Introduction of DevOps

Basic of SDLC Model

The SDLC (Software Development Life Cycle) is a process used to create high-quality software in a systematic, organized way. It breaks down the development of software into different stages to ensure that it meets the client's needs and works efficiently

Stages:

- **Planning:** Understand the project goals, gather requirements, and decide the overall plan for development.
- **Analysis:** Study the user needs and create a detailed plan for how the software will function.
- **Design:** Create the architecture or blueprint for the software, detailing how each component will work together.
- **Development:** Write the actual code based on the design.
- **Testing:** Test the software to find and fix any bugs or issues, ensuring it works as expected.
- **Deployment:** Release the software for users to install and use.
- **Maintenance:** Make updates or improvements, fix bugs, and ensure the software continues to work smoothly over time.

SDLC Models:

Waterfall Model

Structure: Sequential and linear.

Process: Each phase (planning, design, development, testing, deployment) is completed fully before moving to the next one.

Best For: Simple projects with well-defined requirements that are unlikely to change.

Pros:

- Clear structure and easy to manage.
- Suitable for smaller projects with clear goals.

Cons:

- Rigid; hard to go back and make changes once a phase is completed.
- Late discovery of issues since testing happens after development.

Agile Model

Structure: Iterative and incremental.

Process: Development is done in small sprints (usually 2-4 weeks), delivering a working product at the end of each sprint. Continuous feedback and improvements are integrated throughout the process.

Best For: Complex projects with requirements that may change frequently.

Pros:

- Flexible and adaptive to changes.
- Continuous feedback from stakeholders.
- High customer satisfaction.

Cons:

- Can be harder to manage without experience.
- Requires constant communication with the client.

Iterative Model

Structure: Cyclical, with repetitive cycles.

Process: The software is developed and refined in small steps or iterations. Each iteration adds more features until the final product is complete.

Best For: Large projects where the requirements may not be fully clear from the start.

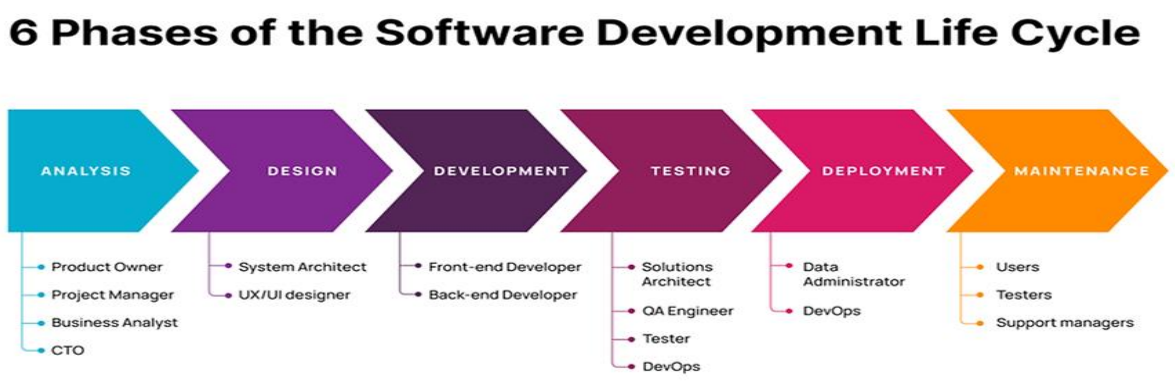
Pros:

- Early detection of issues, as feedback is given in each iteration.
- Adaptable to changes.

Cons:

- May require more resources since multiple iterations are involved.
- Not ideal for small projects.

Phases of SDLC:



What is DevOps?

DevOps is a **set of practices** that combines **software development (Dev)** and **IT operations (Ops)**. The goal of DevOps is to **shorten the development life cycle** and provide continuous delivery of high-quality software. It emphasizes **collaboration**, **automation**, and **continuous feedback** between development and operations teams.

Old Method (Traditional Software Development):

In traditional software development, teams used to work in **silos**, where developers, testers, and system administrators worked separately:

- **Developers** wrote code.
- **Testers** checked the code for bugs.
- **System administrators** deployed the software.

This led to:

- **Long development cycles**.
- **Manual deployments**, prone to errors.
- **Slow feedback loops**, leading to delays in identifying issues.

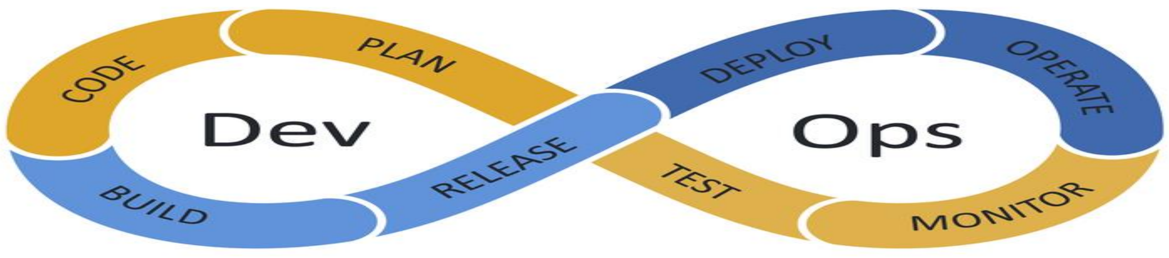
New Method (DevOps Approach):

In DevOps, development, testing, and operations work together **throughout the software development lifecycle**. It focuses on:

- **Automation**: Automating repetitive tasks like code testing and deployment.
- **Collaboration**: Developers, testers, and operations teams collaborate from the beginning.
- **Continuous Integration (CI)**: Developers frequently merge their code into a shared repository, and automated tests are run.
- **Continuous Delivery (CD)**: After testing, code is automatically deployed to production.

DevOps Life Cycle

The **DevOps life cycle** consists of several stages that are interconnected and ensure continuous development, testing, integration, deployment, and monitoring. It promotes collaboration between development and operations teams throughout the software delivery process. Let's break it down step by step:



DevOps Workflow:

The DevOps workflow is a step-by-step process where development (Dev) and operations (Ops) teams collaborate to ensure faster and smoother software delivery. This workflow involves automating key tasks like coding, testing, deployment, and monitoring, making software development more efficient

1. Plan

- **What Happens?** The team plans the software's features, fixes, and updates based on user needs and business goals.
- **Example**: Developers and operations team meet to decide what new features or bug fixes need to be done for the next release.
- **Tools**: Jira, Trello

2. Code

- **What Happens?** Developers write code to create the software or update an existing one.
- **Example**: A developer adds new features like a login system or improves performance.
- **Tools**: Git, GitHub

3. Build

- **What Happens?** Once code is written, it is compiled (put together) into a working application. The code from different developers is combined to ensure it works as one piece.
- **Example**: All the pieces of code from various developers are brought together and packaged into an application.
- **Tools**: Jenkins, GitLab CI

4. Test

- **What Happens?** Automated tests are run on the code to make sure it works as expected and is bug-free.
- **Example**: Automated systems run tests to check if a newly added login feature works correctly.
- **Tools**: Selenium, JUnit

5. Release

- **What Happens?** After testing, the code is deployed (released) to a staging environment, where it is tested in a real-world-like environment.
- **Example**: The login feature is deployed to a test server where it's checked under real-world conditions before going live.
- **Tools**: Jenkins, CircleCI

6. Deploy

- **What Happens?** The tested code is deployed to the live production environment, making it available to actual users.
- **Example**: The new login feature is deployed on the website or app so users can start using it.
- **Tools**: Docker, Kubernetes, Ansible

7. Operate

- **What Happens?** The application is now live and being used by customers. The operations team ensures it runs smoothly.
- **Example:** The operations team monitors the website to make sure it doesn't crash and that the login system is working fine for users.
- **Tools:** AWS, Microsoft Azure, Google Cloud

8. Monitor

- **What Happens?** Continuous monitoring takes place to check the health, performance, and security of the application.
- **Example**: A monitoring system tracks the login feature and alerts the team if there's any issue, like slow performance or downtime.
- **Tools**: Prometheus, Nagios, Grafana

9. Feedback & Improve

- **What Happens?** The team collects feedback from users and monitors logs to improve the application in the next cycle.
- **Example**: If users report login problems or the monitoring tool detects performance issues, the team fixes them and plans improvements for the next release.
- **Tools**: ELK Stack (Elasticsearch, Logstash, Kibana)

Simple DevOps Workflow Visual:

Plan ➡️ 2. Code ➡️ 3. Build ➡️ 4. Test ➡️ 5. Release ➡️ 6. Deploy ➡️ 7. Operate ➡️ 8. Monitor ➡️ Feedback ➡️ Repeat