CI/CD Pipeline Development

WITH GITHUB ACTIONS

What is a Pipeline?

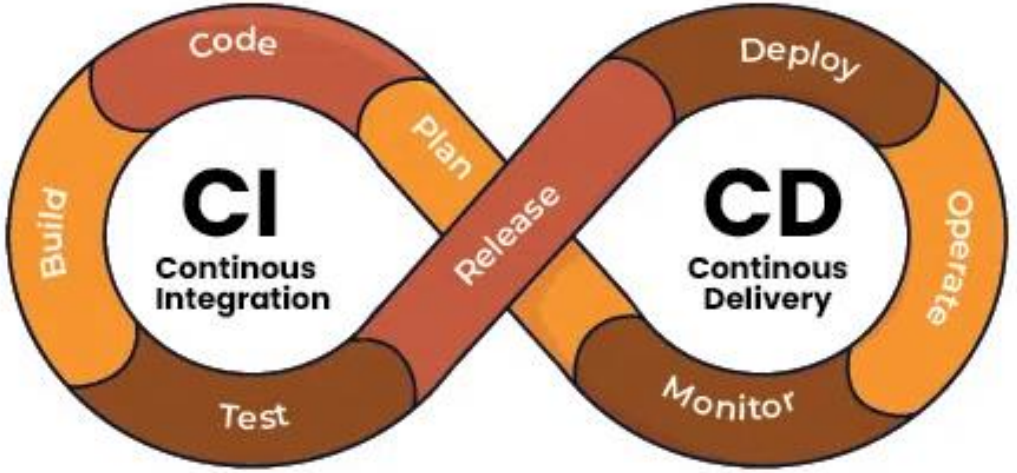
A **Pipeline** is a series of automated steps or processes that allow for the consistent and reproducible flow of code through various stages, from development to deployment. These stages can include:

- **Building the application**
- **Running tests**
- **Deployment** to staging/production environments

Pipelines ensure the consistency, speed, and quality of the deployment process by automating tasks that are usually done manually.

CI/CD Pipeline

System Design



What is CI/CD?

- **CI (Continuous Integration):**
  - Automates the process of merging code changes into the main branch and running tests automatically to ensure the changes don't break the application.
- **CD (Continuous Deployment or Continuous Delivery):**
  - **Continuous Deployment:** Automatically deploys the application to production after successful tests.
  - **Continuous Delivery:** Automates the build and testing process but requires manual approval for deployment.
- GitHub Actions provides a robust framework to implement CI/CD pipelines.

Continuous Integration (CI)

- **Definition**: The practice of merging all developers' working copies to the main branch frequently (usually several times a day).
- **Main Purpose**: To detect issues early, improve software quality, and streamline the merging process.
- **Process**:
  - Developers commit code to the repository.
  - A CI pipeline automatically triggers.
  - The pipeline runs tests and builds the application.
  - If successful, the code is merged to the main branch.

Continuous Deployment (CD)

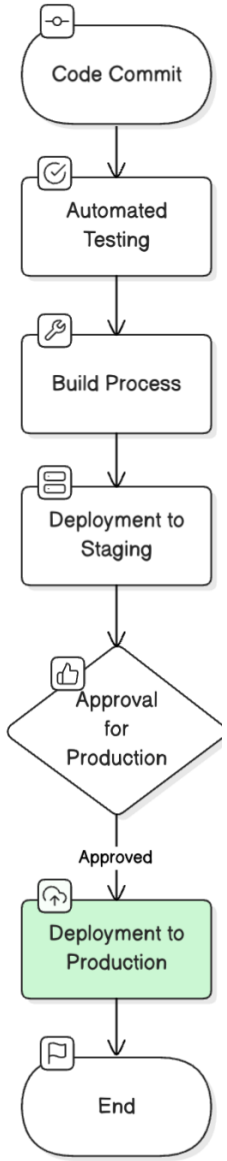
- **Definition**: Automates the entire deployment process so that every change is automatically deployed to production, ensuring that software can be delivered more frequently.
- **Process**:
  - After successful testing, the build gets deployed automatically to a staging or production environment.
  - Any change that passes all testing phases is considered ready for production.

In **Continuous Delivery (CD alternative)**the deployment to production requires manual approval (e.g., staging review).

CI/CD Pipeline: Key Stages

A typical **CI/CD Pipeline** consists of the following key stages:

1. **Code Commit**: Developers commit their code to a shared repository.
2. **Test**: Run automated tests (unit tests, integration tests, etc.).
3. **Build**: The system builds the application, such as compiling the code or bundling assets.
4. **Deploy to Staging**: Deploy the application to a staging environment for QA and user acceptance testing. Depending on type of pipeline it either deploys to production or waits for approval.
5. **Deploy to Production**: Once the application passes all tests and checks, it is deployed to production.



What is GitHub Actions?

GitHub Actions is an automation tool built into GitHub that enables you to set up CI/CD workflows. It allows you to define tasks such as **build**, **test**, **deploy**, and **automate other GitHub tasks** (e.g., creating releases).

GitHub Actions offers:

- Support for **multiple runners** (macOS, Windows, Linux) which can be considered virtual machines when the processes are run.
- **Integration with GitHub's features**, such as pull requests, issues, and releases.

Key Components of a GitHub Actions Workflow

1. **Workflow**: A YAML file that defines the automated tasks to be run on GitHub events.
    1. Stored in ***.github/workflows/*** directory.
    2. Triggered by events such as **push**, **pull request**, or manual triggers.
2. **Job**: A set of steps that run on a runner running in parallel or sequentially.
    1. Each job can run on a different operating system.
3. **Step**: A single task that runs as part of a job.
    1. Steps can use pre-built **GitHub Actions** or run custom commands.
4. **Runner**: A machine that runs the jobs like hosted runners (macOS, Linux, Windows).
5. **Action**: A reusable unit of code that performs a task in a workflow. Examples include:
    1. **setup-node**: Setup a Node.js environment.
    2. **checkout**: Checkout the repository code.
    3. **upload-artifacts**: Upload build artifacts.

Workflow Overview

This GitHub Actions workflow consists of:

1. **Triggering Conditions**: Runs on pushes to the main branch, version tags (v*), and pull requests. Workflow can also be dispatched manually from the Github Actions tab.
2. **Build Process**: Installs dependencies, sets up environments, and builds the Tauri app.
3. **Testing**: Runs basic tests to validate the application.
4. **Release Process**: Creates a GitHub release and uploads the binaries.