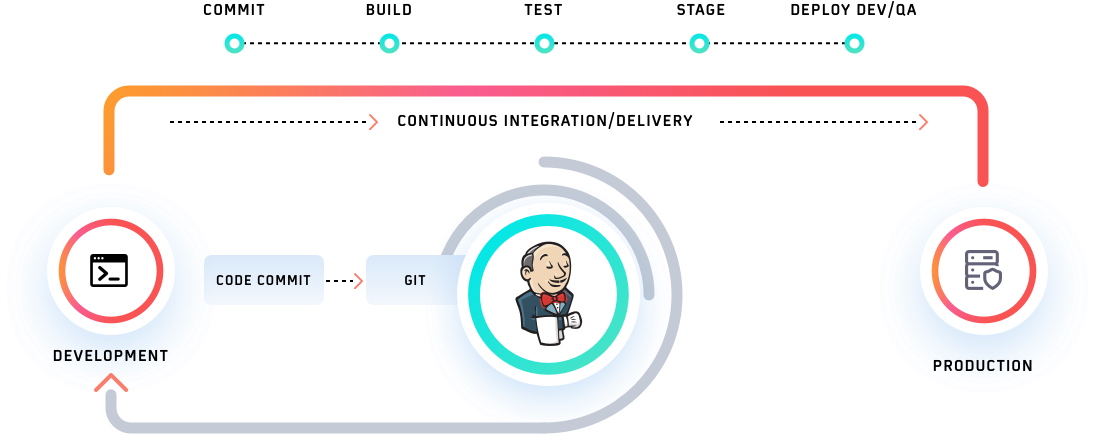
## What Is Jenkins Pipeline?



Inside Jenkins CI/CD, a pipeline is defined as a series of events or tasks which are interconnected in a particular order. In simple terms, Jenkins pipeline is a set of modules or plugins which enable the implementation and integration of Continuous Delivery pipelines within Jenkins.

The Jenkins pipeline has an expandable automation system for building basic or complicated ‘template’ distribution pipelines via the Domain-specific language (DSL) used in the pipeline. There are four states of Continuous Delivery in Jenkins pipeline-

* Build
* Deploy
* Test
* Release

## Why Use Jenkins Pipeline?

Jenkins CI/CD plays a significant role in delivering high-quality applications or products. Jenkins has proved to be a specialist in Continuous Integration, Continuous Testing, and Continuous Delivery. It uses a feature called Jenkins pipeline for Continuous Delivery, which is basically the ability to release apps regularly at an interval. This process ensures that the software is always ready for production.

### Advantages Of Jenkins Pipeline

The highlight of Jenkins pipeline is that it offers the feature to define the complete deployment flow-through configuration and code. It states that all the standard Jenkins jobs can be written manually as an entire script and can be managed with a version control system.

It’s essentially following the discipline of ‘pipeline as code.’ Hence instead of creating multiple jobs for each process, it allows us to code the entire workflow and place it in a Jenkins file. Below are some of the reasons that one might consider before using the Jenkins pipeline for Jenkins test automation with Selenium.

* By using Groovy DSL (Domain Specific Language), it models easy to complex pipelines as code.
* The code is stored in the form of a text file called ‘Jenkinsfile’ that can be scanned into Source Code Management.
* It supports complex pipelines by adding conditional loops, forks, or joining operations and allowing parallel execution tasks.
* It improves user experience by integrating user feedback into the pipeline.
* It’s resilient in terms of Jenkins’ master unplanned restart.
* It can resume from checkpoints saved.
* It can incorporate multiple additional plugins and add-ins.

## What Is Jenkinsfile?

It is a text file that stores the whole process as code in our local machine. It can be reviewed in a Source Code Management (SCM) platform such as Git. It is instrumental as it helps the developers to view, edit, and test the code whenever required.

The Jenkinsfile is written using the Groovy Domain-Specific Language and can be generated using a text editor or the Jenkins instance configuration tab.

There are two different types in which the Jenkins pipeline can be constructed. These are the syntaxes-

* Declarative pipeline syntax
* Scripted pipeline syntax

The Declarative Pipelines is a relatively new feature that supports the concept of code pipeline. It enables the reading and writing of the pipeline code. This code is written within a Jenkinsfile, which can be tested into a tool such as Git for source control.

The **Scripted** pipeline is a typical method of code writing. The Jenkinsfile is written on the Jenkins user interface instance in this pipeline.

While both of these pipelines are Groovy-based, the scripted pipeline uses more strict Groovy-based syntaxes. This is because it was the first groovy foundation pipeline that was created for use. As this Groovy script was not usually suitable to all users, it introduced the declarative pipeline to provide a simpler and more flexible Groovy syntax. The declarative pipeline is defined within a ‘pipeline’ block, while the scripted pipeline is defined within a ‘node’ block.