# DAY-1 | DevOps Explained

DevOps, short for Development and Operations, is a set of practices that emphasizes collaboration, communication, and automation between software development teams (Dev) and IT operations teams (Ops). It aims to streamline the software development and delivery processes, enabling organizations to deliver high-quality software faster and more reliably.

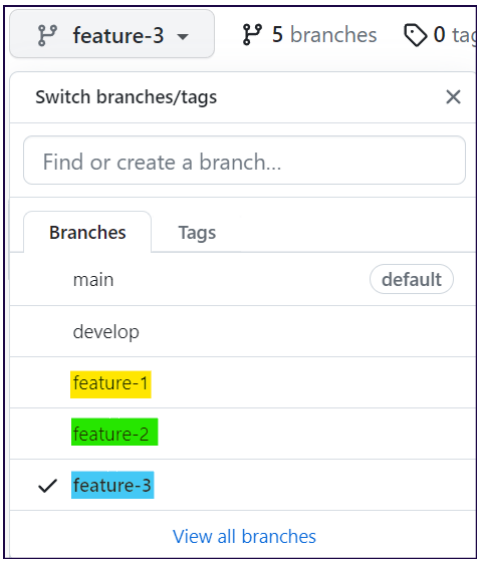
Continuous Integration and Continuous Deployment/Delivery (CI/CD) are essential practices within the DevOps methodology. They focus on automating and accelerating the processes of building, testing, and deploying software.

### Here's a detailed explanation of CI/CD:

## Continuous Integration (CI):

CI is a software development practice that involves frequently integrating code changes from multiple developers into a shared repository. The goal is to detect integration issues early and ensure that the software always remains in a working state. The CI process typically involves the following steps:

### Real-World Scenario for CI : -



Feature-1 → Assigned to Developer1 → Commits code once done

Feature-2 → Assigned to Developer2 → Commits code once done

Feature-3 → Assigned to Developer3 → Commits code once done

When Commit is done by all, then the code for all the features is combined or integrated. This process of adding different features code and building as well as testing at the same time without affecting or causing problems is CI.

1. **Code Commit**: Developers write code and commit their changes to a version control system (such as Git).

2. **Build**: The CI server retrieves the latest code changes from the repository and builds the application. This step involves compiling code, resolving dependencies, and creating executable artifacts.

3. **Automated Tests**: The CI server runs a suite of automated tests to verify the integrity of the codebase. This includes unit tests, integration tests, and other forms of automated testing.

4. **Reporting**: The CI server provides feedback to the development team, indicating whether the build and tests were successful or if any issues were found.

## Continuous Deployment (CD) / Continuous Delivery (CD):

CD is an extension of CI that focuses on automating the deployment and release of software. It enables organizations to release software more frequently, reliably, and with minimal manual intervention. CD encompasses the following stages:

### Real World Scenario:-

Company XYZ is a software development company that has adopted a DevOps approach to enhance their software development and deployment processes. They have a team of developers, testers, and operations professionals working collaboratively to deliver high-quality software products to their customers.

In this scenario, CD is implemented as follows:

1. **Version Control**: The development team uses a version control system like Git to manage their source code. They create different branches for features, bug fixes, and releases.

2. **Automated Build**: Whenever a developer completes a feature or a bug fix, they merge their changes into the main development branch. The source code is then automatically built using a build automation tool like Jenkins.

3. **Continuous Integration**: Once the code is successfully built, it is automatically integrated with the existing codebase. Automated tests are triggered to ensure that the new changes haven't introduced any regressions or issues.

4. **Artifact Repository**: After successful integration and passing the tests, the build artifacts (e.g., compiled binaries, libraries) are stored in an artifact repository such as JFrog Artifactory.

5. **Deployment Pipeline**: The deployment pipeline is set up to automatically deploy the built artifacts to different environments such as development, testing, staging, and production

6. **Automated Testing**: In each environment, automated tests are executed to verify the functionality and performance of the application.

7. **Automated Release**: Once all the automated tests pass in the staging environment, the software is considered ready for release. The release process is automated, and the deployment pipeline promotes the artifacts to the production environment.

8. **Continuous Monitoring**: After the release, the production environment is continuously monitored using tools like Prometheus or ELK (Elasticsearch, Logstash, and Kibana). This helps detect any anomalies, performance issues, or errors in real-time.

9. **Rollbacks and Remediation**: In case any issues arise after the release, an automated rollback process can be triggered to revert to the previous stable version. The DevOps team investigates the root cause and collaborates to fix the issue quickly.

By adopting CI/CD practices, organizations like Company XYZ can achieve the following benefits:

1. **Faster Time-to-Market**: Continuous integration and deployment enable frequent releases, allowing organizations to deliver new features and updates to users quickly.

2. **Early Bug Detection**: Automated tests run during the CI/CD process help identify issues early, preventing them from progressing to later stages or reaching production.

3. **Increased Collaboration**: Developers and operations teams collaborate closely, improving communication and sharing knowledge to resolve issues more effectively.

4. **Improved Quality**: Automated tests and deployments reduce the risk of human errors and inconsistencies, leading to more reliable and higher-quality software.

5. **Agility and Flexibility**: The ability to iterate and release software frequently enables organizations to respond rapidly to changing market demands and user feedback.

In summary, DevOps is a cultural and operational approach that promotes collaboration, automation, and continuous improvement. CI/CD are key practices within the DevOps framework, enabling organizations to automate software builds, testing, and deployments, resulting in faster, more reliable software delivery.