DEVOPS

Assignment – 1

Dave Darshit

4NI19IS026

7th semester, 'B' section

1. What is CI / CD?

CI (Continuous integration) is a modern software development practice in which incremental code changes are made frequently and reliably by being merged into a shared repository.

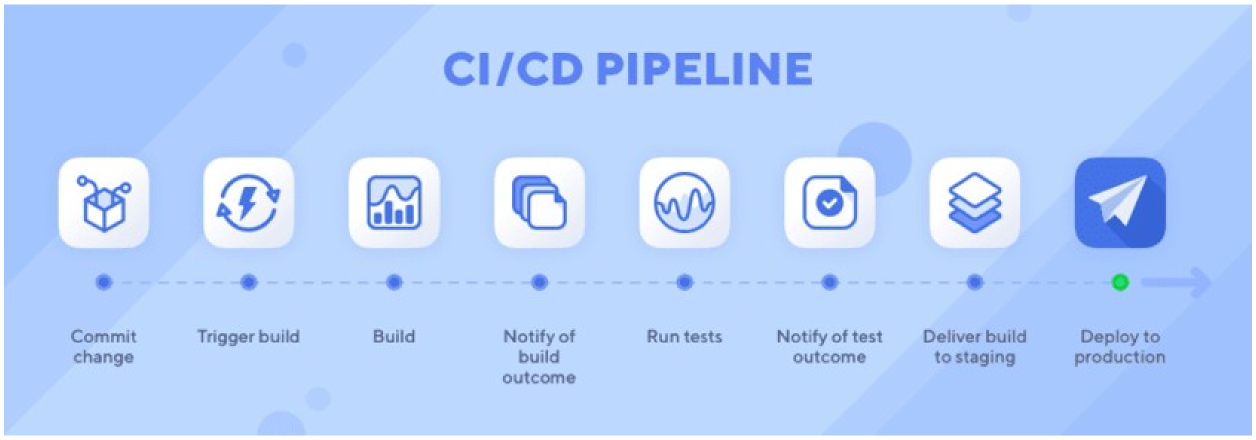
CD (Continuous delivery or Continuous deployment) refers to the automation that enables incremental code changes made by developers to be delivered quickly and reliably to production.

2. What are feature flags?

Feature flags are a system of code that allows conditional features to be accessed only when certain conditions are met. If a flag is on, new code is executed. If the flag is off, the code is skipped.

3. What is CI/CD pipeline?

A CI/CD pipeline automates the process of software delivery. It builds code, runs tests, and helps to safely deploy a new version of the software. Thus, it reduces manual errors, provides feedback to developers, and allows fast product iterations.



A CI/CD pipeline resembles the various stages software goes through in its lifecycle and mimics those stages:

i. **Source stage: Implement a code repository and version control system**

   This initial CI/CD stage involves storing and managing source code in a repository with a version control system (VCS) that supports collaboration and tracking changes across a distributed team. The VCS can trigger a pipeline execution based on various events, such as a branch push or pull request validation. These pipeline runs also can be scheduled or initiated by a user.

ii. **Build stage: Choose a CI engine, compile code, run checks**

   The key to this CI/CD stage is to provide early feedback to developers and maintain the application in a state where it can be released to an environment. It is imperative to provide feedback as soon as a developer checks in new code changes to flag and resolve any issues such as syntax or compilation. Also, compilation ensures that the code can successfully generate artifacts for eventual release into environments further down the CI/CD pipeline.

iii. **Testing stage: Test the build, publish results, release for production**

   The goal of this stage is to ensure that the changes do not break any logic or functionality and that the code is safe to release. In short, testing provides a safety net to release the code. Among the many tests that occur in this stage are unit, integration and functional tests.

   Unit tests examine small units of application code and only test the logic in isolation without any dependencies, although if required, you can simulate them through mocking. For code developed in object-oriented languages such as Java or C#, etc., these small units can be class methods.

   Integration tests analyze individual units of code together in a group. This builds confidence that individual modules integrated to build the entire application won't break under test.

   An end-to-end *functional test* introduces the software into an environment to mimic a production deployment. This step is often automated with tools such as Selenium.

iv. **Deploy stage: Deliver and deploy the final build**

   Once the software is built and tested and artifacts are generated, it is ready to be released into an environment. Ideally, there are multiple environments through which the built artifacts are released and then tested, and if the release passes all the tests, it progresses through to production deployment. This stage also handles adding the required resources to host the application in the cloud.