ISSN: 2633-4828 Vol. 6 No.1, January, 2024

International Journal of Applied Engineering & Technology

ENHANCING HYBRID CLOUD INFRASTRUCTURE RESOURCE PROVISIONING AND MANAGEMENT THROUGH GITOPS WORKFLOW AUTOMATION

Arun Pandiyan Perumal
Illinois Institute of Technology, Fremont, California, USA
apandiyan@hawk.iit.edu

ABSTRACT

GitOps is a contemporary method of software development and deployment that only depends on Git repositories as the definitive source for managing the whole infrastructure and application lifecycle. By using automation, it streamlines the management of your cluster setup and application deployments, which were previously time-consuming tasks. GitOps merges the dependability of version control systems with the explicit characteristics of infrastructure as code (IaC) to provide a resilient and automated process for managing infrastructure. GitOps integrates the capabilities of Git, a version control system, with the concept of infrastructure as code (IaC) to enhance automation, traceability, and stability in infrastructure management. This article evaluates the advancements achieved in the provisioning and administration of hybrid cloud infrastructure via the use of GitOps workflow automation. Examine the fundamental ideas, strategies, and procedures that underpin GitOps, as well as its integration into hybrid cloud infrastructures. This integration results in improved security, consistency, and efficiency. GitOps offers a mechanized method for deploying and managing configurations. Organizations may achieve automation of pipelines for deploying and managing infrastructure and apps by using Git as a centralized and reliable source of truth. This methodology has the potential to decrease the likelihood of human mistakes and facilitate expedited restoration in the event of malfunctions.

Keywords: GitOps; Hybrid Cloud Infrastructure; Workflow Automation; Cloud Infrastructure Management; Cloud Resource Provisioning.

INTRODUCTION

The concept of cloud computing was first established over a decade ago. Over the past decade, it has experienced a surge in popularity, with a significant number of prominent corporations incorporating it to varying degrees. Organizations view it as a cost-effective model that enhances operational efficiency compared to on-premise infrastructure. Nevertheless, the issues of security, trust, and privacy continue to pose a significant problem for several companies when contemplating the migration to the cloud. These challenges impede many firms from fully transitioning to the cloud. There is a notable level of interest within the industry regarding hybrid architectures, which involve hosting enterprise applications both on-premise and in the cloud. Hybrid architectures provide numerous benefits, enabling developers to selectively determine which components should be maintained locally and which components should move (Chittibala, 2020).

The success of any organization in the current dynamic digital environment relies heavily on its ability to manage its infrastructure effectively. Historical infrastructure management systems have been characterized by their tedious nature, susceptibility to errors, and time-consuming nature. GitOps, an innovative approach, is becoming increasingly popular in the DevOps community. GitOps combines the functionalities of Git, a version control system, with the concept of infrastructure as code (IaC) to improve automation, traceability, and stability in infrastructure management. This article explores the concept of GitOps in-depth, focusing on its impact on infrastructure management, process optimization, and system reliability (Chittibala, 2019).

OBJECTIVES OF THE STUDY:

The main objectives of this study are as follows:

- To assess the implementation of GitOps in hybrid cloud infrastructures.
- To examine the impact of Git on improving infrastructure management.

- To identify the most effective methods and difficulties encountered when implementing GitOps.

BACKGROUND AND RELATED WORK

Hybrid Cloud Infrastructure:

A hybrid cloud refers to the combination of two or more separate cloud infrastructures, typically private and public clouds. Organizations employ this concept as an interim measure during the transition to cloud infrastructure. Certain services may remain confidential until all security and performance concerns have been addressed. Ensuring the security and synchronization of data in a hybrid cloud environment requires a significant amount of responsibility and skill (Thompson and Harris, 2020).

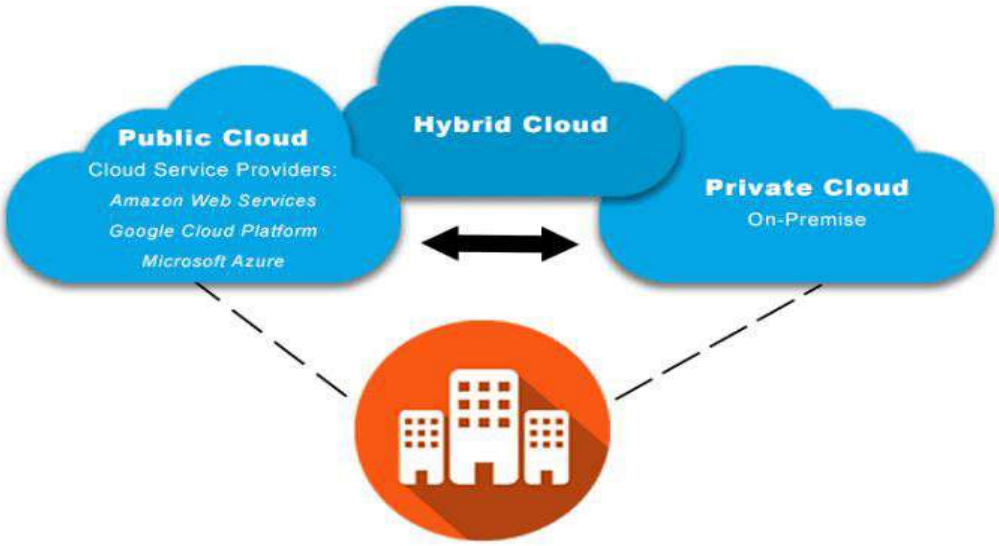

Figure 1: Hybrid cloud infrastructure (Chittibala, 2020)

GitOps: An Overview

GitOps is a contemporary method for continuous delivery that utilizes Git as a central repository for declarative infrastructure and workloads. The tool was first created in 2017 as a framework for operations, with the aim of improving collaboration and productivity in enterprises. In order to provide uninterrupted deployments, modern infrastructure needs to possess elasticity, which allows for efficient management of cloud resources. Contemporary and cloud-native applications prioritize swift development and scalability. Organizations with a robust DevOps culture can deploy code to production numerous times throughout the day, frequently achieving hundreds of deployments. GitOps is employed to automate the provisioning of infrastructure, particularly contemporary cloud infrastructure. Operations teams that adopt GitOps employ configuration files stored as code, similar to how teams use application source code (sometimes referred to as infrastructure as code). GitOps configuration files constantly generate the exact same infrastructure environment every time they are deployed, just like how application source code consistently produces the same application binaries during each build process. (Johnson and Davis, 2021).

The Evolution of Infrastructure Management

- **Traditional Challenges:** Historically, the management of infrastructure through manual processes had challenges in effectively monitoring changes and ensuring uniformity across different contexts. This resulted in operational inconsistencies and errors.
- **The Need for Change:** Organizations sought efficient and automated infrastructure management as the digital environment evolved to accommodate the demands of rapid development and deployment.

- **Introduction to GitOps:** GitOps emerged to enhance predictability and management by utilizing Git as the authoritative source for infrastructure configurations (Florian and Simon, 2021).

GitOps Core Concepts

- **Git as the Single Source of Truth:** GitOps leverages Git repositories to maintain a comprehensive record of all changes made to the infrastructure, guaranteeing accountability, uniformity, and visibility.
- **Infrastructure as Code (IaC):** Infrastructure definitions can be efficiently controlled and updated as Infrastructure as Code (IaC) treats infrastructure configurations as code.
- **Continuous Delivery and Continuous Deployment (CI/CD):** GitOps automates deployments when updates are pushed to Git repositories, leading to faster and more dependable deployments.

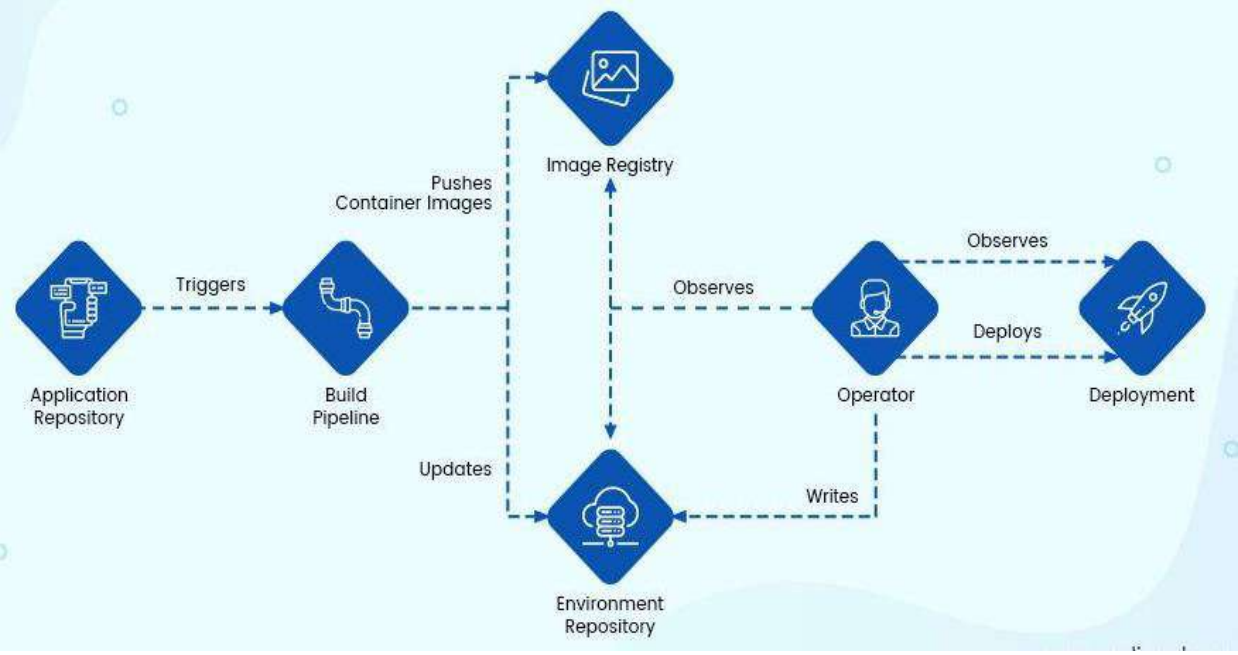

Figure 2: GitOps infrastructure (Florian and Simon, 2021)

Advantages of using the GitOps Methodology

GitOps allows enterprises to optimize their application deployment processes and infrastructure provisioning. This results in a wide array of advantages:

Simplicity in the Administration of Infrastructure:

GitOps enables users to efficiently manage infrastructure as an integral component of the broader DevOps workflow by rapidly testing and deploying modifications with the aid of:

- CI/CD tools
- Automated implementations
- Reduced feedback cycles

By integrating infrastructure into the CI/CD pipeline, any application modification that requires a change in infrastructure can be consolidated and managed as a single unit. Additionally, it enhances the ability to diagnose and resolve operational issues, such as network connectivity problems, as users gain improved clarity regarding the modifications made throughout the deployment process (Ramadoni et al., 2021).

Enhanced Efficiency:
Implementing source-controlled and verified infrastructure reduces the occurrence of configuration issues that can occur during deployments, enabling operations teams to efficiently identify and resolve these errors, resulting in time savings. In addition, source control enables concurrent collaboration across many teams on distinct components of the infrastructure without causing any disruptions to each other's efforts. Consequently, this leads to enhanced efficiency in both the development and operations teams, eventually resulting in expedited software development and deployment processes.

Improved Reliability and Stability:
In an infrastructure method that uses version control, changes to the infrastructure are verified, and the system's state is consistently preserved. When combined with the declarative method of providing infrastructure, it has the potential to decrease errors in the application infrastructure significantly.

- Effortlessly discern discrepancies between the stated infrastructure and the tangible infrastructure.
- Rapidly address or reduce their impact.

Standardization:
GitOps facilitates the standardization of infrastructure installations. Infrastructure may undergo a verification and validation process similar to that of application code, ensuring consistency.

- Complete processes from start to finish
- Uniform code structures
- Documentation
- Methods of testing

Enhanced Security:
The GitOps strategy enables enterprises to enforce security best practices and monitor all infrastructure modifications and their related states, which are accessible through Git SCM. In addition, this systematic technique allows for accurate audit trails to detect specific information pertaining to infrastructure modifications, such as

- Accountable individuals
- Time of deployment data
- Impacted assets

Challenges of GitOps

1. **Learning Curve*:*** Adopting a GitOps approach may provide a challenge for teams who are not experienced with Git and CI/CD pipelines.
2. **Complex CI/CD Pipelines:** Establishing and managing resilient CI/CD pipelines can be intricate, particularly for extensive deployments.
3. **Branch Management:** Handling branches and merges in Git repositories can pose difficulties in intricate systems and with multiple teams.
4. **Security Risks:** It is crucial to ensure that the Git repository is securely protected to safeguard the sensitive information it holds regarding infrastructure and applications.
5. **Limited Use Cases*:*** Although GitOps is very compatible with cloud-native systems, it may not be optimal for managing infrastructure in traditional data center configurations.

Workflow Automation in Cloud Infrastructure Management

Effectively managing IT infrastructure can pose significant challenges. Nevertheless, teams that embrace established software development practices such as version control, code review, and continuous integration/continuous deployment (CI/CD) pipelines have improved efficiency across the entire process. Config files enable the consistent deployment of the same infrastructure environment on every occasion. Numerous organizations are aware that using GitOps leads to enhanced efficiency, collaboration, and stability. However, they may be curious about the exact implications of adopting GitOps (Kamath et al., 2023).

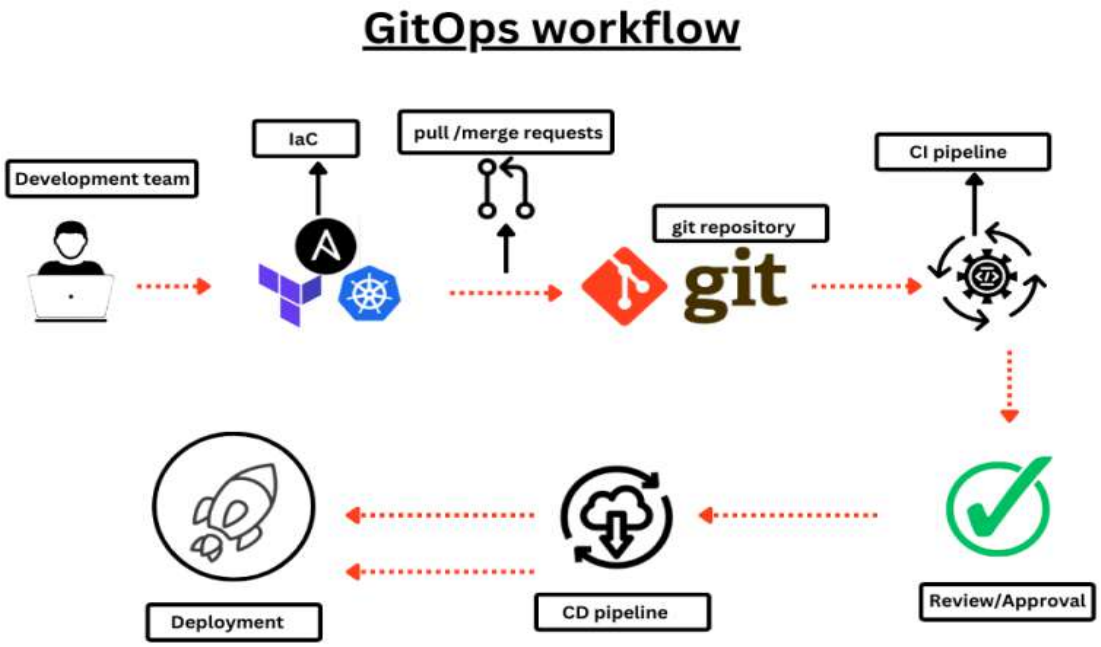

Figure 3: GitOps workflow (Kamath et al., 2023)

Three Components of GitOps Workflows

Infrastructure as Code (IaC):

The initial stage of a GitOps workflow involves establishing all infrastructure as code. Infrastructure as Code (IaC) streamlines the process of setting up IT infrastructure by employing configuration files. Infrastructure as Code (IaC) is a DevOps methodology that enables teams to manage and control infrastructure by treating it as software code. This approach ensures that infrastructure configurations are standardized across several computers, leading to smoother deployment processes (Alsurdeh et al., 2021). Infrastructure code follows a similar procedure as application code, involving continuous integration, version control, testing, and continuous deployment. Automation results in enhanced development efficiency, heightened uniformity, and accelerated time to market. Historically, the administration of infrastructure has been a labor-intensive procedure in which large teams are accountable for overseeing physical servers. Each computer frequently possesses its own configuration, resulting in the creation of snowflake settings. Infrastructure as code enables teams to achieve enhanced visibility, uniformity, reliability, and expandability (Chittibala, 2019).

Merge Requests (Mrs):

Declarative tools, like Kubernetes, allow configuration files to be managed by Git, an open-source version control system that monitors changes in code. GitOps leverages a Git repository as the authoritative and reliable source for infrastructure definitions, hence gaining advantages from a robust and comprehensive audit trail. The second

component of GitOps workflows is merge requests, which serve as the mechanism for implementing infrastructure upgrades (López-Viana et al., 2022). Teams engage in collaborative efforts during merge requests by conducting code reviews, providing comments, and offering suggestions. A merge commit is a record of changes made to the main branch and serves as an audit log. Teams can utilize the built-in rollback features to quickly return to a desired state and experiment with novel approaches to tackle complex difficulties. Merge requests enable the process of experimentation and offer a secure means for team members to promptly acquire feedback from their colleagues and experts in the field (Alsurdeh et al., 2021).

Continuous Integration and Continuous Deployment (CI/CD):

GitOps streamlines infrastructure management by leveraging a Git workflow alongside robust continuous integration and continuous deployment practices. Once the code is merged into the main branch, the CI/CD pipeline triggers the modification in the environment. Manual changes and mistakes made by humans can lead to configuration drift and the creation of unique and fragile environments. However, by implementing GitOps automation and continuous deployment, these issues can be resolved as the environment is consistently restored to its proper state (Kormaník and Porubän, 2023).

GitOps PRINCIPLES AND METHODOLOGIES

Declarative Infrastructure and Configuration

- Git as the Single Source of Truth: Git serves as the primary repository for storing infrastructure and application configurations, establishing a single source of truth for the intended state of the system (Chittibala, 2019).
- Kubernetes Deployment Example: Discover how GitOps enables the definition of Kubernetes manifests, including deployment and service YAML files, within Git, guaranteeing the provision of declarative and replicable infrastructure (Kratzke and Quint, 2017).

Git-Based Continuous Delivery:

- **GitOps Workflow:** Explore how GitOps facilitates continuous delivery through the utilization of Git's pull request and branching features. Automated deployments to the Kubernetes cluster are triggered by changes submitted to particular branches.
- **Kubernetes Deployment Example:** Investigate how a Git push to the production branch can immediately initiate a deployment, guaranteeing a seamless and regulated implementation of modifications to the Kubernetes environment (Koren et al., 2023).

Observability and Compliance:

- **Git's Audit Trail:** GitOps enables enterprises to maintain an inherent audit trail by leveraging Git's commit history. This allows for effective tracking and review of all changes made to their Kubernetes infrastructure.
- **Compliance and Kubernetes Example:** Discover how GitOps facilitates organizational compliance by ensuring that all configuration alterations undergo thorough review and approval procedures, resulting in a traceable record of changes (Chittibala, 2019).

Infrastructure as Code (IaC) and Automation:

- **IaC with GitOps:** GitOps facilitates the adoption of infrastructure as code (IaC) concepts by enabling automated and version-controlled provisioning, scaling, and updating of infrastructure.
- **Kubernetes Deployment Automation Example:** Examine how GitOps solutions, such as Flux or Argo CD, may seamlessly harmonize modifications made to the infrastructure-as-code repository with the Kubernetes cluster, guaranteeing uniform and automated deployment (Alsurdeh et al., 2021).

IMPLEMENTING GitOps IN HYBRID CLOUD ENVIRONMENTS

- **Set Up a Git Repository:** To begin, establish a Git repository that will function as the authoritative source for all modifications made to the infrastructure. The repository should be structured in a manner that accurately represents the distinct environments, including production, staging, and development (Leiter et al., 2023).
- **Define the Infrastructure as Code:** Subsequently, it is essential to establish the infrastructure as code. This entails developing code that encompasses the complete infrastructure, encompassing servers, networking, and storage. The code should be saved in the Git repository and tracked using Git version control.
- **Automate the Deployment Process:** Automating the deployment process is essential to guarantee that any modifications made in the Git repository are automatically applied to the associated environment. This can be achieved by implementing a continuous delivery pipeline that actively monitors the Git repository for any modifications and automatically deploys them to the designated environment (Gupta et al., 2022).
- **Monitor and Audit Changes:** It is crucial to observe and examine all modifications performed on the infrastructure consistently. This can be accomplished by utilizing tools that monitor modifications in the Git repository and offer insight into the complete deployment procedure.

CASE STUDY AND APPLICATION

Case Studies and Real-world Applications of GitOps Using Integration of AgroCD with Crossplane

Implementing GitOps methods in organizational workflows can significantly optimize deployment operations, ensure uniformity, and improve collaboration. ArgoCD is a famous tool that helps with GitOps, namely in the management of Kubernetes (K8s) clusters. Crossplane is a freely available framework for Kubernetes that is designed to provide cloud-native control planes using a uniform API, eliminating the need for manual coding. Crossplane simplifies the process of generating resources on different providers, such as GCP, AWS, and Azure, by leveraging configurations described as Kubernetes resources. This allows for better infrastructure management using standard Kubernetes capabilities. (López-Viana et al., 2022).

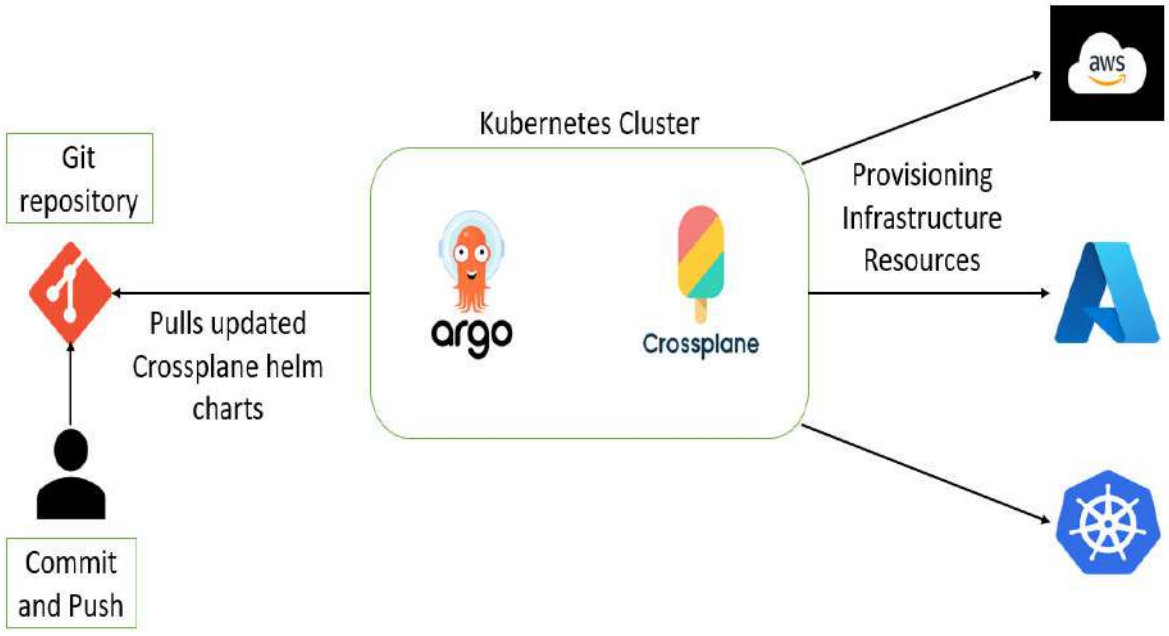

Figure 4: A simple architectural diagram of the GitOps model using Argocd with Crossplane (López-Viana et al., 2022)

Integrating Argo CD with Crossplane offers a robust toolkit for effectively managing infrastructure and application dependencies inside Kubernetes settings. Describing infrastructure as Kubernetes resources simplifies the process of overseeing the complete stack by using GitOps techniques and Kubernetes capabilities. Previously, resources that were indirectly connected to the application may now be efficiently controlled alongside the native resources of Kubernetes using declarative methods. The setting for the Bucket is fully transparent and very repeatable. Developing a straightforward technique to deploy additional instances of the application in any Kubernetes environment using Crossplane and Argo CD configuration.

Case Study: Successful Adoption of GitOps in a Financial Services Company

A well-established financial services organization aimed to improve its deployment processes and infrastructure management. The organization encountered difficulties such as varying settings, labor-intensive deployment procedures, and a lack of transparency in application management (Arundel and Domingus, 2019).

Challenges Encountered:

- **Inconsistent Environments:** The organization had challenges due to inconsistencies among development, staging, and production environments, resulting in frequent problems during deployment.
- **Manual Processes:** Manual interventions in deployment workflows were prone to errors and consumed a significant amount of time.
- **Lack of Transparency:** The deployment procedure was obscure, providing team members with only little insight into the application's status and settings (Zhou et al., 2019).

Implementation of GitOps using ArgoCD:

- **Consistency and Reliability:** The organization established consistency across environments by declaratively specifying the infrastructure and application configurations in Git. ArgoCD maintained a consistent state between the actual configuration in Kubernetes (K8s) and the desired configuration stored in Git (Ramadoni et al., 2021).
- **Automated Deployments:** ArgoCD's automatic synchronization of updates from Git to the K8s clusters led to a significant reduction in manual interventions.
- **Enhanced Transparency and Collaboration:** By incorporating Git into the deployment process, it promoted a culture of collaboration. Adopting pull requests and code reviews as routine practices has improved the quality and security of deployments (Wettinger et al., 2018).

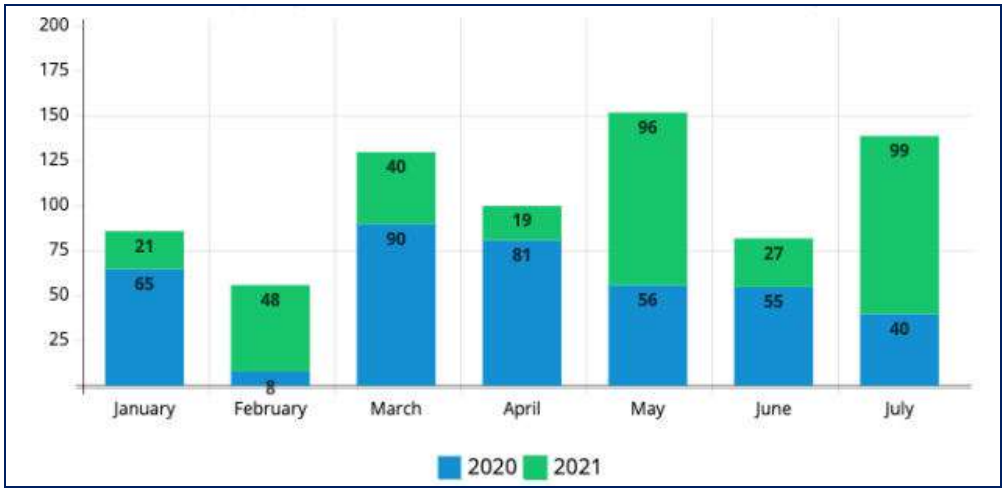

Figure 5: The monthly number of deployments rose from 2020 to 2021 after an application was introduced to GitOps (Ramadoni et al., 2021).

Benefits Realised:

- **Improved Deployment Frequency:** The organization experienced a significant rise in the frequency of deployment, which allowed for quicker implementation of new features.
- **Enhanced Security:** By utilizing the declarative approach and implementing a review process in Git, the security of their applications was enhanced.
- **Reduced Downtime:** The implementation of automated rollbacks and faster recovery processes resulted in decreased downtime and increased availability.

Implementing GitOps with ArgoCD revolutionized the company's operational processes, enhancing efficiency, security, and cooperation. This case study demonstrates the profound capacity of GitOps in practical applications, offering a model for enterprises seeking to improve their DevOps methodologies (Limoncelli,2018).

Use Cases of GitOps

- ***Static Websites:*** For the implementation of intricate static websites that consist of a collection of markdown files, which provide a more convenient way to edit site pages compared to raw HTML. A build is required to render it publishable. Alternatively, one can adjust the photos to be accessible in several dimensions, ensuring optimal visual appeal across numerous platforms. To accomplish these tasks, one can utilize GitOps or just duplicate the deployment onto the webserver (Bryant and Marin-Perez, 2018).
- ***Writing Books:*** Since books primarily consist of text, they can be easily integrated with a Version Control System (VCS), and a GitOps pipeline can be established immediately after completing the writing process. The pipeline has the capability to do grammatical and spelling checks automatically, as well as generate outputs in multiple formats such as doc, pdf, ePUB, and others (Cheng et al., 2022).
- ***GitOps for Documentation:*** Product documentation often utilizes markdown or Ascii documents due to their compatibility with text-based formats. These documents can be conveniently stored in version control systems such as Bitbucket or GitHub (Errea et al., 2023). Subsequently, a Continuous Integration (CI) tool retrieves the modifications and implements the updated version of the documentation. During the deployment stage, the latest version can be uploaded to numerous platforms whenever new changes are committed to the documentation (Goasguen and Hausenblas, 2018).

FUTURE DIRECTIONS AND RESEARCH OPPORTUNITIES

In the future, the evolution of GitOps will be influenced by advanced technology and the ever-changing demands of the industry (Thompson and Harris, 2020). The future trajectory of GitOps is likely to be influenced by various trends and innovations:

- Integration with AI and Machine Learning: Potential advancements in GitOps could entail the incorporation of artificial intelligence and machine learning to facilitate predictive analytics and intelligent automation. These technologies can offer valuable information on the performance of systems, anticipate problems before they happen, and streamline complicated decision-making processes (Gupta et al., 2022).
- **Enhanced Security through Policy:** Ensuring security will remain a top priority. GitOps may experience enhanced integration with policy-as-code frameworks, guaranteeing the constant enforcement of security policies throughout all phases of the deployment pipeline.
- **Hybrid and Multi-Cloud Management:** As enterprises increasingly implement hybrid and multi-cloud strategies, it is necessary for GitOps tools and procedures to adapt in order to handle settings across different cloud environments effectively. This adaptation should include providing unified administration and visibility.
- **Enhanced Observability and Feedback Loops:** It will be crucial to integrate extensive observability and feedback mechanisms into GitOps procedures. By implementing this, any deviations from the ideal state will

be instantly identified, and corrective measures will be automatically initiated, ensuring the system remains resilient.

CONCLUSION

GitOps is a notable improvement in managing hybrid cloud infrastructure. It provides a declarative, automated, and auditable method. GitOps utilizes Git as the authoritative source of information, guaranteeing operational efficiency, uniformity, and responsibility. Additionally, it integrates security and compliance into the fundamental aspects of IT operations. GitOps is characterized by its automated, transparent, and collaborative approach, which goes beyond conventional limits. It promotes a culture of shared ownership and continuous progress. In the future, GitOps is expected to undergo additional development, incorporating improvements such as the integration of artificial intelligence for predictive analytics, the implementation of policy-as-code for improved security, and expanded functionalities for managing hybrid and multi-cloud systems. In the ever-evolving world of infrastructure, organizations are turning to GitOps as a guiding principle to achieve agility, security, and resilience. GitOps ensures that configuration management is not only essential but also a strategic tool for fostering innovation and facilitating growth.

REFERENCES

1. Chittibala, D. (2019). GitOps: Revolutionizing Configuration Management in DevOps, International Journal of Science and Research (IJSR), ISSN: 2319-7064.
2. Thompson, G., & Harris, R. (2020). GitOps: The Future of Deployment. In S. Wallace (Ed.), Proceedings of the 2020 International Conference on Cloud Computing (pp.112 - 120). Tech Conferences.
3. Johnson, L., & Davis, T. (2021). The Impact of GitOps on DevOps Practices. Journal of Modern IT Infrastructure, 14 (3), 234 - 245
4. Florian, B. & Simon, H. (2021). GitOps: The Evolution of DevOps IEEE Software.39.10.1109/MS.2021.3119106.
5. Ramadoni, E. Utami and H. A. Fatta. (2021). Analysis on the Use of Declarative and Pull-based Deployment Models on GitOps Using Argo CD, " 2021 4th International Conference on Information and Communications Technology (ICOIACT), Yogyakarta, Indonesia, 2021, pp.186 - 191,
6. Kamath, S., M, M., Vignesh, S., & G, D. (2023). Revolutionizing Cloud Infrastructure Management: Streamlined Provisioning and Monitoring with Automated Tools and User-Friendly Frontend Interface. 2023 3rd International Conference on Intelligent Technologies (CONIT), 1-6.
7. Kormaník, T., & Porubän, J. (2023). Exploring GitOps: An Approach to Cloud Cluster System Deployment. 2023 21st International Conference on Emerging eLearning Technologies and Applications (ICETA), 318-323.
8. Alsurdeh, R., Calheiros, R., Matawie, K., & Javadi, B. (2021). Hybrid Workflow Scheduling on Edge Cloud Computing Systems. IEEE Access, 9, 134783-134799.
9. Leiter, Á., Hegyi, A., Kispál, I., Böösy, P., Galambosi, N., & Tar, G. (2023). GitOps and Kubernetes Operator-based Network Function Configuration. NOMS 2023-2023 IEEE/IFIP Network Operations and Management Symposium, 1-5.
10. Gupta, S., Bhatia, M., Memoria, M., & Manani, P. (2022). Prevalence of GitOps, DevOps in Fast CI/CD Cycles. 2022 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COM-IT-CON), 1, 589-596.
11. López-Viana, R., Díaz, J., & Pérez, J. (2022). Continuous Deployment in IoT Edge Computing: A GitOps implementation. 2022 17th Iberian Conference on Information Systems and Technologies (CISTI), 1-6.

12. Arundel, J. & Domingus, J. (2019). Cloud Native DevOps with Kubernetes: Building, Deploying, and Scaling Modern Applications in the Cloud.
13. Goasguen, S. and Hausenblas, M. (2018). Kubernetes Cookbook: Building Cloud Native Applications. O'Reilly Media, Inc, p. 215, ISBN: 2013436106.
14. Bryant, D. and Marin-Perez, A. (2018). Continuous Delivery in Java: Essential Tools and Best Practices for Deploying Code to Production. O'Reilly Media, Inc, pp. 2, 20, 70–73, ISBN: 978-1-491-98602-8.
15. Cheng, W., Feng, H., & Liang, G. (2022). Design of IT Infrastructure Multicloud Management Platform Based on Hybrid Cloud. Wireless Communications and Mobile Computing.
16. Errea, J., Quang, H., Verchère, D., Thieu, H., Mazzini, A., Abnaou, L., Pesic, J., Curtol, M., Imadi, A., Ksentini, A., & Zeghlache, D. (2023). Open Disaggregated Optical Network Control with Network Management as Code. 2023 Optical Fiber Communications Conference and Exhibition (OFC), 1-3.
17. Limoncelli, T. (2018). GitOps: A Path to More Self-service IT. Queue, 16, 13 - 26.
18. Zhou, J., Wang, T., Cong, P., Lu, P., Wei, T., & Chen, M. (2019). Cost and makespan-aware workflow scheduling in hybrid clouds. J. Syst. Archit., 100.
19. Wettinger, J., Andrikopoulos, V., Leymann, F., & Strauch, S. (2018). Middleware-Oriented Deployment Automation for Cloud Applications. IEEE Transactions on Cloud Computing, 6, 1054-1066.
20. Kratzke, N., & Quint, P. (2017). Understanding cloud-native applications after 10 years of cloud computing - A systematic mapping study. Journal of Systems and Software. 1-16.
21. Koren, I., Rinker, F., Meixner, K., Kröger, M., & Zeng, M. (2023). Implementing DevOps Practices in CPPS Using Microservices and GitOps. 2023 IEEE 28th International Conference on Emerging Technologies and Factory Automation (ETFA), 1-4.