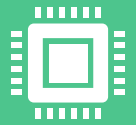
API Gateway

- 1- What is an API Gateway?
- 2 - What is the Typical Use Case for an API Gateway?
- 3 - KONG API Gateway
- 4 - What are the Benefits of Using an API Gateway?
- 5 - All the plugin

What is an API Gateway?

An API Gateway is a reverse proxy that exposes microservices as APIs. As the name implies, it acts as a “gatekeeper” between clients and microservices,



Typical features of an API Gateway include the ability to authenticate requests, enforce security policies, load balance between backend services and throttle them if necessary.

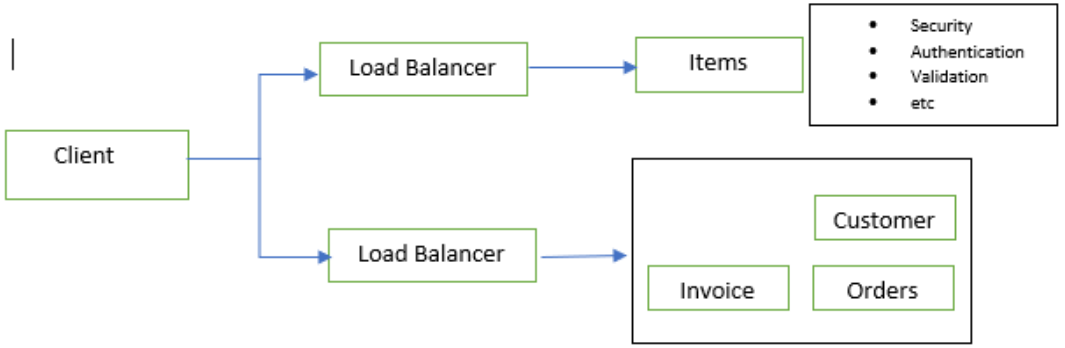
Reduced complexity in their client and server code and a reduction in the overall network latency

What is the Typical Use Case for an API Gateway?

- The fundamental purpose of an API Gateway is to avoid exposing backend services and data sources to the outside world.

Ex - Monolithic Application

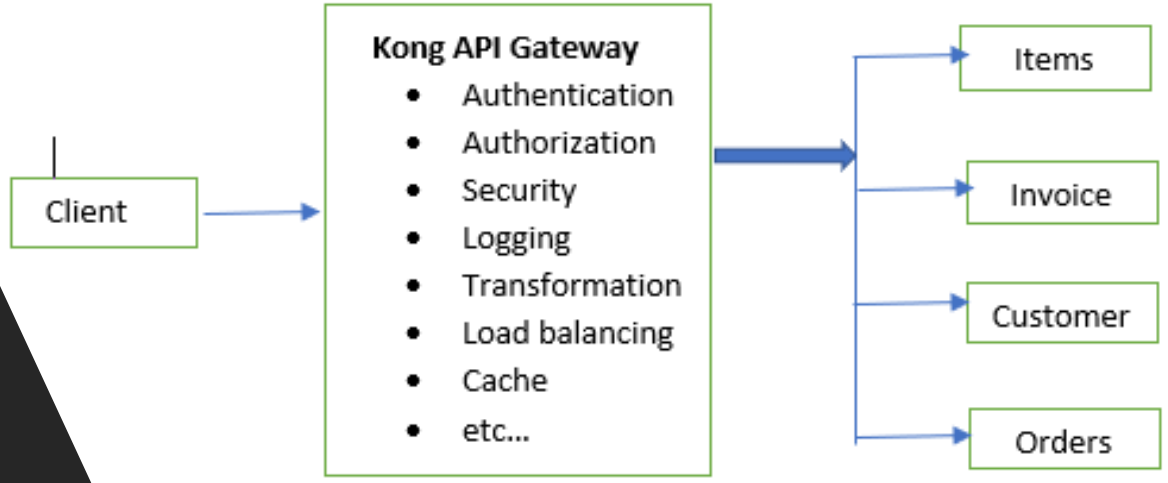
Ex – One Microservice Application out of Monolithic



KONG API Gateway

Kong is an API management solution that acts as an API gateway for microservices. It has a plug-in architecture that allows users to extend Kong's core functionality.

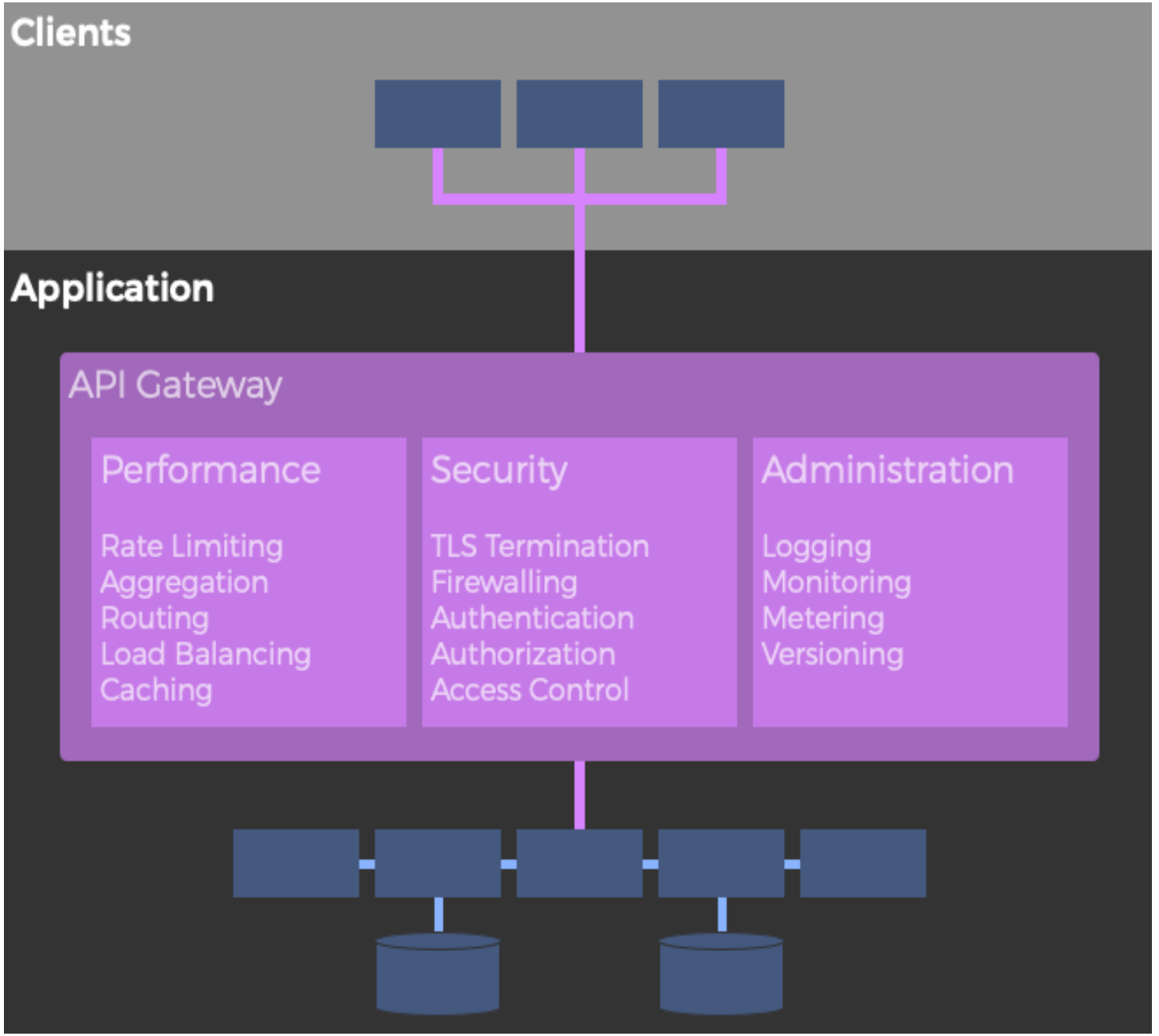
Install and configure a Kong cluster.
Manage a Kong API gateway to provide a single communication point for accessing APIs.
Control web services traffic at a granular level.
Secure APIs by authenticating access, Build with Nginx, Lua language
Control traffic through rate limiting and quotas.
Monitor, log and analyse API traffic using a third-party solution such as ELK stack.
configurable through restful API
add powerful functionality through plugins



Security-related tasks like SSL termination, whitelisting, firewalling, authentication and authorization.

Performance related capabilities like throttling (or rate limiting), request aggregation, routing, load balancing and caching.

Administrative tasks like logging, monitoring, metering and API versioning.



What are the Benefits of Using an API Gateway?

- **Cleaner and Simpler Client Code**

- When client and backend services are decoupled, the client doesn't need to know how the individual services have been decomposed. This decoupling makes it easier to maintain client code and to refactor services without one code base impacting the other. Also, with an API Gateway, developers don't need to build logic into their apps to keep track of endpoints or how to handle request failures so that they don't end up delivering a bad user experience.

- **Less Latency Equals a Better End-User Experience**

- Without an API Gateway, instead of a single request, the client will need to make multiple network round trips to the backend services and in the process, add significant latency. Of course, high latency is directly related to bad user experiences. With an API Gateway, requests can be aggregated and routed efficiently to minimize the "chatter" required to deliver a completed request.

- **Simplified Authentication and Encryption**

- With an API Gateway, organizations have a centralized place from which to handle authentication, SSL, client rate limiting and other security-related policies for all their backend services.

Conclusion

In summary, an API gateway is a reverse proxy that offers up microservices as APIs. An API gateway also helps to minimize the potential dangers of exposing backend services and data sources directly to clients. The use of an API gateway makes for cleaner and simpler client code, less latency, and more simplified authentication and encryption.

All the plugin -
https://github.com/Kong/kong/tree/master/kong/plugins