DEVOPS Assignment (1)

Jitendra Lokesh Bhat – 4NI19IS039

<u>CI/CD Pipelines</u>

1) What is CI/CD?

The basic idea behind CI/CD is to have a workflow that automatically detects code changes, builds the code, runs tests, and deploys the code to production if the tests pass. This allows teams to rapidly and safely deliver new features and updates to users, as the automation helps to catch problems earlier in the development process.

- Continuous Integration (CI) is the practice of automatically building and testing code changes. Developers write code and commit their changes to a version control system, such as Git. CI servers monitor the version control system for new commits, and when it detects a new commit, it automatically builds the code and runs tests to ensure the code changes did not introduce any regressions.
- Continuous Deployment (CD) is a method of automatically releasing code changes to production. Once the code passes the tests in her CI process, it is automatically deployed to production and made available to users. This allows your team to quickly and securely roll out new features and updates to your users.
- It also helps teams deploy software updates faster and reduce errors because automation helps catch problems early in the development process.
- By using CI/CD, teams can release software updates faster and with fewer errors. The automation helps to catch problems earlier in the development process, and the frequent deployments make it easier to roll back changes if there are issues. CI/CD also helps to improve collaboration among developers, as it encourages them to commit code changes frequently and integrates the work of multiple developers into a single codebase.

Overall, CI/CD is a key practice for organizations looking to deliver software updates rapidly and safely. It helps teams to build, test, and deploy code changes automatically, which can lead to faster and more reliable releases.

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

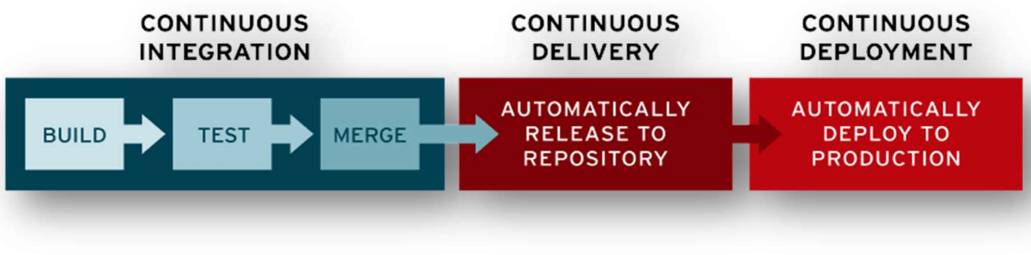
Feature flags can be used to enable or disable specific features in software without deploying a new version of the code. This allows teams to rapidly and safely release new features and updates to users by controlling the release of features through the use of feature flags.

Some common ways that feature flags are used in DevOps include:

- Gradual rollouts: Teams can use feature flags to gradually roll out new features to a small group of users and gradually increase the number of users over time. This allows teams to gather feedback and test the feature with a small group before releasing it to everyone.
- Canary releases: Teams can use feature flags to release a new feature to a small group of users and monitor its performance before releasing it to everyone. This allows teams to test the feature in a production environment and ensure that it is stable before releasing it to all users.
- A/B testing: Teams can use feature flags to conduct A/B tests, where different versions of a feature are released to different groups of users and the results are compared to determine which version performs better.
- Controlling access to features: Teams can use feature flags to control access to features based on user attributes, such as location, account type, or subscription status. This allows teams to selectively release features to specific users or groups.
- Rolling back changes: If there are problems with a new feature, teams can use feature flags to quickly disable the feature for all users until the issue is resolved. This allows teams to fix problems without the need to deploy a new version of the code.
- Overall, feature flags can be a useful tool for teams practicing DevOps to rapidly and safely release new features and updates to users.

3) Explain CI/CD pipeline with block diagram.

A CI/CD pipeline is a series of automated processes that enable software changes to be rapidly and safely delivered to users. The pipeline typically consists of the following stages:



Continuous Integration/Continuous Deployment (CI/CD) is a software engineering practice that aims to minimize the time between writing code and deploying it to production by automatically building, testing, and releasing software updates.

The basic idea behind CI/CD is to have a workflow that automatically detects code changes, builds the code, runs tests, and deploys the code to production if the tests pass. This allows teams to rapidly and safely deliver new features and updates to users, as the automation helps to catch problems earlier in the development process.

Here is a more detailed explanation of the steps involved in a typical CI/CD pipeline:

Developer writes code and commits changes to version control system (e.g. Git): Developers write code and commit their changes to a version control system, such as Git. This allows teams to track changes to the code over time and collaborate on projects. CI server monitors version control system for new commits: The CI server is a software tool that monitors the version control system for new commits. When a new commit is detected, the CI server automatically begins the build process.

CI server automatically builds code and runs tests: When a new commit is detected, the CI server automatically retrieves the latest version of the code from the version control system and builds it. The CI server may also run a variety of tests, such as unit tests, integration tests, and static analysis tools, to ensure that the code changes did not introduce any regressions or other issues. CI server generates artifacts: After the code is built and tests are run, the CI server generates artifacts, which are typically executable files or packages that can be deployed to production.

For example, the CI server may generate a Docker image, a Java JAR file, or a Node.js package.

Artifacts are deployed to staging environment: If the tests pass, the artifacts are automatically deployed to the staging environment. The staging environment is a replica of the production environment that is used for testing purposes.

QA team manually tests staging environment: The Quality Assurance (QA) team manually tests the staging environment to ensure that the code is ready for production. The QA team may use a variety of testing techniques, such as manual testing, automated testing, and performance testing, to ensure that the code is of high quality. Artifacts are deployed to production environment: If the QA team approves, the artifacts are automatically deployed to the production environment, making them available to users. The production environment is the live environment where users access the software.

This pipeline allows teams to automatically build, test, and deploy code changes, which can help to reduce the time it takes to release updates to users. It also helps to catch problems earlier in the development process, as the automated tests can detect issues before the code is deployed to production.

There are many tools and services available to help teams implement a CI/CD pipeline. Some popular tools include Jenkins, Travis CI, and CircleCI. These tools provide a range of features, such as support for different programming languages, integration with a variety of version control systems, and the ability to run tests and deploy code to a variety of environments.

Overall, CI/CD is a powerful software engineering practice that helps teams to deliver software updates faster and with fewer errors. It allows teams to rapidly and safely deliver new features and updates to users, while also helping to catch problems earlier in the development process.