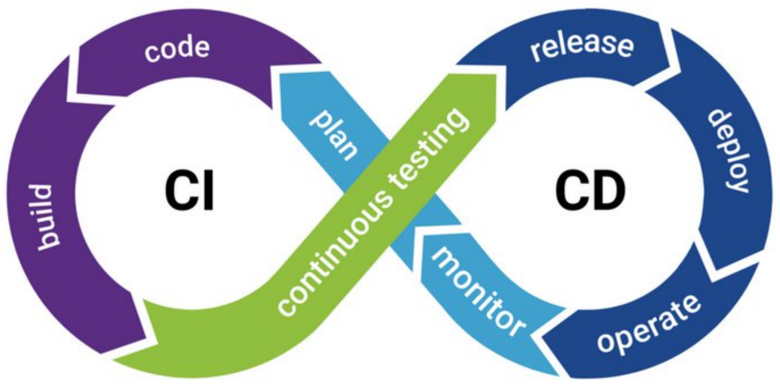
WHAT IS A CI/CD PIPELINE?



A continuous integration and continuous deployment (CI/CD) pipeline is a series of established steps that developers must follow in order to deliver a new version of software. CI/CD pipelines are a practice focused on improving software delivery throughout the software development life cycle via automation.

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.

CI/CD pipelines have advantages for software organizations that use virtual machines as well as container-based cloud native applications. With the ability to more quickly integrate updates and changes to code, teams can respond to user feedback and business changes frequently and effectively, leading to positive outcomes for end users.

HOW DO CI/CD PIPELINES RELATE TO DEVOPS?

CI/CD pipelines, which streamline and accelerate software development, are a reflection of DevOps methodology, a set of ideas and practices that fosters collaboration between developers and IT operations teams.

The CI side of CI/CD refers to continuous integration, which includes building, testing, and merging code. The CD side can stand for continuous delivery, which includes automatically releasing software to a repository. CD can also stand for continuous deployment, which adds the step of automatically deploying software to production.

A CI/CD pipeline guides the process of software development through a path of building, testing, and deploying code. By automating the processes that support CI/CD, development and operations teams can minimize human error and maintain a consistent process for how software is released. Pipelines can include tools for compiling code, unit tests, code analysis, security, and binaries creation. For containerized environments, pipelines will also include tools for packaging the code into a container image to be deployed across a hybrid cloud.

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.

CI/CD BENEFITS

Faster Deployment
Improved Quality
Better Collaboration
Reliable Deployments
Rapid Feedback

Top 5 Benefits of a CI/CD Pipeline:

1. **Faster Development**: A CI/CD pipeline automates manual processes, thereby reducing the time it takes to release new features or bug fixes.

2. **Improved Quality**: Continuous testing and integration catch and address issues early, leading to more reliable software.
3. **Better Collaboration**: Developers can work in smaller, more manageable code increments, reducing conflicts and making collaboration smoother.
4. **Reliable Deployments**: Automation reduces the risk of human error during delivery, leading to more consistent and predictable releases.
5. **Rapid Feedback**: Developers receive quick feedback on their code changes, allowing them to fix issues before they become major problems.

ENVIRONMENTS IN A CI/CD PIPELINE

In a CI/CD pipeline, software passes through multiple environments before reaching production to ensure quality and reliability. The process begins in the development environment, where developers build and test new features locally. Once the code is stable, it moves to the QA environment, where the testing team validates functionality, reports bugs, and ensures fixes are retested until the build is stable. After QA, the application is deployed to the staging or integration environment, which closely replicates real-world scenarios such as order processing or payment flows, allowing testers to verify system behavior under production-like conditions. The next stage is User Acceptance Testing (UAT), where end-users or business stakeholders confirm that the software meets requirements and works as intended. Following UAT, the build is promoted to the Pre-Production Environment (PPE), which mirrors the production setup and serves as the final checkpoint for testing before release. Finally, once all validations are complete, the application is deployed to the Production environment, which is the live system accessed by real users.