KubeCon | CloudNativeCon

North America 2024

Gateway API

What's New, What's Next?

Who are we?

Nick Young
Senior Software Engineer
Isovalent at Cisco
Gateway API maintainer
@youngnick

Guilherme Cassolato
Principal Software
Engineer at Red Hat
@guicassolato

Christine Kim
OSS Dev Experience
Isovalent at Cisco
@xtineskim

Mattia Lavacca
Software Engineer at Kong
Gateway API maintainer
@mlavacca

Agenda

Agenda

- What’s new
- Policies
- User Experience improvements
- What’s next

What’s New

New release cycle and process

For all releases v1.2+ (which includes the 1.3, the current release)

Added release phases:

- Scoping (4-6 weeks)
- GEP iteration and Review (5-7 weeks)
- API Refinement and Documentation (3-5 weeks)
- API Review and Release candidates (2-4 weeks)

New release cycle and process

Scoping

- Determine what features will be included in the release. After this phase, no features are added, but features can fall out of scope
- For a feature to go to Experimental, something must graduate to Standard
- We are in this phase right now for 1.3
- The output of this phase is a list of GEPs, and a target status for that GEP for the release
- Experimental GEPs should generally be Implementable or Experimental, Standard should be Standard or Completed

New release cycle and process

GEP Iteration and review

- We've picked the GEPs to work on, now we work on them
- Experimental GEPs must get to Implementable by the end of this phase
- Standard GEPs must have a TODO list for all remaining work by the end of this phase

New release cycle and process

API Refinement and Documentation

- Implement any required API changes and document everything
- Conformance testing must be included, particularly for GEPs moving to Standard

New release cycle and process

API review and Release Candidates

- As an official `k8s.io` API, Gateway API must be reviewed by designated Kubernetes API reviewers
- API reviewers check that the API is in line with the API conventions, and that the design meets the required quality bar
- Release Candidates are issued once the community is confident that the release is close, and allow implementations to do implementation and early conformance testing

New release cycle and process

Contributions welcome in each phase

	1. Scope	2. GEP	3. API	4. Review
New GEPs	✅	❌	❌	❌
Major GEP Updates	✅	✅	❌	❌
GEP Refinement	✅	✅	✅	❌
API Spec Additions	❌	❌	✅	❌
New Conformance Tests	✅	✅	✅	❌
Bug Fixes	✅	✅	✅	✅
Documentation	✅	✅	✅	✅
Review	✅	✅	✅	✅

Release 1.2 Changes

V1.2 breaking changes

- `GRPCRoute` and `ReferenceGrant` `v1alpha2` removed
  - This affects implementations that are still using the `v1alpha2` version of the resource.
- The experimental `supportedFeatures` field in GatewayClass `status` has changed from being a list of strings to being a list of objects with a `name` field.

V1.2 graduations to Stable

- `Infrastructure` labels and annotations - now Extended in Standard
  - Feature name: `GatewayInfrastructurePropagation`
- HTTPRoute Timeouts and Durations - now Extended in Standard
  - Feature name: `HTTPRouteRequestTimeout`
  - Feature name: `HTTPRouteBackendTimeout`
- BackendProtocol Support - now Extended in Standard
  - Feature name: `HTTPRouteBackendProtocolH2C`
  - Feature name: `HTTPRouteBackendProtocolWebSocket`

V1.2 new experimental features

All of these features are Extended support in the Experimental channel

- HTTPRoute Retry support
  - No feature name or conformance tests as yet
- Percentage-based request mirroring
  - No feature name or conformance tests as yet
- Backend TLS configuration improvements
  - Ability to specify SANs on BackendTLSPolicy
  - Add TLS options to BackendTLSPolicy to mirror TLS Config on Gateways
  - No feature name or conformance tests as yet

Leadership changes

We've also had some leadership changes! Congratulations to everyone here!

Mattia Lavacca
@mlavacca
Promoted to Gateway API maintainer

Mike Morris
@mikemorris
Promoted to GAMMA Lead

Flynn
@kflynn
Promoted to GEP Reviewer

Arko Dasgupta
@arkodg
Promoted to GEP Reviewer

Lior Lieberman
@LiorLieberman
Promoted to Conformance Approver

Candace Holman
@candita
Promoted to Conformance Reviewer

Guilherme Cassolato
@guicassolato
Promoted to `gwctl` Reviewer

Policies

Policies Recap

- Additional spec for Gateway API-related resources “from the outside”
- Based on *Metaresources* (GEP-713)
  - *“Objects that augment the behavior of other objects in a standard way.”*
- “Policies” → *rules* with *context*
  - *Context*: network object that the policy *attaches to* (`targetRefs`)
  - *Rules*: the policy spec proper (e.g. traffic validation rules, mutation rules, etc)
- Applications:
  - Gateway API-related capabilities for stable APIs (e.g. Service)
  - Implementation-specific features (e.g. different flavors of auth)
  - Decoupling of concerns (i.e. personas, delegation, RBAC)
  - 3rd-party extensions (non-Gateway API implementations)

Current state of Policies (as of v1.2)

- 2 classes of policies (since v1.1)
  - Direct Policies (GEP-2648) – experimental
  - Inherited Policies (GEP-2949, aka: defaults and overrides) – provisional
- Common policies specified by Gateway API (target all implementations)
  - BackendTLSPolicy – experimental since v1.0
    - Envoy Gateway
    - Istio
    - NGINX Gateway Fabric
  - BackendLBPolicy – provisional in v0.8, experimental since v1.1

Current state of Policies (as of v1.2)

- Implementation-specific policies (non-extensive list)





Envoy Gateway	Istio	NGINX Gateway Fabric	Gloo Gateway	Kuadrant(*)
ClientTrafficPolicy	EnvoyFilter	ClientSettingsPolicy	ListenerOption	DNSPolicy
BackendTrafficPolicy	RequestAuthentication	ObservabilityPolicy	HTTPListenerOption	TLSpolicy
EnvoyExtensionPolicy	AuthorizationPolicy		RouteOption	AuthPolicy
EnvoyPatchPolicy	WasmPlugin		VirtualHostOption	RateLimitPolicy
SecurityPolicy	Telemetry			

What new about Policies in v1.2

- Multiple targets (`targetRef` → `targetRefs`)
- `targetRefs.sectionName` extended to include HTTPRoute and GRPCRoute sections (GEP-995)
- New fields added to BackendTLSPolicy:
  - `subjectAltNames`
  - `options`

Main challenges of Policies

- Discoverability problem
- Visualization of Effective Policies and the Combinatorial explosion problem

kubernetes-sigs/gwctl

kuadrant/policy-machinery

What to expect about Policies in v1.3+

- BackendTLSPolicy conformance tests and graduation to standard
- API extensions currently under discussion (possibly evolving to new kinds of policies):
  - Timeout (GEP-1742)
  - Circuit breaking (GEP-3358)
  - Auth (GEP-1494)
  - CORS (GEP-1767)
  - DNS (GEP-2627)
  - Retry Budgets (#3403)

What to expect about Policies in v1.3+ (cont.)

- Possible new definitions for Direct and Inherited policies emerging from a generalized concept of ‘merge strategies’
- More dynamic ways to target resources using label selectors
- Narrowing the scope of targets by specifying specific GatewayClass names (#3175)
- Enhancements to policy status
  - Status conditions for policies (#738)
  - PolicyAncestor status type (#2923)

User Experience Updates

Tooling

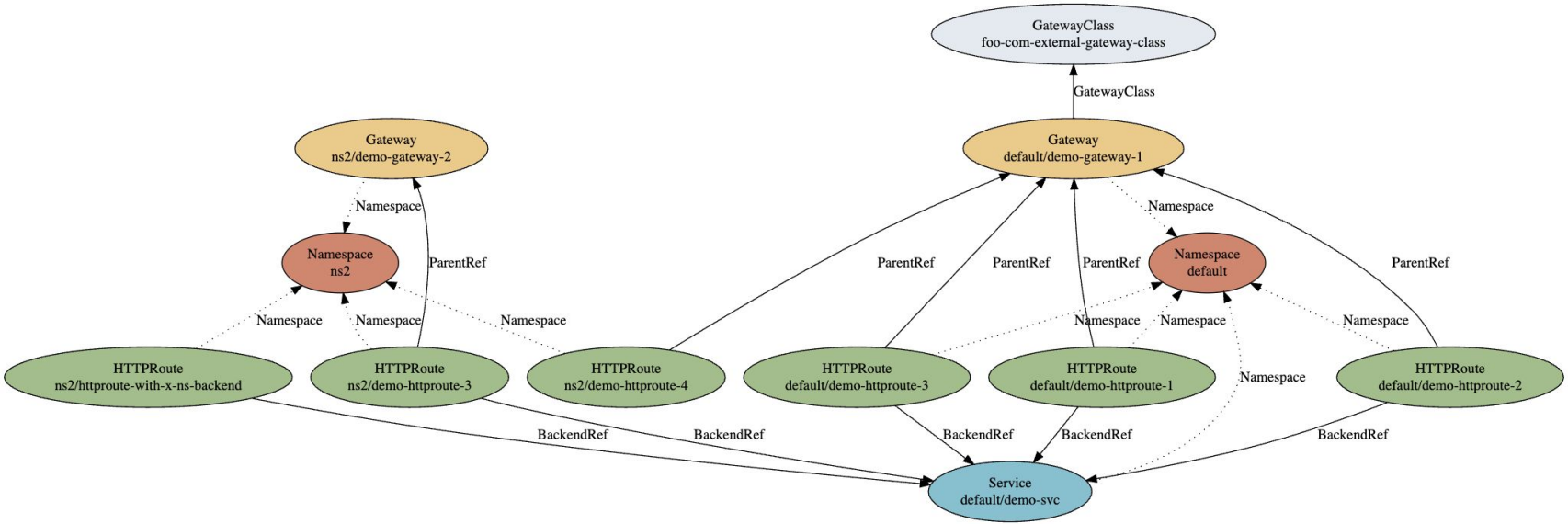
gwctl

- Explore Gateway API resources
- github.com/kubernetes-sigs/gwctl

```
NAME                          KIND                                          TARGET NAME       TARGET KIND  POLICY TYPE  AGE
tls-upstream-dev              BackendTLSPolicy.gateway.networking.k8s.io                                     Direct       6h22m
demo-health-check-1           HealthCheckPolicy.foo.com                     demo-gateway-1    Gateway      Direct       6h23m
demo-retry-policy-1           RetryOnPolicy.foo.com                         demo-gateway-1    Gateway      Direct       6h23m
demo-retry-policy-2           RetryOnPolicy.foo.com
demo-tls-min-version-policy-1 TLSMinimumVersionPolicy.baz.com               demo-httproute-1  HTTPRoute    Direct       6h23m
demo-tls-min-version-policy-3 TLSMinimumVersionPolicy.baz.com               demo-svc          Service      Direct       6h23m
```

`gwctl`

- Explore Gateway API resources
- github.com/kubernetes-sigs/gwctl

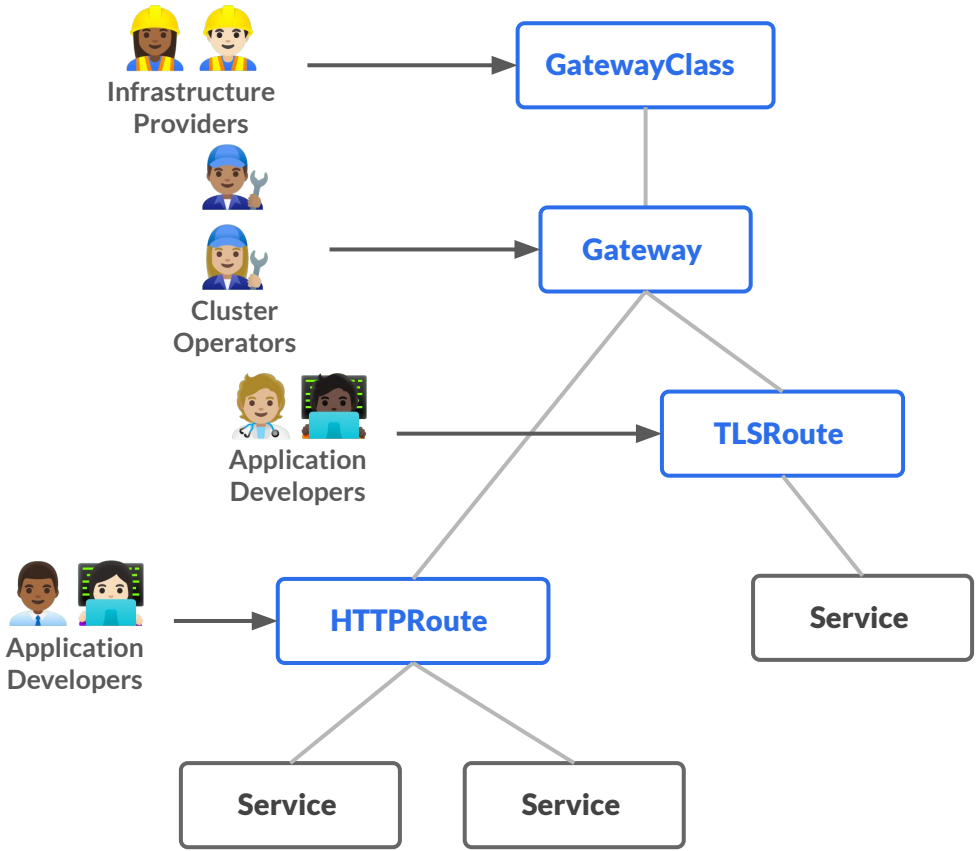


ingress2gateway

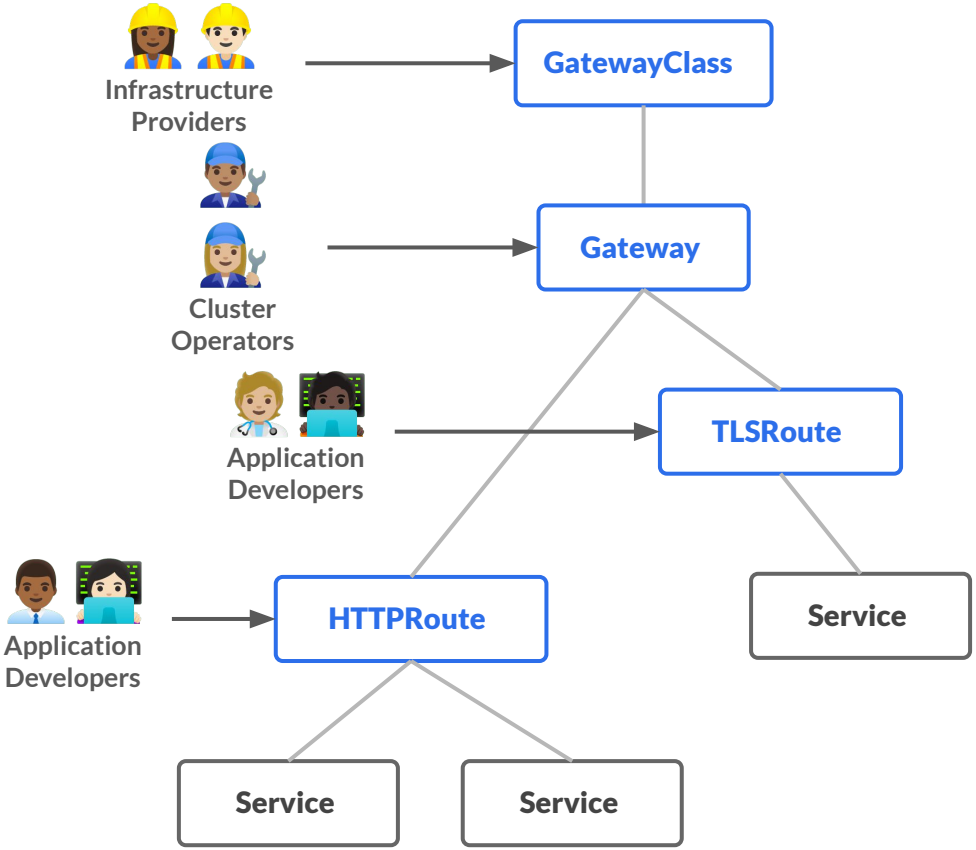
- Translate Ingress and provider specific resources to Gateway API resources
- github.com/kubernetes-sigs/ingress2gateway


```
ingress2gateway print --providers <your-provider-here>
```

Personas


Infrastructure Providers
GatewayClass
Gateway
Cluster Operators
Application Developers
TLSRoute
Application Developers
HTTPRoute
Service
Service
Service

Personas

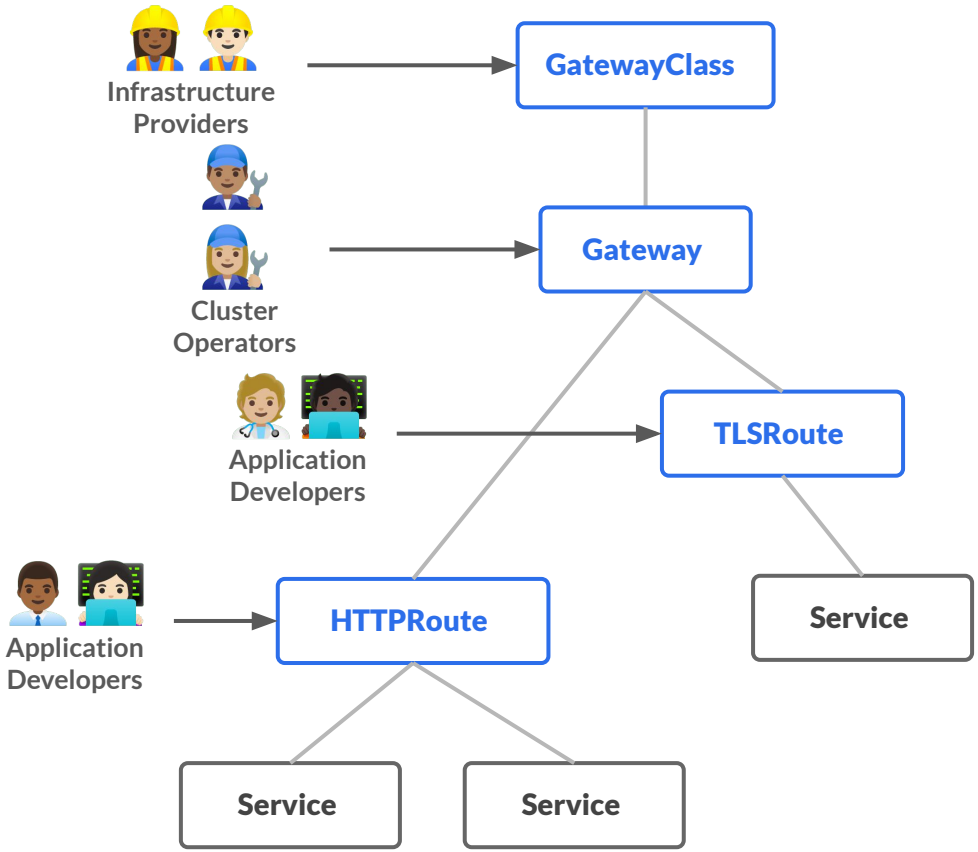


Ian
Infrastructure Providers
(he/him)

Chihiro
Cluster Operators
(they/them)

Ana
Application Developers
(she/her)

Personas



Ian
Infrastructure Providers
(he/him)

Cares about infrastructure

Chihiro
Cluster Operators
(they/them)

Cares about policies, network access, app permissions

Ana
Application Developers
(she/her)

Cares about serving her app

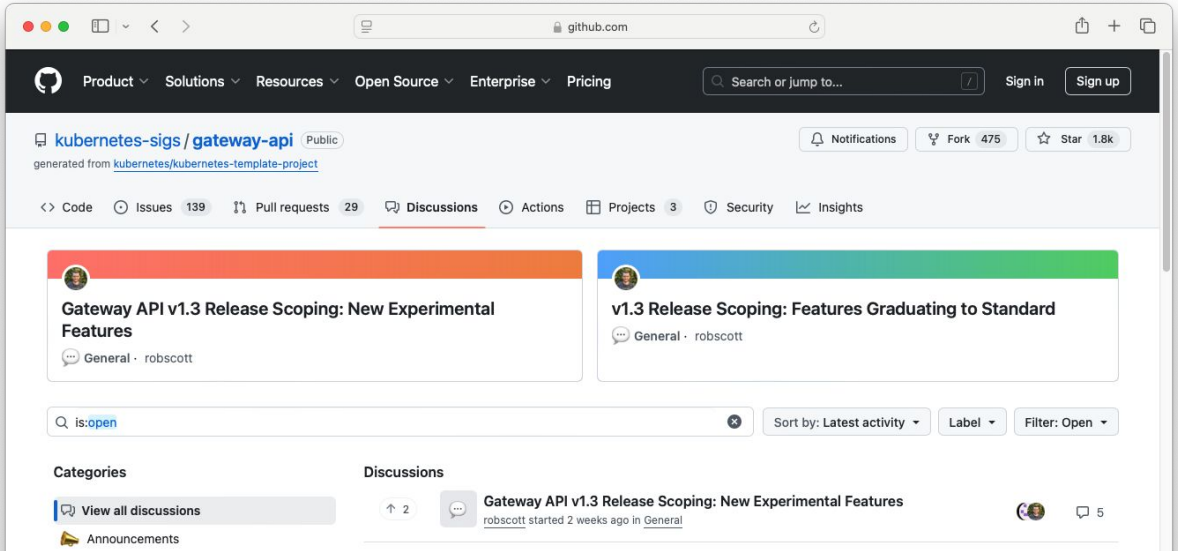
User Experience

- Converting to Gateway API?
- Pain points!
- How do you use Gateway API?

Gateway API survey

User Experience

- Converting to Gateway API?
- Pain points!
- How do you use Gateway API?



Gateway API survey

What’s Next?

We are in the scoping phase

- New features for the experimental channel
  - Retry Budgets
  - Auth* (AuthN / AuthZ)
  - Ready condition on routes
  - GRPC retry
  - CORS filter
  - HTTP Cookie match
  - Query Parameter Filter
  - ListenerSet / Listener Merging

Gateway-api#3403

We are in the scoping phase

- Features graduating to the standard channel
  - Percentage-Based Request Mirroring
  - `BackendTLSPolicy`

Gateway-api#3404

CRD lifecycle management

- 1st CRD-based Kubernetes official API
- No Feature Gates for CRDs
- 2 different channels

```
// Timeouts defines the timeouts that can be configured for an HTTP request.
//
// Support: Extended
//
// +optional
Timeouts *HTTPRouteTimeouts `json:"timeouts,omitempty"`

// Retry defines the configuration for when to retry an HTTP request.
//
// Support: Extended
//
// +optional
// <gateway:experimental>
Retry *HTTPRouteRetry `json:"retry,omitempty"`
```

CRD lifecycle management (cont.)

Standard HTTPRoute

```yaml
apiVersion:
gateway.networking.k8s.io/v1
kind: HTTPRoute
metadata:
  name: example-route
spec:
  parentRefs:
  - name: example-gateway
  rules:
  - timeouts:
      request: 3s
    backendRefs:
    - name: backend
      port: 8080
```

Experimental HTTPRoute

```yaml
apiVersion: gateway.networking.k8s.io/v1
kind: HTTPRoute
metadata:
  name: example-route
spec:
  parentRefs:
  - name: example-gateway
  rules:
  - timeouts:
      request: 3s
    retry:
      codes:
      - 500
    backendRefs:
    - name: backend
      port: 8080
```

CRD lifecycle management (cont.)

- Complexities
  - Updating the CRDs version
  - Changing channel (experimental->standard)

Gateway-api#2655

CRD lifecycle management - possible solutions

- Introduction of separate groups

- Single channel with the introduction of a `ValidatingAdmissionPolicy` based on FG
  - Potential Long Term solution

LLM instance Gateway

- Maintained by the wg-serving
- Proposes to define an inference Gateway
- Based on Envoy
- Based on an extension to target LLM-specific backends

llm-instance-gateway

Thank you and Questions

Gateway API survey

#sig-network-gateway-api

github.com/kubernetes-sigs/gateway-api