



KubeCon



CloudNativeCon

Europe 2026

Exploring ~~kgateway~~ agentgateway to write your own GatewayAPI backend

Ricardo Katz - Red Hat



Who am I

- Software Engineer @ Red Hat - All Gateway API stuff
- Maintainer of Gateway API @ Kubernetes
- Former Ingress NGINX maintainer
- Like Lego, airplanes, traveling and making jokes at the wrong moment



KubeCon



CloudNativeCon

Europe 2026

So you want your own Gateway API implementation.



Motivations



KubeCon



CloudNativeCon

Europe 2026

- Maybe using Apache HTTPD as a backend! (why not?!)
 - Spoiler alert: I did with NGINX because it is my comfort zone
- I DO NOT want my dataplane accessing Kubernetes API - Ingress NGINX trauma!
- I want to make it easy and reusable for other backends



KubeCon

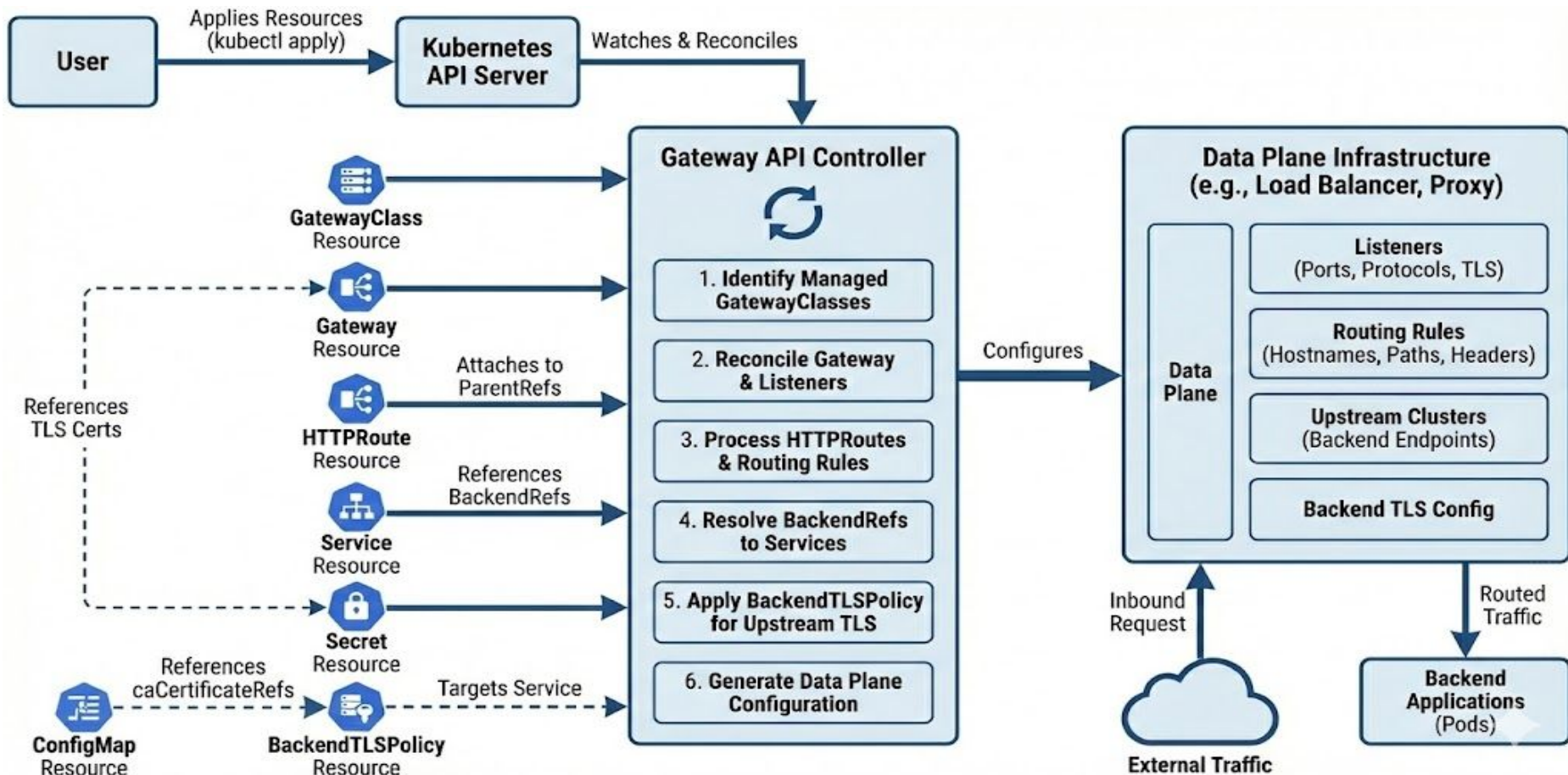


CloudNativeCon

Europe 2026

Creating your own Gateway API implementation is easy...



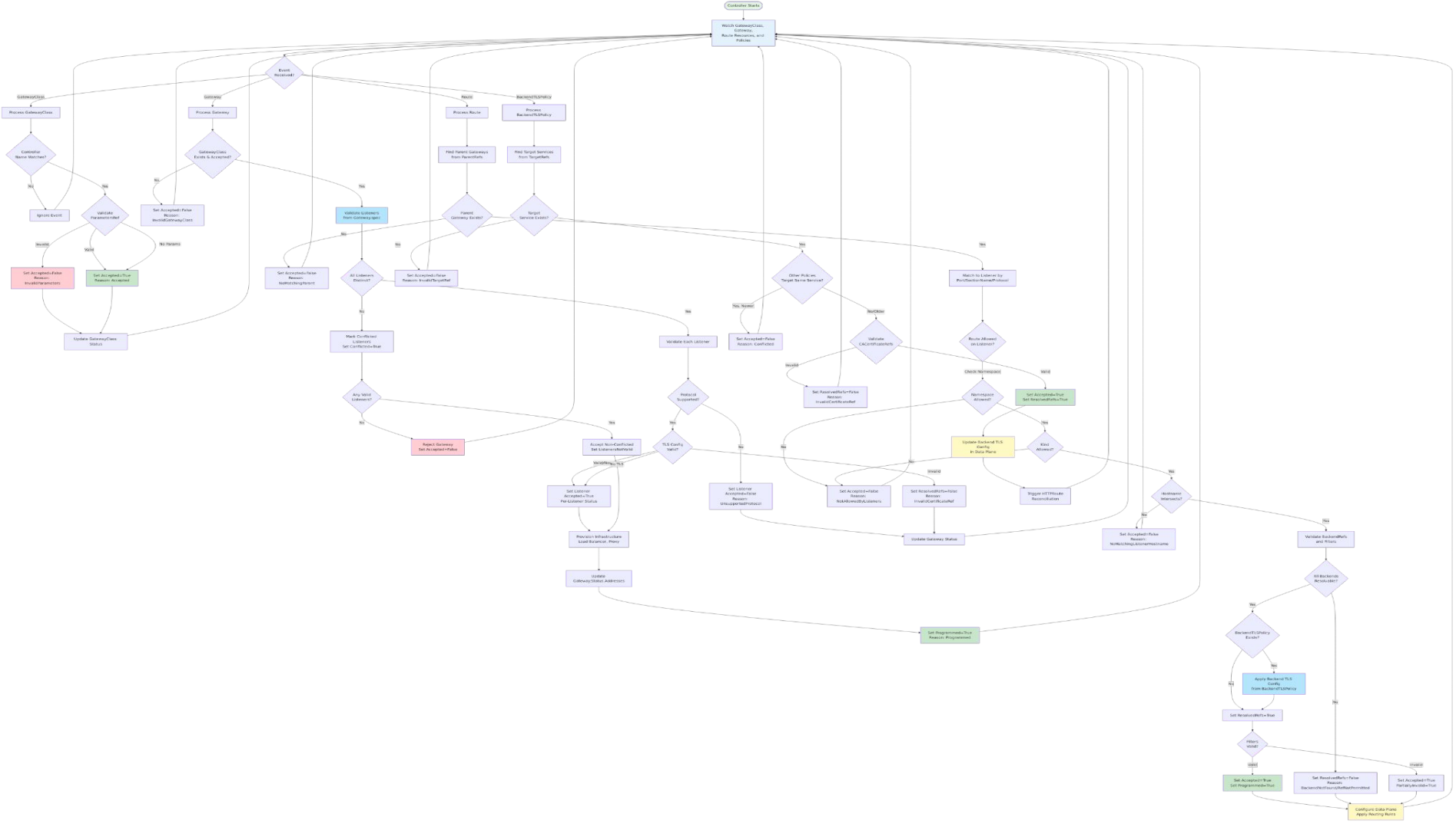


KubeCon



CloudNativeCon

Europe 2026





KubeCon

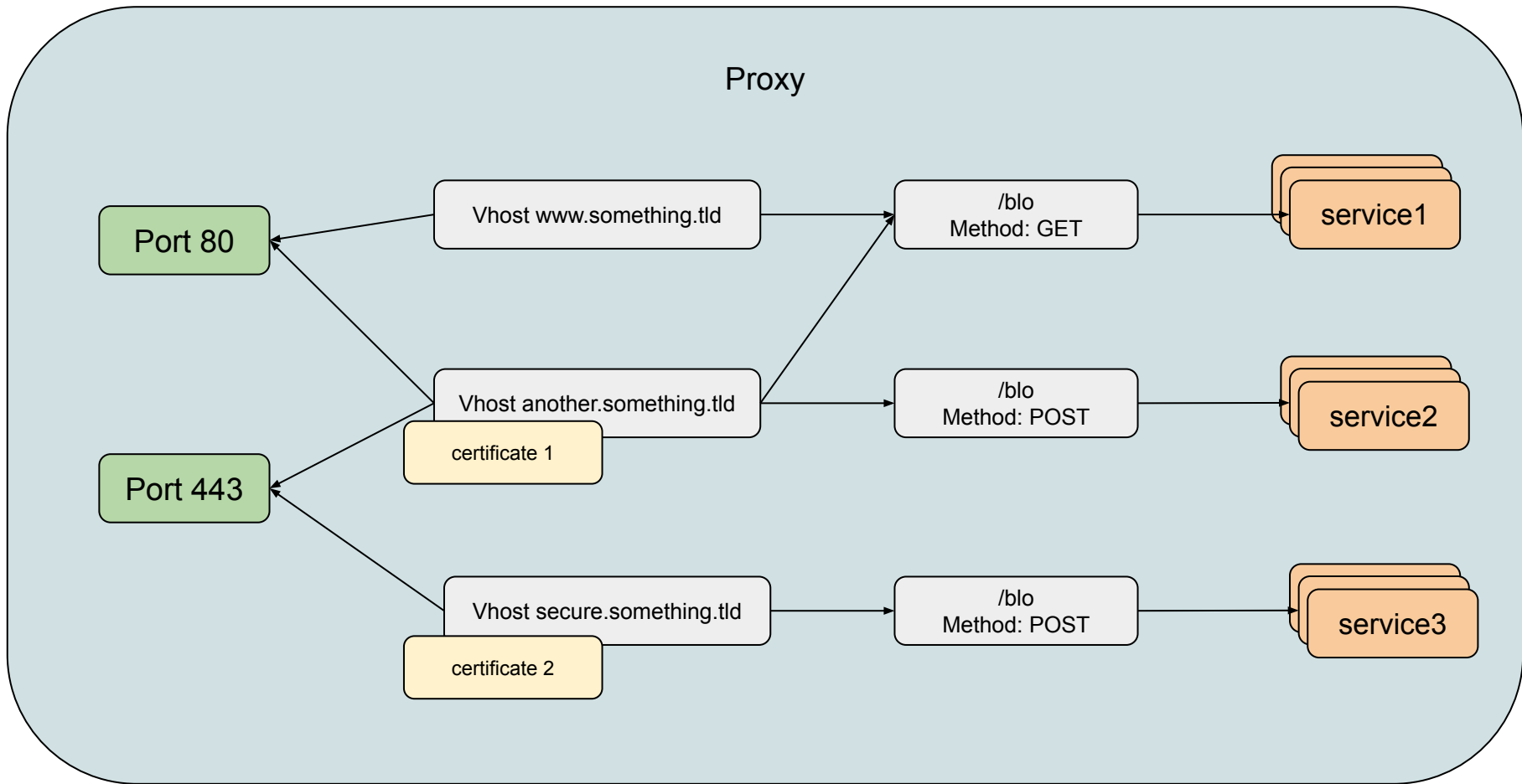


CloudNativeCon

Europe 2026

**But the result is almost
always the same...**





Like nginx



KubeCon



CloudNativeCon

Europe 2026

```
server {
    listen 80 default_server;
    server_name _;
    ... # Locations / Routes come here
}

server {
    listen 443 ssl;
    server_name foo.example.com;
    ssl_certificate /path/to/foo-cert.crt;
    ssl_certificate_key /path/to/foo-cert.key;
    ... # Locations / Routes come here
}

server {
    listen 443 ssl;
    server_name *.example.com;
    ssl_certificate /path/to/wildcard-cert.crt;
    ssl_certificate_key /path/to/wildcard-cert.key;
    ... # Locations / Routes come here
}
```

Or HAProxy



KubeCon



CloudNativeCon

Europe 2026

```
frontend http-in
```

```
    bind *:80
```

```
frontend https-foo
```

```
    bind *:443 ssl crt /etc/haproxy/certs/foo-cert.pem crt-ignore-err all
```

```
    mode http
```

```
    acl is_something req_ssl_sni -i foo.example.com
```

```
    use_backend something_backend if is_something
```

```
frontend https-wildcard
```

```
    bind *:443 ssl crt /etc/haproxy/certs/wildcard-cert.pem crt-ignore-err all
```

```
    mode http
```

```
    acl is_wildcard req_ssl_sni -m end .ricardo.com
```

```
    use_backend wildcard_backend if is_wildcard
```

Or Envoy



KubeCon



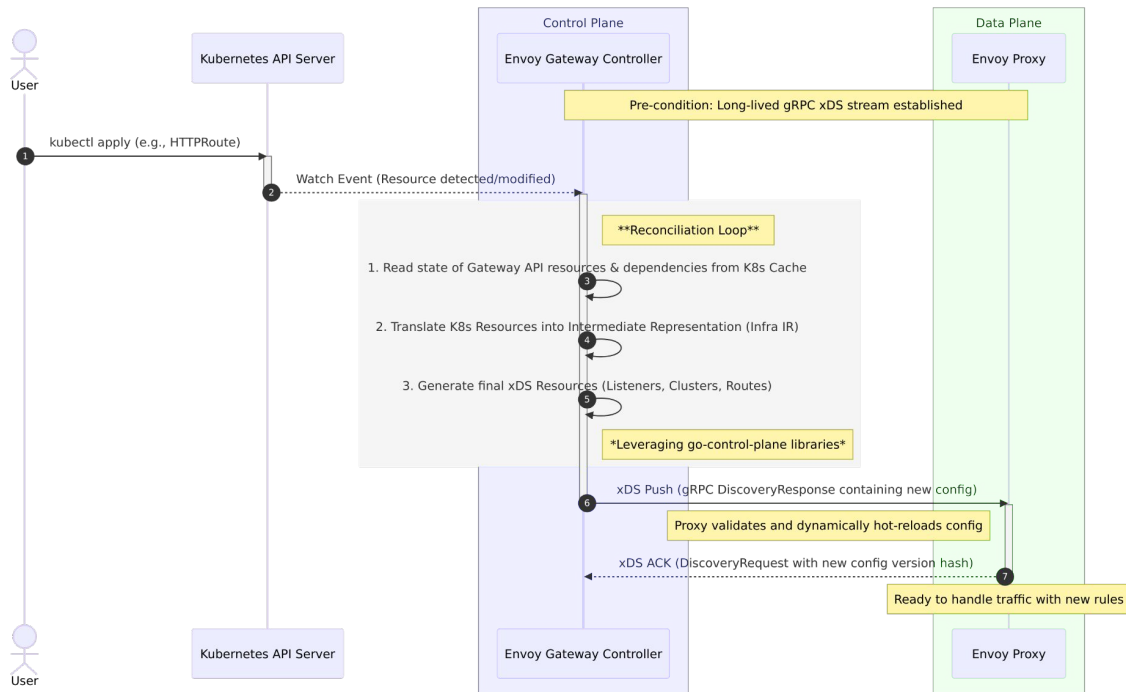
CloudNativeCon

Europe 2026

```
static_resources :
  listeners :
    - name: listener_80
      address :
        socket_address : { address: 0.0.0.0, port_value: 80 }
        # Extra configs and routes here
    - name: listener_443
      address :
        socket_address : { address: 0.0.0.0, port_value: 443 }
      filter_chains :
        - filter_chain_match :
            server_names : ["foo.example.com "]
            transport_socket :
              name: envoy.transport_sockets.tls
              typed_config :
                "@type": type.googleapis.com/envoy.extensions.transport_sockets.tls.v3.DownstreamTlsContext
                common_tls_context :
                  tls_certificates :
                    - certificate_chain :
                        filename : "/etc/envoy/foo-cert.crt "
                      private_key :
                        filename : "/etc/envoy/foo-cert.key "
            filters :
              # Your routes here
        - filter_chain_match :
            server_names : ["*.example.com "]
            transport_socket :
              name: envoy.transport_sockets.tls
              typed_config :
                "@type": type.googleapis.com/envoy.extensions.transport_sockets.tls.v3.DownstreamTlsContext
                common_tls_context :
                  tls_certificates :
                    - certificate_chain :
                        filename : "/etc/envoy/wildcard-cert.crt "
                      private_key :
                        filename : "/etc/envoy/wildcard-cert.key "
            filters :
              # Your routes here
```

Then Envoy Gateway was created

- A Gateway API implementation based on Envoy, that reconciles Gateway API resources
- A control plane that generates Envoy configuration
- The dataplanes connect to the control plane and the control plane pushes the configuration via xDS
- But to consume this on something not Envoy is complicated!



KubeCon



CloudNativeCon

Europe 2026



KubeCon



CloudNativeCon

Europe 2026

Then kgateway and agentgateway appeared

But why they are important?



KubeCon



CloudNativeCon

Europe 2026

- kgateway - A new proposal of a Gateway API controller - Still uses Envoy Proxy
- agentgateway - Gateway API, but oriented to AI workload
- agentgateway is written in Rust, but uses kgateway as its control plane

But why they are important?



KubeCon



CloudNativeCon

Europe 2026

- kgateway - A new proposal of a Gateway API controller - Still uses Envoy Proxy
- agentgateway - Gateway API, but oriented to AI workload
- agentgateway is written in Rust, but uses kgateway as its control plane

A Rust written proxy? But the dataplane in Go?
Maybe there is an opportunity here!

How it works?



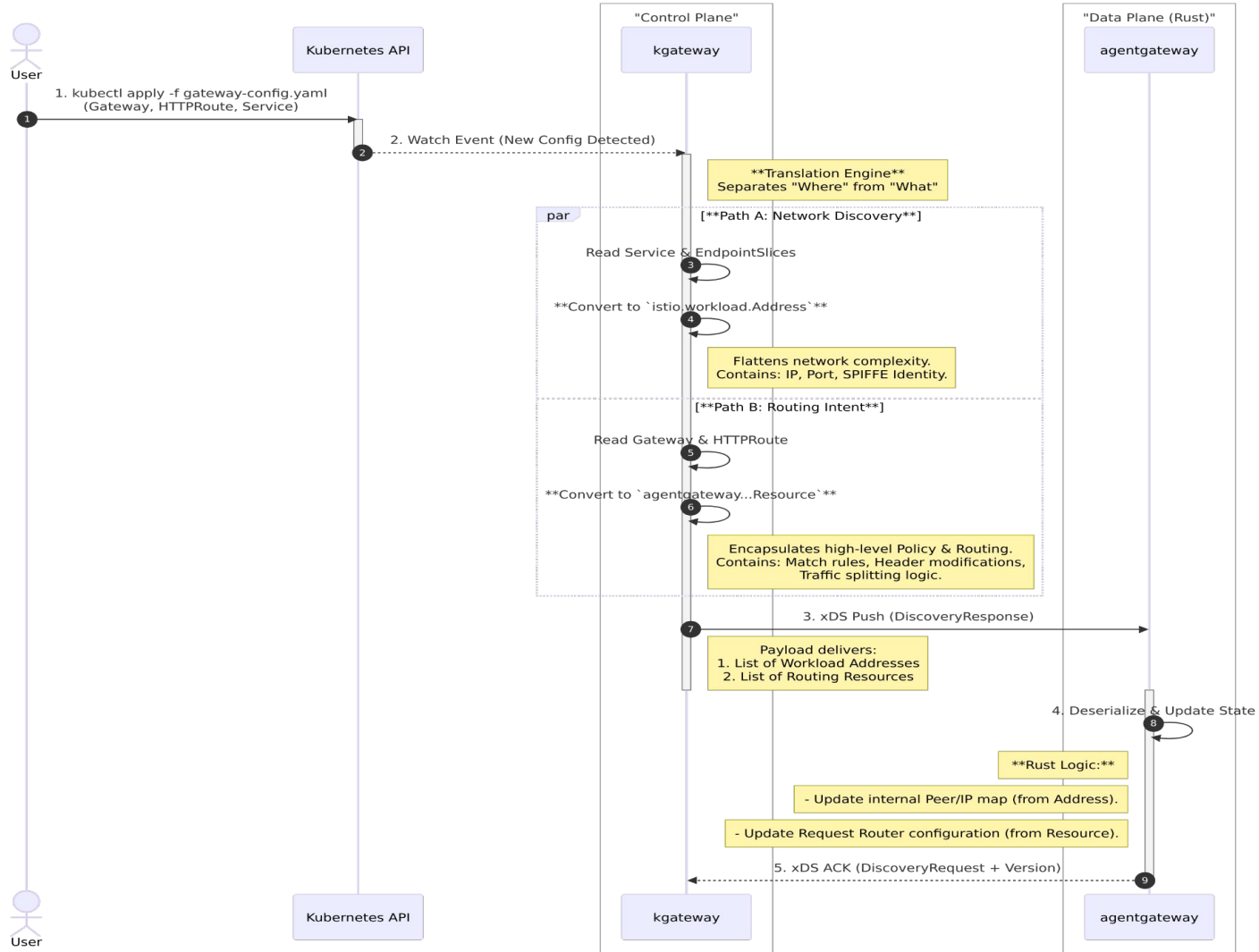
KubeCon



CloudNativeCon

Europe 2026

- kgateway - gRPC / xDS server with Aggregated Discovery Service
- Model generated for agentgateway is based on the intent and much simpler than the Envoy layer
- agentgateway model can be reused by other implementations
 - `type.googleapis.com/istio.workload.Address` = workload IPs, protocols (upstream servers)
 - `type.googleapis.com/agentgateway.dev.resource.Resource` = listeners, routes, certificates, etc
 - Each message can carry a different type of Kind (eg.: Resource/Route, workload/address)
- Will show this working with more details soon



Understanding ADS



KubeCon

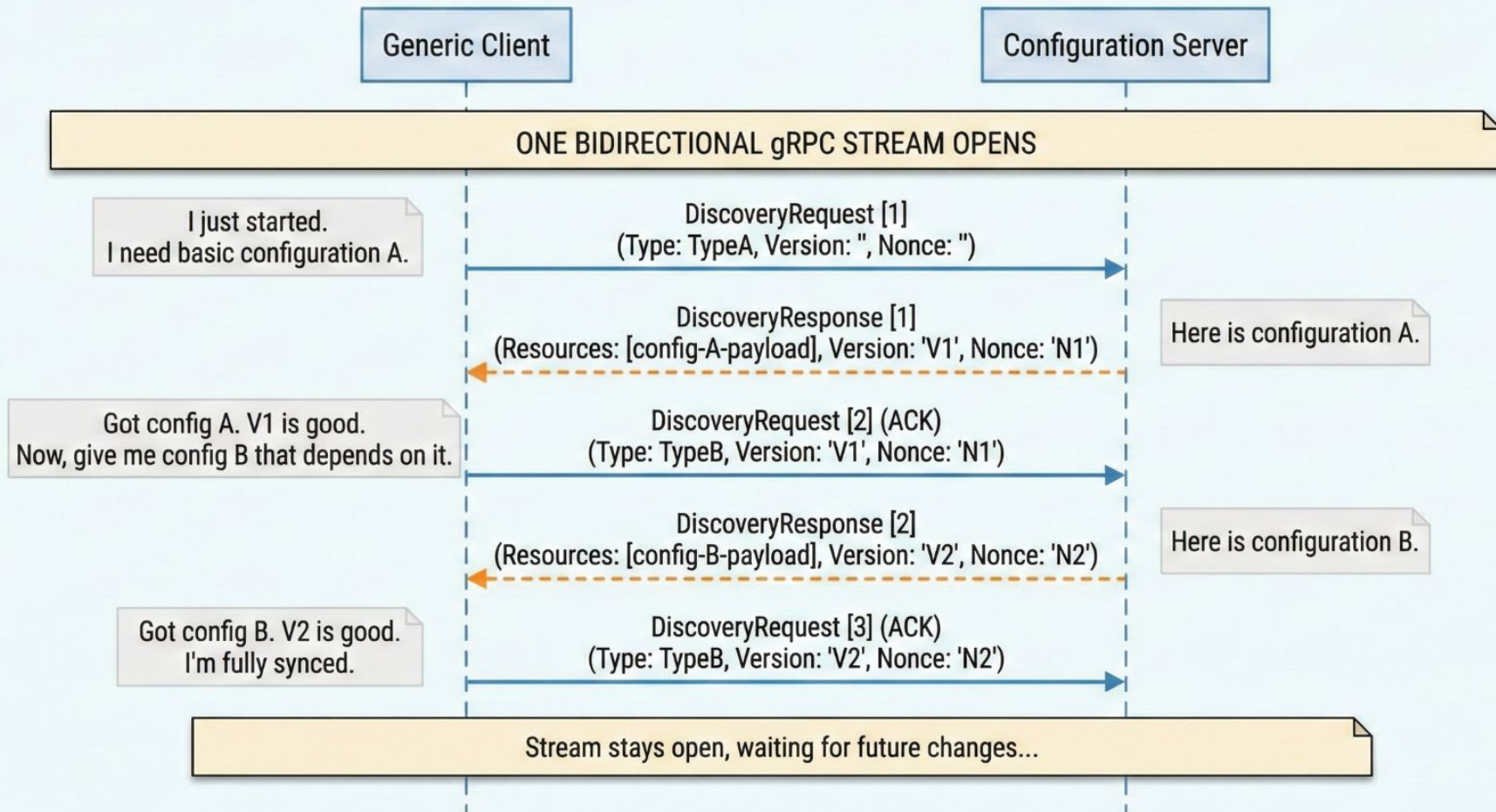


CloudNativeCon

Europe 2026

- Aggregated Discovery Services - Variant of xDS APIs
- Consolidates the information on a single, ordered stream instead of making one update per resource
- Works on a “request/response/ack” model
 - Client authenticates with nodeID and requests a resource and its version (or empty version to get “everything”)
 - Server responds with the resources, the current version and a unique string (nonce) to track the update
 - Client responds with a ACK/NACK

Generic Aggregated Discovery Service (ADS) gRPC Flow



So where to start?

- I wrote a client
- The client will behave as agentgateway proxy
- It will authenticate to agentgateway control plane and get the xDS messages
- But then...

So where to start?

- I wrote a client
- The client will behave as agentgateway proxy
- It will authenticate to agentgateway control plane and get the xDS messages
- But then...

I've found a CVE - [CVE-2025-64323](#) - kgateway Weak authentication

So where to start?

- I wrote a client
- The client will behave as agentgateway proxy
- It will authenticate to agentgateway control plane and get the xDS messages
- But then...

I've found a CVE - [CVE-2025-64323](#) - kgateway Weak authentication

- CVE fixed, continue the client implementation
- A nice TUI that uses this client and shows me how agentgateway control plane generates the configuration for agentgateway proxy

Discovered types - Workload addresses



KubeCon



CloudNativeCon

Europe 2026

type.googleapis.com/istio.workload.Address - ([proto](#))

- Message type “Workload” contains address (IPs)
 - Contain the ports of workloads, can contain specific information
- Message type “Service” contains a group of Workloads
 - Similar to “Workload” and can contain other specific information

Basically contains the endpoint addresses that we care

Discovered types - Resources



KubeCon



CloudNativeCon

Europe 2026

`type.googleapis.com/agentgateway.dev.resource.Resource` - ([proto](#))

- Message type “Bind” contains the bind address for the proxy
 - Eg.: Listen on port 443 TCP
- Message type “Listener” contains a listener (bound to a port)
 - Eg.: Port 443 serves HTTP and uses a certificate
- Message type “Route” contains a match to a traffic
 - Eg.: /app binds to listener-1, routes to backends app1 and uses traffic policy “pol1”
(eg.: BackendTLSPolicy)
- There are other message types (Policy specification, TCPRoute, etc)

A snippet of the client



KubeCon



CloudNativeCon

Europe 2026

```
resourceHandlerStore := xdsclient.ResourceTypeStore{
    "type.googleapis.com/istio.workload.Address":      store.NewAddressStore(),
    "type.googleapis.com/agentgateway.dev.resource.Resource": store.NewResourceStore(),
}

client, err := xdsclient.NewADSClient(xdsclient.Config{
    ServerAddress:      kgatewayEndpoint,
    Namespace:          namespace,
    GatewayName:        gatewayname,
    Token:              token,
    ResourceTypeHandlers: resourceHandlerStore,
})

if err != nil {
    fatal("error initiating the client", err)
}

if err := client.Start(context.Background()); err != nil {
    fatal("error running client", err)
}
```

A snippet of the client



KubeCon



CloudNativeCon

Europe 2026

```
type ResourceTypeHandler interface {
    Remove(ctx context.Context, resourceNames []string) error
    Update(ctx context.Context, resources []*discovery.Resource) error
}

// ResourceTypeStore defines a map of typeURLs and the handler
// that should be called for each type.
type ResourceTypeStore map[string]ResourceTypeHandler
```

2 items

UPDATE - Address

17:32:20

UPDATE - Resource

17:32:20

```
      "bind": {
        "key": "80/agentgateway-system/agentgateway-ricardo1",
        "port": 80
      }
    },
    {
      "listener": {
        "key": "agentgateway-system/agentgateway-ricardo1.http",
        "bindKey": "80/agentgateway-system/agentgateway-ricardo1",
        "name": {
          "gatewayName": "agentgateway-ricardo1",
          "gatewayNamespace": "agentgateway-system",
          "listenerName": "http"
        },
        "protocol": "HTTP"
      }
    },
    {
      "route": {
        "key": "default/echo.0.0.http",
        "listenerKey": "agentgateway-system/agentgateway-ricardo1.http",
        "name": {
          "kind": "HTTPRoute",
          "name": "echo",
          "namespace": "default"
        },
        "hostnames": [
          "some.example.tld"
        ],
        "matches": [
          {
            "path": {
              "pathPrefix": "/"
            }
          }
        ],
        "backends": [
          {
            "backend": {
              "service": {
                "namespace": "default",
                "hostname": "echo.default.svc.cluster.local"
```

ADS Messages

5 items

UPDATE - Address

17:42:52

UPDATE - Resource

17:42:52

UPDATE - Address

17:42:56

REMOVE - Address

17:42:57

UPDATE - Address

17:42:58

```
"/Pod/default/echo-545cbd868c-gpqb2"  
]
```

5 items

UPDATE - Address

17:32:20

UPDATE - Resource

17:32:20

UPDATE - Address

17:33:39

REMOVE - Address

17:33:39

UPDATE - Address

17:33:40

```
{
  "workload": {
    "uid": "//Pod/default/echo-545cbd868c-gpqb2",
    "name": "echo-545cbd868c-gpqb2",
    "namespace": "default",
    "addresses": [
      "CvQACQ=="
    ],
    "serviceAccount": "default",
    "node": "kubeconeu2026-control-plane",
    "canonicalName": "echo",
    "canonicalRevision": "latest",
    "workloadType": "POD",
    "workloadName": "echo",
    "services": {
      "default/echo.default.svc.cluster.local": {
        "ports": [
          {
            "servicePort": 3000,
            "targetPort": 3000
          }
        ]
      }
    }
  }
}
```

6 items

UPDATE - Address

17:32:20

UPDATE - Resource

17:32:20

UPDATE - Address

17:33:39

REMOVE - Address

17:33:39

UPDATE - Address

17:33:40

UPDATE - Resource

17:34:49

```
{
  "route": {
    "key": "default/echo.0.0.http",
    "listenerKey": "agentgateway-system/agentgateway-ricardo1.http",
    "name": {
      "kind": "HTTPRoute",
      "name": "echo",
      "namespace": "default"
    },
    "hostnames": [
      "some.example.tld"
    ],
    "matches": [
      {
        "path": {
          "pathPrefix": "/bla"
        }
      }
    ],
    "backends": [
      {
        "backend": {
          "service": {
            "namespace": "default",
            "hostname": "echo.default.svc.cluster.local"
          },
          "port": 3000
        },
        "weight": 1
      }
    ]
  }
}
```

Implementing a NGINX proxy



KubeCon



CloudNativeCon

Europe 2026

- Use the demo client to connect to agentgateway controlplane
- Triggers different actions for “Resources” or “Address” messages
- For every message
 - Parse the message, write a configuration file (add or remove configuration)
 - Uses [nginx-go-crossplane](#) instead of templates (more trauma from ingress-nginx)
 - Reload NGINX
- Warning: This is NOT a Gateway API conformant implementation

```
apiVersion: agentgateway.dev/v1alpha1
kind: AgentgatewayParameters
metadata:
  name: nginx-gateway
  namespace: agentgateway-system
spec:
  deployment:
    spec:
      template:
        spec:
          containers:
            - args:
                - -token-file
                - /var/run/secrets/xds-tokens/xds-token
                - -endpoint
                - agentgateway.agentgateway-system.svc.cluster.local:9978
              name: agentgateway
              securityContext:
                readOnlyRootFilesystem: false
                runAsUser: 101
          image:
            registry: example.com
            repository: nginx-gateway
            tag: v1771796588
```

```
apiVersion: gateway.networking.k8s.io/v1
kind: GatewayClass
metadata:
  name: nginx
spec:
  controllerName: agentgateway.dev/agentgateway
  description: Specialized class for NGINX using kgateway
  parametersRef:
    name: nginx-gateway
    group: agentgateway.dev
    kind: AgentgatewayParameters
    namespace: agentgateway-system
```

```
apiVersion: gateway.networking.k8s.io/v1
kind: Gateway
metadata:
  name: nginx-gateway
  namespace: agentgateway-system
spec:
  gatewayClassName: nginx
  listeners:
    - protocol: HTTP
      port: 80
      name: http
      allowedRoutes:
        namespaces:
          from: All
```

Demo it working!



KubeCon



CloudNativeCon

Europe 2026

Repository: <https://github.com/rikatz/notyetanotherenvoybackend>

To test:

- Fork the repository
- Execute "make demo" (you need Docker, Kind and Go compiler for now!)
- There are some instructions also for the GUI on README.md

What's next



KubeCon



CloudNativeCon

Europe 2026

- Anyone can write their own implementation relying on kgateway model
- Istio is moving towards supporting agentgateway using the same model - [PRs](#)
- Supporting new features is much easier using this model
- No need to understand Gateway API model, just parse ADS messages correctly
 - Eg.: ListenerSet is HARD to implement
 - So are new AI features (GIE)
 - But on our side, we just need to know how to configure the proxy

Some references



KubeCon



CloudNativeCon

Europe 2026

- github.com/rikatz/notyetanotherenvoybackend
- [agentgateway syncer README](#)
- [Agentgateway KRT translators](#)

Feedbacks are welcome!



KubeCon



CloudNativeCon

Europe 2026





KubeCon



CloudNativeCon

Europe 2026

