Practical No.1

Date of performance:

Date of Submission:

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

What is DevOps?

DevOps is a way of working where the software developers (the people who write the code) and the operations team (the people who manage and run the software on servers) work closely together. Earlier, these two teams worked separately. Developers would build the software, and the operations team would run it. This caused delays and problems.

With DevOps:

- Both teams collaborate (work together).
- They use automation tools to build, test, and deliver software faster.
- This means new features, updates, or fixes can reach users more quickly and with fewer errors.
- It helps companies save time, improve quality, and respond faster to customer needs.

❖ **Key Characteristics of DevOps**

1. Collaboration: Close teamwork between development and operations.
2. Automation: Automate build, test, and deployment processes.
3. Continuous Integration/Continuous Delivery (CI/CD): Frequent code integration and fast
4. delivery.
5. Monitoring: Ongoing tracking of application performance and infrastructure.
6. Infrastructure as Code (IaC): Manage infrastructure through code and automation.
7. Rapid Feedback: Quick identification and resolution of issues.
8. Scalability: Easily scale applications and infrastructure.

❖ **Advantages of DevOps**

1. Faster delivery of software through automation and continuous integration.
2. Better collaboration between development and operations teams.
3. Higher quality software with fewer bugs due to automated testing and monitoring.
4. Increased efficiency by reducing manual work and saving time.
5. Improved customer satisfaction with frequent updates and faster issue resolution.
6. Easy scalability by managing large infrastructure using automation and containers.

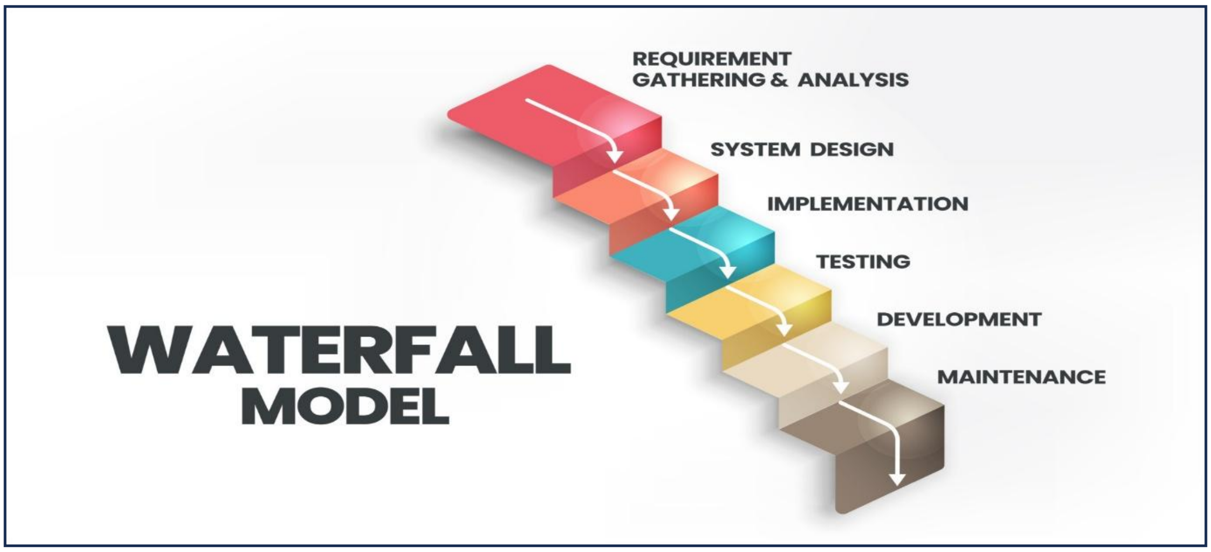
❖ **Disadvantages of DevOps**

1. Requires a cultural shift and change in team working style.
2. Initial setup can be costly and time-consuming.
3. Security challenges due to continuous deployments.
4. More complexity from managing multiple tools and workflows.
5. Needs skilled team members with knowledge of both development and operations.

❖ <u>Traditional Software Development Model (Waterfall Model)</u>

The Waterfall Model is a traditional, linear approach to software development, introduced by Winston W. Royce in 1970. It divides the development process into distinct, sequential phases, where each phase must be completed before moving to the next. This model is widely used for projects with well-defined and stable requirements.
Key Phases of the Waterfall Model

1. **Requirements Analysis and Specification:** This phase involves gathering and documenting all customer requirements in a Software Requirements Specification (SRS) document. It ensures clarity and serves as a contract between stakeholders.
2. **Design:** The design phase translates requirements into a system architecture. It includes: High-Level Design (HLD): Outlines the system's structure and interactions between components. Low-Level Design (LLD): Focuses on detailed specifications for each component.
3. **Development:** Developers write the source code based on the design documents. Unit testing is performed to ensure individual modules function correctly.
4. **Testing and Deployment:** Testing: Integration and system testing are conducted to ensure all modules work together and meet requirements. Deployment: The software is delivered to the customer or end-users, often accompanied by training and setup.
5. **Maintenance: Maintenance:** After the software is released, it is regularly updated to fix problems, adjust to new systems, or improve features.



❖ <u>Advantages of Waterfall Model</u>

1. Each phase has clear goals and is done one after the other.
2. Phases like requirement, design, coding, and testing happen in order.
3. It’s easier to plan and track progress.
4. Works well when requirements are clear and not expected to change.
5. Each stage has proper documentation, which helps in future reference.

❖ <u>Disadvantages of Waterfall Model</u>

1. If customer needs change, it's hard to adapt.

2. Testing happens at the end, so bugs may be found very late.
3. If there's a mistake in the beginning (like in requirements), it affects the whole project.
4. It's hard to handle long-term or evolving projects with this model.

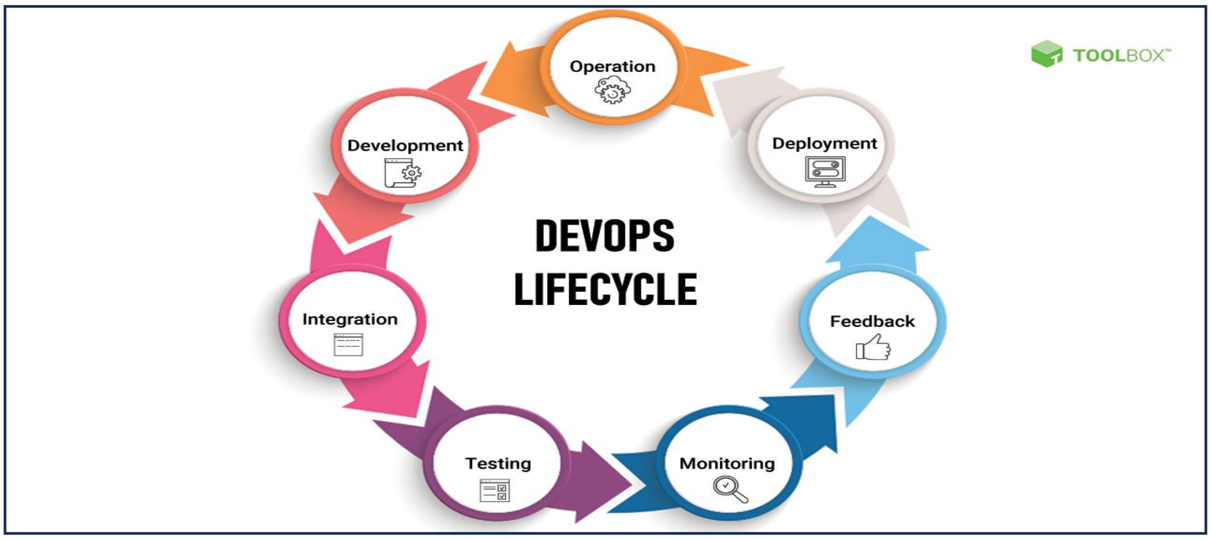
❖ Features of Waterfall Model

1. The process flows in one direction like a waterfall from one phase to the next.
2. The project is divided into separate phases: Requirements, Design, Implementation, Testing, Deployment, and Maintenance.
3. Each phase starts only after the previous one is completed.
4. No overlapping of phases — everything is done step by step.
5. It is easy to manage due to its clear structure and fixed steps.

❖ DevOps Lifecycle

The DevOps lifecycle is a way of working where the development and operations teams work together to deliver software faster and more smoothly. It focuses on teamwork, automation, and continuous feedback.
By using the right tools and methods, DevOps helps:

- Deliver software faster
- Keep systems more stable
- Make teams more efficient



1. **Development**:- Development means creating software step by step in small parts. The team plans, writes code, and prepares features in smaller units instead of doing everything at once.
2. **Integration**:- Integration is a DevOps practice where developers' code is automatically built, tested, and added to the main project.

Main Steps:

- Code is uploaded to GitHub or another code-sharing platform.
- Code is built using tools like Maven.
- Code is checked for bugs using tools like SonarQube.

- Final files are stored in a central place like Nexus.

3. **Testing:-** Testing means checking the software automatically every time the code changes. This helps find and fix bugs early, before the software is released.Tools like Selenium, Testsigma, and LambdaTest are used to test the app automatically instead of doing it manually.With tools like Jenkins, the testing process runs automatically after every code update. This saves time and avoids mistakes.
4. **Deployment:-** is the process of automatically deploying an application into the production environment when it has completed testing and the build stages.
5. **Delivery**:- Delivery means that once the code is built and tested, it is ready to be deployed to the live (production) server — but the deployment is done manually.
6. **Monitoring**:- Monitoring is an important part of the DevOps lifecycle. It means constantly checking the application and systems to make sure everything is working properly.
7. **Feedback**:- Feedback happens after the application is released and used by real users. Users share their experience, suggestions, and problems with the app.The DevOps team reviews this feedback and shares it with the developers, who then improve the code by fixing bugs and making changes.
8. **Operations:-** Operations means keeping the application running all the time with very little or no downtime.

❖ **DevOps Tools**

1. **Puppet:-** Puppet is a tool used to automate the setup and management of servers. It helps keep all servers configured the same way by using code, making things faster and more consistent.
2. **Selenium:-** Selenium is a free tool used to automate testing of websites. It works with many browsers and languages, making it easier to test if the website works correctly.
3. **Jenkins:-** Jenkins is a popular tool that helps automate the software process — like building, testing, and deploying code.It is widely used in Continuous Integration and Continuous Delivery (CI/CD).