Submitted by,

Pradhyumna H P

4NI19IS061

B Section

Continuous Integration and Continuous Delivery

Continuous Integration (CI):

CI is a software development method where members of the team can integrate their work at least once a day. In this method, every integration is checked by an automated build to search the error.

Continuous Delivery (CD):

CD is a software engineering method in which a team develops software products in a short cycle. It ensures that software can be easily released at any time.

CI/CD Pipeline:

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. These connected practices are referred as CI/CD pipeline.

Stages in CI/CD Pipeline:

1. **Source Stage:** In the source stage, CI/CD pipeline is triggered by a code repository. Any change in the program triggers a notification to the CI/CD tool that runs an equivalent pipeline. Other common triggers include user-initiated workflows, automated schedules, and the results of other pipelines.

2. **Build Stage:** This is the second stage of the CI/CD Pipeline in which you merge the source code and its dependencies. It is done mainly to build a runnable instance of software that you can potentially ship to the end-user. Programs that are written in languages like C++, Java, C, or Go language should be compiled. On the other hand, JavaScript, Python, and Ruby programs can work without the build stage. Failure to pass the build stage means there is a fundamental project misconfiguration, so it is better that you address such issue immediately.
3. **Test Stage:** Test Stage includes the execution of automated tests to validate the correctness of code and the behaviour of the software. This stage prevents easily reproducible bugs from reaching the clients. It is the responsibility of developers to write automated tests.
4. **Deploy Stage:** This is the last stage where your product goes live. Once the build has successfully passed through all the required test scenarios, it is ready to deploy to live server.

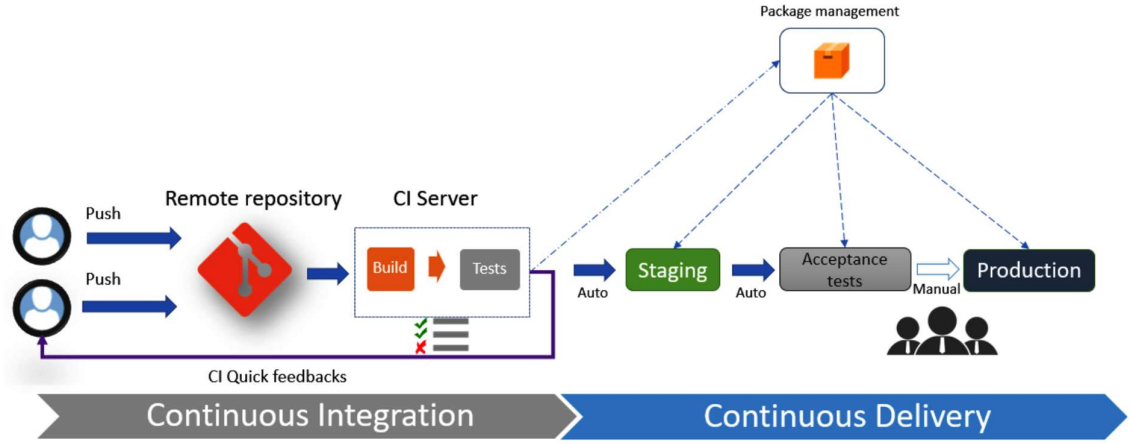

Fig 1 – Stages in a CI/CD Pipeline

Advantages of CI/CD Pipeline:

- Builds and tests can be easily performed and manually.
- Improves the consistency and quality of the code.
- Automates the process of software delivery.
- Improves flexibility and has the ability to ship new functionalities.
- Helps you to achieve faster customer feedback.

- It enables you to remove manual error.
- Reduces costs and labour.
- CI/CD Pipeline makes the software development lifecycle faster.
- CD pipeline gives a rapid feedback loop starting from developer to client.
- It improves the communication between the organization employees.

CI/CD Tools:

- **Jenkins:** It is an open source continuous integration server that helps to achieve the continuous integration process in an automated fashion. Jenkins is free and entirely written in Java.
- **Bamboo:** It is a continuous integration build server that performs automatic build, tests and releases in a single place. It works seamlessly in JIRA and BitBucket.
- **CircleCi:** It is a flexible continuous integration tool that runs in any environment like a cross platform mobile app, Python API server or Docker Cluster. This tool reduces bugs and improves the quality of application.

Feature Flags:

Feature flags (also termed as feature toggles) allow us to dynamically enable or disable a feature of an application without having to redeploy it. They are implemented in application's code. Features encapsulated in feature flags may be necessary either for the running of the application or for an internal purpose such as log activation or monitoring. The activation and deactivation of features can be controlled either by an administrator or directly by users via a graphical interface.

The lifetime of a feature flag can be either of the following:

- **Temporary:** To test a feature. Once validated by the users, the feature flag will be deleted.
- **Definitive:** To leave a feature flagged for a long time.

Thus, using feature flags, a new version of an application can be deployed to the production stage faster. This is done by disabling the new features of the release. Then, we will reactivate these new features for a specific group of users such as testers, who will test these features directly in production.

Moreover, if we notice that one of the application's functionalities is not working properly, it is possible for the feature flags to disable it very quickly, without us having to redeploy the rest of the application.

Feature flags also allow A/B testing—that is, testing the behaviour of new features by certain users and collecting their feedback.

Advantages of Feature Flags:

- You develop and maintain your custom feature flags system, which has been adapted to your business needs. This solution will be suitable for your needs but requires a lot of development time, as well as the necessary considerations of architecture specifications such as the use of a database, data security, and data caching.
- You use an open source tool that you must install in your project. This solution allows us to save on development time but requires a choice of tools, especially in the case of open source tools. Moreover, among these tools, few offer portals or dashboard administration that allows for the management of feature flags remotely. There is a multitude of open source frameworks and tools for feature flags.
- You can use a cloud solution (a platform as a service, or PaaS) that requires no installation and has a back office for managing feature flags, but most of them require a financial investment for large-scale use in an enterprise.