**Jenkins :**

* Jenkins is an open source automation tool written in Java programming language that allows continuous integration.
* Jenkins **builds** and **tests** our software projects which 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 us to continuously **deliver** our software by integrating with a large number of testing and deployment technologies.
* Jenkins offers a straightforward way to set up a continuous integration or continuous delivery environment for almost any combination of languages and source code repositories using pipelines, as well as automating other routine development tasks.
* With the help of Jenkins, organizations can speed up the software development process through automation. Jenkins adds development life-cycle processes of all kinds, including build, document, test, package, stage, deploy static analysis and much more. Jenkins achieves CI (Continuous Integration) with the help of plugins.
* Plugins is used to allow the integration of various DevOps stages. If you want to integrate a particular tool, you have to install the plugins for that tool. For example: Maven 2 Project, Git, HTML Publisher, Amazon EC2, etc.
* **For example:** If any organization is developing a project, then **Jenkins** will continuously test your project builds and show you the errors in early stages of your development.
* Possible steps executed by Jenkins are for example:
* Perform a software build using a build system like Gradle or Maven Apache
* Execute a shell script
* Archive a build result
* Running software tests

Continuous Integration *(CI)*

Continuous Integration *(CI)* is a development practice in which the developers are needs to commit changes to the source code in a shared repository at regular intervals. Every commit made in the repository is then built. This allows the development teams to detect the problems early.

Continuous integration requires the developers to have regular builds. The general practice is that whenever a code commit occurs, a build should be triggered.

**Continuous Integration with Jenkins**

Let's consider a scenario where the complete source code of the application was built and then deployed on test server for testing. It sounds like a perfect way to *develop software*, but this process has many problems.

* Developer teams have to wait till the complete software is developed for the test results.
* There is a high prospect that the test results might show multiple bugs. It was tough for developers to locate those bugs because they have to check the entire source code of the application.
* It slows the software delivery process.
* Continuous feedback pertaining to things like architectural or coding issues, build failures, test status and file release uploads was missing due to which the quality of software can go down.
* The whole process was manual which increases the threat of frequent failure.

It is obvious from the above stated problems that not only the software delivery process became slow but the quality of software also went down. This leads to customer dissatisfaction.

So to overcome such problem there was a need for a system to exist where developers can continuously trigger a build and test for every change made in the source code.

This is what Continuous Integration (CI) is all about. Jenkins is the most mature Continuous Integration tool available so let us see how Continuous Integration with Jenkins overcame the above shortcomings.

### **Jenkins Master**

The main server of Jenkins is the Jenkins Master. It is a web dashboard which is nothing but powered from a war file. By default, it runs on 8080 port. With the help of Dashboard, we can configure the jobs/projects but the build takes place in Nodes/Slave. By default one node (slave) is configured and running in Jenkins server. We can add more nodes using IP address, user name and password using the ssh, jnlp or web start methods.

### **Jenkins Slave**

Jenkins slave is used to execute the build jobs dispatched by the master. We can configure a project to always run on a particular slave machine, or particular type of slave machine, or simple let the Jenkins to pick the next available slave/node.

As we know Jenkins is developed using Java is platform independent thus Jenkins Master/Servers and Slave/nodes can be configured in any servers including Linux, Windows, and Mac.

## Different Types of Jenkins Jobs

Jenkins provides the option of choosing from different types of jobs to build your project.

Below are the types of jobs you can choose from:

* Freestyle

Freestyle build jobs are general-purpose build jobs, which provides maximum flexibility. It can be used for any type of project.

* Pipeline

This project runs the entire software development workflow as code. Instead of creating several jobs for each stage of software development, you can now run the entire workflow as one code.

* Multiconfiguration

The multiconfiguration project allows you to run the same build job on different environments. It is used for testing an application in different environments.

* Folder

This project allows users to create folders to organize and categorize similar jobs in one folder or sub folder.

* GitHub Organization

This project scans your entire GitHub organization and creates Pipeline jobs for each repository containing a Jenkinsfile

* Multibranch pipeline

This project type lets you implement different Jenkinsfiles for different branches of the same project.

## Build Pipeline

Build pipeline can be used to chain several jobs together and run them in a sequence. Let’s see how to install Build Pipeline:

Jenkins Dashboard -> Manage Jenkins -> Manage Plugins -> Available

In the filter text field enter the name of the plugin you want to install.

### **Build Pipeline Example**

Step 1: Create 3 freestyle Jobs (Job1, Job2, Job3)

Step 2: Chain the 3 Jobs together

Job1 -> configure -> Post Build -> Build other projects -> Job2

Job2 -> configure -> Post Build -> Build other projects -> Job3

Step 3: Create a build pipeline view

Jenkins Dashboard -> Add view -> Enter a name -> Build pipeline view -> ok ->

configure -> Pipeline flow -> Select Initial job -> Job1 -> ok

Step 4: Run the Build Pipeline

## Jenkins Pipeline

Jenkins’s pipeline is a single platform that runs the entire pipeline as code. Instead of building several jobs for each phase, you can now code the entire workflow and put it in a Jenkins file.

Jenkins file is a text file that stores the pipeline as code. It is written using the Groovy DSL. It can be written based on two syntaxes:

* Scripted pipeline

Code is written on the Jenkins UI instance and is enclosed within the node block