What is an API Gateway?

An API stands for Application Program Interface. It is a set of instructions, protocols, and tools for building software applications. It specifies how software components should interact.

The API Gateway is a server. It is a single entry point into a system. API Gateway encapsulates the internal system architecture. It provides an API that is tailored to each client. It also has other responsibilities such as **authentication, monitoring, load balancing, caching, request shaping and management,** and **static response handling**.

API Gateway is also responsible for **request routing, composition,** and **protocol translation**. All the requests made by the client go through the API Gateway. After that, the API Gateway routes requests to the appropriate microservice.

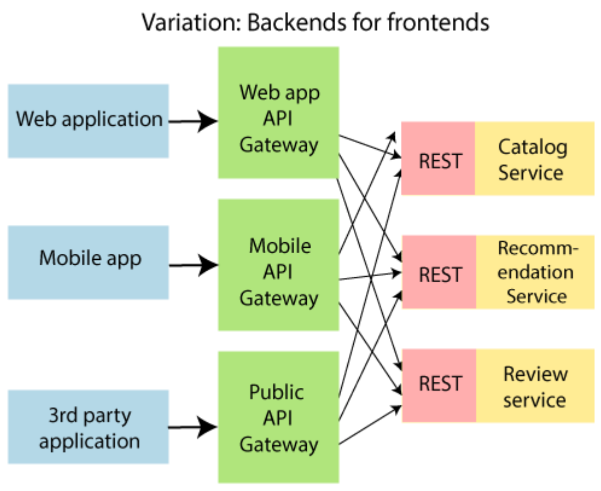
The API Gateway handles the request in one of the two ways:

- It routed or proxied the requests to the appropriate service.
- Fanning out (spread) a request to multiple services.

The API Gateway can provide each client with a custom API. It also translates between two protocols, such as **HTTP, WebSockets,** and **Web-Unfriendly** protocols that are used internally.

Example

The popular example of API Gateway is **Netflix API Gateway**. The Netflix streaming services are available on hundreds of different kinds of devices such as **televisions**, **set-top boxes, smartphones, tablets,** etc. It attempts to provide a **one-size-fits-all** API for its streaming service.



An API Gateway includes:

- Security
- Caching
- API composition and processing
- Managing access quotas
- API health monitoring
- Versioning
- Routing

Advantages of API Gateway

- The most important advantage of API Gateway is that it encapsulates the internal structure of the application.
- Rather than invoking the specific service, the client directly talks to the API Gateway.
- It reduces the number of round trips between client and application.
- It simplifies the client code.
- It reduces coding efforts, makes the application more efficient, decreases errors all at the same time.
- It provides each kind of client with a specific API.

Disadvantages

- It requires routing rules.
- There is a possibility of a single point of failure.
- Risk of complexity due to all the API rules are in one place.

Working of API Gateway

In microservices, we route all the requests through an API. We can implement common features like **authentication, routing, rate limiting, auditing,** and **logging** in the API Gateway.

Consider a scenario in which we do not want to call a microservice more than five times by a particular client. We can do it as a part of the limit in the API Gateway. We can implement the common features across microservices in the API gateway. The **Zuul API Gateway** is a popular API Gateway implementation.

We must implement the following features in all the microservices:

- **Service Aggregation**
- **Authentication, authorization and Security**
- **Rate Limits**
- **Fault Tolerance**

Suppose there is an external consumer who wants to call **fifteen** different services as part of one process. It is better to aggregate those fifteen services and provide one service call for the external consumer. These are the kinds of features that are common across all the microservices. These features are implemented at the level of API.

Instead of allowing microservices to call each other directly, we would do all the calls through API Gateway. API Gateway will take care of common features like authentication, fault tolerance, etc. It also provides aggregation services around all microservices because all calls get routed through the API Gateway.

What does an API gateway do?

An API gateway is an API management tool that sits between a client and a collection of backend services.

An API gateway acts as a reverse proxy to accept all application programming interface (API) calls, aggregate the various services required to fulfill them, and return the appropriate result.

Most enterprise APIs are deployed via API gateways. It’s common for API gateways to handle common tasks that are used across a system of API services, such as user authentication, rate limiting, and statistics.

Check out this on-demand API management and security demo

Why use an API gateway?

At its most basic, an API service accepts a remote request and returns a response. But real life is never that simple. Consider your various concerns when you host large-scale APIs.

- You want to protect your APIs from overuse and abuse, so you use an authentication service and rate limiting.
- You want to understand how people use your APIs, so you’ve added analytics and monitoring tools.
- If you have monetized APIs, you’ll want to connect to a billing system.
- You may have adopted a microservices architecture, in which case a single request could require calls to dozens of distinct applications.
- Over time you’ll add some new API services and retire others, but your clients will still want to find all your services in the same place.

Your challenge is offering your clients a simple and dependable experience in the face of all this complexity. An API gateway is a way to decouple the client interface from your backend implementation. When a client makes a request, the API gateway breaks it into multiple requests, routes them to the right places, produces a response, and keeps track of everything.

An API gateway’s role in API management

An API gateway is one part of an API management system. The API gateway intercepts all incoming requests and sends them through the API management system, which handles a variety of necessary functions.

Exactly what the API gateway does will vary from one implementation to another. Some common functions include authentication, routing, rate limiting, billing, monitoring, analytics, policies, alerts, and security.

How an API gateway supports DevOps and serverless environments

In organizations that follow a DevOps approach, developers use microservices to build and deploy apps in a fast-paced, iterative way. APIs are one of the most common ways that microservices communicate.

Additionally, modern cloud development, including the serverless model, depends on APIs for provisioning infrastructure. You can deploy serverless functions and manage them using an API gateway.

In general, as integration and interconnectivity become more important, so do APIs. And as API complexity increases and usage grows, so does the value of an API gateway.

References
https://www.redhat.com/en/topics/api/what-does-an-api-gateway-do
https://www.javatpoint.com/introduction-to-api-gateways