LOHITH C 4NI19IS043

Assignment 1

1. What is CI/CD?

CI or **Continuous Integration** is the practice of automating the integration of code changes from multiple developers into a single codebase. It is a software development practice where the developers commit their work frequently into the central code repository (GitHub or Stash). Then there are automated tools that build the newly committed code and do a code review, etc. as required upon integration.

The key goals of Continuous Integration are to find and address bugs quicker, make the process of integrating code across a team of developers easier, improve software quality and reduce the time it takes to release new feature updates. Some popular CI tools are Jenkins, TeamCity, and Bamboo.

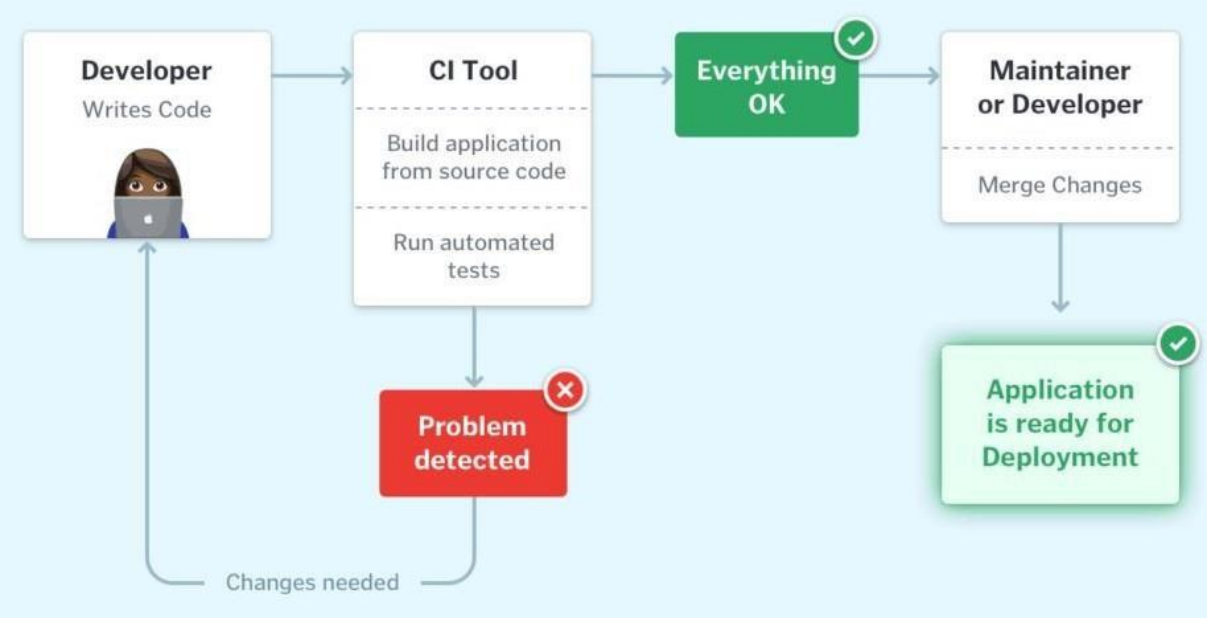


Fig-1 Continuous Integration

There could be scenarios when developers in a team, work in isolation for an extended period of time and only merge their changes to the master branch once their work was completed. This not only makes the merging of code very difficult, prone to conflicts, and time-consuming but also results in bugs accumulating for a long time which are only identified in later stages of development. These factors make it harder to deliver updates to customers quickly.

With Continuous Integration, developers frequently commit to a shared common repository using a version control system such as Git. A continuous integration pipeline can automatically run builds, store the artifacts, run unit tests and even conduct code reviews using tools like Sonar. We can configure the CI pipeline to be triggered every time there is a commit/merge in the codebase.

CD or **Continuous Delivery** is carried out after Continuous Integration to make sure that we can release new changes to our customers quickly in an error-free way. This includes running integration and regression tests in the staging area (similar to the production environment) so that the final release is not broken in production. It ensures to automate the release process so that we have a release-ready product at all times and we can deploy our application at any point in time.

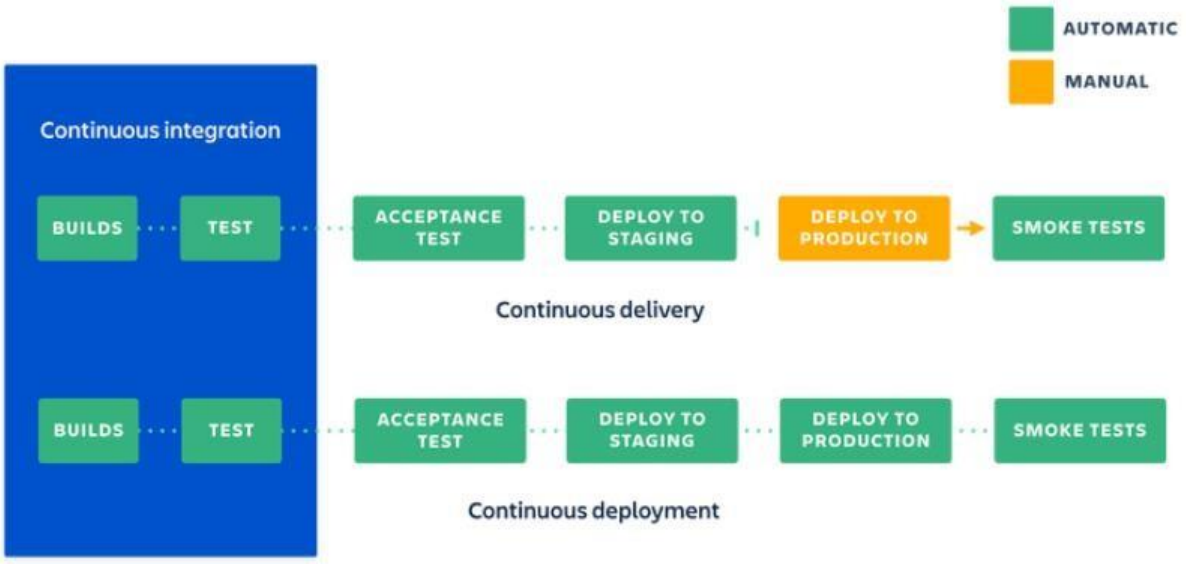
Continuous Delivery automates the entire software release process. The final decision to deploy to a live production environment can be triggered by the developer/project lead as required. Some popular CD tools are AWS CodeDeploy, Jenkins, and GitLab.

Continuous delivery helps developers test their code in a production-similar environment, hence preventing any last moment or post-production surprises. These tests may include UI testing, load testing, integration testing, etc. It helps developers discover and resolve bugs preemptively.

By automating the software release process, CD contributes to low-risk releases, lower costs, better software quality, improved productivity levels, and most importantly, it helps us deliver updates to customers faster and more frequently. If Continuous Delivery is implemented properly, we will always have a deployment-ready code that has passed through a standardized test process.

We have seen how Continuous Integration automates the process of building, testing, and packaging the source code as soon as it is committed to the code repository by the developers. Once the CI step is completed, the code is deployed to the staging environment where it undergoes further automated testing (like Acceptance testing, Regression testing, etc.). Finally, it is deployed to the production environment for the final release of the product.

If the deployment to production is a manual step, the process is called Continuous Delivery whereas if the process of deployment to the production environment is automated, it is referred to as Continuous Deployment.



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

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.

3. Explain CI/CD pipeline with block diagram

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

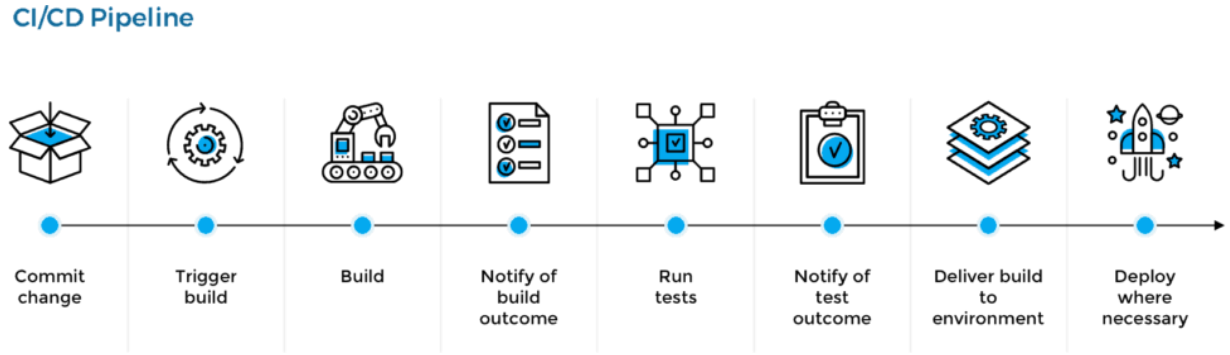
product. It involves from the integration and testing phase to delivery and deployment. These connected practices are referred as CI/CD pipeline.

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.



Advantages:

- 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.