

Beyond Kubernetes Ingress

with Gateway API

Shane Utt

March 2023

Agenda

- Review of the Ingress API
- Introduction to Gateway API
- Overview and anatomy of API Resources
- Talk about Kong + Demo
- Discuss upcoming features
- Talk about sub-projects

Ingress

Review of Ingress

- Simple
- Host and Path Matching
- Forward to Service
- TLS Configuration

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: demo
spec:
  ingressClassName: kong
  rules:
  - http:
    paths:
    - path: /demo
      pathType: Prefix
      backend:
        service:
          name: demo
          port:
            number: 80
```

Limitations of Ingress

- Many **non-portable extensions** among 22+ implementations
- Annotations everywhere
- Insufficient permission model
- Mainly focused on HTTP(S) traffic
- Limited to North/South traffic
- Heavily reliant on Kubernetes **Service API**

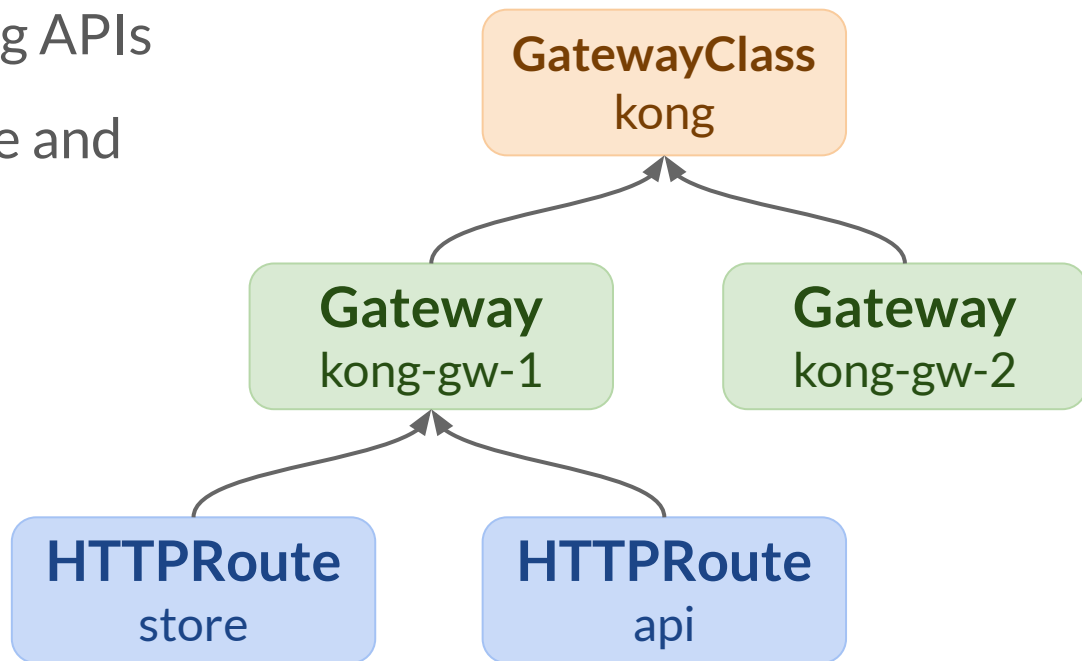
Impetus for change

- Extensibility
- Multi-protocol Support
- Conformance

Gateway API

Gateway API

- Next generation of Kubernetes routing and load balancing APIs
- Designed to be expressive and extensible
- Multiple API Resources
- Role oriented



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

Gateway API Features

- TLS configuration
- HTTP matching
 - Host
 - Path

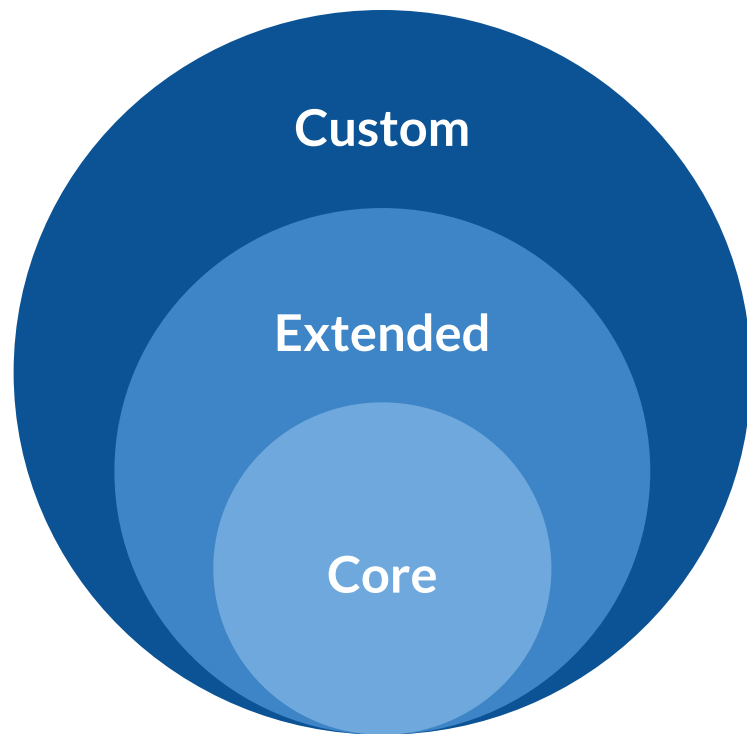
Only in Gateway API

- 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

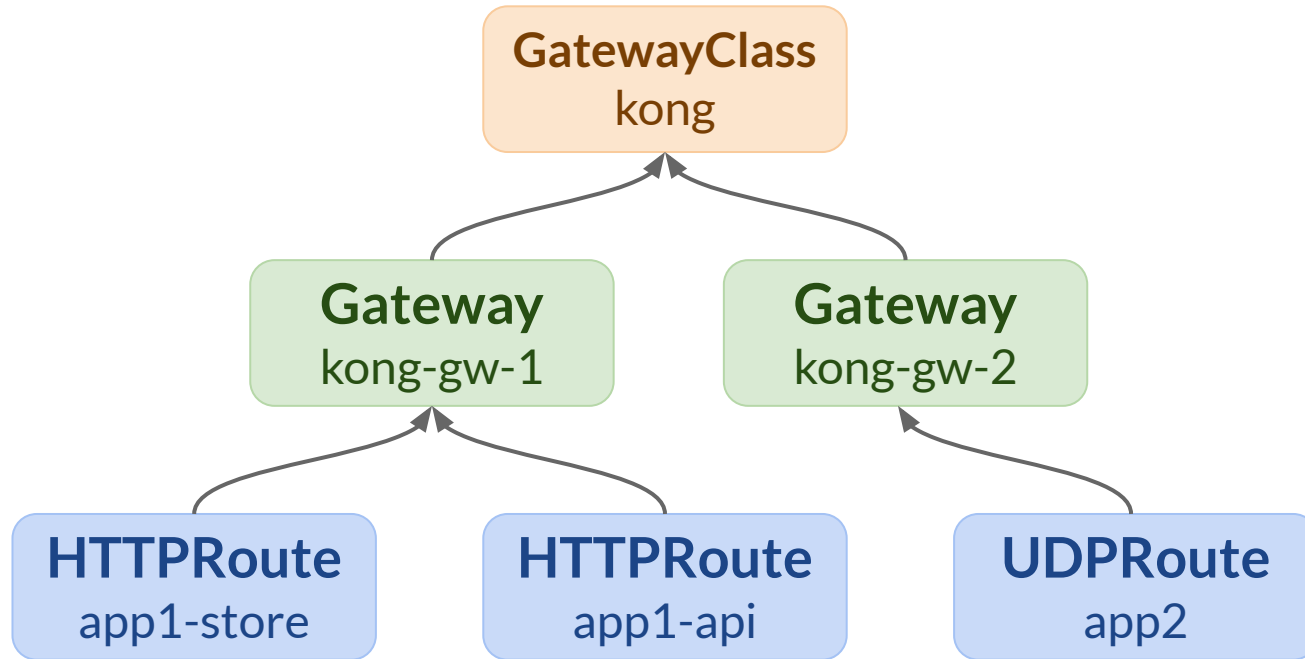
Conformance Levels

- **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*



GatewayClass Gateway HTTPRoute

GatewayClass, Gateway, HTTPRoute



GatewayClass

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

Gateway

`apiVersion`: gateway.networking.k8s.io/v1beta1

`kind`: Gateway

`metadata`:

`name`: kong

`spec`:

`gatewayClassName`: kong

`listeners`:

- `name`: http

`protocol`: HTTP

`port`: 80

HTTPRoute

`apiVersion`: gateway.networking.k8s.io/v1beta1

`kind`: HTTPRoute

`metadata`:

`name`: demo

`spec`:

`parentRefs`:

 - `name`: kong

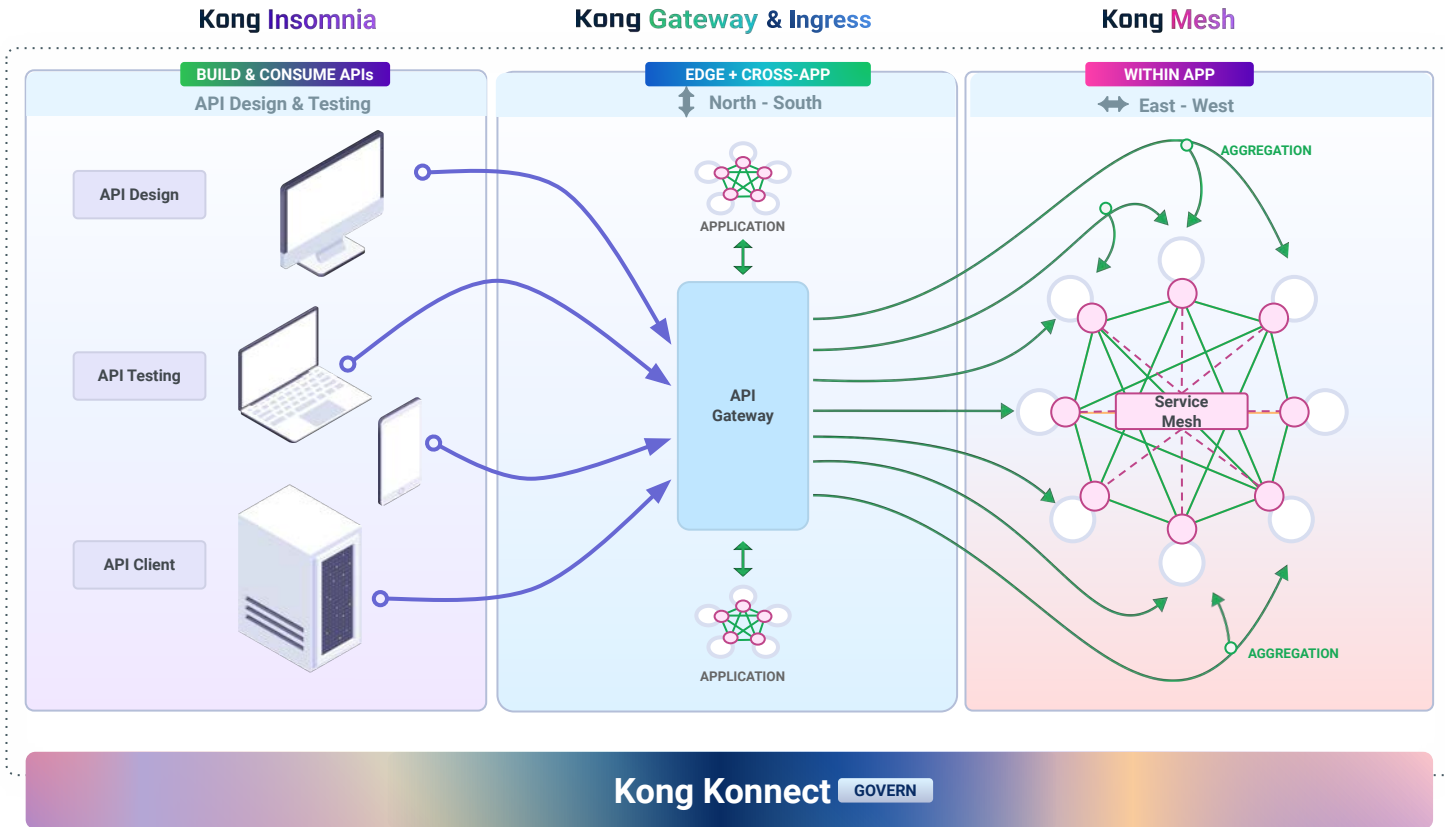
`rules`:

 - `backendRefs`:

 - `name`: demo

`port`: 80

Gateway API at Kong



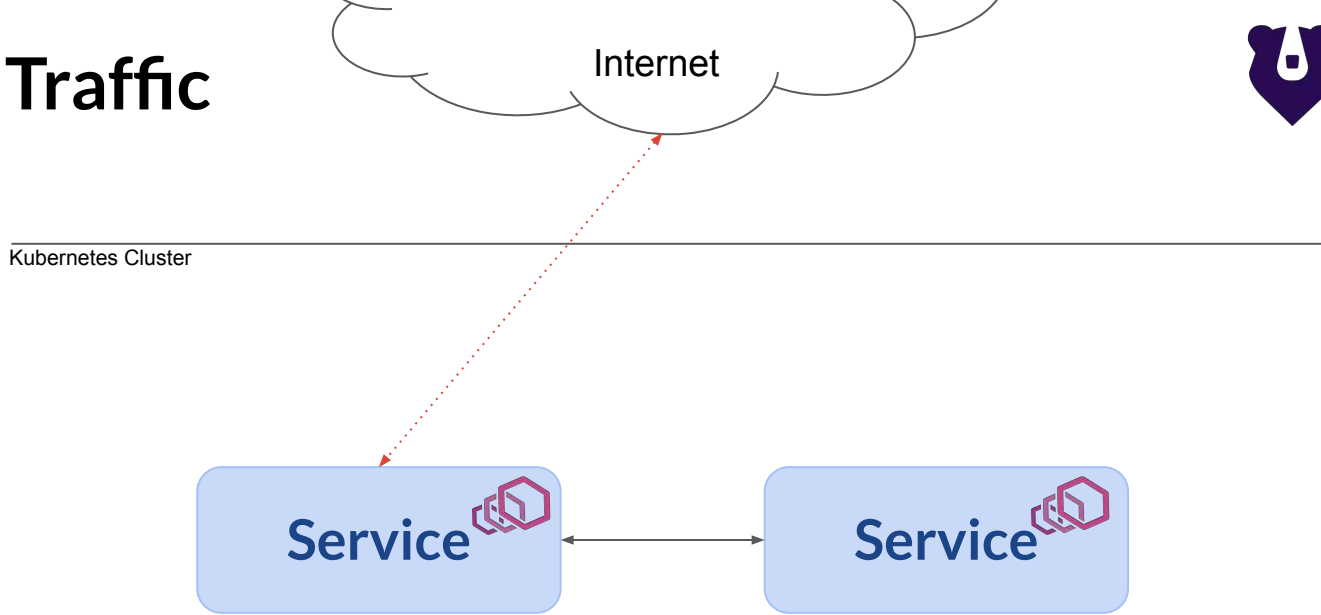
Our Gateway API Journey

- **Ingress** has historically been a top user configuration option
- For **extensibility** we've had multiple implementation-specific configurations
 - Custom Resource Definitions (CRDs)
 - Annotations
- We want users to have **common, upstream APIs** for routing and load-balancing
- We got involved in the early days, and continue to **help drive**
- We currently maintain **3 implementations**
 - Kong Kubernetes Ingress Controller (KIC)
 - Kong Gateway Operator (KGO)
 - Kuma & Kong Mesh

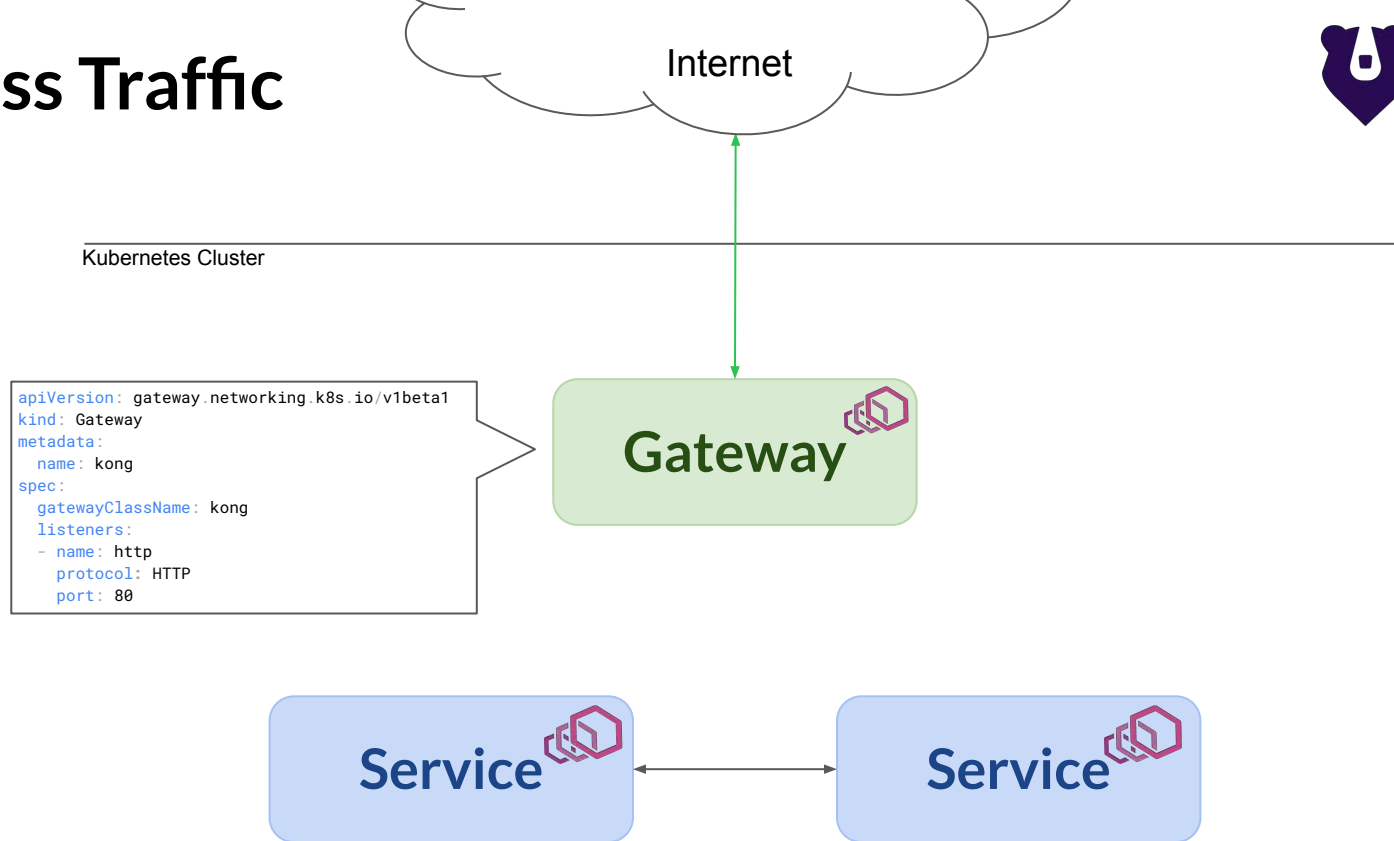
Service Mesh Traffic



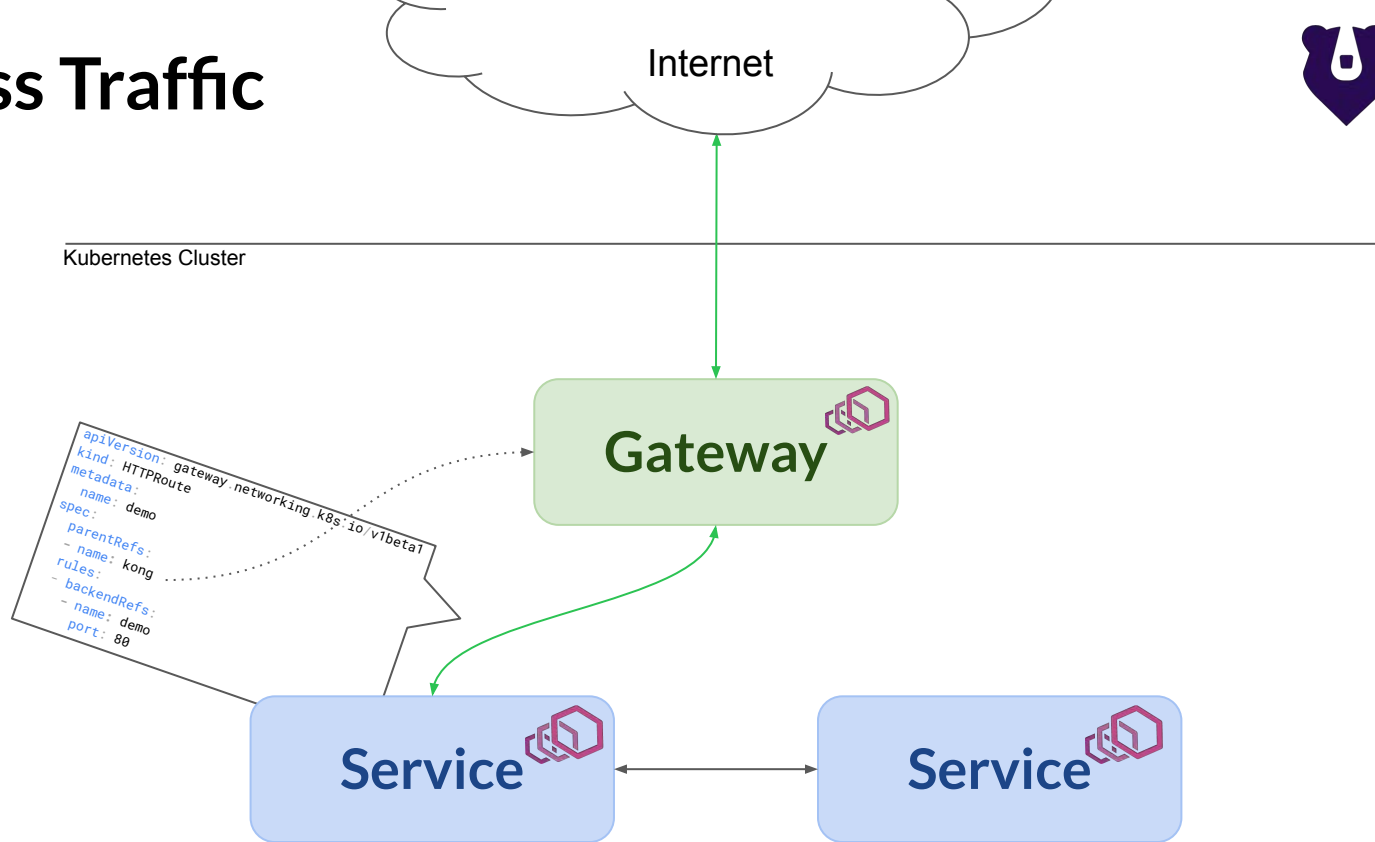
Ingress Traffic



Ingress Traffic



Ingress Traffic



Demo

Beyond HTTP

GRPCRoute

`apiVersion`: gateway.networking.k8s.io/v1alpha2

`kind`: GRPCroute

`metadata`:

`name`: grpcdemo

`spec`:

`parentRefs`:

 - `name`: kong

`hostnames`:

 - "example.com"

`rules`:

 - `backendRefs`:

 - `name`: grpcdemo

`port`: 443

TLSRoute

`apiVersion`: gateway.networking.k8s.io/v1alpha2

`kind`: TLSRoute

`metadata`:

`name`: tls-example

`spec`:

`parentRefs`:

 - `name`: kong

`hostnames`:

 - "example.com"

`rules`:

 - `backendRefs`:

 - `name`: tls-example

`port`: 8443

UDPRoute

`apiVersion`: gateway.networking.k8s.io/v1alpha2

`kind`: UDPRoute

`metadata`:

`name`: coredns

`spec`:

`parentRefs`:

 - `name`: kong

`rules`:

 - `backendRefs`:

 - `name`: coredns

`port`: 53

TCPRoute

`apiVersion`: gateway.networking.k8s.io/v1alpha2

`kind`: TCPRoute

`metadata`:

`name`: echo

`spec`:

`parentRefs`:

 - `name`: kong

`rules`:

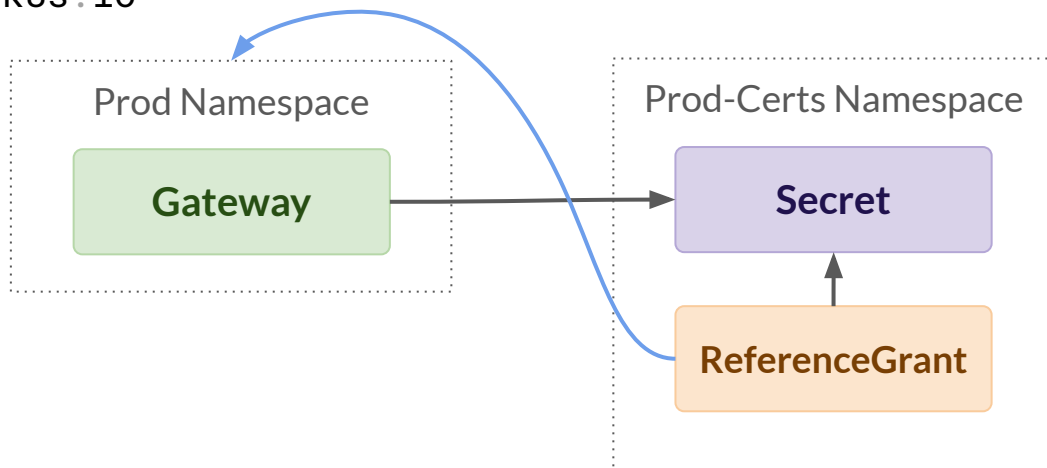
 - `backendRefs`:

 - `name`: echo

`port`: 8888

ReferenceGrant

```
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
```



Coming Next

General Availability (GA)

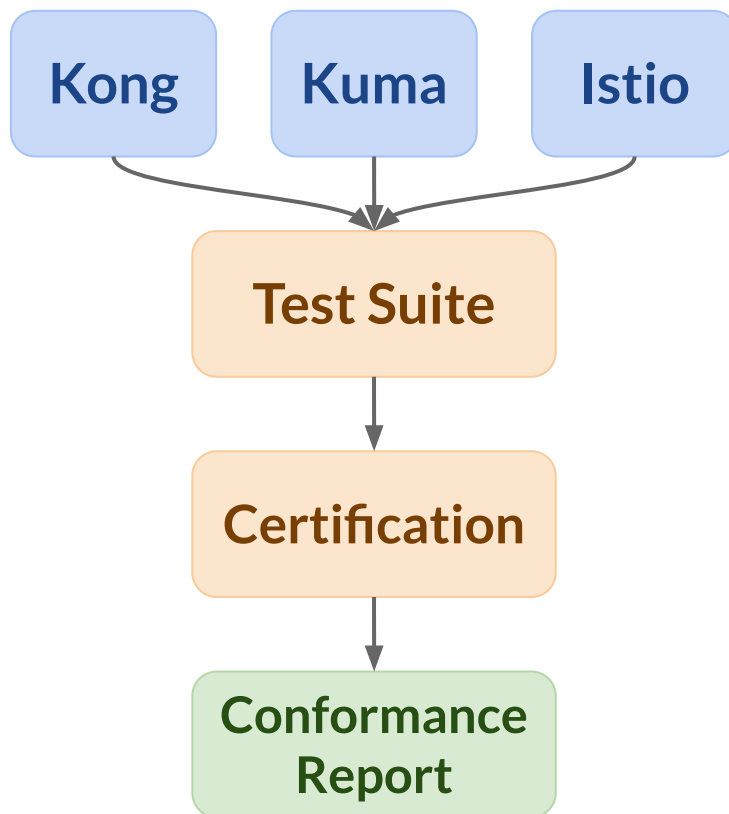
- No due date, but pushing for **this year**
- Version 1 APIs:
 - GatewayClass
 - Gateway
 - HTTPRoute
- Tracking via “Road to GA” project
 - <https://github.com/orgs/kubernetes-sigs/projects/30/views/1>
 - also **v1.0.0** milestone

Gamma Project

- Gateway API for Mesh Management and Administration
- Several contributors:
 - Consul Service Mesh
 - Istio
 - Kuma/Kong Mesh
 - Linkerd
 - and others!
- Using ***Route** resources in a **service mesh context**
- Current development focus is **conformance first HTTPRoute API usage**

Conformance Profiles

- High Level Testing Profiles
 - HTTP
 - GRPC
 - UDP, TCP
 - Mesh
 - e.t.c.
- Tooling
 - Test Suite
 - Automated Reporting
 - Certification Process
- In Development for GA (provisional)



Other Upcoming Prospects & Projects

- Experimental
 - Layer4 Support
 - Policy Attachment
 - Multi-Cluster Services
 - Path Redirects & Rewrites
- Provisional
 - Timeouts
 - Egress

Way down the road: Service Replacement

- Kubernetes **Service** API has become overloaded
- Replacement for **Load-Balancer** type **Services**
- You may start to see more routing and load-balancing functionality **shifted to Gateway API** in future Kubernetes versions

Exploring Further



Trying it out

- 20+ implementations & Integrations!
 - <https://gateway-api.sigs.k8s.io/implementations/>
- Kong's implementations:
 - Kong Kubernetes Ingress Controller (KIC)
 - Standard Ingress Controller, deployed via Helm
 - <https://docs.konghq.com/kubernetes-ingress-controller/latest/concepts/gateway-api/>
 - Kuma / Kong Mesh
 - Service mesh with ingress functionality
 - Actively helping to drive GAMMA
 - <https://kuma.io/docs/latest/explore/gateway-api/>
 - Kong Gateway Operator (KGO)
 - Kubernetes Operator with **Gateway** as a central resource
 - Currently in development preview
 - <https://github.com/kong/gateway-operator-docs>

Subprojects

- **GAMMA Project** (<https://gateway-api.sigs.k8s.io/contributing/gamma/>)
 - Supporting **Service Mesh** use cases and east/west traffic
 - Re-using existing ***Route** APIs as much as feasible
 - Early maturity and active development, please join us!
- **Ingress2Gateway** (<https://github.com/kubernetes-sigs/ingress2gateway>)
 - Converts **Ingress** resources to **Gateway API** resources
 - Annotation support in scope, still under development
 - Written in **Go**, can be used as a **Go library** or via **command line**
- **Blixt** (<https://github.com/kong/blixt> (moving to <https://github.com/kubernetes-sigs>))
 - Layer4 Load-Balancer (**TCPRoute**, **UDPRoute**, e.t.c.)
 - Control-plane in **Golang**, data-plane in **Rust** and **eBPF**
 - **NON-production** use cases: **Reference**, **testing** and **CI**

Community

- Website: <https://gateway-api.sigs.k8s.io/>
 - Main meetings every Monday
 - Gamma (Mesh) meetings every Tuesday
 - Code Jams on Fridays
- Discussion Boards:  <https://github.com/kubernetes-sigs/gateway-api/discussions>
- Kubernetes Slack:  <https://kubernetes.slack.com> (#sig-network-gateway-api)

Maintainers



Rob Scott
@Google



Shane Utt
@Kong



Nick Young
@Isovalent