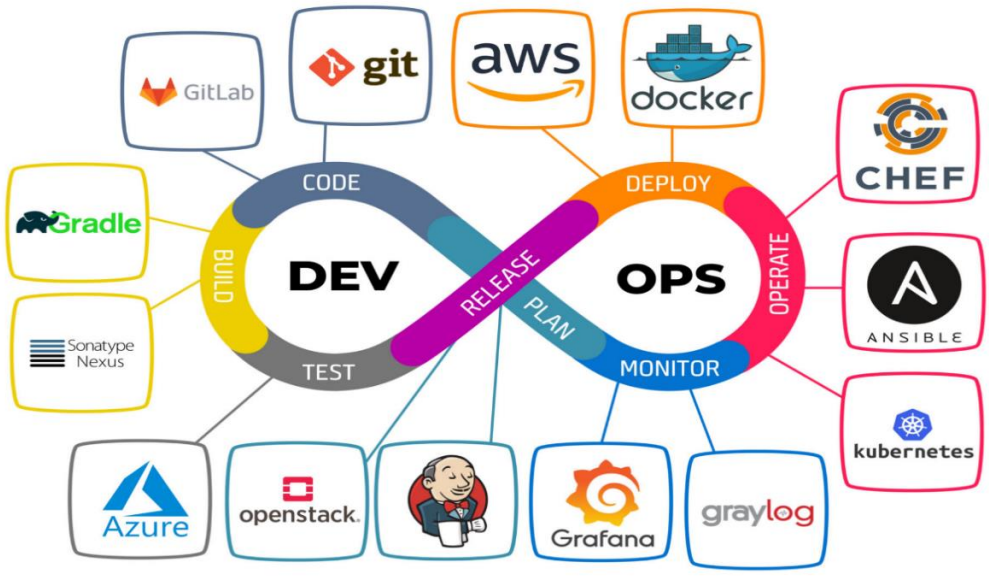
EXPERIMENT NO :1

AIM: TO Understand DevOps: Principles, Practices and DevOps Engineer Role and Responsibilities

DevOps:

DevOps is a combination of software development(dev)and operations(ops). It is defined as a software engineering methodology which aims to integrate the work of development teams and operations teams by facilitating a culture of collaboration and shared responsibilities.



1)Puppet

Puppet is the most widely used DevOps tool. It allows the delivery and release of the technology changes quickly and frequently. It has features of versioning, automated testing, and continuous delivery. It enables to manage entire infrastructure as code without expanding the size of the team.

Features

- Real-time context-aware reporting.
- Model and manage the entire environment.
- Defined and continually enforce infrastructure.
- Desired state conflict detection and remediation.

2) Ansible

Ansible is a leading DevOps tool. Ansible is an open-source IT engine that automates application deployment, cloud provisioning, intra service orchestration, and other IT tools. It makes it easier for DevOps teams to scale automation and speed up productivity.

Ansible is easy to deploy because it does not use any agents or custom security infrastructure on the client-side, and by pushing modules to the clients. These modules are executed locally on the client-side, and the output is pushed back to the Ansible server.

Features

- It is easy to use to open source deploy applications.
- It helps in avoiding complexity in the software development process.
- It eliminates repetitive tasks.
- It manages complex deployments and speeds up the development process.

3) Docker

Docker is a high-end DevOps tool that allows building, ship, and run distributed applications on multiple systems. It also helps to assemble the apps quickly from the components, and it is typically suitable for container management.

Features

- It configures the system more comfortable and faster.
- It increases productivity.
- It provides containers that are used to run the application in an isolated environment.

4) Nagios

Nagios is one of the more useful tools for DevOps. It can determine the errors and rectify them with the help of network, infrastructure, server, and log monitoring systems.

Features

- It provides complete monitoring of desktop and server operating systems.
- The network analyser helps to identify bottlenecks and optimize bandwidth utilization.
- It helps to monitor components such as services, application, OS, and network protocol.
- It also provides to complete monitoring of Java Management Extensions.

5) CHEF

A chef is a useful tool for achieving scale, speed, and consistency. The chef is a cloud-based system and open source technology. This technology uses Ruby encoding to develop essential building blocks such as recipes and cookbooks. The chef is used in infrastructure automation and helps in reducing manual and repetitive tasks for infrastructure management.

Chef has got its convention for different building blocks, which are required to manage and automate infrastructure.

Features

- It maintains high availability.
- It can manage multiple cloud environments.
- It uses popular Ruby language to create a domain-specific language.
- The chef does not make any assumptions about the current status of the node. It uses its mechanism to get the current state of the machine.

6) Jenkins

Jenkins is a DevOps tool for monitoring the execution of repeated tasks. Jenkins is a software that allows continuous integration. Jenkins will be installed on a server where the central build will take place. It helps to integrate project changes more efficiently by finding the issues quickly.

Features

- Jenkins increases the scale of automation.
- It can easily set up and configure via a web interface.
- It can distribute the tasks across multiple machines, thereby increasing concurrency.
- It supports continuous integration and continuous delivery.
- It offers 400 plugins to support the building and testing any project virtually.
- It requires little maintenance and has a built-in GUI tool for easy updates.

7) Git

Git is an open-source distributed version control system that is freely available for everyone. It is designed to handle minor to major projects with speed and efficiency. It is developed to co-ordinate the work among programmers. The version control allows you to track and work together with your team members at the same workspace. It is used as a critical distributed version-control for the DevOps tool.

Features

- It is a free open source tool.
- It allows distributed development.
- It supports the pull request.
- It enables a faster release cycle.

- Git is very scalable.

8) SALTSTACK

Stalkily is a lightweight DevOps tool. It shows real-time error queries, logs, and more directly into the workstation. SALTSTACK is an ideal solution for intelligent orchestration for the software-defined data centre.

Features

- It eliminates messy configuration or data changes.
- It can trace detail of all the types of the web request.
- It allows us to find and fix the bugs before production.
- It provides secure access and configures image caches.

9) Splunk

Splunk is a tool to make machine data usable, accessible, and valuable to everyone. It delivers operational intelligence to DevOps teams. It helps companies to be more secure, productive, and competitive.

Features

- It has the next-generation monitoring and analytics solution.
- It delivers a single, unified view of different IT services.
- Extend the Splunk platform with purpose-built solutions for security.
- Data drive analytics with actionable insight.

10) Selenium

Selenium is a portable software testing framework for web applications. It provides an easy interface for developing automated tests.

Features

- It is a free open source tool.
- It supports multiplatform for testing, such as Android and ios.
- It is easy to build a keyword-driven framework for a WebDriver.

DevOps Lifecycle Phases:

1.Plan

In this phase, the team collaborates with the business stakeholders to define the requirements for the project and identify the features that will be delivered.

The goal is to clearly understand the needs and expectations of the stakeholders.This phase involves creating a roadmap, defining milestones, and setting project goals. An example of this phase in action could be a team using a project management tool like Jira to create a backlog of user stories and tasks.

2. Code

In this phase, the development team creates the code for the application. This involves using best practices to write high-quality, maintainable code that can be easily tested.

The team may use a version control system like Git to manage the codebase. An example of this phase in action could be a team using an integrated development environment (IDE) like Visual Studio Code/ Eclipse to write and debug code.

3. Build

In this phase, the code is compiled and built into executable binaries or packages. This phase includes tasks such as compiling code, running unit tests, and packaging the application.

An example of this phase in action could be a team using a tool like Jenkins / GitLab to automate the build process, ensuring that code changes are integrated and tested regularly.

4. Test

In this phase, the application is tested to ensure that it meets the quality standards and requirements. This includes testing at various levels, such as unit tests, integration tests, and acceptance tests.

An example of this phase in action could be a team using a test automation tool/ platforms like TestSigma/Selenium/Playwright/Cypress to run automated tests for the web application.

5. Release

In this phase, the application code undergoes a final check for production readiness. If all requirements are met and any identified issues are resolved, the project proceeds to the deployment phase.

An example of this phase in action could be a team using Jenkins / Bamboo that automates the entire release process.

6. Deploy

In this phase, the application is deployed to the desired environment which can be stage, UAT or production environment. This involves setting up the infrastructure, configuring the environment, and deploying the application code.

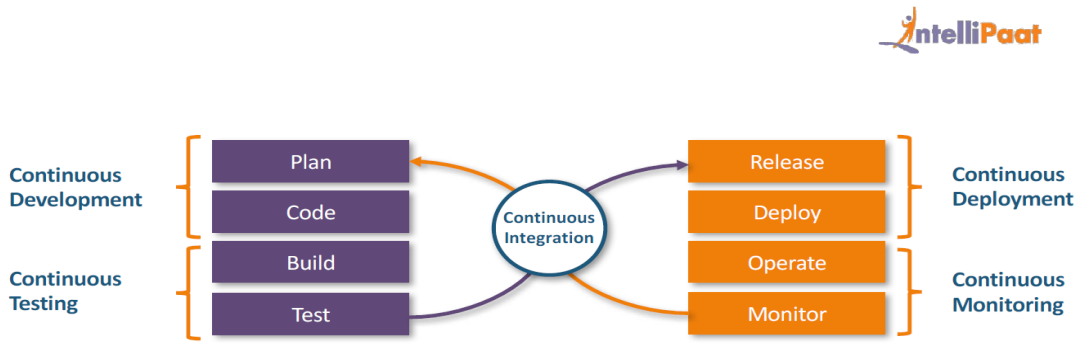
7. Operate

In this phase, the application is monitored and maintained in the production environment. Operations teams monitor the application's performance, availability, and security, and respond to any incidents or issues that arise.

8. Monitor

In this phase, the application is monitored and maintained in the production environment. This includes tasks such as monitoring performance, tracking user behavior, and troubleshooting issues.

CI/CD



Defination:

A continuous integration and continuous deployment (CI/CD) pipeline is a series of steps that must be performed in order to deliver a new version of software. CI/CD pipelines are a practice focused on improving software delivery throughout the software development life cycle via automation.

What is Continuous Integration?

Continuous integration is a process that integrates tools involved in the DevOps lifecycle. The continuous integration tool pulls out the code from the source code management system. When a developer updates the code, the code is built and tested.

Upon successfully passing the testing stage continuous integration tool passes the software to the production server where it is released to the customer. The monitoring tool keeps a log of

customer feedback, the complete process is automated using tools and integrated using a continuous integration tool.

What is Continuous deployment?

Continuous deployment goes one step further than continuous delivery. With this practice, every change that passes all stages of your production pipeline is released to your customers. There's no human intervention, and only a failed test will prevent a new change to be deployed to production.

Continuous deployment is an excellent way to accelerate the feedback loop with your customers and take pressure off the team as there isn't a "release day" anymore. Developers can focus on building software, and they see their work go live minutes after they've finished working on it.

What is continuous testing?

Continuous testing can be defined as undisrupted testing, meaning little-to-no human intervention, that is carried out on a frequent, repeated basis.

Continuous testing is about testing throughout the release pipeline, as soon as a piece of code is delivered by the developer, instead of testing at the very end of a development cycle, as it is done in a more traditional, waterfall approach.

What is Continuous Monitoring?

Continuous Monitoring (CM) is a Lifecycle feature that automatically scans configured applications once a day and sends notifications if new policy violations are discovered. The primary use case of this feature is to monitor deployed applications that are not built regularly. It’s a powerful tool that can keep you informed about your application’s security and health.

What is Continuous Development?

This stage involves committing code to version controls such as Git or SVN for maintaining the different versions of the code, and tools like Ant, Maven, Gradle for building/packaging the code into an executable file that can be forwarded to the QAs for testing.