Akamai

GitOps for Cloud Portability

Akamai

EBOOK

GITOPS

Table of Contents

Introduction

Cloud portability refers to the capability to migrate applications, data, and other key resources seamlessly between different environments or cloud providers. It enables organizations to avoid vendor lock-in and retain flexibility in their cloud strategies. This type of flexibility is crucial for organizations looking to maintain agility in their architecture. The recommended strategy for enabling portable application development by implementing GitOps, or DevOps best practices with Git as the source of truth for managing infrastructure and application deployment.

In this ebook you will learn:

- the differences between cloud-native and platform-native,
- how cloud-native development and cloud portability are linked,
- how cloud portability helps get around vendor lock-in,
- the five characteristics of cloud portability, and
- how leveraging GitOps increases cloud portability.

Cloud-Native vs. Platform-Native

Cloud-native is a widely known, modern approach to software development and deployment that leverages the capabilities and benefits of cloud computing with an emphasis on staying provider-agnostic. Cloud-native takes full advantage of the elasticity and agility of cloud computing resources.

Platform-native is a lesser-used term that describes application architecture that is heavily tied to a single infrastructure provider or deployment platform. This type of architecture is common for organizations that depend on multiple services from AWS or Azure, which typically manifests in an application architecture built on a unique design feature or functionality within that service.

When you build within a specific platform ecosystem and cannot move to another without significantly changing the services you rely on, you experience vendor lock-in. Your workload lacks portability and if, for whatever reason, you needed to move to another cloud provider in order to scale or replace your provider altogether, you would need time and resources to re-architect components of your application.

However, when you begin with a cloud-agnostic and cloud-native approach to leverage tools that can be used with any cloud provider, you have the flexibility to make adjustments as your needs change.

A **cloud portability** strategy also gives you more insight into how, where, and why your application consumes cloud resources. So, instead of working around the limitations of a single cloud provider, you have the power to make decisions based solely on your business and user needs.

GitOps and Cloud-Native Development

GitOps, an operational framework that simplifies the deployment and management of applications and infrastructure using Git, closely aligns with the principles of cloud-native development and enhances cloud portability.

5 Characteristics of a Cloud Portability Strategy

Cloud portability is a strategy for building scalable, resilient cloud-native applications. A portable workload is one that can be easily migrated, deployed, and managed across different computing environments and infrastructure platforms. So how do you know that you're building for portability?

Here are five key characteristics of a cloud portability strategy to consider:

1. Commoditized architecture
A portable workload is designed to operate on core cloud infrastructure primitives. These are the basic, interchangeable services found across all Infrastructure-as-a-Service (IaaS) providers.

Examples of core primitives include things like compute, storage, networking, and declarative APIs. Managed services like Database-as-a-Service (DBaaS) and Kubernetes engines such as AWS EKS and Akamai's LKE can also be considered core cloud infrastructure primitives because they are common across IaaS providers.

It's important to choose an open source version that is compatible with similar managed services on other platforms as well as with self-hosted solutions.

2. Open source software
Portable applications bypass vendor lock-in by avoiding design patterns that depend on proprietary services of a cloud provider. Your entire software stack–everything from front-end frameworks, to back-end APIs, to the database layer, and message brokers–should be built with open source tooling that is capable of running in environments built from core cloud infrastructure primitives.

3. Scalability
Fully portable workloads are capable of horizontal scaling. This can be by metric-based autoscaling functionality, or by predictive (manual) scaling based on resource availability and user demand.

A scalable design with a cloud-agnostic architecture helps to ensure adaptability and flexibility across different environments, allowing your architecture to span multiple cloud platforms.

4. Standardization
A portable workload adheres to a standardized design to simplify testing and deployment. This is achieved by ensuring consistency and repeatability in every aspect.

The design pattern of infrastructure resources, software configurations, and security policies are all templated and thoroughly documented. When deploying multiple times from the same blueprint, processes and tools are in place to ensure idempotency, safeguarding against configuration drift, and ensuring uniformity across deployments.

5. High availability and disaster recovery
A portable architecture is built for resilience by incorporating a design that supports replication and failover strategies to eliminate single points of failure. This design pattern, along with thorough documentation and standardization, results in a lower recovery point objective (RPO) and recovery time objective (RTO) for disaster recovery.

The lower the RPO and RTO, the quicker an organization can recover from a disaster with minimal data loss and downtime, so prioritizing high availability and disaster recovery is necessary for continuous operation and minimal impact in case of failure.

Understanding the Challenges of Cloud-Native Architecture

Adopting a cloud-native architecture is not without its challenges. While the benefits are substantial, familiarizing yourself with the initial challenges is crucial for a successful transition.

The first hurdle is modifying your application to fit the cloud-native model. This process involves re-architecting applications fit into a standardized and vendor-agnostic workflow. It often requires breaking down monolithic applications into microservices, which can be a complex and resource-intensive task - but one that will pay off by providing flexibility in the future.

Another significant challenge is dealing with the complexities of multicloud environments. Each cloud service has different APIs, syntax, and other features, and each comes with its own unique set of “gotchas.” This complexity is compounded by the need to ensure consistency in deployment, security, and operations across various platforms.

Transitioning to a cloud-native architecture also brings about an increase in maintenance tasks. The dynamic and distributed nature of cloud-native applications requires continuous monitoring, updates, and security checks to ensure optimal performance and security.

Handling the Challenges with Cloud-Native Strategies

Transitioning to a cloud-native architecture brings its challenges, but there are effective tools and strategies that facilitate migration, automation, and iterative development. These strategies lie in the innate strengths of cloud-native systems.

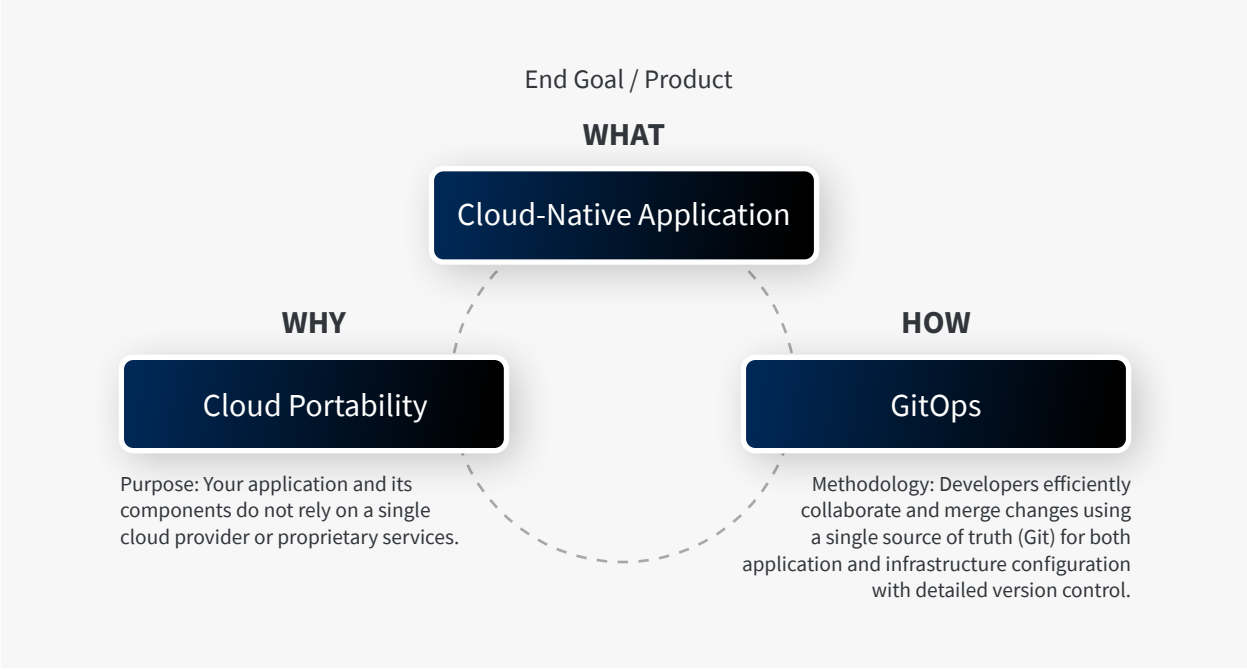
A pivotal advantage of cloud-native architecture is its inherent support for automation. Tools designed for continuous integration and continuous deployment (CI/CD) enable organizations to automate crucial aspects of application deployment, scaling, and management. This not only reduces the need for manual intervention but also significantly lowers the likelihood of errors, streamlining the operational workflow.

The adoption of a microservices approach is another critical aspect of cloud-native applications. By fracturing applications into smaller, independent units, organizations can facilitate more manageable updates and maintenance. This approach allows teams to modify or repair specific parts of the application with minimal impact on the overall system. Such modular architecture enhances scaling efficiency and optimizes resource utilization.

Implementing GitOps practices brings a new level of efficiency to the development and operational processes. This approach emphasizes streamlined collaboration, advanced automation, and a commitment to continuous improvement, making the overall development lifecycle more responsive and agile.

In embracing these cloud-native strategies, organizations can effectively navigate the complexities of transitioning to a cloud-native architecture, setting the stage for enhanced scalability, resilience, and operational efficiency in the long run.

How Leveraging GitOps Increases Cloud Portability



Cloud-native development, cloud portability, and GitOps are strongly linked– in both end goals and principles and methodologies to reach those goals. Cloud portability enables developers to rapidly move applications and data between cloud environments (and providers). Cloud-native is an architectural approach to designing applications that are built in the cloud, and intend to remain as such to reap cloud computing benefits. In all of this, GitOps is the “how”.

The benefits of GitOps impact key portability goals ranging from our failover example to freeing resources from extensive testing and auditing in order to ship new features and improvements.

BENEFIT	HOW	IMPACT ON PORTABILITY
Consistency	GitOps leverages CI/CD pipelines to apply changes whenever a change occurs in the Git repository.	The live state matches the desired state so an environment can be replicated on another cloud provider.
Efficiency	GitOps enables iterative development to support rapid releases and fixes.	Reduce time-to-market for new features and improvements.
Collaboration	Maintaining version control with multiple teams and contributors is the foundation of Git and GitOps.	Developers experience fewer hurdles to contribute their code while maintaining a single source of truth.
Reliability	Version control makes it easy to roll back changes and find changes that resulted in a bug or performance issue.	Decision makers can rapidly assess whether unexpected application behavior is due to a recent change or other potential cause.
Security	Git repositories are equipped with role-based access control, secrets management, and custom security policy codification.	Security features are at the foundation of the application infrastructure– not defined by cloud platform features or external tools.
Compliance	All changes are recorded in Git, making audits and compliance checks more efficient.	Traceability streamlines moving to new environments, improving development flexibility.
Multi-environment Automation	Different environments (i.e. production, staging, dev) are clearly defined. The transition of changes to the production environment is meticulously controlled through branch merges or updates to configuration files.	Any modifications or updates undergo testing in staging or development environments before they are applied to the production environment, maintaining the integrity and stability of the live application.

Akamai and Cloud Portability

Akamai is at the forefront of empowering developers in their cloud-native development and ensuring that workloads are portable and easy to manage. This includes making it easy to get started with IaC tools and our extensive documentation library to set up portable infrastructure.

Here are some resources to help you get started.

- Understanding GitOps (Ebook)
- GitOps: An Overview of Its Principles and Workflow (Guide)
- Resolve Merge Conflicts in Git (Guide)
- Cloud Portability: Building a Cloud-Native Architecture (Ebook)
- View all Git guides in Akamai's documentation library

About Akamai

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