




JENKINS



Jenkins: Continuous Integration and Continuous Delivery (CI/CD) Tool



Introduction

Jenkins is an automation server that is widely used in the field of software development to facilitate Continuous Integration and Continuous Delivery (CI/CD) processes.

It helps in automating various aspects of software development, including building, testing, and deploying applications.



WHY?

Open Source and Extensible:

Jenkins is an open-source tool, which means it's freely available and has a vast community of developers contributing to its growth. This openness allows for easy customization and extension through a wide range of plugins.

Plugin Ecosystem:

Jenkins boasts a rich ecosystem of plugins, offering integrations with numerous tools, version control systems, cloud platforms, and more. This flexibility enables organizations to build tailored CI/CD pipelines that fit their specific needs.



Azure Plugin



AWS Plugin



Github Actions Plugin



WHY?

Automation:

Jenkins automates various aspects of the software development lifecycle, including building, testing, and deployment. Automation reduces the risk of human error, increases efficiency, and allows for quicker releases.

Continuous Integration:

Jenkins promotes the practice of continuous integration, where code changes are regularly integrated into a shared repository. This ensures that code is tested and validated frequently, leading to early detection of bugs and integration issues.

WHY?

Continuous Delivery/Deployment:

Jenkins supports continuous delivery (CD) and continuous deployment (CD) practices. It can automatically trigger deployments to different environments (e.g., development, staging, production) based on predefined criteria.

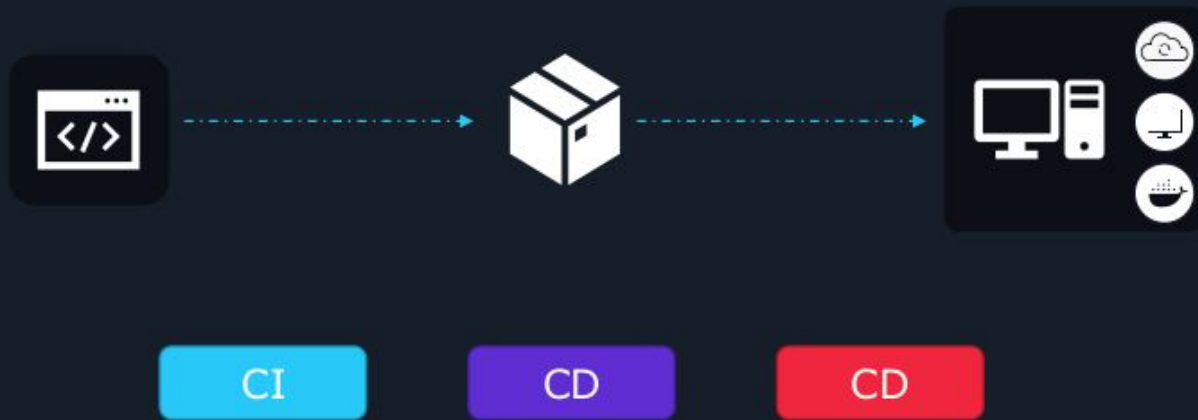
Monitoring and Reporting:

Jenkins offers detailed logs and reports, making it easy to monitor the status of builds and deployments. These insights help teams identify issues quickly and improve the development process.

Continuous Integration and Continuous Delivery/Deployment (CICD)

CI/CD in simple words is a process to take a code, package it up and deploy it to a system that can be serverless, a VM, or a container. CI/CD can be broken down into 3 steps:

- CI – Continuous Integration
- CD – Continuous Delivery
- CD – Continuous Deployment





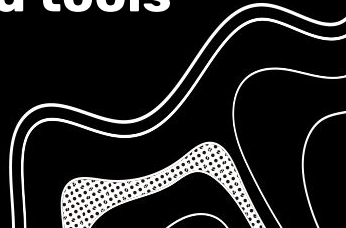
Key Concepts

1. Jobs

Jobs are the fundamental units of work in Jenkins. A job represents a task or a series of tasks that need to be executed. Jobs can be configured to run at specific times or in response to certain events.

2. Builds

In Jenkins, a build refers to the process of compiling source code, running tests, and creating executable artifacts. Jenkins can handle a variety of build tools and programming languages.



Key Concepts

3. Nodes

Nodes are the machines on which Jenkins runs jobs. There are two types of nodes in Jenkins:

Master Node: The master node is the primary Jenkins instance where the Jenkins server is running. It manages the job execution and distributes work to slave nodes.

Slave Nodes: Slave nodes are additional machines that can be used to offload build and test work from the master node. This helps distribute the workload and improve performance.

Key Concepts

4. Plugins

Jenkins has a vast ecosystem of plugins that extend its functionality. Plugins allow you to integrate Jenkins with various tools, version control systems, and cloud platforms.

5. Pipelines

Jenkins Pipelines are a way to define complex automation workflows as code. You can create pipelines using the Pipeline DSL (Domain-Specific Language) to describe the stages of your CI/CD process.

Master Node

The Jenkins master node is the central server responsible for managing jobs and distributing work to slave nodes.

It hosts the Jenkins web interface, where users can configure and monitor jobs.

The master node stores job configurations, build logs, and other essential data.



Slave Nodes

Slave nodes are additional machines that execute jobs offloaded by the master.

Slaves can run on various operating systems and architectures, allowing for flexibility in building and testing environments.

They are configured to connect to the master node for job execution.

Executors

Executors represent the number of concurrent job executions that a node can handle.

Each slave node can have one or more executors, allowing multiple jobs to run concurrently on the same node.

Jenkins Home Directory

This directory stores Jenkins configuration, job configurations, plugins, and build artifacts.

It is essential to back up this directory regularly to prevent data loss.



Jenkinsfile

Jenkinsfile is a text file that contains definitions. This could be templates or instructions. It tells pipelines what they should be doing and what services and plugins they should be interacting with.

Components of Jenkinsfile:

- 1) Pipeline – The task you are trying to accomplish
- 2) Build Agent – The place where you run your pipeline
- 3) Stages – Staging/Production/UAT
- 4) Steps – Work done in the pipeline

The task you are trying to accomplish

Build Agent

Stages

Steps

```
pipeline {  
  agent any
```

```
  stages {  
    stage('Build') {  
      steps {  
        echo 'Building..'  
      }  
    }  
    stage('Test') {  
      steps {  
        echo 'Testing..'  
      }  
    }  
    stage('Deploy') {  
      steps {  
        echo 'Deploying....'  
      }  
    }  
  }  
}
```


Use-cases

Containerized Workflows:

Use Case: Jenkins can build and deploy containerized applications and orchestrate container-based workflows using tools like Docker and Kubernetes.

Benefits: Supports modern microservices architectures, simplifies deployment in container environments, and ensures consistency.



Use-cases

Security Scanning and Compliance:

Use Case: Jenkins can integrate security scanning tools to assess code and container images for vulnerabilities and compliance violations.

Benefits: Enhances security posture, identifies and mitigates risks early in the development process.



Use-cases

Custom Scripting and Automation:

Use Case: Jenkins allows for custom scripting and automation of a wide range of tasks, making it adaptable to unique project requirements.

Benefits: Provides flexibility to cater to specific needs, automates manual tasks, and reduces repetitive work.



Conclusion

Jenkins is a versatile and powerful tool that plays a pivotal role in modern software development and IT operations.

Its ability to automate, integrate, and orchestrate various tasks and processes makes it indispensable for organizations striving to improve their development workflows and achieve a higher level of efficiency and reliability.

Lets MAKE OUR HANDS DIRTY !!!

