Good Fences Make Good Neighbors

Making Cross-Namespace References more secure with ReferenceGrant

Speaker: **Nick Young**

Not this type of Neighbours

What we'll talk about

- Namespaces are one of the most important security boundaries in Kubernetes
- Making references across namespaces is easy to get wrong
- Some prior art
- How Gateway API does cross-namespace references
- What's a ReferenceGrant anyway?
- Next steps

Kubernetes Namespaces

Namespaces are the main way to enclose trust boundaries.

Most users control a whole namespace.

Sometimes, cross-namespace references would be really handy though!

- TLS Secrets you want to use but shouldn’t see the values of.
- Some users want to have ingress config live in one namespace and backends in another.

Cross-namespace references are hard!

How do you ensure that you can only expose what you *should* be?

The key is that the referent and the referrer need to agree.

Prior Art - Contour's TLSCertificateDelegation

```yaml
apiVersion: projectcontour.io/v1
kind: TLSCertificateDelegation
metadata:
  name: example-com-wildcard
  namespace: www-admin
spec:
  delegations:
    - secretName: example-com-wildcard
      targetNamespaces:
      - example-com
---
apiVersion: projectcontour.io/v1
kind: HTTPProxy
metadata:
  name: www
  namespace: example-com
spec:
  virtualhost:
    fqdn: foo2.bar.com
    tls:
      secretName: www-admin/example-com-wildcard
  routes:
    - services:
        - name: s1
          port: 80
```

elegated

s

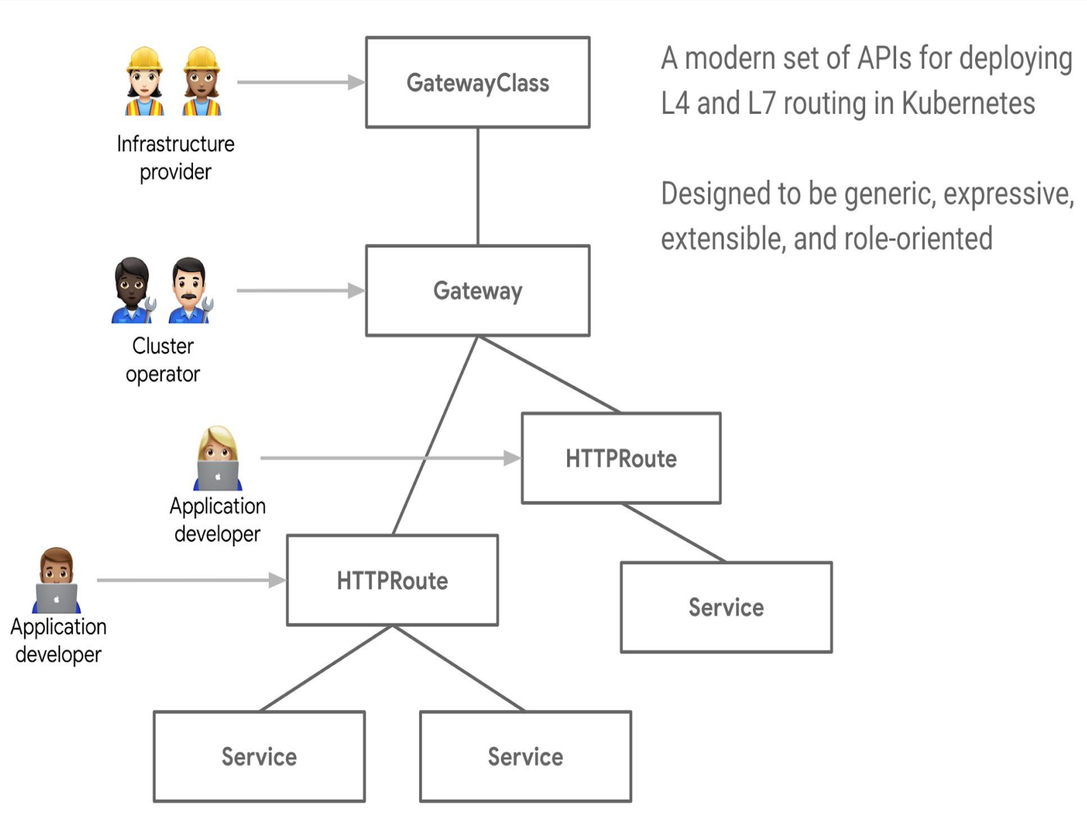
for all

d in Contour's HTTPProxy resource to reference

@youngnick

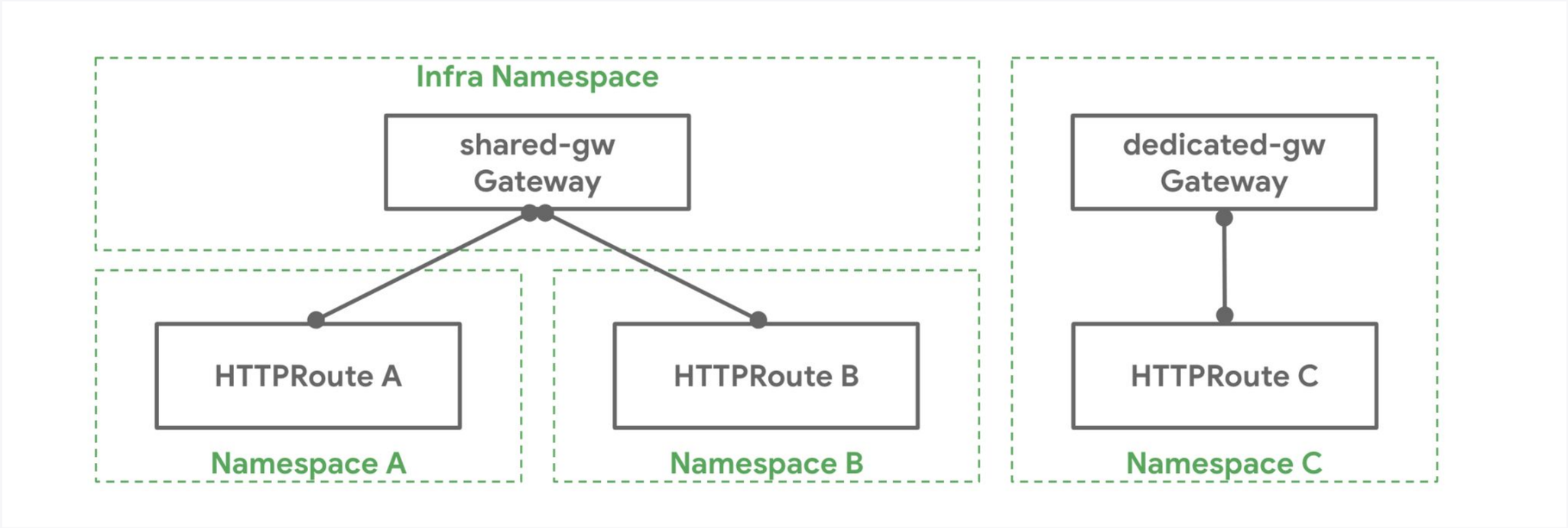
Let’s do better in Gateway API

In Gateway API, we’re trying to do this *right*.



@youngnick

Gateway API - Gateway and Route binding



Gateway API - Gateway and Route binding

```yaml
kind: Gateway
metadata:
  name: shared-gw
  namespace: infra-ns
spec:
  listeners:
  - name: http
    hostname: "foo.example.com"
    allowedRoutes:
      namespaces:
        from: Selector
        selector:
          matchLabels:
            shared-gateway-access: "true"
```

```yaml
kind: Namespace
metadata:
  name: nsA
  labels:
    shared-gateway-access: "true"
---
kind: HTTPRoute
metadata:
  name: HTTPRouteA
  namespace: nsA
spec:
  parentRefs:
  - name: shared-gw
    namespace: infra-ns
  rules:
  - backendRefs:
    - name: home
      port: 8080
```

@youngnick

Gateway API - Gateway and Route binding

Gateway and Route binding forces the Gateway owner and Route owner to agree.

Gateway owners *allow* routes to attach.

Routes *request* to attach to a parent Gateway.

By default, Gateways with no **`allowedRoutes`** will allow any Route *from the same namespace* to attach.

But what about core objects?

Adding **`allowedRoutes`** and **`parentRef`** is fine for *new* objects, but what about if we need to do cross-namespace references on objects we can't add to?

Like:

- Secret, or
- Service

Enter ReferenceGrant

```yaml
apiVersion: gateway.networking.k8s.io/v1alpha2
kind: ReferenceGrant
metadata:
  name: allow-prod-traffic
  namespace: app-ns
spec:
  from:
  - group: gateway.networking.k8s.io
    kind: HTTPRoute
    namespace: prod
  to:
  - group: ""
    kind: Service
```

```yaml
apiVersion: gateway.networking.k8s.io/v1alpha2
kind: ReferenceGrant
metadata:
  name: allow-gateways-prod
  namespace: secretslivehere
spec:
  from:
  - group: gateway.networking.k8s.io
    kind: Gateway
    namespace: prod
  to:
  - group: ""
    kind: Secret
```

ReferenceGrant Design Goals

- For things that we can’t change the spec of easily - like Service and Secret.
- Owned by the owner of the Granted object, lives in the same namespace
- Grants access to things by Group, Kind and Namespace.
- Does *not* do one thing:
    - No label selectors for namespace.

Other notes and future work

- ReferenceGrant must be fully reconciled; Removal of ReferenceGrant means removal of the granted access.
- Controllers are expected to be able to be granted broad *read* access, and then self-limit what they *use* based on ReferenceGrant.
- No way to currently grant access to “all namespaces” - ***** seems right.

Next steps

Next Steps

- Already used in SIG-Storage for cross-namespace datasources in PersistentVolumeClaims. (see the blog about Cross-Namespace Data Sources)
- KEP-3766 opened to move ReferenceGrant to new home under SIG-Auth.

KEP-3766

https://github.com/kubernetes/enhancements/issues/3766

Cross-Namespace
Data sources blog

https://kubernetes.io/blog/2023/01/02/cross-namespace-data-sources-alpha/

@youngnick

KEP-3766

- This KEP was started last week, and has already meant significant changes to the proposed ReferenceGrant
- Most reviewers want this to be an in-tree resource, we were expecting to just move the existing CRD
- Keep an eye on the KEP for more information.

@youngnick

Takeaways

Takeaways

1

Cross namespace refs are hard!

It’s really important to do them correctly, and easy to accidentally grant more access than is needed.

2

The key is agreement between namespaces

Usually, the safest way is to make the referent the one that allows the access.

3

Multiple patterns are possible

Route-Gateway binding and ReferenceGrant are both ways to solve this same problem.

ISOVALENT

Thank you!