

## Experiment 05

**Aim :** To build the pipeline of jobs using Maven/Gradle/Ant in Jenkins , create a pipeline script to test and deploy an application

**Theory :** Jenkins Pipeline (or simply "Pipeline" with a capital "P") is a suite of plugins which supports implementing and integrating *continuous delivery pipelines* into Jenkins.

A *continuous delivery (CD) pipeline* is an automated expression of your process for getting software from version control right through to your users and customers.

the definition of a Jenkins Pipeline is written into a text file (called a [Jenkinsfile](#)) which in turn can be committed to a project's source control repository.

Creating a **Jenkinsfile** and committing it to source control provides a number of immediate benefits:

- Automatically creates a Pipeline build process for all branches and pull requests.
- Code review/iteration on the Pipeline (along with the remaining source code).
- Audit trail for the Pipeline.

By modeling a series of related tasks, users can take advantage of the many features of Pipeline:

- **Code:** Pipelines are implemented in code and typically checked into source control, giving teams the ability to edit, review, and iterate upon their delivery pipeline.
- **Durable:** Pipelines can survive both planned and unplanned restarts of the Jenkins controller.
- **Pausable:** Pipelines can optionally stop and wait for human input or approval before continuing the Pipeline run.
- **Versatile:** Pipelines support complex real-world CD requirements, including the ability to fork/join, loop, and perform work in parallel.
- **Extensible:** The Pipeline plugin supports custom extensions to its DSL <sup>[1]</sup> and multiple options for integration with other plugins.

### Pipeline concepts

#### Pipeline

A Pipeline is a user-defined model of a CD pipeline. Also, a **pipeline** block is a [key part of Declarative Pipeline syntax](#).

#### Node

A node is a machine which is part of the Jenkins environment and is capable of executing a Pipeline.

## Stage







A **stage** block defines a conceptually distinct subset of tasks performed through the entire Pipeline (e.g. "Build", "Test" and "Deploy" stages).

## Step

A single task. Fundamentally, a step tells Jenkins *what* to do at a particular point in time (or "step" in the process). For example, to execute the shell command **make** use the **sh** step: **sh 'make'**.

## Procedure :

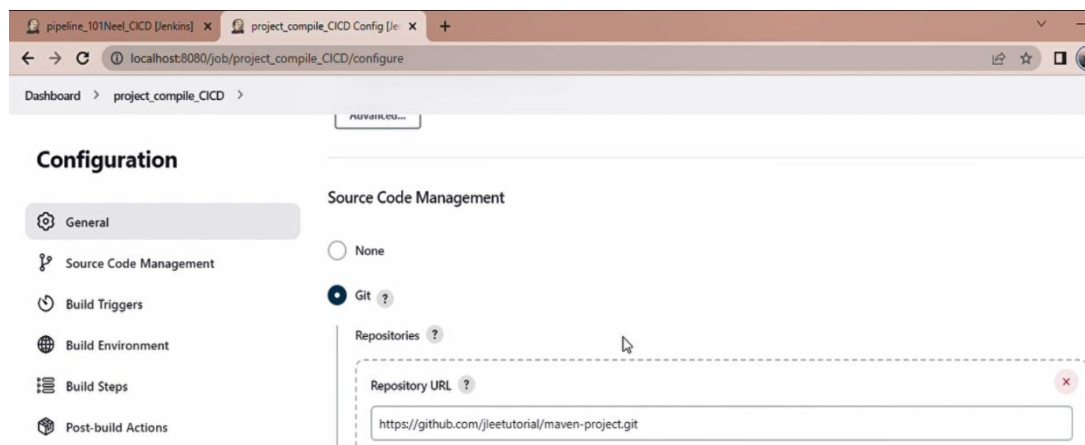
1. First Build a pipeline by naming it and choosing pipeline in the project type section. We will then create three projects to build a pipeline , mainly a test

	project_compile_CICD	13 days #3	N/A	2.9 sec	
	project_package01	13 days #2	N/A	4 sec	
	project_test01	13 days #4	N/A	4.1 sec	

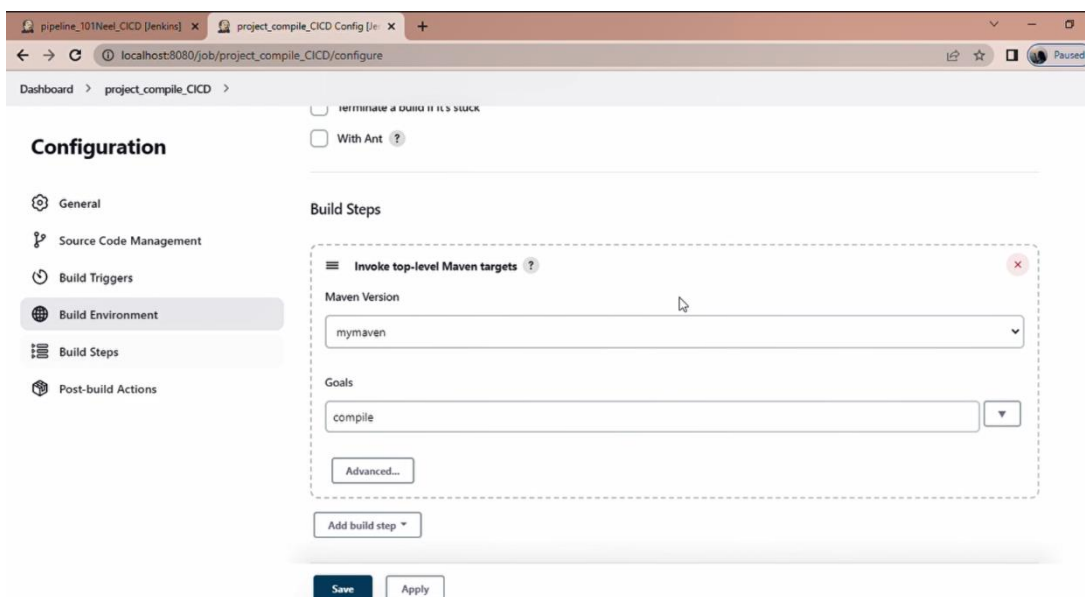
project , a compile project and a package project .

2. Now click on the compile project and perform the following actions , go to the source code management and paste the following Git repo link.

<https://github.com/jleetutorial/maven-project/>



3. Go to the Add Build Step action and select invoke top-level maven targets and select the mymaven version and in goals write compile



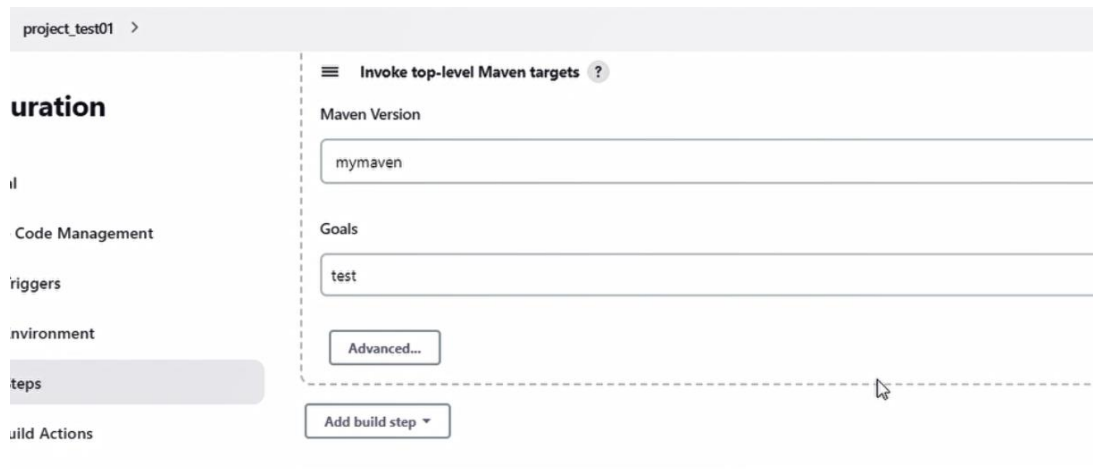
- Go to post build section and choose build other projects and in that add your test project ( desired name ) and keep the default trigger setting . after that click on apply and save respectively .

The screenshot shows the Jenkins Configuration page with the 'Post-build Actions' tab selected in the left sidebar. The main content area is titled 'Post-build Actions' and contains a section for 'Build other projects'. Under this section, there is a text input field labeled 'Projects to build' containing the text 'project\_test01'. Below the input field, there are three radio button options: 'Trigger only if build is stable' (which is selected), 'Trigger even if the build is unstable', and 'Trigger even if the build fails'. At the bottom of the section, there is a button labeled 'Add post-build action'.

- Don't run and build any project right now .
- Now go to your test project , again go to the source code management and paste the following Git repo link.  
<https://github.com/ileetutorial/maven-project>
- Go to build triggers section and choose build after project are build and add the name of the project compile in it.

The screenshot shows the Jenkins Configuration page with the 'Build Triggers' tab selected in the left sidebar. The main content area is titled 'Build Triggers' and contains several options. The first two options are 'Trigger builds remotely (e.g., from scripts)' and 'Build after other projects are built', both with checkboxes. The 'Build after other projects are built' checkbox is checked. Below these options, there is a text input field labeled 'Projects to watch' containing the text 'project\_compile\_CICD'. Below the input field, there are five radio button options: 'Trigger only if build is stable' (which is selected), 'Trigger even if the build is unstable', 'Trigger even if the build fails', and 'Always trigger, even if the build is aborted'. At the bottom of the section, there are three more checkboxes: 'Build periodically', 'GitHub hook trigger for GITScm polling', and 'Build SCM'.

8. Go to the Add Build Step action and select invoke top-level maven targets and select the mymaven version and in goals write test



project\_test01 >

**uration**

il

Code Management

iggers

nvironment

steps

uild Actions

≡ Invoke top-level Maven targets ?

Maven Version

mymaven

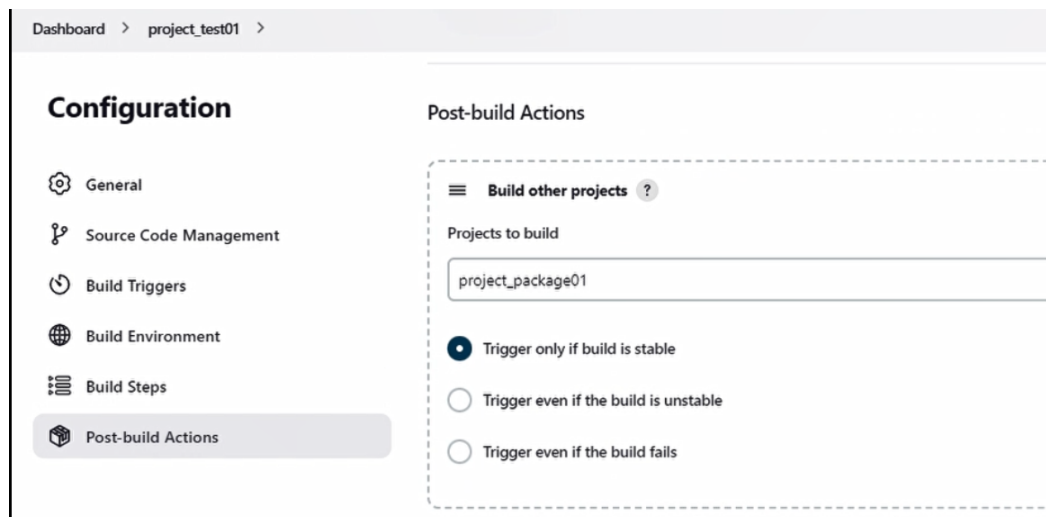
Goals

test

Advanced...

Add build step ▾

9. Go to post-build section and choose your project package name .



Dashboard > project\_test01 >

**Configuration**

General

Source Code Management

Build Triggers

Build Environment

Build Steps

Post-build Actions

Post-build Actions

≡ Build other projects ?

Projects to build

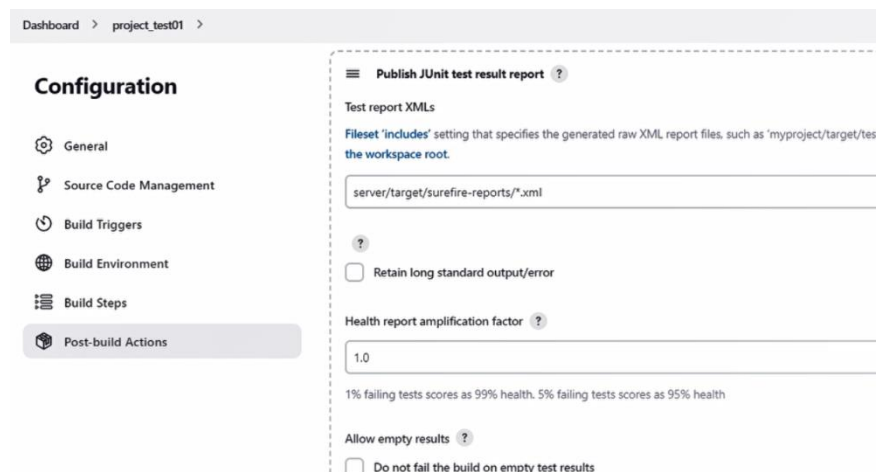
project\_package01

☒ Trigger only if build is stable

☐ Trigger even if the build is unstable

☐ Trigger even if the build fails

10. Now again add publish Junit test result report in post build actions paste the xml file link from the same git repository . keep the rest same and save



Dashboard > project\_test01 >

**Configuration**

General

Source Code Management

Build Triggers

Build Environment

Build Steps

Post-build Actions

Post-build Actions

≡ Publish JUnit test result report ?

Test report XMLs

Fileset 'includes' setting that specifies the generated raw XML report files, such as 'myproject/target/test-reports/\*.xml'.

server/target/surefire-reports/\*.xml

☐ Retain long standard output/error

Health report amplification factor ?

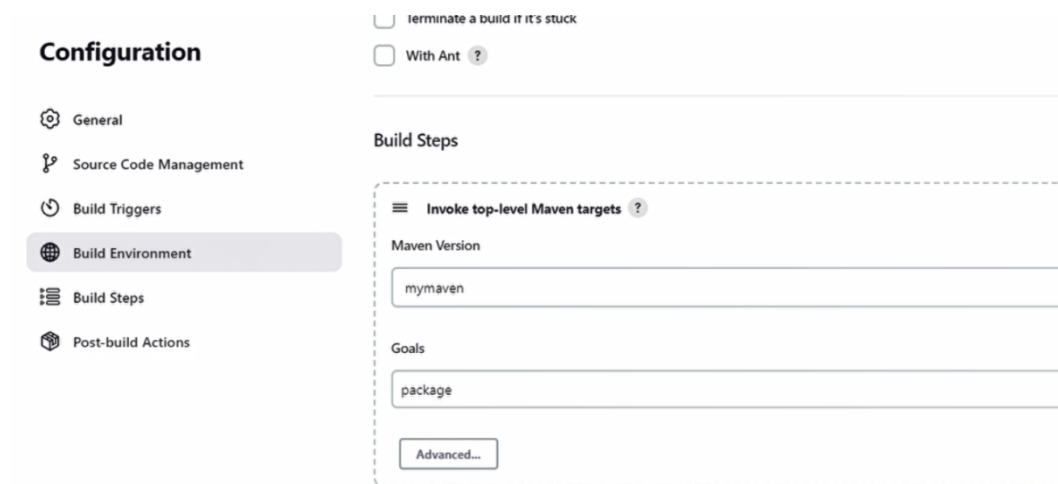
1.0

1% failing tests scores as 99% health. 5% failing tests scores as 95% health

Allow empty results ?

☐ Do not fail the build on empty test results

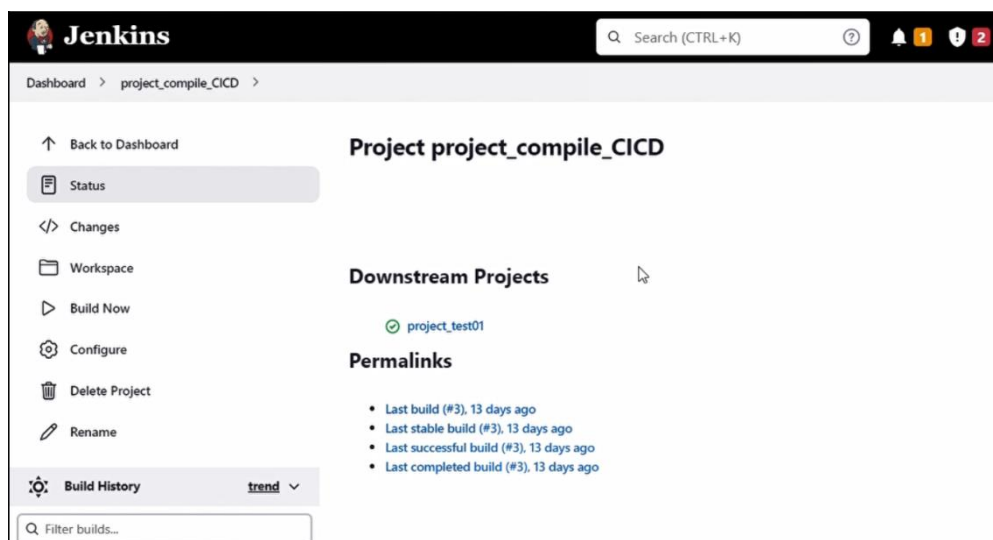
11. Now go to your project package , Now go to your test project , again go to the source code management and paste the following Git repo link.  
<https://github.com/ileetutorial/maven-project>
12. Go to build steps and choose top-level maven targets again choose the mymaven version (as per your pc) and in goals write package



The screenshot shows the Jenkins Configuration page for the Build Environment. On the left, a sidebar lists various configuration sections: General, Source Code Management, Build Triggers, Build Environment (selected), Build Steps, and Post-build Actions. The main area is titled 'Configuration' and contains two sections. The top section has two checkboxes: 'Terminate a build if it's stuck' (unchecked) and 'With Ant' (unchecked). The bottom section is titled 'Build Steps' and contains a dashed box with the title 'Invoke top-level Maven targets'. Inside this box, there are two input fields: 'Maven Version' with the value 'mymaven' and 'Goals' with the value 'package'. Below these fields is an 'Advanced...' button.

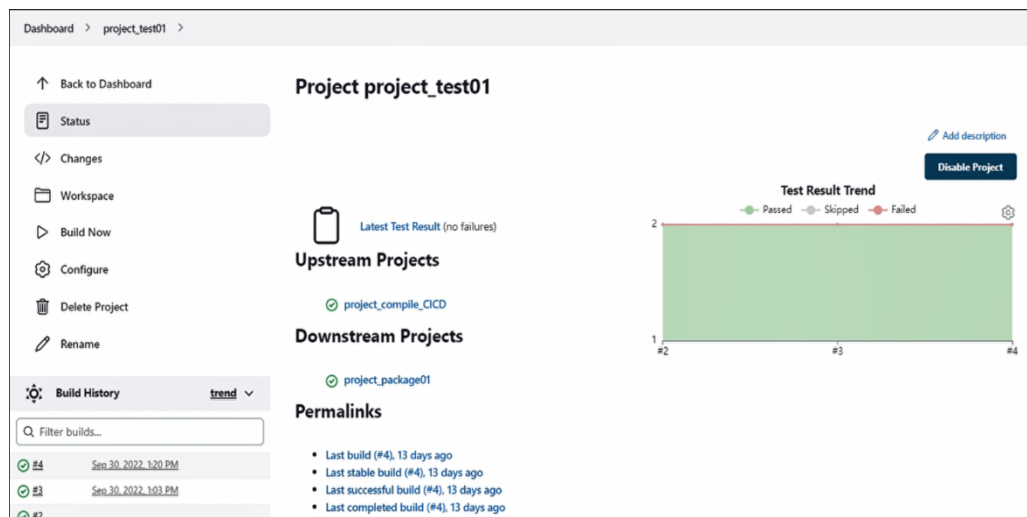
13. Keep the rest setting same and click on apply and save .
14. Now go the dashboard or the project page and build them . after building them all successfully , this should appear in status section .

## Project Compile -

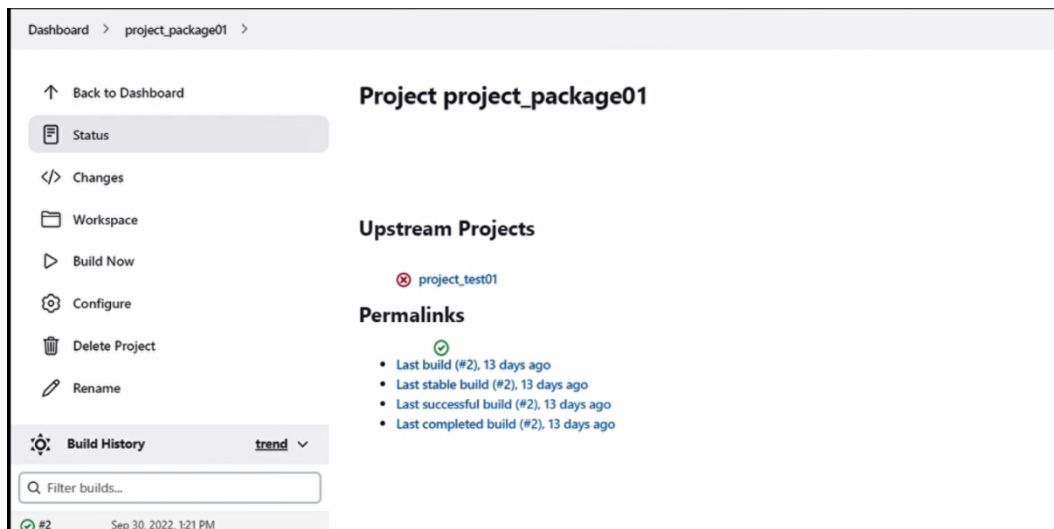


The screenshot shows the Jenkins Project page for 'project\_compile\_CICD'. The top navigation bar includes the Jenkins logo, a search bar, and notification icons. The breadcrumb trail shows 'Dashboard > project\_compile\_CICD >'. The left sidebar contains a list of actions: 'Back to Dashboard', 'Status' (selected), 'Changes', 'Workspace', 'Build Now', 'Configure', 'Delete Project', and 'Rename'. The main content area is titled 'Project project\_compile\_CICD' and contains two sections. The 'Downstream Projects' section shows a single project 'project\_test01' with a green status icon. The 'Permalinks' section lists four links: 'Last build (#3), 13 days ago', 'Last stable build (#3), 13 days ago', 'Last successful build (#3), 13 days ago', and 'Last completed build (#3), 13 days ago'. At the bottom, there is a 'Build History' section with a 'trend' dropdown and a search bar labeled 'Filter builds...'.

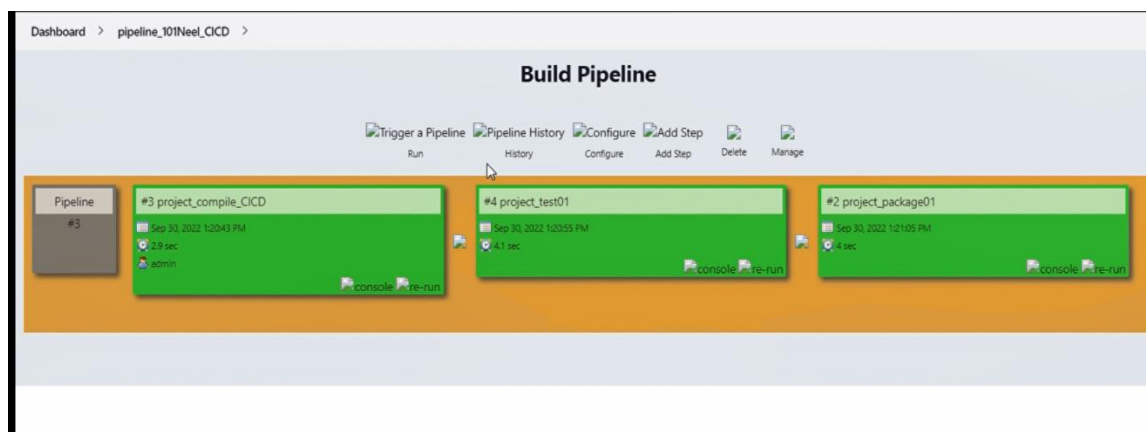
## Project test –



## Project Package –



15. So after all projects are successfully build go to your pipeline in the top views section and then reload the page until the pipeline is successfully created and try to analyse it and manage .



**Conclusion :** Therefore we studied pipelining in Jenkins , it's advantages to developers , successfully created a pipeline and imported pipeline from maven repository from git and analysed it.