
KubeCon
CloudNativeCon
North America 2024

All your routes are ready, *more or less*

David Protasowski

Dave Protasowski

Staff Engineer

Serving Lead & TOC

dprotaso

lintinmybelly

dprotasowski

Knative

- **Serving**
  - Autoscale your workloads based on traffic - can scale to zero
- **Eventing**
  - Declaratively bind consumers and producers of events
- **Client (kn)**
  - Create resources interactively from the command line
- **Functions**
  - A programming model to simplify development

Knative

- **Serving**
  - Autoscale your workloads based on traffic - can scale to zero
- **Eventing**
  - Declaratively bind consumers and producers of events
- **Client (kn)**
  - Create resources interactively from the command line
- **Functions**
  - A programming model to simplify development

Knative Serving

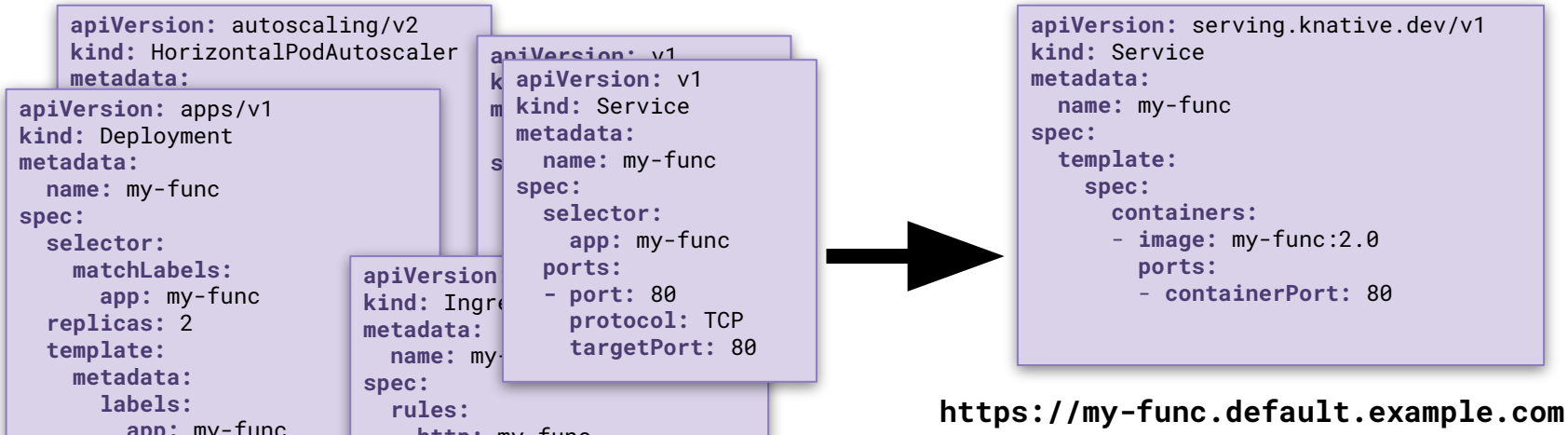
- Higher Level Abstraction than Kubernetes
- Container ⇒ URL
- Autoscales your workloads based on traffic
  - Scales to Zero
- Automatic Certificate Provisioning for HTTPS
- Revision Management & Traffic Splitting
- Automatic Health Checks

Knative Serving


```yaml
apiVersion: serving.knative.dev/v1
kind: Service
metadata:
  name: my-func
spec:
  template:
    spec:
      containers:
      - image: my-func:2.0
        ports:
        - containerPort: 80
  traffic:
  - percent: 0
    revisionName: my-func-0004
    tag: staging
  - percent: 40
    revisionName: my-func-0002
  - percent: 60
    revisionName: my-func-0003
```

https://my-func.default.example.com

Knative Serving



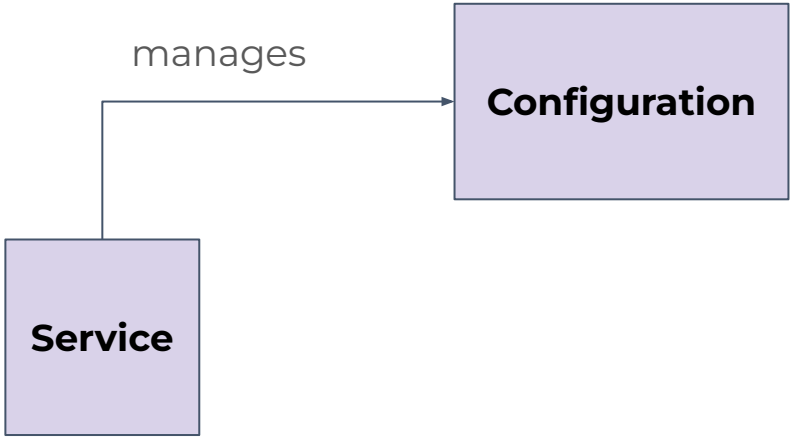
Knative Serving


```yaml
apiVersion: serving.knative.dev/v1
kind: Service
metadata:
  name: my-func
spec:
  template:
    spec:
      containers:
      - image: my-func:2.0
        ports:
        - containerPort: 80
```

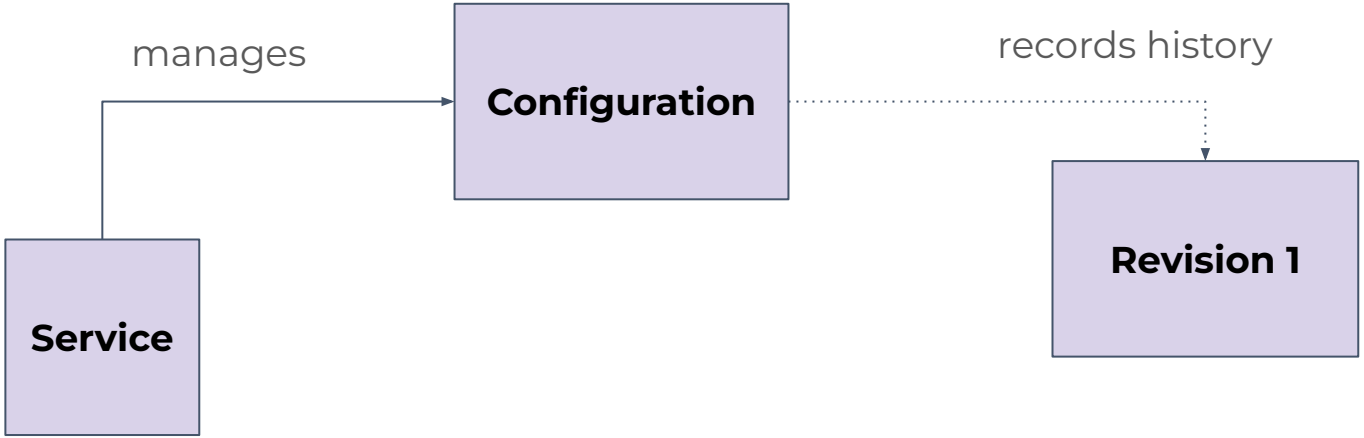
Knative Serving

Knative Serving

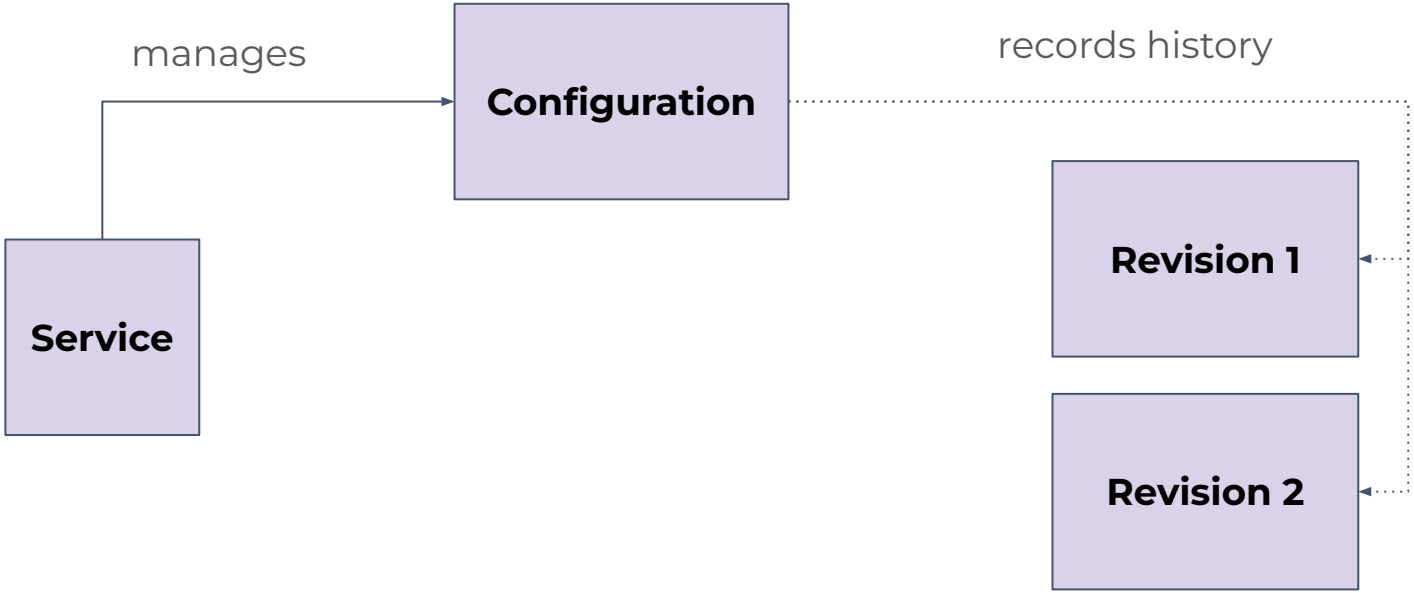
Knative Serving



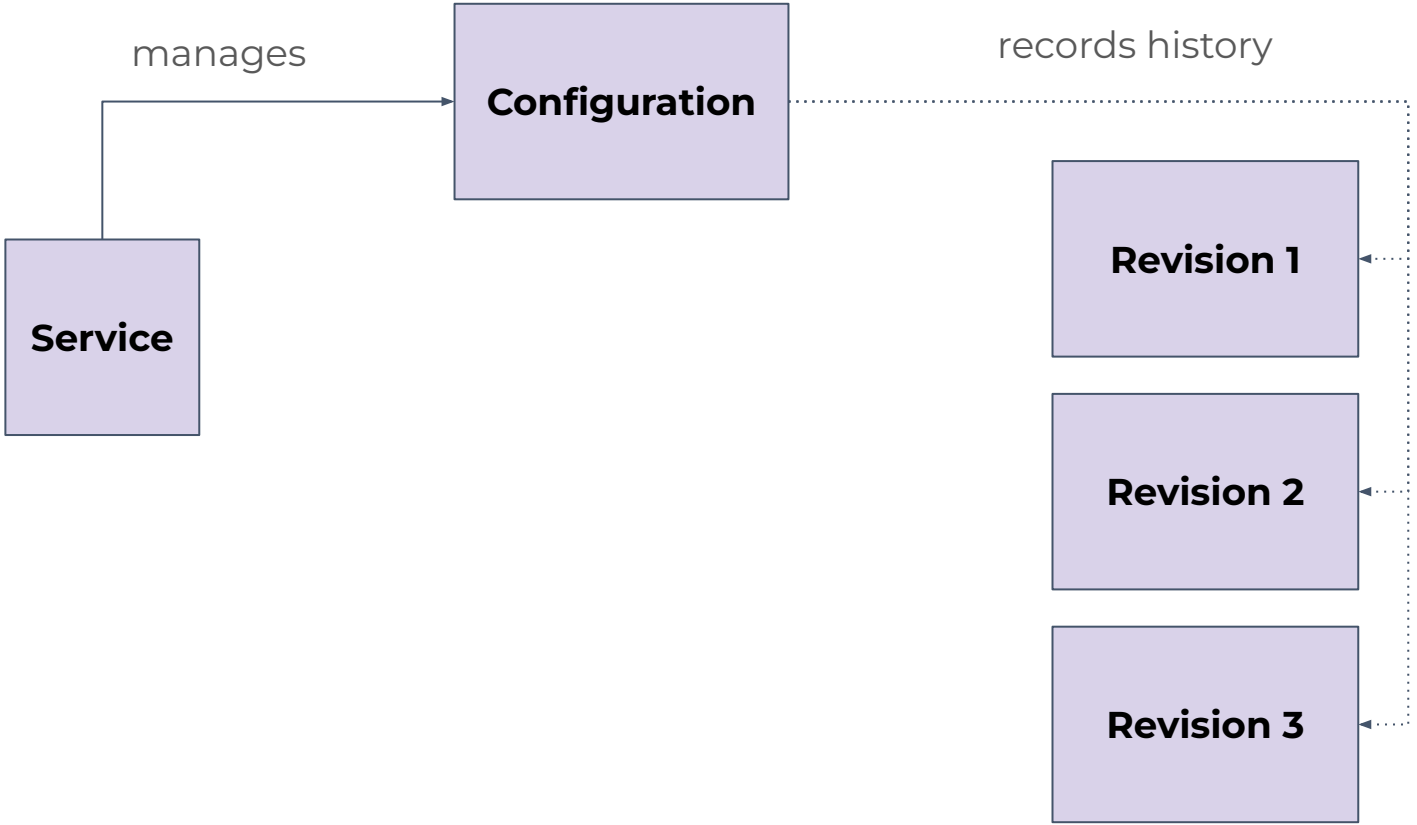
Knative Serving



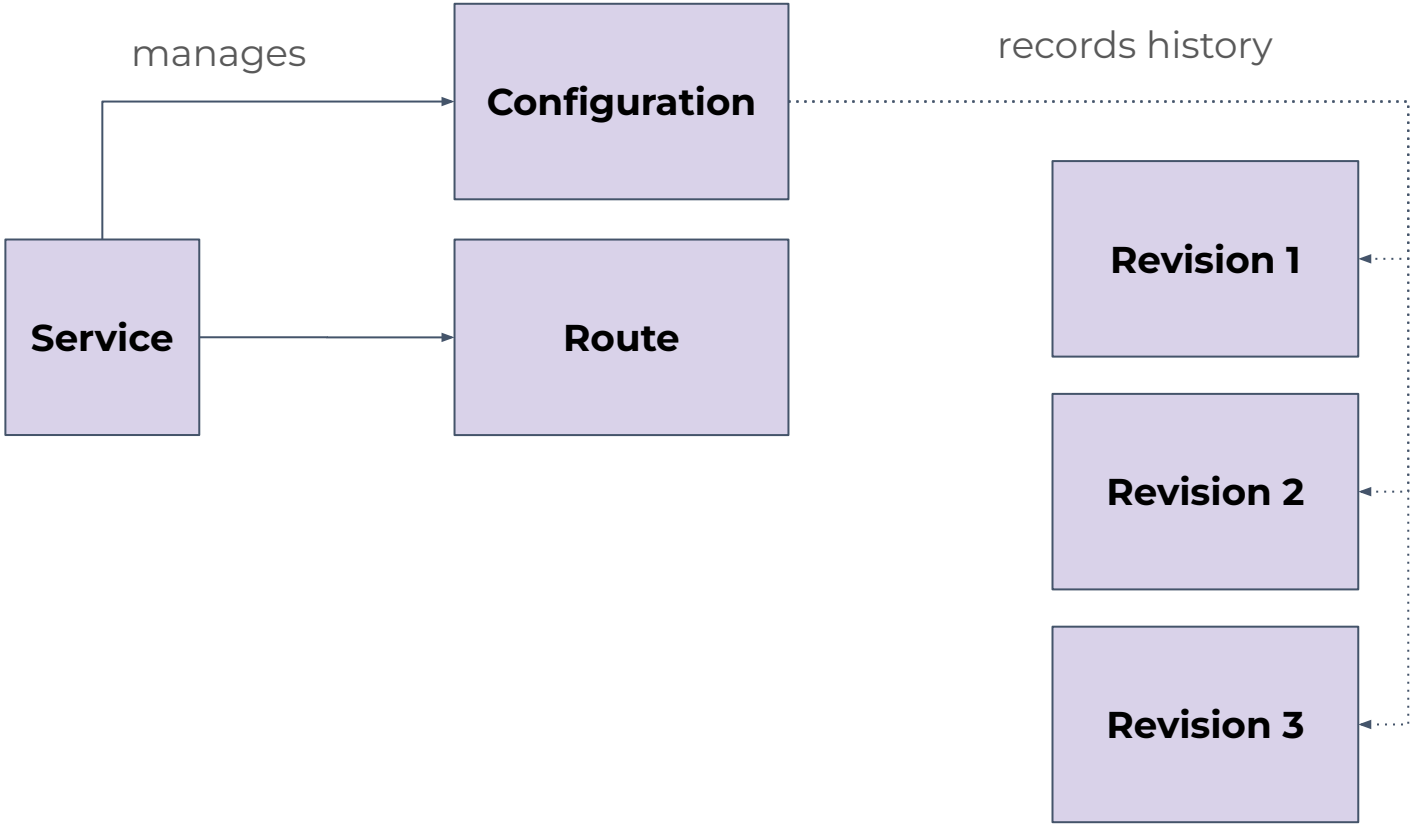
Knative Serving



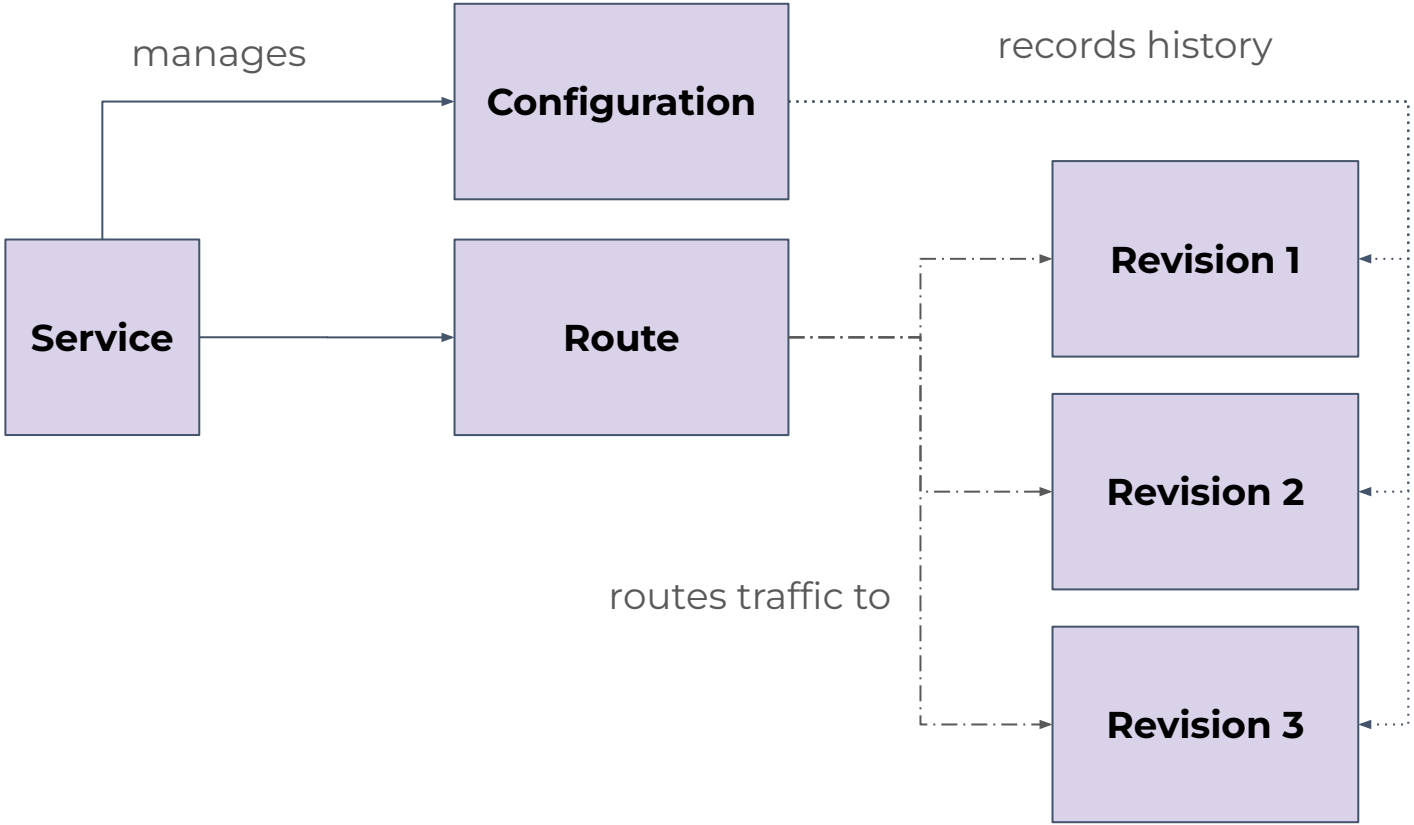
Knative Serving



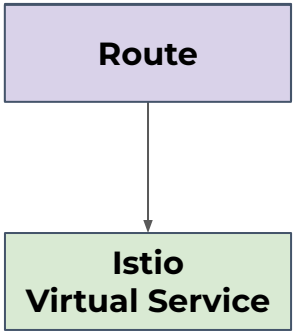
Knative Serving



Knative Serving

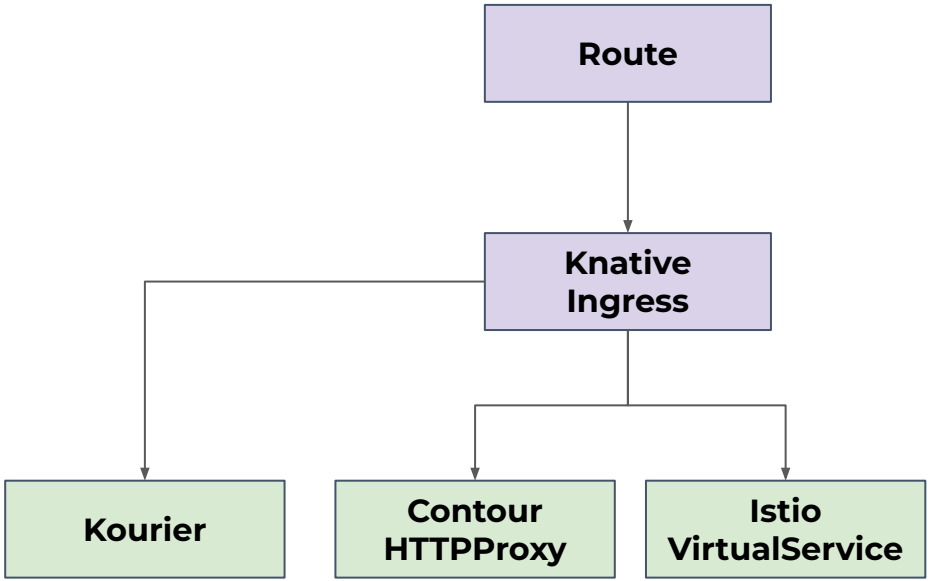


Knative Serving - Routes



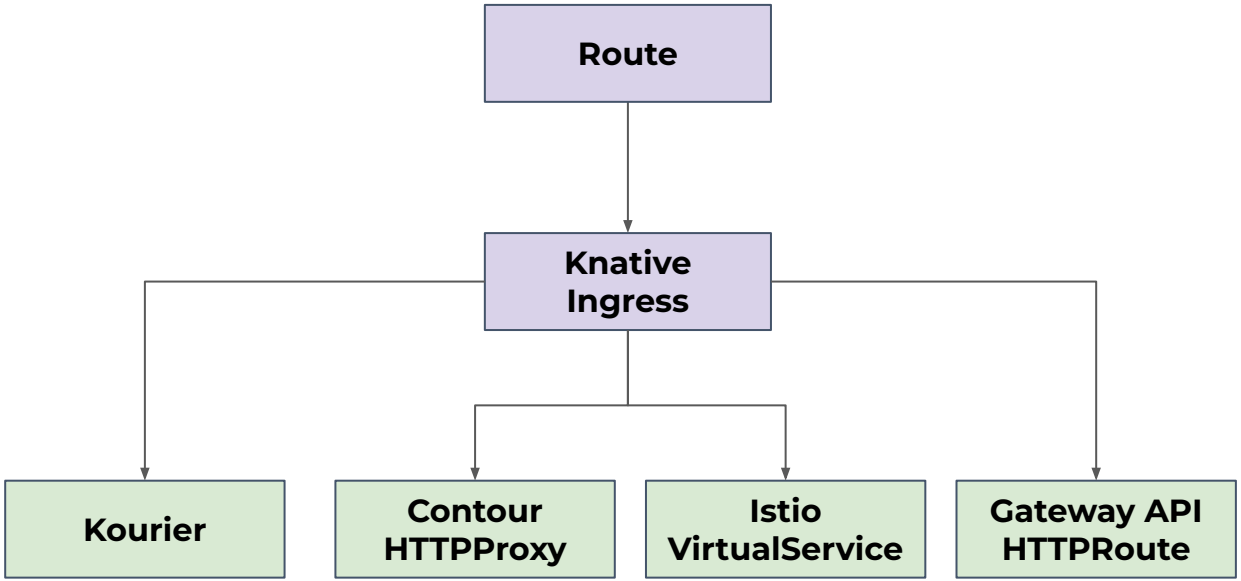
Q1 2018

Knative Serving - Routes



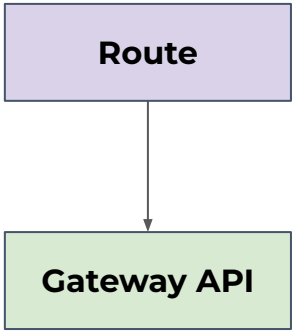
Q4 2018

Knative Serving - Routes



Q4 2021

Knative Serving - Routes



Future

What have we learned?

All your routes are ready, *more or less*

All your routes are **ready**,
more or less

Knative Serving - Ready Condition


```yaml
apiVersion: serving.knative.dev/v1
kind: Service
metadata:
  name: my-func
spec:
  template:
    spec:
      containers:
      - image: my-func:2.0
        ports:
        - containerPort: 80
```

Knative Serving - Ready Condition


```yaml
apiVersion: serving.knative.dev/v1
kind: Service
metadata:
  name: my-func
spec:
  template:
    spec:
      containers:
      - image: my-func:2.0
        ports:
        - containerPort: 80
status:
  ...
  conditions: []
```

Knative Serving - Ready Condition


```yaml
apiVersion: serving.knative.dev/v1
kind: Service
metadata:
  name: my-func
spec:
  template:
    spec:
      containers:
      - image: my-func:2.0
        ports:
        - containerPort: 80
status:
  ...
  conditions:
  - lastTransitionTime: "2024-11-03T18:20:01Z"
    status: "True"
    type: ConfigurationsReady

  - lastTransitionTime: "2024-11-03T18:20:01Z"
    status: "True"
    type: RoutesReady
```

Knative Serving - Ready Condition


```yaml
apiVersion: serving.knative.dev/v1
kind: Service
metadata:
  name: my-func
spec:
  template:
    spec:
      containers:
      - image: my-func:2.0
        ports:
        - containerPort: 80
status:
  ...
  conditions:
  - lastTransitionTime: "2024-11-03T18:20:01Z"
    status: "True"
    type: ConfigurationsReady

  - lastTransitionTime: "2024-11-03T18:20:01Z"
    status: "True"
    type: RoutesReady
```

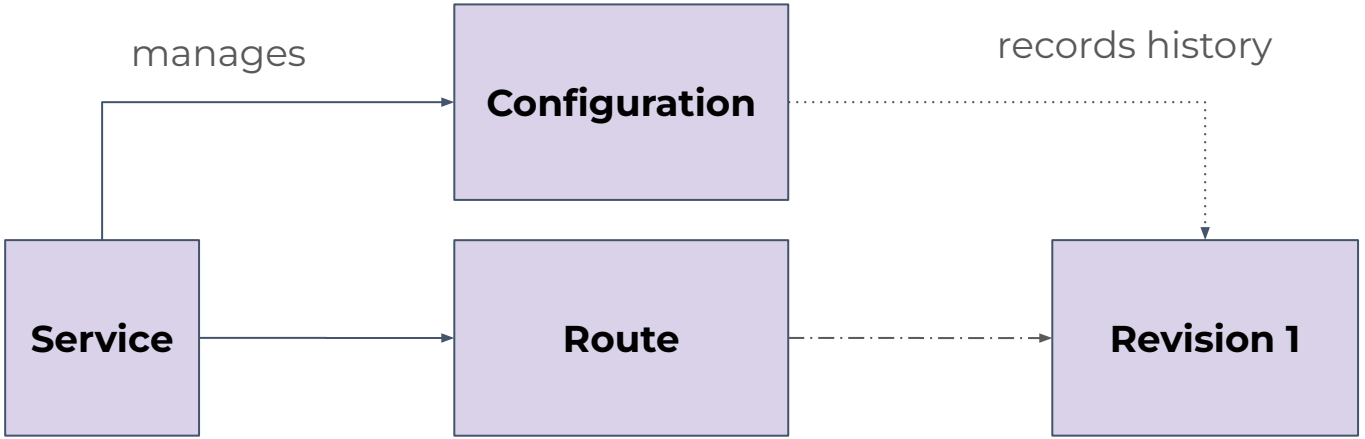
Knative Serving - Ready Condition


```yaml
apiVersion: serving.knative.dev/v1
kind: Service
metadata:
  name: my-func
spec:
  template:
    spec:
      containers:
      - image: my-func:2.0
        ports:
        - containerPort: 80
status:
  ...
  conditions:
  - lastTransitionTime: "2024-11-03T18:20:01Z"
    status: "True"
    type: ConfigurationsReady

  - lastTransitionTime: "2024-11-03T18:20:01Z"
    status: "True"
    type: RoutesReady

  - lastTransitionTime: "2024-11-03T18:20:01Z"
    status: "True"
    type: Ready
```

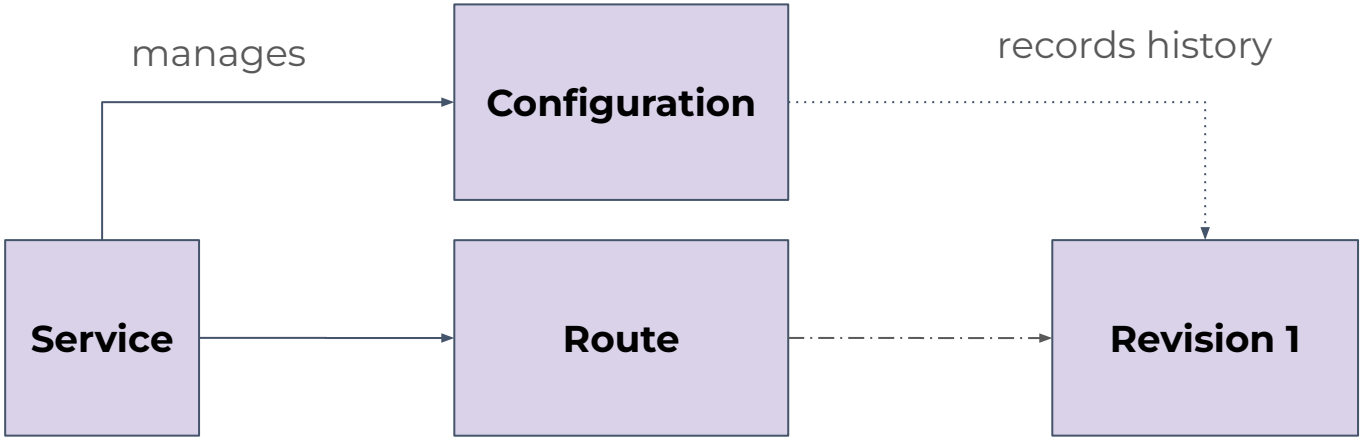
Knative Serving - Ready Condition


```yaml
apiVersion: serving.knative.dev/v1
kind: Service
metadata:
  name: my-func
spec:
  template:
    spec:
      containers:
      - image: my-func:2.0
        ports:
        - containerPort: 80
status:
  ...
  conditions:
  - lastTransitionTime: "2024-11-03T18:20:01Z"
    status: "True"
    type: ConfigurationsReady

  - lastTransitionTime: "2024-11-03T18:20:01Z"
    status: "True"
    type: RoutesReady

  - lastTransitionTime: "2024-11-03T18:20:01Z"
    status: "True"
    type: Ready
```

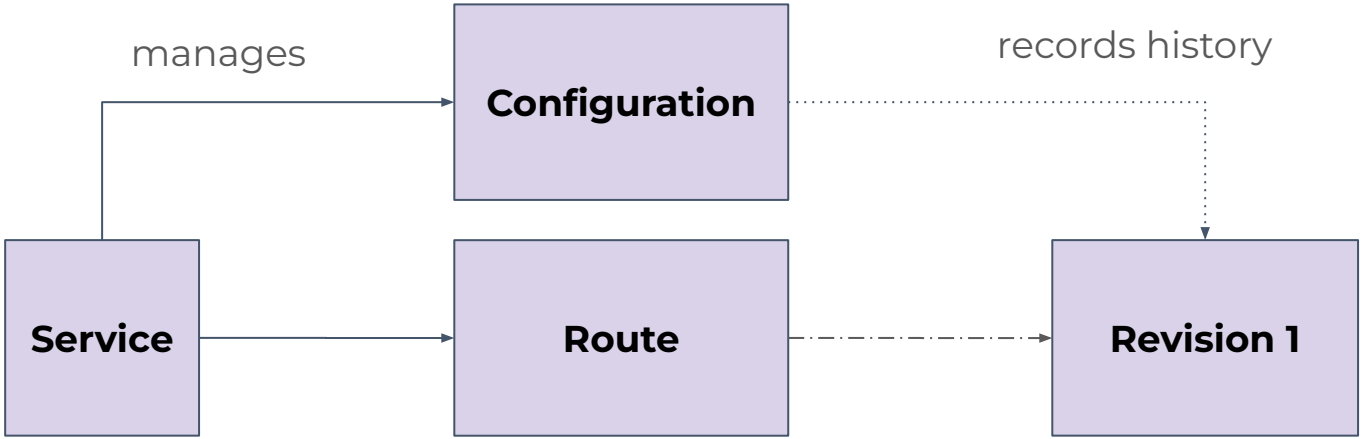
Knative Serving - Ready Condition



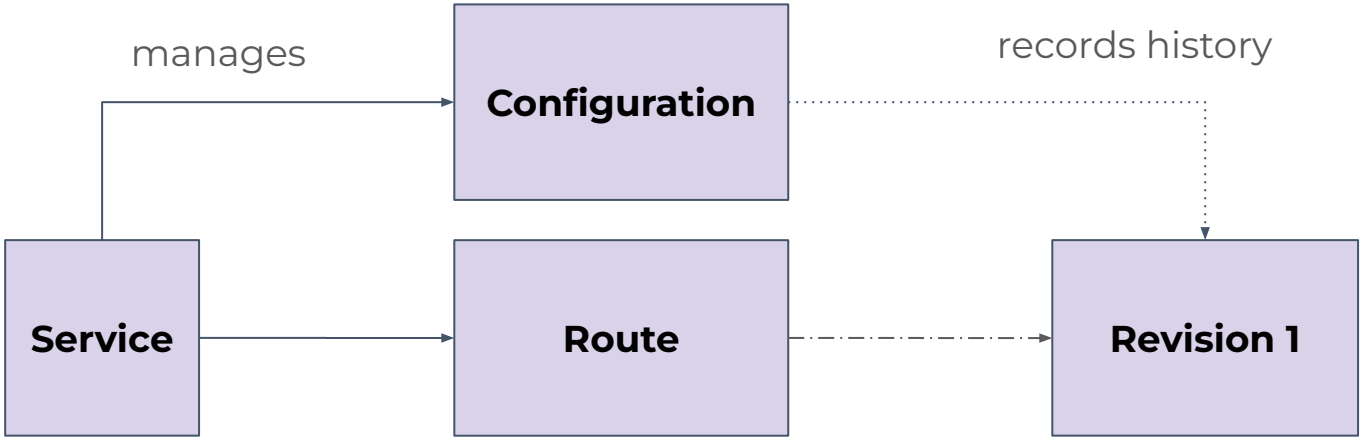
Knative Serving - Ready Condition



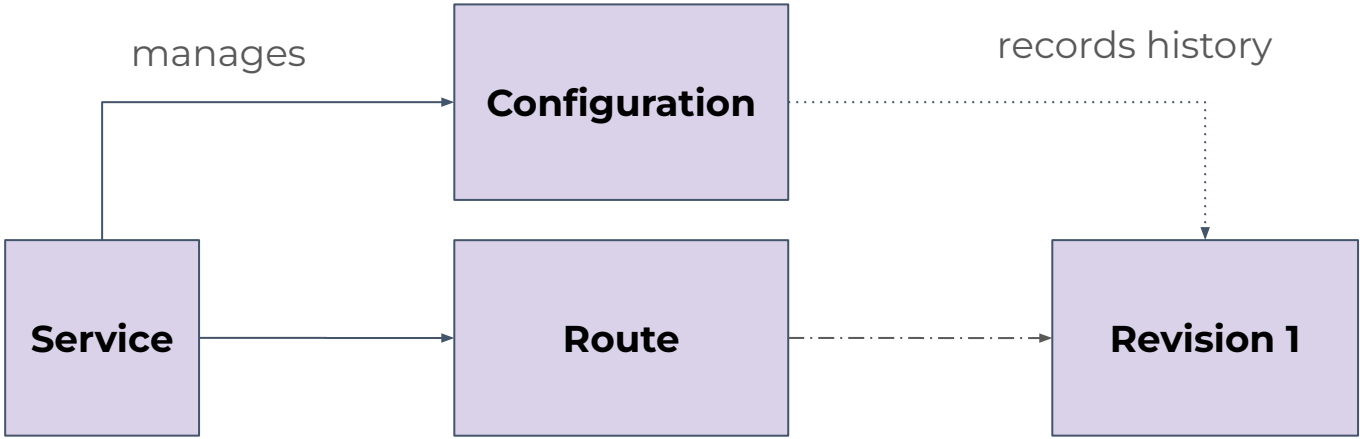
Knative Serving - Ready Condition



Knative Serving - Ready Condition



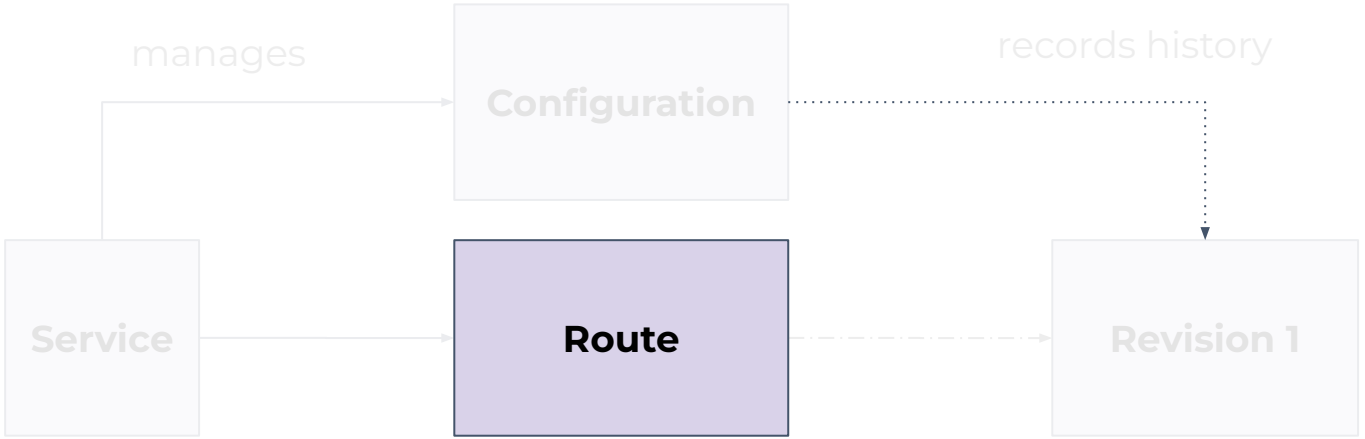
Knative Serving - Ready Condition



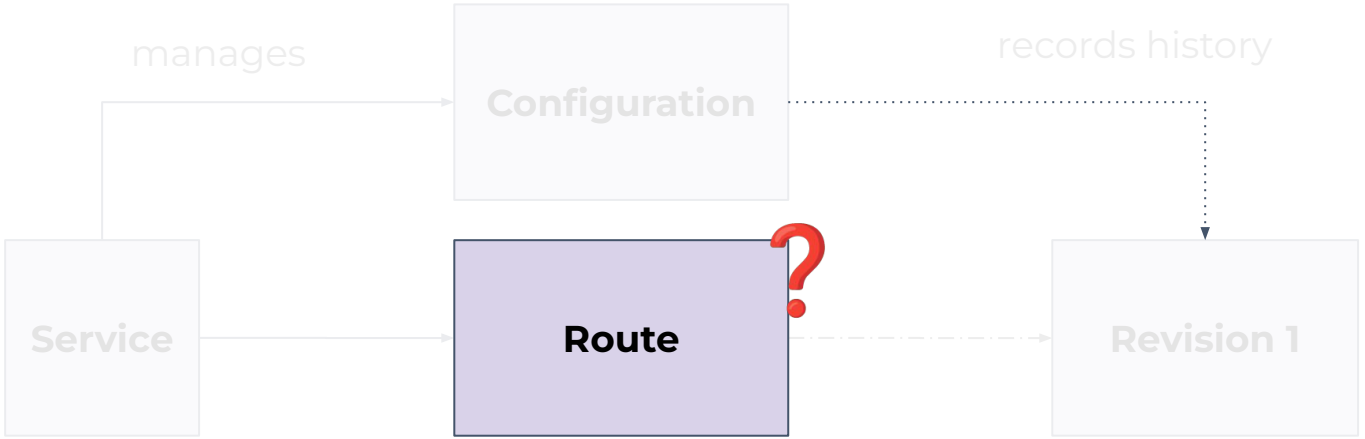
All your routes are **ready**,
more or less

All your **routes** are ready,
more or less

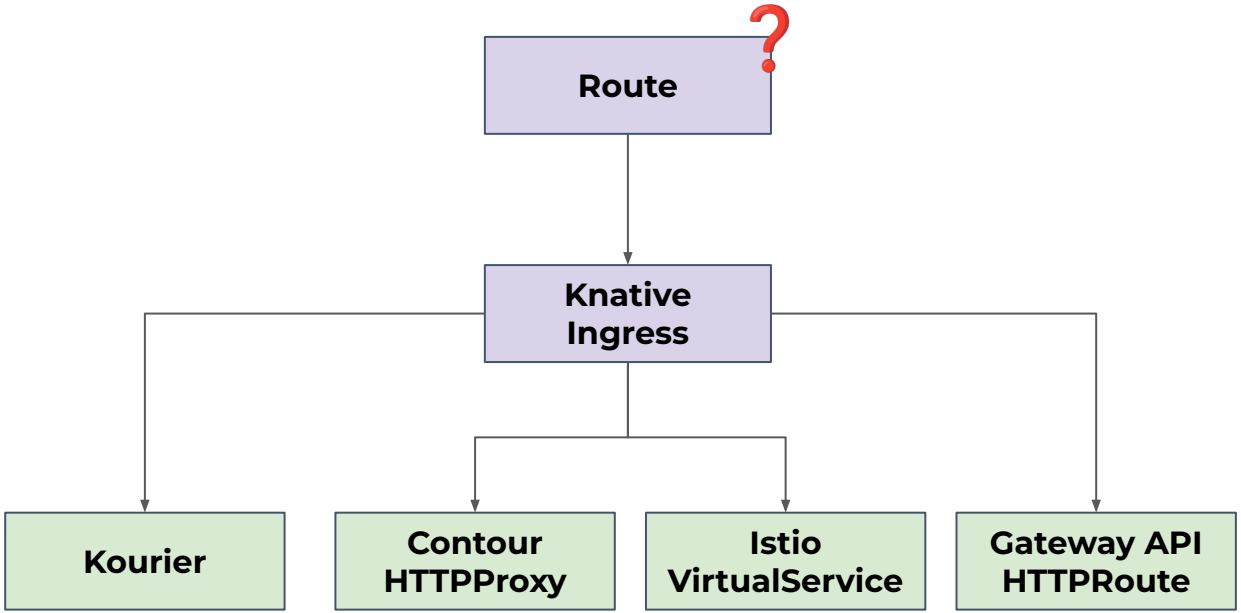
Knative Serving - Routes



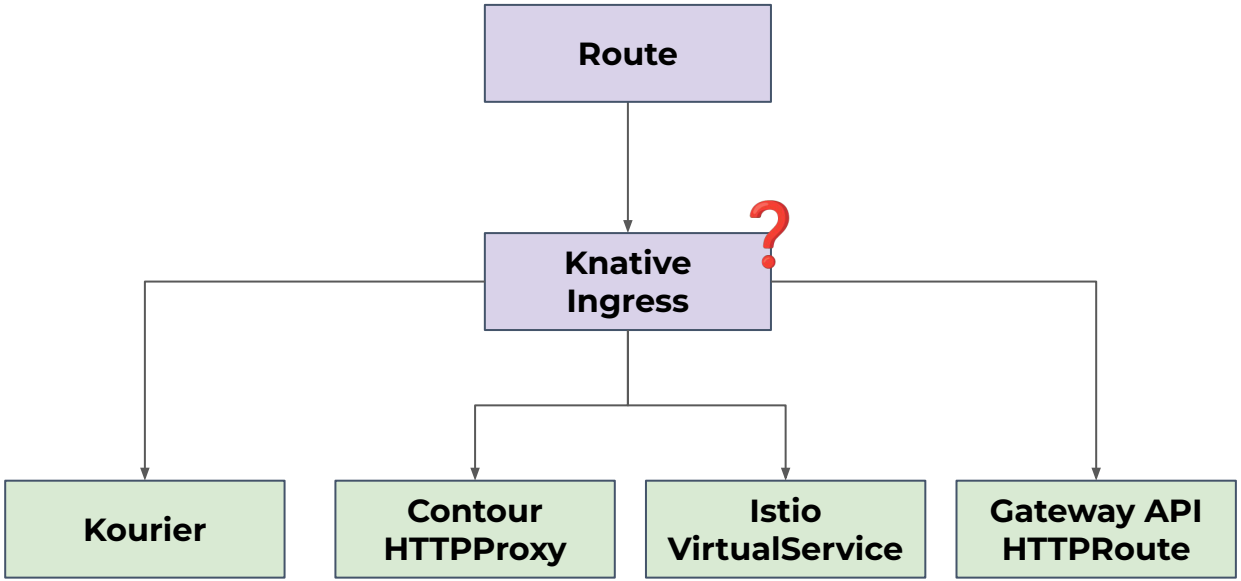
Knative Serving - Routes



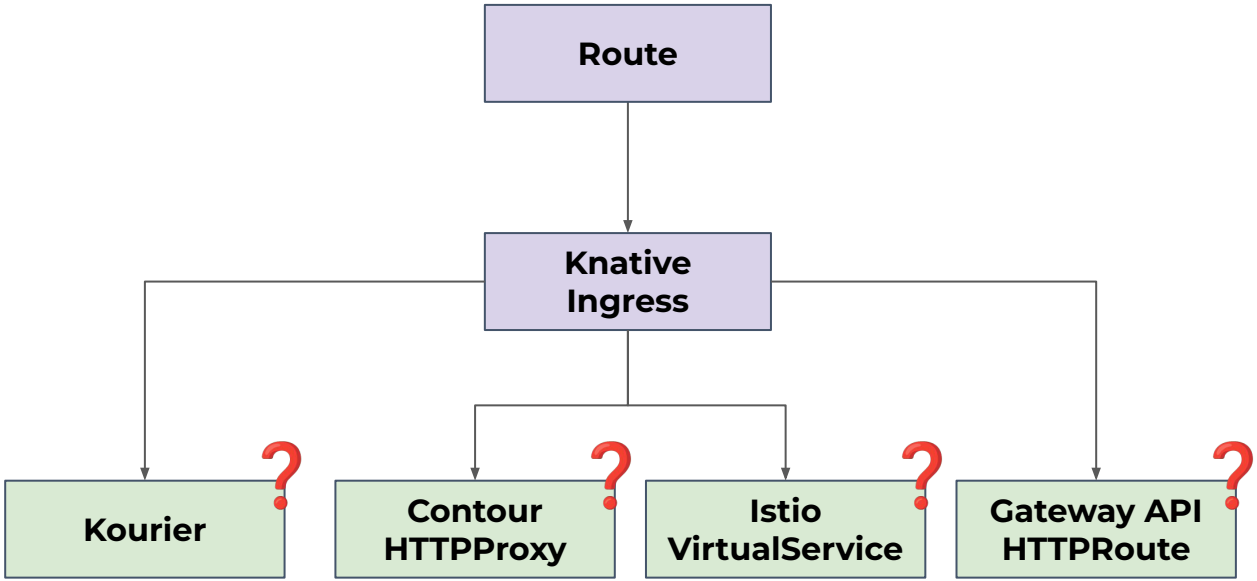
Knative Serving - Routes



Knative Serving - Routes



Knative Serving - Routes



Types of Guarantees

- Controller validates configuration
- Controllers sends configuration to proxy

Is it safe?

Types of Guarantees

❌ Controller validates configuration

❌ Controllers sends configuration to proxy

Types of Guarantees

❌ Controller validates configuration

❌ Controllers sends configuration to proxy

- Proxy responds to config message with an ACK

Types of Guarantees

- ❌ Controller validates configuration
- ❌ Controllers sends configuration to proxy
- ❌ Proxy responds to config message with an ACK

Types of Guarantees

❌ Controller validates configuration

❌ Controllers sends configuration to proxy

❌ Proxy responds to config message with an ACK

- Proxy waits for programming to finish then replies (maybe async)

Types of Guarantees

- ❌ Controller validates configuration
- ❌ Controllers sends configuration to proxy
- ❌ Proxy responds to config message with an ACK
- ❌ Proxy waits for programming to finish then replies (maybe async)

Types of Guarantees

- ❌ Controller validates configuration
- ❌ Controller sends configuration to proxy
- ❌ Proxy responds to configuration message with ACK
- ❌ Proxy waits for programming to finish then replies (maybe async)

DNS

Types of Guarantees

- ✗ Controller validates configuration
- ✗ Controllers sends configuration to proxy
- ✗ Proxy responds to config message with an ACK
- ✗ Proxy waits for programming to finish then replies (maybe asynchronously)

>1 PROXY

Gateway API xRoute Conditions

- **Accepted**
  - If the Route's ParentRef specifies an existing Gateway that supports Routes of this kind AND that Gateway's controller has sufficient access, then that Gateway's controller MUST set the "Accepted" condition on the Route

- **Programmed**
  - This condition indicates whether a Listener has generated some configuration that will soon be ready in the underlying data plane.

NOT SAFE

Other Implementation Quirks

- Some non-Gateway API implementations surface readiness via `status`
- Sometimes it's broken
  - ie. `observedGeneration` is missing or not bumped properly
- tl;dr no reliable way for clients to know when to make a request

Come help!

- **Ready**
  - If used in the future, “Ready” will represent the final state where all configuration is confirmed good *and has completely propagated to the data plane*.

Related Issues

https://github.com/kubernetes-sigs/gateway-api/issues/1156
https://github.com/kubernetes-sigs/gateway-api/issues/1364
https://github.com/kubernetes-sigs/gateway-api/issues/1870

Come help!

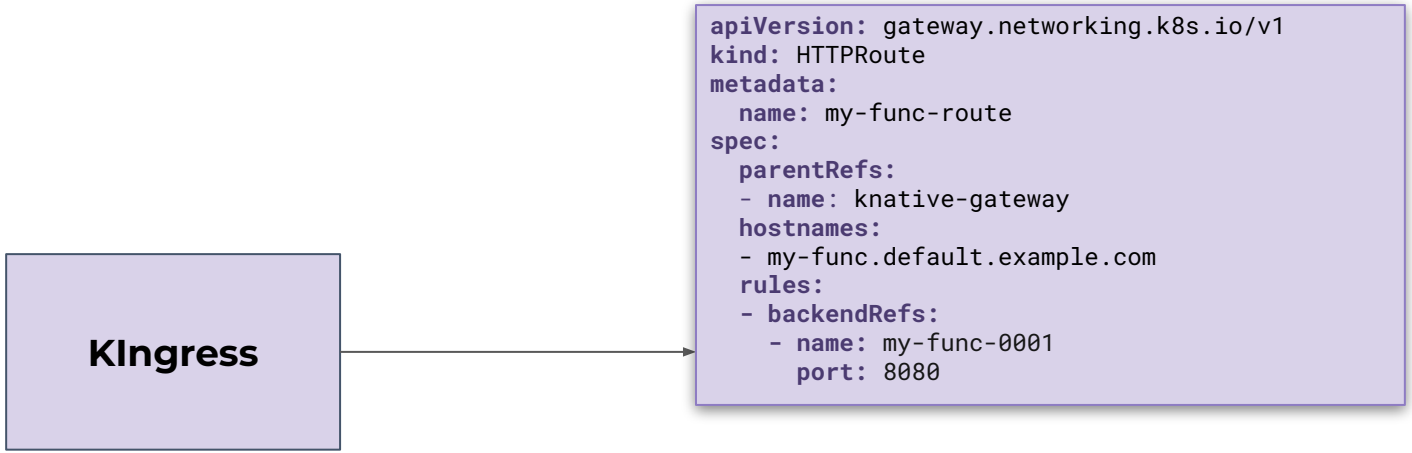
Vote for ‘Ready’ in Gateway v1.3!

https://bit.ly/gateway-ready

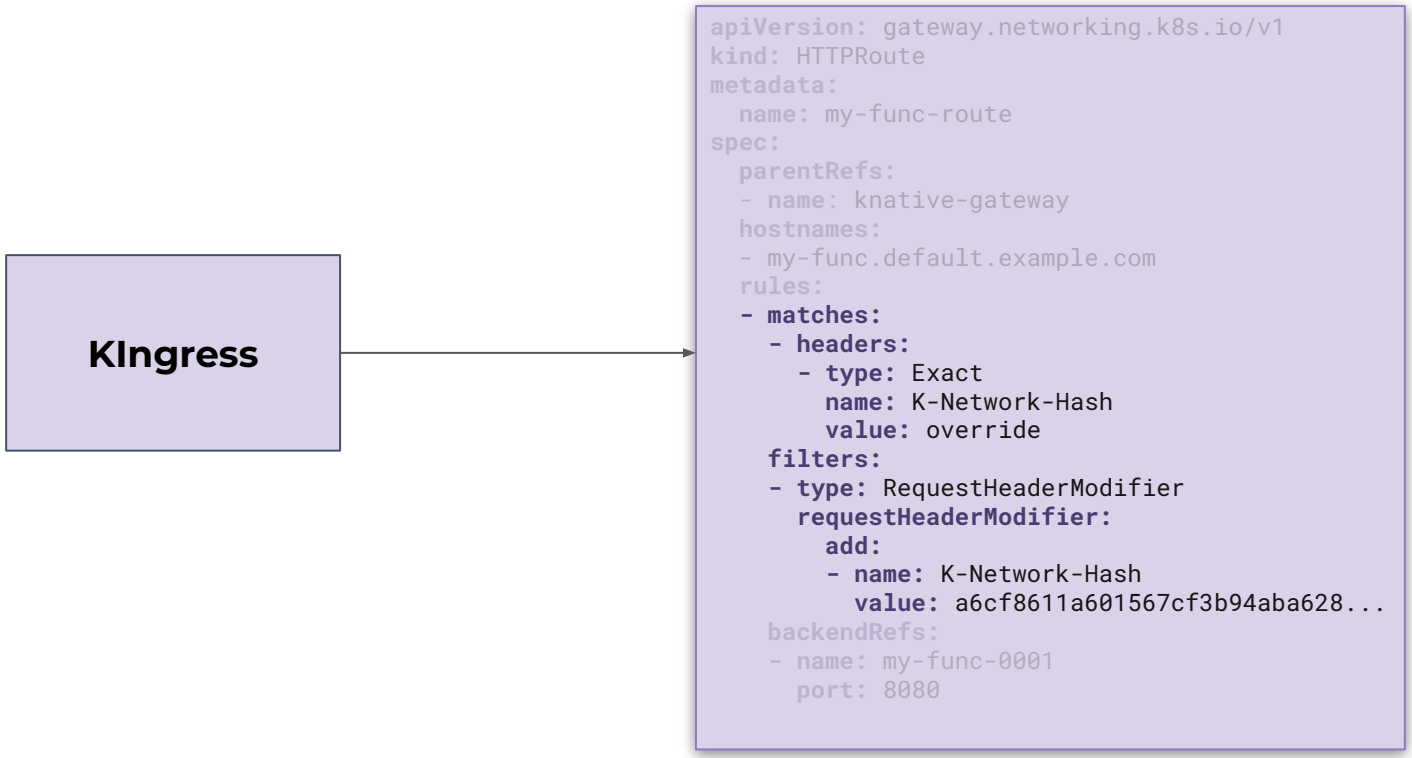

WHAT DO WE DO?

Probing 😭

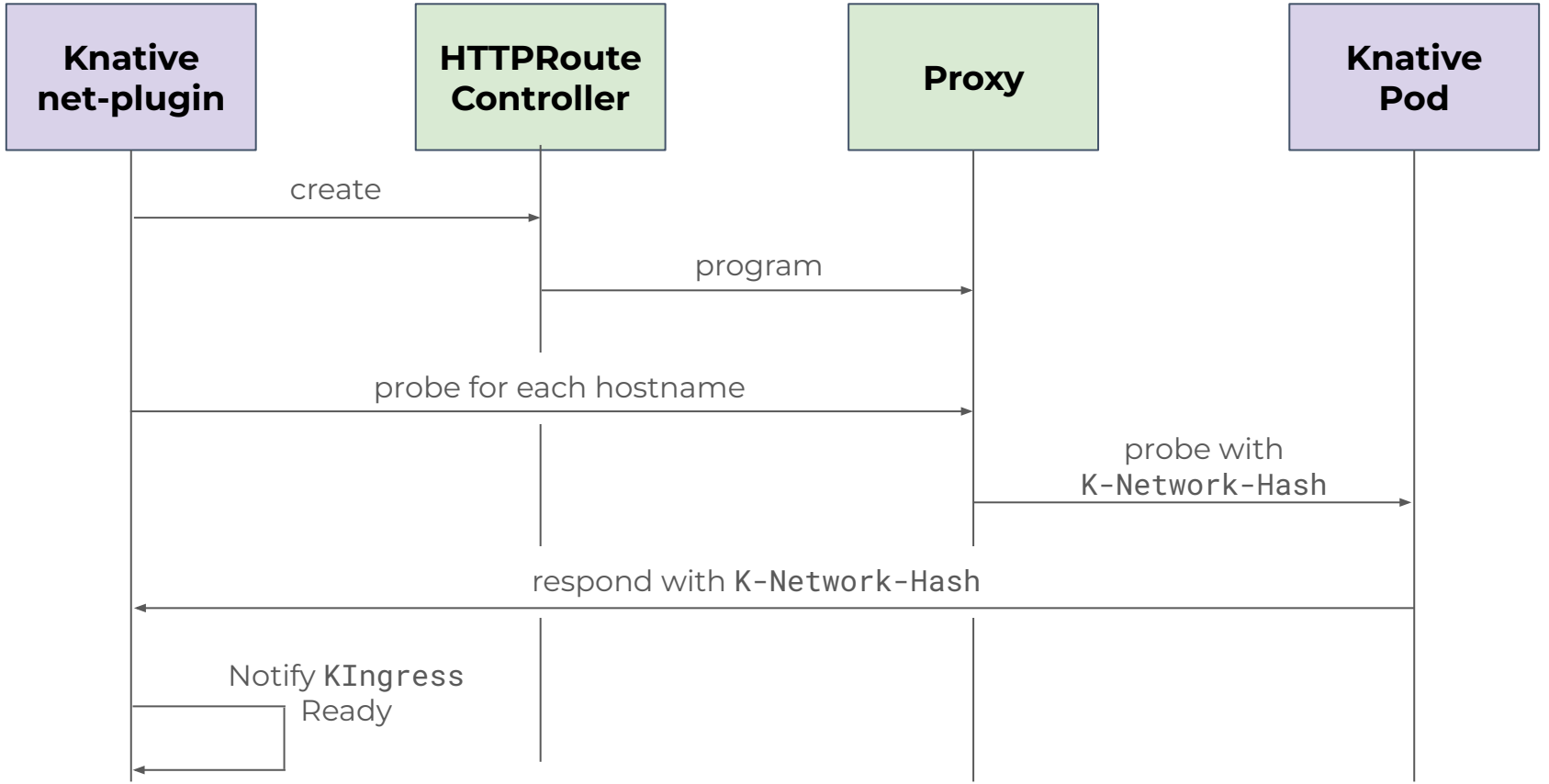
Probing 😭


KIngress
apiVersion: gateway.networking.k8s.io/v1
kind: HTTPRoute
metadata:
name: my-func-route
spec:
parentRefs:
- name: knative-gateway
hostnames:
- my-func.default.example.com
rules:
- backendRefs:
- name: my-func-0001
port: 8080

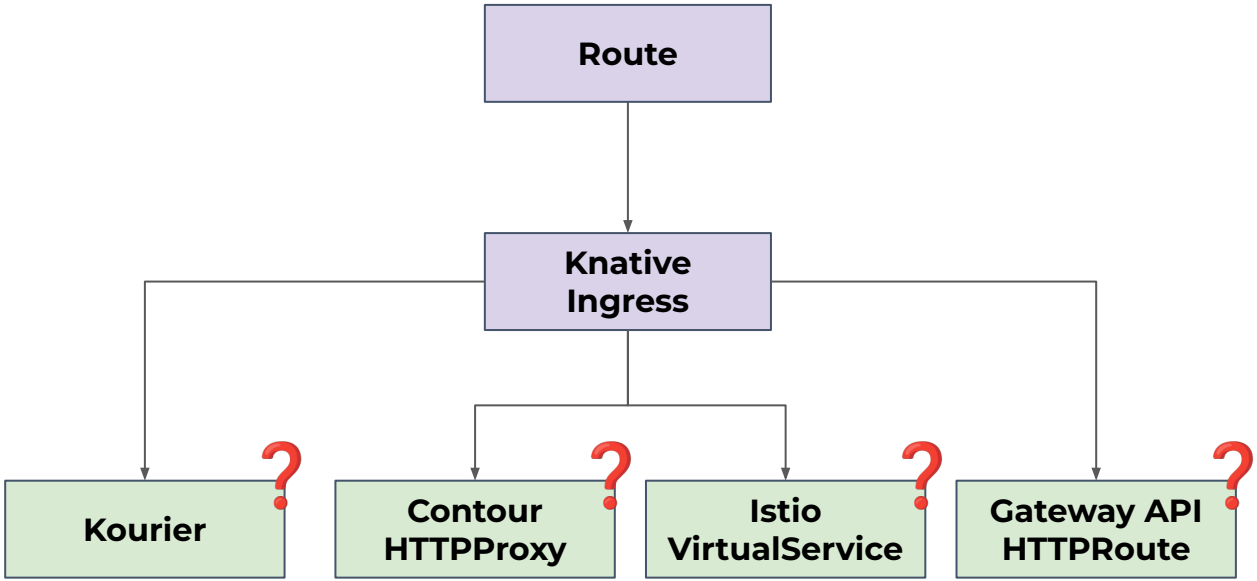
Probing 😭


KIngress
apiVersion: gateway.networking.k8s.io/v1
kind: HTTPRoute
metadata:
name: my-func-route
spec:
parentRefs:
- name: knative-gateway
hostnames:
- my-func.default.example.com
rules:
- matches:
- headers:
- type: Exact
name: K-Network-Hash
value: override
filters:
- type: RequestHeaderModifier
requestHeaderModifier:
add:
- name: K-Network-Hash
value: a6cf8611a601567cf3b94aba628...
backendRefs:
- name: my-func-0001
port: 8080

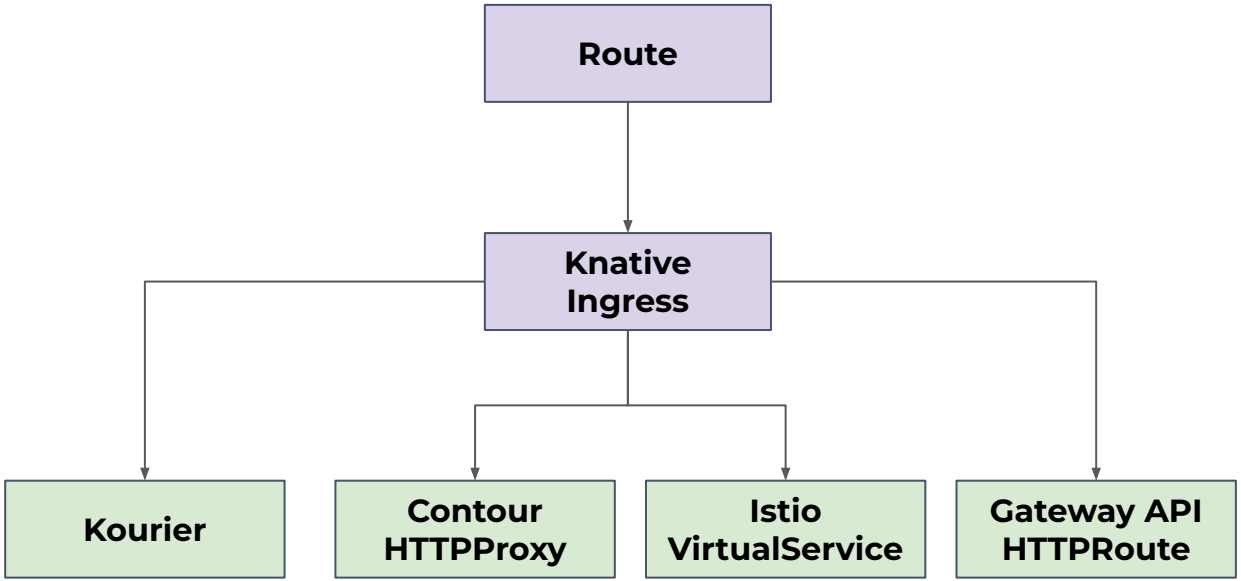
Probing 😭


Knative net-plugin
HTTPRoute Controller
Proxy
Knative Pod
create
program
probe for each hostname
probe with K-Network-Hash
respond with K-Network-Hash
Notify KIngress Ready

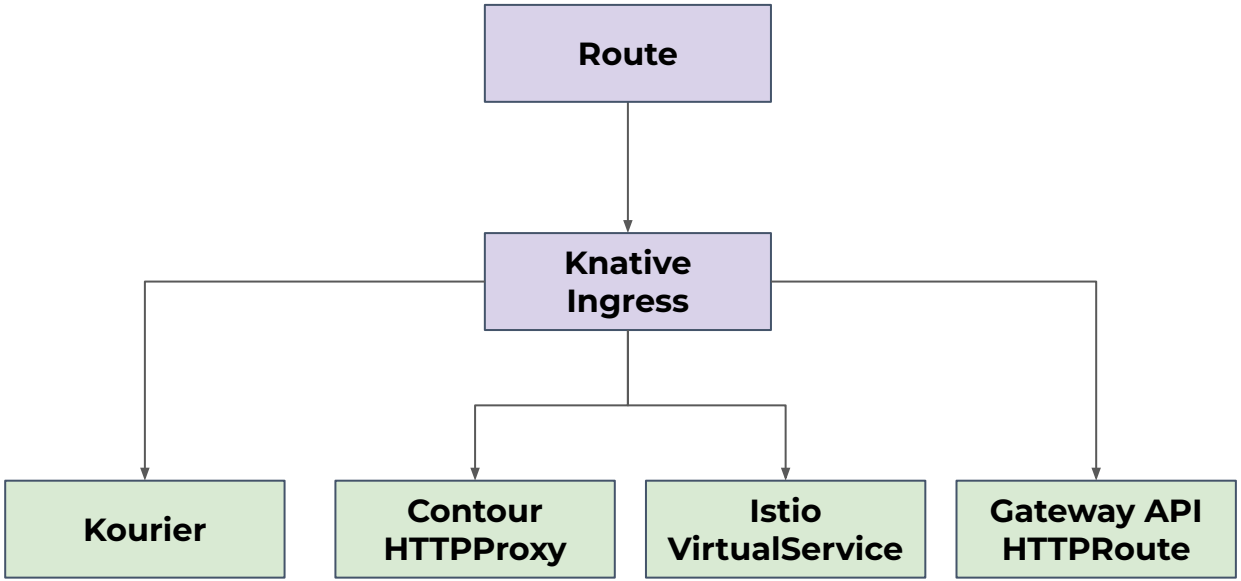
Probing 😭


Route
Knative Ingress
Kourier
Contour HTTPProxy
Istio VirtualService
Gateway API HTTPRoute

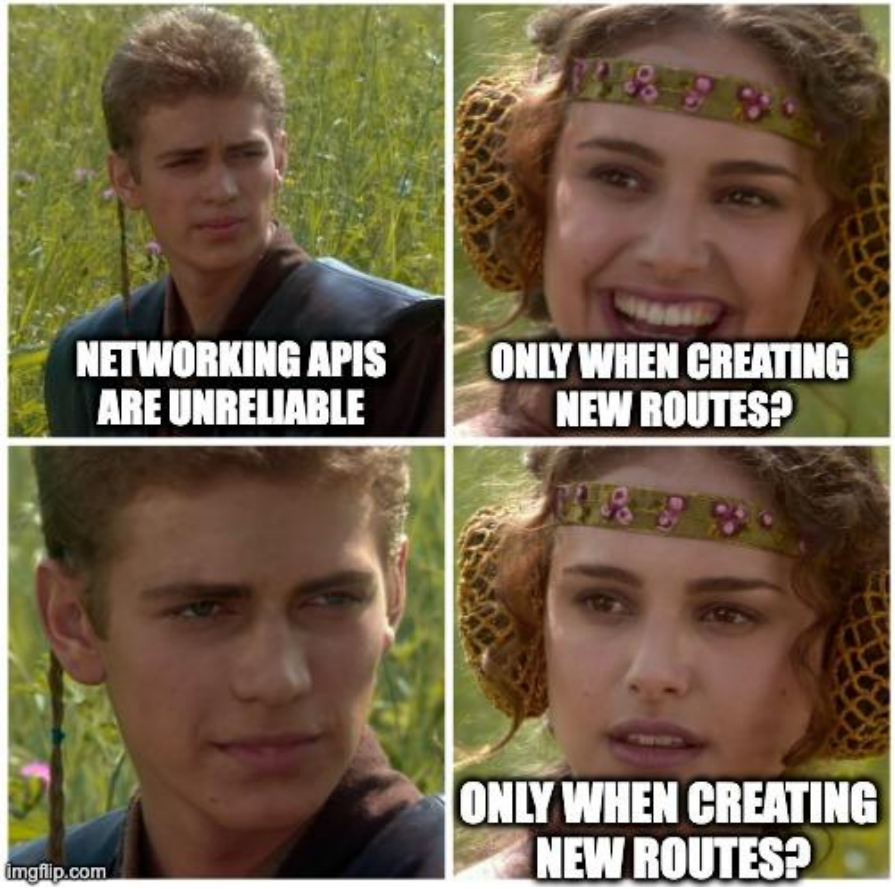
Probing 😭


Route
Knative Ingress
Kourier
Contour HTTPProxy
Istio VirtualService
Gateway API HTTPRoute

Probing 😭


Route
Knative Ingress
Kourier
Contour HTTPProxy
Istio VirtualService
Gateway API HTTPRoute

We ain’t done


NETWORKING APIS ARE UNRELIABLE
ONLY WHEN CREATING NEW ROUTES?
ONLY WHEN CREATING NEW ROUTES?
imgflip.com

Update Operations


```yaml
apiVersion: gateway.networking.k8s.io/v1
kind: HTTPRoute
metadata:
  name: my-func-route
spec:
  parentRefs:
  - name: knative-gateway
  hostnames:
  - my-func.default.example.com
  rules:
  - backendRefs:
    - name: my-func-0001
      port: 8080
```



```yaml
apiVersion: gateway.networking.k8s.io/v1
kind: HTTPRoute
metadata:
  name: my-func-route
spec:
  parentRefs:
  - name: knative-gateway
  hostnames:
  - my-func.default.example.com
  rules:
  - backendRefs:
    - name: my-func-0002
      port: 8080
```

```
go run github.com/rakyll/hey@latest \
  -disable-keepalive \
  -c 10 `# number of workers` \
  -q 5  `# qps per worker` \
  -z 1m `# duration` \
  https://my-func.default.example.com
```

Update Operations are Unsafe

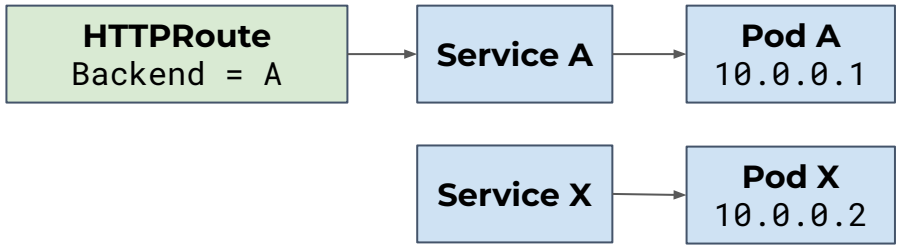
Update Operations are Unsafe

- Ingress doesn't track backend endpoints that aren't reachable via routes

https://github.com/kubernetes-sigs/gateway-api/issues/1870

Update Operations are Unsafe

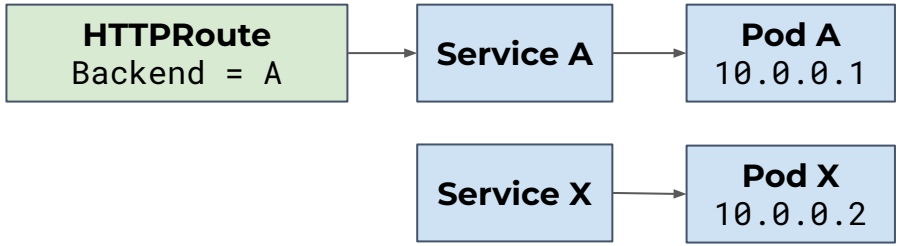
- Ingress doesn't track backend endpoints that aren't reachable via routes

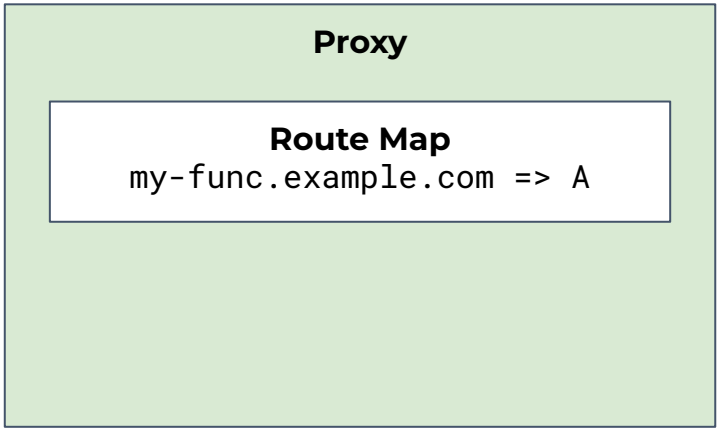


https://github.com/kubernetes-sigs/gateway-api/issues/1870

Update Operations are Unsafe

- Ingress doesn't track backend endpoints that aren't reachable via routes

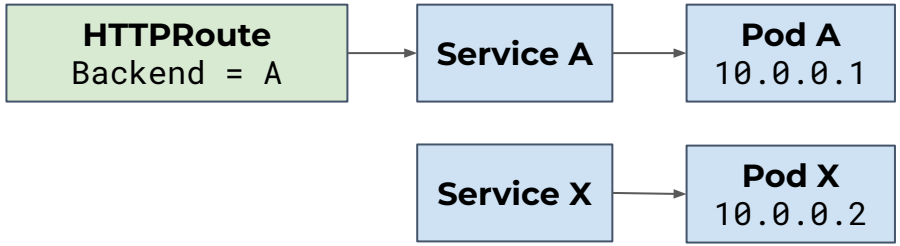


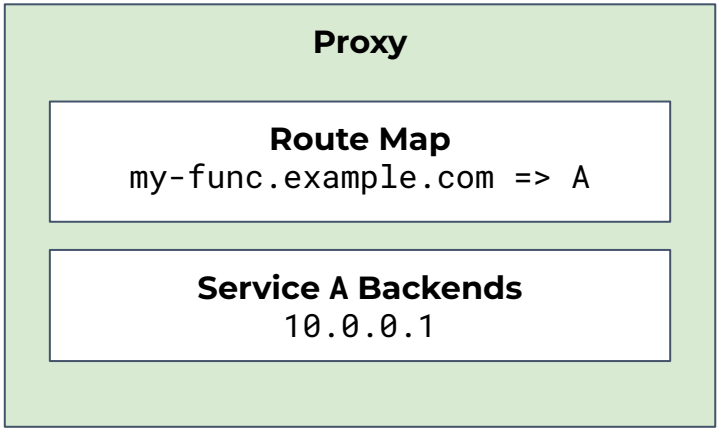


https://github.com/kubernetes-sigs/gateway-api/issues/1870

Update Operations are Unsafe

- Ingress doesn't track backend endpoints that aren't reachable via routes

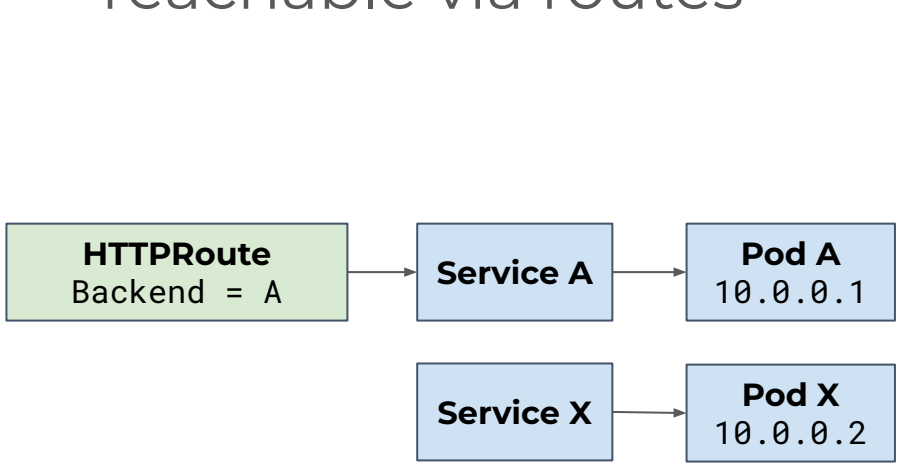


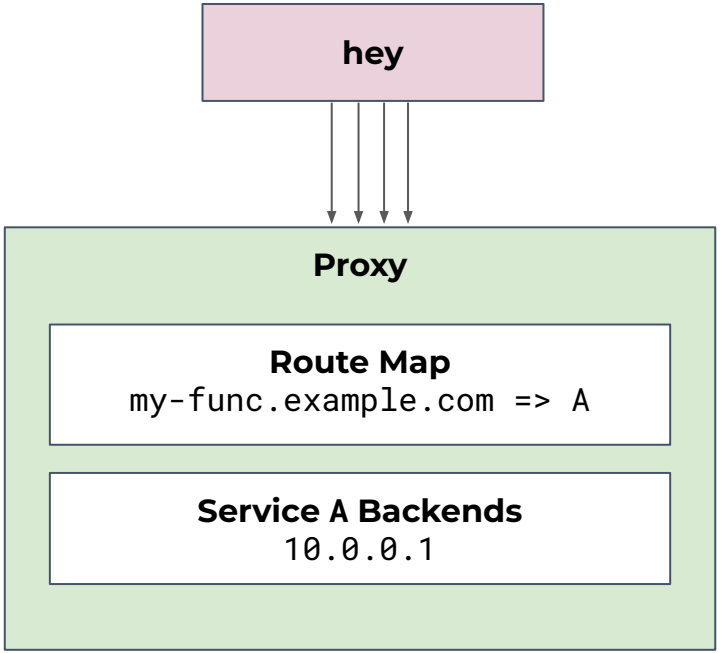


https://github.com/kubernetes-sigs/gateway-api/issues/1870

Update Operations are Unsafe

- Ingress doesn't track backend endpoints that aren't reachable via routes

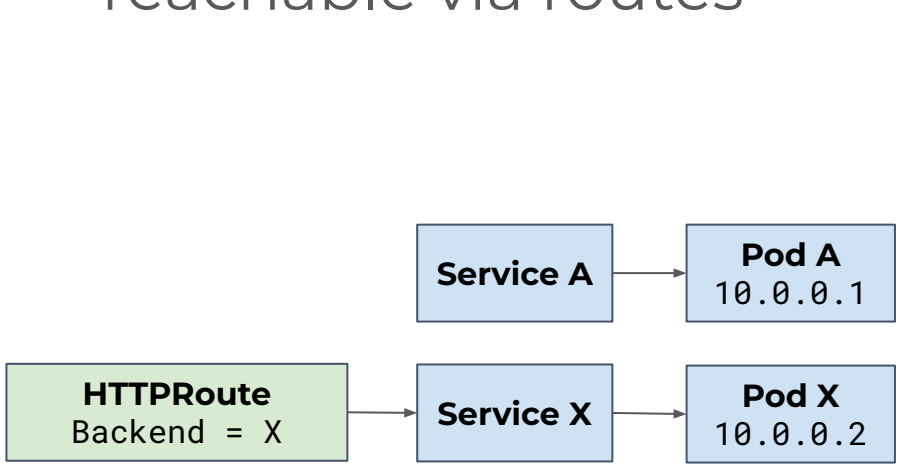


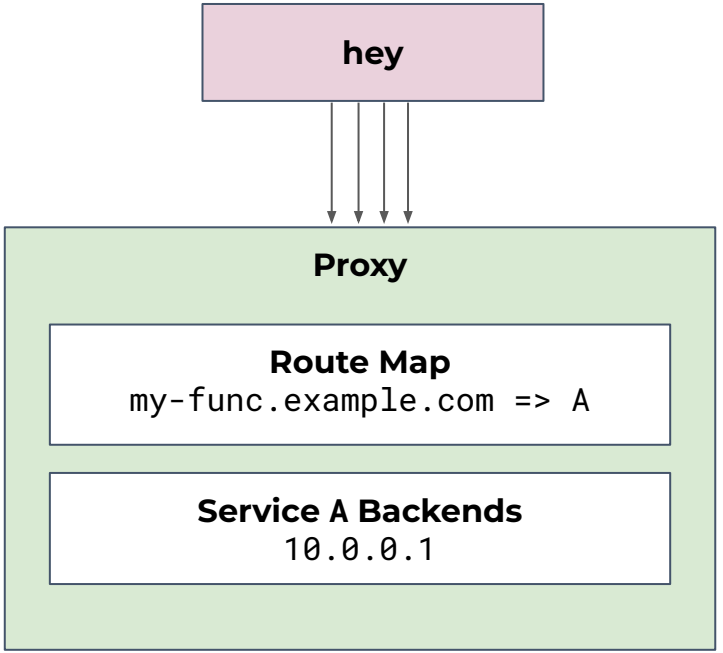


https://github.com/kubernetes-sigs/gateway-api/issues/1870

Update Operations are Unsafe

- Ingress doesn't track backend endpoints that aren't reachable via routes

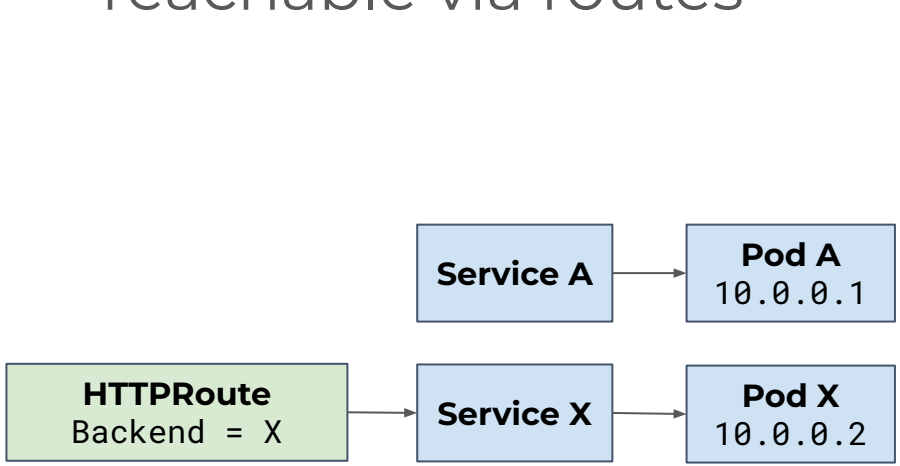


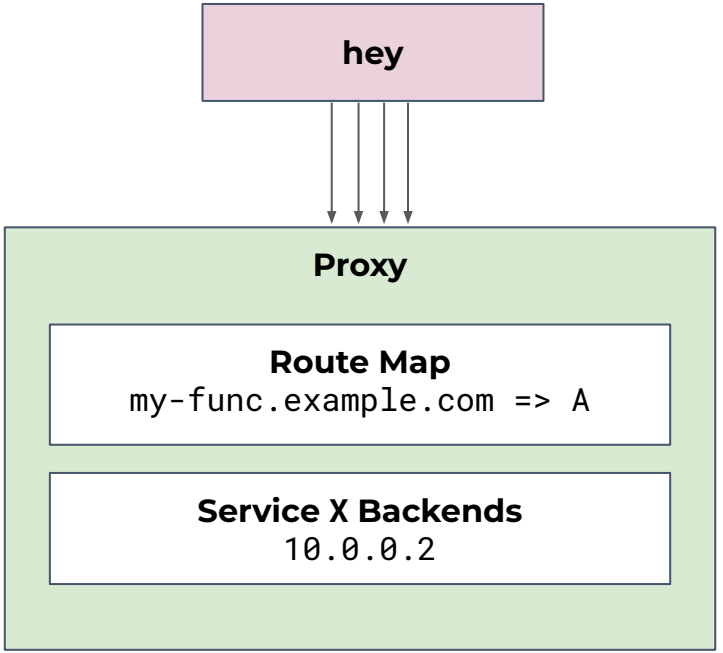


https://github.com/kubernetes-sigs/gateway-api/issues/1870

Update Operations are Unsafe

- Ingress doesn't track backend endpoints that aren't reachable via routes

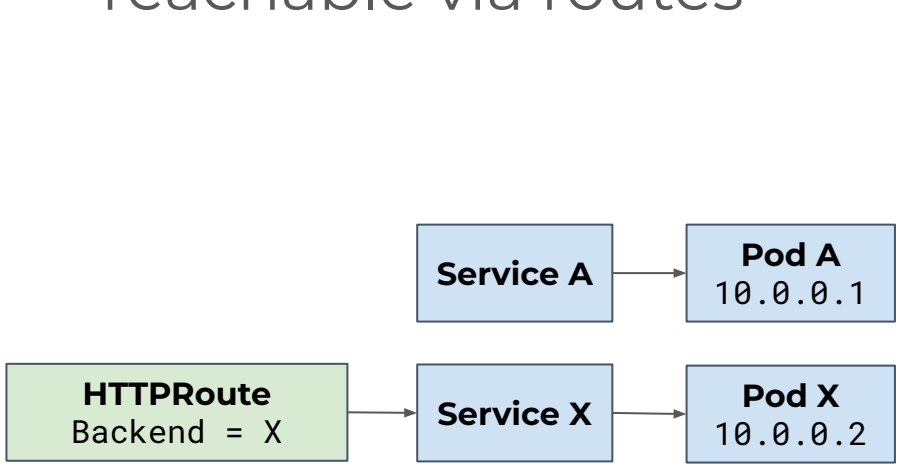


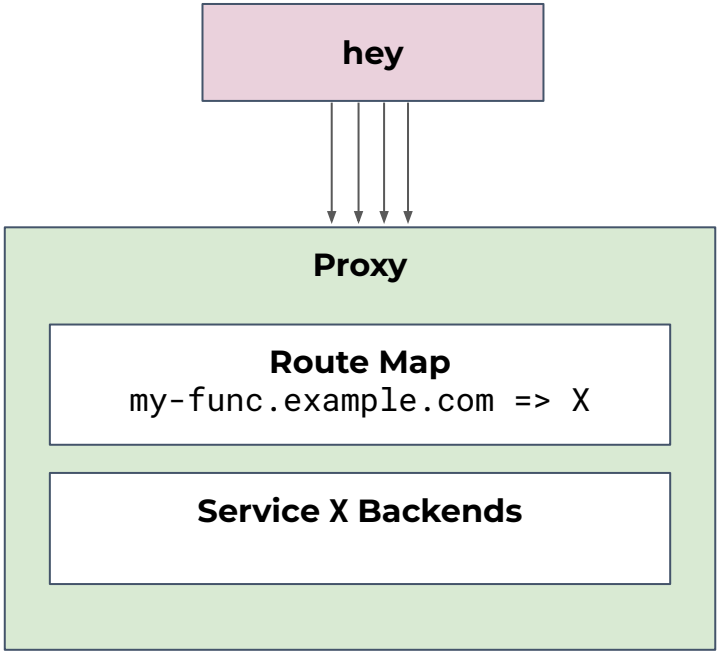


https://github.com/kubernetes-sigs/gateway-api/issues/1870

Update Operations are Unsafe

- Ingress doesn't track backend endpoints that aren't reachable via routes

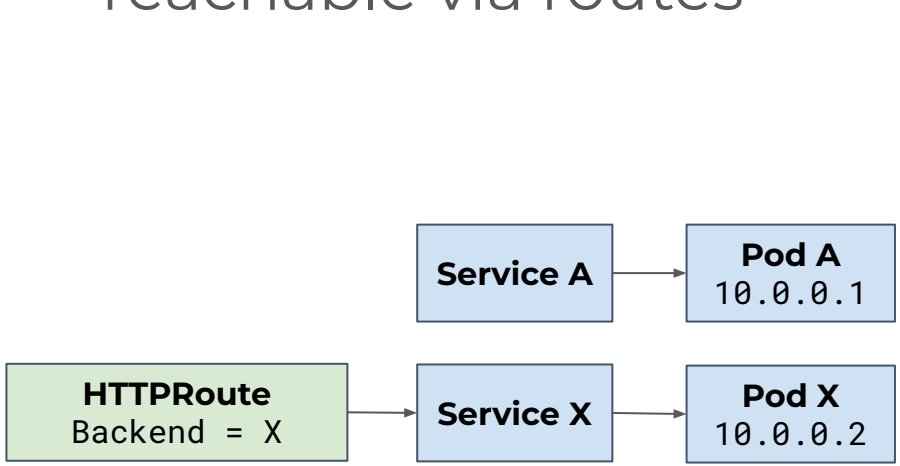


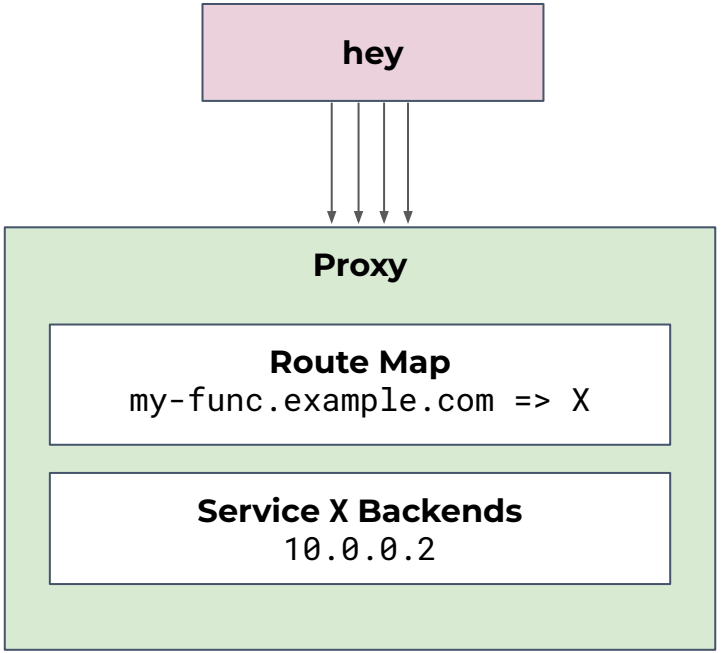


https://github.com/kubernetes-sigs/gateway-api/issues/1870

Update Operations are Unsafe

- Ingress doesn't track backend endpoints that aren't reachable via routes

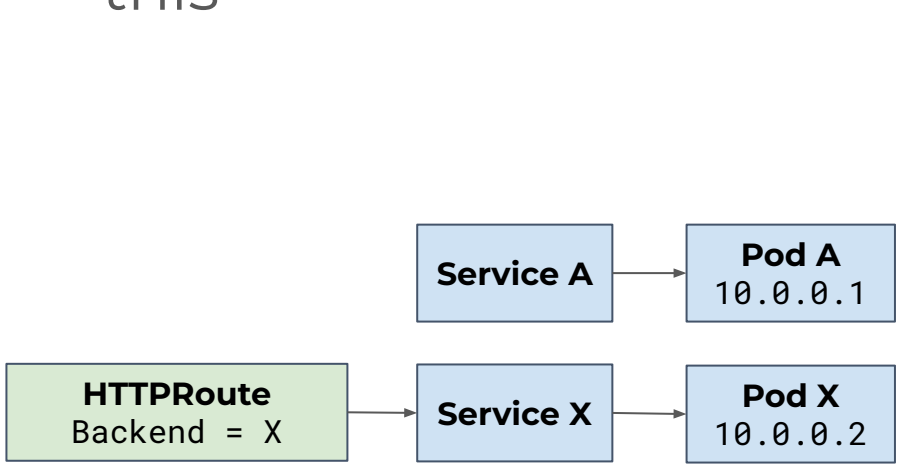


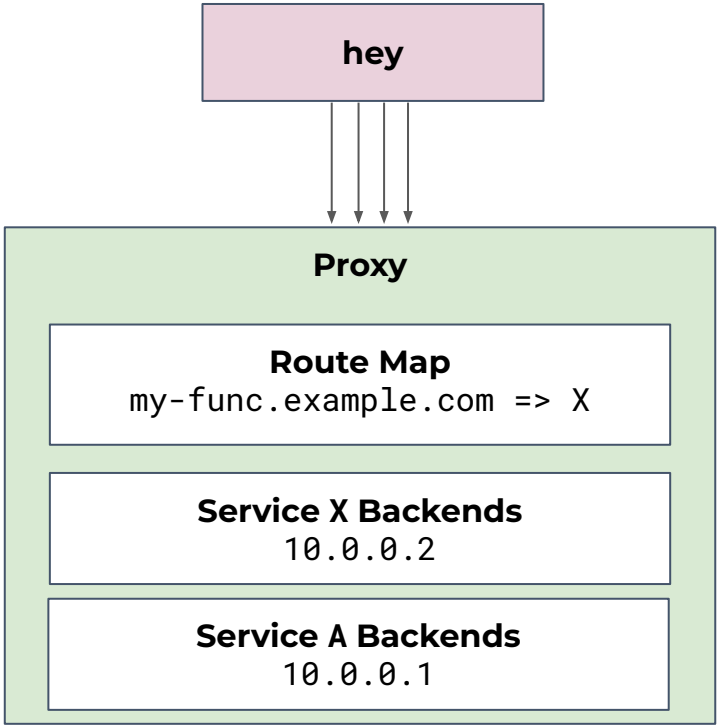


https://github.com/kubernetes-sigs/gateway-api/issues/1870

Update Operations are Unsafe

- Proxy needs to track all potential backends to avoid this

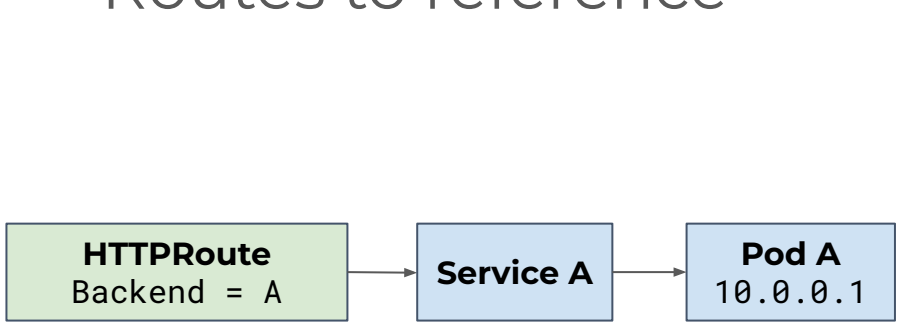


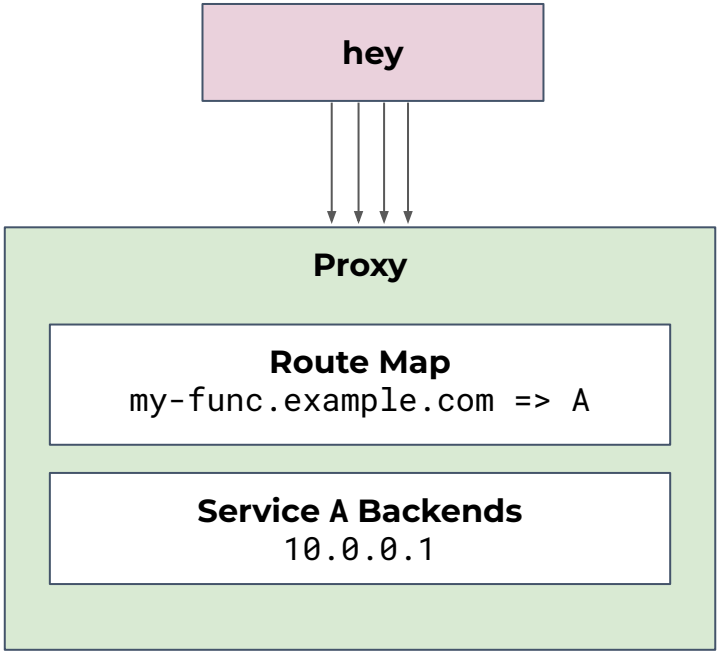


https://github.com/kubernetes-sigs/gateway-api/issues/1870

Update Operations are Unsafe

- Newly deployed healthy backends aren't safe for Routes to reference

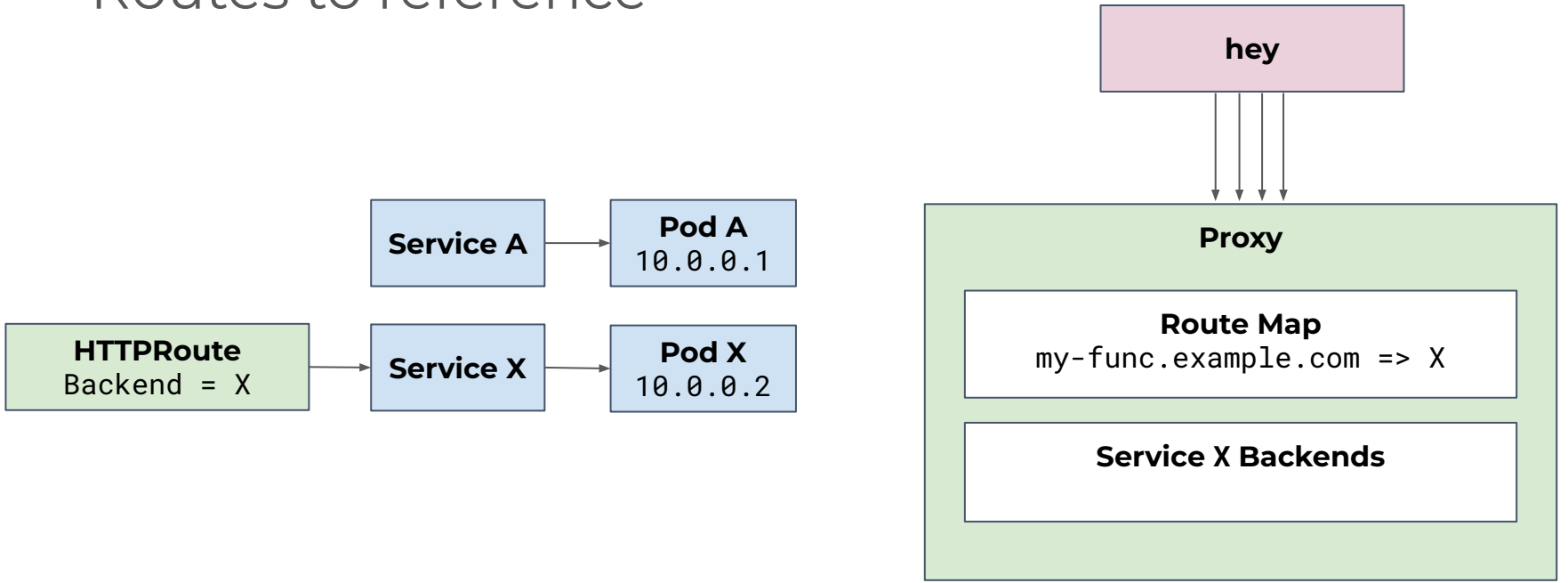




https://github.com/kubernetes-sigs/gateway-api/issues/1870

Update Operations are Unsafe

- Newly deployed healthy backends aren't safe for Routes to reference

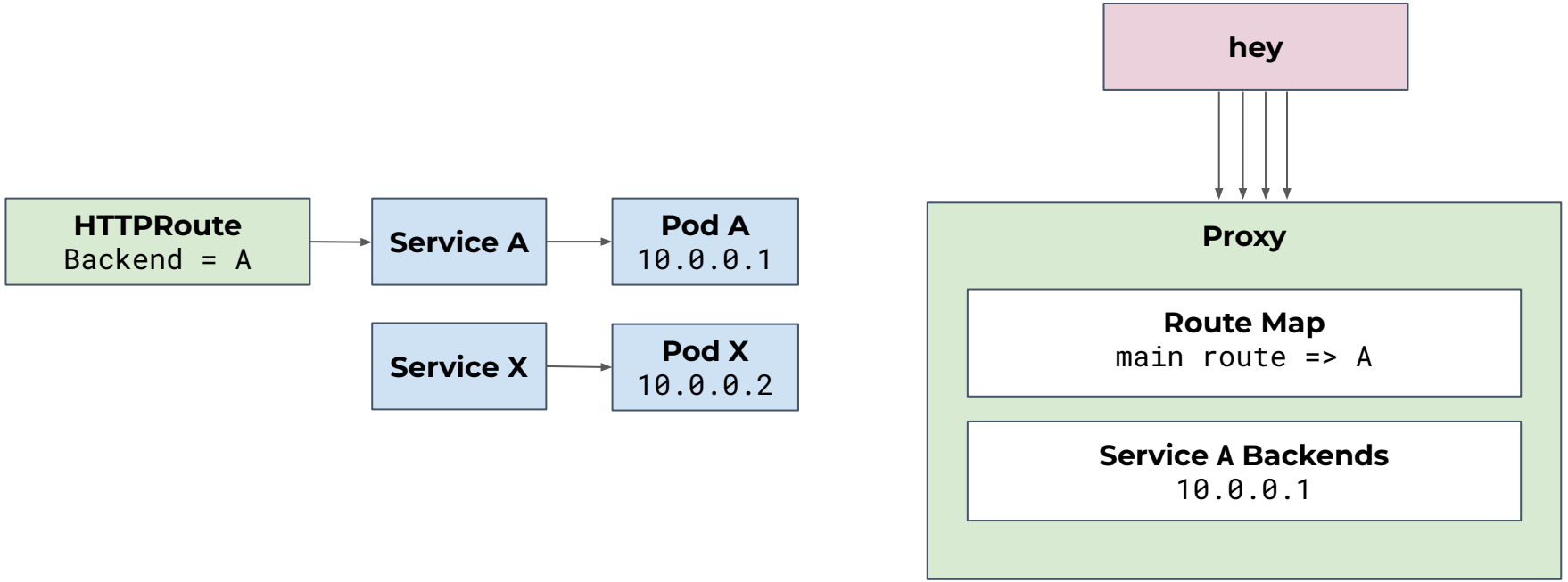


https://github.com/kubernetes-sigs/gateway-api/issues/1870


WHAT DO WE DO?

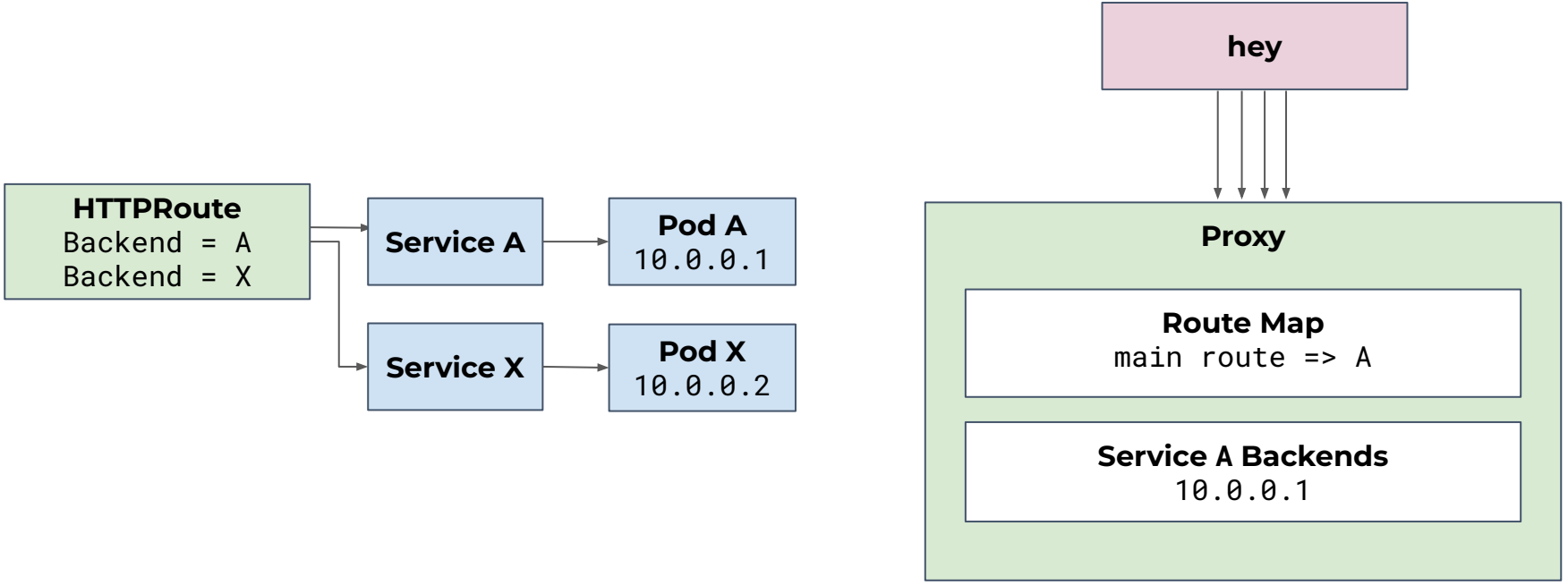
shoutout to @mattmoor

Three Phase Update



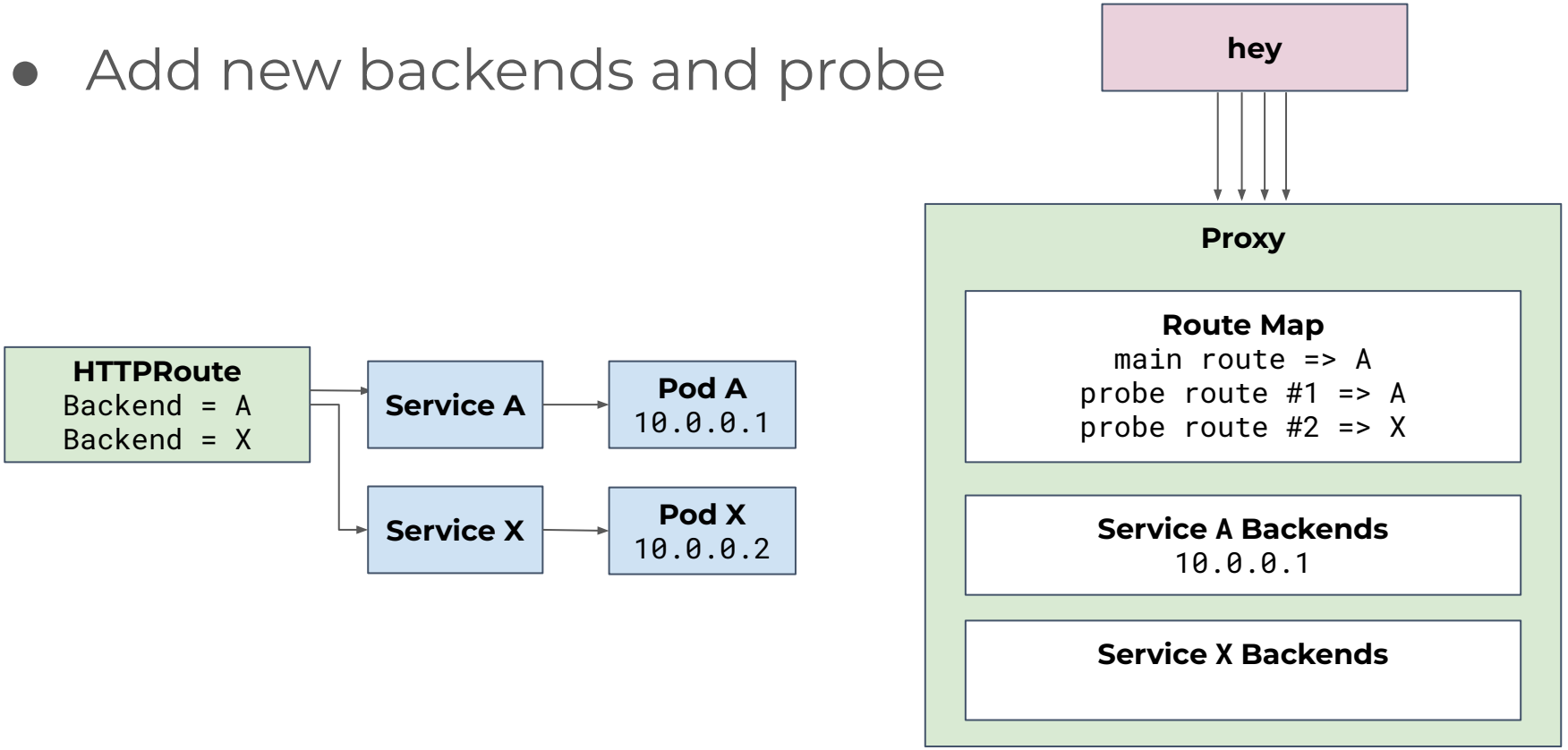
Three Phase Update

- Add new backends and probe



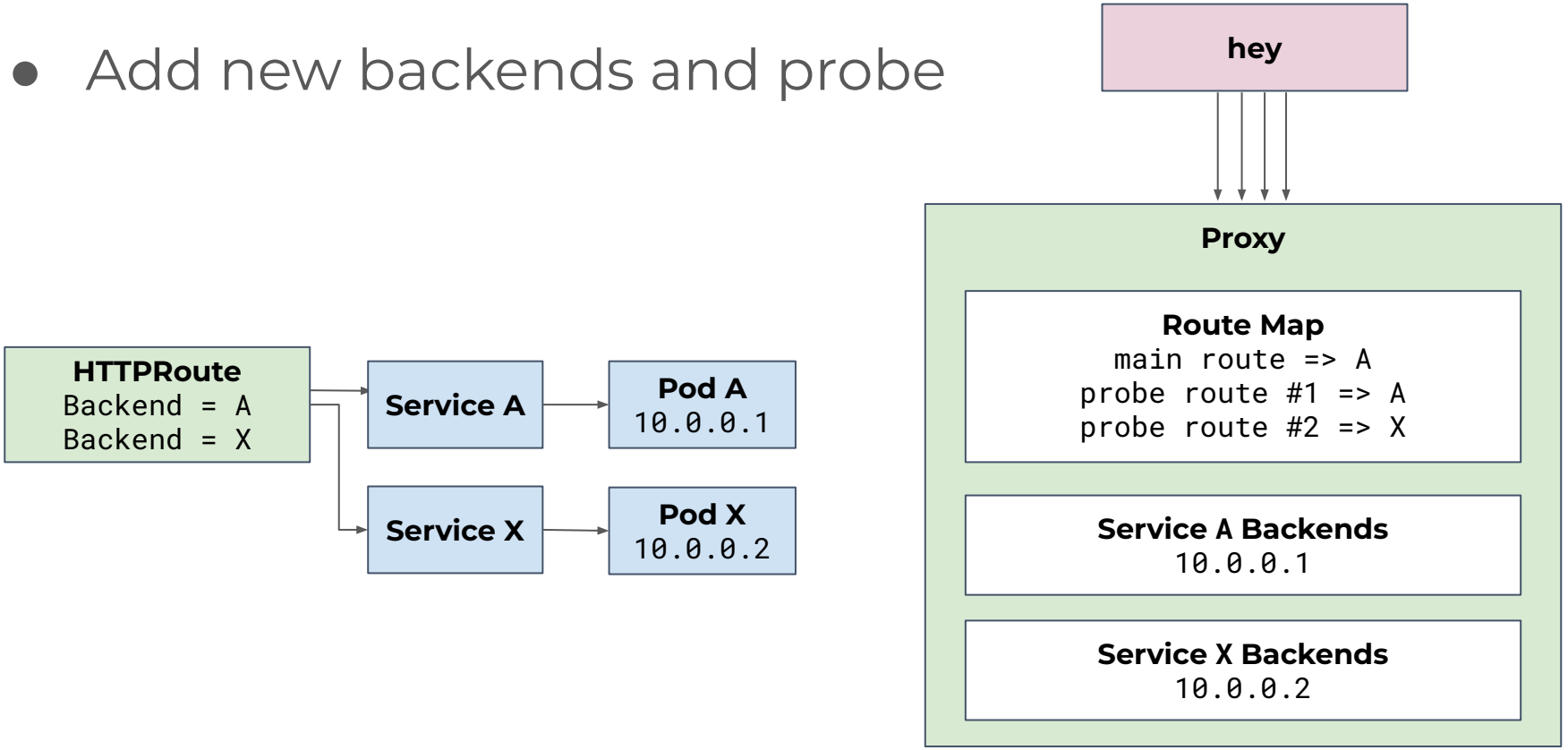
Three Phase Update

- Add new backends and probe



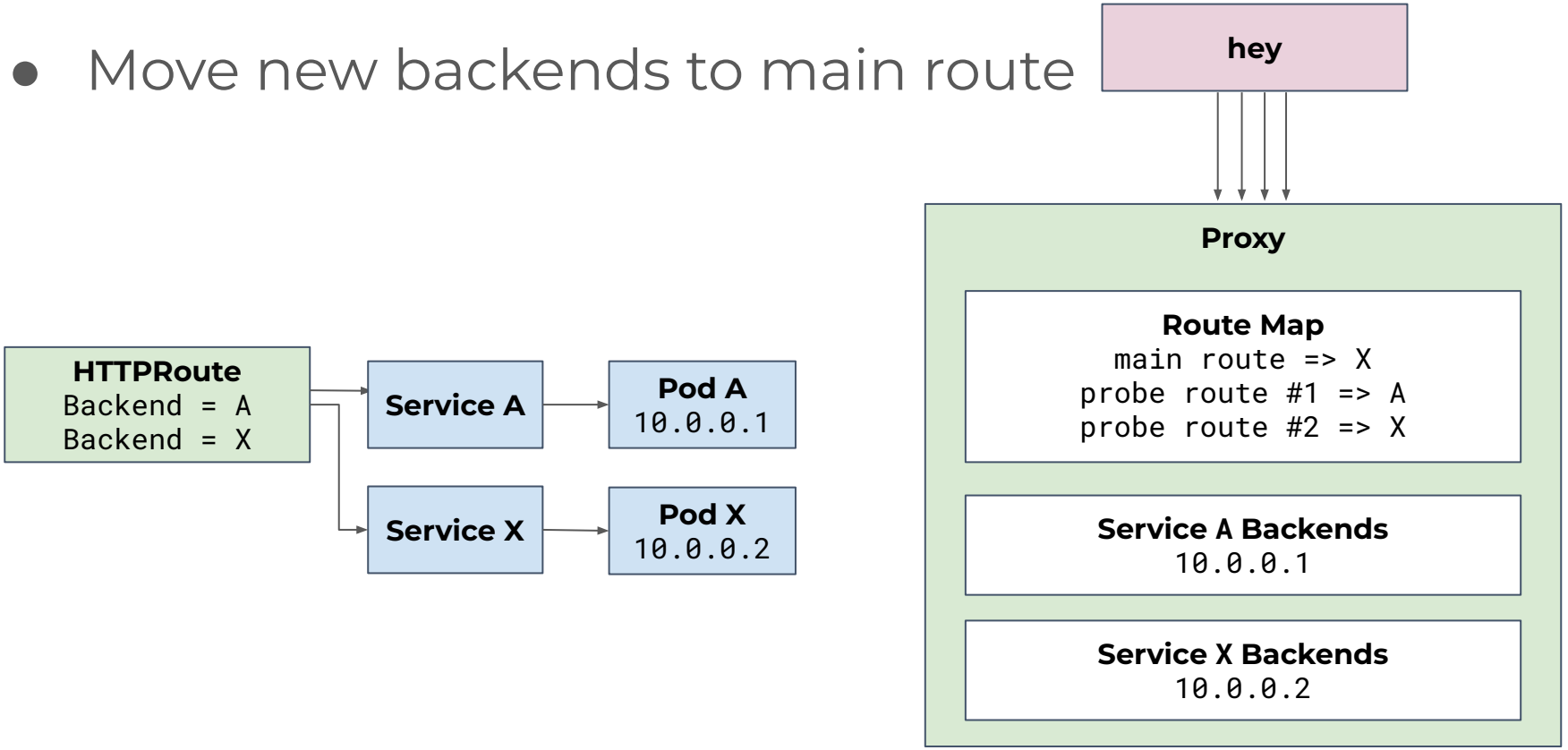
Three Phase Update

- Add new backends and probe



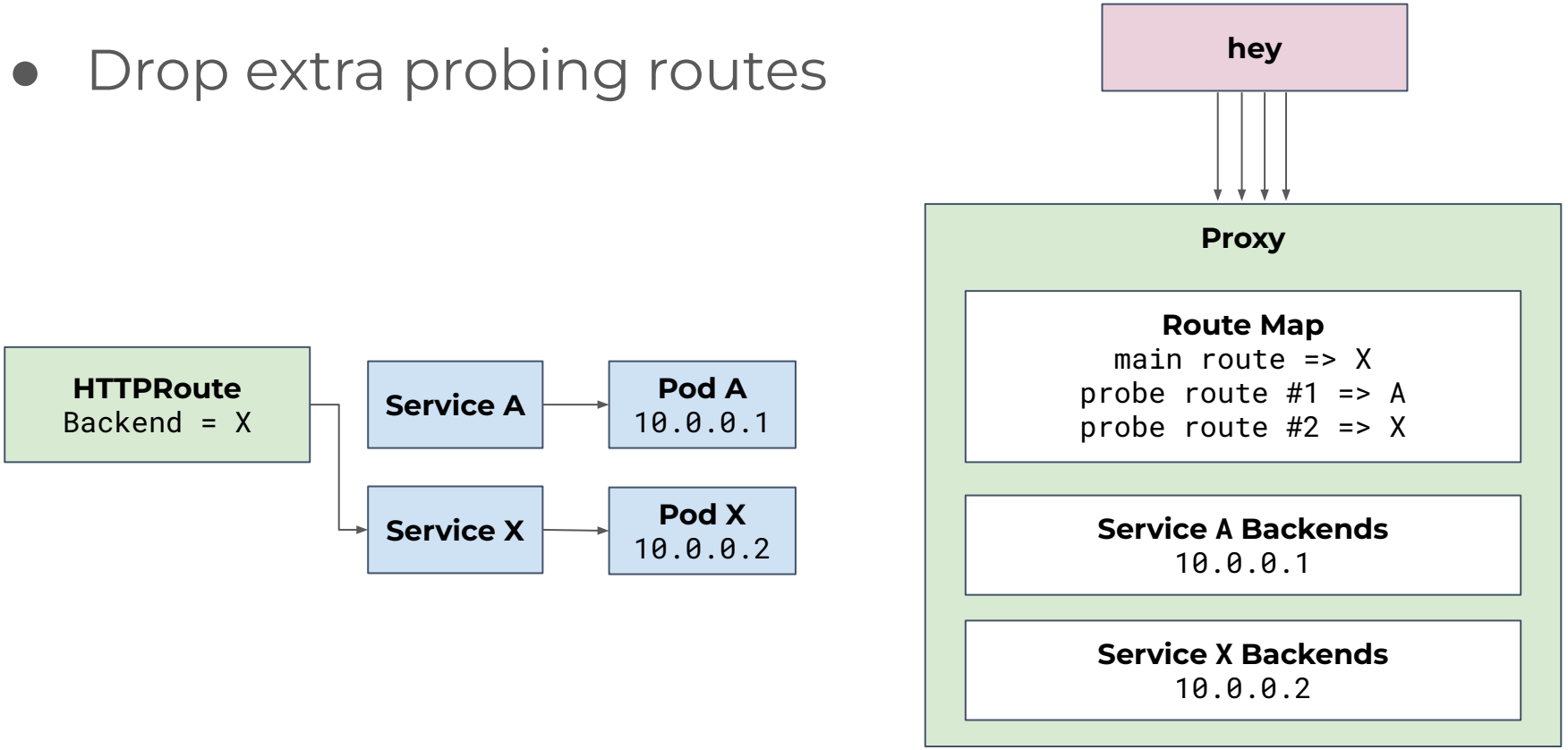
Three Phase Update

- Move new backends to main route



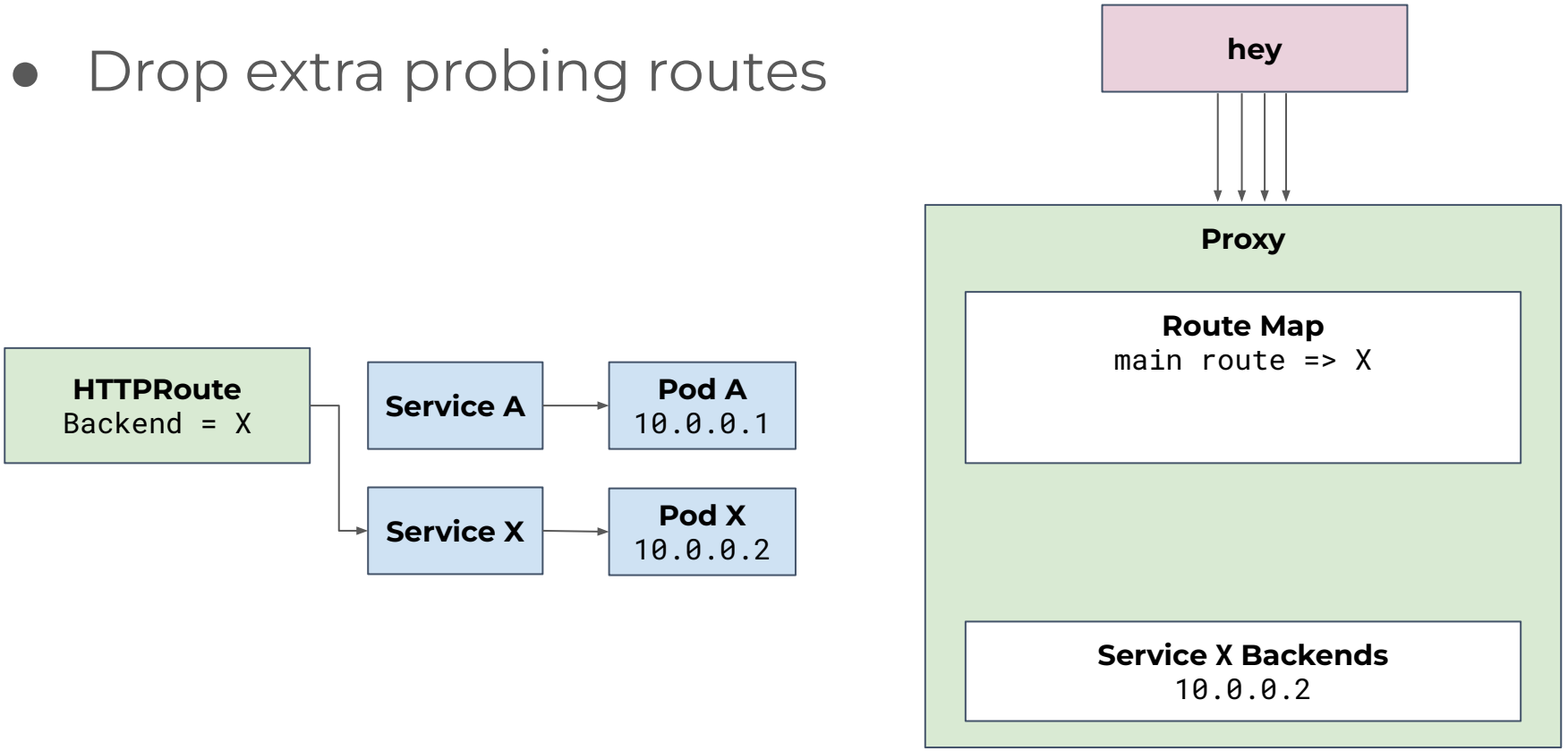
Three Phase Update

- Drop extra probing routes



Three Phase Update

- Drop extra probing routes



Recap: Three Phase Update

Gateway API Endpoint Probing - Design Document

1. Add new backends and probe
2. Move new backends to main route
3. Drop extra probing routes

Three Phase Update

- Proxies that tract a subset of Services in a cluster are most affected

  AND

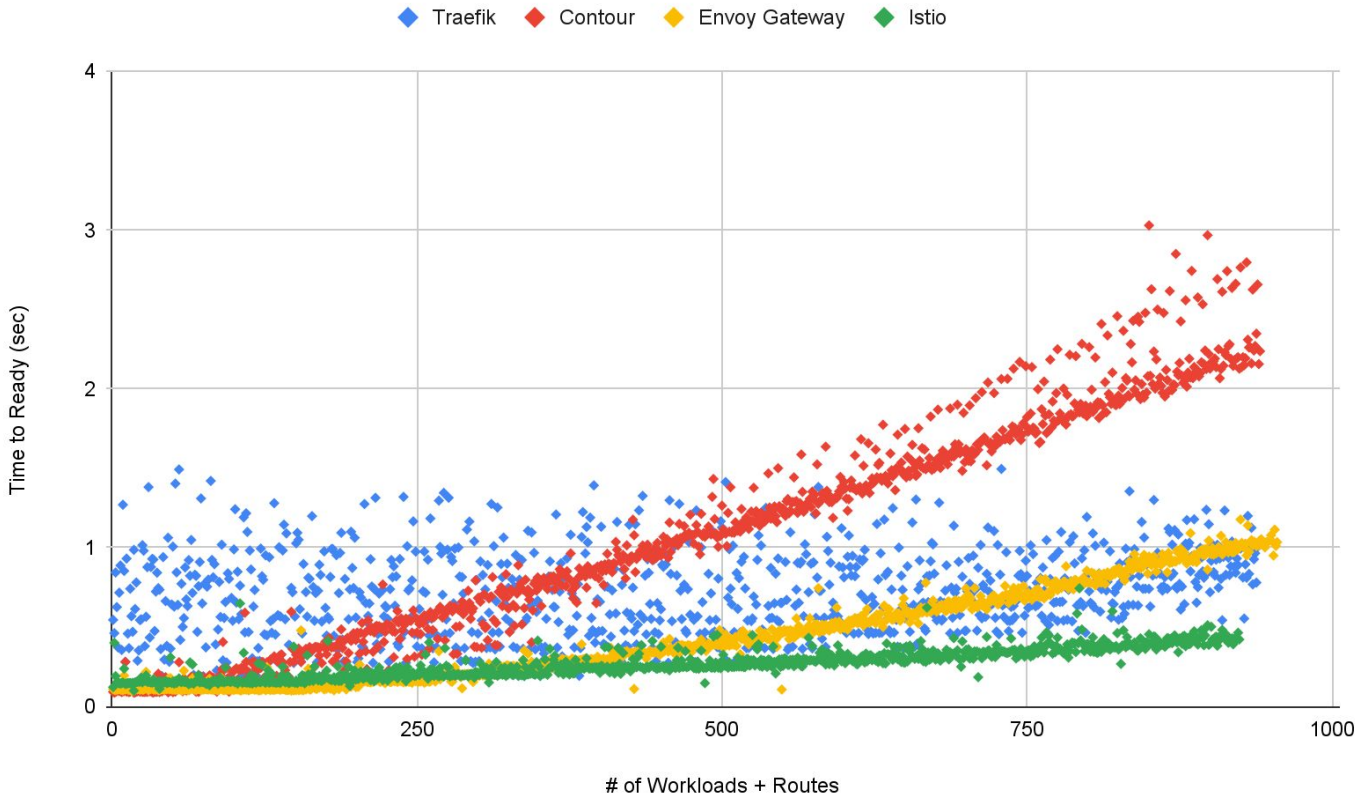
- If you perform route updates too soon you'll get failures
  - Workload is not Ready
  - Endpoints haven't been propagated to control plane

Metrics Time

Ready Time - Creating many HTTPRoutes

1. Create a Service & Pod
    - Wait for Endpoints to have a ready address
2. Create and HTTPRoute with a unique domain
3. Poll the route and determine how long it takes to become ready
4. Repeat 1-3 - 1000 times

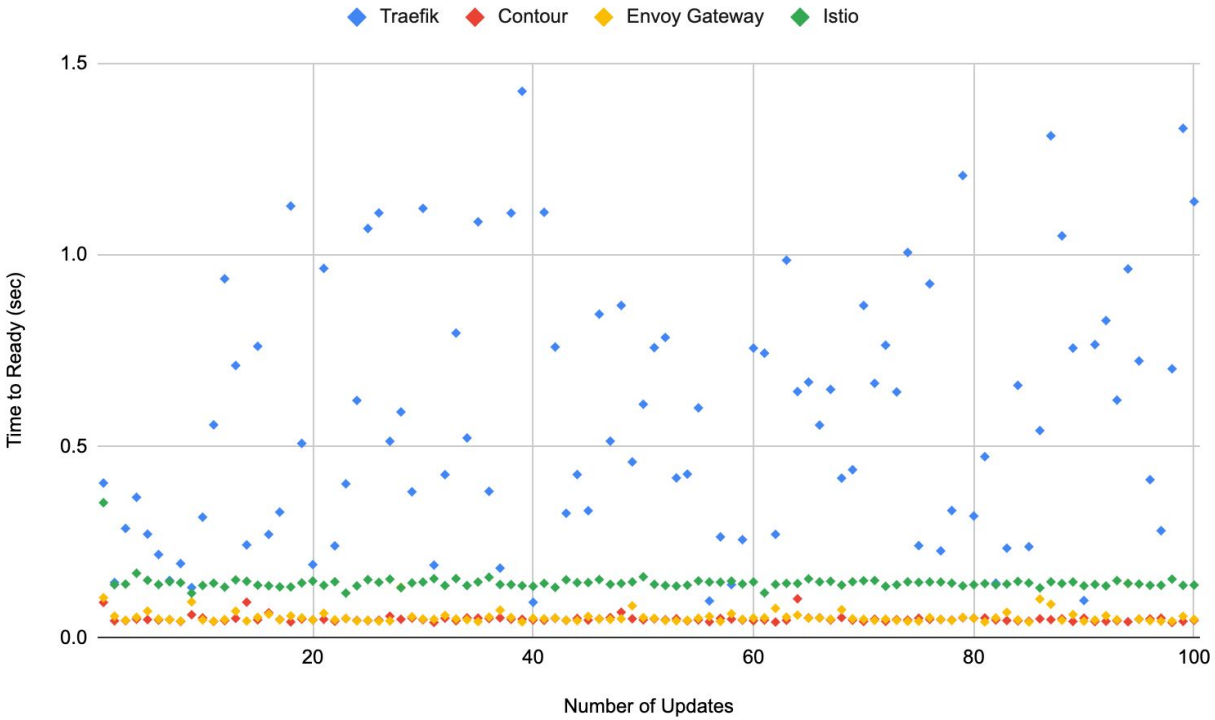
Readiness Time vs. Creation



Ready Time - Updating a Single Route

1. Create an initial Service & Pod
2. Create a single HTTPRoute and start polling
3. Create a new Service & Pod
    - Wait for Endpoints to have a ready address
4. Update the HTTPRoute
5. Repeat 4-5 100 times
6. Check poller for dropped traffic

Readiness Time vs. Updates

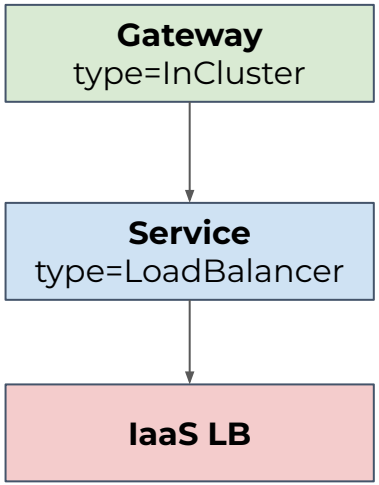


	Traefik	Contour	Envoy Gateway	Istio
SLI	100.00%	99.62%	100.00%	100.00%

GKE Testing

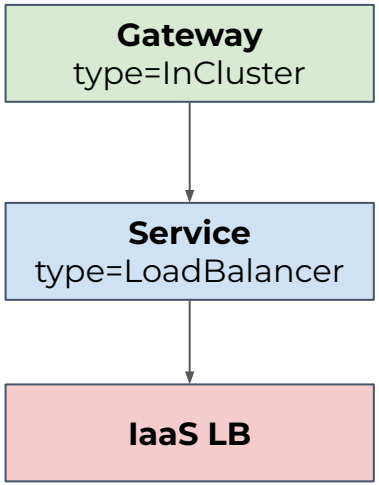
- GKE Managed Gateways
  - `gke-l7-regional-external-managed`
    - Ready time after CREATE **~ 1m29s**
    - Ready time after UPDATE **~1m40s**
      - Continuous Polling - **SLI 64%**
  - `gke-l7-gxlb`
    - Similar creation times as above
    - Updates never succeeded under **~2min**

Underlying K8s Cluster Quirks



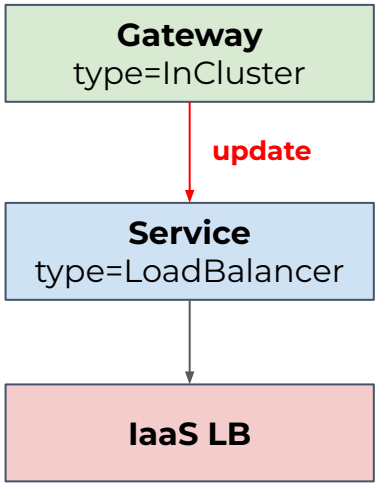

```yaml
apiVersion: gateway.networking.k8s.io/v1
kind: Gateway
metadata:
  name: prod-web
spec:
  gatewayClassName: example
  listeners:
  - protocol: HTTP
    port: 80
    name: web
```

Underlying K8s Cluster Quirks



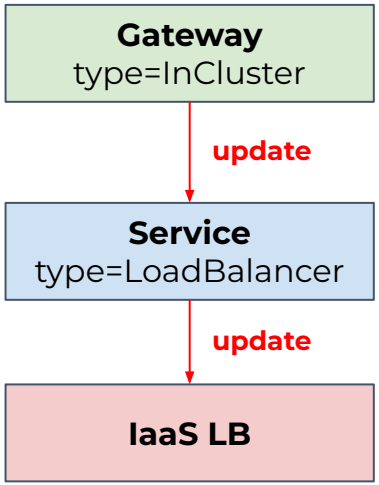

```yaml
apiVersion: gateway.networking.k8s.io/v1
kind: Gateway
metadata:
  name: prod-web
spec:
  gatewayClassName: example
  listeners:
  - protocol: HTTP
    port: 80
    name: web
  - protocol: HTTP
    port: 9090
    name: metrics
```

Underlying K8s Cluster Quirks



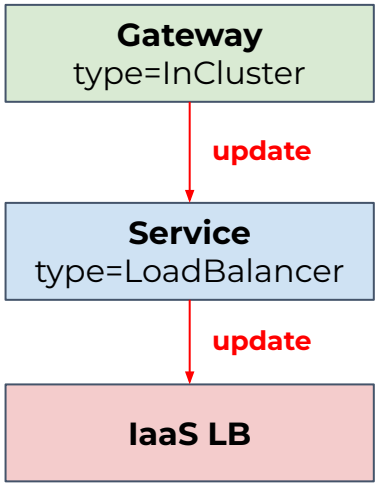

```yaml
apiVersion: gateway.networking.k8s.io/v1
kind: Gateway
metadata:
  name: prod-web
spec:
  gatewayClassName: example
  listeners:
  - protocol: HTTP
    port: 80
    name: web
  - protocol: HTTP
    port: 9090
    name: metrics
```

Underlying K8s Cluster Quirks




```yaml
apiVersion: gateway.networking.k8s.io/v1
kind: Gateway
metadata:
  name: prod-web
spec:
  gatewayClassName: example
  listeners:
  - protocol: HTTP
    port: 80
    name: web
  - protocol: HTTP
    port: 9090
    name: metrics
```

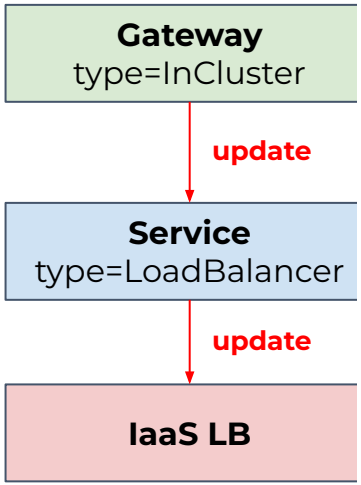
Underlying K8s Cluster Quirks

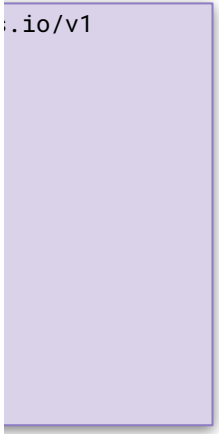



```yaml
apiVersion: gateway.networking.k8s.io/v1
kind: Gateway
metadata:
  name: prod-web
spec:
  gatewayClassName: example
  listeners:
  - protocol: HTTP
    port: 80
    name: web
  - protocol: HTTP
    port: 9090
    name: metrics
```

- On GKE `ServicePort` changes bring down the whole LB
  - In above example continuous traffic hitting port `80` will fail
  - Upvote issue here **https://bit.ly/google-fix-my-lb**

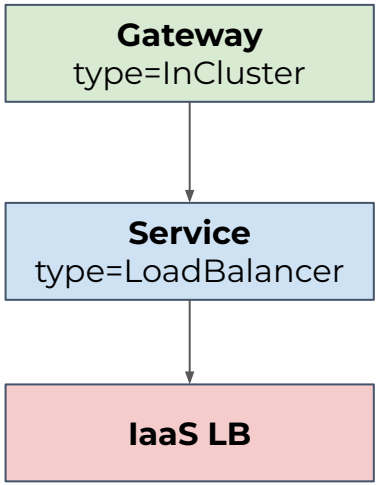
Underlying K8s Cluster Quirks





- On GKE Ser ole LB
  - In above e
  - Upvote iss

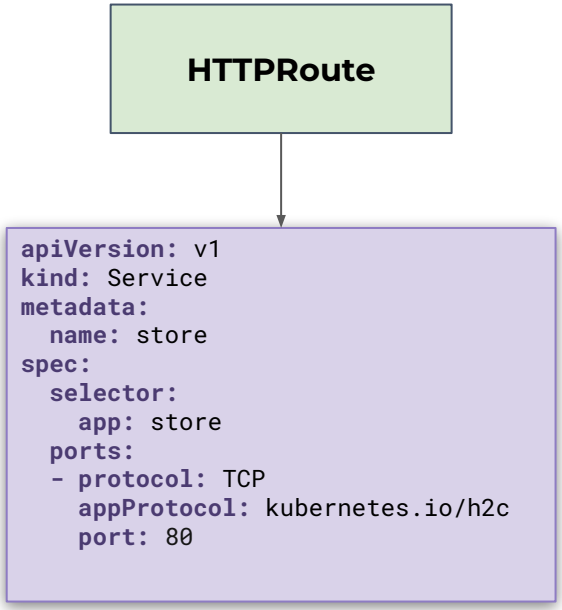
Underlying K8s Cluster Quirks




```yaml
apiVersion: gateway.networking.k8s.io/v1
kind: Gateway
metadata:
  name: prod-web
spec:
  gatewayClassName: example
  listeners:
  - protocol: HTTP
    port: 80
    name: web
  - protocol: HTTP
    port: 9090
    name: metrics
  infrastructure:
    labels: {}
    annotations: {}
```

https://gateway-api.sigs.k8s.io/geps/gep-1867/

Backend Protocol Selection



- `kubernetes.io/h2c` - HTTP/2 Prior Knowledge
- `kubernetes.io/ws` - WebSocket over HTTP

https://gateway-api.sigs.k8s.io/guides/backend-protocol/

GEP-1713: Merging Listeners


```yaml
apiVersion: gateway.networking.k8s.io/v1
kind: Gateway
metadata:
  name: prod-web
spec:
  gatewayClassName: example
  listeners:
  - protocol: HTTP
    port: 80
    name: web
  - protocol: HTTPS
    port: 443
    name: foo
    hostname: foo.example.com
    tls:
      certificateRefs:
      - kind: Secret
        group: ""
        name: foo-example-com-cert
```

Knative Serving

- Automatic Certificate Provisioning for HTTPS

Knative Serving

- Automatic Certificate Provisioning for HTTPS
  - Delegate to cert-manager
  - HTTP-01 is most popular challenge type
    - Doesn’t support wildcard certificates

GEP-1713: Merging Listeners


```yaml
apiVersion: gateway.networking.k8s.io/v1
kind: Gateway
metadata:
  name: prod-web
spec:
  gatewayClassName: example
  listeners:
  - protocol: HTTP
    port: 80
    name: web
  - protocol: HTTPS
    port: 443
    name: foo
    hostname: foo.example.com
    tls:
      certificateRefs:
      - kind: Secret
        group: ""
        name: foo-example-com-cert
```

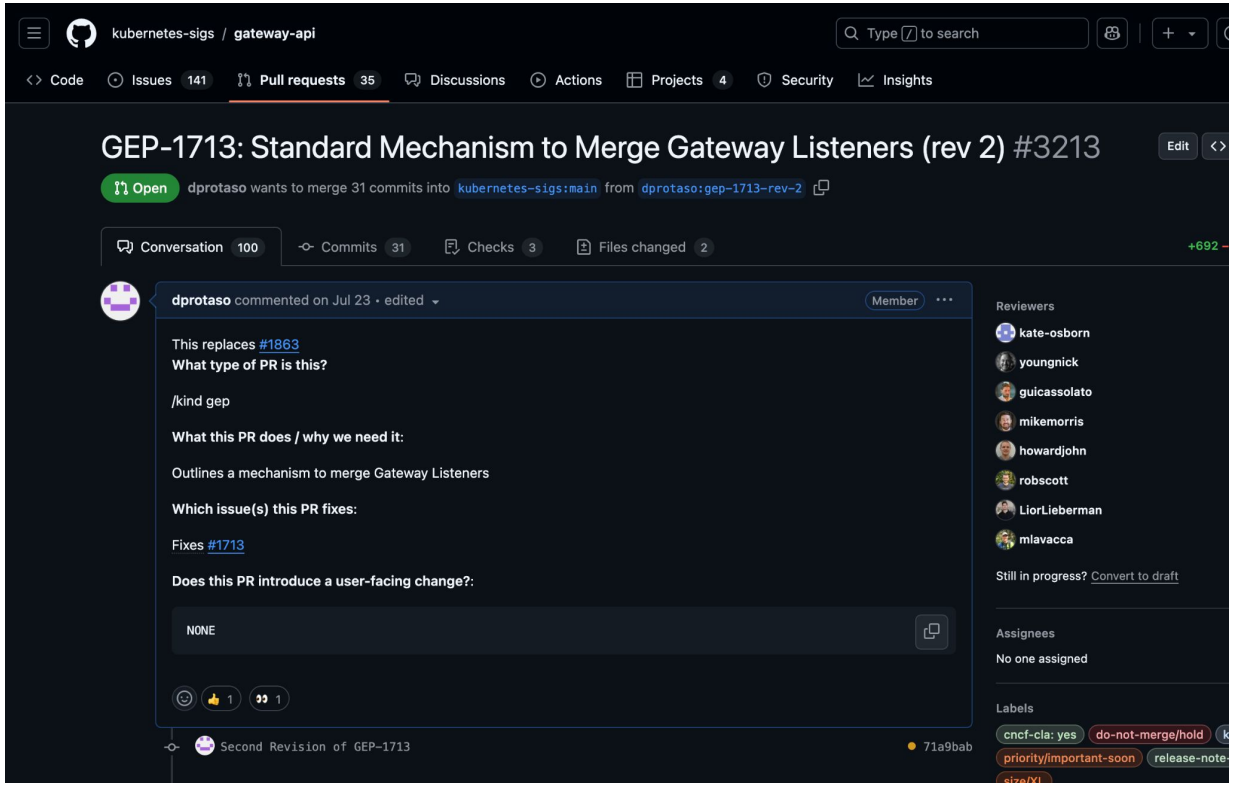
GEP-1713: Merging Listeners


```yaml
apiVersion: gateway.networking.k8s.io/v1
kind: Gateway
metadata:
  name: prod-web
spec:
  gatewayClassName: example
  listeners:
  - protocol: HTTP
    port: 80
    name: web
  - protocol: HTTPS
    port: 443
    name: foo
    hostname: foo.example.com
    tls:
      certificateRefs:
      - kind: Secret
        group: ""
        name: foo-example-com-cert
  - protocol: HTTPS
    port: 443
    name: bar
    hostname: bar.example.com
    tls:
      certificateRefs:
      - kind: Secret
        group: ""
        name: bar-example-com-cert
```

GEP-1713: Merging Listeners


```
//
// Implementations MAY merge separate Gateways onto a single s
// Addresses if all Listeners across all Gateways are compatib
//
// Support: Core
//
// +listType=map
// +listMapKey=name
// +kubebuilder:validation:MinItems=1
// +kubebuilder:validation:MaxItems=64
// +kubebuilder:validation:XValidation:message="tls must not b
// +kubebuilder:validation:XValidation:message="tls mode must
// +kubebuilder:validation:XValidation:message="hostname must
// +kubebuilder:validation:XValidation:message="Listener name
// +kubebuilder:validation:XValidation:message="Combination of
Listeners []Listener `json:"listeners"`
```

GEP-1713: Merging Listeners



https://github.com/kubernetes-sigs/gateway-api/pull/3213

GEP-1713: Merging Listeners

- Discussions are ongoing!
- Comment on PR/original issue
- Maintainer meeting @ KubeCon
  - Ambassador and Maintainer Lounge: 155D
  - Nov 13 - 3pm

Questions and Feedback!

dprotaso

lintinmybelly

dprotasowski