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.

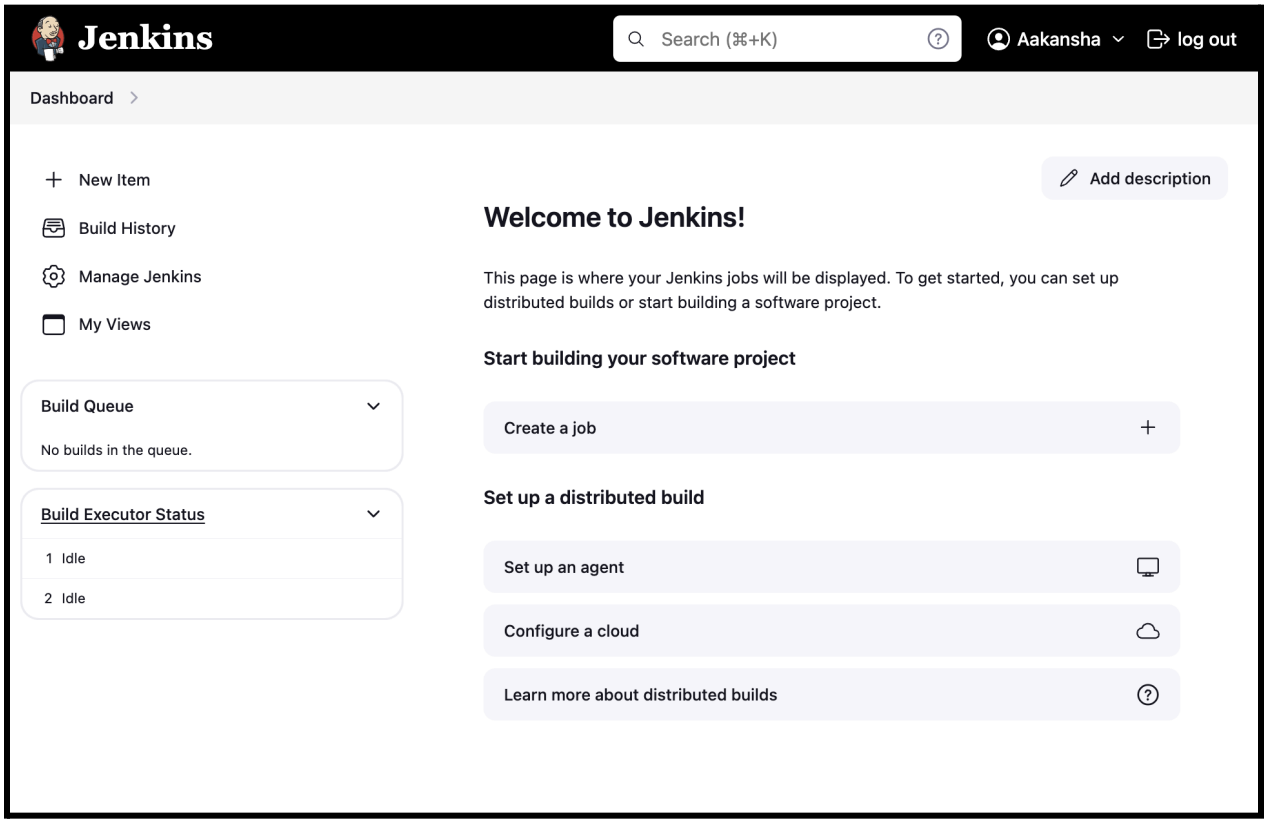
Step-by-Step Process

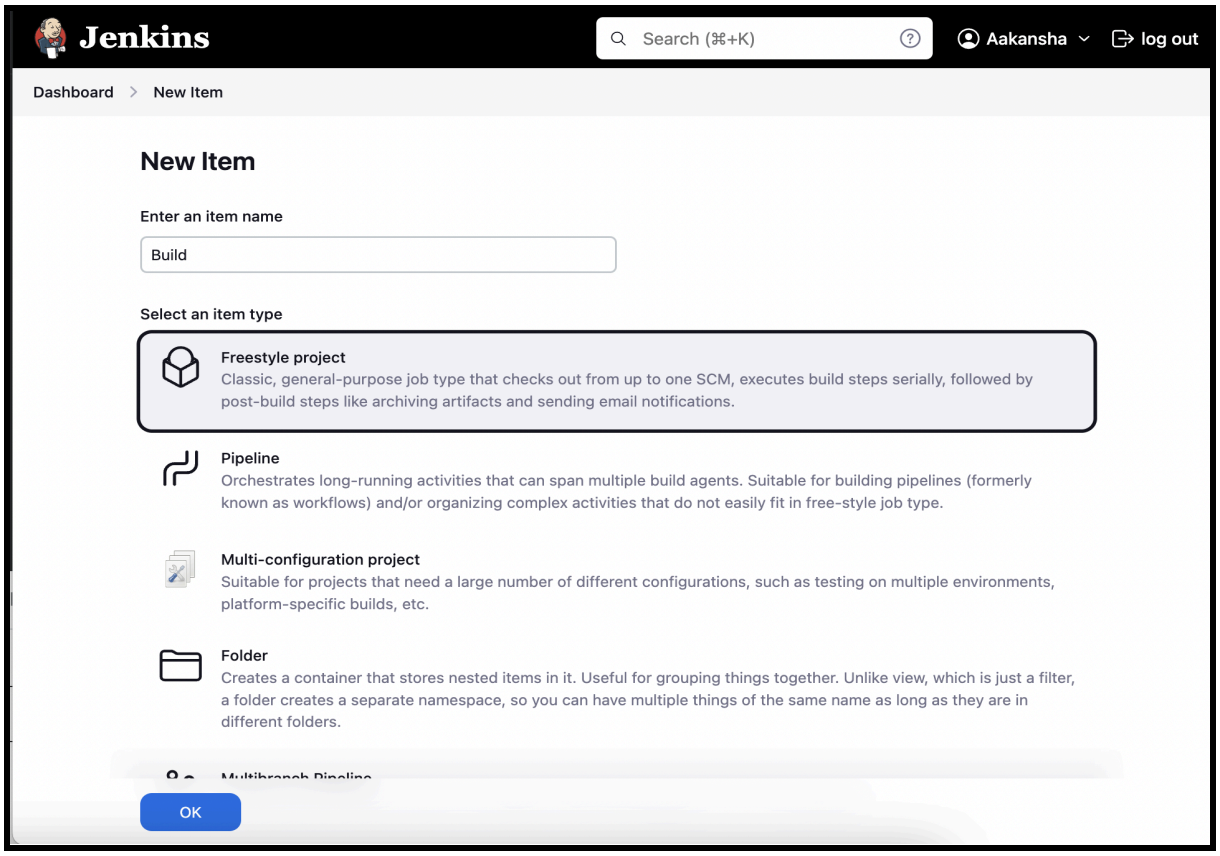
1. **Starting the Build Job:**
    - Initiate the build process by triggering the job in Jenkins. If there are changes in the source code, Jenkins will detect and pull the differences. If no changes exist, it will simply clone the latest version of the source code.
    - The Git plugin is utilized to 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.

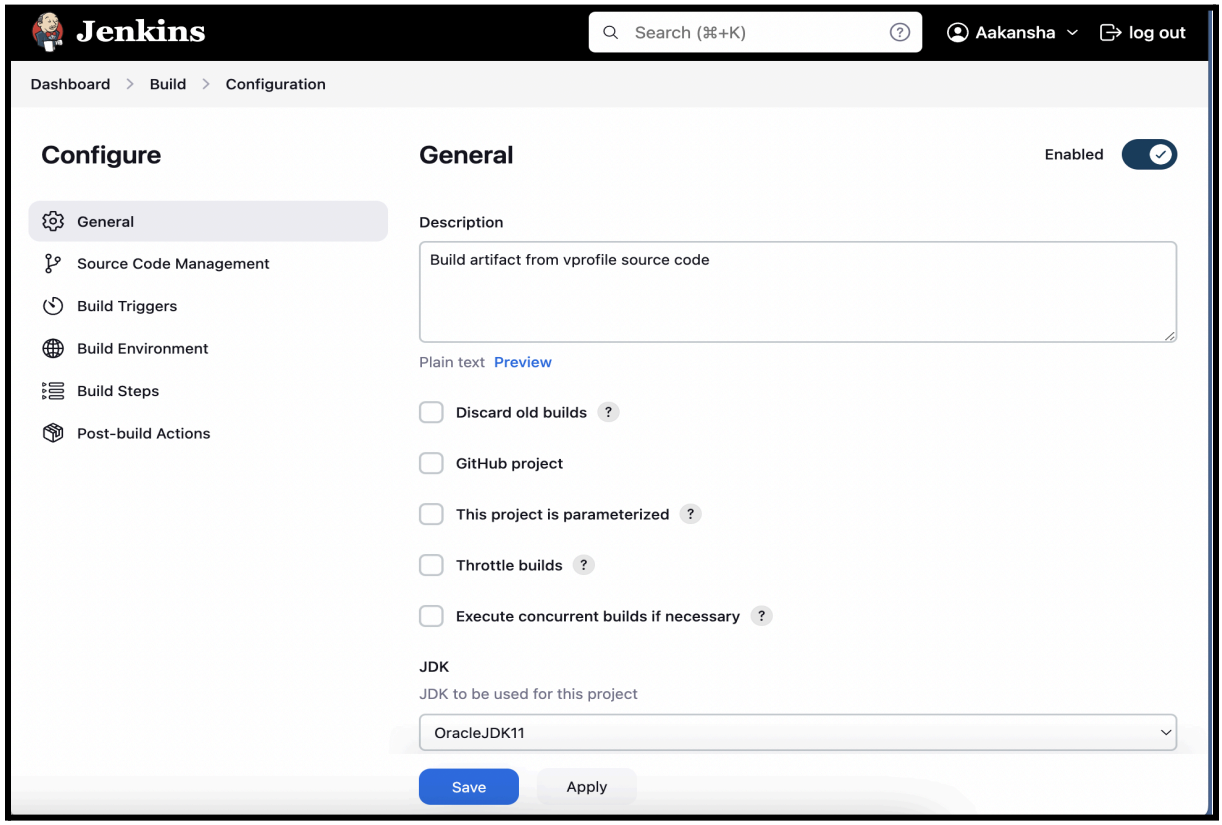
6. **Checking Artifacts:**
    - Once the build is complete, navigate to the build job page and verify that the artifacts are archived. Jenkins will store these files for future reference or deployment.
7. **Cleaning Workspace:**
    - After the job is done, you can clean your Jenkins workspace to clear out unnecessary files. Despite this cleanup, the archived artifact will remain available as it was explicitly saved during the build process.

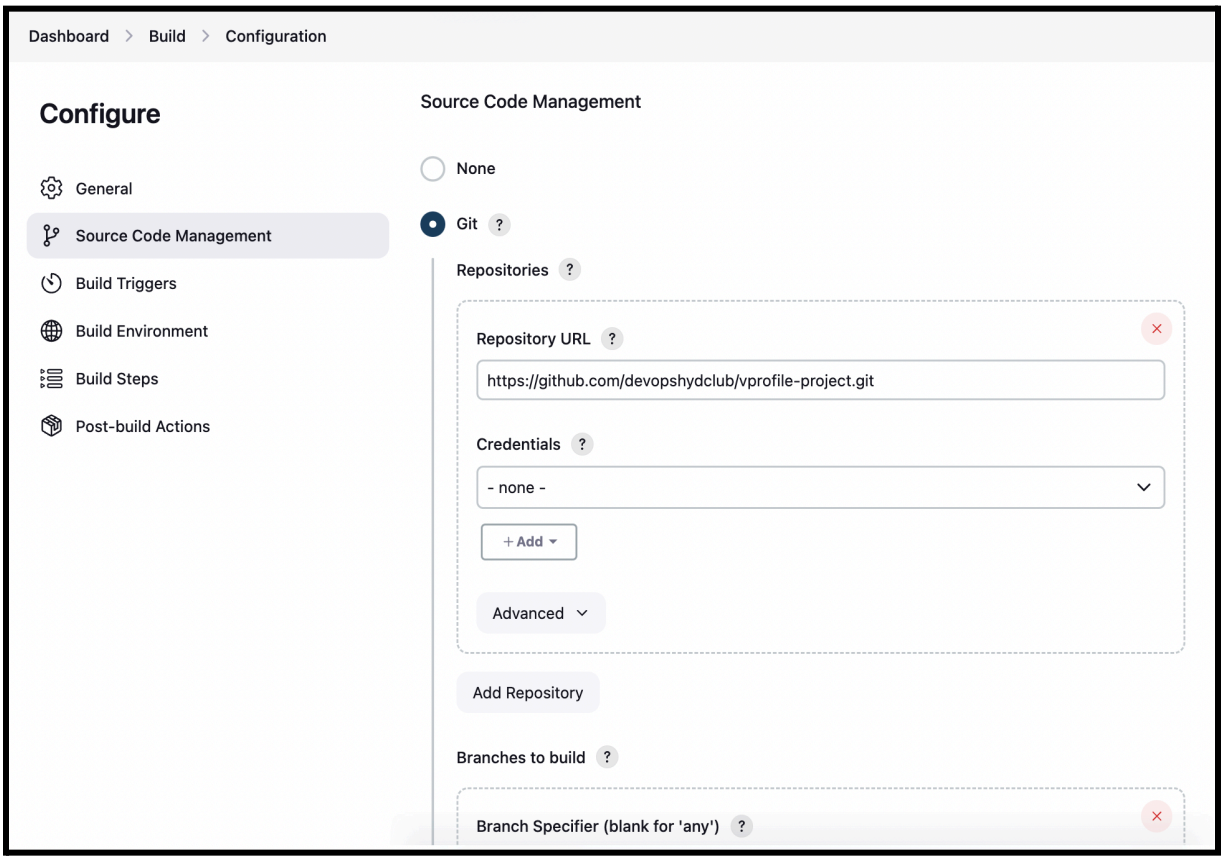
By following these steps, you automate the process of building, testing, and saving artifacts in Jenkins, which is essential for continuous integration and delivery.

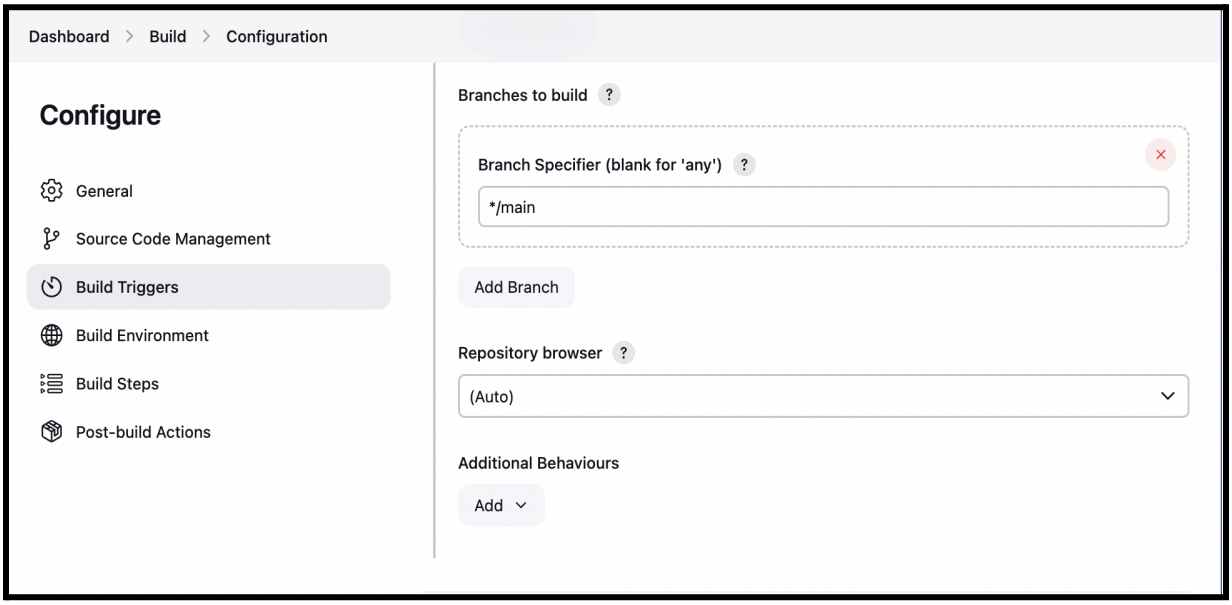
Let's Get Started

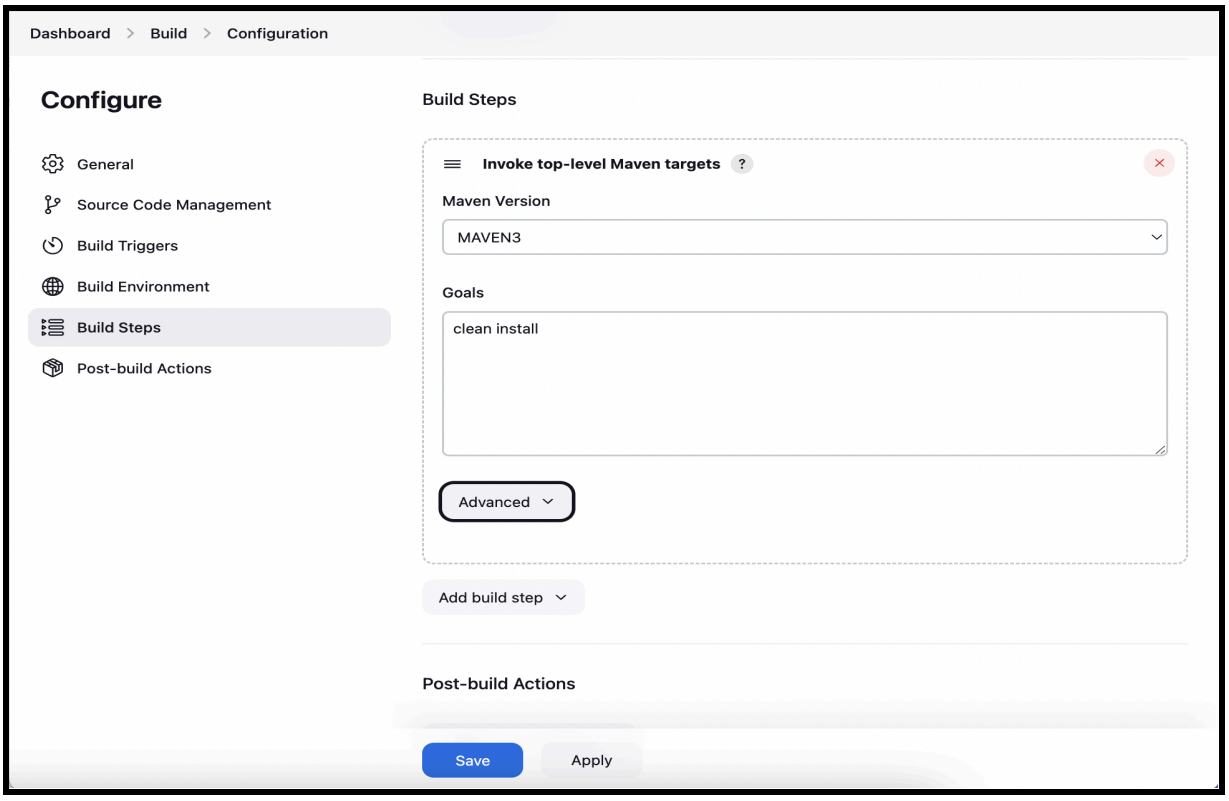


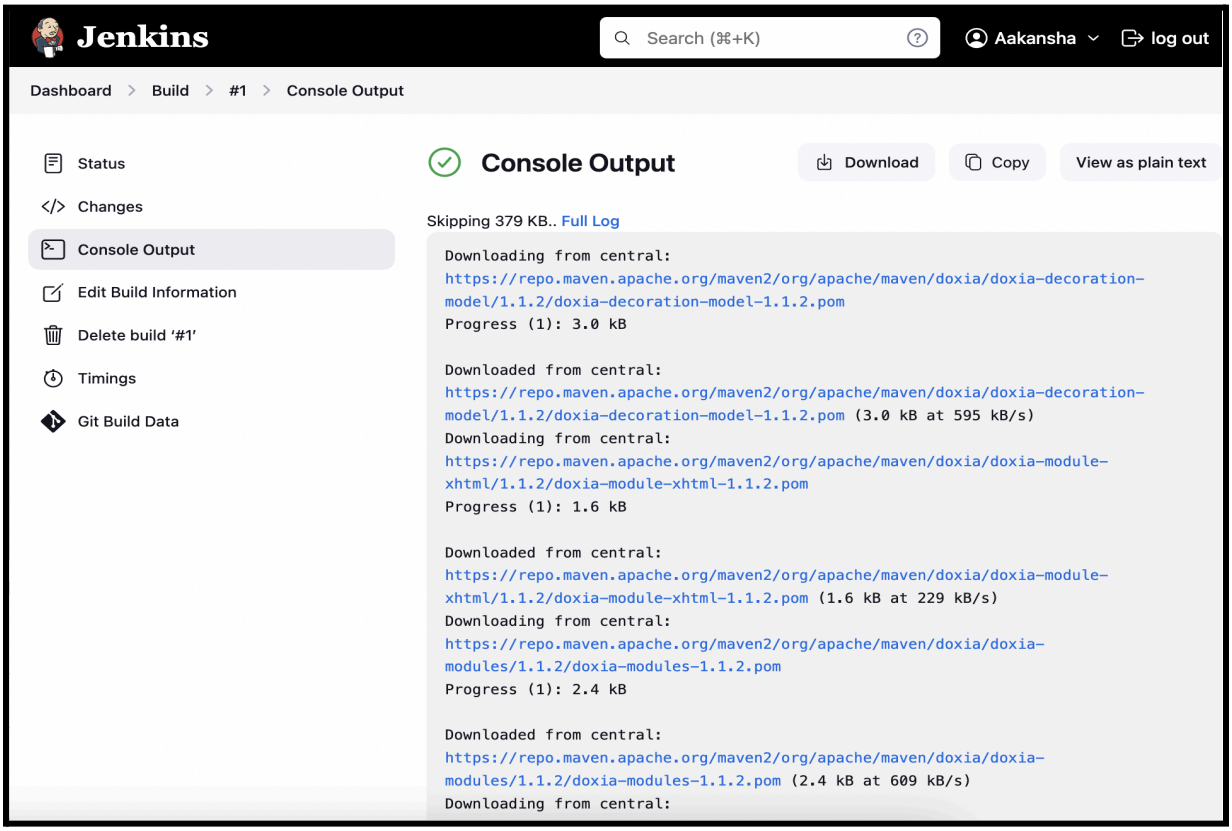

Jenkins
Search (⌘+K)
Aakansha
log out
Dashboard
New Item
New Item
Enter an item name
Build
Select an item type
Freestyle project
Classic, general-purpose job type that checks out from up to one SCM, executes build steps serially, followed by post-build steps like archiving artifacts and sending email notifications.
Pipeline
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.
Multi-configuration project
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.
Folder
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.
OK

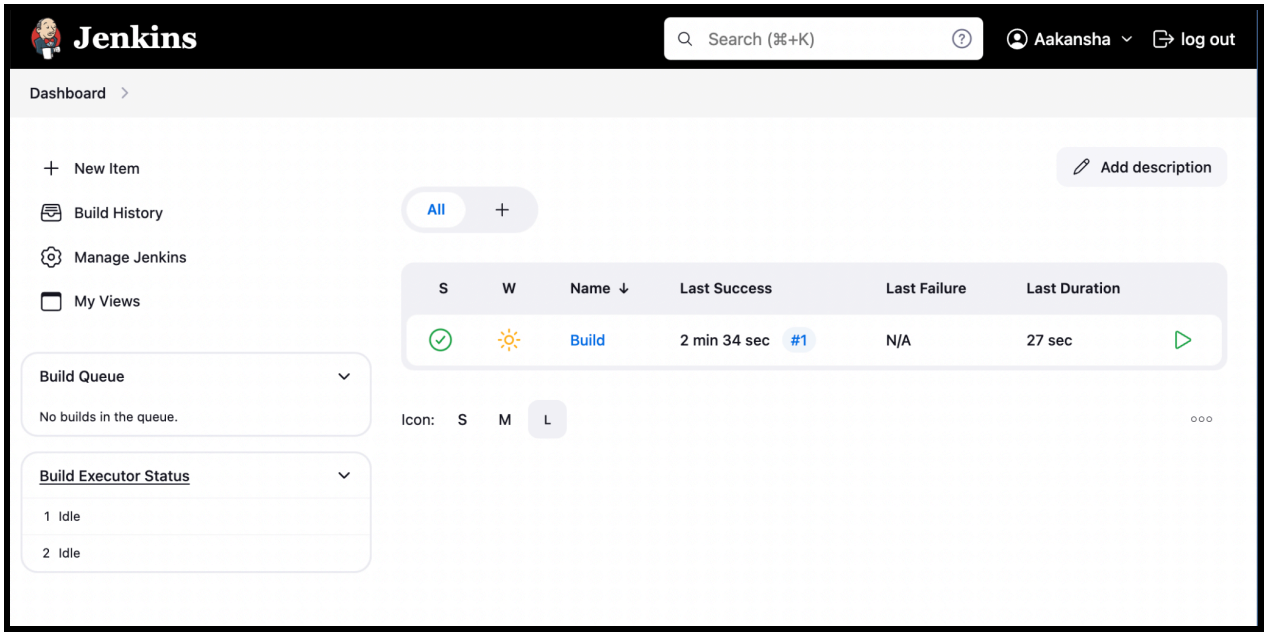

Jenkins
Search (⌘+K)
Aakansha
log out
Dashboard
Build
Configuration
Configure
General
Source Code Management
Build Triggers
Build Environment
Build Steps
Post-build Actions
General
Enabled
Description
Build artifact from vprofile source code
Plain text
Preview
Discard old builds
GitHub project
This project is parameterized
Throttle builds
Execute concurrent builds if necessary
JDK
JDK to be used for this project
OracleJDK11
Save
Apply

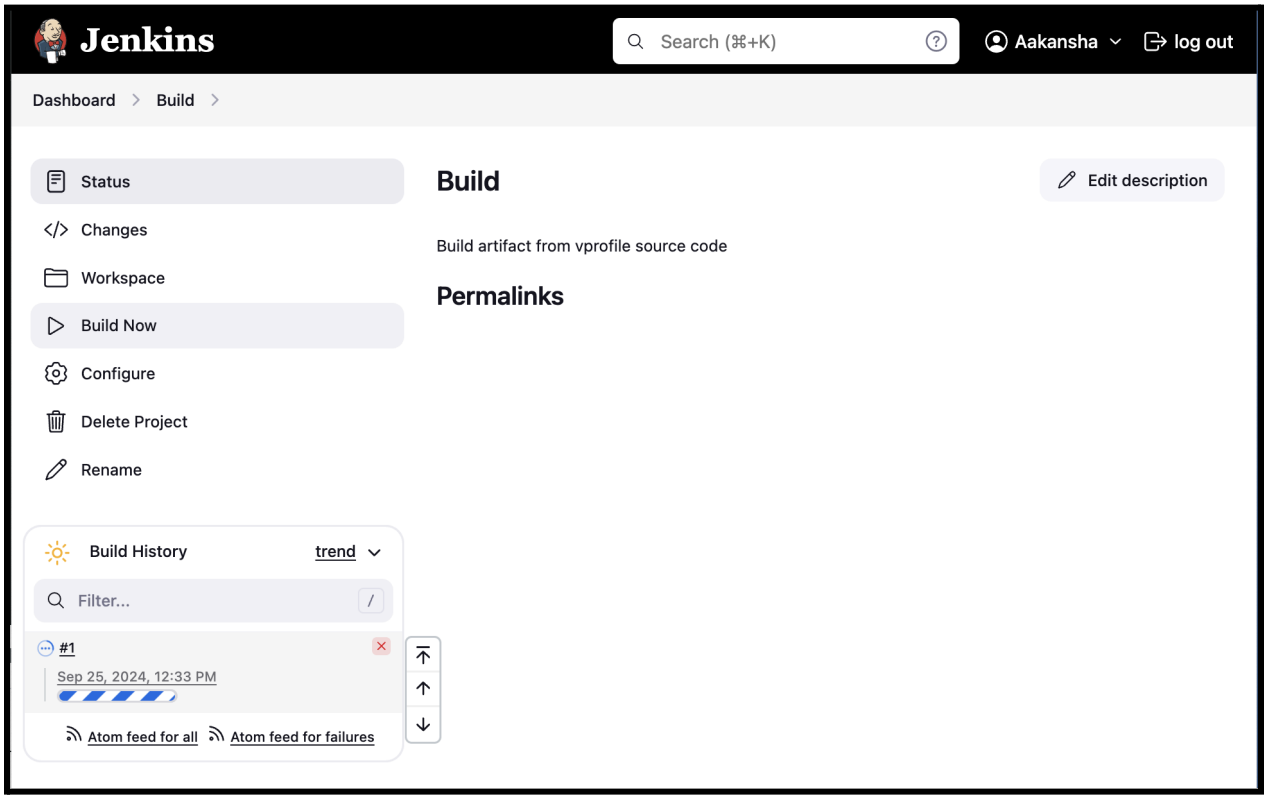

Dashboard
Build
Configuration
Configure
General
Source Code Management
Build Triggers
Build Environment
Build Steps
Post-build Actions
Source Code Management
None
Git
Repositories
Repository URL
https://github.com/devopshydclub/vprofile-project.git
Credentials
- none -
+ Add
Advanced
Add Repository
Branches to build
Branch Specifier (blank for 'any')

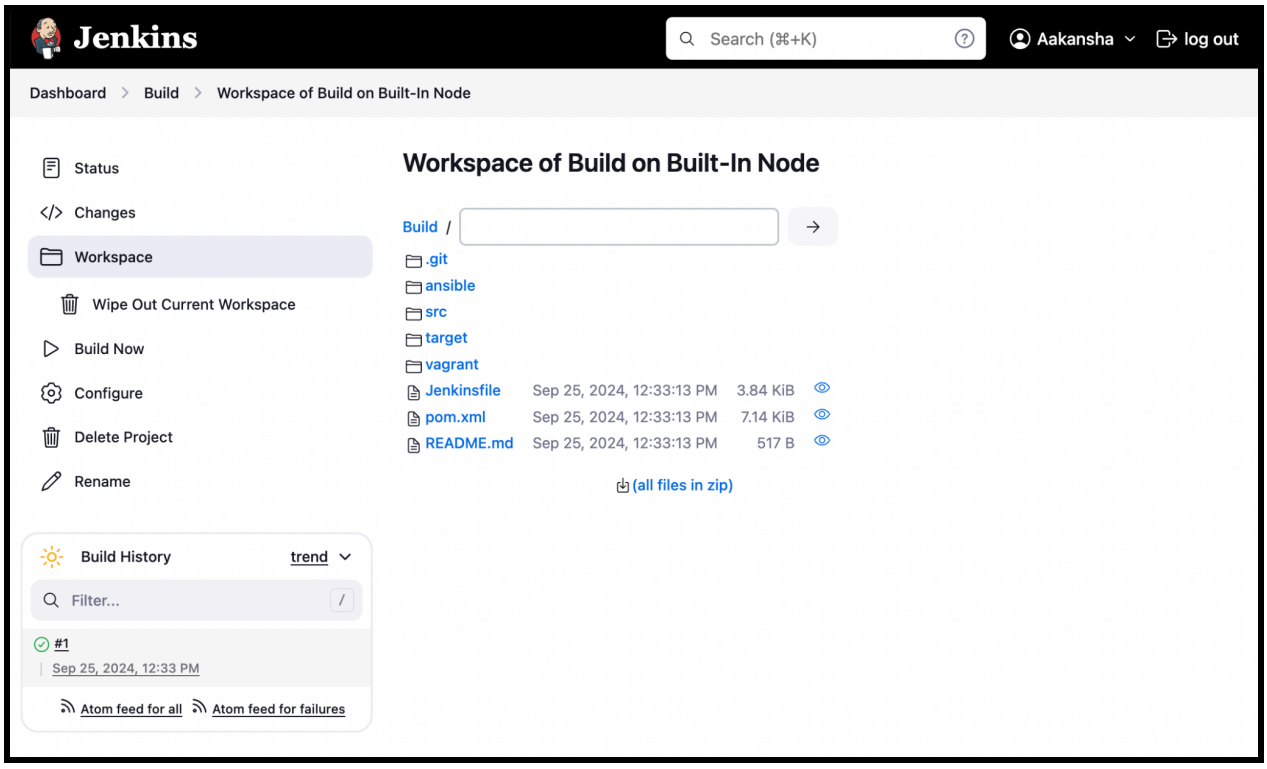

Dashboard
Build
Configuration
Configure
General
Source Code Management
Build Triggers
Build Environment
Build Steps
Post-build Actions
Branches to build
Branch Specifier (blank for 'any')
*/main
Add Branch
Repository browser
(Auto)
Additional Behaviours
Add

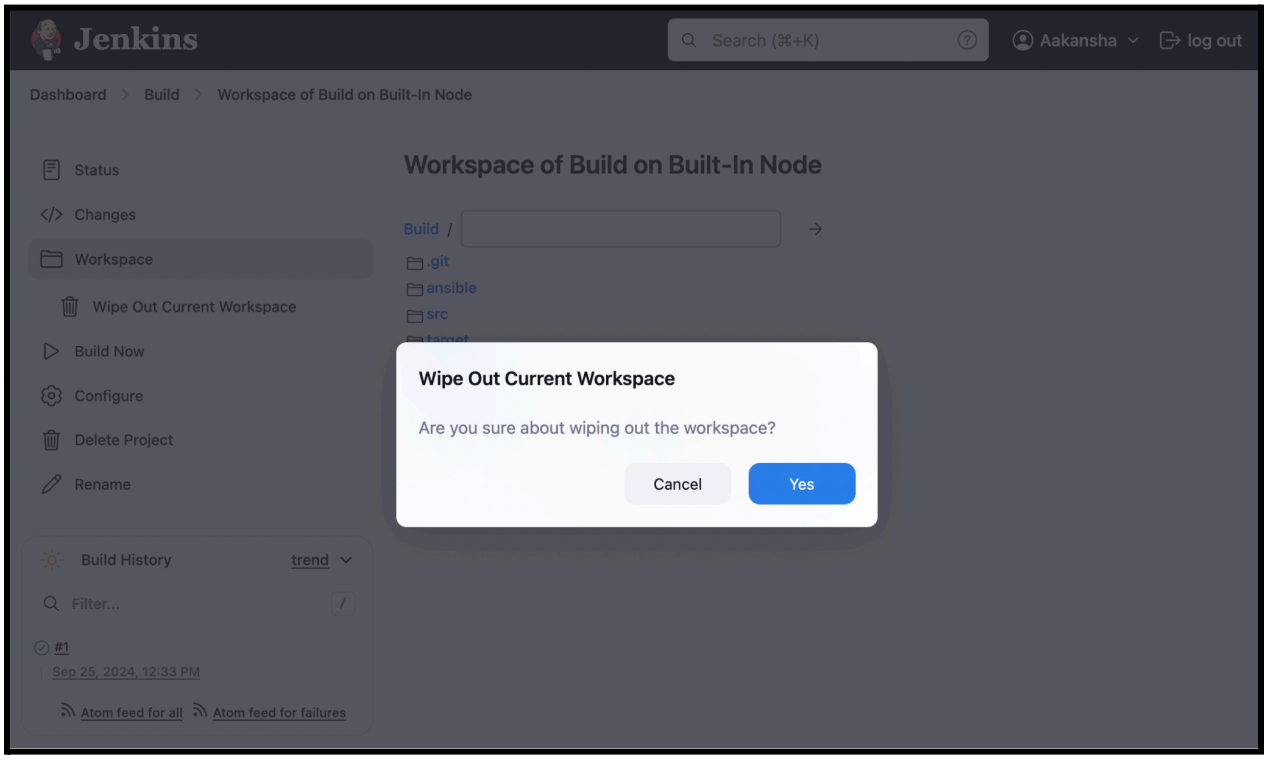

Dashboard > Build > Configuration
Configure
General
Source Code Management
Build Triggers
Build Environment
Build Steps
Post-build Actions
Build Steps
Invoke top-level Maven targets
Maven Version
MAVEN3
Goals
clean install
Advanced
Add build step
Post-build Actions
Save
Apply


Jenkins
Search (⌘+K)
Aakansha
log out
Dashboard > Build > #1 > Console Output
Status
Changes
Console Output
Edit Build Information
Delete build '#1'
Timings
Git Build Data
Console Output
Download
Copy
View as plain text
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:

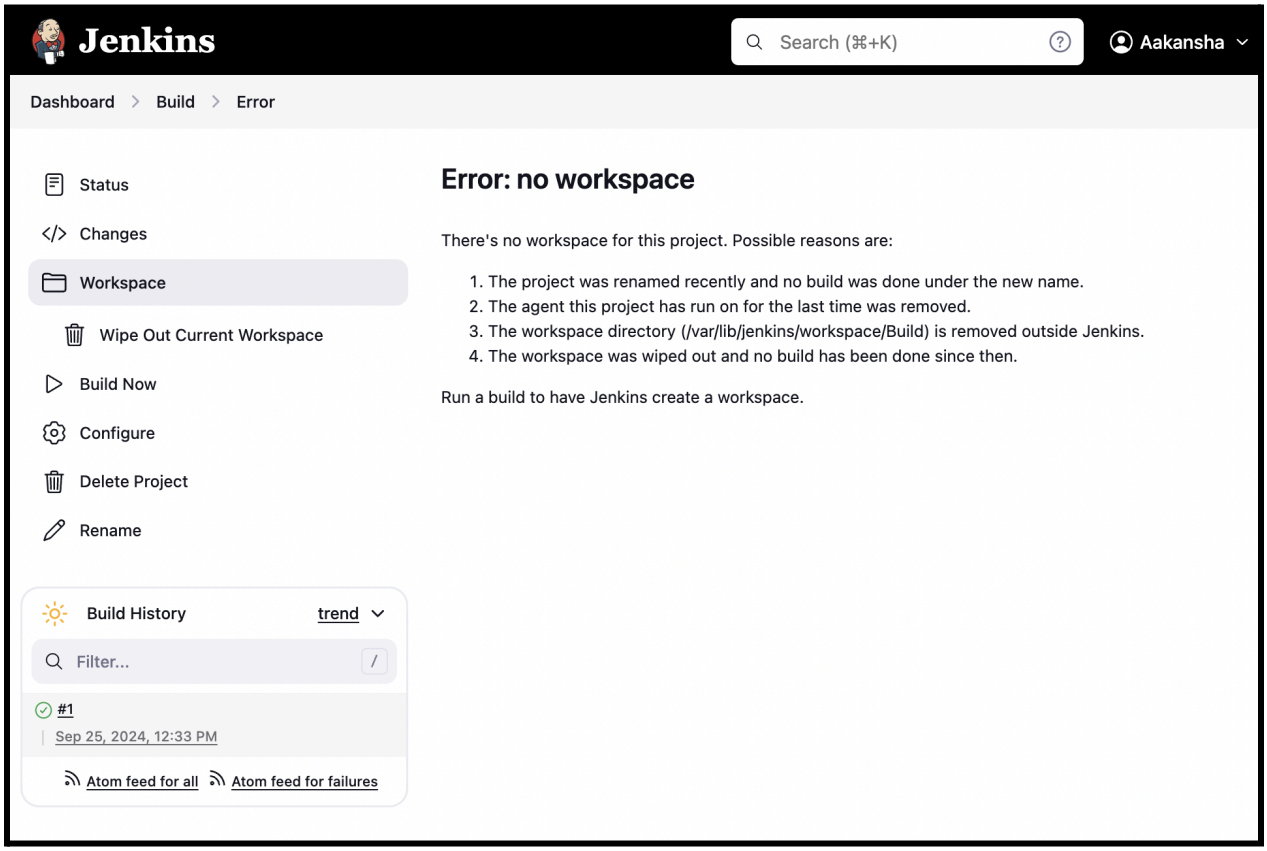

Jenkins
Search (⌘+K)
Aakansha
log out
Dashboard
New Item
Build History
Manage Jenkins
My Views
Add description
All
S
W
Name
Last Success
Last Failure
Last Duration
Build
2 min 34 sec
#1
N/A
27 sec
Build Queue
No builds in the queue.
Build Executor Status
1 Idle
2 Idle
Icon:
S
M
L


Jenkins
Search (⌘+K)
Aakansha
log out
Dashboard
Build
Status
Changes
Workspace
Build Now
Configure
Delete Project
Rename
Build
Edit description
Build artifact from vprofile source code
Permalinks
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
Status
Changes
Workspace
Wipe Out Current Workspace
Build Now
Configure
Delete Project
Rename
Workspace of Build on Built-In Node
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
(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
Status
Changes
Workspace
Wipe Out Current Workspace
Build Now
Configure
Delete Project
Rename
Workspace of Build on Built-In Node
Build /
.git
ansible
src
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.

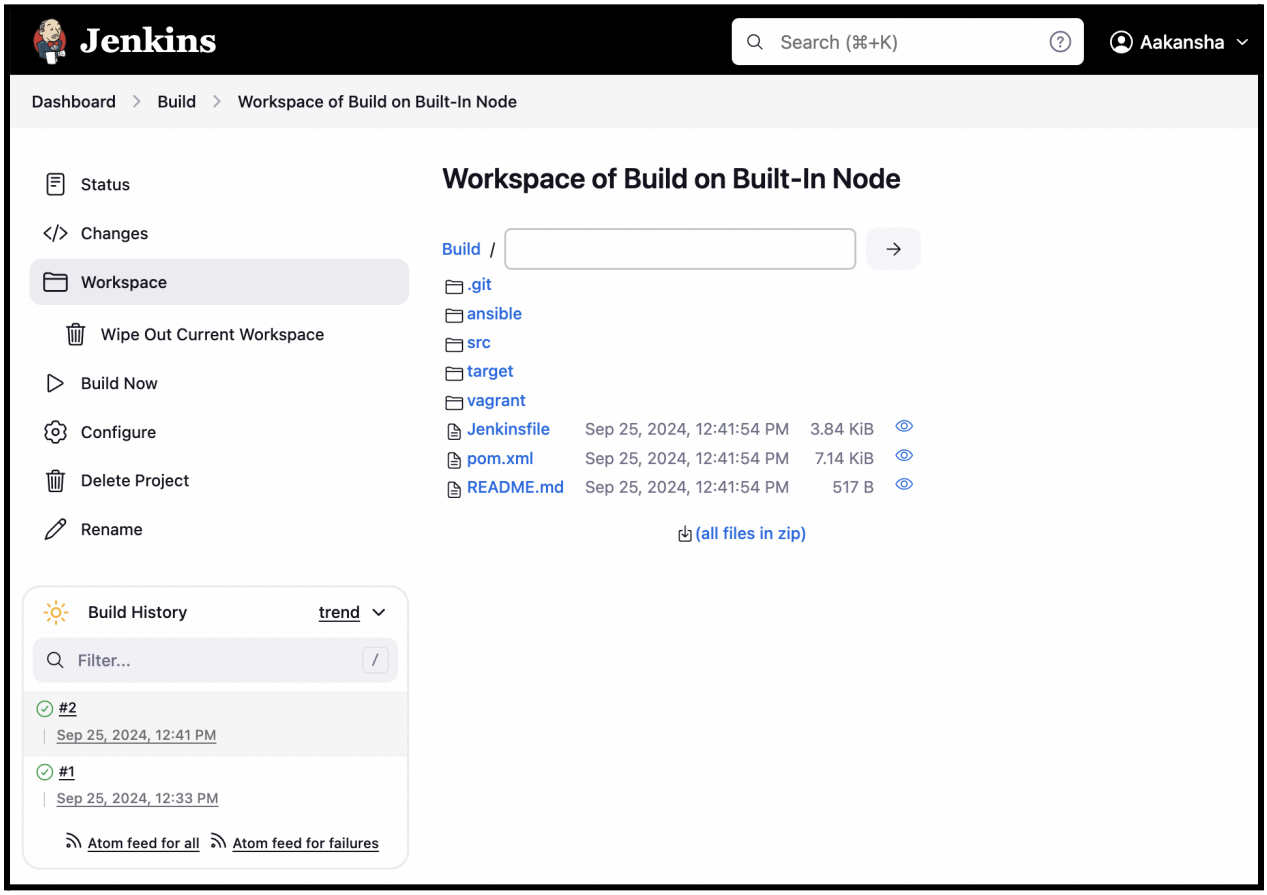


➔ **Now, when we trigger the build, Jenkins will initiate the process by fetching all new data from the configured Git repository. If there are any changes in the source code, Jenkins, with the help of the Git plugin, will automatically pull only the modified portions, ensuring efficient and fast updates. 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.**

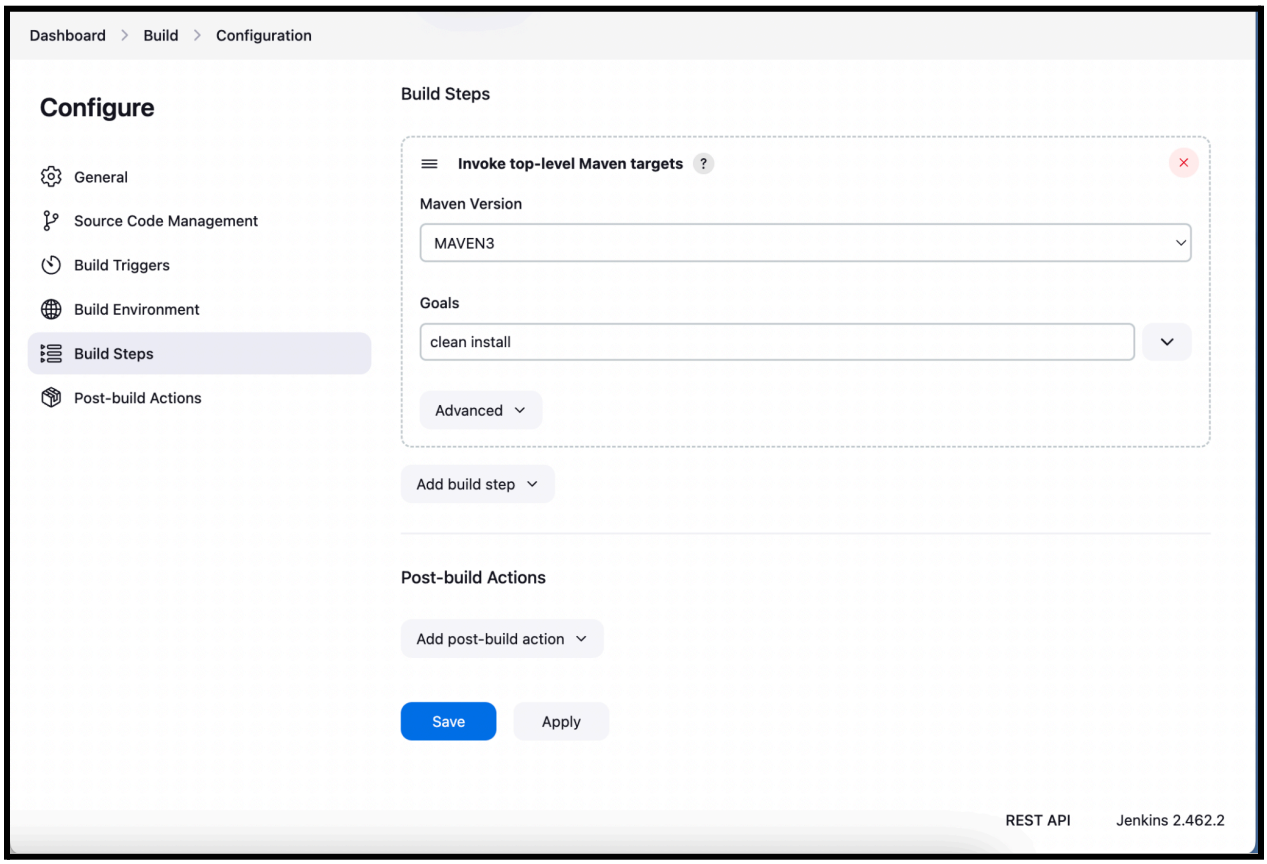
Note: If the workspace has been wiped out (as shown in the error), triggering a new build will automatically recreate the workspace and populate it with the necessary data.

Let’s proceed to build the project again.



Configuring Post-Build Actions

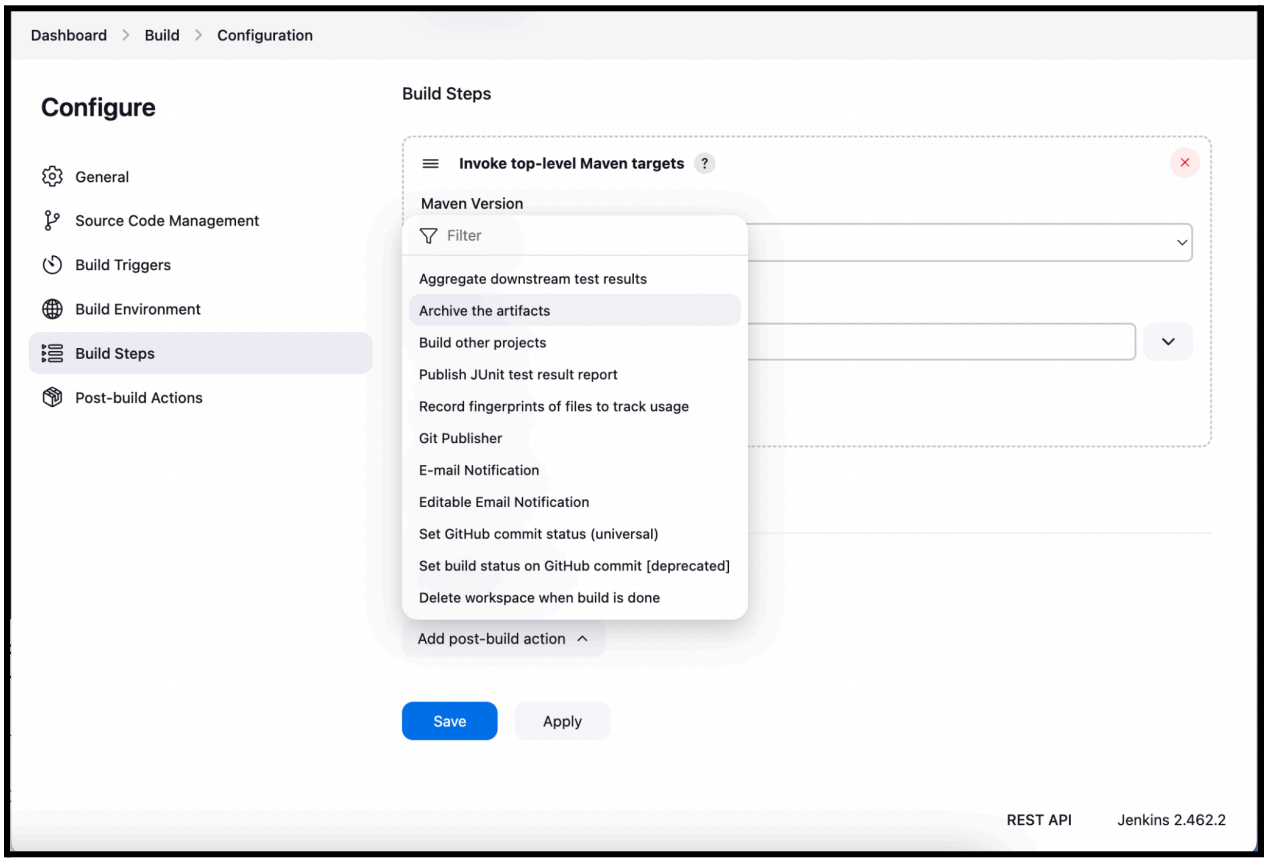
1. **Click on "Configure":**
    - Begin by navigating to your Jenkins job and selecting the Configure option. This allows you to modify the job's settings and behavior.
2. **Navigate to Post-Build Actions:**
    - Scroll down to the Post-Build Actions section. This area defines what Jenkins will do 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:

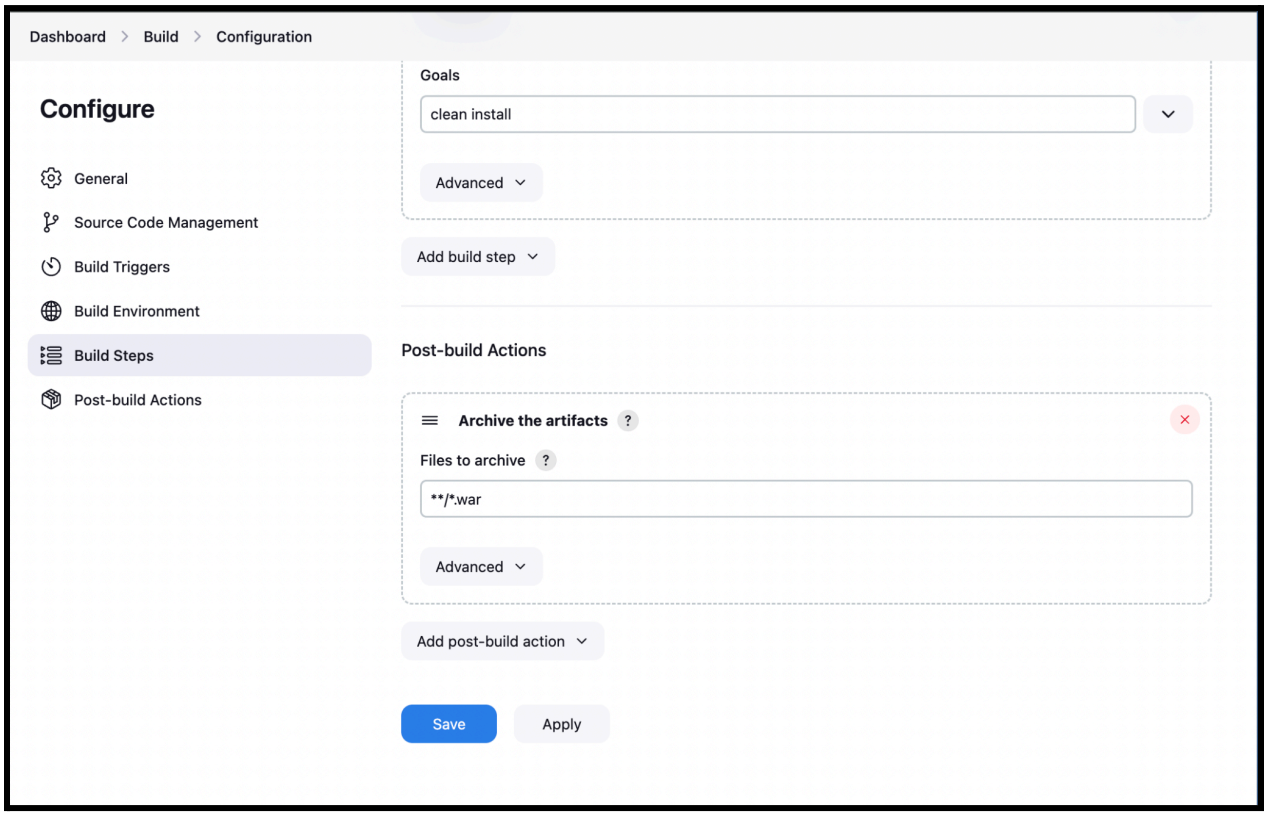
- **Archiving Artifacts:** After the build is complete, you can specify that Jenkins should look for files with a `.war` extension or other relevant formats. These files represent the output of the build (such as packaged applications) and need to be stored for future use or deployment.

- **Recursive Search:** Jenkins can also be configured to look for artifacts recursively in specific directories and subdirectories.



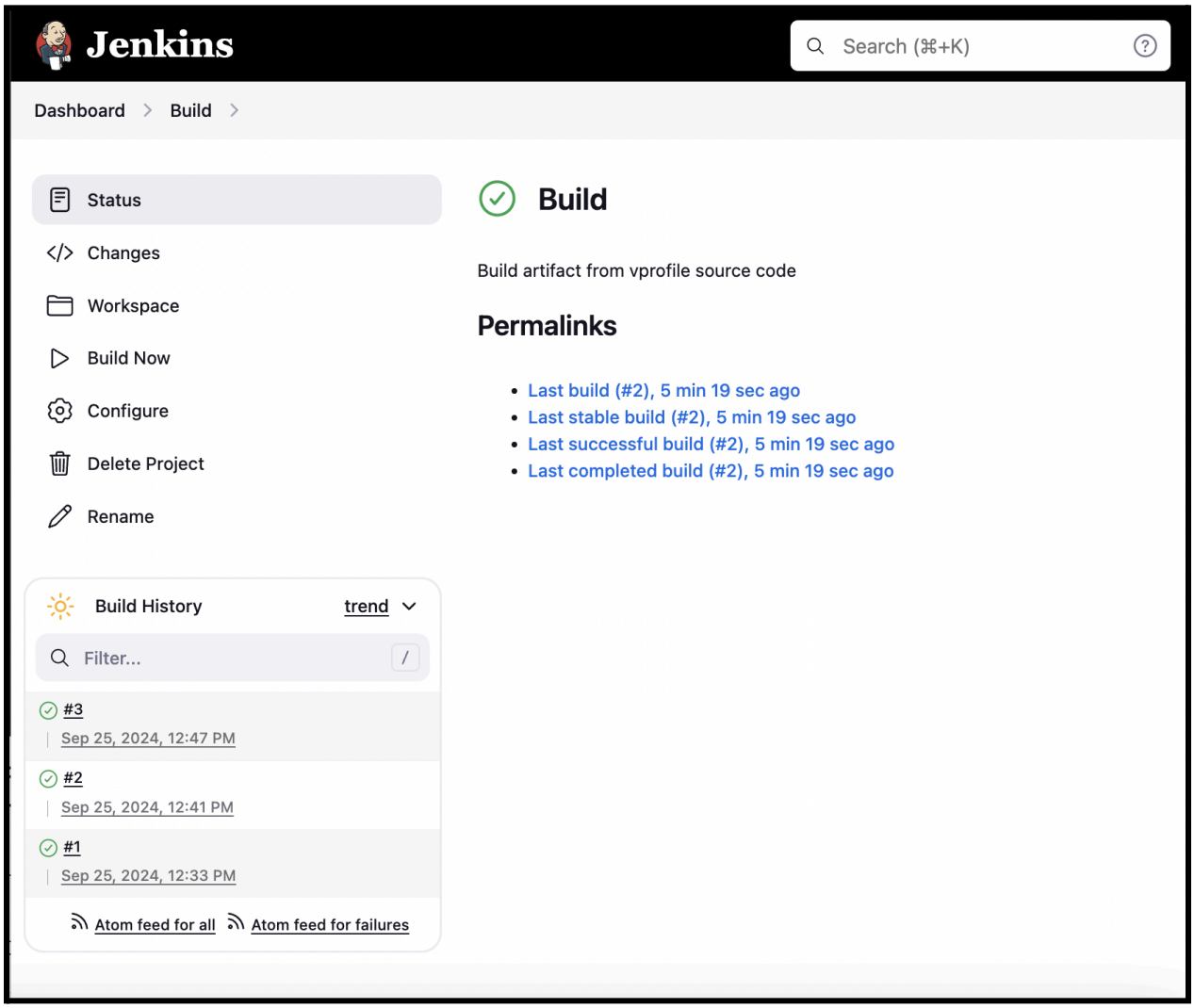
Archive Files with `.war` Extension:

- 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.



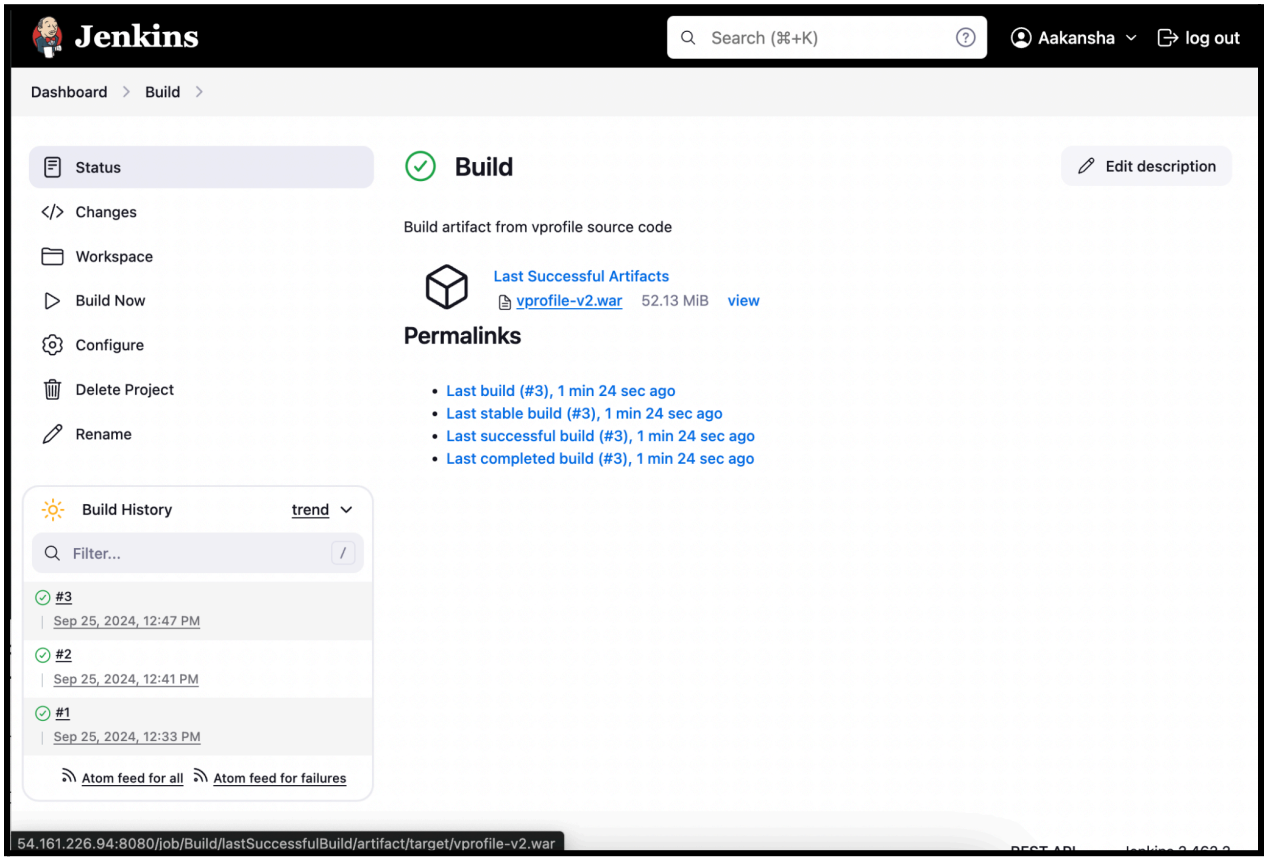
Click on Save.

Start the Build.



Check the Archived Artifact:

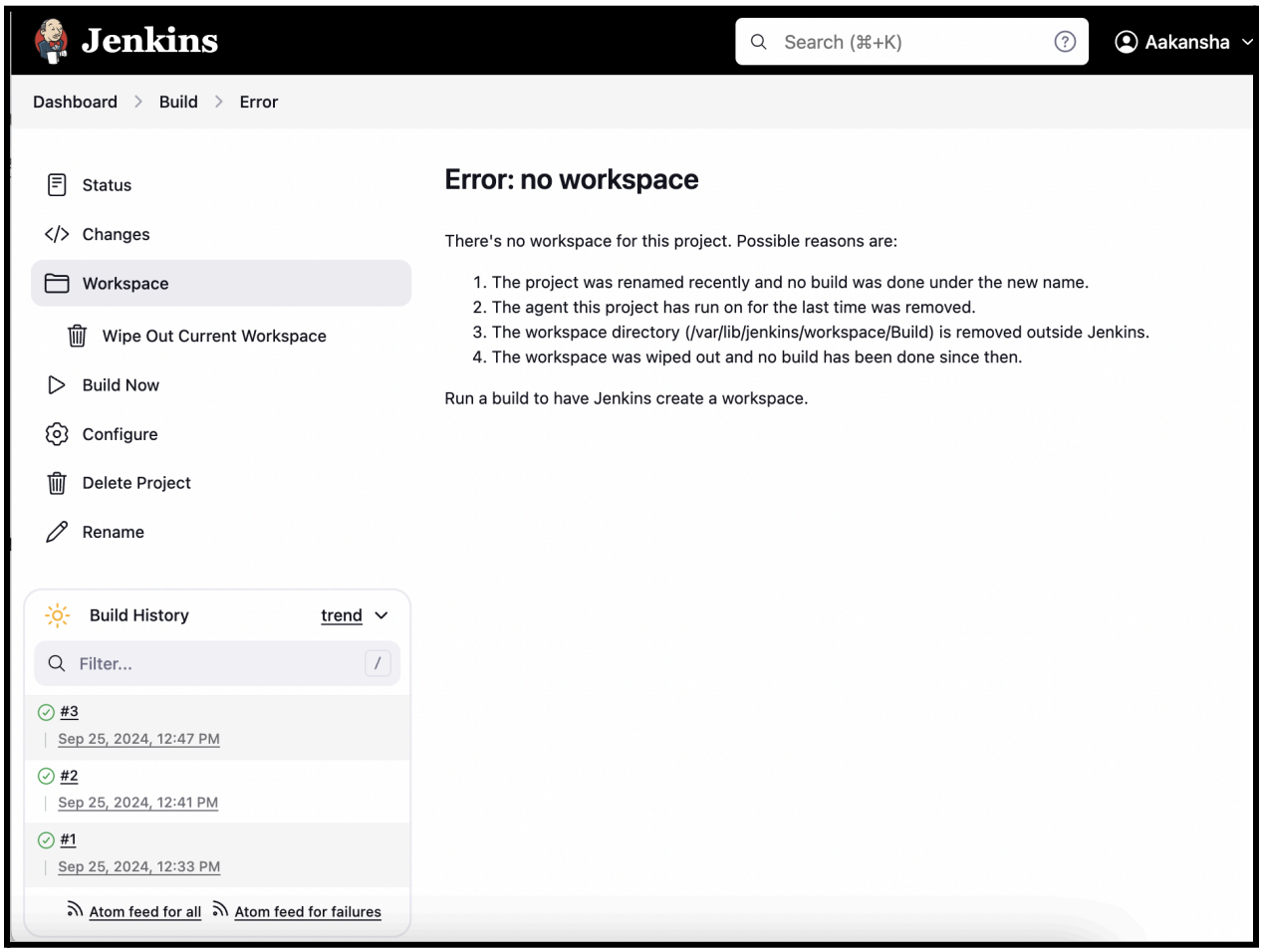
- After the build completes, navigate to the Build History section and click on the most recent build entry. In the build details page, look for the Archived Artifacts section.
- Here, you should see the `.war` file (or other configured file types) that Jenkins has saved from the build. You can download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



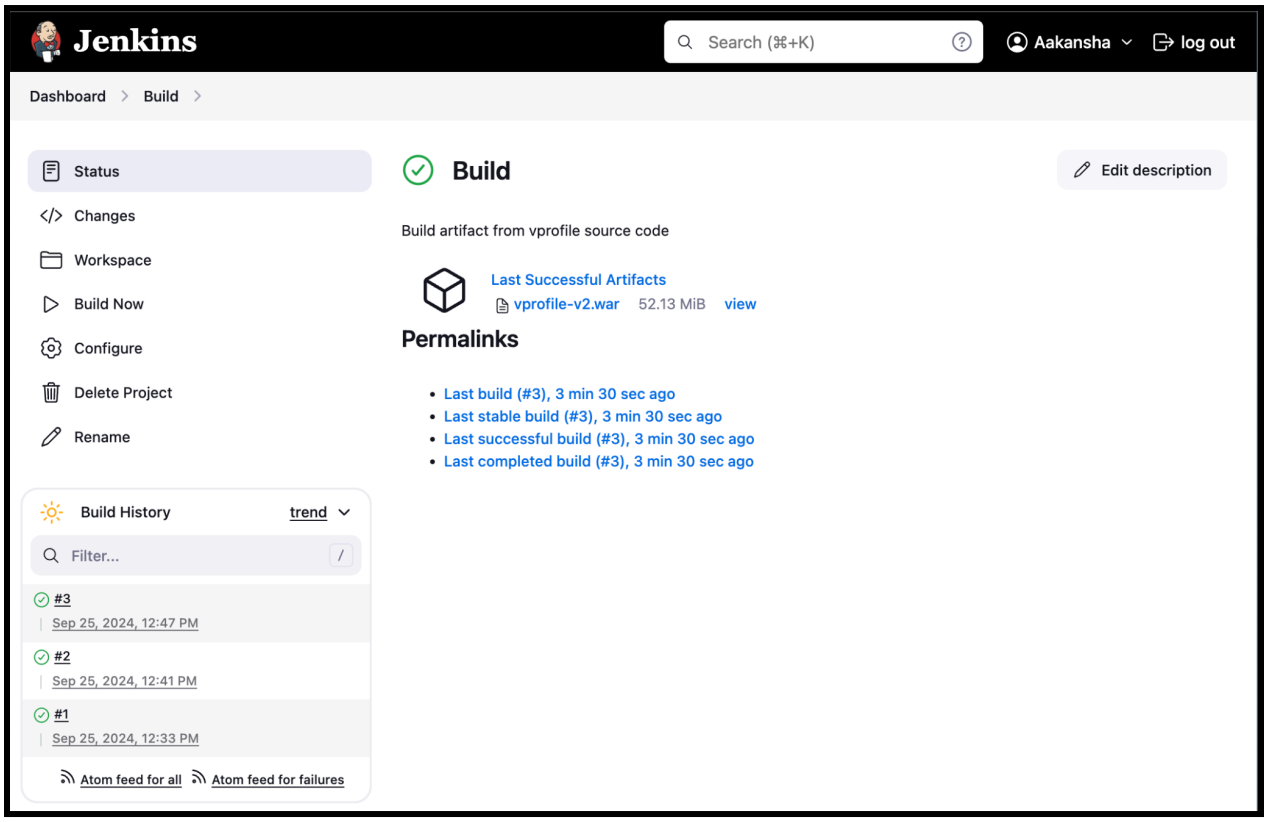
This step confirms that your build artifacts were successfully archived and are accessible, ensuring that critical outputs like your web application archives are stored safely for future use.

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).



Verifying Archived Artifacts After Workspace Cleanup



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.