

# FIRST BUILD JOB



Today, we will create and execute a Jenkins build job to automate a basic task in a CI/CD pipeline. We'll walk through the process of configuring the job, triggering builds, handling source code updates, and archiving build artifacts.

## Step-by-Step Process

### 1. Starting the Build Job:

- Initiate the build process by triggering the job in Jenkins. If there are changes in the source code, Jenkins will detect and pull the differences. If no changes exist, it will simply clone the latest version of the source code.
- The Git plugin is utilized to fetch only the changed parts, optimizing the build process.

### 2. Rebuilding:

- You can rebuild the project as needed. Each time a build is triggered, Jenkins will handle the latest state of the repository.

### 3. Configuring Post-Build Actions:

- After the build process completes, go to Configure in Jenkins and navigate to Post-build Actions. This section defines what Jenkins will do after the build is successful.

### 4. Archiving Artifacts:

- To ensure critical build files are saved, configure the job to archive any files with a `.war` extension (or any other format relevant to your project). Archiving allows you to store the build outputs for future use or deployment.

### 5. Saving and Building:

- After setting the configuration, save the changes and trigger a new build. This ensures the job is properly configured and will archive the desired artifacts.

## **6. Checking Artifacts:**

- Once the build is complete, navigate to the build job page and verify that the artifacts are archived. Jenkins will store these files for future reference or deployment.

## **7. Cleaning Workspace:**

- After the job is done, you can clean your Jenkins workspace to clear out unnecessary files. Despite this cleanup, the archived artifact will remain available as it was explicitly saved during the build process.

**By following these steps, you automate the process of building, testing, and saving artifacts in Jenkins, which is essential for continuous integration and delivery.**

## **Let's Get Started**

The screenshot shows the Jenkins dashboard with the following elements:

- Header:** Jenkins logo, Search bar (Search (⌘+K)), Help icon, Aakansha (User), and Log Out button.
- Left Sidebar:** Dashboard >, New Item, Build History, Manage Jenkins, and My Views.
- Welcome Section:** "Welcome to Jenkins!" message, "Start building your software project" button, and "Create a job" button.
- Build Queue:** "Build Queue" dropdown, "No builds in the queue." message.
- Build Executor Status:** "Build Executor Status" dropdown, showing 1 Idle and 2 Idle.
- Setup Section:** "Set up a distributed build" section with "Set up an agent" (monitor icon), "Configure a cloud" (cloud icon), and "Learn more about distributed builds" (help icon).

 Jenkins

Search (⌘+K) ? Aakansha log out

Dashboard > New Item

## New Item

Enter an item name

Select an item type

 **Freestyle project**  
Classic, general-purpose job type that checks out from up to one SCM, executes build steps serially, followed by post-build steps like archiving artifacts and sending email notifications.

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

 **Multi-configuration project**  
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

 **Folder**  
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.

OK

 Jenkins

Search (⌘+K) ? Aakansha log out

Dashboard > Build > Configuration

## Configure

### General

Enabled ✓

**General**

Description

Plain text [Preview](#)

Discard old builds ?

GitHub project

This project is parameterized ?

Throttle builds ?

Execute concurrent builds if necessary ?

JDK

JDK to be used for this project

Save Apply

Dashboard > Build > Configuration

## Configure

Source Code Management

General

Source Code Management

Build Triggers

Build Environment

Build Steps

Post-build Actions

Git

None

Repositories

Repository URL: `https://github.com/devopshydclub/vprofile-project.git`

Credentials: - none -

+ Add

Advanced

Add Repository

Branches to build

Branch Specifier (blank for 'any'): `*/main`

This screenshot shows the 'Source Code Management' section of a build configuration. The 'Git' provider is selected. The 'Repository URL' field contains the value 'https://github.com/devopshydclub/vprofile-project.git'. There are sections for 'Credentials' (set to none) and 'Advanced' settings, along with buttons for 'Add Repository' and 'Add Branch'. A 'Branches to build' section at the bottom has a 'Branch Specifier' field containing '\*/main'.

Dashboard > Build > Configuration

## Configure

General

Source Code Management

Build Triggers

Build Environment

Build Steps

Post-build Actions

Build Triggers

Branches to build

Branch Specifier (blank for 'any'): `*/main`

Add Branch

Repository browser: (Auto)

Additional Behaviours

Add

This screenshot shows the 'Build Triggers' section of a build configuration. It includes a 'Branches to build' section with a 'Branch Specifier' field containing '\*/main', an 'Add Branch' button, and a 'Repository browser' dropdown set to '(Auto)'. There is also an 'Additional Behaviours' section with an 'Add' button.

Dashboard > Build > Configuration

## Configure

Build Steps

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

Invoke top-level Maven targets

Maven Version: MAVEN3

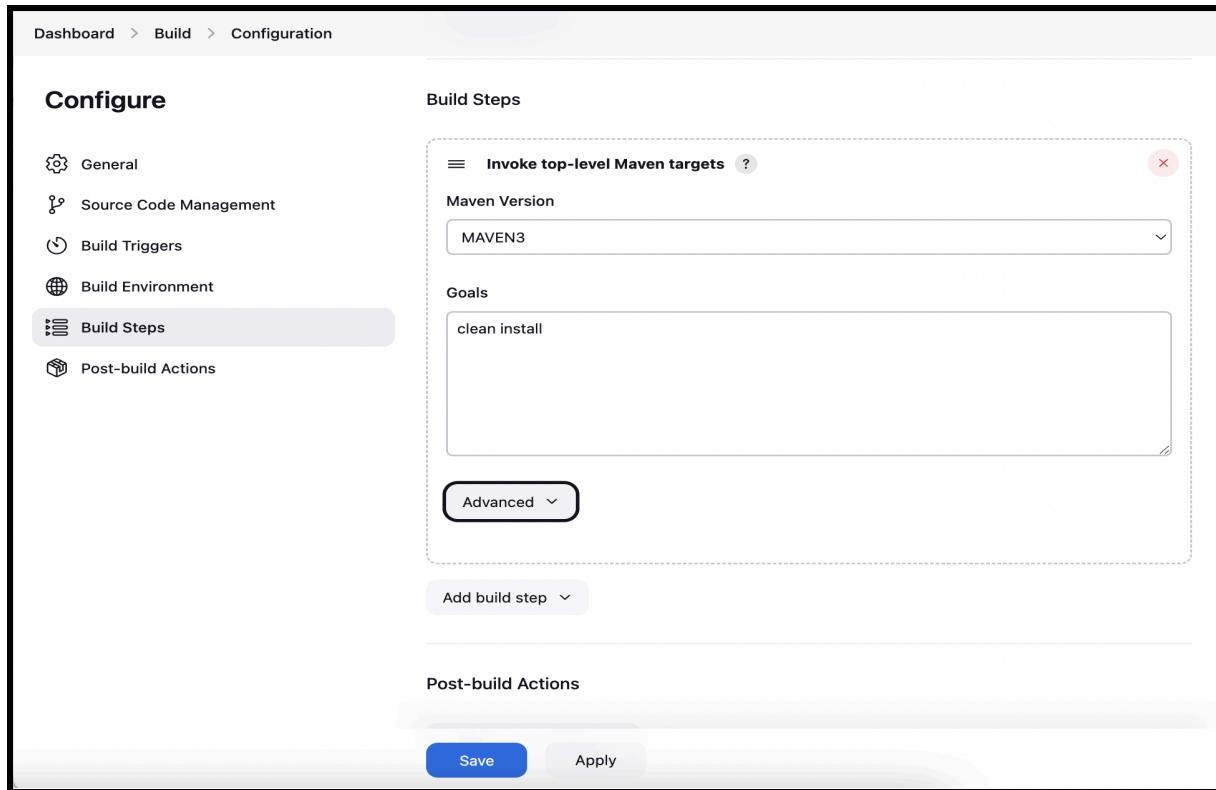
Goals: clean install

Advanced

Add build step

Post-build Actions

Save Apply



Jenkins

Dashboard > Build > #1 > Console Output

Status  
Changes  
**Console Output**  
Edit Build Information  
Delete build '#1'  
Timings  
Git Build Data

## Console Output

Skipping 379 KB.. [Full Log](#)

Downloading from central:  
<https://repo.maven.apache.org/maven2/org/apache/maven/doxia/doxia-decoration-model/1.1.2/doxia-decoration-model-1.1.2.pom>  
Progress (1): 3.0 kB

Downloaded from central:  
<https://repo.maven.apache.org/maven2/org/apache/maven/doxia/doxia-decoration-model/1.1.2/doxia-decoration-model-1.1.2.pom> (3.0 kB at 595 kB/s)

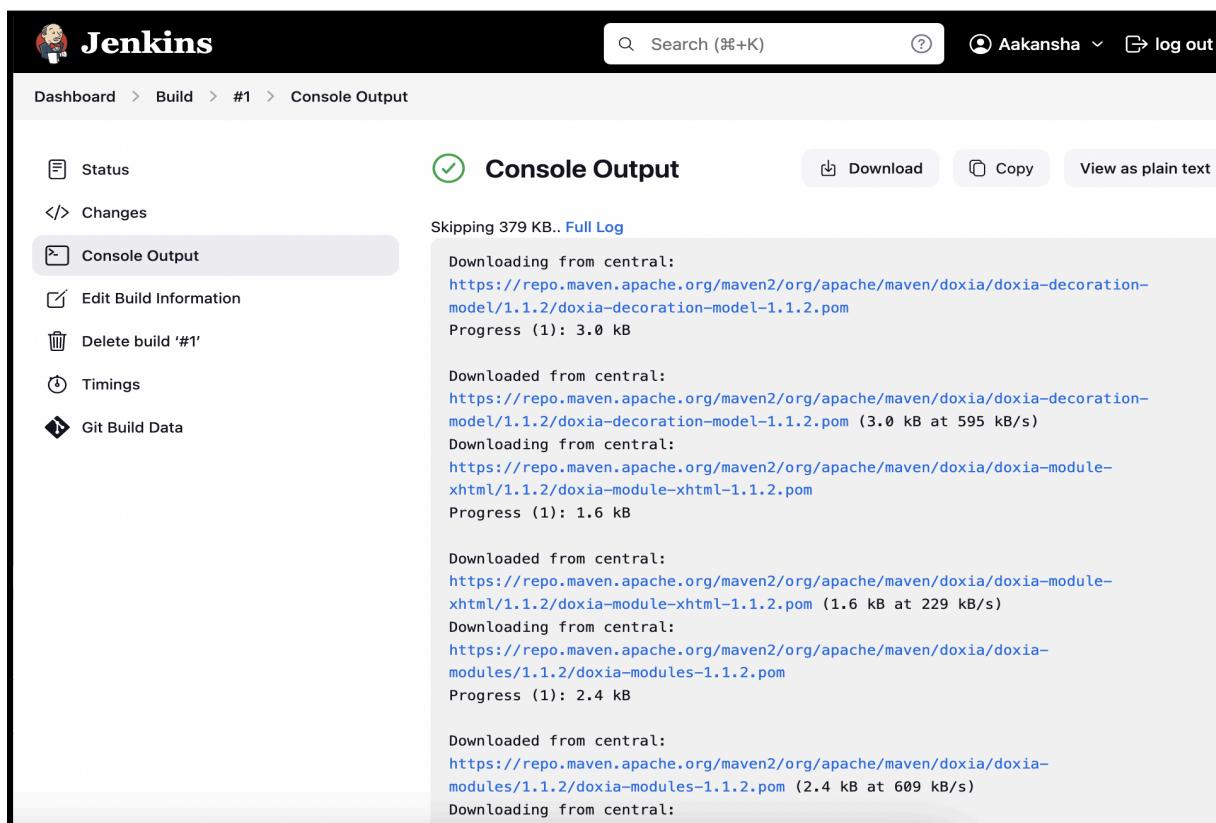
Downloading from central:  
<https://repo.maven.apache.org/maven2/org/apache/maven/doxia/doxia-module-xhtml/1.1.2/doxia-module-xhtml-1.1.2.pom>  
Progress (1): 1.6 kB

Downloaded from central:  
<https://repo.maven.apache.org/maven2/org/apache/maven/doxia/doxia-module-xhtml/1.1.2/doxia-module-xhtml-1.1.2.pom> (1.6 kB at 229 kB/s)

Downloading from central:  
<https://repo.maven.apache.org/maven2/org/apache/maven/doxia/doxia-modules/1.1.2/doxia-modules-1.1.2.pom>  
Progress (1): 2.4 kB

Downloaded from central:  
<https://repo.maven.apache.org/maven2/org/apache/maven/doxia/doxia-modules/1.1.2/doxia-modules-1.1.2.pom> (2.4 kB at 609 kB/s)

Downloading from central:



# Jenkins

Dashboard >

+ New Item      Add description

Build History      All +

Manage Jenkins

My Views

Build Queue: No builds in the queue.

Build Executor Status: 1 Idle, 2 Idle

S	W	Name ↓	Last Success	Last Failure	Last Duration
✓	☀️	Build	2 min 34 sec #1	N/A	27 sec

Icon: S M L      ...

The dashboard shows a single build named 'Build' with a status of 'Success' (green checkmark) and a duration of '2 min 34 sec'. The build number is '#1'. There are no other builds in the queue or executor status.

# Jenkins

Dashboard > Build >

Status      Build      Edit description

</> Changes      Build artifact from vprofile source code

Workspace

Build Now

Configure

Delete Project

Rename

Build History      trend

Filter... /

#1      Sep 25, 2024, 12:33 PM

Atom feed for all Atom feed for failures

The build page for 'Build' shows the build history with one entry. The build was successful on Sep 25, 2024, at 12:33 PM. It includes links for atom feeds.

Jenkins

Search (⌘+K)

Aakansha log out

Dashboard > Build > Workspace of Build on Built-In Node

## Workspace of Build on Built-In Node

Status Changes Workspace

Build /  →

git ansible src target vagrant

Jenkinsfile Sep 25, 2024, 12:33:13 PM 3.84 KiB ↗  
pom.xml Sep 25, 2024, 12:33:13 PM 7.14 KiB ↗  
README.md Sep 25, 2024, 12:33:13 PM 517 B ↗

Wipe Out Current Workspace Build Now Configure Delete Project Rename

(all files in zip)

Build History trend Filter... #1 Sep 25, 2024, 12:33 PM Atom feed for all Atom feed for failures

Jenkins

Search (⌘+K)

Aakansha log out

Dashboard > Build > Workspace of Build on Built-In Node

## Workspace of Build on Built-In Node

Status Changes Workspace

Build /  →

git ansible src target

Wipe Out Current Workspace

Are you sure about wiping out the workspace?

Cancel Yes

Build History trend Filter... #1 Sep 25, 2024, 12:33 PM Atom feed for all Atom feed for failures

## Build Job Execution with Git Integration.

The screenshot shows the Jenkins interface. At the top, there's a navigation bar with a Jenkins logo, a search bar containing 'Search (⌘+K)', and a user dropdown for 'Aakansha'. Below the navigation bar, the page title is 'Dashboard > Build > Error'. On the left, a sidebar lists project actions: Status, Changes, Workspace (which is selected and highlighted in grey), Wipe Out Current Workspace, Build Now, Configure, Delete Project, and Rename. In the main content area, the title 'Error: no workspace' is displayed. A message states: 'There's no workspace for this project. Possible reasons are:' followed by a numbered list: 1. The project was renamed recently and no build was done under the new name. 2. The agent this project has run on for the last time was removed. 3. The workspace directory (/var/lib/jenkins/workspace/Build) is removed outside Jenkins. 4. The workspace was wiped out and no build has been done since then. Below this, a note says 'Run a build to have Jenkins create a workspace.' At the bottom of the main content area, there's a 'Build History' section with a table showing one build entry: '#1 | Sep 25, 2024, 12:33 PM'. There are also links for 'Atom feed for all' and 'Atom feed for failures'.

- Now, when we trigger the build, Jenkins will initiate the process by fetching all new data from the configured Git repository. If there are any changes in the source code, Jenkins, with the help of the Git plugin, will automatically pull only the modified portions, ensuring efficient and fast updates. If no changes are detected, Jenkins will simply clone the source code to ensure the latest version is available for the build.
- This step ensures that the workspace is always up-to-date with the latest version of the code, ready for compilation or further deployment steps.

**Note:** If the workspace has been wiped out (as shown in the error), triggering a new build will automatically recreate the workspace and populate it with the necessary data.

**Let's proceed to build the project again.**

The screenshot shows the Jenkins workspace interface for a build on a built-in node. The left sidebar contains links for Status, Changes, Workspace (which is selected), Build Now, Configure, Delete Project, and Rename. The main area is titled "Workspace of Build on Built-In Node" and shows a file tree with .git, ansible, src, target, and vagrant directories, along with Jenkinsfile, pom.xml, and README.md files. A "Build /" input field with a search icon is also present. Below the file tree, there is a "Build History" section with two entries: #2 (Sep 25, 2024, 12:41 PM) and #1 (Sep 25, 2024, 12:33 PM). At the bottom, there are links for Atom feed for all and Atom feed for failures.

Dashboard > Build > Workspace of Build on Built-In Node

**Workspace of Build on Built-In Node**

Status Changes Workspace Build / →

Wipe Out Current Workspace

Build Now

Configure

Delete Project

Rename

(all files in zip)

**Build History** trend

Filter... /

#2 | Sep 25, 2024, 12:41 PM

#1 | Sep 25, 2024, 12:33 PM

Atom feed for all Atom feed for failures

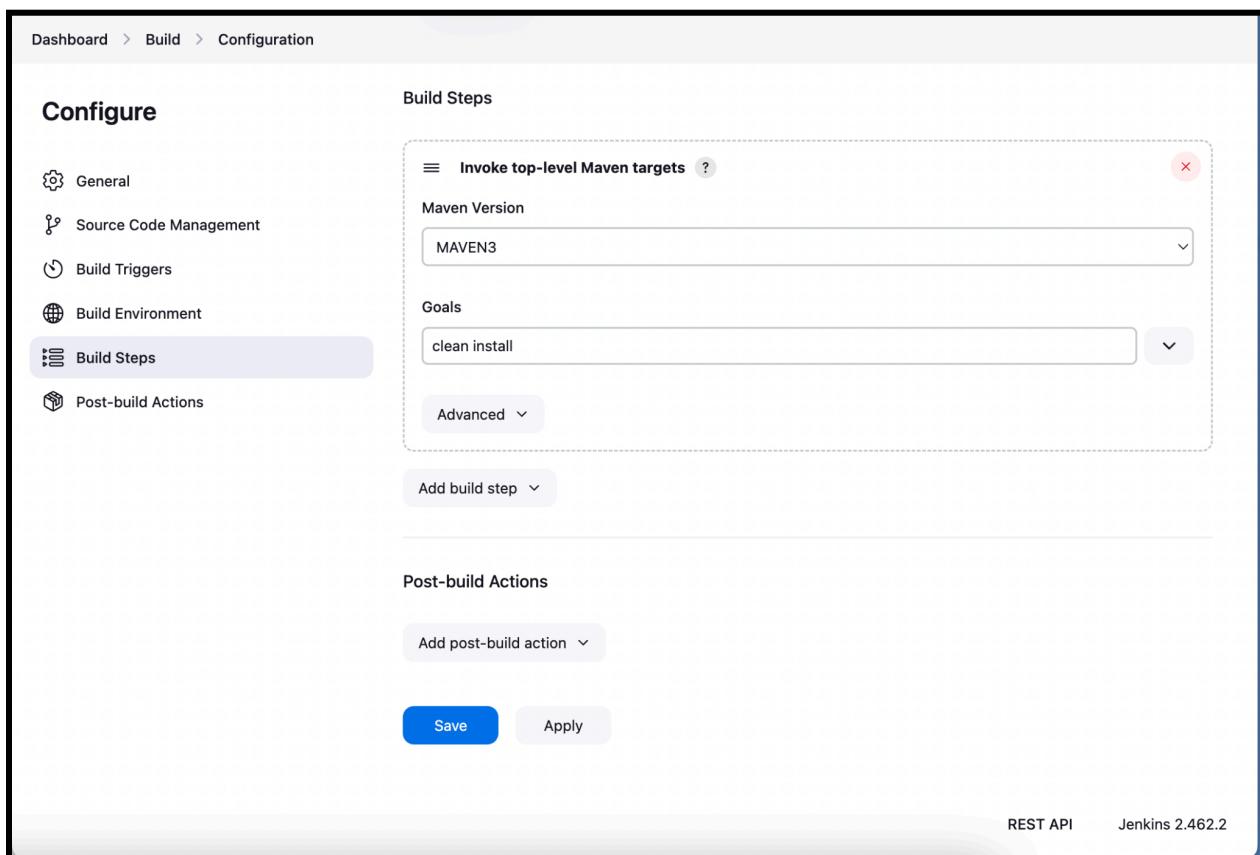
## Configuring Post-Build Actions

### 1. Click on "Configure":

- Begin by navigating to your Jenkins job and selecting the Configure option. This allows you to modify the job's settings and behavior.

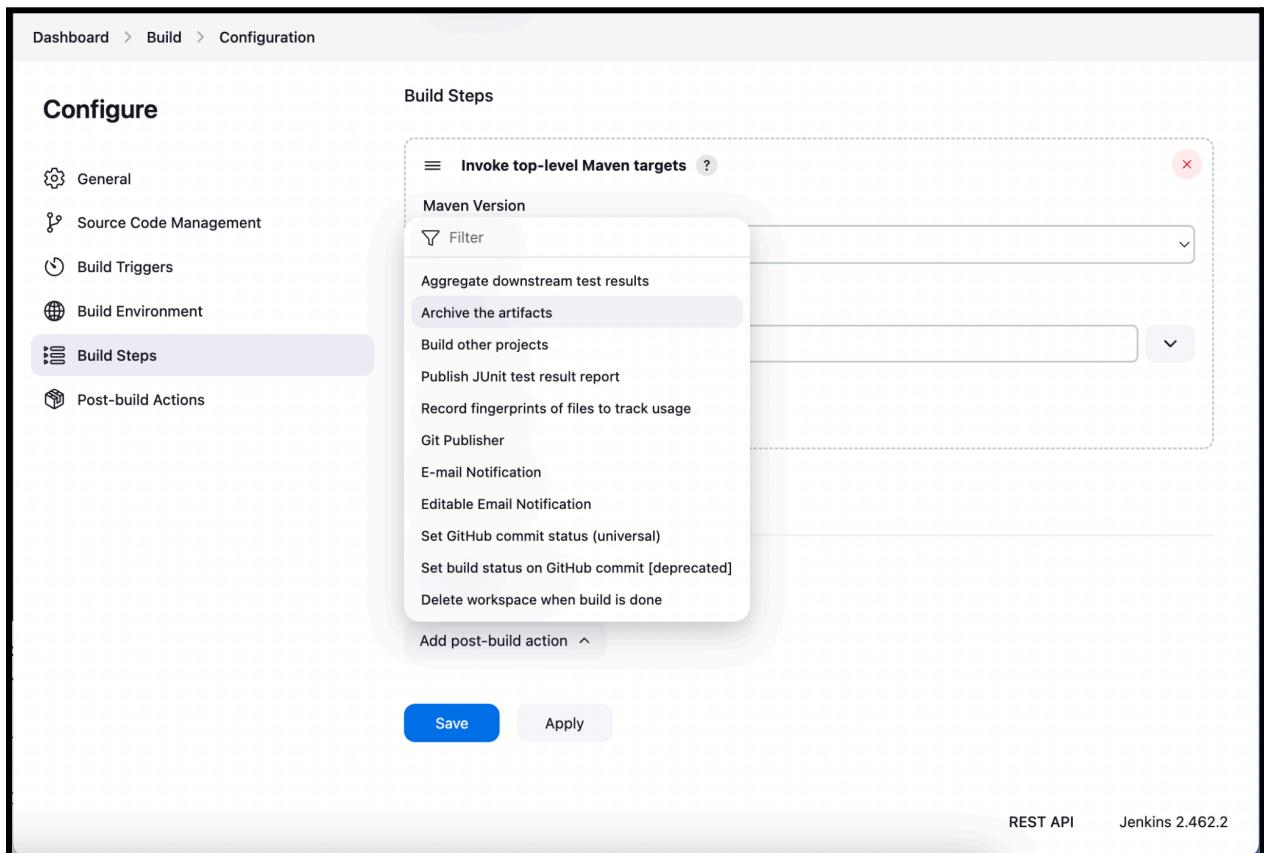
### 2. Navigate to Post-Build Actions:

- Scroll down to the Post-Build Actions section. This area defines what Jenkins will do after successfully completing the build process. Post-build actions are essential for tasks like archiving artifacts, sending notifications, or triggering additional jobs.



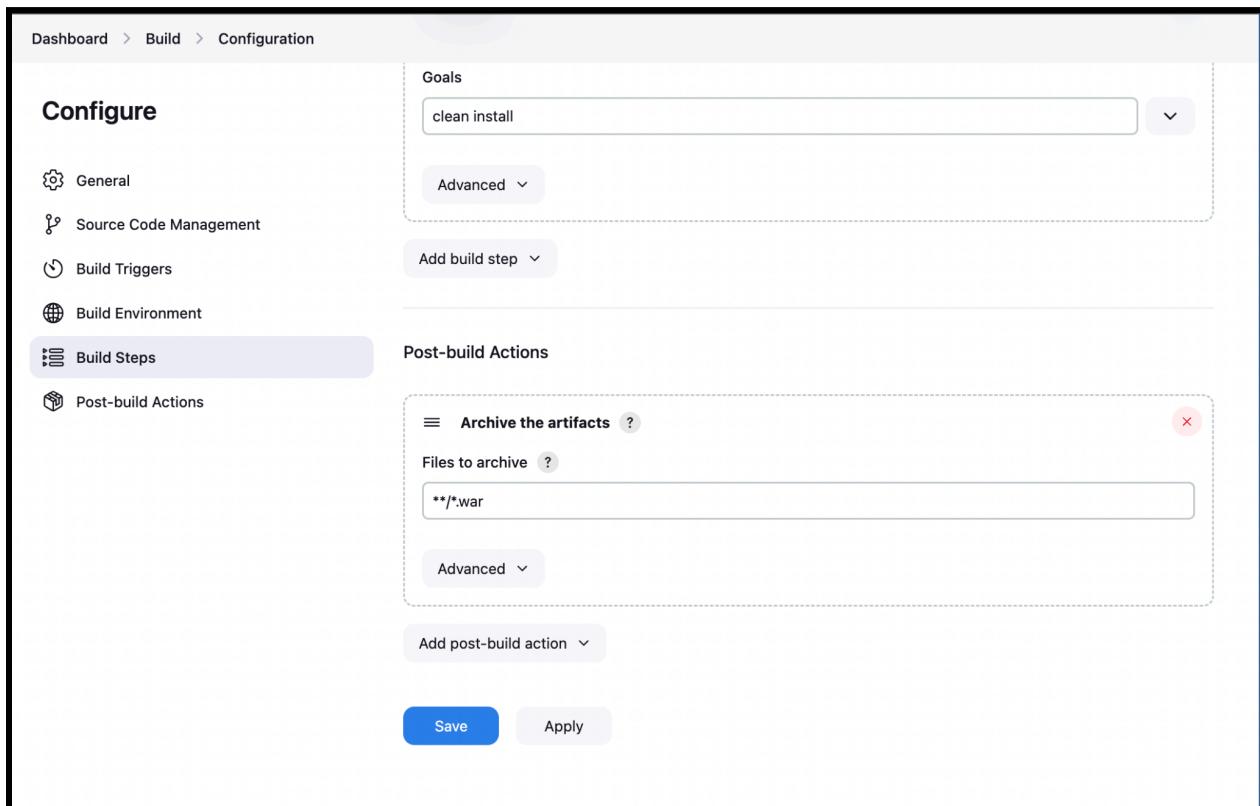
## Now what we can do after the build step is completed:

- **Archiving Artifacts:** After the build is complete, you can specify that Jenkins should look for files with a `.war` extension or other relevant formats. These files represent the output of the build (such as packaged applications) and need to be stored for future use or deployment.
- **Recursive Search:** Jenkins can also be configured to look for artifacts recursively in specific directories and subdirectories.



## Archive Files with `.war` Extension:

- The key focus here is to archive any files that have the `.war` extension. WAR files (Web Application Archives) are typically used in Java-based applications and contain all the components of a web application (e.g., classes, libraries, and configuration files).
- By archiving these `.war` files, Jenkins will store them as build artifacts, ensuring they are preserved for future deployment or download.



**Click on Save.**

## Start the Build.

The screenshot shows the Jenkins interface for a project named "Build".

**Left Sidebar:**

- Status
- </> Changes
- Workspace
- ▷ Build Now
- ⚙️ Configure
- Trash Delete Project
- Pencil Rename

**Right Content Area:**

**Build Status:** ✓ **Build**

Build artifact from vprofile source code

**Permalinks:**

- [Last build \(#2\), 5 min 19 sec ago](#)
- [Last stable build \(#2\), 5 min 19 sec ago](#)
- [Last successful build \(#2\), 5 min 19 sec ago](#)
- [Last completed build \(#2\), 5 min 19 sec ago](#)

**Build History:**

#	Date
#3	Sep 25, 2024, 12:47 PM
#2	Sep 25, 2024, 12:41 PM
#1	Sep 25, 2024, 12:33 PM

Filter... /

Atom feed for all Atom feed for failures

## Check the Archived Artifact:

- After the build completes, navigate to the Build History section and click on the most recent build entry. In the build details page, look for the Archived Artifacts section.
- Here, you should see the `.war` file (or other configured file types) that Jenkins has saved from the build. You can download this file directly or reference it in subsequent steps for deployment.

The screenshot shows the Jenkins interface for a project named "Build". The top navigation bar includes links for "Dashboard", "Build", "Status", "Changes", "Workspace", "Build Now", "Configure", "Delete Project", and "Rename". On the right, there's a search bar, user profile for "Aakansha", and a "log out" button. The main content area is titled "Build" with a green checkmark icon. It displays the message "Build artifact from vprofile source code". Below this, under "Last Successful Artifacts", is a link to "vprofile-v2.war" (52.13 MiB). A "view" link is also present. A "Permalinks" section lists the last four builds. The "Build History" sidebar on the left shows three builds: #3 (Sep 25, 2024, 12:47 PM), #2 (Sep 25, 2024, 12:41 PM), and #1 (Sep 25, 2024, 12:33 PM). At the bottom of the sidebar are links for "Atom feed for all" and "Atom feed for failures". The footer of the page shows the URL "54.161.226.94:8080/job/Build/lastSuccessfulBuild/artifact/target/vprofile-v2.war" and the Jenkins version "Jenkins 2.402.0".

**This step confirms that your build artifacts were successfully archived and are accessible, ensuring that critical outputs like your web application archives are stored safely for future use.**

## Cleaning the Workspace

### 1. Clean Your Workspace:

- After confirming the archived artifacts, go back to the Jenkins job page and click on Wipe Out Current Workspace. This will delete all files and folders in the workspace directory to free up space and prepare for future builds.

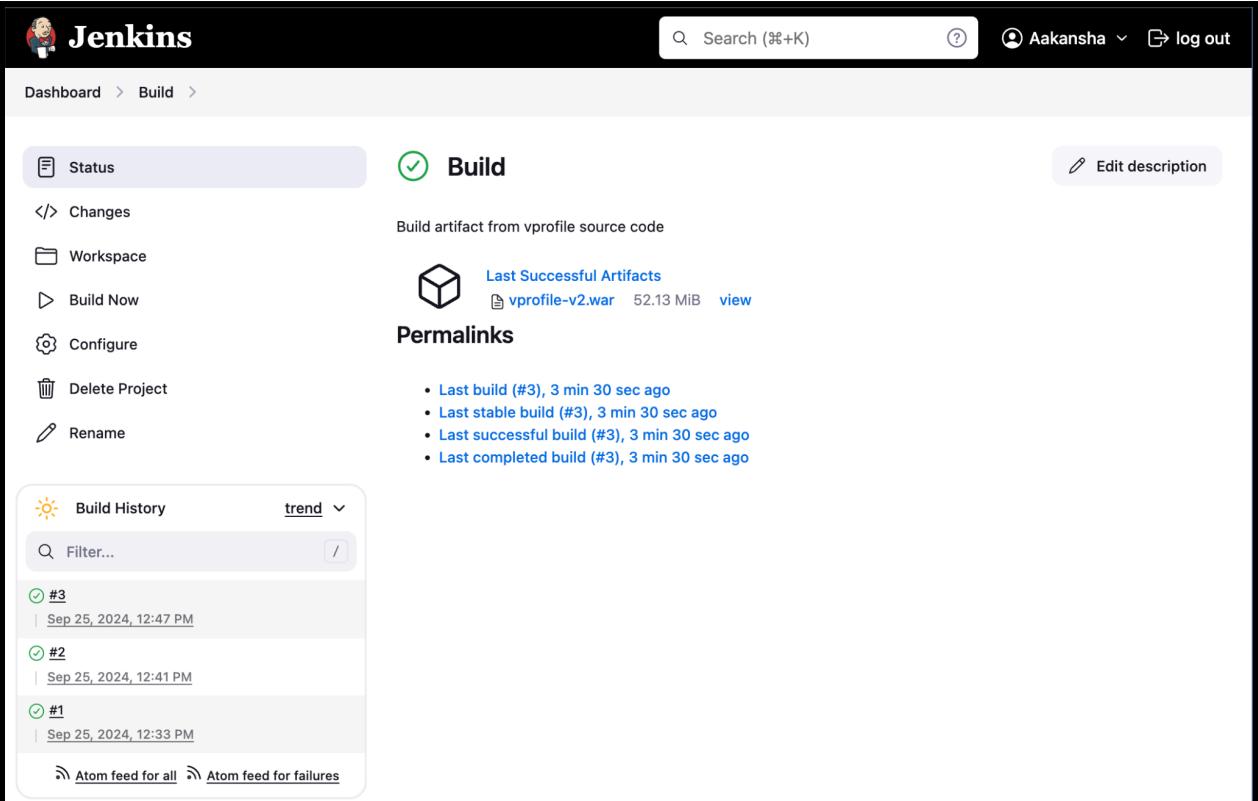
### 2. Path for Cleaning:

- The path where Jenkins stores the workspace files is usually located under `/var/lib/jenkins/workspace/YourProjectName` by default (depending on your Jenkins setup).

The screenshot shows the Jenkins interface with the following details:

- Header:** Jenkins logo, Search bar (Search (⌘+K)), User profile (Aakansha).
- Breadcrumbs:** Dashboard > Build > Error
- Left Sidebar:** Status, Changes, **Workspace** (selected), Wipe Out Current Workspace, Build Now, Configure, Delete Project, Rename.
- Right Content Area:**
  - Error: no workspace**
  - Text: There's no workspace for this project. Possible reasons are:
    1. The project was renamed recently and no build was done under the new name.
    2. The agent this project has run on for the last time was removed.
    3. The workspace directory (/var/lib/jenkins/workspace/Build) is removed outside Jenkins.
    4. The workspace was wiped out and no build has been done since then.
  - Text: Run a build to have Jenkins create a workspace.
- Bottom Panel:** Build History (dropdown: trend), Filter...
  - #3 | Sep 25, 2024, 12:47 PM
  - #2 | Sep 25, 2024, 12:41 PM
  - #1 | Sep 25, 2024, 12:33 PMAtom feed for all | Atom feed for failures

## Verifying Archived Artifacts After Workspace Cleanup



The screenshot shows the Jenkins interface for a build named "Build". The left sidebar has options like Status, Changes, Workspace, Build Now, Configure, Delete Project, and Rename. The main area shows a green checkmark icon and the word "Build". Below it, it says "Build artifact from vprofile source code". A "Last Successful Artifacts" section shows a cube icon and a file icon next to "vprofile-v2.war 52.13 MiB view". Under "Permalinks", there's a list of four recent builds: #3, #2, #1, and completed build #3. On the left, a "Build History" sidebar lists three builds: #3 (Sep 25, 2024, 12:47 PM), #2 (Sep 25, 2024, 12:41 PM), and #1 (Sep 25, 2024, 12:33 PM). At the bottom of the sidebar are links for "Atom feed for all" and "Atom feed for failures".

Even after cleaning the workspace, we can see that the archived artifacts remain intact. This is because Jenkins saves the archived files separately from the workspace, ensuring they are preserved regardless of workspace cleanup.

