Envoy-Proxy Gateway

Mehul Jaiswal

June 2024

1 Introduction to Envoy Proxy

1.1 What is Envoy?

Envoy is an open-source edge and service proxy designed for cloud-native applications. It acts as an intermediary in your network that receives requests from clients and forwards them to the appropriate services. Envoy can perform load balancing, health checking, service discovery, and observability. It's highly extensible through filters and supports dynamic configuration through APIs.

1.2 Why Use Envoy?

- **High Performance:** Envoy is optimized for performance with low latency and high throughput.
- **Resilience:** Supports retries, timeouts, circuit breaking, and fault injection to handle failure scenarios gracefully.
- **Observability:** Rich metrics, logging, and tracing capabilities to monitor and debug applications.
- **Extensibility:** Can be extended through filters to add custom processing logic.
- **Dynamic Configuration:** Supports runtime configuration updates without needing restarts.

2 Static Configuration

2.1 Overview

Static configuration involves setting up Envoy with a predefined configuration file. This configuration defines all the necessary components, such as listeners, routes, clusters, and endpoints. This approach is straightforward and suitable for environments with minimal changes.

2.2 Basic Concepts

2.2.1 Listeners

A listener in Envoy defines where and how Envoy receives incoming traffic. It specifies the IP address and port on which Envoy listens for incoming connections. Each listener can have multiple filter chains to process the incoming requests.

2.2.2 Routes

Routes define how Envoy matches incoming requests to the appropriate upstream services. A route can be configured with various match criteria, such as prefixes, headers, or specific paths, and then directs the request to a specified cluster.

2.2.3 Clusters

Clusters are groups of logically similar endpoints. A cluster typically represents a single service and can contain multiple endpoints (instances of the service). Clusters define how Envoy connects to the upstream services and can include load balancing, health checks, and circuit breakers.

2.2.4 Endpoints

Endpoints are the actual instances of services within a cluster. Each endpoint includes an IP address and a port where the service instance is running.

2.3 Example Configuration

Here's a detailed example of a simple static configuration file in YAML:

```yaml
static_resources:
  listeners:
    - name: listener_0
      address:
        socket_address:
          address: 0.0.0.0  # Bind to all IP addresses
          port_value: 10000  # Listening on port 10000
      filter_chains:
        - filters:
            - name: envoy.filters.network.http_connection_manager
              typed_config:
                "@type": type.googleapis.com/envoy.extensions.filters.network.http_connection_ma
                stat_prefix: ingress_http  # Prefix for statistics
                route_config:
                  name: local_route
                  virtual_hosts:
                    - name: local_service
                      domains: ["*"]  # Match all domains
                      routes:
                        - match: { prefix: "/" }  # Match all paths
                          route: { cluster: service_cluster }
                http_filters:
                  - name: envoy.filters.http.router
                    typed_config: {}
  clusters:
    - name: service_cluster
      connect_timeout: 0.25s  # Connection timeout
      type: strict_dns  # Resolve service.example.com using DNS
      lb_policy: round_robin  # Use round-robin load balancing
      load_assignment:
        cluster_name: service_cluster
        endpoints:
          - lb_endpoints:
              - endpoint:
                  address:
                    socket_address:
                      address: service.example.com  # Service address
                      port_value: 80  # Service port
```

2.4 Detailed Explanation

2.4.1 Listeners

- **name: listener_0:** A unique name for the listener.

- **address:** Specifies the IP address and port.
  - **socket_address:** Defines a socket address.
  - **address: 0.0.0.0:** Bind to all available IP addresses.
  - **port_value: 10000:** Listen on port 10000.
- **filter_chains:** Defines a set of filters for processing incoming connections.
  - **filters:** List of filters applied to incoming requests.
  - **name: envoy.filters.network.http_connection_manager:** An HTTP connection manager filter.
  - **typed_config:** Configuration for the HTTP connection manager.
  - **stat_prefix: ingress_http:** Prefix for statistics.
  - **route_config:** Configuration for routing.
    - **name: local_route:** A unique name for the route.
    - **virtual_hosts:** Defines virtual hosts.
      - **name: local_service:** A unique name for the virtual host.
      - **domains: ["*"]:** Match all domains.
      - **routes:** List of routes.
      - **match: { prefix: "/" }:** Match all paths.
      - **route: { cluster: service_cluster }:** Route to the service_cluster.
  - **http_filters:** List of HTTP filters.
    - **name: envoy.filters.http.router:** A router filter for handling routes.
    - **typed_config: {}:** Configuration for the router filter.

2.4.2 Clusters

- **name: service_cluster:** A unique name for the cluster.
- **connect_timeout: 0.25s:** Timeout for connecting to the service.
- **type: strict_dns:** Use DNS to resolve the service address.
- **lb_policy: round_robin:** Use round-robin load balancing.
- **load_assignment:** Specifies the load assignment for the cluster.
  - **cluster_name: service_cluster:** Name of the cluster.
  - **endpoints:** List of endpoints.
    - **lb_endpoints:** Load balancing endpoints.
      - **endpoint:** Endpoint configuration.
      - **address:** Address of the endpoint.
      - **socket_address:** Defines a socket address.
      - **address: service.example.com:** Service address.
      - **port_value: 80:** Service port.

3 Dynamic Configuration

3.1 Overview

Dynamic configuration allows Envoy to update its configuration at runtime without needing to restart. This is crucial for dynamic environments where services are frequently added or removed. Envoy uses the xDS APIs (LDS, RDS, CDS, EDS) to fetch and update configurations dynamically.

3.2 xDS APIs

- **LDS (Listener Discovery Service):** For discovering listeners.
- **RDS (Route Discovery Service):** For discovering routes.
- **CDS (Cluster Discovery Service):** For discovering clusters.
- **EDS (Endpoint Discovery Service):** For discovering endpoints.
- **SDS (Secret Discovery Service):** For discovering TLS secrets.

3.3 Control Plane Integration

A control plane like Istio, Consul, or a custom-built one manages Envoy's dynamic configuration. The control plane communicates with Envoy via the xDS APIs to provide updated configurations.

3.4 Example Configuration for Dynamic Configuration

A partial dynamic configuration using ADS (Aggregated Discovery Service):

```
dynamic_resources:
  ads_config:
    api_type: GRPC
    grpc_services:
      - envoy_grpc:
          cluster_name: xds_cluster
  cds_config: { resource_api_version: V3 }
  lds_config: { resource_api_version: V3 }

static_resources:
  clusters:
    - name: xds_cluster
      connect_timeout: 0.25s
      type: strict_dns
      lb_policy: round_robin
      http2_protocol_options: {}
      load_assignment:
        cluster_name: xds_cluster
        endpoints:
          - lb_endpoints:
              - endpoint:
                  address:
                    socket_address:
                      address: xds-server.example.com
                      port_value: 50000
```

3.5 Detailed Explanation

3.5.1 dynamic_resources

- **ads_config:** Configuration for ADS (Aggregated Discovery Service).
  - **api_type: GRPC:** Use gRPC for communication.
  - **grpc_services:** List of gRPC services.
    - **envoy_grpc:** Configuration for Envoy's gRPC service.
    - **cluster_name: xds_cluster:** Name of the cluster used for ADS.

- **cds_config:** Configuration for CDS (Cluster Discovery Service).
  - **resource_api_version: V3:** Use version 3 of the API.
- **lds_config:** Configuration for LDS (Listener Discovery Service).
  - **resource_api_version: V3:** Use version 3 of the API.

3.5.2 static_resources

- **clusters:** List of static clusters.
  - **name: xds_cluster:** A unique name for the xDS cluster.
  - **connect_timeout: 0.25s:** Timeout for connecting to the xDS server.
  - **type: strict_dns:** Use DNS to resolve the xDS server address.
  - **lb_policy: round_robin:** Use round-robin load balancing.
  - **http2_protocol_options: {}:** Enable HTTP/2 for the connection.
  - **load_assignment:** Specifies the load assignment for the cluster.
    - **cluster_name: xds_cluster:** Name of the cluster.
    - **endpoints:** List of endpoints.
      - **lb_endpoints:** Load balancing endpoints.
      - **endpoint:** Endpoint configuration.
      - **address:** Address of the endpoint.
      - **socket_address:** Defines a socket address.
      - **address: xds-server.example.com:** xDS server address.
      - **port_value: 50000:** xDS server port.

4 Advanced Topics

4.1 Filter Chains

Envoy filters process requests and responses. Filters can manipulate headers, perform rate limiting, apply authentication, etc.

4.1.1 Example of Filter Chains

```
filters:
  - name: envoy.filters.network.http_connection_manager
    typed_config:
      "@type": type.googleapis.com/envoy.extensions.filters.network.http_connection_manager.v3.H
      http_filters:
        - name: envoy.filters.http.jwt_authn
          typed_config:
            "@type": type.googleapis.com/envoy.extensions.filters.http.jwt_authn.v3.JwtAuthentic
            providers:
              my_jwt_provider:
```

```
      issuer: https://example.com/
      local_jwks:
        inline_string: <JWKS-STRING>
      from_headers:
        - name: "Authorization"
          value_prefix: "Bearer␣"
```

4.1.2 Detailed Explanation

- **filters:** List of network filters.
  - **name: envoy.filters.network.http_connection_manager:** HTTP connection manager filter.
  - **typed_config:** Configuration for the HTTP connection manager.
    - **http_filters:** List of HTTP filters.
      - **name: envoy.filters.http.jwt_authn:** JWT authentication filter.
      - **typed_config:** Configuration for the JWT authentication filter.
      - **providers:** List of JWT providers.
      - **my_jwt_provider:** Configuration for a JWT provider.
      - **issuer: https://example.com/:** JWT issuer.
      - **local_jwks:** Local JWKS (JSON Web Key Set) configuration.
      - **inline_string: ¡JWKS-STRING¿:** Inline JWKS string.
      - **from_headers:** List of headers to extract the JWT from.
      - **name: "Authorization":** Name of the header.
      - **value_prefix: "Bearer ":** Prefix for the JWT token.

4.2 Rate Limiting

Envoy supports rate limiting via external rate limit service integration.

4.2.1 Example of Rate Limiting

```
filters:
  - name: envoy.filters.network.http_connection_manager
    typed_config:
      "@type": type.googleapis.com/envoy.extensions.filters.network.http_connection_manager.v3.H
      http_filters:
        - name: envoy.filters.http.ratelimit
          typed_config:
            "@type": type.googleapis.com/envoy.extensions.filters.http.ratelimit.v3.RateLimit
            domain: my_service
            rate_limit_service:
              grpc_service:
                envoy_grpc:
                  cluster_name: rate_limit_cluster
```

4.2.2 Detailed Explanation

- **filters:** List of network filters.
  - **name: envoy.filters.network.http_connection_manager:** HTTP connection manager filter.
  - **typed_config:** Configuration for the HTTP connection manager.
    - **http_filters:** List of HTTP filters.
      - **name: envoy.filters.http.ratelimit:** Rate limiting filter.
      - **typed_config:** Configuration for the rate limiting filter.
      - **domain: my_service:** Domain for rate limiting.
      - **rate_limit_service:** Configuration for the rate limit service.
      - **grpc_service:** gRPC service configuration.
      - **envoy_grpc:** Configuration for Envoy's gRPC service.
      - **cluster_name: rate_limit_cluster:** Name of the rate limit cluster.

4.3 Fault Injection

Fault injection allows testing of resilience by introducing faults like delays or aborts.

4.3.1 Example of Fault Injection

```
filters:
  - name: envoy.filters.network.http_connection_manager
    typed_config:
      "@type": type.googleapis.com/envoy.extensions.filters.network.http_connection_manager.v3.H
      http_filters:
        - name: envoy.filters.http.fault
          typed_config:
            "@type": type.googleapis.com/envoy.extensions.filters.http.fault.v3.HTTPFault
            delay:
              type: fixed
              fixed_delay: 5s
              percentage: { numerator: 50, denominator: "HUNDRED" }
```

4.3.2 Detailed Explanation

- **filters:** List of network filters.
  - **name: envoy.filters.network.http_connection_manager:** HTTP connection manager filter.
  - **typed_config:** Configuration for the HTTP connection manager.
    - **http_filters:** List of HTTP filters.
      - **name: envoy.filters.http.fault:** Fault injection filter.
      - **typed_config:** Configuration for the fault injection filter.
      - **delay:** Configuration for injecting delays.
      - **type: fixed:** Use a fixed delay.
      - **fixed_delay: 5s:** Introduce a delay of 5 seconds.
      - **percentage: { numerator: 50, denominator: "HUNDRED" }:** Apply the delay to 50% of the requests.

4.4 Circuit Breakers

Envoy supports circuit breaking to handle failure scenarios gracefully.

4.4.1 Example of Circuit Breakers

```
clusters:
  - name: service_cluster
    circuit_breakers:
      thresholds:
        - priority: default
          max_connections: 1000
          max_pending_requests: 1000
          max_requests: 1000
          max_retries: 3
```

4.4.2 Detailed Explanation

- **clusters:** List of clusters.
  - **name: service_cluster:** A unique name for the cluster.
  - **circuit_breakers:** Configuration for circuit breakers.
    - **thresholds:** List of thresholds for circuit breaking.
      - **priority: default:** Default priority.
      - **max_connections: 1000:** Maximum number of connections.
      - **max_pending_requests: 1000:** Maximum number of pending requests.
      - **max_requests: 1000:** Maximum number of requests.
      - **max_retries: 3:** Maximum number of retries.

4.5 Observability

Envoy provides extensive observability features including logging, metrics, and tracing.

4.5.1 Metrics

Envoy exposes a rich set of metrics in Prometheus format. These metrics include information about requests, responses, connections, and more. Metrics can be scraped by Prometheus and visualized using Grafana.

4.5.2 Logging

Envoy supports structured logging for access logs and error logs. Logs can be configured to include various details such as request and response headers, response codes, and more.

4.5.3 Tracing

Envoy integrates with distributed tracing systems like Jaeger and Zipkin. Tracing helps in understanding the flow of requests through the system and identifying performance bottlenecks.

4.6 Security

Envoy supports various security features:

- **mTLS:** Mutual TLS for secure communication between services.
- **JWT Authentication:** Verify JWT tokens.
- **RBAC:** Role-based access control for fine-grained access control.

4.6.1 Example of mTLS Configuration

```
transport_socket:
  name: envoy.transport_sockets.tls
  typed_config:
    "@type": type.googleapis.com/envoy.extensions.transport_sockets.tls.v3.UpstreamTlsContext
    common_tls_context:
      tls_certificates:
        - certificate_chain:
            filename: "/etc/envoy/tls/tls.crt"
          private_key:
            filename: "/etc/envoy/tls/tls.key"
      validation_context:
        trusted_ca:
          filename: "/etc/envoy/tls/ca.crt"
```

4.6.2 Detailed Explanation

- **transport_socket:** Configuration for the transport socket.
  - **name: envoy.transport_sockets.tls:** TLS transport socket.
  - **typed_config:** Configuration for the TLS transport socket.
    - **@type:** Type URL for the TLS context.
    - **common_tls_context:** Common TLS context configuration.
      - **tls_certificates:** List of TLS certificates.
      - **certificate_chain:** Configuration for the certificate chain.
      - **filename: "/etc/envoy/tls/tls.crt":** Path to the certificate file.
      - **private_key:** Configuration for the private key.
      - **filename: "/etc/envoy/tls/tls.key":** Path to the private key file.
      - **validation_context:** Configuration for validating the TLS context.
      - **trusted_ca:** Configuration for the trusted CA.
      - **filename: "/etc/envoy/tls/ca.crt":** Path to the CA certificate file.

5 Conclusion

Envoy is a powerful proxy that supports both static and dynamic configurations, making it suitable for a wide range of use cases from simple setups to complex, dynamic environments. By understanding the basics of static configuration and the advanced capabilities provided by dynamic configuration and xDS APIs, you can effectively leverage Envoy in your infrastructure to achieve high performance, resilience, and observability.