EXPERIMENT NO: 1

Aim: Introduction to DevOps.

Theory:

**What is DevOps?**

DevOps is a combination of Development and Operations. It is a culture, set of practices, and tools that help organizations deliver applications and services faster and more reliably.

It focuses on automation, collaboration, and continuous delivery.

The goal is to break the silos between software developers and IT operations teams.

**Difference Between Traditional SDLC and DevOps:**

|  |  |  |
| --- | --- | --- |
| Feature | Traditional SDLC | DevOps |
| Teams | Developers & operations work separately | Developers & operations work together |
| Delivery | Slow, step-by-step (Waterfall model) | Continuous integration & delivery |
| Testing | Done after development | Automated and continuous testing |
| Deployment | Manual and less frequent | Automated and frequent |
| Feedback | Delayed | Quick feedback loop |
| Risk Handling | High risk during release | Lower risk due to smaller, faster release |

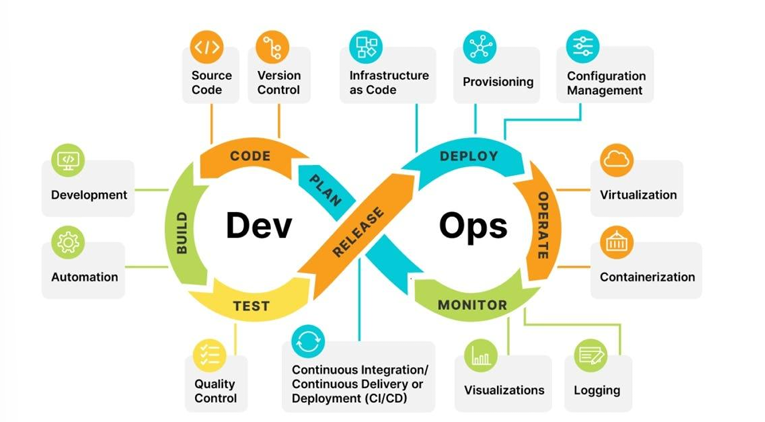


**DevOps Lifecycle & Tools**

The DevOps lifecycle is often represented by an infinity loop, symbolizing the continuous and collaborative nature of software development and operations.

This lifecycle includes 8 main stages: Plan, Code, Build, Test, Release, Deploy, Operate, and Monitor.

Each phase involves specific tools and practices aimed at automating and improving the software delivery process.



Stages Explained

 1. Plan–Define features, requirements, and goals. Tools: Jira, Trello.

 2. Code–Write source code. Tools: Git, GitHub, GitLab.

 3. Build–Compile code into executable form. Tools: Maven, Gradle.

 4. Test–Automated/manual testing to ensure quality. Tools: Selenium, JUnit.

 5. Release–Deliver application to production.

 6. Deploy–Automated deployment using tools. Tools: Ansible, Docker, Kubernetes.

 7. Operate–Run and manage the application environment. Involves provisioning and virtualization.

 8. Monitor–Monitor performance and errors. Tools: Prometheus, Grafana, ELK Stack’

**Waterfall Model**

The Waterfall Model is one of the earliest and most widely used Software Development Life Cycle (SDLC) models. It is a linear and sequential approach, where each phase mustbe completed before the next one begins — just like water flowing down a staircase, hence the name 'Waterfall'.

Phases of the Waterfall Model:

 Requirement Gathering: All system requirements are collected from the client at the beginning.

 System Design: The collected requirements are analyzed, and the system

architecture is designed.

 Implementation (Coding): The Actual source code is written based on the design.

 Testing: After coding, the software is tested to find and fix bugs.

 Deployment: The finished product is delivered to the customer and installed in the live environment.

 Maintenance: Any issues that arise after deployment are handled and fixed in this phase.

Characteristics:

 Easy to understand and manage.

 Each phase has clear deliverables and a review process.

 Works well for small projects with clear, fixed requirements.

Limitations:

 No flexibility to return to previous phases.

 The latest means issues are found later.

 High risk and uncertainty.

 Not suitable for projects with frequent requirement changes.

**Agile Methodology**

Agile is a modern software development methodology that emphasizes flexibility, collaboration, and customer feedback. Unlike the Waterfall Model, Agile divides the development process into smaller, iterative cycles called 'sprints' that typically last 1–4 weeks. Each sprint results in a working version of the software that is reviewed and improved continuously.

**Key Principles of Agile:**

 Customer satisfaction through early and continuous software delivery.

 Welcoming changing requirements even late in development.

 Frequent delivery of working software.

 Close daily collaboration between business people and developers.

 Projects are built around motivated individuals.

 Face-to-face conversation is the best form of communication.

 Working software is the primary measure of progress.

 Sustainable development pace.

 Continuous attention to technical excellence and good design.

 Simplicity—the art of maximizing the amount of work not done—is essential.

**Benefits of Agile:**

 Faster and more flexible development.

 Improved collaboration between teams.

 Higher customer satisfaction.

 Early detection of issues and continuous improvement.

 Better adaptability to changes.

**Why is DevOps Needed?-**

To release software faster and more frequently

To improve collaboration between teams

To increase automation in building, testing, and deploying code

To reduce errors and downtime during releases, to quickly adapt to user and business needs

**Disadvantages of DevOps**:

1. Cultural shift is difficult– Changing mindset and processes can face resistance.

2. Requires skilled professionals– Not all team members may have DevOps experience.

3. High initial setup cost– Tools, automation, and training require investment.

4. Security concerns– Frequent deployments can lead to overlooked vulnerabilities.

5. Not suitable for all projects– Small or simple apps may not need full DevOps setup.

**Advantages of DevOps:**

1. Faster software delivery.

2. Improved collaboration and communication.

3. Better product quality due to continuous testing.

4. Quick bug fixing and recovery.

5. Scalability and reliability through automation.

6. Customer satisfaction– Continuous improvements based on real-time feedback

**Conclusion:**

DevOps fosters collaboration between development and operations for faster, automated, and reliable software delivery. This experiment gave insight into DevOps principles, lifecycle, and its advantages over traditional SDLC models.