GitOps and KRO: A New Way to Control Cloud Infra

GitOpsCon NA

Dec 4, 2025

KORAY

- Consultant and Trainer @ Kubermatic
- Working remotely from Istanbul
- CNCF Ambassador and Kubestronaut
- Kubernetes Contributor
- KCD and DevOpsDays Istanbul Organizer

CANSU

- 👩‍💻 Principal Platform Architect @ Red Hat
- 🌭 Based in Germany
- 🐙 Platform Engineering Ambassador
- 🏅 Team Topologies Advocate
- 🔥 Huge believer in Open Source

GitOps Con
NORTH AMERICA

01

What is KRO?

The Tooling Gap: Common Pain Points

Helm Template Sprawl

Complex charts, client-side rendering, and post-upgrade drift create management overhead.

Crossplane Complexity

Multiple CRDs and external provider pods add latency and operational surface area.

Terraform State Drift

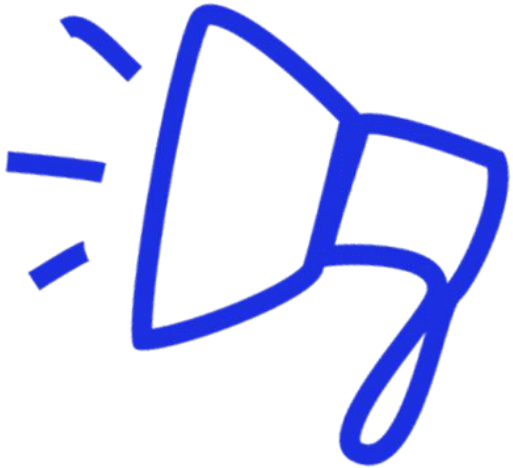
State file management and manual reconciliation disrupt the GitOps flow.

Platform engineers need a lighter way to bundle and govern resources that fits native GitOps workflows.

Meet KRO!

https://kro.run/

KRO is a subproject of Kubernetes SIG Cloud Provider. Runs in-cluster as a native controller. Works with any CRD regardless of API group. Continuous reconciliation with automatic drift detection.



Warning!

The project is still in Alpha status. It is not Production Ready!!!

Kube Resource Orchestrator

Kube Resource Orchestrator (KRO) is a native operator that simplifies complex resource management. It wraps multiple Kubernetes manifests into a single Custom Resource Definition (CRD) using **ResourceGraphDefinition**.

KRO automatically generates dedicated controllers for each CRD, ensuring continuous reconciliation of resources. This eliminates the need for manual intervention and reduces configuration drift.

KRO provides a simplified developer experience by abstracting away the complexity of YAML manifests. Developers can deploy applications using a single CRD, without needing deep Kubernetes expertise.

Core Concepts & Flow

1. Define RGD

Declare resources, defaults, and schema.

2. Compile CRD

KRO generates the custom resource definition.

3. Reconcile

The controller manages instances, ensuring drift correction.

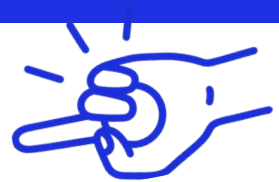
This process provides native validation, events, and owner references, hiding YAML complexity from end-users.

GitOps Con
NORTH AMERICA

02

KRO vs Helm

Templating Model Compared



KRO Templating

KRO employs server-side reconciliation with RGD, ensuring continuous resource management and reducing the risk of configuration drift.

Ecosystem

Helm has a vast ecosystem with thousands of charts, making it ideal for one-time deployments. KRO, though newer, focuses on continuous resource management and integration with GitOps.

01

Helm Templating

Helm uses client-side templating with Go templates, allowing flexible resource generation but often leading to complex and error-prone configurations.

02

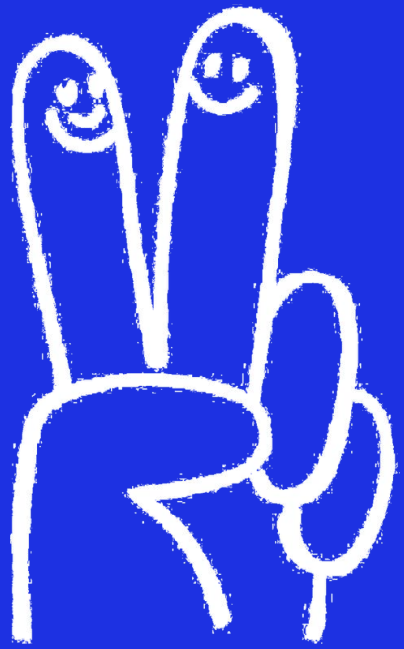
03

Validation

Helm validates templates during rendering, while KRO validates at admission, providing real-time feedback and ensuring compliance with defined policies.

04

Pros & Cons



KRO offers native drift detection, simpler RBAC, and seamless GitOps integration. However, it lacks Helm's extensive chart ecosystem and has a steeper learning curve for CRD authors.

GitOps Con
NORTH AMERICA

03

KRO vs Crossplane

Composition Head-to-Head

01

Crossplane Composition

Crossplane uses CompositeResourceDefinitions and Compositions to create higher-level APIs, requiring multiple CRDs and external provider pods for resource management.

KRO Composition

02

KRO simplifies this process with ResourceGraphDefinition, which directly translates to a single CRD and dedicated controller, eliminating the need for external API translation.

03

Operational Complexity

KRO reduces operational complexity by minimizing the number of components and avoiding external state stores, making it easier to manage and scale.

When to Prefer Which

Choose KRO

Select KRO when you need to compose existing Kubernetes resources quickly and efficiently, leveraging its native Kubernetes integration and GitOps compatibility.

Choose Crossplane

Opt for Crossplane when managing multi-cloud resources or requiring advanced policy enforcement and a mature package management system.

GitOps Con
NORTH AMERICA

04

KRO Meets GitOps

Declarative Story Fit

KRO & GitOps

KRO seamlessly integrates with GitOps tools like Flux and Argo CD, allowing resource definitions to be stored and managed in Git repositories.

Simplified Workflows

Developers can focus on defining resources in Git, while KRO handles the deployment and management, streamlining the GitOps workflow.

Continuous Reconciliation

KRO's server-side controllers continuously reconcile resources, ensuring they match the desired state defined in Git, providing real-time drift detection.

Traceability

Git history provides a clear audit trail of all changes to resource definitions, making it easier to track and revert changes if needed.

Reference Workflow

Workflow Overview

Platform teams define **ResourceGraphDefinitions** in Git, which are synced to the cluster by GitOps tools. KRO generates CRDs and manages instances, ensuring resources are deployed and maintained as specified.

GitOpsCon NORTH AMERICA

05

Live Demo

koksay/gitopscon-kro-demo

kubernetes-sigs/kro/tree/main/examples

KRO Advantages

KRO offers a Kubernetes-native solution for resource composition, simplifying complex deployments and integrating seamlessly with GitOps workflows.

Why Consider KRO?

Lightweight & Native

Delivers Kubernetes-native composition without extra control planes or state files.

Bridges the Gap

Closes the gap between Helm's simplicity and Crossplane's power.

GitOps Native

Slots naturally into existing GitOps workflows with full auditability.

THANK YOU

cansu@redhat.com

linkedin.com/in/ckavili

koray@kubermatic.com

linkedin.com/in/korayoksay