The Evolution of DevOps and the Emergence of GitOps: A Standard Literature Review

Sonal Joshi
Department of Computer Science
University of New Brunswick
Fredericton, NB, Canada
sonal.joshi@unb.ca

***Abstract*—DevOps is a culture of software development that emphasizes collaboration, automation, and communication between development and operations teams to accelerate product delivery, improve quality, and enhance security. This literature review provides an overview of DevOps as a software development methodology, focusing on the recent trends: Infrastructure as Code (IaC), DevSecOps, and GitOps. IaC enables infrastructure to be managed through code, providing greater scalability, repeatability, and consistency. DevSecOps integrates security into DevOps workflows, emphasizing the importance of security as an integral part of the software development process. GitOps is highlighted as the state-of-the-art approach for managing infrastructure and application configuration, enabling teams to manage changes in a more scalable, repeatable, and auditable way. Challenges related to implementing GitOps are also discussed. Finally, the review covers emerging research directions in the field of DevOps, emphasizing the importance of staying up-to-date with evolving practices to improve software delivery and meet the demands of modern IT operations.**

***Index Terms*—DevOps, GitOps, DevSecOps, DevOps Cloud, Infrastructure as Code (IaC), CI/CD**

I. Introduction

DevOps is a software engineering culture and philosophy that emphasizes collaboration, communication, and integration between software development (Dev) and information technology operations (Ops) teams. It aims to shorten the system development life cycle, while continuously delivering high-quality software products. DevOps has gained increasing popularity in recent years due to its ability to speed up software delivery time and improve software quality, security, and collaboration between teams. As DevOps is still an emerging concept, its definitions and best practices are still evolving, which can make its implementation in practice less informed and somewhat risky. Therefore, there is a growing need for in-depth research into the implementation of DevOps in practice to understand its adoption, benefits, and challenges.

Building an effective DevOps process requires a deliberate culture of seamless collaboration. Pipelines, code merges, testing, validation, and a high degree of automation are required to quickly and safely move code from design and build through to deployment and production. This is reflected in Continuous Integration / Continuous Delivery (CI/CD), an instrumental part of the DevOps framework. CI is the philosophy of getting tests integrated with code automatically, and CD involves deploying that code frequently. Using a continuous and iterative process to build, test, and deploy can help avoid bugs and code failures.

DevOps practices allow development, security and operation teams to collaborate and work together to shorten the software development life cycle, sometimes from months to hours.

There currently isn't a single all-encompassing platform to cover the entire scope. DevOps teams usually put together a customized toolchain to connect the various people and workflows that consist of open-source and vendor tools. The output from one is an input for another, and so on. That leads to a very fragmented vendor landscape. Each organization selects the toolchain they want to leverage for its processes. When a company is leveraging DevOps, its toolchain will cover stages like planning, creation, testing, releasing, and monitoring. Most DevOps vendors will specialize in a particular area of the toolchain, such as planning, issue tracking, source code control, code writing, testing, configuration management, deployment, or monitoring. [1]

A. *How is DevOps and CI/CD related?*

CI/CD - the combination of Continuous integration and Continuous delivery is an essential part of DevOps and any modern software development practice. A purpose-built CI/CD platform can maximize development time by improving an organization's productivity, increasing efficiency, and streamlining workflows through built-in automation, continuous testing, and collaboration. As applications grow larger, the features of CI/CD can help decrease development complexity. Adopting other DevOps practices like shifting left on security and creating tighter feedback loops helps break down development silos, scale safely, and get the most out of CI/CD. [2]

II. Literature Review

Authors Ricardo Amaro, Ruben Pereira and Miguel Mira da Silva in [3] conducted a multivocal literature review (MLR) which explores the various perspectives and voices within the literature on DevOps. The authors analyze a total of 93 articles and identify five different categories of literature: conceptual, empirical, experience reports, tool-focused, and opinion-based. They also identify several themes within each category, such as the importance of culture and communication, the use of automation and measurement, and the need for collaboration and integration. The authors note that while there is a broad

consensus on the core principles and practices of DevOps, there are also significant differences in how DevOps is understood and implemented in different contexts. They highlight the importance of understanding the diversity of perspectives and experiences within the DevOps community in order to gain a more nuanced understanding of the field and to identify best practices that are relevant to specific organizational contexts. Table 1 highlights the Publication Properties Identified from the MLR.

Property	Total
Interchangeably mentions Capabilities and Practices	66
Mentions Capabilities directly	19
Presents different or reorganized Capabilities compared to Senapathi et al. [4]	14
Relates Capabilities to Practices or distinguish them	8
Indicates a definition for Capability	6
Indicates a definition for Practice	1

TABLE I
SIX PUBLICATION PROPERTIES IDENTIFIED FROM THE MLR. [3]

Authors Ruth W. Macarthy and Julian M. Bass in [5] present a study on the taxonomy of DevOps practices in software development companies and software-intensive companies in the financial and public sectors based in the UK, Netherlands, and Africa. The study involved eleven practitioners from nine organizations who had at least two years of experience using DevOps practices. The research involved four main aspects of data analysis using Glaserian Grounded Theory: open coding, memoing, constant comparison, and saturation. Open coding involved identifying concepts found within the interview transcripts by line-by-line coding of participants' responses without any pre-determined codes. The authors used brief descriptive phrases to represent codes. After an initial comparison of the two independent transcripts, the codes were merged into a single set of codes. As shown in Table 2, the study classified the organizations based on staff count, adopted from the EU Recommendation 2003/361. The classification of organizations in this study showed a summary of their description. Some had colocated teams, while others were geographically distributed. The diversity in the research sites provided richness to the data and lent credence to the results. The study provided insights into the implementation of DevOps practices in different organizations, highlighting the challenges and benefits of DevOps practices in software development.

Authors Arun Sojan, Ranjit Rajan, and Pasi Kuvaja in [6] discuss a monitoring solution that provides visibility into the security of applications and infrastructure deployed in a cloud environment. The authors proposed an architecture of the monitoring solution, based on open-source tools such as Prometheus and Grafana, and the solution consists of three components: a data collector, data processing and data storage, and a visualization layer. Data collectors are responsible for collecting metrics from a variety of sources, such as Kubernetes clusters, Docker containers, and network devices. The collected data is then processed and stored in a time series database for analysis. The visual layer provides users with a dashboard to view metrics and perform analysis. This proposed solution has benefits such as increased visibility into cloud-native application security, real-time monitoring and alerting, and the ability to monitor a large number of services in an environment. dynamic cloud field. The authors also suggest that the proposed monitoring solution can be used by organizations adopting DevSecOps methodologies and deploying applications in a cloud environment.

In [7], authors Juncal Alonso, Radosław Piliszek, and Matija Cankar discussed the importance of embracing Infrastructure as Code (IaC) through the DevSecOps philosophy, which emphasizes security throughout the development and deployment process. The authors have defined IaC and its key benefits, such as faster provisioning, reduced errors, and better version control and then introduced the concept of DevSecOps, which integrates security into the development and operations process. The article then confers the challenges of implementing IaC in a DevSecOps environment. These challenges include the need for collaboration between development, operations, and security teams, the complexity of managing large infrastructure environments, and the need for automation tools and processes. To address these challenges, the authors propose a reference framework for implementing IaC in a DevSecOps environment as shown in Table 3.

IaC Challenge	DevSecOps framework component
Market fragmentation	DevSecOps framework (all components)
Requirement of wide (IaC) skills	DevSecOps framework (all components)
Definition of well-known IaC code patterns	IaC generator, IaC execution manager, infrastructure monitoring, self-learning IaC, self-healed IaC
Difficulty in replicating errors	IaC scanner, canary sandbox environment, infrastructure monitoring, self-learning IaC, self-healed IaC
IaC languages specificities and tools heterogeneity	DOML editor, IaC generator, IaC execution manager
Security and trustworthiness	Model checker, IaC scanner, infrastructure monitoring
Configuration drift	IDE, IaC generator
Changing infrastructure requirements	DOML editor, infrastructure optimizer, infrastructure monitoring, self-learning IaC, self-healed IaC

TABLE III
THE RELATIONSHIP OF THE CHALLENGES FOR TRUSTWORTHY IAC DEVELOPMENT AND THE COMPONENT ADDRESSING IT IN THE PROPOSED SOLUTION. [7]

The authors Amr Ibrahim, Ahmed H. Yousef and Walaa Medha in [8] discuss the need for integrating security practices into DevOps workflows, specifically in the context of Infrastructure as Code (IaC) over cloud environments. The authors introduced a DevSecOps model that covers different stages of the software development lifecycle, including planning, coding, testing, and deployment. The model is based on the principle of "shift-left," which means integrating security

Organisation	FinCo1	FinCo2	FinCo3	FinCo4	ITCo	RegCo	FreeCo	PubCo	FinCo5
Size	Large	SME	SME	Large	Large	SME	SME	Large	Large
Business Type	Financial	Financial	Insurance	Financial	IT Consulting	Regulatory	IT Consulting	Public	Financial
Team Location	Distributed	Co-located	Co-located	Distributed	Distributed	Co-located	Co-located	Co-located	Co-located
Team Types	Developers DevOps Ops	Developers DevOps Ops	Developers DevOps Ops	Developers DevOps Ops	Developers DevOps	Developers DevOps	Developers Ops	Developers Ops	Developers
Tools	GitHub, Kubernetes, Ansible, Docker, Azure DevOps, Terraform, Istio	GitHub, SonarCube, Docker, Azure DevOps, Selenium, Veracode, Slack	Kubernetes, GitLab, Ansible, Docker, Bamboo, Jenkins, Terraform, Vault	Jenkins, Jira, Slack, Gitlab	Github, AWS cloud formation and other AWS tools, New Relic, Slack, Terraform	Gitlab, Slack, Kibana, Grafana, Jenkins, Jira, Terraform	Azure DevOps, Skype, Slack, GitHub	-	Github, Docker, Ansible, Azure DevOps, Terraform, Slack
Practices	Scrum meetings CI/CD	Scrum meetings CI/CD	Scrum meetings CI/CD	Scrum meetings CI/CD	Scrum meetings CI/CD	Scrum meetings CI/CD	Scrum meetings CI/CD	Scrum meetings	Scrum meetings CI/CD
Software Methodology	Scrum, Spotify	Scrum	Scrum, Kanban	Scrum, Spotify	Scrum	Scrum, Kanban	Scrum	Scrum	Scrum

TABLE II
DESCRIPTION OF ORGANIZATIONS IN THE STUDY [5]

practices earlier in the development process to detect and fix vulnerabilities at an early stage. They have also highlighted the importance of using automated tools and processes to ensure continuous monitoring and testing of the cloud infrastructure. The paper also considered several security challenges that organizations face when implementing IaC over cloud environments. Some of these challenges include securing access keys, securing communication between cloud resources, and ensuring compliance with regulatory requirements. The authors propose several security measures to address these challenges, such as implementing security controls in the IaC templates, using secure communication protocols, and automating compliance checks.

Authors Florian Beetz and Simon Harrer [9] provide an overview of the GitOps approach to software delivery and operations, exploring how it differs from traditional DevOps practices. They argue that GitOps is a natural evolution of DevOps, as it builds on the principles of infrastructure as code (IaC) and automates deployment through the use of Git repositories. It also outlines the key features of GitOps, including the use of a declarative model for infrastructure configuration, continuous delivery through Git repositories, and the use of version control to track changes to the system.

Papers [10] and [11] highlight the growing importance of GitOps in DevOps practices. GitOps is gaining traction as a state-of-the-art approach in DevOps due to its ability to provide version control, auditability, and simplified deployment. The use of GitOps in network management and IoT edge computing also demonstrates the versatility of this approach and its potential to be applied to a wide range of use cases.

In [11], authors Vojdan Kjorveziroski, Łukasz Łopatowski, Pavle V. Vuletic, and Frédéric Loui describe Network Management as a Service (NMaaS), which is a software suite that allows teams to centrally manage multiple remote infrastructures through fast deployment and configuration of network management applications and supporting tools. The authors used a GitOps-based approach for NMaaS to version application configuration and automatically sync it to running instances, enabling easy migration from existing installations and roll-back of recent configuration changes. The paper outlines the NMaaS architecture, discusses possible use cases, and showcases the application deployment process together with the steps required for extending the existing application catalogue. The proposed demo discusses NMaaS deployment options, showcases how the development virtual machine can be used for quick evaluation of the NMaaS software, and demonstrate the extensibility of NMaaS by adding a previously unsupported application to the NMaaS catalogue using the new application wizard directly from the Portal.

[10] presents a GitOps-based approach to continuous deployment in IoT edge computing. According to Ramón López-Viana, Jessica Diaz, and Jorge E. Pérez, authors, GitOps can offer various advantages in this situation, including version control, auditability, and simpler deployment. Additionally, they provide a case study of a continuous deployment system built on GitOps for an IoT edge computing application.

These two papers show how adaptable and scalable GitOps can be as a management strategy for sophisticated infrastructure and systems. These strategies make use of GitOps concepts to manage the network infrastructure and IoT edge computing more effectively and efficiently, advancing the state-of-the-art in DevOps.

III. GITOPS: STATE-OF-THE-ART IN DEVOPS

GitOps utilizes Git as the primary reference point for defining all aspects of a cloud-native system. By declaring all code, configurations, and policies in Git, a GitOps agent (such as Flux) can automatically apply these changes across

various environments such as development, testing, staging, and production. In case there is any inconsistency between the Git repository and the cluster, GitOps alerts developers and Kubernetes reconcilers can update or revert the cluster accordingly. With Git as the core component of delivery pipelines, developers can leverage their familiar tools and make pull requests to expedite and streamline the deployment of applications and policies that govern software delivery from start to finish.

At its core, GitOps is code-based infrastructure and operational procedures that rely on Git as a source control system. It's an evolution of Infrastructure as Code (IaC) and a DevOps best practice that leverages Git as the single source of truth, and control mechanism for creating, updating, and deleting system architecture. More simply, it is the practice of using Git pull requests to verify and automatically deploy system infrastructure modifications. [12]

GitOps incorporates almost all DevOps practices, except for communication and microservices. The most significant DevOps practice in GitOps is the heavy use of CI/CD pipelines, which is extended to continuous infrastructure deployment. This leads to the adoption of IaC in GitOps, with more detailed specifications on how it should be applied compared to DevOps. Monitoring is another practice that is common in both GitOps and DevOps, with the role of continuously receiving feedback about the runtime behaviour of software in production. However, GitOps extends this to infrastructure, enabling operators to recreate misbehaving infrastructure and applications. GitOps also employs monitoring for automation and provides more detailed specifications on how it should be used compared to DevOps.

By offering a single method for managing infrastructure, code, and data, GitOps can be helpful to use MLOps, DevOps, and IaC. GitOps allows machine learning engineers to manage the full ML workflow, from data preparation to model training to deployment, using version control. Over the entire ML pipeline, this strategy assures repeatability, traceability, and transparency. GitOps may also automatically deploy ML models to real-world settings, simplifying the scaling and upkeep of ML systems.

Sample GitOps Workflow for Kubernetes:

This workflow shows how GitOps can be applied to manage infrastructure provisioning and software deployment in any cloud-native environment as depicted in Fig 1:

1) Developers make changes to the application source code, which is stored in a Git repository. After committing these changes, a container image is generated and stored in the container registry.
2) There is a separate Git repository that maintains the deployment configuration for the application.
3) Any updates to the Git repository are tracked by the GitOps agent. Upon detecting a change, the agent triggers the deployment of the application or infrastructure.
4) The updated container image is deployed to either the production or staging environment.

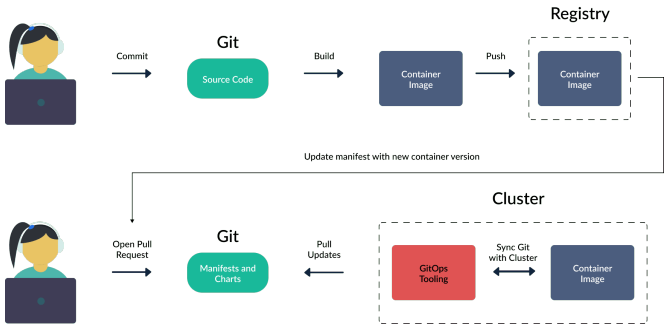

Fig. 1. Basic GitOps workflow for Kubernetes [13]

5) Finally, the developers can test the application and confirm that the cluster state aligns with the Git state, verifying the success of the deployment.

Recently, Weave GitOps was launched which is a tool that helps organizations implement GitOps practices to manage their Kubernetes infrastructure and applications. GitOps is a way of managing infrastructure and applications using version control, with Git as the single source of truth. Weave GitOps allows teams to define their infrastructure and application configurations as code, and then automatically apply changes to their Kubernetes clusters based on changes to the Git repository.

Weave GitOps offers different tiers to cater to different stages of operational maturity. The first tier is Weave GitOps Core, an open-source and free solution that automates the application lifecycle for developers and DevOps teams. The second tier is Weave GitOps Enterprise, which is designed to bring continuous operations and control to Kubernetes fleets for platform teams and operators. Weave GitOps Enterprise supports managed Kubernetes services like AKS, EKS, and GKE and self-installed clusters from Rancher, OpenShift, and Tanzu.

A. *Benefits*

There are several benefits of using GitOps for managing infrastructure and applications, including:

- **Faster deployments:** GitOps streamlines the deployment process, enabling faster application releases by keeping everything version-controlled in a single Git repository, eliminating the need for switching between different tools.
- **History of environment changes:** GitOps allows for easy tracking of changes made to application environments by storing all configuration updates in a Git repository. This results in a full history of the desired state changes, including who made the changes and why.
- **Secure deployments:** GitOps provides a secure deployment process by enabling management from within the environment, thus avoiding the need for developer access to the environment.

- **Tool-independent architecture:** GitOps architecture is flexible and not tool-specific, allowing for the integration of different tools that work best for the team.
- **Built-in backups:** Since the state of the environments is stored in Git repositories, teams never lose the data even if something happens to the cluster, making the Git repository a natural backup of the cluster state.
- **Ease of adoption:** GitOps is easy to adopt as it utilizes common tools, and Git is the de facto standard for version control systems.
- **Transparency and clarity:** GitOps provides transparency and clarity to an organization's infrastructure needs around a central repository, scaling contribution input from team members and building communication loops for infrastructure changes.

B. Comparison

GitOps is a modern approach to infrastructure and application management that has evolved from DevOps practices. While there are similarities between the two approaches, there are also some key differences as seen in table 4.

	DevOps	**GitOps**
Comparison by Principles		
Holism	Yes	No (Developer-centric)
Incremental approach	Yes (Advances product as a whole)	Yes (Advances infrastructure)
Continuity	Yes	Yes (CD technique with specific pattern)
Automation	Yes	Yes (CI/CD pipelines, IaC)
Self-service	Yes (Create environments via environment repository)	Yes (Declaratively describable environments)
Iterative approach	Yes	No (Automatic deployments make it easy to iterate)
Collaboration	Yes	No (A reduced collaboration between teams, but still possible)
Comparison by Practices		
Infrastructure as Code (IaC)	Yes (Formulate requests to self-service environments)	Yes (Specifies when and how IaC should be applied)
Continuous Integration/-Continuous Deployment (CI/CD)	Yes	Yes (Extended with continuous infrastructure deployment)
Monitoring	Yes (Observability of software in production)	Yes (Observability of infrastructure for pull-based deployments)
Communication	Yes	No (Less important due to self-service environments)
Microservices	Yes	No (Not directly employed, but not restricted)

TABLE IV
COMPARISON OF DEVOPS AND GITOPS BY PRINCIPLES AND PRACTICES

C. Use Cases

Some of the use cases of GitOps are:

- **Smart city services:** Implementing smart city services can be challenging, requiring the ability to roll out and manage a complex platform. GitOps practices can help by providing a way to manage configuration as code and automate the deployment of changes to the infrastructure and applications.
- **Network slicing:** GitOps can enable service providers to differentiate service tiers and allow users to pay only for the bandwidth they need or use. This can lead to premium pricing for high-bandwidth services like video streaming and lower prices for connected IoT devices.
- **Edge computing:** Managing a large number of edge nodes can be complex, especially when trying to provision and manage them at scale. GitOps can help by providing a standardized approach to configuration management and automation, making it easier to manage edge computing infrastructure.
- **Disaster recovery:** GitOps can be used to automate disaster recovery processes. By using Git to manage infrastructure as code, teams can easily recreate production environments in the event of a disaster, reducing downtime and minimizing the impact on business operations.
- **Canary releases:** GitOps can also enable canary releases, which involve gradually rolling out a new version of the application to a small subset of users to test its performance and stability before rolling it out to all users. This can help minimize the risk of downtime or performance issues.

D. Tools

There are several GitOps tools available in the market that can be used to implement GitOps workflows. Below are the most popular GitOps tools:

1) **ArgoCD**: It is a GitOps continuous delivery solution that is open-source and made especially for Kubernetes. Using Git as the sole source of truth, it automates the deployment of applications to a Kubernetes cluster. ArgoCD offers a web-based user interface with a lot of plugins and extension flexibility for managing deployments.
2) **Flux:** It is a GitOps tool that is open-source and automates the distribution of applications to Kubernetes clusters. Flux operates by continuously scanning a Git repository for updates and applying them to the Kubernetes cluster automatically. GitOps' best practices like declarative configuration and version control are supported by Flux.
3) **Jenkins X:** It is a GitOps continuous delivery solution that is cloud-native and optimized for Kubernetes. By leveraging Git as the sole source of truth, Jenkins X automates the whole CI/CD pipeline. It offers a subjective perspective on software development and best practices for deployment, versioning, and testing.
4) **GitLab:** It is an all-encompassing DevOps platform with GitOps features. For source code management, continuous integration and delivery, and container registry, GitLab offers a full suite of tools. It supports GitOps best practices including declarative setup and version control and has a web-based interface for managing deployments.

5) **Weave Flux:** It is a fork of Flux that has been modified and enhanced by WeaveWorks. Weave Flux uses an operator that watches for changes and applies them to the cluster. Weave Flux has some additional features such as multi-tenancy support and automated rollbacks.

IV. Challenges

- DevOps requires collaboration between teams, but silos between teams can hinder collaboration and slow down the development process. Teams need to work together to break down silos and improve communication.
- DevOps requires the integration of multiple tools and technologies, such as source control, continuous integration, and continuous delivery. Integrating these tools can be challenging, and organizations need to ensure that they work together seamlessly.
- Due to their potential incompatibility with current tools and technology, legacy systems might be a barrier to the implementation of DevOps principles. Companies must carefully consider their strategy before merging legacy systems with DevOps techniques.
- It can be difficult to develop the continuous improvement culture that is necessary for DevOps. When making improvements to their processes, organizations must be receptive to criticism.
- GitOps requires knowledge of several tools and technologies, such as Kubernetes, Git, and CI/CD pipelines. The learning curve can be steep, and organizations may need to invest in training and upskilling employees.
- GitOps requires access to sensitive information, such as passwords and credentials, which can be a security risk if not managed properly. Organizations need to implement strong security practices, such as role-based access control and encryption, to mitigate these risks.
- When a company builds a complex application using GitOps, it can become challenging to audit the final product. This is because the application may have multiple repositories, numerous configuration files, and a large number of data points. Although enabling additional observability tooling can provide better visibility into the application's operations, it goes against the core principles of GitOps, which aim to break down the barriers between developers and operators.

V. Future research directions

Several research areas require future research as DevOps develops and gains popularity in the software world. These research directions include, among others:

- Cloud-Native DevOps: As more organizations move towards cloud-native architectures, there is a need for research on how DevOps practices can be applied to these architectures. Topics of research in this area may include the use of container orchestration tools such as Kubernetes and the application of DevOps practices to serverless architectures.
- There is a need for research on how to include security principles into the DevOps process as firms place a greater emphasis on security. Secure code review, threat modelling, and penetration testing are just a few examples of the subjects that might be studied in this field.
- DevOps approaches that can enable serverless computing are required because it has evolved as a new paradigm for creating and deploying applications. Future studies in this field can look at how DevOps methodologies can be modified to support serverless computing and how they can be utilized to enhance the scalability, availability, and performance of serverless applications.
- Study of the interactions between DevOps and cutting-edge technologies including blockchain, quantum computing, and edge computing, as well as their prospective effects on DevOps techniques.
- DevOps as a Service: AI/MLops can be used to create a new class of DevOps services that are fully automated and intelligent. This could include automated code review and optimization, intelligent deployment pipelines, or self-optimizing infrastructure management.

VI. Conclusion

Bringing together development and operations teams to increase collaboration, automation, and continuous delivery, DevOps has become a crucial method of software development and delivery. DevOps deployment has its difficulties, including cultural resistance, toolchain complexity, security, legacy systems, and a lack of skills and knowledge, but it is quickly emerging as a crucial strategy for businesses looking to remain competitive. DevOps has developed, and new subtypes like DevSecOps, IaC and GitOps have appeared.

GitOps has grown in prominence as a cutting-edge method for managing infrastructure, in which the entire infrastructure is documented and controlled as code in a Git repository. Future research in DevOps and GitOps should focus on several different aspects such as the use of machine learning and artificial intelligence, the usage of serverless architectures, the creation of new tools and technologies, the enhancement of security, and the spread of DevOps to other divisions of the company outside of development and operations. The need for DevOps and GitOps will only grow as software development picks up speed, and enterprises will need to invest in platforms that can serve as the key building blocks of the software development process to stay competitive.

References

[1] C. Research, "Evolution of devops." https://research.contrary.com/reports/evolution-of-devops, 2019.

[2] "The four phases of devops." https://about.gitlab.com/topics/devops/#the-four-phases-of-dev-ops. Accessed: March 25, 2023.

[3] R. Amaro, R. Pereira, and M. M. da Silva, "Capabilities and practices in devops: A multivocal literature review," *IEEE Transactions on Software Engineering*, vol. 49, no. 2, pp. 883–901, 2023.

[4] D. Berkholz, "Six core capabilities of a devops practice," *The New Stack*, October 2020.

[5] R. Macarthy and J. Bass, "An empirical taxonomy of devops in practice," pp. 221–228, 08 2020.

[6] A. Sojan, R. Rajan, and P. Kuvaja, "Monitoring solution for cloud-native devsecops," in *2021 IEEE 6th International Conference on Smart Cloud (SmartCloud)*, pp. 125–131, 2021.
[7] J. Alonso, R. Piliszek, and M. Cankar, "Embracing iac through the devsecops philosophy: Concepts, challenges, and a reference framework," *IEEE Software*, vol. 40, no. 1, pp. 56–62, 2023.
[8] A. Ibrahim, A. H. Yousef, and W. Medhat, "Devsecops: A security model for infrastructure as code over the cloud," in *2022 2nd International Mobile, Intelligent, and Ubiquitous Computing Conference (MIUCC)*, pp. 284–288, 2022.
[9] F. Beetz and S. Harrer, "Gitops: The evolution of devops?," *IEEE Software*, vol. 39, no. 4, pp. 70–75, 2022.
[10] R. López-Viana, J. Díaz, and J. E. Pérez, "Continuous deployment in iot edge computing : A gitops implementation," in *2022 17th Iberian Conference on Information Systems and Technologies (CISTI)*, pp. 1–6, 2022.
[11] V. Kjorveziroski, P. Vuletic, Łopatowski, and F. Loui, "On-demand network management with nmaas: Network management as a service," 04 2022.
[12] Atlassian, "Gitops: Next big thing in devops?: Atlassian git tutorial." https://www.atlassian.com/git/tutorials/gitops, 2021.
[13] Codefresh, "Gitops at your service - codefresh." https://codefresh.io/learn/gitops/#section-gitops-at-your-service, Accessed: March 30, 2023.