WSO2CON AFRICA

SEPT 22 – 24, 2026 | NAIROBI, KENYA

One Way to Ship

Build your internal developer platform with OpenChoreo

Kavith Lokuhewage
Associate Director / Architect

Isuru Wimalasundera
Associate Enterprise Architect

Resilient is solved. Shipping is not.

What every organisation here is already building

Resilient systems on Kubernetes — self-healing workloads, horizontal scale, rolling upgrades, failover across zones. That is a well-understood problem with a well-understood answer, and most of you have already bought it.

What still has to be decided

How that system actually gets shipped — the whole path from a commit to production, and everything it has to touch on the way. None of it is a runtime question, so none of it comes in the box.

And in a regulated institution, the governance around it

Who approved the change, what was actually running on the day of the audit, and whether a control is enforced or merely described. A resilient cluster answers none of those.

Everyone ends up with a way to ship. This session is about choosing it on purpose.

The decisions Kubernetes leaves to you

A declarative API, a scheduler, a reconciliation loop — all of that works. What it does not ship is a path from a commit to production.

Getting it built

Repository layout. The build. Container image, registry, signing, scanning. Which CI system — and who fixes it at two in the morning.

Getting it deployed

Manifest templating and its values. Environments, and what may differ between them. Promotion and rollback. Secrets, routes, network policy.

Keeping it alive

Logs, metrics, traces and alerts. Autoscaling and limits. Access control and an audit trail. Cost attribution. And the docs for all of it.

All of it gets built. The only question is whether it gets built once, or once per team.

Why you have to build a platform

You already have one

Every organisation running Kubernetes in production already has a path from commit to release. It was assembled from scripts, conventions and one person's memory — but it exists.

Nobody decided it

It was never designed, reviewed or owned. It accumulated one reasonable decision at a time, under deadline — which is why it is inconsistent, and why nobody can describe it.

The only real choice

Not whether to have a platform, but whether the one you have is designed on purpose and owned by someone. That decision is the whole of it.

So what do you build it with?

Every box is defensible. Knowing which ones you need takes requirements you do not have yet.

THE DEMO USE CASE

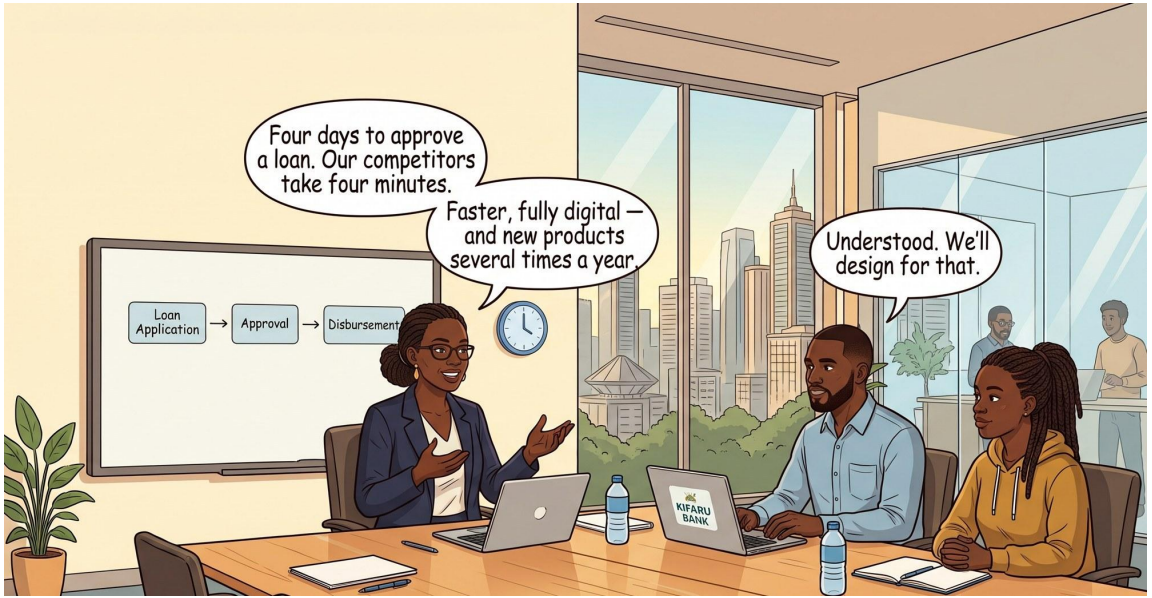
Kifaru Bank

WSO2

The mandate

Njeri sets two targets: loan decisions in minutes, and new products several times a year.

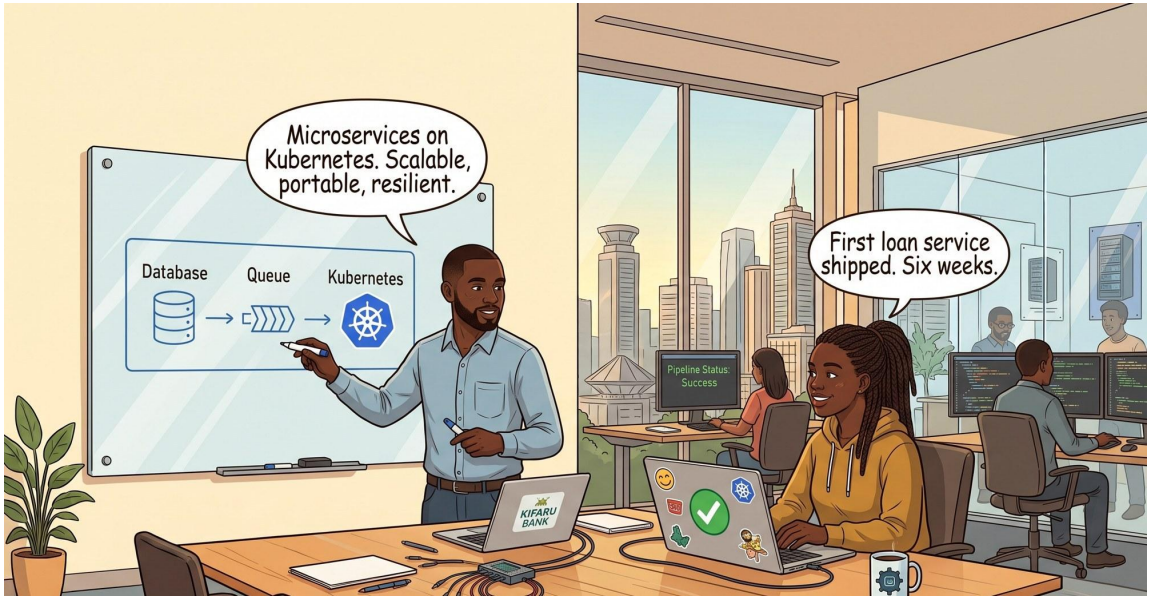
The second is the hard one. Anyone can digitise one product.



And it worked

Microservices on Kubernetes. Wanjiru ships the first loan service in six weeks.

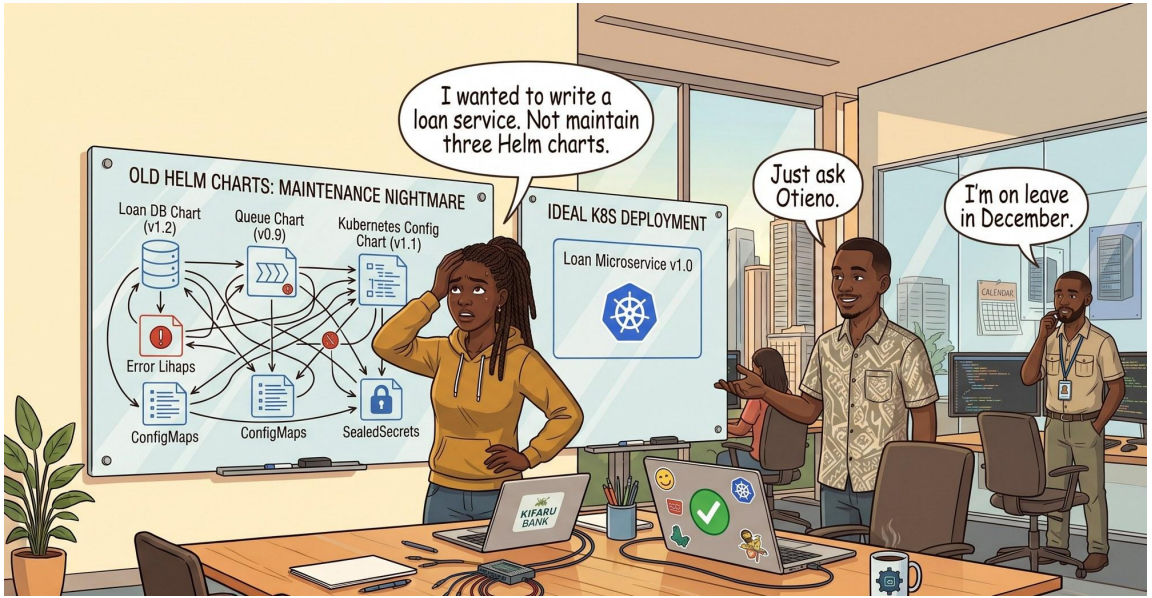
Nothing went wrong here. Kubernetes did what it promised.



Six months later

Three Helm charts, two pipelines, and one person who knows how all of it works.

Nobody decided this. Each step was reasonable on the day.



Then the regulator asks

An auditor asks Otieno to show how access to the scoring engine is controlled.

The controls are real. The evidence is not.

So what is an Internal Developer Platform?

Not a portal and not a wrapper — the designed path from code to production, and the contract between the two roles who maintain it.

Self-service

Databases, environments and deployments without a ticket. An opinion about lending, not about ingress controllers.

Golden paths

One supported route, genuinely easier than going around it. Nothing left to diverge from.

Guardrails

A versioned file with an author and a date, not a person and a wiki page. The control becomes the evidence.

Kubernetes stays visible and operable. Developers get a simpler surface, not a blindfold.

An IDP is not a tool you install. It is the set of decisions you stopped leaving to chance.

So we'll build it ourselves

Developer portal	Identity — OIDC, OAuth2	Authorization — RBAC, ABAC	Application abstraction
Build, CI, registry	GitOps and promotion	Managed databases, caches, queues	API gateway and rate limits
Network policy, zero trust	Logs, metrics, traces, alerts	Secret management	Cost and FinOps

Every one of these has a good open-source answer. None of them knows about the other eleven.

And you maintain all of it through every Kubernetes upgrade.

Three ways that goes wrong

Not our view — this is what practitioners report.

No MVP

Nothing gets an internal platform right first time. It has to start small and iterate, and keep proving its value as it goes.

No evidence

Without measured developer experience and delivery outcomes, the platform cannot justify itself when budgets are reviewed.

No boundary

Nobody writes down who owns what, so the platform team drifts back into being a help desk.

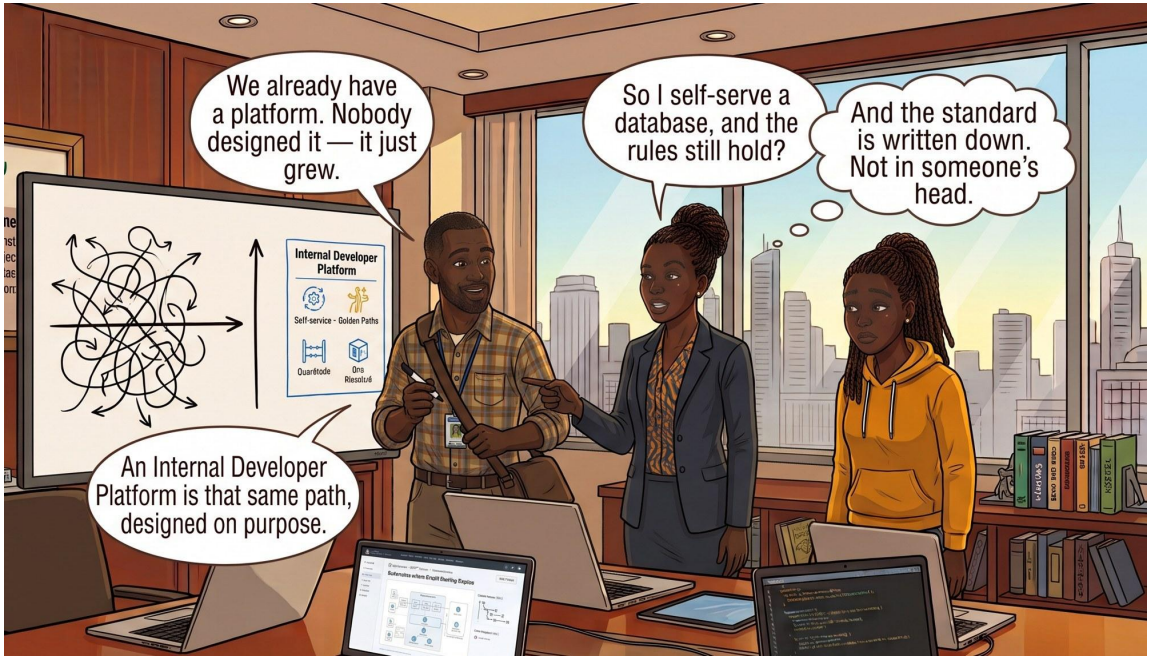
None of these is a technology problem. All three decide whether the platform survives.

They already had a platform

Three charts, two pipelines and one engineer's memory were already a platform.

Nobody designed it. An IDP is that same path, designed on purpose.

So Kifaru picks a foundation they can inspect first — open source, CNCF Sandbox, Kubernetes-native.



THE PLATFORM WE
BUILD ON

OpenChoreo

WSO2

What OpenChoreo is

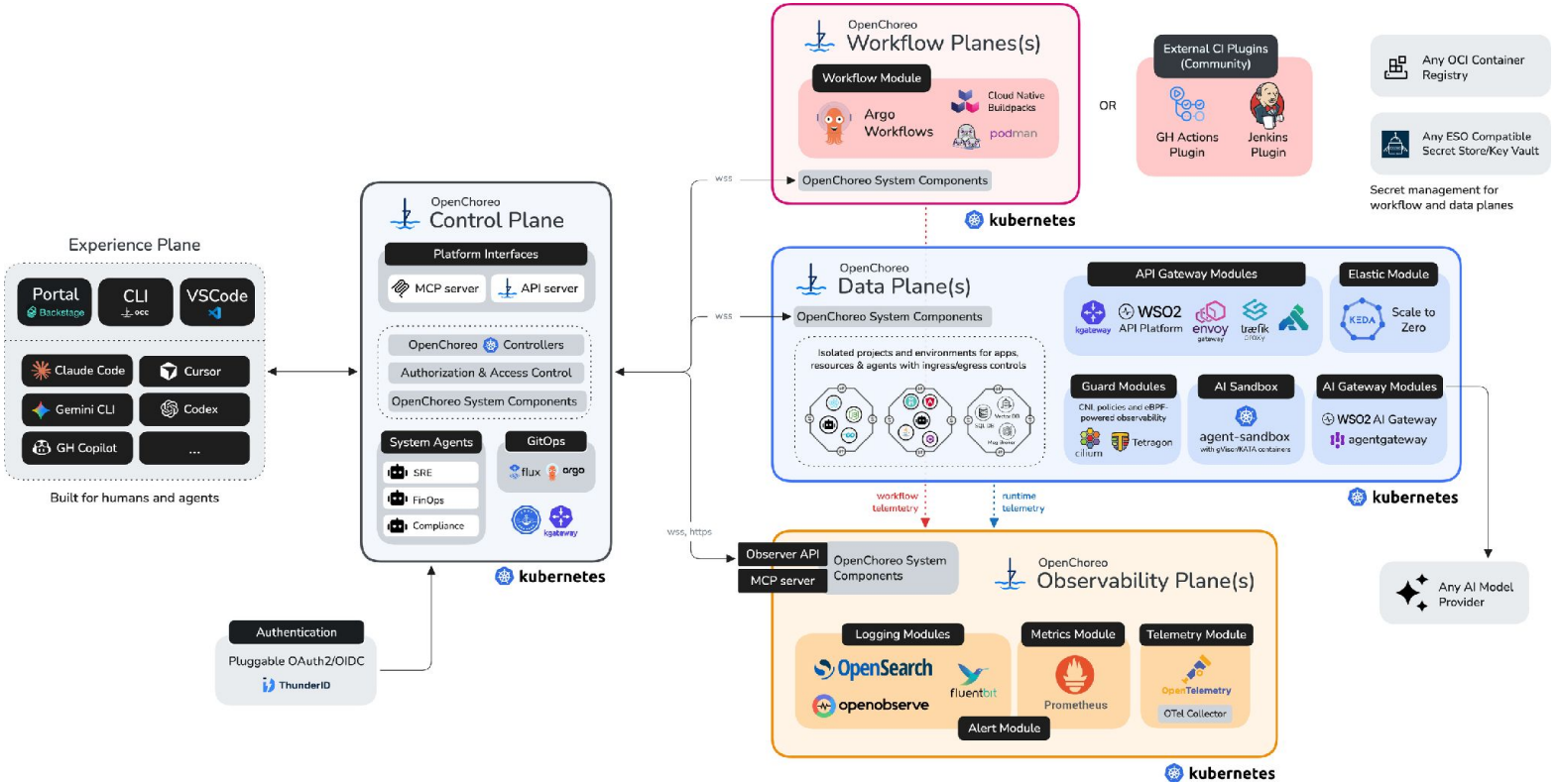
A complete, modular, open-source internal developer platform for Kubernetes.

Built by WSO2 out of years of running a developer platform as a service, and contributed to the community. A CNCF Sandbox project since January 2026.

Kubernetes-native: everything is a CRD, reconciled by controllers. You can always look underneath.

It gives you the abstractions, not the answers. You define what a service means at your bank.

Five planes


Experience Plane
Portal
Backstage
CLI
occ
VSCode
Claude Code
Cursor
Gemini CLI
Codex
GH Copilot
...
Built for humans and agents
OpenChoreo
Control Plane
Platform Interfaces
MCP server
API server
OpenChoreo Controllers
Authorization & Access Control
OpenChoreo System Components
System Agents
SRE
FinOps
Compliance
GitOps
flux
argo
kgateway
kubernetes
Authentication
Pluggable OAuth2/OIDC
ThunderID
wss
wss
wss, https
OpenChoreo
Workflow Planes(s)
Workflow Module
Argo Workflows
Cloud Native Buildpacks
podman
OpenChoreo System Components
kubernetes
OR
External CI Plugins (Community)
GH Actions Plugin
Jenkins Plugin
Any OCI Container Registry
Any ESO Compatible Secret Store/Key Vault
Secret management for workflow and data planes
OpenChoreo
Data Plane(s)
OpenChoreo System Components
Isolated projects and environments for apps, resources & agents with ingress/egress controls
API Gateway Modules
kgateway
WSO2 API Platform
envoy gateway
Traefik
Elastic Module
KEDA
Scale to Zero
Guard Modules
CNI policies and eBPF-powered observability
cilium
Tetragon
AI Sandbox
agent-sandbox
with gVisor/KATA containers
AI Gateway Modules
WSO2 AI Gateway
agentgateway
kubernetes
workflow telemetry
runtime telemetry
Observer API
MCP server
OpenChoreo System Components
OpenChoreo
Observability Plane(s)
Logging Modules
OpenSearch
openobserve
fluentbit
Metrics Module
Prometheus
Telemetry Module
OpenTelemetry
OTel Collector
Alert Module
kubernetes
Any AI Model Provider

Two hats, one contract

I need a service and a database, and the service talks to scoring.

I will define what a service is, provision the databases, and define the standard routes.

OpenChoreo

they meet at the Release

The Developer Declares Intent

The Platform Engineer Defines the Standard

They never edit each other's files.

The abstractions

Platform API — the platform engineer writes these

ProjectType · ComponentType · ResourceType · Trait · Workflow · Environment · DeploymentPipeline · DataPlane

Developer API — the developer writes these

Project · Component · Workload · Resource

At release time, they meet

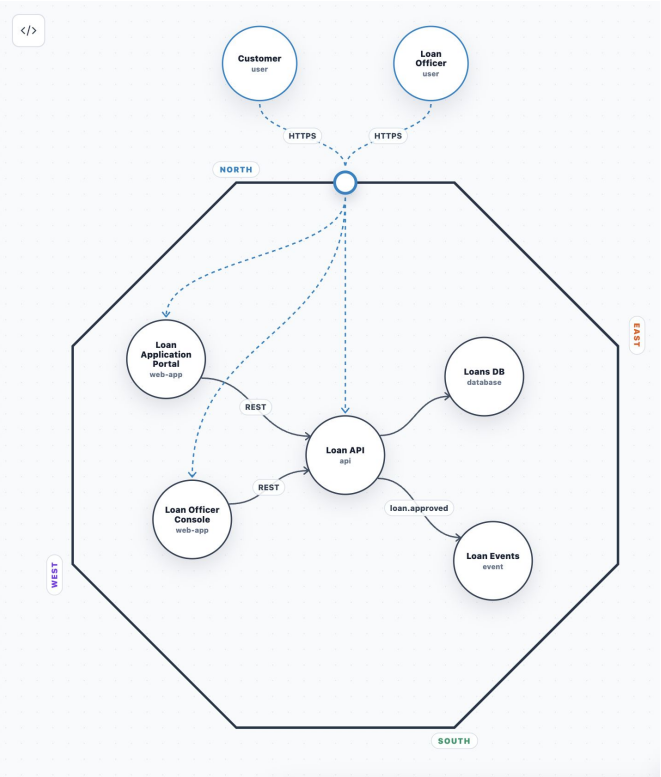
An immutable, reproducible artifact — bound per environment, then rendered.

Cells: the runtime model

A Project becomes a cell — its own namespace and its own network boundary, in every environment.

Components inside talk freely. Everything crossing the boundary goes through a gateway.

Architectural boundaries enforced by infrastructure, not by code review.

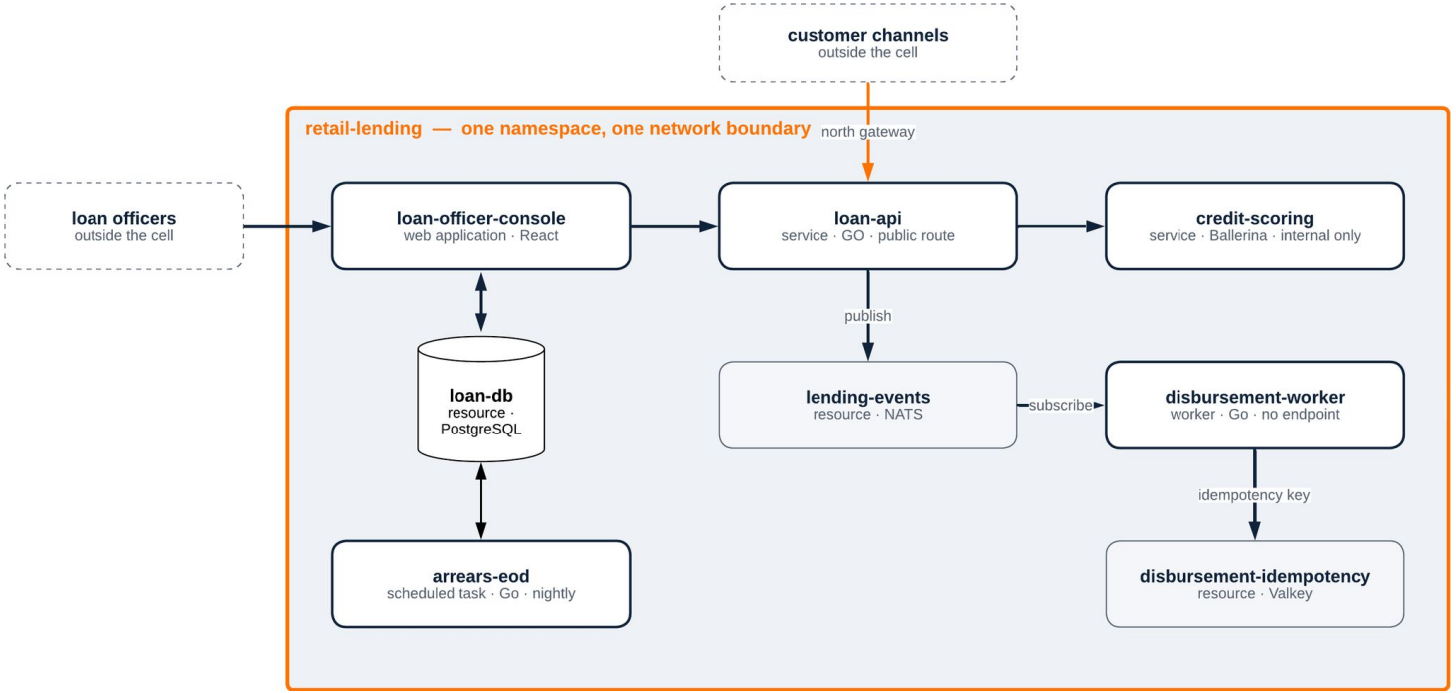


Demo, part one

Clone it now and follow along:

github.com/wso2con/2026-NBO-Internal-Developer-Platform-tutorial-1

Demo use case — Kifaru Lending System



Every box here needs a ComponentType, a ResourceType or a Workflow. That is the golden path.

Platform Engineer Tasks

What we build

Three environments and one DeploymentPipeline

A ProjectType — every project starts closed

Four ComponentTypes and three ResourceTypes

Four Workflows and one alert-rule Trait

How it is written

Manifest templates with CEL expressions

includeWhen — no route unless an endpoint exists

forEach — one resource per declared item

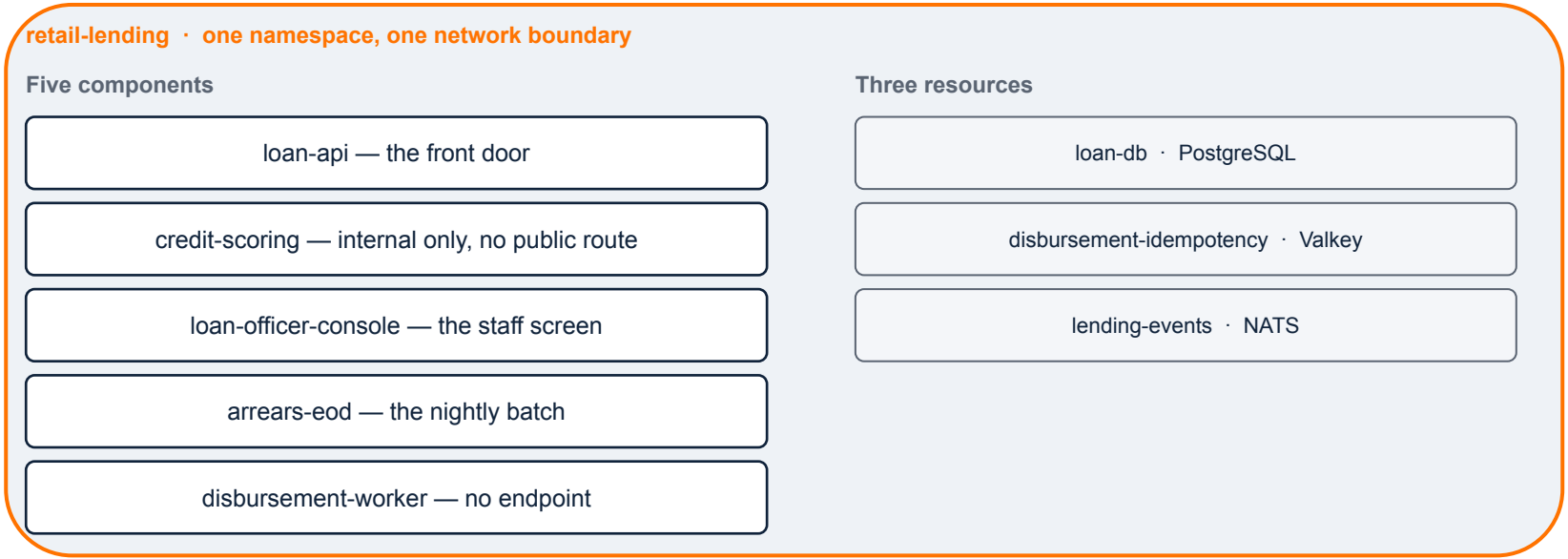
No controller code — a file in Git, with an author

Installing OpenChoreo is not platform engineering. Deciding what a service means at this bank is.

Demo, part two

The Kifaru retail lending system

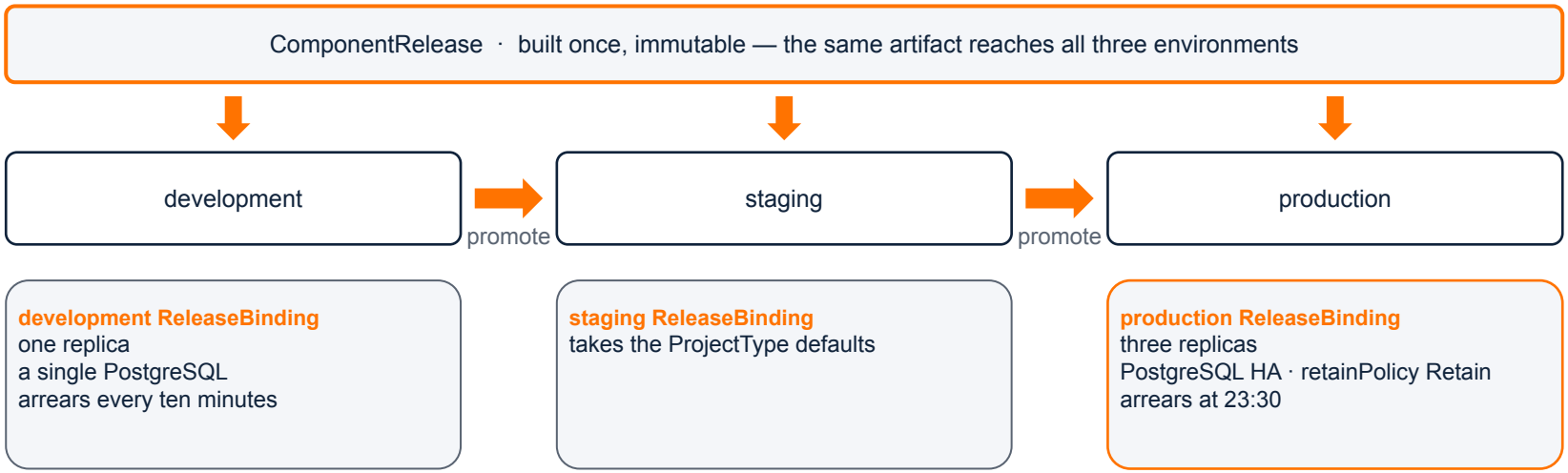
One project becomes one cell — one namespace, one network boundary.



Three resources, four lines of YAML each. No ticket, no module, no credential handled by the developer.

Promotion, and the overrides

Development to staging to production. Two commands. No YAML re-applied, no rebuild.



The developer wrote none of that. The platform engineer did.

Observability

Logs, metrics, traces and alerts scoped to a component, not to raw Kubernetes objects.

One trace for APP-100244: gateway, loan-api, credit-scoring, the broker, the worker. Two languages, four services, one request.

The alert rule came from a five-line trait on the component. Nobody logged into Prometheus.

Cost insights attach to the release binding — governance, not a stray Deployment.

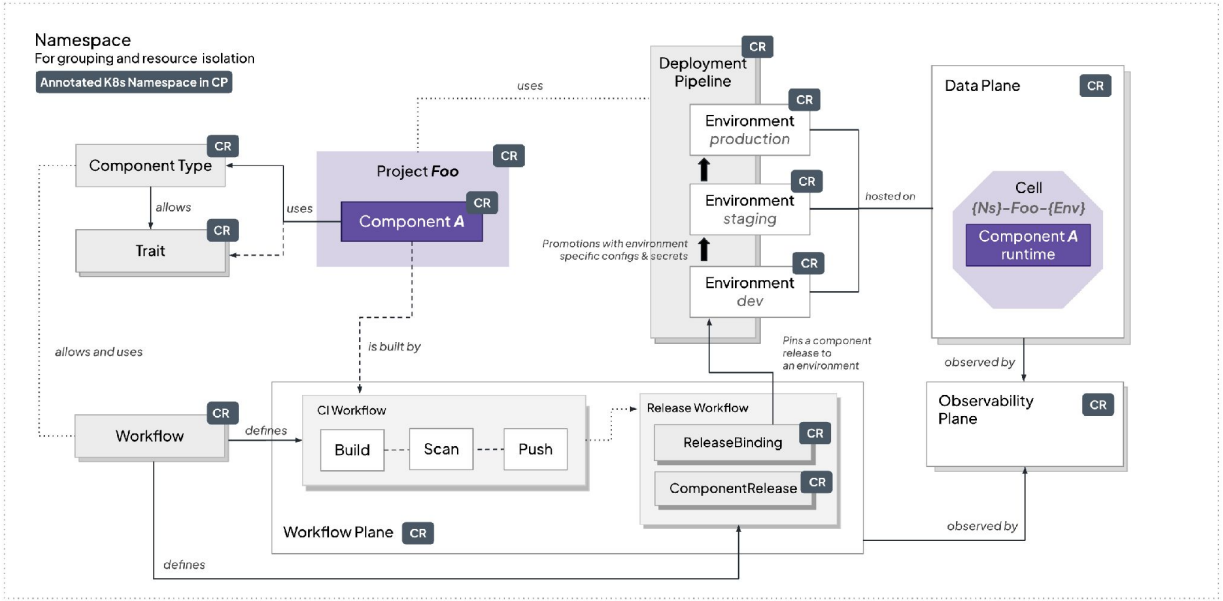
IF TIME PERMITS

Architecture In-Depth

Reference material. We run these live only if the demo leaves room — the slides stand on their own either way.

WSO2

The Platform API, end to end

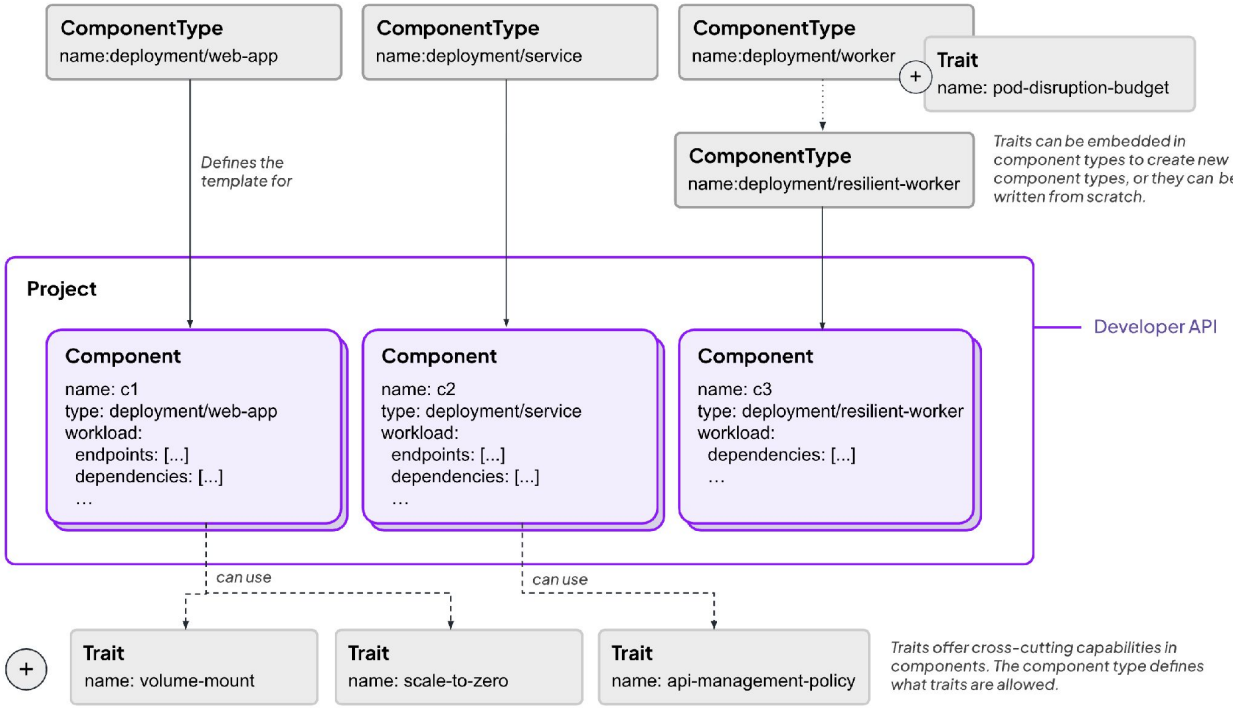


Cluster-level (*global*) resources that can shared across any control plane namespace
(for resource and infrastructure sharing)

- Cluster Data Plane
- Cluster Workflow Plane
- Cluster Observability Plane
- Cluster Environment
- Cluster Deployment Pipeline
- Cluster Workflow
- Cluster Component Type
- Cluster Trait

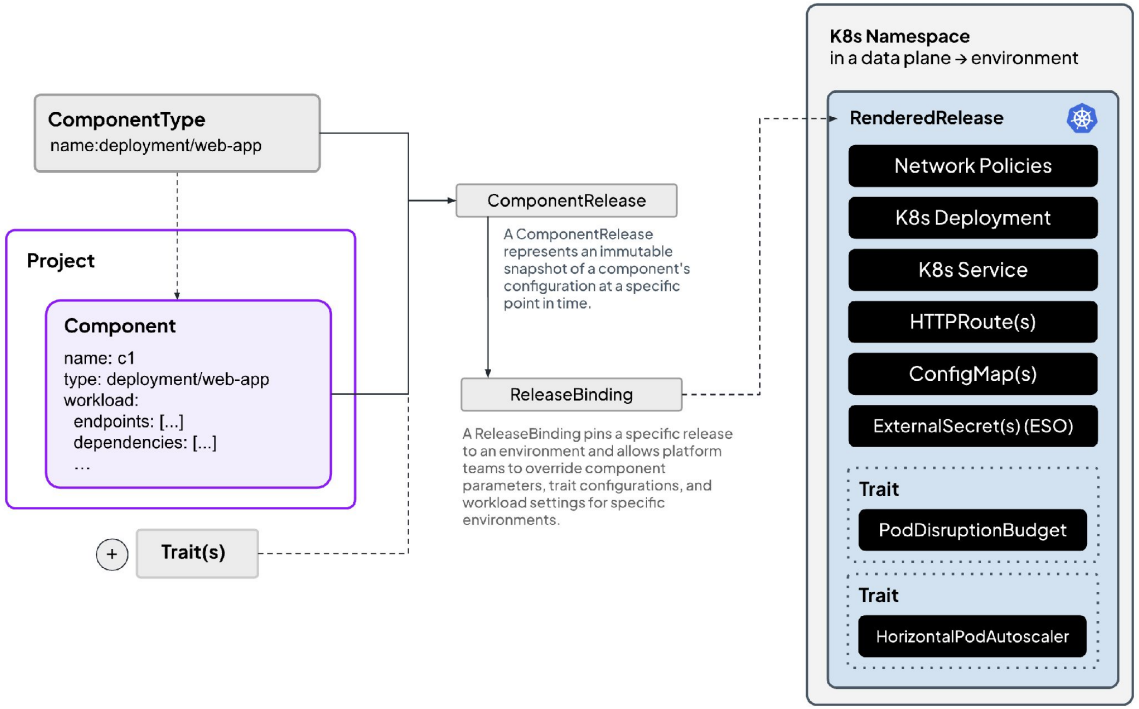
Every box marked CR is a Kubernetes custom resource. There is no platform database.

The Developer API



The developer writes the purple boxes. The ComponentType decides which traits are even offered.

What actually gets created



Nothing is hidden. Every object on the right is plain Kubernetes, and the developer wrote none of them.

Questions we get asked

Does this hide Kubernetes from my team?

No. Everything is a CRD and the rendered resources are right there. It removes the boilerplate, not the visibility.

Do we have to replace our CI?

No. Jenkins, GitHub Actions and GitLab CI register builds through the Platform API and appear in the same build history.

What does adoption actually cost?

Start with one project type and one component type. A platform that tries to cover everything on day one covers nothing well.

Questions

https://openchoreo.dev/

CNCF Slack: #openchoreo

WSO2

Keep an eye out for

Day 3 (Thursday, Sep 24th)

11:00 a.m. (30 mins)

Custom Resources, Not Custom Forks – Curating OpenChoreo for Your Enterprise

Tishan Dahanayakage
Director - Head of Engineering, Choreo BU

WSO2

Yellow Room

10:30 a.m. (30 mins)

The Platform Abstractions: What Developers Declare, What Platform Engineers Govern in OpenChoreo

Kavith Lokuhewage
Associate Director & Architect

WSO2

Yellow Room

Thank You