CD Tools (Jenkins, GitLab CI, Spinnaker)

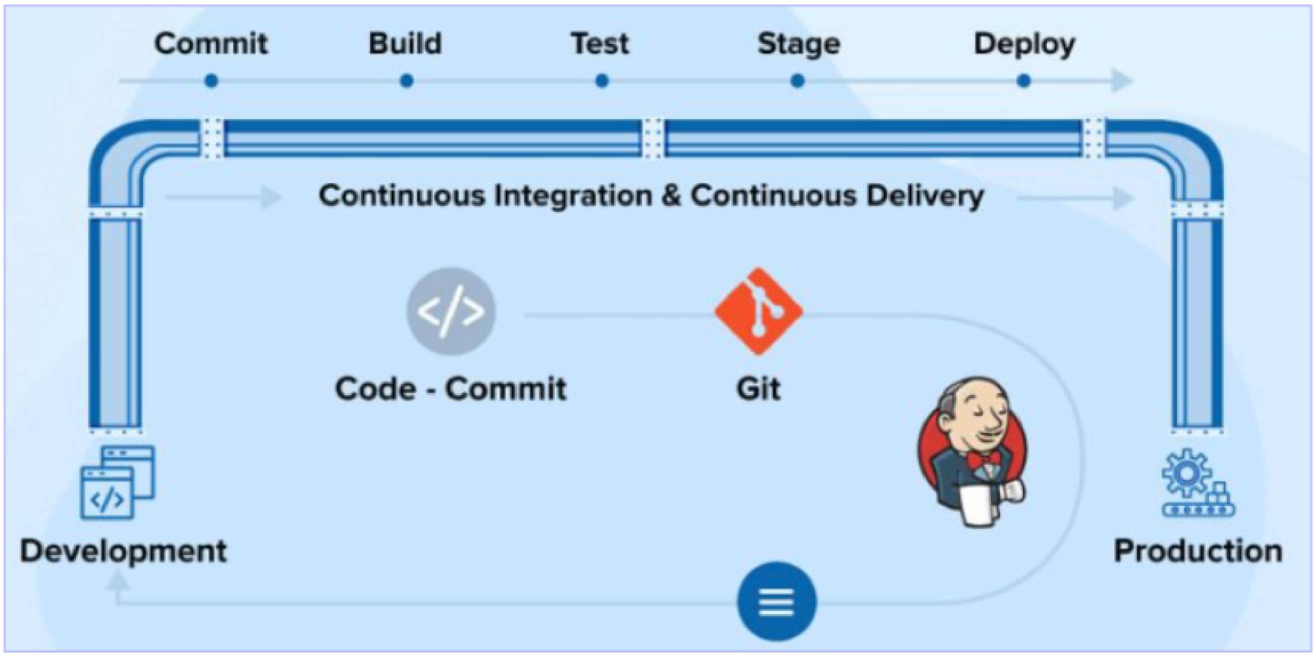
Lesson Plan

PW SKILLS

Continuous Deployment (CD) Tools like Jenkins, GitLab CI, and Spinnaker are crucial for automating and streamlining the software release process. Here's a brief overview of each:

1. Jenkins:

Jenkins is an open-source automation server that helps automate parts of the software development process related to building, testing, and deploying, facilitating continuous integration and continuous delivery (CI/CD).



Key Components

- **Jenkins Master:** The central control unit that manages and coordinates jobs, schedules build tasks, and dispatches them to appropriate agents. It also provides the user interface and monitors job status.
- **Jenkins Agents (Slaves):** Machines that perform the actual build tasks. Agents can be on different operating systems, allowing for a wide range of environments.

Jenkins Pipeline

- **Pipeline as Code:** Jenkins allows defining build processes with code using a Jenkinsfile. This file describes the stages and steps of the pipeline, enabling version control and easier replication of build environments.

- **Declarative vs. Scripted Pipelines:**
- **Declarative:** A simpler, structured approach with predefined syntax.
- **Scripted:** More flexible and powerful, but requires deeper Groovy scripting knowledge.

Plugin Ecosystem

- **Extensive Library:** Jenkins has over 1,500 plugins, making it possible to integrate with almost any tool or service, including SCM tools (Git, SVN), build tools (Maven, Gradle), and cloud services (AWS, GCP).

- **Common Plugins:**

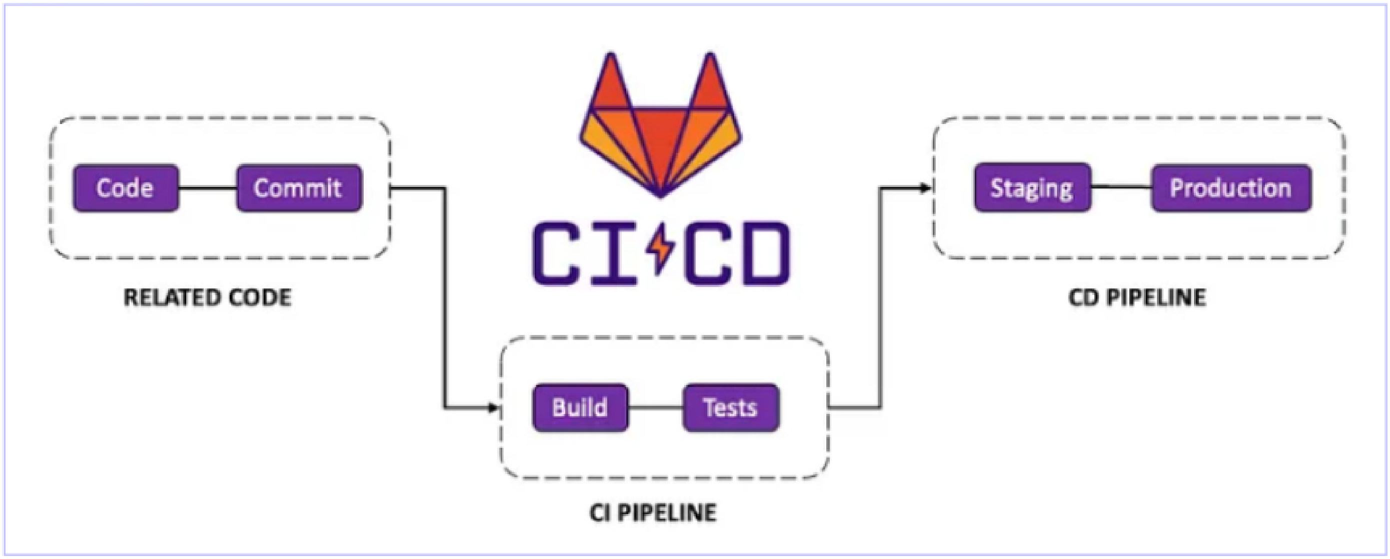
  - **Git Plugin:** For integrating with Git repositories.
  - **Pipeline Plugin:** Enables defining and managing Jenkins pipelines.
  - **Docker Plugin:** For building and deploying Docker containers.
  - **Blue Ocean:** A modern interface for Jenkins with a focus on pipelines.

Jenkins in a CI/CD Pipeline

- **Integration with Version Control:** Automatically trigger builds based on commits or pull requests.
- Build Automation: Compile code, run tests, and package applications.
- **Artifact Management:** Store and manage build artifacts using tools like Artifactory or Nexus.
- **Automated Deployment:** Deploy applications to staging or production environments automatically or with manual approval gates.

2. GitLab CI:

GitLab CI is a robust and integrated continuous integration and delivery tool that's part of the broader GitLab platform. It allows you to automate your build, test, and deployment pipelines directly from your Git repository. Here's a more detailed look at GitLab CI:



Key Concepts:

Pipelines:

- A pipeline in GitLab CI consists of a series of stages that define the lifecycle of a CI/CD process (e.g., build, test, deploy).
- Pipelines are defined in a .gitlab-ci.yml file located in the root directory of your repository.
- Each pipeline can run multiple jobs in parallel or sequentially, depending on the configuration.

Jobs:

- Jobs are individual tasks within a stage, such as compiling code, running tests, or deploying to a server.
- Jobs run in a specified environment (e.g., Docker container) and can be configured with different parameters like scripts, artifacts, and tags.

Stages:

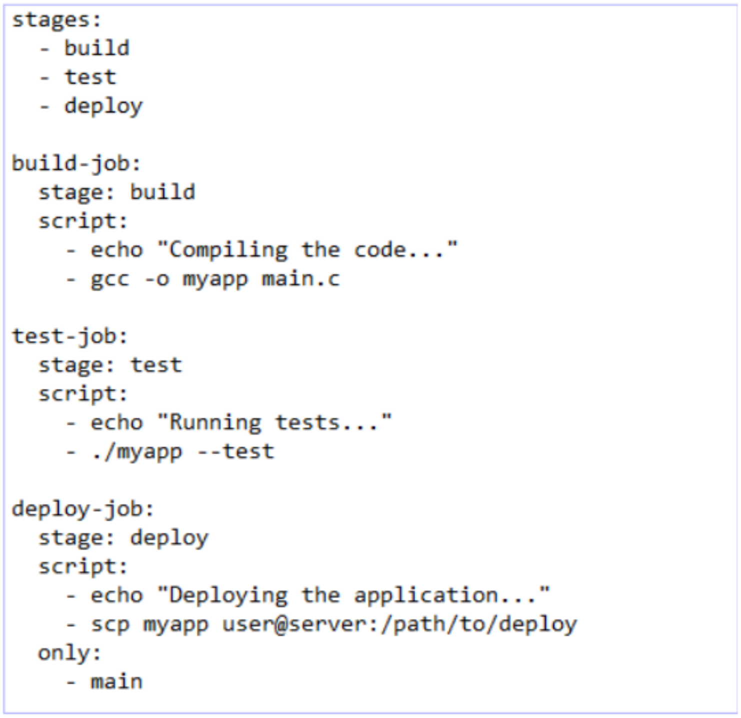
- Stages organize jobs within a pipeline. Jobs in a stage run concurrently, and the next stage begins only when all jobs in the current stage have completed successfully.
- Common stages include build, test, and deploy, but you can customize stages to suit your workflow.

Runners:

- GitLab Runners are agents that execute the jobs defined in your pipeline. They can run on various environments, such as virtual machines, Docker containers, or Kubernetes clusters.
- Runners can be shared (available for all projects) or specific (dedicated to a particular project or group).

.gitlab-ci.yml File

This file is the heart of GitLab CI, where you define the pipeline structure. Here's a simple example:



```
stages:
  - build
  - test
  - deploy

build-job:
  stage: build
  script:
    - echo "Compiling the code..."
    - gcc -o myapp main.c

test-job:
  stage: test
  script:
    - echo "Running tests..."
    - ./myapp --test

deploy-job:
  stage: deploy
  script:
    - echo "Deploying the application..."
    - scp myapp user@server:/path/to/deploy
  only:
    - main
```

Key Features:

- Built-in CI/CD within the GitLab ecosystem.
- Pipelines as code (using .gitlab-ci.yml).
- Supports Docker and Kubernetes integrations.
- Security testing and monitoring within CI/CD.

3. Spinnaker:

Spinnaker is a powerful, open-source continuous delivery platform that focuses on automating and managing the deployment of applications across various cloud environments. Developed by Netflix and later open-sourced, Spinnaker excels in multi-cloud deployments and offers sophisticated deployment strategies that help ensure smooth, reliable releases.

Key Features of Spinnaker

Multi-Cloud Deployment Support:

- Spinnaker is designed to work seamlessly with multiple cloud providers, including AWS, Google Cloud Platform (GCP), Microsoft Azure, Kubernetes, and OpenStack. This makes it a great choice for organizations using or transitioning to a multi-cloud strategy.

Deployment Strategies:

- **Canary Releases:** Deploy a new version to a small subset of users before rolling it out to the entire user base, allowing for real-time monitoring and feedback.
- **Blue/Green Deployments:** Keep two separate environments (blue and green), with one serving live traffic while the other is updated. After the update, traffic is switched to the updated environment.
- **Rolling Updates:** Gradually replace instances of the current version with the new version, reducing the risk of downtime.

Pipeline Management:

- Spinnaker's pipelines are highly customizable and can be configured to perform a series of automated steps such as building, testing, and deploying code. It integrates with other CI/CD tools like Jenkins to manage the entire process from code commit to production.

Application Management:

- Spinnaker provides a unified view of all your applications and their deployments across different environments. This helps teams monitor the status of their applications in real time.

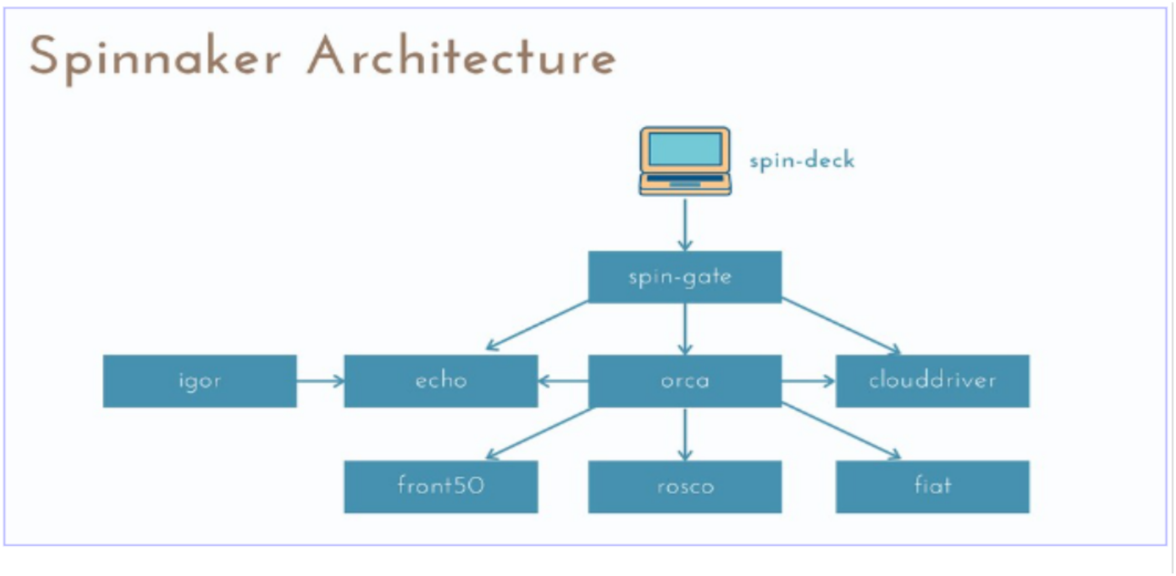
Monitoring and Rollback:

- Spinnaker integrates with monitoring tools like Prometheus and Datadog to keep track of application performance during and after deployment. If issues are detected, Spinnaker can automatically trigger a rollback to the previous stable version, minimizing downtime.

Extensibility:

- Spinnaker's microservices architecture allows it to be extended with custom stages, plugins, and integrations. This flexibility enables organizations to tailor the platform to their specific needs.

Architecture Overview



Spinnaker's architecture is composed of several microservices, each responsible for a specific aspect of the deployment process:

- **Orca:** Manages the orchestration of pipelines and workflows.
- **Clouddriver:** Provides cloud provider-specific functionality, such as creating, deleting, and managing resources.
- **Deck:** The UI for Spinnaker, allowing users to manage applications, pipelines, and deployments visually.
- **Gate:** The API gateway for interacting with Spinnaker.
- **Igor:** Integrates with CI servers like Jenkins to trigger pipelines based on CI events

.

- **Front50:** Stores the metadata of applications, pipelines, and other Spinnaker objects.
- **Rosco:** Manages the creation of machine images for cloud deployments.
- **Echo:** Manages notifications and triggers within Spinnaker.