



KubeCon



CloudNativeCon

Europe 2026

#KubeCon #CloudNativeCon

Dynamic Cluster Wide Quotas with kro and Kyverno



Introductions

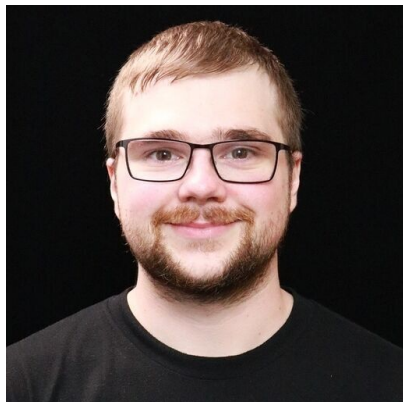


KubeCon



CloudNativeCon

Europe 2026



Adam Crowder



Mariam Fahmy

You run a multi-team Kubernetes cluster



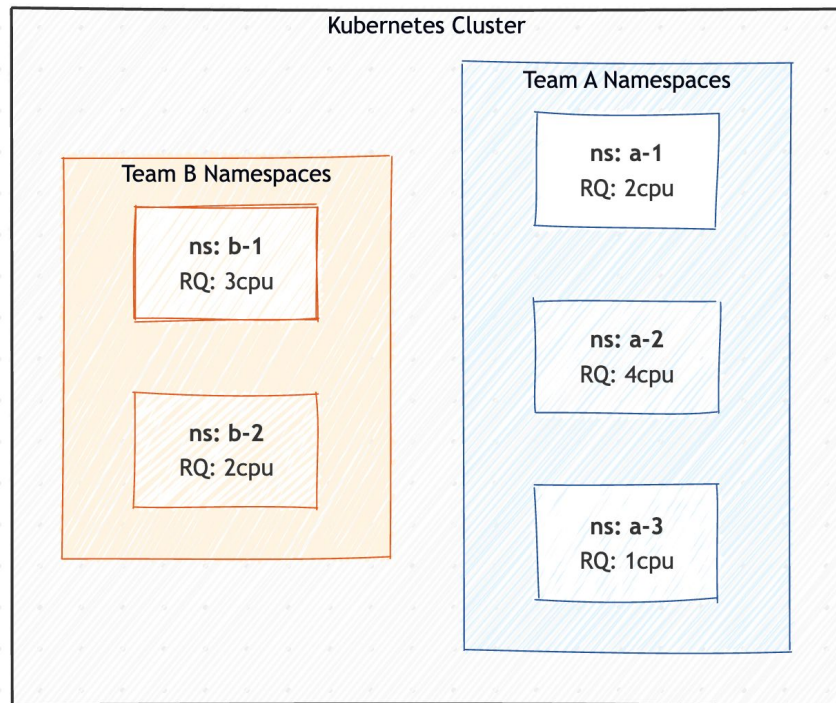
KubeCon



CloudNativeCon

Europe 2026

Each team owns multiple namespaces, each with its own ResourceQuota



Challenge #1: No Cluster-Wide Visibility



KubeCon

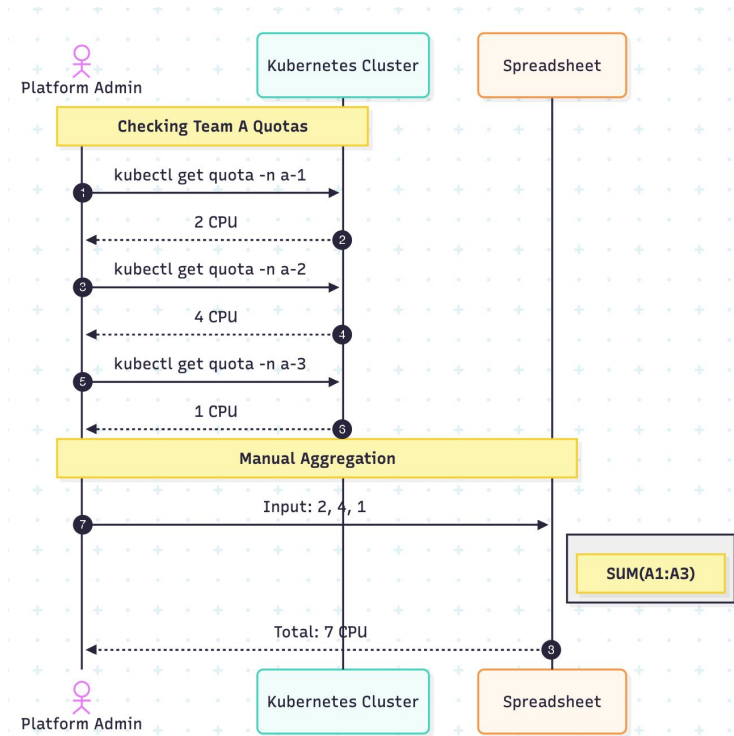


CloudNativeCon

Europe 2026

How much CPU is Team A using across all namespaces?

- No single object aggregates quotas across namespaces
- Kubernetes ResourceQuota is namespace-scoped by design
- Manual scripts break when namespaces are added or removed



Challenge #2: No Budget Enforcement



KubeCon

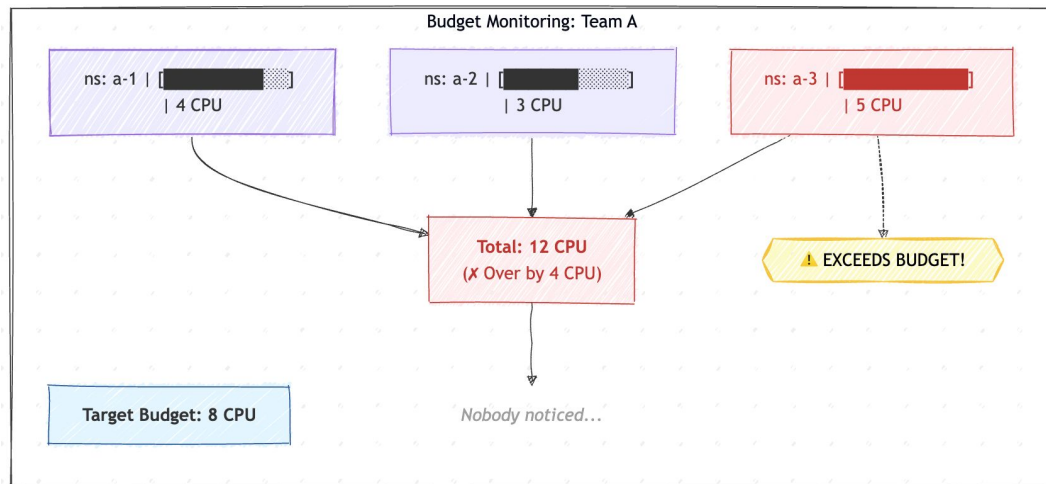


CloudNativeCon

Europe 2026

Team A was allocated 8 CPU.
They just requested 12.

- ResourceQuota only enforces within a single namespace.
- Nothing stops a team from creating a new namespace and requesting more.
- Leads to overprovisioning and noisy-neighbor issues



Challenge #3: Operational Toil



KubeCon

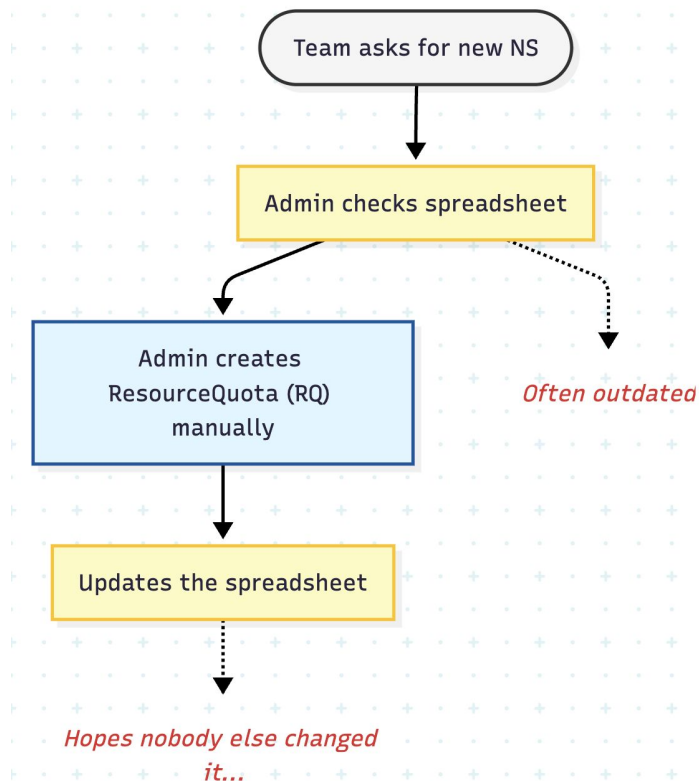


CloudNativeCon

Europe 2026

What platform teams do today:

- Manual tracking via spreadsheets, wikis, or tickets.
- Drift between actual and tracked state.
- Doesn't scale beyond a handful of teams.



What We Actually Need



KubeCon



CloudNativeCon

Europe 2026



Visibility: One object that shows a team's total resource usage across all namespaces, updated in real time



Enforcement: Reject any ResourceQuota that would push a team over their budget, at admission time



Automation: No manual steps, no scripts, no spreadsheets. Just Kubernetes-native resources

The Tools



KubeCon



CloudNativeCon

Europe 2026

Tool	The Role	The Superpower
kro	The Brain	- Define custom APIs using ResourceGraphDefinition - Manages child resources as a single unit - One kubectl apply provisions the entire stack per team
Kyverno	The Enforcer	- Kubernetes-native policy engine - ValidatingPolicy intercepts the admission request - Deny requests that would exceed a team's cluster-wide budget

What is kro?



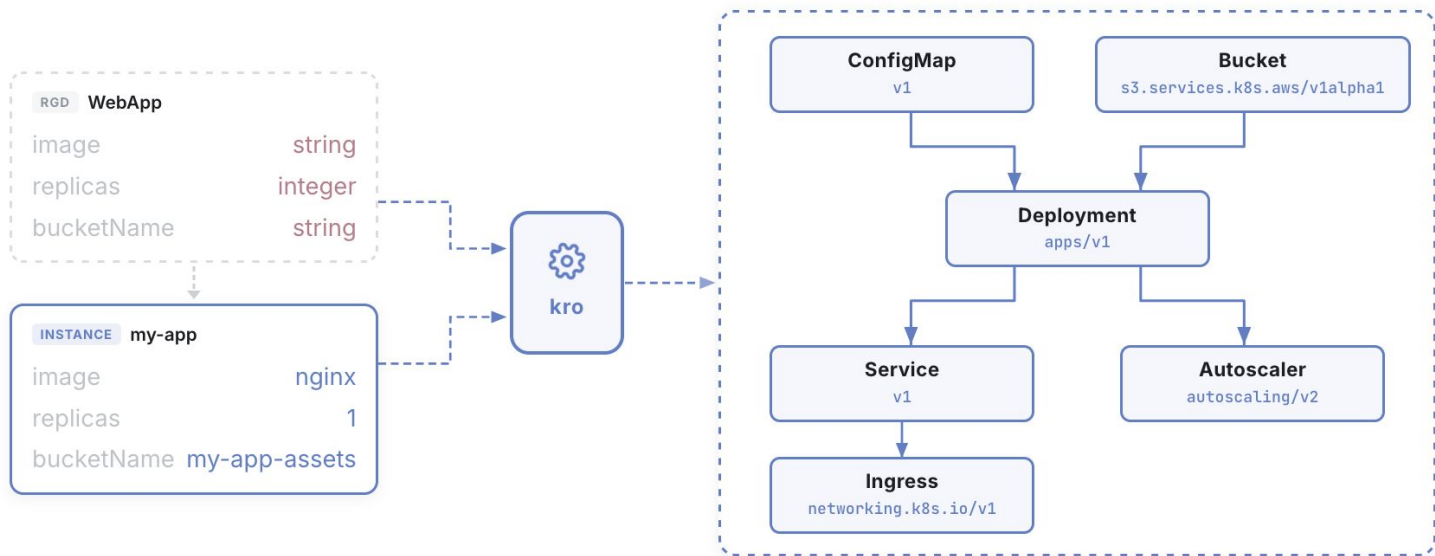
KubeCon



CloudNativeCon

Europe 2026

- Extensible kubernetes operator for kubernetes resources
- Define new, re-usable APIs [CRDs] with YAML and CEL (as an RGD)
- kro automatically reconciles resources for all defined RGDs



What is Kyverno?



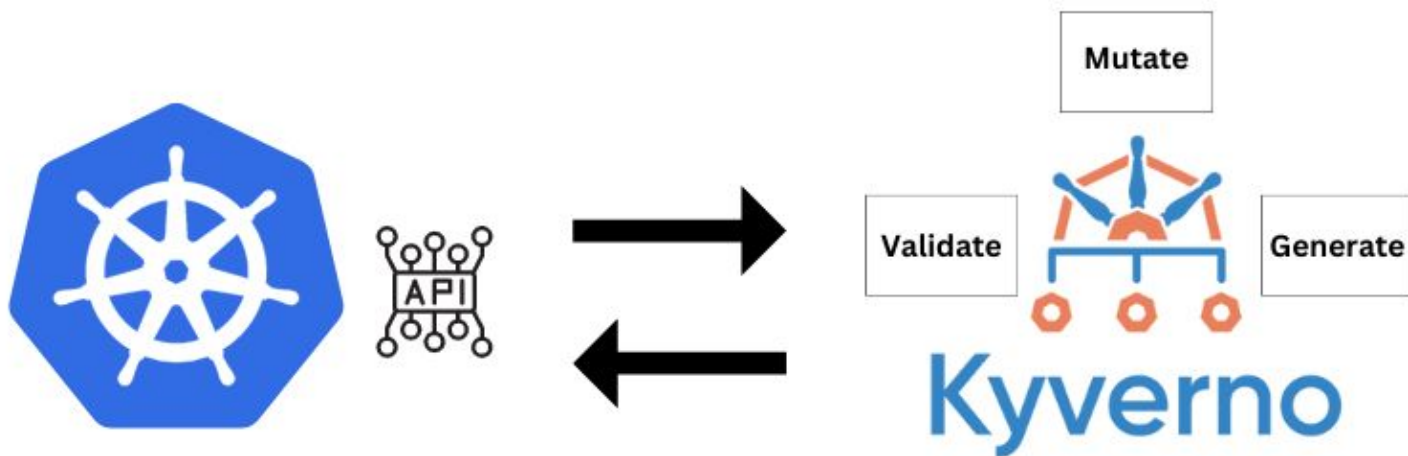
KubeCon



CloudNativeCon

Europe 2026

- Extensible policy engine for kubernetes
- Define kubernetes policies (i.e. validation) with YAML and CEL
- Automatic enforcement in kubernetes via AdmissionWebhooks





KubeCon



CloudNativeCon

Europe 2026

Building Cluster-Wide Quotas — Step by Step



The Architecture



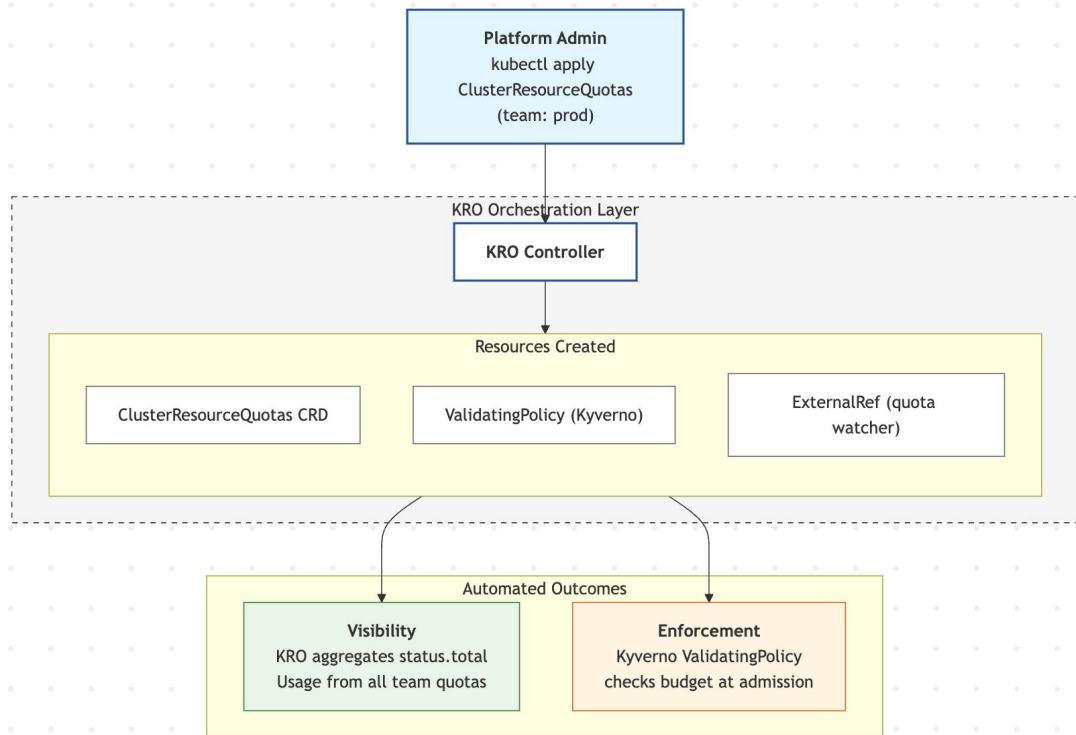
KubeCon



CloudNativeCon

Europe 2026

One *kubectl apply* provisions the entire stack for a team.



Step 1: The ResourceGraphDefinition



KubeCon



CloudNativeCon

Europe 2026

kro to define your custom resource:

- *spec.teamName* →

parameterizes everything per team

- *spec.budget* → the hard cap for the team

- *status.totalUsage* →

computed live by kro from actual ResourceQuotas

```
rgd.yaml

apiVersion: kro.run/v1alpha1
kind: ResourceGraphDefinition
metadata:
  name: cluster-resource-quotas.kro.run
spec:
  schema:
    apiVersion: v1alpha1
    kind: ClusterResourceQuotas
    spec:
      teamName: string
      budget:
        requests.cpu: string
        requests.memory: string
        limits.cpu: string
        limits.memory: string
    status:
      totalUsage:
        requests.cpu: ${teamQuotas.items.map(q, int(q.spec.hard['requests.cpu'])).sum()}
        requests.memory: ${teamQuotas.items.map(q, int(q.spec.hard['requests.memory'])).sum()}
        limits.cpu: ${...}
        limits.memory: ${...}
```

Step 1: The Managed Resources

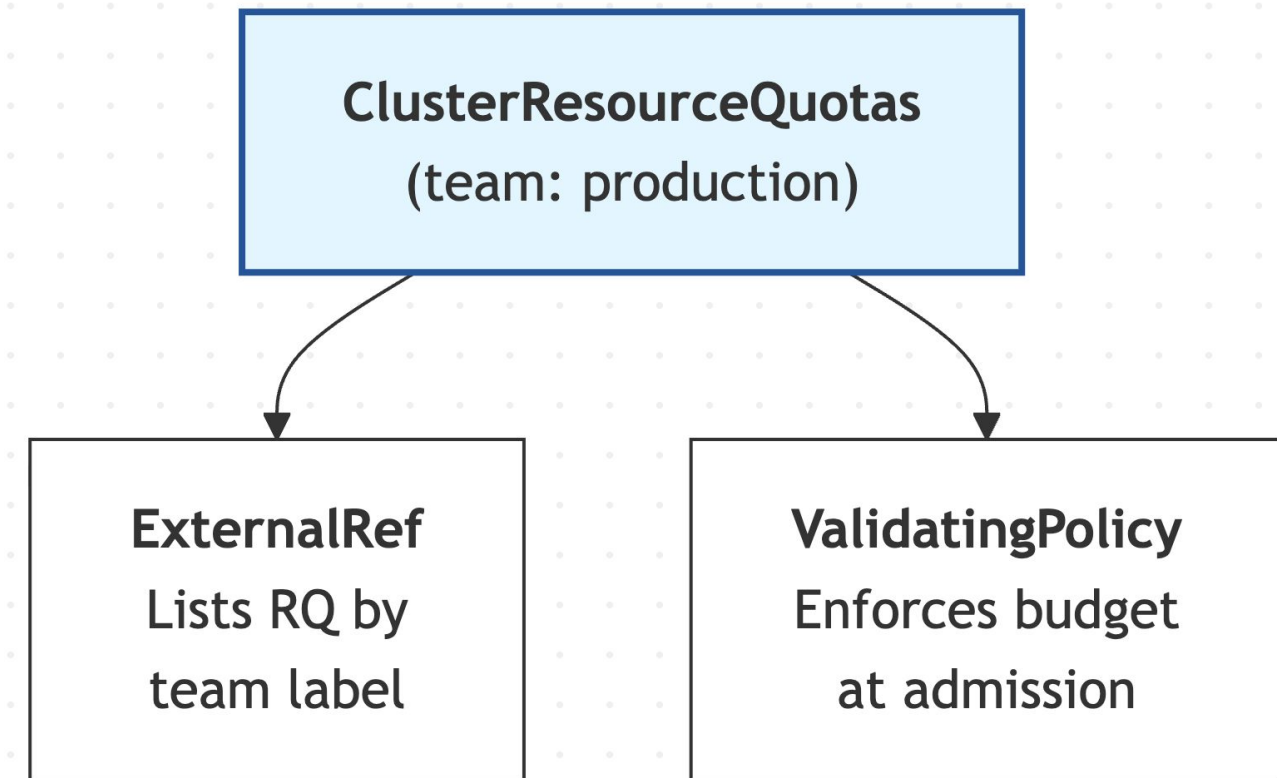


KubeCon



CloudNativeCon

Europe 2026



Watching team quotas across the cluster

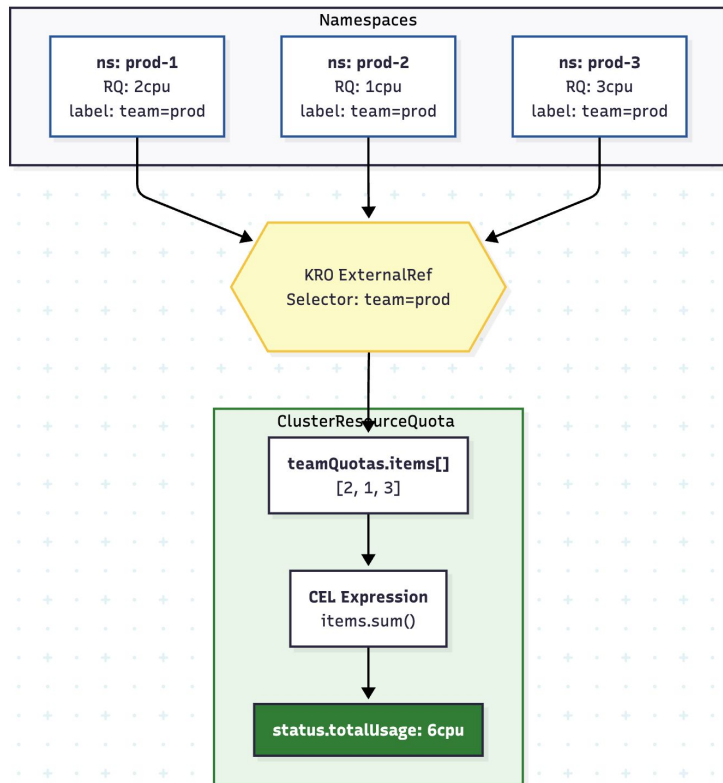


KubeCon



CloudNativeCon

Europe 2026



```
rgd.yaml

apiVersion: kro.run/v1alpha1
kind: ResourceGraphDefinition
metadata:
  name: cluster-resource-quotas.kro.run
spec:
  schema:
    apiVersion: v1alpha1
    kind: ClusterResourceQuotas
    spec:
      teamName: string
      budget:
        ...
  status:
    totalUsage:
      requests.cpu: ${teamQuotas.items.map(q, int(q.spec.hard['requests.cpu'])).sum()}
      requests.memory: ${teamQuotas.items.map(q, int(q.spec.hard['requests.memory'])).sum()}
    ...
  resources:
    - id: teamQuotas
      externalRef:
        apiVersion: v1
        kind: ResourceQuota
        labelSelector:
          team: ${schema.spec.teamName}
```

Budget enforcement via Kyverno



KubeCon



CloudNativeCon

Europe 2026

```
rgd.yaml

- id: validatingPolicy
  template:
    apiVersion: policies.kyverno.io/v1beta1
    kind: ValidatingPolicy
    metadata:
      name: validate-quota-${schema.spec.teamName}
    spec:
      matchConstraints:
        objectSelector:
          matchExpressions:
            - key: team
              operator: In
              values: ["${schema.spec.teamName}"]
      resourceRules:
        - operations: ["CREATE", "UPDATE"]
          resources: ["resourcequotas"]
      variables:
        - name: other_team_quotas
          expression: >-
            resource.List("v1", "resourcequotas", "", {"team": "..."}).items
              .filter(q, q.metadata.name != object.metadata.name)
        - name: current_cpu_usage
          expression: >-
            dyn(variables.other_team_quotas).map(q, quantity(...).asApproximateFloat()).sum()
      validations:
        - expression: >-
            requested_cpu + current_cpu_usage <= budget_cpu
```

How Enforcement Works



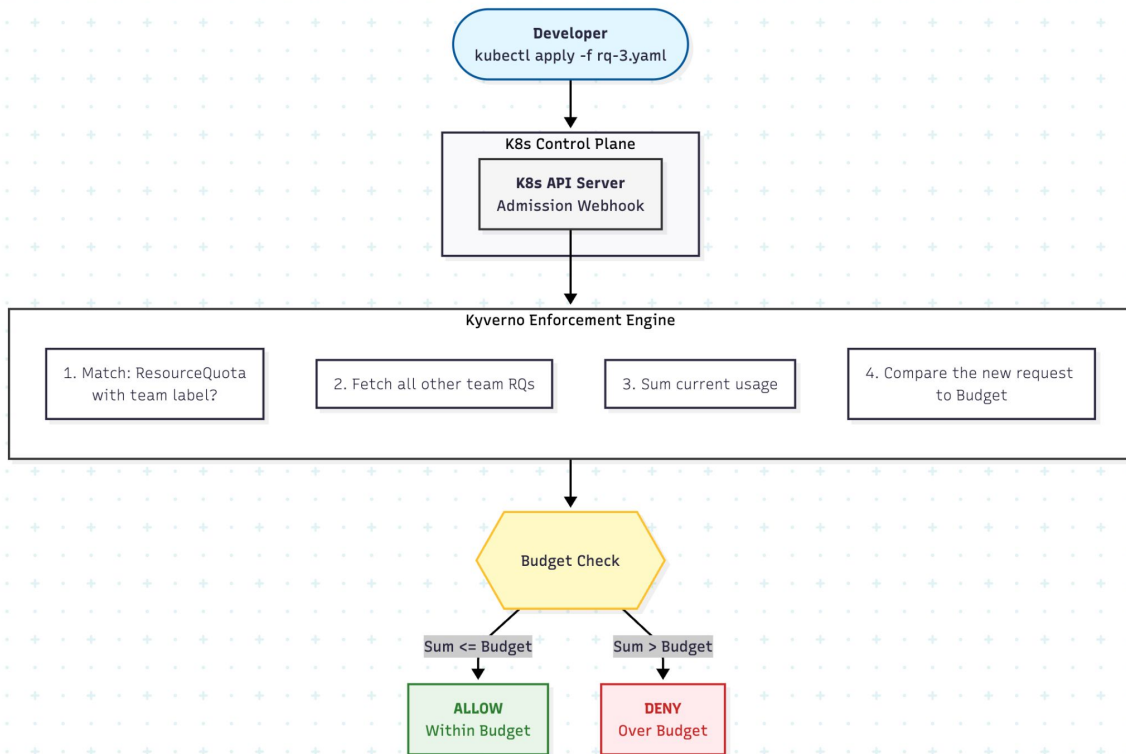
KubeCon



CloudNativeCon

Europe 2026

What happens when a developer creates a ResourceQuota



Step 2: Create a Team Instance



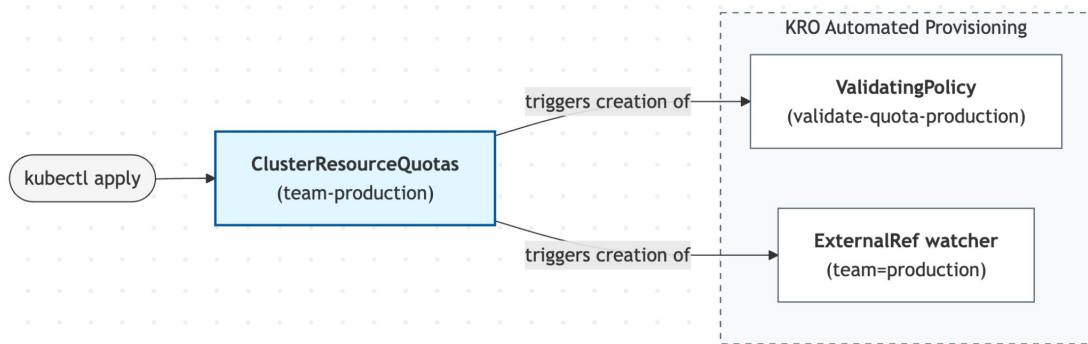
KubeCon



CloudNativeCon

Europe 2026

One resource per team, that's it!



ClusterResourceQuotas

```
apiVersion: kro.run/v1alpha1
kind: ClusterResourceQuotas
metadata:
  name: team-production
  labels:
    team: production
spec:
  teamName: production
  budget:
    requests.cpu: "8"
    requests.memory: "16Gi"
    limits.cpu: "16"
    limits.memory: "32Gi"
```

What the Dashboard Looks Like



KubeCon



CloudNativeCon

Europe 2026

Real-time visibility via *kubectl get*

```
dashboard

$ kubectl get clusterresourcequotas team-production -o yaml

status:
  totalUsage:
    requests.cpu: "5"
    requests.memory: "10Gi"
    limits.cpu: "10"
    limits.memory: "20Gi"
  conditions:
  - type: InstanceSynced
    status: "True"
    reason: ReconciliationSucceeded
    state: ACTIVE
```

What Rejection Looks Like



KubeCon



CloudNativeCon

Europe 2026

```
dashboard

$ kubectl apply -f rq-new.yaml

Error from server:
resource ResourceQuota/prod-3/rq-new was blocked
validate-quota-production:
  Team budget exceeded for 'production'.
  CPU request would bring total to 12,
  but budget is 8.
  Memory request would bring total to 24Gi,
  but budget is 16Gi.
```

Multi-Team at Scale

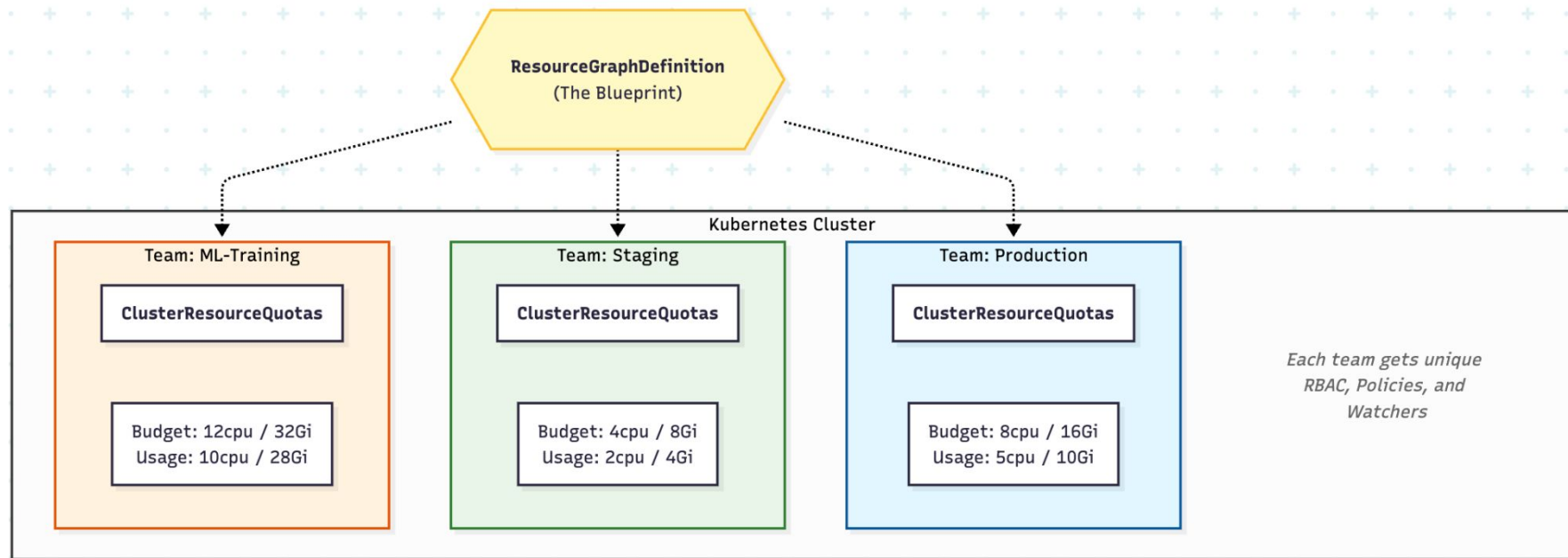


KubeCon



CloudNativeCon

Europe 2026





KubeCon



CloudNativeCon

Europe 2026

Demo



Key Takeaways



KubeCon



CloudNativeCon

Europe 2026

- Kubernetes ResourceQuota is namespace-scoped
- kro lets you define a ClusterResourceQuotas custom resource that aggregates usage in real time
- Kyverno enforces team budgets at admission time -- over-budget requests are rejected before they land
- One ResourceGraphDefinition + one CR per team = full visibility and enforcement, zero scripts
- The pattern scales: add a new team with a single kubectl apply



KubeCon



CloudNativeCon

Europe 2026

Resources



Demo Repo



kro



Kyverno





KubeCon



CloudNativeCon

Europe 2026

Thank You!

