Name of Student: Ravikumar Rai			
Roll Number: 42		**LAB Assignment Number:** 1	
Title of LAB Assignment: Case Study on DevOps: Principles, Practices, and DevOps Engineer Role and Responsibilities			
DOP: 18-01-2024		**DOS:** 24-01-2024	
CO **Mapped:** CO1	**PO** **Mapped:** PO2, PO5, PSO1	**Signature:**	**Marks:**

Aim: Case Study on DevOps: Principles, Practices, and DevOps Engineer Role and Responsibilities.

Things to be added in case study.
What is DevOps?
DevOps benefit over existing culture
DevOps architecture and Life Cycle with detail explanation.
Explanation of tools like GIT, GitHub, Jenkin, Docker, Selenium, Ansible, Nagios

What is DevOps?

DevOps, a compound of "development" (Dev) and "operations" (Ops), is a software engineering methodology that aims to integrate the work of development and operations teams. It's a blend of practices and tools designed to improve and shorten the systems development life cycle.

#Here are some key aspects of DevOps:

1. DevOps Model: DevOps helps organizations deliver applications and services efficiently. It keeps development and operations together, working as a team throughout the software development process, from development, testing, deployment, to updates. It also aligns with development, QA, and IT operations efforts.

2. Principles of DevOps: The DevOps methodology follows four key principles:
- Automation of the software development lifecycle.
- Collaboration and communication.
- Continuous improvement and minimization of waste.
- Hyper-focus on user needs with short feedback loops.

3. DevOps Engineer: A DevOps Engineer is an IT professional who works with both development and operations teams.

4. Benefits of DevOps: DevOps speeds the delivery of higher-quality software by combining and automating the work of software development and IT operations teams. It also meets software users' ever-increasing demand for frequent, innovative new features and uninterrupted performance and availability.

In essence, a DevOps engineer is responsible for the smooth operation of a company's IT infrastructure. They work with developers to deploy and manage code changes, and with operations staff to ensure that systems are up and running smoothly.

#DevOps Principles

DevOps is a mindset, a cultural shift, where teams adopt new ways of working. The key principles that help DevOps teams deliver applications and services at a faster pace and higher quality are:

1. Collaboration: Development and operations teams coalesce into a functional team that communicates, shares feedback, and collaborates throughout the entire development and deployment cycle.

2. Automation: An essential practice of DevOps is to automate as much of the software development lifecycle as possible.

3. Continuous Improvement: It's the practice of focusing on experimentation, minimizing waste, and optimizing for speed, cost, and ease of delivery.

4. Customer-centric action: DevOps teams use short feedback loops with customers and end users to develop products and services centered around user needs.

#DevOps Practices

DevOps practices include:

1. Agile Project Management: Agile is an iterative approach to project management and software development that helps teams deliver value to their customers faster and with fewer headaches.

2. Shift Left with CI/CD: When teams "shift left", they bring testing into their code development processes early.

3. Automation: Build with the right tools. A DevOps toolchain requires the right tools for each phase of the DevOps lifecycle, with key capabilities to improve software quality and speed of delivery.

#DevOps Engineer Role and Responsibilities

A DevOps engineer is a professional who combines expertise in software development and IT operations to streamline and automate the process of creating, testing, and deploying software applications. Their responsibilities include:

1. **Understan dingcustomer requirements and project KPIs.**

2. **Implementing** various development, testing, automation tools, and IT infrastructure.

3. **Planning** the team structure, activities, and involvement in project management activities.

4. **Managing** stakeholders and external interfaces.

#DevOps benefit over existing culture

DevOps offers several benefits over traditional IT culture:

1. Faster Deployment: DevOps ensures faster deployment of high-quality software applications.

2. Stable Work Environment: It provides a reliable infrastructure with a lower risk of failures.

3. Improved Productivity: The collaboration of teams in a DevOps culture improves the productivity of the organization.

4. Rapid Problem-Solving: Rapid feedback improves problem-solving ability.

5. Increased Business Agility: DevOps promotes agility in your business.

6. Improved Product Quality: There's a significant improvement in product quality.

7. Innovation: Automation in repetitive tasks leaves more room for innovation.

8. Continuous Delivery: DevOps promotes the continuous delivery of software.

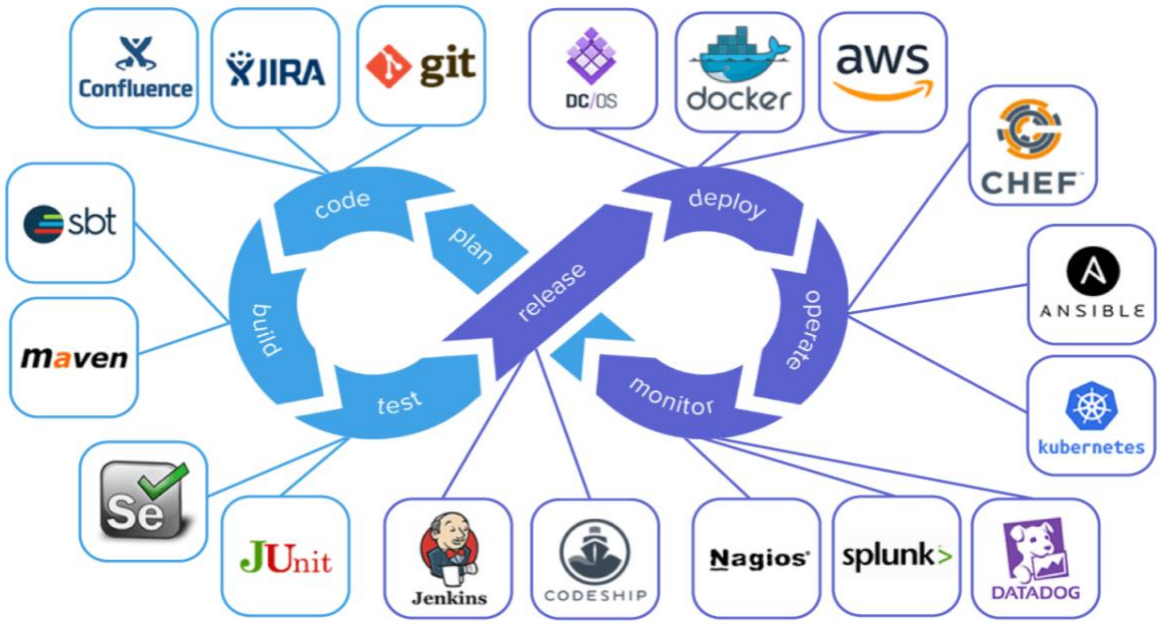
9. Transparency: Transparency leads to high productivity.

DevOps architecture and Life Cycle with detailed explanation.

#DevOps Architecture

DevOps architecture is not a framework for building computing systems. Instead, it's a mindset, a culture, and a philosophy that emphasizes collaboration, communication, and continuous improvement as a means of rapidly and sustainably releasing better software in the most efficient way possible. It enables collaboration between teams, which is one of the essential

features of delivery. It helps in improving the work culture among the teams to remain in sync to understand the status of work related to other teams.

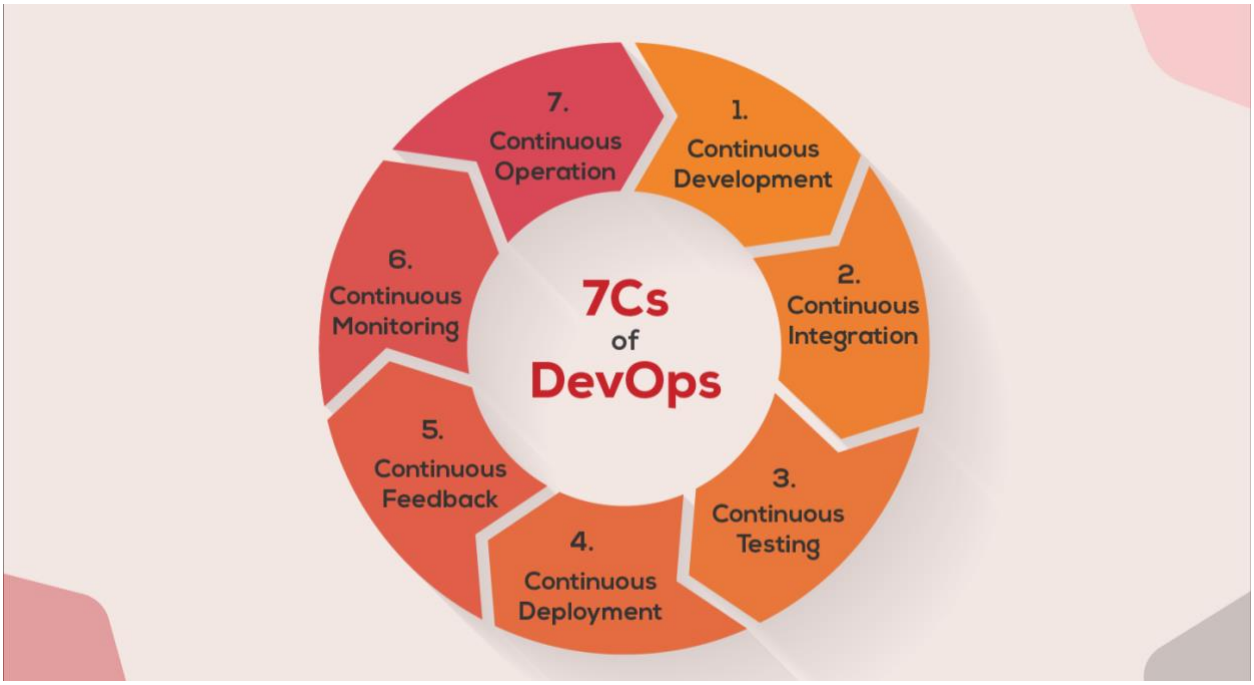


Here are the various components used in the DevOps architecture:

1. Plan: DevOps use Agile methodology to plan the development. With the operations and development team in sync, it helps in organizing the work to plan accordingly to increase productivity.
2. Code: Practices such as Git enable the code to be used, which ensures writing the code for business, helps to track changes, getting notified about the reason behind the difference in the actual and the expected output, and if necessary reverting to the original code developed.
3. Build: With DevOps, the usage of cloud, sharing of resources comes into the picture, and the build is dependent upon the user's need, which is a mechanism to control the usage of resources or capacity.
4. Test: The application will be ready for production after testing. The testing can be automated, which decreases the time for testing so that the time to deploy the code to production can be reduced as automating the running of the scripts will remove many manual steps.
5. Release: The primary goal of the release phase is to ensure that the infrastructure and environment are set up for a successful launch of the updated software. This can include spinning up or down servers, updating operating system configurations, or setting up any other necessary infrastructure to support the application.
6. Deploy: Many systems can support the scheduler for automated deployment.
7. Operate: This stage is by no means the final step. Rather, it informs future development cycles and manages the configuration of the production environment and the implementation of any runtime requirements.
8. Monitor: Continuous monitoring is used to identify any risk of failure. Also, it helps in tracking the system accurately so that the health of the application can be checked.

#DevOps Lifecycle

The DevOps lifecycle is a series of automated development processes or workflows within an iterative development lifecycle. It consists of distinct phases, including planning, coding, testing, deployment, monitoring, and feedback. Here are the key phases:



1. Continuous Development: In Continuous Development, code is written in small, continuous bits rather than all at once. This improves efficiency every time a piece of code is created, it is tested, built, and deployed into production.
2. Continuous Integration: Continuous Integration can be explained mainly in 4 stages in DevOps. They are as follows: Getting the SourceCode from SCM, Building the code, Code quality review, Storing the build artifacts.
3. Continuous Testing: The application will be ready for production after testing. In the case of manual testing, it consumes more time in testing and moving the code to the output.
4. Continuous Deployment/Delivery: Many systems can support the scheduler for automated deployment. The cloud management platform enables users to capture accurate insights and view the optimization scenario, analytics on trends by the deployment of dashboards.
5. Continuous Monitoring: Continuous monitoring refers to the process and technology required to incorporate monitoring across each phase of DevOps and IT operations lifecycles.
6. Continuous Feedback: This phase involves getting feedback from the end-users and making necessary changes and improvements.
7. Continuous Operations: This phase ensures that the updated application is correctly deployed in the production environment.

Explanation of tools like GIT, GitHub, Jenkin, Docker, Selenium, Ansible, Nagios

1. Git: Git is an open-source, command-line IT automation software application written in Python. It can configure systems, deploy software, and orchestrate advanced workflows to support application deployment, system updates, and more.

2. GitHub: GitHub is a hosting platform used by developers to save their code online and track its changes to see what will work when it comes to them working on their projects. GitHub is popular among people in the IT industry due to its easy-to-understand UI and its ability to act as a progress tracker using a Version Control System to track changes.

3. Jenkins: Jenkins is a tool that is used for automation, and it is an open-source server that allows all the developers to build, test and deploy software. It works or runs on java as it is written in java.

4. Docker: Docker is an open-source project for automating the deployment of applications as portable, self-sufficient containers that can run on the cloud or on-premises. Docker is also a company that promotes and evolves this technology, working in collaboration with cloud, Linux, and Windows vendors, including Microsoft.

5. Selenium: Selenium is a free (open-source) automated testing framework used to validate web applications across different browsers and platforms. You can use multiple programming languages like Java, C#, Python, etc to create Selenium Test Scripts.

6. Ansible: Ansible is an open-source, command-line IT automation software that can configure systems, deploy software, and orchestrate workflows. It uses a human-readable language, OpenSSH, and a simple inventory file to connect to and manage remote devices.

7. Nagios: Nagios is a powerful monitoring system that enables organizations to identify and resolve IT infrastructure problems before they affect critical business processes.

Conclusion:

In essence, our discussion revolved around the understanding of DevOps, its principles, practices, benefits, architecture, lifecycle, and the tools used in the DevOps ecosystem. DevOps is about creating a culture of collaboration and shared responsibility, with the ultimate goal of providing value to customers. It’s a key methodology in today’s fast-paced, agile software development environment.