# Understanding GitOps

EBOOK

GITOPS

# Table of Contents

# Introduction

GitOps is an operational framework that aims to streamline and automate the deployment and management of applications and infrastructure using Git as the single source of truth. GitOps is popular in the cloud-native ecosystem, particularly in Kubernetes-based environments, where managing Infrastructure as Code (IaC) is crucial.

In this ebook, you will learn:

- what GitOps is and the core principles that support it;
- why it's compatible with DevOps;
- how it automates pipelines with IaC and Continuous Integration and Delivery (CI/CD); and
- what a GitOps workflow looks like.

# What is GitOps?

GitOps is an operational framework that aims to streamline and automate the deployment and management of applications and infrastructure using Git as the single source of truth. The term “GitOps” combines the version control system Git with operations (Ops), emphasizing using Git principles and workflows for managing infrastructure and applications.

In GitOps, the system’s desired state (including the application code, configuration, and infrastructure) is defined and stored in a Git repository. This repository serves as the central source of truth for the entire system. Any changes to the system are made by modifying the Git repository, which triggers an automated process to reconcile the current state with the desired one. GitOps is possible due to IaC tools that allow you to create and manage your infrastructure using declarative configuration files.

GitOps is a paradigm that empowers developers to manage tasks that operations might otherwise handle. Traditionally, a company’s IT department oversees operations processes and services, including:

- technology and infrastructure management (including software);
- quality assurance;
- network administration; and
- device management.

In that model, developers are siloed from these activities. GitOps aims to **remove** those silos, and enable operations to employ the same tools and methodologies developers use for efficient collaboration.

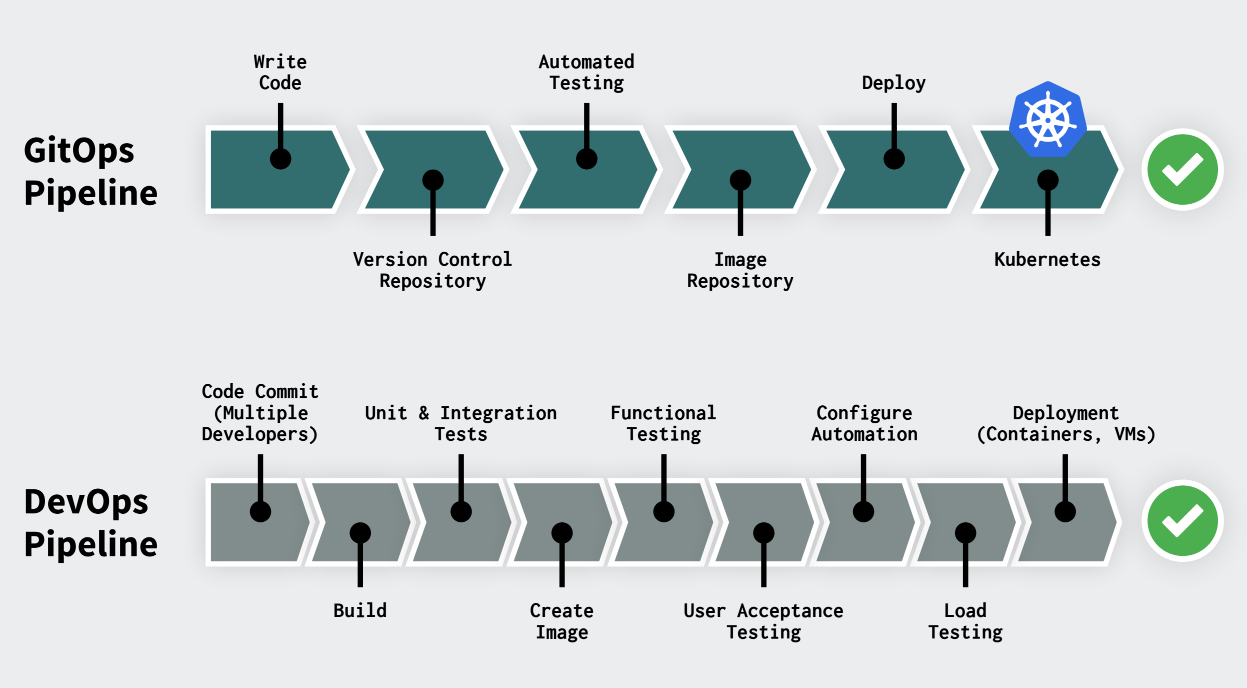
# GitOps and DevOps

To those familiar with DevOps methodologies, GitOps may sound familiar. While they share similarities, they focus on different aspects of the software development lifecycle.

DevOps is an approach that combines development and operations practices to foster collaboration and efficiency throughout the software development lifecycle. It emphasizes automation, continuous integration and continuous delivery (CI/CD), and cross-functional teamwork. DevOps aims to break down silos between development, operations, and other teams to enable faster and more reliable software delivery.

GitOps is a specific implementation of DevOps that emphasizes using Git as the single source of truth for declarative infrastructure and application code. As we learned, it focuses on managing the entire software delivery process using Git repositories. In GitOps, the system's desired state is versioned and stored in a Git repository, and a reconciliation loop continuously monitors and enforces that desired state.

> ***DevOps assists in the automation of the software development lifecycle, while GitOps contributes to the automation of infrastructure.***



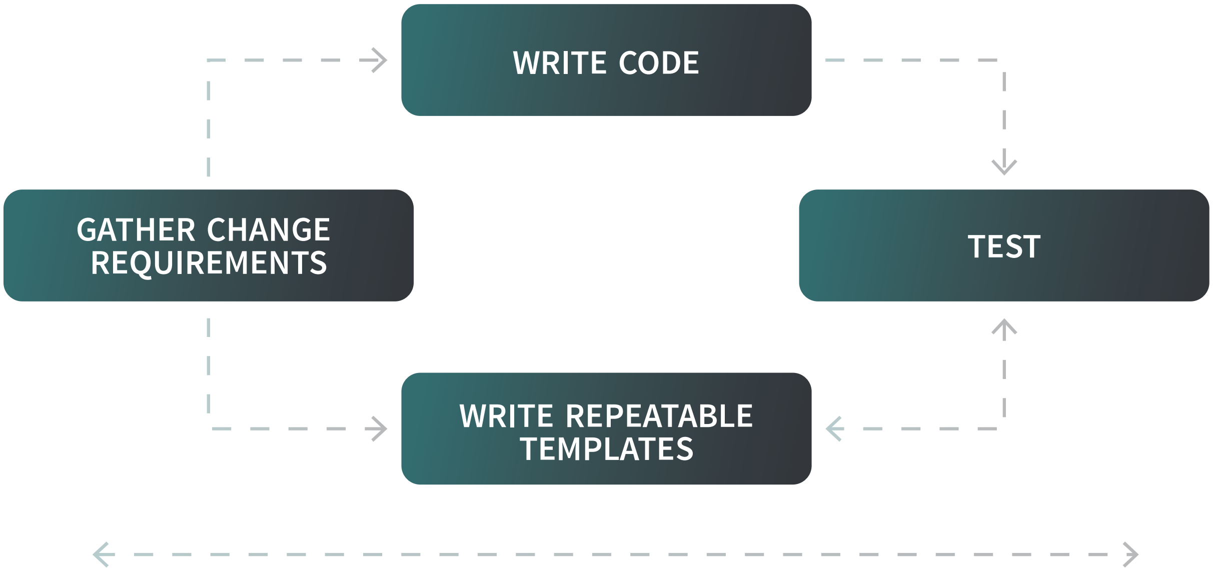
*GitOps borrows best practices from DevOps and applies them to infrastructure automation, including version control, collaboration, compliance, and CI/CD.*

Tools like Kubernetes support software development lifecycle automation. Because many businesses containerize deployments to scale applications and services, they often depend on third-party, cloud-based services to host their infrastructure. This dependence has led to the rise of infrastructure automation to achieve a level of elasticity not practical with traditional, or on-premise, infrastructure.

# Automating Pipelines with IaC and CI/CD

As you learn more about GitOps, it is also important to better understand two core DevOps practices: Infrastructure as Code (IaC) and CI/CD.

**IaC** is a technique for deploying and managing infrastructure through portable configuration files instead of a manual or interactive process. This approach is not limited to just infrastructure primitives or managed cloud services–it is applicable to all manageable aspects, such as resource deployment, software installations, and network and security policies, among others. This type of unified management is commonly referred to as “X as Code,” which provides a templated and documented desired state for your deployment. It allows infrastructure provisioning and management to be treated as code, thereby enabling automation, version control, and reproducibility. In GitOps, the system’s desired state is versioned and stored in a Git repository, and a reconciliation loop continuously monitors and enforces that desired state.

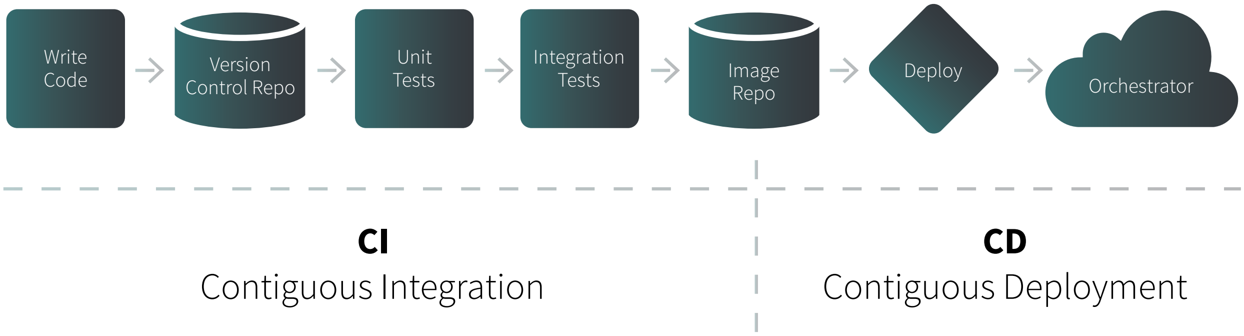


**Iterate. Test. Repeat.**

*At a high level, GitOps enables iterative development.*

**CI/CD** is a methodology and set of best practices allowing developers to rapidly and reliably deliver quality and securely coded applications. Continuous Integration (CI) is the process of automatically validating the changes by building, testing, and packaging the updated application code reliably and consistently. Triggering a CI workflow on every push event provides a smoother, faster development process. Developers can discover bugs, security vulnerabilities, and conflicts before changes are deployed to an environment.

A Continuous Delivery (**CD**) workflow takes over after successfully completing the CI workflow. This is the automated and consistent process of delivering the updated application code to a selected environment. This workflow can facilitate deploying the application directly onto staging or production servers, or shipping a new release to a container registry or mobile distribution platform.



A GitOps approach puts these core DevOps practices of IaC and CI/CD into operation. GitOps utilizes IaC to define infrastructure as code and CI/CD practices to automate the deployment process. This results in more efficient, reliable, and scalable infrastructure and application management.

# GitOps Principles and Workflow

While DevOps is a broader philosophy that encompasses the entire software development lifecycle and promotes collaboration and automation, GitOps is a specific implementation of DevOps that relies on Git as the single source of truth and focuses on managing the system's desired state using declarative infrastructure and continuous deployment. GitOps can be seen as a specialization or extension of DevOps.

Here are the five core GitOps principles:

### Declarative Infrastructure

The desired state of the infrastructure and applications is declared in a Git repository using declarative configuration files. These files describe the desired state of the system rather than specifying the steps to reach that state.

### Version Control

Git is used as the version control system, providing a history of changes made to the system over time. This enables rollbacks, auditing, and collaboration among team members.

### Continuous Deployment and Automation

GitOps promotes automation by using CI/CD pipelines to automatically apply changes to the system whenever there is a change in the Git repository. This process eliminates manual intervention and ensures consistency between the desired and actual system states.

### Reconciliation Loop

A GitOps tool or operator continuously monitors the Git repository for changes. When a change is detected, the tool reconciles the system's current state with the desired state defined in the repository. It applies any necessary updates or rollbacks to achieve the desired state.

### Observability and Auditing

GitOps emphasizes observability by providing real-time visibility into the system's state and changes. This allows teams to monitor and troubleshoot issues more effectively. Additionally, since all changes are recorded in Git, it is easy to meet auditing and compliance requirements.

Within the GitOps workflow, there is a high level of automation, and a much higher probability that your deployment will work exactly as expected.

In addition to deployment efficiency, using Git as a single source of truth allows you to rely on in-depth version control in case you need to roll back changes. Historical information is readily available to see what change caused a bug or other problems. This ultimately improves your reliability posturing and provides additional security guardrails.

# Using GitOps with Akamai Cloud Computing Services

Akamai cloud computing services make it easy to get started with IaC tools that are integral to GitOps. Get a jump start with our Try IaC ebook and on-demand course by instructor Justin Mitchel of Coding for Entrepreneurs.

Here are other resources to help you get started with Git to create your own GitOps pipeline.

## Resources

- Set up Git (GitHub documentation)
- Install GitLab with Docker
- Resolve Merge Conflicts in Git
- View all Git guides in Akamai's documentation library

# Conclusion

You can see that while GitOps and DevOps share many of the same core components, GitOps focuses more on tools and frameworks to enable the culture changes of DevOps. With GitOps, you'll accelerate DevOps practices and apply them to automation and deployment cycles, enabling developers to work in code repositories they're familiar with while empowering operations to build automated and efficient environments.

# About Akamai Cloud Computing

Akamai accelerates innovation with scalable, simple, affordable, and accessible Linux cloud solutions and services. Our products, services, and people give developers and enterprises the flexibility, support, and trust they need to build, deploy, secure, and scale applications more easily and cost-effectively from cloud to edge on the world's most distributed network.

www.akamai.com
www.linode.com

# Cloud Computing Services Developers Trust

linode.com | Support: 855-4-LINODE | Sales: 844-869-6072
249 Arch St., Philadelphia, PA 19106 Philadelphia, PA 19106