



Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

Aim: To understand Continuous Integration, install and configure Jenkins with Maven.

Objective: The objective of understanding Continuous Integration (CI) involves grasping the principles and benefits of automating the process of integrating code changes from multiple developers into a shared repository.

Theory:

Continuous integration is a DevOps software development practice where developers regularly merge their code changes into a central repository, after which automated builds and tests are run. Continuous integration most often refers to the build or integration stage of the software release process and entails both an automation component (e.g. a CI or build service) and a cultural component (e.g. learning to integrate frequently). The key goals of continuous integration are to find and address bugs quicker, improve software quality, and reduce the time it takes to validate and release new software updates.

Why is Continuous Integration Needed?

In the past, developers on a team might work in isolation for an extended period of time and only merge their changes to the master branch once their work was completed. This made merging code changes difficult and time-consuming, and also resulted in bugs accumulating for a long time without correction. These factors made it harder to deliver updates to customers quickly.

Benefits of Continuous Integration:

- 1. Improve Developer Productivity:** Continuous integration helps your team be more productive by freeing developers from manual tasks and encouraging behaviors that help reduce the number of errors and bugs released to customers.
- 2. Find and Address Bugs Quicker:** With more frequent testing, your team can discover and address bugs earlier before they grow into larger problems later.
- 3. Deliver Updates Faster:** Continuous integration helps your team deliver updates to their customers faster and more frequently.

How does Continuous Integration Work?

With continuous integration, developers frequently commit to a shared repository using a version control system such as Git. Prior to each commit, developers may choose to run local unit tests on their code as an extra verification layer before integrating. A continuous integration service automatically builds and runs unit tests on the new code changes to immediately surface any



Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

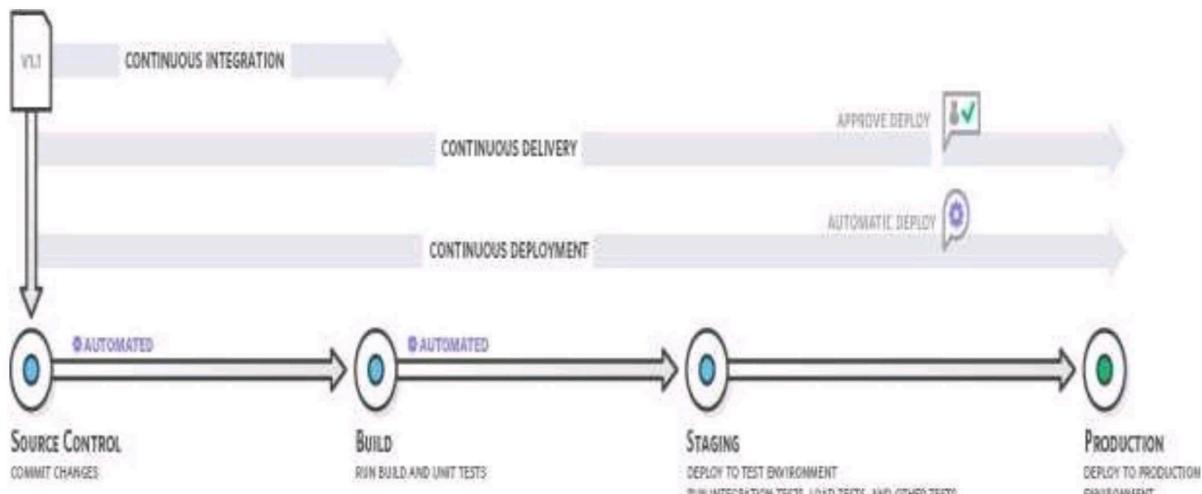


Fig. 4.1 Working of Continuous Integration

Continuous integration refers to the build and unit testing stages of the software release process. Every revision that is committed triggers an automated build and test.

With continuous delivery, code changes are automatically built, tested, and prepared for a release to production. Continuous delivery expands upon continuous integration by deploying all code changes to a testing environment and/or a production environment after the build stage.

Continuous Integration Tools:

- Jenkins.
- CircleCI.
- TeamCity.
- Travis CI.
- Buddy.
- GitLab.
- Bamboo.
- Buildbot.

What is Jenkins and Why we use it?

Jenkins is an open-source automation tool written in Java with plugins built for continuous integration. Jenkins is used to build and test your software projects continuously making it easier for developers to integrate changes to the project, and making it easier for users to obtain a fresh build. It also allows you to continuously deliver your software by integrating with a large number of testing and deployment technologies.

With Jenkins, organizations can accelerate the software development process through automation. Jenkins integrates development life-cycle processes of all kinds, including build, document, test, package, stage, deploy, static analysis, and much more.



Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

With Jenkins, organizations can accelerate the software development process through automation. Jenkins integrates development life-cycle processes of all kinds, including build, document, test, package, stage, deploy, static analysis, and much more.

Jenkins achieves Continuous Integration with the help of plugins. Plugins allow the integration of Various DevOps stages. If you want to integrate a particular tool, you need to install the plugins for that tool. For example Git, Maven 2 project, Amazon EC2, HTML publisher etc.

Advantages of Jenkins include:

1. It is an open-source tool with great community support.
2. It is easy to install.
3. It has 1000+ plugins to ease your work. If a plugin does not exist, you can code it and share it with the community.
4. It is free of cost.
5. It is built with Java and hence, it is portable to all the major platforms.

What is Maven?

Maven is a powerful project management and comprehension tool that provides complete build life cycle framework to assist developers. It is based on the concept of a POM (Project Object Model) that includes project information and configuration information for Maven such as construction directory, source directory, test source directory, dependency, Goals, plugins etc.

Maven is build automation tool used basically for Java projects, though it can also be used to build and manage projects written in C#, Scala, Ruby, and other languages. Maven addresses two aspects of building software: 1st it describes how software is build and 2nd it describes its dependencies.

Maven when integrated with Jenkins through plugins aids to automate the complete build.

Steps to install and configure Jenkins with Maven :

Step 1: Download Jenkins war file

War is required to install Jenkins

The official website for Jenkins is <https://jenkins.io/>

Click on download button

Click on Generic Java Package (.war) to download the Jenkins war file.

Step 2: Now go to the location where the file is downloaded.

Open the command prompt and go to the directory where the Jenkins.war file is located. And then run the following command:

```
java -jar jenkins.war
```



Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

Step 3: Accessing Jenkins--Open your browser and type the following url on your browser:
<http://localhost:8080>

This url will bring up the Jenkins dashboard.

Step 4: Now go to the path C:\Users\jenkins\secrets as per your System. Here you will find your Admin Password to continue the process of installation (Copy that password and paste it in Administrator field).

Step 5: After entering the password you will be redirected to the page select suggested plugins option. Click on Install Suggested Plugins.

Step 6: After Jenkins finishes installing the plugins, enter the required information on the Create First Admin User page.

Step 7: On the Instance Configuration page, confirm the port number you want Jenkins to use and click Save and Finish. (Default port number is 8080)

Step 8: After that go the port 8080 (Make sure that your (.war) file is running on cmd)

Step 9: You should know your user name and password that you have created previously. Click on sign in.

Step 10: Download Maven

The official website for Apache Maven is <https://maven.apache.org/download.cgi>.

Go to the files section and download the Maven by the given link for Binary zip archive file. Once the file is downloaded, extract the file into your system.

Step 11: Setting Up Java and Maven in Jenkins

First of all, you have to set the JAVA_HOME and MAVEN_HOME environment variable in your system.

To set the JAVA_HOME and MAVEN_HOME path:

- i. Make sure JDK is installed, and JAVA_HOME environment variable is configured. You can verify that the JAVA_HOME environment variable is properly configured or not by using the following command: C:\java -version
- ii. Add a MAVEN_HOME system variables, and point it to the Maven folder.
Press Windows key, type adva and clicks on the View advanced system settings
- iii. In System Properties dialog, select Advanced tab and clicks on the Environment Variables... button.
- iv. In “Environment variables” dialog, System variables, Clicks on the New... button and add a MAVEN_HOME variable and point it to c:\opt\apache-maven-3.6.0



Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

- v. Add %MAVEN_HOME%\bin To PATH : In system variables, find PATH, clicks on the Edit... button. In “Edit environment variable” dialog, clicks on the New button and add this %MAVEN_HOME%\bin
- vi. Verification: Done, start a new command prompt, type mvn –version

Step 12: Now, in the Jenkins dashboard (Home screen) click on manage Jenkins from the left-hand side menu.

The screenshot shows the Jenkins dashboard. On the left, there is a sidebar with the following options: 'New Item', 'People', 'Build History', 'Manage Jenkins', and 'My Views'. Under 'Build Queue', it says 'No builds in the queue.' Under 'Build Executor Status', it shows '1 Idle' and '2 Idle'. The main content area has a heading 'Welcome to Jenkins!' followed by a sub-heading 'Start building your software project'. It features several buttons: 'Create a job' (highlighted in blue), 'Set up a distributed build', 'Set up an agent', 'Configure a cloud', and a link 'Learn more about distributed builds'.

Step 13: Click on "Global Tool Configuration" option.



Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

The screenshot shows the Jenkins Manage Jenkins interface. On the left, there is a sidebar with links: New Item, People, Build History, Manage Jenkins (which is highlighted), and My Views. Below this are two dropdown menus: Build Queue (No builds in the queue) and Build Executor Status (1 idle, 2 idle). The main area is titled "Manage Jenkins" and contains several sections: System Configuration (with System, Tools, Plugins, Nodes, Clouds, and Appearance), Security, and a summary table for Jenkins instances.

Step 14: To configure Java, click on "Add JDK" button in the JDK section.

The screenshot shows the Jenkins Tools configuration page. It includes sections for Maven Configuration (Default settings provider: Use default maven settings) and JDK installations. The JDK installations section has a "Add JDK" button, which opens a modal dialog. In the dialog, there is a "Name" field containing "JAVA_HOME" and an "Install automatically" checkbox. At the bottom of the dialog are "Save" and "Apply" buttons.

Step 15: Give a Name and JAVA_HOME path, or check on install automatically checkbox.



Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

The screenshot shows the Jenkins 'Tools' section under 'Manage Jenkins'. The 'JDK installations' tab is selected. A new entry is being created with the name 'JAVA_HOME' and the path 'C:\Program Files\Java\jdk-21'. The 'Install automatically' checkbox is left unchecked.

Step 16: And now, to configure Maven, click on "Add Maven" button in the Maven section, give any Name and MAVEN_HOME path or check to install automatically checkbox.

The screenshot shows the Jenkins 'Tools' section under 'Manage Jenkins'. The 'Maven installations' tab is selected. A new entry is being created with the name 'MAVEN_HOME' and the path 'C:\Program Files\apache-maven-3.8.6'. The 'Install automatically' checkbox is left unchecked. At the bottom, there are 'Save' and 'Apply' buttons.

Then, click on the "**Save**" button at the end of the screen.

Now, you can create a job with the Maven project. To do that, click on the **New Item** option or **create a new job** option.



Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

The screenshot shows the Jenkins dashboard. On the left, there is a sidebar with links: New Item, People, Build History, Manage Jenkins, Credentials, and New View. Below this are two summary boxes: 'Build Queue' (No builds in the queue) and 'Build Executor Status' (1 Idle, 2 Idle). The main area features a large 'Welcome to Jenkins!' message with a call to action: 'Please [create new jobs](#) to get started.' There is also a search bar at the top right.

Step 17: Enter the **Item Name** and select the **Maven Project**.

Click OK.

The screenshot shows the 'New Item' creation dialog. A search bar at the top has 'EP4' typed into it. Below it is a list of project types: 'Freestyle project', 'Pipeline', 'Multi-configuration project', 'Folder', 'Multibranch Pipeline', and 'Organization Folder'. Each item has a small icon and a brief description. At the bottom of the list is a blue 'OK' button and a note: 'If you want to create a new item from other existing, you can use this option'.

Step 18: Now configure the job. Give the description and in the **Source Code Management** section, select the required option.



Vidya Vardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

The screenshot shows the Jenkins configuration interface for a project. The top navigation bar includes 'Dashboard', 'Jobs', and 'Configurations'. The main area has tabs for 'Configure' and 'General'. In the 'General' tab, the 'Enabled' switch is turned on. The 'Source Code Management' section is selected, showing options for 'None' or 'Git'. The 'Build Triggers' section contains several checkboxes: 'Trigger builds remotely (e.g., from scripts)', 'Build after other projects are built', 'Build periodically', 'GitHub hook trigger for GitHub pull requests', and 'Poll SCM'. The 'Build Environment' section includes checkboxes for workspace cleanup, timestamps in output, and inspecting build log. Buttons for 'Save' and 'Apply' are at the bottom.

Step 19: In the Build Triggers section, there are multiple options, select the required one.

Add the pom.xml file's path in the **Root POM** option. Configure the other fields as per your requirement and then click on the **Save** button.



Vidya Vardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

The screenshot shows the Jenkins Pipeline configuration interface. The left sidebar lists sections: General, Source Code Management, Build Triggers, Build Environment, Build Step (which is selected), and Post-build Actions. The main panel displays a 'Build Step' configuration for 'Execute shell'. It includes a 'Command' field containing the placeholder 'See the list of available environment variables.' Below the command field is an 'Advanced' dropdown and a 'Add build step' button. At the bottom of the panel are 'Save' and 'Apply' buttons.

Conclusion:

1. What are the requirements for using Jenkins?
 - 1.Jenkins requires Java to run, so you need to install the latest Java Development Kit (JDK) or Java Runtime Environment (JRE) versions based on your system configuration, 32-bit or 64-bit.
 - 2.Jenkins requires at least 512 MB of RAM and 1 GB of storage on your hard drive.
 - 3.You can download Jenkins from the official website (<<https://jenkins.io/download/>>) and select the platform you intend to download Jenkins. For Windows, you can download the .exe file, while for other platforms, you can download the .war file.
 4. use Jenkins for continuous integration and continuous delivery, you need to define a set of instructions in a script, usually written in Groovy. This script includes stages, steps, and triggers to automate the software delivery process.
 5. Jenkins provides a vast number of plugins that can be installed to extend its functionality. For example, the Docker Pipeline plugin is required to run a Jenkins pipeline using Docker.
2. Name the two components that Jenkins is mostly integrated with
Jenkins is mainly integrated with two components: version systems and build tools. Version systems like Git and SVN are examples of the former, while build tools like Apache Maven are examples of the latter. This integration allows Jenkins to efficiently manage code deployment processes by monitoring changes in the source code repository, preparing builds, testing them, and deploying the code in various environments. The continuous automation and feedback loop provided by Jenkins streamline the development and deployment processes, ensuring efficient and error-free software delivery.



Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

3. Name some of the useful plugins in Jenkins.

1.Git Plugin: This plugin integrates Jenkins with Git version control systems, providing features like pulling, fetching, checking out, branching, merging, tagging, and pushing repositories. It also supports distributed version control, security, and flexibility for effective utilization

2.Kubernetes Plugin: This plugin enables Jenkins to run jobs on Kubernetes clusters, allowing for dynamic scaling and resource management

3.GitHub Integration Plugin: This plugin integrates Jenkins with GitHub, providing features like automatic build triggering, pull request integration, and issue tracking

4.Maven Integration Plugin: This plugin integrates Jenkins with Maven, enabling seamless build, testing, and deployment of Maven projects

5.Docker Plugin: This plugin allows Jenkins to build, test, and deploy Docker applications, and also supports containerization and orchestration

6.JUnit Plugin: This plugin integrates Jenkins with JUnit testing framework, providing features like test results visualization, trend tracking, and failure analysis.