

## Code testing (second job of SDLC)

After doing code compile job

Lets now create code testing

Click on new item

Dashboard > All >

Enter an item name

code-testing-akshu  
» Required field

**Freestyle project**  
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

**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.

<https://github.com/akshu20791/DevOpsClassCodes>

## 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/akshu20791/DevOpsClassCodes

Credentials ?

- none -

Click on build steps

## Configure

General

Source Code Management

Build Triggers

Build Environment

Build Steps

Post-build Actions

### Build Steps

Add build step

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

DOCUMENTATION

Maven Plugins

Maven Extensions

Index (category)

User Centre

Maven in 5 Minutes

Getting Started Guide

Naming Conventions

The Build Lifecycle

The POM

Profiles

Standard Directory Layout

Dependency Mechanism

Getting Help

Running Maven

## Build Lifecycle Basics

Maven is based around the central concept of a build lifecycle. What this means is that the process for building and distributing a particular artifact (project) is clearly defined.

For the person building a project, this means that it is only necessary to learn a small set of commands to build any Maven project, and the `POM` will ensure you get the results they desired.

There are three built-in build lifecycles: `default`, `clean`, and `site`. The `default` lifecycle handles your project deployment, the `clean` lifecycle handles project cleaning, while the `site` lifecycle handles the creation of your project's web site.

### A Build Lifecycle is Made Up of Phases

Each of these build lifecycles is defined by a different list of build phases, wherein a build phase represents a stage in the lifecycle.

For example, the default lifecycle comprises of the following phases (for a complete list of the lifecycle phases, refer to the [Lifecycle Reference](#)):

- `validate` - validate the project is correct and all necessary information is available
- `compile` - compile the source code of the project
- `test` - test the compiled source code using a suitable unit testing framework. These tests should not require the code be packaged or deployed
- `package` - take the compiled code and package it in its distributable format, such as a JAR.
- `verify` - run any checks on results of integration tests to ensure quality criteria are met
- `install` - install the package into the local repository, for use as a dependency in other projects locally
- `deploy` - done in the build environment, copies the final package to the remote repository for sharing with other developers and projects.

These lifecycle phases (plus the other lifecycle phases not shown here) are executed sequentially to complete the `default` lifecycle. Given the lifecycle above, this means that when the default lifecycle is used, Maven will first validate the project, then will try to compile the sources, run those against the test package the binaries (e.g. jar), run integration tests against that package, verify the integration tests, install the verified package to the local repository, then deploy the installed package to a remote repository.

Dashboard > code-testing-akshat > Configuration

Configure

General

Source Code Management

Build Triggers

Build Environment

Build Steps

Post-build Actions

Build Steps

Invoke top-level Maven targets ?

Goals

test

Advanced

Add build step

Save

Apply

Add timestamps to the Console Output

☐ Inspect build log for published build scans


☐ Terminate a build if it's stuck

☐ With Ant ?

 Status

 Changes

 Workspace

 Build Now


 Configure

 Delete Project

 Rename

## Project code-testing-akshat

### Permalinks

 Build History trend ▾

##

 Status

 Changes

 Workspace

 Build Now

 Configure


 Delete Project


 Rename



## Project code-testing-akshat

### Permalinks

- [Last build \(#1\), 46 sec ago](#)

 Build History trend ▾

Q Filter builds... 

 #1 Jun 3, 2023, 5:03 PM 

Build would be successful