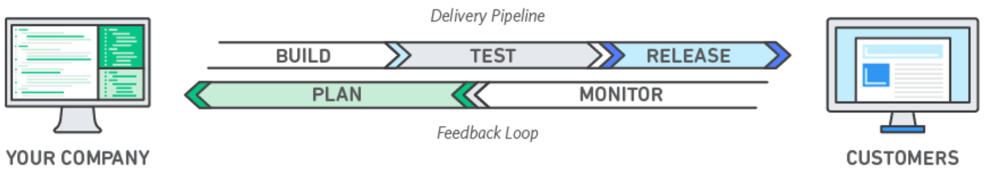
Experiment No.01

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

Theory:

DevOps Model Defined

DevOps is the combination of cultural philosophies, practices, and tools that increases an organization's ability to deliver applications and services at high velocity: evolving and improving products at a faster pace than organizations using traditional software development and infrastructure management processes. This speed enables organizations to better serve their customers and compete more effectively in the market.



How DevOps Works

Under a DevOps model, development and operations teams are no longer "siloed." Sometimes, these two teams are merged into a single team where the engineers work across the entire application lifecycle, from development and test to deployment to operations, and develop a range of skills not limited to a single function.

In some DevOps models, quality assurance and security teams may also become more tightly integrated with development and operations and throughout the application lifecycle. When security is the focus of everyone on a DevOps team, this is sometimes referred to as Develops.

These teams use practices to automate processes that historically have been manual and slow. They use a technology stack and tooling which help them operate and evolve applications quickly and reliably. These tools also help engineers independently accomplish tasks (for example, deploying code or provisioning infrastructure) that normally would have required help from other teams, and this further increases a team's velocity.

DevOps Practices

Continuous Integration

Continuous integration is a software development practice where developers regularly merge their code changes into a central repository, after which automated builds and tests are run. The key goals of continuous integration are to find and address bugs quicker, improve software quality, and reduce the time it takes to validate and release new software updates.

Continuous Delivery

Continuous delivery is a software development practice where code changes are automatically built, tested, and prepared for a release to production. It expands upon continuous integration by deploying all code changes to a testing environment and/or a production environment after the build stage. When continuous delivery is implemented properly, developers will always have a deployment-ready build artifact that has passed through a standardized test process.

Micro services

The micro services architecture is a design approach to build a single application as a set of small services. Each service runs in its own process and communicates with other services through a well-defined interface using a lightweight mechanism, typically an HTTP-based application programming interface (API). Micro services are built around business capabilities; each service is scoped to a single purpose.

You can use different frameworks or programming languages to write micro services and deploy them independently, as a single service, or as a group of services.

Infrastructure as Code

Infrastructure as code is a practice in which infrastructure is provisioned and managed using code and software development techniques, such as version control and continuous integration. The cloud's API-driven model enables developers and system administrators to interact with infrastructure programmatically, and at scale, instead of needing to manually set up and configure resources. Thus, engineers can interface with infrastructure using code-based tools and treat infrastructure in a manner similar to how they treat application code. Because they are defined by code, infrastructure and servers can quickly be deployed using standardized patterns, updated with the latest patches and versions, or duplicated in repeatable ways.

Configuration Management

Developers and system administrators use code to automate operating system and host configuration, operational tasks, and more. The use of code makes configuration changes repeatable and standardized. It frees developers and systems administrators from manually configuring operating systems, system applications, or server software.

Policy as Code

With infrastructure and its configuration codified with the cloud, organizations can monitor and enforce compliance dynamically and at scale. Infrastructure that is described by code can thus be tracked, validated, and reconfigured in an automated way. This makes it easier for organizations to govern changes over resources and ensure that security measures are properly enforced in a distributed manner (e.g., information security or compliance with PCIDSS or HIPAA). This allows teams within an organization to move at higher velocity since noncompliant resources can be automatically flagged for further investigation or even automatically brought back into compliance.

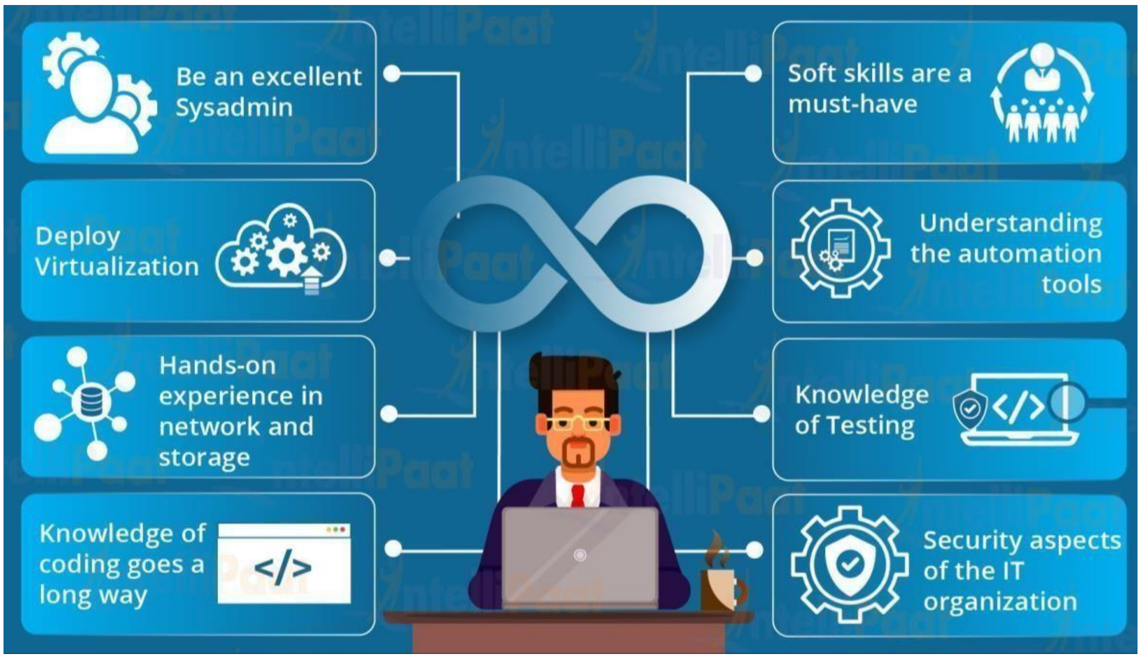
Monitoring and Logging

Organizations monitor metrics and logs to see how application and infrastructure performance impacts the experience of their product's end user. By capturing, categorizing, and then analysing data and logs generated by applications and infrastructure, organizations understand how changes or updates impact users, shedding insights into the root causes of problems or unexpected changes. Active monitoring becomes increasingly important as services must be available 24/7 and as application and infrastructure update frequency increases. Creating alerts or performing real-time analysis of this data also helps organizations more proactively Monitor their services.

Communication and Collaboration

Increased communication and collaboration in an organization is one of the key cultural aspects of DevOps. The use of DevOps tooling and automation of the software delivery process establishes collaboration by physically bringing together the workflows and responsibilities of development and operations. Building on top of that, these teams set strong cultural norms around information sharing and facilitating communication through the use of chat applications, issue or project tracking systems, and wikis. This helps speed up communication across developers, Operations, and even other teams like marketing or sales, allowing all parts of the organization to align More closely on goals and projects.

DevOps Engineer Role


Be an excellent Sysadmin
Deploy Virtualization
Hands-on experience in network and storage
Knowledge of coding goes a long way
Soft skills are a must-have
Understanding the automation tools
Knowledge of Testing
Security aspects of the IT organization

- **Application and infrastructure planning, testing and development**

The entire DevOps team is in charge of application and infrastructure planning, testing and development. Sharing the responsibility for the development and release pipeline leads to more reliable services. Being part of a DevOps team doesn't mean you have an entire team made up of "DevOps Engineers" – but, a DevOpsoriented company will spread the accountability for application development, testing and release across the entire engineering and IT organization.

- **Maintaining CI/CD pipelines**

Being part of a DevOps team means you have a responsibility for building a CI/CD pipeline and optimizing processes, people and tooling. DevOps-minded engineers will see ways they can constantly improve the pipeline – from people to processes. The team will shift testing and QA further left into the development cycle, allowing the team to continuously test without restricting speed.

- **Automation implementation**

As we mentioned above, automation is a core principle of DevOps. So, the implementation of said automation clearly falls on the shoulders of DevOps teams. It's the responsibility of everyone from the data team to the frontend team to automate tasks and improve the efficiency of engineering and IT. By constantly automating mundane tasks, you're able to focus more on strategic development and driving business value.

- **On-call, incident response and incident management**

With shared accountability and code ownership, DevOps teams need to take on-call responsibilities and incident management work. But, the more time the DevOps team spends responding to incidents in production, the more they learn about their systems. So, over time, developers start to write code that better fits into their applications and infrastructure – leading to fewer incidents.

And, IT teams gain more influence in the development lifecycle, helping them proactively deepen the reliability of services being deployed.

- **Monitoring**

Last but not least, DevOps teams are responsible for the implementation of actionable monitoring solutions. The organization needs to collect data and know how they can take action with it. The DevOps team (aka everyone) is responsible for exposing blind spots in their applications and infrastructure, and then figuring out how they can monitor those services. Monitoring is just one small step into building highly observable systems – but it's an important start for building reliable systems.

Conclusion:

DevOps is important because it's a software development and operations approach that enables faster development of new products and easier maintenance of existing deployments. DevOps teams monitor the entire development lifecycle, from planning, development, integration and testing, deployment, and operations. This allows teams to respond to any degradation in the customer experience, quickly and