UTT

UNIVERSIDAD TECNOLÓGICA DE TIJUANA

GOBIERNO DE BAJA CALIFORNIA

Assignment:

Introduction to DevOps

BY:

Daniel Chavez Madrigal

GROUP:

9-B

SUBJECT:

Software development process management

PROFESSOR:

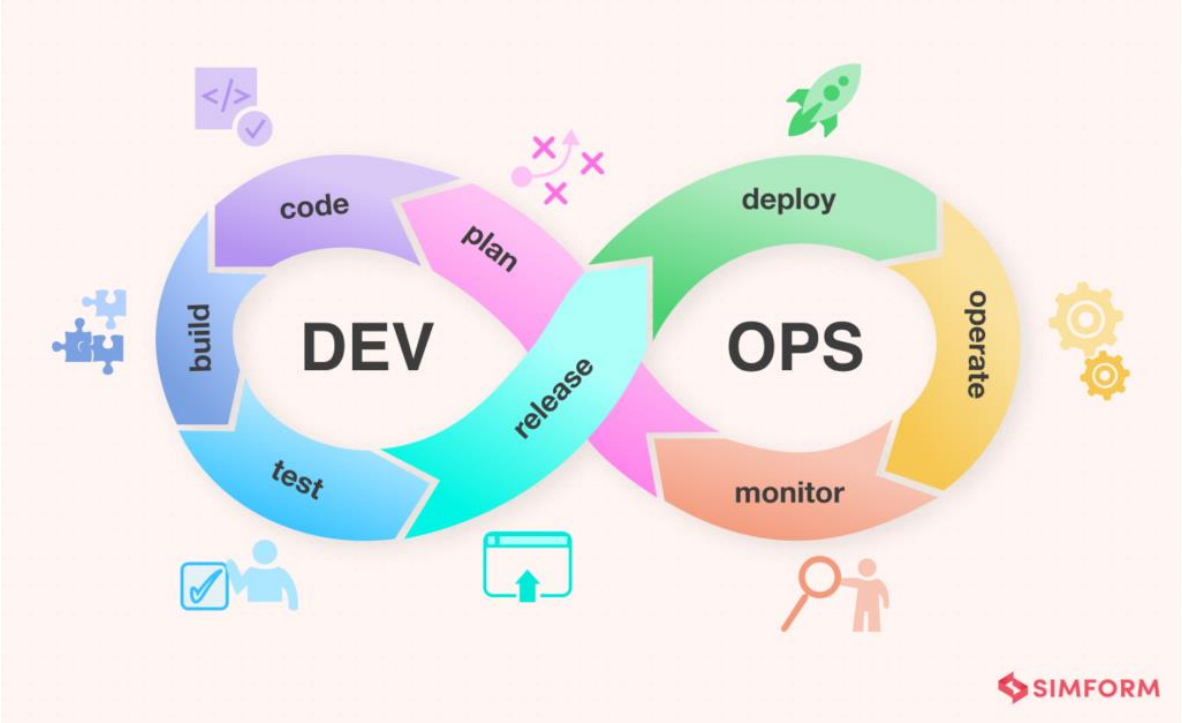
Ray Brunett Parra Galaviz

Tijuana, Baja California, 10 de enero del 2025

DevOps is a set of practices that combines software development (Dev) and IT operations (Ops). It aims to shorten the systems development cycle and provide continuous delivery with high software quality. Here are some key points to understand:

- **Culture of Collaboration**: DevOps emphasizes a culture of collaboration between development and operations teams. This involves breaking down silos and fostering a shared responsibility for the success of the software.
- **Continuous Integration and Continuous Delivery (CI/CD)**: These are key practices in DevOps. Continuous Integration involves frequently merging code changes into a central repository, while Continuous Delivery ensures that the code is always in a deployable state.
- **Infrastructure as Code (IaC)**: This practice involves managing and provisioning computing infrastructure through machine-readable definition files, rather than physical hardware configuration or interactive configuration tools.
- **Automation**: Automation is a core principle of DevOps, aiming to automate repetitive tasks to improve efficiency and reduce the risk of human error.
- **Monitoring and Logging**: Continuous monitoring and logging are essential to gain insights into the performance and health of applications and infrastructure.

DevOps lifecycle:



1. Planning

In this phase, teams define the project requirements and plan the development process. This includes setting goals, creating a roadmap, and identifying the necessary tools and technologies.

2. Development

During development, the actual coding takes place. Developers write code, perform unit tests, and ensure that the code meets the defined requirements. This phase often involves using version control systems like Git to manage code changes.

3. Continuous Integration (CI)

Continuous Integration involves merging code changes into a shared repository frequently. Automated builds and tests are run to detect issues early. This helps in maintaining a consistent and functional codebase.

4. Continuous Testing

In this phase, automated tests are executed to ensure the quality of the code. Continuous testing helps identify bugs and issues before the code is deployed to

production. Tools like Selenium and JUnit are commonly used for automated testing.

5. Continuous Deployment (CD)

Continuous Deployment automates the release of code to production. Once the code passes all tests, it is automatically deployed to the production environment. This ensures that new features and updates are delivered quickly and reliably.

6. Monitoring and Logging

After deployment, continuous monitoring and logging are essential to track the performance and health of the application. Monitoring tools like Prometheus and Grafana help in identifying and resolving issues in real-time.

7. Feedback and Optimization

Feedback from users and stakeholders is gathered to improve the application. This phase involves analyzing the feedback, identifying areas for improvement, and optimizing the application accordingly. The cycle then repeats, incorporating the feedback into the planning phase.

References:

Dhaduk, H. (2023, July 31). DevOps Lifecycle: 7 Phases Explained in Detail with Examples. *Simform - Product Engineering Company*. https://www.simform.com/blog/devops-lifecycle/

Wwlpublish. (n.d.). *Introduction to DevOps - Training*. Microsoft Learn. https://learn.microsoft.com/en-us/training/modules/introduction-to-devops/