EVOY - Open source edge and service proxy, designed for cloud-native applications

Filip Kitka, Mateusz Furga, Norbert Morawski, Łukasz Wala
Group 6 (Thursday 13:15) — Year: 2024

Kraków, April 4, 2024

Contents

1 Introduction

Envoy proxy is an open-source edge and service proxy designed for cloud-native applications. It is widely used in modern microservices architectures and service mesh implementations.

Envoy is built to be highly performant and extensible, with features like dynamic service discovery, load balancing, TLS termination, HTTP/2 and gRPC proxying, circuit breaking, retries, rate limiting, observability, and more. It is often deployed as a sidecar alongside each service instance in a microservices architecture, facilitating communication between services and providing various traffic management functionalities.

2 Theoretical background/technology stack

This chapter will describe some of the core concepts behind the Envoy Proxy, as well as some other technologies used in the case study.

2.1 Envoy Proxy

Envoy proxy is a high-performance open-source project developed for modern cloud-native architectures, particularly microservices. It acts as a sidecar proxy, meaning it runs alongside applications and handles communication between them.

Key features of Envoy proxy:

1. **L7 traffic management:** Envoy can route traffic based on various factors like URLs, headers, and more. It can also handle tasks like load balancing, which distributes traffic evenly across different service instances, and circuit breaking, which prevents overloading services.

2. **Security:** Envoy provides features like TLS termination for encryption and automatic service discovery to ensure applications connect to the correct services securely.

3. **Observability:** Envoy offers extensive monitoring and logging capabilities, making it easier to troubleshoot network issues and gain insights into service communication.

Envoy possible use cases:

1. **Service mesh:** Envoy is a popular choice for building service meshes, which provide a dedicated infrastructure layer for managing communication between services.

2. **API Gateway:** Envoy can act as an API gateway, a single entry point for external clients to access multiple backend services.

2.2 Service mesh

A service mesh is a dedicated infrastructure layer designed to simplify and manage communication between services, particularly in microservice architectures. Provides a separate layer for service communication.

Using service mesh:

1. **Simplified Communication:** The service mesh handles tasks like load balancing, service discovery, and security, offloading these burdens from individual services. This makes development and maintenance easier.
2. **Security:** A service mesh can enforce security policies like encryption and authorization throughout your application, improving overall security posture.
3. **Observability:** Service meshes offer features for monitoring and tracing service communication, providing valuable insights for troubleshooting and performance optimization.

2.3 gRPC

gRPC (gRPC Remote Procedure Calls) is an open-source framework that facilitates high-performance communication between services. gRPC allows to invoke methods on a remote server as if they were local methods on the machine. This simplifies development by enabling writing code without worrying about the underlying network communication details. gRPC is highly performant, language agnostic and strongly typed, in contrast to e.g the typical REST API.

Some use cases for gRPC:

1. **Microservices Communication:** gRPC is a popular choice for building microservice architectures. It enables efficient communication between independent services within an application.
2. **Real-time Communication:** gRPC's streaming capabilities make it suitable for building real-time applications like chat or data feeds.

2.4 Front proxy

In the context of Envoy proxy, a front-proxy refers to a specific deployment configuration where Envoy acts as an intermediary server sitting in front of one or more backend services. It essentially functions as a forward proxy, handling client requests and directing them to the appropriate backend service.

How a front-proxy with Envoy works:

1. **Client Sends Request:** A client (like a web browser) initiates a request by sending it to the Envoy front-proxy. 2. **Envoy Routes Request:** Envoy, based on pre-defined rules, routes the request to the most suitable

backend service. This routing can be based on factors like URLs, headers, or load balancing strategies. 3. **Backend Service Processes:** The chosen backend service receives the request from Envoy and processes it. 4. **Response Sent Back:** The backend service sends its response back to Envoy. 5. **Envoy Delivers Response:** Finally, Envoy delivers the response from the backend service to the client.

2.5 Envoy as postgres sniffer

Envoy offers a built-in Postgres protocol filter. This filter acts like a sniffer, decoding the communication between Postgres clients and the server without interfering with the actual data flow. It can decode Postgres traffic in its non-SSL form. This means it won't modify or decrypt encrypted communication, ensuring a transparent monitoring approach for your Postgres database.

Using Envoy for Postgres statistics:

1. **Lightweight Monitoring:** Envoy's sniffer approach avoids additional load on the Postgres server, making it a lightweight solution for gathering statistics.
2. **Real-time Insights:** Envoy provides near real-time visibility into Postgres activity, allowing you to monitor performance and identify potential issues quickly.
3. **Improved Observability:** The captured statistics can be integrated with monitoring tools, providing valuable insights into your database's overall health and performance.

2.6 Statistics

Prometheus is an open-source monitoring system designed to scrape metrics from various sources, including Envoy. Envoy exposes various metrics related to its own operations and the traffic it handles. Prometheus acts as a time series database, storing the collected metrics over time. This allows you to analyze trends and identify patterns in your monitoring data.

Grafana is an open-source platform specifically designed for data visualization. It integrates seamlessly with Prometheus, allowing you to create dashboards and graphs to visualize the metrics collected by Envoy.

Envoy can be integrated with Prometheus to expose various metrics about its health and the traffic it manages. These metrics can then be visualized and analyzed using Grafana. The combination offers a comprehensive view of Envoy deployment, including its performance, traffic patterns, and resource utilization.

3 Case study concept description

The case study will consist of a typical Envoy Proxy use case—a service mesh with a front proxy, as shown on the Figure 1.

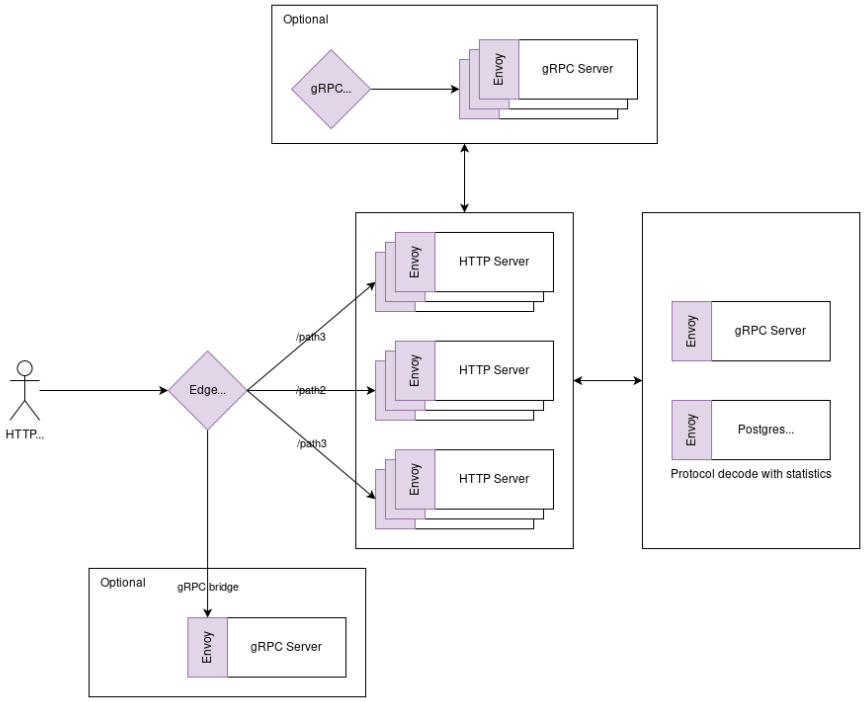

Figure 1: The case study architecture

3.1 Front Proxy

One instance of Envoy will serve as an edge reverse proxy and, therefore, be the entry point for incoming HTTP requests.

It will:

1. terminate TLS and perform authentication (e.g., using oAuth and JWT tokens). Services behind the proxy will not use encrypted communication,
2. route the HTTP requests to appropriate services,
3. load balance the requests within groups of the same service.

3.2 HTTP/REST Services

The edge proxy will route to groups of services based on HTTP request paths. Within each group, services will be duplicated (to enable load balancing) and respond to HTTP requests, sometimes also communicating with other services in the mesh, such as gRPC services and the database. All services will include a sidecar instance of Envoy.

3.3 gRPC Services

Just as with HTTP services, the mesh will contain a group of gRPC services. These services will not be directly accessed by the edge proxy but will communicate with the HTTP services. The gRPC services group may also include an internal load balancer. All services will include a sidecar instance of Envoy.

3.4 Database

The mesh will include a single instance of the Postgres database that might be queried by both the HTTP and gRPC services. The database will also include a sidecar instance of Envoy that can decode the Postgres protocol and collect statistics on the database queries.

3.5 Additions

The simple case study can be expanded with:

1. an additional edge proxy to create the "Double proxy" setup that allows for client/TLS connection termination closer (geographically) to the user, which offers benefits such as a shorter round-trip time for the TLS handshake.
2. gRPC bridge, where the edge proxy upgrades the incoming HTTP request to gRPC, enabling direct interaction between clients and the gRPC services,

3. rate limiting on the edge proxy (which, in real use cases, can prevent DDoS attacks),

4. proxying WebSocket connections by the edge proxy.

The service mesh will come with built-in support for functionalities like service discovery, health checks, and dynamic configuration (using, for example, the go-control-plane). The case study will also leverage Envoy's excellent observability features by setting up Grafana visualization, which will utilize Prometheus metrics.