Experiment No. 1

Aim: To understand DevOps: Principles, Practices, and DevOps Engineer Role and Responsibilities.

LO No. & Statement: (LO1): Describe the fundamentals of DevOps engineering and be fully proficient with DevOps terminologies, concepts, benefits, and deployment options to meet your business requirements.

What is DevOps?

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

If a service disruption happens unexpectedly, this may be the result of a team structure issue where developers and operators are working in silos. The structure of these teams restricts collaboration and obscures accountability.

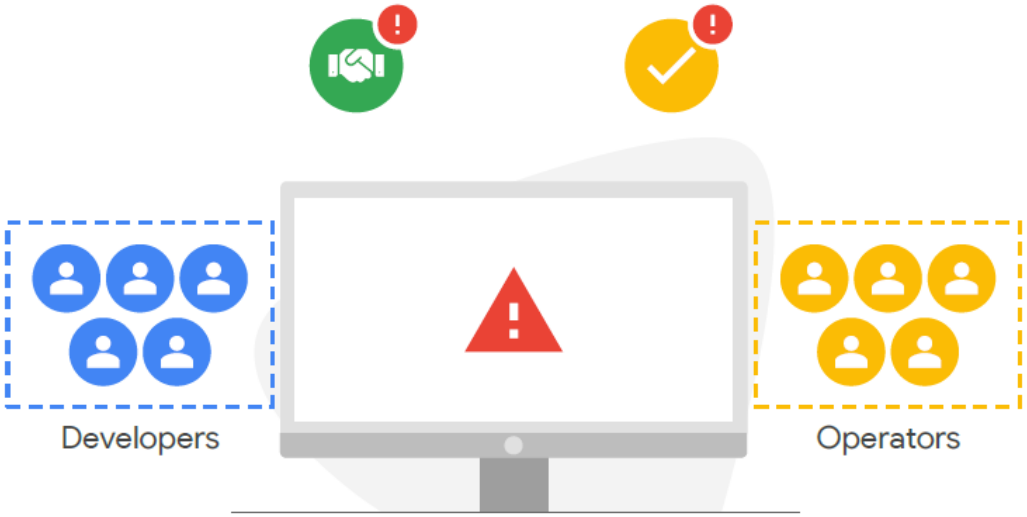

Figure 1: DevOps

A philosophy that seeks to create a more collaborative and accountable culture within developer and operations teams. The philosophy highlights how IT teams can operate but doesn't give explicit guidance on how an organization should implement practices to be successful.

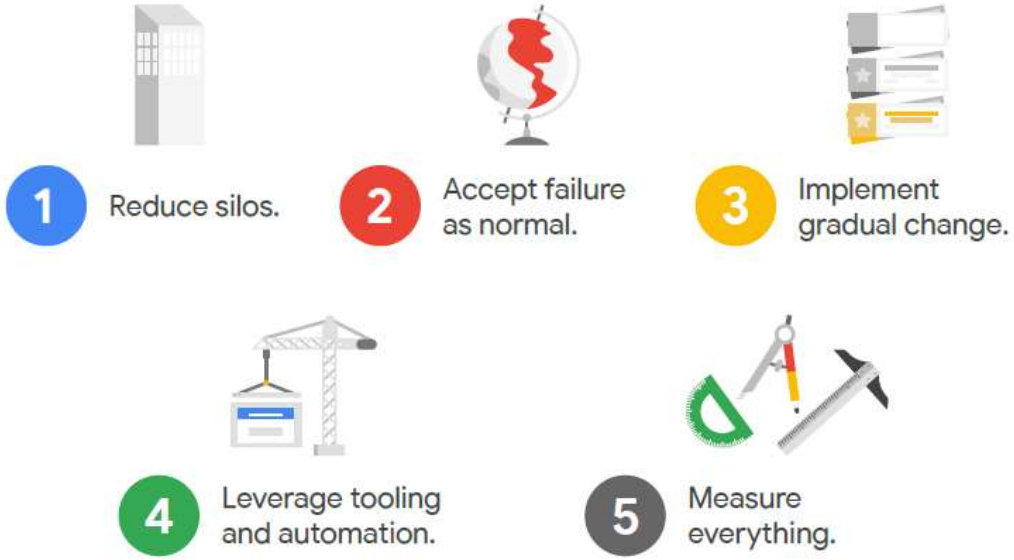

Figure 2: Objective of DevOps

Principles of DevOps:

DevOps is more than just development and operations teams working together. It's more than tools and practices. DevOps is a mindset, a cultural shift, where teams adopt new ways of working.

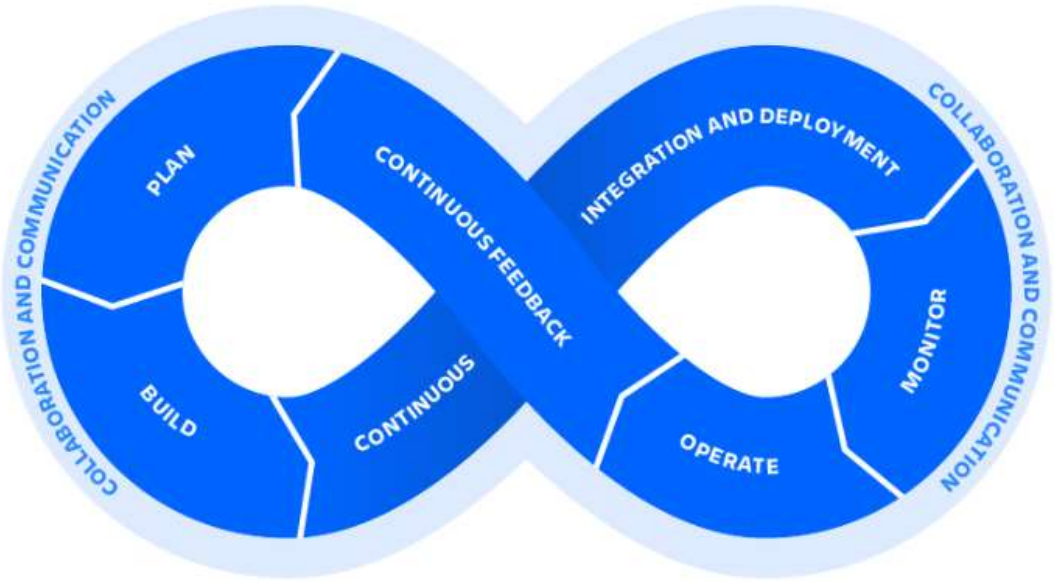

Figure 3: Principles of DevOps

A DevOps culture means developers get closer to the user by gaining a better understanding of user requirements and needs. Operations teams get involved in the development process and add maintenance requirements and customer needs. It means adhering to the following key principles that help DevOps teams deliver applications and services at a faster pace and a higher quality than organizations using the traditional software development model.

1. Collaboration: The key premise behind DevOps is collaboration. Development and operations teams coalesce into a functional team that communicates, shares feedback, and collaborates throughout the entire development and deployment cycle. Often, this means development and operations teams merge into a single team that works across the entire application lifecycle.
2. Automation: An essential practice of DevOps is to automate as much of the software development lifecycle as possible. This gives developers more time to write code and develop new features. Automation is a key element of a CI/CD pipeline and helps to reduce human errors and increase team productivity. With automated processes, teams achieve continuous improvement with short iteration times, which allows them to quickly respond to customer feedback.
3. Continuous Improvement: Continuous improvement was established as a staple of agile practices, as well as lean manufacturing and Improvement Kata. It's the practice of focusing on experimentation, minimizing waste, and optimizing for speed, cost, and ease of delivery. Continuous improvement is also tied to continuous delivery, allowing DevOps teams to continuously push updates that improve the efficiency of software systems. The constant pipeline of new releases means teams consistently push code changes that eliminate waste, improve development efficiency, and bring more customer value.
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 enable rapid collection and response to user feedback through the use of real-time live monitoring and rapid deployment. Teams get immediate visibility into how to live users interact with a software system and use that insight to develop further improvements.
5. Create with the end in mind: This principle involves understanding the needs of customers and creating products or services that solve real problems. Teams shouldn't 'build in a bubble', or create software based on assumptions about how consumers will use the software. Rather, DevOps teams should have a holistic understanding of the product, from creation to implementation.

Practices of DevOps:

Figure 4: Developer Operations

1. Continuous development: This practice spans the planning and coding phases of the DevOps lifecycle. Version-control mechanisms might be involved.
2. Continuous testing: This practice incorporates automated, prescheduled, continued code tests as application code is being written or updated. Such tests can speed the delivery of code to production.
3. Continuous integration (CI): This practice brings configuration management (CM) tools together with other test and development tools to track how much of the code being developed is ready for production. It involves rapid feedback between testing and development to quickly identify and resolve code issues.
4. Continuous delivery: This practice automates the delivery of code changes, after testing, to a preproduction or staging environment. A staff member might then decide to promote such code changes into production.
5. Continuous deployment (CD): Similar to continuous delivery, this practice automates the release of new or changed code into production. A company doing continuous deployment might release code or feature changes several times per day. The use of container technologies, such as Docker and Kubernetes, can enable continuous deployment by helping to maintain consistency of the code across different deployment platforms and environments.
6. Continuous monitoring: This practice involves ongoing monitoring of both the code in operation and the underlying infrastructure that supports it. A feedback loop that reports on bugs or issues then makes its way back to development.
7. Infrastructure as code: This practice can be used during various DevOps phases to automate the provisioning of infrastructure required for a software release. Developers add infrastructure “code” from within their existing development tools. For example, developers might create a storage volume on demand from Docker, Kubernetes, or OpenShift. This practice also allows operations teams to monitor environment configurations, track changes, and simplify the rollback of configurations.

DevOps Engineer Role & Responsibilities:

Developers are responsible for writing code for systems and applications, and operators are responsible for ensuring that those systems and applications operate reliably.

Figure 5: DevOps Engineer Role & Responsibilities

Developers are expected to be agile. They aim to release new functions frequently, increase core business value with new features, and release fixes fast for an overall better user experience. In contrast, operators are expected to keep systems stable, and so they often prefer to work more slowly to ensure reliability and consistency.

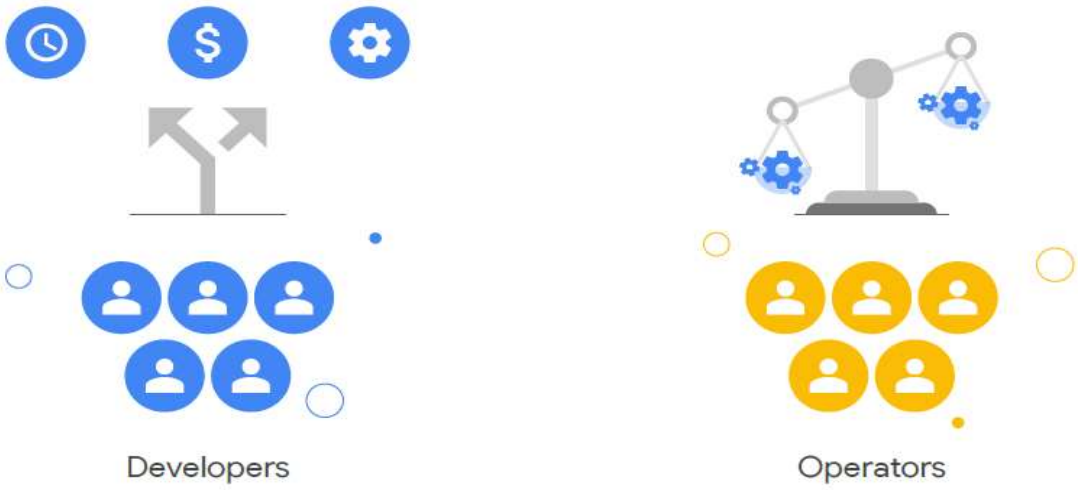

Figure 6: DevOps Engineer Role & Responsibilities

Conclusion:

In this experiment, we have learned about the DevOps, DevOps Principles, Practices, and DevOps Engineer Role and Responsibilities.

We have achieved LO1 from this experiment.

We have also achieved Program Outcomes PO1, PO2, PO3, PO4.