# SOFTWARE ENGINEERING

## UNIT – 1

## TOPIC – 9

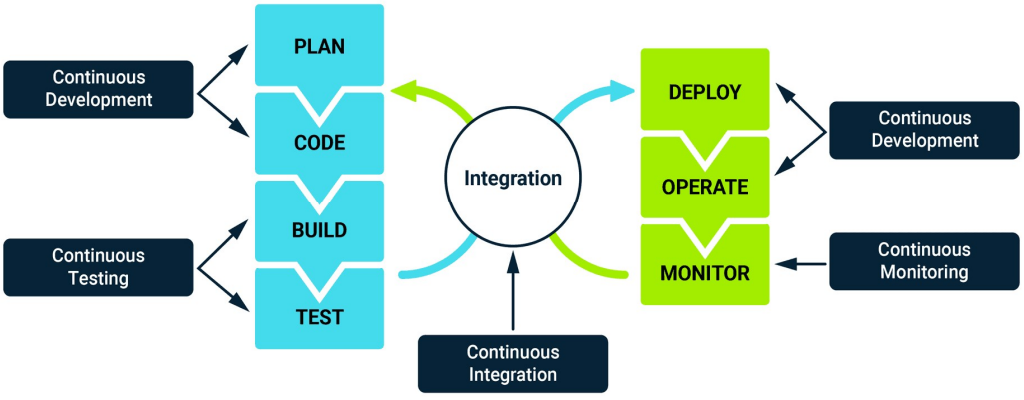
# DEVOPS LIFECYCLE

## I. Introduction to DevOps

DevOps is a combination of "Development" and "Operations". It is a set of practices that aims to automate and integrate the processes of software development (coding, testing) and IT operations (deployment, monitoring) to enhance collaboration and productivity between these teams.


The DevOps lifecycle consists of several key phases that are continuously repeated until the desired quality is achieved.

## II. Phases of the DevOps Lifecycle:




**Planning:** In this phase, teams discuss what needs to be built and how it will meet the user's needs. Communication between team members is crucial here.

Key Point: There aren't specific tools used in planning, but clear communication is the main requirement.

**Coding:** Developers write the code based on the project's requirements. There are various tools to manage this code.

Tools Used: Version control tools: These helps keep track of code changes over time (**e.g., Git, SVN**).

**Building:** The code written by the developers is transformed into software that can be run. This is known as a build.

Tools Used: Build tools: Examples include **Ant, Maven, and Gradle**.

**Integration:** Developers continuously integrate their work, ensuring that the new code works well with what has already been built.

Tools Used: **Jenkins**: It automatically takes the latest code and compiles it into a build.

**Testing:** Automated tests check for issues in the software. If bugs are found, they are reported, fixed, and tested again.

Tools Used: Testing tools like **Selenium, TestNG, and JUnit**.

**Deployment:** The software is moved from a development or testing environment to production (where users can interact with it).

Tools Used: Configuration management tools: These ensure the software runs consistently across different servers (e.g., **Puppet, Chef, Ansible**). Containerization tools: Tools like **Docker** ensure the software behaves the same way across all environments.

**Operation:** Once deployed, the software is operational, and the team monitors it to ensure smooth functioning.

**Monitoring:** Monitoring tools check if the software is performing well and detect any issues. If problems arise, they are fed back to the development team for fixing.

Tools Used: Examples include **Splunk, Nagios, and New Relic**.

## Continuous Loops in DevOps:

**Continuous Development:** The process of planning, coding, building, and testing keeps going in cycles. This ensures that new features are always being added and improved.

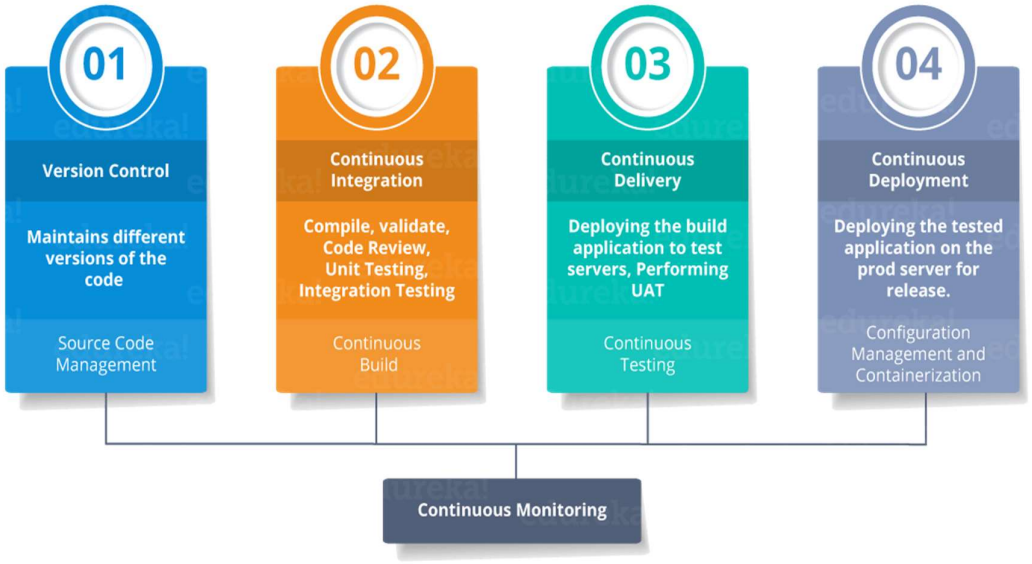
**Continuous Testing:** Regular testing ensures that bugs are caught early, before they reach the users.

**Continuous Integration:** This means merging code changes frequently to ensure everything works together smoothly.

**Continuous Deployment:** The software is continuously deployed to the live environment, so users always have the latest version.

**Continuous Monitoring:** After deployment, the software is constantly monitored to track its performance and identify any issues.

## III. DevOps Stages




**The stages of DevOps — Version Control, Continuous Integration, Continuous Delivery, and Continuous Deployment** — ensure that software development is more collaborative, automated, and efficient. Each stage builds upon the previous one, leading to faster software releases with fewer errors. Continuous monitoring is an essential part of the process, providing real-time feedback that helps improve the software continuously. By integrating these stages, organizations can release high-quality software at a much faster pace, adapting quickly to user needs and market changes.

**Stage 1: Version Control** Version control is the backbone of DevOps practices. It helps teams track and manage changes to the codebase. Every time developers make changes, these are saved in different versions, so teams can easily go back to previous versions if needed. This

system is essential when multiple people work on the same project, as it allows them to collaborate without overwriting each other's work.

- Tools Used: Popular version control tools include **Git, SVN, and Mercurial**. These tools keep the source code organized, allowing teams to review previous changes, identify issues, and roll back to older versions if necessary.

**Stage 2: Continuous Integration (CI)** Continuous Integration (CI) involves frequently merging all the code changes from developers into the main codebase. It ensures that new code works seamlessly with the already existing code. This process involves several steps like compiling the code, running unit tests, and performing integration tests to validate the new changes. This phase ensures that bugs and integration issues are detected early. Continuous Integration automates this process, ensuring the software build is always up-to-date and stable.

- Tools Used: **Jenkins, Travis CI, and CircleCI** are examples of tools that help automate CI by constantly integrating, testing, and building the software.

**Stage 3: Continuous Delivery (CD)** Once the code passes through integration and testing, the next phase is Continuous Delivery. In this stage, the application is automatically delivered to a testing or pre-production environment, where it is further tested before being released to users. This ensures that every change made to the software can be automatically prepared for a release, which means the code is always ready for deployment at any time. The build application is deployed to test servers, and user acceptance testing (UAT) is performed to ensure that the product is ready for production.

- Tools Used: Tools like **Jenkins, GitLab, and Bamboo** help automate delivery processes, allowing smoother transitions between stages.

**Stage 4: Continuous Deployment** Continuous Deployment takes things a step further by automatically releasing the tested software to the production environment. This stage involves releasing the application into real-world environments where users can interact with it. In this phase, configuration management and containerization play a big role in ensuring the software works across various environments without manual intervention. This reduces human error and speeds up the time to market.

- Tools Used: **Docker, Kubernetes, Ansible, and Chef** are tools that handle configuration management and containerization, ensuring the application behaves the same across all environments.

**Continuous Monitoring (Across All Stages)** Continuous monitoring is an ongoing activity that runs across all DevOps stages. This phase ensures that once the software is deployed, its performance is constantly tracked. Monitoring tools help identify any potential issues such as server crashes, slow performance, or security vulnerabilities. If problems are detected, feedback is sent back to the development team, and the process starts again.

- Tools Used: Popular tools for continuous monitoring include **Nagios, Splunk, New Relic, and Prometheus.**

## IV. DevOps Phases


DevOps is an ongoing process that integrates development and operations to ensure continuous delivery of high-quality software. By focusing on collaboration, automation, and feedback, the DevOps approach allows teams to build, test, and release applications faster and more reliably.

The key phases in the DevOps process:

**1. Plan**

Planning is the first and most critical step. It involves deciding what kind of application needs to be built, identifying its goals, and outlining the steps required to achieve those goals. You gather all the requirements from stakeholders and create a detailed plan or roadmap.

In this phase, you figure out what you're going to build, why it's needed, and how to go about building it.

**2. Code**

This is the phase where the actual development happens. Developers write the code for the application based on the plan. The code must align with user needs and business objectives, and collaboration among team members is essential.

This is where you write the actual instructions (code) that make the application work.

**3. Build**

Once the code is written, it needs to be compiled and combined to create the full application. During this phase, different modules or parts of the code are brought together to form a complete system. Automation tools are often used to speed up this process and avoid errors.

You bring together all the code pieces to create a functioning version of the application.

**4. Test**

Testing ensures that the application works as expected. Various tests (like unit tests, integration tests, and performance tests) are run to find bugs or issues. If something doesn't work, it's fixed and tested again until it's right.

You check to make sure the application works properly by looking for any errors or bugs.

**5. Integrate**

During integration, code from different developers is combined into a single, cohesive system. This phase helps ensure that everyone's work fits together and functions smoothly without conflicts. Continuous Integration (CI) tools help to automate this process, reducing manual effort.

You combine the work of different developers into one unified application.

**6. Deploy**

Deployment is about making the application available to users. The software is placed on servers (usually in the cloud) so that it can be accessed and used by people. This phase often involves automating the release process to make deployment faster and more reliable.

You make the application live, so people can start using it.

**7. Operate**

After the application is deployed, the operations team ensures it runs smoothly. This includes fixing any issues that come up, managing server resources, and handling routine maintenance tasks. The goal is to keep the application functioning well for users.

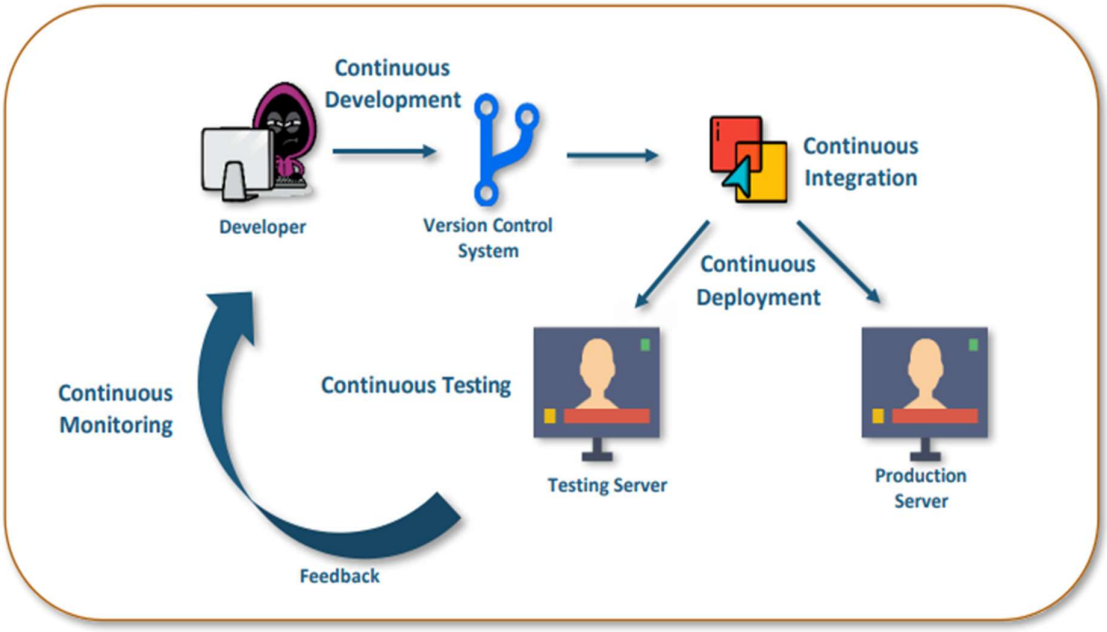
You take care of the application after it's live by fixing problems and keeping everything running smoothly.

**8. Monitor**

Continuous monitoring is key to ensuring the application performs well over time. Tools are used to track performance, uptime, and other key metrics. If issues arise, changes or fixes are made quickly to ensure users remain happy with the service.

You keep an eye on how well the application is working and make improvements when necessary to keep users satisfied.

## Continuous DevOps Process




The DevOps process is a continuous flow, where software is constantly developed, tested, and improved in a cycle:

It all begins with the Developer writing new code or updating existing code. This code is then stored in a Version Control System, which helps keep track of all changes and versions of the software.

Once the code is ready, it moves to Continuous Integration, where it's automatically combined with the code from other developers. This step ensures that the new code works well with the rest of the project.

After integration, the process moves into Continuous Deployment. Here, the software is automatically sent to different servers. First, it goes to a Testing Server, where the code undergoes Continuous Testing to check if everything works as expected.

If the tests are successful, the code is then deployed to the Production Server, where users can access the new or updated software.

While the software is running on the Production Server, Continuous Monitoring takes place. This helps track performance and find any issues in real-time. Based on the results of this monitoring, feedback is collected and sent back to the development team.

This feedback initiates another cycle of development, ensuring the software is constantly improved and updated without any breaks in the process.

Thus, the process forms a loop that ensures continuous development, testing, deployment, and monitoring, making the software more efficient and reliable over time.