What is Jenkins?

<In slide>

Various tools and programming languages that are supported by Jenkins are as follows.

- Programming languages:
    - Java
    - C#
    - C/C++
    - Python
    - JavaScript
- Test case tools:
    - TestLink
    - JIRA
    - PractiTest
    - TestRail
- Bug management tools:
    - SpiraTeam
    - Zoho
    - JIRA
    - Bugzilla
- Source code management tools (version control tools):
    - GitHub
    - GitLab
    - CVS (Concurrent Version System)
    - BitBucket

Formally speaking, Jenkins is an open-source automation server. It helps automate the parts of software development related to building, testing, deploying, facilitating continuous integration and continuous delivery.

Features

Can be installed on multiple OS like Windows or Linux

Developed in Java

- OpenJDK 11 or 17 in Linux are mandatory for installation on Ubuntu
- Latest JDK needed for installation on Windows

Continuous Integration/Continuous Delivery

This is a method that facilitates parallel development, incremental addition of features and deployment of these features in the system under test, to the deployment server through which the testers and end-users can access the system for various testing, verification and validation purposes.

Continuous integration is a coding philosophy and set of practices that drive development teams to frequently implement small code changes and check them in to a version control repository. Most modern applications require developing code using a variety of platforms and tools, so teams need a consistent mechanism to integrate and validate changes. Continuous integration establishes an automated way to build, package, and test their applications. Having a consistent integration process encourages developers to commit code changes more frequently, which leads to better collaboration and code quality.

Automatic project builds

- Time based
- Trigger based
- Nightly execution and reporting
- Helpful in management-level decision making

How does Jenkins work?

Here, the developer will write the project code and unit test scripts for the software code locally in a code, also the tester will write automation scripts in another code editor like Eclipse, NetBeans, etc.

This code will be pushed by the developer or tester from the code editor to the version control tools like GitHub or GitLab.

Jenkins will pull that code from GitLab, download all the dependencies like jar files, web browser supporting files, etc. through Maven.

For reporting the test script execution output, graphical reports are generated by Jenkins, error codes, status codes, error messages, etc. are displayed onto the Jenkins Console, test cases written in test case management tools are updated as passed, failed or blocked automatically based on the execution status of their corresponding test script code. Also, as per requirement the status of test script execution is sent on mail to all the registered stakeholders on Jenkins.

Using this Jenkins architecture, we can collectively work as a whole team. Here, we have developers and testers working together to prepare quality software.

Also, the senior management can work together with the developers and testers to track the progress of the overall project through Jenkins dashboard.

Give a demo of QA Jenkins with status of each service as projects, along with build status, decision making by management according to the status of project builds.

All these operations that Jenkins can perform are through a major functionality called Plugin.

Plugin

Plugin is basically a protocol or language in which Jenkins will connect with third-party tools, execute certain commands, display reports, etc.

All the tasks that we have performed in the above schematic diagram are through various plugins. Here, separate plugins for GitLab, Maven, TestNG, Java, Selenium, HTML reports, TestLink test cases, mail notifications, etc. are used for all the above tasks.

Slave node in Jenkins

In a production environment, there are a lot of builds need to run in parallel, in a software project. This is the time Jenkins slave nodes come into the play. We can run builds in separate Jenkins slave nodes. This will reduce build overhead to the Jenkins master and we can run build in parallel.

Explain the schematic diagram of master-slave architecture.

Steps:

1) We need to have at least 2 computers, connected to LAN.
2) Decide on a master computer of your choice.
3) Install Jenkins on each of the computers.
4) Add the details of the slave computer into Jenkins installed on master computer.
    a) Go to Manage Jenkins-->Manage Nodes-->New Node.
    b) Enter the slave node IP address, username and password.
    c) Connection will be done through SSH protocol.