DevOps Curriculum

Overview of DevOps Architecture Design.

UNIT-I DevOps Workflow

1. Introduction to devOps

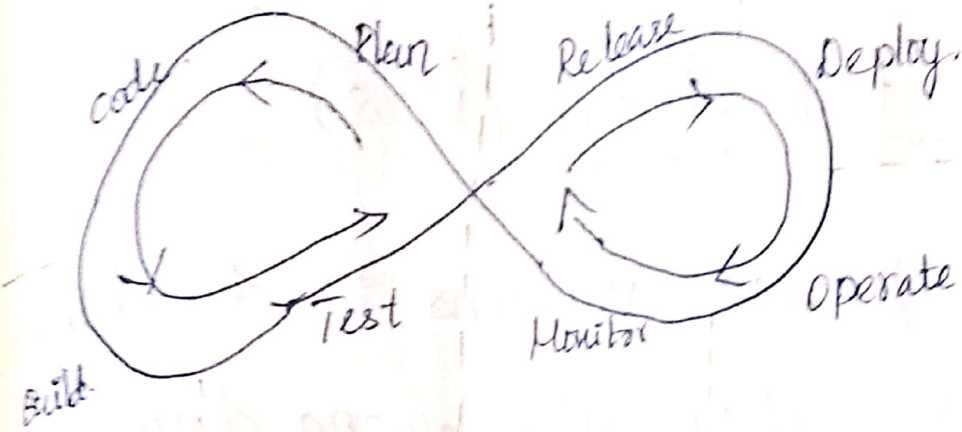
i) Definition and goals of DevOps.
ii) DevOps Architecture
iii) DeVOps Architecture Workflow.

) DEFINITION AND GOALS OF DEVOPS.

The main goals of DevOps are to improve the speed, efficiency, and quality of software development and delivery. Here are the primary Objectives:

- ✓) Increase Deployment Frequency
- ✓) Improve Deployment Quality
- ✓) Reduce Lead Time for changes
- ✓) Enhance Collaboration and Communication
- ✓) Improve Recovery Time
- ✓) Automate and streamline Processes.

DevOps Architecture Diagram



plan: requirement gather

code: step by step coding(or) coded which technology

Build: Build process (setup) (EX:1)

Test: Testing the process

Release: The process is released.

Deploy: For Enduser is deployed the code.

operate: The process is going (or) not Ensure that

Monitor: Connection is rtly established or not.
Continuous performance monitor..

:) DevOps Architecture

Key Components of DevOps Architecture.

>) Version Control System (VCS)

Purpose: Manages code versions, tracks changes, and facilitates collaboration among developers.

>) Continuous Integration (CI)

Purpose: Automate the process of integration code changes from multiple contributors into a single software project.

>) Continuous Delivery / Continuous Deployment (CD)

Purpose: Automates the deployment of code changes to various environments, ensuring that software can be released reliably at any time.

>) Configuration Management:

Purpose: Manages and maintains consistency in software environments (development, testing, production).

4) Infrastructure as Code (IaC)

Purpose: Manages & provisions computing infrastructure through machine-readable definition files, rather than physical hardware or interactive configuration tools.

5) Containerization and Orchestration

Purpose: Packages applications and their dependencies into containers to ensure consistency across environments and simplifies deployment.

6) Continuous Monitoring and Logging

Purpose: Monitors applications and infrastructure to detect performance issues, errors, security threats

7) Collaboration and Communication Tools

Purpose: Facilitties communication and collaboration among team members, enabling faster decision-making and issue resolution.

DevOps Workflow:

Code: Developers write and commit code to a version control system (e.g., Git).

Build: The CI server automatically builds the code into executable files; creating artifacts that can be deployed.

Test: Automated tests are run to ensure the quality of the code. This includes unit tests, integration tests, and sometimes security checks.

Release: If all test pass, the code is packaged and prepared for deployment.

Deploy: The code is automatically deployed to the target environment (e.g., staging, production). Continuous Deployment involves deploying to Production automatically, whereas continuous Delivery might require manual approval.

Operate: The deployed applications are monitored for performance, reliability, and security. Continuous monitoring tools collect metrics and logs, providing

application's behavior

monitor: feedback is collected from monitoring and users, providing data for continuous improvement. Any issues detected are fed back into the development process for resolution

1.2 DevOps vs. Traditional IT Operations

1.2.1 Differences between DevOps and traditional software development and IT operations.

1.2.2 Benefits of adopting DevOps practices

1.2.3 Building a culture of collaboration and communication between development and operations teams.

1.2.4. The role of automation and monitoring in enhancing team efficiency.

Differences between DevOps and traditional Software development and IT operations.

1) Collaboration and Communication.

i) Traditional Approach:

Development and IT operations teams work in silos. Developers focus on writing code, and operations teams are responsible for deploying and maintaining the application. This often leads to miscommunication, delays, and a lack of shared understanding.

ii) DevOps Approach:

DevOps encourages continuous collaboration and communication between development and operations teams. Both teams work together throughout the software development lifecycle, fostering a culture of shared responsibility.

2) Process and Workflow:

i) Traditional Approach: Uses a Sequential

elopment process (e.g. Waterfall model) where each phase must be completed before the next begins. This can create ttle necks and slow down the process.

) DevOps Approach: Follows an agile and iterative approach where development, sting, and deployment are done ntinuously and concurrently. This helps dentify and fix issues earlier in the development process.

Water fall Model.

y) It can make your projects flow smoothly, void bottle necks, help you hit deadlines, nsures deliverables are met before the next phase begins, and allow the team verall to shine with perfection. This in-depth guide analyses the advantages of the waterfall methodology.

Requirement Gathering & Analysis

↓

System Design

↓

Implementation.

Testing.

↓

Development Deployment.

↓

Maintainance.

Agile:

Agile developement is important because it helps to ensure that development teams complete projects on time and within budget. It also helps to improve communication between the development team and the product ownner. Additionally, Agile development methodology can help reduce the risks associated with complex projects.