**EXPERIMENT 1**

**AIM:** A case study on DevOps.

**THEORY:**

**What is DevOps?**

DevOps is a set of practices that combines [software development](https://en.wikipedia.org/wiki/Software_development) (Dev) and [IT operations](https://en.wikipedia.org/wiki/IT_operations) (Ops). It aims to shorten the [systems development life cycle](https://en.wikipedia.org/wiki/Systems_development_life_cycle) and provide [continuous delivery](https://en.wikipedia.org/wiki/Continuous_delivery) with high [software quality](https://en.wikipedia.org/wiki/Software_quality). DevOps is complementary with [Agile software development](https://en.wikipedia.org/wiki/Agile_software_development); several DevOps aspects came from the Agile methodology. DevOps is a new term emerging from the collision of two major related trends. The first was also called “agile infrastructure” or “agile operations”; it sprang from applying Agile and Lean approaches to operations work.  The second is a much-expanded understanding of the value of collaboration between development and operations staff throughout all stages of the development lifecycle when creating and operating a service, and how important operations has become in our increasingly service-oriented world.

**History of DevOps**

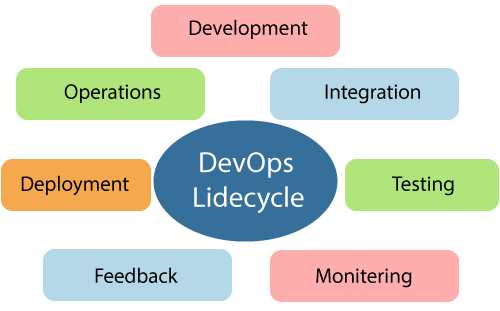
The history of DevOps is simple, yet revolutionary. The concept of DevOps emerged out of a discussion between Andrew Clay and Patrick Debois in 2008. They were concerned about the drawbacks of Agile and wanted to come up with something better. The idea slowly began to spread and after the DevOpsDays event held in Belgium in 2009, it became quite a buzzword. What is best about DevOps is that there is more to it than it seems. It’s not just an attempt at efficiency, it is a step towards cultural change. We can call it a blend of agile philosophy with lean thinking. DevOps unites the efforts of every team involved in the project and that too with tighter integration. This cross-department integration between developers, QA engineers, and system admins is more amazing than it sounds. DevOps implementation ensures that developers can now take part in deployment, admins can write scripts and QA engineers know how to figure out issues other than testing. Processes can be automated and no one has to wait as they can now work more closely and come up with quicker and better solutions. Better communication and understanding would also help teams to recognize the priorities of each other. And all of these benefits would mean skyrocketing productivity and swift delivery speeds.

**Evolution of DevOps**

DevOps practices have evolved in recent years in many organizations seeking to respond more effectively to their business challenges. While DevOps previously focused primarily on IT services, it now extends across the entire enterprise, impacting processes and data flows and driving deep organizational changes. Indeed, DevOps is not just about choosing the right automation solution. It requires an accompanied transition, wherein lies the role and responsibility of the solution vendor. In a DevOps project, levels of maturity and understanding differ between organizations. A DevOps solution provider therefore has a duty to advise and support in the management of change and should add value to the project beyond a simple automation. Company specifics must be taken into account, and in particular, the scope and diversity of development cultures and technology platforms contained in the application portfolio. Without this, a DevOps project has no chance of success.

**DevOps Lifecycle**

DevOps defines an agile relationship between operations and Development. It is a process that is practiced by the development team and operational engineers together from beginning to the final stage of the product.



Learning DevOps is not complete without understanding the DevOps lifecycle phases. The DevOps lifecycle includes seven phases as given below:

1) Continuous Development

This phase involves the planning and coding of the software. The vision of the project is decided during the planning phase. And the developers begin developing the code for the application. There are no DevOps tools that are required for planning, but there are several tools for maintaining the code.

2) Continuous Integration

This stage is the heart of the entire DevOps lifecycle. It is a software development practice in which the developers require to commit changes to the source code more frequently. This may be on a daily or weekly basis. Then every commit is built, and this allows early detection of problems if they are present. Building code is not only involved compilation, but it also includes unit testing, integration testing, code review, and packaging. Jenkins is a popular tool used in this phase. Whenever there is a change in the Git repository, then Jenkins fetches the updated code and prepares a build of that code, which is an executable file in the form of war or jar. Then this build is forwarded to the test server or the production server.

3) Continuous Testing

This phase, where the developed software is continuously testing for bugs. For constant testing, automation testing tools such as TestNG, JUnit, Selenium, etc are used. These tools allow QAs to test multiple code-bases thoroughly in parallel to ensure that there is no flaw in the functionality. In this phase, Docker Containers can be used for simulating the test environment. Selenium does the automation testing, and TestNG generates the reports. This entire testing phase can automate with the help of a Continuous Integration tool called Jenkins.

4) Continuous Monitoring

Monitoring is a phase that involves all the operational factors of the entire DevOps process, where important information about the use of the software is recorded and carefully processed to find out trends and identify problem areas. Usually, the monitoring is integrated within the operational capabilities of the software application.

5) Continuous Feedback

The application development is consistently improved by analysing the results from the operations of the software. This is carried out by placing the critical phase of constant feedback between the operations and the development of the next version of the current software application.

6) Continuous Deployment

In this phase, the code is deployed to the production servers. Also, it is essential to ensure that the code is correctly used on all the servers. The new code is deployed continuously, and configuration management tools play an essential role in executing tasks frequently and quickly. Here are some popular tools which are used in this phase, such as Chef, Puppet, Ansible, and SaltStack.

7) Continuous Operations

All DevOps operations are based on the continuity with complete automation of the release process and allow the organization to accelerate the overall time to market continuingly. With DevOps, we can make any software product more efficient and increase the overall count of interested customers in your product.

**DevOps Tools:**



* Puppet is the most widely used DevOps tool. It allows the delivery and release of the technology changes quickly and frequently. It has features of versioning, automated testing, and continuous delivery. It enables to manage entire infrastructure as code without expanding the size of the team.
* Nagios is one of the more useful tools for DevOps. It can determine the errors and rectify them with the help of network, infrastructure, server, and log monitoring systems.
* A chef is a useful tool for achieving scale, speed, and consistency. The chef is a cloud-based system and open-source technology. This technology uses Ruby encoding to develop essential building blocks such as recipes and cookbooks. The chef is used in infrastructure automation and helps in reducing manual and repetitive tasks for infrastructure management.
* Jenkins is a DevOps tool for monitoring the execution of repeated tasks. Jenkins is a software that allows continuous integration. Jenkins will be installed on a server where the central build will take place. It helps to integrate project changes more efficiently by finding the issues quickly.
* Git is an open-source distributed version control system that is freely available for everyone. It is designed to handle minor to major projects with speed and efficiency. It is developed to co-ordinate the work among programmers. The version control allows you to track and work together with your team members at the same workspace. It is used as a critical distributed version-control for the DevOps tool.
* SaltStack is a lightweight DevOps tool. It shows real-time error queries, logs, and more directly into the workstation.

**CONCLUSION:**

Therefore, we have successfully performed a case study on DevOps.