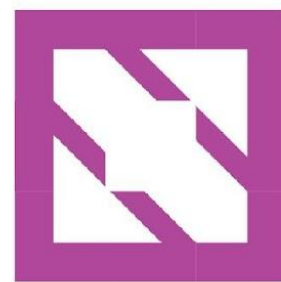




KubeCon



CloudNativeCon

Europe 2025



Securing the Gateway



KubeCon



CloudNativeCon

Europe 2025

A Deep Dive into Envoy Gateway's Advanced Security Policy

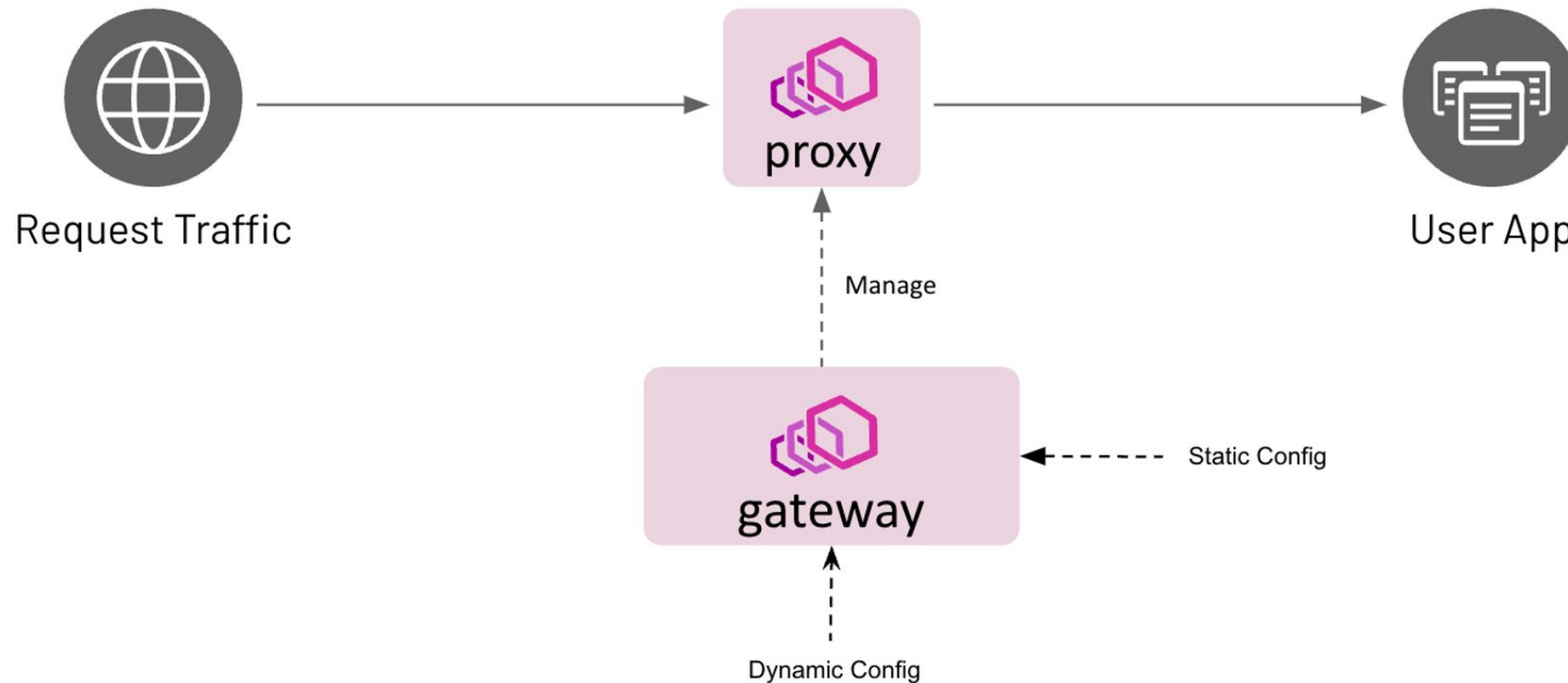
Robin Zhao, Tetrade

Envoy Gateway Maintainer
Envoy Contributor



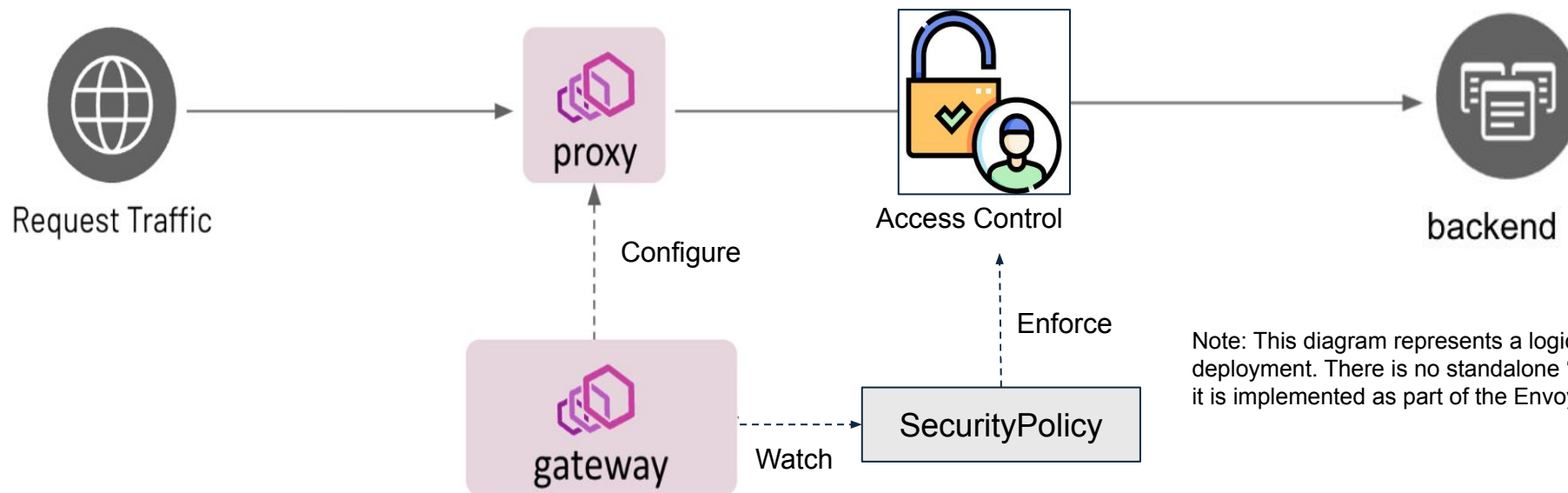
Background: Envoy Gateway

- **Envoy Gateway** is an open source project for managing **Envoy Proxy** as a standalone or Kubernetes-based application gateway.
- **Kubernetes Gateway API** resources are used to automatically provision and configure the managed Envoy Proxies.



SecurityPolicy - Secure your Gateway

Security Policy is a Gateway API extension to provide advanced security features for the ingress traffic.



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.

Secure Your Gateway with Envoy Gateway's Security Policy

CORS (Cross-Origin Resource Sharing)

- Restrict or allow access **based on request origin** to prevent unauthorized cross-site requests.

Authentication - Verify Who's Accessing

-  **HTTP Basic Auth** – Simple username/password protection.
-  **JWT (JSON Web Token)** – Secure, token-based authentication.
-  **API Key Authentication** – Control access with API keys.
-  **OIDC (OpenID Connect)** – Federated authentication with **Google, AWS, Azure AD, Auth0, Keycloak, Okta,** and more.

Authorization - Control What Users Can Do

-  **Principal** – Identify users based on **IP, JWT claims, Basic Auth credentials, HTTP header and method, etc.**
-  **Action** – **Allow/Deny** access to specific **HTTP routes** based on rules.

 **Ext Auth** – Seamlessly integrate with **custom authorization systems** for customized security requirements.

How SecurityPolicy Works

Targets

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

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

```
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
```

```
apiVersion: gateway.envoyproxy.io/v1alpha1
kind: SecurityPolicy
metadata:
  name: security-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:
          clientCIDs:
            - 10.0.1.0/24
```

SecurityPolicy Example

CORS

```
apiVersion: gateway.envoyproxy.io/v1alpha1
kind: SecurityPolicy
metadata:
  name: cors-example
spec:
  targetRefs:
  - group: gateway.networking.k8s.io
    kind: HTTPRoute
    name: backend
  cors:
    allowOrigins:
    - "http://*.foo.com"
    - "http://*.foo.com:80"
    allowMethods:
    - GET
    - POST
    allowHeaders:
    - "x-header-1"
    - "x-header-2"
    exposeHeaders:
    - "x-header-3"
    - "x-header-4"
```

Basic Auth

```
apiVersion: gateway.envoyproxy.io/v1alpha1
kind: SecurityPolicy
metadata:
  name: basic-auth-example
spec:
  targetRefs:
  - group: gateway.networking.k8s.io
    kind: HTTPRoute
    name: backend
  basicAuth:
    users:
    - kind: Secret
      name: "basic-auth"
```

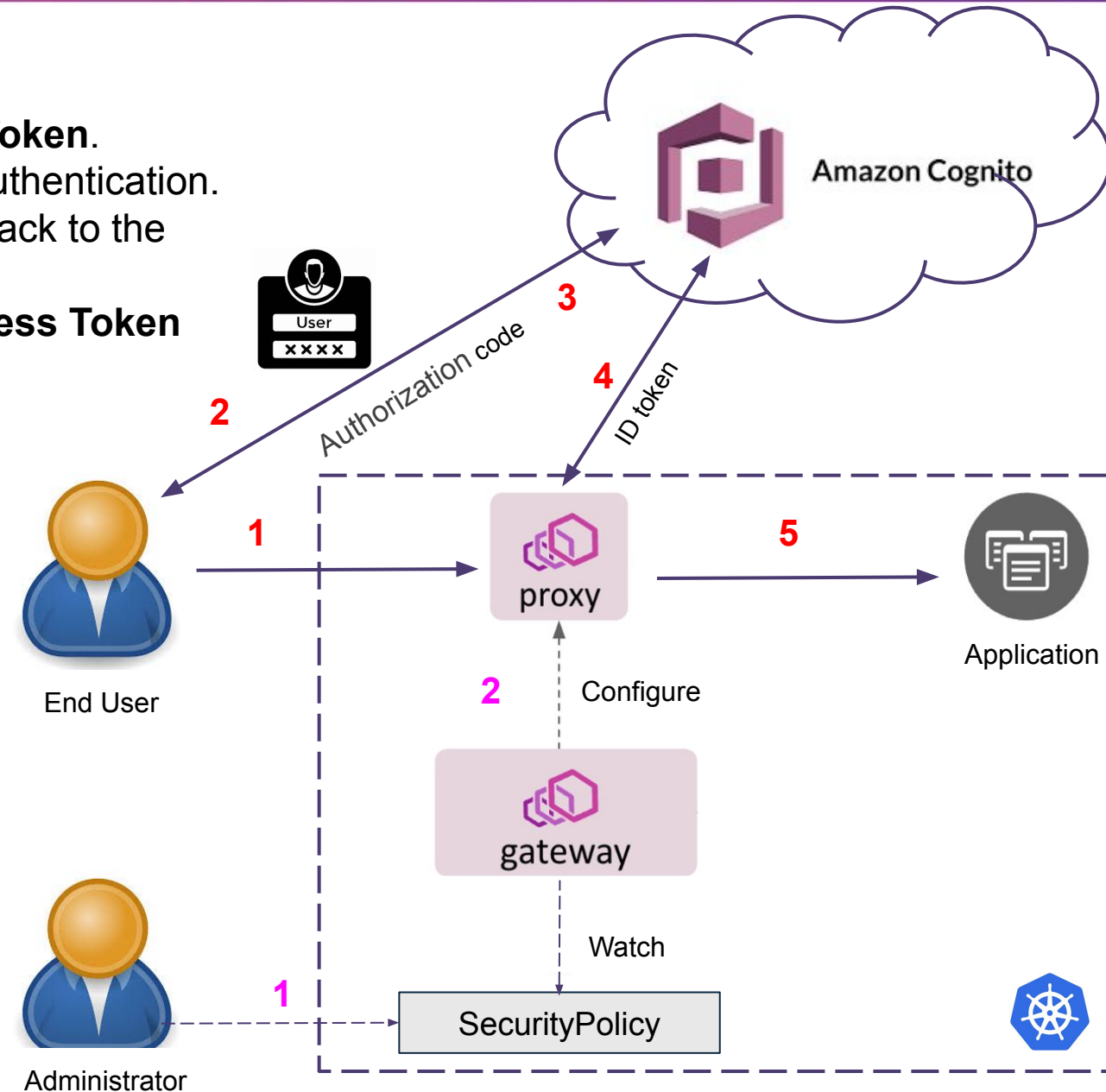
API Key Auth

```
apiVersion: gateway.envoyproxy.io/v1alpha1
kind: SecurityPolicy
metadata:
  name: apikey-auth-example
spec:
  targetRefs:
  - group: gateway.networking.k8s.io
    kind: HTTPRoute
    name: backend
  apiKeyAuth:
    credentialRefs:
    - group: ""
      kind: Secret
      name: apikey-secret
    extractFrom:
    - headers:
      - x-api-key
```

Demo: OIDC Auth with Amazon Cognito

Data Plane:

1. The user sends a request to the **Envoy** without an **ID Token**.
2. The user is redirected to the **Cognito login page** for authentication.
3. After successful authentication, the user is redirected back to the Envoy with an **Authorization Code**.
4. **Envoy** exchanges the **Authorization Code** for an **Access Token** and **ID Token**.
5. The user's request is then proxied to the application.



Demo: JWT Claim Authorization

- The **ID Token** is a **JWT token** that contains claims about the authenticated user.
- These claims can be used to enforce **fine-grained access control** at the HTTPRoute level.

(Note: This applies not only to **OIDC**, but to any **JWT token**, and it also supports authorization on JWT scopes.

JSON CLAIMS TABLE

COPY

```
{
  "at_hash": "4t5MMYUI9I0cuchiM9Y0gw",
  "sub": "89eed458-a031-70a2-f079-21ce7ccae5f0",
  "iss": "https://cognito-idp.ap-southeast-2.amazonaws.com/ap-southeast-2_ewsdqjtaD",
  "cognito:username": "89eed458-a031-70a2-f079-21ce7ccae5f0",
  "origin_jti": "78e6f428-c8c1-49ad-b02c-a75e8416f3d3",
  "aud": "3su394h1en0hpdfd861ddh9fkd",
  "event_id": "c07859dc-f7b6-47bb-93a2-f65af707f046",
  "token_use": "id",
  "auth_time": 1741148685,
  "exp": 1741152285,
  "iat": 1741148685,
  "jti": "f603f9a5-1664-4d2e-b996-1b6ddf1cb5d5",
  "email": "zhaohuabing@gmail.com"
}
```

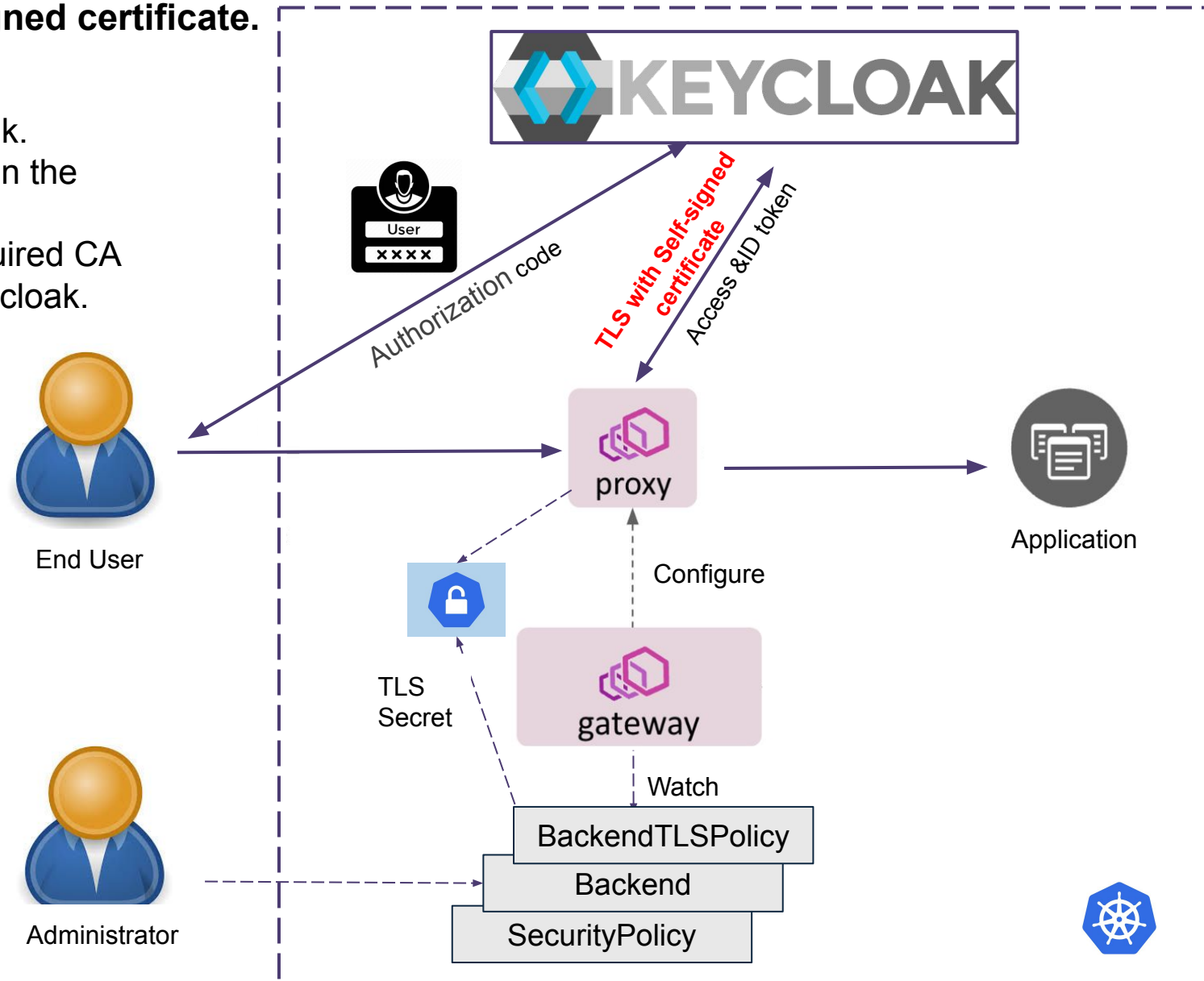
```
apiVersion: gateway.envoyproxy.io/v1alpha1
kind: SecurityPolicy
metadata:
  name: oidc-example
spec:
  targetRefs: ...
  oidc: ...
  jwt: ...
  authorization:
    defaultAction: Deny
    rules:
      - action: Allow
        name: allow
        principal:
          jwt:
            provider: aws-cognito
            claims:
              - name: sub
                values: [89eed458-a031-70a2-f079-21ce7ccae5f0]
              - name: email
                values: [zhaohuabing@gmail.com]
```

Demo: Work with Self-Signed Certificate

Keycloak is deployed as an IDP using a self-signed certificate.

To connect to it:

1. Create a **Backend** resource pointing to Keycloak.
2. Reference that **Backend** as the OIDC provider in the **SecurityPolicy**.
3. Use a **BackendTLSPolicy** to configure the required CA certificate for establishing a TLS connection with Keycloak.



Note: In this demo, a Backend resource is not a must because Keycloak is deployed inside the cluster. However, if Keycloak is deployed outside the cluster, you *will* need to define a Backend to connect to it properly.

OIDC Provider with Self-Signed Certificate



KubeCon



CloudNativeCon

Europe 2025

- Use the **Backend** API to reference to a Keycloak deployed within the cluster.
- Use the **BackendTLSPolicy** API to configure CA for TLS.

```
apiVersion: gateway.envoyproxy.io/v1alpha1
kind: SecurityPolicy
metadata:
  name: oidc-example
spec:
  targetRefs:
  - group: gateway.networking.k8s.io
    kind: HTTPRoute
    name: myapp
  oidc:
    provider:
      backendRefs:
      - group: gateway.envoyproxy.io
        kind: Backend
        name: backend-keycloak
        port: 443
      issuer: "https://my.keycloak.com/realms/master"
      authorizationEndpoint: "https://my.keycloak.com/realms/
      tokenEndpoint: "https://my.keycloak.com/realms/master/p
    clientID: "${CLIENT_ID}"
    clientSecret:
      name: "my-app-client-secret"
    redirectURL: "http://www.example.com/myapp/oauth2/callback
    logoutPath: "/myapp/logout"
```

```
apiVersion: gateway.envoyproxy.io/v1alpha1
kind: Backend
metadata:
  name: backend-keycloak
spec:
  endpoints:
  - fqdn:
      hostname: 'my.keycloak.com'
      port: 443
---
apiVersion: gateway.networking.k8s.io/v1alpha3
kind: BackendTLSPolicy
metadata:
  name: policy-btls
spec:
  targetRefs:
  - group: gateway.envoyproxy.io
    kind: Backend
    name: backend-keycloak
  validation:
    caCertificateRefs:
    - name: backend-tls-certificate
      group: ""
      kind: ConfigMap
      hostname: my.keycloak.com
```

Thank you

- CORS
- Basic Authn
- API Key Authn
- JWT Authn
- OIDC Authn
- Client IP Authz
- JWT Claim/Scope Authz
- HTTP Header/Method Authz



attend **Docker's Envoy Gateway Journey**

15:00 - 15:30 Level 3 | ICC Capital Suite 14-16

Join us in the envoy **#gateway-users** slack channel