All Aboard!

Kubernetes Routes Now Available for All Destinations North/South and East/West

Presented by Jef Spaleta - Isovalent

This is a story about Gateway API

This talk is also sort of an exploration of popular music about trains
(check the qr codes in the talk)

ISOVALENT

What is the Gateway API?

Gateway API is a redesign of Kubernetes Ingress, informed by years of operational experience, to address some short-comings in the original Ingress design.

Gateway API is the future of Ingress(North/South) ***and*** **Service Mesh(East/West) routing.**

The outcome of this work, in my opinion, is a much more durable set of routing abstractions that Kubernetes cluster operators and cluster users will be able to make use of with less friction.

Kubernetes Ingress is still being maintained, but the the cool new features to support more uses cases are happening in Gateway API

"Train kept a-rolling all night long"
- Train kept a-rolling (Aerosmith)

ISOVALENT

Imagine your cluster networking is a series of train routes

North/South Routes:
In/Out of your Kubernetes cluster

East/West Routes:
Between services in your cluster

Ref: https://www.wikihow.com/Play-Ticket-to-Ride

"Everybody loves a train in the distance"
- Train in the Distance (Paul Simon)

ISOVALENT

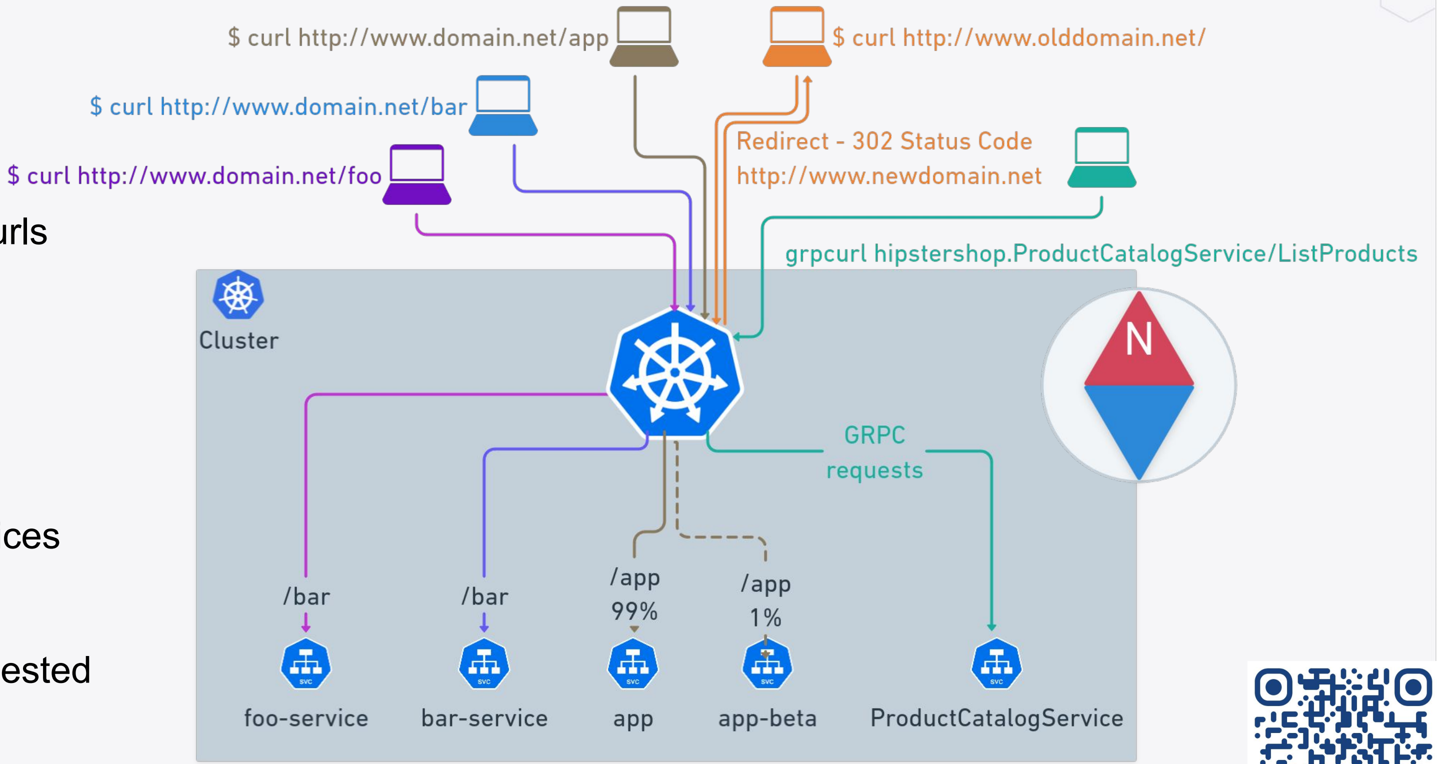
Traveling southbound into your cluster

North:
External Clients

Requesting a variety of urls using multiple protocols

South:
Multiple application services

Serving portions of the supported url paths requested by clients.



$ curl http://www.domain.net/app
$ curl http://www.olddomain.net/
$ curl http://www.domain.net/bar
Redirect - 302 Status Code
http://www.newdomain.net
$ curl http://www.domain.net/foo
grpcurl hipstershop.ProductCatalogService/ListProducts
Cluster
N
GRPC
requests
/bar
/bar
/app
99%
/app
1%
foo-service
bar-service
app
app-beta
ProductCatalogService

"Leaving, leaving here on a southbound train this morning, oh yes I am early this morning"
- Nobody Knows The Way I Feel This Morning (Aretha Franklin)

ISOVALENT

Traveling southbound into your cluster

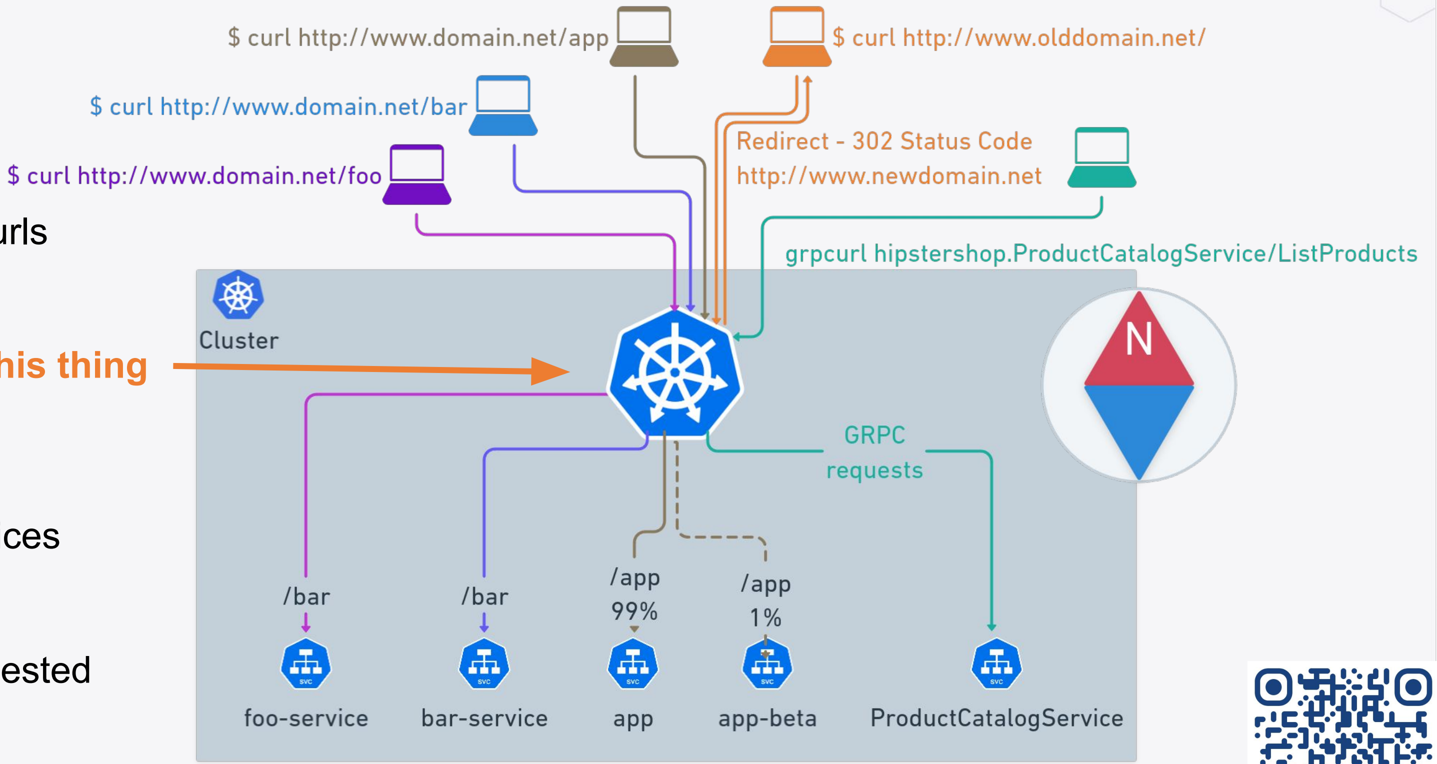
North:
External Clients

Requesting a variety of urls using multiple protocols

Let's talk about this thing

South:
Multiple application services

Serving portions of the supported url paths requested by clients.


$ curl http://www.domain.net/app
$ curl http://www.olddomain.net/
$ curl http://www.domain.net/bar
Redirect - 302 Status Code
http://www.newdomain.net
$ curl http://www.domain.net/foo
grpcurl hipstershop.ProductCatalogService/ListProducts
Cluster
N
GRPC requests
/bar
/bar
/app 99%
/app 1%
foo-service
bar-service
app
app-beta
ProductCatalogService

"Leaving, leaving here on a southbound train this morning, oh yes I am early this morning"
- Nobody Knows The Way I Feel This Morning (Aretha Franklin)

ISOVALENT

You need a bridge between the outside world and your cluster

Two prominent in-cluster solutions

Kubernetes Ingress:
This is the traditional choice and used heavily in production right now.

Gateway API:
Built on the experience of operating multiple Kubernetes Ingress implementations, addressing some of the shortcomings in the original design to help scale better in multiple team orgs.

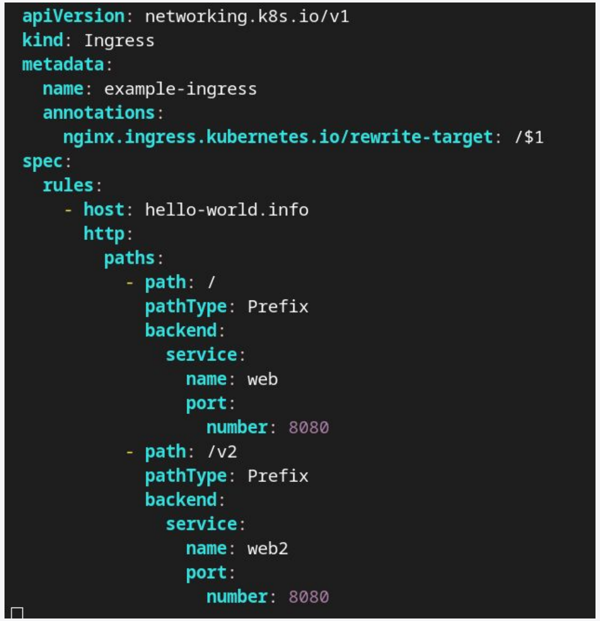
Both implement:
Reverse proxy functionality and loadbalancing capabilities inside your cluster.

The intent is for the Gateway API design to be more durable, verifiable, and extensible

"From the wrong side of the tracks, from the bad side of the town, that's where she grew up"
- The Wrong Side of the Tracks (Brian Setzer)

ISOVALENT

Traditional Kubernetes Ingress - A reverse proxy for your cluster



```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: example-ingress
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /$1
spec:
  rules:
    - host: hello-world.info
      http:
        paths:
          - path: /
            pathType: Prefix
            backend:
              service:
                name: web
                port:
                  number: 8080
          - path: /v2
            pathType: Prefix
            backend:
              service:
                name: web2
                port:
                  number: 8080
```

Works well when the user facing web service is a collection of services inside a single K8s namespace.

Not designed to access cross-namespace services.

What happens when multiple teams working in their own dedicated namespace need their own ingress?

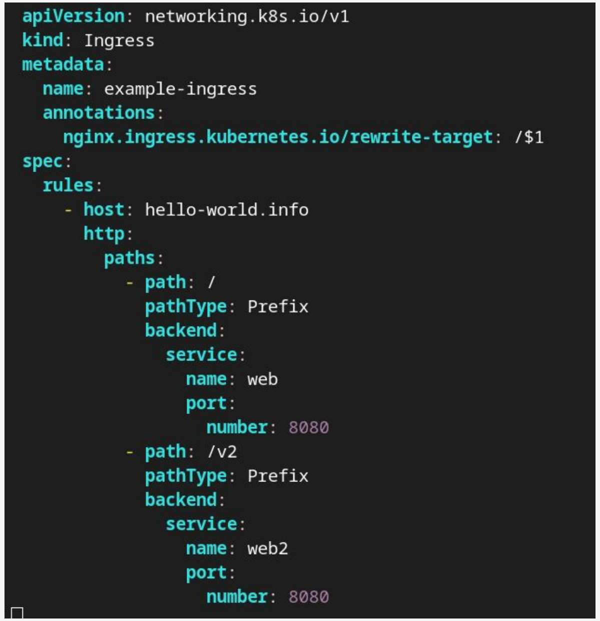
What happens when teams working in different namespaces need to depend on each other?

[Ref: https://kubernetes.io/docs/tasks/access-application-cluster/ingress-minikube/](https://kubernetes.io/docs/tasks/access-application-cluster/ingress-minikube/)

"Take the last train to Clarksville, and I'll meet you at the station"
- The Last Train to Clarksville (The Monkees)

ISOVALENT

The Problem - Who owns the Ingress resource?



```yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: example-ingress
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /$1
spec:
  rules:
    - host: hello-world.info
      http:
        paths:
          - path: /
            pathType: Prefix
            backend:
              service:
                name: web
                port:
                  number: 8080
          - path: /v2
            pathType: Prefix
            backend:
              service:
                name: web2
                port:
                  number: 8080
```

Ingresses can be in multiple namespaces

Application Developer

The paths are an application developer concern!

Only application developers know what's needed here.

Cluster Operator

The host field is a cluster operator concern!

What happens if ingresses in different namespaces refer to the same host?

Needs a way to prevent rule collisions across teams.

"Runaway train never going back. Wrong way on a one-way track."
- Runaway Train (Soul Asylum)

ISOVALENT

The Solution - Gateway API design driven by personas

Infrastructure Provider

Provisions cluster and provides external connectivity

Cluster Operator

Ensures the cluster is configured so it works well for all cluster users
May have broad cluster admin access

Application Developer

Controls a specific cluster application namespace
Access tightly restricted by cluster RBAC
Must coordinate with other teams or operators for changes outside of namespace

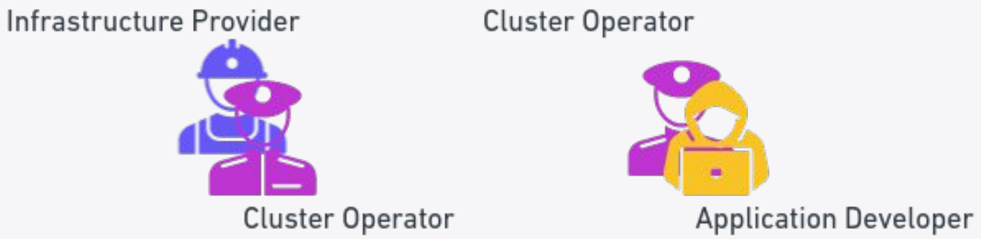
"And the train conductor says, take a break, driver 8. Driver 8, take a break, we can reach our destination..."
- Driver 8 (R. E. M.)

ISOVALENT

Key Idea: Scoping resources to personas

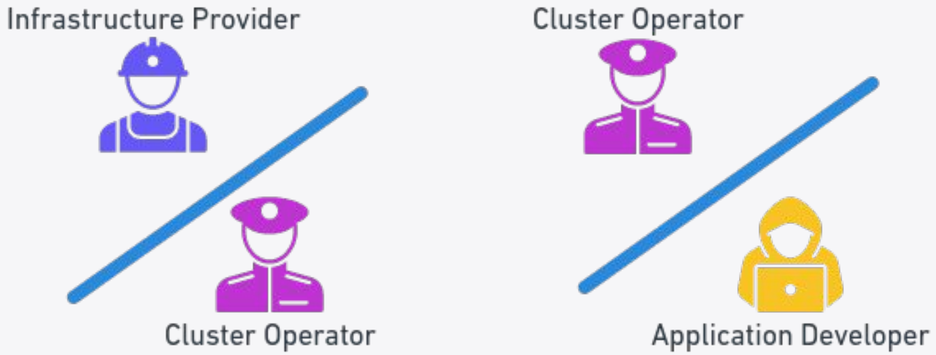
Kubernetes Ingress:
Difficult to separate out
day-to-day activities by persona

Adds friction to app team self-servicing



Gateway API:
Better separation of concerns
through well scoped resources

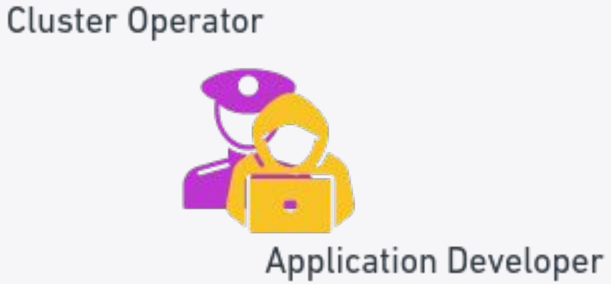
Lowers friction to app team self-servicing

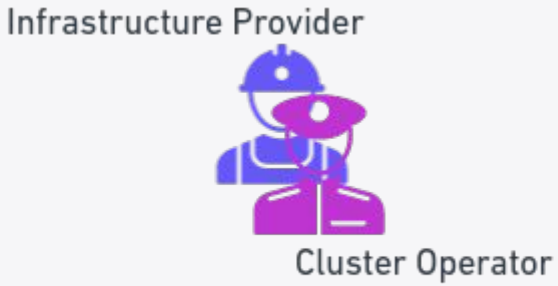


"Mental wounds still screaming. Driving me insane,I'm going off the rails on a crazy train"
- Crazy Train (Ozzy Osbourne)

ISOVALENT

Traditional Kubernetes Ingress - Resource configuration crosses persona boundaries





Technical Details:

Ingress: The ***resource*** you as a cluster operator/user would create with desired traffic routing rules

IngressClass: A ***resource*** with additional configuration for the underlying controller(s)

IngressController: is the implementation responsible for fulfilling the Ingress

Ingress Loadbalancing Service:
A service created by the controller to expose the Ingress to the outside world, may need manual configuration

"I read my schedule but my train wasn't there. I re-read my ticket, it said going nowhere"
- Right Train, Wrong Track (Cyndi Lauper)

ISOVALENT

Gateway API design - Ingress resources better scoped to human roles

Infrastructure Provider

Gateway Controller: is the implementation responsible for fulfilling the gateway routing rules, usually provisioned by

GatewayClass: A ***resource*** with additional configuration for the underlying controller(s)

Cluster Operator

Gateway: The ***resource*** you as a cluster operator would configure gateway loadbalancing services, and set policy as to which application namespaces were allowed to attach routes

Gateway LoadBalancer Service:
A service created by *most* implementations to expose configured Gateway resources to the outside world. Note: this may also be an externally provided LoadBalancer.

Application Developer

Routes:** The ***resources that can be attached to Gateways that define application specific routing rules.

ReferenceGrants: The ***resources*** used by application teams to coordinate with each other and allow cross-namespace routing.

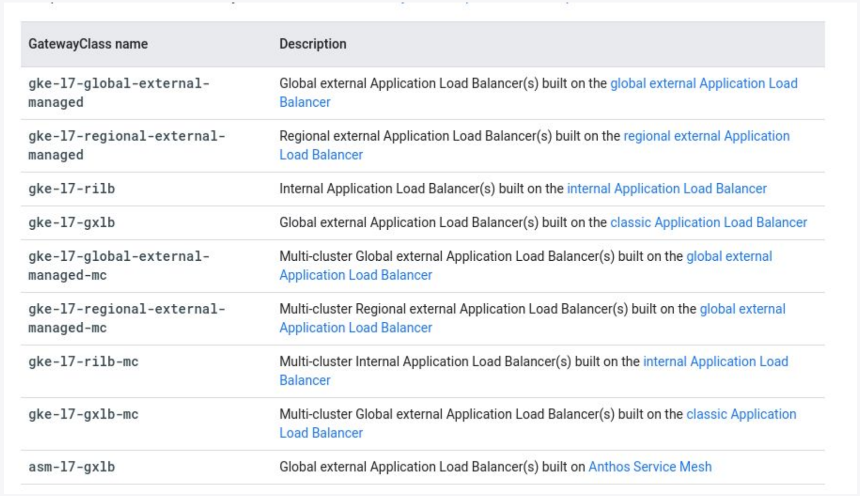
"Train, train, comin' 'round, 'round the bend"
- Mystery Train (Elvis Presley)

ISOVALENT

The GatewayClass Resource: Multiple classes for different infrastructure scopes

The *GatewayClass* resources is meant to encode infrastructure capabilities, and is extensible via *parametersRef*

Infrastructure Provider



GatewayClass name	Description
`gke-l7-global-external-managed`	Global external Application Load Balancer(s) built on the global external Application Load Balancer
`gke-l7-regional-external-managed`	Regional external Application Load Balancer(s) built on the regional external Application Load Balancer
`gke-l7-rilb`	Internal Application Load Balancer(s) built on the internal Application Load Balancer
`gke-l7-gxlb`	Global external Application Load Balancer(s) built on the classic Application Load Balancer
`gke-l7-global-external-managed-mc`	Multi-cluster Global external Application Load Balancer(s) built on the global external Application Load Balancer
`gke-l7-regional-external-managed-mc`	Multi-cluster Regional external Application Load Balancer(s) built on the global external Application Load Balancer
`gke-l7-rilb-mc`	Multi-cluster Internal Application Load Balancer(s) built on the internal Application Load Balancer
`gke-l7-gxlb-mc`	Multi-cluster Global external Application Load Balancer(s) built on the classic Application Load Balancer
`asm-l7-gxlb`	Global external Application Load Balancer(s) built on Anthos Service Mesh

Ref: https://cloud.google.com/kubernetes-engine/docs/concepts/gateway-api

"The work is hard in a railroad yard, Hey, hey, gotta' make it today to punch a time card. Workin' on the railroad."
- The Railroad (Grand Funk Railroad)

ISOVALENT

The Gateway Resource: The cluster operator's friend!

Cluster Operator

Questions for cluster operator to think about when setting up any Ingress:

- Do you want a common shared gateway for all teams or do you want each namespace to have its own gateway? Maybe something in-between?
- What protocols do you want the gateway to handle and what ports do you want to listen on?
- How will hostnames be matched to applications namespaces?

The Gateway resources is meant to be used to encode cluster operations policy to answer cluster operational questions like these

"Hold that train, conductor, please don't let that engineer start"
- Hold That Train (BB King)

ISOVALENT

The *Routes resources: What application developers actually care about

Application Developer

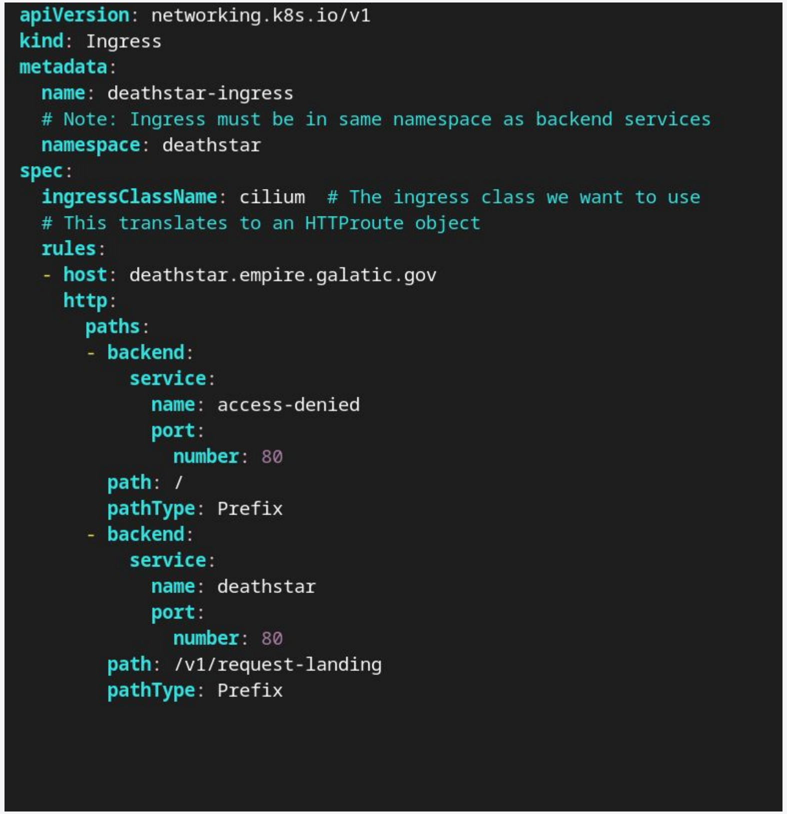
Questions for app developer teams

- Which URL paths do you want to route to which services
- Do you need to route to services maintained by other teams in namespaces you don't control?
- Do you need to reference any TLS certificates or other namespaced resources?

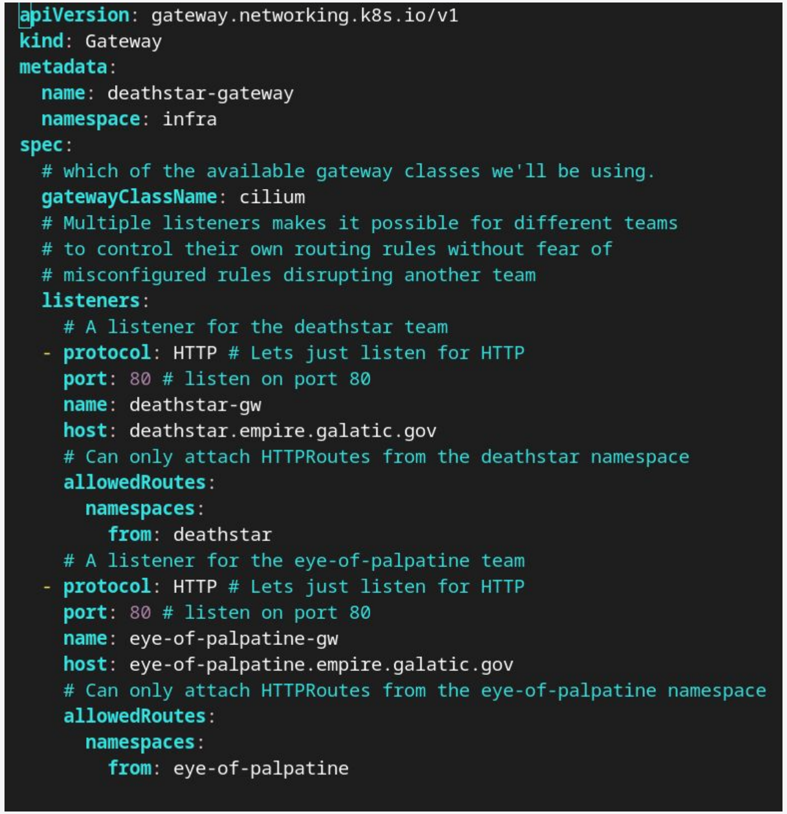
"Well, pistons keep on churnin' and the wheels go 'round and 'round"
- Long Runnin Train (The Doobie Brothers)

ISOVALENT

A Gateway and Ingress example comparison



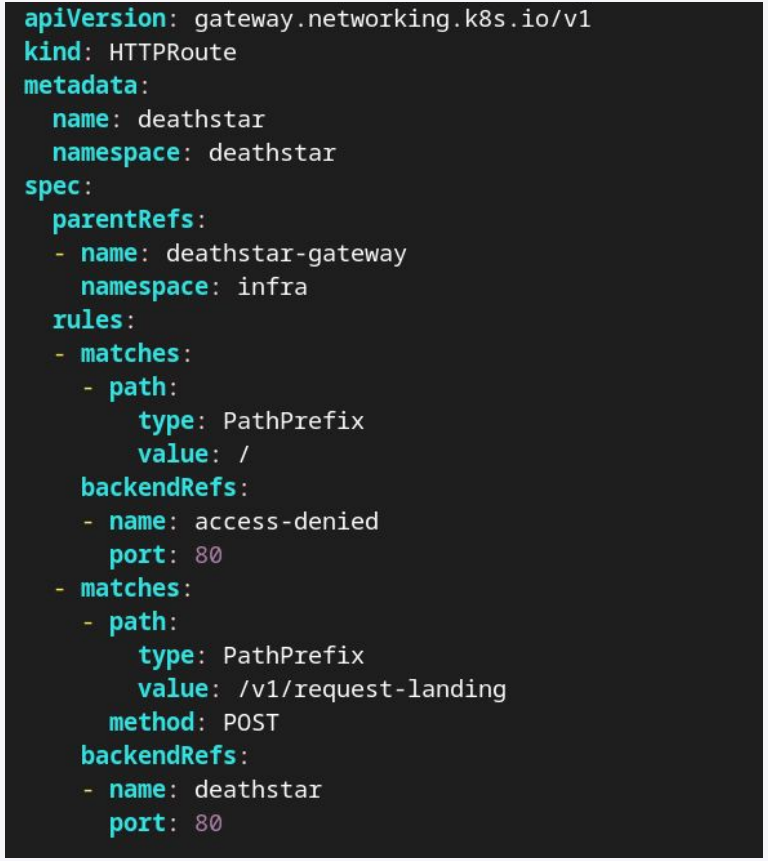
```yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: deathstar-ingress
  # Note: Ingress must be in same namespace as backend services
  namespace: deathstar
spec:
  ingressClassName: cilium  # The ingress class we want to use
  # This translates to an HTTProute object
  rules:
  - host: deathstar.empire.galatic.gov
    http:
      paths:
      - backend:
          service:
            name: access-denied
            port:
              number: 80
        path: /
        pathType: Prefix
      - backend:
          service:
            name: deathstar
            port:
              number: 80
        path: /v1/request-landing
        pathType: Prefix
```



```yaml
apiVersion: gateway.networking.k8s.io/v1
kind: Gateway
metadata:
  name: deathstar-gateway
  namespace: infra
spec:
  # which of the available gateway classes we'll be using.
  gatewayClassName: cilium
  # Multiple listeners makes it possible for different teams
  # to control their own routing rules without fear of
  # misconfigured rules disrupting another team
  listeners:
    # A listener for the deathstar team
  - protocol: HTTP # Lets just listen for HTTP
    port: 80 # listen on port 80
    name: deathstar-gw
    host: deathstar.empire.galatic.gov
    # Can only attach HTTPRoutes from the deathstar namespace
    allowedRoutes:
      namespaces:
        from: deathstar
    # A listener for the eye-of-palpatine team
  - protocol: HTTP # Lets just listen for HTTP
    port: 80 # listen on port 80
    name: eye-of-palpatine-gw
    host: eye-of-palpatine.empire.galatic.gov
    # Can only attach HTTPRoutes from the eye-of-palpatine namespace
    allowedRoutes:
      namespaces:
        from: eye-of-palpatine
```

ISOVALENT

The HTTPRoute Resource: Where the application specific routing rules live



```yaml
apiVersion: gateway.networking.k8s.io/v1
kind: HTTPRoute
metadata:
  name: deathstar
  namespace: deathstar
spec:
  parentRefs:
  - name: deathstar-gateway
    namespace: infra
  rules:
  - matches:
    - path:
        type: PathPrefix
        value: /
    backendRefs:
    - name: access-denied
      port: 80
  - matches:
    - path:
        type: PathPrefix
        value: /v1/request-landing
      method: POST
    backendRefs:
    - name: deathstar
      port: 80
```

Looks a lot like the Ingress rules...

Except both ***ParentRefs*** and ***BackendRefs*** can cross namespace boundaries making several new use-cases possible.

Note: ****Refs*** in general are intended to be extensible and its expected implementations will support a variety of extended types in ****Refs***

ISOVALENT

Routes aren't just for the HTTP protocol!

The available protocols a Gateway controller implementation supports is intended to be extensible.

Each supported protocol will have an associated Route resource.

Routes in development as part of Gateway API:

- HTTPRoute: reached GA status in the Gateway API v1 release
- TLSRoute
- GRPCRoute
- TCPRoute
- UDPRoute

[Ref: https://gateway-api.sigs.k8s.io/reference/spec/#gateway.networking.k8s.io%2fv1alpha2HTTPRoute](https://gateway-api.sigs.k8s.io/reference/spec/#gateway.networking.k8s.io%2fv1alpha2HTTPRoute)

"I said, Now look a yonder coming, coming down that long railroad track"
- Long Black Train (Conway Twitty)

ISOVALENT

Ingress migration tool in development!

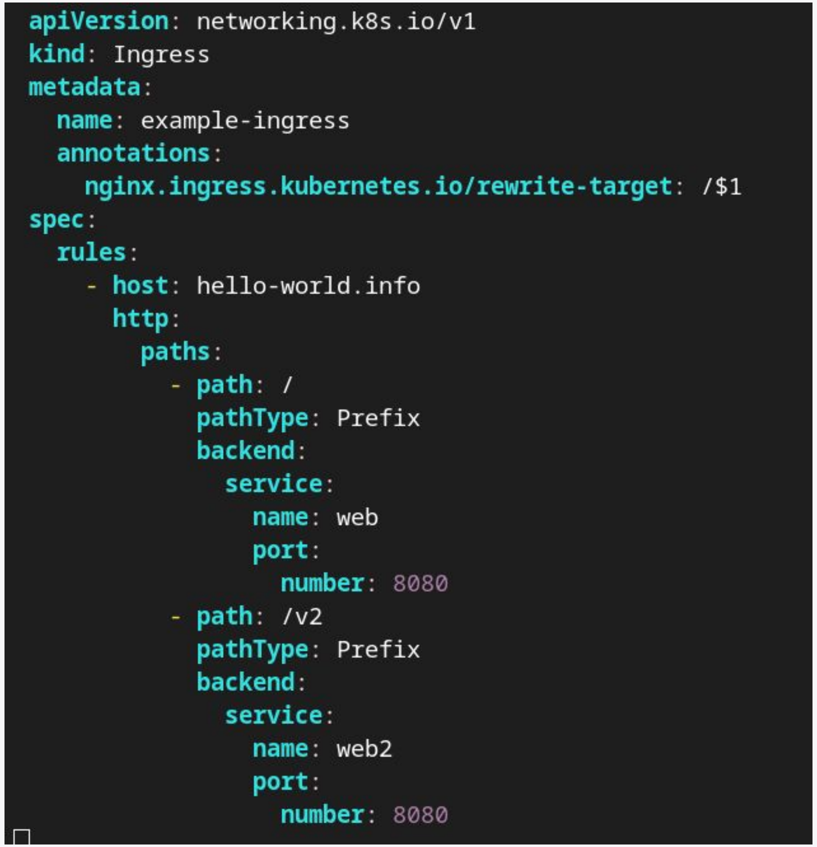
https://github.com/kubernetes-sigs/ingress2gateway

"Ingress2gateway helps translate Ingress and provider-specific resources (CRDs) to Gateway API resources. Ingress2gateway is managed by the Gateway API SIG-Network subproject."

"I don't know what train I'm on. Won't you tell me, before I'm gone"
- Freight Train (Van Morrison)

ISOVALENT

Separation of concerns isn't the only concern with Kubernetes Ingress



```yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: example-ingress
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /$1
spec:
  rules:
    - host: hello-world.info
      http:
        paths:
          - path: /
            pathType: Prefix
            backend:
              service:
                name: web
                port:
                  number: 8080
          - path: /v2
            pathType: Prefix
            backend:
              service:
                name: web2
                port:
                  number: 8080
```

Kubernetes Ingress implementations make heavy use of implementation specific annotations.

Example: extending ingress to GRPC

Annotations are notoriously hard to validate.

Are you sure you are using the correct annotations to match to features available in an Ingress controller implementation?

Annotations are NOT portable across Ingress implementations

Gateway API's design is meant to address this by making sure there are resource based extension points in key places (such as the ***Refs**)

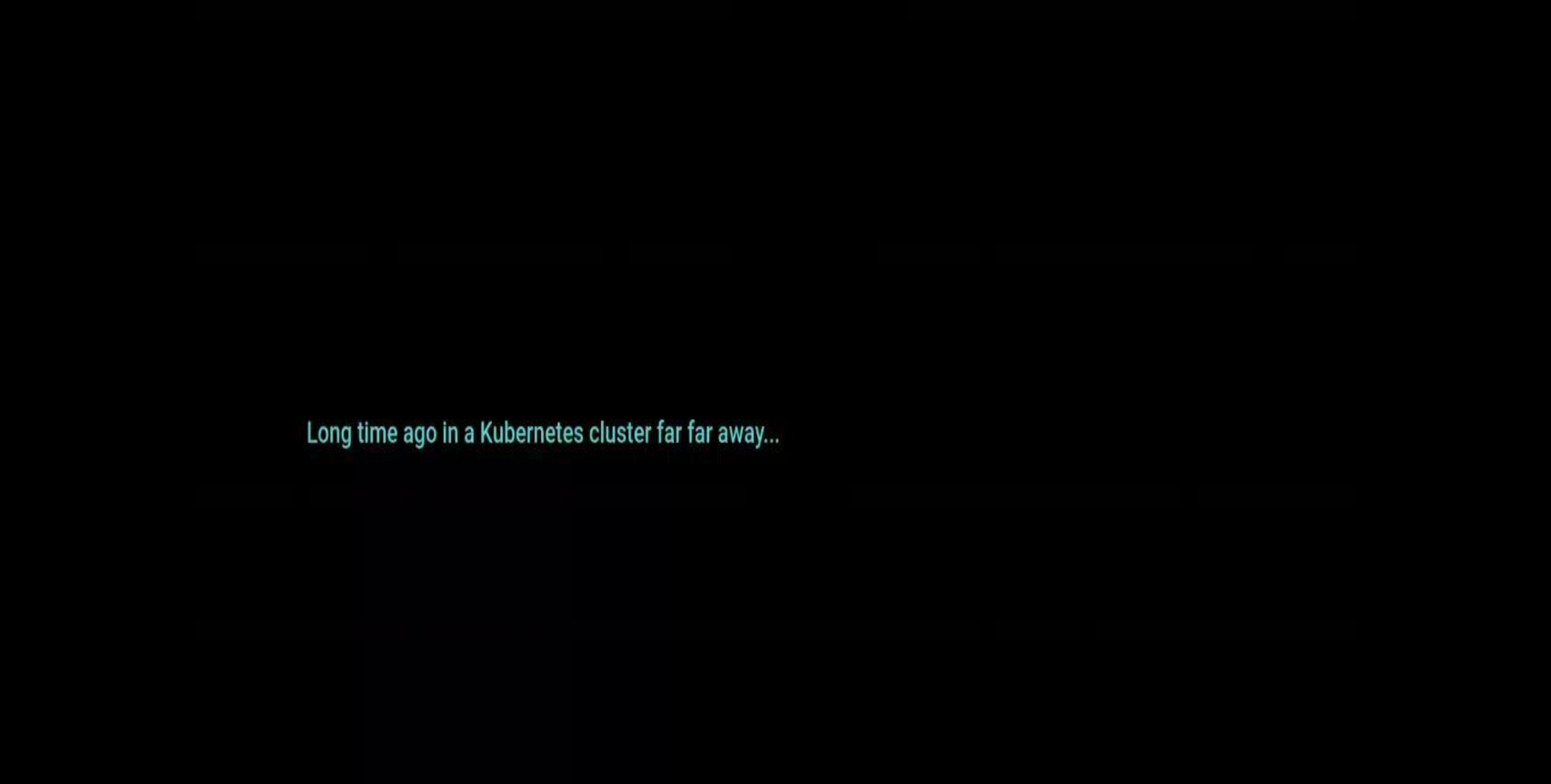
Star Wars
Gateway API Demo
Episode IV

A New Ingress

You're part of the Empire's platform engineering team, and you need to roll out a centralized landing request service that ALL the Imperial bases and secret superweapon application developers can use.

It's become clear from past incidents that trying to keep individual landing request services up-to-date with the current codes has been less than successful. Purging older codes hasn't been top priority everywhere, and it's led to some unfortunate minor security breaches at some orbital battle station platforms.

Demo Time


Long time ago in a Kubernetes cluster far far away...

Made with: https://starwarsintrogenerator.com/

Terminal Time

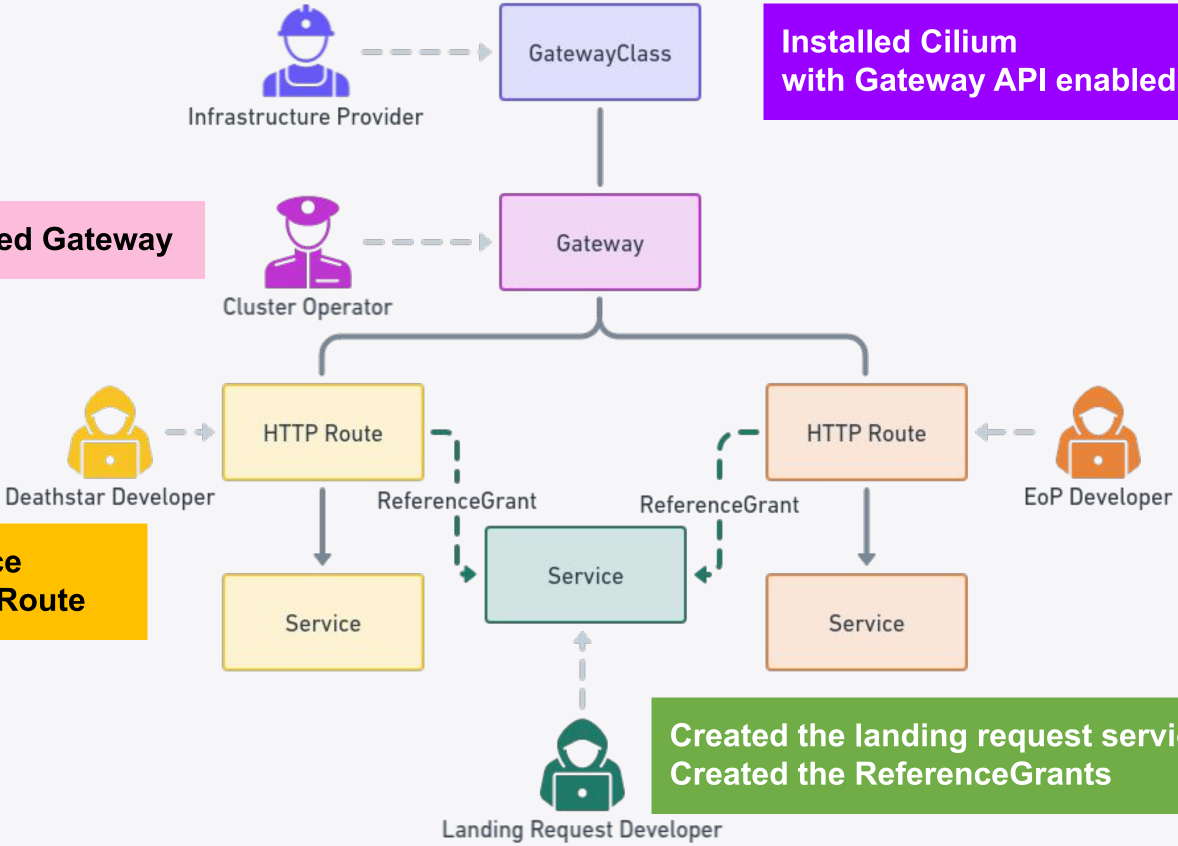
Star Wars Gateway API Demo

Episode IV: A New Ingress

Set up a Shared Gateway API Gateway and HTTPRoutes for a common landing service

ISOVALENT

Let's break the demo actions down by persona



Installed Cilium with Gateway API enabled

Created the shared Gateway

Created the deathstar service
Created the deathstar HTTPRoute

Created the landing request service
Created the ReferenceGrants

ISOVALENT

Key demo takeaways

Gateways map host,port,address into listeners and set policy as to which namespaces can attach which kind of routes.

***Routes** express application specific routing rules, and can even reference services across namespace boundaries.

ReferenceGrants are really powerful ways for developers to be explicit about intended use of resource dependencies that cross namespace boundaries.

Note: its also possible to use ***ReferenceGrants*** to control access to namespaced certificate secrets for TSLRoutes and HTTPSRoutes in a cross-namespace way.

ISOVALENT

The ReferenceGrant resource: Makes cross-namespace dependencies explicit

Application Developer

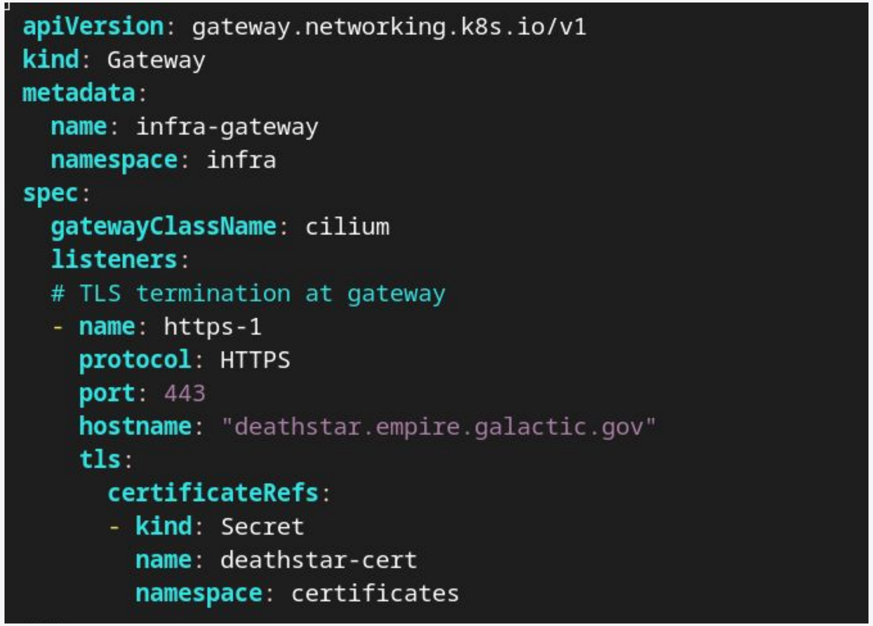
Questions for app developer teams

- Do you want other application teams working (and constrained by RBAC) in other namespaces to be able to route to your services?
- What about other resources (like TLS certs?)
- How do you know which other teams are routing to your team's services or depending on your TLS certs?

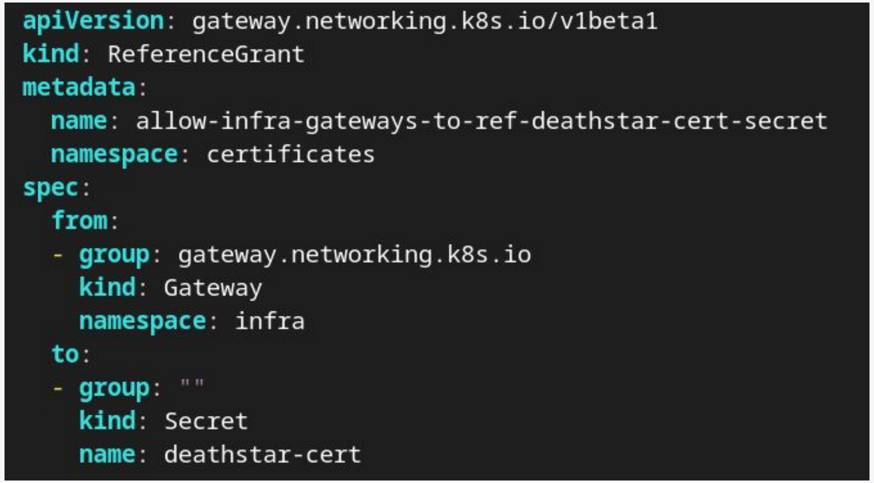
""People all over the world (everybody) join hands (join) start a love train, love train
- Love Train (The O'Jays)

ISOVALENT

Example of ReferenceGrant for TLS Certificate Secret



```yaml
apiVersion: gateway.networking.k8s.io/v1
kind: Gateway
metadata:
  name: infra-gateway
  namespace: infra
spec:
  gatewayClassName: cilium
  listeners:
  # TLS termination at gateway
  - name: https-1
    protocol: HTTPS
    port: 443
    hostname: "deathstar.empire.galactic.gov"
    tls:
      certificateRefs:
      - kind: Secret
        name: deathstar-cert
        namespace: certificates
```



```yaml
apiVersion: gateway.networking.k8s.io/v1beta1
kind: ReferenceGrant
metadata:
  name: allow-infra-gateways-to-ref-deathstar-cert-secret
  namespace: certificates
spec:
  from:
  - group: gateway.networking.k8s.io
    kind: Gateway
    namespace: infra
  to:
  - group: ""
    kind: Secret
    name: deathstar-cert
```

Gateway needs TLS cert in different namespace to perform TLS termination
Perhaps there is a separate security persona in charge of the certs?

Demo Time

Star Wars
Gateway API Demo
Episode V

The Traffic Control Team Strikes Back

The Traffic Control team has introduce a new version of the landing request service.
You'r job is to update the Deathstar's HTTPRoute to split traffic between multiple versions of the Traffic Control team's landing request service

Long time ago in a Kubernetes cluster far far away...

Made with: https://starwarsintrogenerator.com/

Terminal Time

ISOVALENT

Key demo takeaways

Gateways are cluster operational: They must be configured correctly to avoid having multiple teams disrupting each other, regardless if they are in a single namespace or not. Misconfigured gateways (like Ingress) can disrupt multiple teams.

Shared Gateways can potentially help protect teams from disrupting each by isolating teams through the use of multiple ***listeners***.

The ****Routes*** attached to the ***shared Gateway*** are application specific policy. If they are misconfigured they only impact the application team(s) allowed to attach to a given ***Gateway listener***.

Using multiple weighted ***BackendRefs*** in ***HTTPRoute*** rules provides loadbalancer-like behavior with fail-over protection.

ISOVALENT

Gateway API North/South Recap

Gateway API reached v1.0 in the past year with support for:

- Gateway
- GatewayClass
- HTTPRoute

This forms the initial basis for being the next generation of Kubernetes Ingress. But its just the start! There are many ***Gateway enhancement proposals (GEPs)*** being worked on by the community.

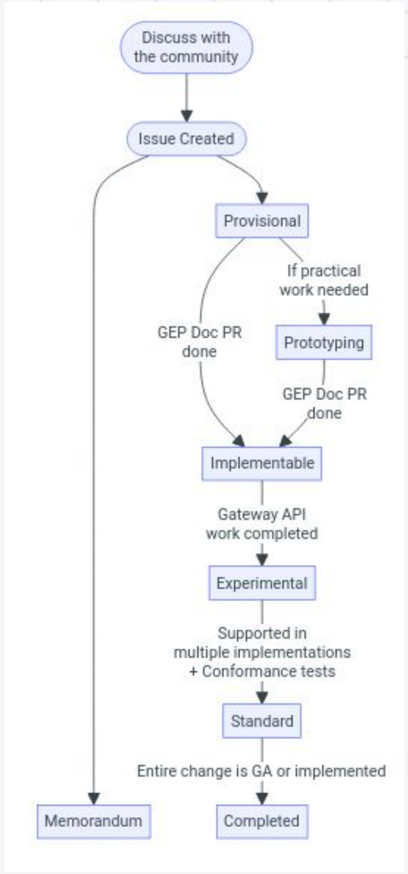
If you're interested by what you've seen so far, you should keep tabs on the in-progress GEPs, and see how the community is working to build on this foundation.

ISOVALENT

Some experimental GEPs of note

- **Route Port Matching [#957](#)**
- **Mesh Service Binding [#1294](#) -> East/West routing!**
- **GatewayClass status supported features [#2162](#)**

Ref: https://github.com/orgs/kubernetes-sigs/projects/20/views/1?filterQuery=status%3AExperimental+



Discuss with the community
Issue Created
Provisional
If practical work needed
Prototyping
GEP Doc PR done
GEP Doc PR done
Implementable
Gateway API work completed
Experimental
Supported in multiple implementations + Conformance tests
Standard
Entire change is GA or implemented
Memorandum
Completed

The GEP lifecycle

"I hear the train a comin' It's rolling round the bend"
- Folsom Prison Blues (Jonny Cash)

ISOVALENT

Can Gateway API handle East/West routes? Yes!

Made possible by a series of GEPs introduced by the "GAMMA Initative"

Implemented by extending ***Route ParentRef*** fields
to allow ***kind: Service*** instead of assuming ***kind: Gateway***

Makes it possible to use ***Route*** resources to instruct ***service mesh controllers*** to route traffic between ***service facets*** inside your cluster

Note: This is experimental/beta feature in several service mesh controllers (kuma, linkerd, istio) and is under active development.

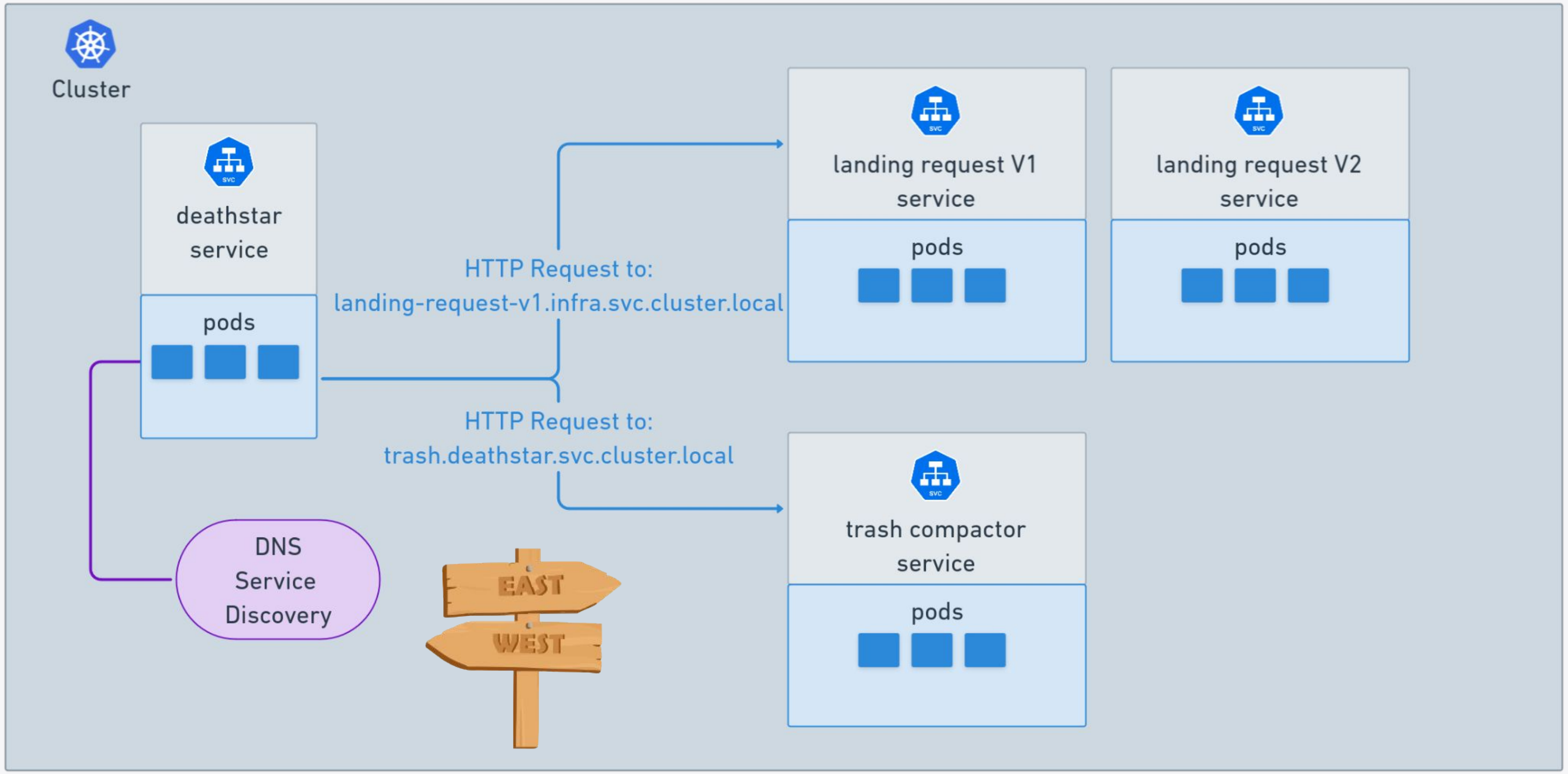
Note: Planned to be included in Cilium as well

Ref: https://github.com/cilium/cilium/issues/22512

"Roll down your window, brother, shout out my name You're on route 66, on a blacktop train"
- Black Top Train (Ellis Paul)

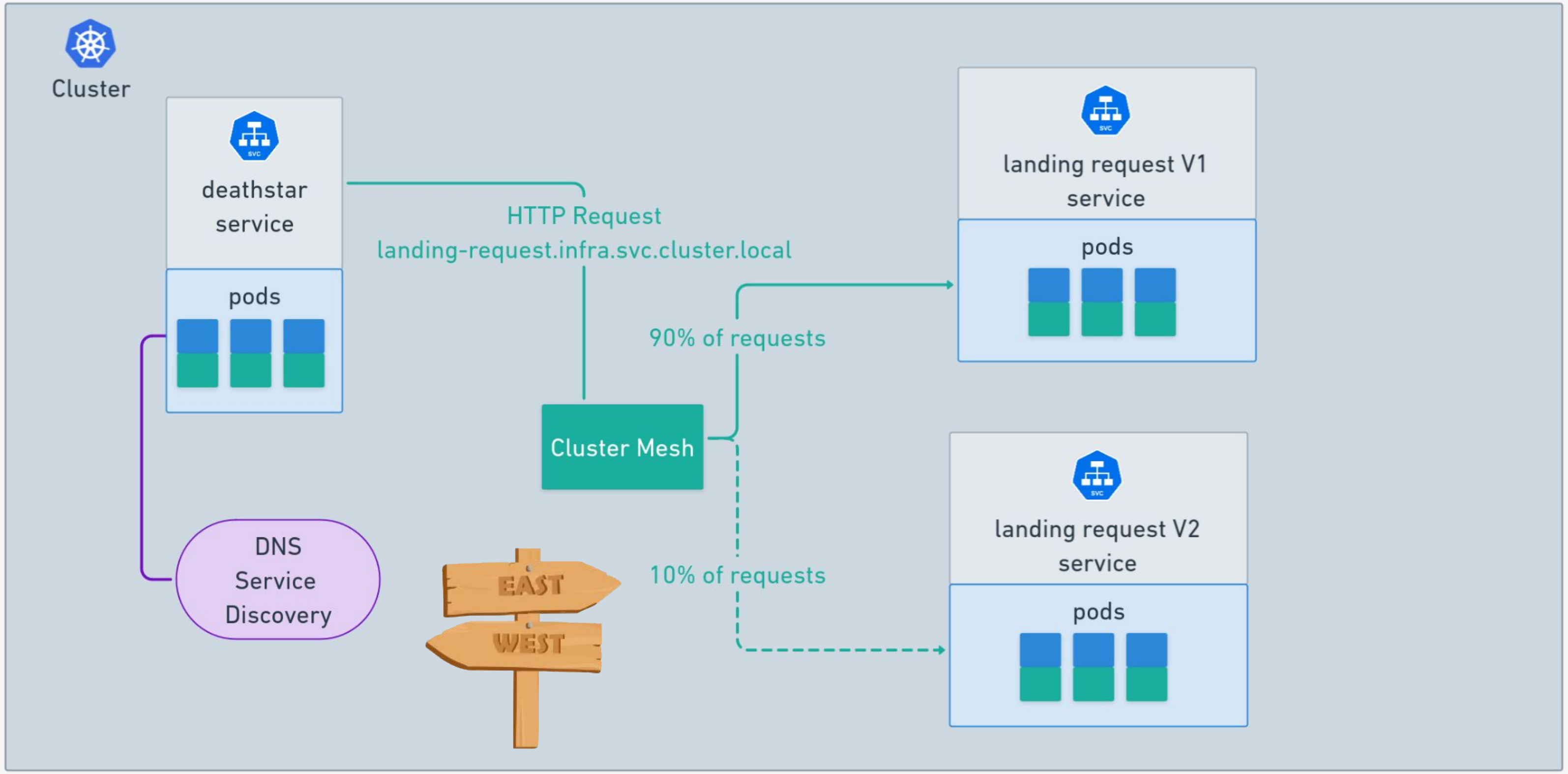
ISOVALENT

East/West Routes - When a cluster service calls another


Cluster
deathstar service
pods
HTTP Request to:
landing-request-v1.infra.svc.cluster.local
landing request V1 service
pods
landing request V2 service
pods
HTTP Request to:
trash.deathstar.svc.cluster.local
trash compactor service
pods
DNS Service Discovery
EAST
WEST

ISOVALENT

Cluster Mesh adds East/West routing flexibility like a gateway!



ISOVALENT

What is a Kubernetes Service really?

A Kubernetes ***Service*** is a construct that connects ***frontend*** endpoint(s) with a set of ***backend*** endpoints

The ***backends*** are an explicit ***facet*** of a Service resource and are *usually*[1] pods matching some labeling criteria.

The ***frontends*** are an implicit ***facet*** of a Service resource, controlled by the Service type:

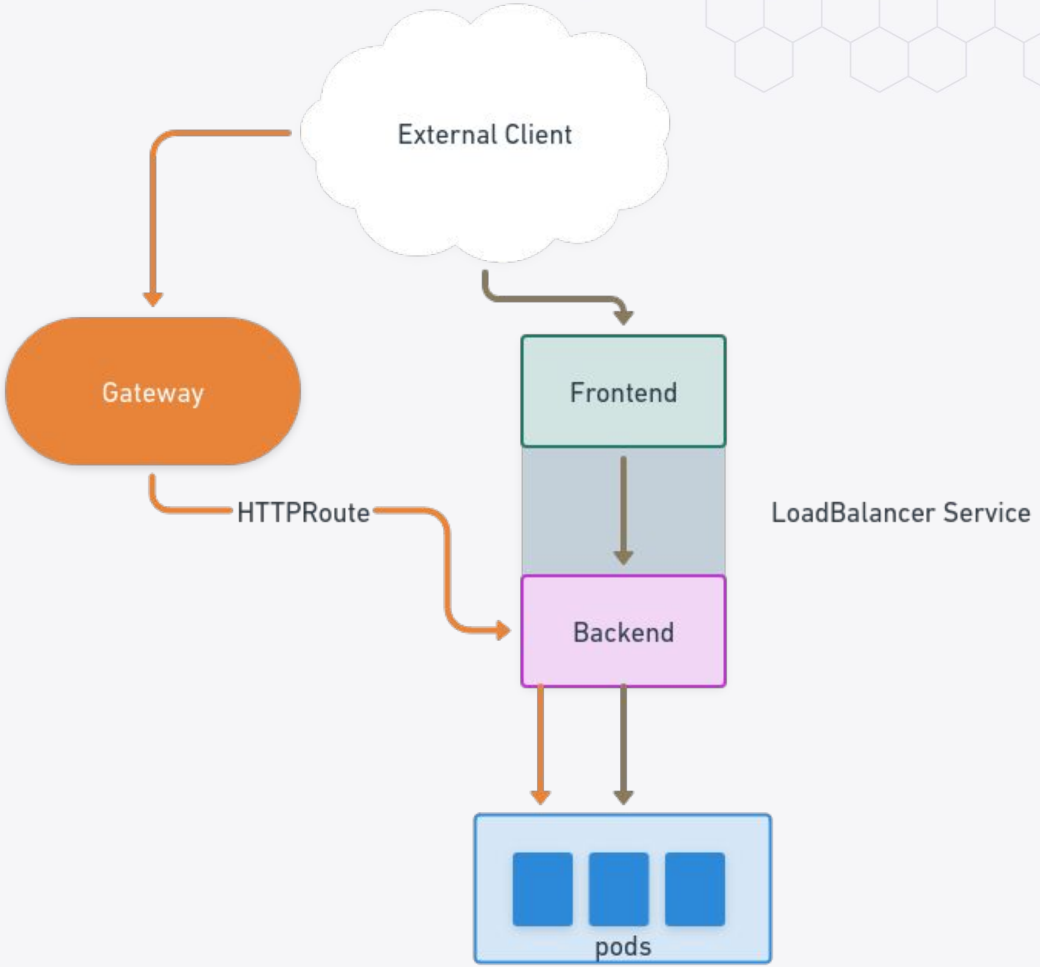
- For ClusterIP services the frontend endpoints are given IP addresses routable only inside the cluster, and a cluster-internal DNS record is created for service discovery.
- For LoadBalancer services the frontend endpoints are additionally given an externally routable IP address.
- For NodePort services the frontend endpoints are additionally assigned a port on the host's externally addressable interface.

Note 1: Headless services are effectively frontend facet with no backend facet. Neither Gateway nor Service ParentRefs work with Headless services.

ISOVALENT

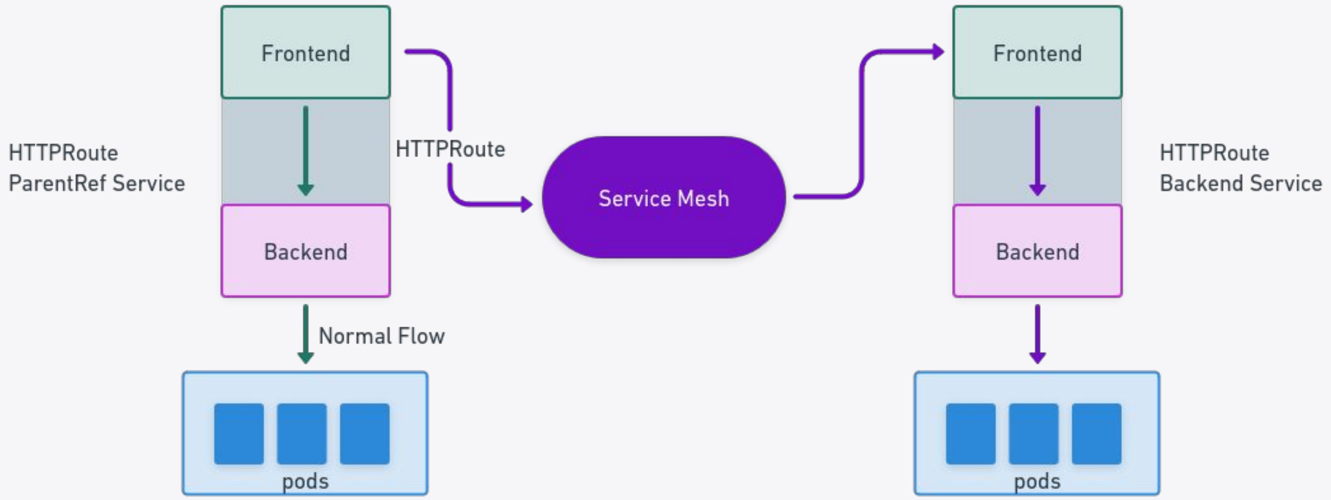
Service facets are fascinating

When referencing a ***Gateway*** as the ***ParentRef*** in an ***HTTPRoute***, you are essentially ignoring the **frontend** facet of the ***HTTPRoute BackendRef Services***.



ISOVALENT

Service facets are fascinating



When referencing a ***ParentRef Service*** in an ***HTTPRoute***, you are essentially ignoring the ***backend*** facet of that ***ParentRef Service***, and telling the ***Service Mesh*** controller to use the ***frontend*** of ***HTTPRoute BackendRef Services*** instead.

Demo Time

Long time ago in a Kubernetes cluster far far away...

Made with: https://starwarsintrogenerator.com/

Terminal Time

Star Wars
Gateway API Demo
Episode VI

The Return of the Service Mesh

Use Linkerd's experimental support for Service ParentRef targets in HTTPRoutes for cross-namespace routing between micro-services.

ISOVALENT

Key demo takeaway

Gateway API ***Route*** objects are well scoped for micro-service routing policy.

Allows application teams to build and control explicit policy around how backend services interconnect.

Individual consumer service teams, can build and implement ***Route*** resources without disrupting other consumers.

Producer service teams, can build and implement ***Route*** resources that control all, or some internal clients as needed.

ISOVALENT

Parting thought: *Routes are durable and expressive abstraction for routing policy

And the future is now!!!!!

I think the GAMMA Initative's adaptation of ***Routes** as Service Mesh policy proves the Gateway API decisions are heading in the right direction.

Consolidating around *_**Routes**_ as a standard way to express routing policy between services has huge benefits.

Introduction of ***ReferenceGrants*** as a sort of contract between teams working in different namespaces, makes dependencies between teams more explicit, which I think serves as an interesting model for resource access across all of Kubernetes more broadly.

I think this design really powerful for both application teams and cluster operators and makes it easier and safer for teams to self-service application routing.

ISOVALENT

Thanks for coming to my talk!

Here are some additional learning resources:

Kubernetes Gateway API SIG resources:
https://gateway-api.sigs.k8s.io/

Demo Git Repo:
https://isogo.to/scale21x-gateway-demos

Cilium Gateway API Features (North/South):
https://docs.cilium.io/en/v1.15/network/servicemesh/gateway-api/gateway-api/

Linkerd Gateway API Features (East/West):
https://linkerd.io/2.15/features/httproute/

Song Playlist: https://isogo.to/scale21x-playlist