Rekha Maruti Shet
4NI19IS075
B - Section

ASSIGNMENT 1

1) What is Continuous Integration and Continuous Deployment (CI/CD)

CI/CD is a method to frequently deliver apps to customers by introducing automation into the stages of app development. The main concepts attributed to CI/CD are continuous integration, continuous delivery, and continuous deployment. CI/CD is a solution to the problems integrating new code can cause for development and operations teams (AKA "integration hell").

Specifically, CI/CD introduces ongoing automation and continuous monitoring throughout the lifecycle of apps, from integration and testing phases to delivery and deployment. Taken together, these connected practices are often referred to as a "CI/CD pipeline" and are supported by development and operations teams working together in an agile way with either a DevOps or site reliability engineering (SRE) approach.

Difference between CI and CD

Continuous Integration	**Continuous Delivery**
CI is a development process of automatically building and performing unit tests upon making changes to the source code.	CD is an extension of continuous integration. It is a process in which DevOps teams develop and deliver complete portions of software to a repository such as GitHub.
CI requires development teams to integrate code changes into a shared source code repository multiple times every day.	Cd makes releases regular and predictable events for DevOps staff, and seamless for end-users.
The main goal of continuous integration is to create a consistent, ongoing method to	The main goal of continuous delivery is to always keep code in a deployable state so that

automatically build and test applications ensuring a change from one developer is suitable for use in the entire code base.	updates can go live at a moment's notice with little or no issues.
Through continuous integration, developers can solve problems they face when writing. Integrating, testing and delivering software applications to end users.	Programmers are working in a one to two week sprint rather than a months ling development of an update

2) What are feature flags and how it is used

Feature flags are system of codes that allows conditional features to be accessed only when certain conditions are met. In other words, if a flag is on, new code is executed if the flag is off, the code is skipped. It is also referred to as release toggles, feature flags are a best practice in DevOps, often occurring within distributed version control systems.

Example code:

```
if(activateFeature("addTaxToOrder")==True)
{
ordervalue = ordervalue + tax
}
else
{
 ordervalue = ordervalue
 }
```

In this example code, the activateFeature function allows us to find out whether the application should add the tax to order according to the addTaxToOrder parameter, which is specified outside the application (such as in a database or configuration file).

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.

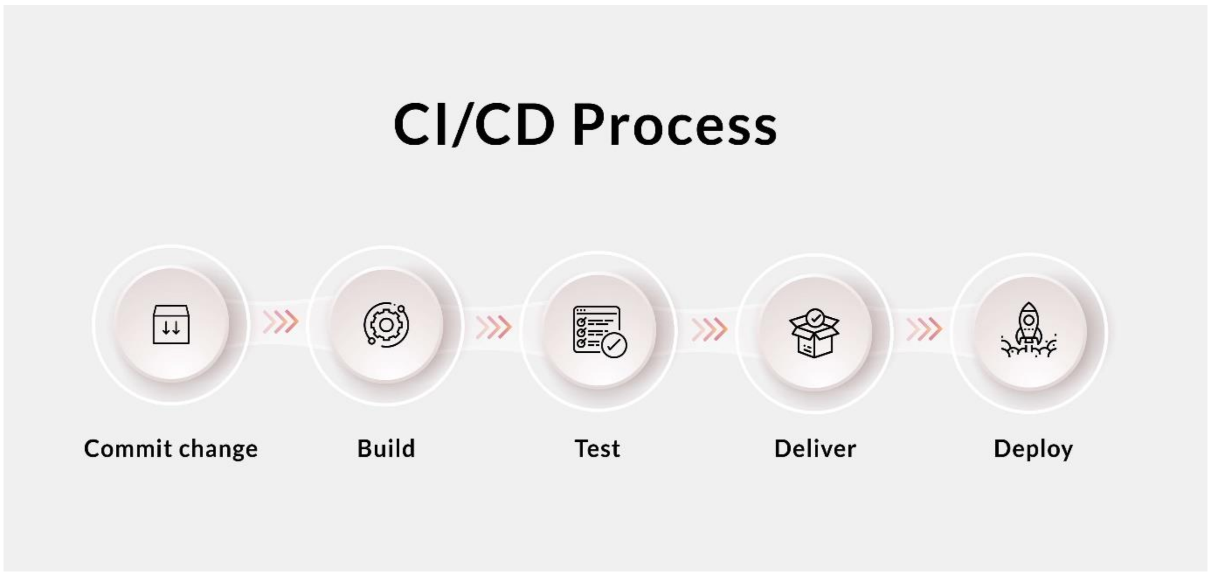
A key characteristic of the CI/CD pipeline is the use of automation to ensure code quality. As the software changes progress through the pipeline, test automation is used to identify dependencies and other issues earlier, push code changes to different environments and deliver applications to production 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.

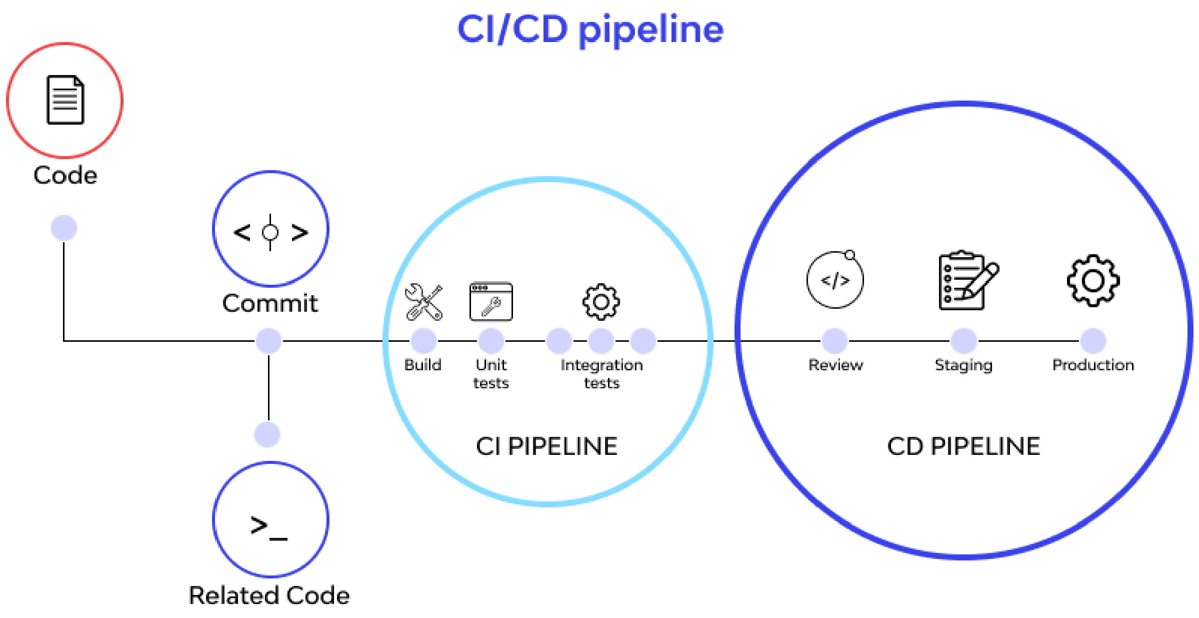
The ability to automate various phases of the CI/CD pipeline helps development teams improve quality, work faster and improve other DevOps metrics.

CI/CD pipeline phases

From source code to production, these phases make up the development lifecycle and workflow of the CI/CD pipeline:

- **Build:** This phase is part of the continuous integration process and involves the creation and compiling of code. Teams build off of source code collaboratively and integrate new code while quickly determining any issues or conflicts.
- **Test:** At this stage, teams test the code. Automated tests happen in both continuous delivery and deployment. These tests could include integration tests, unit tests, and regression tests.
- **Deliver:** Here, an approved codebase is sent to a production environment. This stage is automated in continuous deployment and is only automated in continuous delivery after developer approval.
- **Deploy:** Lastly, the changes are deployed and the final product moves into production. In continuous delivery, products or code are sent to repositories and then moved into production or deployment by human approval. In continuous deployment, this step is automated.





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).