KubeCon CloudNativeCon

THE LINUX FOUNDATION

OPEN SOURCE SUMMIT

AI_dev

Open Source GenAI & ML Summit

China 2024

KubeCon

CloudNativeCon

THE LINUX FOUNDATION
OPEN SOURCE SUMMIT

AI_dev
Open Source GenAI & ML Summit

China 2024

Gateway API and Beyond: Introducing Envoy Gateway's Gateway API Extensions

Huabing Zhao, Tetrate
Envoy Gateway Maintainer

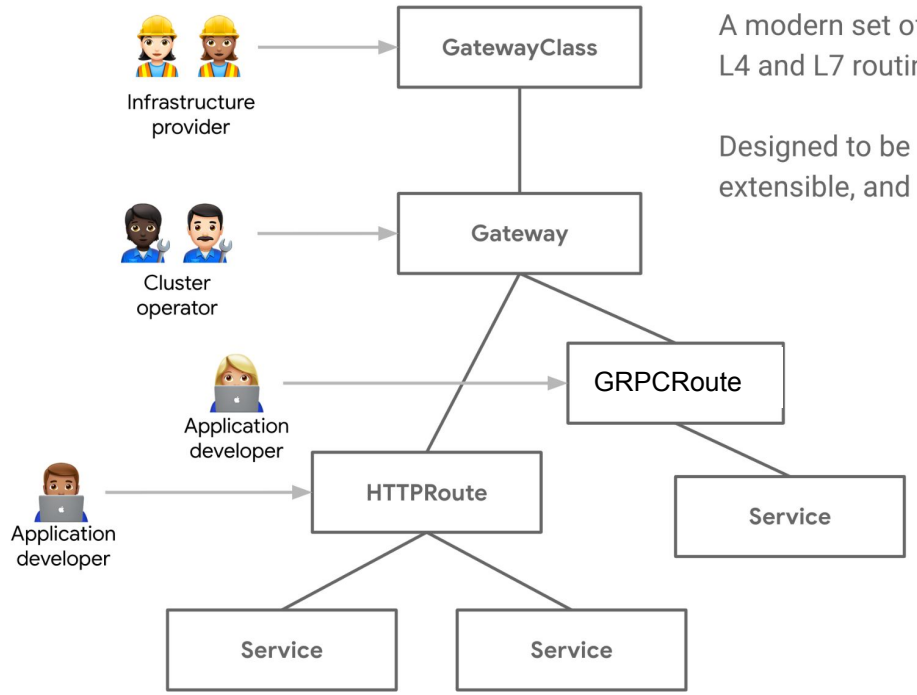
Gateway API as Management API

Ingress

```yaml
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: test
spec:
  rules:
    - host: foo.bar.com
      http:
        paths:
          - path: /foo
            backend:
              serviceName: s1
              servicePort: 80
          - path: /bar
            backend:
              serviceName: s2
              servicePort: 80
```

- HTTP host matching
- HTTP path matching
- TLS

Gateway API



A modern set of APIs for deploying L4 and L7 routing in Kubernetes

Designed to be generic, expressive, extensible, and role-oriented

Powerful traffic management

- HTTP header-based matching
- HTTP header manipulation
- Weighted traffic splitting
- gRPC,UDP,TCP routing
- Role-oriented resource model

Flexible extension mechanism

- Arbitrary backend
- Custom filters
- Policy Attachment

A Minimum configuration

GatewayClass: Defines a class of Gateways that share a common configuration and behaviour

- The controller that is managing Gateways of this class
- Configuration parameters of this GatewayClass

Infrastructure provider

```yaml
apiVersion: gateway.networking.k8s.io/v1
kind: GatewayClass
metadata:
  name: eg
spec:
  controllerName: gateway.envoyproxy.io/gatewayclass-controller
  parametersRef:
    group: gateway.envoyproxy.io
    kind: EnvoyProxy
    name: config
    namespace: envoy-gateway-system
```

```yaml
apiVersion: gateway.envoyproxy.io/v1alpha1
kind: EnvoyProxy
metadata:
  namespace: envoy-gateway-system
  name: config
spec:
  telemetry:
    tracing:
      samplingRate: 100
      provider:
        host: otel-collector.monitoring.svc.cluster.local
        port: 4317
        type: OpenTelemetry
```

A Minimum configuration

Gateway: Specifies how traffic can enter the cluster

- The GatewayClass used for this Gateway
- Listeners that accepts network connections

Cluster operator


```yaml
apiVersion: gateway.networking.k8s.io/v1
kind: Gateway
metadata:
  name: foo-gateway
spec:
  gatewayClassName: eg
  listeners:
    - name: https
      protocol: HTTPS
      hostname: "foo.example.com"
      port: 443
      tls:
        mode: Terminate
        certificateRefs:
          - name: server-cert
```

A Minimum configuration

xRoute: Define protocol-specific rules for mapping requests from a Gateway to Backend Services.

- How to match, process, and forward the request to a backend service

Site Developer →

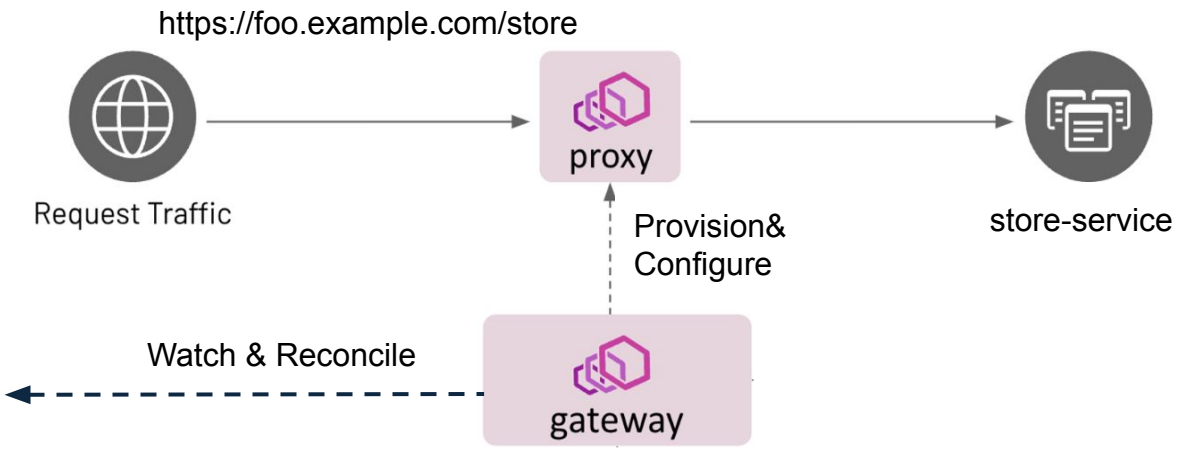
```yaml
apiVersion: gateway.networking.k8s.io/v1
kind: HTTPRoute
metadata:
  name: store-route
spec:
  parentRefs:
    - name: foo-gateway
  hostnames:
    - "foo.example.com"
  rules:
    - matches:
      - path:
          value: "/store"
      backendRefs:
        - kind: Service
          name: store-service
          port: 3000
```

A Minimum configuration

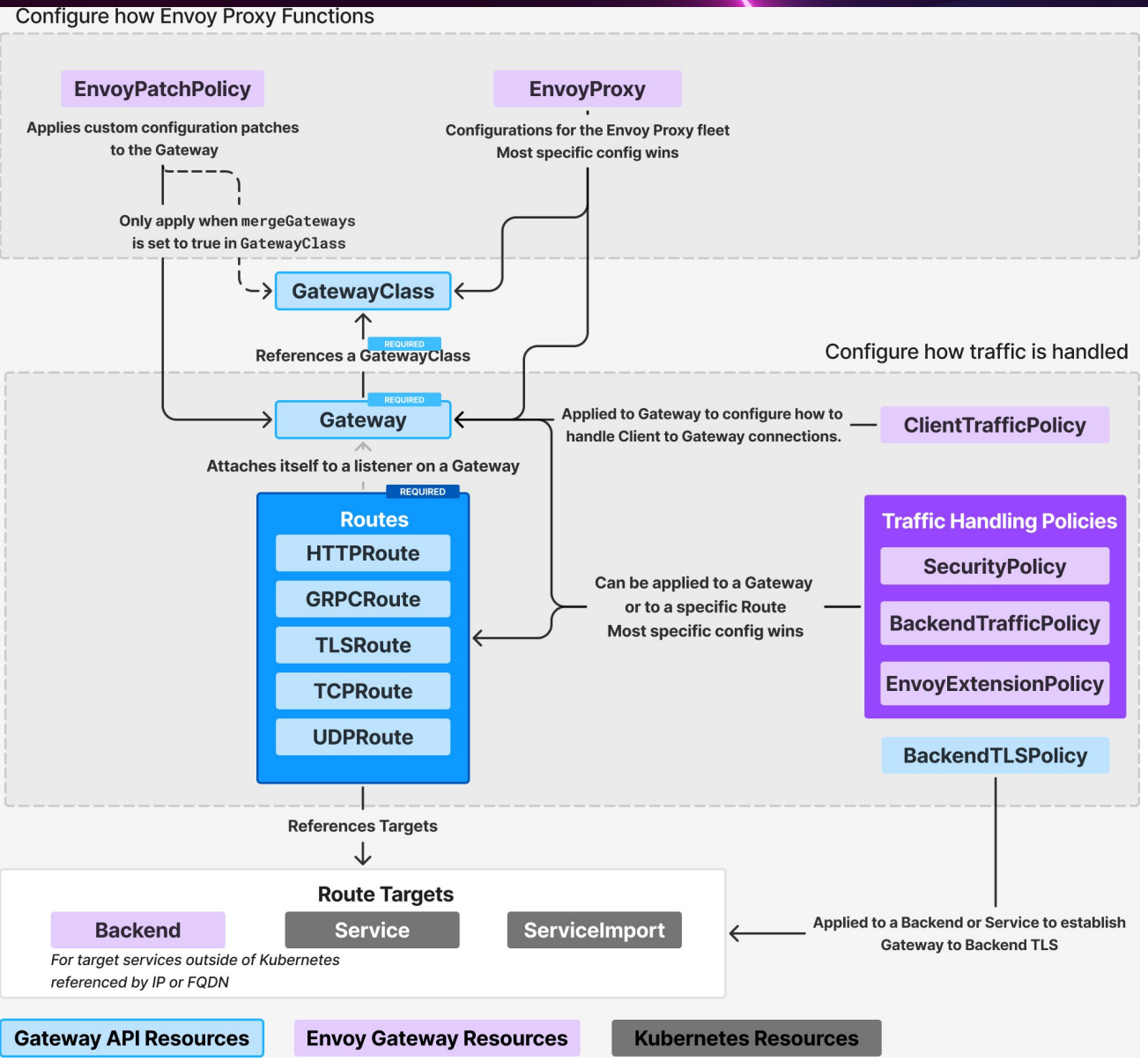

```yaml
apiVersion: gateway.networking.k8s.io/v1
kind: GatewayClass
metadata:
  name: eg
spec:
  controllerName: gateway.envoyproxy.io/gatewayclass-controller
  parametersRef:
    group: gateway.envoyproxy.io
    kind: EnvoyProxy
    name: config
    namespace: envoy-gateway-system
```

```yaml
apiVersion: gateway.networking.k8s.io/v1
kind: Gateway
metadata:
  name: foo-gateway
spec:
  gatewayClassName: eg
  listeners:
    - name: https
      protocol: HTTPS
      hostname: "foo.example.com"
      port: 443
      tls:
        mode: Terminate
        certificateRefs:
          - name: server-cert
```

```yaml
apiVersion: gateway.networking.k8s.io/v1
kind: HTTPRoute
metadata:
  name: store-route
spec:
  parentRefs:
    - name: foo-gateway
  hostnames:
    - "foo.example.com"
  rules:
    - matches:
      - path:
          value: "/store"
      backendRefs:
        - kind: Service
          name: store-service
          port: 3000
```



Envoy Gateway Resources



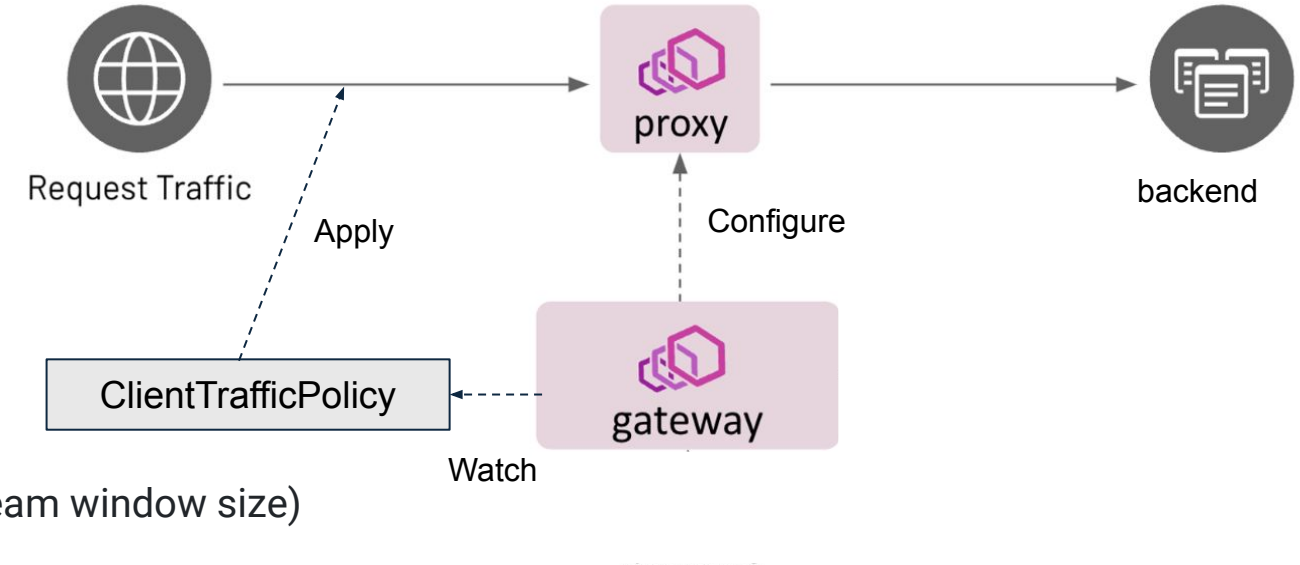
Envoy Gateway API Resources

- **EnvoyProxy:** Represents the deployment and configuration of the Envoy proxy within a Kubernetes cluster, managing its lifecycle and settings.
- **Backend:** A resource that makes routing to cluster-external backends easier and makes access to external processes via Unix Domain Sockets possible.
- **xPolicy :** Additional policies to enhance Gateway API resources
  - **ClientTrafficPolicy: :** Configuration for downstream client to Envoy listener.
  - **BackendTrafficPolicy :** Configuration for Envoy to backend service.
  - **SecurityPolicy :** Configuration for security settings.
  - **EnvoyExtensionPolicy :** Configuration for Enovy extensions(Wasm, ExtProc).
  - **EnvoyPatchPolicy :** Abritary patches to the generated xDS.

ClientTrafficPolicy

Traffic policy for client connections (connections between client and Envoy)

- TCP settings for downstream client connections
  - TCP Keepalive
  - TCP Timeout (TCP Idle time)
  - Connection Limit
  - Socket and Connection Buffer size
- TLS settings for the downstream client connections
  - Should and how to verify the client cert
  - TLS options: version, ciphers, ALPN, etc.
- HTTP settings for downstream client connections
  - HTTP Timeout (Request timeout, HTTP Idle time)
  - HTTP1/HTTP2/HTTP3 settings (For example: HTTP2 stream window size)
- Other downstream client connections related configurations
  - Wether Proxy protocol is enabled or not on the client connection
  - How to detect the original client IP of the client request



Please note: not all features can be applied to all Listener types.

If a targeeted Listener does not support a feature, the feature will be ignored. For example, the HTTP2 setting will be ignored if the Listener is a TCP Listener.

ClientTrafficPolicy

Targets

- Gateway: ClientTrafficPolicy applies on all listeners on the targeted Gateway
- Listener: ClientTrafficPolicy applies on the specified Listener only

```yaml
apiVersion: gateway.networking.k8s.io/v1
kind: Gateway
metadata:
  name: eg
spec:
  gatewayClassName: internet
  listeners:
    - name: http
      protocol: HTTP
      port: 80
    - name: https
      protocol: HTTPS
      port: 443
      tls:
        mode: Terminate
        certificateRefs:
          - name: server-cert
```

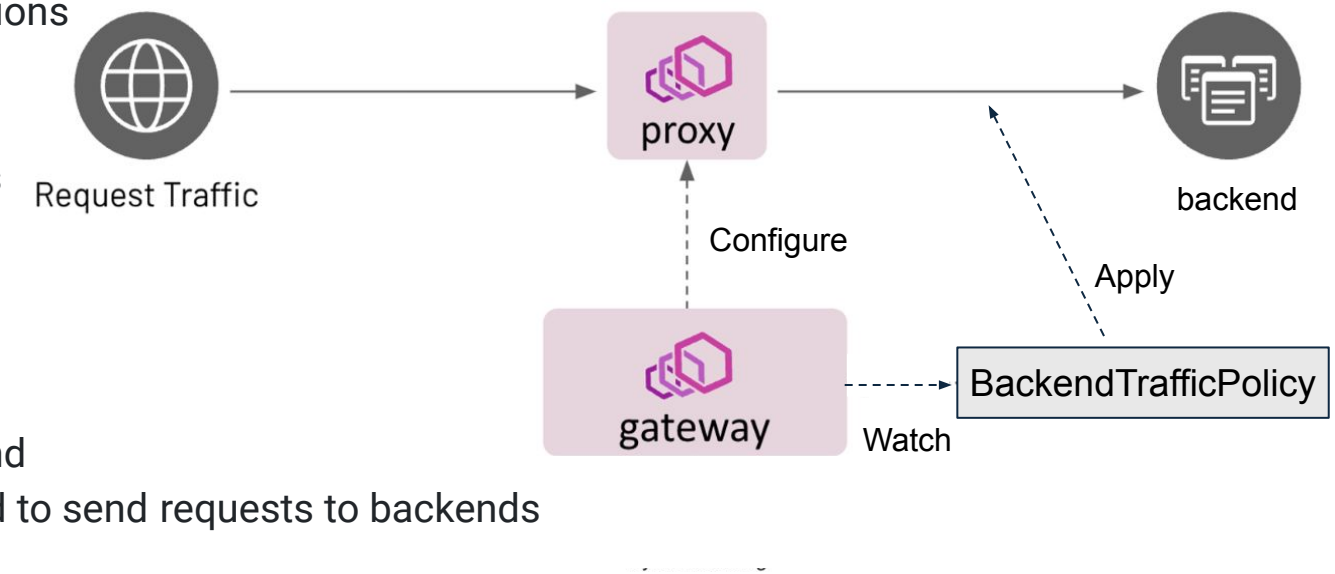
```yaml
apiVersion: gateway.envoyproxy.io/v1alpha1
kind: ClientTrafficPolicy
metadata:
  name: client-traffic-policy-gateway
spec:
  targetRefs:
    - group: gateway.networking.k8s.io
      kind: Gateway
      name: eg
  tcpKeepalive:
    idleTime: 20m
    interval: 60s
    probes: 3
  connection:
    bufferLimit: 16Ki
  clientIPDetection:
    xForwardedFor:
      numTrustedHops: 2
  timeout:
    http:
      requestReceivedTimeout: 2s
      idleTimeout: 5s
```

```yaml
apiVersion: gateway.envoyproxy.io/v1alpha1
kind: ClientTrafficPolicy
metadata:
  name: client-traffic-policy-https-listener
spec:
  targetRefs:
    - group: gateway.networking.k8s.io
      kind: Gateway
      name: eg
      sectionName: https
  tcpKeepalive:
    idleTime: 20m
    interval: 60s
    probes: 3
  connection:
    bufferLimit: 16Ki
  clientIPDetection:
    xForwardedFor:
      numTrustedHops: 2
  timeout:
    http:
      requestReceivedTimeout: 2s
      idleTimeout: 5s
  tls:
    clientValidation:
      caCertificateRefs:
        - name: client-ca
```

BackendTrafficPolicy

Traffic policy for backend connections (connections between Envoy and backend)

- Global and Local RateLimit
- Retries
- Load Balancing
  - Algorithms: ConsistentHash, LeastRequest, Random, RoundRobin
  - Support SlowStart to gradually warm up JAVA applications
- Circuit Breaker
  - Max connections/requests per connection
  - Max pending requests/parallel requests/parallel retries
- TCP Keepalive
- Socket and Connection Buffer size
- TCP and HTTP Timeout
- Active and Passive Health Check (Outlier Detection)
- Enable Proxy protocol when communicating with the backend
- Use the same HTTP protocol that the incoming request used to send requests to backends
- DNS refresh rate and TTL for DNS type backend cluster
- HTTP2 settings (For example: HTTP2 stream window size and max concurrent streams)



Please note: not all features can be applied to all xRoute types. If a targeted xRoute does not support a feature, the feature will be ignored. For example, the RateLimit setting will be ignored if the Route is a TCP Route.

BackendTrafficPolicy

Targets

- Gateway: BackendTrafficPolicy applies on all xRoute on the targeted Gateway
- xRoute: BackendTrafficPolicy applies on the specified xRoute only

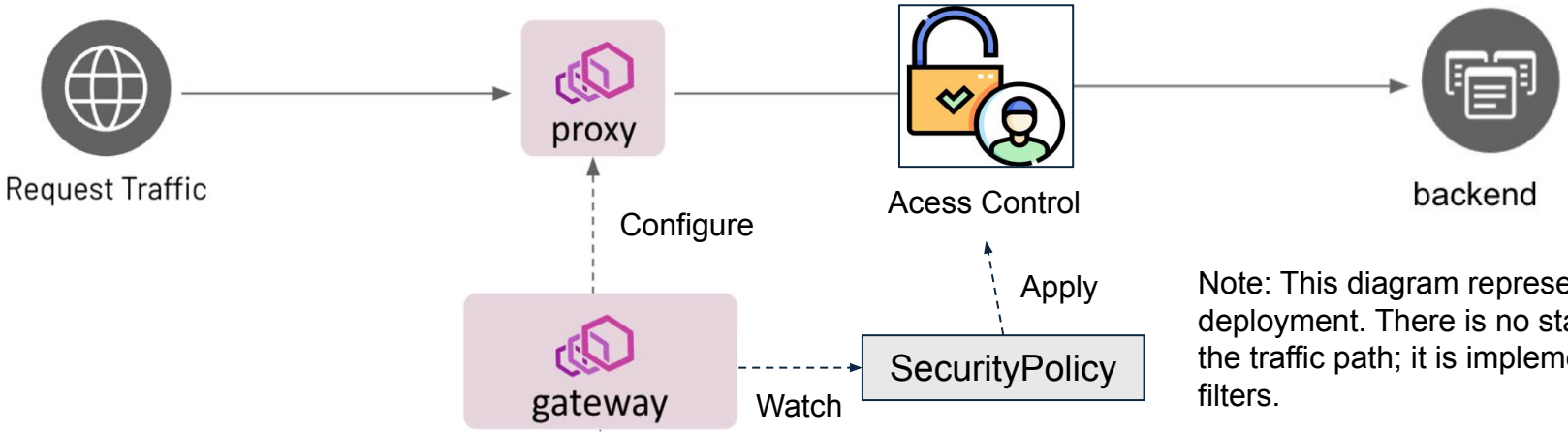
```yaml
apiVersion: gateway.networking.k8s.io/v1
kind: Gateway
metadata:
  name: eg
spec:
  gatewayClassName: internet
  listeners:
    - name: http
      protocol: HTTP
      port: 80
```

```yaml
apiVersion: gateway.networking.k8s.io/v1
kind: HTTPRoute
metadata:
  name: http-route
spec:
  parentRefs:
    - name: eg
  hostnames:
    - "foo.bar.com"
  rules:
    - backendRefs:
        - name: foo-svc
          port: 8080
```

```yaml
apiVersion: gateway.envoyproxy.io/v1alpha1
kind: BackendTrafficPolicy
metadata:
  name: backend-traffic-policy-http-route
spec:
  targetRefs:
  - group: gateway.networking.k8s.io
    kind: HTTPRoute
    name: http-route
  rateLimit:
    type: Global
    global:
      rules:
      - clientSelectors:
        - headers:
          - type: Distinct
            name: x-user-id
        limit:
          requests: 100
          unit: Second
    loadBalancer:
      type: ConsistentHash
      consistentHash:
        type: SourceIP
    circuitBreaker:
      maxPendingRequests: 1024
      maxParallelRequests: 1024
```

Security Settings for Gateway(Access Control, Authentication, and Authorization Policies)

- CORS: Access control based on the origin of the request
- Authenticaion
  - HTTP Basic Auth
  - OIDC
    - Integrate with any IdPs: Google, Auth0, Azure AD, Keycloak, Okta, etc
    - Support SSO across multiple applications
  - JWT Auth
- Authorization
  - Principal: Original request IP, JWT Claims, Basic Auth user, etc.
  - Action: allow/deny access to targeted HTTP Routes
- Ext Auth: Integrate with user provided authorization systems



Note: This diagram represents a logical structure rather than an actual deployment. There is no standalone ‘Access Control’ component in the traffic path; it is implemented as part of the Envoy using HTTP filters.

Targets

- Gateway: SecurityPolicy applies on all xRoute on the targeted Gateway
- xRoute: SecurityPolicy applies on the specified xRoute only

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

```yaml
apiVersion: gateway.networking.k8s.io/v1
kind: HTTPRoute
metadata:
  name: http-route
spec:
  parentRefs:
    - name: eg
  hostnames:
    - "foo.bar.com"
  rules:
    - backendRefs:
        - name: foo-svc
          port: 8080
```

```yaml
apiVersion: gateway.envoyproxy.io/v1alpha1
kind: SecurityPolicy
metadata:
  name: scurity-policy-http-route
spec:
  targetRefs:
    - group: gateway.networking.k8s.io
      kind: HTTPRoute
      name: http-route
  oidc:
    provider:
      issuer: "https://accounts.google.com"
    clientID: "client1.apps.googleusercontent.com"
    clientSecret:
      name: "my-app-client-secret"
    redirectURL: "https://www.example.com/myapp/oauth2/callback"
    logoutPath: "/myapp/logout"
  authorization:
    defaultAction: Deny
    rules:
    - action: Allow
      principal:
        clientCIDRs:
        - 10.0.1.0/24
```

EnvoyExtensionPolicy

Expand Envoy's functionality with custom extensions, currently supports:

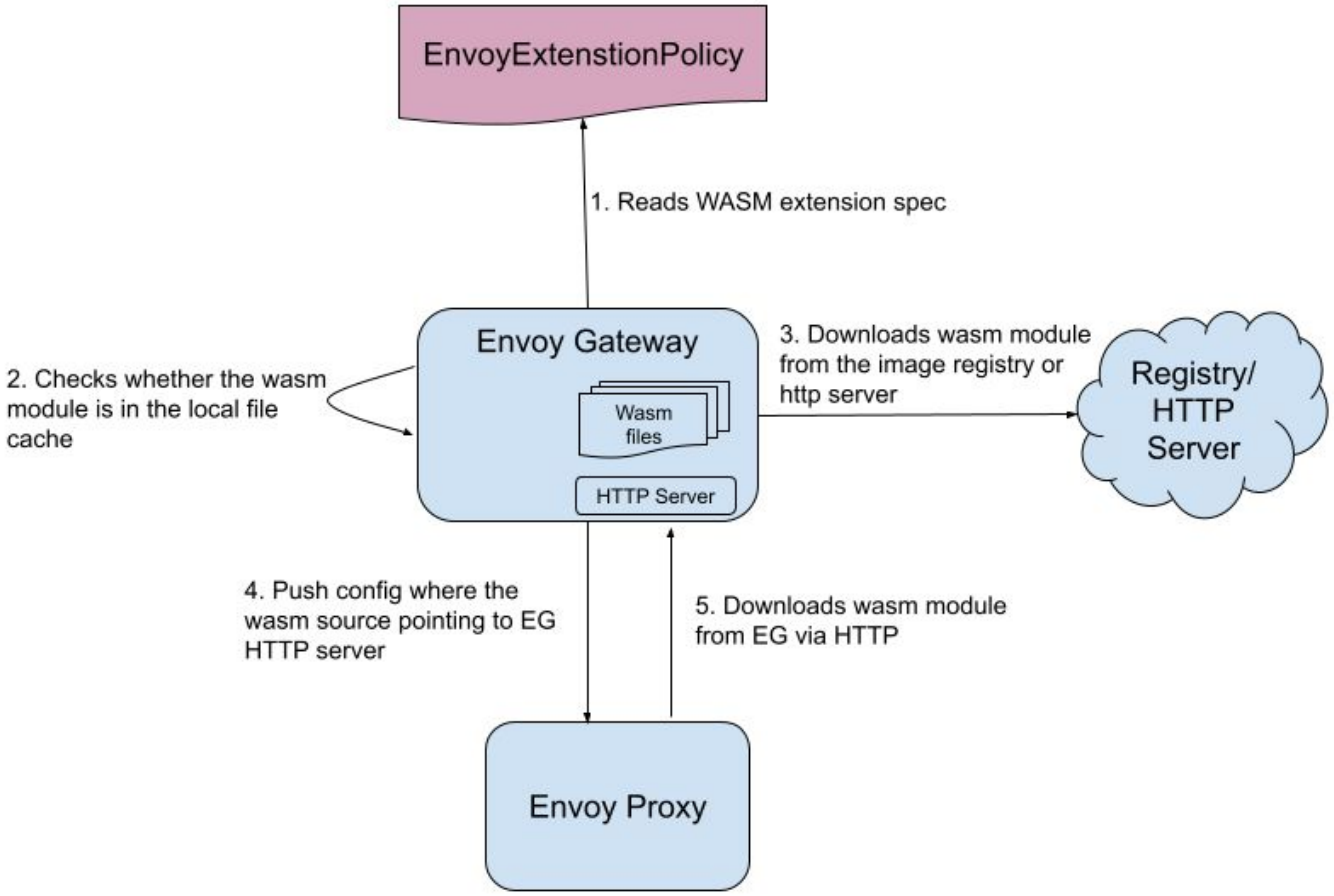
- WASM (WebAssembly)
  - Lightweight, secure, and run within Envoy’s process
  - Support OIC Image and HTTP WASM source for flexible deployment
  - Version managment and access control (OCI Image)
  - Slightly better performance and less moving parts than External Process
- External Process
  - Scalable, flexible, and isolated from Envoy’s core process
  - Independently scale external processing services
  - Use any language supporting gRPC, with no system call restrictions
  - Runs in separate processes, minimizing risk of Envoy crashes
  - Heavier than WASM: additional network calls, external process deployment

Consider selecting the appropriate extension mechanism that best aligns with your specific use

EnvoyExtensionPolicy

Envoy Gateway support Wasm OCI image as a remote wasm code source.

- **Versioning**: Users can use the tag feature of the OCI image to manage the version of the Wasm module.
- **Security**: Users can use private registries to store the Wasm module.
- **Distribution**: Users can use the existing distribution mechanism of the OCI registry to distribute the Wasm module.



EnvoyExtensionPolicy

OCI Image Wasm source

```yaml
apiVersion: gateway.envoyproxy.io/v1alpha1
kind: EnvoyExtensionPolicy
metadata:
  name: wasm-test
spec:
  targetRefs:
  - group: gateway.networking.k8s.io
    kind: HTTPRoute
    name: backend
  wasm:
  - name: wasm-filter-1
    rootID: my_root_id
    code:
      type: Image
      image:
        url: zhaohuabing/testwasm:v0.0.1
  - name: wasm-filter-2
    rootID: "my-root-id"
    code:
      type: Image
      image:
        url: oci://my.private.regisgtry/wasm-filter-2:v1.0.0
        pullSecretRef:
          name: my-pull-secret
        sha256: a1efca12ea51069abb123bf9c77889fcc2a31cc5483f
    config:
      parameter1: value1
      parameter2: value2
```

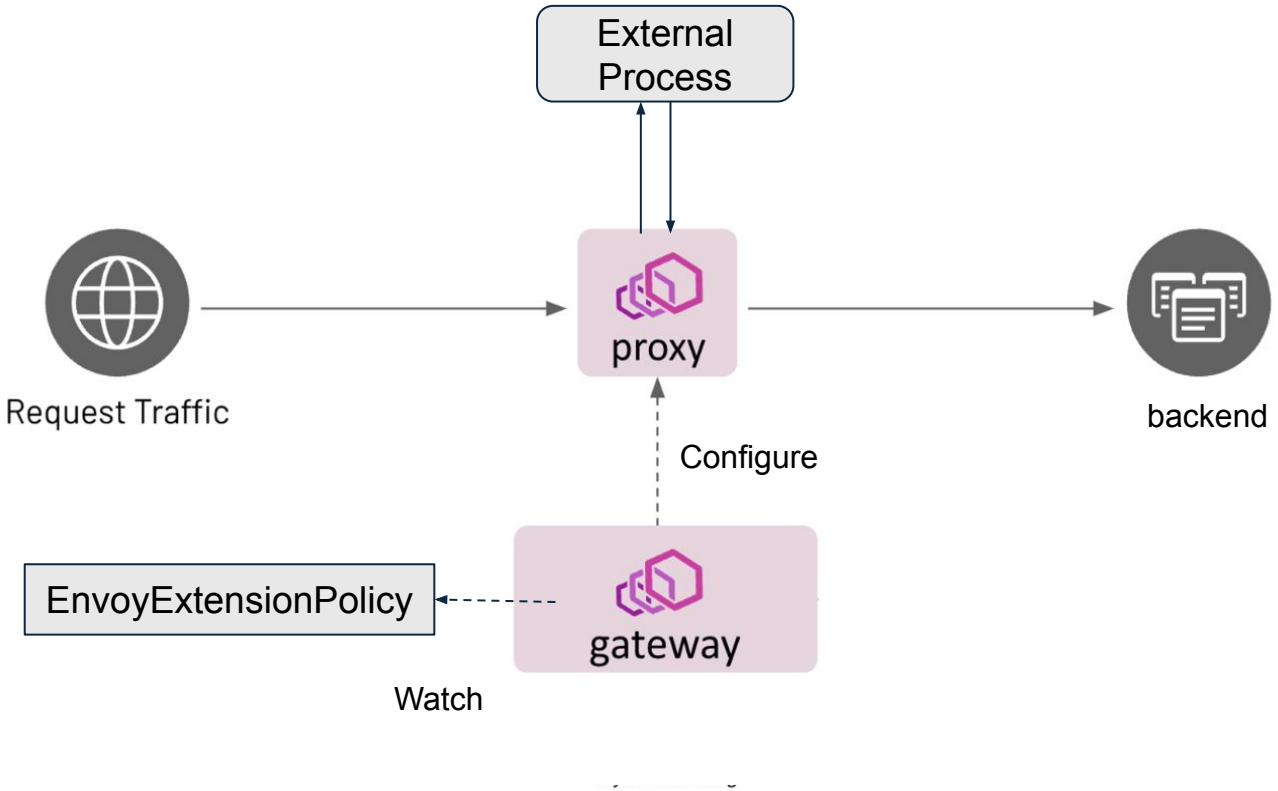
HTTP Wasm source

```yaml
apiVersion: gateway.envoyproxy.io/v1alpha1
kind: EnvoyExtensionPolicy
metadata:
  name: wasm-test
spec:
  targetRefs:
  - group: gateway.networking.k8s.io
    kind: HTTPRoute
    name: backend
  wasm:
  - name: wasm-filter-1
    code:
      type: HTTP
      http:
        url: https://www.example.com/wasm-filter-1.wasm
        sha256: 746df05c8f3a0b07a46c0967cfbc5cbe5b9d48d0f79
    config:
      parameter1:
        key1: value1
        key2: value2
      parameter2: value3
```

EnvoyExtensionPolicy

External Process Extension

Note: The deployment of the External Process is managed independently, outside the scope of Envoy Gateway (EG).

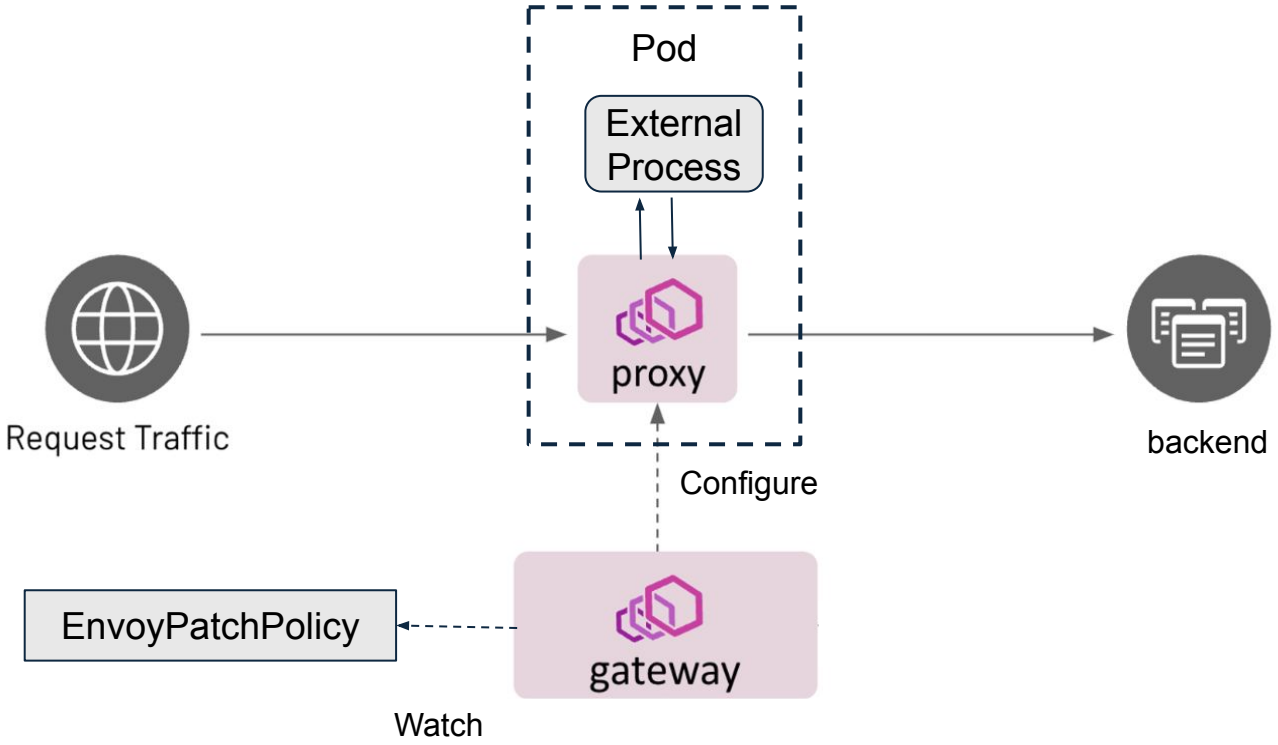



```yaml
apiVersion: gateway.envoyproxy.io/v1alpha1
kind: EnvoyExtensionPolicy
metadata:
  namespace: default
  name: policy-for-http-route
spec:
  targetRef:
    group: gateway.networking.k8s.io
    kind: HTTPRoute
    name: httproute-1
  extProc:
  - backendRefs:
    - Name: my-ext-proc-svc
      Port: 8000
    messageTimeout: 1s
    faileOpen: true
    processingMode:
      request:
        body: Buffered
      response:
        body: Buffered
```

EnvoyExtensionPolicy

External Process Extension

- The External Process can be deployed as a sidecar to minimize network latency.
- A Unix Domain Socket (UDS) type of Backend API can be used to reference the sidecar service.




```yaml
apiVersion: gateway.envoyproxy.io/v1alpha1
kind: EnvoyExtensionPolicy
metadata:
  namespace: default
  name: policy-for-http-route
spec:
  targetRef:
    group: gateway.networking.k8s.io
    kind: HTTPRoute
    name: httproute-1
  extProc:
  - backendRefs:
    - Name: uds-ext-proc
      kind: Backend
      group: gateway.envoyproxy.io
```

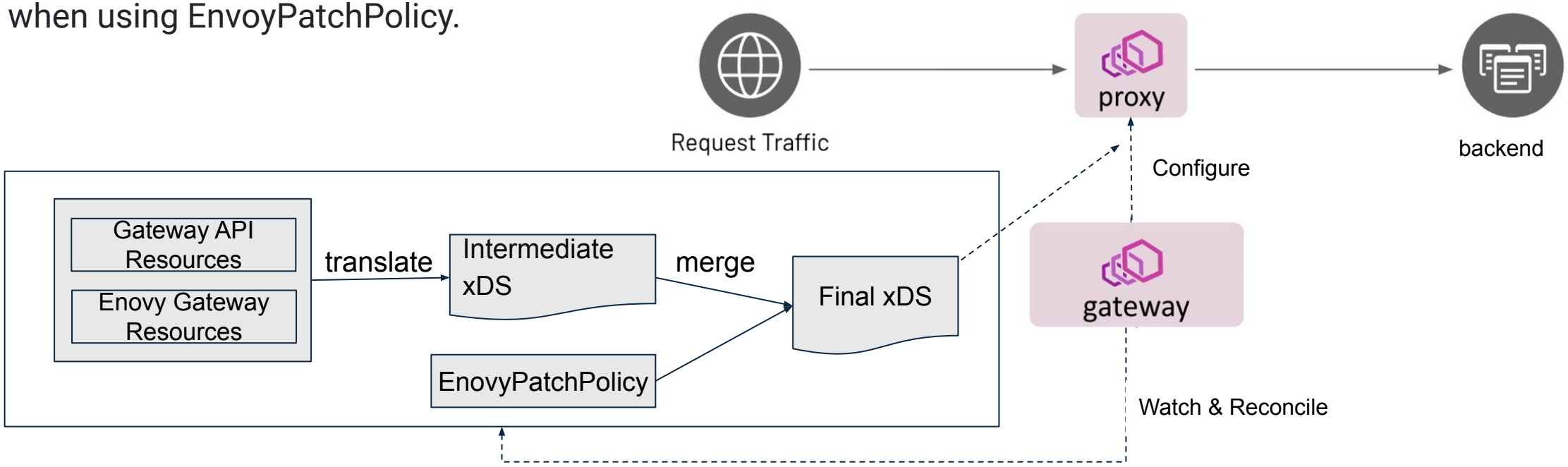
```yaml
apiVersion: gateway.envoyproxy.io/v1alpha1
kind: Backend
metadata:
  name: uds-ext-proc
spec:
  endpoints:
  - unix:
      path: /var/run/ext-proc/extproc.sock
```

EnvoyPatchPolicy

Add arbitrary patches to the generated xDS, especially useful for:

- Verifying prototype before formally landing a feature to GE API.
- Implementing temporary workarounds for features not yet supported by EG.

Caveat: The compatibility of EnvoyPatchPolicy is not guaranteed. An EnvoyPatchPolicy that functions with a specific version may not work following an upgrade due to changes in the xDS translation. Please consider this when using EnvoyPatchPolicy.



EnvoyPatchPolicy

Enable EnvoyPatchPolicy

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: envoy-gateway-config
  namespace: envoy-gateway-system
data:
  envoy-gateway.yaml: |
    apiVersion: gateway.envoyproxy.io/v1alpha1
    kind: EnvoyGateway
    provider:
      type: Kubernetes
    gateway:
      controllerName: gateway.envoyproxy.io/gatewayclass-controller
    extensionApis:
      enableEnvoyPatchPolicy: true
```

EnvoyPatchPolicy for custom response

```yaml
apiVersion: gateway.envoyproxy.io/v1alpha1
kind: EnvoyPatchPolicy
metadata:
  name: custom-response-patch-policy
  namespace: default
spec:
  targetRef:
    group: gateway.networking.k8s.io
    kind: Gateway
    name: eg
  type: JSONPatch
  jsonPatches:
    - type: "type.googleapis.com/envoy.config.listener.v3.Listener"
      # The listener name is of the form <GatewayNamespace>/<GatewayName>/<GatewayListenerName>
      name: default/eg/http
      operation:
        op: add
        path: "/default_filter_chain/filters/0/typed_config/local_reply_config"
        value:
          mappers:
          - filter:
              status_code_filter:
                comparison:
                  op: EQ
                  value:
                    default_value: 404
                    runtime_key: key_b
            status_code: 406
            body:
              inline_string: "could not find what you are looking for"
```

Backend

Supports non-k8s backends for xRoute and xPolicy

- **IP:** IP based TCP/UDP socket address, for example: *10.0.0.1:8080*
- **FQDN**: Hostname based TCP/UDP socket address, for example: *foo.bar.com:443*
- **UDS**: Unix Domain socket address, for example: */var/run/ext-proc/extproc.sock*

Enable Backend API

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: envoy-gateway-config
  namespace: envoy-gateway-system
data:
  envoy-gateway.yaml: |
    apiVersion: gateway.envoyproxy.io/v1al
    kind: EnvoyGateway
    provider:
      type: Kubernetes
    gateway:
      controllerName: gateway.envoyproxy.i
    extensionApis:
      enableEnvoyPatchPolicy: true
```

Define a Backend pointing to httpbin.org

```yaml
apiVersion: gateway.envoyproxy.io/v1alpha1
kind: Backend
metadata:
  name: httpbin
  namespace: default
spec:
  endpoints:
    - fqdn:
        hostname: httpbin.org
        port: 80
```

Define a HTTPRoute that references this Backend

```yaml
apiVersion: gateway.networking.k8s.io/v1
kind: HTTPRoute
metadata:
  name: backend
spec:
  parentRefs:
    - name: eg
  hostnames:
    - "www.example.com"
  rules:
    - backendRefs:
        - group: gateway.envoyproxy.io
          kind: Backend
          name: httpbin
      matches:
        - path:
            type: PathPrefix
            value: /
```

What do we expect for v1.2.0 ?

- Deploy EG on non-k8s environment
- Control plane memory usage optimization
- More authorization capabilities: authorization based on JWT claim, Basic Auth user, HTTP headers, …
- OIDC enhancement on upstream and EG: retries, sub-domain token sharing, single logout, state/nonce support …
- Any features that you want? - raise an issue on Github and join the community meeting to discuss

Thanks To All The Contributors!

KubeCon | CloudNativeCon | THE LINUX FOUNDATION OPEN SOURCE SUMMIT | AI_dev

China 2024

Get Involved!

Docs: gateway.envoyproxy.io

Project: github.com/envoyproxy/gateway

微信：联系 zhao_huabing 加入 EG 中国社区微信群

Slack channel

Community meeting