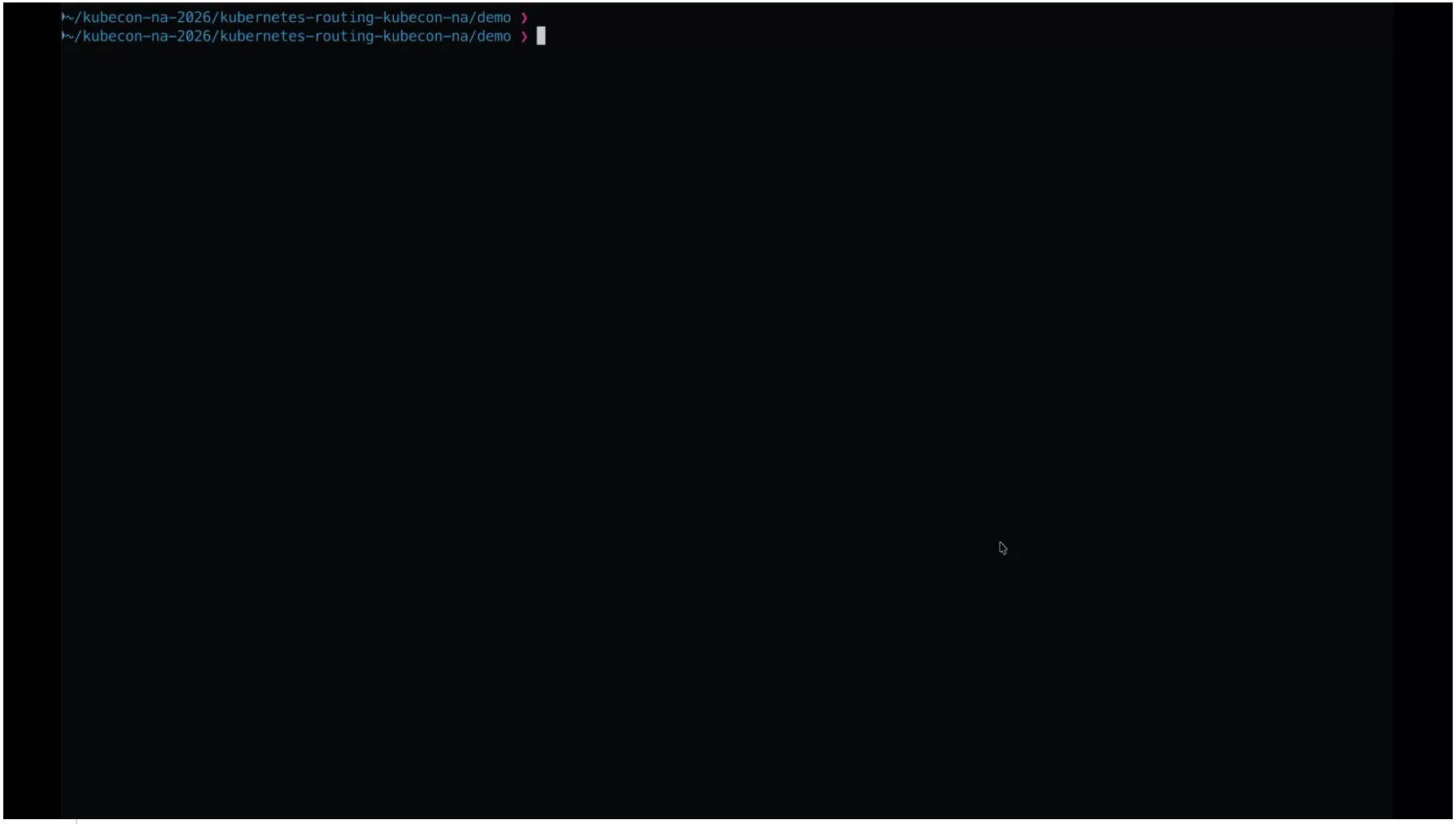


Four Resources, One Object: Surviving Gateway API After Ingress NGINX

Sitesh Pattanaik
Software Development Engineer, AWS EKS | Co-founder, Seattle Systems




~/kubecon-na-2026/kubernetes-routing-kubecon-na/demo
~/kubecon-na-2026/kubernetes-routing-kubecon-na/demo

```
$ kubectl get httproute -A
NAMESPACE   NAME      HOSTNAMES                AGE
team-chat   chat-v2   ["chat.example.com"]     3m21s

$ kubectl get securitypolicy chat-auth -n team-chat -o yaml | yq .spec

targetRefs:
  - kind: HTTPRoute
    name: chat

$ kubectl get securitypolicy chat-auth -n team-chat -o yaml | yq .status

      message: Policy has been accepted.
      reason: Accepted
      status: "True"

$ c -H "Host: chat.example.com"        # no token
200
```

Ingress NGINX is retired. Ingress is not.

2025-11-11
Retirement announced

March 2026
Best-effort maintenance ends

2026-03-24
Repository archived
no releases, no CVE fixes

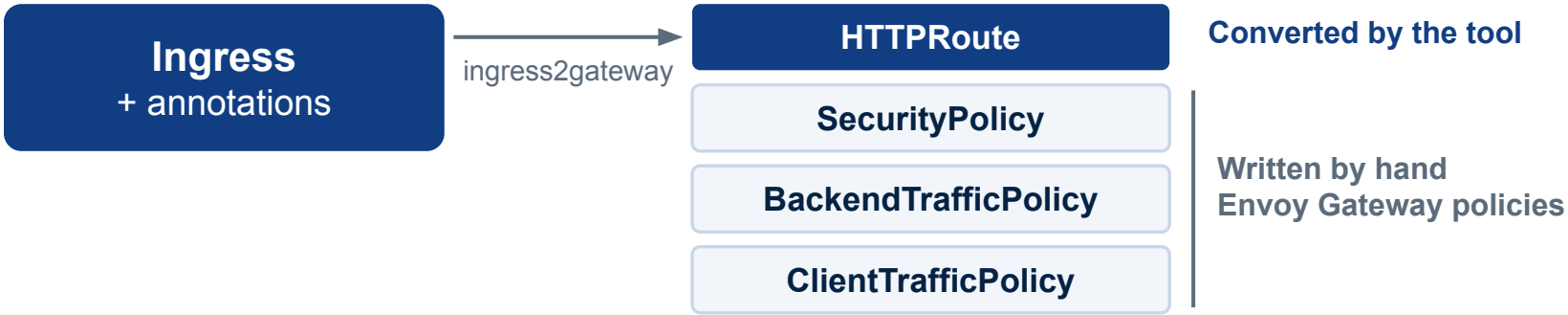
Retired: Ingress NGINX
the implementation

Still supported: Ingress API
GA and frozen, no plans to remove it

kubernetes.io/blog/2025/11/11/ingress-nginx-retirement · kubernetes.io/docs/concepts/services-networking/ingress

Two migrations happen at once

	API	Implementation	Features beyond the spec
Before	Ingress	Ingress NGINX	Annotations, inside the Ingress
After	Gateway API	Envoy Gateway	Policy CRDs, separate objects



LIVE DEMO · SWITCH TO TERMINAL

Live: the platform, then four objects

Commands: demo/README.md, Act 1 setup + Demo 1 · run from demo/ after the shell setup

RMINAL

```
$ kubectl get pods -n envoy-gateway-system
NAME                              READY   STATUS    AGE
envoy-gateway-666cb7f75f-wgqkd    1/1     Running   6h48m

$ kubectl apply -f platform/
namespace/platform created
namespace/team-chat created
gatewayclass.gateway.networking.k8s.io/eg created
gateway.gateway.networking.k8s.io/shared-gw created
deployment.apps/model-70b created

configmap/chat-jwks created

$ kubectl get pods -n envoy-gateway-system
NAME                                          READY   STATUS    AGE
envoy-gateway-666cb7f75f-wgqkd                1/1     Running   6h48m
envoy-platform-shared-gw-4974d13c-757566879d-mc6nr   2/2     Running   17s

$ kubectl -n platform wait gateway/shared-gw --for=condition=Programmed
gateway.gateway.networking.k8s.io/shared-gw condition met

$ c
404
```

```
$ kubectl apply -f before/chat.yaml
httproute.gateway.networking.k8s.io/chat created
securitypolicy.gateway.envoyproxy.io/chat-auth created
backendtrafficpolicy.gateway.envoyproxy.io/chat-ratelimit created
clienttrafficpolicy.gateway.envoyproxy.io/chat-client created

$ kubectl get httproute,securitypolicy,backendtrafficpolicy -n team-chat
NAME                                    HOSTNAMES               AGE
httproute.gateway.networking.k8s.io/chat    ["chat.example.com"]   18s
NAME                                                AGE
securitypolicy.gateway.envoyproxy.io/chat-auth      18s
NAME                                                AGE
backendtrafficpolicy.gateway.envoyproxy.io/chat-ratelimit   18s

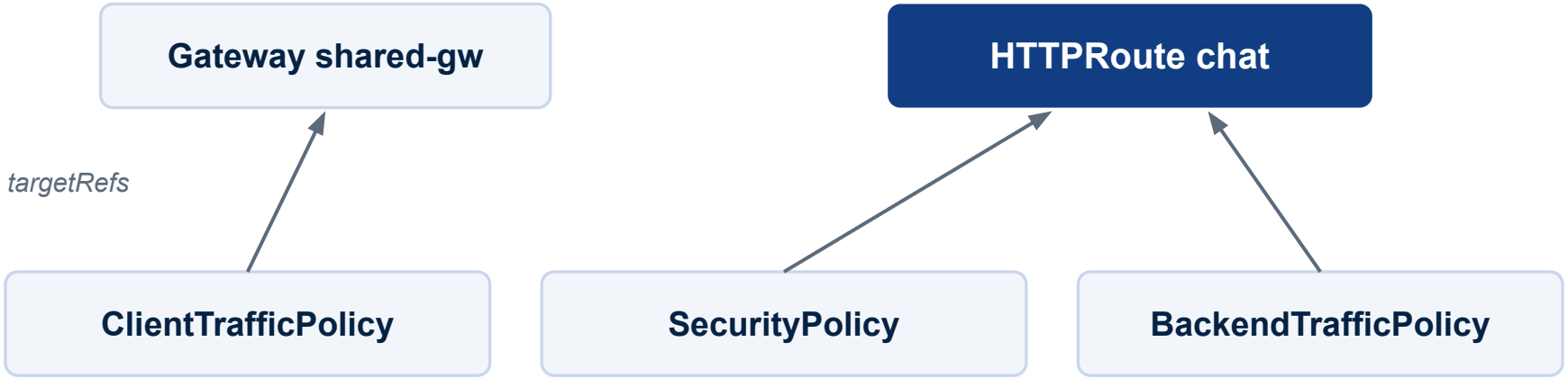
$ kubectl get clienttrafficpolicy -n platform
NAME          AGE
chat-client   28s
```

```
$ c -H "Host: chat.example.com"        # no token
401
$ c -H "Host: chat.example.com" -H "Authorization: Bearer $TOKEN"
200

$ kubectl get backendtrafficpolicy chat-ratelimit -n team-chat -o yaml | yq .spec
rateLimit:
  local:
    rules:
      - clientSelectors:
          - headers:
              
                name: x-tier
                value: premium
        limit:
          requests: 3
          unit: Minute


$ for i in 1 2 3 4 5; do c -H "Host: chat.example.com" \
    -H "Authorization: Bearer $TOKEN" -H "x-tier: premium"; done
200
200
200
429
429
```

The arrows point one way



Policies point at their targets. Targets never point back.

How Policy attachment looks like

```
kind: SecurityPolicy
spec:
  targetRefs:
  - kind: HTTPRoute
    name: chat    # a string
```

A name, not ownership

No ownerReference. Deleting the route leaves the policy.

No existence check

Admission accepts a target that doesn't exist.

No reverse link

The route has no field listing its policies.

Pattern: Gateway API (GEP-713) · Behaviour shown: Envoy Gateway v1.9

Envoy Gateway: missing target, no status

```
route, ok := routes[key]

// Route not found
// It's not an error if the gateway
// is not found ...
if !ok {
    return nil, nil
}
```

What happens

The target isn't resolved

No status condition is written

The policy stays in the cluster

kubectl apply: succeeded

envoyproxy/gateway v1.9.3 · internal/gatewayapi/securitypolicy.go, backendtrafficpolicy.go

LIVE DEMO · SWITCH TO TERMINAL

Live: a one-line rename

Commands: demo/README.md, Demo 2 · run from demo/ after the shell setup

```
$ kubectl apply -f before/break-1-route-v2.yaml
httproute.gateway.networking.k8s.io/chat-v2 created
$ kubectl get httproute -A
NAMESPACE   NAME      HOSTNAMES              AGE
team-chat   chat      ["chat.example.com"]   14s
team-chat   chat-v2   ["chat.example.com"]   5s

$ kubectl delete httproute -n team-chat chat        # the GitOps prune
httproute.gateway.networking.k8s.io "chat" deleted from team-chat namespace
$ kubectl get httproute -A
NAMESPACE   NAME      HOSTNAMES              AGE
team-chat   chat-v2   ["chat.example.com"]   12s

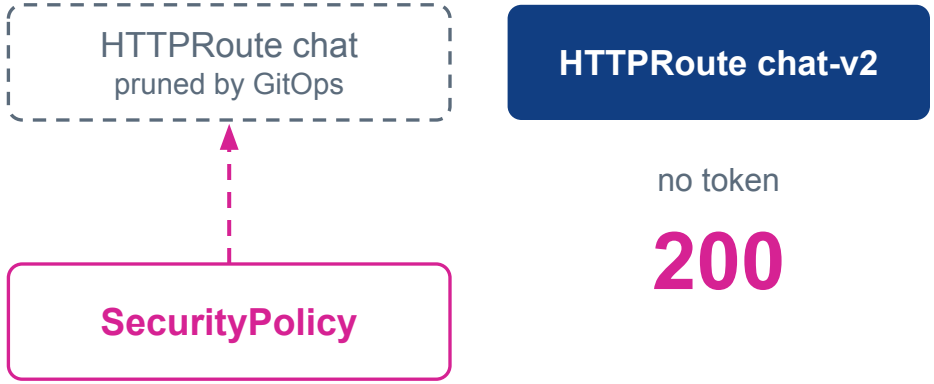
$ c -H "Host: chat.example.com"        # no token
200
$ kubectl get securitypolicy chat-auth -n team-chat -o yaml | yq .status
ancestors:
  - ancestorRef:
      kind: Gateway
      name: shared-gw
      
    conditions:
      
      message: Policy has been accepted.
      reason: Accepted
      status: "True"
```

Break 1: a one-line rename drops auth

The pull request

```
- name: chat
+ name: chat-v2
```

One file changed. The policy isn't in it.



The policy still points at the old name. Auth fails open.

LIVE DEMO · SWITCH TO TERMINAL

Live: a new route named chat

Commands: demo/README.md, Demo 3 · run from demo/ after the shell setup

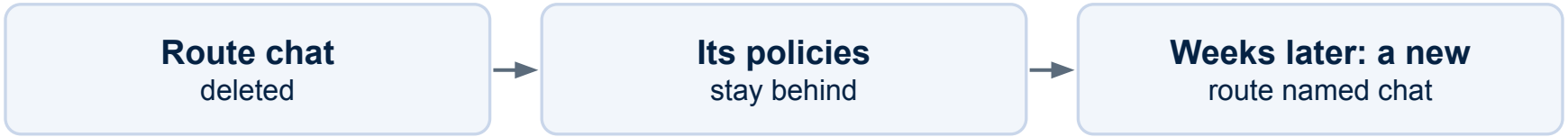
```
$ kubectl delete httproute -n team-chat chat-v2
httproute.gateway.networking.k8s.io "chat-v2" deleted from team-chat namespace

$ kubectl apply -f before/break-2-new-chat.yaml   # a status page, also named chat
httproute.gateway.networking.k8s.io/chat created
$ kubectl get httproute -A
NAMESPACE   NAME   HOSTNAMES                 AGE
team-chat   chat   ["status.example.com"]    7s

$ c -H "Host: status.example.com"        # no token
401
```


Commands: demo/README.md, Demo 2 · run from demo/ after the shell setup

Break 2: the leftover policy is a landmine





The status page now returns 401. Old intent, new target.

LIVE DEMO · SWITCH TO TERMINAL

Live: one typo

Commands: demo/README.md, Demo 4 · run from demo/ after the shell setup

```
$ kubectl apply -f before/break-3-ctp-typo.yaml
clienttrafficpolicy.gateway.envoyproxy.io/chat-client-typo created
$ kubectl get clienttrafficpolicy -A
NAMESPACE   NAME               AGE
platform    chat-client        2m52s
platform    chat-client-typo   4s

$ kubectl get clienttrafficpolicy chat-client-typo -n platform -o yaml | yq .spec
targetRefs:
  - group: gateway.networking.k8s.io
    kind: Gateway
    name: shared-gateway        # the real one is shared-gw
timeout:
  http:
    streamIdleTimeout: 300s

$ kubectl get clienttrafficpolicy chat-client-typo -n platform -o yaml | yq .status
null
```


Commands: demo/README.md, Demo 4 · run from demo/ after the shell setup

Break 3: a typo has nowhere to report

```
kind: ClientTrafficPolicy
targetRefs:
- kind: Gateway
  name: shared-gateway
```




```
status: {}
```

No ancestor exists, so there's nothing to report TargetNotFound against.

github.com/envoyproxy/gateway/issues/10215

Why review and admission miss it

Valid objects

Each manifest passes schema validation on its own.

Local admission

ValidatingAdmissionPolicy CEL can't look up whether another object exists.

Local diffs

Review shows the renamed route, not the policies that point at it.

Validity is per object. Correctness is per graph.

The spec calls it the discoverability problem

> *“… it does come with meaningful challenges, such as the Discoverability problem and the Fanout status update problem.”*

GEP-713, Metaresources and Policy Attachment

A documented trade-off. Implementations decide how to show it.

Change the unit of ownership

Before: the app team writes

- **HTTPRoute**
- **SecurityPolicy**
- **BackendTrafficPolicy**
- **ClientTrafficPolicy**



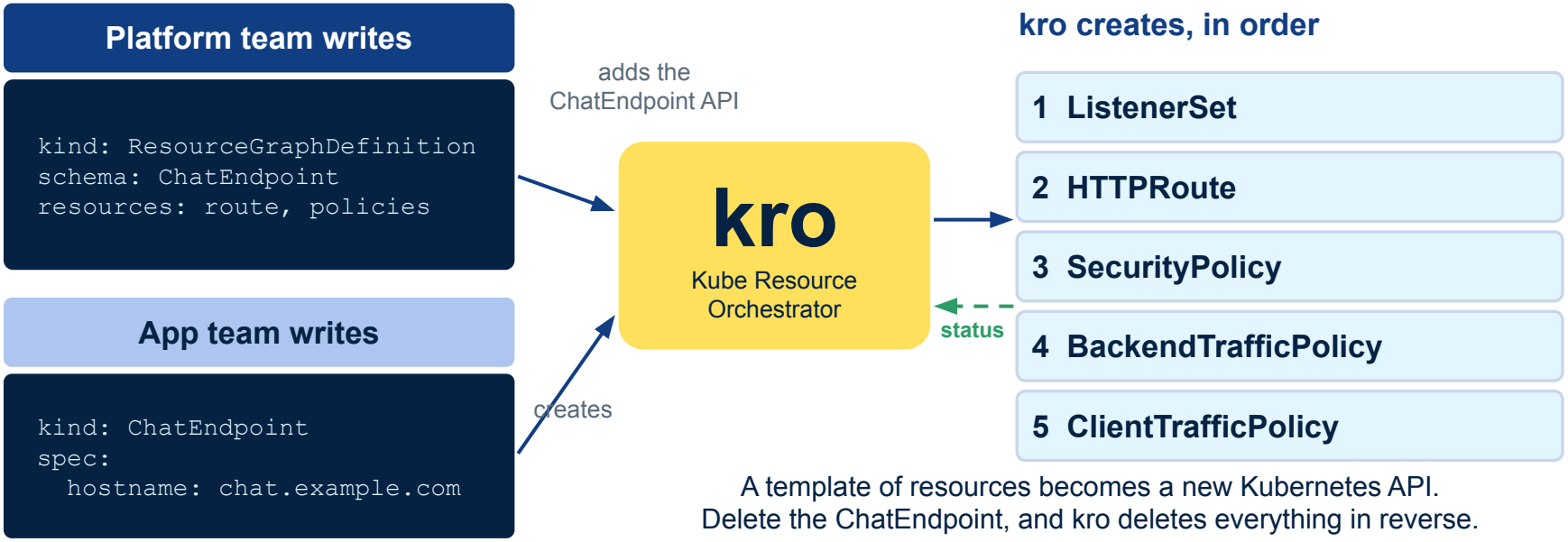
After: the app team writes

ChatEndpoint chat
one custom resource, 7 lines

The platform team owns the RGD that expands it into the routing graph.

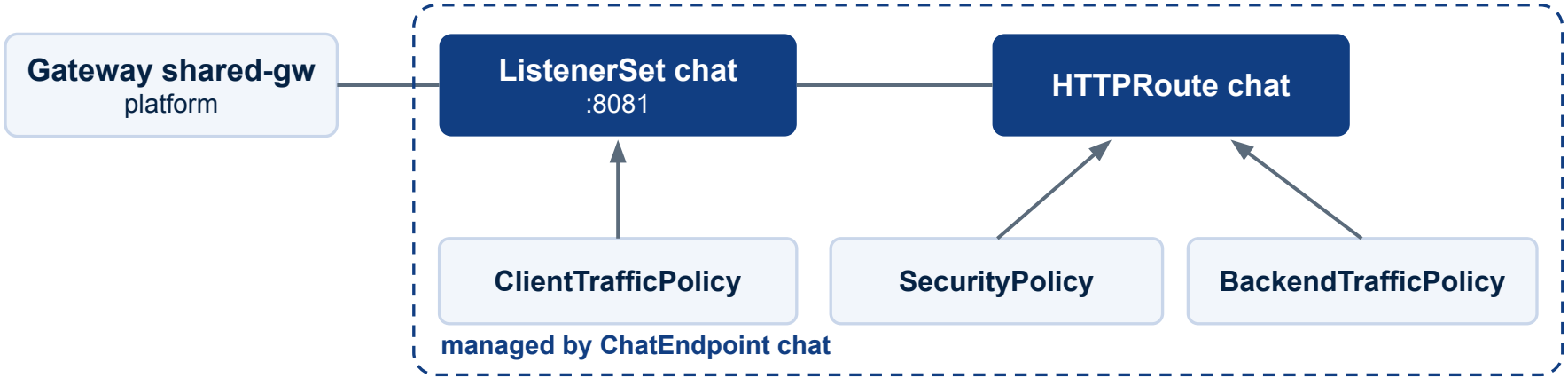
RBAC: app teams can create ChatEndpoint, not raw policies.

kro in one slide



A template of resources becomes a new Kubernetes API.
Delete the ChatEndpoint, and kro deletes everything in reverse.

One object owns the whole graph



The ListenerSet has its own port because plain-HTTP listeners on one port share one Envoy listener. In production, use HTTPS on 443: Envoy splits listeners by hostname.

LIVE DEMO · SWITCH TO TERMINAL

Live: one object, the whole graph

Commands: demo/README.md, Demo 5 · run from demo/ after the shell setup

```
$ kubectl delete chatendpoint -n team-chat chat
chatendpoint.platform.example.com "chat" deleted from team-chat namespace
$ kubectl get chatendpoint -A
NAMESPACE   NAME      STATE    READY   AGE
team-chat   chat-v2   ACTIVE   True    36s

$ kubectl get securitypolicy -A
NAMESPACE   NAME           AGE
team-chat   chat-v2-auth   14s

$ c -H "Host: chat.example.com"        # no token
401
$ c -H "Host: chat.example.com" -H "Authorization: Bearer $TOKEN"
200
```


Commands: demo/README.md, Demo 5 · run from demo/ after the shell setup

References become dependencies

```
resources:
  - id: route
    template:
      kind: HTTPRoute
  - id: auth
    includeWhen:
      - ${schema.spec.auth.enabled}
    template:
      kind: SecurityPolicy
      spec:
        targetRefs:
          - kind: HTTPRoute
            name: ${route.metadata.name}
```

- **The target is an expression**

  It reads the route's name, so the policy depends on the route.

- **So kro knows the order**

  ```
  gateway → listenerSet → route
  → auth → rateLimit → clientPolicy
  ```

  Created in this order, deleted in reverse.

- **includeWhen makes it optional**

  Turn auth off, and kro removes the SecurityPolicy.

Excerpt from demo/rgd/chatendpoint-rgd.yaml

One status answers: is it attached?

What the app team writes

```yaml
apiVersion: platform.example.com/v1alpha1
kind: ChatEndpoint
metadata:
  name: chat
  namespace: team-chat
spec:
  hostname: chat.example.com
```

What the app team watches

```yaml
status:
  state: ACTIVE
  listenerAccepted: true
  routeAccepted: true
  rateLimitAttached: true
  clientPolicyAttached: true
```

One object to get, watch, and alert on.

LIVE DEMO · SWITCH TO TERMINAL

Live: replay the rename and the drift

Commands: demo/README.md, Demo 6 · run from demo/ after the shell setup

LIVE DEMO · SWITCH TO TERMINAL

```
$ kubectl apply -f rgd/chat-v2-instance.yaml
chatendpoint.platform.example.com/chat-v2 created
$ kubectl get chatendpoint -A
NAMESPACE   NAME      STATE         READY   AGE
team-chat   chat      ACTIVE        True    5m21s
team-chat   chat-v2   IN_PROGRESS   False   7s

$ kubectl get chatendpoint -n team-chat chat-v2 -o yaml | yq .status
conditions:
  -   # InstanceManaged, GraphResolved: True
  - message: '... waiting for node "listenerSet": readyWhen condition
      evaluated to false ... (waiting for readiness)'
    reason: NotReady
    type: ResourcesReady
  -   # Ready: False, same message
listenerAccepted: false
state: IN_PROGRESS
```

LIVE DEMO · SWITCH TO TERMINAL

Live: replay the landmine and the typo

Commands: demo/README.md, Demo 6 · run from demo/ after the shell setup

```
$ kubectl delete chatendpoint -n team-chat chat-v2
chatendpoint.platform.example.com "chat-v2" deleted from team-chat namespace
$ kubectl get securitypolicy,backendtrafficpolicy -A
No resources found

$ kubectl apply -f rgd/typo-instance.yaml        # gatewayName: shared-gateway
chatendpoint.platform.example.com/chat-typo created
$ kubectl get clienttrafficpolicy -A
No resources found
$ kubectl get chatendpoint -A
NAMESPACE   NAME        STATE         READY   AGE
team-chat   chat-typo   IN_PROGRESS   False   17s

$ kubectl get chatendpoint -n team-chat chat-typo -o yaml | yq .status
conditions:
  □   # InstanceManaged, GraphResolved: True
  - message: 'resource reconciliation failed: apply results contain errors:
      waiting for external reference "gateway": not found'
    reason: NotReady
    type: ResourcesReady
  □   # Ready: False, same message
state: IN_PROGRESS
```

Here’s what we saw…

Break	Loose objects	With the RGD
Rename	Auth silently drops	New instance brings its own policies
Hand-deleted route	Policies orphaned	kro recreates the route
Name reuse	Old policy reattaches	Nothing left behind to attach
Gateway typo	Empty policy status	ChatEndpoint stays not Ready, with a reason

When is an RGD enough?

	Helm	kro RGD	Crossplane	Operator
Keeps pieces together after install	No	Yes	Yes	Yes
Reports whether policies attached	No	readyWhen	Readiness	Any logic
Logic	Templates	CEL	Functions	Code
Cross-instance coordination	No	No	Partly	Yes
External side effects	No	Other CRDs	Providers	Yes
You maintain	Charts	1 controller + RGDs	Core, functions	Your code

Take it home



This talk's repo

In the repository

- The ResourceGraphDefinition and ChatEndpoint
- The starting manifests and three breaks
- Offline k3d demo: install, reset, cleanup



Learn kro: kro.run

Don't review policies. Review graphs.

KubeCon CloudNativeCon
North America 2026

Thank you

Connect on LinkedIn

Questions?

