+33 (0)1 48 10 09 46
formation@ambient-it.net 140 Avenue Jean Lolive 93500 Pantin - France

Updated on 08/21/2026

Sign Up

Kro Training: Kube Resource Orchestrator

3 days (21 hours)

Overview

kro is a native Kubernetes orchestrator that transforms sets of resources into reusable, declarative, and consistent Kubernetes APIs. Using ResourceGraphDefinitions and CEL expressions, kro simplifies the creation of standardized platform components.

Our Kro: Kube Resource Orchestrator training will enable you to design abstractions tailored to the needs of development and operations teams. You'll learn how to assemble native or custom Kubernetes resources, structure their dependencies, and expose easy-to-use interfaces.

You'll master defining a ResourceGraphDefinition, designing its schema, generating CRDs, and using resource templates. CEL expressions will allow you to pass parameters, leverage statuses, and set up deployment conditions.

By the end of the course, you'll be able to create reliable platform building blocks, reduce manifest duplication, and deliver consistent deployments. You'll also learn how to diagnose reconciliation issues, control access, and evolve your APIs without disrupting their consumers.

Finally, this training covers the integration of kro into a GitOps workflow, controller observability, and best practices for versioning. This will enable you to industrialize the composition of application and infrastructure resources in your Kubernetes clusters.

Like all our training courses, this one will introduce you to **the latest stable version** of the technology and its new features.

Objectives

- Understand the architecture and operation of kro in a Kubernetes cluster.
- Design ResourceGraphDefinitions to create reusable platform APIs.
- Define schemas, composite resources, and dependencies between objects.
- Use CEL expressions to configure, condition, and enrich compositions.
- Diagnose reconciliation, secure access, and version kro abstractions.

Target Audience

- Kubernetes Administrator
- DevOps Engineer
- Platform Engineer
- Cloud Engineer
- Backend Developer

Prerequisites

- Proficiency with YAML manifests and fundamental Kubernetes objects.
- Experience using kubectl to deploy and troubleshoot resources.
- Knowledge of Kubernetes CRDs, controllers, and the reconciliation mechanism.
- Basic understanding of declarative deployment with Git and GitOps.

Technical Requirements

- Computer running Linux, macOS, or Windows with at least 8 GB of RAM.
- Access to a Kubernetes lab cluster or a local installation using kind or minikube.
- kubectl version 1.28 or later and Helm version 3 installed and configured.
- A code editor such as Visual Studio Code with YAML support.
- A stable internet connection that allows for the download of container images and Helm packages.

Our Kro Training Course: Kube Resource Orchestrator

[Day 1 - Morning]

Kro Fundamentals and Architecture

- Kro’s role in the Kubernetes ecosystem and use cases for platform teams.
- Principles of declarative APIs, CRDs, and continuous reconciliation.
- Role of the ResourceGraphDefinition, the KRO controller, and the generated controllers.
- The lifecycle of an abstraction, from defining an RGD to creating its instances.
- Installing kro, verifying components, and discovering available resources in the cluster.

[Day 1 - Afternoon]

Creating Your First Platform API

- Structure of a ResourceGraphDefinition and naming conventions for resource identifiers.
- Designing the SimpleSchema, field typing, default values, and required fields.
- Dynamically generating a CRD and making the resulting API available to developers.
- Composing a Deployment, a Service, and a ConfigMap into a single abstraction.
- Hands-on workshop: Create and deploy an RGD WebApp that generates a configurable Deployment, Service, and ConfigMap.

[Day 2 - Morning]

Orchestration and CEL Expressions

- Syntax and principles of CEL expressions in kro resource models.
- Referencing schema data, metadata, and attributes generated by resources in the graph.
- Automatic inference of the dependency graph and the order in which resources are created.
- Value propagation between resources, utilization of status fields, and exposure of useful outputs.
- Validity checks, static analysis, and identification of reference errors prior to deployment.

[Day 2 - Afternoon]

Conditional and observable compositions

- Implementing conditional logic with `includeWhen` to adapt a composition as needed.
- Defining readiness criteria to manage the availability status of managed resources.
- Creating custom status fields to provide a usable API for consumers.
- Using native resources and third-party CRDs in a single composition.
- Hands-on workshop: Enhancing the WebApp API with a conditional Ingress, status outputs, and availability rules.

[Day 3 - Morning]

Industrializing kro abstractions

- Designing reusable, scalable abstractions that adhere to platform standards.

- Collection management to dynamically generate multiple resources from a single model.
- External references, composition with installed operators, and integration of cloud resources.
- Chaining ResourceGraphDefinitions to structure platform building blocks at different levels.
- Integration into GitOps workflows with code review, manifest management, and
  .

[Day 3 - Afternoon]

Securing, diagnosing, and scaling

- Permission control using RBAC and the principle of least privilege for managed resources.
- Monitoring events, conditions, logs, and metrics from kro controllers.
- Diagnosing schema errors, cyclic dependencies, reconciliation failures, and deletions.
- API versioning strategies, schema compatibility, and management of incompatible changes
  .
- Hands-on workshop: Design a comprehensive platform API, secure it, diagnose a reconciliation error, and prepare for its evolution.

Target Audience

This training is intended for both individuals and companies—large or small—that wish to train their teams in a new advanced IT technology or to acquire specific business knowledge or modern methods.

Entry-Level Assessment

The pre-training assessment complies with Qualiopi quality standards. Upon final registration, the learner receives a self-assessment questionnaire that allows us to evaluate their estimated proficiency level across various types of technologies, as well as their expectations and personal goals for the upcoming training, within the limits imposed by the selected format. This questionnaire also allows us to anticipate certain connection or internal security issues within the company (intra-company or virtual classroom) that could pose challenges for monitoring and ensuring the smooth operation of the training session.

Teaching Methods

Hands-On Training: 60% hands-on, 40% theory. Training materials distributed in digital format to all participants.

Organization

The course alternates between theoretical input from the instructor—supported by examples and reflection sessions—and group work.

Assessment

At the end of the session, a multiple-choice quiz is administered to verify that participants have successfully acquired the skills.

Certification

A certificate will be issued to each trainee who has completed the entire training program.