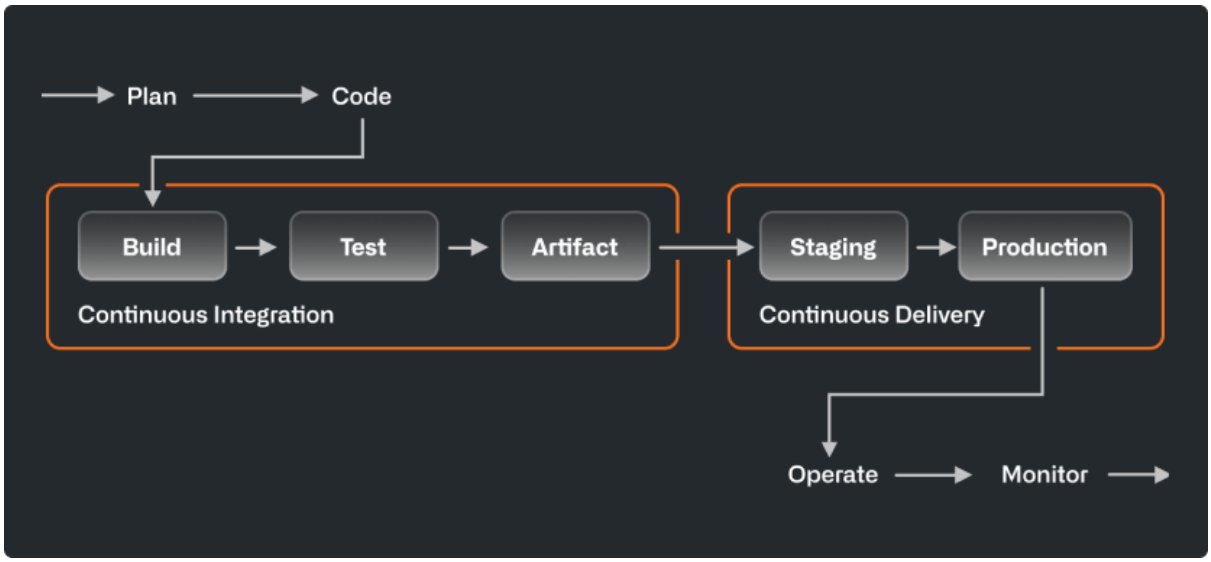
CI/CD PIPELINE

CI/CD stands for Continuous Integration and Continuous Deployment (or Continuous Delivery). It's a set of practices and tools designed to improve the software development process by automating builds, testing, and deployment, enabling you to ship code changes faster and reliably.

- **Continuous integration (CI):** Automatically builds, tests, and integrates code changes within a shared repository.
- **Continuous delivery (CD):** Automatically delivers code changes to production-ready environments for approval.
- **Continuous deployment (CD):** Automatically deploys code changes to customers directly.

CI/CD comprises of continuous integration and continuous delivery or continuous deployment. Put together, they form a “CI/CD pipeline”—a series of automated workflows that help DevOps teams cut down on manual tasks.



By automating CI/CD throughout development, testing, production, and monitoring phases of the software development lifecycle, teams are able to develop higher quality code, faster and more securely. Automated testing also allows dependencies and other issues to be identified earlier in the software development lifecycle, saving time later. Although it's possible to manually execute each of the steps of a CI/CD pipeline, the true value of CI/CD pipelines is realized through automation.

Continuous delivery vs. Continuous deployment

In a CI/CD pipeline that uses continuous delivery, automation pauses when developers push to production. A human still needs to manually sign off before final release, adding more delays. On the other hand, continuous deployment automates the entire release process. Code changes are deployed to customers as soon as they pass all the required tests.

Continuous deployment is the ultimate example of DevOps automation. Since continuous deployment relies on rigorous testing tools and a mature testing culture, most software teams start with continuous delivery and integrate more automated testing over time.

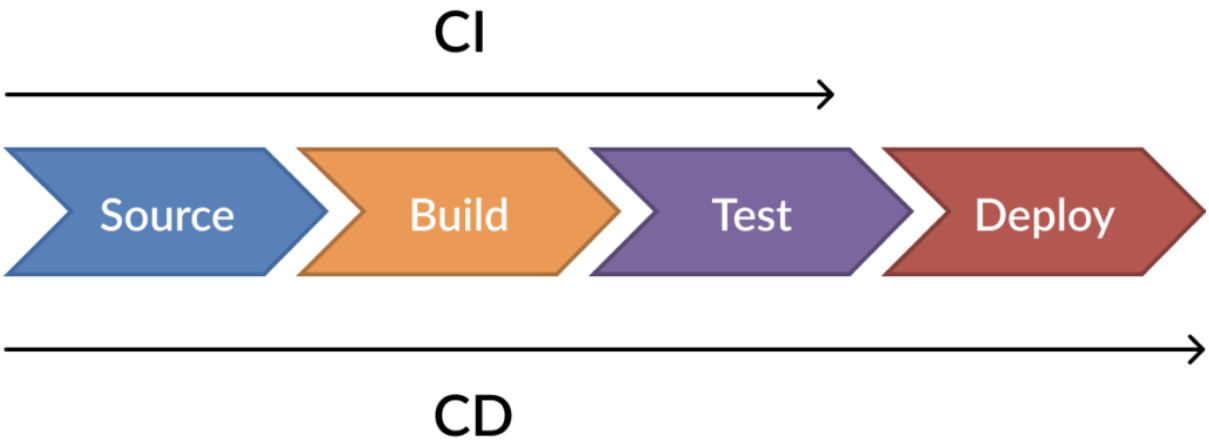
How do CI/CD pipelines relate to DevOps?

Both CI/CD and DevOps focus on automating processes of code integration, thereby speeding up how an idea (like a new feature, a request for enhancement, or a bug fix) goes from development to deployment in a production environment where it can provide value to the user. Developers, usually coding in a standard development environment, work closely with IT operations to speed software builds, tests, and releases—without sacrificing reliability.

How can CI/CD pipelines support security?

Without proper security, a rapid development and deployment process can expose an organization to risks. Common risks can include revealing sensitive data to outside sources, incorporating insecure code or third-party components, or exposing source code repositories or build tools to unauthorized access. Automated checks and testing within a CI/CD pipeline can safeguard code and prevent vulnerabilities in software delivery. Incorporating security into a pipeline helps to protect code from attack, prevent data leaks, comply with policies, and ensure quality assurance. Identifying and mitigating vulnerabilities throughout the development cycle assures that code changes are thoroughly tested and adhere to security standards before they are deployed to production.

CI/CD stages:



Source:

The source stage serves as the foundation of the CI/CD pipeline, where developers push code changes to a version control system like Git. This stage manages code versions, enables branching strategies (e.g., GitFlow), and triggers the pipeline on commits or pull requests. It may also include initial checks like linting to ensure coding standards are followed.

Build:

In the build stage, the source code is compiled and packaged into deployable artifacts (e.g., JAR, WAR, Docker image). It handles dependency resolution, asset bundling, and runs unit tests or static code analysis. If the build or tests fail, the pipeline stops, promoting early detection of issues—following the 'fail fast' principle.

Test:

The test stage runs automated tests to verify the application's functionality, performance, security, and integration between components. This includes unit tests, integration tests, UI/functional tests, and load/security testing to ensure the product meets quality standards before deployment.

Deploy:

The deploy stage releases the tested build to the production environment through automated processes. It may involve pushing code to servers, containers, or cloud platforms. Post-deployment smoke tests ensure the app works as expected, marking the final step where the software becomes available to end users.

Advantages of CI/CD pipelines:

- **Automation**: A good CI/CD workflow automates builds, testing, and deployment so you have more time for code.
- **Transparency**: Logs, visual workflow builders, and deeply integrated tooling make it easier for developers to troubleshoot, understand complex workflows, and share their status with the larger team.
- **Speed**: CI/CD contributes to your overall DevOps performance, particularly speed.

- **Resilience**: When used with other approaches like test coverage, observability tooling, and feature flags, CI/CD makes software more resistant to errors.
- **Security**: Automation includes security. With DevOps gaining traction, a future-proof CI/CD pipeline has checks in place for code and permissions, and provides a virtual paper trail for auditing security breaches, non-compliance events.
- **Scalability**: A robust CI/CD setup should effortlessly expand with your growing development team and project complexity. This means it can efficiently handle increased workloads as your software development efforts grow, maintaining productivity and efficiency.