



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?

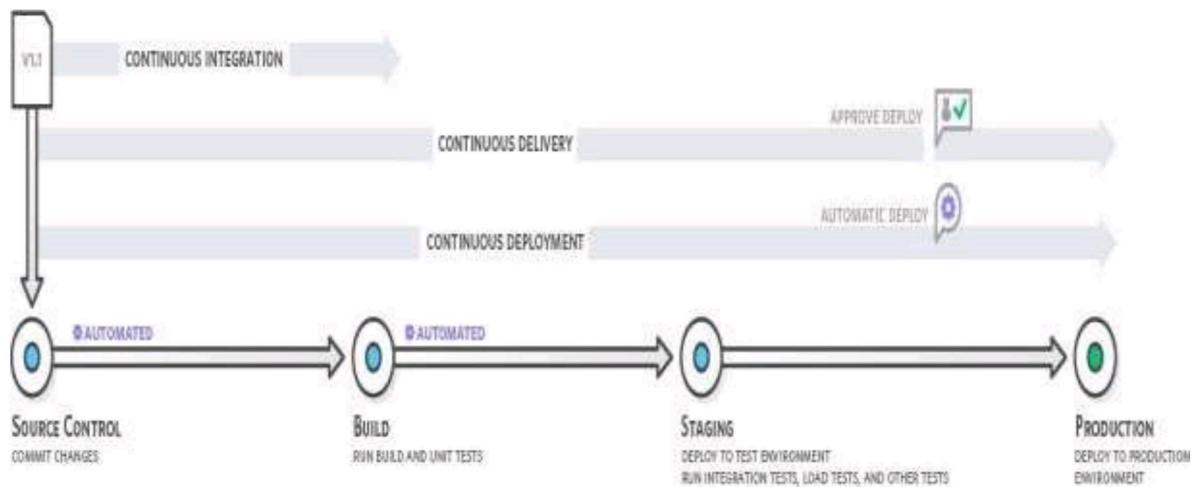
With continuous integration, developers frequently commit to a shared repository using a version control system such as Git. Prior to each commit, developers may choose to run local unit tests on their code as an extra verification layer before integrating. A continuous integration service automatically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

errors.





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.



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!' with a sub-instruction: 'This page is where your Jenkins jobs will be displayed. To get started, you can set up distributed builds or start building a software project.' Below this, there are three buttons: 'Create a job' (with a plus sign icon), 'Set up a distributed build', and 'Set up an agent'. Further down are 'Configure a cloud' and a link 'Learn more about distributed builds' with a question mark icon.

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's a sidebar with links for New Item, People, Build History, Manage Jenkins (which is selected), and My Views. Below this are sections for Build Queue and Build Executor Status. The main area is titled 'Manage Jenkins' and contains a 'System Configuration' section. It includes icons and descriptions for System (configure global settings), Tools (configure tools and paths), Plugins (add or remove plugins), Nodes (manage nodes), Clouds (configure cloud instances), and Appearance (configure look and feel). There's also a 'Security' link at the bottom.

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

The screenshot shows the Jenkins Tools configuration page. It has sections for Maven Configuration and JDK installations. Under Maven Configuration, there are dropdowns for 'Default settings provider' (set to 'Use default maven settings') and 'Default global settings provider' (set to 'Use default maven-global settings'). Under 'JDK installations', there's a 'Add JDK' button. A modal dialog is open, showing a list with one item: '= /DK' (Name: JAVA_HOME). 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

JDK installations

Add JDK

JDK

Name: JAVA_HOME

JAVA_HOME: C:\Program Files\java\jdk-21

Install automatically

Add JDK

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.

Maven installations

Maven installations

Add Maven

Maven

Name: MAVEN_HOME

MAVEN_HOME: C:\Program Files\apache-maven-3.9.0

Install automatically

Add Maven

Save Apply

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 sections: '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 button to 'create new jobs'.

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

Click OK.

The screenshot shows the 'New Item' dialog in Jenkins. It lists several project types: 'FreeStyle project' (selected), 'Pipeline', 'Multi-configuration project', 'Folder', 'Multibranch Pipeline', and 'Organization Folder'. A note at the bottom says '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.



VidyaVardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

The screenshot shows two stacked Jenkins configuration pages. The top page is for a job named 'General'. It has tabs for 'Configure' and 'General'. Under 'General', the 'Enabled' switch is turned on. The 'Description' field is empty. Under 'Post-build Actions', there are several checkboxes: 'Delete old builds', 'GitHub project', 'This project is parameterized', 'Notify users', and 'Execute concurrent builds if necessary'. The bottom page is for 'Source-Code Management'. It also has tabs for 'Configure' and 'Source-Code Management'. Under 'Source-Code Management', the 'Name' dropdown is set to 'git'. The 'Build Triggers' section contains checkboxes for 'Trigger build remotely (e.g., from script)', 'Build after other projects are built', 'Build periodically', 'GitHub hook trigger for GitHub pull requests', and 'Poll SCM'. The 'Build Environment' section contains checkboxes for 'Delete workspace before build starts', 'Use user-defined workspace', 'Add timestamps to the console output', and 'Inspect build log for published build status'. At the bottom of both pages are 'Save' and 'Apply' buttons.

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.



Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

The screenshot shows the Jenkins configuration interface for a job. The left sidebar has links for Dashboard, JIRA, Configuration, General, Source Code Management, Build Triggers, Build Environment, Build Step (which is selected and highlighted in grey), and Post-build Actions. The main area is titled 'Configure' and contains a 'Build Step' section with a 'Command' input field containing 'sh -c echo \$env'. Below this is a 'Post-build Actions' section with a 'Add post-build action' link. At the bottom are 'Save' and 'Apply' buttons.

Conclusion:

1. What are the requirements for using Jenkins?

Ans: A machine with:

256 MB of RAM, although more than 2 GB is recommended

10 GB of drive space (for Jenkins and your Docker image)

The following software installed:

Java 11, 17, or 21

Docker (navigate to Get Docker site to access the Docker download that's suitable for your platform)

2. Name the two components that Jenkins is mostly integrated with.

Ans: Jenkins is typically integrated with these two components: Version Control systems like Git and SVN (Apache Subversion) Build tools like Maven.

3. Name some of the useful plugins in Jenkins.

Ans: Some of the plugins in Jenkins include:

1. Maven 2 project.
2. Amazon EC2.
3. Copy artifact.
4. Join.
5. HTML publisher.
6. Green Balls.