DEVOPS ASSIGNMENT 1

---SAMIKSHA(4NI19IS082)

SECTION A

Introduction

DevOps is the acronym given to the combination of Development and Operations. It refers to a collaborative approach to make the Application Development team and the IT Operations team of an organization to seamlessly work with better communication. It is a philosophy that encourages adopting iterative software development, automation, and programmable infrastructure deployment and maintenance.

DevOps emphasizes building trust and better liasioning between developers and system administrators. This helps the organization in aligning technological projects to business requirements. Changes rolled out are usually small and reversible, which the entire team begins to comprehend.DevOps is visualized as an infinite loop comprising the steps: plan, code, build, test, release, deploy, operate, monitor, then back to plan, and so on.

Its not a technology but an environment and includes the following:

1. Continuous integration and continuous delivery or continuous deployment (CI/CD) tools,with an emphasis on task automation.

2. Systems and tools that support DevOps adoption, include real-time monitoring and collaboration platform.

3. Cloud computing

While DevOps is seen as a natural extension of Agile, and somewhat of an anathema for ITIL, it does not have its own framework – and can potentially be relevant for a variety of situations.

What is CI/CD?

Continuous Integration and Continuous Delivery (CI/CD) is a strategic approach to automate the application development process with the aim of speeding up the turnaround time. CI, or continuous integration, involves frequently merging of the application code changes into a shared branch. CD refers to continuous deployment or delivery that involves automating the application release process and rolling out the app post-merging.

Continuous Integration in DevOps-

CI is a set of application coding practices that enables the Dev teams to continuously implement minor code updates and simultaneously validate it in a version control repository. Today's digitally advanced and complex applications are coded over diverse frameworks and tools. CI ensures the teams have a consistent process to integrate and validate the updates.

CI enables the adoption of an automated approach to create, package, and test the apps. When digital businesses have a uniform and reliable integration process in place, the Dev team is encouraged to do code modifications more frequently, leading to optimized code quality and team collaboration.

CI facilitates the continuous merging of code updates in a shared or trunk branch. The CI approach reduces the incidents of merger conflicts and ensures the bugs reported are less severe and take less time to resolve.

Continuous Delivery in DevOps-

CD comes into play where CI ends and automates the application delivery to the desired production, Dev, and QA environments. In other words, CD is an automated approach to moving code updates to different environments.

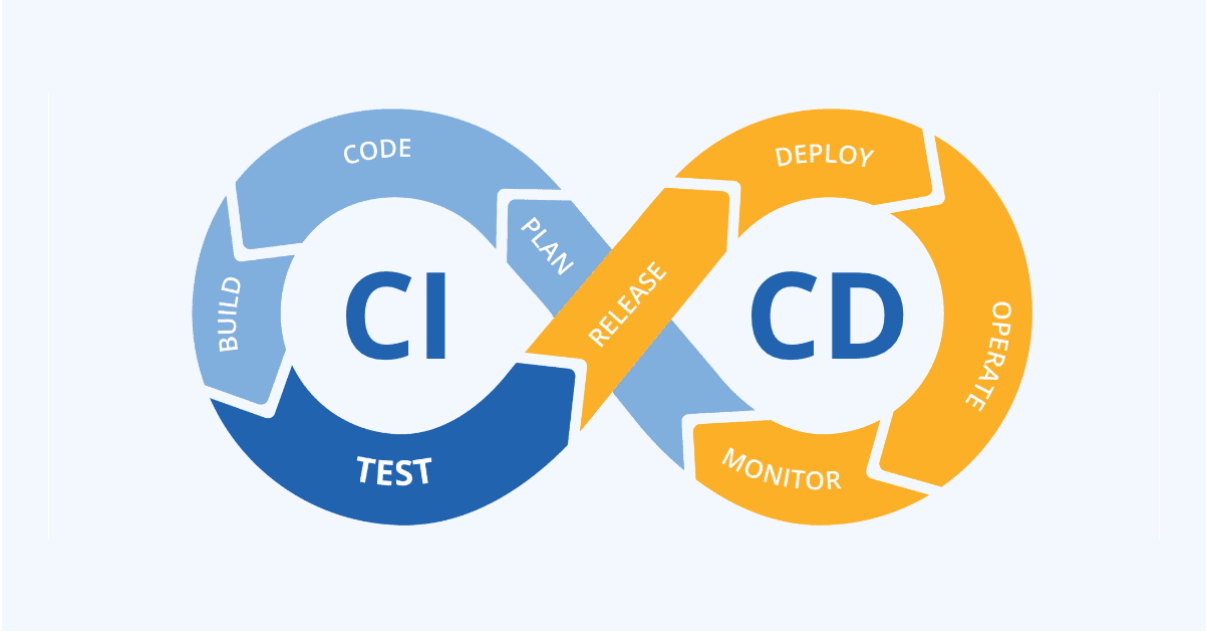
CD is the next step in the automated SDLC. Once the code is validated for bugs and errors, it is pushed to a shared repository like Bitbucket. CD

optimizes the overall visibility and communication between the Dev and Ops departments by accelerating the delivery of production-ready code.

However, it is important to note that CD is only effective after CI implementation. Code validation in the CI stage is essential to ensure that the code can be pushed to the repository and prevent any defective code from being delivered to production.

CI/CD pipelines are agile workflows that ensure continuous, error-free, and reliable software delivery. The methodology enables the DevOps teams to create, integrate, test, and deliver the code, as well as deploy application updates in real time.

A critical feature of a CI/CD pipeline is quality-driven automation. As the application updates are made through the pipeline, test automation detects and highlights the dependencies and other issues, pushes updates to various environments, and moves the application to production. Test automation performs quality assurance over critical aspects such as performance, security, and API integration. CI/CD pipelines ensure that the updates made by the Dev, Ops, and QA teams are aligned and perform as expected.



Stages:

1.Build : The build stage involves the code being written. This is typically done by multiple people in a team, and for larger projects, multiple teams. Code is held in a version control system (VCS).

2. Test: testing code leads to greater confidence that the code will perform as expected. Testing of your code can be automated. In general, this is normally a repetitive, complex and sometimes tedious process to perform manually.

3.Deliver: After the code has been tested, the code is packaged up as an artifact and committed to a repository.

4.Deploy: The deploy stage allows the orchestration of the artifact release. Usually, teams will deploy to multiple environments, including environments for internal use such as development and staging, and Production for end-user consumption.

CI/CD Tools:

A good CI/CD tool can help you create a strong CI/CD pipeline. Popular CI/CD tools include:

- **Jenkins**: Jenkins is an open-source, Java-based automation server that supports building, deploying, and automating software development processes.
- **CircleCI**: CircleCI supports software development and publishing. It allows you to automate the entire pipeline, and integrate with services like GitHub, GitHub Enterprise, and Bitbucket to perform builds when code is committed.
- **GitLab**: GitLab provides a suite of tools for managing the software development lifecycle. You can perform builds, run tests, and deploy code. It also allows you to build jobs in a VM, Docker container, or a different server.

Major cloud providers, such as Microsoft, Amazon, and Google, also offer CI/CD process tools:

- **Azure DevOps**: Azure DevOps provides a variety of CI/CD tools, like Git repo management, testing, reporting, and more. It provides support for Azure, Kubernetes, and VM-based resources.
- **AWS CodePipeline**: AWS CodePipeline is a continuous delivery service that allows you to automate release pipelines. It easily integrates with third-party services like GitHub.
- **Cloud Build from Google Cloud Platform (GCP)**: Cloud Build from GCP is a serverless CI/CD platform that allows you to build software across all languages, such as Java and Go, deploy across multiple environments, and access cloud-hosted CI/CD workflows within your own private network.

Benefits of CI/CD Pipeline:

- **Reduced risk**- CI/CD pipelines encourage developers to make regular commits to a shared repository. This means commit cycles are much smaller, thus fewer changes are being made to the codebase at any one time.
- **Faster time to market**- A prerequisite for continuous delivery is to ensure code is always in a releasable state. To achieve this it's important the entire system is always moving. This means before we can make continuous delivery work effectively, we must ensure code is continuously being integrated into the shared repository.
- **Fault Detection and isolation-** Fault detection, isolation, and recovery is a subfield of control engineering which concerns itself with monitoring a system, identifying when a fault has occurred, and pinpointing the type of fault and its location.
- **Smaller Backlog-** CI/CD pipelines provide developers with a chance to reduce the number of non-critical defects in the team's backlog. The team can reap many benefits from solving non-critical issues before they can become harmful.
- **Better teamwork**- CI/CD encourages transparency and accountability among team members.

REFERENCES

1. https://medium.com/pykes-technical-notes/7-benefits-of-ci-cd-pipelines-22f807e81266
2. https://www.testingxperts.com/blog/what-is-ci-cd-in-devops
3. https://www.tutorialworks.com/cicd-pipeline-stages/
4. https://www.educative.io/blog/what-is-ci-cd-devops