Jenkins for Beginners

Agenda

What is Jenkins?

Why use Jenkins?

Jenkins Architecture

Installation (Docker Desktop)

Jenkins Concepts

Basic to Intermediate Pipelines

Hands-On Lab

What is Jenkins?

Jenkins is an open-source automation server.

Helps automate parts of software development.

Used for building, testing, and deploying code.

Why Use Jenkins?

Continuous Integration/Delivery

Easy plugin ecosystem

Open source and widely adopted

Supports distributed builds

Jenkins Architecture

Master-Agent architecture

Jenkins Master schedules jobs, manages agents

Agents execute build tasks

Key Terminologies

Jobs: Configurable tasks in Jenkins that automate processes like building, testing, or deploying code.

Pipelines: A series of automated steps defined in code (often via Jenkinsfile) to manage and visualize complex workflows.

Nodes: Machines or agents in Jenkins that execute jobs, distributing workload across environments.

Plugins: Extensions that add functionality to Jenkins, enabling integration with tools or custom features.

Workspace: A directory on a node where Jenkins stores job-related files and data during execution

Installing Docker Desktop

Visit docker.com and download Docker Desktop

Install and run Docker

Ensure Docker is running and working

Installing Jenkins on Docker

- Run *docker run -p 8080:8080 -p 50000:50000 jenkins/jenkins:lts*
- Access via *http://localhost:8080*
- Initial Admin Password and Setup Wizard

First Look at Jenkins Dashboard

Main View: Lists all jobs with their names, status (e.g., success, failure, in-progress), last build details, and build history.

Build Queue: Shows pending or running jobs waiting for execution on available nodes.

Build Executor Status: Displays active nodes and their current task load, indicating resource usage.

Tabs/Links: Provides quick access to "New Item" (create jobs), "Manage Jenkins" (system settings), "My Views" (custom job filters), and "Build History" (timeline of all builds).

Search Bar: Allows searching for specific jobs or configurations.User Interface Options: Includes links for user account settings, logout, and Jenkins version information.

Jenkins Plugin Ecosystem

Git Plugin: Enables integration with Git repositories for source code management, supporting cloning, fetching, and webhooks.

Pipeline Plugin: Provides tools for creating and managing scripted or declarative pipelines using Jenkinsfile for complex, automated workflows.

Docker Plugin: Allows Jenkins to interact with Docker, enabling building, running, and managing containers as part of jobs or nodes.

Blue Ocean Plugin: Offers a modern, visual interface for pipelines, improving user experience with detailed build insights.

Credentials Plugin: Securely manages and stores credentials (e.g., usernames, tokens) for use in jobs and integrations.

Slack Notification Plugin: Sends build status notifications to Slack channels for real-time team updates.

JUnit Plugin: Publishes and visualizes test results from JUnit or similar frameworks in job reports.

Introduction to Pipelines

What is a Jenkins Pipeline?

A **Jenkins Pipeline** is a suite of plugins and tools in Jenkins that enables defining and automating continuous integration and delivery workflows as code, providing a structured, repeatable, and visual way to manage complex build, test, and deployment processes.

Scripted vs Declarative

Syntax: Scripted is free-form Groovy; Declarative uses a strict, user-friendly structure.

Ease of Use: Declarative is easier for beginners; Scripted suits complex, custom needs.

Error Handling: Declarative has built-in constructs (e.g., post block); Scripted requires manual try-catch.

Tooling: Declarative integrates better with Blue Ocean and syntax validators.

Jenkinsfile Introduction

A **Jenkinsfile** is a text file (written in Groovy) that defines a Jenkins Pipeline, stored in the source code repository alongside the project code, enabling version control and collaboration.

Purpose: Codifies the entire CI/CD workflow, making it reproducible, trackable, and reviewable.

Location: Typically placed in the root of the repository (e.g., Jenkinsfile).

Create Your First Pipeline

Jenkinsfile Basics

Key Components

pipeline:

The root block that encapsulates the entire pipeline definition.

Required for Declarative Pipelines; all other blocks (e.g., agent, stages) are nested inside it.

Example: pipeline { ... }

agent:

Specifies where the pipeline or its stages will run (e.g., on a specific node, Docker container, or any available agent).

Common options: any (any available agent), none (define agents per stage), or specific labels (e.g., label 'linux').

Example: agent any or agent { docker { image 'node:16' } }

stages:

A block containing one or more stage blocks, each representing a distinct phase of the pipeline (e.g., Build, Test, Deploy).

Stages are executed sequentially and displayed visually in Jenkins' Stage View or Blue Ocean.

Example: stages { stage('Build') { ... } }

steps:

Nested within a stage, this block defines the actual tasks or commands to execute (e.g., shell scripts, plugin actions).

Common steps include sh (run shell commands), echo (print messages), or plugin-specific actions like archiveArtifacts.

Example: steps { sh 'npm install' }