WSO2CON AFRICA

SEPT 22 – 24, 2026 | NAIROBI, KENYA

Custom Resources, Not Custom Forks

Curating OpenChoreo for Your Enterprise

Tishan Dahanayakage
Director & Head of Engineering, Choreo BU, WSO2

Thursday 24 Sept 2026 · 11:00
Yellow Room · 30 min

99

WSO2

Data residency is already law, or nearly, across the continent

Nigeria	CBN directive, effective January 2027: payment data generated in Nigeria stays in Nigeria.
Kenya	Data Protection Act s.50: a serving copy in Kenya for strategic-interest data.
Ghana	Cyber & Information Security Directive 2026: sensitive financial data stays in Ghana.
Egypt	PDPL: the one-year grace period ends 1 November 2026.
South Africa	Reserve Bank consulting on data offshoring; localisation proposals still open.

I'm an engineer, not your compliance officer; check every one of those with your own counsel.

THE ENGINEERING FACT UNDERNEATH ALL OF IT

Nobody ships your regulator.

It can't. And none of it is a failure of the platform you picked.

WSO2

THE ENGINEERING FACT UNDERNEATH ALL OF IT

Nobody ships your data classification scheme.

It can't. And none of it is a failure of the platform you picked.

WSO2

THE ENGINEERING FACT UNDERNEATH ALL OF IT

Nobody ships your change advisory board.

It can't. And none of it is a failure of the platform you picked.

WSO2

THE ENGINEERING FACT UNDERNEATH ALL OF IT

Nobody ships your definition of ready.

It can't. And none of it is a failure of the platform you picked.

WSO2

FOUR THINGS NOBODY SHIPS

Your regulator.

Your data classification scheme.

Your change advisory board.

Your definition of ready.

That layer is the platform you picked, plus you.

The next twenty minutes are about what building it actually costs.

WSO2

A platform is a fabric, not a product

CONTROL PLANE	DATA PLANE	BUILD PLANE	OBSERVABILITY PLANE
Desired-state API	Ingress & routing	Source checkout	Log collection
Reconciliation	Service identity	Image build	Metrics
Catalogue & RBAC	Secret material	Artifact registry	Traces
Developer portal	Workload runtime	Attestation & signing	Query & dashboards

The platform is not any of those capabilities. The platform is the contracts between them.

Every one of those boxes is a commodity you can already buy or already run. What takes eighteen months is the wiring.

Nothing in that fabric has a shape for any of these

CONTROL PLANE	DATA PLANE	BUILD PLANE	OBSERVABILITY PLANE
Desired-state API	Ingress & routing	Source checkout	Log collection
Reconciliation	Service identity	Image build	Metrics
Catalogue & RBAC	Secret material	Artifact registry	Traces
Developer portal	Workload runtime	Attestation & signing	Query & dashboards

FOUR THINGS NOBODY SHIPS

Your regulator.

Your data classification scheme.

Your change advisory board.

Your definition of ready.

None of them is a box. Every one of them is a contract, and the contracts are the part you author.

Three ways to close the gap

	OPTION	WHAT IT IS	WHAT IT COSTS YOU
	**Fork it**	Patch the platform's source to do what you need.	You now own a codebase, its CVEs, and a rebase you'll stop doing.
	**Route around it**	A team deploys outside the platform to get the thing they need.	You lose visibility, governance and the audit trail: exactly what you needed.
	**Author it inside**	Add the capability as a first-class, supported object.	Only viable if authoring is cheap.

Why option three used to lose

Extending a platform meant writing code *inside* it: in the platform's own language, against its internal APIs, compiled or deployed as part of it. And then owning that code for as long as you ran the platform.

BACKSTAGE · TODAY

Adding one plugin drags its whole dependency tree and rebuilds the entire portal.

You maintain a fork, or you wait.

JENKINS · 20 YEARS

1,600+ plugins. 140 supported. Extension was cheap and uncontracted.

Every instance became a snowflake. Every snowflake was somebody's reasonable Tuesday.

SAP · 30 YEARS

Customization meant modifying the core.

Every upgrade meant auditing every line. Companies sat on old versions for a decade.

Different mechanisms. Same outcome: the customization ended up outside the upgrade path.

An entire industry already ran this experiment

SAP's Clean Core model grades every extension A to D.

A

Built only on publicly released, stability-contracted interfaces.

B

Classic APIs. Documented, generally upgrade-stable.

C

Reaches into internal objects. No contract behind it.

D

Modification, direct table writes, implicit enhancements.

Twenty years and a great deal of ERP money went into learning that distinction.

SAP calls grade D “the highest risk” and says such extensions “create significant technical debt.”

A custom resource is a versioned API

```
apiVersion: openchoreo.dev/v1alpha1
kind: ComponentType
```

Group, version, kind: in every object you apply.

A named kind

The API server validates it at admission. kubectl, RBAC and audit work day one.

A version

v1alpha1 to v1beta1 to v1, with declared conversion. Deprecation on a schedule, not in a patch release.

A contract

The schema is the interface, not the implementation behind it: grade A, in SAP’s terms.

A ConfigMap holds strings. A custom resource holds a schema you can version, validate and deprecate.

Argo CD’s lead maintainer is now proposing to move the project’s own configuration out of **ConfigMaps** , CLI flags and environment variables and into a single **CRD**. His word for the current state: “messy.”

Michael Crenshaw, “Configuring Argo CD is Messy: Here’s How We Can Fix It” · ArgoCon Japan 2026

WHAT CHANGED

Almost everything in OpenChoreo is a custom resource.

UNTIL RECENTLY · WRITE A CONTROLLER

- Reconcile loop
- CRD plumbing
- Go codebase
- Container image
- Release process
- On-call rotation

An operator that is now yours to build, ship and run.

NOW · APPLY A RESOURCE

```
kind: ClusterComponentType
spec:
  parameters: { ...schema }
  resources:  [ ...templates ]   # CEL
```

Declarative. CEL-templated. No controller. No image. No release of your own.

Custom resources, not custom forks.

The shipped set: **github.com/openchoreo/openchoreo** → config/crd/bases

> *"Platform engineers can define abstractions without writing Kubernetes controllers."*
>
> **InfoQ, on OpenChoreo 1.0**

Curating the platform: the catalogue

Five additions. Fourteen months. One payments company operating in Kenya, Nigeria and Ghana.

0	0	0	0
ComponentTypes	**Traits**	**Workflows**	**ResourceTypes**

The descending cost of each addition is the argument.

Addition 1: the house standard

MONTH 1 · A DAY · ClusterComponentType

“Every service is HTTP, autoscaled, behind the gateway, with a disruption budget.”

ONE DECLARATION RENDERS FIVE RESOURCES

payment-service
ClusterComponentType

- **Deployment**
- **Service**
- **HTTPRoute**
- **PodDisruptionBudget**
- **HorizontalPodAutoscaler**

TWO KNOBS EXPOSED. EVERYTHING ELSE IS FIXED.

replicas	port	autoscaling
2 (min 2)	8080	always on

```yaml
kind: ClusterComponentType
metadata:
  name: payment-service
spec:
  parameters:
    openAPIV3Schema:
      properties:
        replicas:
          { type: integer, default: 2, minimum: 2 }
        port: { type: integer, default: 8080 }
  preRenderValidations:
    - rule: ${parameters.replicas >= 2}
      message: "must run 2+ replicas"
  resources:
    - id: deployment   # apps/v1 Deployment
    - id: service      # v1 Service
    - id: httproute    # gateway.networking.k8s.io/v1
    - id: pdb          # policy/v1 PodDisruptionBudget
    - id: hpa          # autoscaling/v2 HPA
```

CATALOGUE | **1** ComponentTypes **0** Traits **0** Workflows **0** ResourceTypes

Addition 1: the house standard

MONTH 1 · A DAY · ClusterComponentType

Two knobs, and `minimum: 2` makes one replica unaskable.

```yaml
kind: ClusterComponentType
metadata:
  name: payment-service
spec:
  parameters:
    openAPIV3Schema:
      properties:
        replicas:
          { type: integer, default: 2, minimum: 2 }
        port: { type: integer, default: 8080 }
  preRenderValidations:
    - rule: ${parameters.replicas >= 2}
      message: "must run 2+ replicas"
  resources:
    - id: deployment   # apps/v1 Deployment
    - id: service      # v1 Service
    - id: httproute    # gateway.networking.k8s.io/v1
    - id: pdb          # policy/v1 PodDisruptionBudget
    - id: hpa          # autoscaling/v2 HPA
```

CATALOGUE | 1 ComponentTypes | 0 Traits | 0 Workflows | 0 ResourceTypes

Addition 1: the house standard

MONTH 1 · A DAY · ClusterComponentType

Validation at author time, not at review time.

```
kind: ClusterComponentType
metadata:
  name: payment-service
spec:
  parameters:
    openAPIV3Schema:
      properties:
        replicas:
          { type: integer, default: 2, minimum: 2 }
        port: { type: integer, default: 8080 }
  preRenderValidations:
    - rule: ${parameters.replicas >= 2}
      message: "must run 2+ replicas"
  resources:
    - id: deployment   # apps/v1 Deployment
    - id: service      # v1 Service
    - id: httproute    # gateway.networking.k8s.io/v1
    - id: pdb          # policy/v1 PodDisruptionBudget
    - id: hpa          # autoscaling/v2 HPA
```

CATALOGUE | **1** ComponentTypes **0** Traits **0** Workflows **0** ResourceTypes

Addition 1: the house standard

MONTH 1 · A DAY · ClusterComponentType

```
kind: ClusterComponentType
metadata:
  name: payment-service
spec:
  parameters:
    openAPIV3Schema:
      properties:
        replicas:
          { type: integer, default: 2, minimum: 2 }
        port: { type: integer, default: 8080 }
  preRenderValidations:
    - rule: ${parameters.replicas >= 2}
      message: "must run 2+ replicas"
  resources:
    - id: deployment   # apps/v1 Deployment
    - id: service      # v1 Service
    - id: httproute    # gateway.networking.k8s.io/v1
    - id: pdb          # policy/v1 PodDisruptionBudget
    - id: hpa          # autoscaling/v2 HPA
```

One declaration renders five resources.
Autoscaling is not optional; it is rendered.

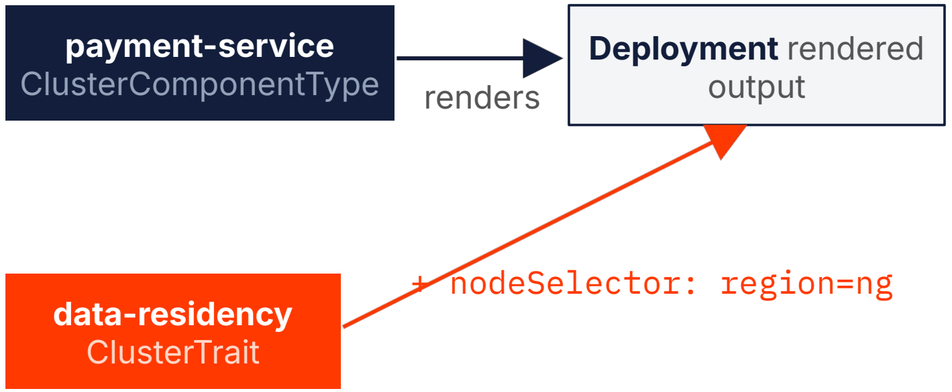
CATALOGUE | **1** ComponentTypes **0** Traits **0** Workflows **0** ResourceTypes

Addition 2: the regulator's rule

MONTH 2 · AN AFTERNOON · ClusterTrait

“Payment data generated in Nigeria stays in Nigeria.”

A TRAIT PATCHES WHAT IT DIDN’T CREATE



The component type is never opened. A post-render rule then makes the wrong data plane unrenderable.

```
kind: ClusterTrait
metadata:
  name: data-residency
spec:
  parameters:
    openAPIV3Schema:
      properties:
        jurisdiction: { enum: [ng, ke, gh, za] }
  patches:
    - target: { group: apps, kind: Deployment }
      operations:
        - op: add
          path: /spec/template/spec/nodeSelector
          value:
            topology.kubernetes.io/region:
              ${parameters.jurisdiction}
  preRenderValidations:
    - when: ${parameters.jurisdiction == 'ng'}
      rule: ${metadata.dataPlaneName.startsWith('ng-')}
      message: >-
        Nigerian payment data may only render to a
        Nigerian data plane (CBN, effective 2027-01-01)
```

CATALOGUE | 1 ComponentTypes | 1 **Traits** | 0 Workflows | 0 ResourceTypes

Addition 2: the regulator's rule

MONTH 2 · AN AFTERNOON · ClusterTrait

It patches a Deployment it did not create.

```
kind: ClusterTrait
metadata:
  name: data-residency
spec:
  parameters:
    openAPIV3Schema:
      properties:
        jurisdiction: { enum: [ng, ke, gh, za] }
  patches:
    - target: { group: apps, kind: Deployment }
      operations:
        - op: add
          path: /spec/template/spec/nodeSelector
          value:
            topology.kubernetes.io/region:
              ${parameters.jurisdiction}
  preRenderValidations:
    - when: ${parameters.jurisdiction == 'ng'}
      rule: ${metadata.dataPlaneName.startsWith('ng-')}
      message: >-
        Nigerian payment data may only render to a
        Nigerian data plane (CBN, effective 2027-01-01)
```

Addition 2: the regulator's rule

MONTH 2 · AN AFTERNOON · ClusterTrait

Residency becomes a nodeSelector. The component type never changes.

```
kind: ClusterTrait
metadata:
  name: data-residency
spec:
  parameters:
    openAPIV3Schema:
      properties:
        jurisdiction: { enum: [ng, ke, gh, za] }
  patches:
    - target: { group: apps, kind: Deployment }
      operations:
        - op: add
          path: /spec/template/spec/nodeSelector
          value:
            topology.kubernetes.io/region:
              ${parameters.jurisdiction}
  preRenderValidations:
    - when: ${parameters.jurisdiction == 'ng'}
      rule: ${metadata.dataPlaneName.startsWith('ng-')}
      message: >-
        Nigerian payment data may only render to a
        Nigerian data plane (CBN, effective 2027-01-01)
```

Addition 2: the regulator's rule

MONTH 2 · AN AFTERNOON · ClusterTrait

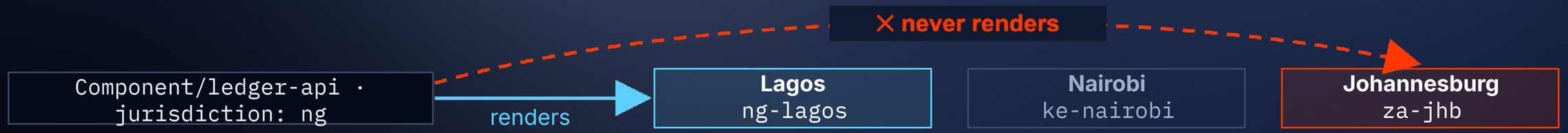
```
kind: ClusterTrait
metadata:
  name: data-residency
spec:
  parameters:
    openAPIV3Schema:
      properties:
        jurisdiction: { enum: [ng, ke, gh, za] }
  patches:
    - target: { group: apps, kind: Deployment }
      operations:
        - op: add
          path: /spec/template/spec/nodeSelector
          value:
            topology.kubernetes.io/region:
              ${parameters.jurisdiction}
  preRenderValidations:
    - when: ${parameters.jurisdiction == 'ng'}
      rule: ${metadata.dataPlaneName.startsWith('ng-')}
      message: >-
        Nigerian payment data may only render to a
        Nigerian data plane (CBN, effective 2027-01-01)
```

The wrong data plane stops being expressible.

CATALOGUE | 1 ComponentTypes | 1 **Traits** | 0 Workflows | 0 ResourceTypes

RETARGET THE SAME COMPONENT AT THE JOHANNESBURG DATA PLANE

It never renders.




```
$ occ apply -f release-za.yaml

Error: render failed for Component/ledger-api
  postRenderValidation "data-residency" failed:
  Nigerian payment data may only render to a Nigerian data plane
  (CBN, effective 2027-01-01)
```

It doesn't fail health checks.

It doesn't get rolled back.

It was never expressible.

The component type from month one, unchanged

```
$ git diff clustercomponenttype-payment-service.yaml
$
```

The regulator changed my platform. It did not change my component type.

Custom resources, not custom forks.

CATALOGUE	1 ComponentTypes	1 **Traits**	0 Workflows	0 ResourceTypes

Addition 3: governing the pipeline

MONTH 3 · AN HOUR · ClusterWorkflow

“Nothing reaches production without an API-spec lint and a signed SBOM.”

The runTemplate is composition, not a pipeline.

Each step is a reference to a template somebody authored once. The addition is “compose five steps”.

```yaml
kind: ClusterWorkflow
metadata:
  name: governed-build
spec:
  ttlAfterCompletion: "30d"    # audit retention
  parameters:
    openAPIV3Schema:
      properties:
        failOn: { enum: [critical, high, medium] }
  runTemplate:                 # Argo Workflow
    spec:
      templates:
        - name: governed-build
          steps:
            - checkout-source      # platform
            - spectral-lint        # platform
            - containerfile-build  # platform
            - publish-image        # platform
            - sbom-and-sign        # yours
```

CATALOGUE | 1 ComponentTypes | 1 Traits | 1 **Workflows** | 0 ResourceTypes

Addition 3: governing the pipeline

MONTH 3 · AN HOUR · ClusterWorkflow

```yaml
kind: ClusterWorkflow
metadata:
  name: governed-build
spec:
  ttlAfterCompletion: "30d"    # audit retention
  parameters:
    openAPIV3Schema:
      properties:
        failOn: { enum: [critical, high, medium] }
  runTemplate:                 # Argo Workflow
    spec:
      templates:
        - name: governed-build
          steps:
            - checkout-source      # platform
            - spectral-lint        # platform
            - containerfile-build  # platform
            - publish-image        # platform
            - sbom-and-sign        # yours
```

Four of the five steps already ship with the platform.

CATALOGUE | 1 ComponentTypes | 1 Traits | 1 **Workflows** | 0 ResourceTypes

The gate is two lines on the component type

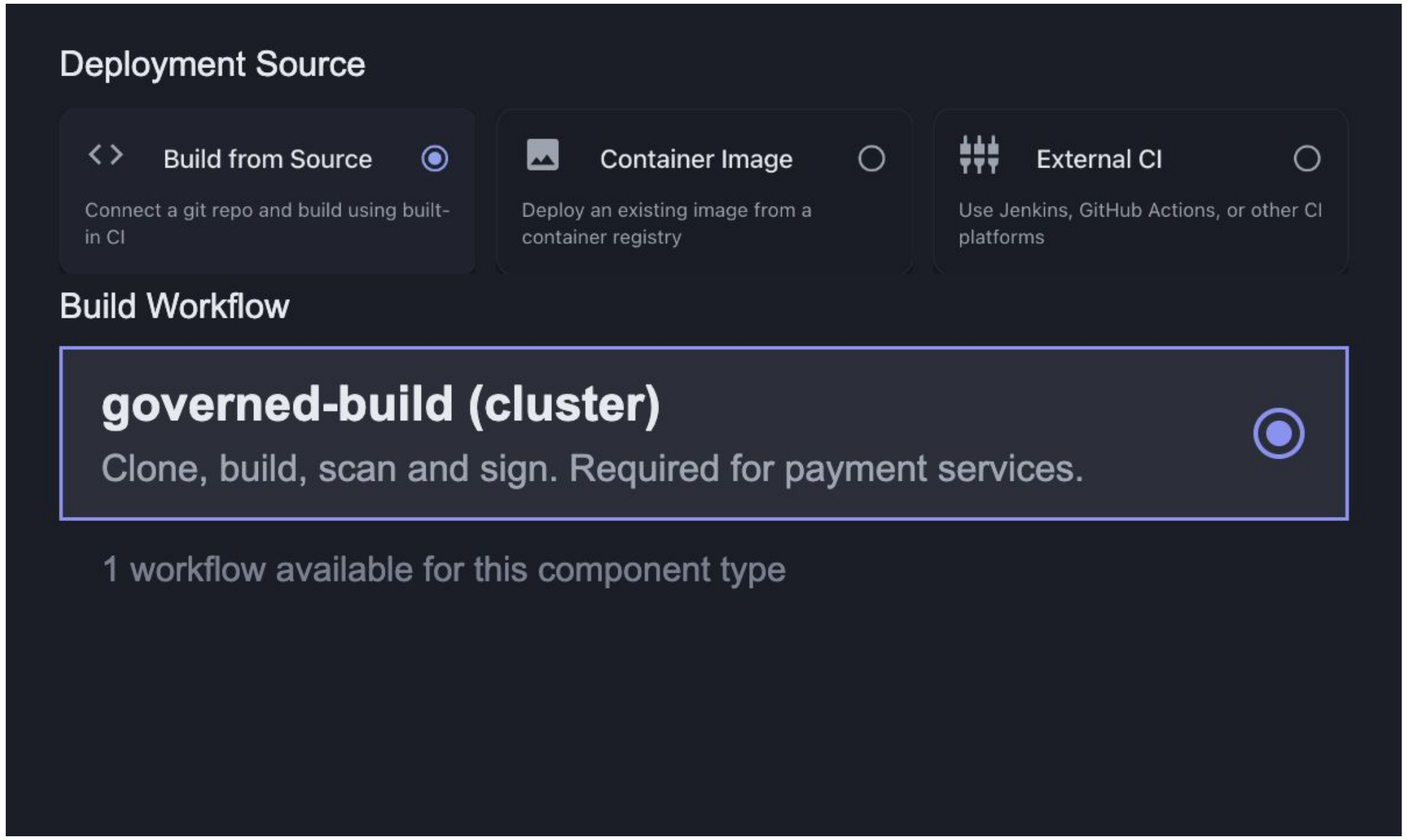
```
# ClusterComponentType/payment-service
spec:
  allowedWorkflows:
    - kind: ClusterWorkflow
      name: governed-build
```

Not a pipeline template a team copied into their repo and can quietly edit. Not a policy document. On the type.

The guardrail isn't enforced at review time. It's enforced at the point where the option would have appeared.

CATALOGUE | 1 ComponentTypes | 1 Traits | 1 **Workflows** | 0 ResourceTypes

What the developer sees



OPENCHOREO PORTAL · CREATE SERVICE · BUILD & DEPLOY

If you are a payment service, this is the build that exists for you.

The `x-openchoreo-component-parameter-*` extensions bind schema fields to the component's own repository settings, so the developer never retypes a Git URL.
Schema drives form.

CATALOGUE | 1 ComponentTypes | 1 Traits | 1 **Workflows** | 0 ResourceTypes

Addition 4: infrastructure the platform doesn't ship

MONTH 5 · 20 MINUTES · ClusterResourceType

“Payments needs a Postgres. In Nigeria. Self-service.”

It provisions nothing itself. It governs a claim.

My infrastructure team already runs the provisioner, a Crossplane XRD, in our case. OpenChoreo doesn't ship a production database provisioner and doesn't need to: a ResourceType template emits whatever Kubernetes resources you point it at.

```
kind: ClusterResourceType
metadata:
  name: postgres-regulated
spec:
  parameters:
    openAPIV3Schema:
      properties:
        version:      { enum: ["15", "16"] }
        jurisdiction: { enum: [ng, ke, gh, za] }
  environmentConfigs:        # regulatory, per-env
    backupRetentionDays: { default: 35 }
  retainPolicy: Retain       # never auto-deleted
  outputs:
    - { name: host, value: "${applied.claim.host}" }
    - { name: port, value: "5432" }
    - name: password         # never in the clear
      secretKeyRef: { name: "${metadata.name}-conn" }
  resources:
    - id: claim              # the infra team's own XRD
      template:
        apiVersion: database.bank.internal/v1alpha1
        kind: PostgreSQLInstance
        metadata: { name: ${metadata.name} }
        spec: { region: ${parameters.jurisdiction} }
```

CATALOGUE | 1 ComponentTypes | 1 Traits | 1 Workflows | 1 **ResourceTypes**

Addition 4: infrastructure the platform doesn't ship

MONTH 5 · 20 MINUTES · ClusterResourceType

Retention and retain policy are the regulator's, not the developer's.

```
kind: ClusterResourceType
metadata:
  name: postgres-regulated
spec:
  parameters:
    openAPIV3Schema:
      properties:
        version:      { enum: ["15", "16"] }
        jurisdiction: { enum: [ng, ke, gh, za] }
  environmentConfigs:        # regulatory, per-env
    backupRetentionDays: { default: 35 }
  retainPolicy: Retain       # never auto-deleted
  outputs:
    - { name: host, value: "${applied.claim.host}" }
    - { name: port, value: "5432" }
    - name: password         # never in the clear
      secretKeyRef: { name: "${metadata.name}-conn" }
  resources:
    - id: claim              # the infra team's own XRD
      template:
        apiVersion: database.bank.internal/v1alpha1
        kind: PostgreSQLInstance
        metadata: { name: ${metadata.name} }
        spec: { region: ${parameters.jurisdiction} }
```

CATALOGUE | 1 ComponentTypes | 1 Traits | 1 Workflows | 1 **ResourceTypes**

Addition 4: infrastructure the platform doesn't ship

MONTH 5 · 20 MINUTES · ClusterResourceType

Only the reference travels. The credential stays in the data plane.

```
kind: ClusterResourceType
metadata:
  name: postgres-regulated
spec:
  parameters:
    openAPIV3Schema:
      properties:
        version:      { enum: ["15", "16"] }
        jurisdiction: { enum: [ng, ke, gh, za] }
  environmentConfigs:        # regulatory, per-env
    backupRetentionDays: { default: 35 }
  retainPolicy: Retain       # never auto-deleted
  outputs:
    - { name: host, value: "${applied.claim.host}" }
    - { name: port, value: "5432" }
    - name: password         # never in the clear
      secretKeyRef: { name: "${metadata.name}-conn" }
  resources:
    - id: claim              # the infra team's own XRD
      template:
        apiVersion: database.bank.internal/v1alpha1
        kind: PostgreSQLInstance
        metadata: { name: ${metadata.name} }
        spec: { region: ${parameters.jurisdiction} }
```

CATALOGUE | 1 ComponentTypes | 1 Traits | 1 Workflows | 1 **ResourceTypes**

Addition 4: infrastructure the platform doesn't ship

MONTH 5 · 20 MINUTES · ClusterResourceType

```
kind: ClusterResourceType
metadata:
  name: postgres-regulated
spec:
  parameters:
    openAPIV3Schema:
      properties:
        version:      { enum: ["15", "16"] }
        jurisdiction: { enum: [ng, ke, gh, za] }
  environmentConfigs:        # regulatory, per-env
    backupRetentionDays: { default: 35 }
  retainPolicy: Retain       # never auto-deleted
  outputs:
    - { name: host, value: "${applied.claim.host}" }
    - { name: port, value: "5432" }
    - name: password         # never in the clear
      secretKeyRef: { name: "${metadata.name}-conn" }
  resources:
    - id: claim              # the infra team's own XRD
      template:
        apiVersion: database.bank.internal/v1alpha1
        kind: PostgreSQLInstance
        metadata: { name: ${metadata.name} }
        spec: { region: ${parameters.jurisdiction} }
```

It provisions nothing. It governs somebody else's XRD.

CATALOGUE | 1 ComponentTypes | 1 Traits | 1 Workflows | 1 **ResourceTypes**

The developer side is four lines

```yaml
kind: Resource
metadata:
  name: ledger-db
spec:
  owner: { projectName: payments }
  type:
    kind: ClusterResourceType
    name: postgres-regulated
  parameters:
    version: "16"
    jurisdiction: ng
```

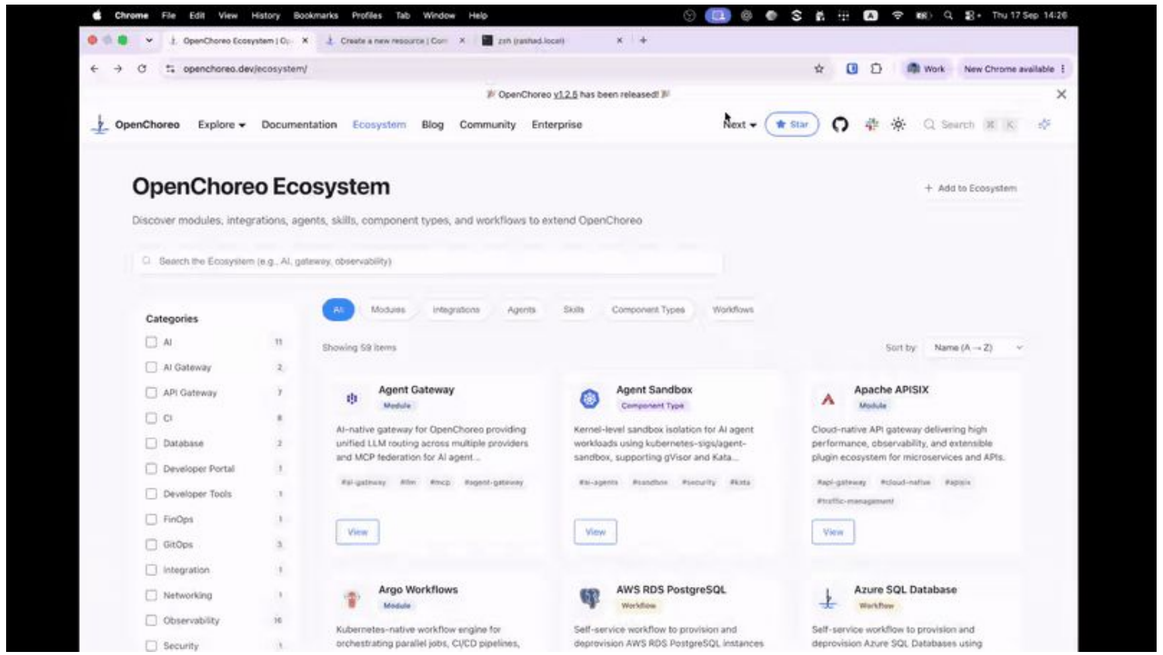
```yaml
# Workload
spec:
  dependencies:
    resources:
      - ref: ledger-db
        envBindings:
          host:     PGHOST
          port:     PGPORT
          username: PGUSER
          password: PGPASSWORD
```

No database operator was written. No ticket was filed. The catalogue answered.

CATALOGUE | 1 ComponentTypes | 1 Traits | 1 Workflows | 1 **ResourceTypes**

Addition 5: someone else's addition

MONTH 14 · ONE COMMAND · COMMUNITY MODULE



"The integration team wants WSO2 Micro Integrator on the platform."

One command. Counter goes up. Nothing was authored.

CATALOGUE	2 ComponentTypes	1 Traits	1 Workflows	1 ResourceTypes

FOUR THINGS NOBODY SHIPS

Your regulator. — `ClusterTrait` **an afternoon**

Your data classification scheme. — `ClusterResourceType` **20 minutes**

Your change advisory board. — `ClusterWorkflow` **an hour**

Your definition of ready. — `ClusterComponentType` **a day**

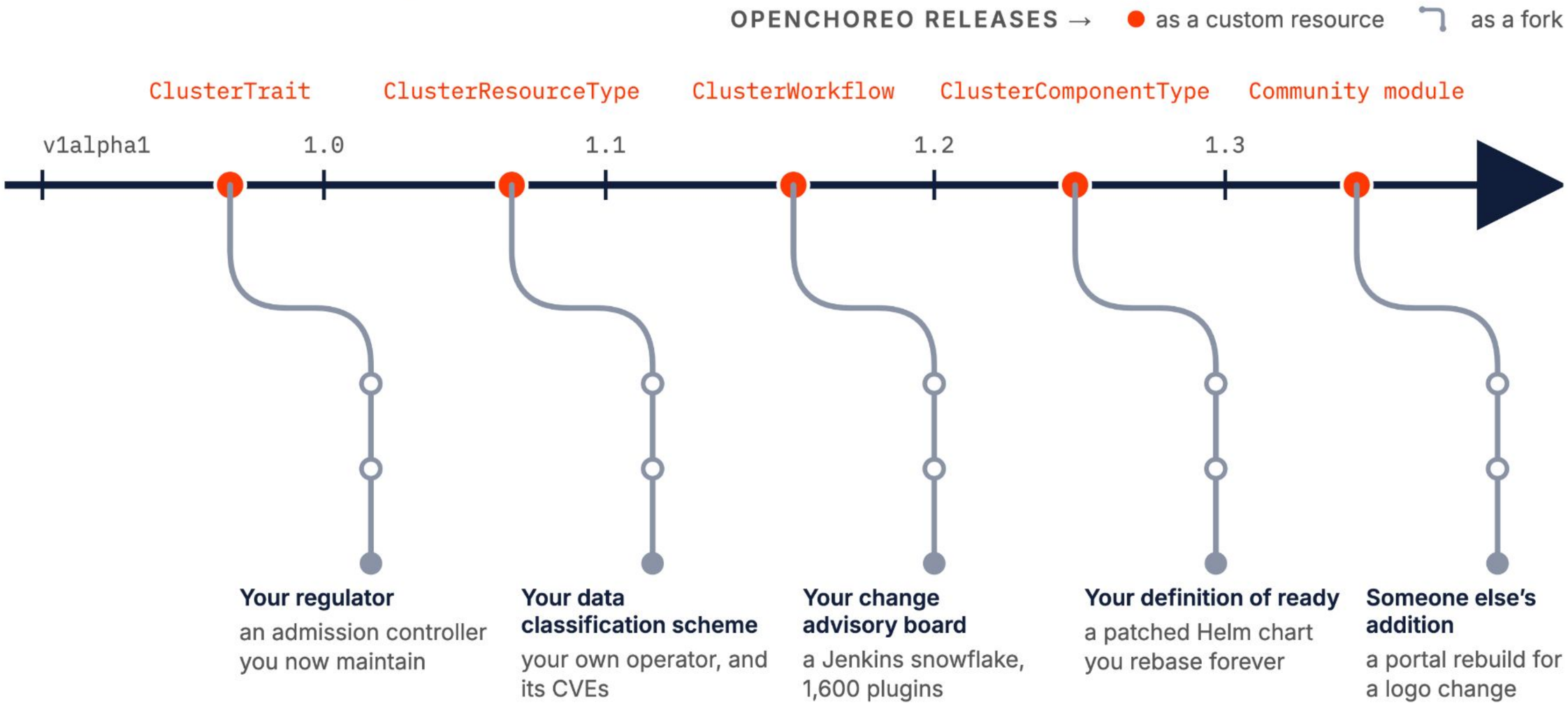
And someone else's. — `Community module` **one command**

Shipped. All five. Fourteen months, no fork.

Zero controllers written, zero portal rebuilds. By SAP's rubric, all five are grade A.

WSO2

The same five, as forks



A custom resource rides the trunk. A fork leaves it the day it's written and never catches up.

KUBERNETES ARCHITECTURE DESIGN DOCUMENT

“We want others to be able to extend Kubernetes functionality … without modification of core Kubernetes source. Therefore, its API isn’t just targeted at end users, but at tool and extension developers.”

OpenChoreo is that sentence, one layer up.

Custom resources, not custom forks.

WSO2CON AFRICA

SEPT 22 – 24, 2026 | NAIROBI, KENYA

Thank You!

Curate it. The next addition is the only number that matters.

Tishan Dahanayakage
Director & Head of Engineering, Choreo BU, WSO2