Name of Student: Pushkar Sane		
Roll Number: 45	**Lab Assignment Number: 1**	
Title of Lab Assignment: To understand DevOps:Principles, Practices, andDevOps Engineer Role and Responsibilities.		
DOP: 23-01-2024	**DOS: 24-01-2024**	
CO Mapped: **CO1**	**PO Mapped:** **PO2, PO5, PSO1**	**Signature:**

Practical No. 1

What is DevOps?

The term "DevOps," which stands for "Development and Operations," refers to a collection of procedures, tenets, and cultural ideologies that are intended to enhance coordination and dialogue between teams working on software development (Dev) and IT operations (Ops). Streamlining and improving the software delivery process to make it more effective, dependable, and responsive to business requirements is the main objective of DevOps.

Compared to traditional processes, software development and delivery may be done more quickly, securely, and efficiently with the help of DevOps, which merges development and operations. Businesses and their clients benefit from a more agile software development lifecycle and competitive advantages.

DevOps' benefit over existing culture:

By encouraging a collaborative and integrated approach throughout the software development lifecycle, DevOps represents a revolutionary step forward from traditional software development and operations cultures. DevOps dramatically reduces time to market by focusing on automation, continuous integration, and continuous deployment. This helps companies to provide new features and upgrades more quickly and consistently. The development and operations teams' collaborative culture dismantles organizational silos and promotes easy communication and shared accountability. Increased software quality and dependability results from automated testing and monitoring procedures, which also lower deployment failure rates and improve resource efficiency. Additionally, DevOps fosters a culture of continuous improvement by encouraging flexibility, scalability, and an increased emphasis on client happiness.

DevOps architecture and Life Cycle:

DevOps design aims to foster seamless collaboration between development and operations teams, leveraging the integration of various technologies and methodologies. By implementing continuous integration, continuous deployment, version control, and infrastructure as code, organizations can streamline workflows, enhance communication, and achieve faster, more reliable software delivery. This approach promotes a culture of collaboration, automation, and efficiency throughout the entire software development lifecycle.

Here are the following are the main elements of a DevOps architecture:

1. **Version Control:** Goal: Monitors and controls modifications to the source code.
   Tools: Mercurial, SVN, and Git.
2. **Continuous Integration (CI):** Goal: Automates code change testing and development.
   Tools: GitLab CI, Travis CI, and Jenkins.
3. **Continuous Deployment (CD):** Goal: Automates the release of code updates to the live environment.
   Tools: Chef, Ansible, Puppet, Docker, and Kubernetes.
4. **Constant Monitoring: Goal:** Tracks infrastructure and application performance.
   Tools: ELK Stack, Grafana, and Prometheus.
5. The goal of Infrastructure as Code (IaC) is to use code to define and manage infrastructure.
   Tools: Ansible, CloudFormation, and Terraform.
6. **Collaboration and Communication:** Goal: Helps teams communicate and work together.
   Resources: Jira, Microsoft Teams, and Slack.
7. **Containerization:** Purpose: Encapsulates applications and their dependencies.
   Tools: Podman and Docker.
8. **Orchestration:** Oversees and streamlines the deployment of apps that are containerized.
   Tools: Docker Swarm, Kubernetes.

DevOps Life Cycle

Planning, development, testing, deployment, and monitoring are all included in the DevOps lifecycle. Throughout these phases, it places a strong emphasis on ongoing cooperation, automation, and feedback loops, promoting an efficient and iterative method of software delivery that guarantees quick development, top-notch releases, and ongoing improvement.

There are multiple interrelated stages that make up the DevOps lifecycle:

1. **Planning:** Teams work together to prioritize activities, set project goals, and schedule development cycles. The groundwork for an organized and effective workflow is laid at this step.
2. **Development:** Code is written by developers with the intended features and enhancements in mind. A centralized and well-organized codebase is ensured by version control systems, which aid in change management.
3. **Testing:** Tools for automated testing are used to evaluate the security, performance, and functionality of the code. Early problem detection and resolution in the development process is facilitated by continuous testing.
4. **Integration:** Coders merge changes into a common repository on a regular basis as part of continuous integration (CI) processes. The bundled code operates without a hitch thanks to automated build and test procedures.
5. **Deployment:** Code updates are automatically released to production environments through Continuous Deployment (CD). During this phase, programs must be deployed consistently and reliably to reduce downtime and possible problems.
6. **Monitoring:** Real-time application and infrastructure performance is tracked via continuous monitoring technologies. This phase ensures that problems are found early, which allows for quick fixes and ongoing development.
7. **Feedback:** Continuous feedback loops are essential, both within and across teams. Retrospective studies, user input, and monitoring offer valuable insights for process improvement and future development cycle optimization.

DevOps Tools:

1. **GIT:** Developed to manage source code changes during software development, Git is a distributed version control system. It enables the management and documentation of codebase modifications by numerous developers working together on projects. Developers are able to work on many branches, integrate their modifications, and keep an extensive version history.
2. **GitHub:** GitHub is an online platform that was developed on top of Git and offers a central area for hosting and working together on Git repositories. It provides tools for project management, pull requests, code reviews, and bug tracking. A popular tool for both private and open-source software projects, GitHub promotes teamwork among development teams.

3. **Jenkins:** Build, test, and deploy code changes with the help of this open-source automation server. It makes continuous integration and delivery (CI/CD) easier by automating tedious processes like deployment, testing, and code builds. Jenkins is extremely configurable for diverse development contexts thanks to its integrations with a wide range of plugins and tools.
4. **Docker:** Docker is a containerization technology that lets developers bundle dependencies and applications into small, lightweight containers. An application may operate consistently regardless of the underlying infrastructure thanks to containers, which offer consistency across several environments. Applications may be deployed, scaled, and versioned more easily with Docker.
5. **Selenium:** An open-source web application testing framework is called Selenium. It enables functional testing, browser interaction automation, and web application behavior verification for developers and testers. Selenium is a potent tool for web application testing and regression testing since it supports a wide range of programming languages and browsers.
6. **Ansible:** Ansible is an open-source automation tool used for job automation, application deployment, and configuration management. It defines automation tasks in an easy-to-read manner. Ansible is simple to set up and scale for managing infrastructure and applications because it runs over SSH and doesn't require agents to be installed on managed systems.
7. **Nagios:** An open-source monitoring and alerting system called Nagios is used to keep an eye on the functionality and condition of IT infrastructure, such as servers, networks, and services. With Nagios's real-time monitoring, alerting, and reporting capabilities, businesses may find and fix problems before they affect the functionality and availability of vital systems.

DevOps Principles:

The core of DevOps is teamwork, as development and operations teams come together to form a single, unified entity that manages the whole development and deployment process. This frequently leads to a combined team managing end-to-end tasks, encouraging a stronger commitment and connection and producing higher-quality outputs.

DevOps relies heavily on automation to streamline the software development lifecycle. Teams may minimize errors, increase productivity, and react quickly to customer input by automating

procedures, including those in CI/CD pipelines. This enables continuous improvement with short iteration durations.

A fundamental component of DevOps is continuous improvement, which complements lean and agile approaches. It emphasizes trial and error, cutting down on waste, and cost and speed optimization. Continuous delivery is a tool that DevOps teams use to deploy changes often, eliminating inefficiencies and increasing customer value.

The focus of DevOps is on customer-centric action, with short feedback loops to enable quick user needs response. Teams may quickly acquire insights into user interactions through real-time monitoring and deployment, which helps to drive ongoing improvements.

"Create with the end in mind" sums up the idea of comprehending client needs and finding practical solutions. A comprehensive understanding of the product life cycle is prioritized by DevOps teams, who also make sure that solutions are customized to real user experiences and avoid making assumptions.

DevOps Engineer Role and Responsibilities:

DevOps Engineer is somebody who understands the Software Development Lifecycle and has the outright understanding of various automation tools for developing digital pipelines. In today's software development environment, a DevOps Engineer is essential, concentrating on improving efficiency and communication between development and operations teams. These experts create and oversee continuous integration and continuous delivery (CI/CD) pipelines, use infrastructure as code to guarantee consistency, and make efficient use of orchestration and containerization. They are in charge of automating the software development lifecycle. In order to assess system performance, handle security concerns through preventative measures, and work on incident response and troubleshooting, DevOps Engineers incorporate monitoring and logging systems. Along with a dedication to capacity planning and scalability, their position encompasses scripting, coding, and version control. By placing a strong emphasis on teamwork, communication, and the smooth integration of development and operations processes to produce high-quality software, DevOps engineers help to foster a culture of continuous improvement.