DevOps in Cloud Computing

SEMINAR REPORT

SUBMITTED IN THE PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE AWARD OF THE DEGREE OF

BACHELOR OF ENGINEERING

IN

ELECTRONICS AND COMMUNICATIONS ENGINEERING

SUBMITTED BY:

KRISHNA KAPOOR

ROLL NO. UE215051

EIGHTH SEMESTER

DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING

UNIVERSITY INSTITUTE OF ENGINEERING AND TECHNOLOGY,

PANJAB UNIVERSITY, CHANDIGARH, INDIA

MAY 2025

CANDIDATE'S DECLARATION

I hereby certify that the work presented in the Seminar entitled "DevOps in Cloud Computing" in partial fulfillment of the requirement for the award of the degree of Bachelor of engineering in electronics and communications engineering from the University Institute of Engineering and technology, Panjab University, Chandigarh, is an authentic record of my work carried out under the supervision and guidance of Dr. Daljeet Kaur.

Date: 30-04-2025 (Krishna Kapoor)

Chandigarh (UE215051)

CERTIFICATE

This is to certify that the above statement made by the candidate is correct to the best of my knowledge and belief.

Date: 30-04-2025 Dr. Daljeet Kaur

Chandigarh ECE, UIET, Panjab University

ABSTRACT

In the rapidly evolving landscape of software development and IT operations, DevOps has emerged as a transformative methodology that bridges the gap between development and operations teams. When integrated with cloud computing, DevOps gains enhanced flexibility, scalability, and automation capabilities, enabling organizations to deliver software faster, more reliably, and with greater efficiency. This synergy not only accelerates the software development lifecycle but also ensures continuous integration, continuous delivery (CI/CD), and real-time monitoring in a highly dynamic environment. DevOps in cloud computing introduces several key features such as automated infrastructure provisioning, containerization, microservices architecture, version control integration, and real-time collaboration. These features collectively support rapid deployment, consistent environments, and reduced human error, all while maintaining high availability and performance. The need for DevOps in cloud computing stems from the growing demand for faster release cycles, improved system reliability, cost efficiency, and seamless scalability. Traditional software development approaches often fail to meet these demands, leading to silos, delays, and operational inefficiencies. DevOps, powered by cloud platforms, overcomes these limitations by fostering a culture of collaboration, automation, and continuous improvement. Applications of DevOps in cloud environments span across industries, including e-commerce, finance, healthcare, and telecommunications. It supports automated testing, infrastructure as code (IaC), continuous security integration (DevSecOps), and real-time performance monitoring. Leading cloud service providers like AWS, Azure, and Google Cloud offer robust DevOps toolchains and services that streamline these operations. In conclusion, the integration of DevOps with cloud computing represents a paradigm shift in software engineering, enabling organizations to innovate rapidly, respond to market changes swiftly, and maintain high-quality standards. As digital transformation accelerates, the adoption of DevOps in the cloud will continue to be a critical enabler of operational excellence and business agility.

ACKNOWLEDGEMENT

I would like to express my deep gratitude to Faculty-in-charge Dr. Daljeet Kaur for their patient guidance, enthusiastic encouragement, and useful critiques during the completion of my seminar named “DevOps in Cloud Computing”.

I would also like to extend my thanks to the University Institute of Engineering & Technology (UIET), Panjab University for giving me such an opportunity to develop an understanding of cloud computing.

Further, I want to thank my parents for supporting me during my seminar and helping me procure the required resources for the fulfilment of my report.

Table of Contents

CHAPTER 1

CLOUD COMPUTING

The rise of cloud computing has significantly reshaped the way businesses deploy and manage applications. One of the most revolutionary technologies that has emerged in the cloud-native ecosystem is Kubernetes, an open-source container orchestration platform. Kubernetes has become a fundamental tool for automating the deployment, scaling, and management of containerized applications. This report explores the interplay between Kubernetes and cloud computing, examining their respective roles, benefits, and how they work together to drive modern application infrastructure.

Understanding Cloud Computing

Cloud computing refers to the delivery of computing services—such as storage, processing power, databases, networking, software, and analytics—over the internet. Rather than owning and maintaining physical data centers, businesses can rent these services on-demand from cloud providers like Amazon Web Services (AWS), Microsoft Azure, and Google Cloud Platform (GCP).

Key characteristics of cloud computing include:

1. Scalability: The ability to scale resources up or down based on demand.

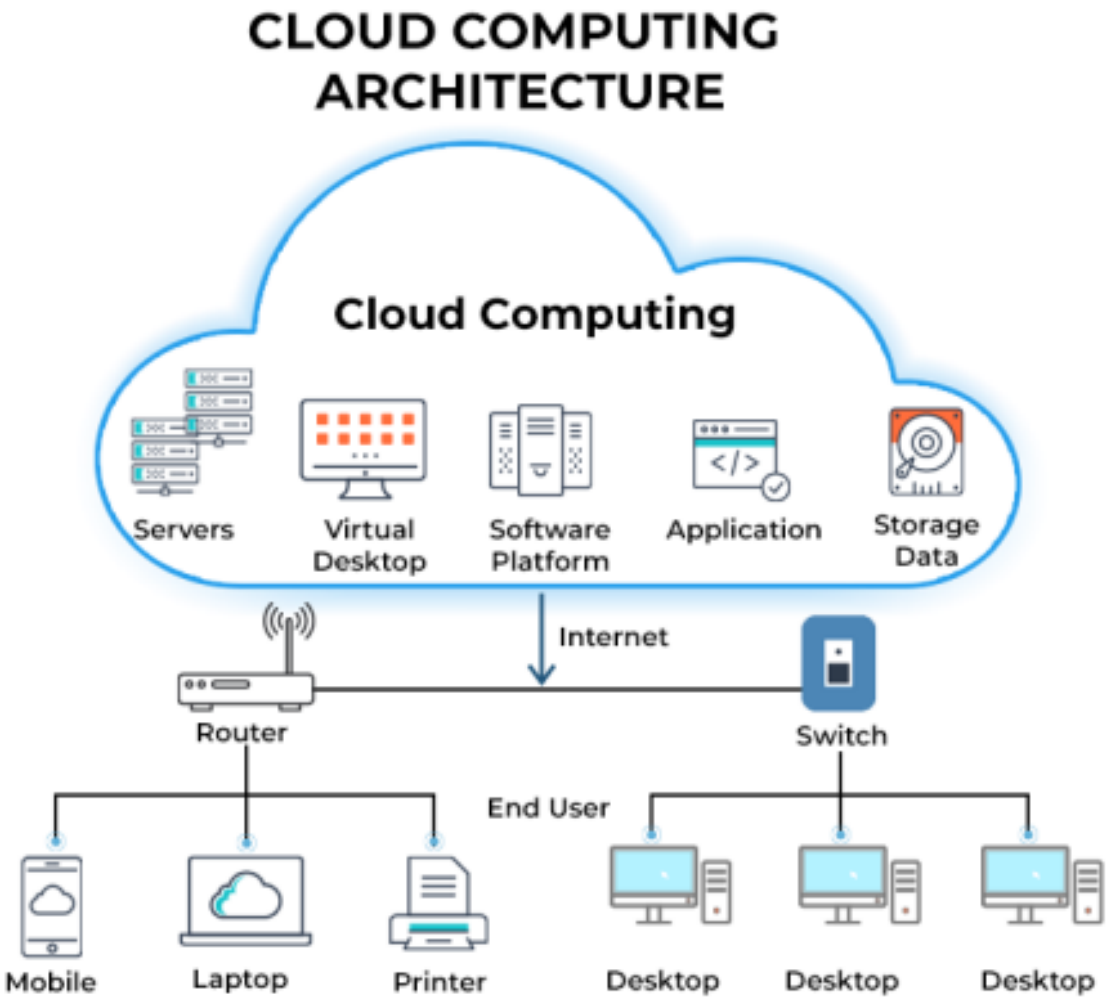
2. Flexibility: Cloud services offer a wide range of services that can be customized to meet specific needs.

3. Cost Efficiency: Pay-as-you-go models allow organizations to optimize their expenditure by only paying for what they use.

4. Reliability: High availability and disaster recovery options are typically built into cloud services.

5. Global Reach: Cloud providers offer data centers worldwide, enabling low-latency access and global service delivery.

Instead of storing files on a storage device or hard drive, a user can save them on cloud, making it possible to access the files from anywhere, as long as they have access to the web.



The services hosted on cloud can be broadly divided into infrastructure-as-a-service (IaaS), platform-as-a-service (PaaS), and software-as-a-service (SaaS). Based on the deployment model, cloud can also be classified as public, private, and hybrid cloud.

Further, cloud can be divided into two different layers, namely, front-end and back-end. The layer with which users interact is called the front-end layer. This layer enables a user to access the data that has been stored in cloud through cloud computing software.

The layer made up of software and hardware, i.e., the computers, servers, central servers, and databases, is the back-end layer. This layer is the primary component of cloud and is entirely responsible for storing information securely. To ensure seamless connectivity between devices

linked via cloud computing, the central servers use a software called middleware Opens a new window that acts as a bridge between the database and applications.

Fundamentals of DevOps:

DevOps is a set of practices, cultural philosophies, and tools designed to increase an organization's ability to deliver applications and services at high velocity. It aims to unify software development (Dev) and software operations (Ops) for faster, more reliable, and efficient delivery.

Here are the core fundamentals of DevOps:

1. Collaboration and Communication

DevOps promotes a shared responsibility between development and operations teams. By breaking down silos and encouraging collaboration, teams can:

- Work together throughout the software lifecycle
- Share feedback continuously
- Improve productivity and reduce misunderstandings

2. Automation

Automation is the backbone of DevOps, helping streamline repetitive and error-prone tasks:

- **CI/CD pipelines** for automated code integration and delivery
- **Automated testing** for faster and more reliable quality checks
- **Infrastructure as Code (IaC)** for provisioning and managing infrastructure

3. Continuous Integration (CI)

CI involves merging code changes from multiple contributors into a shared repository several times a day. It ensures:

- Early detection of bugs
- Faster integration
- Shorter development cycles

4. Continuous Delivery (CD)

CD ensures that code changes are automatically built, tested, and prepared for release to production:

- Reduces time-to-market
- Improves release reliability
- Supports frequent, incremental updates

DevOps lifecycle

The DevOps lifecycle is a continuous software development process that employs DevOps best practices to plan, build, integrate, deploy, monitor, operate, and offer continuous feedback throughout the software's lifecycle.

It is often represented by a continuous loop diagram as seen below –

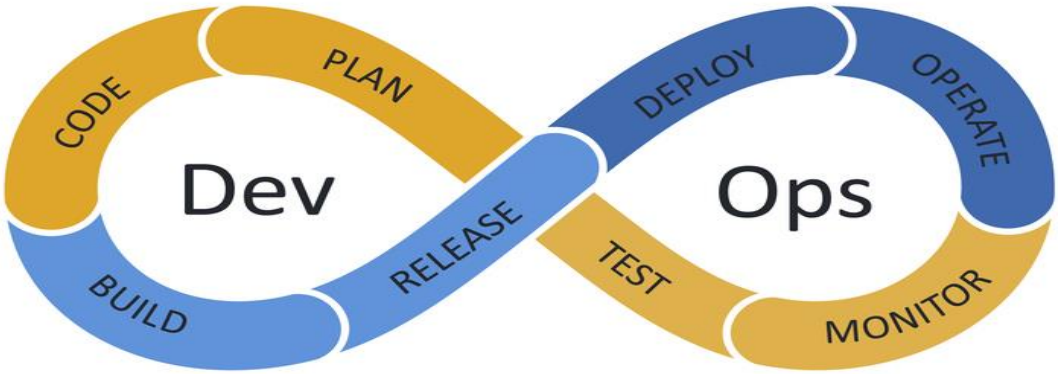


fig: DevOps lifecycle diagram

- **Plan** – The planning phase is exactly what it sounds like: planning the project's lifecycle. In contrast to conventional methods to the development lifecycle, this model assumes that each stage will be repeated as necessary. In this manner, the DevOps workflow is planned with the likelihood of future iterations and likely prior versions in mind. This implies that we will likely have information from past iterations that will better inform the next iteration, and that the present iteration will likewise inform the

next iteration. This stage often involves all teams to ensure that no area of the planning is ignored or forgotten.

- **Code** – The developers will write the code and prepare it for the next phase during the coding stage. Developers will write code in accordance with the specifications outlined in the planning phase and will ensure that the code is created with the project's operations in mind.
- **Build** – Code will be introduced to the project during the construction phase, and if necessary, the project will be rebuilt to accommodate the new code. This can be accomplished in a variety of ways, although GitHub or a comparable version control site is frequently used. The developer will request the addition of the code, which will then be reviewed as necessary. The request will be approved if the code is ready to be uploaded, and the code will be added to the project. Even when adding new features and addressing bugs, this method is effective.
- **Test** – Throughout the testing phase, teams will do any necessary testing to ensure the project performs as planned. Teams will also test for edge and corner case issues at this stage. An "edge case" is a bug or issue that only manifests during an extreme operating event, whereas a "corner case" occurs when many circumstances are met.
- **Release** – The release phase occurs when the code has been verified as ready for deployment and a last check for production readiness has been performed. The project will subsequently enter the deployment phase if it satisfies all requirements and has been thoroughly inspected for bugs and other problems.
- **Deploy** – In the deploy phase, the project is prepared for the production environment and is operating as planned in that environment. This would be the responsibility of the operations team; in DevOps, it is a shared responsibility. This shared duty pushes team members to collaborate to guarantee a successful deployment.
- **Operate** – In the operating phase, teams test the project in a production environment, and end users utilise the product. This crucial stage is by no means the final step. Rather, it informs future development cycles and manages the configuration of the production environment and the implementation of any runtime requirements.
- **Monitor** – During the monitoring phase, product usage, as well as any feedback, issues, or possibilities for improvement, are recognized and documented. This information is

then conveyed to the subsequent iteration to aid in the development process. This phase is essential for planning the next iteration and streamlines the pipeline's development process.

DevOps key features:

1. Continuous Integration and Continuous Delivery (CI/CD)

- Automates the process of code integration, testing, and deployment.
- Ensures faster, more frequent, and reliable software releases.
- Minimizes human error and manual intervention.

2. Automation

- Automates repetitive tasks like builds, tests, deployments, infrastructure provisioning.
- Enhances speed, consistency, and efficiency in the software development lifecycle.

3. Continuous Monitoring and Feedback

- Real-time monitoring of applications and infrastructure.
- Collects logs, metrics, and user feedback to detect and fix issues early.
- Helps improve performance and user satisfaction.

4. Collaboration and Communication

- Breaks down silos between development, operations, QA, and other teams.
- Encourages a culture of shared responsibility and transparency.
- Uses tools like Slack, Microsoft Teams, or Jira to streamline team workflows.

5. Infrastructure as Code (IaC)

- Manages infrastructure (servers, networks, configurations) using code and automation tools.
- Ensures consistency, version control, and reproducibility of environments.
- Tools: Terraform, Ansible, AWS CloudFormation.

6. Security Integration (DevSecOps)

- Embeds security checks early into the development pipeline.
- Automates vulnerability scanning, compliance checks, and secure coding practices.
- Balances speed with security and risk mitigation.

Concept of dockers and containers:

Containers are packages of software that contain all of the necessary elements to run in any environment. In this way, containers virtualize the operating system and run anywhere, from a private data center to the public cloud or even on a developer's personal laptop. Containers make it easy to share CPU, memory, storage, and network resources at the operating systems level and offer a logical packaging mechanism in which applications can be abstracted from the environment in which they actually run.

Docker is the world's leading software container platform. It was launched in 2013 by a company called Dotcloud, Inc which was later renamed Docker, Inc. It is written in the Go language. It has been just six years since Docker was launched yet communities have already shifted to it from VMs. Docker is designed to benefit both developers and system administrators making it a part of many DevOps toolchains. Developers can write code without worrying about the testing and production environment. Sysadmins need not worry about infrastructure as Docker can easily scale up and scale down the number of systems. Docker comes into play at the deployment stage of the software development cycle.

Docker is the containerization platform that is used to package your application and all its dependencies together in the form of containers to make sure that your application works seamlessly in any environment which can be developed or tested or in production. Docker is a tool designed to make it easier to create, deploy, and run applications by using containers.

Benefits of DevOps:

- **Faster Time to Market:** CI/CD pipelines and automation lead to quicker release cycles.
- **Improved Quality:** Automated testing and continuous integration catch bugs early, resulting in higher quality software.
- **Increased Scalability:** DevOps practices enable organizations to scale their infrastructure quickly and efficiently to meet demand.
- **Better Resource Utilization:** Automation and optimization of processes improve resource utilization.
- **Enhanced Security:** Continuous monitoring and testing help identify and mitigate security vulnerabilities.

Future scope of DevOps:

The future of DevOps is very optimistic and more organizations are going to adopt this technology soon. DevOps processes are evolving as new technologies and tools are introduced. In the domain of software development, the additional operating costs and times will continue

if the development processes aren't optimized or are ineffective. DevOps helps to reduce redundancy in development processes by forming a single line of communication between developers and the operations team. This essentially establishes a continuous feedback loop, making it easy to find bugs and quickly fix them. The demand for DevOps will get at its peak as companies are including DevOps in their company processes. With this, they will have more control over the whole pipeline, which will allow the teams to operate more efficiently with less repetition. The DevOps future scope will be extremely competitive and growing rapidly, with organizations that range from large businesses to startups enjoying the advantages. Furthermore, many reports predict that DevOps industry trends will continue to rise, as companies in the software development sector focus more on obtaining advanced DevOps services and tools to optimize specific functions of management and software deployment processes.

Conclusion:

In conclusion, the integration of DevOps with cloud computing has revolutionized the way software is developed, deployed, and maintained. By combining the agility of DevOps with the scalability and flexibility of the cloud, organizations can deliver high-quality software at a faster pace. Key features like automation, continuous integration, and Infrastructure as Code enable teams to innovate rapidly while maintaining system reliability and security. Cloud platforms such as AWS, Azure, and Google Cloud offer robust toolsets that support the full DevOps lifecycle. The synergy between DevOps and the cloud reduces operational complexity, enhances collaboration, and improves resource utilization. As digital transformation continues to grow, adopting DevOps in cloud environments is no longer optional but essential. It empowers businesses to remain competitive, resilient, and customer-focused. Ultimately, DevOps in cloud computing represents a strategic approach to achieving efficiency, speed, and continuous improvement in modern IT operations.

REFERENCES

[1]. https://www.browserstack.com/guide/devops-lifecycle

[2]. https://katalon.com/resources-center/blog/devops-lifecycle

[3]. https://www.geeksforgeeks.org/introduction-to-docker/?ref=rbp

[4]. https://about.gitlab.com/topics/devops/

[5]. https://en.wikipedia.org/wiki/DevOps

[6]. https://www.techtarget.com/searchitoperations/definition/DevOps