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

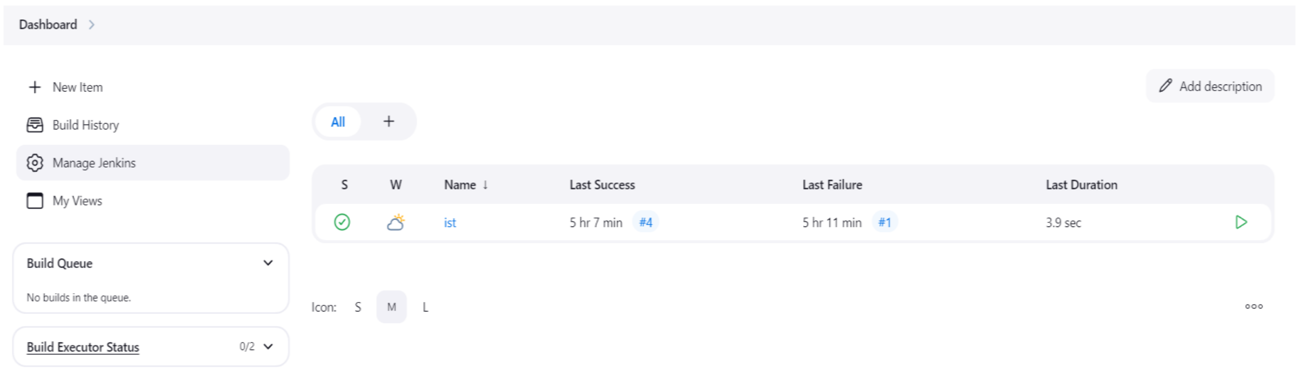
- Open a SSH connection to the specified host as the specified username,
- Check the default version of Java for that user,
- [not implemented yet] If the default version is not compatible with Jenkins's agent.jar, try to find a proper version of Java
- Once it has a suitable version of Java, copy the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



S	W	Name ↓	Last Success	Last Failure	Last Duration
✓	⛅	ist	5 hr 7 min #4	5 hr 11 min #1	3.9 sec

Build Queue

No builds in the queue.

Build Executor Status 0/2

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):

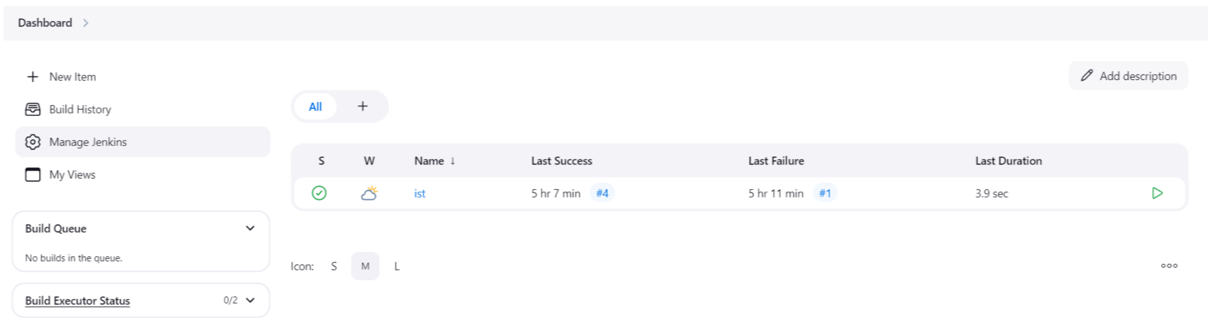
Allows users to add a description for the current job or dashboard view, which can be helpful for tracking purpose or adding notes.

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

27-Nov-2024

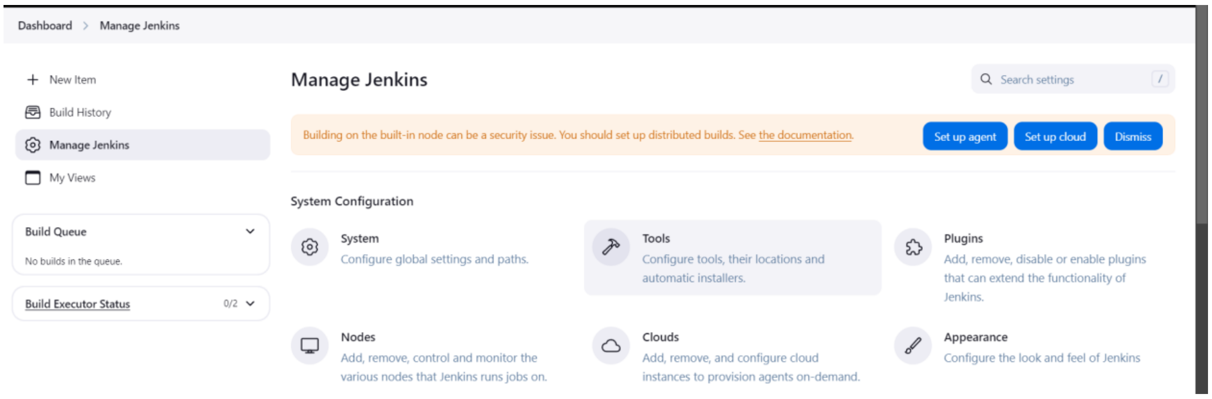
Internship Day - 88 Report:

1. **Click on dashboard & Click Manage Jenkins**



S	W	Name ↓	Last Success	Last Failure	Last Duration
✓	⛅	ist	5 hr 7 min #4	5 hr 11 min #1	3.9 sec

2. **Click on Tools**



Building on the built-in node can be a security issue. You should set up distributed builds. See the documentation.

System — Configure global settings and paths.

Tools — Configure tools, their locations and automatic installers.

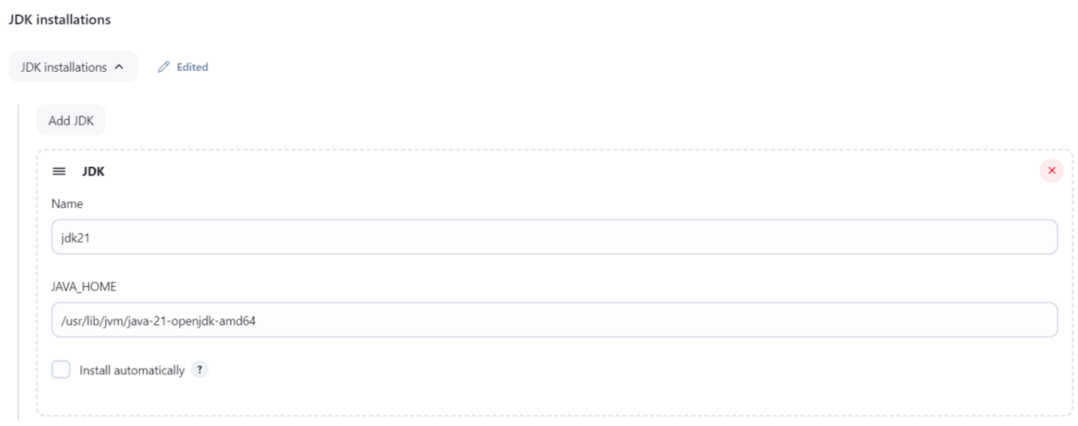
Plugins — Add, remove, disable or enable plugins that can extend the functionality of Jenkins.

Nodes — Add, remove, control and monitor the various nodes that Jenkins runs jobs on.

Clouds — Add, remove, and configure cloud instances to provision agents on-demand.

Appearance — Configure the look and feel of Jenkins

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

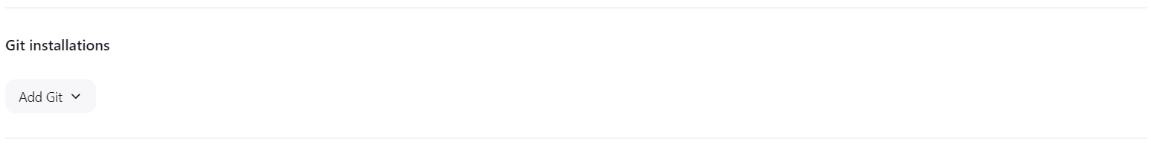


Name: jdk21

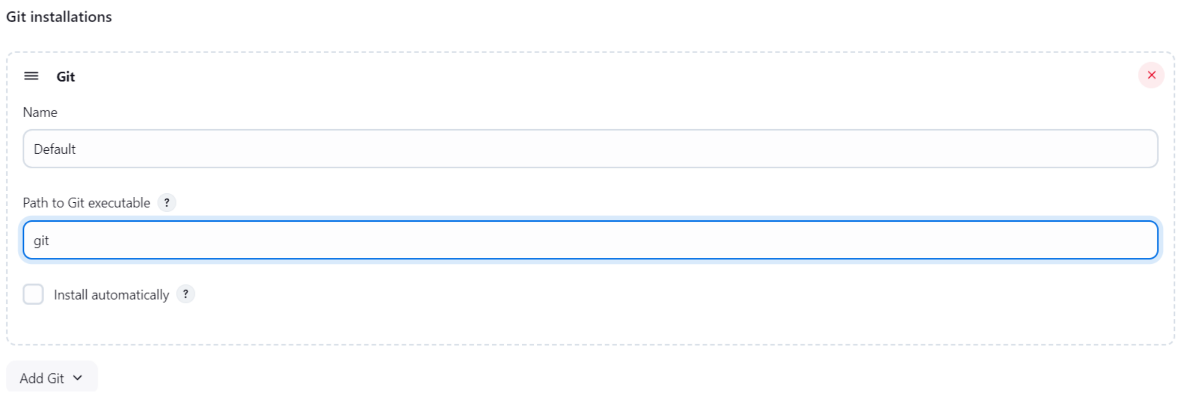
JAVA_HOME: /usr/lib/jvm/java-21-openjdk-amd64

☐ Install automatically

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


Git installations

5. **Then Fill Details:**


Git installations

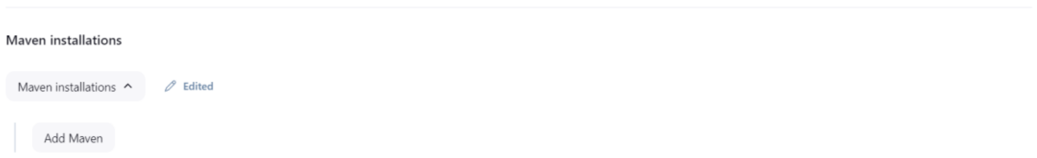
Git

Name: Default

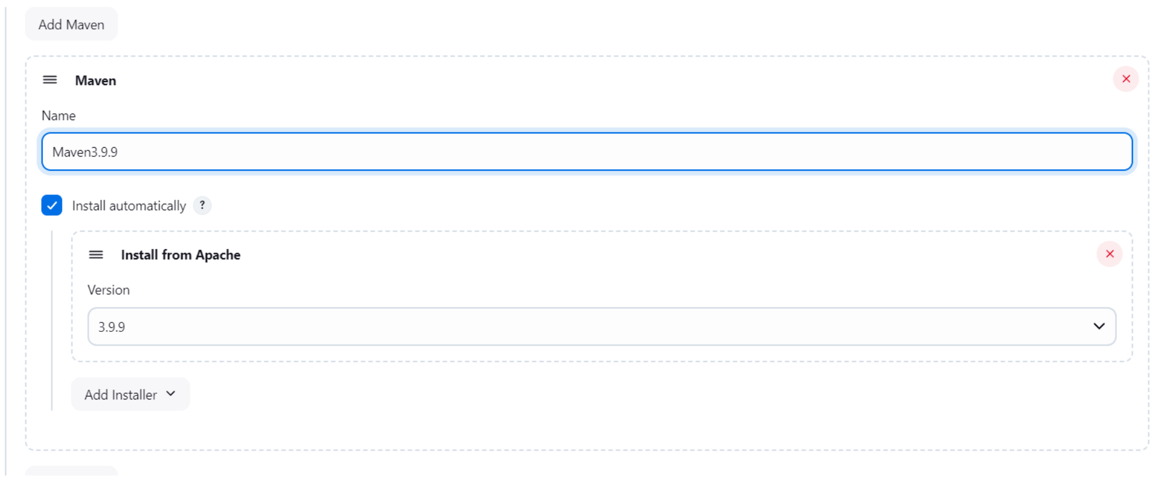
Path to Git executable: git

Install automatically: ☐

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


Maven installations

7. **Enter Name and version**



Maven

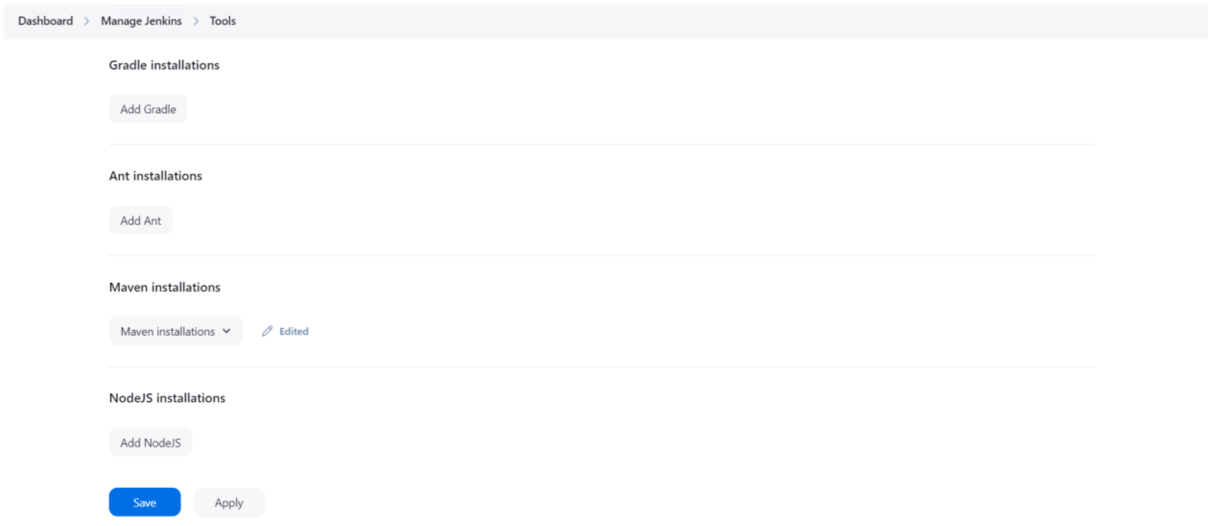
Name: Maven3.9.9

Install automatically: ☑

Install from Apache

Version: 3.9.9

8. After that click on apply & then save


Dashboard > Manage Jenkins > Tools

Gradle installations

Add Gradle

Ant installations

Add Ant

Maven installations

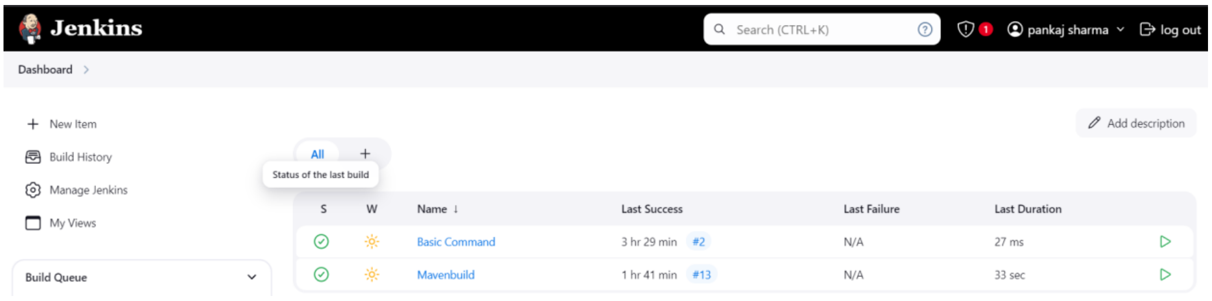
Maven installations ⌄ Edited

NodeJS installations

Add NodeJS

Save Apply

9. Go to dashboard & Click on New Item:


Jenkins — Search (CTRL+K) — pankaj sharma — log out

Dashboard >

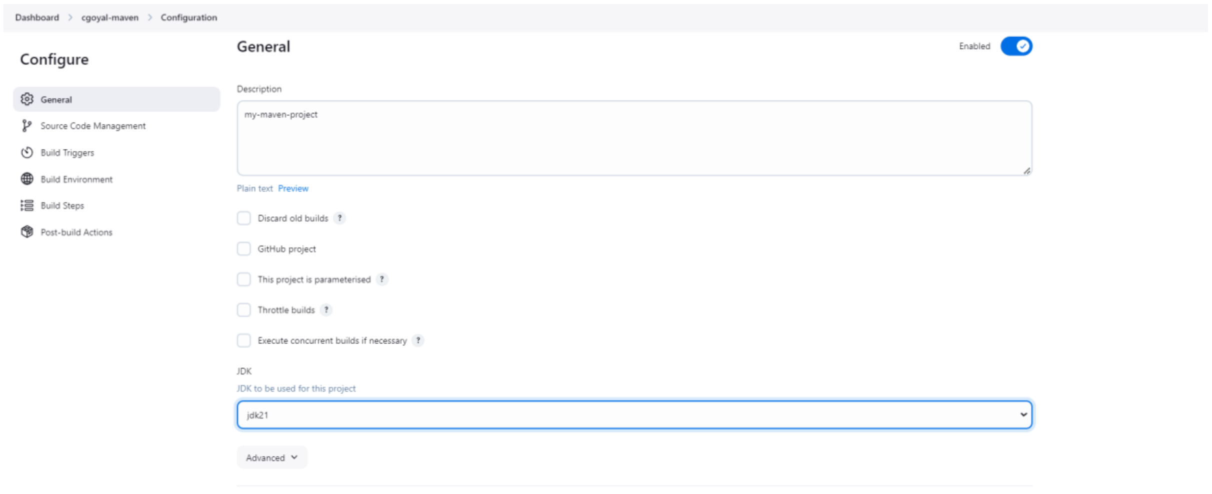
New Item
Build History
Manage Jenkins
My Views
Build Queue

Add description

All +
Status of the last build

S	W	Name ↓	Last Success	Last Failure	Last Duration
✓	☀	Basic Command	3 hr 29 min #2	N/A	27 ms
✓	☀	Mavenbuild	1 hr 41 min #13	N/A	33 sec

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


Dashboard > cgoyal-maven > Configuration

Configure

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

General — Enabled

Description

my-maven-project

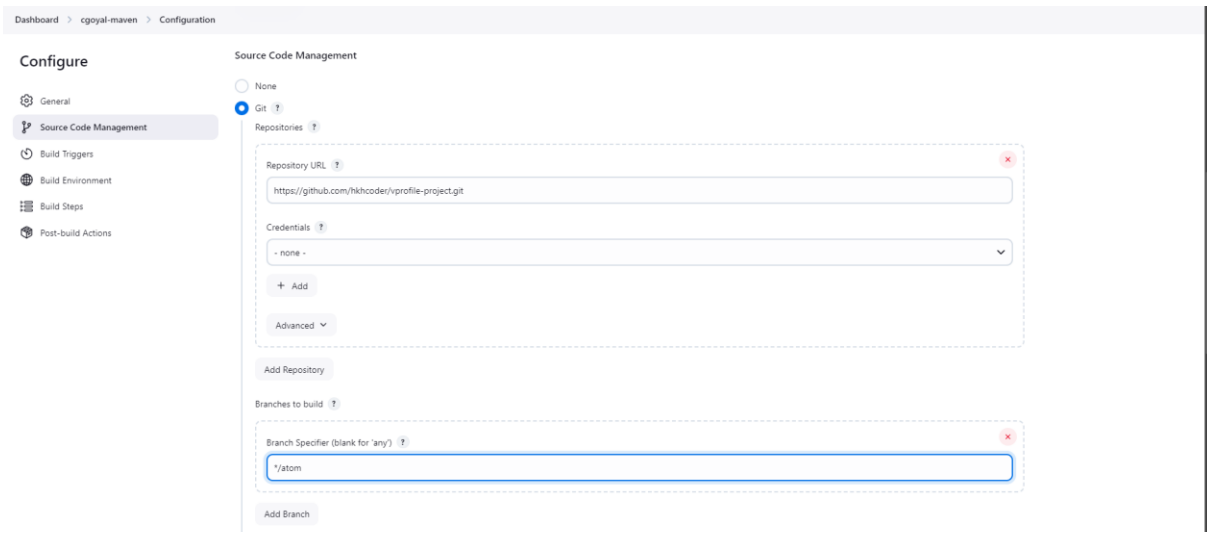
Plain text Preview

- [ ] Discard old builds
- [ ] GitHub project
- [ ] This project is parameterised
- [ ] Throttle builds
- [ ] Execute concurrent builds if necessary

JDK
JDK to be used for this project
jdk21

Advanced ⌄

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


Dashboard > cgoyal-maven > Configuration

Configure: General, Source Code Management, Build Triggers, Build Environment, Build Steps, Post-build Actions

Source Code Management

None
Git (selected)

Repositories

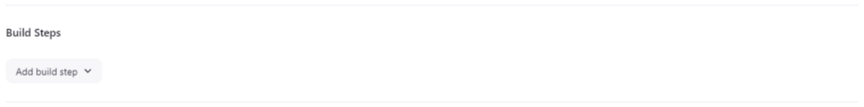
Repository URL: https://github.com/hkhcoder/vprofile-project.git

Credentials: - none -

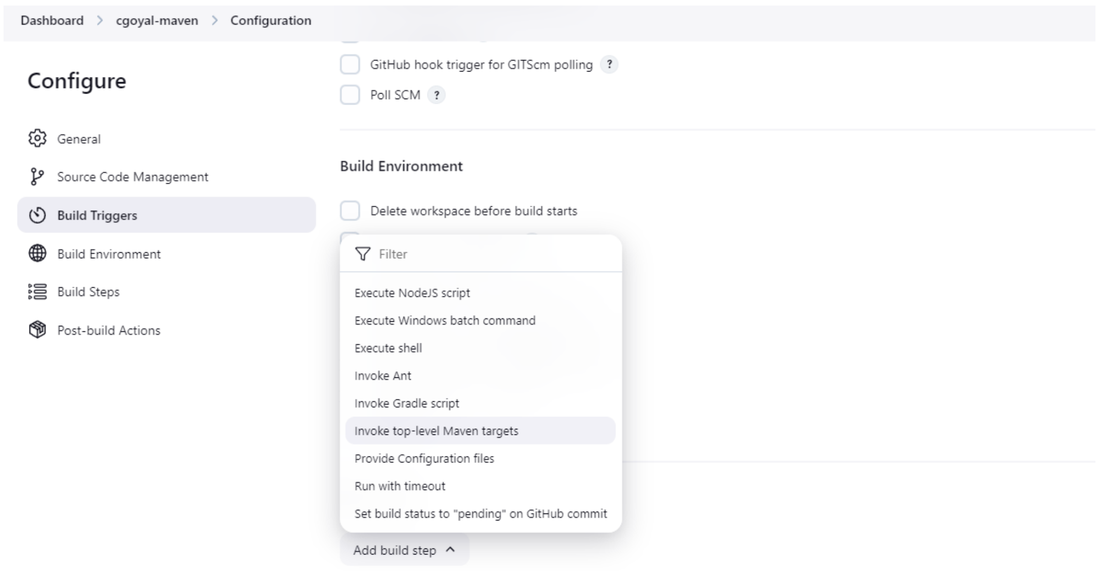
Branches to build

Branch Specifier (blank for 'any'): */atom

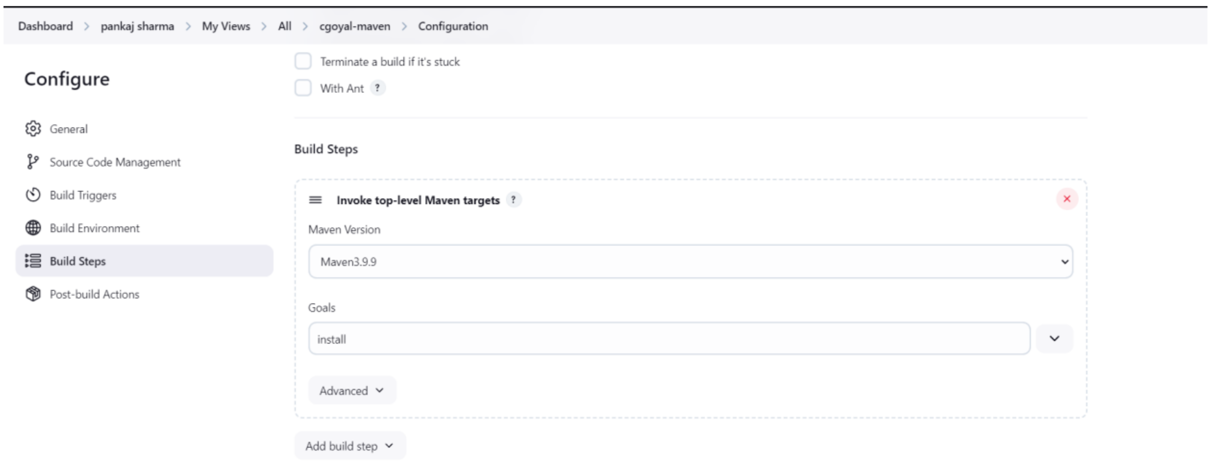
13. **Click on Add Build Setups**



14. **Select Invoke top-level Maven Targets**


Dashboard > cgoyal-maven > Configuration

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


Dashboard > pankaj sharma > My Views > All > cgoyal-maven > Configuration

Configure

Terminate a build if it's stuck

With Ant

Build Steps

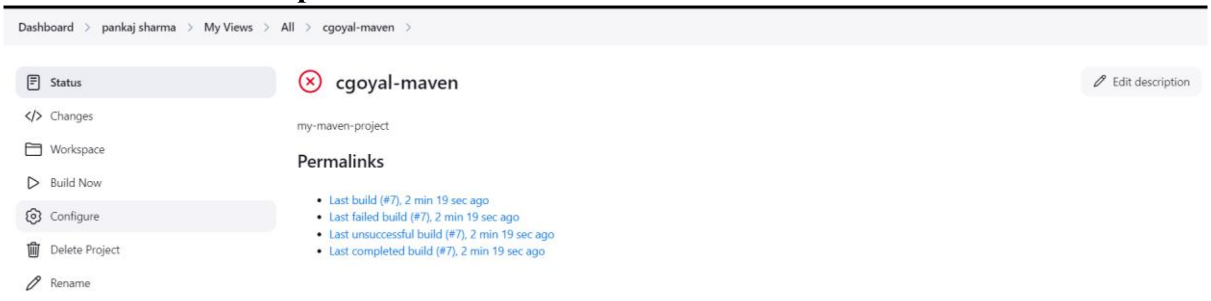
Invoke top-level Maven targets

Maven Version: Maven3.9.9

Goals: install

16. Click Save and Apply

17. Click to Build Now Option


Dashboard > pankaj sharma > My Views > All > cgoyal-maven >

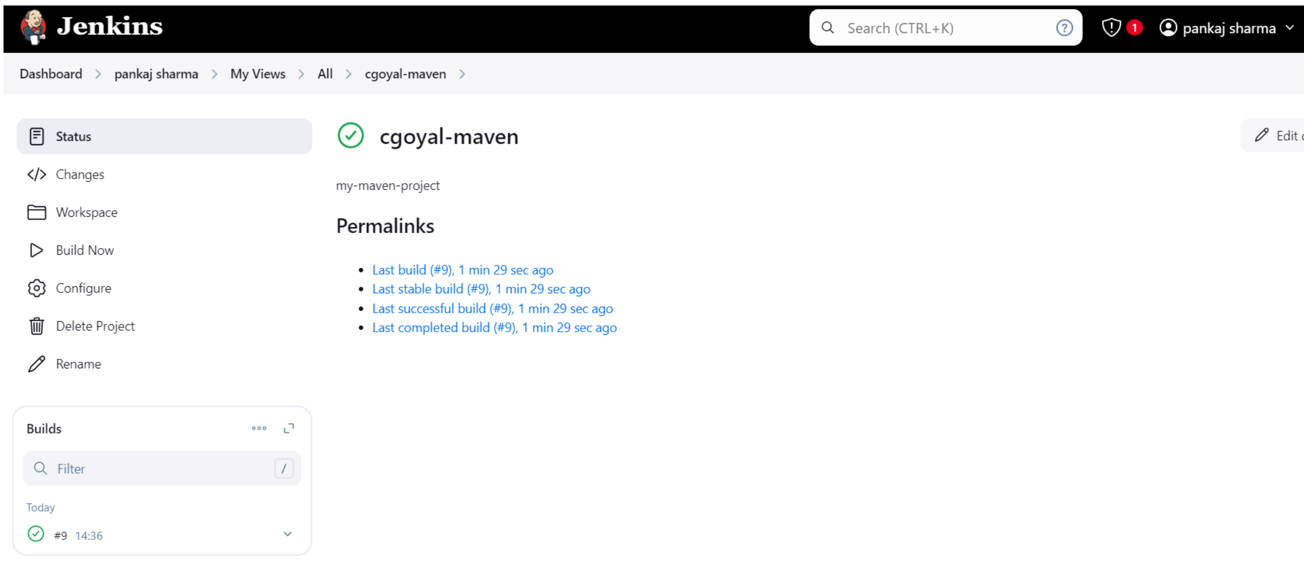
cgoyal-maven

my-maven-project

Permalinks

- Last build (#7), 2 min 19 sec ago
- Last failed build (#7), 2 min 19 sec ago
- Last unsuccessful build (#7), 2 min 19 sec ago
- Last completed build (#7), 2 min 19 sec ago

18. Now, Your Build now is done


Jenkins

Dashboard > pankaj sharma > My Views > All > cgoyal-maven >

cgoyal-maven

my-maven-project

Permalinks

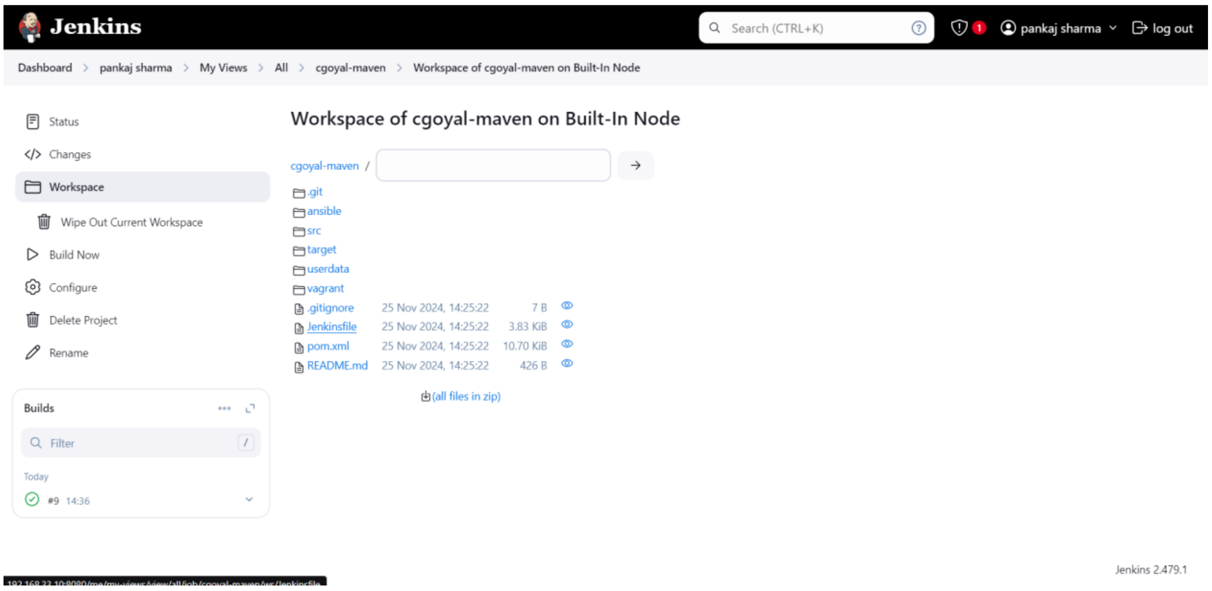
- Last build (#9), 1 min 29 sec ago
- Last stable build (#9), 1 min 29 sec ago
- Last successful build (#9), 1 min 29 sec ago
- Last completed build (#9), 1 min 29 sec ago

Builds

Today

#9 14:36

19. Go to Workspace and see your all project files



Jenkins | Search (CTRL+K) | pankaj sharma | log out

Dashboard > pankaj sharma > My Views > All > cgoyal-maven > Workspace of cgoyal-maven on Built-In Node

Status
Changes
Workspace
Wipe Out Current Workspace
Build Now
Configure
Delete Project
Rename

Builds
Filter
Today
#9 14:36

Workspace of cgoyal-maven on Built-In Node

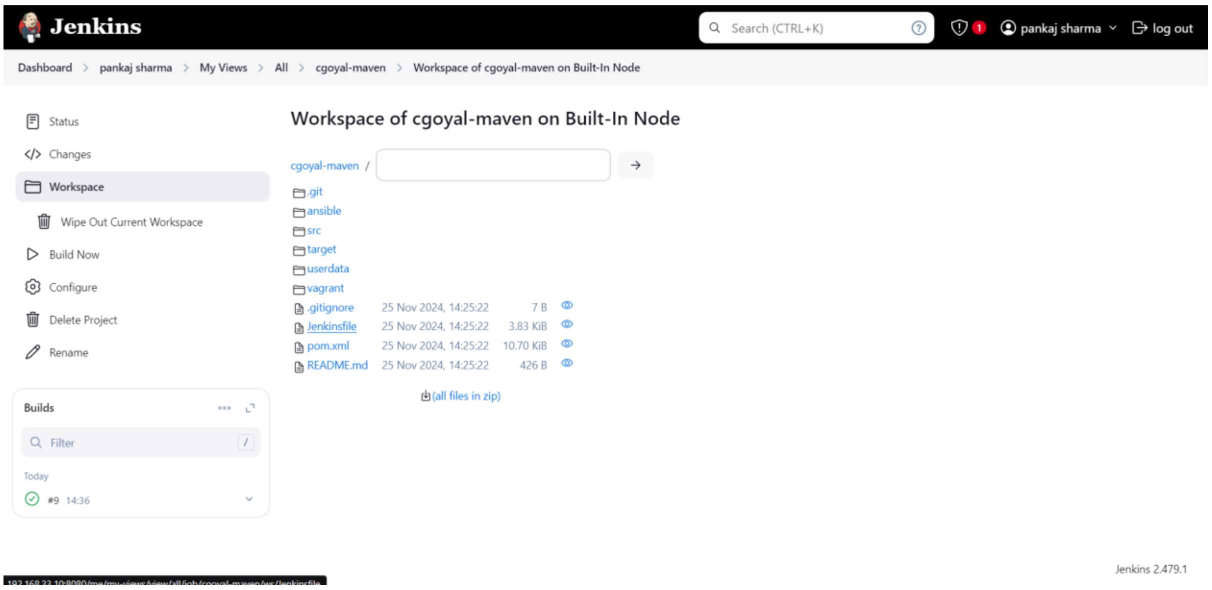
cgoyal-maven /

.git			
ansible			
src			
target			
userdata			
vagrant			
.gitignore	25 Nov 2024, 14:25:22	7 B	
Jenkinsfile	25 Nov 2024, 14:25:22	3.83 KiB	
pom.xml	25 Nov 2024, 14:25:22	10.70 KiB	
README.md	25 Nov 2024, 14:25:22	426 B	

(all files in zip)

Jenkins 2.479.1

20. Click on target and see your .war files



Jenkins | Search (CTRL+K) | pankaj sharma | log out

Dashboard > pankaj sharma > My Views > All > cgoyal-maven > Workspace of cgoyal-maven on Built-In Node

Status
Changes
Workspace
Wipe Out Current Workspace
Build Now
Configure
Delete Project
Rename

Builds
Filter
Today
#9 14:36

Workspace of cgoyal-maven on Built-In Node

cgoyal-maven /

.git
ansible
src
target
userdata
vagrant
.gitignore 25 Nov 2024, 14:25:22 7 B
Jenkinsfile 25 Nov 2024, 14:25:22 3.83 KiB
pom.xml 25 Nov 2024, 14:25:22 10.70 KiB
README.md 25 Nov 2024, 14:25:22 426 B

(all files in zip)

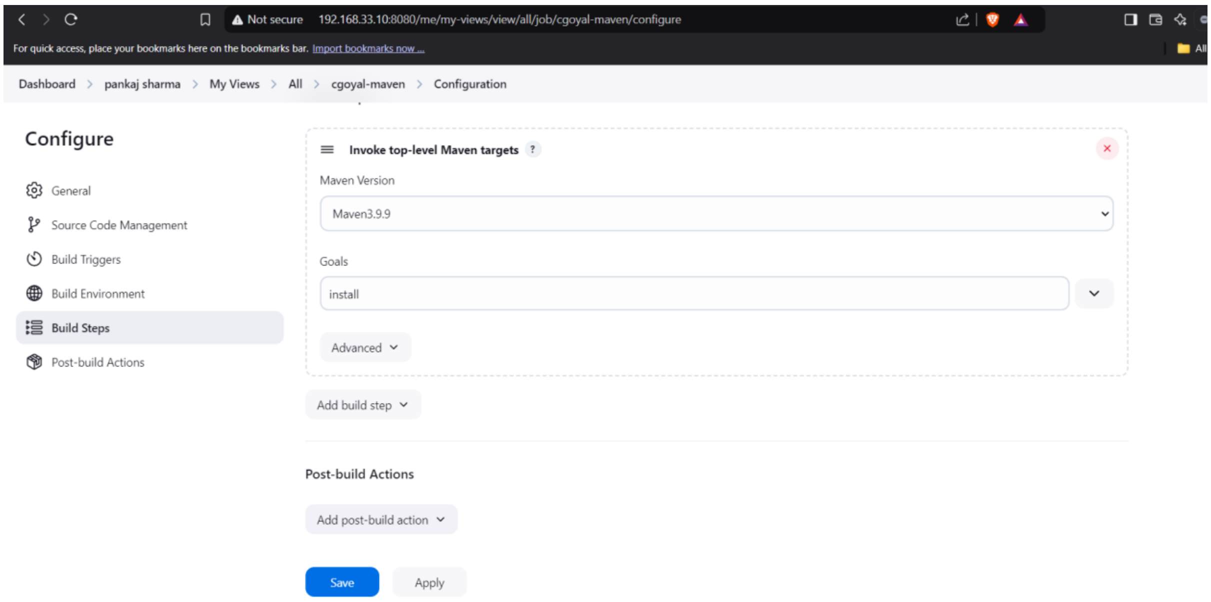
Jenkins 2.479.1

28-Nov-2024

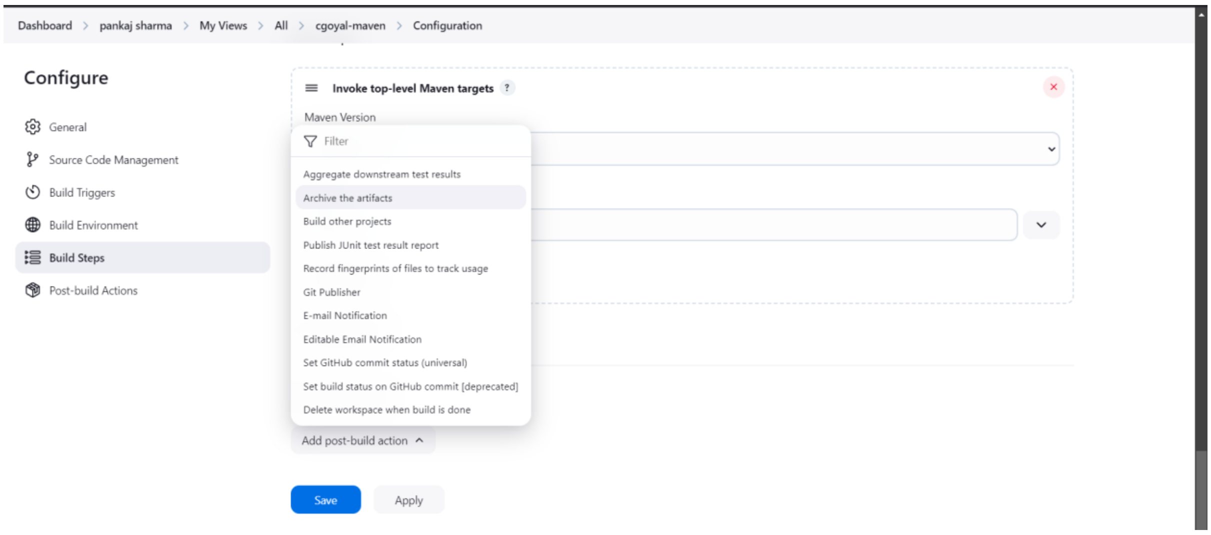
Internship Day - 87 Report:

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

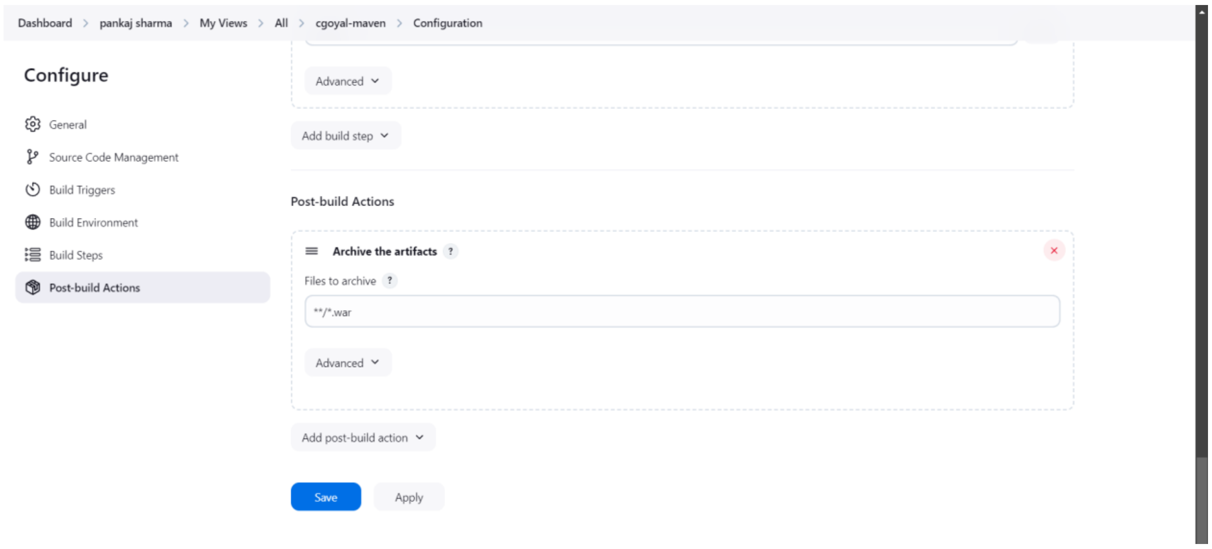
1. Click on Configure and Click Add post-build action


Not secure 192.168.33.10:8080/me/my-views/view/all/job/cgoyal-maven/configure
For quick access, place your bookmarks here on the bookmarks bar. Import bookmarks now ...
Dashboard > pankaj sharma > My Views > All > cgoyal-maven > Configuration
Configure
General
Source Code Management
Build Triggers
Build Environment
Build Steps
Post-build Actions
Invoke top-level Maven targets
Maven Version
Maven3.9.9
Goals
install
Advanced
Add build step
Post-build Actions
Add post-build action
Save
Apply

2. Click on Archive the aircrafts


Dashboard > pankaj sharma > My Views > All > cgoyal-maven > Configuration
Configure
General
Source Code Management
Build Triggers
Build Environment
Build Steps
Post-build Actions
Invoke top-level Maven targets
Maven Version
Filter
Aggregate downstream test results
Archive the artifacts
Build other projects
Publish JUnit test result report
Record fingerprints of files to track usage
Git Publisher
E-mail Notification
Editable Email Notification
Set GitHub commit status (universal)
Set build status on GitHub commit [deprecated]
Delete workspace when build is done
Add post-build action
Save
Apply

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

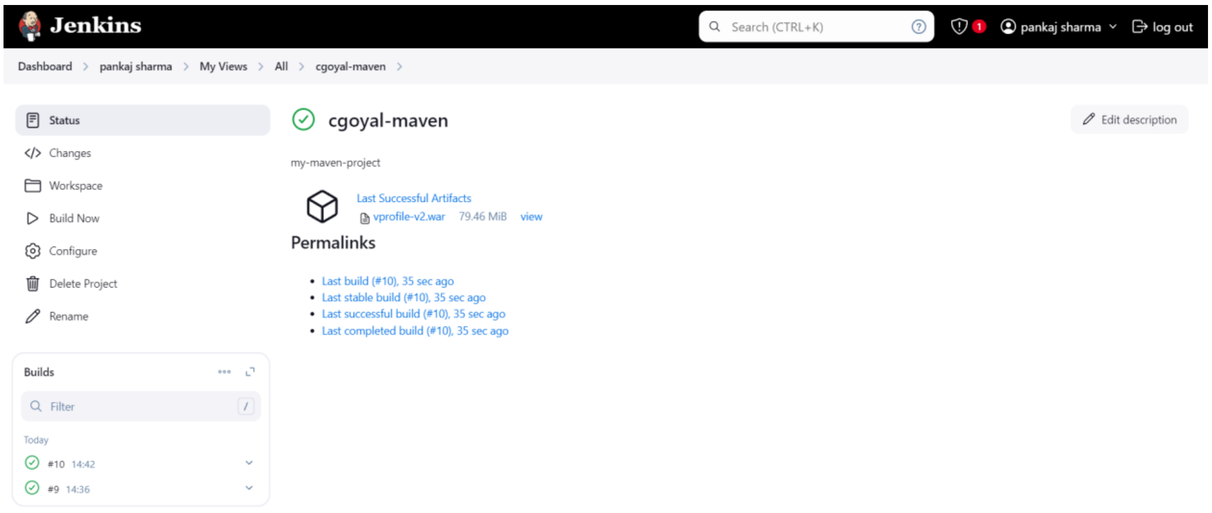


Post-build Actions

Archive the artifacts

Files to archive: **/*.war

4. Now you see your file in front



cgoyal-maven

my-maven-project

Last Successful Artifacts
vprofile-v2.war 79.46 MiB view

Permalinks

- Last build (#10), 35 sec ago
- Last stable build (#10), 35 sec ago
- Last successful build (#10), 35 sec ago
- Last completed build (#10), 35 sec ago

#10 14:42
#9 14:36

29-Nov-2024

Internship Day - 89 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).

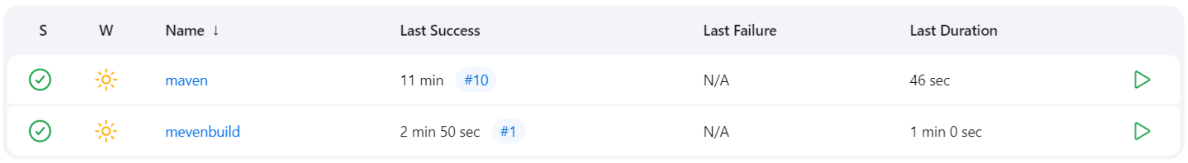
This integration ensures consistency and enables continuous integration and delivery (CI/CD) by automating processes around version-controlled code.

DIFFERENT TYPES OF METHOD OF VERSIONS

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

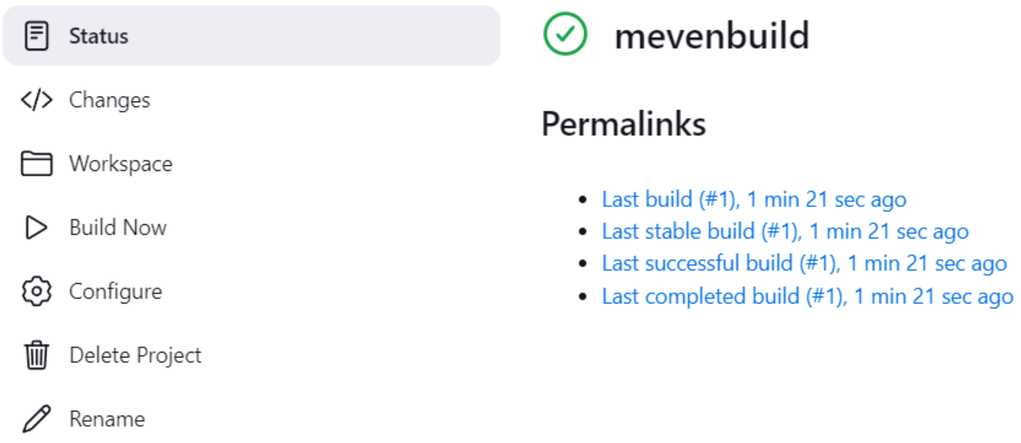
1st: Implementation of manual versioning

1. Click on dashboard and select job(mavenbuild)



S	W	Name ↓	Last Success	Last Failure	Last Duration	
✓	☀	maven	11 min #10	N/A	46 sec	▷
✓	☀	mevenbuild	2 min 50 sec #1	N/A	1 min 0 sec	▷

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

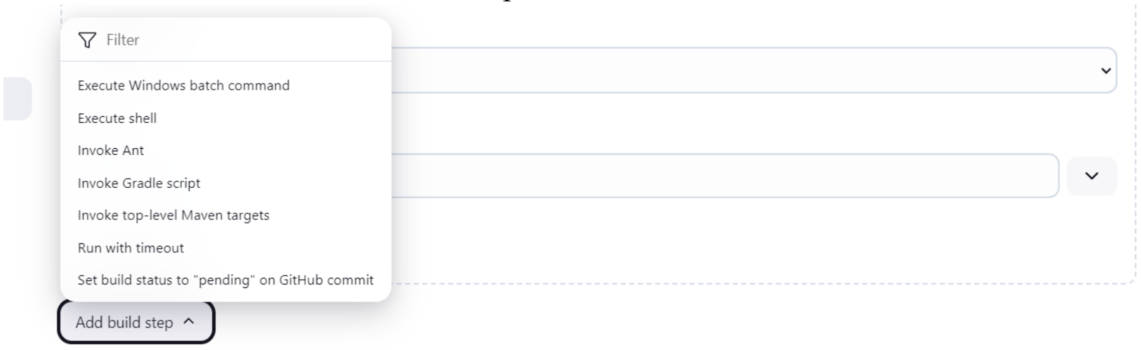


mevenbuild

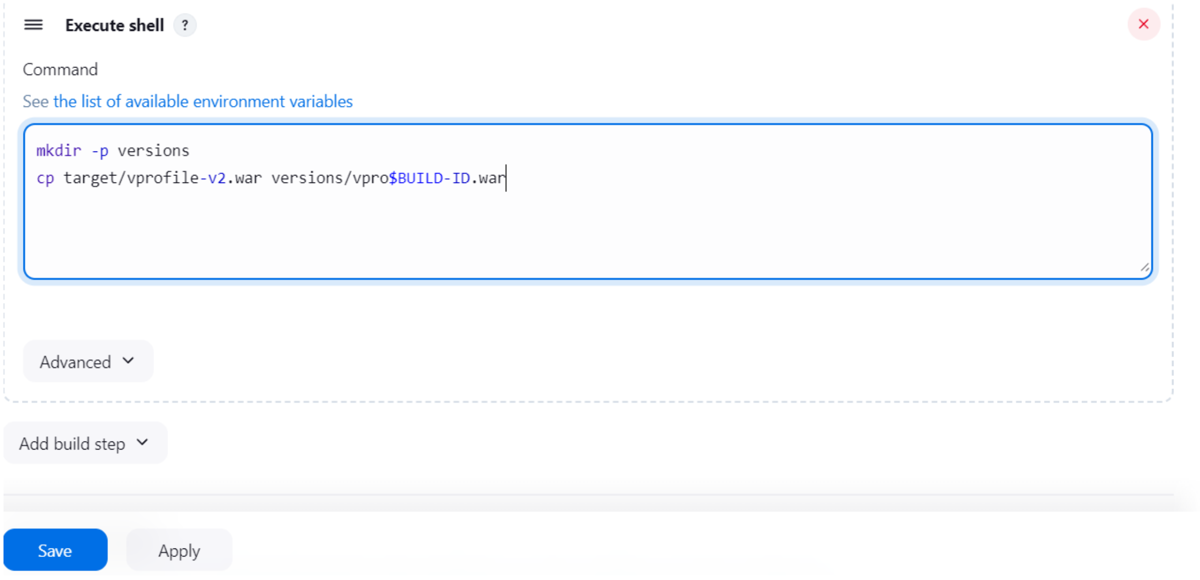
Permalinks

- Last build (#1), 1 min 21 sec ago
- Last stable build (#1), 1 min 21 sec ago
- Last successful build (#1), 1 min 21 sec ago
- Last completed build (#1), 1 min 21 sec ago

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

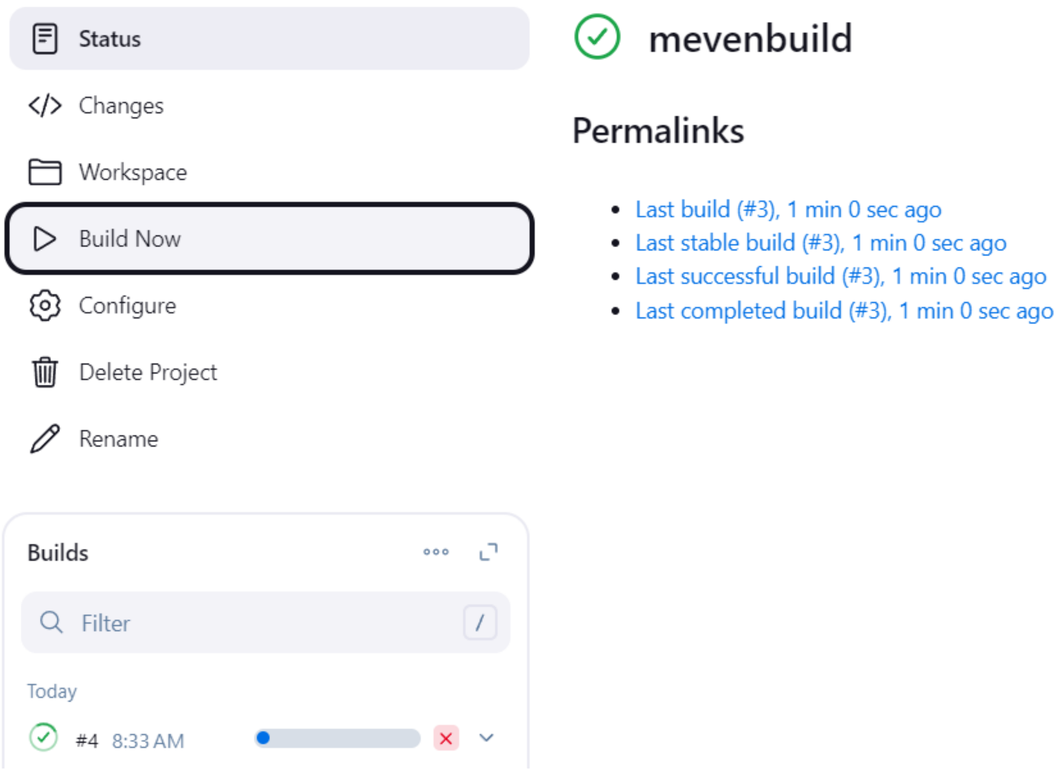

Filter
Execute Windows batch command
Execute shell
Invoke Ant
Invoke Gradle script
Invoke top-level Maven targets
Run with timeout
Set build status to "pending" on GitHub commit
Add build step

4. Write command in execute shell & save it



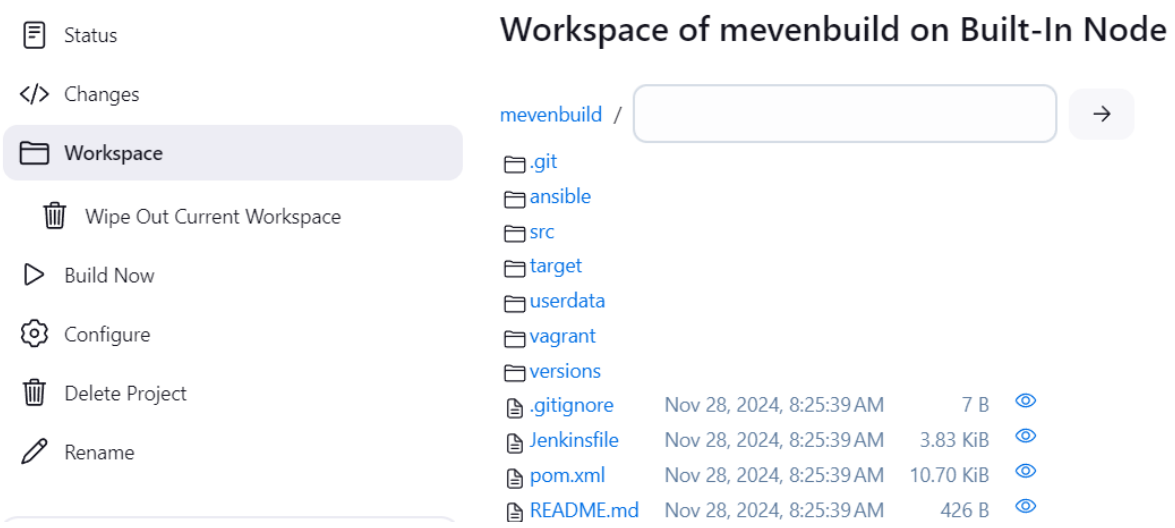
```
mkdir -p versions
cp target/vprofile-v2.war versions/vpro$BUILD-ID.war
```

5. Click on build now



- Last build (#3), 1 min 0 sec ago
- Last stable build (#3), 1 min 0 sec ago
- Last successful build (#3), 1 min 0 sec ago
- Last completed build (#3), 1 min 0 sec ago

6. Go to workspace & you see versions folder

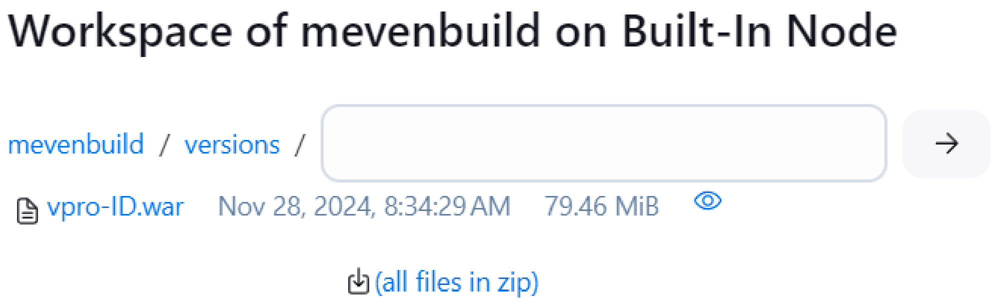


Workspace of mevenbuild on Built-In Node

mevenbuild /

.git		
ansible		
src		
target		
userdata		
vagrant		
versions		
.gitignore	Nov 28, 2024, 8:25:39 AM	7 B
Jenkinsfile	Nov 28, 2024, 8:25:39 AM	3.83 KiB
pom.xml	Nov 28, 2024, 8:25:39 AM	10.70 KiB
README.md	Nov 28, 2024, 8:25:39 AM	426 B

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



Workspace of mevenbuild on Built-In Node

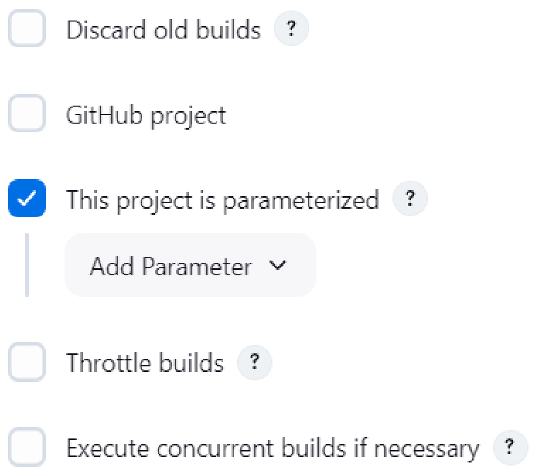
mevenbuild / versions /

vpro-ID.war	Nov 28, 2024, 8:34:29 AM	79.46 MiB

(all files in zip)

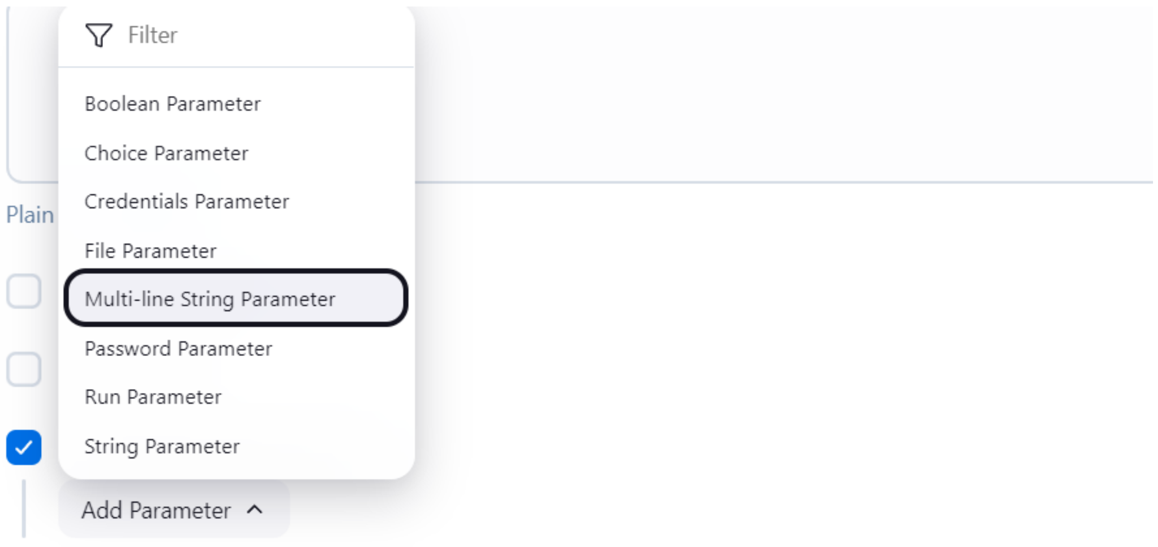
2nd: Implementation of parameter versioning

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

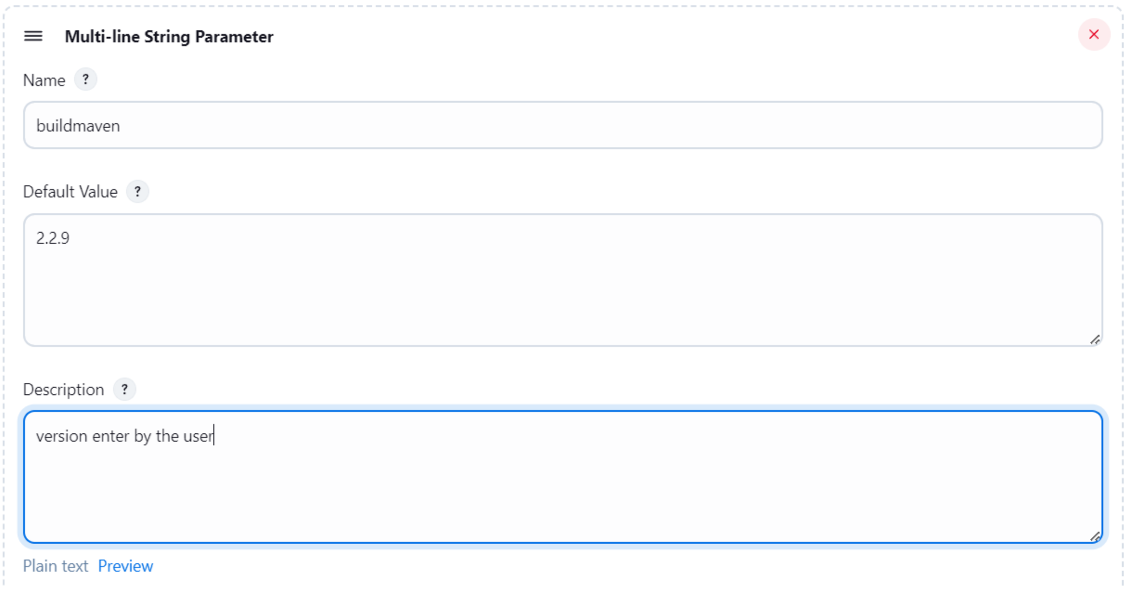


Discard old builds
GitHub project
This project is parameterized
Add Parameter
Throttle builds
Execute concurrent builds if necessary

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



3. Then add name, default value, description



Name

buildmaven

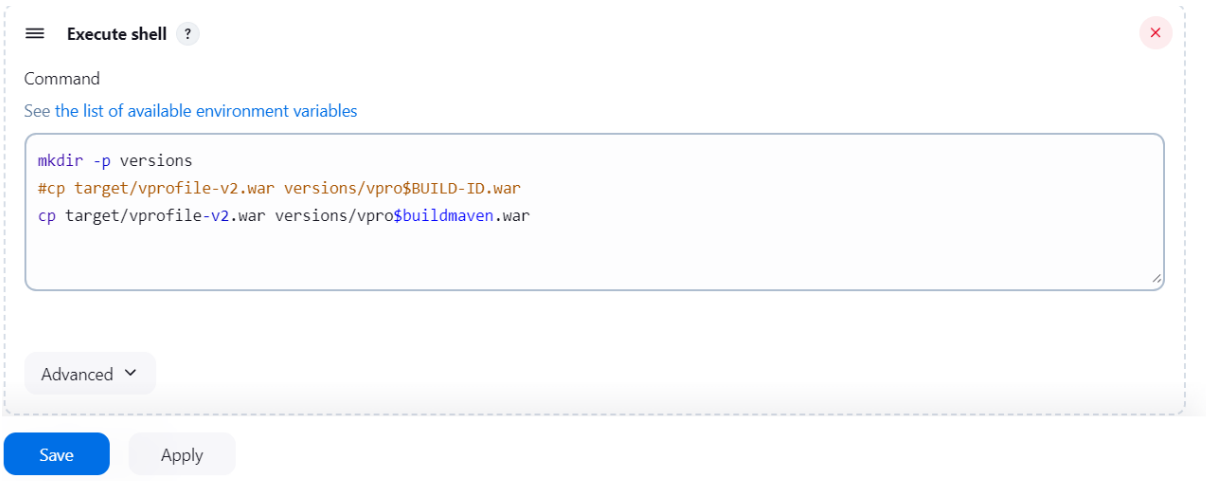
Default Value

2.2.9

Description

version enter by the user

4. Then write command on "execute shell" & click on save

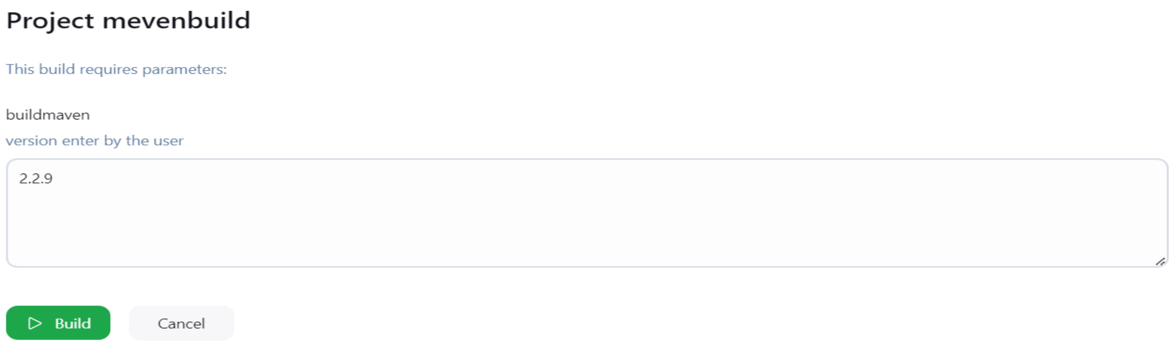


Command

See the list of available environment variables

```
mkdir -p versions
#cp target/vprofile-v2.war versions/vpro$BUILD-ID.war
cp target/vprofile-v2.war versions/vpro$buildmaven.war
```

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



Project mevenbuild

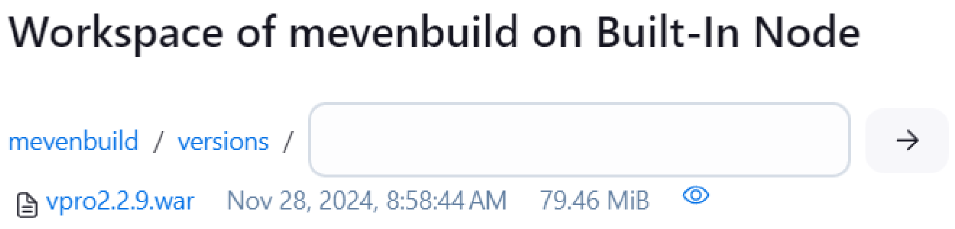
This build requires parameters:

buildmaven

version enter by the user

2.2.9

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



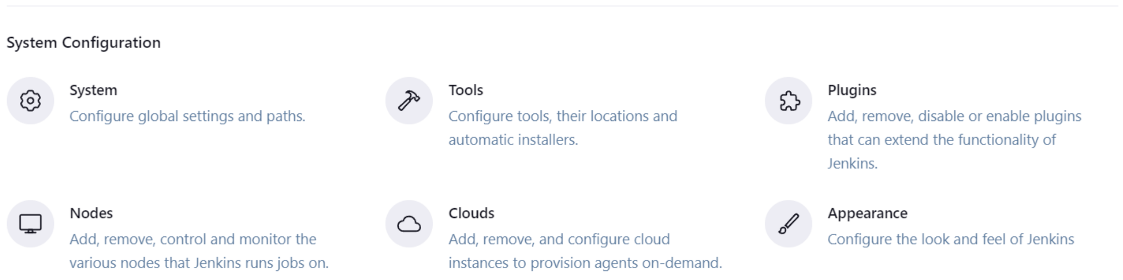
Workspace of mevenbuild on Built-In Node

mevenbuild / versions /

vpro2.2.9.war Nov 28, 2024, 8:58:44 AM 79.46 MiB

3rd: Implementation of parameter versioning

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


System Configuration

System
Configure global settings and paths.

Tools
Configure tools, their locations and automatic installers.

Plugins
Add, remove, disable or enable plugins that can extend the functionality of Jenkins.

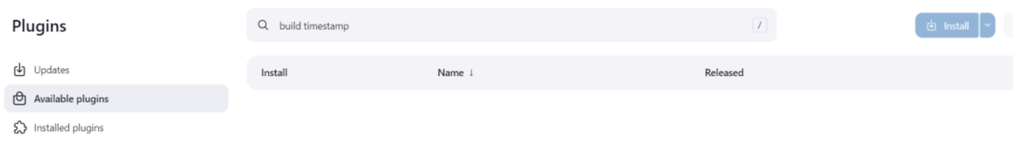
Nodes
Add, remove, control and monitor the various nodes that Jenkins runs jobs on.

Clouds
Add, remove, and configure cloud instances to provision agents on-demand.

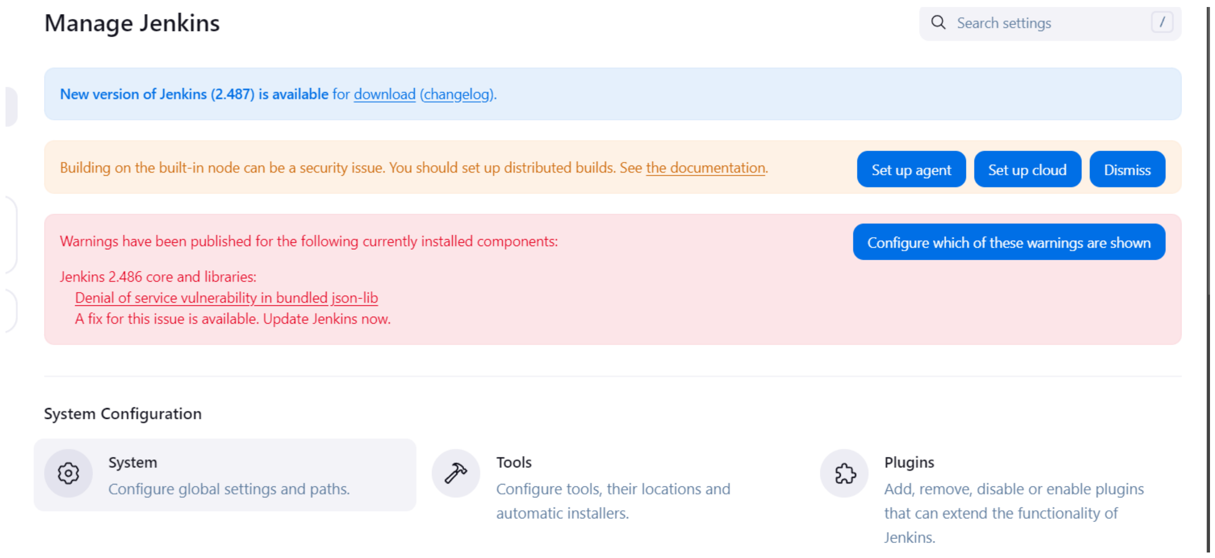
Appearance
Configure the look and feel of Jenkins

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


Manage Jenkins

Search settings

New version of Jenkins (2.487) is available for download (changelog).

Building on the built-in node can be a security issue. You should set up distributed builds. See the documentation.

Set up agent | Set up cloud | Dismiss

Warnings have been published for the following currently installed components:

Jenkins 2.486 core and libraries:
Denial of service vulnerability in bundled json-lib
A fix for this issue is available. Update Jenkins now.

Configure which of these warnings are shown

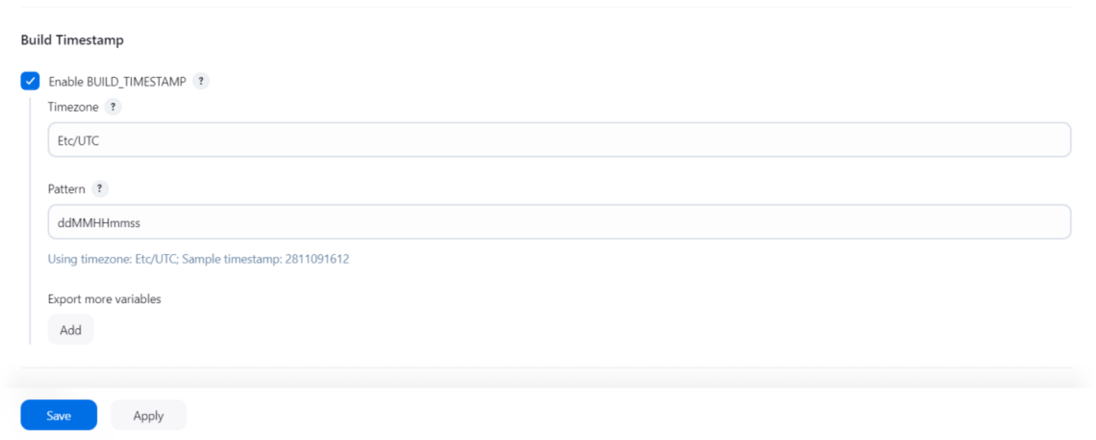
System Configuration

System
Configure global settings and paths.

Tools
Configure tools, their locations and automatic installers.

Plugins
Add, remove, disable or enable plugins that can extend the functionality of Jenkins.

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



Build Timestamp

☑ Enable BUILD_TIMESTAMP

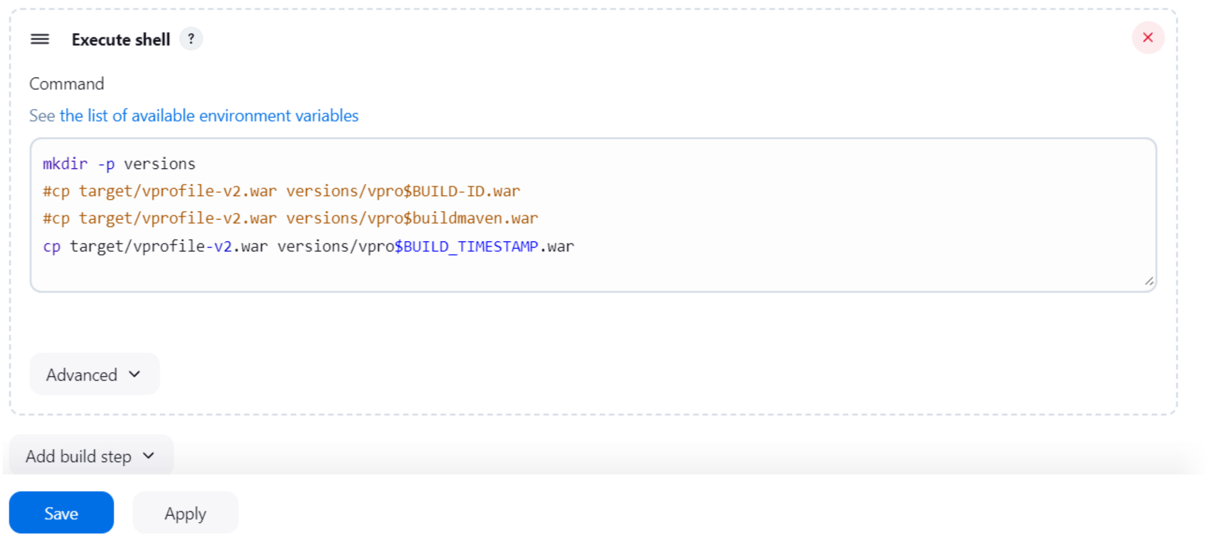
Timezone: Etc/UTC

Pattern: ddMMHHmmss

Using timezone: Etc/UTC; Sample timestamp: 2811091612

Export more variables

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



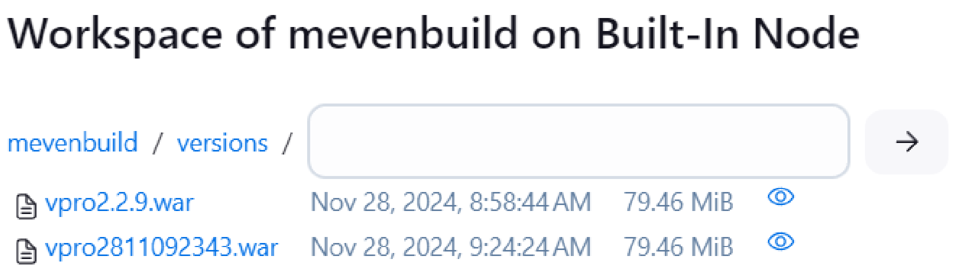
Execute shell

Command

See the list of available environment variables

```
mkdir -p versions
#cp target/vprofile-v2.war versions/vpro$BUILD-ID.war
#cp target/vprofile-v2.war versions/vpro$buildmaven.war
cp target/vprofile-v2.war versions/vpro$BUILD_TIMESTAMP.war
```

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



Workspace of mevenbuild on Built-In Node

mevenbuild / versions /

vpro2.2.9.war	Nov 28, 2024, 8:58:44 AM	79.46 MiB
vpro2811092343.war	Nov 28, 2024, 9:24:24 AM	79.46 MiB