SUJAY MUDGAL
4NI19IS101
ISE “A”

1. What is 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").

The "CD" in CI/CD refers to continuous delivery and/or continuous deployment, which are related concepts that sometimes get used interchangeably. Both are about automating further stages of the pipeline, but they’re sometimes used separately to illustrate just how much automation is happening.

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

Feature flags (also known as feature toggles or feature switches) are a software development technique that turns certain functionality on and off during runtime, without deploying new code. This allows for better control and more experimentation over the full lifecycle of features.

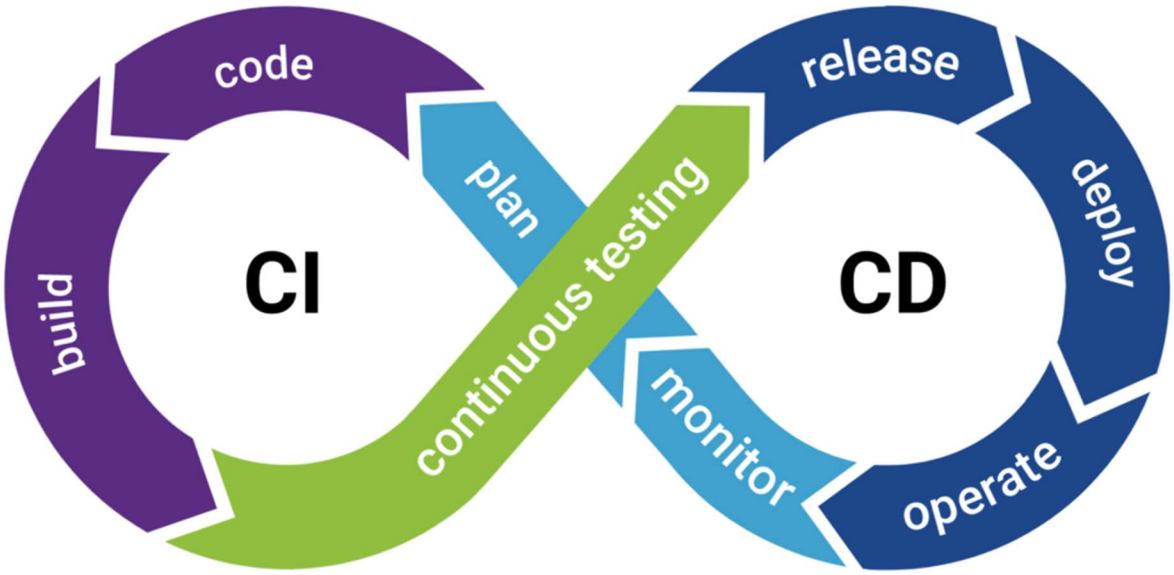
Feature flags are a software development concept that allow you to enable or disable a feature without modifying the source code or requiring a redeploy. They are also commonly referred to as feature toggles, release toggles or feature flippers. Feature flags determine at runtime which portions of code are executed. This allows new features to be deployed without making them visible to users or, even more importantly, you can make them visible to only a specific subset of users.

SUJAY MUDGAL
4NI19IS101
ISE “A”

3. Explain CI/CD pipeline with block diagram?

The code will progress from code check-in through the test, build, deploy, and production stages. Engineers over the years have automated the steps for this process. The automation led to two primary processes known as **Continuous Integration** and **Continuous Delivery**.

Standard Definition: A CI/CD pipeline is defined as a series of interconnected steps that include stages from code commit, testing, staging, deployment testing, and finally, deployment into the production servers. We automate most of these stages to create a seamless software delivery.



For a CI CD pipeline to work, we require a series of sub-processes or stages that need to continuously check and verify the code updates. These sub-stages are as follows-

SUJAY MUDGAL
4NI19IS101
ISE “A”

Continuous Integration:

- Code Commit
- Static Code Analysis
- Build, and
- Test stages/scenarios

Continuous Delivery:

- Bake
- Deploy
- Deployment Testing and Verification
- Monitoring, and
- Feedback

1. Comparison between hypervisor and docker?

- Functioning Mechanism

The most significant difference between hypervisors and Dockers is the way they boot up and consume resources.

Hypervisors are of two types – the bare metal works directly on the hardware while type two hypervisor works on top of the operating system.

Docker, on the other hand, works on the host kernel itself. Hence, it does not allow the user to create multiple instances of operating systems.

Instead, they create containers that act as virtual application environments for the user to work on.

- Number of Application Instances Supported

SUJAY MUDGAL
4NI19IS101
ISE “A”

A hypervisor allows the users to generate multiple instances of complete operating systems. Dockers can run multiple applications or multiple instances of a single application. It does this with containers.

- Memory Requirement

Hypervisors enable users to run multiple instances of complete operating systems. This makes them resource hungry.

They need dedicated resources for any particular instance among the shared hardware which the hypervisor allocates during boot.

Dockers, however, do not have any such requirements. One can create as many containers as needed.

Based on the application requirement and availability of processing power, the Docker provides it to the containers.

- Boot-Time

As Dockers do not require such resource allocations for creating containers, they can be created quickly to get started.

One of the primary reasons why the use of Dockers and containers is gaining traction is their capability to get started in seconds.

A hypervisor might consume up to a minute to boot the OS and get up and running.

Docker can create containers in seconds, and users can get started in no time.

- Architecture Structure

If we consider both hypervisor and Docker’s architecture, we can notice that the Docker engine sits right on top of the host OS.

It only creates instances of the application and libraries.

Hypervisor though, has the host OS and then also has the guest OS further. This creates two layers of the OS that are running on the hardware.

If you are to run a portable program and want to run multiple instances of it, then containers are the best way to go. Hence you can benefit significantly with a Docker.

Dockers help with the agile way of working. Within each container, different sections of the program can be developed and tested.

In the end, all containers can be combined into a single program. Hypervisors do not provide such capability.

- Security

Hypervisors are much more secure since the additional layer helps keep data safe.

One of the major differences between the two is the capability to run operating systems or rather run-on operating systems.

- OS Support

Hypervisors are OS agnostic. They can run across Windows, Mac, and Linux.

SUJAY MUDGAL
4NI19IS101
ISE “A”

Dockers, on the other hand, are limited to Linux only. That, however, is not a deterrent for Dockers since Linux is a strong eco-system. Many major players are entering into the Dockers’ fray

2. Comparison between Containers and Virtual machines?

SNo.	Virtual Machines(VM)	Containers
1	VM is piece of software that allows you to install other software inside of it so you basically control it virtually as opposed to installing the software directly on the computer.	While a container is a software that allows different functionalities of an application independently.
2.	Applications running on VM system can run different OS.	While applications running in a container environment share a single OS.
3.	VM virtualizes the computer system.	While containers virtualize the operating system only.
4.	VM size is very large.	While the size of container is very light; i.e. a few megabytes.
5.	VM takes minutes to run, due to large size.	While containers take a few seconds to run.
6.	VM uses a lot of system memory.	While containers require very less memory.

SUJAY MUDGAL
4NI19IS101
ISE “A”

7.	VM is more secure.	While containers are less secure.
8.	VM’s are useful when we require all of OS resources to run various applications.	While containers are useful when we are required to maximise the running applications using minimal servers.
9.	Examples of VM are: KVM, Xen, VMware.	While examples of containers are:RancherOS, PhotonOS, Containers by Docker.