Jenkins

# Introduction

Continuous Integration is the most important part of DevOps that is used to integrate various [DevOps stages](https://www.edureka.co/devops). Jenkins is the most famous Continuous Integration tool.

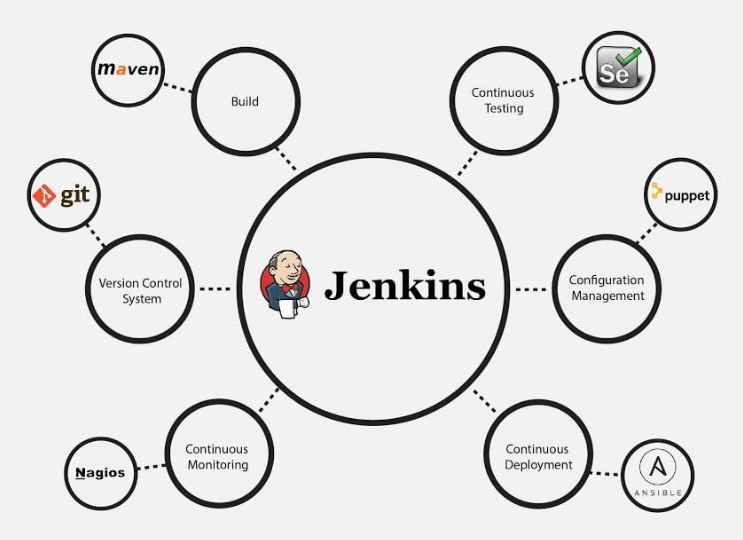
**WHAT IS Jenkins?**

Jenkins is an open source automation tool written in Java with plugins built for Continuous Integration purpose. 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 many 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.

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

The image below depicts that Jenkins is integrating various DevOps stages:



**Advantages of Jenkins include:**

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

There are certain things about Jenkins that separates it from other the Continuous Integration tool.

**Jenkins Key Metrics**

Following are some facts about Jenkins that makes it better than other Continuous Integration tools:

* Adoption: Jenkins is widespread, with more than 147,000 active installations and over 1 million users around the world.
* Plugins: Jenkins is interconnected with well over 1,000 plugins that allow it to integrate with most of the development, testing and deployment tools.

 It is evident from the above points that Jenkins has a very high demand globally. Before we dive into Jenkins it is important to know what is Continuous Integration and why it was introduced.

**What is Continuous Integration?**

Continuous Integration is a development practice in which the developers are required to commit changes to the source code in a shared repository several times a day or more frequently. Every commit made in the repository is then built. This allows the teams to detect the problems early. Apart from this, depending on the Continuous Integration tool, there are several other functions like deploying the build application on the test server, providing the concerned teams with the build and test results etc.

Let us understand its importance with a use-case.

**Continuous Integration in Nokia**

I am pretty sure you all have used Nokia phones at some point in your life. In a software product development project at Nokia there was a process called Nightly builds. Nightly builds can be thought of as a predecessor to Continuous Integration. It means that every night an automated system pulls the code added to the shared repository throughout the day and builds that code. The idea is quite similar to Continuous Integration, but since the code that was built at night was quite large, locating and fixing of bugs was a real pain. Due to this, Nokia adopted Continuous Integration (CI). As a result, every commit made to the source code in the repository was built. If the build result shows that there is a bug in the code, then the developers only need to check that particular commit. This significantly reduced the time required to release new software.

**Continuous Integration With Jenkins**

Let us imagine 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 a software, but, this process has many flaws. I will try to explain them one by one:

* Developers have to wait till the complete software is developed for the test results.
* There is a high possibility 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 coding or architectural 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 risk of frequent failure.

It is evident 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 a chaos there was a dire 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 CI is all about. Jenkins is the most mature CI tool available so let us see how Continuous Integration with Jenkins overcame the above shortcomings.

I will first explain you a generic flow diagram of Continuous Integration with Jenkins so that it becomes self explanatory, how Jenkins overcomes the above shortcomings:

## Download

The following is a link to the open-source version of Jenkins: <https://jenkins.io/>

## Benefits

**CI/CD**As an extensible automation server, Jenkins can be used as a simple CI server or turned into the continuous delivery hub for any project.

**installation**Jenkins is a self-contained Java-based program, ready to run out-of-the-box, with packages for Windows, Mac OS X and other Unix-like operating systems.

**Configuration**Jenkins can be easily set up and configured via its web interface, which includes on-the-fly error checks and built-in help.

**Plugins**With hundreds of plugins in the Update Center, Jenkins integrates with practically every tool in the continuous integration and continuous delivery toolchain.

**Extensible**Jenkins can be extended via its plugin architecture, providing nearly infinite possibilities for what Jenkins can do.

**Distributed**Jenkins can easily distribute work across multiple machines, helping drive builds, tests and deployments across multiple platforms faster.

## Jenkins

Video: <https://www.youtube.com/watch?v=WWcijE7ifcA>

Source: <https://www.edureka.co/blog/what-is-jenkins/>  
 <https://jenkins.io/>