'. The second step's 'Goals' field contains 'install'. Both steps have 'MAVEN\_HOME' selected in their 'Maven Version' dropdowns. At the bottom, there are buttons for 'Add build step', 'Save', and 'Apply'.

## └─ 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

  └─ 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 contains two main sections: 'Archive the artifacts' and 'Build other projects'.

**Archive the artifacts:** Set to archive all files (\*\*/\*). An 'Advanced' dropdown is available.

**Build other projects:** Set to build 'Lakshmitha\_Jenkins\_MavenWeb\_Test'. A note says 'No such project 'Lakshmitha\_Jenkins\_MavenWeb\_Test''. Trigger options:  Trigger only if build is stable,  Trigger even if the build is unstable,  Trigger even if the build fails.

An 'Add post-build action' dropdown is present, and there are 'Save' and 'Apply' buttons at the bottom.

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

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

└─ Click "OK"

### New Item

Enter an item name

Lakshmitha\_Jenkins\_MavenWeb\_Test

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.



**Maven project**  
Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.



**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

## └─ Configure the project:

  └─ Description: "Test demo"

  └─ Build Environment:

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

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

    └─ Project name: MavenWeb\_Build

    └─ Build: Stable build only

    └─ Artifacts to copy: \*\*/\*

## 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 ?

## Build Steps

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

### ≡ Copy artifacts from another project

Project name ?

Lakshmitha\_Jenkins\_MavenWeb\_Build

Which build ?

Latest successful build

Stable build only

Artifacts to copy ?

\*\*/\*

Artifacts not to copy ?

Save

Apply

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

  └─ Maven version: MAVEN\_HOME

    └─ Goals: test

└─ **Post-build Actions:**

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

  └─ Files to archive: \*\*/\*

The screenshot shows the Jenkins configuration interface for a job. At the top, there's a section for 'Invoke top-level Maven targets' with fields for 'Maven Version' (set to 'MAVEN\_HOME') and 'Goals' (set to 'test'). Below this is an 'Advanced' dropdown and a 'Add build step' button. The next section is 'Post-build Actions' with a sub-section for 'Archive the artifacts'. It has a field for 'Files to archive' containing '\*\*/\*' and an 'Advanced' dropdown. There's also a 'Add post-build action' button.

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

  └─ Projects to build: MavenWeb\_Deploy

  └─ Apply and Save

The screenshot shows the Jenkins configuration interface for a job. It's under the 'Build other projects' section. The 'Projects to build' field contains 'Lakshmitha\_Jenkins\_MavenWeb\_Deploy'. A warning message says 'No such project 'Lakshmitha\_Jenkins\_MavenWeb\_Deploy'. Did you mean 'Lakshmitha\_Jenkins\_MavenWeb\_Build''? Below 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 4: Create Freestyle Project (e.g., MavenWeb\_Deploy)

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

  └─ Click "OK"

### New Item

Enter an item name

Lakshmitha\_Jenkins\_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.



Maven project

Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.



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

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:

OK

## └─ Configure the project:

  └─ Description: "Web Code Deployment"

  └─ Build Environment:

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

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

    └─ Project name: MavenWeb\_Test

    └─ Build: Stable build only

    └─ Artifacts to copy: \*\*/\*

## 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 ?

## Build Steps

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

### Copy artifacts from another project

Project name ?

Lakshmitha\_Jenkins\_MavenWeb\_Test

Which build ?

Latest successful build

- Stable build only

Artifacts to copy ?

\*\*/\*

Save

Apply

## 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.

Deploy war/ear to a container

WAR/EAR files ?

Context path ?

Containers

Tomcat 9.x Remote

Credentials  + Add

Tomcat URL ?

Advanced ▼

Add Container ▼

Deploy on failure

Save Apply

## └─ Step 5: Create Pipeline View for MavenWeb

- └─ Click "+" beside "All" on the dashboard
- └─ Enter name: MavenWeb\_Pipeline
- └─ Select "Build pipeline view"

Jenkins / New view

+ New Item

Build History

Build Queue  
No builds in the queue.

Build Executor Status 0/2

New view

Name

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
- └─ Initial job: MavenWeb\_Build
- └─ Apply and Save

## Edit View

### Name

Lakshmitha\_Jenkins\_MavenWeb\_pipeline

### Description

Describe the purpose of this view.

[Plain text](#) [Preview](#)

### Build Pipeline View Title

### Pipeline Flow

#### Layout

Based on upstream/downstream relationship

This layout mode derives the pipeline structure based on the upstream/downstream trigger extension.

#### Upstream / downstream config

Select Initial Job [?](#)

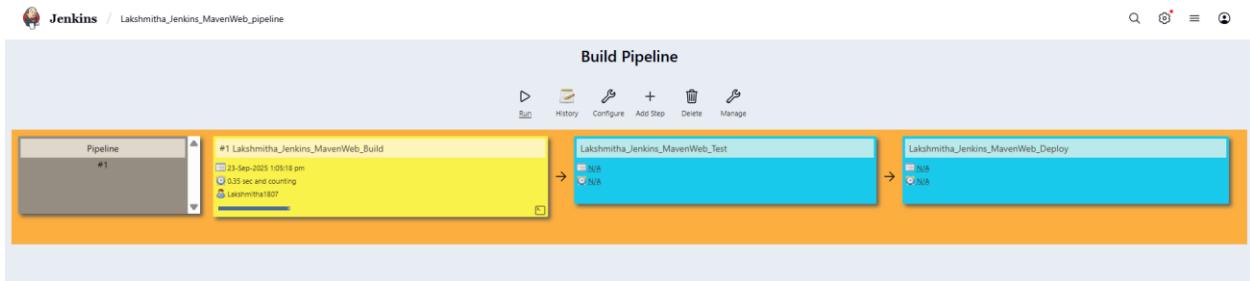
Lakshmitha\_Jenkins\_MavenWeb\_Build

[Save](#)

[Apply](#)

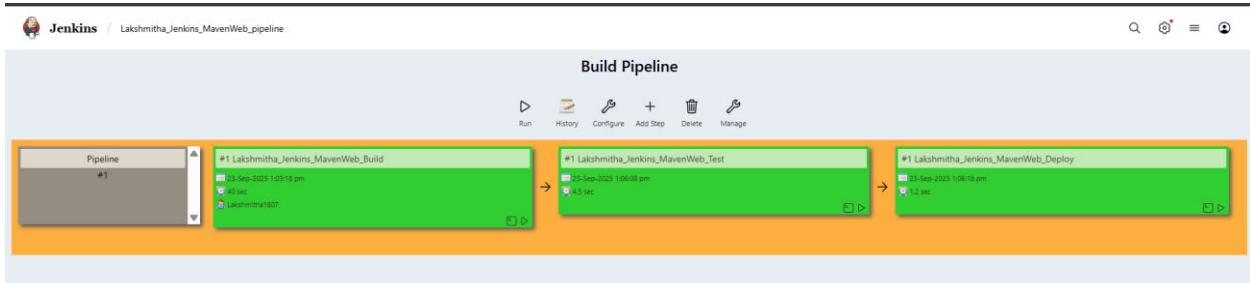
## └─ Step 6: Run the Pipeline and Check Output

└─ 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



o

└─ Open Tomcat homepage in another tab  
└─ Click on the "/webpath" option under the manager app

Note:

1. It ask for user credentials for login ,provide the credentials of tomcat.
2. It provide the page with out project name which is highlighted.
3. After clicking on our project we can see output.

## III. Questions on Jenkins

1. What is Jenkins primarily used for?

Jenkins is primarily used for **Continuous Integration (CI) and Continuous Delivery/Deployment (CD)** to automate building, testing, and deploying applications.

2. What is feature of Jenkins?

**Extensibility through plugins** (over 1,800 available).

**Pipeline as code** (using Jenkinsfile).

**Distributed builds** across multiple nodes.

**Integration with version control systems**.

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

8080

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

Git, Github, BitBucket, SVN

5. What is the purpose of Jenkins plugins?

Plugins extend Jenkins' core functionality by adding **new integrations, build steps, UI enhancements, SCM support, and pipeline features**.

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

FreeStyle Job

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

Using the **Credentials Plugin**, which stores secrets securely and injects them into builds.

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

A **modern, user-friendly UI** for Jenkins pipelines with visualization of stages and steps.

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

A **modern, user-friendly UI** for Jenkins pipelines with visualization of stages and steps.

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

**Jenkins Master-Agent architecture** (agents handle 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.

**Maven Integration Plugin** – Automates building, testing, and packaging microservices using Maven.

**Build Pipeline Plugin** – Visualizes and manages multi-stage CI/CD workflows.

**Pipeline Utility Steps Plugin** – Provides extra steps for handling files, JSON, and YAML in pipelines.

**Copy Artifacts Plugin** – Shares build outputs between dependent Jenkins jobs.

**Deploy to Container Plugin** – Automates deployment of applications to servers like Tomcat/JBoss.

**Git SCM / Blue Ocean Plugin** – Integrates Git repos and offers a modern UI for pipeline visualization.

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?

Go to **Manage Jenkins → Manage Plugins**.

Search in the **Available** tab.

Select plugin → click **Install without restart** (or with restart if required).

**Considerations:**

Ensure plugin version is **compatible with Jenkins version**.

Check for **dependency requirements**.

Avoid outdated or unmaintained plugins.

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?

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?

- Go to **Manage Jenkins → Global Tool Configuration**.
- Add Git installation path.
- In your pipeline/job, configure Git repository URL and credentials.
- Test connection to ensure Jenkins can access the repo.

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?

**Dependency conflicts** – One plugin may require a different version of another.

**Version incompatibility** – Plugins may not support the installed Jenkins core.

**Security vulnerabilities** – Outdated plugins may have security issues.

**Troubleshooting:**

Check **Jenkins logs** for errors.

Review plugin documentation & compatibility matrix.

Upgrade/downgrade plugin versions carefully.

Use **Jenkins Plugin Manager → Updates** to patch issues.

If broken, uninstall plugin or roll back using a backup.

**Conclusion:** In this week student learnt automating Maven projects through Jenkins.