**Experiment 1**

**Aim:** To understand the Devops principles,practices and Devops engineer role and responsibilities.

**Introduction to DevOps:**

For a long time, development and operations were isolated modules. Developers wrote code; the system administrators were responsible for its deployment and integration. As there was limited communication between these two silos, specialists worked mostly separately within a project. That was fine when Waterfall development dominated. But since Agile and continuous workflow have taken over the world of software development, this model is out of the game. Short sprints and frequent releases occurring every two weeks or even every day require a new approach and new team roles.

Today, DevOps is one of the most discussed software development approaches. It is applied in Facebook, Netflix, Amazon, Etsy, and many other industry-leading companies. So, if you are considering embracing DevOps for the sake of better performance, business success, and competitiveness, you take the first step and hire a DevOps engineer. But first, let’s look at what DevOps is all about and how it helps improve product delivery.

**What Is DevOps?**

DevOps stands for development and operations. It’s a practice that aims at merging development, quality assurance, and operations (deployment and integration) into a single, continuous set of processes. This methodology is a natural extension for Agile and continuous delivery approaches.



**What DevOps looks like**.

But DevOps isn’t merely a set of actions. It’s more of a culture or even a philosophy that fosters cross-functional team communication. One of the main benefits of DevOps is that it doesn’t require substantial technical changes being rather oriented to changing the way a team works. Teamwork is a crucial part of DevOps culture: The whole success of a process depends on it, and there are principles and practices that DevOps teams use.

**DevOps Principles**

The main principles of DevOps are automation, continuous delivery, and fast reaction to feedback. You can find a more detailed explanation of DevOps pillars in the CAMS acronym:

Culture represented by human communication, technical processes, and tools

Automation of processes

Measurement of KPIs

Sharing feedback, best practices, and knowledge

Adherence to these principles is achieved through a number of DevOps practices that include continuous delivery, frequent deployments, QA automation, validating ideas as early as possible, and in-team collaboration.

**DevOps Model and Practices**

DevOps requires a delivery cycle that comprises planning, development, testing, deployment, release, and monitoring with active cooperation between different members of a team.



**DevOps lifecycle.**

**The core practices that constitute the DevOps:**

**Agile Planning**

In contrast to traditional approaches of project management, Agile planning organizes work in short iterations (e.g. sprints) to increase the number of releases. This means that the team has only high-level objectives outlined while making detailed planning for two iterations in advance. This allows for flexibility and pivots once the ideas are tested on an early product increment. Check our Agile infographics to learn more about the different methods applied.

**Continuous Delivery and Automation**

Continuous delivery, detailed in our dedicated article, is an approach that merges development, testing, and deployment operations into a streamlined process as it heavily relies on automation.

Development. Engineers commit code in small chunks multiple times a day for it to be easily tested.

Continuous automated testing and integration. A quality assurance team sets committed code testing using automation tools like Selenium, Ranorex, UFT, etc. If bugs and vulnerabilities are revealed, they are sent back to the engineering team. This stage also entails version control to detect integration problems in advance. A Version Control System (VCS) allows developers to record changes in the files and share them with other members of the team, regardless of its location. The code that passes automated tests is integrated in a single, shared repository on a server. Frequent code submissions prevent a so-called “integration hell” when the differences between individual code branches and the mainline code become so drastic over time that integration takes more than actual coding. The most popular tools for continuous integration are Jenkins, GitLab CI, Bamboo, and TeamCity.

**Continuous deployment**

.At this stage, the code is deployed to run in production on a public server. Code must be deployed in a way that doesn’t affect already functioning features and can be available for a large number of users. Frequent deployment allows for a “fail fast” approach, meaning that the new features are tested and verified early. There are various automated tools that help engineers deploy a product increment. The most popular are Chef, Puppet, Azure Resource Manager, and Google Cloud Deployment Manager.

**Continuous monitoring.**

The final stage of the DevOps lifecycle is oriented to the assessment of the whole cycle. The goal of monitoring is detecting the problematic areas of a process and analyzing the feedback from the team and users to report existing inaccuracies and improve the product’s functioning.

**Infrastructure as Code**

Infrastructure as a code (IaC) is an infrastructure management approach that makes continuous delivery and DevOps possible. It entails using scripts to automatically set the deployment environment (networks, virtual machines, etc.) to the needed configuration regardless of its initial state. Without IaC, engineers would have to treat each target environment individually, which becomes a tedious task as you may have many different environments for development, testing, and production use. Having the environment configured as code, you 1) can test it the way you test the source code itself and 2) use a virtual machine that behaves like a production environment to test early. Once the need to scale arises, the script can automatically set the needed number of environments to be consistent with each other.

**A DevOps Engineer: Role and Responsibilities**

In the book Effective DevOps by Ryn Daniels and Jennifer Davis, the existence of a specific DevOps person is questioned:

“It doesn’t usually make much sense to have a director of DevOps or some other position that puts one person in charge of DevOps. DevOps is at its core a cultural movement, and its ideas and principles need to be used throughout entire organizations in order to be effective.”

Some other DevOps experts partly disagree with this statement. They also believe that a team is key to effectiveness. But in this interpretation, a team – including developers, a quality assurance leader, a code release manager, and an automation architect – work under the supervision of a DevOps engineer.

So, the title of a DevOps Engineer is an arguable one. Nonetheless, DevOps engineers are still in demand on the IT labor market. Some consider this person to be either a system administrator who knows how to code or a developer with a system administrator’s skills.

**DevOps Engineer Responsibilities**

In a way, both definitions are fair. The main function of a DevOps engineer is to introduce the continuous delivery and continuous integration workflow, which requires the understanding of the mentioned tools and the knowledge of several programming languages. Depending on the organization, job descriptions differ. Smaller businesses look for engineers with broader skillsets and responsibilities. For example, the job description may require product building along with the developers. Larger companies may look for an engineer for a specific stage of the DevOps lifecycle that will work with a certain automation tool.



**DevOps Engineer Role and Requirements.**

The basic and widely-accepted responsibilities of a DevOps engineer are:

Writing specifications and documentation for the server-side features

Management of continuous deployment and continuous integration (CI/CD)

CI/CD script writing

Performance assessment and monitoring

**DevOps Engineer Skillset**

While this title doesn’t require a candidate to be a system administrator or a developer, this person must have experience in both fields. When hiring a DevOps engineer, pay attention to the following characteristics:

Tech background. A DevOps engineer must hold a degree in computer science, engineering, or other related fields. Work experience must be greater than 2 years. This includes work as a developer, system administrator, or one of the members of a DevOps-driven team. This is an important requirement along with an understanding of all IT operations.

Automation tool experience. The knowledge of open source solutions for testing and deployment is a must for a DevOps engineer. If you use a cloud server, make sure that your candidate has experience with such tools as GitHub, Chef, Puppet, Jenkins, Ansible, Nagios, and Docker. A candidate for this job also must have experience with public clouds such as Amazon AWS, Microsoft Azure, and Google Cloud.

Programming skills. An engineer not only has to know off-the-shelf tools, but also must have programming experience to cover scripting and coding. Scripting skills usually entail the knowledge of Bash or PowerShell scripts, while coding skills may include Java, C#, C++, Python, PHP, Ruby, etc., or at least some of these languages.

Knowledge of database systems. At the deployment stage, an engineer works with data processing, which requires experience with both SQL or NoSQL database models.

Communication and interpersonal skills. Although a good candidate must be well-versed in tech aspects, a DevOps expert must have strong communication talents. He/she must ensure that a team functions effectively, receives and shares feedback to support continuous delivery. The outcome – a product – depends on his/her ability to effectively communicate with all team members.

**Conclusion:**

By embracing DevOps principles and practices and leveraging the expertise of DevOps engineers, companies successfully transforms its development processes.The collaborative approach, automation,and contionous improvement mindset paves the way for faster, higher-quality software delivery.