Api Gateway

Matteo Moi

Contents

1 Introduction

An API Gateway is a critical component in a microservices architecture, acting as a single entry point for client requests and routing them to the appropriate backend services.

1.1 Features

- **Routing:** It routes incoming requests to the appropriate microservice. It abstracts the complexity of the underlying microservices, allowing clients to interact with a unified API
- **Load Balancing:** It distributes incoming requests across multiple instances of a service to ensure optimal resource utilization and high availability, improving performance and reliability.
- **Protocol Translation:** It can translate protocols, enabling clients to use a simple protocol like HTTP/HTTPS while the backend services may use other protocols such as gRPC, WebSocket, or SOAP. This allows seamless communication between different types of services and clients.
- **Authentication and Authorization:** It provides a centralized point for implementing authentication and authorization. It can integrate with OAuth, JWT, API keys.
- **Rate Limiting:** To protect services from being overwhelmed, the API Gateway can enforce rate limits and throttling policies, controlling the number of requests a client can make within a specified period.
- **Transformation:** It can transform incoming requests before forwarding them to the backend services. This includes modifying headers, rewriting URLs, or transforming the request payload.
- **Aggregation:** It can aggregate responses from multiple microservices into a single response, reducing the number of round trips between the client and server.
- **Logging:** The API Gateway can collect logs, metrics, and other analytics data. This includes tracking request rates, response times, error rates, and more, providing insights into the system's performance and health.
- **Caching:** It can cache responses from backend services to improve performance and reduce the load on services.
- **Cross-Origin Resource Sharing (CORS):** It can manage CORS policies, enabling or restricting access to resources from different origins.
- **Failover:** It can handle failover scenarios by routing requests to backup services or returning cached responses if the primary service is unavailable.
- **Circuit Breaking:** It can implement circuit breaker patterns to prevent cascading failures in the system by temporarily blocking requests to failing services and allowing them to recover.

- **Service Discovery Integration:** It can integrate with service discovery mechanisms to dynamically route requests to the correct instances of microservices. This ensures that the gateway always knows the available services and their locations.
- **API Versioning:** It can handle different versions of APIs, allowing clients to use specific versions while enabling the development of new versions without breaking existing clients.

Example of API Gateways

- **NGINX**
- **AWS API Gateway**
- **Netflix Zuul**
- **Spring Cloud Gateway**
- **Kong**