Jenkins Pipeline

In Jenkins, a pipeline is a collection of events or jobs which are interlinked with one another in a sequence.

It is a combination of plugins that support the integration and implementation of continuous delivery pipelines using Jenkins.

In other words, a Jenkins Pipeline is a collection of jobs or events that brings the software from version control into the hands of the end users by using automation tools. It is used to incorporate continuous delivery in our software development workflow.

A pipeline has an extensible automation server for creating simple or even complex delivery pipelines "as code", via DSL (Domain-specific language).

Continuous Delivery Pipeline

In a Jenkins Pipeline, every job has some sort of dependency on at least one or more jobs or events.

Jenkins Pipeline

The above diagram represents a continuous delivery pipeline in Jenkins. It contains a collection of states such as build, deploy, test and release. These jobs or events are interlinked with each other. Every state has its jobs, which work in a sequence called a continuous delivery pipeline.

A continuous delivery pipeline is an automated expression to show your process for getting software for version control. Thus, every change made in your software goes through a number of complex processes on its manner to being released. It also involves developing the software in a repeatable and reliable manner, and progression of the built software through multiple stages of testing and deployment.

JenkinsFile

Jenkins Pipeline can be defined by a text file called JenkinsFile. You can implement pipeline as code using JenkinsFile, and this can be defined by using a DSL (Domain Specific Language). With the help of JenkinsFile, you can write the steps required for running a Jenkins Pipeline.

The benefits of using JenkinsFile are:

You can make pipelines automatically for all branches and can execute pull requests with just one JenkinsFile.
This is the singular source for your pipeline and can be customized by multiple users.
JenkinsFile can be defined by using either Web UI or with a JenkinsFile.

Pipeline syntax
Two types of syntax are used for defining your JenkinsFile.

Declarative
Scripted
Declarative:

Declarative pipeline syntax offers a simple way to create pipelines. It consists of a predefined hierarchy to create Jenkins pipelines. It provides you the ability to control all aspects of a pipeline execution in a simple, straightforward manner.

Scripted:

Scripted Jenkins pipeline syntax runs on the Jenkins master with the help of a lightweight executor. It uses very few resources to convert the pipeline into atomic commands.

Both scripted and declarative syntax are different from each other and are defined totally differently.

Why Use Jenkins Pipeline?
Jenkins is a continuous integration server which has the ability to support the automation of software development processes. You can create several automation jobs with the help of use cases, and run them as a Jenkins pipeline.

the reasons why you should use Jenkins pipeline:

Jenkins pipeline is implemented as a code which allows several users to edit and execute the pipeline process.
Pipelines are robust. So if your server undergoes an unpredicted restart, the pipeline will be automatically resumed.
You can pause the pipeline process and make it wait to continue until there is an input from the user.
Jenkins Pipelines support big projects. You can run many jobs, and even use pipelines in a loop.
Jenkins Pipeline Concepts
Pipeline: This is the user-defined block, which contains all the processes such as build, test, deploy, etc. it is a group of all the stages in a JenkinsFile. All the stages and steps are defined in this block. It is used in declarative pipeline syntax.

GitHub Setup for Jenkins

Jenkins is a CI (Continuous Integration) server and this means that it needs to check out source code from a source code repository and build code. Jenkins has outstanding support for various source code management systems like Subversion, CVS etc.

Github is the fast becoming one of the most popular source code management systems. It is a web based repository of code which plays a major role in DevOps. GitHub provides a common platform for many developers working on the same code or project to upload and retrieve updated code, thereby facilitating continuous integration. Jenkins works with Git through the Git plugin.

Connecting a GitHub private repository to a private instance of Jenkins can be tricky.

To do the GitHub setup, make sure that internet connectivity is present in the machine where Jenkins is installed.

- In the Home screen of the Jenkins (Jenkins Dashboard), click on the **Manage Jenkins** option on the left hand side of the screen.

- Now, click on the **Manage Plugins** option.

- In the next page, click on the "Available tab".

- The "Available" tab gives a list of plugins which are available for downloading. In the Filter tab type, type the "Git Plugin".
- Select the Git Plugin.
- Click on the "**install without restart**". The plugin will take some time to finish downloading depending on your internet connection, and will be installed automatically.
- You can also click on "**Download now and install after restart**" button in which the git plugin is installed after restart.
- If you already have the Git plugin installed then go to "Installed" tab and in filter option type Git plugin.
- Once all the installations are completed, restart Jenkins by giving the following command in the browser.

  After Jenkins is restarted, Git will available as an option while configuring jobs.

Integrating Jenkins with GitHub

Let's see the process of integrating GitHub into Jenkins in a windows system.

- First create a new job in Jenkins, open the Jenkins Dashboard and click on "create new jobs".
- Now enter the item name and select the job type. For example, item name is **javaTpoint**" and job type is "**Freestyle project**".
- Click on **OK**.
- Once you click OK, the page will be redirected to its project configuration. Enter the project information:
- Now, under the "Source Code Management" you will see the Git option, if your **Git** plugin has been installed in Jenkins:
- Enter the Git repository URL on the "Repository URL" option to pull the code from GitHub.
- You might get an error when first time you enter the repository URL. For example:

This happens if you don't have Git installed in your system. To install the Git in your system, download the appropriate Git setup according to your operating system. I am installing for windows. Once the download is completed, install the Git.

Complete the following instructions to install the Git:

You can execute Git repositories in your Jenkins once Git has been installed on your system. To check if the Git has been installed on your system, open the command prompt, type Git and press Enter.

In the above screen, you observe that syntax and different options come up for Git. This means that Git has been installed in your machine.

- Now try to add the Git URL into Jenkins.
- Git is now successfully configured on your system.

Maven Setup

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.

Downloading Maven

The official website for Apache Maven is https://maven.apache.org/download.cgi. Click on the given link to download the Maven. When you click on the given link, you will get the home page of the official Maven website as given below:

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.

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.