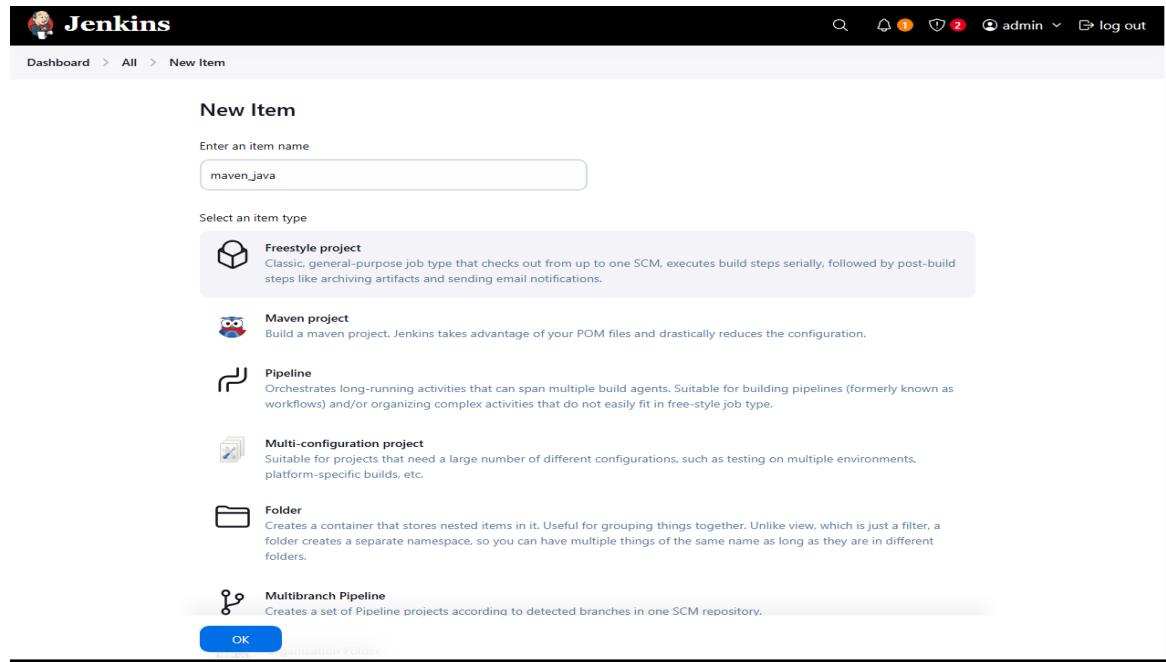


8. Jenkins Automation

Steps for MavenJava Automation

Step 1: Open Jenkins (localhost:8888)

Click on "New Item" (left side menu) and name it as maven_java > select freestyle project > click on "OK"



Step 2: Configuration of maven_java project

Give the description

The screenshot shows the Jenkins configuration page for a job named "Mavenjava". The "General" tab is selected. The "Description" field contains "Java Build demo". There are two checkboxes at the bottom: "Discard old builds" and "GitHub project", both of which are unchecked.

In the source code management select git and give the git repo link

The screenshot shows the Jenkins configuration page for a job named "Mavenjava Config [Jenkins]". The "Source Code Management" tab is selected. The "Git" option is selected, and the "Repository URL" field contains "https://github.com/SarvikaSomishetty/eclipse-maven-projects.git".

In the build steps click on add build step > give maven version as MAVEN_HOME > select invoke top-level maven targets > goals as clean

The screenshot shows the Jenkins configuration page for a job named "Mavenjava". The left sidebar lists various configuration sections: General, Source Code Management, Triggers, Environment, Build Steps (which is selected and highlighted in grey), and Post-build Actions. The main content area is titled "Configure" and shows the "Build Steps" section. It contains two entries under the heading "Invoke top-level Maven targets". Both entries have "MAVEN_HOME" selected in the "Maven Version" dropdown and "clean" in the "Goals" input field. There are "Save" and "Apply" buttons at the bottom of the build step list.

In the build steps click on add build step > give maven version as MAVEN_HOME > select invoke top-level maven targets > goals as install

The screenshot shows the Jenkins configuration interface for a job named 'Mavenjava'. The left sidebar lists configuration sections: General, Source Code Management, Triggers, Environment, Build Steps (which is selected), and Post-build Actions. The main area displays a 'Build Steps' section with a 'Goals' field containing 'clean' and an 'Advanced' dropdown. Below this is a 'Post-build Actions' section with a note about defining actions like notifications or artifact archiving. At the bottom are 'Save' and 'Apply' buttons.

In the post build actions > click on add post build action > select the archive the artifacts > in the file to archive give “**/*”

For the second post build action,

In the post build actions > click on add post build action >select build other projects > give projects to build as MavenJava_Test

Click on apply and save

The screenshot shows the Jenkins configuration page for the 'Mavenjava' job. The 'Post-build Actions' section is expanded, displaying two actions:

- Archive the artifacts**: Set to archive all files (**/*).
- Build other projects**: Set to build 'MavenJava_Test'. The trigger option 'Trigger only if build is stable' is selected.

At the bottom, there are 'Save' and 'Apply' buttons.

If the build is success:

The screenshot shows a web browser window with three tabs open: 'maven_web_build [Jenkins]', 'Edigirala-Neksha/se-lab-intern...', and 'SSL connection issue fix'. The main content area is the Jenkins 'maven_web_build' project page. At the top, there's a navigation bar with a Jenkins logo, user information ('Verify it's you'), and a 'log out' button. Below the navigation is a breadcrumb trail: 'Dashboard > maven_web_build >'. The main content area has a title 'maven_web_build' with a green checkmark icon and the subtitle 'Status'. It includes sections for 'Changes', 'Workspace', 'Build Now', 'Configure', 'Delete Project', and 'Rename'. A 'Last Successful Artifacts' section shows a cube icon and a link to 'maven_web_test'. Another section titled 'Downstream Projects' lists 'maven_web_test' with a green checkmark icon. A 'Permalinks' section provides links to recent builds: 'Last build (#2), 27 min ago', 'Last stable build (#2), 27 min ago', 'Last successful build (#2), 27 min ago', and 'Last completed build (#2), 27 min ago'. On the left, a 'Builds' sidebar shows a table with two rows: '#2 11:43 AM' and '#1 11:42 AM', both marked with green checkmarks. At the bottom right, there are links for 'REST API' and 'Jenkins 2.489'. The taskbar at the bottom of the screen shows the Windows logo, a search bar with 'Type here to search', and icons for various applications like File Explorer, Edge, and Google Chrome. System status icons include battery level, signal strength, and network connectivity. The date and time are shown as '07-10-2025 12:11'.

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

Click on new item > give item name as mavaen_java_test or MavenJava_Test and select free style project and click ok

New Item

Enter an item name
maven_java_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.

OK

Step 4: Configuration of maven_java project

Give the description

MavenJava_Test Config [Jenkins]

localhost:8888/job/MavenJava_Test/configure

Verify it's you

Configure General

Enabled

General

Description
Test demo

Plain text Preview

Discard old builds ?

GitHub project

In the source code management select none and in environment select “delete workspace before build starts”

The screenshot shows the Jenkins configuration page for the 'MavenJava_Test' job. Under 'Source Code Management', 'None' is selected. Under 'Environment', the checkbox 'Delete workspace before build starts' is checked. Other environment options like 'Use secret text(s) or file(s)', 'Provide Configuration files', 'Add timestamps to the Console Output', and 'Inspect build log for published build scans' are unchecked. At the bottom, there are 'Save' and 'Apply' buttons.

In the build steps> select add a build step> select “copy artifacts from another project” > give project name as Maven java and artifacts to copy as **/*

The screenshot shows the Jenkins configuration page for the 'MavenJava_Test' job. Under 'Build Steps', a 'Copy artifacts from another project' step is added. The 'Project name' is set to 'Mavenjava'. The 'Which build' dropdown is set to 'Latest successful build' with the 'Stable build only' checkbox checked. The 'Artifacts to copy' field contains '**/*'. The 'Target directory' field is empty. There are checkboxes for 'Flatten directories', 'Optional', 'Fingerprint Artifacts', and 'Include Build Number', with 'Fingerprint Artifacts' checked. At the bottom, there are 'Save' and 'Apply' buttons.

In the post build actions> select archive the artifacts and enter files as **/*

Click on apply and save

The screenshot shows the Jenkins configuration page for the 'MavenJava_Test' job. In the 'Post-build Actions' section, there is a step titled 'Archive the artifacts' with the pattern '**/*' entered. Below the configuration area, there are 'Save' and 'Apply' buttons.

In the dashboard you will find MavenJava and MavenJava_Test

The screenshot shows the Jenkins dashboard with several jobs listed in a table:

S	W	Name	Last Success	Last Failure	Last Duration
🔴	☁️	INTERNAL_JAVA	9 mo 3 days #34	40 sec #15454	0.67 sec
🟢	☀️	Mavenjava	13 days #2	N/A	11 sec
🟢	☀️	MavenJava_Test	13 days #3	N/A	3.4 sec
🔴	☁️	new	9 mo 3 days #3	13 days #4	31 sec
🟢	☀️	web_build	9 mo 9 days #8	N/A	8.2 sec
🔴	☁️	web_deploy	N/A	9 mo 9 days #15	0.31 sec
🟢	☀️	web_test	9 mo 9 days #12	N/A	3.4 sec

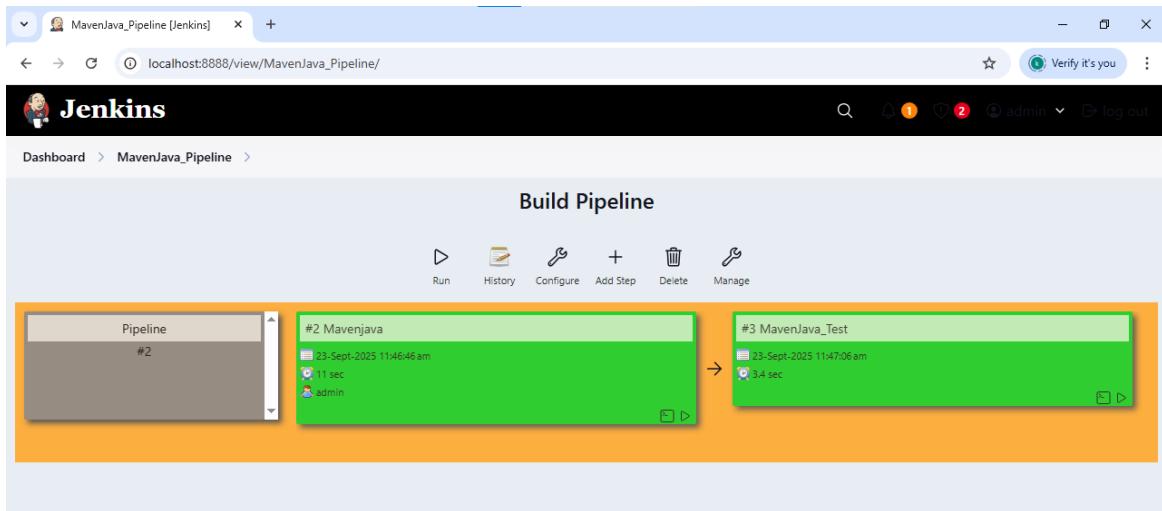
If you open the MavenJava file the following will be shown in case on no errors

The screenshot shows the Jenkins interface for the 'Mavenjava' job. The top navigation bar includes links for 'Dashboard', 'Mavenjava', 'Status', 'Changes', 'Workspace', 'Build Now', 'Configure', 'Delete Project', and 'Rename'. The main content area is titled 'Mavenjava' with a green checkmark icon. It displays the 'Java Build demo' configuration. On the left, there's a sidebar with 'Builds' (listing #2 from September 23, 2025, at 11:46 AM and #1 at 11:45 AM) and 'Permalinks' (link to last build). The right side shows 'Last Successful Artifacts' with a list of files and their sizes, such as .classpath (1.65 KiB), .project (1.06 KiB), and Dockerfile (131 B). Below this is a section for 'Downstream Projects' with a link to 'MavenJava_Test'. A 'Permalinks' section at the bottom lists the last build (#2) and last stable build (#2).

If you open the MavenJava_Test file the following will be shown in case on no errors

The screenshot shows the Jenkins interface for the 'MavenJava_Test' job. The top navigation bar includes links for 'Dashboard', 'MavenJava_Test', 'Status', 'Changes', 'Workspace', 'Build Now', 'Configure', 'Delete Project', and 'Rename'. The main content area is titled 'MavenJava_Test' with a green checkmark icon. It displays the 'Test demo' configuration. On the left, there's a sidebar with 'Builds' (listing #3 from September 23, 2025, at 11:47 AM, #2 at 11:46 AM, and #1 at 11:45 AM) and 'Permalinks' (link to 'Mavenjava'). The right side shows 'Last Successful Artifacts' with a list of files and their sizes, identical to the MavenJava job. Below this is a section for 'Upstream Projects' with a link to 'Mavenjava'. A 'Permalinks' section at the bottom lists the last build (#3), last stable build (#3), and last successful build (#3), all dated 13 days ago.

MavenJava_pipeline

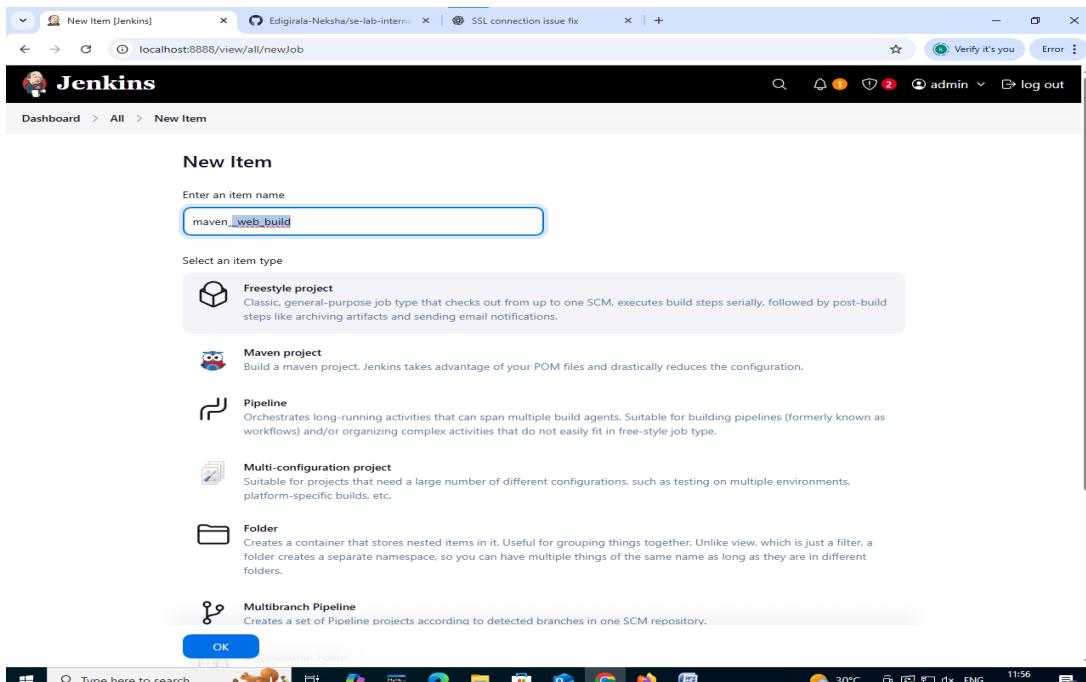


II. Maven Web Automation Steps:

Create Freestyle Project (e.g., MavenWeb_Build)

Step 1: Open Jenkins (localhost:8888)

Click on "New Item" (left side menu) and name it as maven_web_build > select freestyle project > click on "OK"



Step 2: Configuration of maven_web_build project

Give the description

The screenshot shows the Jenkins configuration interface for the 'maven_web_build' project. The 'General' tab is selected. In the 'Description' field, the text 'web build demo' is entered. The 'Enabled' toggle switch is turned on. On the left sidebar, there are tabs for General, Source Code Management, Triggers, Environment, Build Steps, and Post-build Actions. Under 'Source Code Management', there is a note: 'Connect and manage your code repository to automatically pull the latest code for your builds.' At the bottom, there are 'Save' and 'Apply' buttons.

In the source code management select git and give the git repo link

The screenshot shows the Jenkins configuration interface for a job named "maven_web_build". The left sidebar lists several configuration sections: General, Source Code Management (selected), Triggers, Environment, Build Steps, and Post-build Actions. The main content area is titled "Source Code Management" and contains instructions to "Connect and manage your code repository to automatically pull the latest code for your builds." A radio button is selected for "Git", and the "Repositories" section is expanded. It shows a "Repository URL" input field containing "https://github.com/Edigirala-Neksha/se-lab-internal-1.git" and a "Credentials" dropdown set to "- none -". Below these are "Add Repository" and "Advanced" buttons. Another section, "Branches to build", is also expanded, showing a "Branch Specifier (blank for 'any')" input field containing "*/main" and an "Add Branch" button. At the bottom of the configuration area are "Save" and "Apply" buttons. The browser's address bar shows the URL "localhost:8888/job/maven_web_build/configure". The taskbar at the bottom of the screen includes icons for File Explorer, Edge, File, Mail, Google Chrome, and Firefox, along with system status indicators like battery level, temperature (30°C), and date/time (07-10-2025).

In the build steps click on add build step > give maven version as MAVEN_HOME > select invoke top-level maven targets > goals as clean

For the second build step,

In the build steps click on add build step > give maven version as MAVEN_HOME > select invoke top-level maven targets > goals as install

The screenshot shows the Jenkins configuration page for the 'maven_web_build' job. The left sidebar lists 'General', 'Source Code Management', 'Triggers', 'Environment', 'Build Steps' (which is selected), and 'Post-build Actions'. The main area is titled 'Build Steps' with the sub-instruction 'Automate your build process with ordered tasks like code compilation, testing, and deployment.' It displays two 'Invoke top-level Maven targets' steps. The first step has 'MAVEN_HOME' selected in the Maven Version dropdown and 'clean' in the Goals dropdown. The second step also has 'MAVEN_HOME' selected in the Maven Version dropdown and 'install' in the Goals dropdown. Both steps have an 'Advanced' dropdown menu. At the bottom, there are 'Save' and 'Apply' buttons, and a task list with four items.

In the post build actions > click on add post build action > select the archive the artifacts > in the file to archive give “**/*”

For the second post build action,

In the post build actions > click on add post build action > select build other projects > give projects to build as maven_web_test

Click on apply and save

The screenshot shows the Jenkins configuration page for the 'maven_web_build' job. The 'Post-build Actions' section is active. It contains two actions:

- Archive the artifacts**: Set to archive files matching the pattern '**/*'. An 'Advanced' dropdown is visible.
- Build other projects**: Set to build the project 'maven_web_test'. Trigger options include 'Trigger only if build is stable' (selected), 'Trigger even if the build is unstable', and 'Trigger even if the build fails'.

At the bottom, there are 'Save' and 'Apply' buttons. The status bar at the bottom right shows 'REST API' and 'Jenkins 2.489'.

Create Freestyle Project (e.g., MavenWeb_Test):

Step 1: Open Jenkins (localhost:8888)

Click on "New Item" (left side menu) and name it as maven_web_test > select freestyle project > click on "OK"

New Item

Enter an item name
maven_web_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.

OK

Step 2: Configuration of maven_web_test project

Give the description

Configure

General

Enabled

Description
test demo

Plain text [Preview](#)

Discard old builds ?

GitHub project

In the source code management select none and in environment select “delete workspace before build starts”

The screenshot shows the Jenkins configuration page for the 'maven_web_test' job. The 'Source Code Management' section is set to 'None'. The 'Environment' section has the 'Delete workspace before build starts' checkbox checked. The 'Triggers' and 'Build Steps' sections are also visible.

Source Code Management
Connect and manage your code repository to automatically pull the latest code for your builds.
 None
 Git ?

Triggers
Set up automated actions that start your build based on specific events, like code changes or scheduled times.
 Trigger builds remotely (e.g., from scripts) ?
 Build after other projects are built ?
 Build periodically ?
 GitHub hook trigger for GITScm polling ?
 Poll SCM ?

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) ?
 Provide Configuration files ?
 Add timestamps to the Console Output
 Inspect build log for published build scans
Terminate a build if it's stuck

Save Apply

In the build steps click on add build step > select copy artifacts from another project > give project name as maven_web_build > give artifacts to copy as **/*

The screenshot shows the Jenkins configuration interface for the 'maven_web_test' job. In the 'Build Steps' section, a 'Copy artifacts from another project' step is selected. The configuration includes:

- Project name: maven_web_build
- Which build: Latest successful build
- Stable build only: Checked
- Artifacts to copy: **/*
- Target directory: (empty)
- Parameter filters: (empty)

At the bottom, there are 'Save' and 'Apply' buttons.

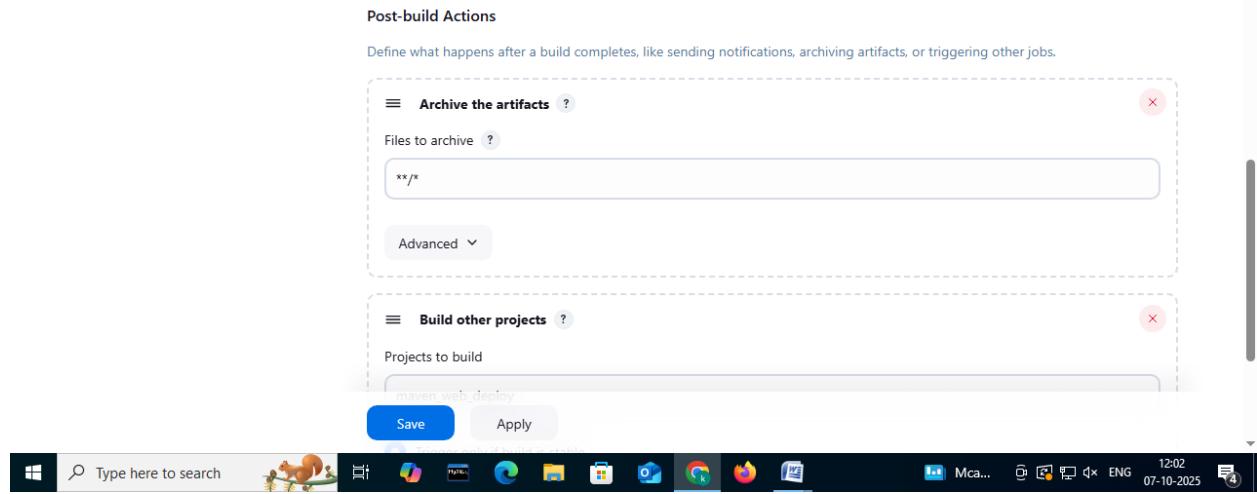
In the build steps click on add build step > give maven version as MAVEN_HOME > select invoke top-level maven targets > goals as test

The screenshot shows the Jenkins configuration interface for the 'maven_web_test' job. In the 'Build Steps' section, an 'Invoke top-level Maven targets' step is selected, with the following configuration:

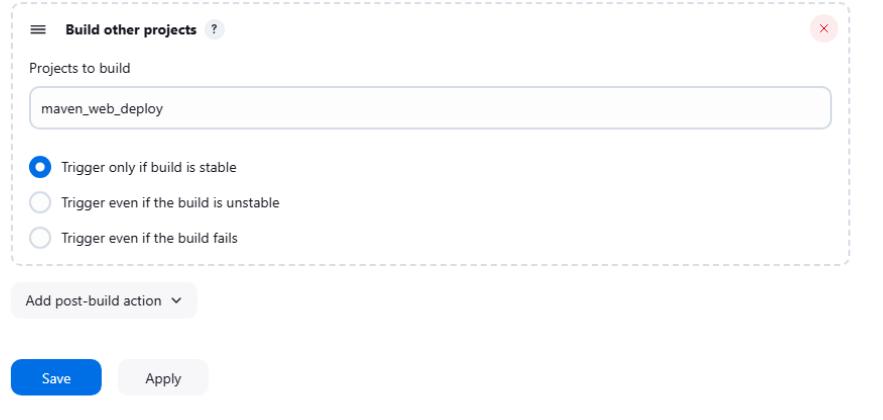
- Maven Version: MAVEN_HOME
- Goals: test

An 'Advanced' dropdown menu is open. At the bottom, there is an 'Add build step' button.

In the post build actions > click on add post build action > select the archive the artifacts > in the file to archive give **/*



In the post build actions > click on add post build action >select build other projects > give name as maven_web_deploy> select “trigger only if build is stable”



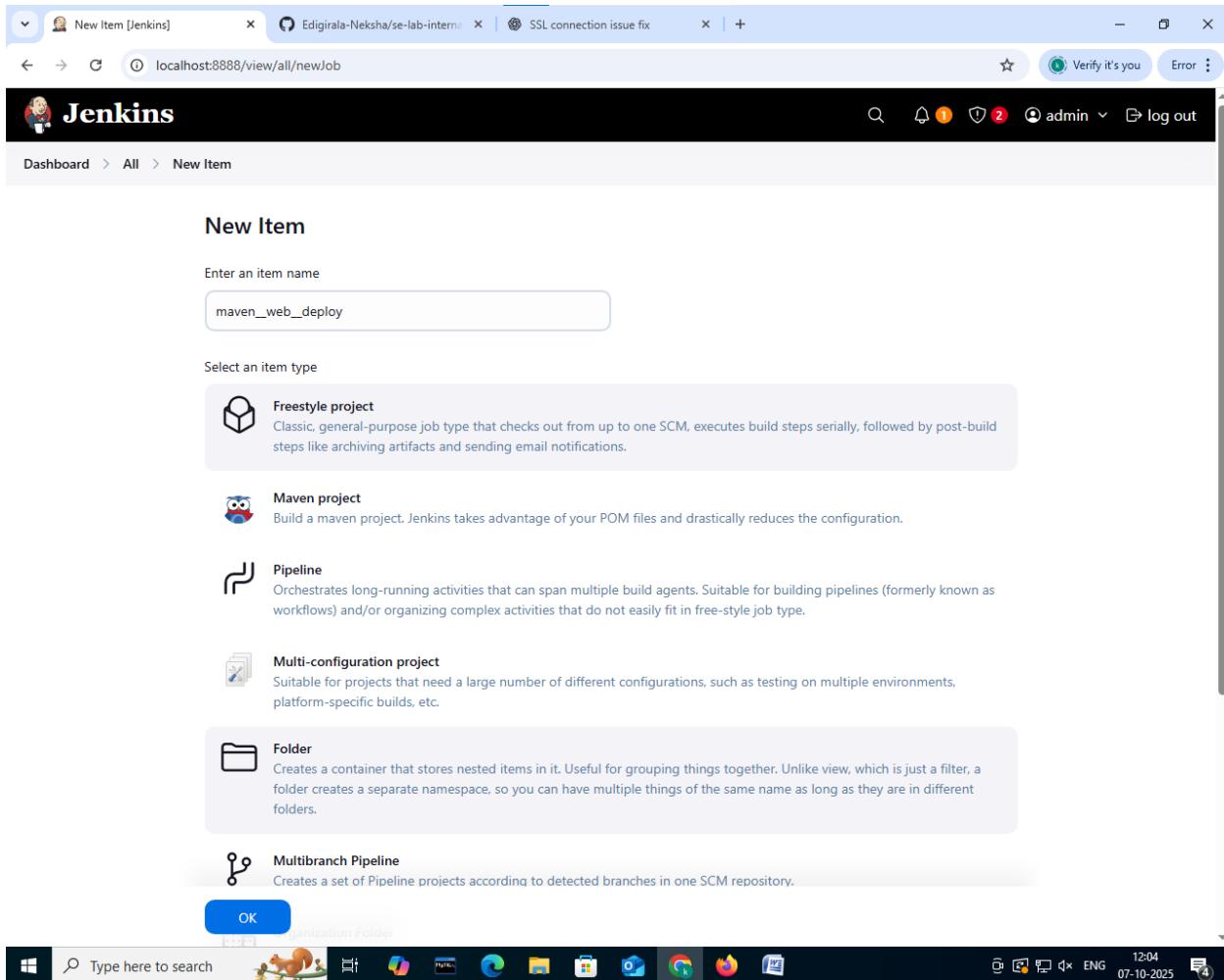
If the build is success:

The screenshot shows a Windows desktop environment. At the top, there is a taskbar with several pinned icons: Start, File Explorer, Task View, Microsoft Edge, File History, Google Chrome, Mozilla Firefox, and FileZilla. To the right of the taskbar, the system tray displays the date (07-10-2025), time (12:37), battery level (30°C), and language (ENG). The main window is a web browser displaying the Jenkins interface for a project named "maven_web_test". The browser tabs include "maven_web_test [Jenkins]", "Edigirala-Neksha/se-lab-intern", "Apache Tomcat/9.0.98", and "Jenkins support for Java 21". The Jenkins page shows the project status as "green" (successful), the last build was #4 (1 min 30 sec ago), and the upstream project "maven_web_build" is also successful. The downstream project "maven_web_deploy" is also successful. The taskbar also shows a search bar with "Type here to search" and a pinned icon for a squirrel.

Create Freestyle Project (e.g., MavenWeb_Deploy):

Step 1: Open Jenkins (localhost:8888)

Click on "New Item" (left side menu) and name it as maven_web_deploy > select freestyle project > click on "OK"



Step 2: Configuration of maven_web_deploy project

Give the description

The screenshot shows the Jenkins configuration page for a job named "maven_web_deploy". The "General" tab is selected. In the "Description" field, the text "deploy demo" is entered. The "Enabled" checkbox is checked. On the left sidebar, there are tabs for General, Source Code Management, Triggers, Environment, Build Steps, and Post-build Actions. Under "Source Code Management", there are several checkboxes: "Discard old builds", "GitHub project", "Permission to Copy Artifact", "This project is parameterized", "Throttle builds", and "Execute concurrent builds if necessary". A "Save" button is visible at the bottom.

In the source code management select none and in environment select “delete workspace before build starts”

The screenshot shows the Jenkins configuration interface for the 'maven_web_deploy' job. The left sidebar lists several tabs: General, Source Code Management (selected), Triggers, Environment, Build Steps, and Post-build Actions. The main content area is titled 'Configure' and contains three sections: 'Source Code Management', 'Triggers', and 'Environment'. In 'Source Code Management', the 'None' option is selected. In 'Triggers', several options are listed, with 'Trigger builds remotely (e.g., from scripts)' checked. In 'Environment', 'Delete workspace before build starts' is checked, and there is an 'Advanced' dropdown menu. At the bottom of the configuration page are 'Save' and 'Apply' buttons. The browser address bar shows 'localhost:8888/job/maven_web_deploy/configure'. The system tray at the bottom right indicates it's 12:07 on 07-10-2025.

In the build steps click on add build step > select copy artifacts from another project > give project name as maven_web_test > give artifacts to copy as **/*

The screenshot shows the Jenkins configuration interface for the 'maven_web_deploy' job. The 'Build Steps' section is active. A 'Copy artifacts from another project' step is defined, pointing to the 'maven_web_test' project and its latest successful build. Artifacts to copy are set to '**/*'. The 'Stable build only' checkbox is checked. Other options like 'Flatten directories' and 'Optional' are available but unchecked. The 'Save' and 'Apply' buttons are at the bottom.

In the post build actions > click on add post build actions > select deploy war/ear to a container > enter war/ear files as **/*.war> context path as webpath > give the credentials and tomcat URL

The screenshot shows the Jenkins configuration interface for a job named "maven_web_deploy". The left sidebar lists various configuration sections: General, Source Code Management, Triggers, Environment, Build Steps, and Post-build Actions. The "Post-build Actions" section is currently selected and highlighted with a grey background.

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: `**/*.war`

Context path: `webpath`

Containers

Tomcat 9.x Remote

Credentials: `admin/*****`

+ Add

Tomcat URL: `https://localhost:8080/`

Advanced ▾

Save Apply

The Jenkins interface is running on a Windows 10 desktop, as indicated by the taskbar at the bottom which includes icons for File Explorer, Edge, and various system status indicators.

If the build is success:

The screenshot shows a Jenkins job page for 'maven_web_deploy'. The top navigation bar includes tabs for 'maven_web_deploy [jenkins]', 'Edigirala-Neksha/se-lab-intern...', 'Apache Tomcat/9.0.98', 'Jenkins support for Java 21', and a '+' button. The URL is 'localhost:8888/job/maven_web_deploy/'. The main content area has a dark header with the Jenkins logo and a green checkmark icon. Below it, the job name 'maven_web_deploy' is displayed with a green checkmark. A status message 'deploy demo' is shown. On the left, a sidebar lists project actions: 'Status', 'Changes', 'Workspace', 'Build Now', 'Configure', 'Delete Project', and 'Rename'. A 'Builds' section shows a table of recent builds, with the most recent one (#13) marked as successful (green circle). To the right, sections for 'Upstream Projects' (listing 'maven_web_test') and 'Permalinks' (listing recent build links) are present. The bottom of the screen shows a Windows taskbar with various icons and system status.

Create Pipeline View for MavenWeb

Click "+" beside "All" on the dashboard and Enter name as maven_web_pipeline

Select type as build pipeline view

The screenshot shows the Jenkins interface for creating a new view. The title bar indicates the URL is `localhost:8888/newView`. The main area is titled "New view". On the left, there's a sidebar with links like "New Item", "Build History", "Project Relationship", "Check File Fingerprint", "Manage Jenkins", and "My Views". A dropdown menu for "Build Queue" shows "No builds in the queue". Another dropdown for "Build Executor Status" shows "0/2". The main form has a "Name" field containing "maven_web_pipeline" and a "Type" section where "Build Pipeline View" is selected (indicated by a blue circle). Below it, "List View" and "My View" are shown as options. At the bottom right of the form is a "Create" button. The footer of the page includes links for "REST API" and "Jenkins 2.489".

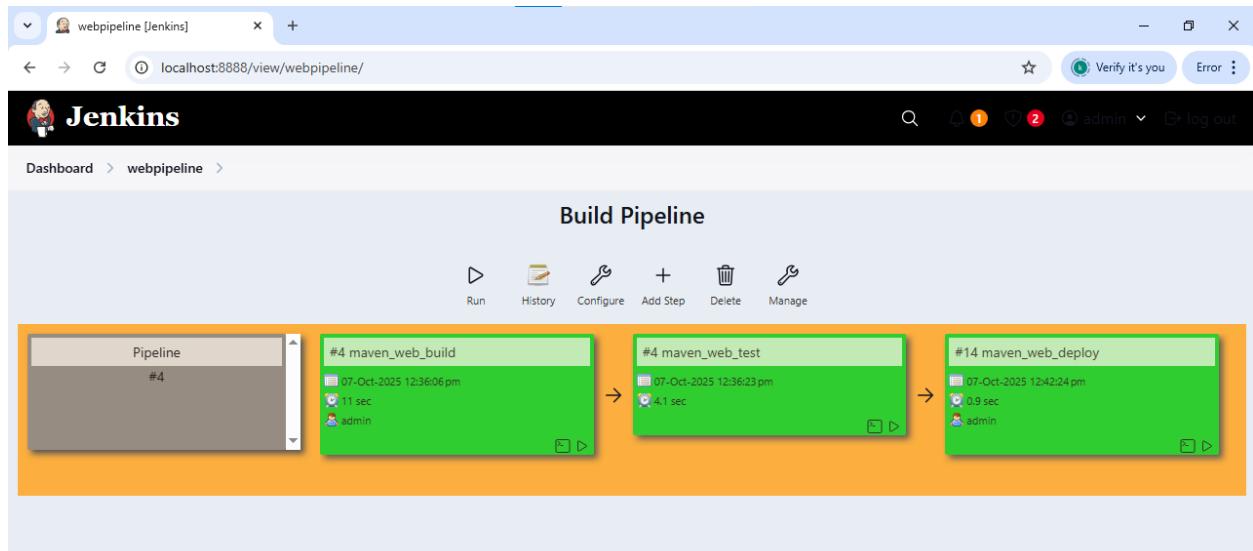
Give the description and in the upstream directly the maven_web_build will be shown

The screenshot shows the Jenkins 'Edit View' configuration page for a view named 'maven_web_pipeline'. The 'Pipeline Flow' section is set to 'Based on upstream/downstream relationship'. Under 'Trigger Options', there are 'Save' and 'Apply' buttons. The taskbar at the bottom shows various application icons and the date/time as 07-10-2025.

Click on apply and save

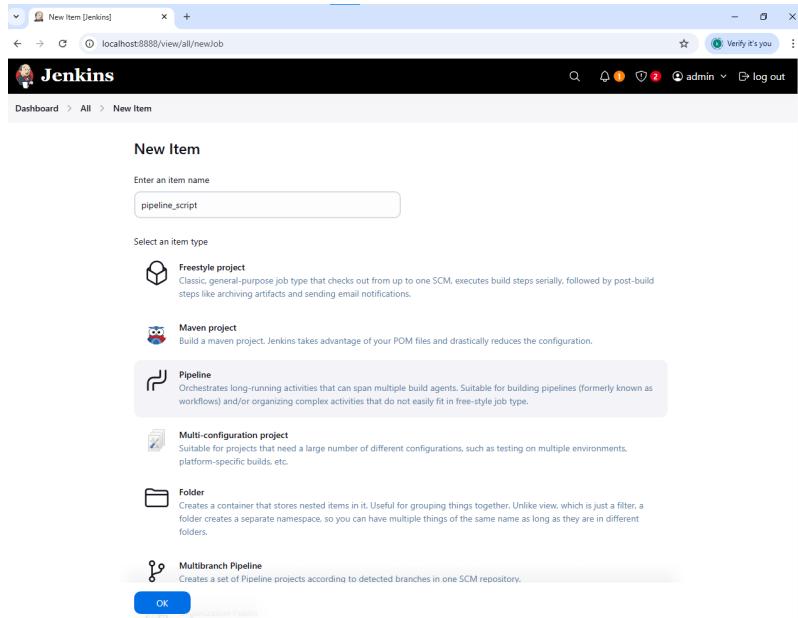
The screenshot shows the Jenkins 'Edit View' configuration page for a view named 'maven_web_pipeline'. The 'Widgets' section contains two checked checkboxes: 'Filter build queue' and 'Filter build executors'. Below these checkboxes are 'Save' and 'Apply' buttons. The taskbar at the bottom shows various application icons and the date/time as 07-10-2025.

In the stage view it we be shown as:

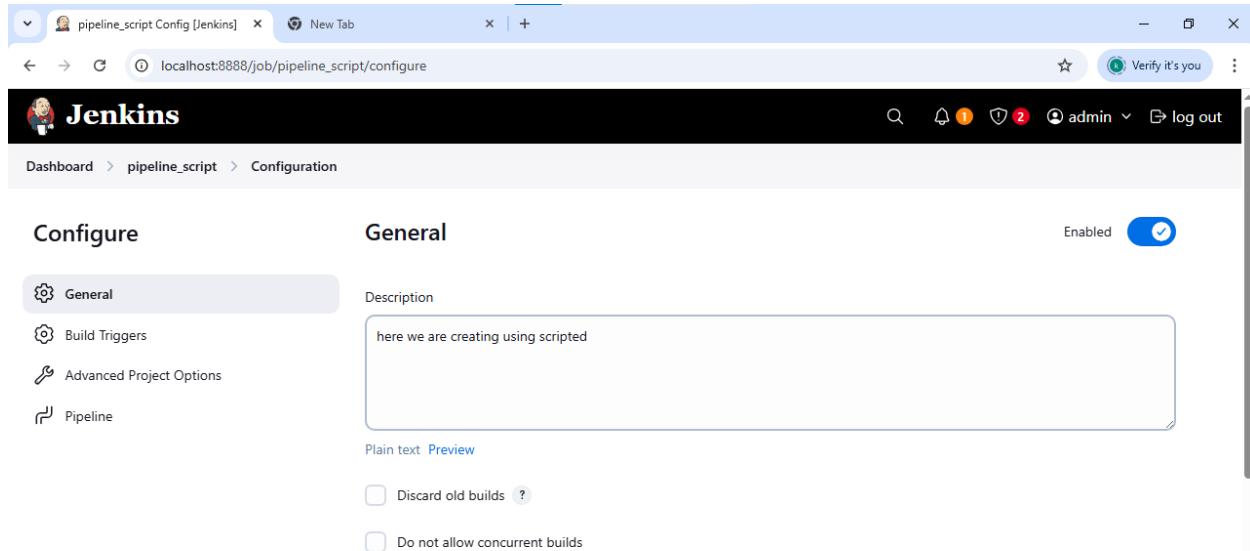


9.Pipeline Creation using script

Step 1: In the Jenkins select the new item and give the name as pipeline_script and select pipeline and click ok



Step 2: In the configuration, give the description



Step 3: In the pipeline section give definition as pipeline script and enter the script with git reop link and project name

The screenshot shows the Jenkins configuration interface for a job named "pipeline_script". The "General" tab is selected. Under "Build Triggers", several options like "Build after other projects are built", "Build periodically", and "GitHub hook trigger for GITScm polling" are listed. The "Advanced Project Options" section has an "Advanced" dropdown. The "Pipeline" section shows the "Definition" set to "Pipeline script". A code editor displays a Groovy script:

```
1 > pipeline {
2   agent any
3   tools
```

At the bottom are "Save" and "Apply" buttons.

Step 4: click on apply and then save

The screenshot shows the Jenkins 'Configure' screen for a project named 'pipeline_script'. The 'Advanced Project Options' tab is selected. The 'Pipeline' section is expanded, showing the 'Definition' set to 'Pipeline script'. Below this is a code editor containing a Groovy pipeline script:

```
1 > pipeline {
2   >   agent any
3   >   tools{
4     >     maven 'MAVEN-HOME'
5   >
6   >   stages {
7     >     stage('git repo & clean') {
8       >       steps {
9         >         //bat "rmdir /s /q mavenjava"
10        >         bat "git clone https://github.com/SarvikaSomishetty/eclipse-maven-projects.git"
11        >         bat "mvn clean -f eclipse-maven-projects"
12      >
13    >   }
14   >   stage('install') {
15     >     steps {
16       >       bat "mvn install -f eclipse-maven-projects"
17     >
18   >
}
```

Below the script, there is a checkbox labeled 'Use Groovy Sandbox' which is checked. At the bottom of the page, there are 'Save' and 'Apply' buttons. The status bar at the bottom right indicates 'REST API Jenkins 2.489'. The taskbar at the bottom of the browser window shows various open tabs and icons.

Step 8: Check the stage view. If is successful.

The screenshot shows the Jenkins interface for the 'pipeline_script' job. The top navigation bar includes a user icon, the job name 'pipeline_script [Jenkins]', and a 'Verify it's you' button. The main content area has a dark header with the Jenkins logo and the job name. Below the header, a breadcrumb trail shows 'Dashboard > pipeline_script >'. The left sidebar contains links for Status, Changes, Build Now, Configure, Delete Pipeline, Full Stage View, Stages, Rename, and Pipeline Syntax. The right side features a 'Stage View' section with a table showing execution times for five stages: Declarative: Tool Install (296ms), git repo & clean (5s), install (9s), test (3s), and package (4s). A summary at the top of the table states 'Average stage times: (full run time: ~26s)'. Below the table, a box indicates 'Oct 07 11:02' and 'No Changes'. The bottom section displays a 'Builds' table with one entry: '#2 11:02 AM'. The 'Permalinks' section lists four recent builds.

Declarative: Tool Install	git repo & clean	install	test	package
296ms	5s	9s	3s	4s
Oct 07 11:02	No Changes			
296ms	5s	9s	3s	4s

Permalinks

- Last build (#2), 4 min 29 sec ago
- Last stable build (#2), 4 min 29 sec ago
- Last successful build (#2), 4 min 29 sec ago
- Last completed build (#2), 4 min 29 sec ago

10. Kubernetes Using Minikube:

Step -1:

Start Minikube : Command- minikube start

- First, you need to start your Kubernetes cluster using Minikube.
- When you start it, Minikube sets up a lightweight virtual machine on your system and runs a local Kubernetes node inside it.

Step-2:Then check for the status Minikube status

Step-3:Create an image

```
PS C:\Users\User>
PS C:\Users\User> kubectl delete deployment mynginx
deployment.apps "mynginx" deleted
PS C:\Users\User> kubectl create deployment mynginx --image=nginx
deployment.apps/mynginx created
PS C:\Users\User> kubectl expose deployment mynginx --type=NodePort --port=80
service/mynginx exposed
PS C:\Users\User> kubectl get pods
NAME           READY   STATUS    RESTARTS   AGE
mynginx-79bb8756c7-wpslj   1/1     Running   0          34s
```

Step-4: Check the NGINX Service Details

- After creating the service, check its details to see which port Kubernetes assigned to it.

```
DownwardAPI:           true
QoS Class:            BestEffort
Node-Selectors:        <none>
Tolerations:          node.kubernetes.io/not-ready:NoExecute op=Exists for 300s
                      node.kubernetes.io/unreachable:NoExecute op=Exists for 300s
Events:
  Type    Reason     Age   From           Message
  ----  -----   ---   ----           -----
  Normal  Scheduled  68s  default-scheduler  Successfully assigned default/mynginx-79bb8756c7-wpslj to minikube
  Normal  Pulled     67s  kubelet         Pulling image "nginx"
  Normal  Pulled     65s  kubelet         Successfully pulled image "nginx" in 2.416s (2.416s including waiting). Image size: 159974475 bytes.
  Normal  Created    65s  kubelet         Created container nginx
  Normal  Started   64s  kubelet         Started container nginx
PS C:\Users\User> kubectl scale deployment mynginx --replicas=4
deployment.apps/mynginx scaled
PS C:\Users\User> kubectl get service mynginx
Error from server (NotFound): services "mynginx" not found
PS C:\Users\User> kubectl port-forward svc/mynginx 8081:80
Forwarding from 127.0.0.1:8081 -> 80
Forwarding from [:1]:8081 -> 80
```

Step-5:check the detail of the kubectl .

```
PS C:\Users\User> kubectl describe pods
Name:           mynginx-79bb8756c7-wpslj
Namespace:      default
Priority:      0
Service Account: default
Node:          minikube/192.168.49.2
Start Time:    Tue, 14 Oct 2025 12:38:19 +0530
Labels:        app=mynginx
               pod-template-hash=79bb8756c7
Annotations:   <none>
Status:        Running
IP:            10.244.0.16
IPs:
  IP:          10.244.0.16
Controlled By: ReplicaSet/mynginx-79bb8756c7
Containers:
  nginx:
    Container ID:  docker://675066efbd98a54ba39177103943b196de2c61f01d820ede859b48578f3e245e
    Image:         nginx
    Image ID:     docker-pullable://nginx@sha256:3b7732505933ca591ce4a6d860cb713ad96a3176b82f7979a8dfa9973486a0d6
    Port:          <none>
    Host Port:    <none>
    State:        Running
      Started:   Tue, 14 Oct 2025 12:38:22 +0530
    Ready:        True
    Restart Count: 0
    Environment:  <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-nh2rw (ro)
Conditions:
  Type        Status
  PodReadyToStartContainers  True
  Initialized  True
  Ready        True
  ContainersReady  True
  PodScheduled  True
Volumes:
  kube-api-access-nh2rw:
    Type:       Projected (a volume that contains injected data from multiple sources)
    TokenExpirationSeconds: 3607
    ConfigMapName:      kube-root-ca.crt
    ConfigMapOptional:  <nil>
    DownwardAPI:       true
  QoS Class:      BestEffort
  Node-Selectors:  <none>
  Tolerations:    node.kubernetes.io/not-ready:NoExecute op=Exists for 300s
                  node.kubernetes.io/unreachable:NoExecute op=Exists for 300s
Events:
```

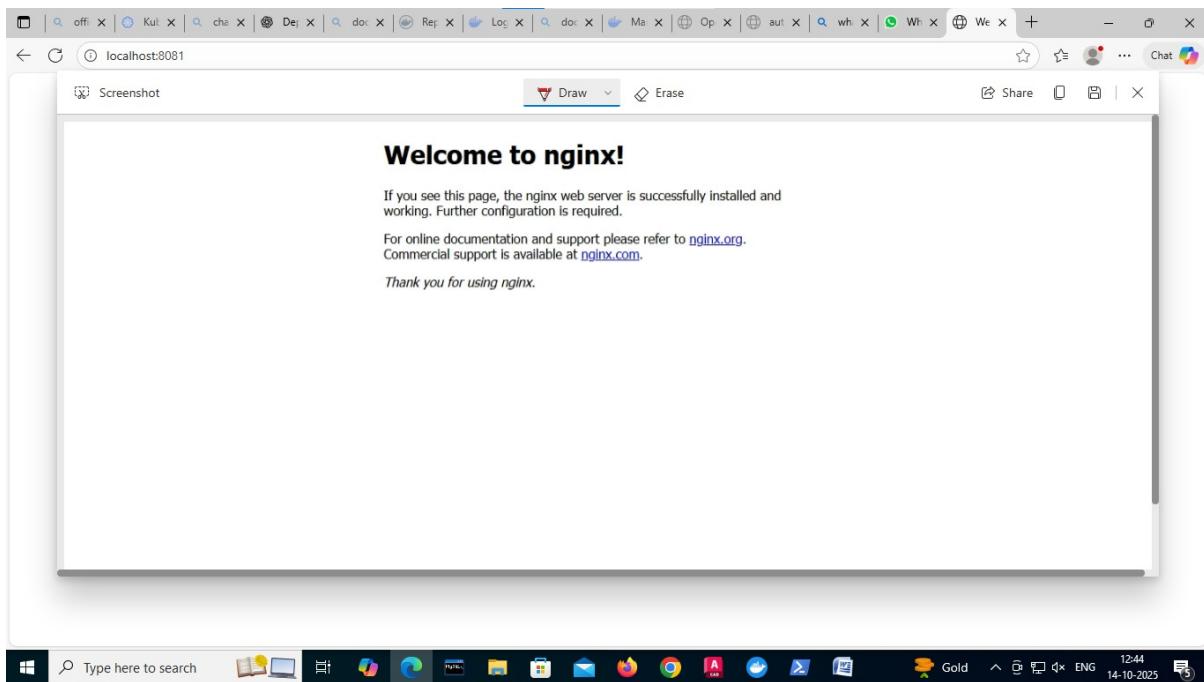
Step-6:Check the NGINX Service Details

- After creating the service, check its details to see which port Kubernetes assigned to it.

```
PS C:\Users\User> kubectl port-forward svc/mynginx 8081:80
Forwarding from 127.0.0.1:8081 -> 80
Forwarding from [::1]:8081 -> 80
Handling connection for 8081
Handling connection for 8081
Handling connection for 8081
Handling connection for 8081
```

Step-7: Open NGINX in the Browser

- Now that your service is exposed, you can open NGINX in your browser.



11. Jenkins-CI/CD

Setting Up Jenkins CI-----using GitHub Webhook with Jenkins

Step 1: Take the authentication key from the ngrok and setup in ngrok terminal

```
tcp          start a TCP tunnel
tls          start a TLS endpoint
update      update ngrok to the latest version
version     print the version string

EXAMPLES:
# forward http traffic from assigned public URL to local port 80
ngrok http 80
# port 8080 available at baz.ngrok.dev
ngrok http --url baz.ngrok.dev 8080
# tunnel arbitrary TCP traffic to port 22
ngrok tcp 22
# secure your app with oauth
ngrok http 80 --oauth=google --oauth-allow-email=foo@foo.com

Paid Features:
ngrok http 80 --url mydomain.com                               # run ngrok with your own custom domain
ngrok http 80 --cidr-allow 2600:8c00::a03c:91ee:fe69:9695/32 # run ngrok with IP policy restrictions
Upgrade your account at https://dashboard.ngrok.com/billing/subscription to access paid features

Upgrade your account at https://dashboard.ngrok.com/billing/subscription to access paid features

Flags:
-h, --help      help for ngrok

Use "ngrok [command] --help" for more information about a command.

ngrok is a command line application, try typing 'ngrok.exe http 80'
at this terminal prompt to expose port 80.
C:\Windows\System32>ngrok config add-authtoken 34gKWhQDcoITj34K6eN73XoYG6J_58fBgmpjM5ikZVdKVdyCe|
```

Step-2: Execute the following command using the port number on which Jenkins is running

```
C:\Windows\System32>ngrok.exe http 8888|
```

- Following output will be given:

```
ngrok
(Ctrl+C to quit)

◆ Block threats before they reach your services with new WAF actions → https://ngrok.com/r/waf

Session Status           online
Account                  Neksha Edigirala (Plan: Free)
Update                   update available (version 3.32.0, Ctrl-U to update)
Version                 3.24.0-msix
Region                  India (in)
Latency                 147ms
Web Interface           http://127.0.0.1:4040
Forwarding              https://corkier-darla-handsome.ngrok-free.dev -> http://localhost:8888

Connections             ttl     opn      rt1      rt5      p50      p90
                        2       0       0.00    0.00    30.28    30.47

HTTP Requests
-----
11:35:59.377 IST POST /github-webhook/          200 OK
11:34:29.479 IST POST /github-webhook/          200 OK
```

Go to Jenkins:

Step-3: Create the Jenkins job in the source code management select the git and enter git repo url and make sure the branch is same (i.e., main)

The screenshot shows the Jenkins job configuration page for 'job_webhook_java'. Under 'Source Code Management', the 'Git' option is selected. The 'Repository URL' field contains 'https://github.com/Edigirala-Neksha/se-lab-internal-1.git'. The 'Branches to build' field contains '/main'. Other tabs like General, Triggers, Environment, Build Steps, and Post-build Actions are visible on the left.

Step-4: In the triggers section select “Github hook trigger for GITScm polling”

The screenshot shows the Jenkins job configuration page for 'job_webhook_java'. Under 'Triggers', the 'GitHub hook trigger for GITScm polling' checkbox is checked. Other options like 'Trigger builds remotely' and 'Build after other projects are built' are available but not selected. Buttons for 'Save' and 'Apply' are at the bottom.

Click on apply and save

Step-6: open the git hub repo open setting of repo and then go to webhooks

The screenshot shows the GitHub repository settings for 'se-lab-internal-1'. The 'General' tab is selected. In the left sidebar, 'Webhooks' is highlighted. The main area shows the repository name 'se-lab-internal-1' and a section for 'Require contributors to sign off on web-based commits'. Below that is the 'Default branch' section, which currently has 'main' selected. The 'Releases' section is also visible.

Step-7: Click on add a webhook and take the forwarding URL from ngrok and paste in payload URL and add /github-webhook/ along with the forwarding url

Forwarding URL: <https://corkier-darla-handsome.ngrok-free.dev>

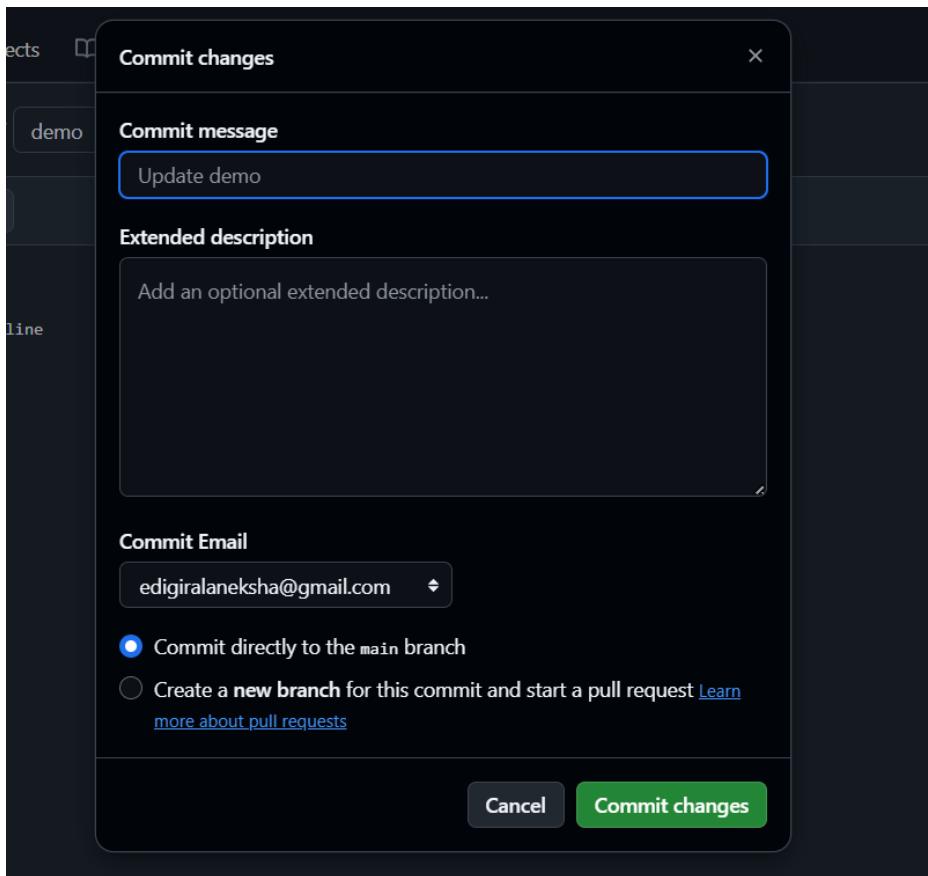
Payload url: <https://corkier-darla-handsome.ngrok-free.dev/github-webhook/>

The screenshot shows the 'Webhooks / Add webhook' form. The 'Payload URL' field contains 'https://corkier-darla-handsome.ngrok-free.dev/github-webhook/'. The 'Content type' dropdown is set to 'application/x-www-form-urlencoded'. The 'SSL verification' section has 'Enable SSL verification' selected. The 'Which events would you like to trigger this webhook?' section has 'Just the push event.' selected.

Step 8: make changes in the files in github

A screenshot of a GitHub repository named "se-lab-internal-1". The "Code" tab is selected. In the center, there's a file editor for a file named "demo" in the "main" branch. The code contains three lines: "demooooooo", "webhook", and "xxxxxx-new line!". Above the code editor, there are buttons for "Cancel changes" and "Commit changes...". On the left, there's a sidebar with a tree view of the project structure, showing ".settings", "src", "target", ".classpath", ".project", "Dockerfile", "demo", and "pom.xml". A status bar at the bottom indicates keyboard navigation instructions.

Step 9: click on commit changes

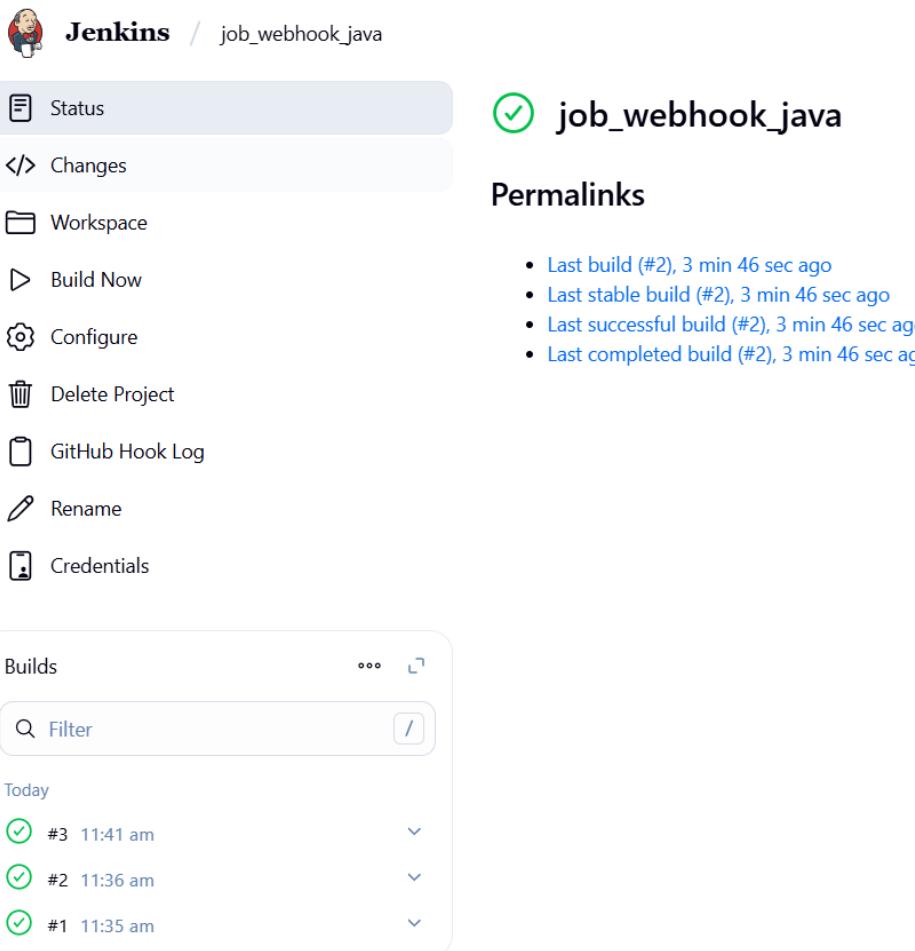


Step 10: open Jenkins the build will start automatically

The screenshot shows the Jenkins interface for the 'job_webhook_java' project. The top navigation bar includes the Jenkins logo and the project name 'job_webhook_java'. A sidebar on the left lists various project management options: Status (highlighted), Changes, Workspace, Build Now, Configure, Delete Project, GitHub Hook Log, Rename, and Credentials. The main content area is titled 'Permalinks' and lists four recent builds. Below this is a 'Builds' section with a 'Pending' row for build #3 and a 'Today' row for build #2 (11:36 am). A 'Filter' input field is also present.

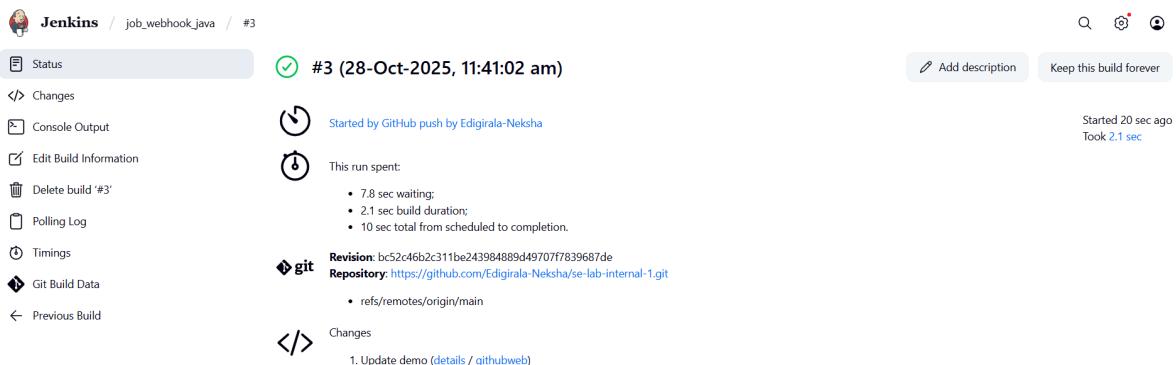
- Last build (#2), 3 min 46 sec ago
- Last stable build (#2), 3 min 46 sec ago
- Last successful build (#2), 3 min 46 sec ago
- Last completed build (#2), 3 min 46 sec ago

Build	Timestamp
#3	In the quiet period. Expires in 2.9 sec
#2	11:36 am



The screenshot shows the Jenkins job_webhook_java dashboard. On the left is a sidebar with various options: Status (highlighted), Changes, Workspace, Build Now, Configure, Delete Project, GitHub Hook Log, Rename, and Credentials. Below the sidebar is a 'Builds' section with a 'Filter' input field. It lists three builds from today: #3 (11:41 am), #2 (11:36 am), and #1 (11:35 am). Each build entry has a dropdown arrow next to it.

You can check status : started by git hub push



This screenshot shows the details of build #3. The top bar includes links for Status, Changes, Console Output, Edit Build Information, Delete build '#3', Polling Log, Timings, Git Build Data, and Previous Build. The main content area shows the build status as 'Started by GitHub push by Edigirala-Neksha' on 28-Oct-2025 at 11:41:02 am. It also displays the revision (bc52c46b2c311be243984889d4970f7839687de) and repository (https://github.com/Edigirala-Neksha/se-lab-internal-1.git). The build took 2.1 seconds and spent 10 seconds total from scheduled to completion. A note indicates the build was started 20 seconds ago. There are buttons for 'Add description' and 'Keep this build forever'. Below the build summary, there's a 'Changes' section showing a single commit: '1. Update demo (details / githubweb)'.

Setting Up Jenkins Email Notification Setup (Using Gmail with AppPassword)

Step-1: Creation of app password

Gmail: Enable App Password (for 2-Step Verification)

ii. Enable 2-Step Verification

iii. Generate App Password for Jenkins

- Go to:
 - Security → App passwords
- Select:
 - **App:** Other (Custom name)
 - **Name:** Jenkins-Demo
- Click **Generate**
- Copy the **16-digit app password**
 - Save it in a secure location (e.g., Notepad)

2. Jenkins Plugin Installation

i. Open Jenkins Dashboard

ii. Navigate to:

- Manage Jenkins → Manage Plugins

iii. Install Plugin:

- Search for and install:
 - Email Extension Plugin

The screenshot shows the Jenkins 'Plugins' page. A search bar at the top contains the text 'email'. Below it, a table lists several plugins:

- Email Extension**: Version 1925.v1598902b_58dd. Status: 100, Enabled, with a red 'X' icon.
- Email Extension Template Plugin**: Version 233.v1eb_88fc160b_5. Status: 100, Enabled, with a red 'X' icon.
- Mailer Plugin**: Version 522.va_995fa_cfb_8b_d. Status: 100, Enabled, with a red 'X' icon.

A message box in the center says: 'Failed to load: Pipeline (workflow-aggregator 608.v67378e9d3db_1) - Failed to load: Pipeline: Basic Steps (workflow-basic-steps 1098.v808b_fd7f8cf4)'. A toggle switch is shown below the message.

3. Configure Jenkins Global Email Settings

Go to:

- Manage Jenkins → Configure System

A. E-mail Notification Section

Field	Value
SMTP Server	smtp.gmail.com
Use SMTP Auth	<input checked="" type="checkbox"/> Enabled
User Name	Your Gmail ID (e.g., archanareddykmit@gmail.com)
Password	Paste the 16-digit App Password
Use SSL	<input checked="" type="checkbox"/> Enabled
SMTP Port	465
Reply-To Address	Your Gmail ID (same as above)

► Test Configuration

- Click: Test configuration by sending test e-mail
- Provide a valid email address to receive a test mail
- Should receive email from Jenkins

Jenkins / Manage Jenkins / System

E-mail Notification

SMTP server

smtp.gmail.com

Default user e-mail suffix ?

Advanced ^ Edited

Use SMTP Authentication ?

User Name
edigiralaneksha@gmail.com

Password
 Concealed Change Password

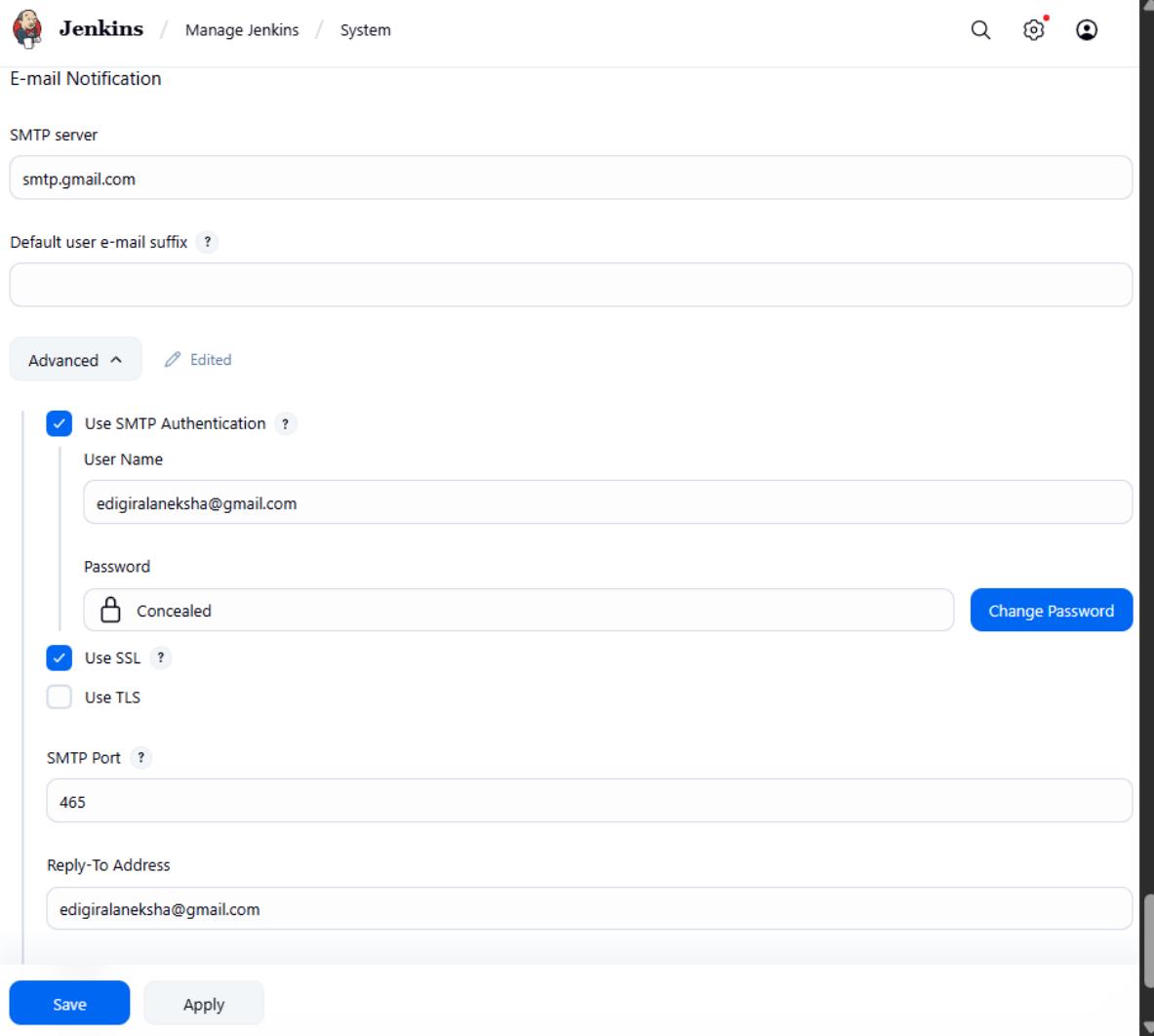
Use SSL ?

Use TLS

SMTP Port ?
465

Reply-To Address
edigiralaneksha@gmail.com

Save Apply



B. Extended E-mail Notification Section

Field	Value
SMTP Server	smtp.gmail.com
SMTP Port	465
Use SSL	<input checked="" type="checkbox"/> Enabled
Credentials	Add Gmail ID and App Password as Jenkins credentials
Default Content Type	text/html or leave default
Default Recipients	Leave empty or provide default emails
Triggers	Select as per needs (e.g., Failure)

Extended E-mail Notification

SMTP server
smtp.gmail.com

SMTP Port
465

Advanced ^ Edited

Credentials
edigiralaneksha@gmail.com/***** (frst) + Add

Use SSL
 Use TLS
 Use OAuth 2.0

Advanced Email Properties

Save Apply

Default Triggers ^

Default Triggers ?

Aborted
 Always
 Before Build
 Failure - 1st
 Failure - 2nd
 Failure - Any
 Failure - Still
 Failure - X
 Failure -> Unstable (Test Failures)
 Fixed
 Not Built
 Script - After Build
 Script - Before Build
 Status Changed
 Success
 Test Improvement
 Test Regression
 Unstable (Test Failures)
 Unstable (Test Failures) - 1st
 Unstable (Test Failures) - Still
 Unstable (Test Failures)/Failure -> Success

Content Token Reference ?

4. Configure Email Notifications for a Jenkins Job

i. Go to:

- Jenkins → Select a Job → Configure

The screenshot shows the Jenkins configuration interface for a job named "job_webhook_java".

General Section:

- Description:** java webhook
- Enabled:** Yes (checkbox checked)
- Discard old builds:** Unchecked
- GitHub project:** Unchecked
- Notify when Job configuration changes:** Unchecked
- This project is parameterized:** Unchecked
- Throttle builds:** Unchecked
- Execute concurrent builds if necessary:** Unchecked
- Advanced:** A dropdown menu.

Source Code Management Section:

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

- None:** Unchecked
- Git:** Checked
- Repositories:** A dropdown menu.

Buttons: Save (blue), Apply.

ii. In the Post-build Actions section:

- Click: Add post-build action → **Editable Email Notification**

A. Fill in the fields:

Field	Value
Project Recipient List	Add recipient email addresses (comma-separated)
Content Type	Default (text/plain) or text/html
Triggers	Select events (e.g., Failure, Success, etc.)
Attachments	(Optional) Add logs, reports, etc.

iii. Click Save

Post-build Actions

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

≡ **Editable Email Notification** ? X

Allows the user to disable the publisher, while maintaining the settings

Disable Extended Email Publisher ?

Project From

Project Recipient List ?

Comma-separated list of email address that should receive notifications for this project.

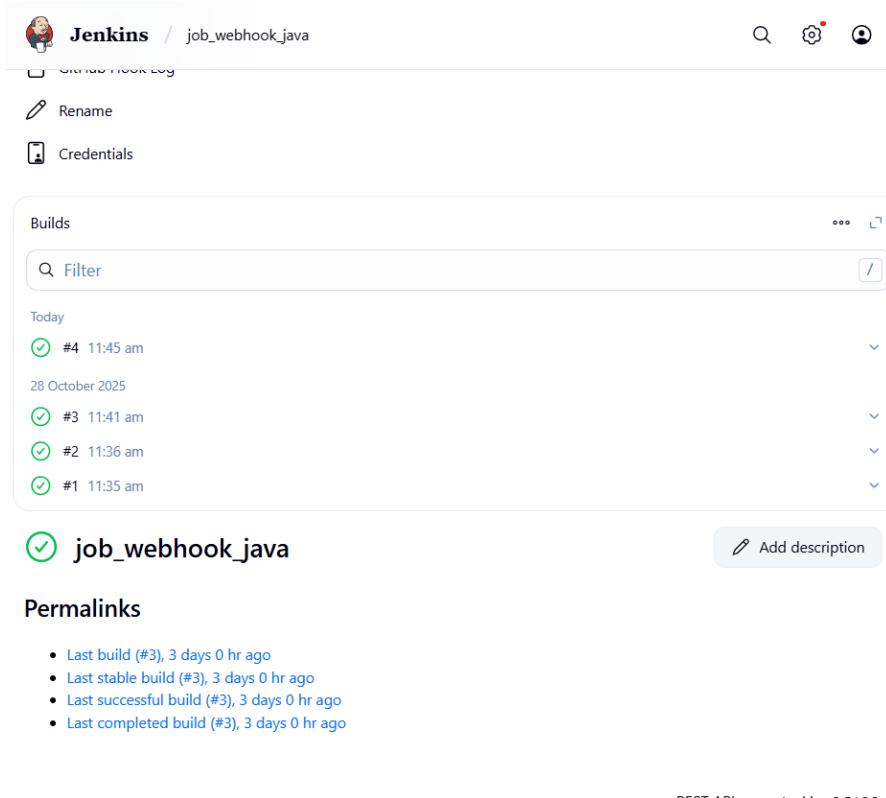
edigiralaneksha@gmail.com,nekshasri99@gmail.com

Project Reply-To List ?

Comma-separated list of email address that should be in the Reply-To header for this project.

\$DEFAULT_REPLYTO

Save **Apply**



The screenshot shows the Jenkins interface for a job named "job_webhook_java".

Job Information:

- Icon: Jenkins logo
- Name: Jenkins / job_webhook_java
- Build Log: Shows "jenkins.log" and "jenkins.log.1" with a "View Log" link.
- Rename: Option to rename the job.
- Credentials: Option to manage credentials.

Build History:

- Builds: A table showing build history from today to October 28, 2025.
- Today:
 - #4 11:45 am (green circle)
 - #3 11:41 am (green circle)
 - #2 11:36 am (green circle)
 - #1 11:35 am (green circle)
- 28 October 2025:
 - #4 11:45 am (green circle)
 - #3 11:41 am (green circle)
 - #2 11:36 am (green circle)
 - #1 11:35 am (green circle)

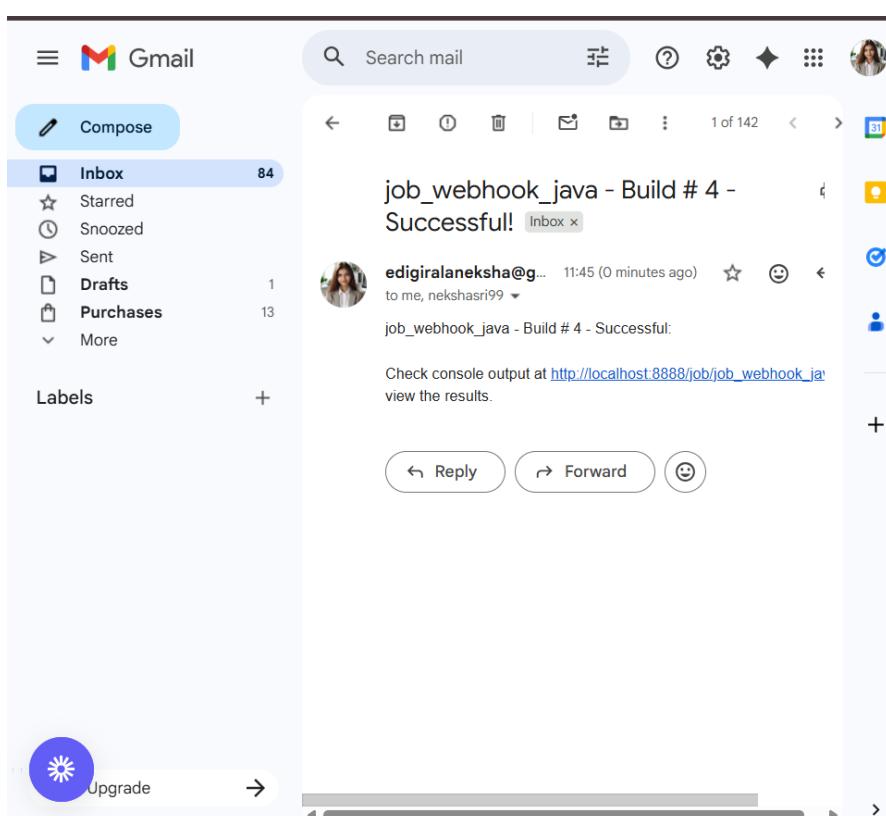
Job Details:

- Job Name: job_webhook_java
- Add description: Option to add a description.

Permalinks:

- Last build (#3), 3 days 0 hr ago
- Last stable build (#3), 3 days 0 hr ago
- Last successful build (#3), 3 days 0 hr ago
- Last completed build (#3), 3 days 0 hr ago

REST API Jenkins 2.516.3



The screenshot shows a Gmail inbox with one unread email from "edigiralaneksha@gmail.com" regarding a Jenkins build.

Inbox Summary:

- Compose button
- Inbox (84 messages)
- Starred (1)
- Snoozed (1)
- Sent (1)
- Drafts (1)
- Purchases (13)
- More

Email Preview:

job_webhook_java - Build # 4 - Successful!

edigiralaneksha@gmail.com 11:45 (0 minutes ago) to me, nekshasri99

job_webhook_java - Build # 4 - Successful!

Check console output at http://localhost:8888/job/job_webhook_java view the results.

Bottom Navigation:

- Reply
- Forward
- Compose

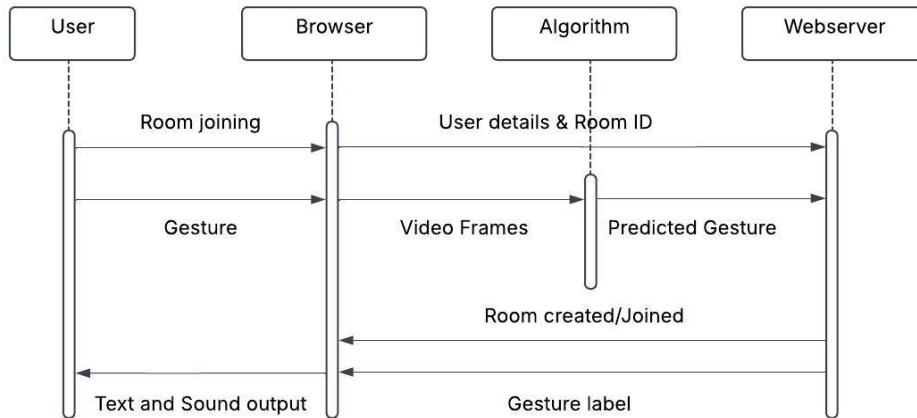
TUNEORA – A Music Web

App

1. Sequence Diagram:

A sequence diagram shows how objects interact in a particular scenario of a use case.

It focuses on the time order of messages exchanged between different components in a system.



2. Class Diagram:

A class diagram represents the static structure of a system by showing classes, their attributes, methods, and relationships.

It is mainly used for object-oriented design and modeling data structures.

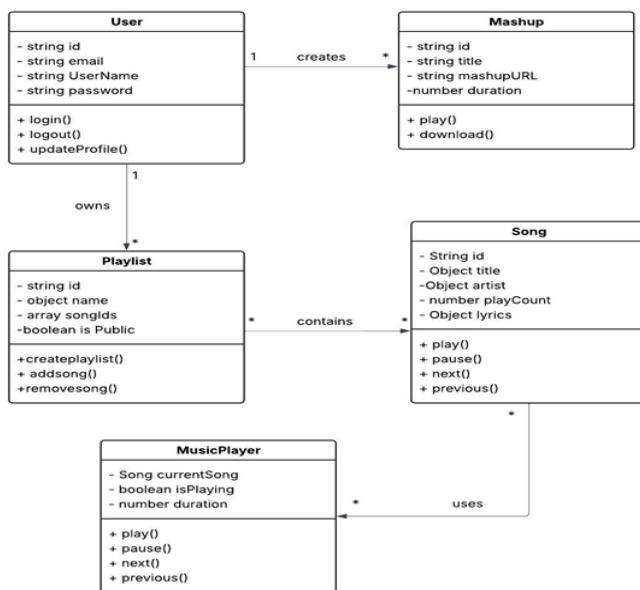
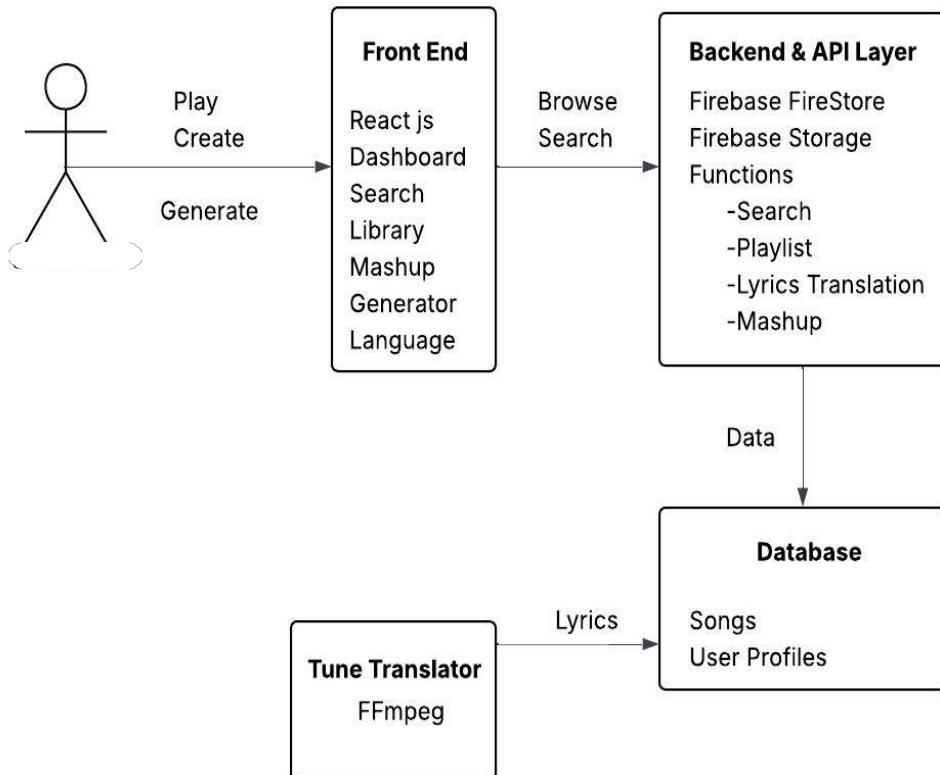


Fig 5: Class Diagram for TuneOra

3. Component Diagram:

A component diagram illustrates how different software components are connected and interact to form a complete system.

It helps visualize the organization and dependencies among modules or subsystems.



12. Creation of virtual machine for Ubuntu OS and Deploying the web application

DEPLOYMENT OF INDEX.HTML USING EC2 INSTANCE in AWS

Step 1: Click on Modules.

The screenshot shows the AWS Academy Learner Lab interface. On the left, there's a sidebar with icons for Account, Dashboard, Courses, Calendar, Inbox, History, and Help. The main area has a breadcrumb navigation: ALLv2EN-US... > Modules > AWS Academ... > Launch AWS Academy Learner Lab. At the top, it says "Used \$0 of \$50" and has buttons for Start Lab, End Lab, AWS Details, Readme, and Reset. A dropdown menu shows "EN-US". The central part is titled "Learner Lab" and contains a "Environment Overview" section with links to Environment Navigation, Access the AWS Management Console, Region restriction, Service usage and other restrictions, Using the terminal in the browser, Running AWS CLI commands, Using the AWS SDK for Python, Preserving your budget, Accessing EC2 Instances, SSH Access to EC2 Instances, and SSH Access from Windows. Below this is a large blue V-shaped graphic with orange arrows at the top.

Step 2: Scroll down and select Launch AWS Academy Lab

The screenshot shows the AWS Academy module page. The sidebar includes Account, Dashboard, Courses, Calendar, Inbox, History, and Help. The main content area has a "Home" tab selected. Under "AWS Academy Learner Lab", there's a link labeled "Launch AWS Academy Learner Lab". Other sections include "AWS Academy Learner Lab Compliance and Security" (with a "Complete All Items" button) and "Module Knowledge Check" (worth 100 pts). The bottom of the page shows a URL: <https://awsacademy.instructure.com/courses/141967/modules/items/13720316>.

Step 3: click on start lab

The screenshot shows the AWS Academy Learner Lab interface. On the left is a sidebar with icons for Account, Dashboard, Courses, Calendar, Inbox, History, and Help. The main area has a top navigation bar with 'AWS' (red dot), 'Used \$0 of \$50', '00:00', and buttons for 'Start Lab', 'End Lab', 'AWS Details', 'Readme', 'Reset', and a close button. Below the navigation is a terminal window showing a command prompt: 'eee_W_5353255@runweb195092:~\$'. To the right is a 'Learner Lab' panel with a dropdown menu set to 'EN-US'. The 'Learner Lab' section contains a list of links: Environment Overview, Environment Navigation, Access the AWS Management Console, Region restriction, Service usage and other restrictions, Using the terminal in the browser, Running AWS CLI commands, Using the AWS SDK for Python, Preserving your budget, Accessing EC2 Instances, SSH Access to EC2 Instances, and SSH Access from Windows.

Step 4: click on AWS and in the services select EC2

This screenshot is identical to the one above, showing the AWS Academy Learner Lab interface. The 'AWS' service is selected in the top navigation bar. The terminal window shows a command prompt: 'eee_W_3940257@runweb155453:~\$'. The 'Learner Lab' panel on the right is visible with its list of links.

Step 5: select instances and select instance click on launch instance

The screenshot shows the AWS EC2 Dashboard in the N. Virginia region. The left sidebar includes sections for Instances, Images, and Elastic Block Store. The main area displays 'Resources' and 'Launch instance' sections. The 'Launch instance' section contains a 'Launch instance' button and a note about launching in the US East (N. Virginia) Region. To the right, there's a 'Service health' section showing the AWS Health Dashboard and a status message for the EC2 service. The 'Account attributes' section lists various VPC and security settings. A 'Explore AWS' sidebar on the right provides tips for cost reduction.

Step 6: Give the name of the machine “week-122”

The screenshot shows the 'Launch an instance' wizard in the us-east-1 region. The first step, 'Name and tags', has 'Name' set to 'week-122'. The second step, 'Application and OS Images (Amazon Machine Image)', shows the 'Quick Start' tab selected, displaying options for Amazon Linux, macOS, Ubuntu, Windows, Red Hat, SUSE Linux, and Debian. A search bar and a 'Browse more AMIs' link are also present. The third step, 'Summary', shows one instance selected, the software image as Canonical, Ubuntu, 24.04, amd64, and the virtual server type as t3.micro. It also shows a new security group and 1 volume(s) - 8 GiB. Buttons for 'Cancel', 'Launch instance', and 'Preview code' are at the bottom.

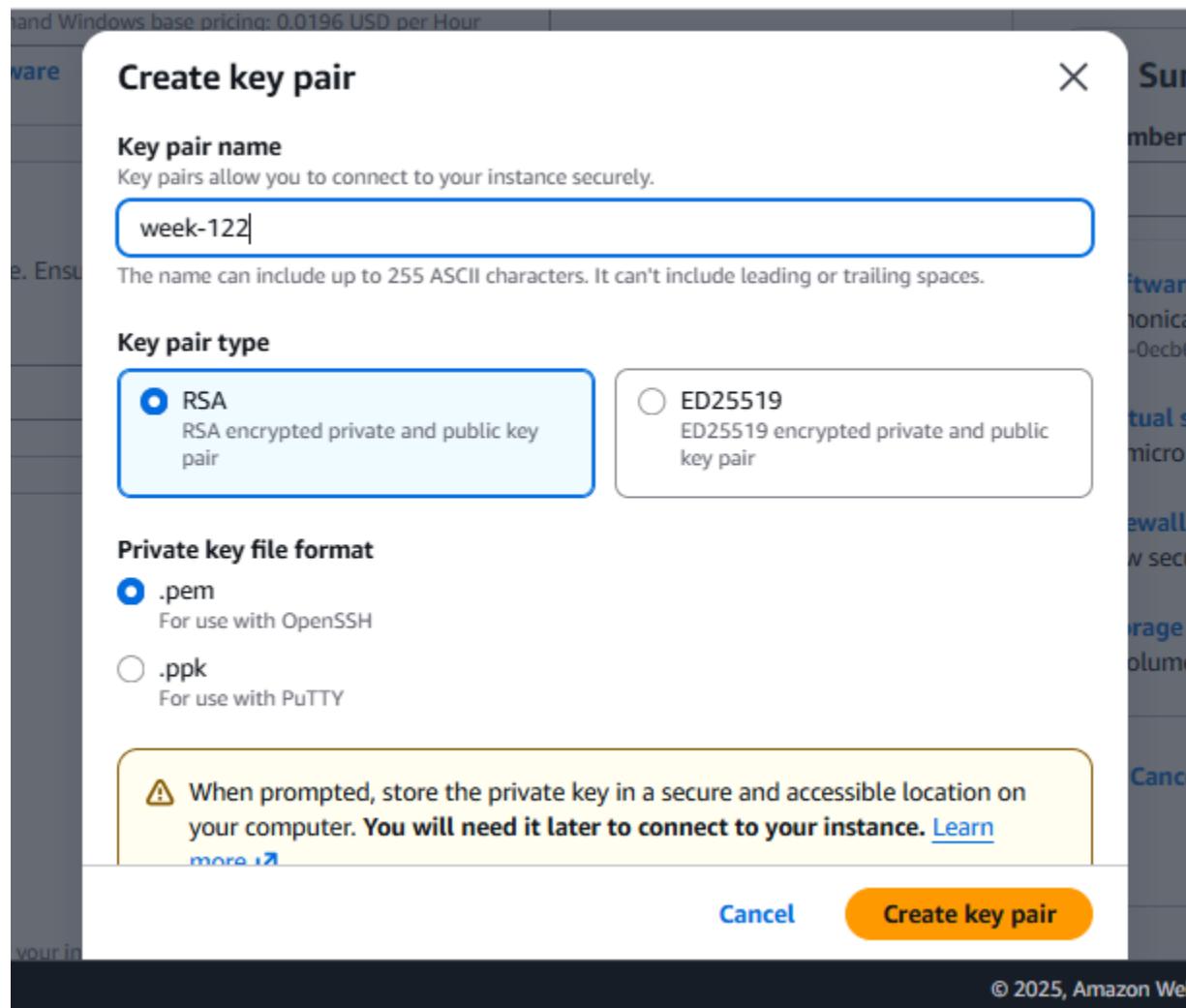
Step 6: Select the ubuntu server

The screenshot shows the AWS EC2 'Launch an instance' wizard. In the left sidebar, under 'Quick Start', the 'Ubuntu' option is selected. Below it, the 'Amazon Machine Image (AMI)' section displays the 'Ubuntu Server 24.04 LTS (HVM), SSD Volume Type' AMI (ami-0ecb62995f68bb549). The 'Description' panel states: 'Ubuntu Server 24.04 LTS (HVM). EBS General Purpose (SSD) Volume Type. Support available from Canonical (<http://www.ubuntu.com/cloud/services>).'. The 'Summary' panel on the right shows the configuration: 1 instance, Software Image (AMI) set to Canonical, Ubuntu, 24.04, amd64, Virtual server type set to t3.micro, and Storage (volumes) set to 1 volume(s) - 8 GiB. The 'Launch instance' button is highlighted.

Step 7: select architecture as 64-bit and instance type as t3.micro(i.e., they are free)

The screenshot shows the AWS EC2 'Launch an instance' wizard. The 'Architecture' dropdown is set to '64-bit (x86)'. The 'Instance type' dropdown is set to 't3.micro'. The 'Summary' panel on the right shows the configuration: 1 instance, Software Image (AMI) set to Canonical, Ubuntu, 24.04, amd64, Virtual server type set to t3.micro, and Storage (volumes) set to 1 volume(s) - 8 GiB. The 'Launch instance' button is highlighted.

Step 8: Create a new keypair and select type as RSA and .pem option and click on create key pair



Step 9: In network settings select “create security group” and give the security group name

▼ Network settings [Info](#)

VPC - required | [Info](#)

vpc-05a9ef3852073b114 (default) ▾ [Create new VPC](#)

Subnet | [Info](#)

No preference ▾ [Create new subnet](#)

Availability Zone | [Info](#)

No preference ▾ [Enable additional zones](#)

Auto-assign public IP | [Info](#)

Enable ▾

Firewall (security groups) | [Info](#)

A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.

Create security group Select existing security group

Security group name - required

week-122

This security group will be added to all network interfaces. The name can't be edited after the security group is created. Max length is 255 characters. Valid characters: a-z, A-Z, 0-9, spaces, and _.-~/()#,@[]+=&;!\$*

Description - required | [Info](#)

Launch wizard-1 created 2025-11-11T05:36:49.724Z

Step 10: Click on edit button on the top right corner and select

Type: ssh

Source: anywhere

EC2 > Instances > Launch an instance

week-122

This security group will be added to all network interfaces. The name can't be edited after the security group is created. Max length is 255 characters. Valid characters: a-z, A-Z, 0-9, spaces, and _.-~/()#,@[]+=&;!\$*

Description - required | [Info](#)

Launch wizard-1 created 2025-11-11T05:36:49.724Z

Inbound Security Group Rules

▼ Security group rule 1 (TCP, 22, 0.0.0.0/0)

Type	Protocol	Port range	Action
ssh	TCP	22	Remove
Source type	Source	Add CIDR, prefix list or security group	Description - optional
Anywhere		e.g. SSH for admin desktop	0.0.0.0/0

⚠ Rules with source of 0.0.0.0/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.

[Add security group rule](#)

Summary

Number of instances | [Info](#)

1

Software Image (AMI)

Canonical, Ubuntu, 24.04, amd64... [read more](#)

ami-0ebc62995f69bb549

Virtual server type (instance type)

t3.micro

Firewall (security group)

New security group

Storage (volumes)

1 volume(s) - 8 GiB

[Cancel](#) [Launch instance](#) [Preview code](#)

Step 11: in configure storage give 8GB and give number of instances as 2 and click on launch instance

The screenshot shows the AWS EC2 'Launch an instance' configuration interface. In the 'Configure storage' section, a root volume of 8 GiB (gp3) is selected. The 'Number of instances' dropdown is set to 2. The 'Launch instance' button is visible at the bottom right.

Step 12: The launching of instance will start and successful message will be shown

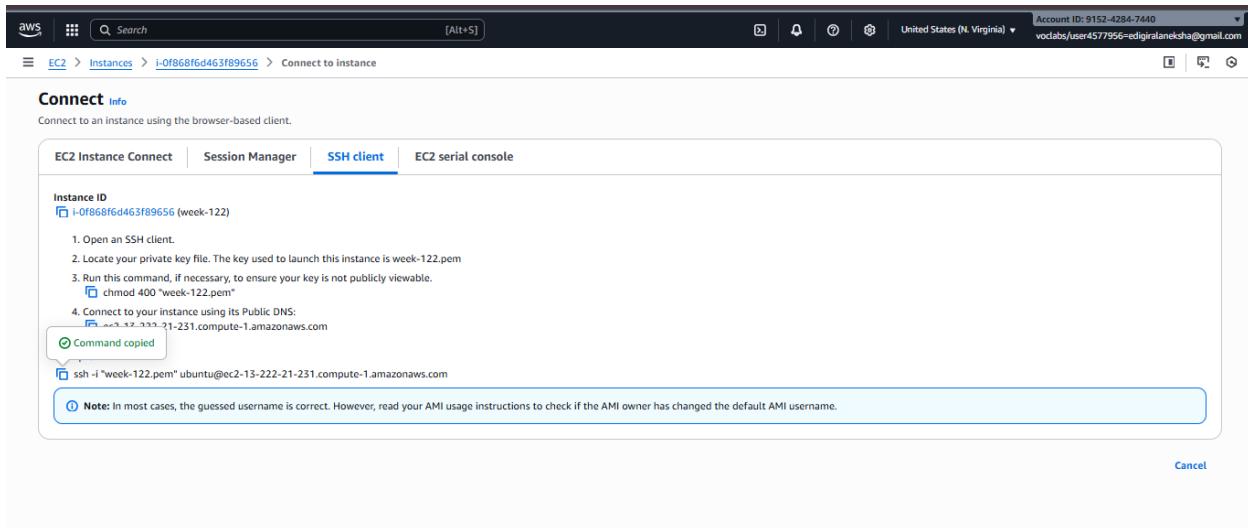
The screenshot shows a progress bar indicating the instance is launching. The status message reads: 'Please wait while we launch your instance. Do not close your browser while this is loading.' A progress bar shows 33% completion.

The screenshot shows the AWS EC2 Instances Launch an instance page. At the top, there is a green success message: "Successfully initiated launch of Instances (i-0f868f6d463f89656, i-0a5aa6fe5d0039e34)". Below this, there is a "Launch log" section. Under "Next Steps", there are six cards with links: "Create billing usage alerts", "Connect to your instance", "Connect an RDS database", "Create EBS snapshot policy", "Manage detailed monitoring", "Create Load Balancer", "Create AWS budget", and "Manage CloudWatch alarms". The bottom of the page includes standard AWS navigation links like CloudShell, Feedback, and a footer with copyright information.

Step 13: In the instances the created ones will be shown, you can also rename the instance , changed week-1222 to “webapp”

The screenshot shows the AWS EC2 Instances page. On the left, there is a sidebar with navigation links: Dashboard, EC2 Global View, Events, Instances (selected), Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity Reservations, Capacity Manager, Images, AMIs, AMI Catalog, and Elastic Block Store. The main area displays a table of instances with columns: Name, Instance ID, Instance state, Instance type, Status check, Alarm status, Availability Zone, Public IPv4 DNS, and Public. There are four instances listed: "week-12" (terminated), "week-122" (terminated), "webapp" (running), and another unnamed instance (terminated). Below the table, the details for the "webapp" instance are shown, including its instance ID, public and private IP addresses, and instance state. The bottom of the page includes standard AWS navigation links like CloudShell, Feedback, and a footer with copyright information.

Step 14: click on connect and select “SSH Client” and copy the ssh command



Step 15: Navigate to the path where the file with .pem extension is present(week-122.pem) and paste the command

```

PS C:\Users\User\downloads> ssh -i "week-122.pem" ubuntu@ec2-13-222-21-231.compute-1.amazonaws.com
The authenticity of host 'ec2-13-222-21-231.compute-1.amazonaws.com (13.222.21.231)' can't be established.
ED25519 key fingerprint is SHA256:NEGegcHQjt8om/1AVLSqfmafnMphv5Ad4A1Mwo8qECo.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-13-222-21-231.compute-1.amazonaws.com' (ED25519) to the list of known hosts.
Welcome to Ubuntu 24.04.3 LTS (GNU/Linux 6.14.0-1015-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

System information as of Tue Nov 11 05:50:06 UTC 2025

  System load:  0.08           Temperature:      -273.1 °C
  Usage of /:   25.9% of 6.71GB Processes:          118
  Memory usage: 24%
  Swap usage:   0%            Users logged in:    0
                                         IPv4 address for ens5: 172.31.9.83

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

The list of available updates is more than a week old.
To check for new updates run: sudo apt update

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/*copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-172-31-9-83:~$
```

Step 16: check the docker and git version

If they are not present, then go to administrative terminal using command

“sudo su”

Then update using the command “sudo apt-get update”

```
ubuntu@ip-172-31-9-83:~$ docker --version
Command 'docker' not found, but can be installed with:
sudo snap install docker          # version 28.4.0, or
sudo snap install docker          # version 28.1.1+1
sudo apt install docker.io        # version 28.2.2-0ubuntu1~24.04.1
sudo apt install podman-docker    # version 4.9.3+ds1-1ubuntu0.2
See 'snap info <snapname>' for additional versions.
ubuntu@ip-172-31-9-83:~$ git --version
git version 2.43.0
ubuntu@ip-172-31-9-83:~$ sudo su
root@ip-172-31-9-83:/home/ubuntu# sudo apt-get update
```

Step 17: use command “sudo apt-get install docker.io” to install docker

```
Get:50 http://security.ubuntu.com/ubuntu noble-security/restricted amd64 Components [212 B]
Get:51 http://security.ubuntu.com/ubuntu noble-security/multiverse amd64 Packages [27.4 kB]
Get:52 http://security.ubuntu.com/ubuntu noble-security/multiverse Translation-en [5708 B]
Get:53 http://security.ubuntu.com/ubuntu noble-security/multiverse amd64 Components [212 B]
Get:54 http://security.ubuntu.com/ubuntu noble-security/multiverse amd64 c-n-f Metadata [384 B]
Fetched 38.6 MB in 6s (6197 kB/s)
Reading package lists... Done
root@ip-172-31-9-83:/home/ubuntu# sudo apt-get install docker.io
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  bridge-utils containerd dns-root-data dnsmasq-base pigz runc ubuntu-fan
Suggested packages:
  ifupdown aufs-tools cgroupfs-mount | cgroup-lite debootstrap docker-buildx docker-compose-v2 docker-doc rinse
  zfs-fuse | zfsutils
The following NEW packages will be installed:
  bridge-utils containerd dns-root-data dnsmasq-base docker.io pigz runc ubuntu-fan
0 upgraded, 8 newly installed, 0 to remove and 10 not upgraded.
Need to get 76.0 MB of archives.
After this operation, 288 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 pigz amd64 2.8-1 [65.6 kB]
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 bridge-utils amd64 1.7.1-1ubuntu2 [33.9 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 runc amd64 1.3.3-0ubuntu1~24.04.2 [8815 kB]
Get:4 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 containerd amd64 1.7.28-0ubuntu1~24.04.1 [38.4 MB]
Get:5 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 dns-root-data all 2024071801~ubuntu0.24.04.1 [5918 B]
Get:6 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 dnsmasq-base amd64 2.90-2ubuntu0.1 [376 kB]
Get:7 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe amd64 docker.io amd64 28.2.2-0ubuntu1~24.04.1 [28.3 MB]
Get:8 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe amd64 ubuntu-fan all 0.12.16+24.04.1 [34.2 kB]
Fetched 76.0 MB in 1s (81.3 MB/s)
Preconfiguring packages ...
Selecting previously unselected package pigz.
(Reading database ... 71735 files and directories currently installed.)
Preparing to unpack .../0-pigz_2.8-1_amd64.deb ...
Unpacking pigz (2.8-1) ...
Selecting previously unselected package bridge-utils.
Preparing to unpack .../1-bridge-utils_1.7.1-1ubuntu2_amd64.deb ...
Unpacking bridge-utils (1.7.1-1ubuntu2) ...
Selecting previously unselected package runc.
Preparing to unpack .../2-runc_1.3.3-0ubuntu1~24.04.2_amd64.deb ...
Unpacking runc (1.3.3-0ubuntu1~24.04.2)
```

Step 18: Clone the git repo that has maven project and change to that directory

```
No VM guests are running outdated hypervisor (qemu) binaries on this host.  
root@ip-172-31-9-83:/home/ubuntu# git clone https://github.com/Gayathri2608-hub/maven-practice.git  
Cloning into 'maven-practice'...  
remote: Enumerating objects: 60, done.  
remote: Counting objects: 100% (60/60), done.  
remote: Compressing objects: 100% (40/40), done.  
remote: Total 60 (delta 11), reused 34 (delta 2), pack-reused 0 (from 0)  
Receiving objects: 100% (60/60), 13.39 KiB | 3.35 MiB/s, done.  
Resolving deltas: 100% (11/11), done.  
root@ip-172-31-9-83:/home/ubuntu# ls  
maven-practice  
root@ip-172-31-9-83:/home/ubuntu# cd maven-practice  
root@ip-172-31-9-83:/home/ubuntu/maven-practice# ls  
Dockerfile demo pom.xml readme src target  
root@ip-172-31-9-83:/home/ubuntu/maven-practice#
```

Step 19: change to the project directory and check for Dockerfile, if not present create the Dockerfile – “nano Dockerfile” and then build the image

“sudo docker build -t image_name .” name of image:img1

```
root@ip-172-31-9-83:/home/ubuntu/maven-practice# ls  
Dockerfile demo pom.xml readme src target  
root@ip-172-31-9-83:/home/ubuntu/maven-practice# sudo docker build -t dep1 .  
DEPRECATED: The legacy builder is deprecated and will be removed in a future release.  
Install the buildx component to build images with BuildKit:  
https://docs.docker.com/go/buildx/  
  
Sending build context to Docker daemon 120.8kB  
Step 1/4 : FROM tomcat:9.0  
9.0: Pulling from library/tomcat  
4b3ffd8ccb52: Pulling fs layer  
b48f960b380d: Pulling fs layer  
58424d8c3e86: Pulling fs layer  
4f4fb700ef54: Pulling fs layer  
37b617836889: Pulling fs layer  
891b6ad931b7: Pulling fs layer  
ac0beccecf50: Pulling fs layer  
4f4fb700ef54: Waiting  
37b617836889: Waiting  
891b6ad931b7: Waiting  
ac0beccecf50: Waiting  
b48f960b380d: Verifying Checksum  
b48f960b380d: Download complete  
4b3ffd8ccb52: Verifying Checksum  
4b3ffd8ccb52: Download complete  
4f4fb700ef54: Verifying Checksum  
4f4fb700ef54: Download complete  
37b617836889: Verifying Checksum  
37b617836889: Download complete  
891b6ad931b7: Verifying Checksum  
891b6ad931b7: Download complete  
ac0beccecf50: Verifying Checksum  
ac0beccecf50: Download complete  
58424d8c3e86: Verifying Checksum  
58424d8c3e86: Download complete  
4b3ffd8ccb52: Pull complete
```

Step 20: Run the image “sudo docker run -d -p 8081:8080 img1”

```
root@ip-172-31-9-83:/home/ubuntu/ar/maven-practice# sudo docker run -d -p 8081:8080 img1
c5fd91cf9a9b4f0625d4d2c0d896406e8da76929ed75a3f9ccc1699fb08535
root@ip-172-31-9-83:/home/ubuntu/ar/maven-practice#
```

Step 21: Check the images and the containers

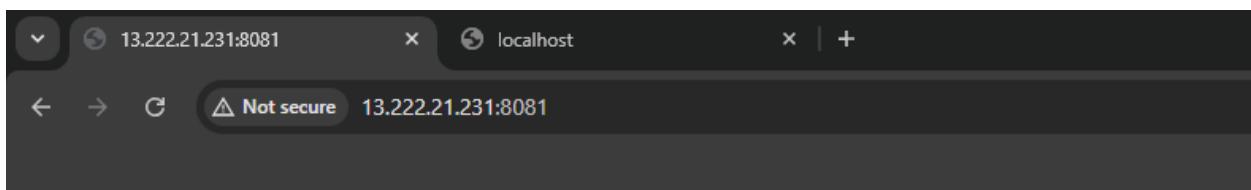
```
root@ip-172-31-9-83:/home/ubuntu/ar/maven-practice# sudo docker images
REPOSITORY      TAG      IMAGE ID      CREATED      SIZE
img1           latest    a67a112ce8ac  2 minutes ago  413MB
dep1           latest    28efbe56e633  29 minutes ago  413MB
tomcat         9.0      2a4887a16e43  12 hours ago   413MB
root@ip-172-31-9-83:/home/ubuntu/ar/maven-practice# docker ps
CONTAINER ID   IMAGE      COMMAND      CREATED      STATUS      PORTS
 NAMES
c5fd91cf9a9b  img1      "catalina.sh run"  About a minute ago  Up About a minute  0.0.0.0:8081->8080/tcp, [::]:8081->8080/tcp
84e7f9ce5ec2  dep1      "catalina.sh run"  9 minutes ago    Up 9 minutes  0.0.0.0:8080->8080/tcp, [::]:8080->8080/tcp
b62aedc8bb3b  dep1      "catalina.sh run"  27 minutes ago   Up 27 minutes  0.0.0.0:7070->8080/tcp, [::]:7070->8080/tcp
root@ip-172-31-9-83:/home/ubuntu/ar/maven-practice#
```

Step 22: Take the public IP address from the instances in AWS and open it in chrome along with the port number mapped.

Public IP- 13.222.21.231

Port used: 8081

Use: 13.222.21.231:8081, you will find your application that is deployed



Hello World! practice