cloud consulting

Betaflux

whitepaper

GITOPS : AUTOMATING YOUR CLOUD INFRASTRUCTURE WITH GIT

march, 2023

> *GitOps is the next step in the evolution of infrastructure automation, bringing version control, collaboration, and developer best practices to the management of infrastructure.*

- Kelsey Hightower

CONTENTS OF THIS PAPER

1. Introduction to GitOps
2. Benefits of GitOps for engineering teams
3. GitOps architecture and tools (with Bonus Case Study)
4. Adopting GitOps for your engineering team
5. Conclusion

1 Introduction to GitOps

GitOps is a modern software development practice that leverages Git as a single source of truth for managing infrastructure and application deployment.

In GitOps, changes to infrastructure and application configurations are version-controlled, auditable, and declarative, allowing engineering teams to apply the time-tested principles of software development to managing cloud infrastructure.

At its core, GitOps is a declarative, continuous delivery model that automates the entire deployment process, from code changes to production deployments. GitOps helps engineering leaders eliminate manual intervention and reduce the likelihood of human error, resulting in faster, more reliable deployments.

THE ROAD TO GITOPS

In the early days of DevOps, engineering teams used tools like Puppet and Chef to manage infrastructure as code. However, these tools were often complex and required significant expertise to use effectively. As containerization and microservices became more popular, teams began to adopt new tools like Kubernetes, which offered greater scalability and flexibility.

At the same time, developers were increasingly using Git as a version control system for managing code changes. This led to the idea of using Git as the single source of truth for infrastructure changes as well. By defining infrastructure as code and storing it in Git, teams could easily collaborate, track changes, and roll back deployments if necessary.

2 Benefits of GitOps for engineering teams

The benefits of GitOps go beyond just faster and more reliable deployments. GitOps also offers improved collaboration and better security for engineering teams and leaders. By using Git to version control all configuration changes, teams can easily track who made changes and when they were made, ensuring that security policies are enforced.

1. **Faster deployments:** GitOps enables faster deployments by using infrastructure as code, continuous integration and deployment pipelines, faster rollbacks, and greater collaboration and transparency. By automating infrastructure changes and tracking them in Git, teams can deploy changes quickly and reliably, catching errors early and avoiding lengthy downtimes.

2. **Increased reliability:** GitOps ensures that all deployments are consistent and repeatable, reducing the likelihood of errors and increasing the reliability of deployments. Since all configurations are version controlled in Git, it's easy to roll back changes if something goes wrong.

3. **Improved Collaboration:** GitOps makes it easy for teams to collaborate on infrastructure and application deployment. By using Git as a single source of truth, all changes are tracked and audited, making it easy for team members to collaborate and review changes.

4. **Better Security:** GitOps can enable better security through version control, code review, automation, and role-based access control. By providing greater visibility, control, and automation over infrastructure changes, GitOps can reduce the risk of security threats and enable quick responses to any incidents that may occur.

3 GitOps architecture and tools (+ CASE STUDY)

A typical GitOps architecture consists of a Git repository, a continuous deployment tool, and an Infrastructure-as-Code (IaC) tool

The GIT Repository

The Git repository serves as the single source of truth for all infrastructure and application configuration management. It contains all the configuration files necessary to deploy and manage the infrastructure resources.
Any changes made to the configuration files are committed to the Git repository and then deployed automatically to the target environment.

Engineering teams could use any GIT tools like GitHub, GitLab, Bitbucket, etc.

Continuous Deployment Tool

The continuous deployment tool is responsible for monitoring the Git repository for changes and applying those changes to the target environment automatically. It ensures that the deployment process is automated and reliable.

One modern tool that can be used for continuous deployment in a GitOps architecture is Argo CD. Argo CD is an open-source continuous delivery tool that provides a web UI and a command-line interface (CLI) to manage and deploy applications. It uses Git as a source of truth for the desired state of the system and automatically deploys changes to the target environment.

Argo CD is designed to work with Kubernetes, a popular container orchestration platform. It provides a declarative approach to deploying applications to Kubernetes clusters, which makes it well-suited for GitOps.

Infrastructure-as-Code Tool

The Infrastructure-as-Code (IaC) tool is responsible for managing the infrastructure resources in a cloud provider like AWS or GCP. It allows teams to define their infrastructure as code, which means that the infrastructure can be version controlled and managed in the same way as application code.

One modern tool that can be used for IaC in a GitOps architecture is Terraform. Terraform is an open-source tool that allows teams to define their infrastructure as code using a declarative language. It supports a wide range of cloud providers, including AWS, GCP, and Azure.

Terraform uses the concept of "resources" to represent infrastructure components like EC2 instances, load balancers, and security groups. These resources are defined in Terraform configuration files and then deployed to the target environment automatically.

BETAFLUX CASE STUDY: GITOPS <> KUBERNETES

Problem Statement

Managing a Kubernetes cluster was turning out to be complex and time-consuming for one of our Fintech clients — requiring manual configuration changes and frequent updates to infrastructure and application code.

Without proper version control and automation, it was difficult to ensure that changes were made consistently and with a high level of quality, with an increased risk of errors and security vulnerabilities. Additionally, without proper access control, it was difficult for the team to manage who can make changes to the infrastructure, adding to the security risks.

Enter GitOps

GitOps now allows this engineering team to easily manage and deploy changes to a Kubernetes cluster using code and automation. By leveraging Git as a version control system and automating the deployment process, they can ensure that changes are made quickly and effectively, while also maintaining security and compliance requirements.

Infrastructure as Code: The infrastructure for the Kubernetes cluster is defined in code using a tool like Terraform or CloudFormation. The code is stored in a Git repository that serves as the single source of truth for the infrastructure.

CI & CD: Changes to the infrastructure code are automatically built, tested, and deployed using a CI/CD pipeline. The pipeline is triggered whenever a

change is pushed to the Git repository, ensuring that changes are deployed quickly and with a high level of quality.

Kubernetes Configuration Management: The Kubernetes configuration is defined in code using YAML files, which are stored in the same Git repository as the infrastructure code. The configuration is automatically deployed to the Kubernetes cluster using a GitOps tool like Flux or Argo CD.

Version Control: All changes to the infrastructure and configuration code are tracked in Git, providing a complete history of changes over time. This allows teams to easily identify and respond to security threats and other issues.

Role-based Access Control (RBAC): Access to the Git repository and Kubernetes cluster is controlled using RBAC, ensuring that only authorised users can make changes.

CONCLUSION

By using GitOs to manage their Kubernetes cluster, the team was able to streamline their deployment process and reduce the time and effort required to make changes to their infrastructure and application code.

They were able to achieve greater consistency and reliability in their deployments, reducing the risk of errors and security vulnerabilities.

With the ability to track all changes in Git and use automation to deploy changes quickly and consistently, the team was able to respond quickly to new requirements and ensure that their infrastructure was always up-to-date and secure.

Additionally, by using RBAC to control access to the Git repository and Kubernetes cluster, the team was able to maintain a high level of security and ensure that only authorised users could make changes.

Overall, the use of GitOps allowed the team to improve the quality, security and reliability of their Kubernetes Custer, while also increasing their efficiency and agility.

4 Adopting GitOps for your engineering team

Adopting a new process like GitOps can be a significant shift for teams that are used to making small, manual changes to infrastructure. Here are some steps that can help ease the transition —

1. **Start small:** Begin by identifying a small project or service to pilot the GitOps process. This will help the team to become familiar with the workflow and identify any issues or challenges that need to be addressed.

2. **Define all infrastructure as config files:** Ensure all the infrastructure you want to manage via GitOps for this project is described in IaC config files. Ideally, these files should be written in declarative code. This means you describe the end state of what you want rather than instructions on how to get there.
   If you're already using IaC and you want to automate it with GitOps, start by adding your infrastructure code to the Git repository you plan to use for GitOps.

3. **Document what you can't automate:** In case you have some legacy environments that need manual attention. Document these instances so that they're accounted for.

4. **Outline a code review and merge request process:** It's important to familiarise GitOps teams with Git and code reviews. Some teams already use a Git repository as a place to store config code, but don't use features like merge requests. For teams new to GitOps, another option is to set up "optional reviews" rather than set up "required blocking reviews".

5. **Consider multiple environments:** It's good practice to have multiple environments. One example you might follow is the DTAP environments: Development, Test, Acceptance, and Production. Code can be rolled out to the Development or Test environment, after which you can test

whether the services are still available and working as expected.
If they are, you can further roll out your changes to the next environments

After you have rolled out your code into your environments, it's important to keep your code in sync with your running services. Once you know there's a difference between your system and your configuration, you can fix either one. A solution to this problem is to use immutable images, such as containers so that it's less likely to have differences.

6. **Make CI/CD the access point to resources:** One practice that encourages a GitOps workflow, and reduces manual changes to cloud infrastructure, is to make your CI/CD tooling the access point for cloud resources. Of course, having this access during initial development can help teams write their code and you may need incidental access for various reasons. However, switching your mindset from "access unless" to "access because" can help in adopting and following the GitOps process.

5 CONCLUSION

In summary, GitOps is a powerful methodology for managing and deploying cloud infrastructure and applications efficiently and securely. However, it can be a complex process that requires expertise and resources.

Betaflux can guide you through the process of implementing GitOps to help you achieve faster delivery times, improved collaboration, reduced risks, and increased compliance in your cloud environment.

If you're interested in exploring how our cloud consulting services can help you leverage GitOps and accelerate your cloud transformation journey, please reach out to us via the link to schedule a free discovery call to understand how we can help.

ABOUT BETAFLUX

Betaflux is a technology consulting company specialising in cloud infrastructure consulting, cloud security & compliance consulting, SRE consulting, software development, UX research and UI/UX Design.

Since 2018, we been working with growth stage companies and enterprises across the Fintech, Automotive, F&B and logistics industries to help them architect scalable and secure software.

www.betaflux.co

Betaflux Consulting Private Limited
4th Floor, WeWork Prestige Central,
Infantry Road, Tasker Town,
Bangalore - 560001, KA, India.

sales@betaflux.co

Betaflux