
KubeCon
CloudNativeCon
North America 2022
BUILDING FOR THE ROAD AHEAD
DETROIT 2022
TiKV
TiKV
HELM

BUILDING FOR THE ROAD AHEAD

DETROIT 2022

Exiting Ingress with the Gateway API

Rob Scott, Google
Shane Utt, Kong

2 Questions

1. How many of you have used Ingress API?

2. How many of you have used Gateway API?

KubeCon | CloudNativeCon North America 2022

BUILDING FOR THE ROAD AHEAD

DETROIT 2022

Rob Scott, Google
@robertjscott

Shane Utt, Kong
@shaneutt

API Overviews

The Ingress API

- Host and Path Matching
- Forward to Service
- TLS Configuration
- Stable for 5 years
- **Simple and broadly implementable**

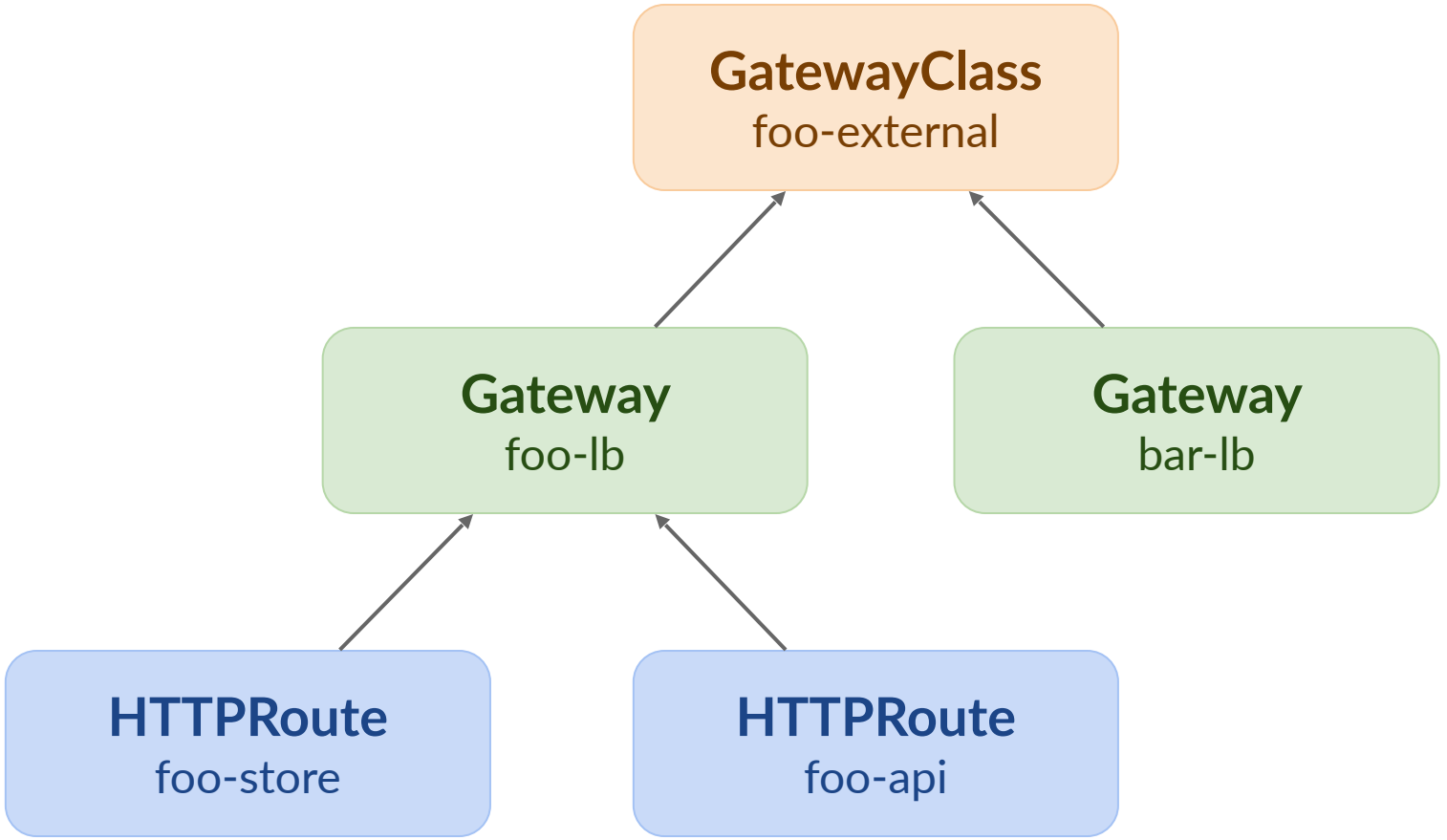
```yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: minimal-ingress
spec:
  ingressClassName: nginx-example
  rules:
  - http:
      paths:
      - path: /testpath
        pathType: Prefix
        backend:
          service:
            name: test
            port:
              number: 80
```

Ingress Limitations

- Not enough features
- Custom extensions everywhere
- Extensions were not portable:
  - Traffic splitting
  - Header matching
  - Sticky sessions
- Annotations were wild west
- Insufficient permission model

Gateway API

- Next generation of Kubernetes routing and load balancing APIs
- Designed to be expressive and extensible
- Role oriented resource model
- 15 implementations
- Graduated to beta in July



Gateway API Features

- TLS configuration
- HTTP matching
  - Host
  - Path
  - Header
  - Method
  - Query Param
- Cross namespace Gateway -> Route binding
- Cross namespace forwarding
- HTTP filters
  - Header modifier
  - Request mirroring
  - Request redirects
  - URL rewrites
- Weight based traffic splitting
- *More in progress…*

Gateway API Features

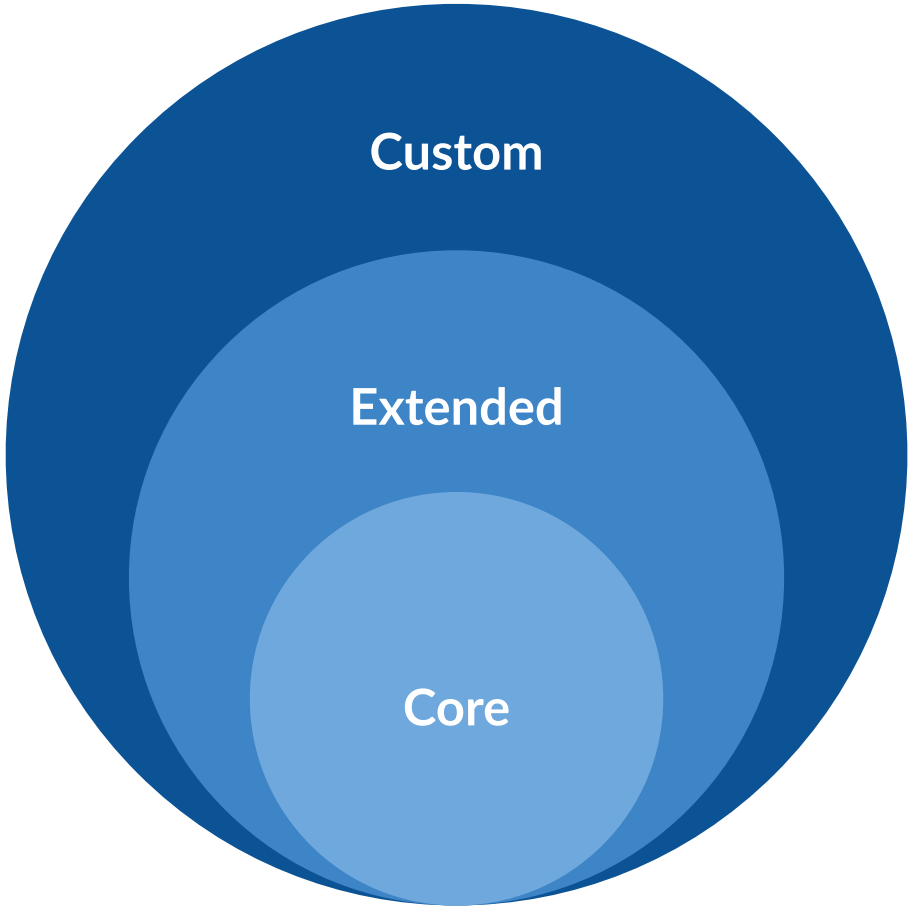
- TLS configuration
- HTTP matching
  - Host
  - Path

Only in Gateway API

- Header
- Method
- Query Param
- Cross namespace Gateway -> Route binding
- Cross namespace forwarding
- HTTP filters
  - Header modifier
  - Request mirroring
  - Request redirects
  - URL rewrites
- Weight based traffic splitting
- *More in progress…*

Conformance Levels

- **Core:** Every implementation expected to support feature in consistent way
  - *HTTP Prefix Path Matching*
- **Extended:** When this is supported, it must be done according to spec
  - *HTTP Header Modification*
- **Custom:** Implementations may have some variation in how they support this feature
  - *HTTP Regex Path Matching*



Ingress vs. HTTPRoute

Simple Path Match

Ingress

```yaml
ingressClassName: nginx
rules:
- http:
    paths:
    - path: /login
      pathType: Prefix
      backend:
        service:
          name: auth-svc
          port:
            number: 8080
```

HTTPRoute

```yaml
parentRefs:
- name: nginx
rules:
- matches:
  - path:
      type: PathPrefix
      value: /login
  backendRefs:
  - name: auth-svc
    port: 8080
```

Simple Path Match

Ingress

```yaml
ingressClassName: nginx
rules:
- http:
    paths:
    - path: /login
      pathType: Prefix
      backend:
        service:
          name: auth-svc
          port:
            number: 8080
```

HTTPRoute

```yaml
parentRefs:
- name: nginx
rules:
- matches:
  - path:
      type: PathPrefix
      value: /login
  backendRefs:
  - name: auth-svc
    port: 8080
```

Simple Path Match

Ingress

```
ingressClassName: nginx
rules:
- http:
    paths:
    - path: /login
      pathType: Prefix
      backend:
        service:
          name: auth-svc
          port:
            number: 8080
```

HTTPRoute

```
parentRefs:
- name: nginx
rules:
- matches:
  - path:
      type: PathPrefix
      value: /login
  backendRefs:
  - name: auth-svc
    port: 8080
```

Adding a Parent

HTTPRoute

```
parentRefs:
- name: nginx
- name: contour
rules:
- matches:
  - path:
      type: PathPrefix
      value: /login
  backendRefs:
  - name: auth-svc
    port: 8080
```

Adding a Parent

Ingress

```yaml
ingressClassName: nginx
rules:
- http:
    paths:
    - path: /login
      pathType: Prefix
      backend:
        service:
          name: auth-svc
          port:
            number: 8080
```

```yaml
ingressClassName: contour
rules:
- http:
    paths:
    - path: /login
      pathType: Prefix
      backend:
        service:
          name: auth-svc
          port:
            number: 8080
```

Adding a Path Match

HTTPRoute

```yaml
parentRefs:
- name: nginx
rules:
- matches:
  - path:
      type: PathPrefix
      value: /login
  - path:
      type: Exact
      value: /auth
  backendRefs:
  - name: auth-svc
    port: 8080
```

Adding a Path Match

Ingress

```
ingressClassName: nginx
rules:
- http:
    paths:
    - path: /login
      pathType: Prefix
      backend:
        service:
          name: auth-svc
          port:
            number: 8080
```

```
- path: /auth
  pathType: Exact
  backend:
    service:
      name: auth-svc
      port:
        number: 8080
```

Adding Host Match

Ingress

```yaml
ingressClassName: nginx
rules:
- host: example.net
  http:
    paths:
    - path: /login
      pathType: Prefix
      backend:
        service:
          name: auth-svc
          port:
            number: 8080
```

HTTPRoute

```yaml
parentRefs:
- name: nginx
hostnames:
- example.net
rules:
- matches:
  - path:
      type: PathPrefix
      value: /login
  backendRefs:
  - name: auth-svc
    port: 8080
```

Adding Host Match

HTTPRoute

```yaml
parentRefs:
- name: nginx
hostnames:
- example.net
- example.com
rules:
- matches:
  - path:
      type: PathPrefix
      value: /login
  backendRefs:
  - name: auth-svc
    port: 8080
```

Adding Host Match

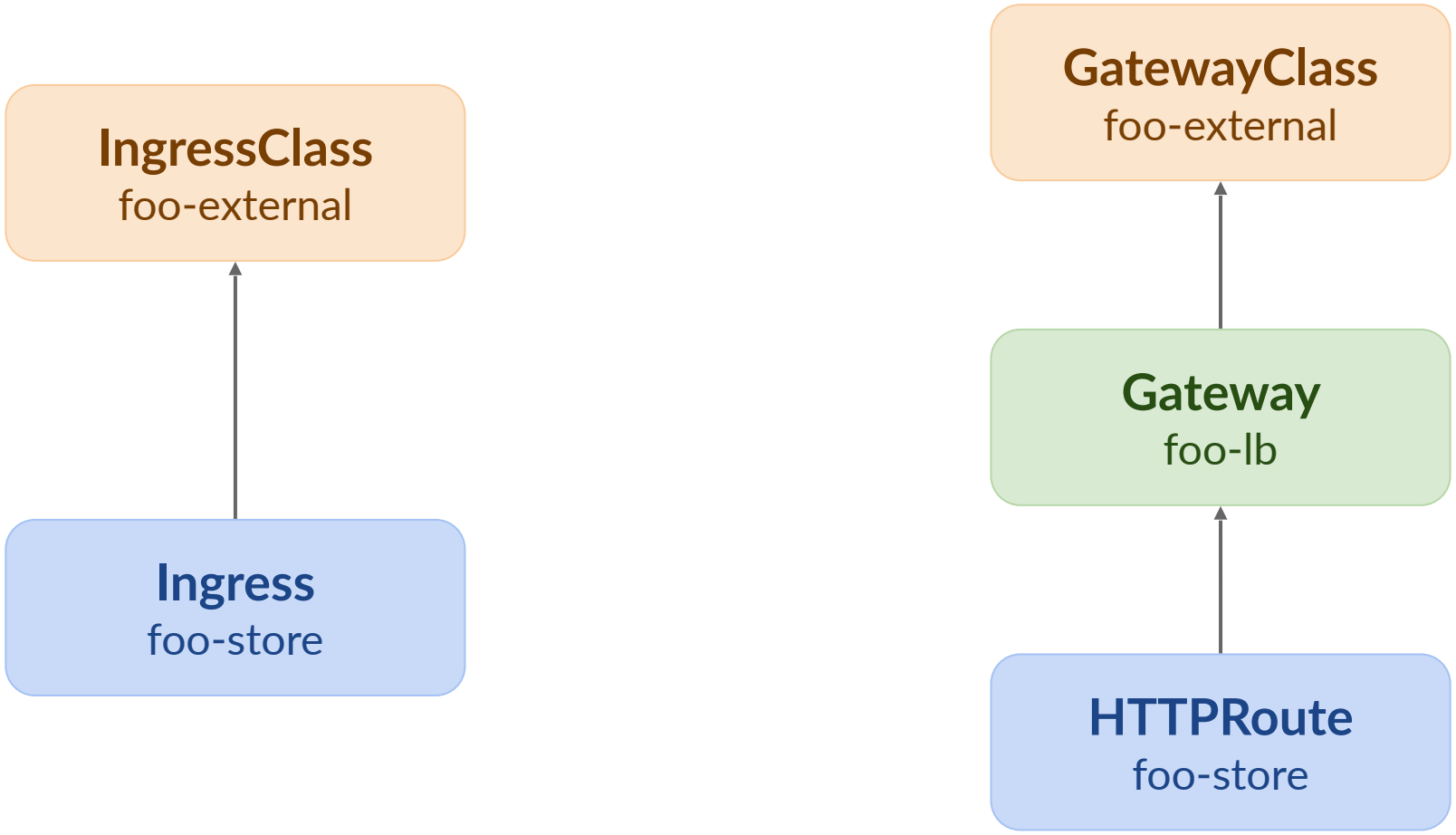
Ingress

```
ingressClassName: nginx
rules:
- host: example.net
  http:
    paths:
    - path: /login
      pathType: Prefix
      backend:
        service:
          name: auth-svc
          port:
            number: 8080
```

```
- host: example.com
  http:
    paths:
    - path: /login
      pathType: Prefix
      backend:
        service:
          name: auth-svc
          port:
            number: 8080
```

Gateways

Resource Models



Gateways

- Represents an instance of a load balancer/proxy
- Defines listeners and addresses
- Attach routes
- Config stays the same across implementations

```yaml
apiVersion: gateway.networking.k8s.io/v1beta1
kind: Gateway
metadata:
  name: prod-web
spec:
  gatewayClassName: gke-l7-gxlb
  listeners:
  - protocol: HTTPS
    port: 443
    hostname: example.com
    tls:
      certificateRefs:
      - name: example-com-cert
        namespace: prod-certs
```

Common Types of Implementations

In-Cluster

- Deploying a **Gateway** resource results in proxy **Pods** for serving traffic (e.g. the “data plane”)
- Often the addresses for the **Gateway** are provided via **LoadBalancer** type **Services**
- Behaves the same way on *any* Kubernetes cluster

Cloud Provider

- Deploying a **Gateway** resource provisions **Cloud Load Balancers**
- Load balance directly from Cloud LB to Pods, **no intermediate hop**
- Usually only available on clusters managed by the Cloud Provider

Route Attachment

- Routes attach to Gateways with **parentRefs** field
- Same route can be attached to multiple Gateways
- Useful for migrating between implementations

```yaml
apiVersion: gateway.networking.k8s.io/v1beta1
kind: HTTPRoute
metadata:
  name: prod-web
spec:
  parentRefs:
  - name: nginx
  - name: contour
  rules:
  - matches:
    - path:
        type: PathPrefix
        value: /login
    backendRefs:
    - name: auth-svc
      port: 8080
```

Allowed Routes

- **Gateway owners** can specify where Routes can be attached from
- **Developers** can attach their routes to Gateways in different namespaces

```yaml
apiVersion: gateway.networking.k8s.io/v1beta1
kind: Gateway
metadata:
  name: prod-web
spec:
  gatewayClassName: gke-l7-gxlb
  listeners:
  - name: http
    protocol: HTTP
    port: 80
    allowedRoutes:
      namespaces:
        from: Selector
        selector:
          matchLabels:
            app: store
```

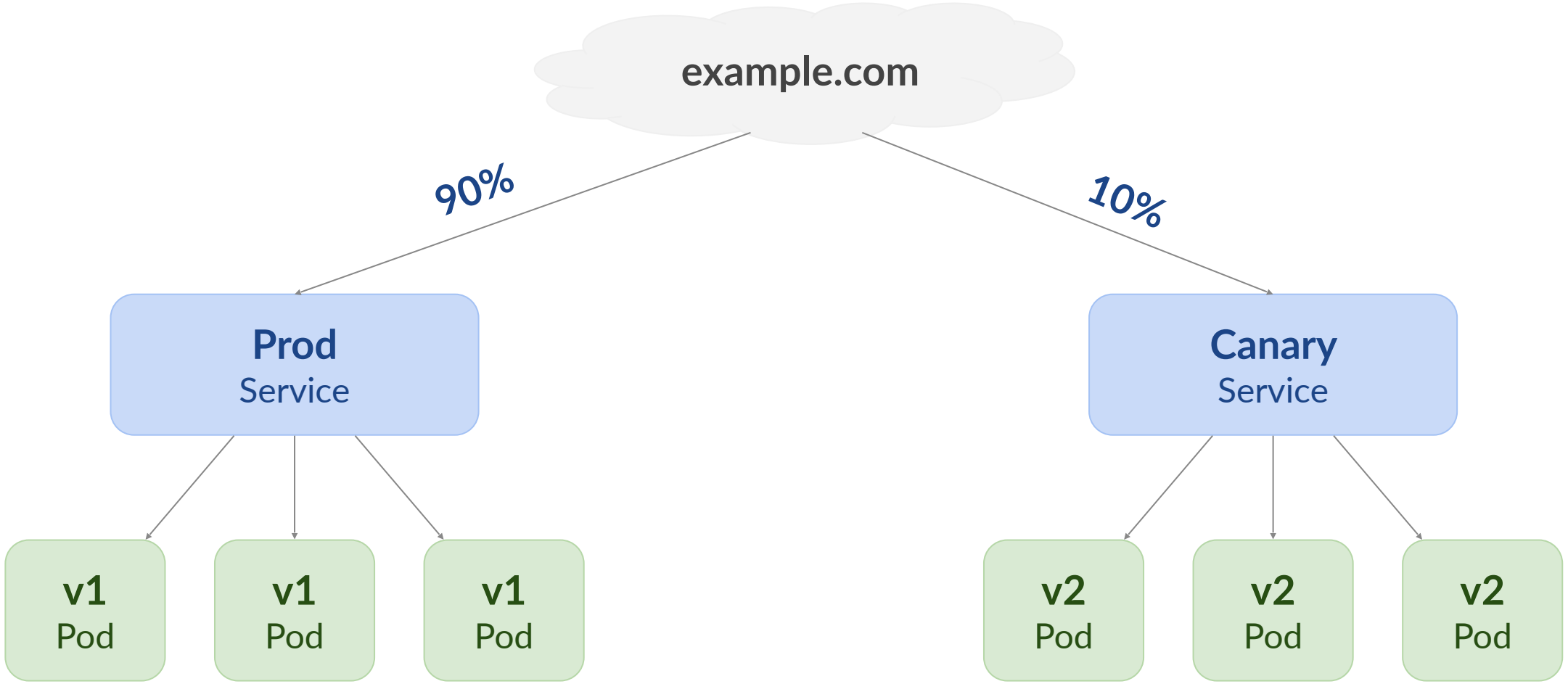
Allowed Routes

- Can specify different trusted namespaces per domain
- Optional extension, same namespace just works

```yaml
listeners:
- name: store
  hostname: store.example.com
  protocol: HTTP
  port: 80
  allowedRoutes:
    namespaces:
      from: Selector
      selector:
        matchLabels:
          app: store
- name: api
  hostname: api.example.com
  protocol: HTTP
  port: 80
  allowedRoutes:
    namespaces:
      from: Selector
        selector:
        matchLabels:
          app: api
```

Advanced Examples

Canary Routing with Ingress



Canary Routing with Ingress

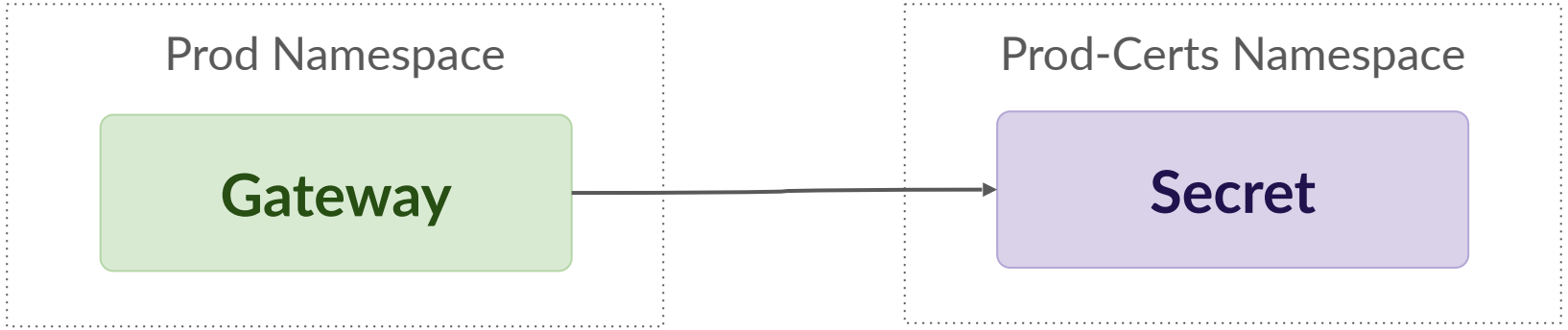

```yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: production
  annotations:
    kubernetes.io/ingress.class: nginx
spec:
  rules:
  - host: example.com
    http:
      paths:
      - pathType: Prefix
        path: "/"
        backend:
          service:
            name: production
            port:
              number: 80
```

```yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: canary
  annotations:
    kubernetes.io/ingress.class: nginx
    nginx.ingress.kubernetes.io/canary: "true"
    nginx.ingress.kubernetes.io/canary-weight: "10"
spec:
  rules:
  - host: example.com
    http:
      paths:
      - pathType: Prefix
        path: "/"
        backend:
          service:
            name: canary
            port:
              number: 80
```

Canary Routing with Gateway API


```yaml
apiVersion: gateway.networking.k8s.io/v1beta1
kind: HTTPRoute
metadata:
  name: canary-example
spec:
  hostnames:
  - example.com
  rules:
  - backendRefs:
    - name: canary
      port: 80
      weight: 10
    - name: production
      port: 80
      weight: 90
    matches:
    - path:
        type: PathPrefix
        value: "/"
```

Cross-Namespace Secret Refs



Cross-Namespace Secret Refs

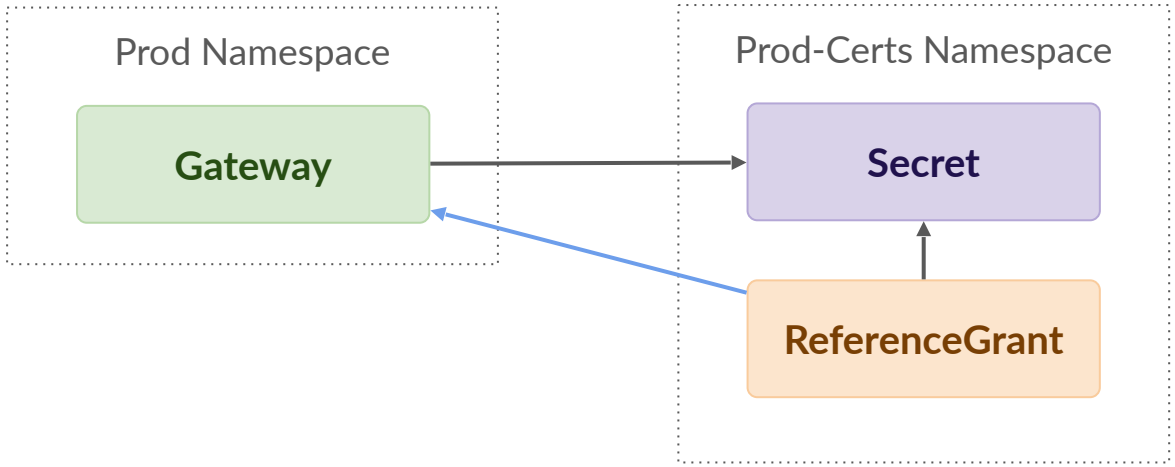

```yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: production
  annotations:
    projectcontour.io/tls-cert-namespace: prod-certs
spec:
  tls:
  - hosts:
      - example.com
    secretName: testsecret-tls
  rules:
  - host: example.com
    http:
      paths:
      ...
```

Cross-Namespace Secret Refs


```yaml
apiVersion: gateway.networking.k8s.io/v1beta1
kind: Gateway
metadata:
  name: tls-basic
spec:
  gatewayClassName: acme-lb
  listeners:
  - name: example-com-https
    protocol: HTTPS
    port: 443
    hostname: example.com
    tls:
      certificateRefs:
      - name: example-com-cert
        namespace: prod-certs
```

ReferenceGrant


```yaml
kind: ReferenceGrant
metadata:
  name: allow-prod-cert-refs
  namespace: prod-certs
spec:
  from:
  - group: networking.gateway.k8s.io
    kind: Gateway
    namespace: prod
  to:
  - group: ""
    kind: Secret
    name: example-com-cert
```



Should you switch?

Should you switch?

No

- Ingress API and ecosystem are quite stable
- If everything you need is covered by Ingress today, there's no need to upgrade

Should you switch?

No

- Ingress API and ecosystem are quite stable
- If everything you need is covered by Ingress today, there's no need to upgrade

Yes

- Ingress is not getting new features
- Gateway API is much more expressive, extensible, and portable
- Broader than ingress, same APIs can be used for:
  - Mesh
  - L4 (TCP, UDP)
- Gateway API has graduated to beta
- Gateway API can be installed in any Kubernetes 1.16+ cluster

Getting Started

First Steps

- Install CRDs
- Install an implementation
- Follow our guides at gateway-api.sigs.k8s.io/guides
  - Simple Gateway (a good one to start out with)
  - HTTP routing
  - HTTP redirects and rewrites
  - HTTP traffic splitting
  - Routing across Namespaces
  - Configuring TLS

Picking an implementation

gateway-api.sigs.k8s.io/implementations

and several more…

Demo

Create a Cluster


```
gcloud container clusters create gw-demo \
  --gateway-api=standard
```

```
➜  gw-demo
```

```
➜  gw-demo
```

```
➜  gw-demo
```

Create a Gateway


```yaml
apiVersion: gateway.networking.k8s.io/v1beta1
kind: Gateway
metadata:
  name: gke-gateway
spec:
  gatewayClassName: gke-l7-rilb
  listeners:
  - name: http
    protocol: HTTP
    port: 80
```

```
➜  gw-demo
```

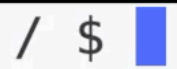
```
➜  gw-demo kubectl get gateway --watch
NAME           CLASS          ADDRESS   READY   AGE
gke-gateway    gke-l7-rilb                      14s
```

Deploy a HTTPRoute


```yaml
apiVersion: gateway.networking.k8s.io/v1beta1
kind: HTTPRoute
metadata:
  name: store
spec:
  parentRefs:
  - name: gke-gateway
  rules:
  - backendRefs:
    - name: store-v1
      port: 8080
```

```
➜  gw-demo
```

```
➜  gw-demo
```



```
}/ $ curl -s http://10.128.15.206 | grep "namespace\|pod"
```

Kong Gateway Operator

- Technical Preview, “Kong Incubator”
- In-Cluster Implementation
- Gateway/Proxy in Pods

```
→ gw-demo
```

Install Kong GatewayClass


```yaml
apiVersion: gateway.networking.k8s.io/v1beta1
kind: GatewayClass
metadata:
  name: kong
spec:
  controllerName: konghq.com/gateway-operator
```

```
→ gw-demo
```

Deploy a Kong Gateway


```yaml
apiVersion: gateway.networking.k8s.io/v1beta1
kind: Gateway
metadata:
  name: kong-gateway
spec:
  gatewayClassName: kong
  listeners:
  - name: http
    protocol: HTTP
    port: 80
```

```
→ gw-demo
```

```
→ gw-demo kubectl apply -f kong-gateway.yaml
gateway.gateway.networking.k8s.io/kong created
→ gw-demo kubectl wait --for=condition=Ready=true gateways.gateway.networking.k8s.io/kong
```

Add Kong Gateway to HTTPRoute


```yaml
apiVersion: gateway.networking.k8s.io/v1beta1
kind: HTTPRoute
metadata:
  name: store
spec:
  parentRefs:
  - name: gke-gateway
  - name: kong-gateway
  rules:
  - backendRefs:
    - name: store-v1
      port: 8080
```

```
→ gw-demo
```

Traffic Splitting

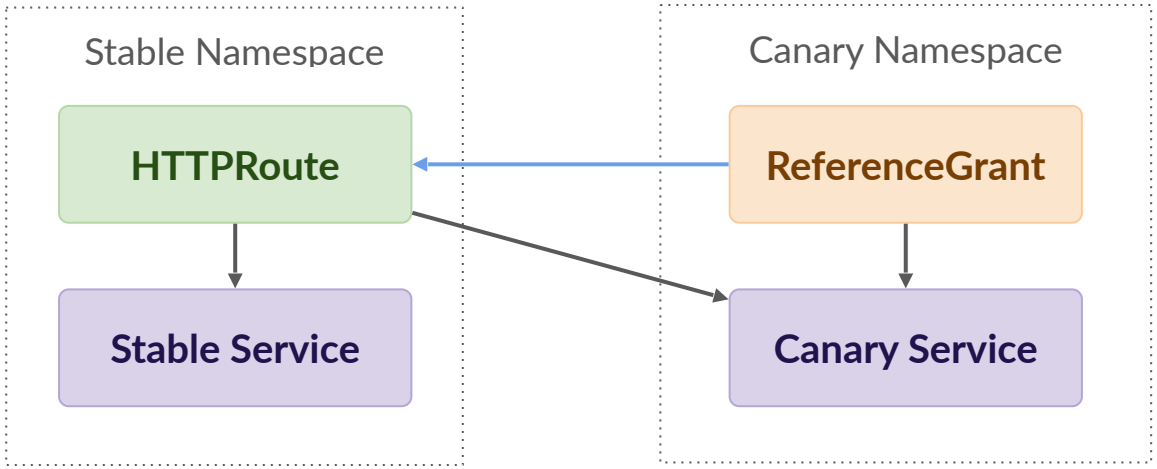

```yaml
parentRefs:
- name: gke-gateway
- name: kong-gateway
rules:
- backendRefs:
  - name: store-v1
    port: 8080
    weight: 50
  - name: store-canary
    port: 8080
    weight: 50
```

```
→ gw-demo
```

```
/ $ curl -s http://10.128.15.206 | grep "namespace\|pod"
 "namespace": "default",
 "pod": "store-v1-6bdfffb54-78nfb"
/ $ curl -s http://10.128.15.206 | grep "namespace\|pod"
 "namespace": "default",
 "pod": "store-canary-5c748f4f6b-l6pw9"
/ $
```

Cross Namespace Traffic Splitting?


```yaml
kind: ReferenceGrant
metadata:
  name: allow-prod-cert-refs
  namespace: canary
spec:
  from:
  - group: networking.gateway.k8s.io
    kind: HTTPRoute
    namespace: stable
  to:
  - group: ""
    kind: Service
    name: canary
```



This looks like a **lot of work**

Ingress2Gateway

- Automatically reads Ingress config from cluster
- Outputs corresponding Gateways and HTTPRoutes
- github.com/kubernetes-sigs/ingress2gateway

```
$ go install sigs.k8s.io/ingress2gateway
$ ingress2gateway
```

What's next?

What's Next?

- **Service Mesh:**
  - GAMMA Initiative
- **GatewayClass, Gateway, and HTTPRoute**:
  - May graduate to v1 soon
- **TCPRoute, TLSRoute, UDPRoute:**
  - Need more feedback, want to get to beta
- **GRPCRoute:**
  - alpha in v0.6.0
- stretch goal: **Gateways as an alternative to Service load-balancing**

Get Involved

- Weekly community meetings on Mondays
- Weekly GAMMA meetings on Tuesdays
- Contributors from all backgrounds welcome

gateway-api.sigs.k8s.io

#sig-network-gateway-api

kubernetes-sigs/gateway-api

KubeCon | CloudNativeCon North America 2022

BUILDING FOR THE ROAD AHEAD

DETROIT 2022

Questions?

Leave us some feedback


KubeCon
CloudNativeCon
North America 2022
BUILDING FOR THE ROAD AHEAD
DETROIT 2022
TiKV
TiKV
HELM