AET’s

Atharva College of Engineering, Malad(W)

Department of Information Technology Academic Year 2021-22

| **Subject**: - DevOps (ITL503) | | | |
| --- | --- | --- | --- |
| **Year: -** TE IT | **Class: -**IT-2 | | **Batch: -**I2 |
| **Roll No: -**35 | **Student Name: -**Bablu Gupta | | |
| **Date of Experiment: -**29.07.22 | | **Date of submission: -** 29.07.22 | |

# Experiment No-1

**Title: -**Introduction to DevOps.

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

# Theory: -

**What is DevOps?**

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.

**Different Phases in DevOps**

DevOps lifecycle is a combination of different phases of continuous software development, integration, testing, deployment, and monitoring. A competent DevOps lifecycle is necessary to leverage the full benefits of the DevOps methodology.

The DevOps approach embraces continuous innovation, agility, and scalability to build, test, consume, and evolve software products. It promotes a culture of experimentation, feedback, and constant learning to reinvent products, services, and processes. However, to implement DevOps, a proper understanding of different phases of the DevOps lifecycle is crucial.

**Advantages/Benefits of DevOps**

* Renews focus on the customers.
* Unites teams for faster product shipments.
* Simplifies development focus.
* Introduces automation to the development process.
* Supports end-to-end responsibility.

**DevOps**

* **Continuous Integration-**

Continuous integration is aDevOps software development practice where developers regularly merge their code changes into a central repository, after which automated builds and tests are run. Continuous integration most often refers to the build or integration stage of the software release process and entails both an automation component (e.g. a CI or build service) and a cultural component (e.g. learning to integrate frequently). 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 (CD) is an approach to software engineering based on producing software in short cycles. By developing in short cycles, teams can reliably release their software at any time. With CD, development teams can build, test, and release software faster and more frequently. As a result, they can reduce the cost, time, and risk of delivering each change.

* **Continuous Deployment-**

Continuous deployment is basically when teams rely on a fully-automated pipeline. This practice fully eliminates any manual steps and automates the entire process. Therefore, continuous deployment ensures that code is continuously being pushed into production.

* **Continuous Testing-**

As the term suggests, Continuous Testing refers to the execution of automated tests that are carried out at regular intervals every time code changes are made. These tests are conducted as a part of the software delivery pipeline to drive faster feedback on recent changes pushed to the code repository.

* **Continuous Monitoring-**

Continuous monitoring in DevOps is the process of identifying threats to the security and compliance rules of a software development cycle and architecture. Also known as continuous control monitoring or CCM, this is an automated procedure that can be extended to detect similar inconsistencies in IT infrastructures. Continuous monitoring helps business and technical teams determine and interpret analytics to solve crucial issues, as mentioned above, instantaneously.

* **Version control-**

Version control, also known as source control, is the practice of tracking and managing changes to software code. Version control systems are software tools that help software teams manage changes to source code over time.

* **Containerization-**

Containerization is the process of packaging software code, its required dependencies, configurations, and other detail to beeasily developed in the same or another computing environment. In simpler terms, containerization is the encapsulation of an application and its required environment.

* **Microservices-**

Microservices architecture is tailor-made for DevOps with its services-based approach that allows organizations to break down the application into smaller services. This enables delivery teams to tackle individual services as separate entities—ultimately simplifying the development, testing, and deployment.

* **Infrastructure as a code-**

Infrastructure as Code enables DevOps teams to test applications in production-like environments early in the development cycle. These teams expect to provision multiple test environments reliably and on demand. Infrastructure represented as code can also be validated and tested to prevent common deployment issues.

**Different Tools used in DevOps**

1. Jenkins is an open source and free automation server that helps automate software development processes such as building, facilitating CI/CD, deploying, and testing.
2. A leader in software containerization,Docker is used by 11 million+ developers across the world. Solomon Hykes is its original author and it was released in 2013 by Docker, Inc.
3. Developed by Puppet, Inc. and founded in 2005 by Luke Kanies,Puppet is an open source tool for software configuration management.
4. Accelerate your software development productivity usingGradle. This is also amongst open source DevOps tools for build automation, especially for multi-language application or software development.

**DevOps Engineer Role and Responsibilities**

Typical responsibilities for devOps engineers include:

* building and setting up new development tools and infrastructure
* understanding the needs of stakeholders and conveying this to developers
* working on ways to automate and improve development and release processes
* testing and examining code written by others and analysing results
* ensuring that systems are safe and secure against cybersecurity threats
* identifying technical problems and developing software updates and ‘fixes’
* working with software developers and software engineers to ensure that development follows

**Conclusion:** By using the following information we understood DevOps: Principles, Practices, and DevOps Engineer Role and Responsibilities.