



KubeCon



CloudNativeCon

Europe 2026

# Testing the Waters: Getting Started with kgateway

Mayowa Fajobi, Steven Thwaites, David Jumani, Nina Polshakova



# Introductions



KubeCon



CloudNativeCon

Europe 2026



**Nina Polshakova**

Solo.io



**David Jumani**

Solo.io



**Mayowa Fajobi**

Marsh



**Steven Thwaites**

Solo.io

# Environment Setup



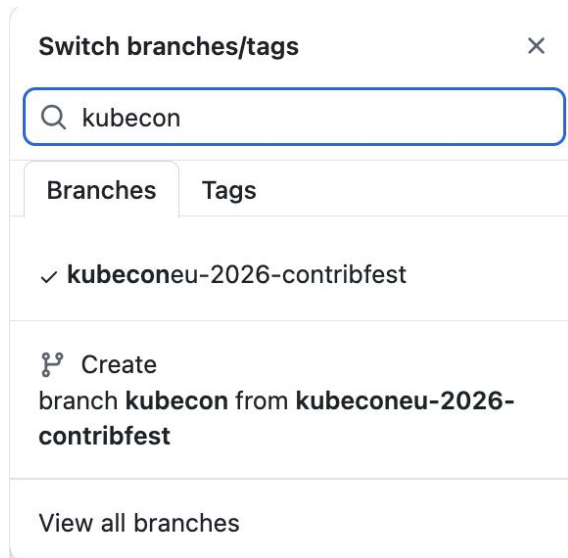
KubeCon



CloudNativeCon

Europe 2026

<https://github.com/kgateway-dev/kgateway/tree/kubecone-2026-contribfest>



# Forking



KubeCon



CloudNativeCon

Europe 2026

- kgateway uses a “fork and PR” model for code contributions, creating your fork is easy though.
- Navigate to <https://github.com/kgateway-dev/kgateway/> in your browser.



Click the drop down arrow in the Fork element located near the top right corner, then select “Create new fork” from the context menu

# What Development Environment Should I Use?



KubeCon



CloudNativeCon

Europe 2026

- kgateway contributors use many different DEs successfully
  - VSCode
  - GoLand
  - Cursor
  - (neo)vim
  - Emacs
- For the purposes of ContribFest, **Github's Codespaces is highly recommended**
  - DevContainers are another strong option, however the image is quite large and pulling it over conference wifi is unlikely to work well

# Launch your Codespace



KubeCon



CloudNativeCon

Europe 2026

From your fork, click the Code drop down, then switch to the Codespaces tab  
Click the “...” (breadcrumbs) and select “New with options..” from the menu

The screenshot shows the GitHub repository page for 'kgateway.dev'. The 'Code' button is highlighted in green. Below it, the 'Codespaces' tab is selected. A dropdown menu is open, showing options: 'New with options...', 'Configure dev container', 'Set up prebuilds', 'Manage codespaces', 'Share a deep link', and 'What are codespaces?'. The 'New with options...' option is highlighted. The main content area shows 'No codespaces' and a 'Create codespace on main' button. The footer indicates 'Codespace usage for this repository is paid for by npolshakova.'

The screenshot shows the 'Create codespace for npolshakova/kgateway' dialog. It has four sections: 'Branch' (set to 'main'), 'Region' (set to 'US East'), 'Machine type' (set to '8-core'), and a 'Create codespace' button. Below the dialog, a list of instance types is shown: '2 cores, 8 GB RAM, 32 GB storage', '4 cores, 16 GB RAM, 32 GB storage', '8 cores, 32 GB RAM, 64 GB storage' (highlighted), and '16 cores, 64 GB RAM, 128 GB storage'.

# Dev Containers



KubeCon



CloudNativeCon

Europe 2026

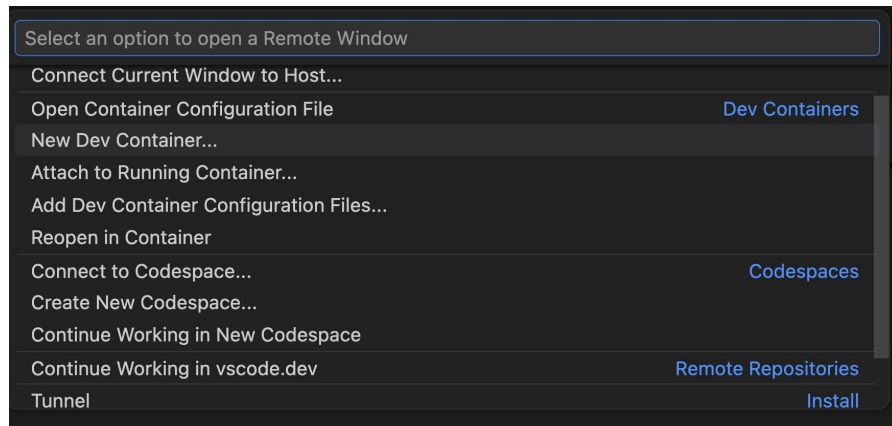
## Step 1:

```
git clone
```

<https://github.com/<fork>/kgateway.git>

## Step 2: Checkout branch **kubeconeu-2026-contribfest**

## Step 3: Open in VSCode



# Agenda



KubeCon



CloudNativeCon

Europe 2026

- ✓ Basic Environment Setup
- ☐ Kgateway Background + Why YOU Should Contribute!
- ☐ Kgateway Architecture Overview
- ☐ Kgateway Code Walkthrough
- ☐ Testing in kgateway
  - ☐ Conformance
  - ☐ Unit
  - ☐ Benchmarks
  - ☐ E2E
  - ☐ Manual Testing
- ☐ How to Join the Kgateway Community



KubeCon



CloudNativeCon

Europe 2026

# Background on kgateway



# What is a Gateway?



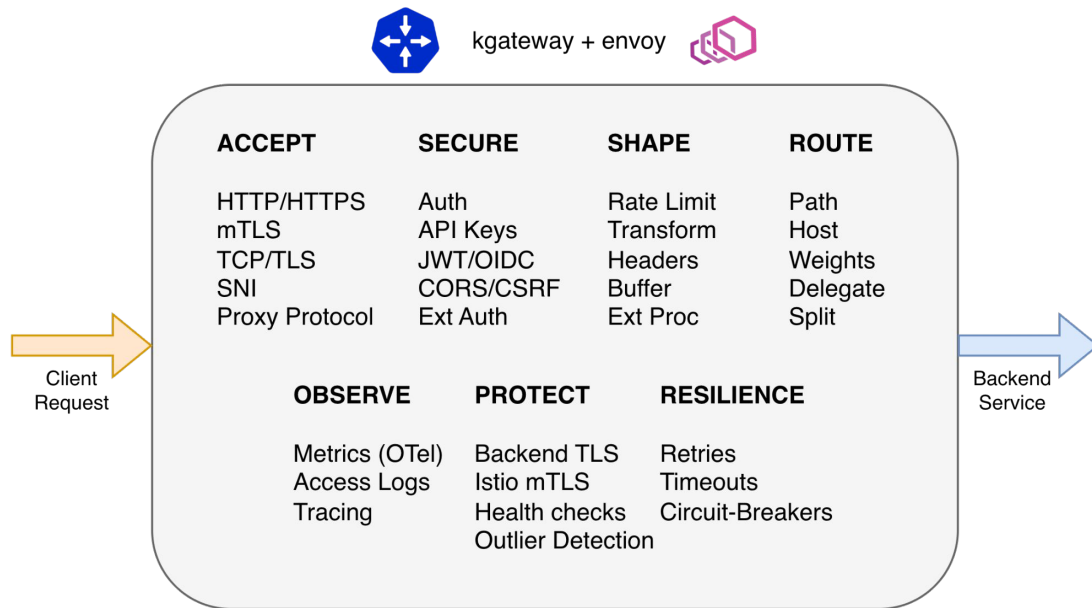
KubeCon



CloudNativeCon

Europe 2026

A gateway is a centralized server that sits between clients and backend services, routing, securing, and managing API requests.



<https://landscape.cncf.io/>

# How is a Gateway Deployed?



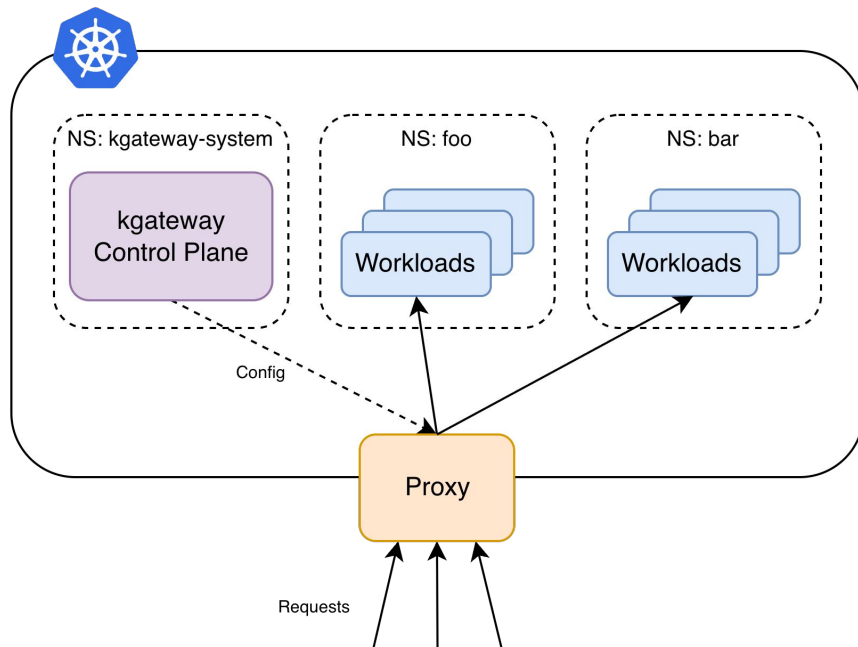
KubeCon



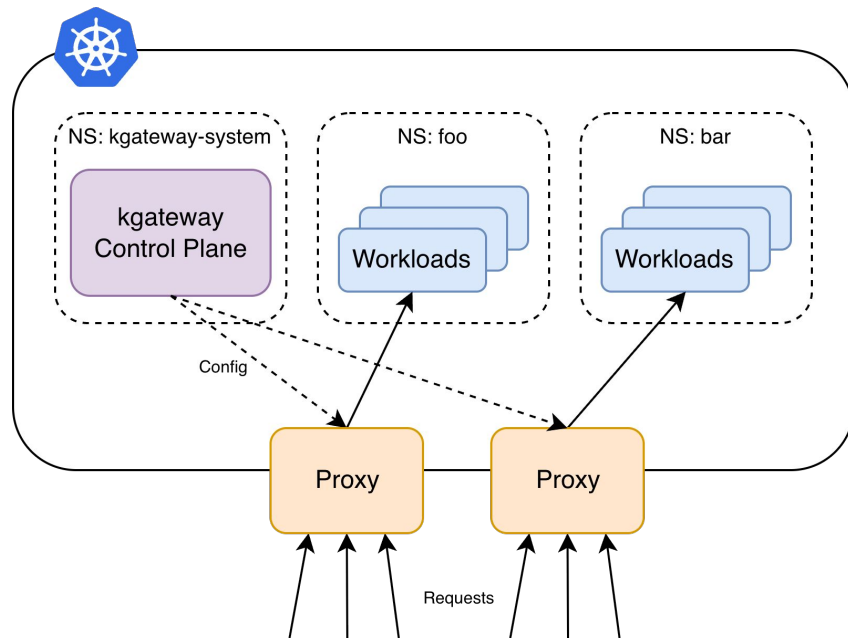
CloudNativeCon

Europe 2026

## Single Proxy



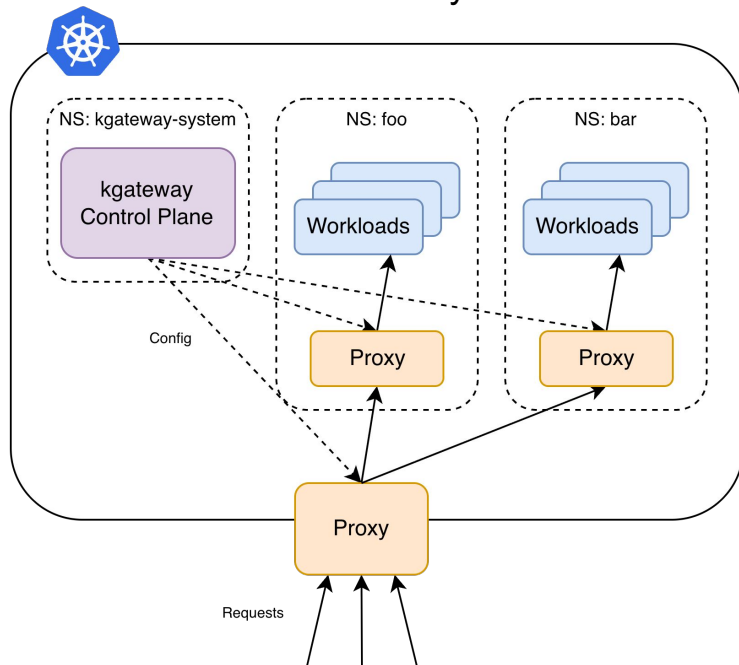
## Sharded Proxy



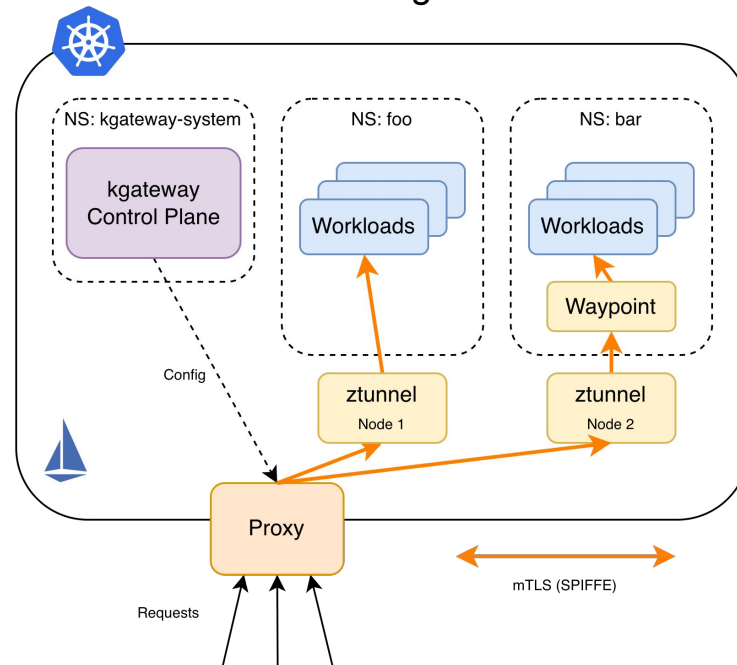
<https://kgateway.dev/docs/envoy/latest/about/deployment-patterns/>

# How is a Gateway Deployed?

## Tiered Proxy



## Mesh Integrated



<https://kgateway.dev/docs/envoy/latest/about/deployment-patterns/>

# kgateway: A History

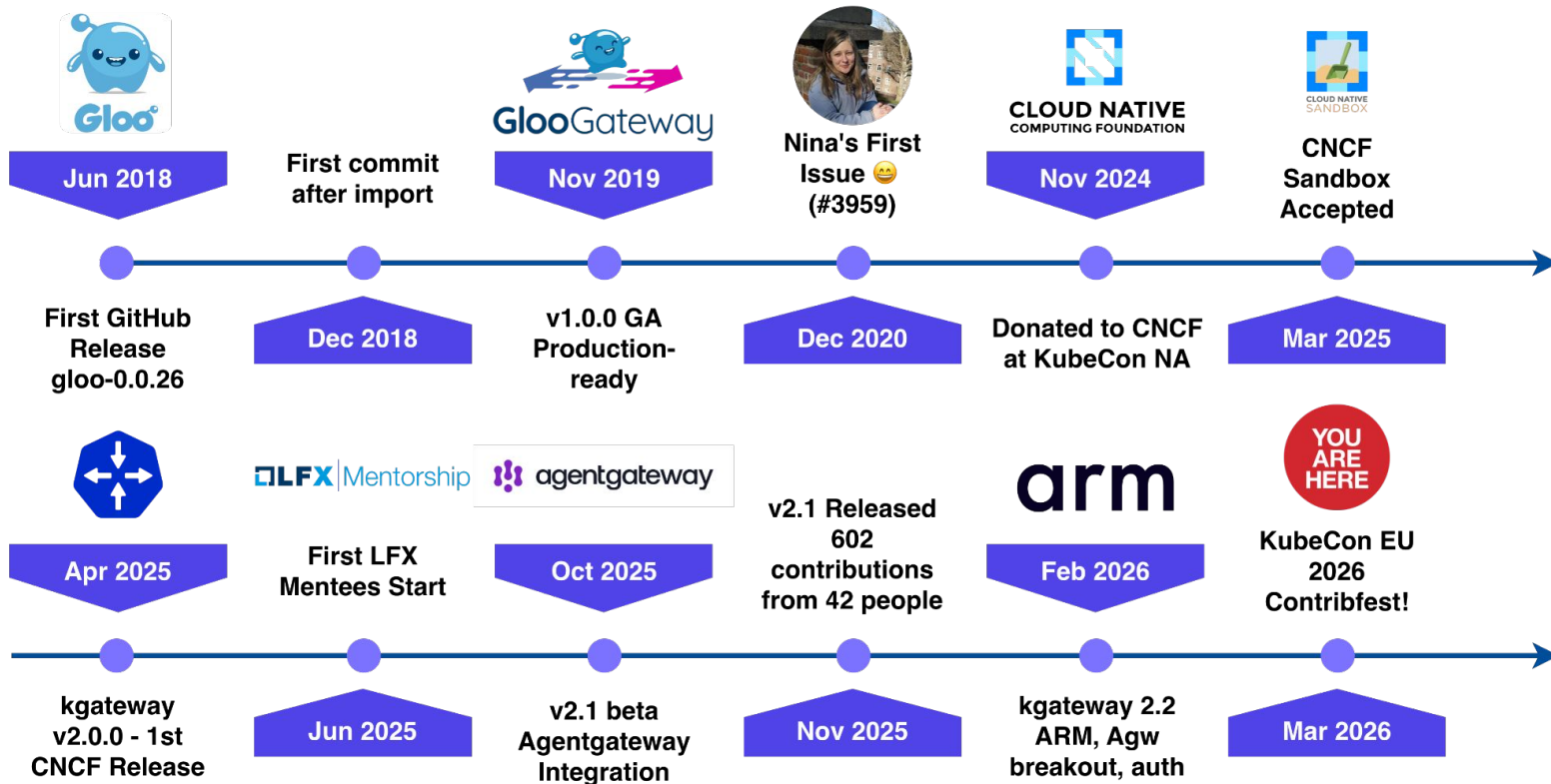


KubeCon



CloudNativeCon

Europe 2026





KubeCon



CloudNativeCon

Europe 2026

# Architecture Overview



# Kgateway Components

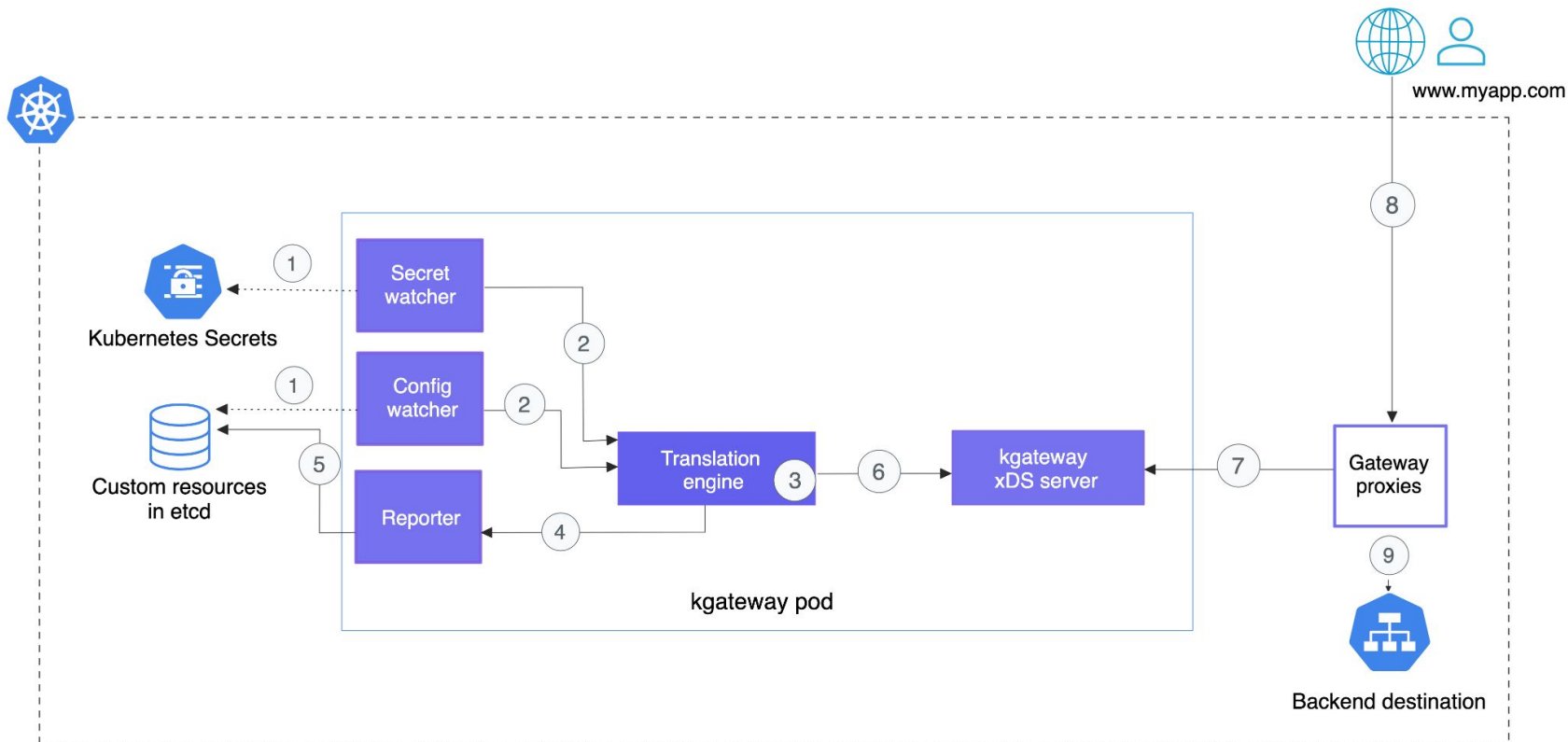


KubeCon



CloudNativeCon

Europe 2026



# How does kgateway work?

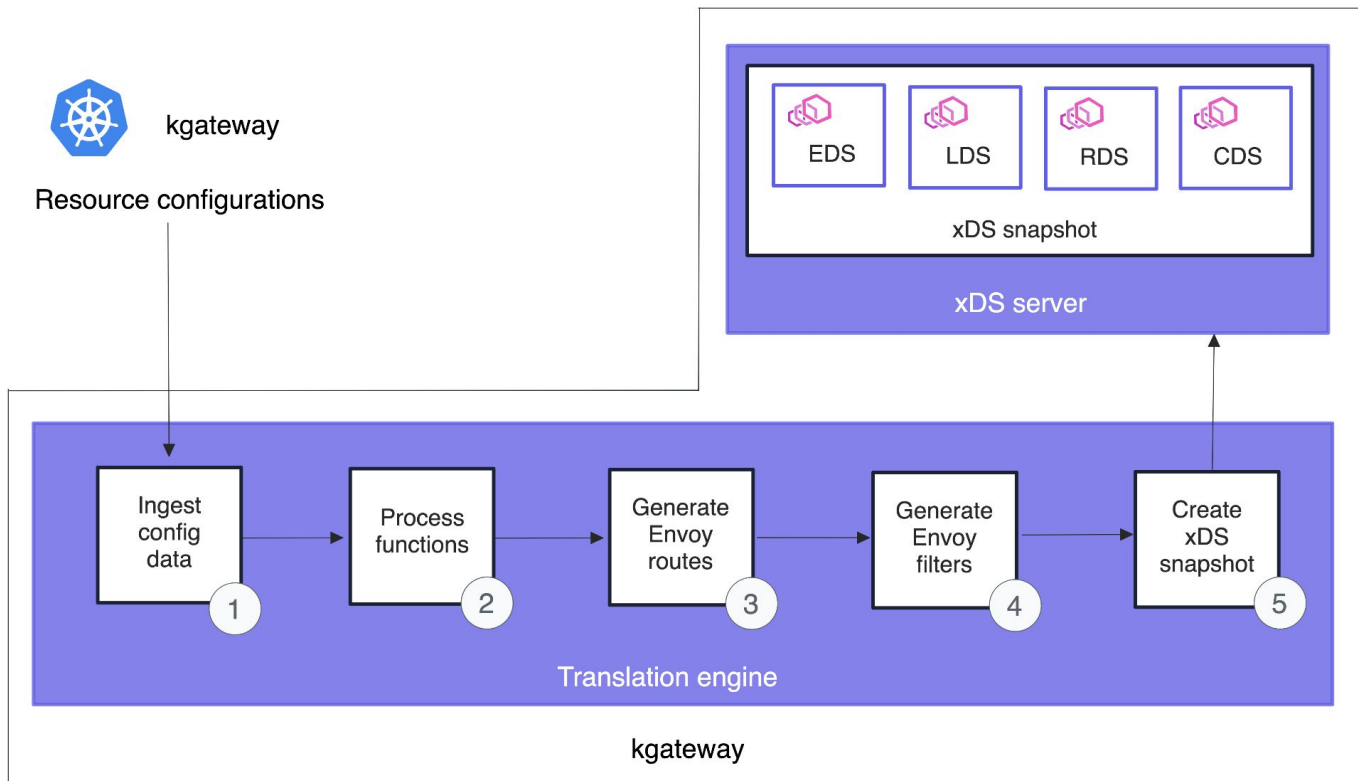


KubeCon

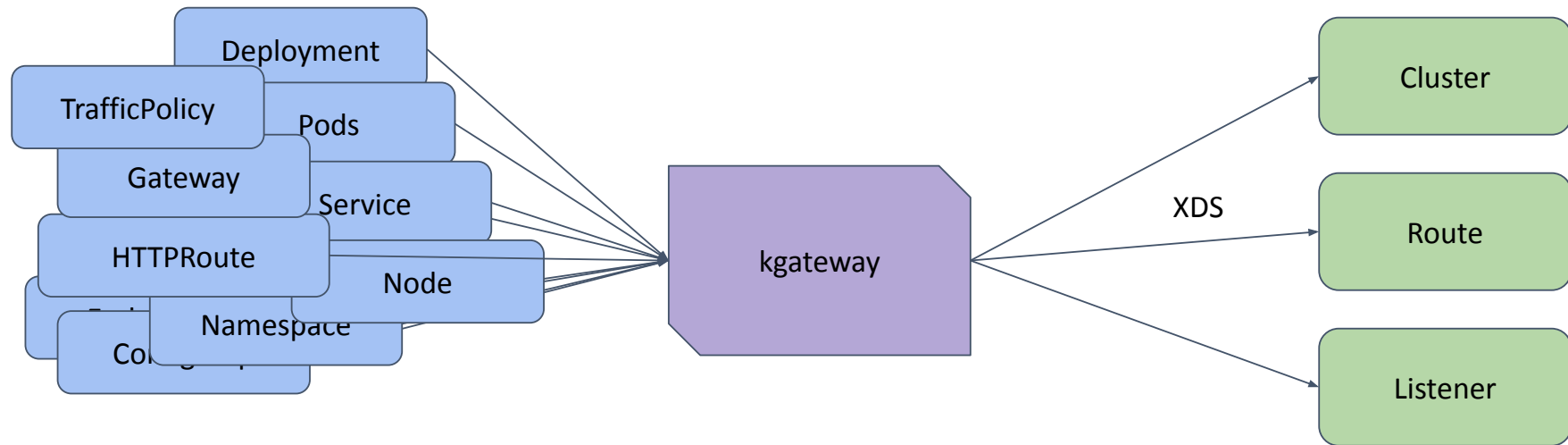


CloudNativeCon

Europe 2026



# Control Plane Internals



1. Read a bunch of Kubernetes objects
2. Translate them into envoy config

# Control Plane Internals

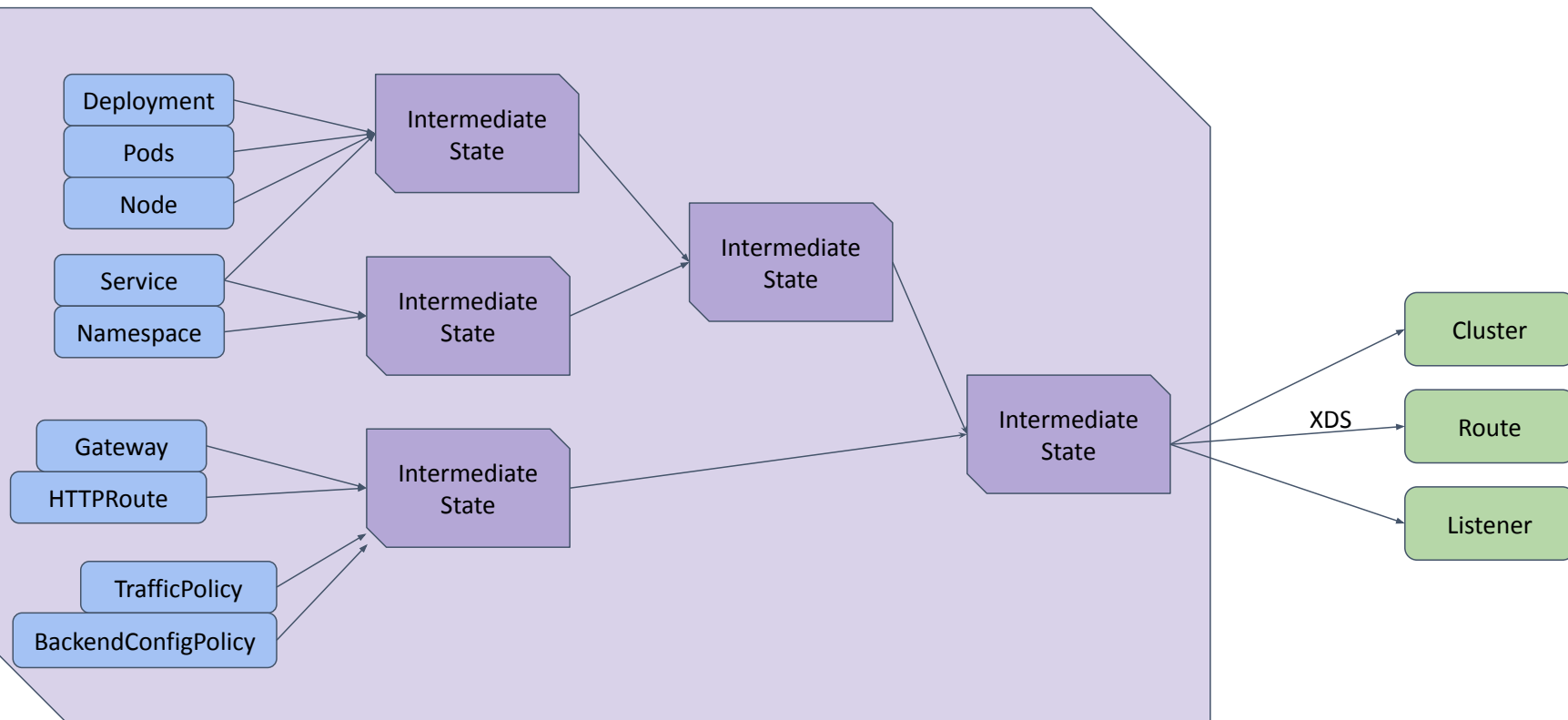


KubeCon



CloudNativeCon

Europe 2026



# What is xds?

- Envoy Proxy dynamically discovers and update its configuration
- At runtime
- Without restarting
- (xDS) x Discovery Service
  - LDS (Listener Discovery Service)
  - RDS (Route Discovery Service)
  - CDS (Cluster Discovery Service)

# Kubernetes Gateway API



KubeCon



CloudNativeCon

Europe 2026

- SIG-Network project being built to standardize service networking in Kubernetes
- Portable API specifications are defined as custom resources and are supported by many implementations.

```
apiVersion: gateway.networking.k8s.io/v1
kind: Gateway
metadata:
  name: ingress-gateway
spec:
  gatewayClassName: kgateway
  listeners:
  - name: http
    port: 80
    protocol: HTTP
```



# Kubernetes Gateway API



KubeCon



CloudNativeCon

Europe 2026

## Infrastructure Provider:

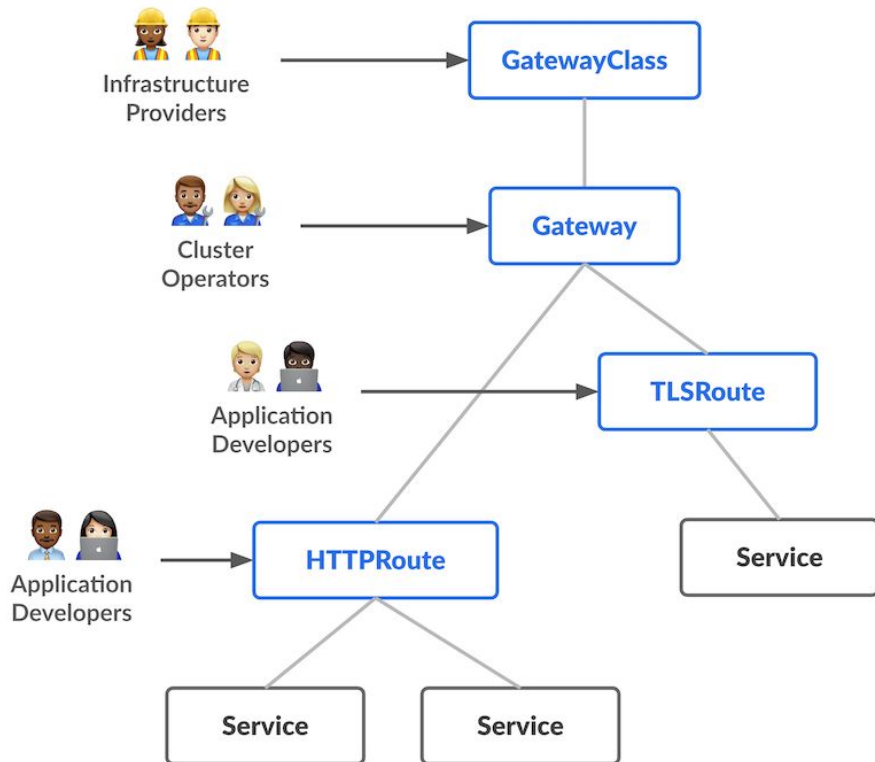
- Manages infrastructure
  - allows multiple isolated clusters to serve multiple tenants, e.g. a cloud provider

## Cluster Operator:

- Manages clusters
  - policies, network access, application permissions, etc.

## Application Developer:

- Manages an application
  - application-level configuration and Service composition





KubeCon



CloudNativeCon

Europe 2026

# Code Base Walkthrough



# Navigating the Codebase



KubeCon



CloudNativeCon

Europe 2026

## kgateway Repo

- `/api`: Kubebuilder api definitions
- `/install`: Helm charts
- `/pkg`: shared packages across some/all binaries
  - `/kgateway` - Core controller logic
    - `/extensions2/plugins` - integrations that extend translation (eg: Policies)
- `/tests/e2e`: e2e tests



KubeCon



CloudNativeCon

Europe 2026

# Testing: Conformance Tests



# What do Conformance Tests Test?



KubeCon



CloudNativeCon

Europe 2026

- Verify that implementations match the Kubernetes Gateway API specification
- Behaviour-driven tests
  - Traffic Routing && Modification
  - Resource Status
- Core vs Extended Tests

# Where do Conformance Tests Come From?



KubeCon



CloudNativeCon

Europe 2026

<https://github.com/kubernetes-sigs/gateway-api/tree/main/conformance>

kubernetes-sigs / gateway-api

Code

Issues 154

Pull requests 56

Agents

Discussions

Actions

Projects

Security

Insights

Q Type to search

+

main

gateway-api / conformance / tests

Go to file

Add file

zirain

fix GatewayFrontendInvalidDefaultClientCertificateValidation test (#4629)

b91ed0c · 3 hours ago

History

| Name                                             | Last commit message                                                      | Last commit date |
|--------------------------------------------------|--------------------------------------------------------------------------|------------------|
| ..                                               |                                                                          |                  |
| mesh                                             | Optimize mesh (GRPC HTTP)RouteWeight tests (#4300)                       | 3 months ago     |
| backendtlspolicy-conflict-resolution.go          | Issue 3940: Move BackendTLSPolicy to standard (#4074)                    | 6 months ago     |
| backendtlspolicy-conflict-resolution.yaml        | fix: use correct API version in BackendTLSPolicy conformance tests (#... | 4 months ago     |
| backendtlspolicy-invalid-ca-certificate-ref.go   | Issue 3940: Move BackendTLSPolicy to standard (#4074)                    | 6 months ago     |
| backendtlspolicy-invalid-ca-certificate-ref.yaml | Issue 3940: Move BackendTLSPolicy to standard (#4074)                    | 6 months ago     |
| backendtlspolicy-invalid-kind.go                 | Issue 3940: Move BackendTLSPolicy to standard (#4074)                    | 6 months ago     |
| backendtlspolicy-invalid-kind.yaml               | Issue 3940: Move BackendTLSPolicy to standard (#4074)                    | 6 months ago     |

# Levels of Conformance

- Conformant
  - Core
  - Claimed Extended Features
- Partially Conformant
  - Subset
  - Aspire to be Conformant
- Stale
  - Were conformant but haven't submitted reports recently
- <https://gateway-api.sigs.k8s.io/implementations/#conformant>

# Code Time: Run the Conformance Tests



KubeCon



CloudNativeCon

Europe 2026

```
# Setup cluster with kgw installed
```

```
$ make run
```

```
# Run specific conformance test
```

```
$ make conformance-HTTPRouteHeaderMatching
```

```
# Run all conformance tests
```

```
$ make all-conformance
```



KubeCon



CloudNativeCon

Europe 2026

# How does kgateway extends the Gateway API



# The Gap Gateway API Intentionally Leaves



KubeCon



CloudNativeCon

Europe 2026

The Gateway API defines several resource types. The most common ones you will encounter are:

|              |                                                            |
|--------------|------------------------------------------------------------|
| GatewayClass | - which implementation handles the Gateway (e.g. kgateway) |
| Gateway      | - the front door (addresses, listeners, TLS)               |
| HTTPRoute    | - HTTP routing rules (which path goes to which Service)    |
| GRPCRoute    | - gRPC routing rules                                       |
| TLSRoute     | - TLS passthrough routing                                  |
| TCPRoute     | - TCP routing (experimental)                               |

That covers **routing**. Production systems need more. The Gateway API spec deliberately does not define how these work. It leaves room for each implementation to add these features via their own CRDs.

| Need                   | Example                                   |
|------------------------|-------------------------------------------|
| Rate limiting          | Max 100 requests/second per client        |
| Authentication         | Only allow requests with a valid JWT      |
| CORS                   | Tell browsers which origins are permitted |
| Request transformation | Strip a header before forwarding          |
| Fault injection        | Inject delays for chaos testing           |

This is the **Policy Attachment** pattern:

- You create a project specific policy resource.
- You point it at a Gateway, HTTPRoute, or other supported resource via `targetRefs`.
- The controller applies the behavior.

# How kgateway fills that gap?



KubeCon



CloudNativeCon

Europe 2026

kgateway's primary policy CRD is `TrafficPolicy`.

Here is a minimal example attaching rate limiting to a route:

```
apiVersion: gateway.kgateway.dev/v1alpha1
kind: TrafficPolicy
metadata:
  name: api-rate-limit
  namespace: team-a
spec:
  targetRefs:
    - group: gateway.networking.k8s.io
      kind: HTTPRoute
      name: my-api-route          # which HTTPRoute to attach to
  rateLimit:
    local:
      tokenBucket:
        maxTokens: 100          # burst capacity
        tokensPerFill: 10       # tokens added each interval
        fillInterval: 1s        # refill every 1 second = 10 req/s steady state
```

Key idea: `targetRefs` is what links the policy to the route.

Without it the policy does nothing.

The `targetRefs` mechanism is standardized across all Gateway API implementations.

The contents of `spec` (the actual policy fields) are kgateway-specific.

# What TrafficPolicy Supports



KubeCon



CloudNativeCon

Europe 2026

TrafficPolicy supports many fields — here are some of them:

| Field                  | What it does                                                    |
|------------------------|-----------------------------------------------------------------|
| rateLimit              | Token-bucket (local) or external service (global) rate limiting |
| cors                   | CORS headers and preflight handling                             |
| extAuth                | Delegate auth to an external authorization service              |
| jwtAuth                | Validate JWTs, extract claims to headers                        |
| transformation         | Rewrite request/response headers and bodies                     |
| retry                  | Automatic retries with backoff                                  |
| timeouts               | Per-route request timeouts                                      |
| basicAuth / apiKeyAuth | Simple authentication methods                                   |

<https://medium.com/@mayorfaj/gateway-api-kgateway-vs-ingress-controllers-a-comprehensive-comparison-b7567533f138>

# What Other Policies kgateway Supports



KubeCon



CloudNativeCon

Europe 2026

kgateway also provides other policy CRDs for different concerns:

ibbo42

| CRD                 | Purpose                                                                           |
|---------------------|-----------------------------------------------------------------------------------|
| HTTPListenerPolicy  | Access logging, custom error responses                                            |
| ListenerPolicy      | Proxy protocol, connection limits                                                 |
| Backend             | Custom backend types (AWS Lambda, static upstreams)                               |
| BackendConfigPolicy | Health checks, circuit breaking                                                   |
| GatewayParameters   | Customize the Envoy proxy deployment itself                                       |
| DirectResponse      | Return a static HTTP response without forwarding to a backend                     |
| GatewayExtension    | Reference external services used by policies (rate limit server, ext-auth server) |

# The Translation Pipeline



KubeCon



CloudNativeCon

Europe 2026

When you apply a `TrafficPolicy`, kgateway translates it in three steps before Envoy ever sees it:

Your YAML → IR (internal model) → Merge with routes → xDS (Envoy config) → Envoy

**IR (Intermediate Representation)** is kgateway's internal Go struct — a middle step between your human-readable YAML and Envoy's low-level config format. Think of it as a blueprint: not the final product, but structured enough to work with and easy to test.

1. **CRD -> IR:** The plugin converts your YAML into an internal Go struct (IR). Close to Envoy's format, but not yet.
2. **Policy Attachment:** kgateway merges the IR onto the matching HTTPRoute or Gateway listener.
3. **IR -> xDS:** The final Envoy configuration is generated and streamed to Envoy — no restart needed.

The IR step is what makes translation testable: the golden file tests verify step 3 by comparing the xDS output against a saved expected file.



KubeCon



CloudNativeCon

Europe 2026

# Testing: Unit Tests



# How We Test Translation - Golden File Tests



KubeCon



CloudNativeCon

Europe 2026

kgateway uses golden file tests (also called snapshot tests) to verify translation.

How they work:

1. You write an input YAML (Gateway + HTTPRoute + TrafficPolicy etc.).
2. The test runs that through the translator.
3. The output xDS config is compared to a saved expected output file (the golden file).
4. If the output differs from the golden file, the test fails.

Where these live:

```
pkg/kgateway/translator/gateway/gateway_translator_test.go  <- test runner
pkg/kgateway/translator/gateway/testutils/inputs/            <- input YAMLs
pkg/kgateway/translator/gateway/testutils/outputs/           <- golden files
```

When you add a new feature or fix a bug:

```
# Regenerate golden files after making intentional changes
REFRESH_GOLDEN=true go test ./pkg/kgateway/translator/gateway/...
```

Then inspect the output diff to confirm the change is exactly what you intended.

# Hands-on Exercise: Break it, Find it , Fix it



KubeCon



CloudNativeCon

Europe 2026

**Background:** Now that we understand how plugins translate `TrafficPolicy` into Envoy config, let's look at a real bug. There's a mistake in the global rate limit plugin — the kind you might make in a real PR. Envoy would quietly receive invalid config, and rate limiting would silently do nothing. Your job: run the test, read the diff, find the bug, fix it.

## Step 1: Check out the exercise branch

```
git fetch origin
git checkout kubecone-2026-contribfest
```

## Step 2: Run the failing test

```
go test ./pkg/kgateway/translator/gateway/... \
-v -run "TestBasic/TrafficPolicy_RateLimit_Full_Config"
```

You should see a test failure. Look at the diff carefully — `-` lines are what the golden file expects, `+` lines are what the code actually produces.

**Tip:** Scan the diff for field names that look subtly wrong.

# Understand the tests



KubeCon



CloudNativeCon

Europe 2026

## Step 3: Compare Output

The test reads this input file:

```
pkg/kgateway/translator/gateway/testutils/inputs/traffic-policy/rate-limit-full-config.yaml
```

and compares the translator output against:

```
pkg/kgateway/translator/gateway/testutils/outputs/traffic-policy/rate-limit-full-config.yaml
```

Find the `rateLimit.global` section in the input. Notice:

```
rateLimit:
  global:
    descriptors:
      - entries:
          - type: Path
```

The `type: Path` entry tells Envoy to rate limit based on the request path.

# Find the bug and Fix it



KubeCon



CloudNativeCon

Europe 2026

## Step 4: Find the bug

Open the global rate limit plugin:

```
pkg/kgateway/extensions2/plugins/trafficpolicy/global_rate_limit_plugin.go
```

Look at the `createRateLimitActions` function. Find the `case kgateway.RateLimitDescriptorEntryTypePath:` block.

What `HeaderName` value is being set? What should it be?

▼ Hint

The header name should be `:path`.

## Step 5: Fix it

Make the fix, then re-run the test:

```
go test ./pkg/kgateway/translator/gateway/... \
-v -run "TestBasic/TrafficPolicy_RateLimit_Full_Config"
```

## Step 6: Verify all rate limit tests pass

```
go test ./pkg/kgateway/translator/gateway/... \
-v -run "TestBasic/TrafficPolicy.*[Rr]ate"
```



KubeCon



CloudNativeCon

Europe 2026

# Testing: benchmarks



# Benchmarks in kgateway



KubeCon



CloudNativeCon

Europe 2026

**Gateway API** implementation puts a proxy (Envoy, NGINX, etc.) in front of traffic. The proxy is fast. The bottleneck is how quickly the ***\*control plane\**** translates your Kubernetes resources into proxy configuration.

The Gateway API exposes an `attachedRoutes` status field on the Gateway object. It reports how many Route resources have been successfully attached to the Gateway.

In this test, we create a large number of routes and measure how quickly that status field reflects the updated count.

gateway-api-bench: <https://github.com/howardjohn/gateway-api-bench?tab=readme-ov-file>

## Focus on Attached Routes

### What Attached Routes measures

1. Create N `HTTPRoute` resources attached to a Gateway.
2. Watch `.status.listeners[].attachedRoutes`
3. Measure how long it takes for status to catch up.

### Key metrics to call out

- **Setup time:** time to add one more route after the baseline is already in place
- **Teardown time:** time to remove that route and return to the baseline state
- **Total writes:** how much status churn the controller produces

kgateway load testing:

<https://github.com/kgateway-dev/kgateway/blob/main/test/e2e/features/loadtesting/README.md>

# How Attached Routes Works



KubeCon



CloudNativeCon

Europe 2026

## Methodology

1. Create a baseline of N routes, for example 1000.
2. Wait until those routes are fully processed.
3. Start the stopwatch and add exactly 1 more route.
4. Stop when `attachedRoutes` changes from N to N+1.
5. Delete that route and measure how long it takes to return from N+1 back to N.

## What the numbers mean

- **Setup time** answers: how quickly can the control plane absorb one more route at scale?
- **Teardown time** answers: how quickly can it remove that route and stabilize again?
- This isolates steady-state behavior better than bulk create timings.

# How kgateway Runs Attached Routes



KubeCon



CloudNativeCon

Europe 2026

- Designed in **EP #11210** and implemented in `test/e2e/features/loadtesting/`
- Integrated into the existing **testify-based e2e framework**. No separate benchmarking binary.
- Uses **VClusterSimulator** to create fake nodes, services, and `EndpointSlice` objects.
- Uses **event-driven monitoring** to record `attachedRoutes` changes over time.
- kgateway uses **KRT** for dependency tracking.
- Adding route #1001 should not force expensive reprocessing of routes #1 through #1000.

- Records two metrics for this talk:
  - **Setup time**
  - **Teardown time**
- Uses **scale-aware thresholds**:
  - Baseline: 1000 routes
  - Production: 5000 routes

```
568 attachedroutes_suite.go:357: Gateway test-gateway: AttachedRoutes=1000 (expecting 1000)
569 attachedroutes_suite.go:333: Translation completed: all baseline routes are attached
570 attachedroutes_suite.go:367: Verifying routes are valid for gateway: test-gateway
571 attachedroutes_suite.go:388: Route test-gateway-route-0 is valid and accepted
572 attachedroutes_suite.go:279: Starting event-driven monitoring for gateway: kgateway-loadtest-
1773898838081890790/test-gateway
573 attachedroutes_suite.go:308: Registering event handler for gateway: kgateway-loadtest-
1773898838081890790/test-gateway
574 attachedroutes_suite.go:232: Phase 9: === STARTING STOPWATCH === Creating 1 incremental HTTPRoute
575 attachedroutes_suite.go:397: Creating single incremental HTTPRoute
576 attachedroutes_suite.go:432: Created incremental route: incremental-route-1773898870464798232
577 attachedroutes_suite.go:238: Phase 10: Probing gateway until incremental route returns 200
578 attachedroutes_suite.go:437: Probing gateway until incremental route returns 200
579 attachedroutes_suite.go:291: Gateway event #1: kgateway-loadtest-1773898838081890790/test-gateway
AttachedRoutes=1001 (ResourceVersion=5567)
580 attachedroutes_suite.go:304: Status update recorded: AttachedRoutes=1001 (total samples: 1)
581 attachedroutes_suite.go:473: Probing: AttachedRoutes=1001 (expecting 1001)
582 attachedroutes_suite.go:477: Gateway condition ready met! Time: 1.039007043s
583 attachedroutes_suite.go:243: Phase 11: === STOPWATCH STOPPED === User Time: 1.043413927s (incremental
```



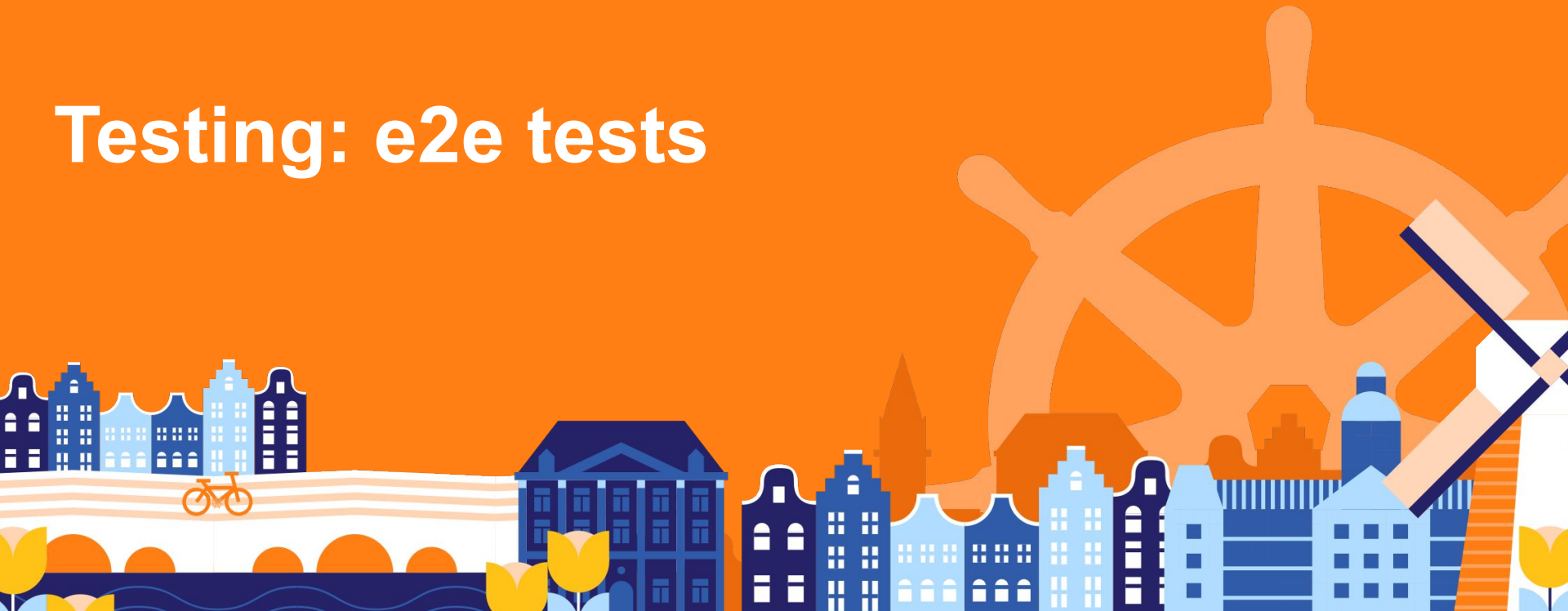
KubeCon



CloudNativeCon

Europe 2026

# Testing: e2e tests



# Code Time: Run the e2e tests



KubeCon



CloudNativeCon

Europe 2026

1. Go to GitHub Codespace
2. Open **test/e2e/features/header\_modifiers/suite.go**
3. Follow instructions in the comments to run the test

# Recap: Testing 101



KubeCon



CloudNativeCon

Europe 2026

- Test Like a User
  - If users wouldn't do it in Kubernetes, our e2e tests shouldn't either.
    - Kubernetes manifests
    - Helm installs
    - CI/CD-style updates
- Test Real Behavior
  - Validate real traffic through the gateway, not just internal state
- Make Tests Reusable
- Optimize for Developers



KubeCon



CloudNativeCon

Europe 2026

# Testing: manual!



# Code Time: Labs in kgateway



KubeCon



CloudNativeCon

Europe 2026

**Sometimes you just need to see a feature in action:**

The screenshot shows a web interface for a 'Cloud Connectivity' track. At the top, it says 'TRACK 02' and 'API gateways & service mesh'. Below this, there's a section for 'kgateway' described as 'Cloud-native API Gateway built on Envoy Proxy and Kubernetes Gateway API.' with '3 LABS'. Three lab cards are listed: '01 Basic Authentication with kgateway', '02 Migrate to kgateway with the ingress2gateway tool', and '03 Basics of Kubernetes Gateway API'. Each card has a 'Launch' button with an external link icon.



<https://play.instruqt.com/soloio/invite/uys5cvqduc0e>

# More: Labs in kgateway

**Homework: test kgateway features in sandbox**



<https://kgateway.dev/resources/labs/>



<https://kgateway.dev/learn/>



KubeCon



CloudNativeCon

Europe 2026

# Kgateway community



# Community Structure



KubeCon



CloudNativeCon

Europe 2026

Contributor Ladder:

- **Code Owner**
  - Maintainer with special review permissions for sensitive sections of the codebase
- **Maintainer**
  - Approve pull requests in specific directories
  - Recommend and review other contributors to become Maintainers
- **Org Member**
  - May be assigned Issues and Reviews.
  - May give commands to CI/CD automation.
  - May label/triage issues.
- **Contributor**

**Community Meetings biweekly Tuesday 2pm ET**



# Become a kgateway contributor!



KubeCon



CloudNativeCon

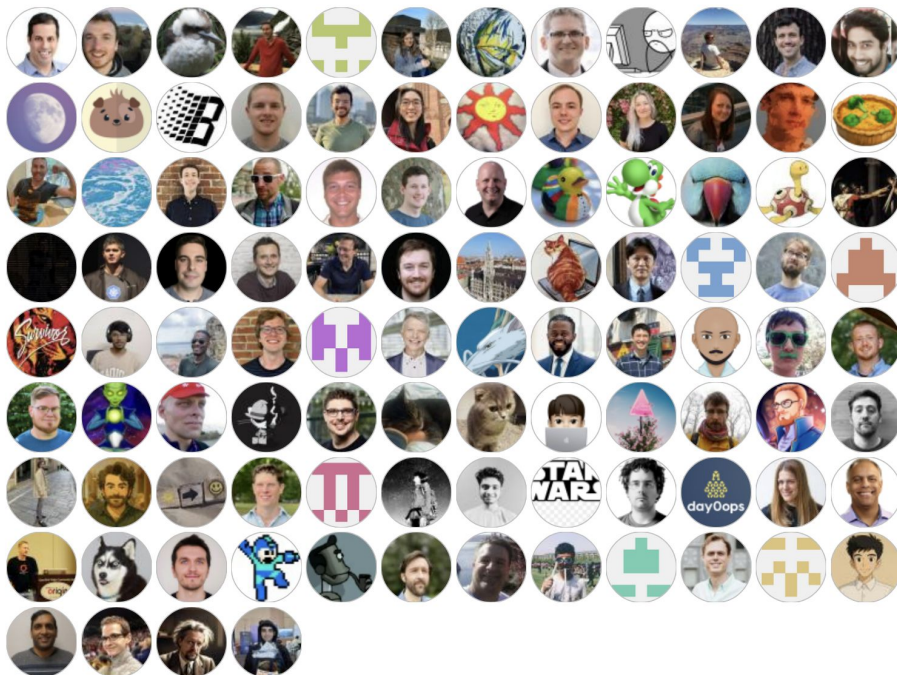
Europe 2026

## Contributions are not only code! You can...

- Documentation fixes and updates
- Report issues
- Show up to meetings, take notes
- Answer questions from other community members on slack/GitHub
- Submit feedback on issues and PRs
- Test releases and patches
- Run events
- Promote the project in public, and more!

## Contributors

Thanks to all contributors who are helping to make kgateway better!



# How to Join the Community?



KubeCon



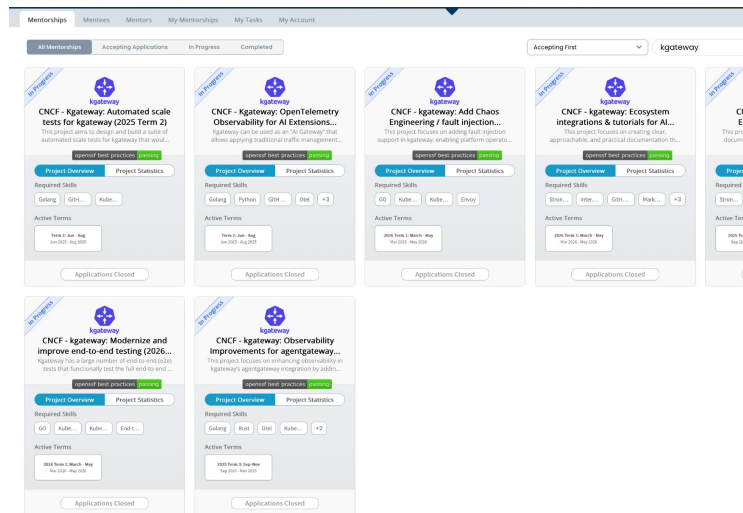
CloudNativeCon

Europe 2026

- CNCF slack! #kgateway
- Bi-weekly Community Meeting on Tuesday 2pm ET
- LFX Mentorship Program & Google Summer of Code
- Good First Issues on GitHub

## Want to learn more?

- Try out the project by following the docs:  
<https://kgateway.dev/docs/>
- Instruqt Labs: <https://www.solo.io/resources/lab>



# LFX Mentorship Program



KubeCon



CloudNativeCon

Europe 2026

Get paid to contribute to Open Source!

- Three applications cycles per year, often multiple projects per cycle!

The screenshot displays the LFX Mentorship Program application interface. At the top, there are navigation tabs: Mentorships, Mentees, Mentors, My Mentorships, My Tasks, and My Account. Below these, there are filters for 'All Mentorships', 'Accepting Applications', 'In Progress', and 'Completed'. The main content area shows a grid of project cards, each representing a different mentorship opportunity. Each card includes the project title, a brief description, a 'Project Overview' tab, a 'Project Statistics' tab, 'Required Skills', 'Active Terms', and an 'Applications Closed' button. The projects are organized into two rows of five cards each. The first row includes projects like 'CNCF - Kgateway: Automated scale tests for kgateway (2025 Term 2)', 'CNCF - Kgateway: OpenTelemetry Observability for AI Extensions...', 'CNCF - kgateway: Add Chaos Engineering / fault injection...', 'CNCF - kgateway: Ecosystem integrations & tutorials for AI...', and 'CNCF - kgateway: Improve Ecosystem Integrations...'. The second row includes 'CNCF - kgateway: Modernize and improve end-to-end testing (2026...)' and 'CNCF - kgateway: Observability Improvements for agentgateway...'. Each card also features a 'passing' status indicator and a 'Project Overview' button.



KubeCon



CloudNativeCon

Europe 2026

# Thank you!

