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, I know you are curious to know the reason behind the popularity of Jenkins, and if Jenkins is easy to learn. I am pretty sure after reading this ***What is Jenkins*** blog, all your questions will get answered.

**What is Jenkins and why we use it?**

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

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.

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 it 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. Let us take a look on those points.

**enkins Features**

The 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 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 software, but, this process has many flaws. I will try to explain them one by one:

DevOps Training



The above diagram is depicting the following functions:

* First, a developer commits the code to the source code repository. Meanwhile, the Jenkins server checks the repository at regular intervals for changes.
* Soon after a commit occurs, the Jenkins server detects the changes that have occurred in the source code repository. Jenkins will pull those changes and will start preparing a new build.
* If the build fails, then the concerned team will be notified.
* If built is successful, then Jenkins deploys the built in the test server.
* After testing, Jenkins generates a feedback and then notifies the developers about the build and test results.
* It will continue to check the  source code repository for changes made in the source code and the whole process keeps on repeating.

**What is Continuous Deployment (CD)?**

As a reminder, [**Continuous Integration**](https://www.padok.fr/en/blog/devops-continuous-integration) (CI) is an **automated testing phase integrated into the deployment flow**.

**Continuous Deployment** (CD) is the **continuation of Continuous Integration**. Once the tests have been validated on the dev environment, it must be put into production. Continuous deployment, therefore, **consists of automating deployment actions** that were previously performed manually. This is why we often talk about CI/CD together.

To automate deployment actions, **tests on the qualification environment need to be automated** to ensure that the new functionality to be pushed works properly.

**The difference between continuous**

**The difference between continuous deployment and continuous delivery**

**Continuous deployment** is an ideal that few companies have implemented. Most of the time, IT teams prefer to have control over the last stage of the deployment. In this case, we talk about **continuous delivery, all the steps of the deployment are automated except for the last one**: the production launch.

