25-Nov-2024

Internship Day - 86 Report:

Jenkins is an open-source automation server. It helps automate the parts of software development related to building, testing, and deploying, facilitating continuous integration, and continuous delivery. It is a server-based system that runs in servlet containers such as Apache Tomcat, or by default as a stand-alone web-application in co-bundled Eclipse Jetty. It supports version control tools, including CVS, Subversion, Git, Mercurial, Perforce, ClearCase, and RTC, and can execute Apache Ant, Apache Maven, and set based projects as well as arbitrary shell scripts and Windows batch commands.

BUILDS

Builds can be triggered by various means, for example:

- a webhook that gets triggered upon pushed commits in a version control system
- scheduling via a cron-like mechanism
- requesting a specific build URL.
- after the other builds in the queue have completed
- invoked by other builds

PLUGINS

Plugins have been released for Jenkins that extend its use to projects written in languages other than Java. Plugins are available for integrating Jenkins with most version control systems and bug databases. Many build tools are supported via their respective plugins. Plugins can also

change the way Jenkins looks or add new functionality. There are a set of plugins dedicated for the purpose of unit testing that generate test reports in various formats (for example, JUnit bundled with Jenkins, MSTest, NUnit, etc. and automated testing that supports automated tests. Builds can generate test reports in various formats supported by plugins (JUnit support is currently bundled) and Jenkins can display the reports and generate trends and render them in the GUI.

MAILER

Allows configuring email notifications for build results. Jenkins will send emails to the specified recipients whenever a certain important event occurs, such as:

- Failed build.
- Unstable build.
- Successful build after a failed build, indicating that a crisis is over.
- Unstable build after a successful one, indicating that there's a regression.

CREDENTIALS

Allows storing credentials in Jenkins. Provides a standardized API for other plugins to store and retrieve different types of credentials.

MONITORING EXTERNAL JOBS

Adds the ability to monitor the result of externally executed jobs

SSH AGENTS

This plugin allows managing agents (formerly known as slaves) running on nix machines over SSH. It adds a new type of agent launch method. This launch method will

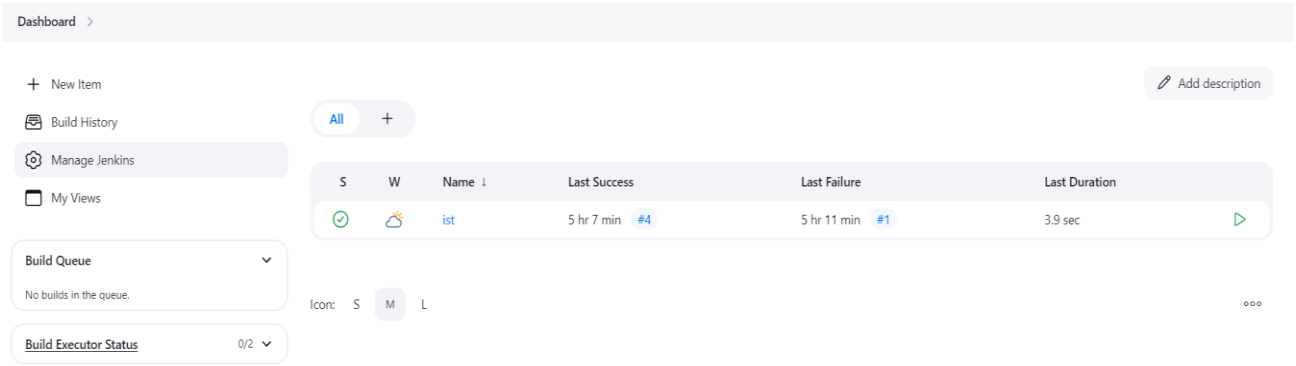
- Open a SSH connection to the specified host as the specified username,
- Check the default version of Java for that user,
- [not implemented yet] If the default version is not compatible with Jenkins's agent.jar, try to find a proper version of Java
- Once it has a suitable version of Java, copy the latest agent.jar via SFTP (falling back to scp if SFTP is not available),

Start the agent process

26-Nov-2024

Internship Day - 87 Report:

Click on dashboard & Click Manage Jenkins



1. Menu (Left Side):

- New Item: Allows users to create a new Jenkins project, which could be a Freestyle project, Pipeline, etc.
- Build History: Displays a log of past builds. Users can click this to view the results and status of previous jobs.
- Manage Jenkins: This navigates to the Jenkins management and configuration settings (like in the previous image).
- My Views: Users can create custom views for managing different jobs or seeing data tailored to their preferences.

2. Build Queue:

No builds in the queue: Indicates that no jobs are currently in the queue for execution. If jobs were queued for execution, they would appear here.

3. Build Executor Status:

Shows the number of available or busy executors (agents that run the build jobs). In this case, it shows 0/2, indicating that both executors are idle and no jobs are running.

4. Main Job List:

This section provides details about Jenkins jobs and their build history.

- All (+ icon): Allows you to create new jobs or filter the display to show specific jobs.
- Columns:
- S (Success Indicator): The green checkmark indicates that the last build was successful.
- W (Warning Indicator): This column may display a warning symbol if the build encountered issues but was not a total failure.
- Name: The name of the Jenkins job. In this case, the job name is "ist."
- Last Success: Indicates the time since the job was last successfully completed. Here, it was 5 hours and 7 minutes ago, and the build number (#4) is provided for reference.
- Last Failure: Indicates the time since the job last failed. In this case, the job failed 5 hours and 11 minutes ago, with build number (#1).
- Last Duration: Displays the amount of time it took for the last build to complete. In this case, the last successful build ran for 3.9 seconds.
- Play Icon: A button that allows the user to manually trigger the job to run again.

5. Icon Size:

S, M, L: Options to change the icon size for better visibility on the Jenkins dashboard (small, medium, or large).

6. Add Description (Top Right):

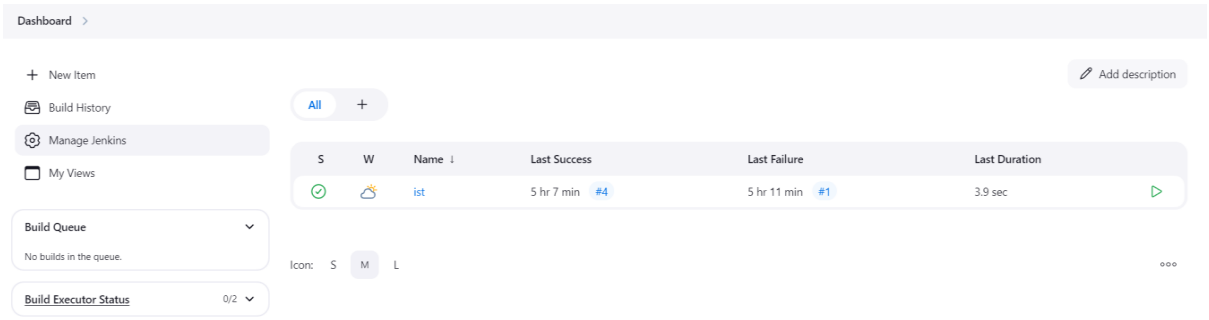
Allows users to add a description for the current job or dashboard view, which can be helpful for tracking purpose or adding notes.

This dashboard provides a concise overview of your Jenkins jobs, their recent build history, and their current status.

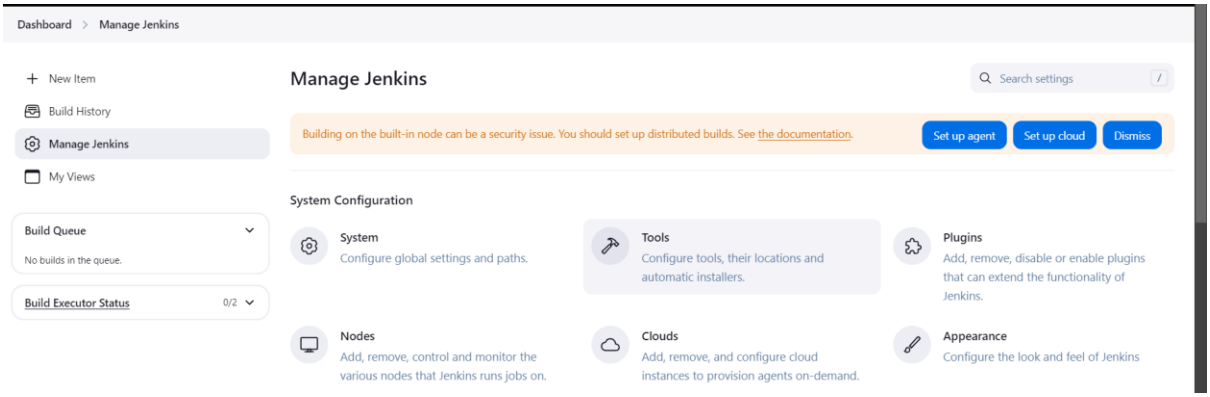
27-Nov-2024

Internship Day - 88 Report:

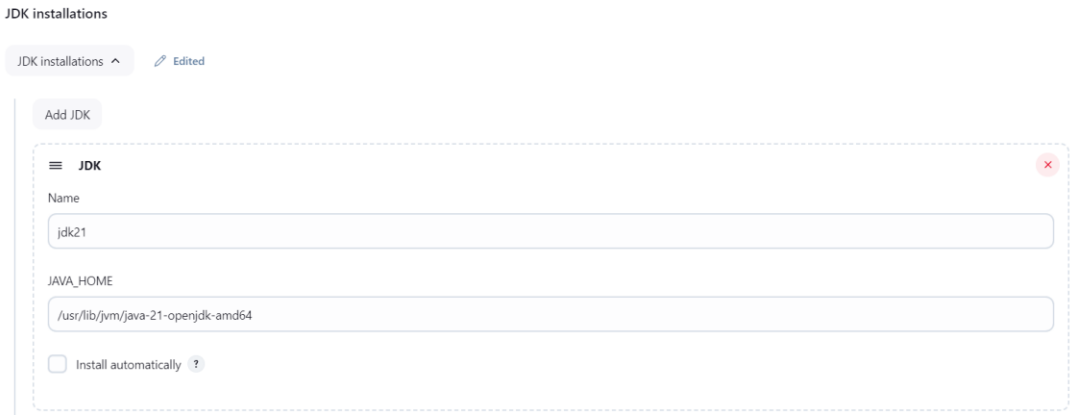
1. Click on dashboard & Click Manage Jenkins



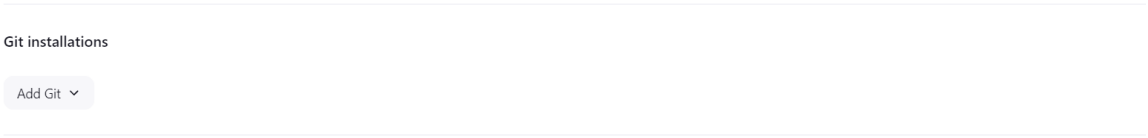
2. Click on Tools



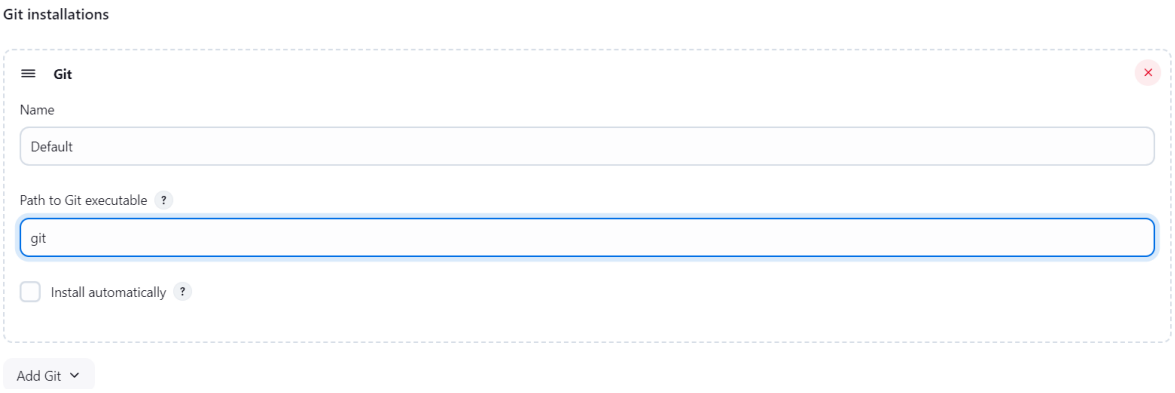
3. click on JDK installations & Click Add jdk : Enter Name & Location of jdk



4. **Click on git installations & Click on Add Git**

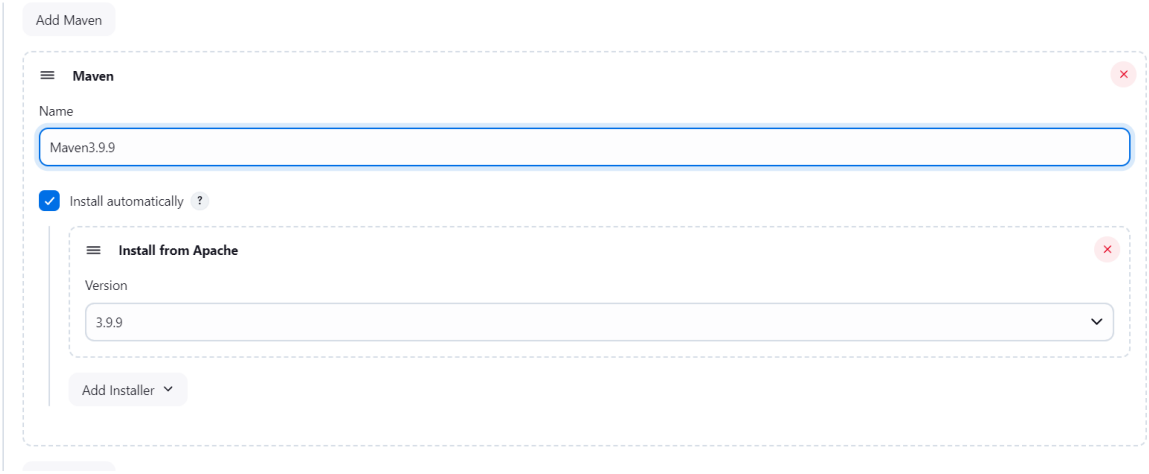


5. **Then Fill Details:**

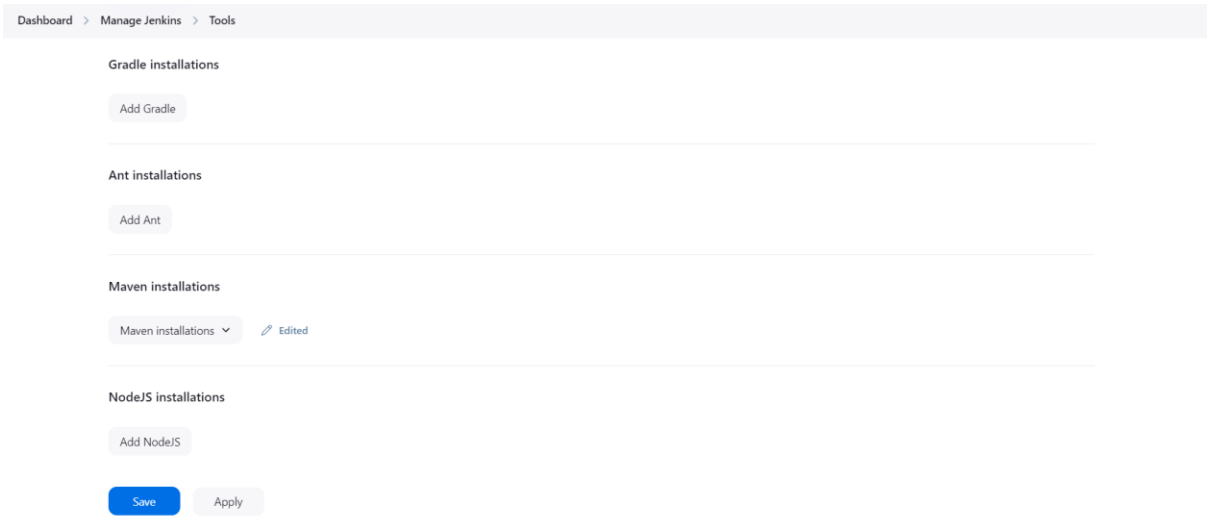


6. **Click on maven installations & Click Add Maven**

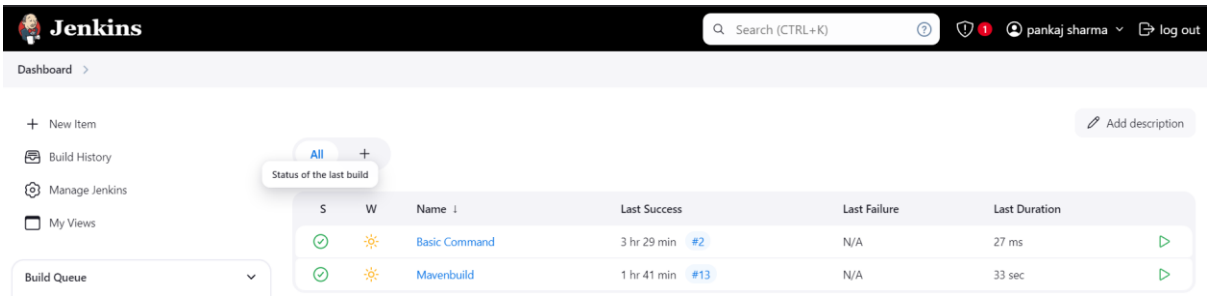
7. **Enter Name and version**



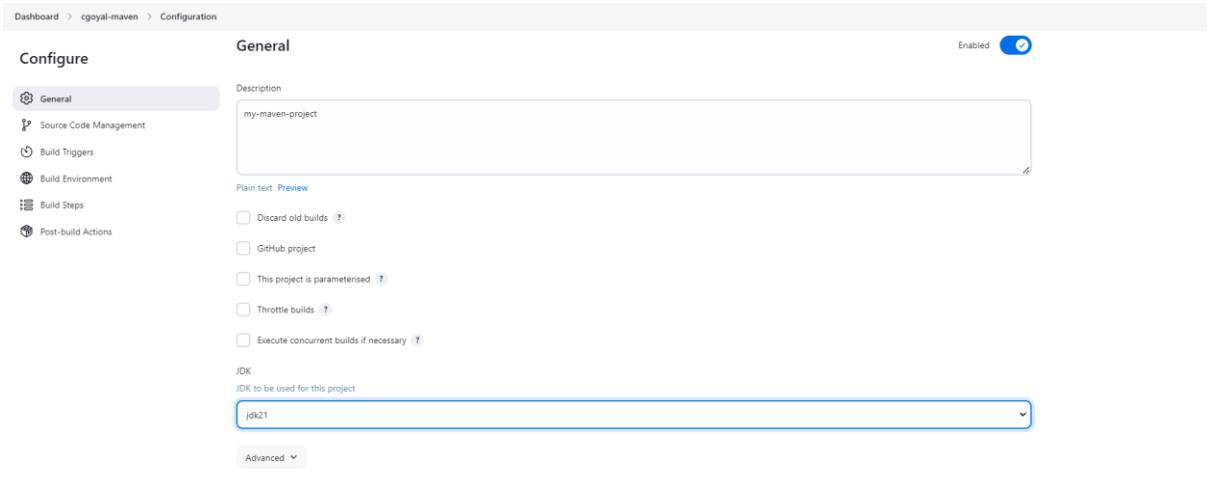
8. **After that click on apply & then save**



9. Go to dashboard & Click on New Item:

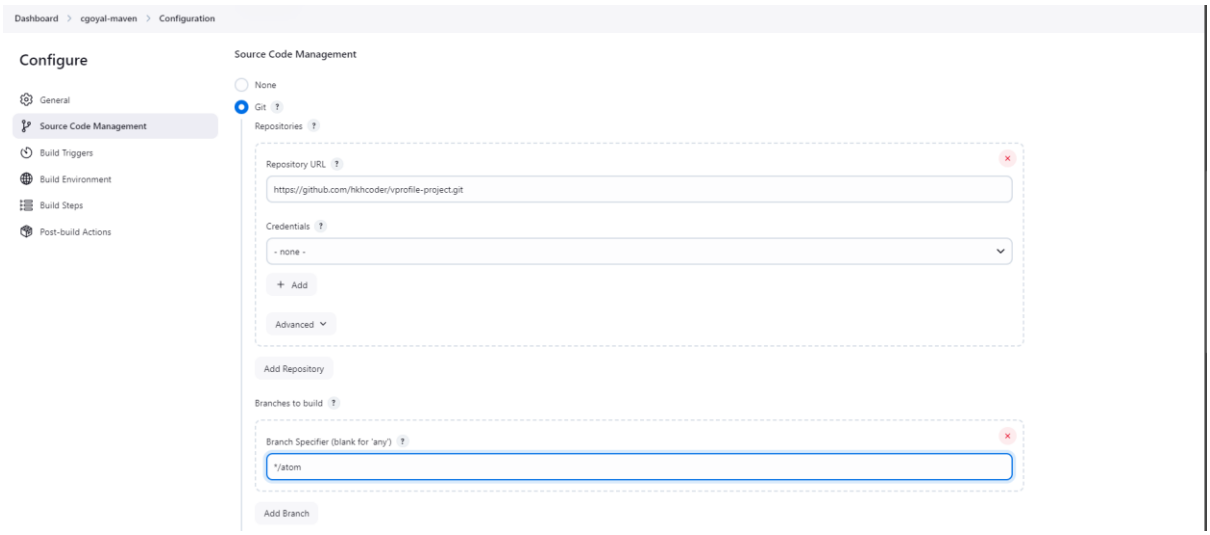


10. **Enter New Item name and select freestyle Project & Click Ok**

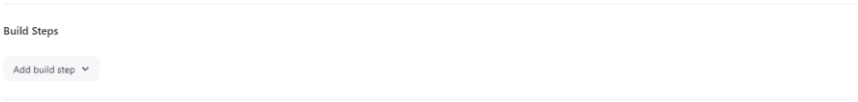


11. Enter Description & then Select JDK

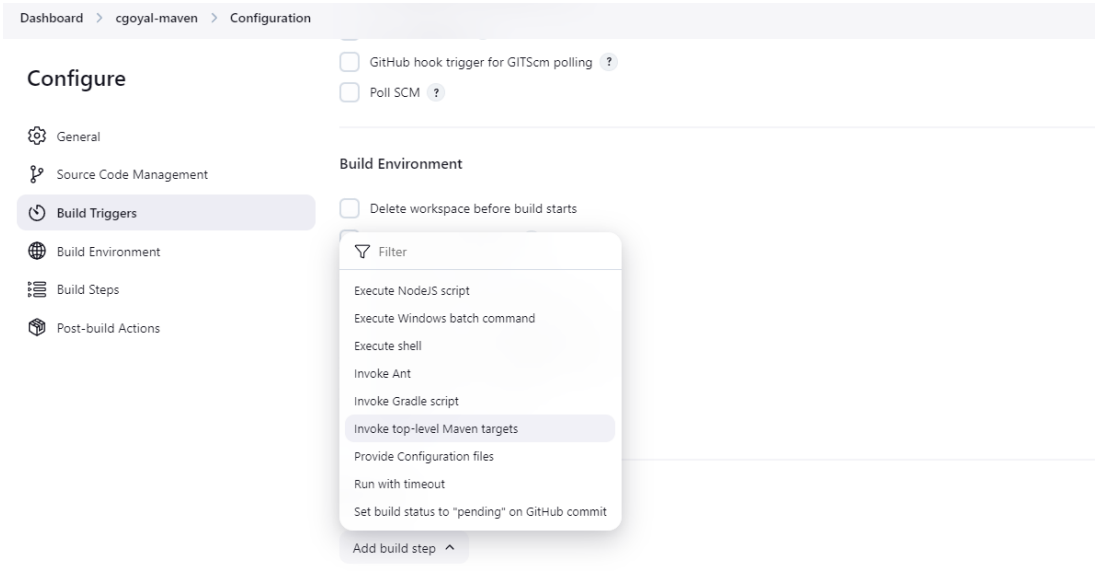
12. **Click on source code management & select git then Enter Git repository URL and Enter Branch name**



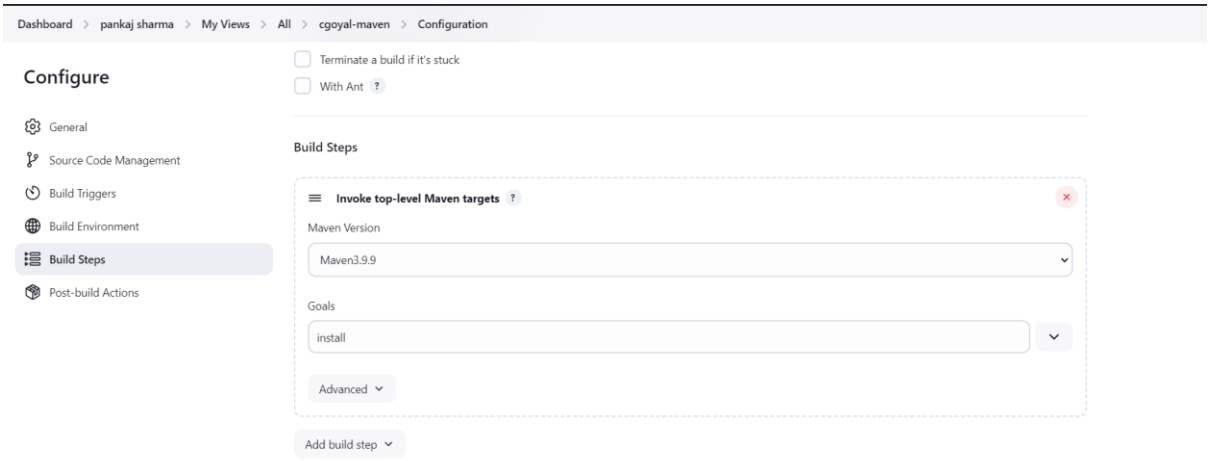
13. **Click on Add Build Setups**



14. Select Invoke top-level Maven Targets



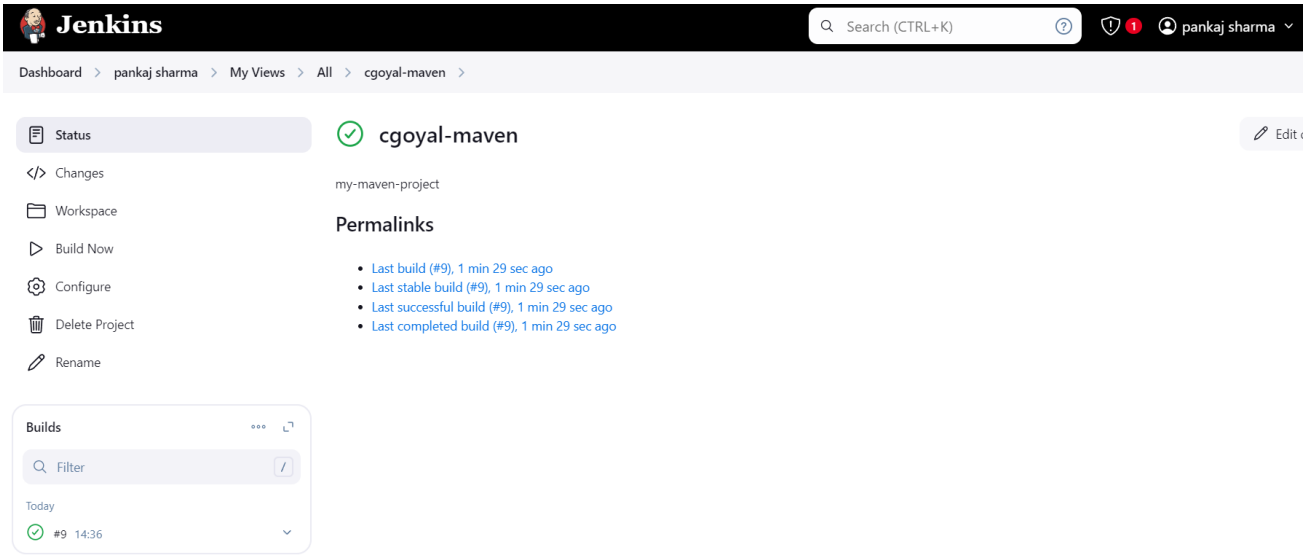
15. Select Maven Version & In Goals write (Install) and Click to Save



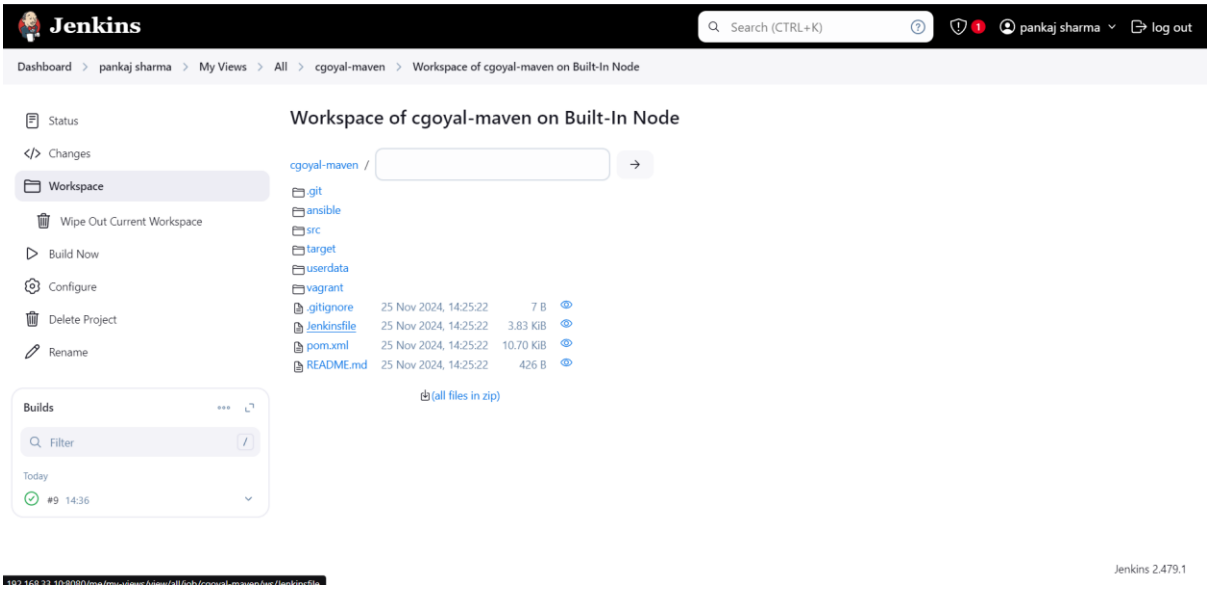
16. Click Save and Apply

17. Click to Build Now Option

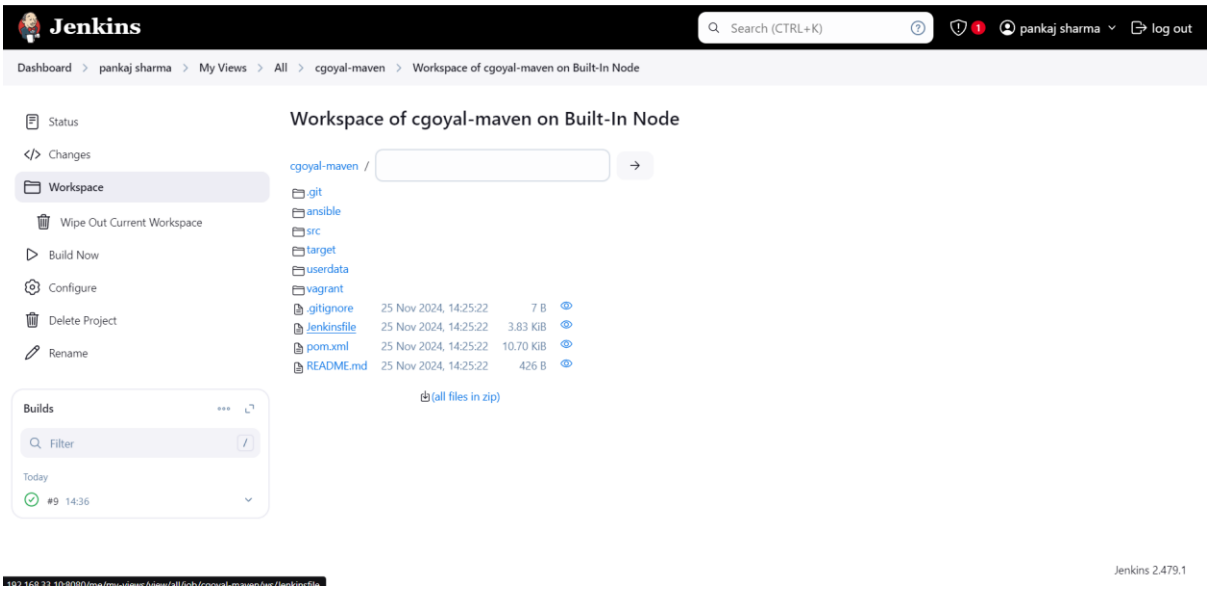
18. **Now, Your Build now is done**



19. Go to Workspace and see your all project files



20. Click on target and see your .war files

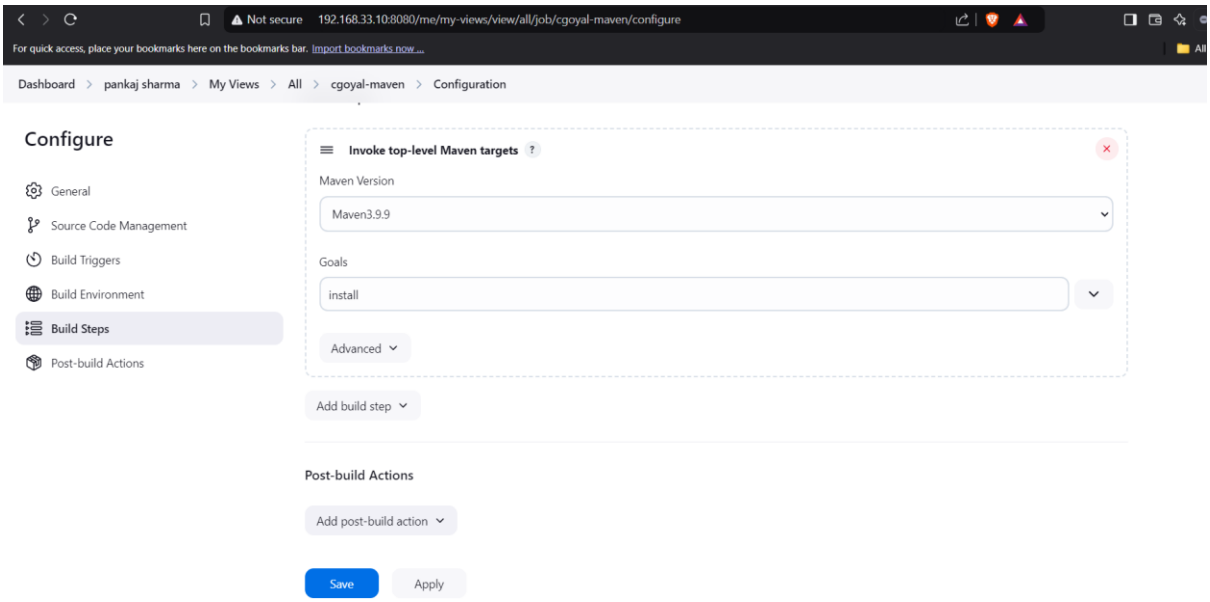


28-Nov-2024

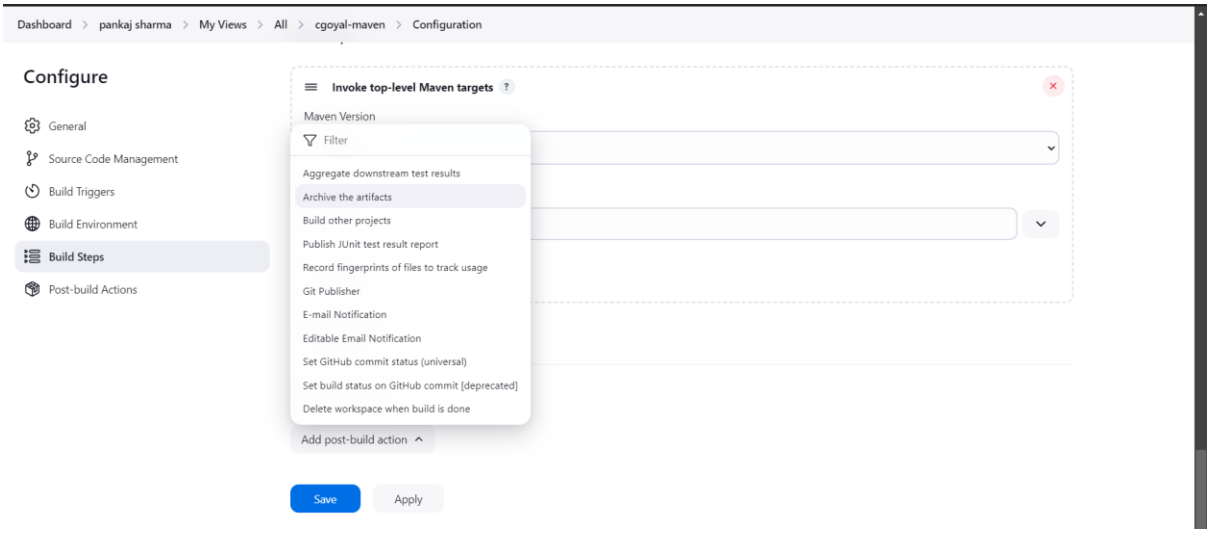
Internship Day - 89 Report:

Now if you want to see it your war files in front then follow these steps:

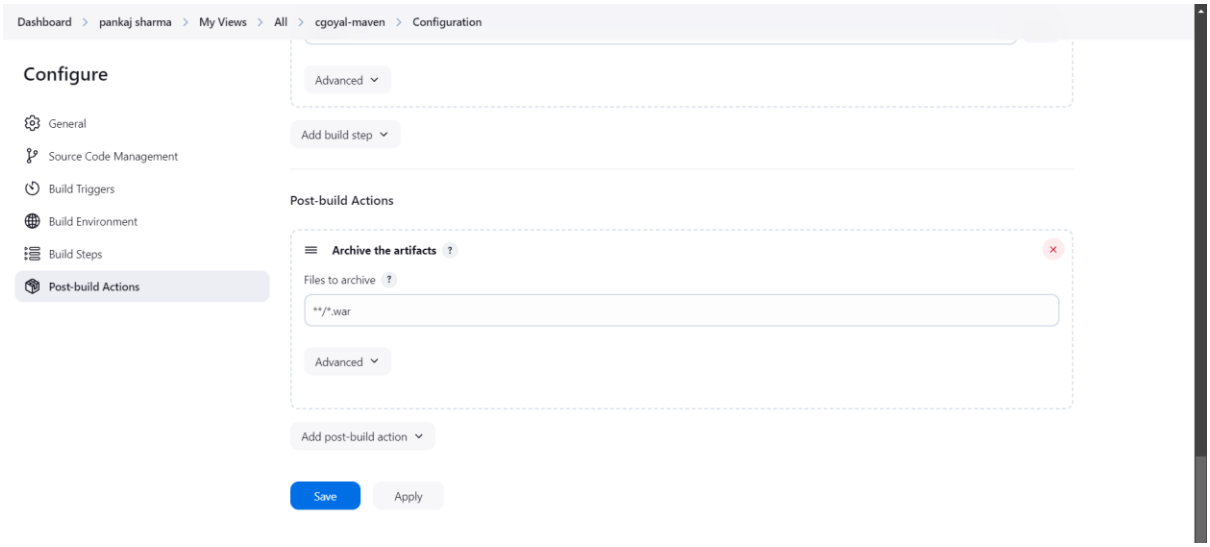
1. Click on Configure and Click Add post-build action



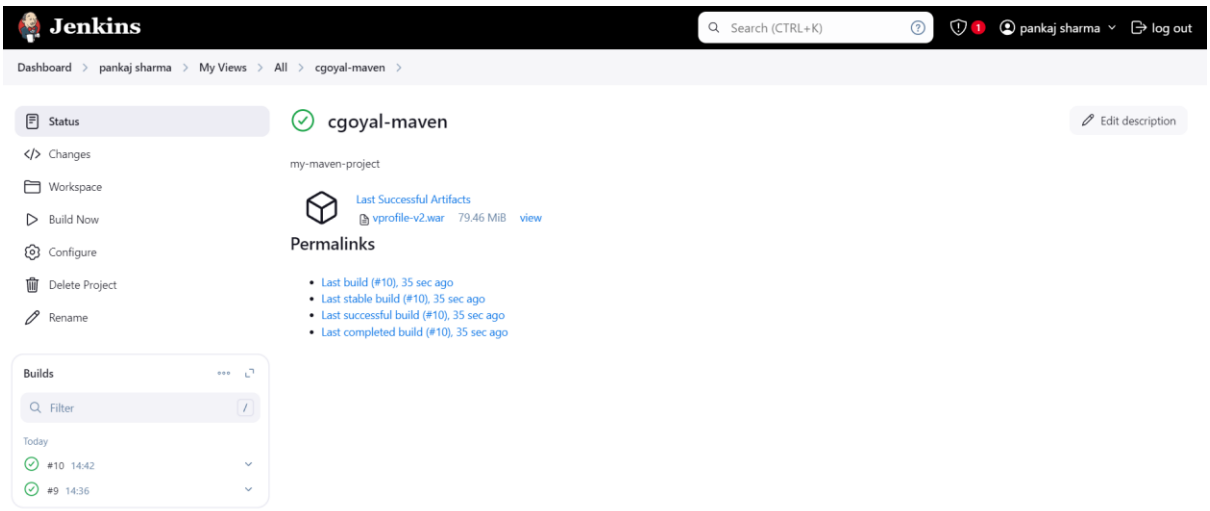
2. Click on Archive the aircrafts



3. Type **/*.war then save & Apply



4. Now you see your file in front



29-Nov-2024

Internship Day - 90 Report:

Version Control System (VCS)

In Jenkins, a **Version Control System (VCS)** is used to manage the source code and configuration files for projects. Jenkins integrates with various VCSs like **Git, SVN, Mercurial**, and others to automate builds and deployments. By connecting to a VCS, Jenkins can pull the latest code changes, trigger automated pipelines, run tests, and deploy applications based on the version-controlled files.

Jenkins uses VCS to:

- Fetch code from repositories.
- Track specific branches, commits, or tags.
- Automatically trigger builds on code changes (e.g., via webhooks or polling).

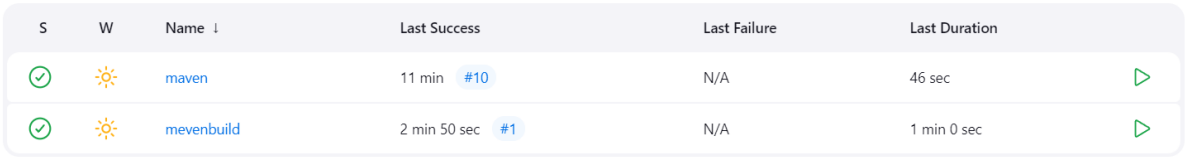
This integration ensures consistency and enables continuous integration and delivery (CI/CD) by automating processes around version-controlled code.

DIFFERENT TYPES OF METHOD OF VERSIONS

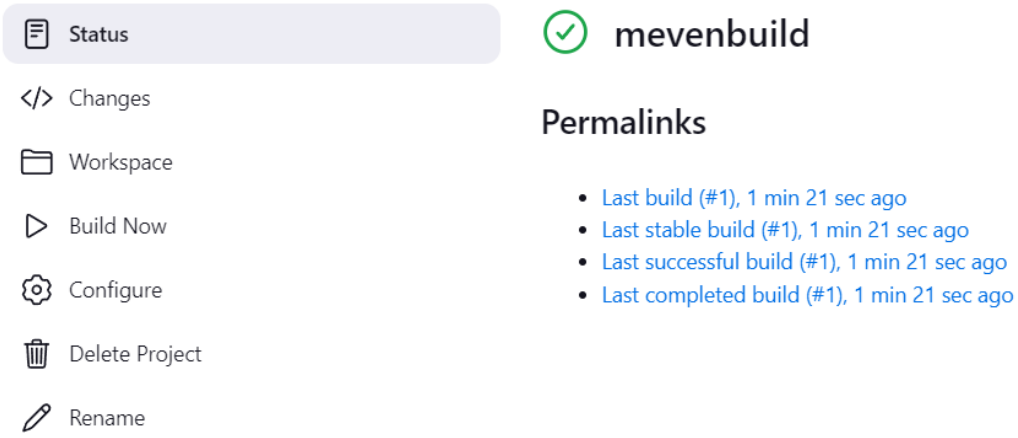
1. MANUAL VERSIONING
2. PARAMETER VERSIONING
3. PLUGIN VERSIONG

1st: Implementation of manual versioning

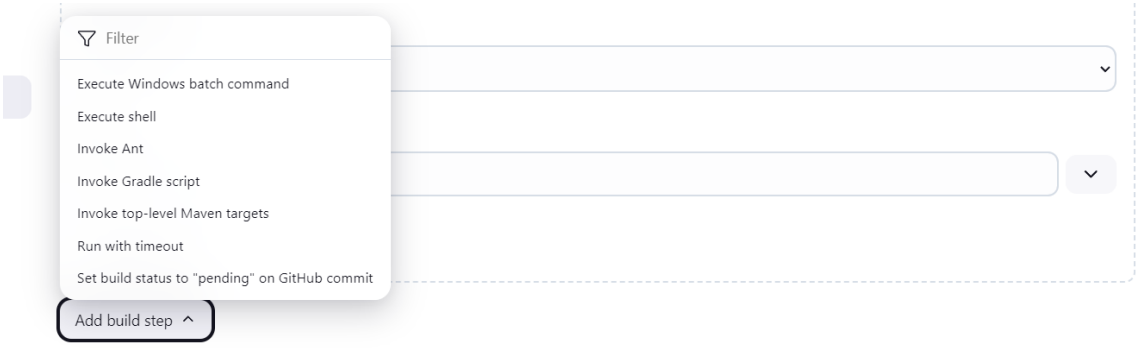
1. Click on dashboard and select job(mavenbuild)



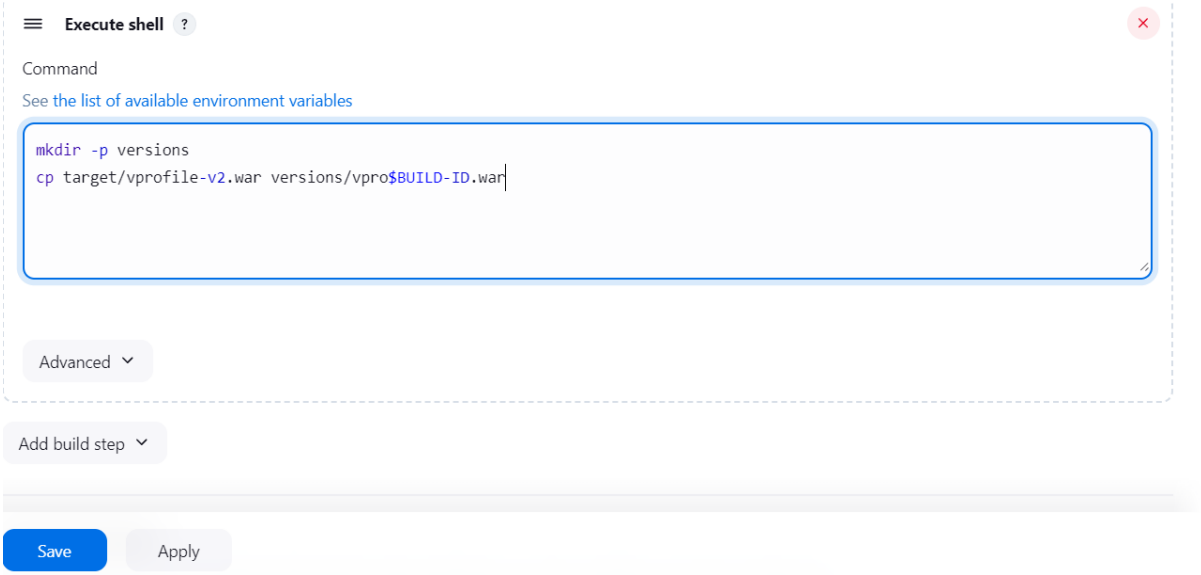
2. Then click on job(mavenbuild) & click on configure



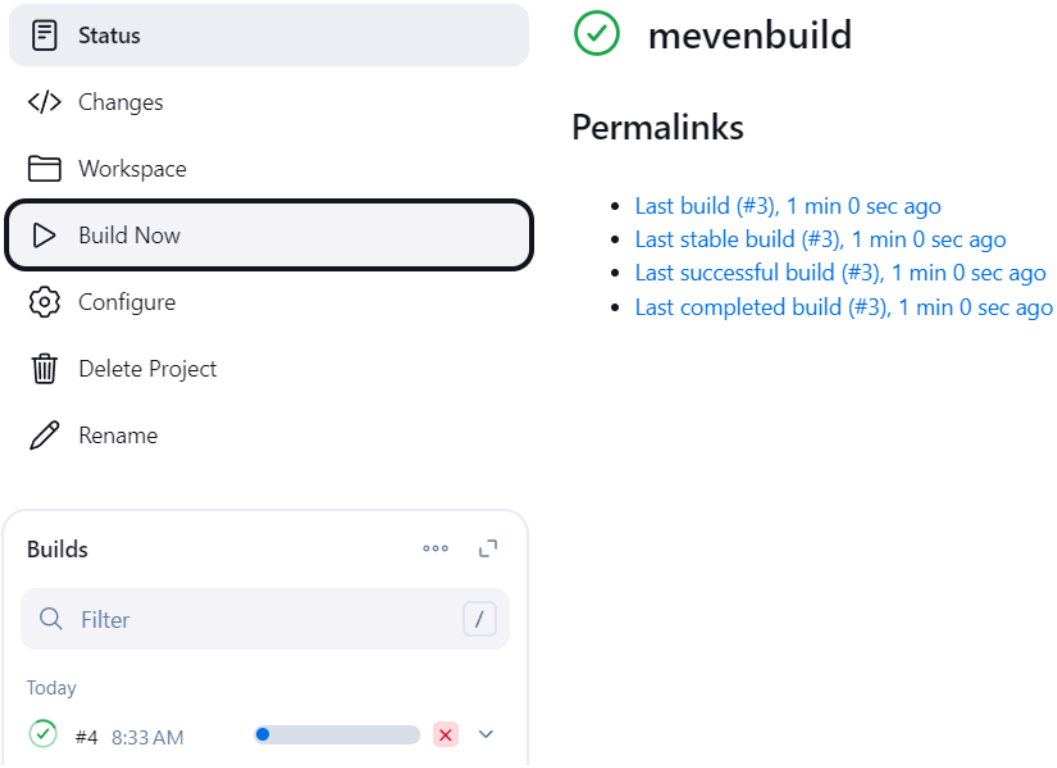
3. After that click on add build & select option “execute shell”



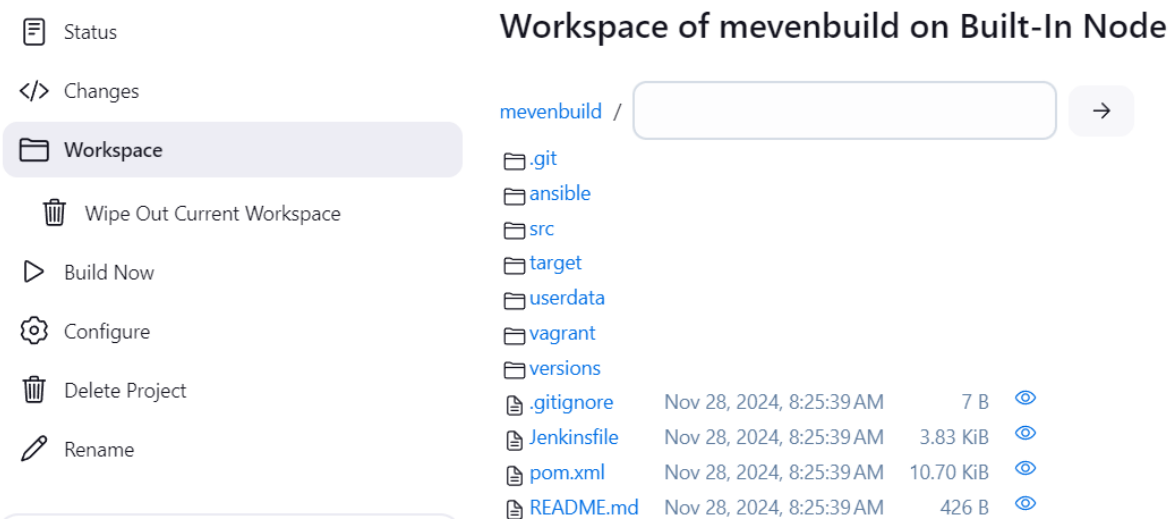
4. Write command in execute shell & save it



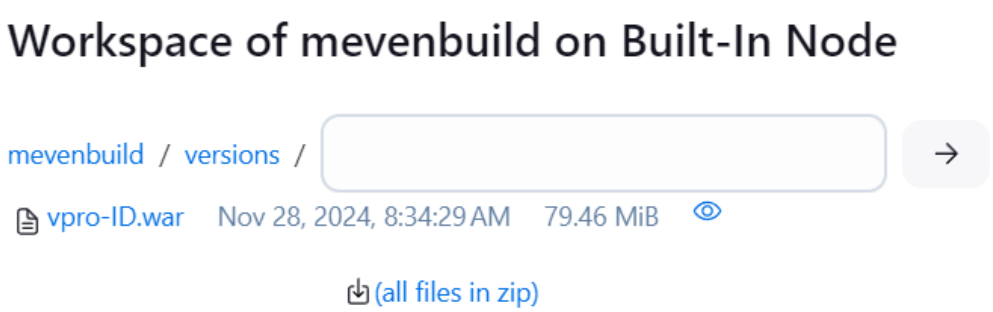
5. Click on build now



6. Go to workspace & you see versions folder

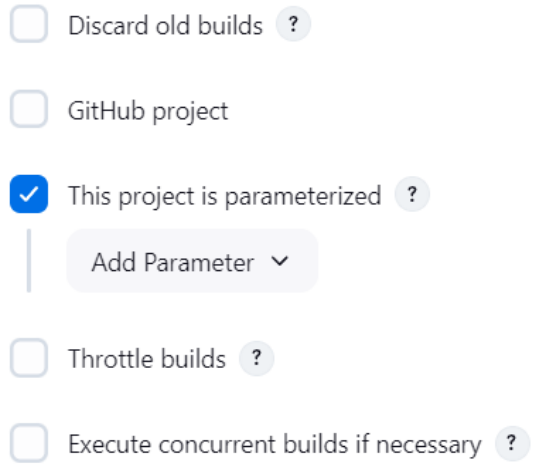


7. Then click on versions & you see your .war file

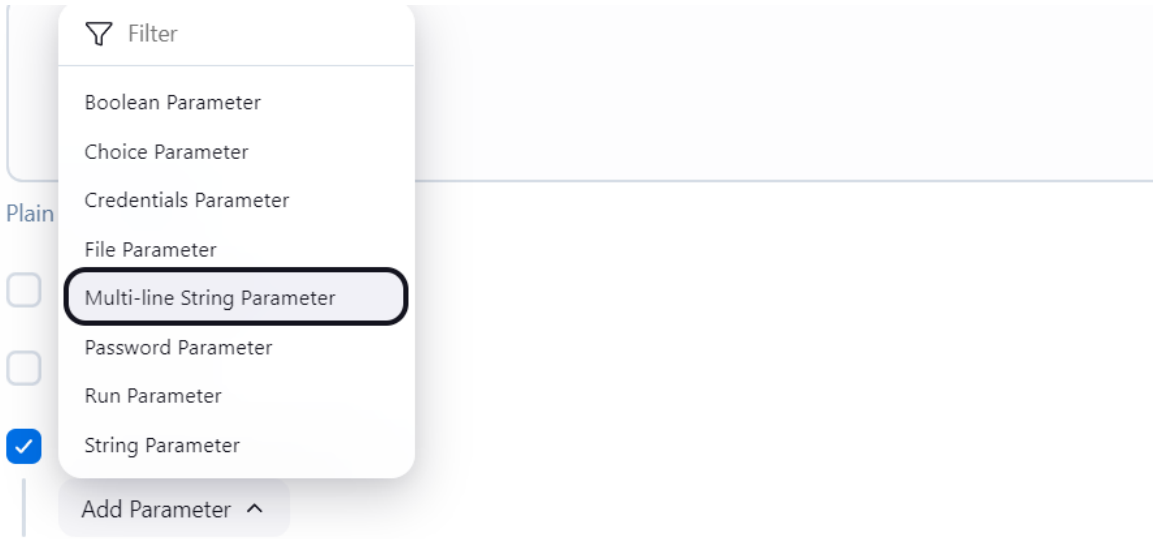


2nd: Implementation of parameter versioning

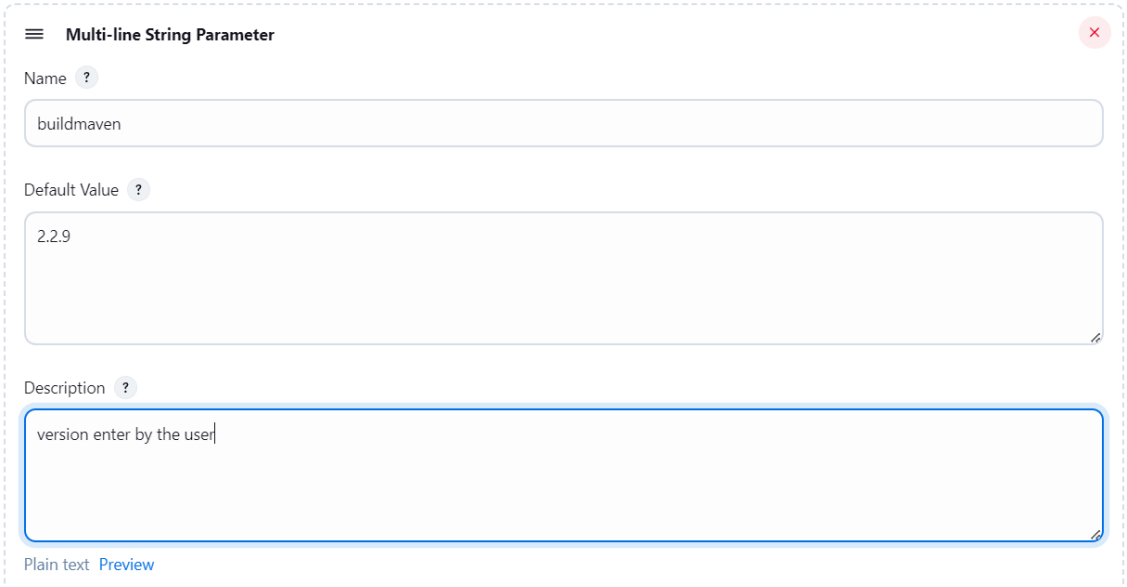
1. Go to mavenbuild & click on configure select option “This project is parameterized”



2. After that click on add parameter & select multi-line string parameter

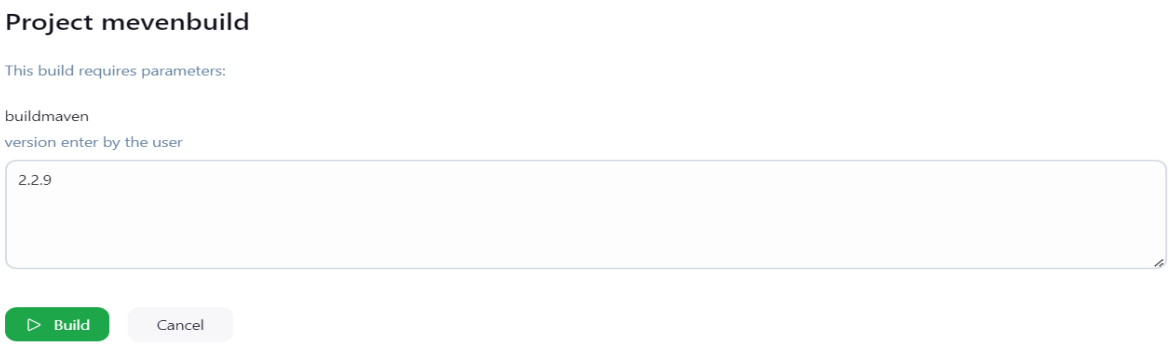


3. Then add name, default value, description

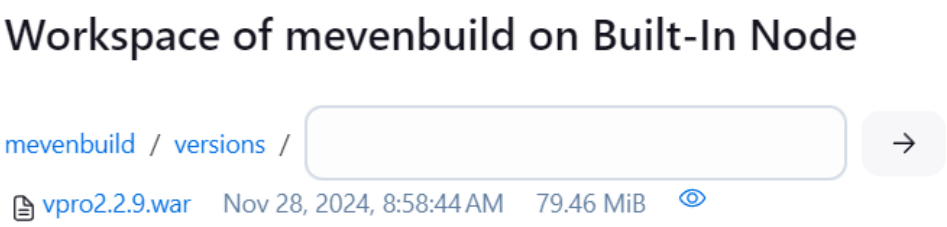


4. Then write command on “execute shell” & click on save

5. Click on build with parameters, enter any versions & click on build

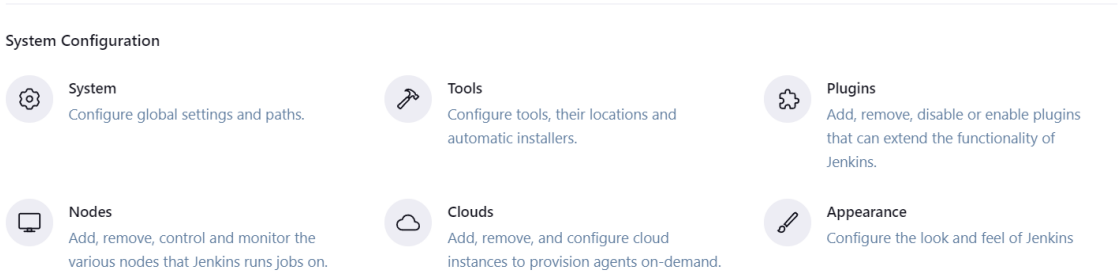


6. After that click on workspace choose versions & click on it you get your file.



3rd: Implementation of parameter versioning

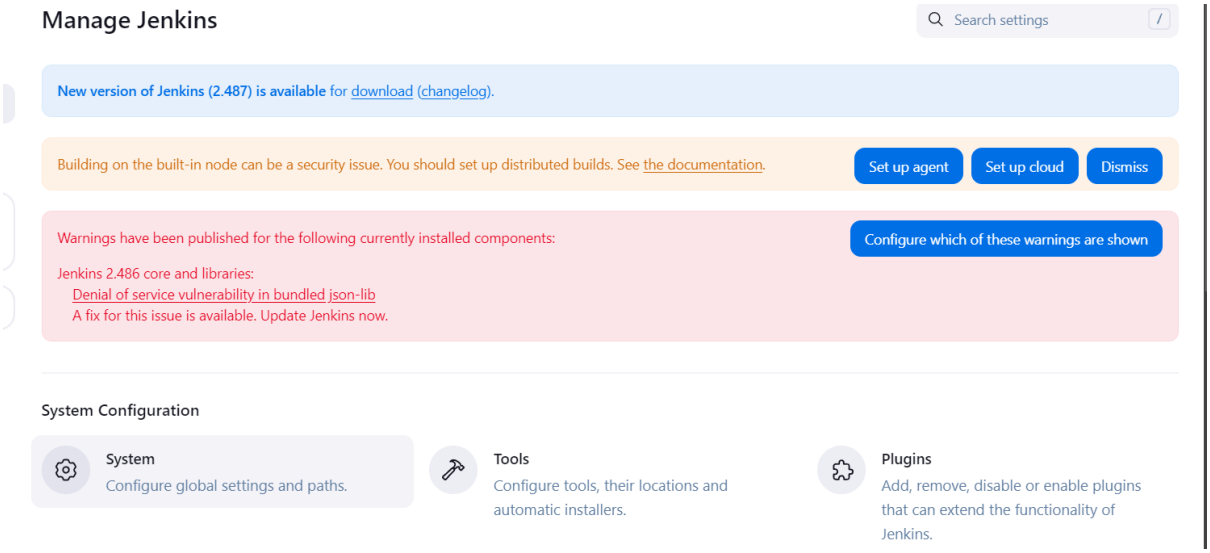
1. Go to dashboard, click on Jenkins manage & click on plugins



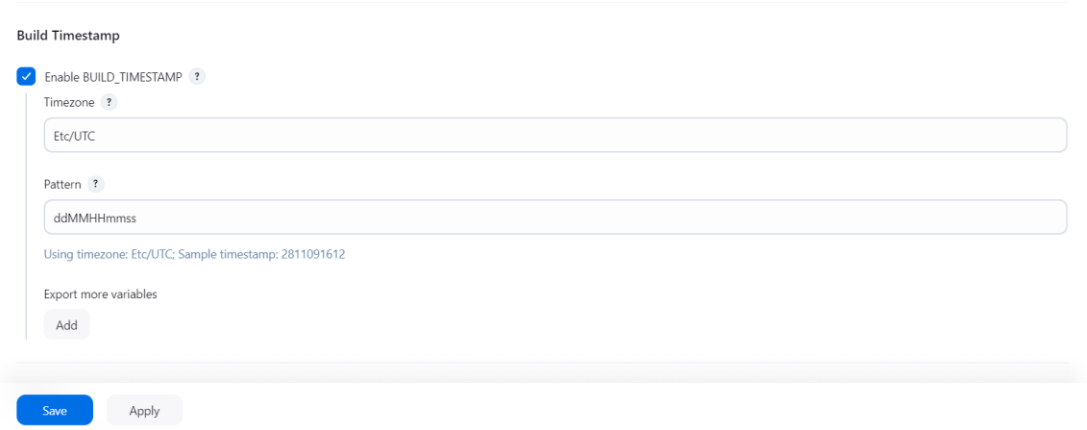
2. Click on Available plugins & search timestamp select it and install it

   I already installed so, it doesn't show

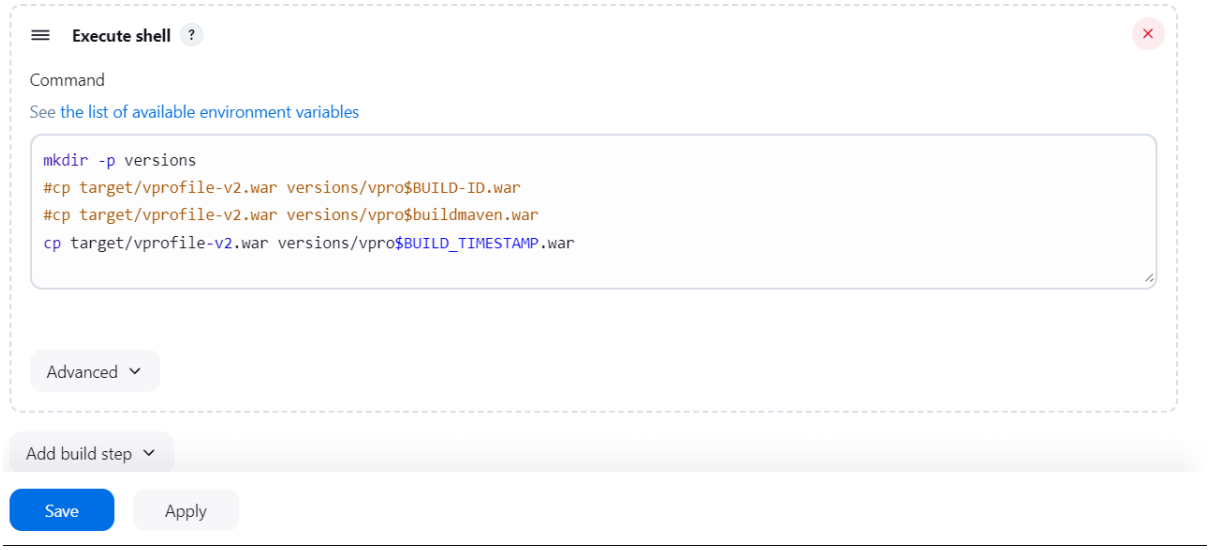
3. Go back to Jenkins manage & click on system



4. Select build timestamp & enable build stamp, change pattern then save it.



5. Go back to configure & write command in “execute shell” and save it.



6. After that click on workspace choose versions folder & open it you get your file.