Title: Jenkins – Continuous Integration and Continuous Delivery Tool

1. Introduction to Jenkins

Jenkins is an open-source automation server widely used in DevOps for Continuous Integration (CI) and Continuous Delivery (CD). It helps automate the parts of software development related to building, testing, and deploying, facilitating continuous development and integration.

2. Key Features of Jenkins

Open Source: Free to use and supported by a strong community.

Platform Independent: Java-based; runs on Windows, macOS, and Unix systems.

Extensible: Over 1800 plugins available to integrate with various tools (Git, Maven, Docker, etc.).

Easy Configuration: Web-based GUI for managing projects and pipelines.

Distributed Builds: Supports master-slave architecture for parallel builds.

3. Jenkins Architecture

Components:

Master Node: Controls the Jenkins environment, schedules builds, and manages plugins.

Agent (Slave) Nodes: Perform the actual build tasks on different machines.

Jobs: Tasks to be performed, such as building or testing code.

Plugins: Extend Jenkins functionality for integration with version control, build tools, testing, etc.

4. Jenkins Workflow

Step-by-Step CI/CD Pipeline:

Code Commit: Developers commit code to a version control system like Git.

Trigger Build: Jenkins detects the changes via webhooks or polling.

Build Execution: Jenkins builds the code using tools like Maven or Gradle.

Testing: Automated tests are executed (unit tests, integration tests).

Reporting: Results are reported to the development team.

Deployment: If successful, Jenkins deploys the code to the staging/production environment.

5. Jenkins Pipeline

A **Pipeline** is a suite of plugins supporting integration and implementation of continuous delivery pipelines.

Two Types:

Declarative Pipeline: Simplified and structured syntax using `Jenkinsfile`.

Scripted Pipeline: More flexibility, written in Groovy.

Example (Declarapipeline {

```
  agent any

  stages {

    stage('Build') {

      steps {

        echo 'Building...'

        sh 'mvn clean package'

      }

    }

    stage('Test') {

      steps {

        echo 'Testing...'

        sh 'mvn test'

      }

    }

    stage('Deploy') {

      steps {
```

```
                echo 'Deploying...'

                sh 'scp target/app.jar user@server:/deploy/path'

            }

        }

    }

}tive):
```

6. Common Use Cases

- Continuous integration for software development.
- Automated testing and build verification.
- Scheduled or triggered builds.
- Automated deployment.
- Monitoring and reporting build status.

7. Advantages of Jenkins

- Improves code quality through automated testing.
- Reduces manual effort and chances of human error.
- Fast feedback to developers.
- Easily integrates with popular tools like GitHub, Docker, Kubernetes, etc.

8. Real-World Scenario

In a team using GitHub, Jenkins can automatically pull code on every commit, run tests, build a Docker image, and deploy it to Kubernetes — all without manual intervention