THE NATIONAL INSTITUTE OF ENGINEERING, MYSURU

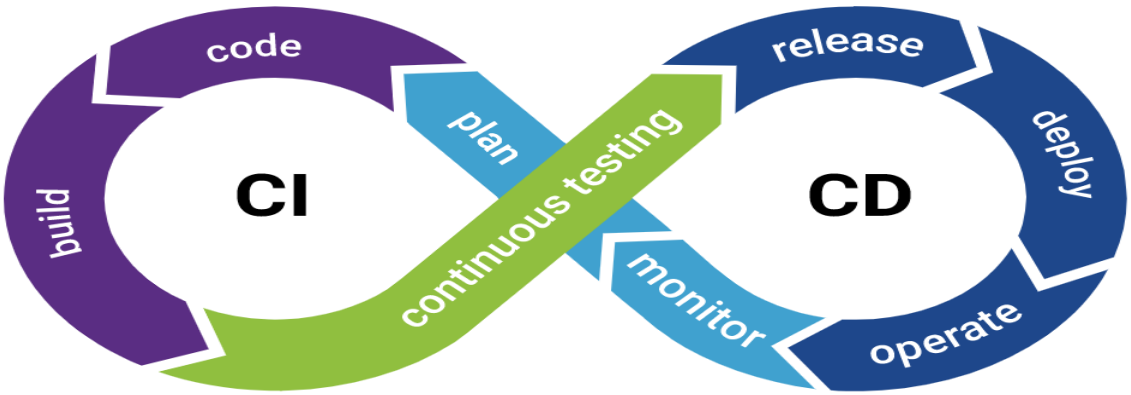
DEPARTMENT OF INFORMATION SCIENCE & ENGINEERING

DevOps (IS7I04)
Assignment_1

Name-PRIYANKA H R
USN- 4NI19IS069
Section-B

1. What is CI/CD?

CI and CD stand for continuous integration and continuous delivery/continuous deployment. In very simple terms, CI is a modern software development practice in which incremental code changes are made frequently and reliably. Automated build-and-test steps triggered by CI ensure that code changes being merged into the repository are reliable. The code is then delivered quickly and seamlessly as a part of the CD process. In the software world, the CI/CD pipeline refers to the automation that enables incremental code changes from developers' desktops to be delivered quickly and reliably to production.A CI/CD pipeline automates the process of software delivery. It builds code, runs tests, and helps you to safely deploy a new version of the software. CI/CD pipeline reduces manual errors, provides feedback to developers, and allows fast product iterations.CI/CD pipeline introduces automation and continuous monitoring throughout the lifecycle of a software product. It involves from the integration and testing phase to delivery and deployment.. CI/CD is a solution to the problems integrating new code can cause for development and operations teams.



DevOps is a set of practices and tools designed to increase an organization's ability to deliver applications and services faster than traditional software development processes. The increased speed of DevOps helps an organization serve its customers more successfully and be more competitive in the market. In a DevOps environment, successful organizations "bake security in" to all phases of the development life cycle, a practice called DevSecOps. The CI/CD pipeline is part of the broader DevOps/DevSecOps framework. In order to successfully implement and run a CI/CD pipeline, organizations need tools to prevent points of friction that slow down integration and delivery. Teams require an integrated toolchain of technologies to facilitate collaborative and unimpeded development efforts.

2. What are Feature flags and how it is used?

Feature flags, also known as feature toggles, are a technique used in software development and DevOps to enable or disable features in a software application without the need to deploy a new version of the application. This allows developers to work on new features or changes to the application without disrupting the functionality of the current version for users.

• Feature flags can be used to test new features with a subset of users before releasing them to the entire user base, or to gradually roll out a feature to all users.

• They can also be used to quickly disable a feature in the event of a problem or to experiment with different versions of a feature.

• In DevOps, feature flags are often used in conjunction with continuous delivery and deployment practices.

• Feature flags are typically implemented using a combination of code and configuration.

• They can be managed through a variety of tools and platforms, including feature flag management platforms, version control systems, and configuration management systems.

An example of how feature flags might be used in a DevOps workflow:

• A developer writes and tests a new feature on their local machine.

• The developer commits the code changes to a version control system (such as Git) and pushes them to a central repository.

• The code is automatically built and deployed to a staging environment.

• The developer creates a feature flag to toggle the availability of the new feature on or off.

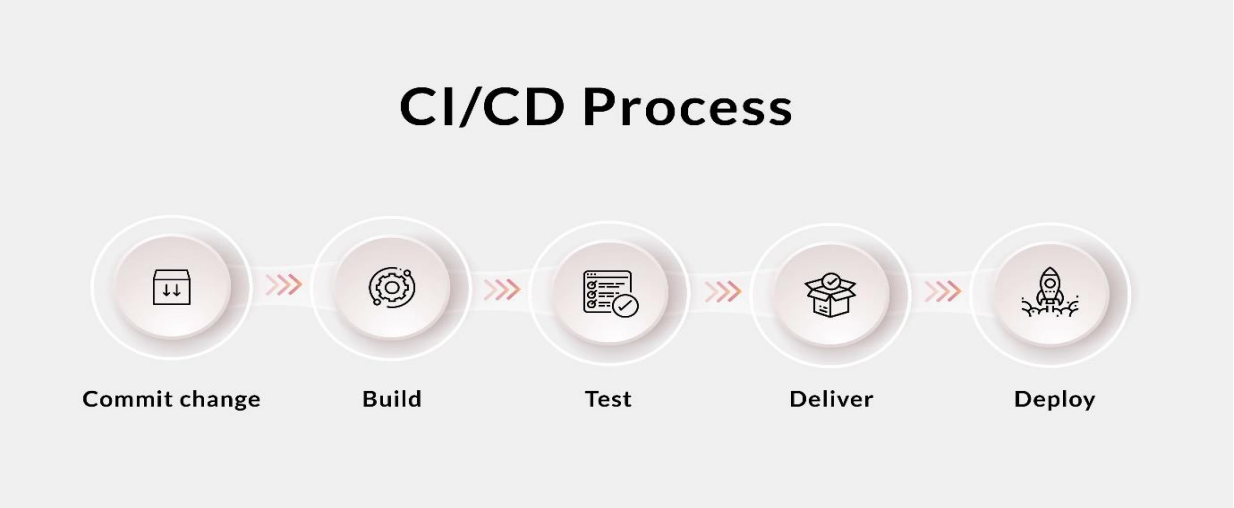
• The developer uses the feature flag to enable the new feature for a small group of users, such as a subset of the staging environment or a group of beta testers.

• The team monitors the usage and performance of the new feature to ensure that it is working as expected.

• If the new feature is stable and performs well, the team can use the feature flag to roll it out to a wider audience. If there are any issues with the feature, the team can use the feature flag to quickly disable it.

3. Explain CI/CD pipeline with block diagram.

The continuous integration/continuous delivery (CI/CD) pipeline is an agile DevOps workflow focused on a frequent and reliable software delivery process. The methodology is iterative, rather than linear, which allows DevOps teams to write code, integrate it, run tests, deliver releases and deploy changes to the software collaboratively and in real-time.A key characteristic of the CI/CD pipeline is the use of automation to ensure code quality. As the software changes progress through the pipeline, test automation is used to identify dependencies and other issues earlier, push code changes to different environments and deliver applications to production environments. Here, the automation’s job is to perform quality control, assessing everything from performance to API usage and security. This ensures the changes made by all team members are integrated comprehensively and perform as intended.The ability to automate various phases of the CI/CD pipeline helps development teams improve quality, work faster and improve other DevOps metrics.



Implementation of CI/CD pipeline stages

Plan

Propose a feature that needs to be built, or outline an issue that warrants change.

Code

Turn use case suggestions and flowcharts into code, peer-review code changes, and obtain design feedback.

Test

Verify code changes through testing, preferably automated testing. At this point, unit testing will usually suffice.

Repository

Push code to a shared repository like Github that uses version control software to keep track of changes made. Some consider this as the first phase of the CI/CD pipeline.

Set up an Integration Testing Service

For example, Travis CI to continuously run automated tests, such as regression or integration tests, on software hosted on services like Github and BitBucket.

Set up a Service To Test Code Quality

Better Code Hub can help continuously check code for quality in CI/CD pipeline. This will enable the software development team to spend less time fixing bugs. Better Code Hub uses 10 guidelines to gauge quality and maintainability in order to future proof the code.

Build

This might overlap with compilation and containerization. Assuming the code is written in a programming language like Java, it'll need to be compiled before execution. Therefore, from the version control repository, it'll go to the build phase where it is compiled. Some pipelines prefer to use Kubernetes for their containerization.

Testing Phase

Build verification tests as early as possible at the beginning of the pipeline. If something fails, you receive an automated message to inform you the build failed, allowing the DevOps to check the continuous integration logs for clues.Smoke testing might be done at this stage so that a badly broken build can be identified and rejected, before further time is wasted installing and testing it. Push the container(Docker) image created to a Docker hub: This makes it easy to share your containers in different platforms or environments or even go back to an earlier version.

Deploy the App, Optionally To a Cloud-Based Platform

The cloud is where most organizations are deploying their applications. Heroku is an example of a relatively cheap cloud platform. Others might prefer Microsoft Azure or Amazon Web Service (AWS).