**DEVOPS**

**SELF-LEARNING ASSIGNMENT-1**

**SHIVANI L H**

**4NI19IS089**

**A Section**

* **INTRODUCTION**

## Before we get into the topic that is given in the self-learning exercise. Let us start from where it all originated and that would be DevOps.

## DevOps, which stands for development and operations, is considered as a solution to the problem of how we can collaboratively develop, test, operate, deploy in addition to sharing tasks performed by a company's application development and IT operations teams. It can also be expressed as an infinite loop that is meant to improve the work of software development life cycle, consisting of stages being: build, test, release, operate and monitor. Once feedback is given, the loop resets.

## In a broader sense, DevOps endorses for better communication and joint effort between teams within an organization. However, In its most narrow perception, DevOps makes sure to illustrate the adoption of iterative software development, automation, and programmable infrastructure deployment and maintenance.

. DevOps usually implement a few common methods in their environment, such as:

## Continuous integration and Continuous Delivery or Continuous Deployment (CI/CD).

* Real-time monitoring, incident management, configuration management and collaboration platforms.
* cloud computing, microservices and containers which are implemented within the DevOps environment.

## 

## **PROBLEM STATEMENT**

Now that we have understood the overview of DevOps, let us consider why exactly we need to know about CI/CD.

Imagine a pair of teammates within a company who work on a web application/website, where one person can work on the backend and the other with the frontend. They could be using their own applications and code for the project which works well for them. However, even though they have been working on it for months, when the time comes for them to merge these features, they could find that both the features are completely incompatible with each other, and the build fails. This incident would cost a lot of time and money for the company to resolve all the conflicts.

## **What is CI/CD?**

CI/CD essentially stands for Continuous Integration (CI) and Continuous Delivery (CD) or Continuous Deployment (CD). It is the method which creates a faster and more accurate way of combining the work of different people into one united product. Within DevOps, CI/CD is a set of practices or tools that streamlines the product development process. The ultimate goal is to deliver quality products.

* **What is the CI/CD pipeline?**

The continuous integration and continuous delivery (CI/CD) pipeline is a collection of operating principles workflow that is focused on the frequent code changes, development and software delivery process. As stated before, this method allows development and operation, which is considered as the DevOps teams to write the code in the repository, integrate it with the server, run tests for any error, deliver releases and deploy changes in real-time to the software collectively as a whole.

The main point or the basic idea of CI/CD pipeline is the use of automation to ensure code/product quality for the end-users in a reliable way. The automation of the Devops lifecycle as much as possible, that automation is known as a pipeline. The code would be pushed up, tested for any issues, checked for quality, made to go through the user acceptance tests and finally deployed to the users. 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 of a company or organization, are integrated comprehensively, and perform as intended.

The various stages that ensure quality product to the customer/end-users through the CI/CD pipeline are:

1. **Continuous Integration (CI):** It is a development practice that enables the developers to merge their code changes in the central repository of the server, which helps the already automated build and tests to run. The process ensures to avoid merge conflicts by continuous testing of the build.
2. **Continuous Delivery (CD):** It is a practice that refers to the building, testing, and delivering improvements to the software code made by the developers. This can be considered as a phase which is an extension of the Continuous Integration phase. This process of phase ensures that the product is delivered to the customers/users in a timely manner.
3. **Continuous Deployment (CD):** It is an extended part of Continuous Development which develops into its own phase. This is the final stage in the pipeline that ensures the release of any development changes by the developers for the software in the repository to the production. This stage ensures that the end-users receive the software even as the development of the software may still be released to production.

* **BLOCK DIAGRAM OF CI/CD PIPELINE**

Continuous Integration

Continuous Delivery

Changes released to the repository automatically

Continuous Deployment

Development changes in repository deployed the end-users

Build

Test

Merge

**Fig 1. CI/CD block diagram**

* **PROBLEM STATEMENT**

One of the biggest challenges in product development is delivering and controlling new quality features to the end-user. It's difficult to test new features before releasing them without knowing how the feature could possibly affect. As the team doesn't want to risk a bad release, they must be careful.

* **What are feature flags?**

Feature flags are a system of code that separates the release of new features from the actual deployment of the code made by the DevOps team. It only allows these features to be accessed when certain conditions are met. It can alternatively be known as a toggle button to turn the features on/off. Such as, if a flag is on, the new code provided by the Devops team is executed; if the flag is off, then the code will be skipped. It is also referred to as Release Toggle.

The few benefits to feature flags are:

1. Turn it on/off without deployment.
2. Test directly into the production environment.
3. Segment the users based on their different attributes.
4. Continuous delivery to the production.
5. Reduce the risk when releasing the new features to the production.
6. Faster release modulations.

It is very easy to implement feature flags in the code by using a feature management tool like DevCycle to create the flags within the platform, and then wrap new features within the code with flags when the team wants to implement them.

**REFERENCES**

DevOps: <https://www.techtarget.com/searchitoperations/definition/DevOps>

CI/CD:

1. <https://www.ibm.com/cloud/blog/ci-cd-pipeline>
2. <https://katalon.com/resources-center/blog/continuous-delivery-vs-continuous-deployment>

Feature flags : <https://devcycle.com/blog/what-are-feature-flags-in-devops#:~:text=the%20feature%20flag.-,Feature%20flags%20are%20features%20turned%20on%2Foff%20during%20runtime%20without,deploy%20software%20to%20their%20users>.