

October 2026

Cell-based Isolation for Multi-Tenant Platforms on Kubernetes

Lahiru De Silva

Associate Technical Lead · WSO2

OpenChoreo Maintainer



October 2026

Multi-tenancy in Kubernetes

Who is a tenant?

SOFT MULTI-TENANCY

HARD MULTI-TENANCY



Multiple teams

Implicitly trusted. A crossed boundary is often taken lightly.



Business units, partners

Contractual boundaries and separate budgets.



External customers

Mutually untrusted. One breach ends the contract.

TRUST

Assume good faith

Assume nothing

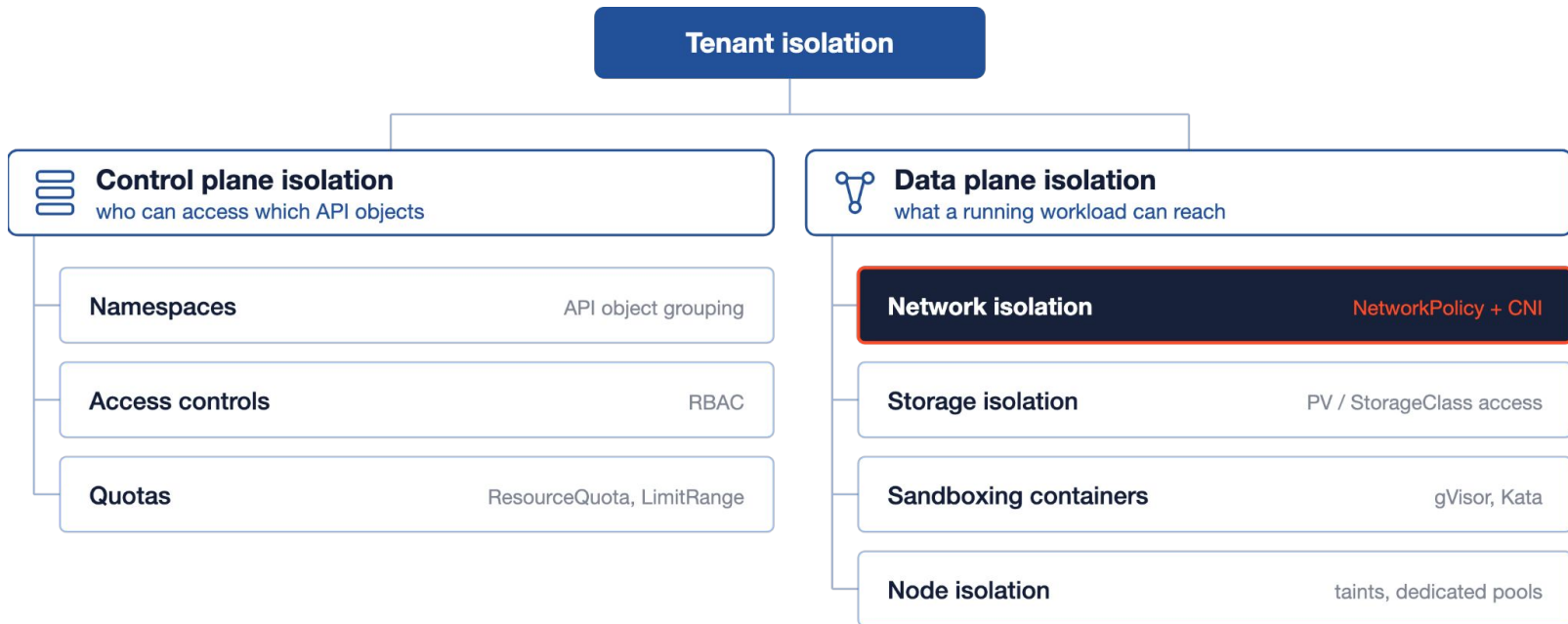
ENFORCED BY

Convention — people agree not to

Mechanism — and you can show it holds



Tenant Isolation



Why network isolation is difficult

☰ The others: pick from a short list

the platform team decides once, per tenant

Namespaces

1 per tenant

a fixed few

Access controls

admin

dev

sre

Node isolation

shared

dedicated

Sandboxing

runc

gVisor

Quotas

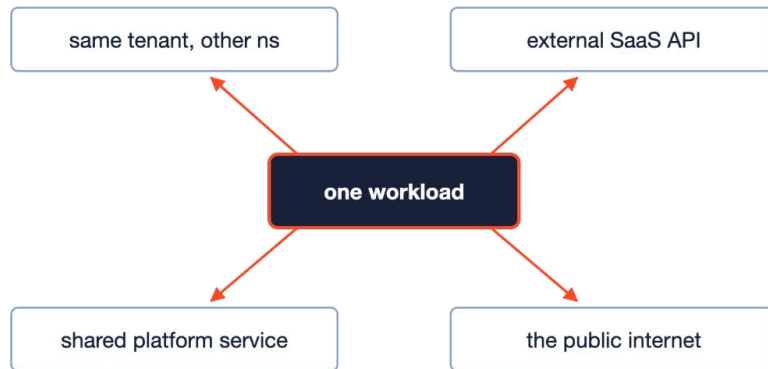
a number

Small, enumerable, stable.



Network isolation: describe a graph

each application decides — and it changes every release



Unbounded, per-application, continuously changing.

The platform team owns the left. The application teams own the right — and there are a lot of them.



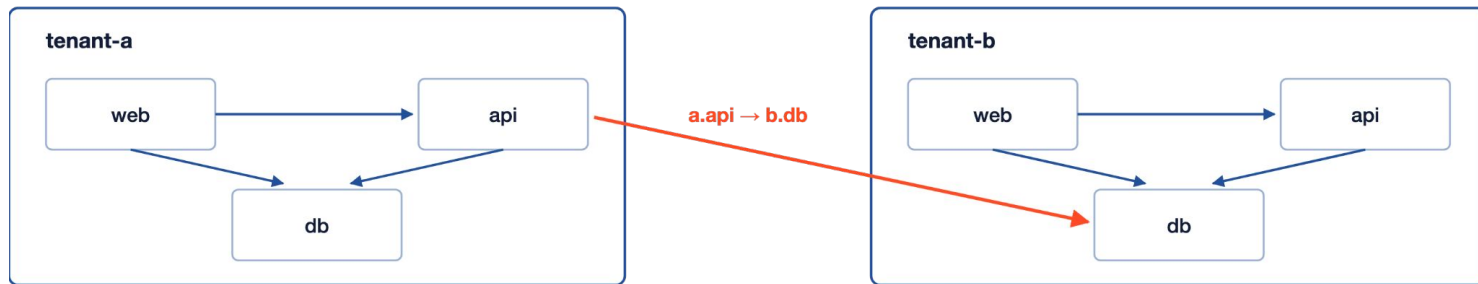
October 2026

Network micro-segmentation in Kubernetes

Namespaces and NetworkPolicy

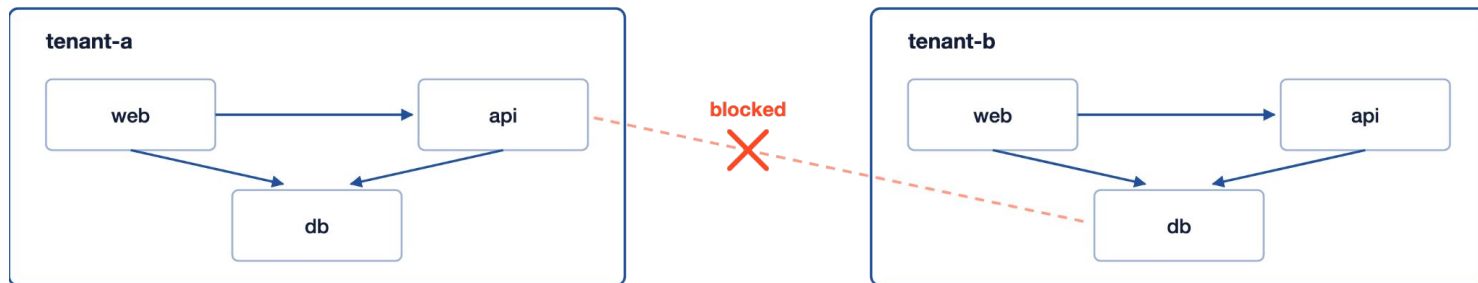
No NetworkPolicy

every pod reaches every pod, in every namespace



With NetworkPolicy

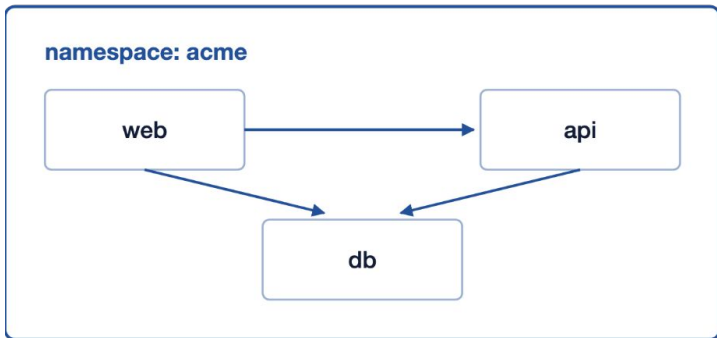
the cross-tenant call is refused — everything inside each tenant still flows



How the boundary fragments as tenants grow

YEAR ONE

One tenant, one namespace, three services.



The boundary is an object.

All three connections stay inside the namespace, so a single rule shuts the tenant off from everyone else.

Ask what separates this tenant from the next, and you point at a name.

3 connections · 1 rule

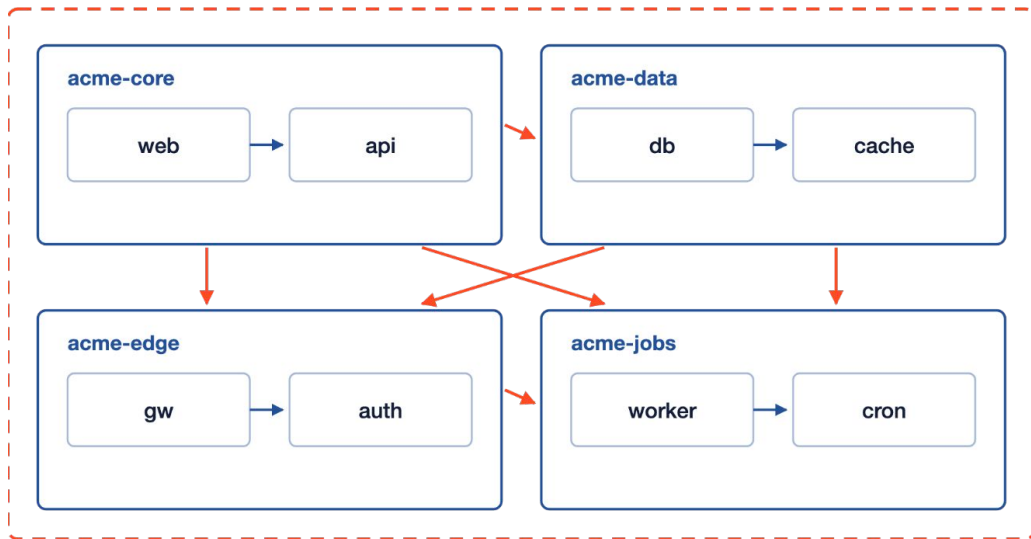
When the tenancy boundary and the namespace boundary are the same object, correctness is obvious.



How the boundary fragments as tenants grow

YEAR TWO

Same tenant, same application — now spread across four namespaces.



— STAYS INSIDE — NO NEW RULE — CROSSES AN EDGE — NEEDS ITS OWN RULE

Now it's a dashed line.

Separate namespaces for core, data, edge and batch — for good reasons, usually RBAC or release lifecycle.

10 connections · 7 rules

The isolation rule from year one is still there — plus one more for every crossing, each written, reviewed and kept correct as the services change.

Nothing in the cluster represents the tenant.

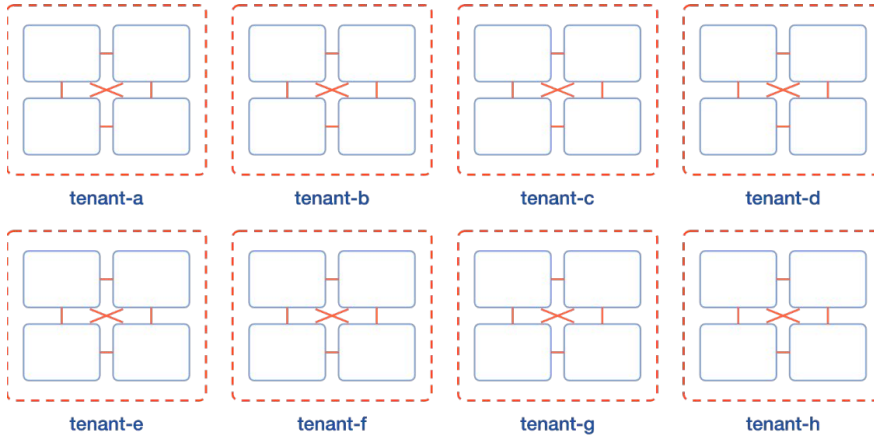
Nothing went wrong. The tenant grew — and the boundary stopped being a thing you can point at.



How the boundary fragments as tenants grow

AND THEN — TWENTY OF THEM

Every tenant has its own shape, and its own connectivity graph.



× 20 tenants

Seven rules for a tenant this shape
— and every tenant is a different
shape. One wrong selector anywhere
opens a path that should not exist.

**~140 rules
that must all be right.**

The rule count grows with tenants × services. The number of people who can hold it in their head does not.



A design problem, not a configuration problem

WHAT'S ACTUALLY BROKEN



So the fix isn't a better rule. It's giving the boundary a name, a shape, and a default.



October 2026

Cell-based Architecture

Cell-based Architecture

A cell is a collection of components, grouped from design and implementation into deployment — independently deployable, manageable, and observable.

Asanka Abeysinghe & Paul Fremantle · WSO2 Reference Architecture, first published Summer 2018

Self-contained

Components talk freely inside. Everything else leaves through a gateway.

One way in

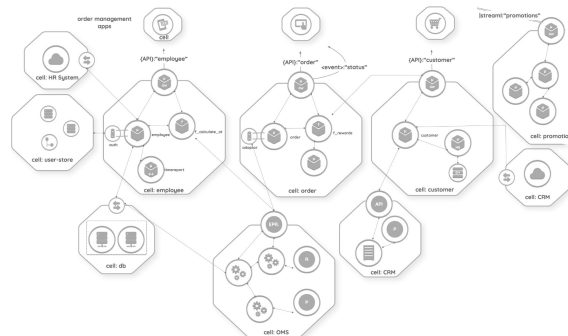
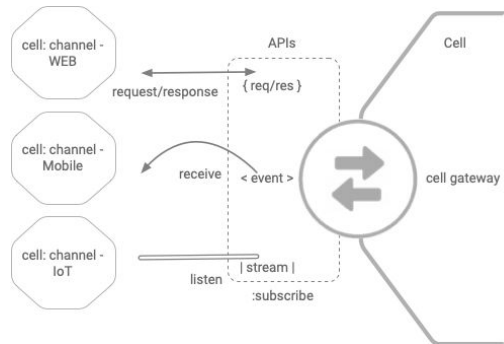
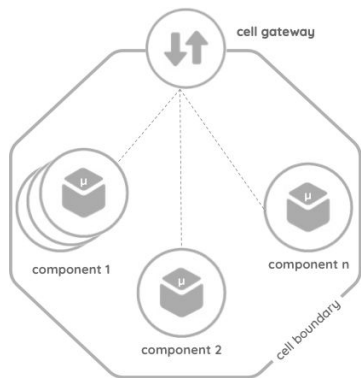
The gateway is the only access point — and the policy enforcement point.

Bounded context

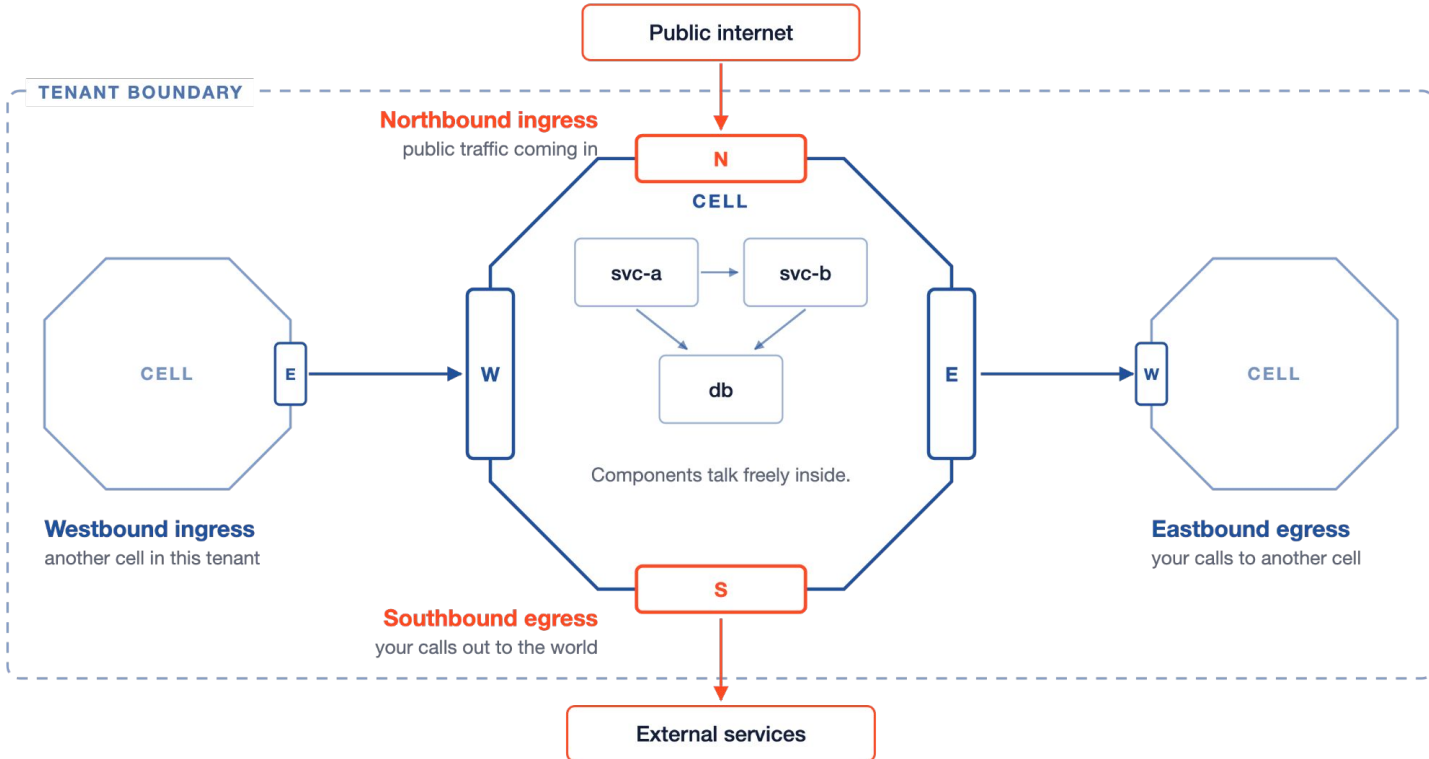
One functional domain,
owned by one team that
builds and runs it.

Versioned, immutable

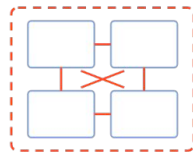
Built, deployed and replaced as one unit, not patched in place.



Cell gateways and connectivity

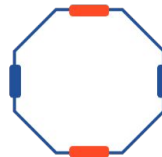


Rules vs. Cell



RULES

a pile you maintain



CELL

a boundary you declare

SCALES AS

tenants × services

one more cell

THE DEFAULT IS

whatever you remembered to write

deny, except through the gateways

"WHAT CAN THIS REACH?"

run an experiment

read it

THE BOUNDARY IS

a convention

an object — with a name

Same services. Same enforcement. Only one of them is a thing you can point at.



October 2026

OpenChoreo as a reference implementation

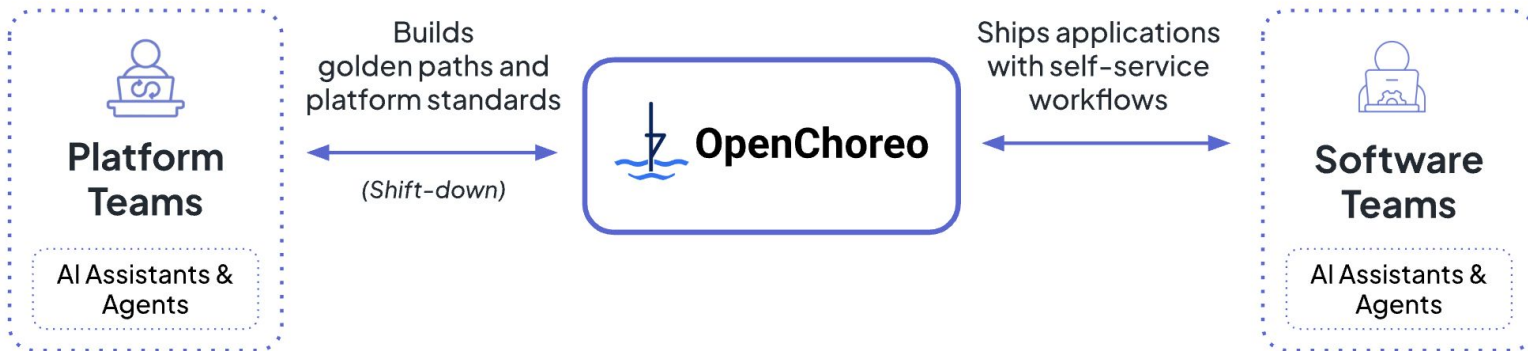


CLOUD NATIVE
COMPUTING FOUNDATION

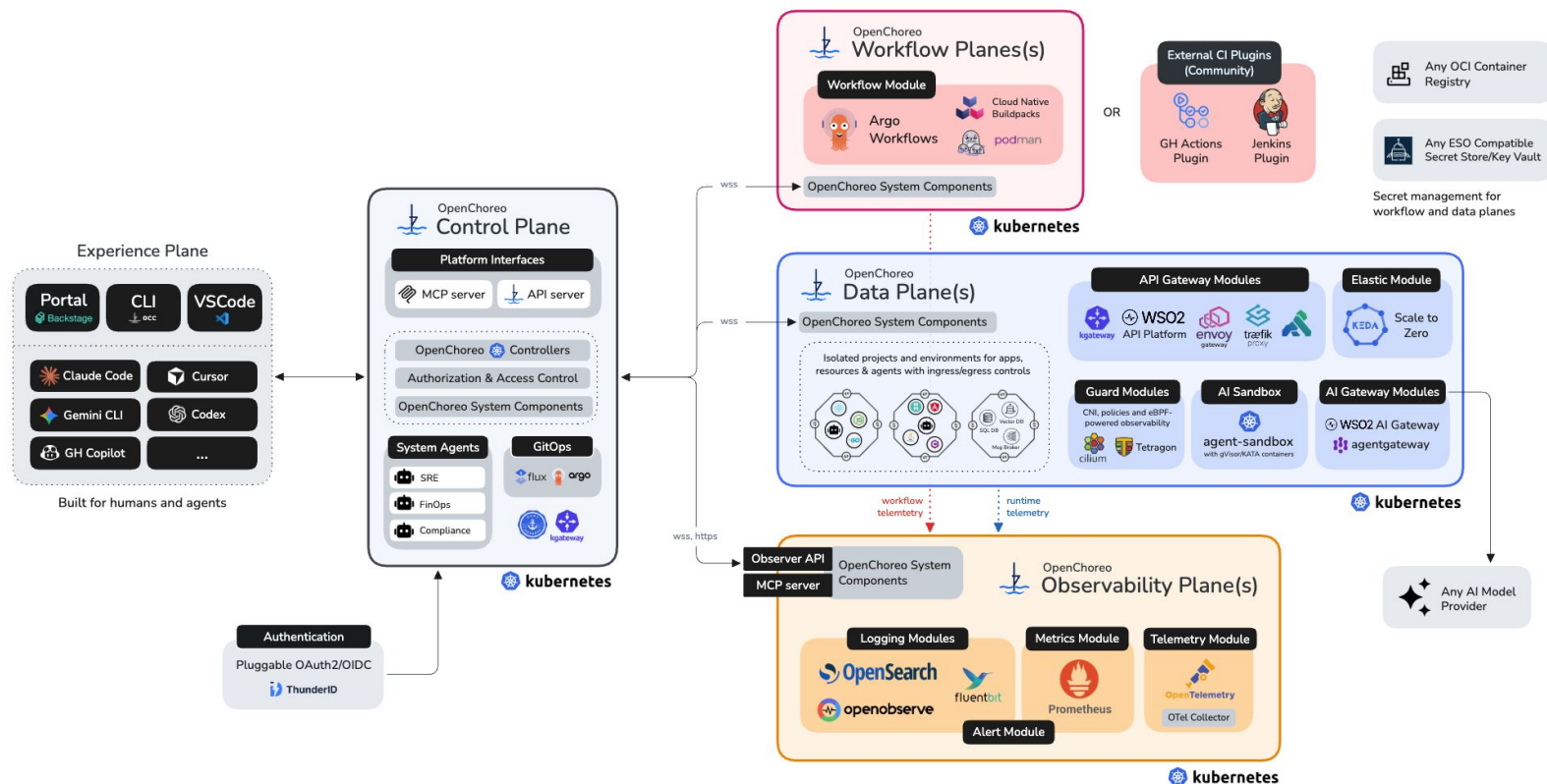


What is OpenChoreo?

- A complete, modular, open-source developer platform for Kubernetes
- Developed by WSO2 and donated to CNCF as a CNCF Sandbox Project in January 2026
- High level platform and developer abstractions

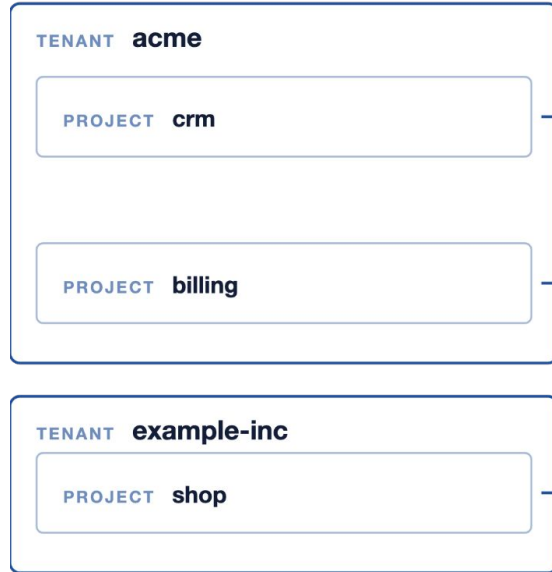


OpenChoreo multi-plane architecture

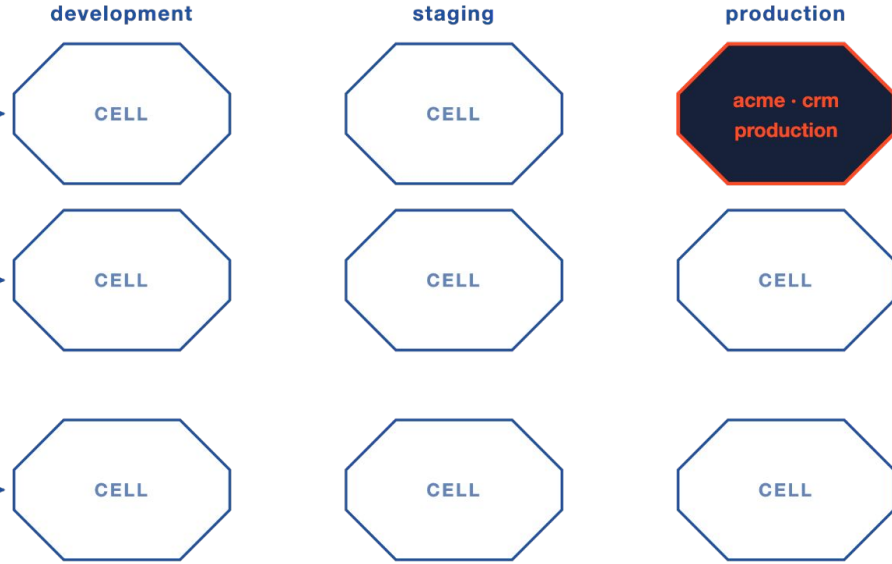


Representation of a Cell in OpenChoreo

CONTROL PLANE



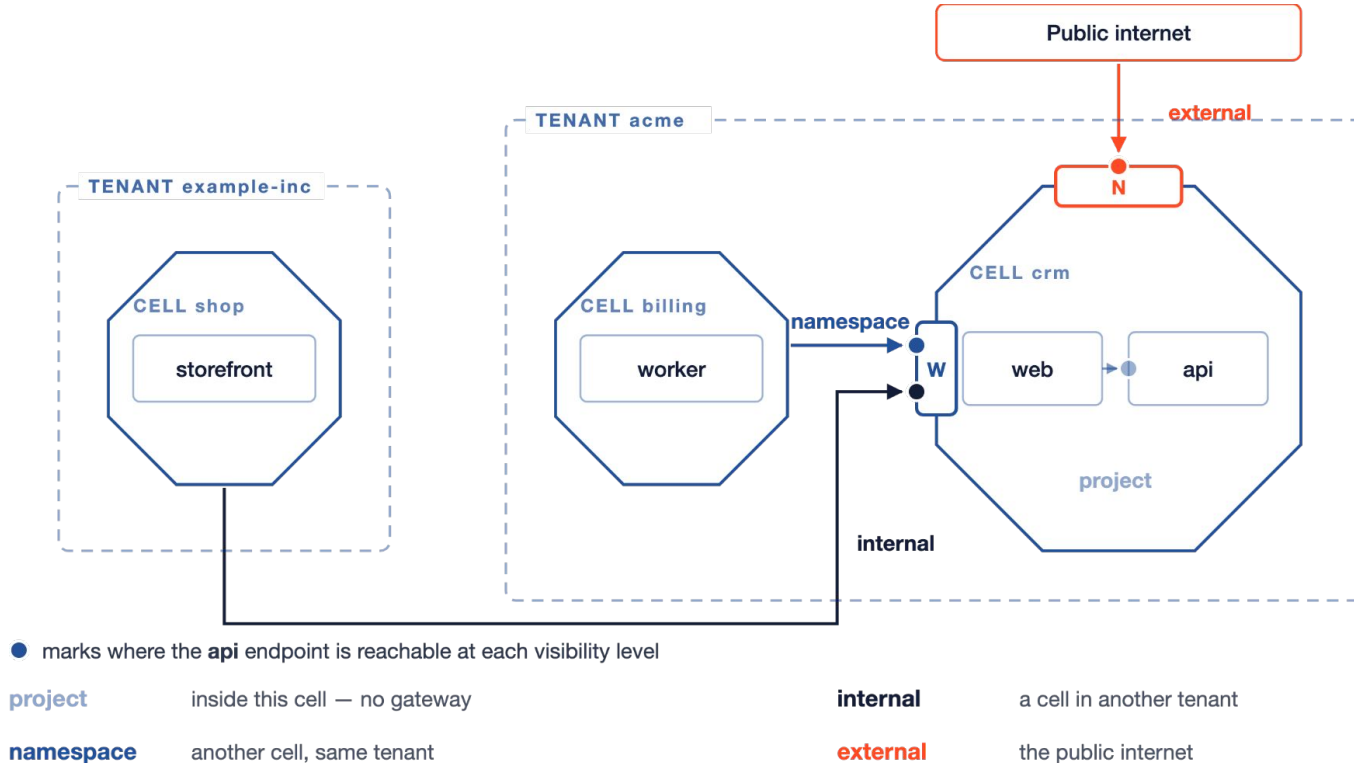
DATA PLANE – ONE CELL PER PROJECT, PER ENVIRONMENT



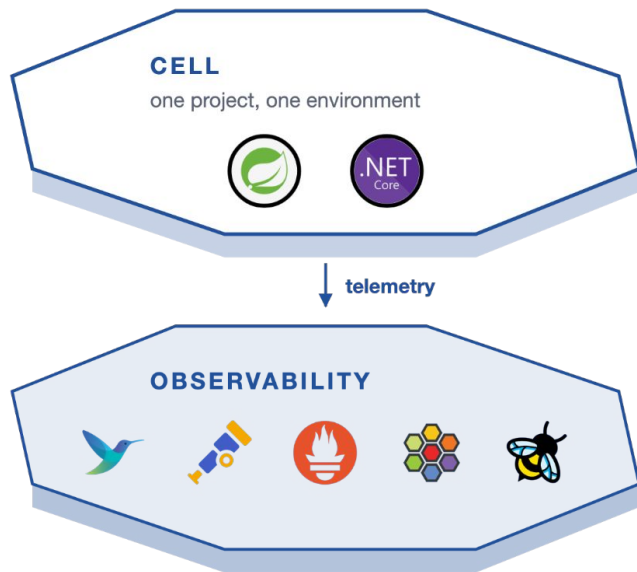
3 projects × 3 environments = 9 cells



OpenChoreo Endpoints and Cells



Cell Observability in OpenChoreo



WHAT THE CELL EMITS

Logs

every container in the cell

Metrics

CPU, memory, HTTP

Traces

OTLP, from the app

Wire logs

network-level detail

COLLECTED BY

Fluent Bit · OpenTelemetry · Prometheus

Cilium + eBPF

network visibility without touching the app

If you build a boundary and can't see across it, you haven't finished.

Every signal is tagged at collection with the cell's identity — **namespace · project · environment · component**



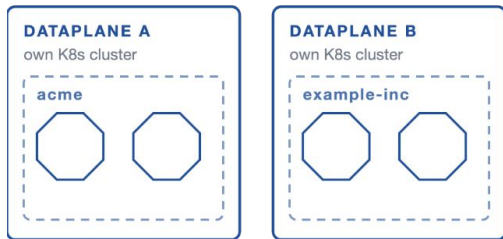
Multi-tenancy patterns in OpenChoreo

MOST ISOLATION

LEAST ISOLATION

PATTERN 01

DataPlane per tenant

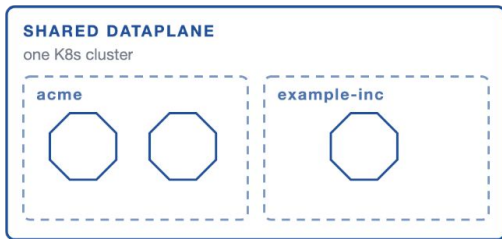


Each tenant gets its own DataPlane, pointing at its own Kubernetes cluster.

Physical isolation — at full cost.

PATTERN 02

Namespace per tenant

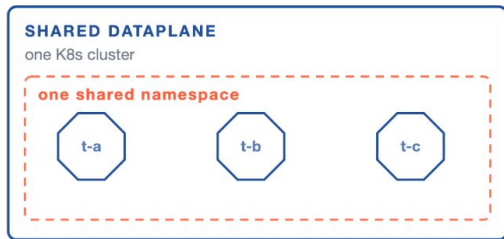


Tenants share a DataPlane. Each gets its own namespace, and a cell per project.

The usual middle ground.

PATTERN 03

Project per tenant



Tenants share a namespace. Each tenant is a project — one cell each.

Sub-teams inside one team.



October 2026

OpenChoreo Product Walkthrough