Caltech | Center for Technology & Management Education

Post Graduate Program in DevOps

simplilearn

DevOps

Caltech | Center for Technology & Management Education

CI/CD Pipeline with Jenkins

simplilearn

Source Control Management, Build Tools, and Test Reports

Learning Objectives

By the end of this lesson, you will be able to:

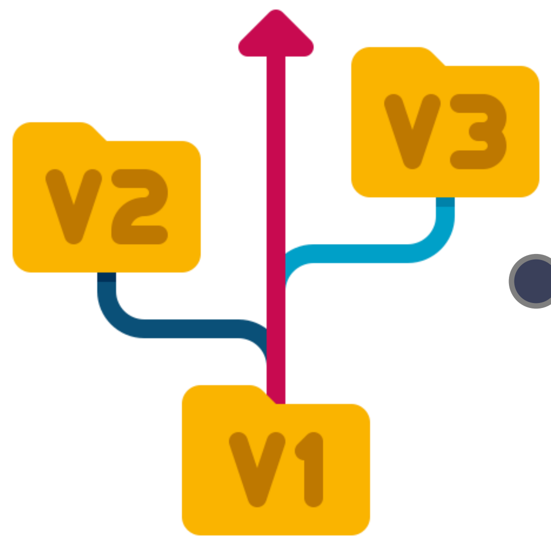
- List the benefits of version control system
- Discuss how to set up Git configurations in Jenkins job
- Outline Build Automation using Jenkins
- Configure unit test cases
- Discuss how to generate test reports in Jenkins

Version Control System

Introduction to Version Control System

A version control system allows users to track changes in software development projects.

It enables developers to collaborate within the team

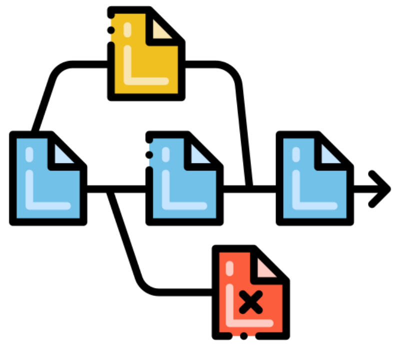


It helps developers to distribute tasks through branches

Version Control System

Introduction to Version Control System

Based on the number of collaborators, there can be several branches in the version control system.



Branches maintain code individually with every change in a specified branch.

They enable developers to combine the changes in the code whenever required.

Introduction to Version Control System

With the help of branches, developers can:

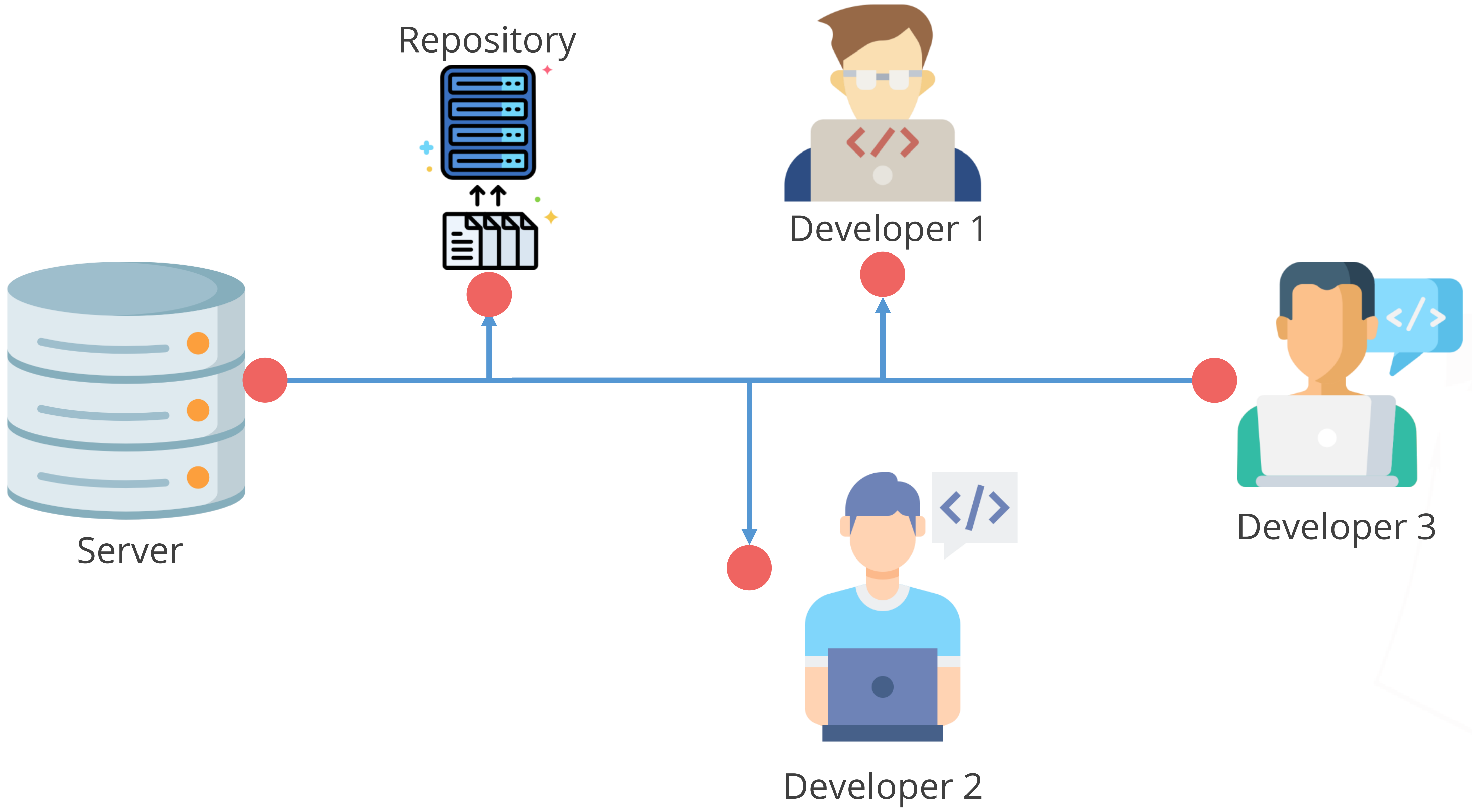
View change history

Navigate to required versions

Manage code versions

Introduction to Version Control System

Each modification made to a file in version control system creates a new version of the file in the server or repository.



Benefits of Version Control System

The benefits of version control system are:

1. Helps developers to work in isolation
2. Creates separate versions for each modification
3. Acts like a backup if the source code is accidently lost
4. Tracks and fetches the version of the source code easily
5. Integrates all the changes of the source code into a single repository

Version Control Systems Tools

Version control system includes:

- Subversion
- TFS
- Git
- Mercurial
- CVS

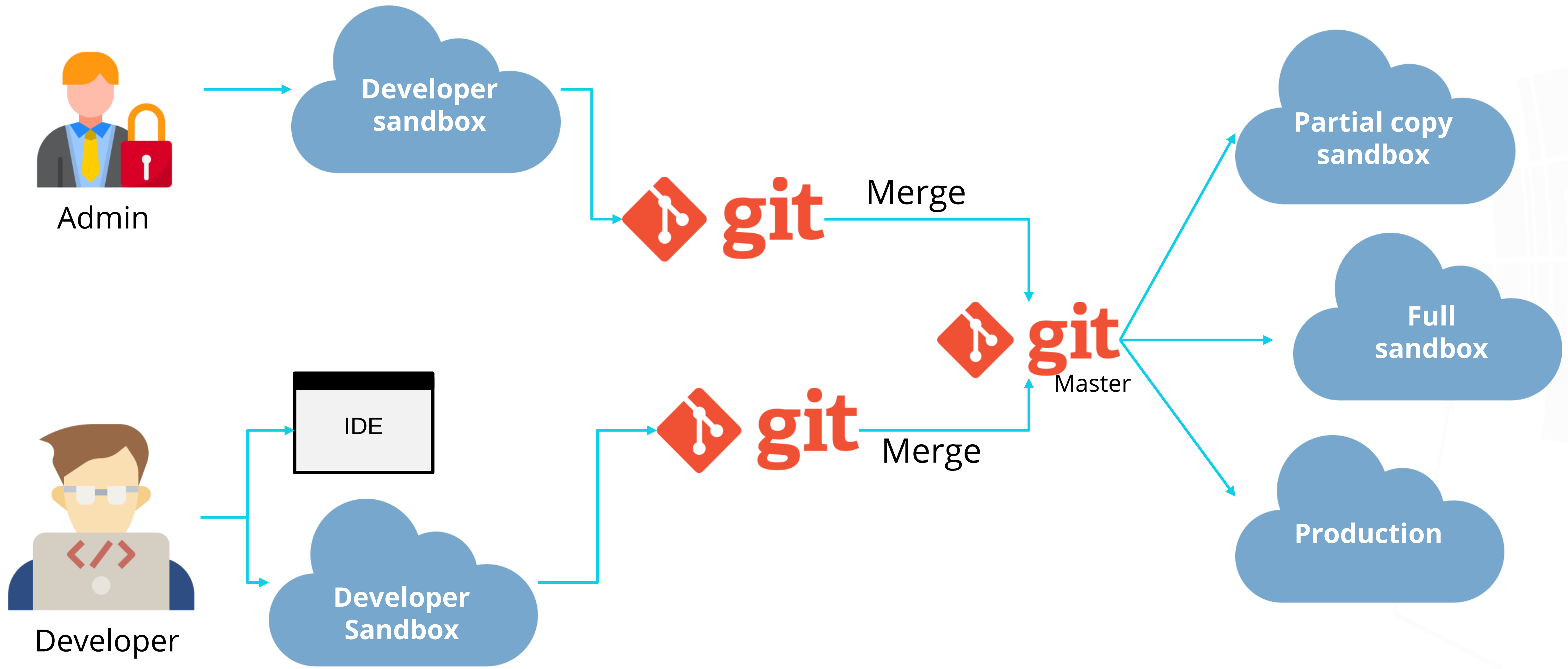
Caltech | Center for Technology & Management Education

simplilearn

Build Tools Integration

Introduction to Git

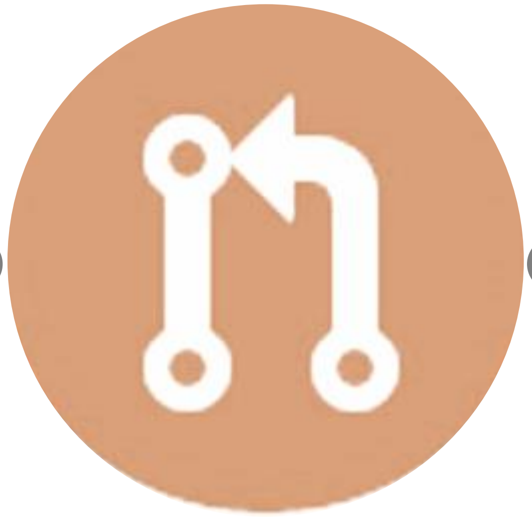
Git is the most widely used version control system. It is considered the modern standard for software development.



Introduction to Git

Git is a distributed version control system used to track changes in source code during software development.

Allows teams to collaborate on Git branches using Pull requests



Reviews each other's code efficiently using Pull requests

Pull Request

Git

Benefits

- Helps collaborate with programmers
- Tracks changes in any set of files

Goals

- Speed
- Data integrity
- Distributed workflows
- Non-linear workflows

Git Terminologies

Some of the terminologies used in GIT are:

Master: The default branch in case of Git

Clone: Used to clone remote repository as local repository in developer's machine

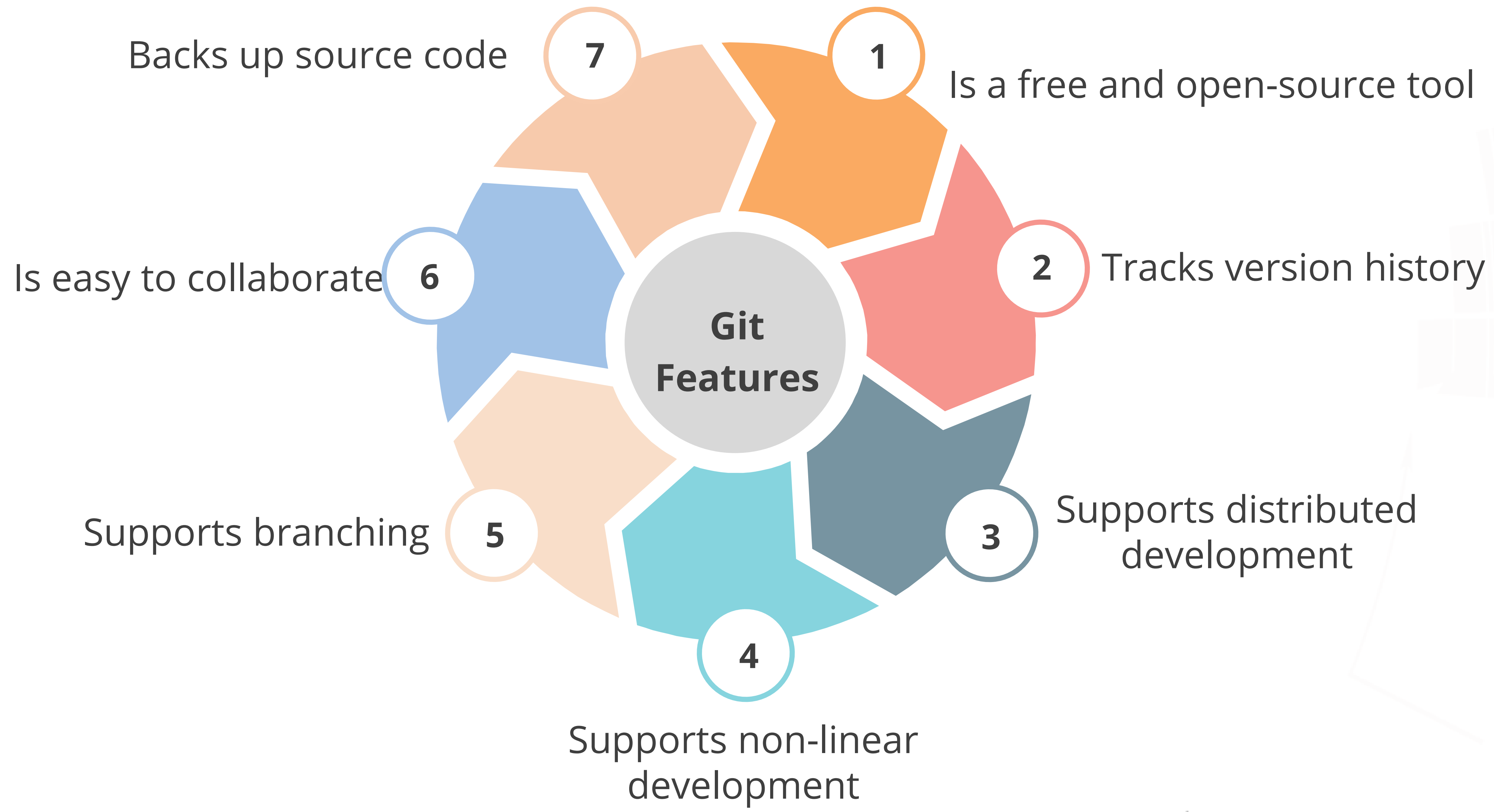
Commit: Command used to save commits to Git local repository

Push and Pull: Commands used to push and pull changes from and to remote repository respectively

HEAD: Refers to the latest commit in Git repository

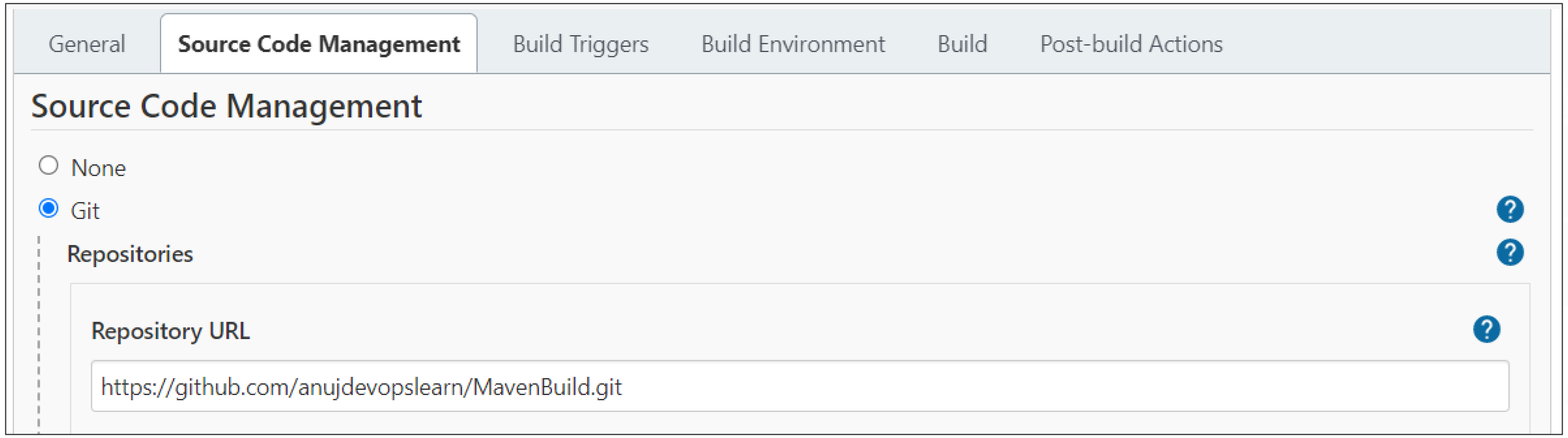
Features of Git

Some of the features of Git are:



Jenkins Job: Setting Up Git Configuration

Shown below is a screenshot of the Source Code Management tab. Developers use this tab to provide Repository configuration and check out the source code before executing Build process.



Source Code Management

None [✓ Git]

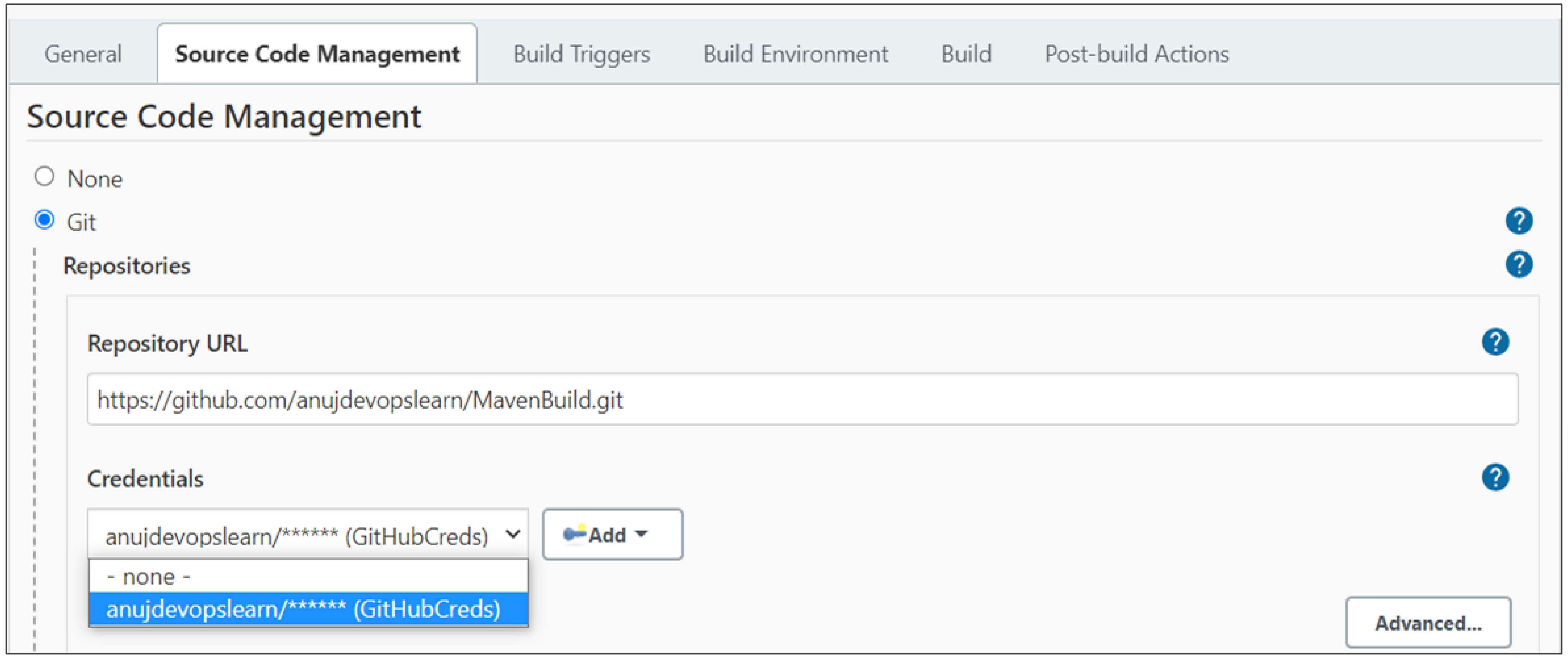
Repositories

Repository URL

https://github.com/anujdevopslearn/MavenBuild.git

Jenkins Job: Setting Up Git Configuration

Shown below is a screenshot of the Source Code Management tab. This tab allows the developer to configure credentials while configuring Private Git repositories.



Source Code Management

None [✓ Git]

Repositories

Repository URL

https://github.com/anujdevopslearn/MavenBuild.git

Credentials

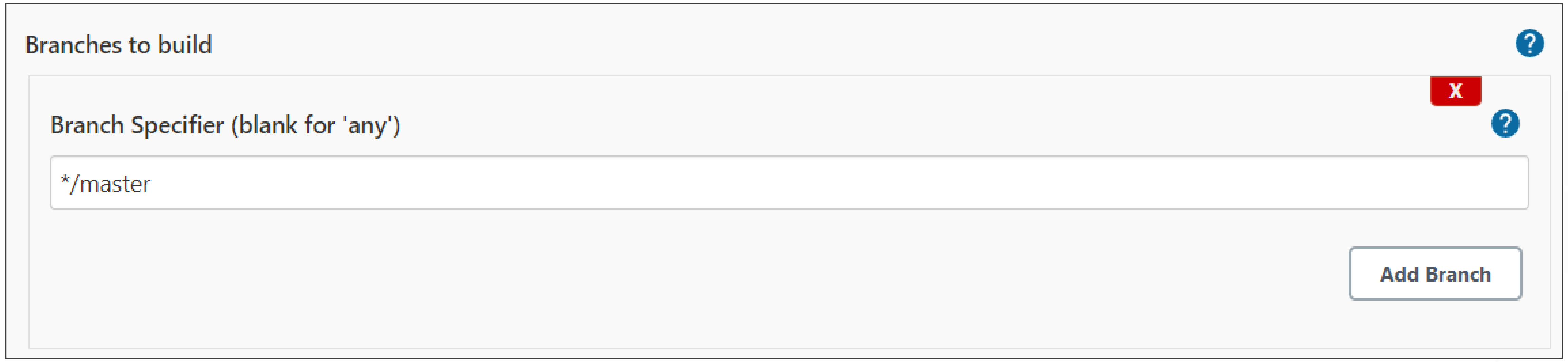
anujdevopslearn/****** (GitHubCreds)

- none -

anujdevopslearn/****** (GitHubCreds)

Jenkins Job: Setting Up Git Configuration

Shown below is a screenshot of the Branches to build screen. This is used to configure a branch and to check out the source code while running Build in Jenkins.



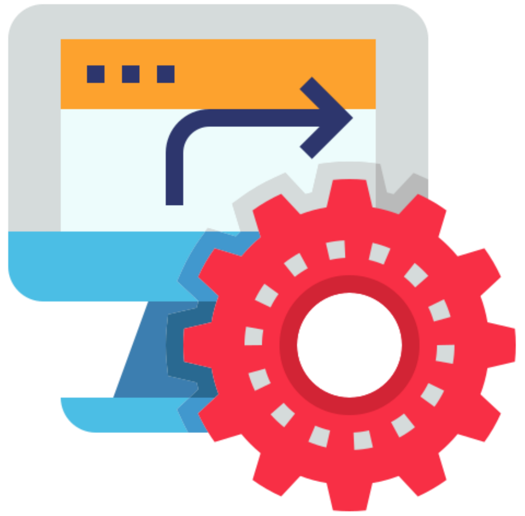
Branches to build

Branch Specifier (blank for 'any')

*/master

Build Automation

Build Automation is a process of automating preparation of executables by compiling the application source code.



- Is the first step implemented while setting up Pipeline
- Ensures fast feedback, which will help developers to fix bugs in the early stages
- Standardizes Build and eliminates human errors and bugs

Build Automation

Build is a process of compiling and packaging code into a executable binaries.

Small Projects

- Developers perform all Build activities

Large Projects

- Developers need Build scripts for Build automation
- Developers can store Build scripts with source code inside version control system

Build Automation Tools

Build Tools like Ant, Maven, Gradle, Rake, and Nant can be used to perform Build Automation and IDE integration.

Gradle

Rake

Apache Ant

Maven

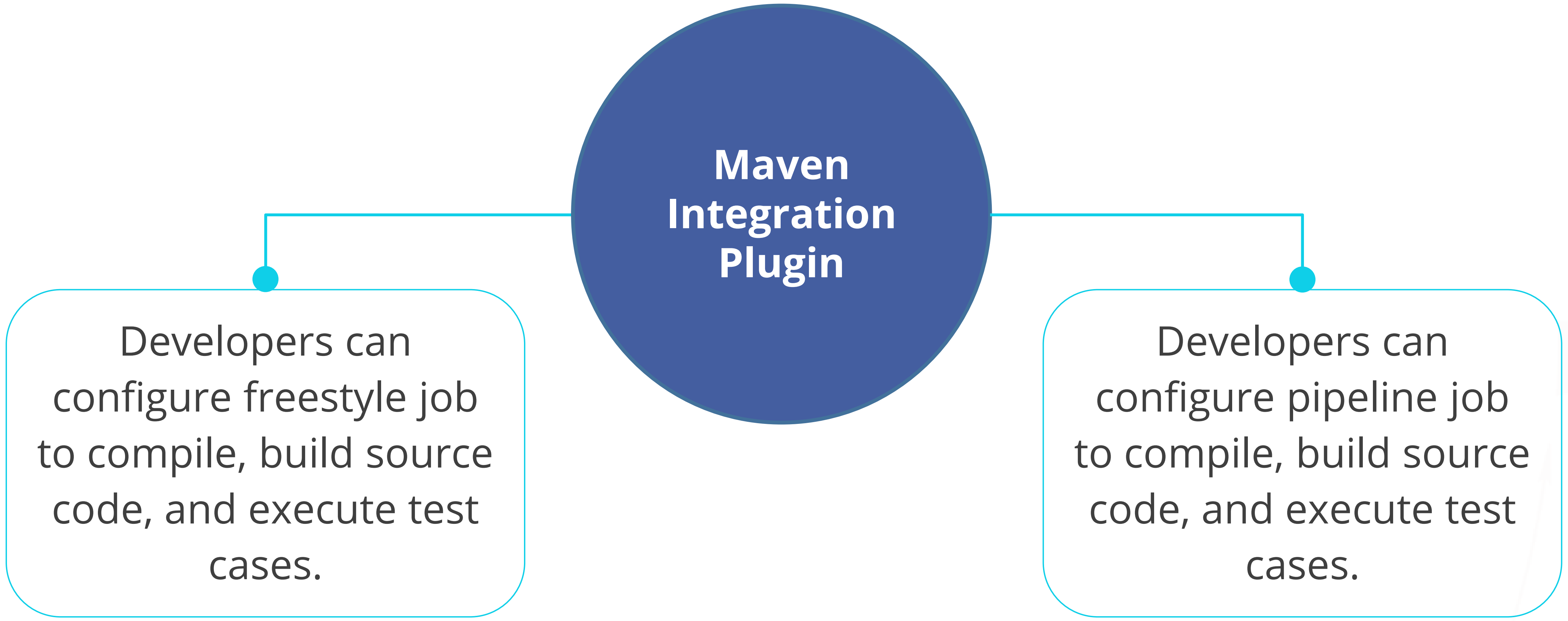
Nant

Note

Build tools are scripts that automate preparation of executable applications from source code.

Maven Integration with Jenkins

Maven Integration plugin is supported by Jenkins to configure and build maven based projects.



Maven Integration Plugin

Developers can configure freestyle job to compile, build source code, and execute test cases.

Developers can configure pipeline job to compile, build source code, and execute test cases.

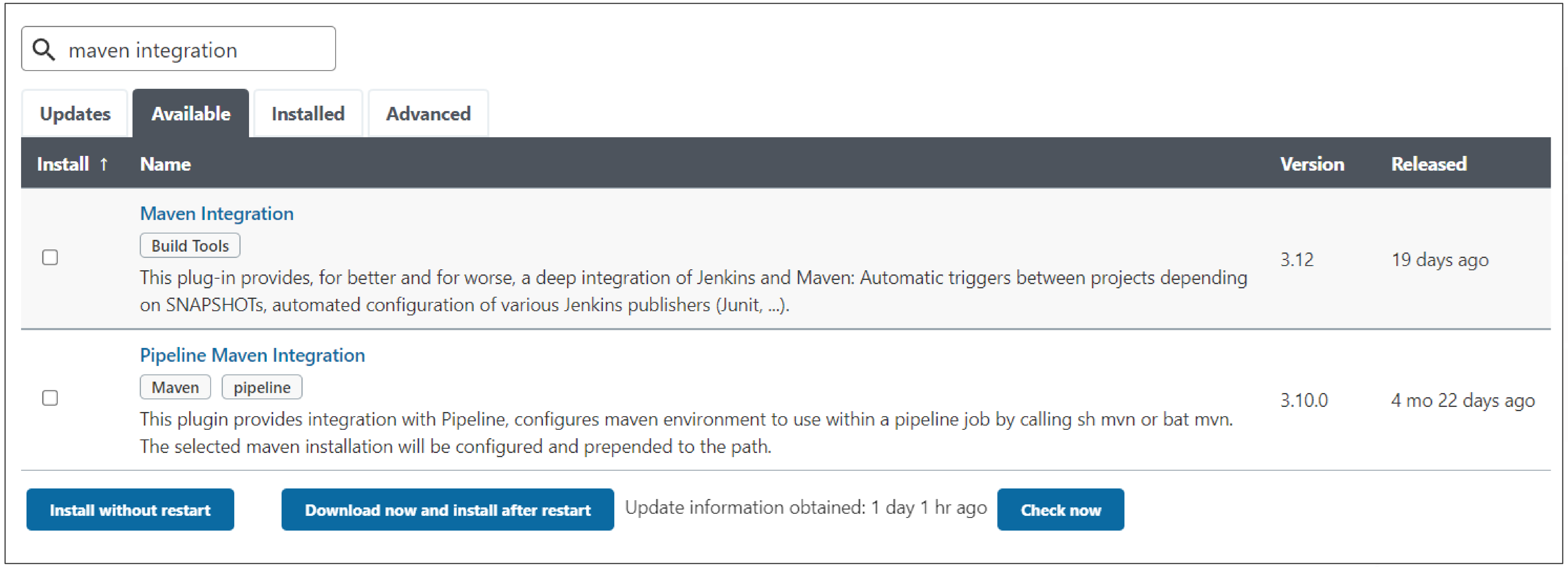
Maven Installation with Jenkins

Follow below steps to install Maven plugin on Jenkins:

Click Manage Jenkins and click on Manage plugins → Under plugin manager access Available tab, search for Maven integration → Select the required plugin and click Download → Restart the machine → Click the checkbox to restart Jenkins

Maven Integration with Jenkins

Shown below is the Available tab that helps to install Maven plugin.

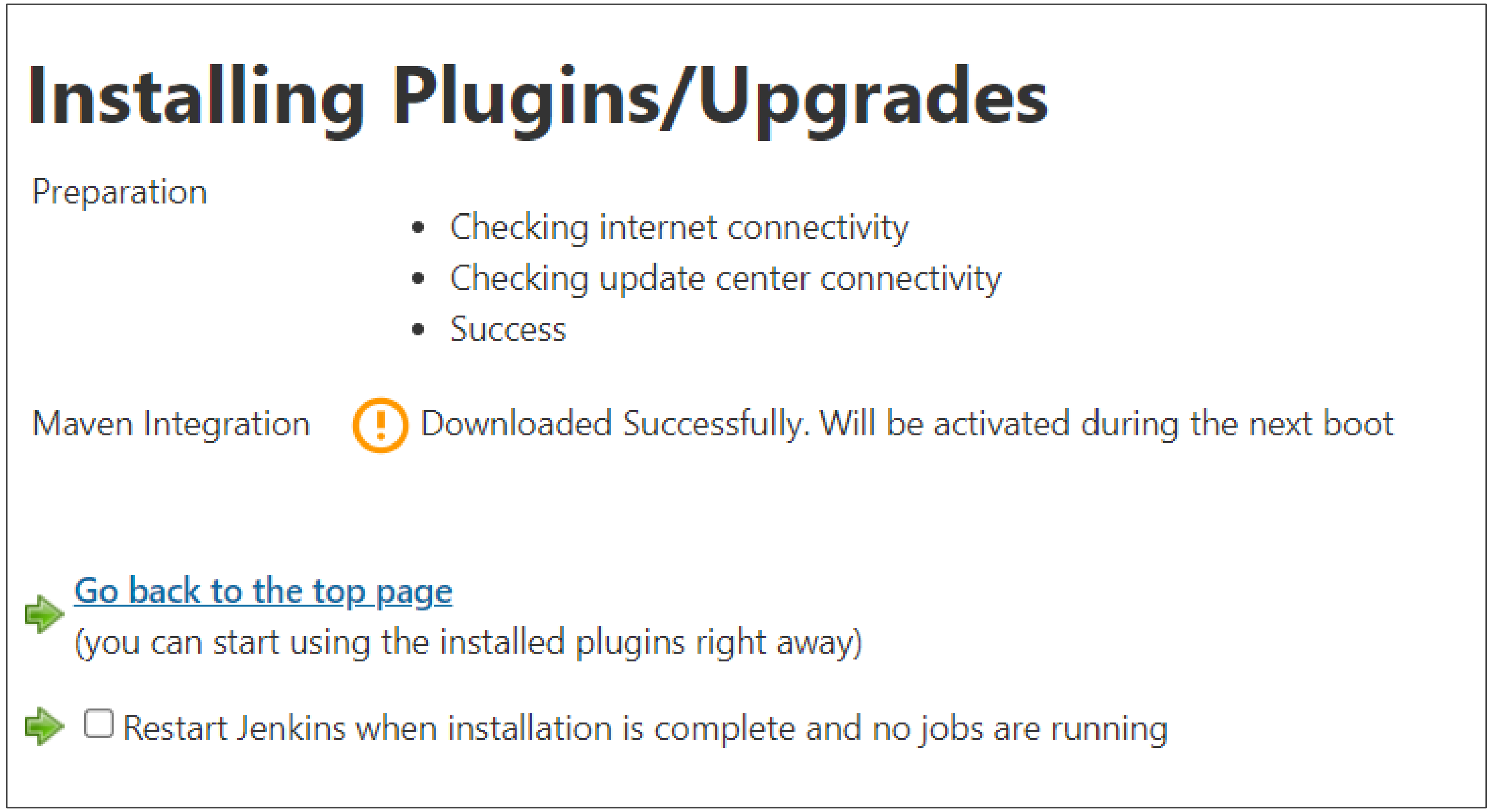


Install ↑	Name	Version	Released
☐	Maven Integration Build Tools This plug-in provides, for better and for worse, a deep integration of Jenkins and Maven: Automatic triggers between projects depending on SNAPSHOTs, automated configuration of various Jenkins publishers (Junit, ...).	3.12	19 days ago
☐	Pipeline Maven Integration Maven pipeline This plugin provides integration with Pipeline, configures maven environment to use within a pipeline job by calling sh mvn or bat mvn. The selected maven installation will be configured and prepended to the path.	3.10.0	4 mo 22 days ago

Update information obtained: 1 day 1 hr ago

Maven Integration with Jenkins

Shown below is the status of plugins installation.



Installing Plugins/Upgrades

Preparation

- Checking internet connectivity
- Checking update center connectivity
- Success

Maven Integration ! Downloaded Successfully. Will be activated during the next boot

Go back to the top page
(you can start using the installed plugins right away)

☐ Restart Jenkins when installation is complete and no jobs are running

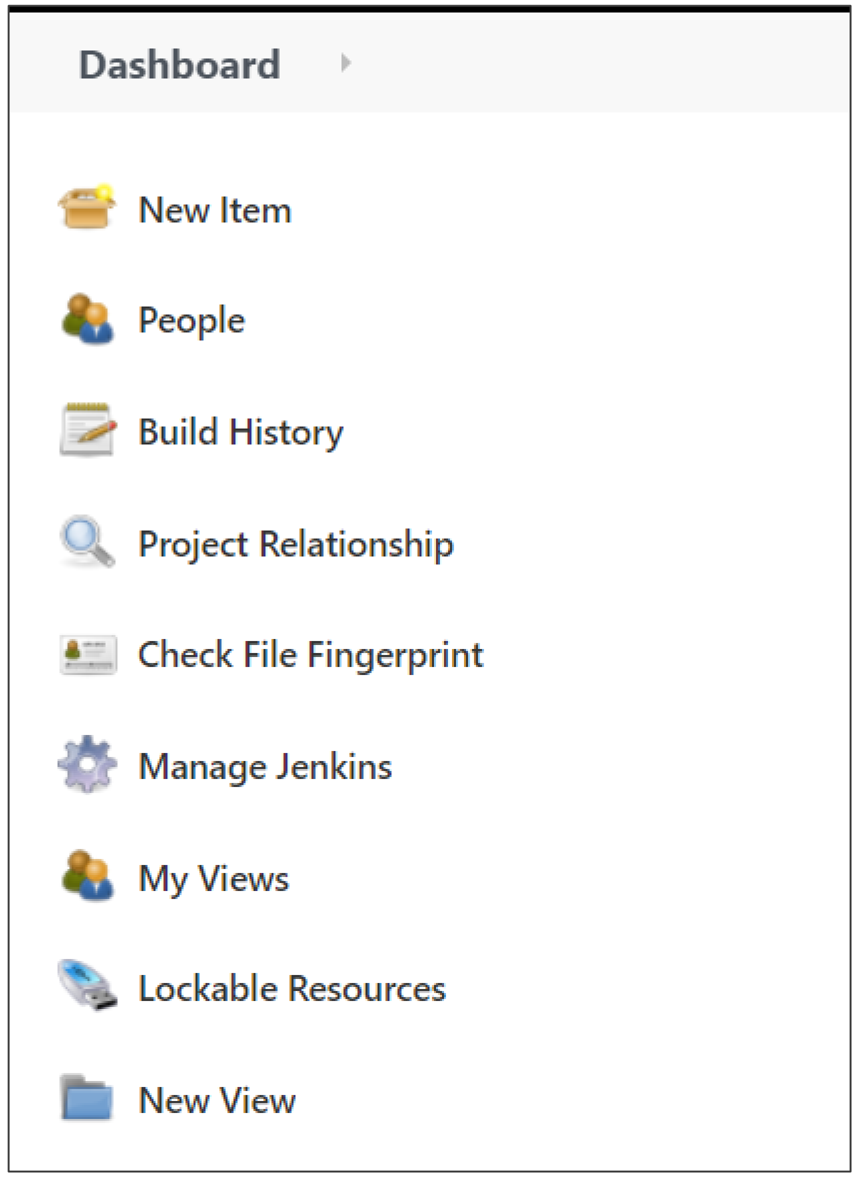
Maven Integration with Jenkins

For integrating Maven with Jenkins follow the below steps:

Click New Item and then provide Job name → Configure Git repository details to checkout source code → Configure Maven tool in Global Tool configuration → Click Add Maven → Enter tool name and select Install automatically to add Maven installation in Jenkins.

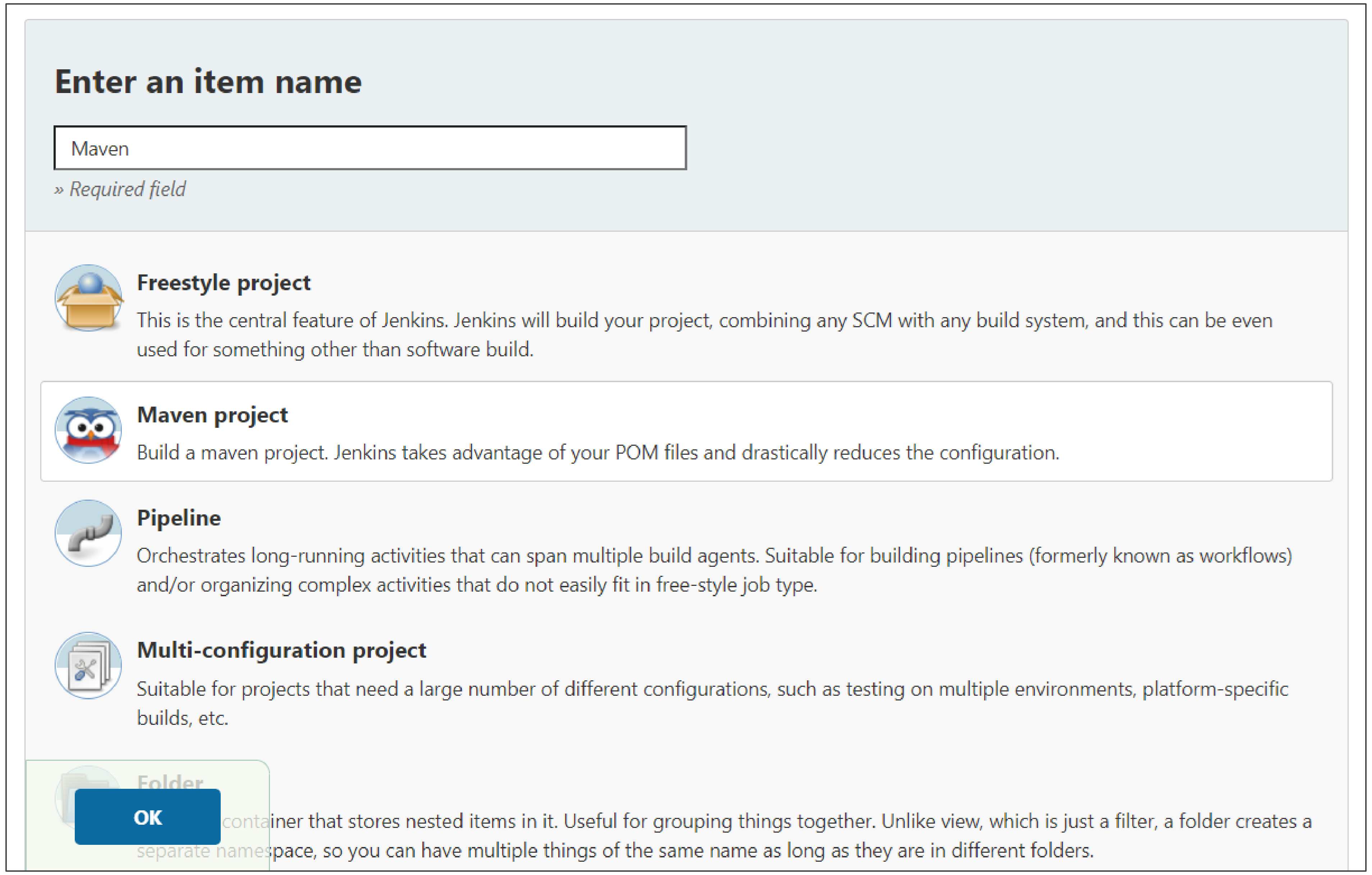
Maven Integration with Jenkins

Shown below is the **Dashboard** section that allows the developer to create new item.



Maven Integration with Jenkins

Shown below is the **Enter an item name** section that allows the developers to enter the item.



Enter an item name

Maven

» *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.

Maven project
Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.

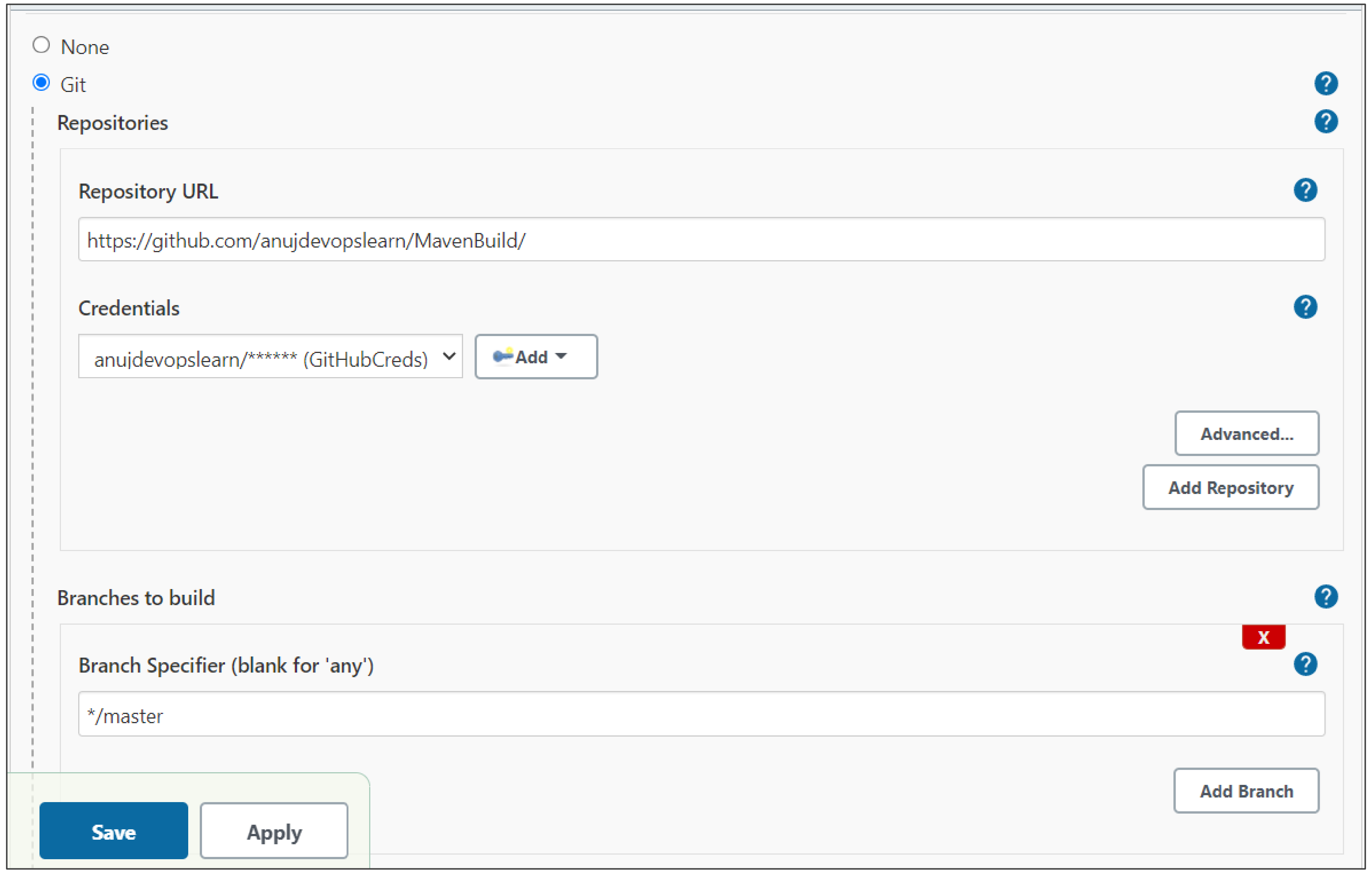
Pipeline
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

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
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.

Maven Integration with Jenkins

Shown below is the **Repositories** section that allows the developers to add the repository URL and credentials.



None

[✓ Git]

Repositories

Repository URL: https://github.com/anujdevopslearn/MavenBuild/

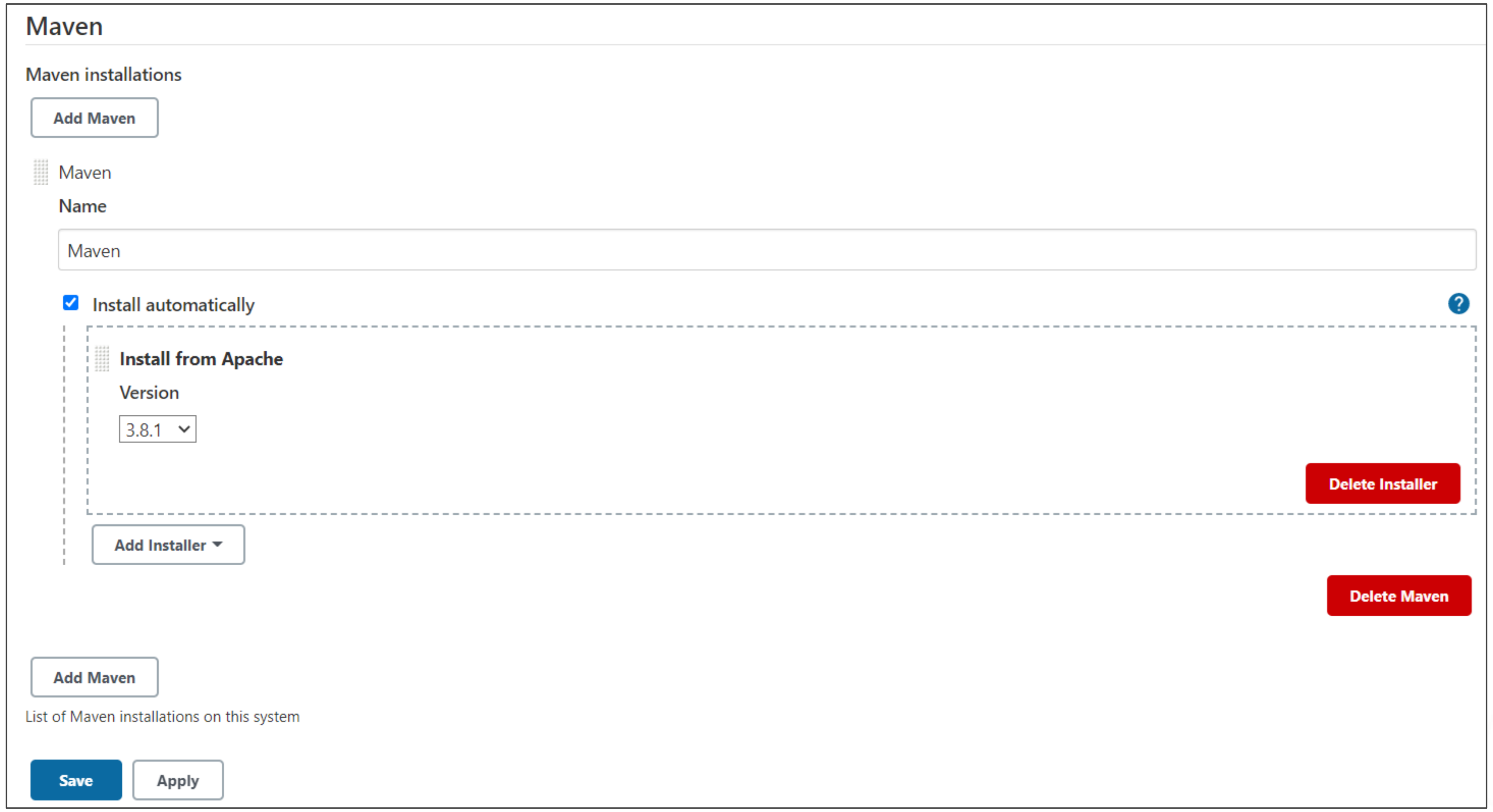
Credentials: anujdevopslearn/****** (GitHubCreds)

Branches to build

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

Maven Integration with Jenkins

Shown below is the **Maven** section that allows the developers to click Install automatically checkbox.



Maven

Maven installations

Add Maven

Maven

Name

Maven

☑ Install automatically

Install from Apache

Version

3.8.1

Delete Installer

Add Installer

Delete Maven

Add Maven

List of Maven installations on this system

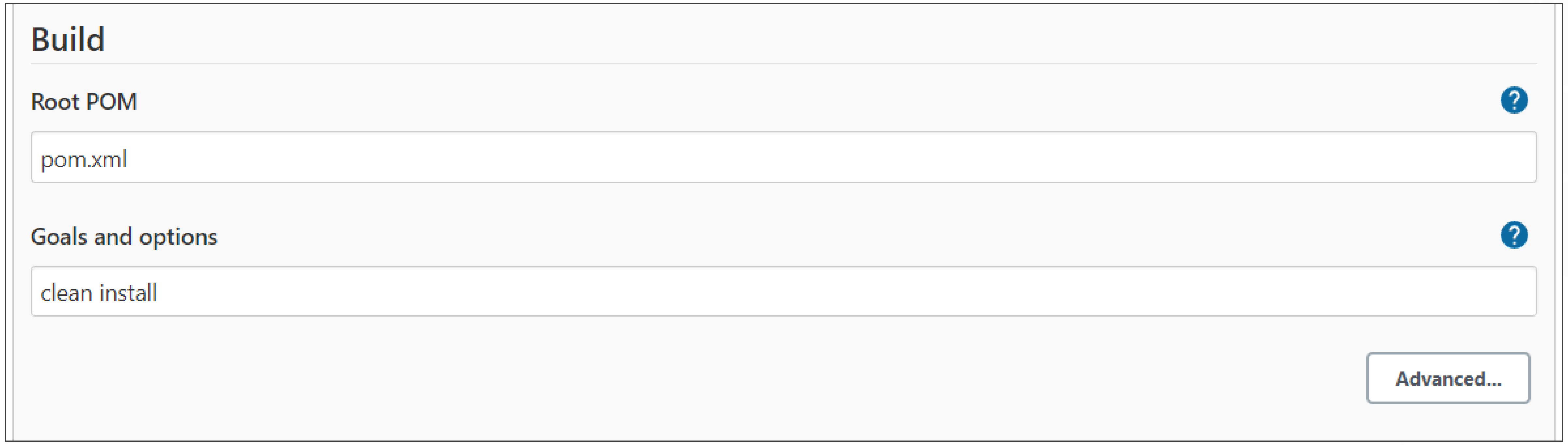
Save Apply

Maven Integration with Jenkins

Once Maven Integration plugin is configured in Jenkins, start configuring Maven build scripts with Jenkins job.

1. Provide pom.xml relative path in **Root POM**

2. Provide Goal details as "clean install" in **Goals and options**



Build

Root POM: pom.xml

Goals and options: clean install

Test Executions

Software testing is important to understand whether a software application is capable of fulfilling all business requirements or not.

Traditional Approach

- Depends on the QA professional for testing
- Performs manual testing for software
- Is time-consuming
- Is expensive

Modern Approach

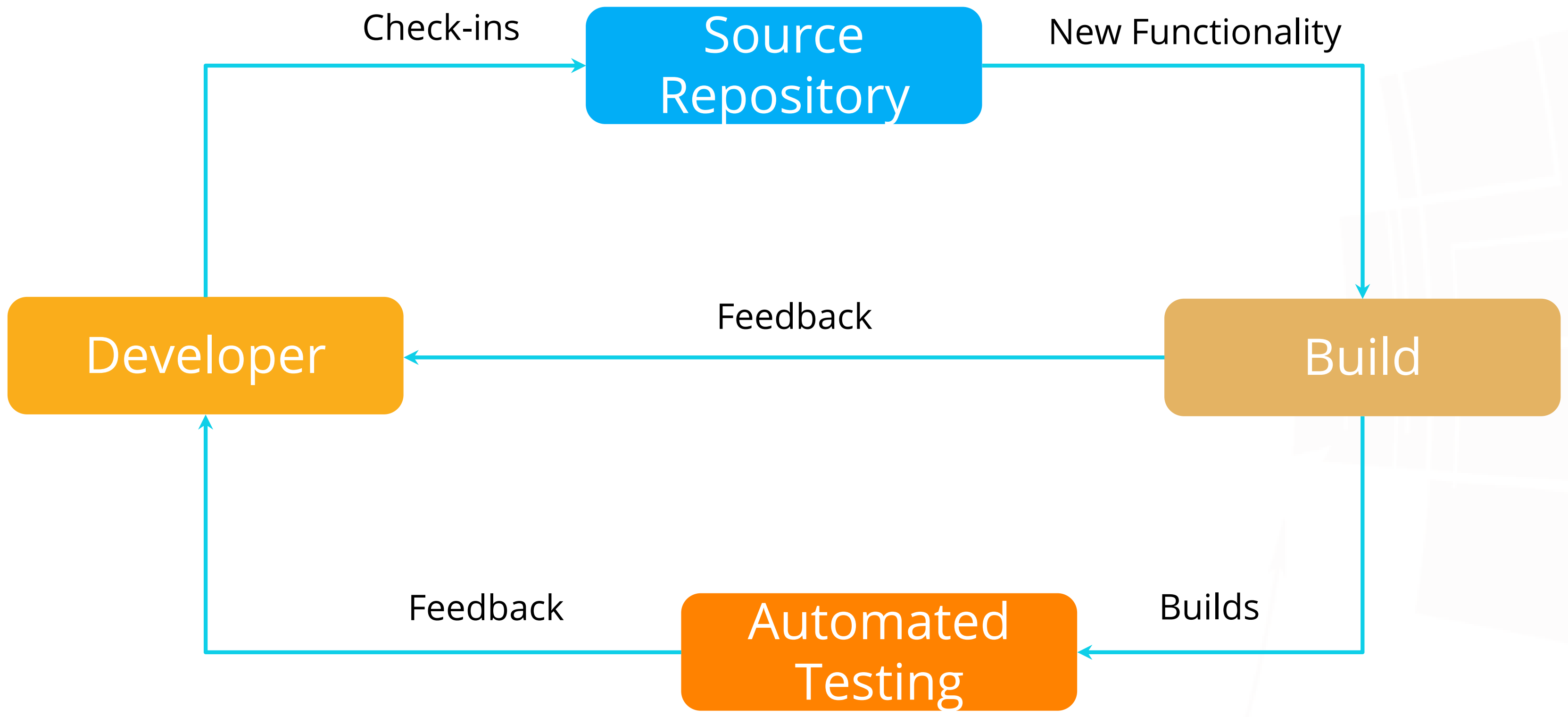
- Does not depend on the QA professional for testing
- Performs automated testing for software
- Is easy to perform
- Is cheap and economic

Test Executions

Shown below is the flow diagram for automated testing:

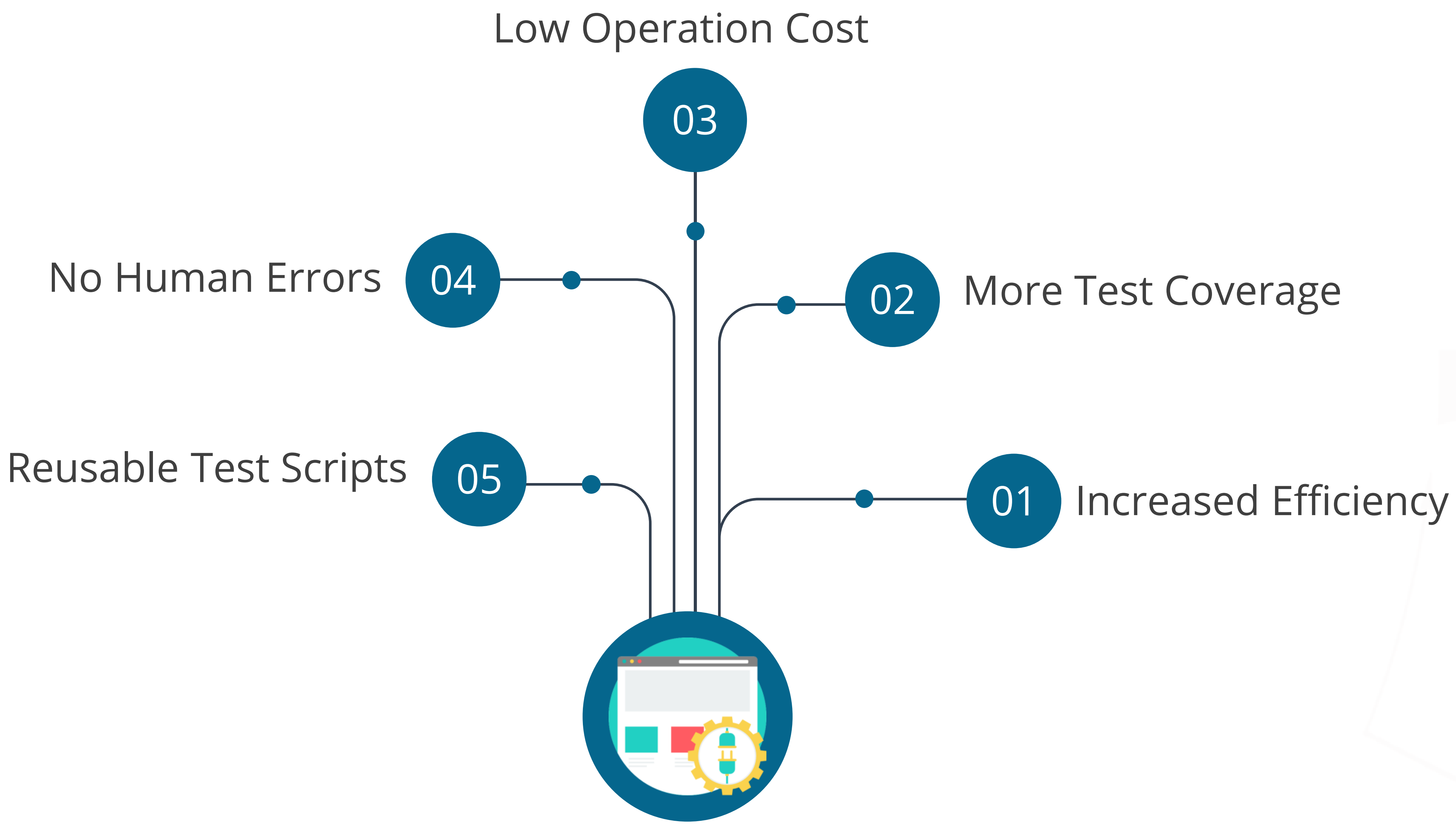
Automate application backend builds and create complex rule-based processes, actions, and workflows.

Developers can schedule triggers, tasks, and updates using this build tool.



Benefits of Test Executions

The benefits of automated test executions are:



Jenkins Test Reporting

In Jenkins, a report is a graphical representation that is used to visualize test results and outputs.

Used to communicate test results to team members

Used to communicate test results to stakeholders

Generated from Jenkins server XML files

Jenkins Test Reporting

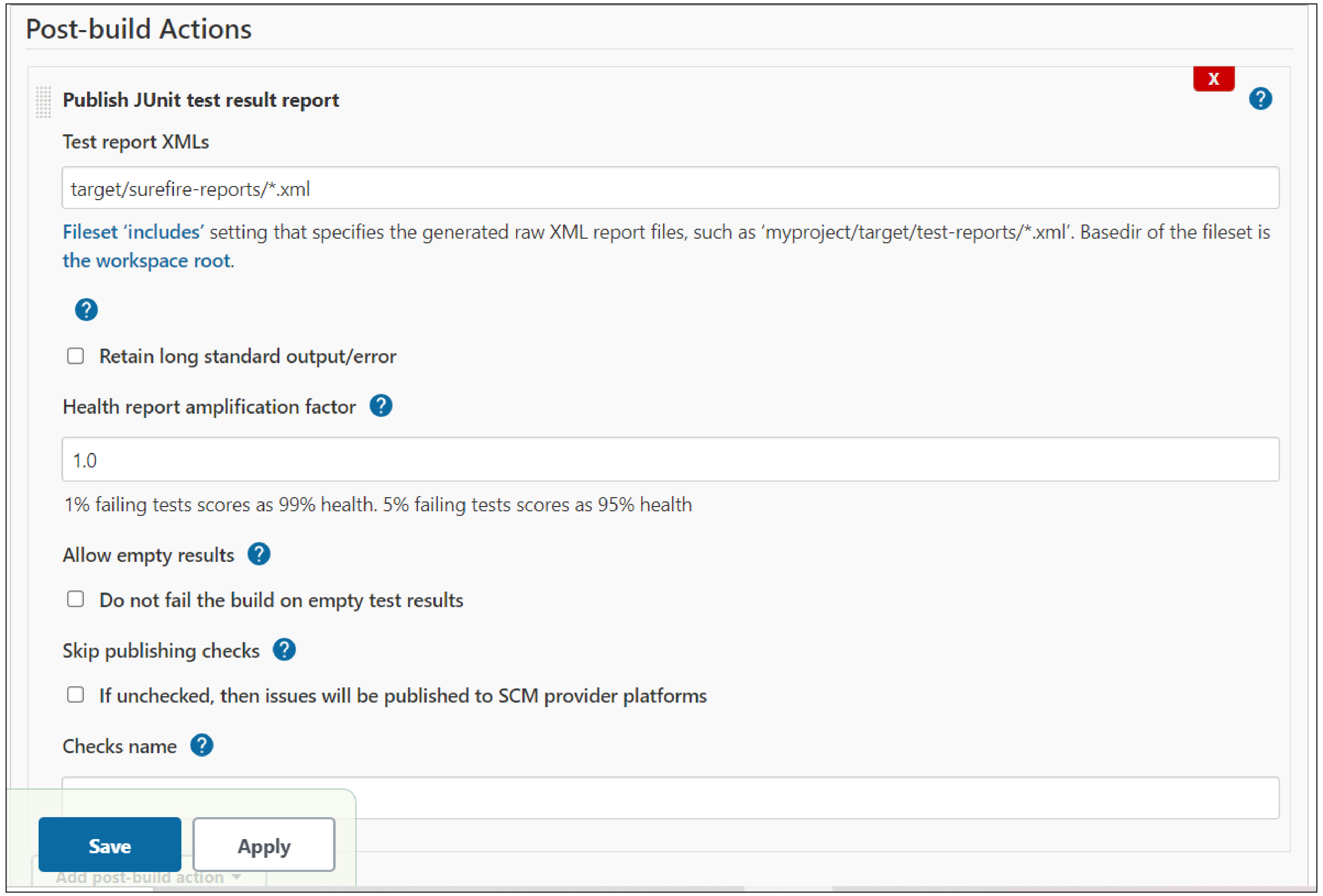
Developers can define the reports that are to be created during the post-Build action for any job.

How to define the reports that are to be created?

1. Navigate to Job Configure
2. Scroll down and click on the Post-build action button
3. Select Publish JUnit test result report option

Jenkins Test Reporting

Shown below is the Post-build Actions screen:



Post-build Actions

Publish JUnit test result report

Test report XMLs

target/surefire-reports/*.xml

Fileset 'includes' setting that specifies the generated raw XML report files, such as 'myproject/target/test-reports/*.xml'. Basedir of the fileset is the workspace root.

☐ 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

Skip publishing checks

☐ If unchecked, then issues will be published to SCM provider platforms

Checks name

Jenkins Test Reporting

To generate a test report in Jenkins, follow the steps shown below.

Step 1

- Once job is saved, click **Build Now** to build the project and execute unit test cases.

Step 2

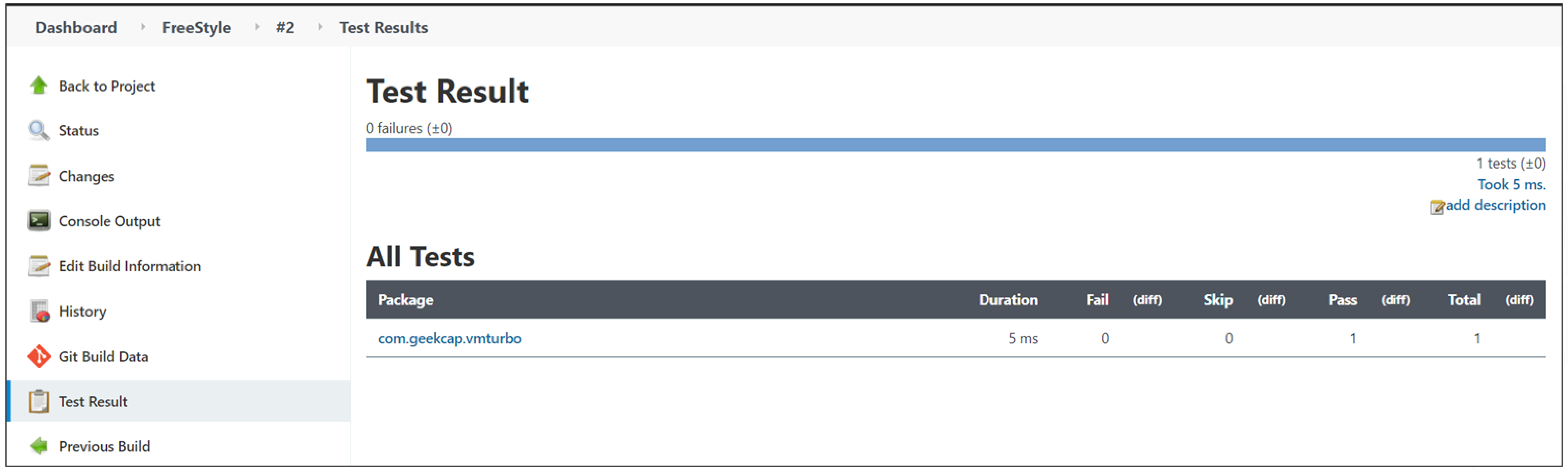
- In the **Build history** section, select the Build and open the Build results by clicking on Test Result.

Step 3

- In the **Test Result** option, the test results are generated in a simple and graphical format.

Jenkins Test Reporting

Shown below is the **Test Result** screen displaying the generated reports.



Test Result

0 failures (±0)

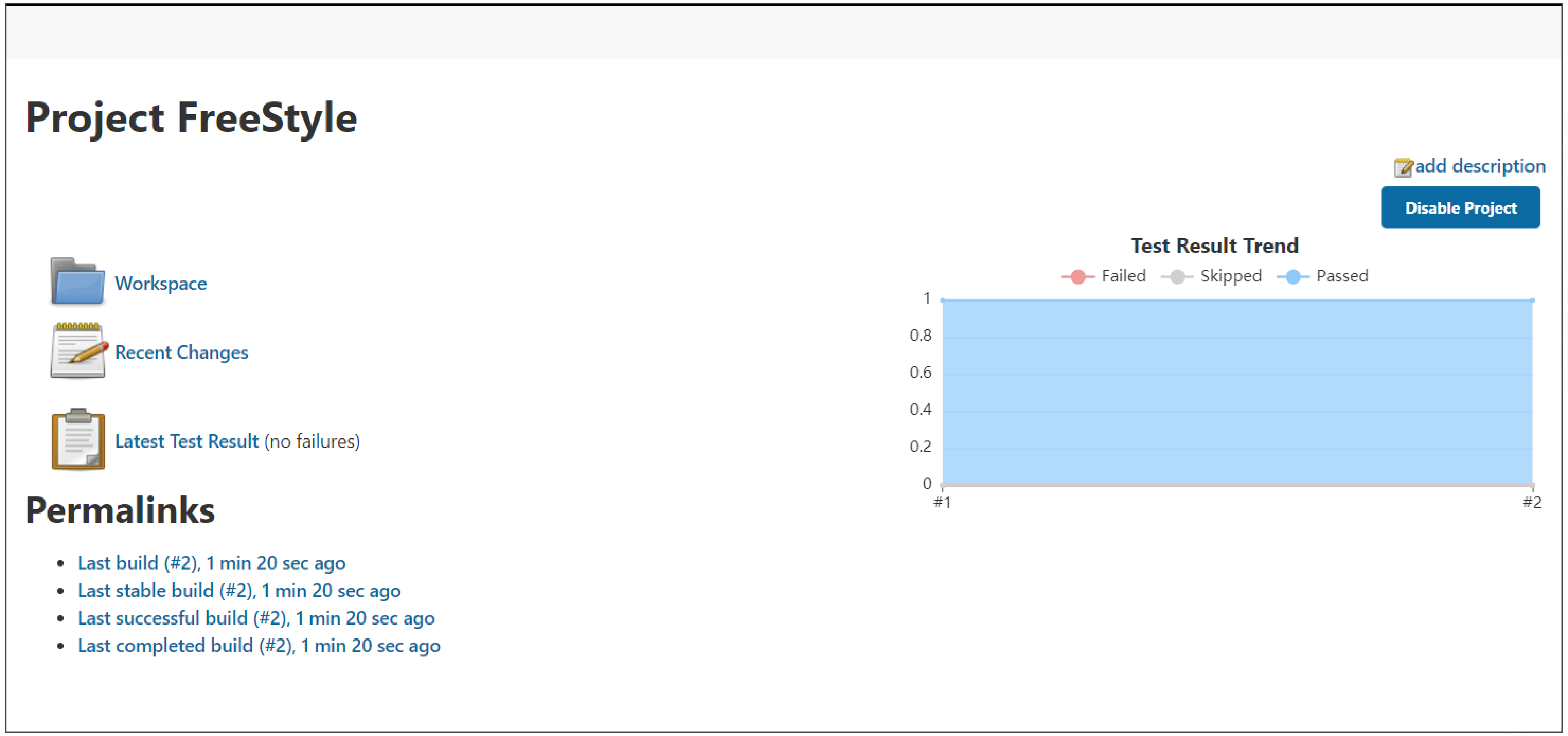
1 tests (±0)
Took 5 ms.
add description

All Tests

Package	Duration	Fail	(diff)	Skip	(diff)	Pass	(diff)	Total	(diff)
com.geekcap.vmturbo	5 ms	0		0		1		1	

Jenkins Test Reporting

Shown below is the generated report of the test result trend:



Project FreeStyle

- Workspace
- Recent Changes
- Latest Test Result (no failures)

Permalinks

- Last build (#2), 1 min 20 sec ago
- Last stable build (#2), 1 min 20 sec ago
- Last successful build (#2), 1 min 20 sec ago
- Last completed build (#2), 1 min 20 sec ago

Assisted Practice

Setting up Maven Build Job in Jenkins

Duration: 10 min

Problem Statement:

1. Set up a Maven Build job in Jenkins.

Assisted Practice: Guidelines

Steps to demonstrate how to set up a Maven Build job in Jenkins:

1. Log in to Jenkins CI tool and configure freestyle job to run maven Build

Assisted Practice

Publishing Test Case Reports in Jenkins

Duration: 15 min

Problem Statement:

1. Demonstrate how test case reports can be published in Jenkins.

Assisted Practice: Guidelines

Steps to demonstrate how to publish test cases reports in Jenkins:

1. Log in to Jenkins CI tool and configure freestyle job to run maven Build and Unit Test cases.

Key Takeaways

- A version control system allows users to track changes in software development projects.
- Git is a distributed version control system used to track changes in source code during software development.
- Build is a process of compiling and packaging code into executable binaries.
- Software testing is important to understand whether a software application is capable of fulfilling all business requirements or not.
- In Jenkins, a report is a graphical representation that is used to visualize test results and outputs.

Lesson-End Project

Maven Tool Configuration

Problem Statement:

Perform the following:

- Configure Maven tool
- Send Junit test case reports using email notification

Access: Click on the **Labs** tab on the left side panel of the LMS. Copy or note the username and password that is generated. Click on the **Launch Lab** button. On the page that appears, enter the username and password in the respective fields, and click **Login**.

Thank You