WSO2CON AFRICA

SEPT 22 – 24, 2026 | NAIROBI, KENYA

The Platform Abstractions

What Developers Declare, What Platform Engineers Govern

Kavith Lokuhewage

Associate Director & Architect

The state of developer platforms

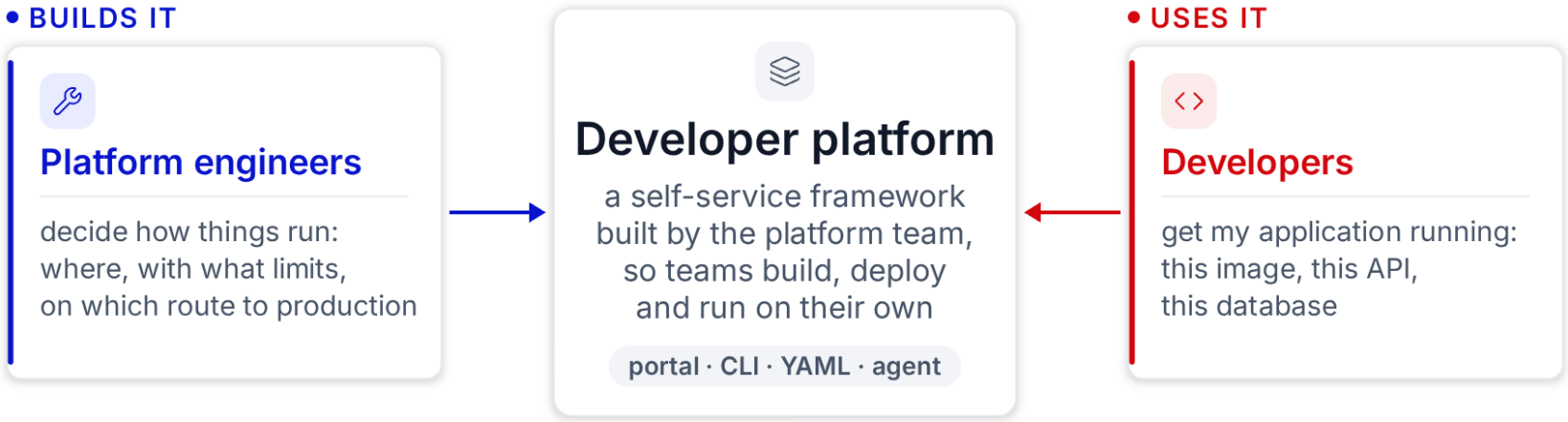
80%

of large software engineering organisations will have platform engineering teams by 2026

Gartner

- Grown out of DevOps, or built from scratch
- Self-service, so teams ship on their own
- One side builds it. The other uses it.

Two roles, one platform



Every platform decision serves one of them — usually at the other's expense.

Why platforms fail

Developers

must learn the platform to use it

tickets, then waiting

so they use kubectl instead

Platform engineers

twenty tools to stitch

every team's exception

cannot change what is already exposed

Developers should not have to learn the platform to use it.

The wrong abstraction

Meant: image, port, database.

Got: 200 lines, six resources.

Infrastructure, not an application.

WHAT THE DEVELOPER MEANT

what I want

```
image: acme/catalog:1.4
port: 8080
needs: postgres
```

WHAT THEY ARE HANDED

catalog-service.yaml

```
kind: Deployment
spec:
  replicas: 2
  template:
    containers:
      - image: acme/catalog:1.4
        ports:
          - containerPort: 8080
        resources:
          limits:
            cpu: 500m
        securityContext:
          runAsNonRoot: true
---
```

... about 180 more lines, in six resources

Kubernetes solved this for routing

• INGRESS: ONE OBJECT, ALL ROLES

Ingress

Three roles edit the same object, and annotations carry the rest.

infra provider · cluster operator · app developer · all three, one file

Gateway API

• GATEWAY API: ONE RESOURCE PER ROLE

GatewayClass
owned by the infrastructure provider

Gateway
owned by the cluster operator

HTTPRoute
owned by the application developer

Kubernetes split routing by role — on purpose.

The Gateway API design calls itself role-oriented. Routing only — nothing like it exists for workloads.

A Deployment has two owners

ONE FILE, COLOURED BY WHO EDITS EACH LINE

deployment.yaml

```yaml
kind: Deployment
spec:
  replicas: 2
  template:
    containers:
      - image: acme/catalog:1.4
        env:
          - name: LOG_LEVEL
            value: debug
        resources:
          limits:
            cpu: 500m
            memory: 512Mi
        securityContext:
          runAsNonRoot: true
    topologySpreadConstraints:
```

The developer changes

`image · env · config · args`

The platform changes

`replicas · limits · security · spread`

Same file, two owners — and nothing in the API says so.

A typical division of ownership, not a rule.

The workarounds

Mutating admission

strip, then inject

GitOps

a base from dev, overlays from the platform

A chart — or an in-house operator

values from the teams, or your own CRD

Same resource. Two sources of truth.

Kubernetes is not the problem.
Its abstractions solve a different problem.

Every layer gave us simpler words

	• WHAT IT HID	• WHAT IT GAVE
The developer platform — ?	about twelve Kubernetes resources per service	**?**
Kubernetes	machines, networking, storage	you describe the result, not the steps
Containers	namespaces, cgroups, file systems	one image that runs anywhere
Operating system	the processor, memory, devices	programs and files

What each side needs

THE DEVELOPER

“This image. An API on port 8080. A database. Run it.”

THE PLATFORM ENGINEER

“Where it runs, with what limits, behind which gateway.”

Two languages. Neither speaks the other's.

Stilt fishing, southern Sri Lanka

THE STORY BEHIND THE MARK

OpenChoreo

Our logo is a fishing stilt from the south of Sri Lanka.

The mark is the handcrafted perch used in traditional stilt fishing — a single pole planted in the restless sea, with a fisher balanced calmly above the waves.

The waves are everything under the hood — Kubernetes, GitOps, networking, observability. Powerful, ever-moving, and not the developer's problem.

The stilt is the platform — just enough to perch on, keeping developers steady, focused, and above the chaos.

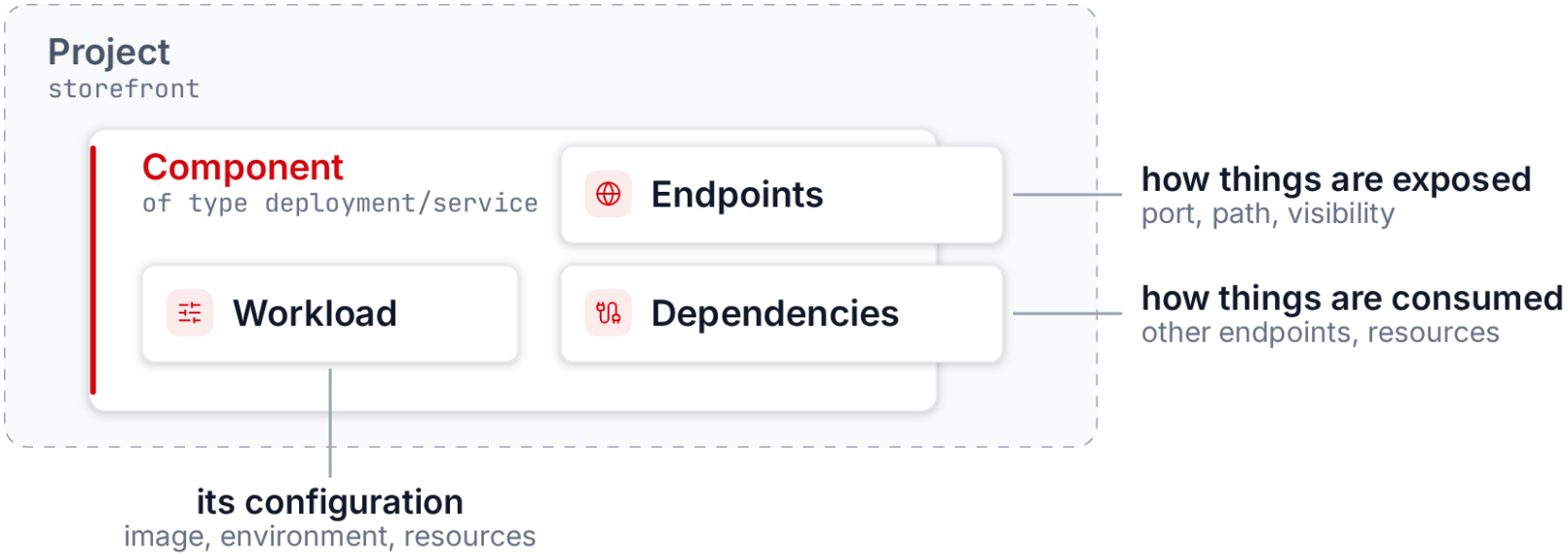
READ THE FULL STORY

openchoreo.dev/blog

How Sri Lankan stilt fishing inspired the logo


WSO2CON
AFRICA
SEPT 22 - 24, 2026 | NAIROBI, KENYA
The developer's side
What I want to run

Developer abstractions



Three things to declare. Everything else is the platform's job.

Everything the developer writes

THE DEVELOPER WRITES THIS

component.yaml

```yaml
kind: Component
metadata:
  name: catalog
spec:
  owner:
    projectName: shop
  componentType:
    kind: ClusterComponentType
    name: deployment/service
```

Two files, 27 lines.

The platform writes the rest.

workload.yaml

```yaml
kind: Workload
metadata:
  name: catalog
spec:
  owner:
    projectName: shop
    componentName: catalog
  container:
    image: acme/catalog:1.4
  endpoints:
    http:
      type: HTTP
      port: 8080
  dependencies:
    resources:
      - ref: postgres
        envBindings:
          url: DATABASE_URL
```

AND NEVER WRITES THIS

- Deployment
- Service
- HTTPRoute
- NetworkPolicy
- ExternalSecret
- HorizontalPodAutoscaler

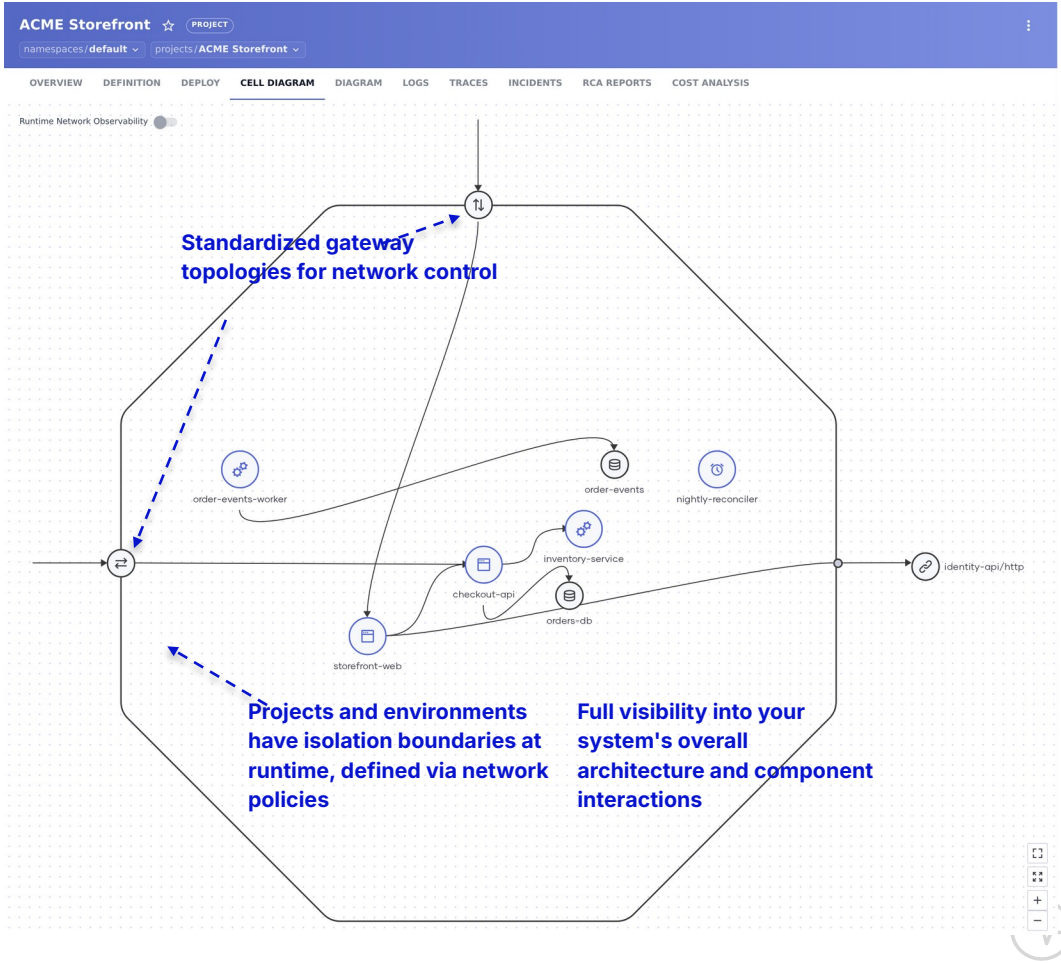
Runtime semantics

Each project becomes a cell.

The developer sets visibility.

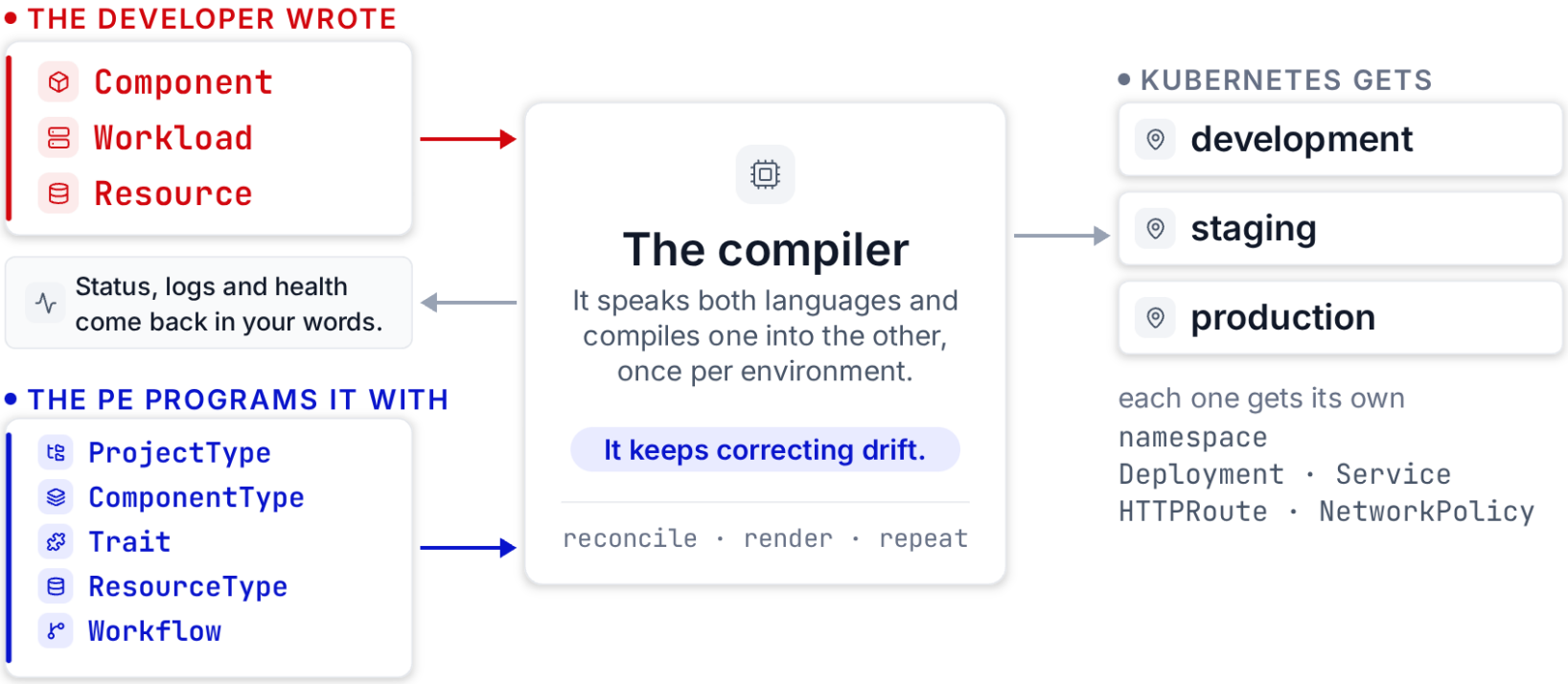
The platform writes the policy.

Routes come with the cell.




WSO2CON
AFRICA
SEPT 22 - 24, 2026 | NAIROBI, KENYA
The platform engineer's side
What is allowed

The control plane compiles intent



Who decided what a service is?

THE DEVELOPER WRITES

the declaration

```
type: deployment/service
port: 8080
basePath: /catalog
visibility: [external]
```

four values

THE PLATFORM ENGINEER DEFINED

ComponentType

cluster-scoped or namespaced

workloadType	kind of workload
parameters	what may be set
environmentConfigs	what varies per env
allowedTraits	add-ons allowed
allowedWorkflows	build paths allowed
validations	rules that must pass
resources	the templates

KUBERNETES GETS

- Deployment
- Service
- HTTPRoute
- NetworkPolicy
- Autoscaler
- ExternalSecret

The application team never sees these resources.

What the platform team defines

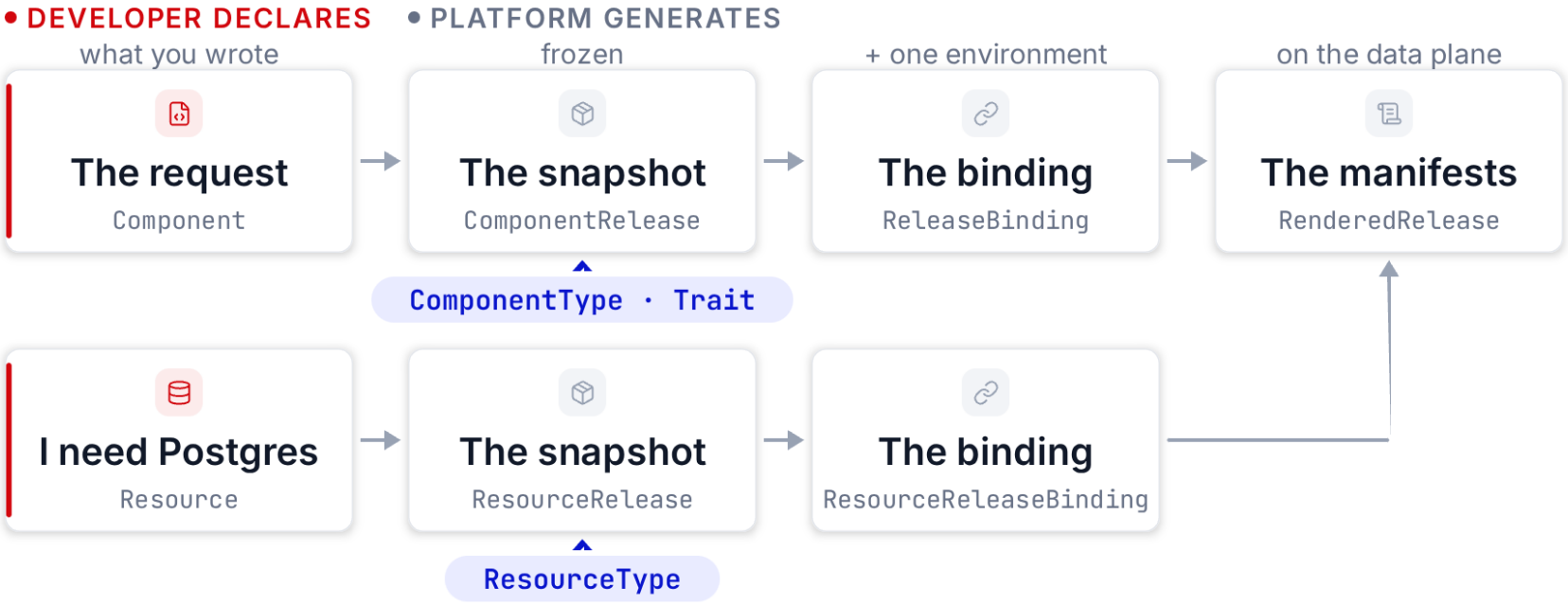
	WHAT IT DEFINES	FOR EXAMPLE
`ProjectType`	what a project is, in every environment	the cell namespace, shared policy
`ComponentType`	what a component is, and what it creates	Deployment, Service, route
`Trait`	behaviour you attach to a component	alert rules, autoscaling
`ResourceType`	how infrastructure is actually provided	PostgreSQL, cache, queue
`Workflow`	how source code becomes an image	Docker, buildpacks

All five are configuration the platform team versions in Git.

Two scopes: namespaced (one organisation) or Cluster* (shared platform-wide).


WSO2CON
AFRICA
SEPT 22 - 24, 2026 | NAIROBI, KENYA
Where the two sides meet
Release time

Release, binding, rendered



The component's binding takes its connection details from the resource's binding — and reports the dependency as pending until that resource is ready.

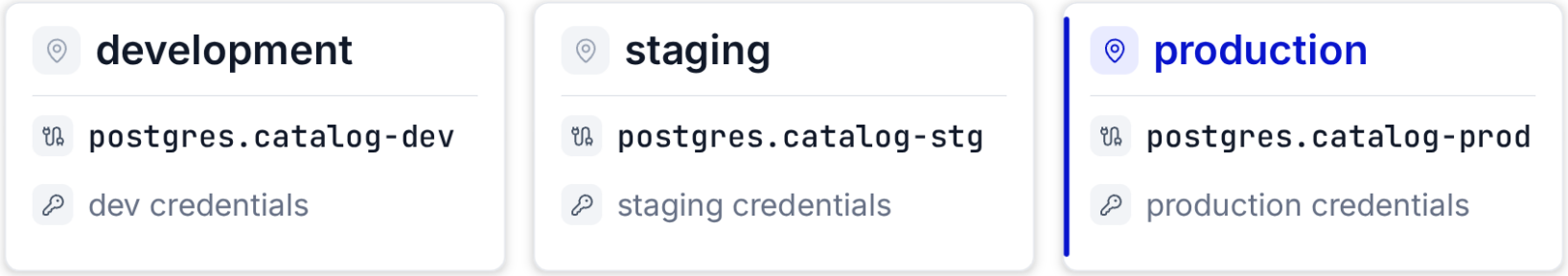
Declare a dependency once

THE DEVELOPER WRITES IT ONCE

workload.yaml

```
dependencies:
  - resource: postgres
```

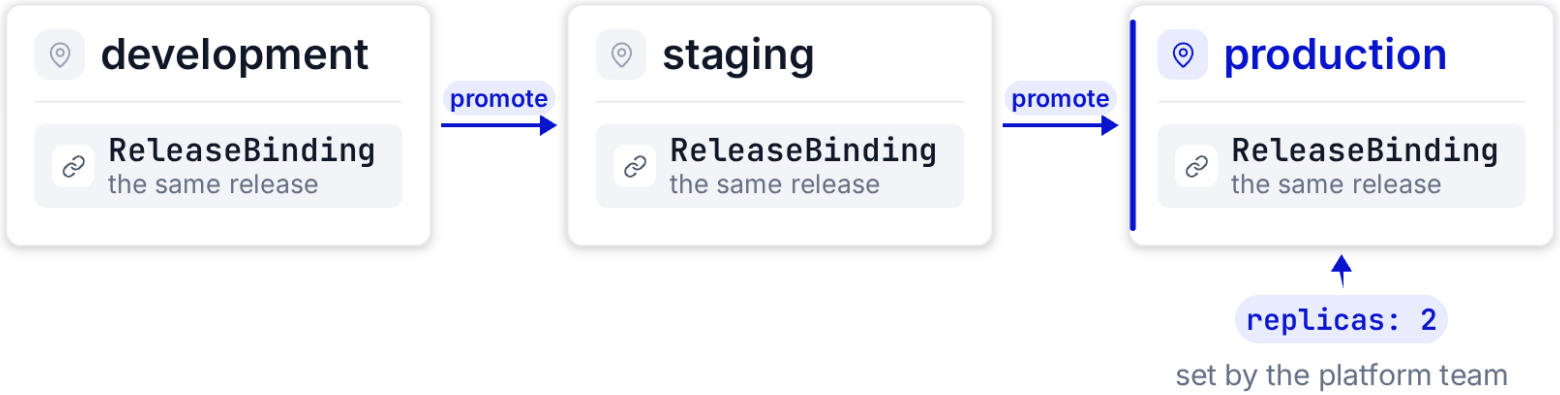
THE PLATFORM SUPPLIES THE REST, PER ENVIRONMENT



No host names in the code. No settings that drift apart between environments.

One approved route to production

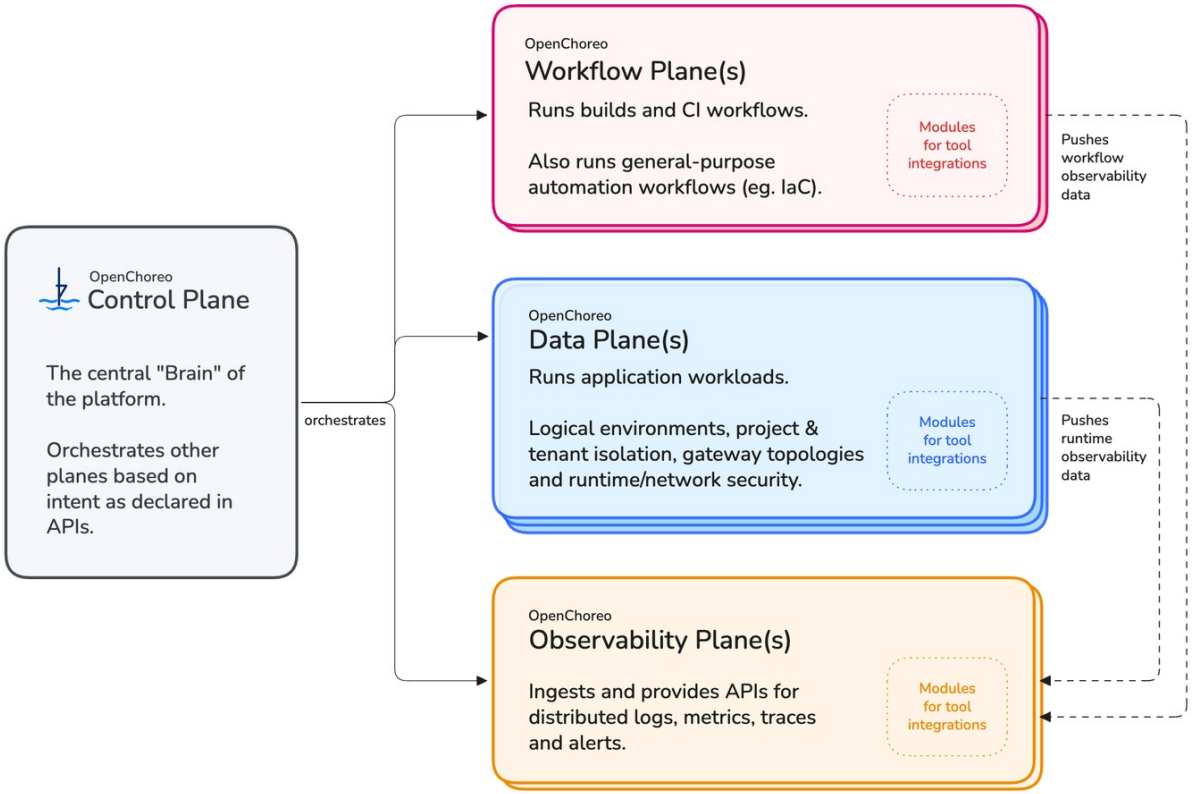
- THE DEPLOYMENT PIPELINE DEFINES THE APPROVED ROUTE



Nobody re-applies files into production. The same release moves.

The per-environment settings sit on the binding — the platform team's half of the contract.

A system, not a stack



What else the platform owns

Where things run

`DataPlane` `WorkflowPlane` `ObservabilityPlane`

Each can be its own cluster — production in its own region.

Who may do what

`AuthzRole` `AuthzRoleBinding`

Roles bound to your identity provider's claims, per project.

Where secrets live

`SecretReference`

Values come from your existing secret store, not OpenChoreo.

What gets watched

`ObservabilityAlertRule` `ObservabilityAlertsNotificationChannel`

Generated by a Trait, so every component gets the same alerts.

How images get built

`Workflow` `WorkflowRun`

The build paths the platform allows; each run is a record.

Where, in what order

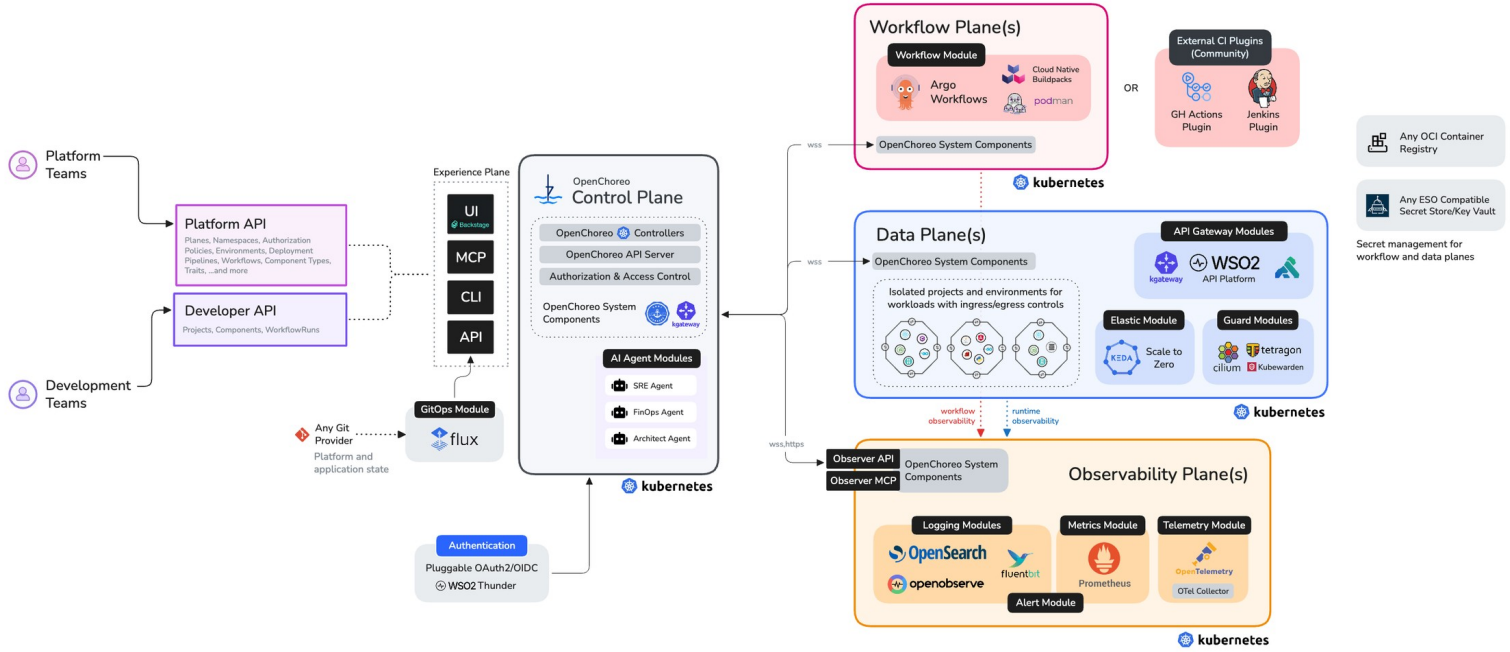
`Environment` `DeploymentPipeline`

The environments, and the approved routes between them.

None of this is developer-facing — it is the platform team's side of the contract.

There is no control-plane kind: the control plane is the thing doing the registering.

Three layers. One system.



You can change the defaults

Types, traits and workflows — all replaceable.

Go deeper

Custom Resources,
Not Custom Forks
Tishan — 11:00, this room

Try it

openchoreo.dev
github.com/openchoreo

Ask

CNCF Slack
#openchoreo

Scan for the docs

openchoreo.dev/docs

A CNCF Sandbox project.

Day 1's lab repo has the full walkthrough.

WSO2CON AFRICA

SEPT 22 – 24, 2026 | NAIROBI, KENYA

Thank You!

Kavith Lokuhewage

Associate Director & Architect