**Experiment- 10**

**AIM:To study version control tool -- JENKINS**

**THEORY:**

**Jenkins**

When we talk about continuous integration or continuous delivery (CI/CD) tools, one tool that tops that list: Jenkins. It is the most famous open-source CI/CD tool used to support DevOps with several other cloud-native tools.

So, what is Jenkins? It’s an open-source automation server. It is possible to escalate the software development process with Jenkins automation. Throughout the life cycle, it helps you manage software delivery processes. Various life cycle stages include building, testing, documenting, packaging, staging, deployment, and more.

This article will assist you in understanding the Jenkins tool in detail. Now, let’s define Jenkins.

**What is Jenkins? Jenkins Explained**

Kohsuke Kawaguchi created Jenkins in 2004. The suite’s sole objective is continuous integration. Jenkins is the most popular solution for continuous delivery across the globe.

Written in Java, Jenkins allows continuous development, testing, and deployment of newly created codes.

The community originally wrote Jenkins as a build and release tool. It is a server-based application and requires a web server like Apache Tomcat. Jenkins allows users to build and test projects continuously, making it easier for developers to integrate changes to the project, and for users to obtain a fresh build.

Now, let’s analyze the concept of continuous integration!

**The actual involvement of Jenkins in the development process is as follows:**

1.Developers begin committing code to the repository, which Jenkins checks regularly

2.If there is a change in the code, Jenkins compiles the code

3.If there is a failed build, the developer gets notified

4.If the build is successful, the code is deployed to production

5.Its several large plugins are also useful, as they offer a lot of customizability. As a whole, it’s quite flexible and can work in different environments. Jenkins also allows you to schedule tests based on particular events, and you can set the frequency of these tests. You don’t have to run the test manually.

**Why Use Jenkins?**

So, Jenkins is a CI/CD tool. But there are many of those out there. Why would you use Jenkins over any other CI/CD solution?

It turns out that Jenkins has a wide array of benefits. It has a thriving, active community, it’s easy to deploy, and supports most environments. It’s a free, open-source solution, and you can customize your Jenkins distribution to your needs.

Jenkins has remained one of the most popular build/test solutions for Agile development, CI/CD, and DevOps with Jenkins because it is ubiquitous. Its thriving community provides phenomenal support, and its code base is incredibly stable.

Before using Jenkins, an organization might need to manually launch its testing solutions – lengthening the QA process and making it substantially more expensive. After Jenkins, an organization can instead automatically create a build and push it toward testing – making it easier to push commits down the CI/CD pipeline.

But this is only one use of Jenkins. Jenkins, as an all-around orchestration tool, has many advantages when used for automation.

**Features of Jenkins**

Jenkins has some features that really sell it as a CI/CD tool. These are some of them:

1.Plugins

2.Easy to set up

3.Supports most environments

4.Open-source

5.Easy distribution

**How Jenkins Works**

With all that in mind, what does Jenkins do? How does Jenkins work to improve your processes?

Initially, a developer commits the code to the source code. Now, the Jenkins server keeps a regular check at the repository. As soon as the commit occurs, the Jenkins server discovers the changes and pulls them. Further, it prepares a new build.

If the build does not pass in between tests, the developer is instantly notified to take suitable action. Now, Jenkins is ready to deploy the build to an environment that allows any User Acceptance Testing (UAT) before the release into the production pipeline.

Image: Jenkins Code Base Management.png, Alt-Text: A screenshot of Jenkins’ code base management.

The above-performed UAT tests are automated for continuous delivery with a tool known as Selenium. The code is appended to the master branch to create a golden build if the test is passed. It is then directly deployed into the production pipeline.

After testing, Jenkins notifies the developers of the test results and build.

Now, the whole cycle continues repeatedly.

In Jenkins, you can easily figure out which commit caused the build’s failure. However, if all the unit tests pass, the build pipeline will move forward to the next stage.

Jenkins runs parallel builds across several machines to decrease the total time consumed to complete the other activities.

Moreover, several multinational companies like Google and Amazon have achieved the continuous delivery milestone. It helps them deploy to production several times a day.

Jenkins’ continuous integration pipeline is incredibly powerful and inhabits several tools for different purposes, like hosting, monitoring, testing code, and compilation.

**Here is a list of such tools:**

1.Continuous integration server: Jenkins, TeamCity, etc

2.Source control tool: CVS, GIT, SVN, etc

3.Automation testing framework: Appium, Selenium, UFT, etc

4.Build tool: ANT, Ivy, Make, Gradle

**What is the Jenkins Pipeline?**

The Jenkins Pipeline is the interconnection of several sequential tasks and events. It is a pool of plugins that helps the continuous delivery pipelines with easy integration and implementation.

The primary feature of a Jenkins pipeline is that every task or job is dependent on another task or job. However, there are different states in the case of continuous delivery pipelines: build, test, deploy, release, etc. These states are interconnected.

A CD pipeline is a sequence of events in which these states work. Every change made to the software has to pass via multiple complex processes before the release.

**How to Implement the Pipelines**

JenkinsFile is used to define the Jenkins pipeline. It is a text file used to implement pipelines in code. This process is explained using DSL.

**Two syntaxes define the JenkinsFile.**

1.Scripted pipeline syntax: A syntax with resources that convert pipelines into atomic commands, running on Jenkins master.

2.Declarative pipeline syntax: A simple syntax with easy ways to control several pipeline execution aspects.

**Some Jenkins advantages include:**

1.It is easier to review the pipeline’s code.

2.You can conduct an audit.

3.It helps execute full requests for pipelines created for several branches.

4.Jenkins automates the software development process with continuous integration abilities.

**Architecture of Jenkins**

Jenkins adheres to a master-slave architecture technique to manage distributed builds.

**Jenkins came up with a distributed master-slave architecture:**

1.Jenkins Master

2.Jenkins Slave

3.Jenkins Master

**Advantages of Jenkins**

1.User-friendly and easy to install: You do not require extra installations or tools with it.

2.Platform-independent: You can find Jenkins on several platforms and operating systems such as Windows, Linux, OS X, etc.

3.Easy configuration: You can configure, modify or extend it. Moreover, code deployment and the report generation are quick. For continuous delivery and continuous integration, you can configure Jenkins according to your requirements.

4.Rich plugin ecosystem: Jenkins’ massive plugin collection makes it more flexible and permits building, deploying, and automating across several platforms. Free. Need we say more?

5.Incredible support: Jenkins is the most popular open-source server available; therefore, agile teams can resolve your queries all across the globe.

6.Automated integration. This helps save money and time in the journey of every project.

7.Easily detectable errors. This saves you from needing large-scale error-coated integrations. Moreover, in collaboration with Docker, Jenkins escalates the progress of the development team working on different projects. It also provides the software infrastructure with Agile development throughout the journey.

**Disadvantages of Jenkins**

1.Difficult management. It runs on a server and requires skills as a server administrator to monitor all associated activities.

2.In some aspects, it lacks user-friendliness when compared to current UI trends.

3.The process to install and configure Jenkins is pretty tedious.

4.The continuous integration pipeline breaks whenever changes occur in the settings. The integration halts and needs developer interference.

**Conclusion:**

What is Jenkins All About? also advantages and disadvantages.