
KubeCon
CloudNativeCon
North America 2024

Extending the Gateway API: The Power and Challenges of Policy

Kate Osborn, NGINX

What is the Gateway API?

“Next generation of Kubernetes Ingress, Load Balancing, and Service Mesh APIs”

Extending Ingress Today

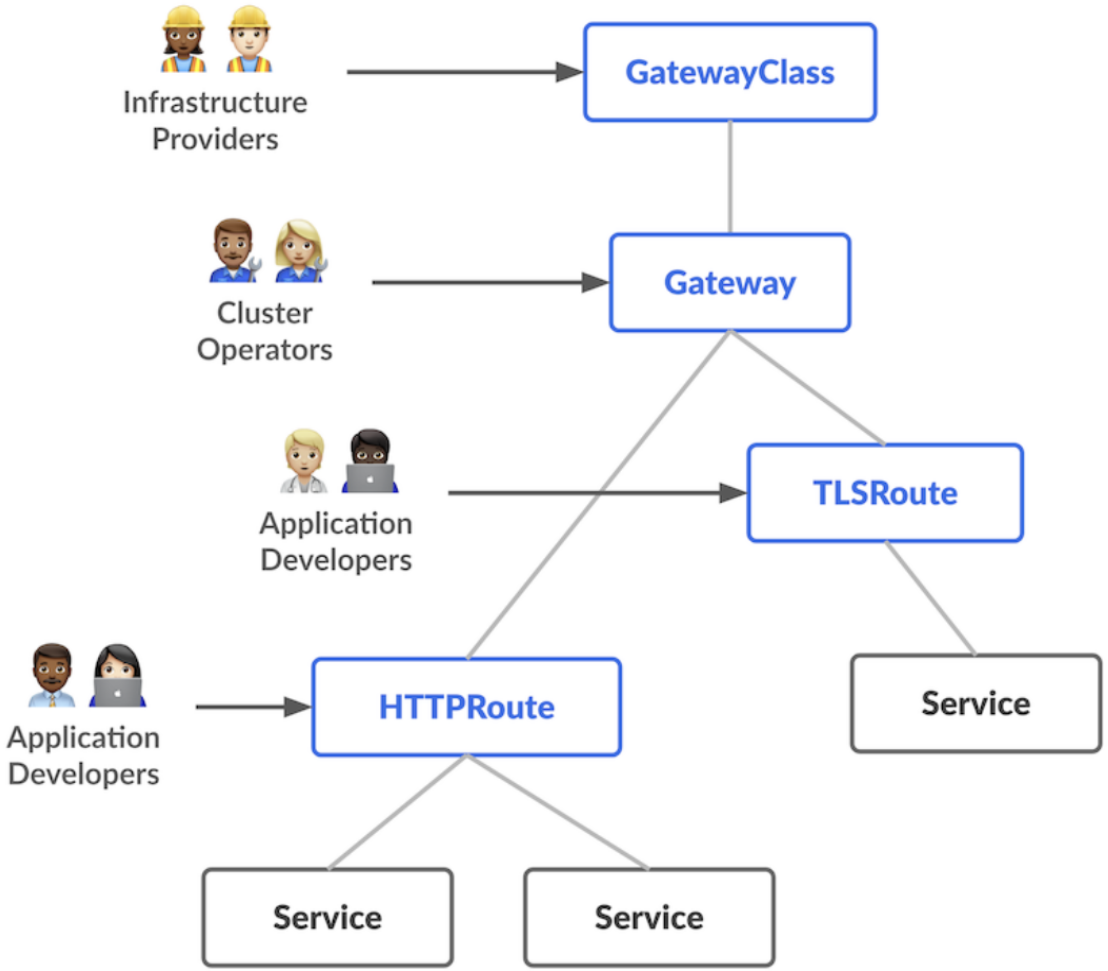
- nginx.org/proxy-connect-timeout
- nginx.org/proxy-read-timeout
- nginx.org/proxy-send-timeout
- nginx.org/client-max-body-size
- nginx.org/proxy-buffering
- nginx.org/proxy-buffers
- nginx.org/proxy-buffer-size
- nginx.org/proxy-max-temp-file-size
- nginx.org/server-tokens
- nginx.org/path-regex
- nginx.org/proxy-hide-headers
- nginx.org/proxy-pass-headers
- nginx.org/rewrites
- nginx.org/proxy-set-headers
- nginx.org/redirect-to-https
- nginx.org/hsts
- nginx.org/hsts-max-age
- nginx.org/hsts-include-subdomains
- nginx.org/hsts-behind-proxy
- nginx.org/basic-auth-secret
- nginx.org/basic-auth-realm
- nginx.com/jwt-key
- nginx.com/jwt-realm
- nginx.com/jwt-token
- nginx.com/jwt-login-url
- nginx.org/listen-ports
- nginx.org/listen-ports-ssl
- nginx.org/lb-method
- nginx.org/ssl-services
- nginx.org/grpc-services
- nginx.org/websocket-services
- nginx.org/max-fails
- nginx.org/max-conns
- nginx.org/upstream-zone-size
- nginx.org/fail-timeout
- nginx.com/sticky-cookie-services
- nginx.org/keepalive
- nginx.com/health-checks
- nginx.com/health-checks-mandatory
- nginx.com/health-checks-mandatory-queue
- nginx.com/slow-start
- nginx.org/use-cluster-ip
- nginx.org/limit-req-rate
- nginx.org/limit-req-key
- nginx.org/limit-req-zone-size
- nginx.org/limit-req-delay
- nginx.org/limit-req-no-delay
- nginx.org/limit-req-burst
- nginx.org/limit-req-dry-run
- nginx.org/limit-req-log-level
- nginx.org/limit-req-reject-code
- nginx.org/limit-req-scale
- nginx.org/location-snippets
- nginx.org/server-snippets
- appprotect.f5.com/app-protect-policy
- appprotect.f5.com/app-protect-enable
- appprotect.f5.com/app-protect-security-log-enable
- appprotect.f5.com/app-protect-security-log
- appprotect.f5.com/app-protect-security-log-destination

nginx.org/rewrites: "serviceName=service1 rewrite=rewrite1[;serviceName=service2 rewrite=rewrite2;...]"

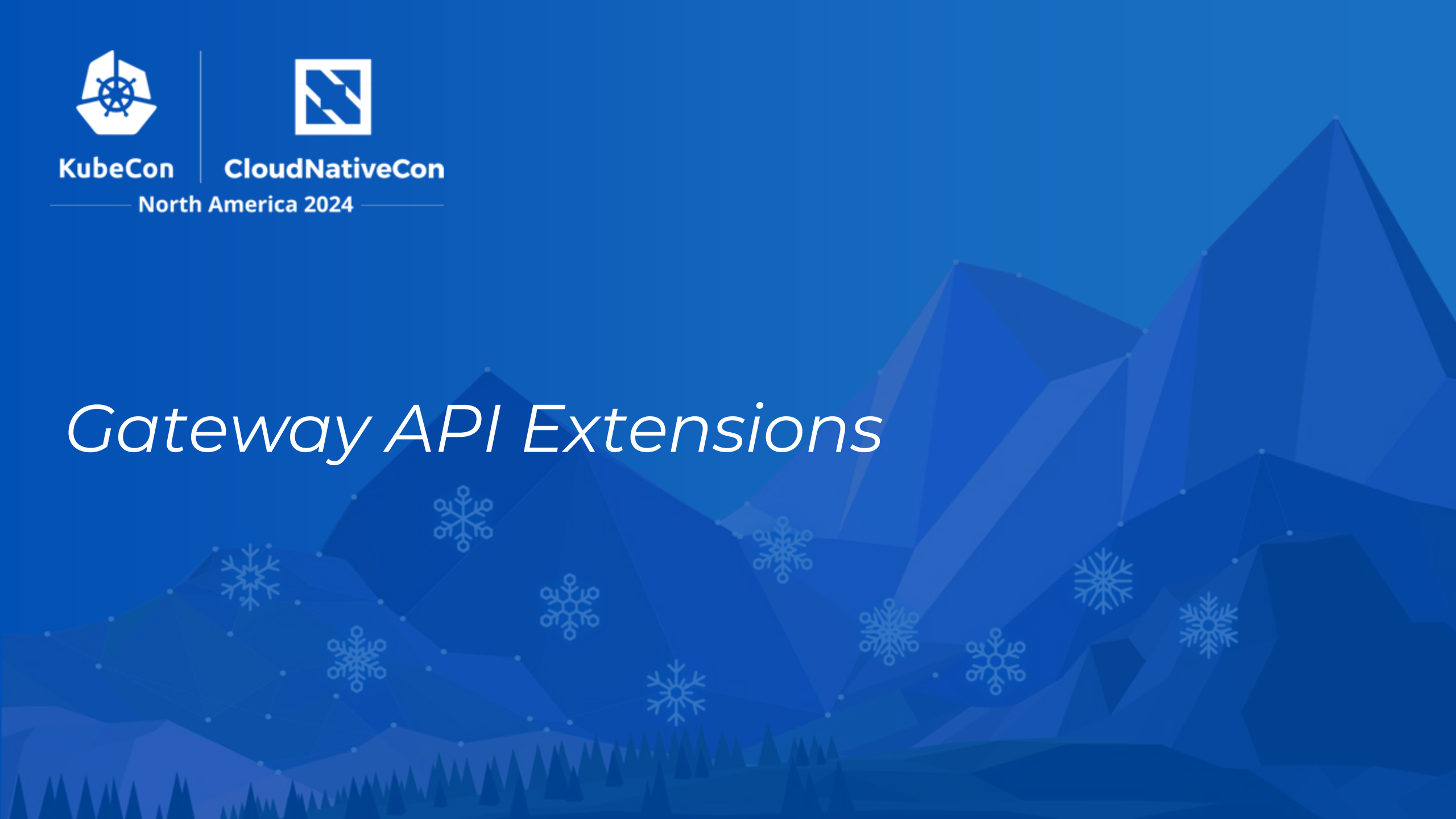
Extending Ingress Example


```yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: cafe-ingress
  annotations:
    nginx.org/proxy-connect-timeout: "30s"
    nginx.org/proxy-read-timeout: "20s"
    nginx.org/client-max-body-size: "4m"
    nginx.org/rewrites: "serviceName=tea-svc rewrite=/;serviceName=coffee-svc rewrite=/beans/"
spec:
  rules:
    - host: cafe.example.com
      http:
        paths:
          - path: /tea/
            backend:
              service:
                name: tea-svc
          - path: /coffee/
            backend:
              service:
                name: coffee-svc
```

Gateway API Concepts

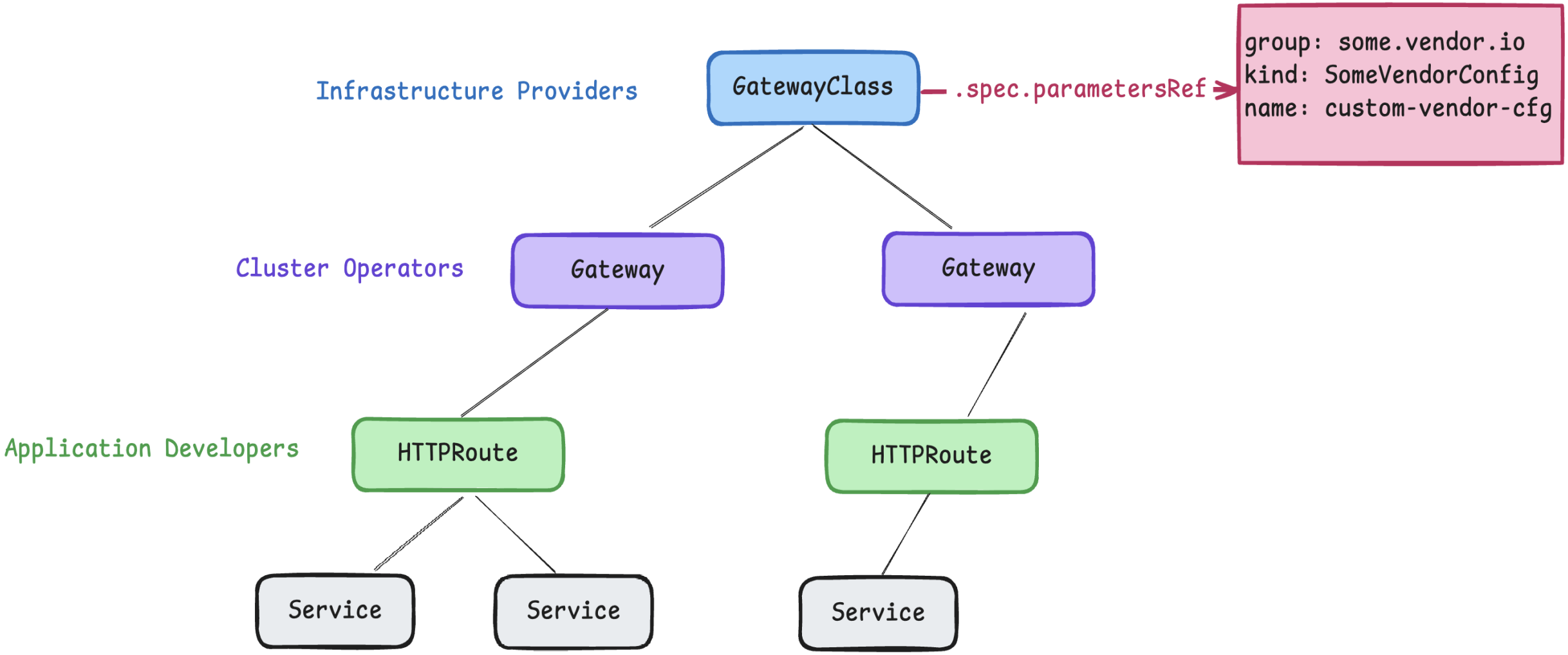


- Role-Oriented
- Expressive
- Portable
- Extensible

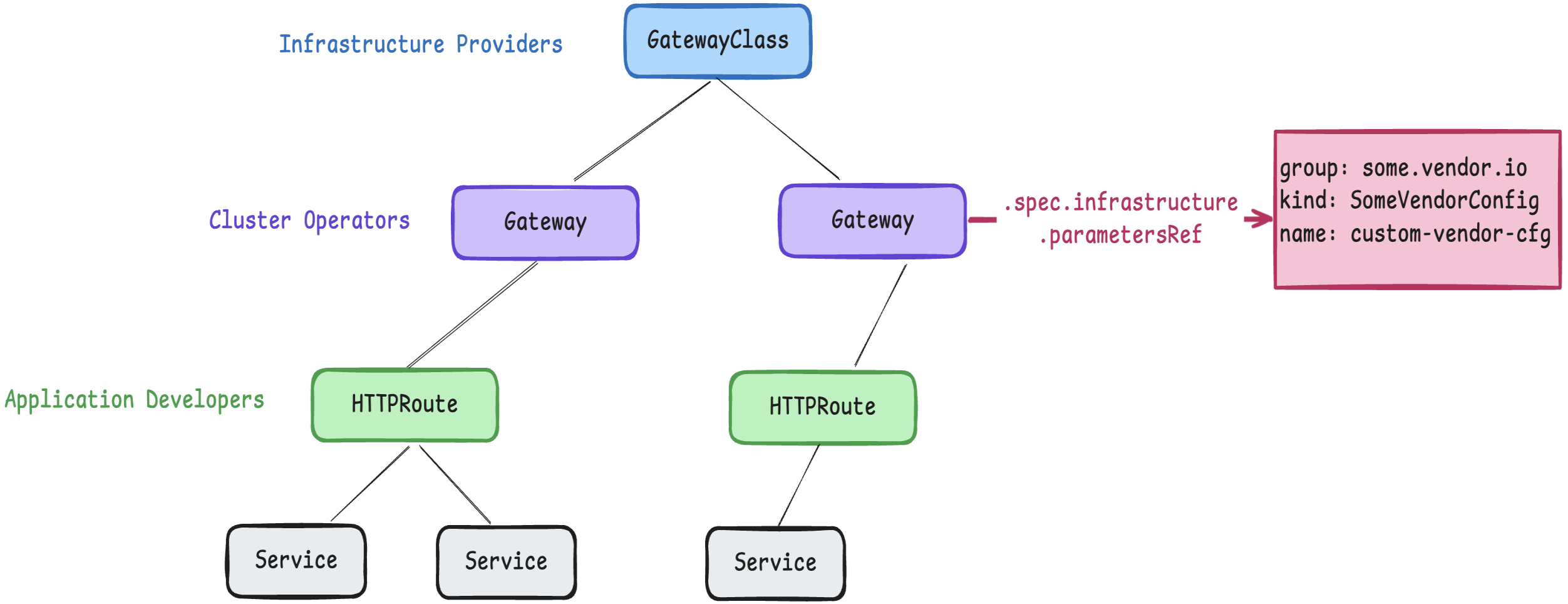


Gateway API Extensions

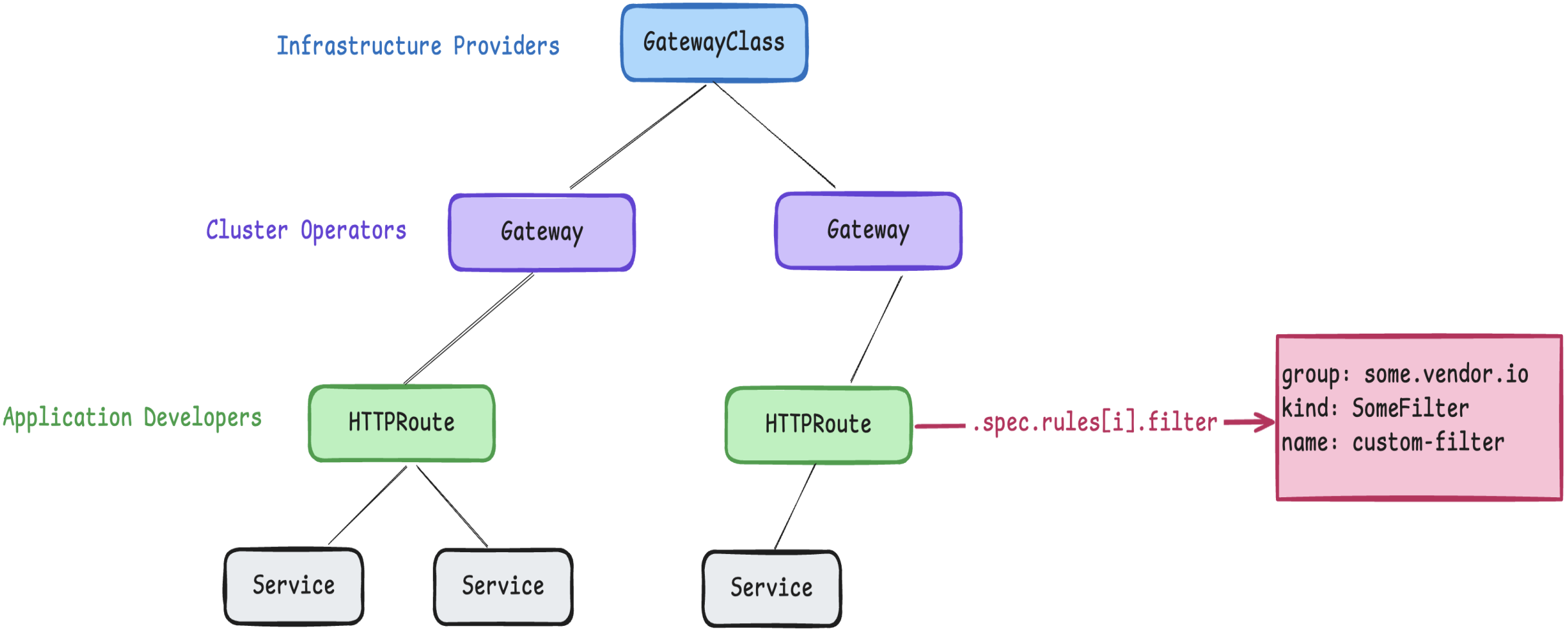
GatewayClass ParametersRef



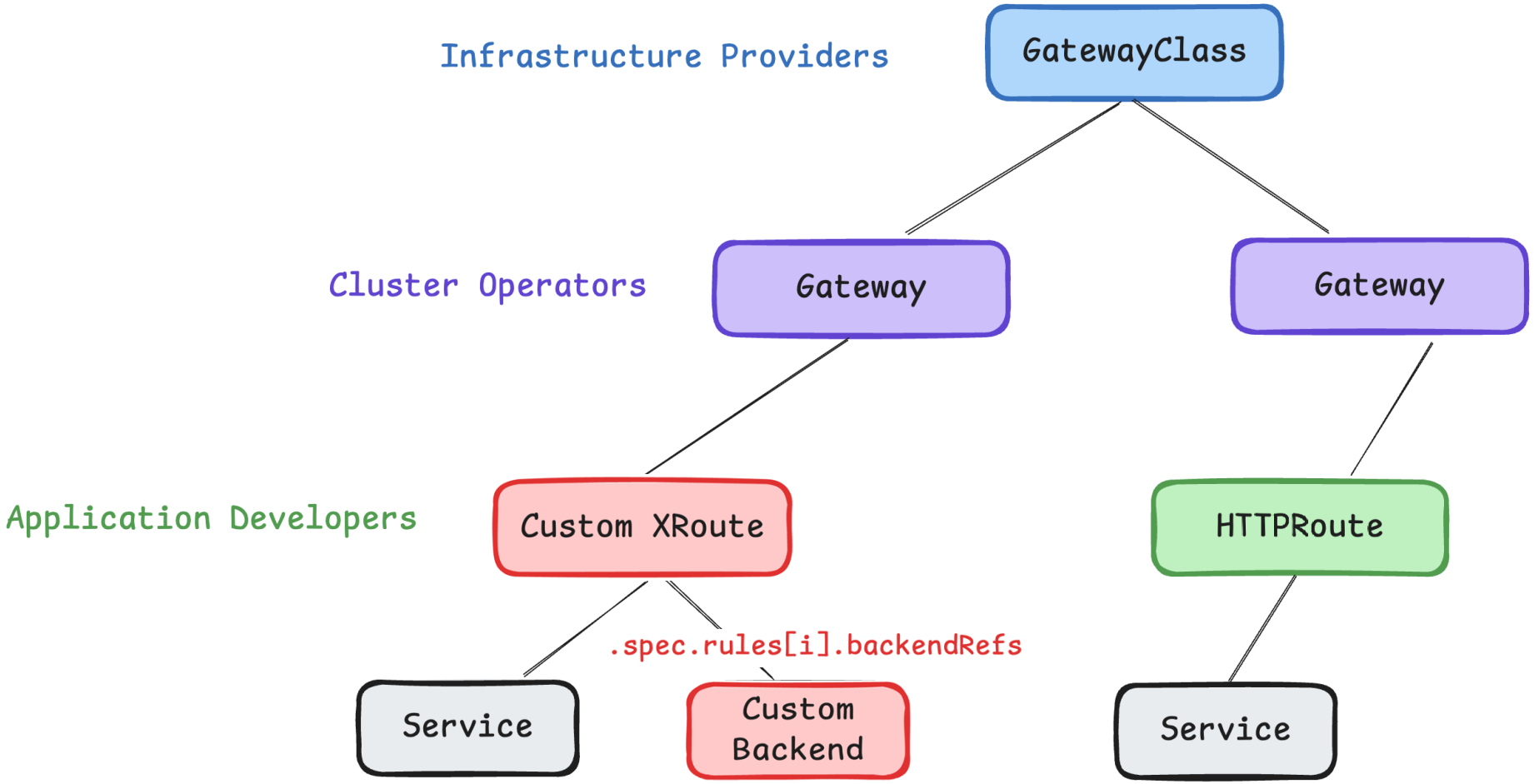
Gateway Infrastructure



Route Filters



Custom BackendRefs and Routes



Is this extensible enough?

The Case for More Extensibility

Cluster Operators

*"We want to **mandate** security policies for all applications."*

*"We want to set **sane defaults** for all applications."*

Application Developers

*"We want to **fine-tune** settings based on our application's behavior."*

Any

*"We want to **change** the behavior of **Services** or **Namespaces** without changing their specs."*

Is it possible to satisfy these use cases while maintaining a consistent user experience across implementations?

Policy Attachment

Disclaimer: Policy Attachment is a work in progress! The details in the following slides may change.

What is Policy Attachment?

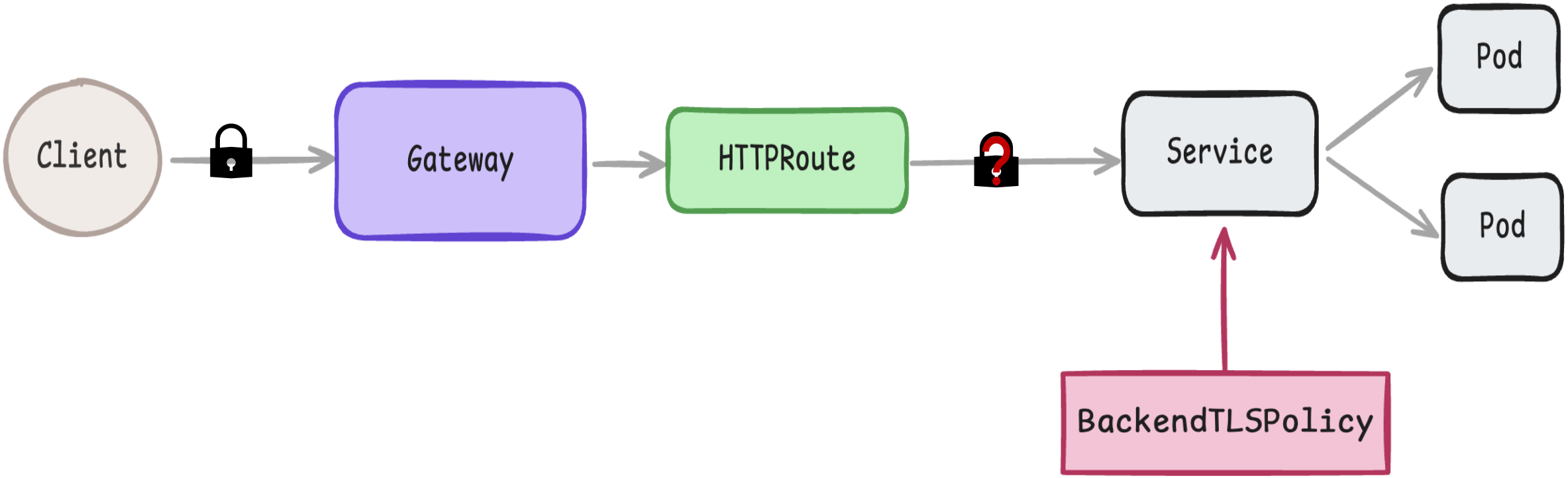
Policy Attachment: a specific type of metaresource that can affect specific settings across either one object ("Direct Policy Attachment"), or objects in a hierarchy ("Inherited Policy Attachment").

Metaresource: a Kubernetes object that augments the behavior of an object in a standard way.

Policy: a metaresouce. Also, a CRD.

Direct Policies

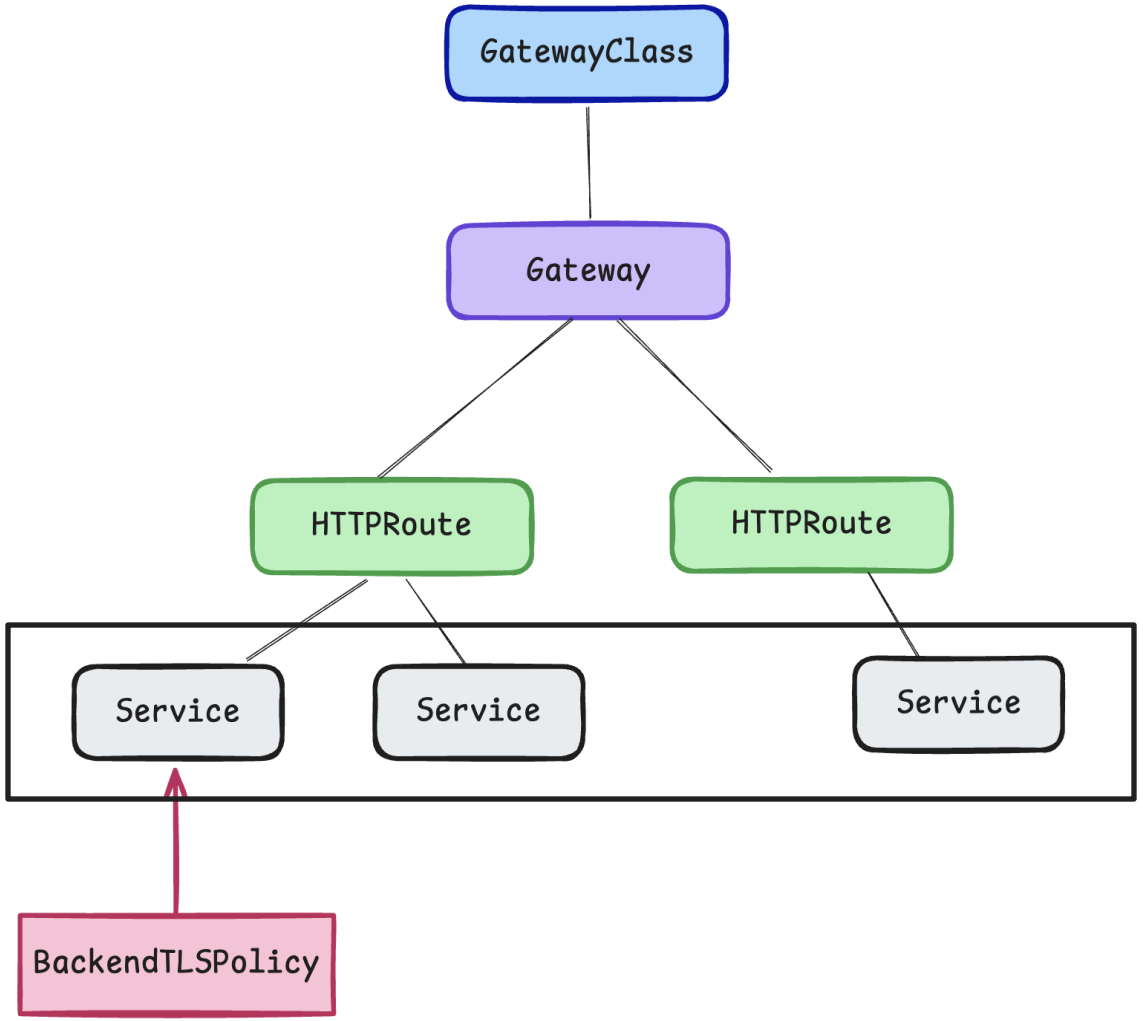
BackendTLSPolicy



BackendTLSPolicy Continued


```yaml
apiVersion: gateway.networking.k8s.io/v1alpha3
kind: BackendTLSPolicy
metadata:
  name: secure-foo-service
spec:
  targetRefs:
  - kind: Service
    name: foo
  validation:
    caCertificateRefs:
    - name: foo-cert
      kind: ConfigMap
    hostname: foo.example.com
```

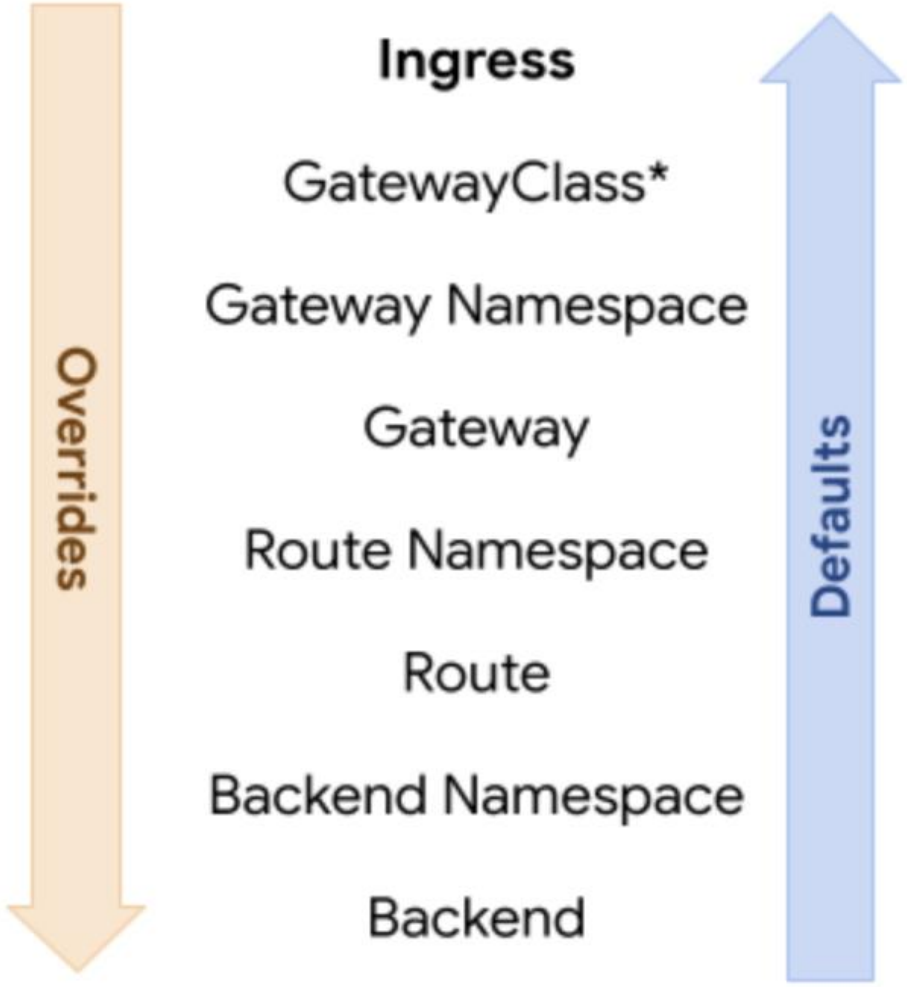
Why is BackendTLSPolicy Direct?



- Only affects the Service it targets
- Tightly bound to the Kind Service
- Attaches to a single layer in hierarchy

Inherited Policies

Hierarchy for Inherited Policies



Default values are given precedence from the **bottom-up**.

Override values are **top-down**.

The **default** attached to a **Backend** will have the **highest** precedence among default values

The **override** value attached to a **GatewayClass** will have the **highest** precedence **overall.**

NGINX Gateway Fabric's First Policy

The Issue

Unable to use nginx gateway fabric in front of docker registry service (need to set client_max_body_size) area/nginx-configuration enhancement

A user wants to be able to set the `client_max_body_size` for their application.

The User Stories

1

As a Cluster Operator

I want to set ***sane defaults*** for client body settings that will work for most applications.

2

As an Application Developer

I want to be able to ***configure*** the client body settings for my application based on its behavior or requirements.

3

As an Application Developer

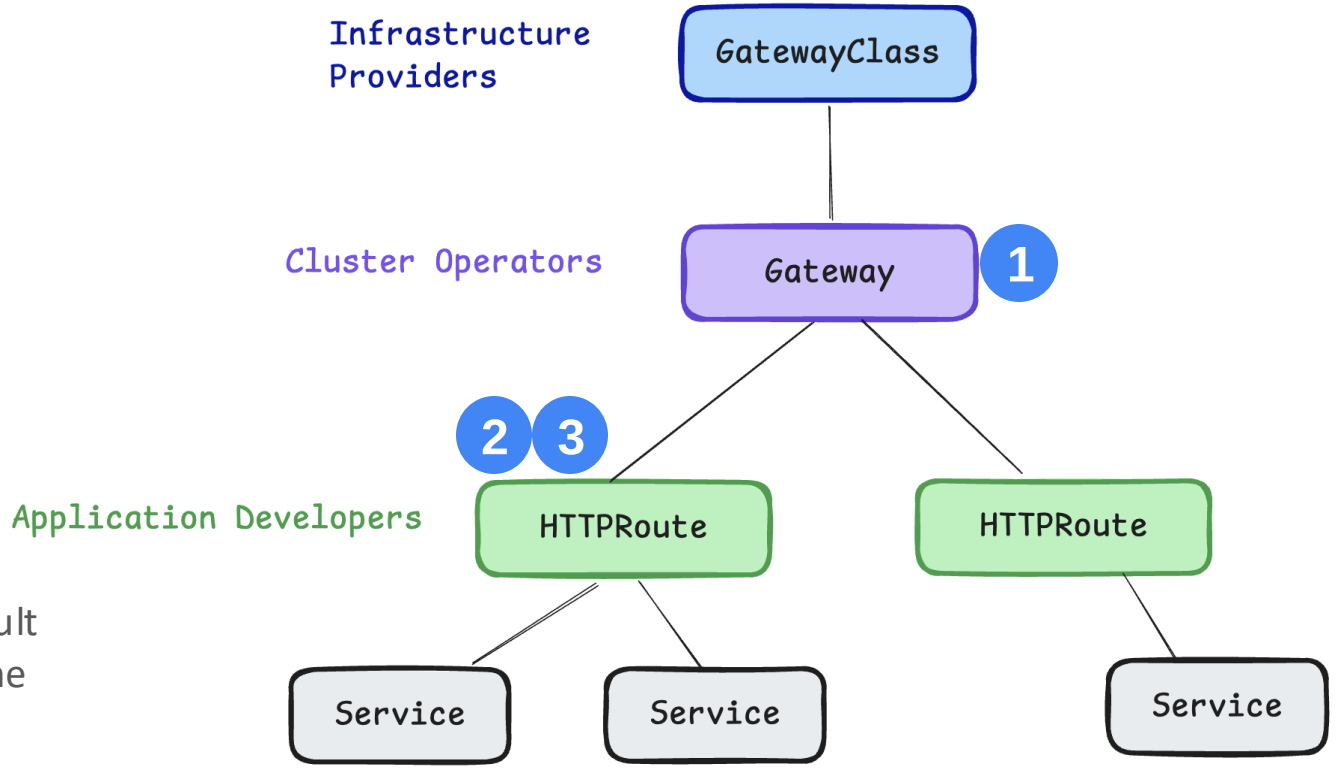
I want to ***fine-tune*** the default client body settings set by the Cluster Operator because the defaults do not satisfy my application's requirements.

Direct or Inherited?

1. **As a Cluster Operator**, I want to set ***sane defaults*** for client body settings that will work for most applications.

2. **As an Application Developer**, I want to be able to ***configure*** the client body settings for my application based on its behavior or requirements.

3. **As an Application Developer**, I want to ***fine-tune*** the default client body settings set by the Cluster Operator because the defaults do not satisfy my application's requirements.



Does the Policy affect *any* other object aside from the one it targets? *It depends...*

If a Policy can be used as an Inherited Policy, it MUST be treated as an Inherited Policy.

Writing the Policy

MUST be a CRD

MAY be included in the Gateway API group or be defined by implementations

MUST be clearly named to indicate that they are Policy metaresources

MUST include a label on the CRD that specifies it is an Inherited Policy.

MUST include both `spec` and `status` stanzas

```yaml
apiVersion: gateway.nginx.org/v1alpha1
kind: ClientSettingsPolicy
metadata:
  name: example-client-settings
  namespace: default
  labels:
    gateway.networking.k8s.io/policy: inherited

spec:

status:
```

Writing the Policy Spec

MUST include a `TargetRef` struct in the `spec`

MAY specify a `default` stanza, an `override` stanza, or both

```yaml
default:
  body:
    maxSize: 10m
    timeout: 30s
```

```yaml
body:
  maxSize: 10m
  timeout: 30s
```

```yaml
spec:
  targetRef:
    group: gateway.networking.k8s.io
    kind: HTTPRoute|Gateway
    name: resource-name
  body:
    maxSize: 10m
    timeout: 30s
```

Writing the Policy Status

SHOULD use the upstream `PolicyAncestorStatus` struct in the `status` stanza

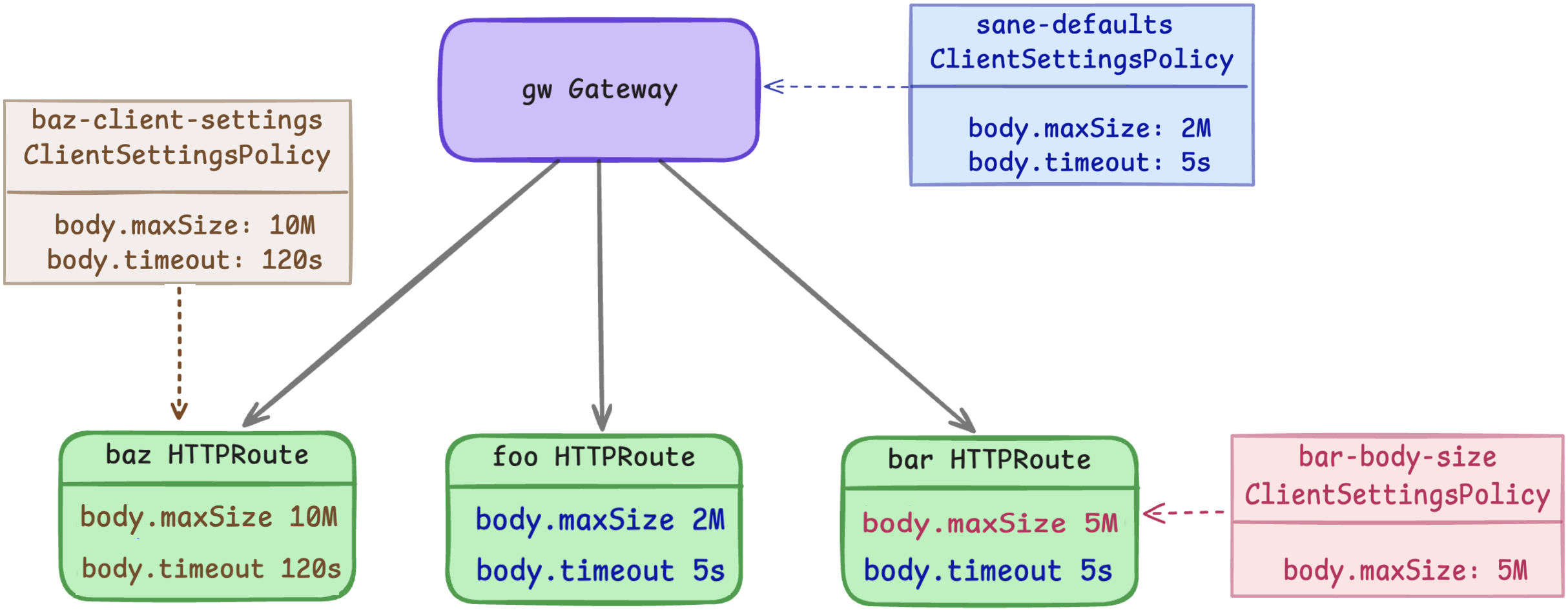
MUST have the `status` stanza include a conditions section using upstream `Condition` type

```yaml
status:
  ancestors:
  - ancestorRef:
      group: gateway.networking.k8s.io
      kind: HTTPRoute|Gateway
      name: resource-name
    controllerName : my-controller
    conditions:
    - type: Accepted
      status: "True"
      reason: Accepted
      message: Policy is accepted
```

Putting in All Together


```yaml
apiVersion: gateway.nginx.org/v1alpha1
kind: ClientSettingsPolicy
metadata:
  name: example-client-settings
  namespace: default
spec:
  targetRef:
    group: gateway.networking.k8s.io
    kind: Gateway
    name: example-gateway
  body:
    maxSize: 10m
    timeout: 30s
status:
  ancestors:
    ancestorRef:
      group: gateway.networking.k8s.io
      kind: Gateway
      name: example-gateway
    controllerName: my-controller
    conditions:
      - type: Accepted
        status: "True"
        reason: Accepted
        message: Policy is accepted
```

ClientSettingsPolicy in Practice



Revisiting the Case for More Extensibility

Cluster Operators

*"We want to **mandate** security policies for all applications."*

*"We want to set **sane defaults** for all applications."*

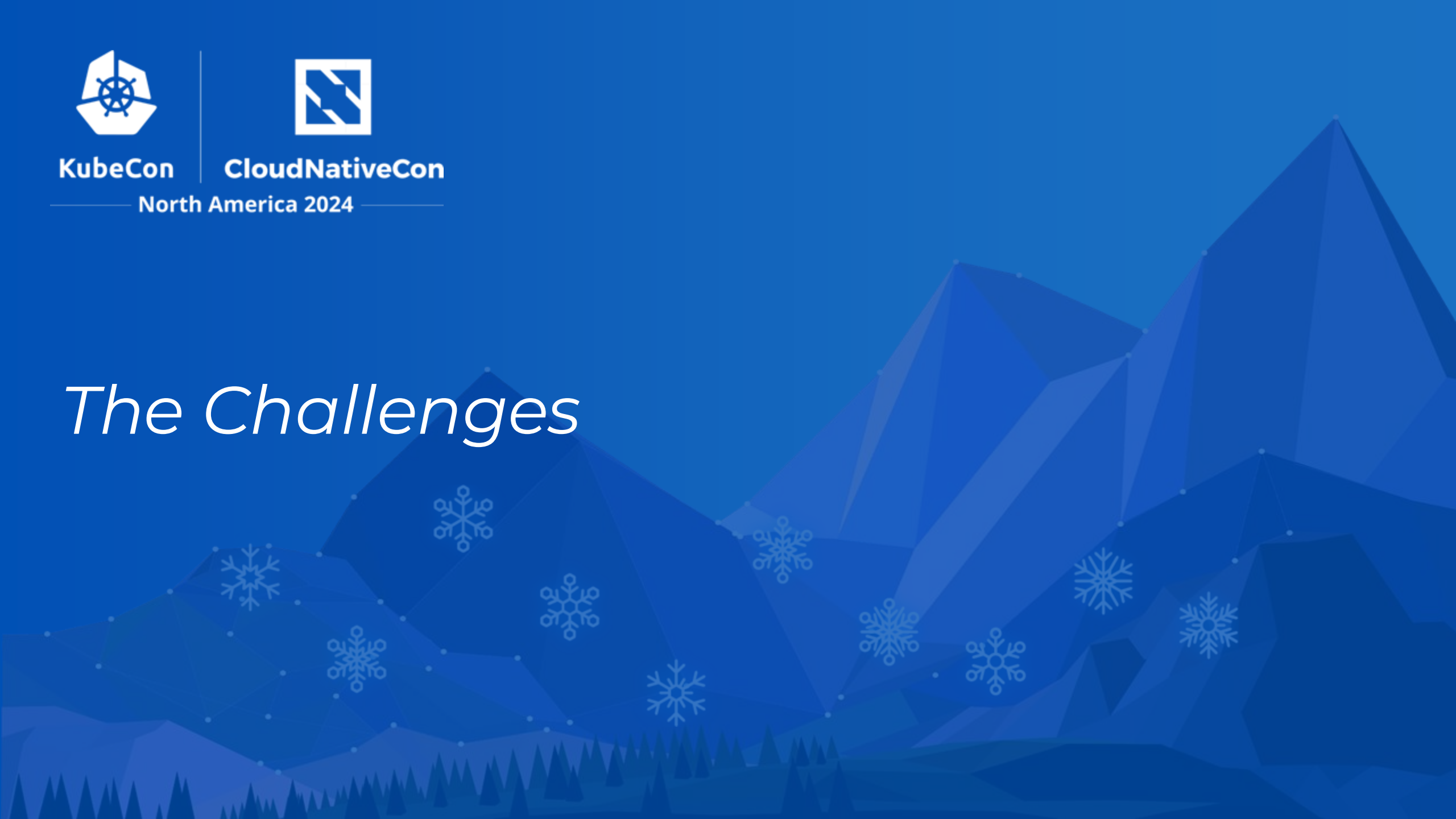
Application Developers

*"We want to **fine-tune** settings based on our application's behavior."*

Any

*"We want to **change** the behavior of **Services** or **Namespaces** without changing their specs."*

How's the user experience?



The Challenges

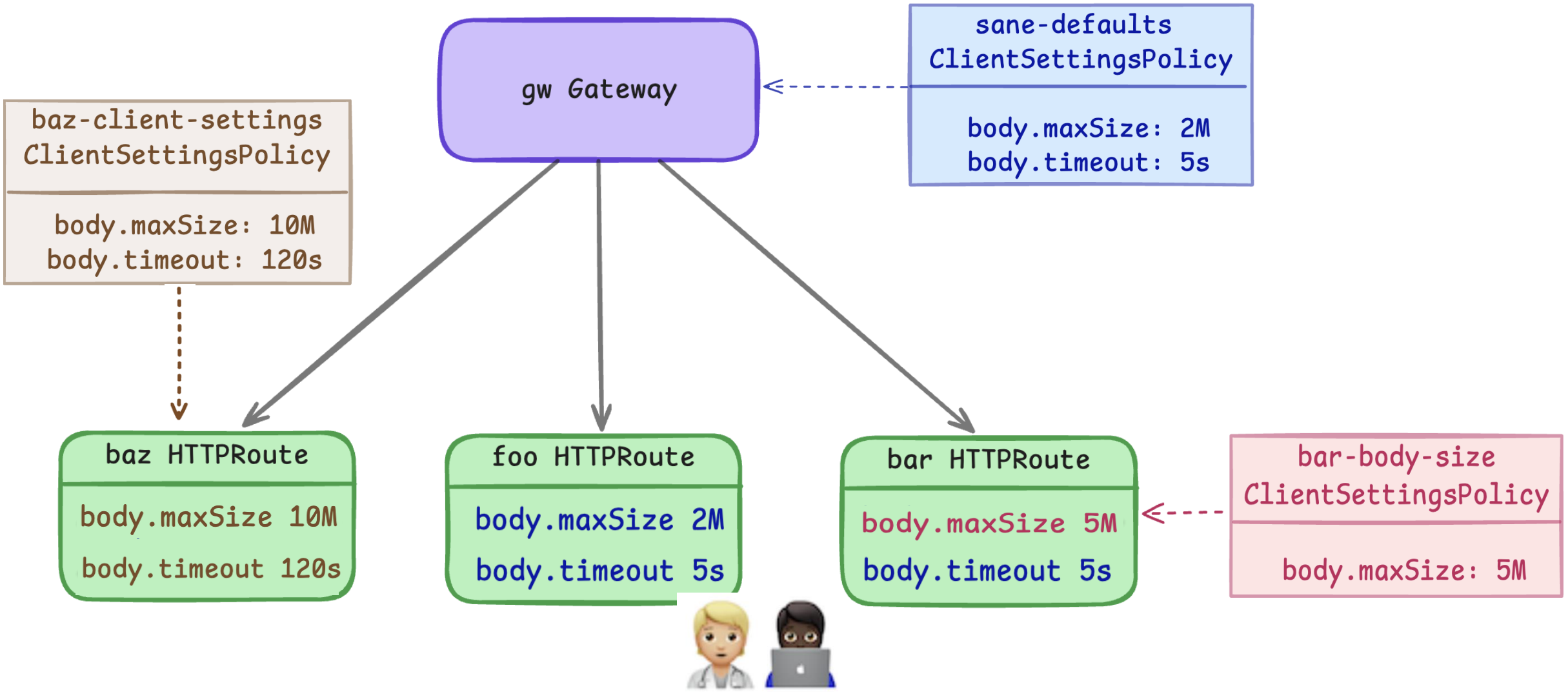
It’s complex

Before and After


```
nginx.org/client-max-body-size: 2M
```

```yaml
apiVersion: gateway.nginx.org/v1alpha1
kind: ClientSettingsPolicy
metadata:
  name: gateway-client-settings
spec:
  targetRef:
    group: gateway.networking.k8s.io
    kind: Gateway
    name: gateway
  body:
    maxSize: "2M"
```

The Discoverability Problem



Troubleshooting

gwctl

- Command-line tool for Gateway API
- Can show how policies impact your resources
- **With Great Flexibility Comes Great Complexity – Friday @ 4:55pm, 155 E**

status

- `kubectl describe` is your friend
- No status? Check the ref

docs

- Knowledge is power

Portability Concerns

Is this better than annotations?

How many CRDs is too many?

What's Next for Policy?

- Open GitHub discussion
- Improved discoverability
- More policies?
- Policy Machinery by Kuadrant
- You tell us...

Want more of the Gateway API?

Check out these talks at KubeCon:

How to Move from Ingress to Gateway API with Minimal Hassle Thurs 11:55am, 155 E

Tutorial: Live with Gateway API V1.2 - Thurs 2:30pm, Grand Ballroom G

Gateway API: What's New, What's Next? - Thurs 4:30pm, Regency Ballroom A

Tutorial: No Mess Rollouts with Gateway API – Thurs 4:30pm, Grand Ballroom G

One Gateway API to Rule Them All (and in the Cluster Configure Them) - Thurs 5:25pm, 155 E

With Great Flexibility Comes Great Complexity – Fri 4:55pm, 155 E

The Gateway API wants your feedback!

This Gateway API survey aims to understand how widely Gateway API is used, identify user needs and pain points, and gather valuable feedback from the community. Whether you're a seasoned user or just starting out, your input is crucial.


KubeCon
CloudNativeCon
North America 2024
Thank you!

Resources and Links

- Gateway API documentation
- Policy & Metaresources GEP
- Inherited Policy GEP
- Direct Policy GEP
- Discussion on Policy
- Gateway API Repo
- NGINX Gateway Fabric Repo
- Kuadrant's Policy Machinery