III B.Tech I Sem CSE

23CS503 : DevOps

Unit-IV

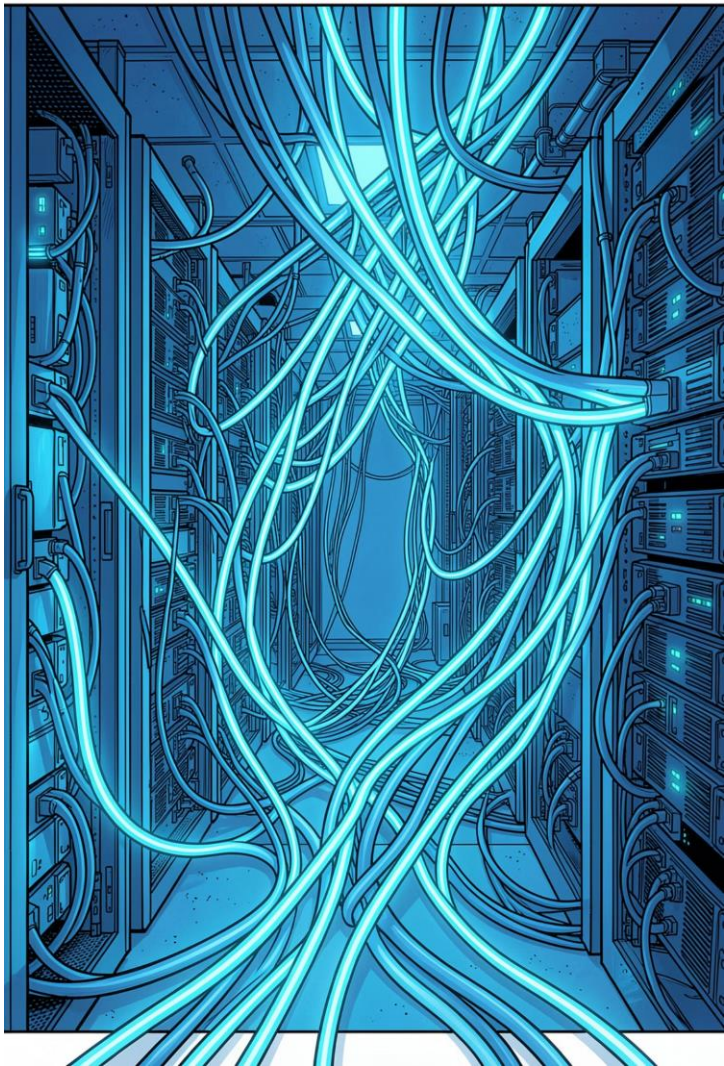
Integrating the System

Mr. P. Thirupathi
Assistant Professor, Department of CSE NRCM

Integrating the System

DEVOPS (23CS503) · UNIT – IV

Department of Computer Science and Engineering · Narsimha Reddy Engineering College (Autonomous)



Learning Outcomes

After completing this unit, you will be able to:

1

Build Systems

Understand how build systems automate compilation, testing, and packaging

2

Jenkins Architecture

Explain the master-agent model, plugins, and pipeline workflow

3

Build Pipelines

Configure and manage CI/CD pipelines in Jenkins

4

Build Dependencies

Manage external libraries and packages using modern tools

5

Infrastructure as Code

Implement IaC practices for reproducible environments

6

Quality Metrics

Analyze test coverage, code quality, and build health

Introduction to System Integration

System Integration is the process of combining different software components and modules into a unified, functioning system — enabling teams to deliver software faster and more reliably.

Seamless Communication

Components interact reliably across services and layers

Automated Builds

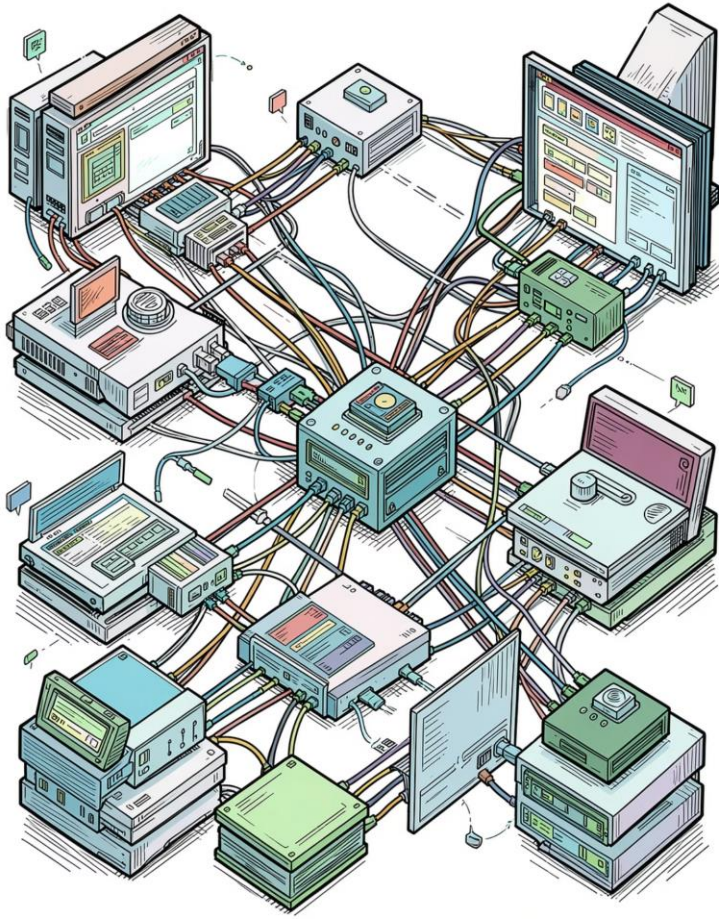
Code is compiled and packaged without manual intervention

Faster Deployment

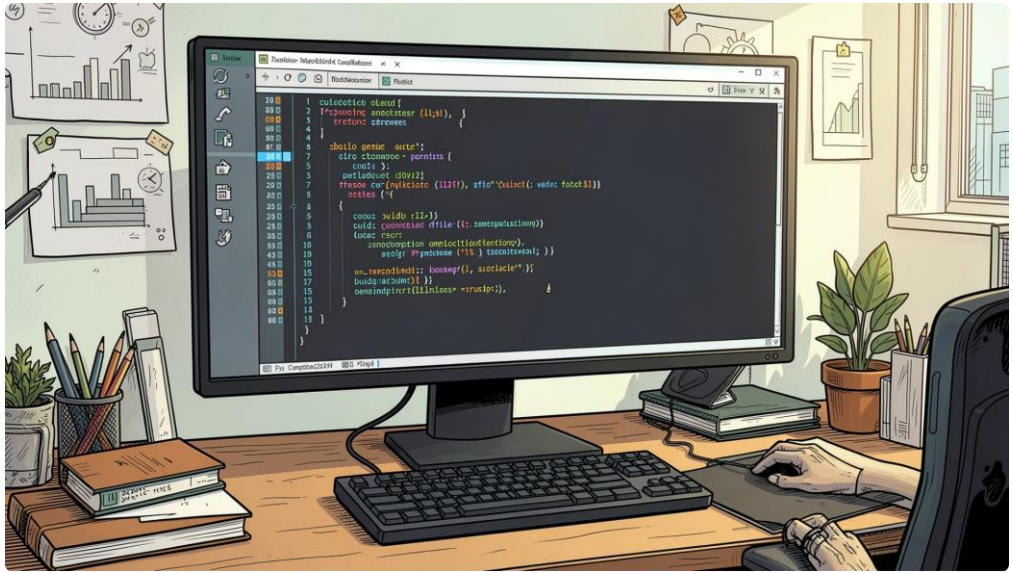
Changes reach production quickly and safely

Improved Quality

Automated testing catches defects early in the cycle



Build Systems



What is a Build System?

A build system automates the entire software delivery workflow — from source code to deployable artifact.

Popular Tools: Maven, Gradle, Ant, Jenkins

Compilation

Translates source code into executable binaries

Testing

Runs unit and integration tests automatically

Packaging

Bundles artifacts into JAR, WAR, or Docker images

Deployment

Pushes artifacts to servers or registries

Why Build Systems Matter

Manual builds are slow, error-prone, and inconsistent. Build systems eliminate these problems by enforcing automation at every stage of development.

Automation

Repetitive tasks run without human input

Faster Development

Teams ship features more rapidly

Fewer Errors

Consistent processes reduce human mistakes

Continuous Integration

Every commit triggers an automated build

Consistent Results

Same output regardless of developer or machine

Jenkins Build Server

Jenkins is the world's most popular open-source automation server, powering CI/CD pipelines for millions of projects worldwide.

Continuous Integration (CI)

Automatically builds and tests code on every commit

Continuous Delivery (CD)

Deploys validated builds to staging or production

Build Automation

Handles compilation, packaging, and artifact management

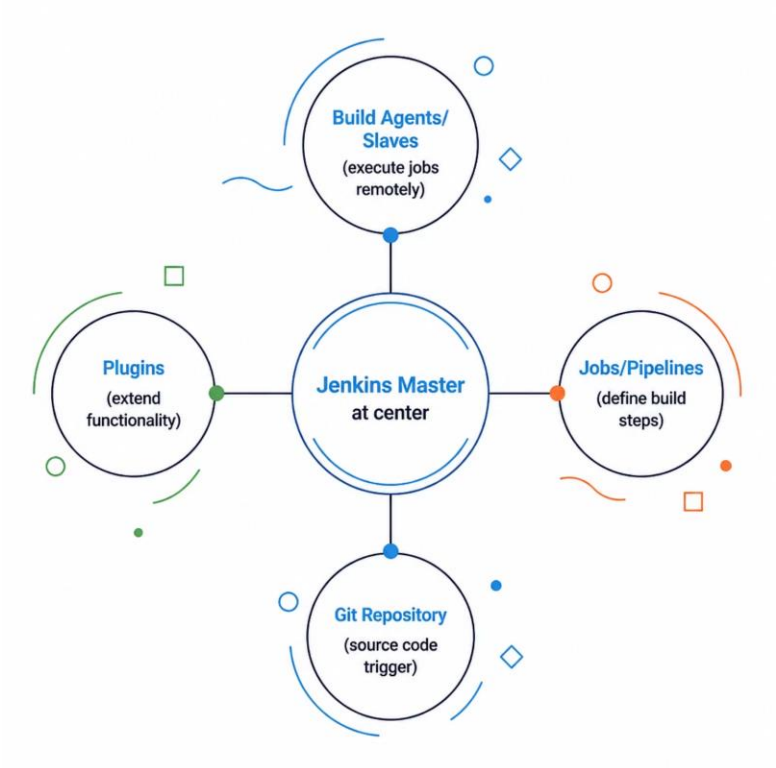
Extensible via Plugins

1,500+ plugins integrate with virtually any tool

Distributed Builds

Scales across multiple agents for parallel execution

Jenkins Architecture



Core Components

1 Jenkins Master

Orchestrates jobs, schedules builds, and manages agents

2 Build Agents (Slaves)

Execute build tasks distributed across machines

3 Plugins

Add integrations for Git, Docker, Maven, and more

4 Jobs & Pipelines

Define the sequence of build, test, and deploy steps

Typical Workflow: Developer → Git → Jenkins → Build → Test → Deploy

Managing Build Dependencies

What are Build Dependencies?

External libraries, frameworks, and packages that your application requires to compile and run. Proper dependency management ensures reproducible builds across all environments.

Key Advantages

Version Control

Pin exact library versions for consistency

Easy Updates

Upgrade dependencies with a single command

Reusability

Share configurations across projects and teams

Transitive Resolution

Automatically fetch nested dependencies

Maven
Java projects

Gradle
Java/Kotlin

npm
JavaScript

pip
Python

Jenkins Plugins

Plugins are the heart of Jenkins' extensibility — they integrate third-party tools, add new features, and customize the pipeline experience.

Git Plugin

Connects Jenkins to Git repositories for source control triggers

Docker Plugin

Builds and pushes Docker images as part of the pipeline

Maven Plugin

Invokes Maven goals for Java project builds

Pipeline Plugin

Enables Jenkinsfile-based declarative and scripted pipelines

Email Notification

Sends build status alerts to developers via email

Jenkins File System Layout

Directory Structure

All Jenkins data is stored under `JENKINS_HOME/`. Understanding this layout helps with backup, migration, and troubleshooting.

```
JENKINS_HOME/
├── jobs/          # Job configs & build history
├── plugins/       # Installed plugin files
├── users/         # User accounts & credentials
├── workspace/     # Active build workspaces
└── logs/          # System & build logs
```

Why It Matters

Backup & Recovery

Backing up `JENKINS_HOME/` preserves all jobs and configs

Migration

Copy the directory to move Jenkins to a new server

Debugging

Logs and workspace files help diagnose build failures

Key Takeaway: `JENKINS_HOME/` is the single source of truth for your entire CI/CD setup.