Name: Param Shukla

Roll No: C107

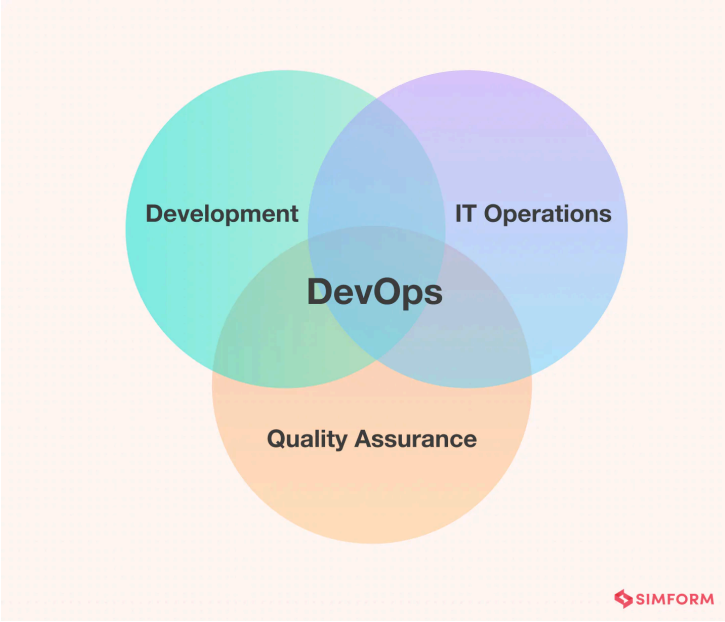
Sap ID: 60004220046

DIV: C-2

Experiment No: 1 - Case Study

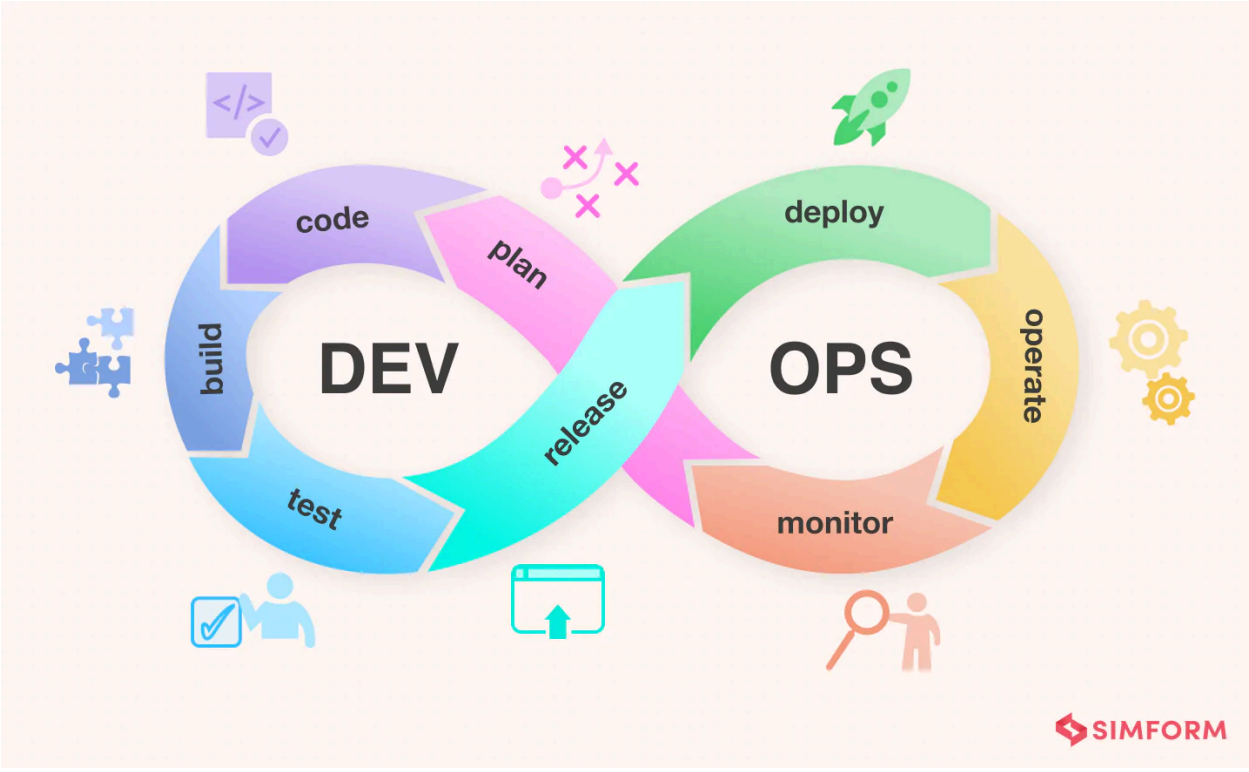
What is DevOps?

DevOps is a work culture primarily centered around collaboration, communication, and integration among the development teams. It was introduced to address the disconnect primarily between the development, operations, and quality assurance teams. As a result, it's becoming crucial for businesses to adopt DevOps practices, not only for seamless software development and operations but also for the high quality of deployment for successful product delivery.



What is DevOps lifecycle?

DevOps lifecycle is a series of automated development processes or workflows within an iterative development lifecycle. It follows a continuous approach; hence its lifecycle is symbolized in the form of an infinity loop. This loop depicts the collaborative and iterative approach throughout the application lifecycle, consisting of tools and technology stacks for each stage. The left part deals with software development and testing. And in contrast, the right side of the infinity loop represents the deployment and operations cycle.



How Does the DevOps lifecycle work?

1. Plan: In this stage, teams identify the business requirement and collect end-user feedback. They create a project roadmap to maximize the business value and deliver the desired product during this stage.

2. Code: The code development takes place at this stage. The development teams use some tools and plugins like Git to streamline the development process, which helps them avoid security flaws and lousy coding practices.

3. Build: In this stage, once developers finish their task, they commit the code to the shared code repository using build tools like Maven and Gradle.

4. Test: Once the build is ready, it is deployed to the test environment first to perform several types of testing like user acceptance test, security test, integration testing, performance testing, etc., using tools like JUnit, Selenium, etc., to ensure software quality.

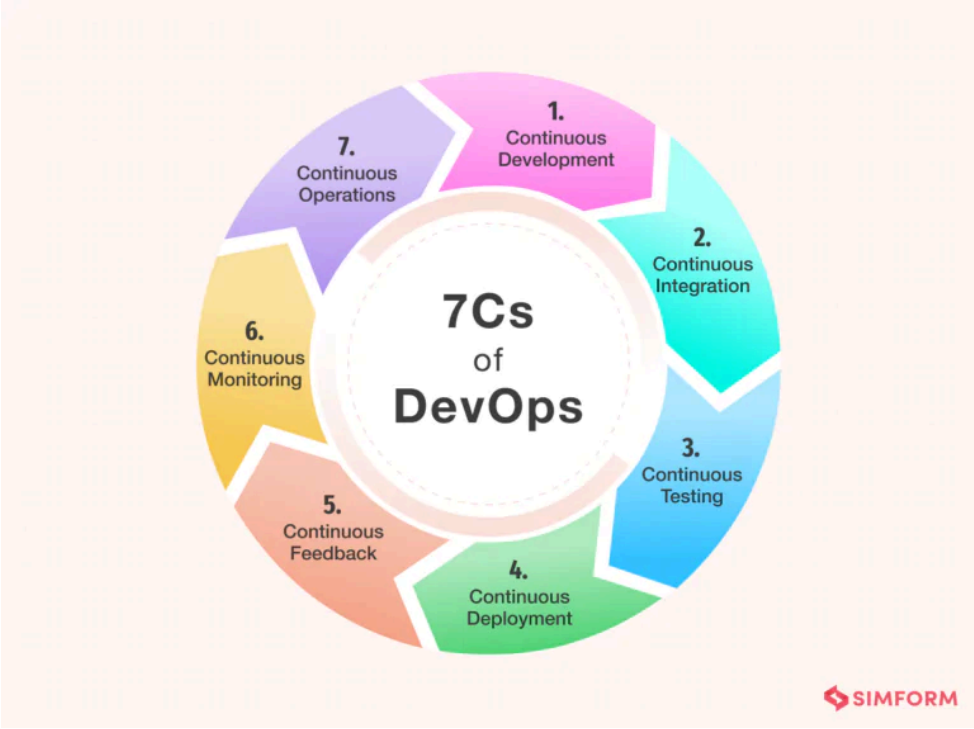
5. Release: The build is ready to deploy on the production environment at this phase. Once the build passes all tests, the operations team schedules the releases or deploys multiple releases to production, depending on the organizational needs.

6. Deploy: In this stage, Infrastructure-as-Code helps build the production environment and then releases the build with the help of different tools.

7. Operate: The release is live now to use by customers. The operations team at this stage takes care of server configuring and provisioning using tools like Chef.

8. Monitor: In this stage, the DevOps pipeline is monitored based on data collected from customer behavior, application performance, etc. Monitoring the entire environment helps teams find the bottlenecks impacting the development and operations teams' productivity.

DevOps lifecycle phases:
The 7Cs of DevOps lifecycle



1. Continuous Development

This phase is crucial in defining the vision for software development, focusing on project planning and coding. Requirements are gathered and discussed with stakeholders, and a product backlog is maintained based on customer feedback. The development team continuously codes to accommodate changes or performance issues.

Nordstrom adopted DevOps to improve development speed after facing issues with the waterfall model. By shifting to continuous planning and development with a single backlog, they enhanced build quality, reduced bugs, and increased product releases from twice a year to monthly.

Common tools for version control include GitLab, GIT, TFS, SVN, and BitBucket, while Jira, Scrum, Lean, and Kanban support collaboration. GIT and Jira are widely used for managing complex projects effectively.

2. Continuous Integration

Continuous integration is a crucial DevOps phase where updated code, features, and bug fixes are continuously integrated into existing code. Unit testing is performed at every step to detect and resolve issues, ensuring seamless integration with each commit. Test planning also takes place in this phase.

DocuSign, known for its e-signature technology, initially followed Agile but faced collaboration challenges between development and operations. To enhance integration and delivery, they adopted DevOps and implemented a mock tool for internal APIs, improving product development, incident management, and testing.

Common CI tools include Jenkins, Bamboo, GitLab CI, Buddy, TeamCity, Travis, and CircleCI. Jenkins is a widely used open-source tool, while CircleCI and Buddy are commercial options. The choice of tools should align with business and project needs.

3. Continuous Testing

Some teams carry out the continuous testing phase before the integration occurs, while others do it after the integration. Quality analysts continuously test the software for bugs and issues during this stage using Docker containers. In case of a bug or an error, the code is sent back to the integration phase for modification. Automation testing also reduces the time and effort to deliver quality results. Teams use tools like Selenium at this stage. Moreover, continuous testing enhances the test evaluation report and minimizes the provisioning and maintenance cost of the test environments.

Tools Used: JUnit, Selenium, TestNG, and TestSigma are a few DevOps tools for continuous testing. Selenium is the most popular open-source automation testing tool that supports multiple platforms and browsers. TestSigma, on the other hand, is a unified AI-driven test automation platform that eliminates the technical complexity of test automation through artificial intelligence.

4. Continuous Deployment

Continuous deployment is a crucial DevOps phase where the final code is deployed to production servers. Configuration management ensures smooth and accurate deployment, while containerization tools maintain consistency across development, testing, staging, and production environments, enabling continuous delivery.

Adobe adopted DevOps to release small software updates continuously. Using CloudMunch's DevOps platform, Adobe improved product management by understanding the impact of changes across its products, ensuring faster and more efficient software delivery.

Common deployment tools include Ansible, Puppet, and Chef for configuration management, while Docker and Vagrant handle scalability. Spinnaker and ArgoCD are open-source tools used for continuous delivery, especially for Kubernetes environments.

5. Continuous Feedback

Continuous feedback is essential for analyzing and improving application code by evaluating customer behavior after each release. Businesses gather feedback through structured methods like surveys or unstructured sources like social media. This phase ensures continuous delivery by refining future updates.

Tangerine Bank, a Canadian bank, leveraged continuous feedback to enhance its mobile experience. By collecting valuable insights quickly, they identified issues, improved their application, and optimized resources effectively.

Common tools include Pendo for product analytics and Qentelli's TED for tracking DevOps processes and identifying bugs.

6. Continuous Monitoring

During this phase, the application's functionality and features are monitored continuously to detect system errors such as low memory, non-reachable server, etc. This process helps the IT team quickly identify issues related to app performance and the root cause behind it. If IT teams find any critical issue, the application goes through the

entire DevOps cycle again to find the solution. However, the security issues can be detected and resolved automatically during this phase.

Tools Used: Nagios, Kibana, Splunk, PagerDuty, ELK Stack, New Relic, and Sensu are a few DevOps tools used to make the continuous monitoring process fast and straightforward.

7. Continuous Operations

The last phase in the DevOps lifecycle is crucial for reducing the planned downtime, such as scheduled maintenance. Generally, developers are required to take the server offline to make the updates, which increases the downtime and might even cost a significant loss to the company. Eventually, continuous operation automates the process of launching the app and its updates. It uses container management systems like Kubernetes and Docker to eliminate downtime. These container management tools help simplify the process of building, testing, and deploying the application on multiple environments. The key objective of this phase is to boost the application’s uptime to ensure uninterrupted services. Through continuous operations, developers save time that can be used to accelerate the application’s time-to-market.

Tools Used: Kubernetes and Docker Swarm are the container orchestration tools used for the high availability of the application and to make the deployment faster.

What is Agile?

Agile is a term that describes approaches to software development that emphasize incremental delivery, team collaboration, continual planning, and continual learning. The term Agile was coined in 2001 in the Agile Manifesto. The manifesto set out to establish principles to guide a better approach to software development. At its core, the manifesto declares four value statements that represent the foundation of the Agile movement. As written, the manifesto states:

We have come to value:

1. Individuals and interactions over processes and tools.
2. Working software over comprehensive documentation.
3. Customer collaboration over contract negotiation.
4. Responding to change over following a plan.

The manifesto doesn't imply that the items on the right side of these statements aren't important or needed. Rather, items on the left are simply more valued.

Agile methods and practices

It's important to understand that Agile isn't a thing. You don't do Agile. Rather, Agile is a mindset that drives an approach to software development. Because there's no single approach that works for all situations, the term Agile has come to represent various methods and practices that align with the value statements in the manifesto.

Agile methods, which are often called frameworks, are comprehensive approaches to phases of the DevOps lifecycle: planning, development, delivery, and operations. They prescribe a method for accomplishing work, with clear guidance and principles.

Scrum is the most common Agile framework, and the one that most people start with. Agile practices, on the other hand, are techniques that are applied during phases of the software development lifecycle.

1. Planning Poker is a collaborative estimation practice that's designed to encourage team members to share their understanding of what done means. Many people find the process fun, and it has proven to help foster teamwork and better estimates.

2. Continuous integration (CI) is a common Agile engineering practice that involves integrating code changes into the main branch frequently. An automated build verifies changes. As a

result, there's a reduction in integration debt and a continually shippable main branch.

These practices, like all Agile practices, carry the Agile label, because they're consistent with the principles in the Agile manifesto.

Why Agile?

So why would anyone consider an Agile approach? It's clear that the rules of engagement around building software have fundamentally changed in the last 10-15 years. Many of the activities look similar, but the landscape and environments where we apply them are noticeably different.

1. Compare what it's like to purchase software today with the early 2000s. How often do people drive to the store to buy business software?

2. Consider how feedback is collected from customers about products. How did a team understand what people thought about their software before social media?

3. Consider how often a team desires to update and improve the software that they deliver. Annual updates are no longer feasible against modern competition.