API Gateways

An overview

finleap

API Gateway – What Is It?

It's a pattern turned into a really versatile tool

- It's an Edge Service – i.e. can sit on the edge of your network
- It's a Proxy
- It's a **Reverse** Proxy
- It's a Level 7 load balancer – or even more than that
- It's a Router
- It provides Access Control
- It can translate protocols and formats
- It takes care of common functionality and help separate concerns in your architecture
- It can be an API Orchestrator
- It can help to slice and dice a portfolio of APIs to better suit certain use cases (ex. Backend-for-Frontend).
- Provide an API Portal
- The silver bullet that solves all your problems (Quote: Vendor XYZ)

Toolbox vs. Swiss Army Knife

vs

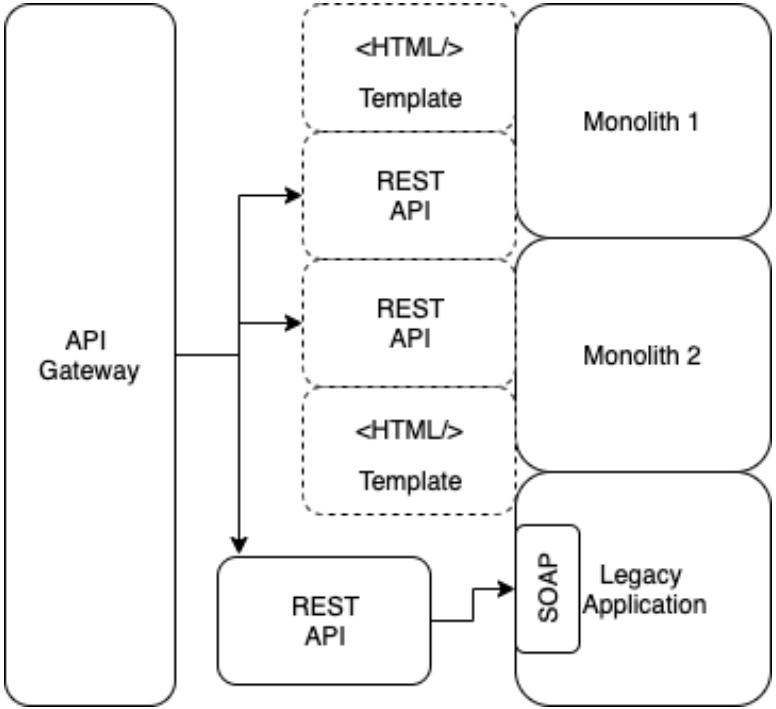
API Gateways

Common Patterns and Anti-Patterns

API Gateways – The History

Grossly simplified and highly subjective….

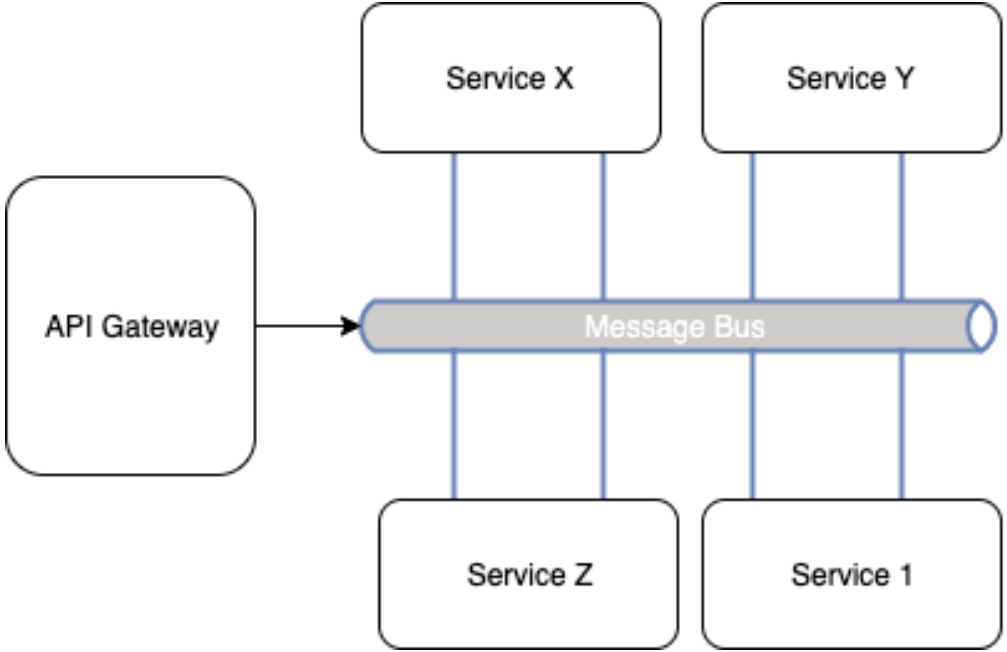
- In the old days, we built Apps as one or more (Web-) “Monoliths”. Suddenly we needed to add an API because AJAX and REST became fashionable
- API Gateway was built as a separate Application that was forwarding API Calls from the outside
- Maybe you mapped to a legacy SOAP API



API Gateways – The RPC or CQRS Entry Gate

Your backend doesn't follow REST Architecture? You still need to provide an API

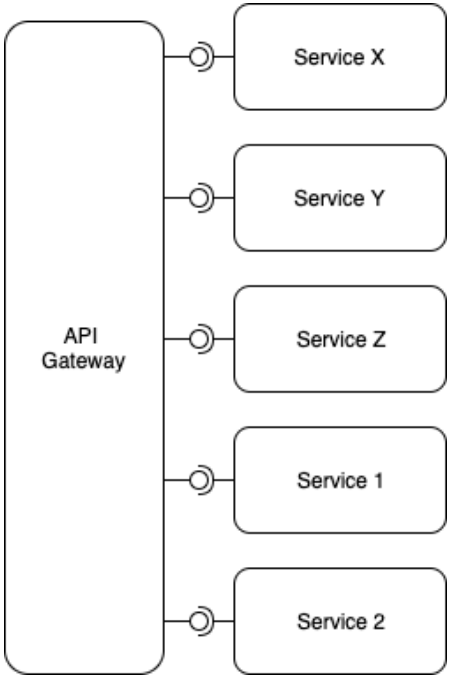
- An asynchronous architecture in the backend such as RPC or CQRS using protocols such as JMS, ActiveMQ, RabbitMQ, or Protocol Buffers, or we have a "Service Bus"
- How did messages go into the bus from the outside? Put an API Service, aka. API Gateway, in front
- Gives Freedom to also ad REST-style Resources
- Some products out there come with adapters for common protocols such as for Kafka, AMQP and more



API Gateways – The Indespensible Ingredient

The Key to Microservice Architecture

- In the “microservice age”, we built lots of services with their own API. We needed a way to cover common functions, unify the interface and add a routing layer so the frontend doesn’t need to know about every single backend service
- Suddenly, a client needed to know about all those services, needed to authenticate for each one separately, and potentially speak a different protocol or format
- Solution: Put an API Service, aka. API Gateway, in front, to take care of routing and service discovery

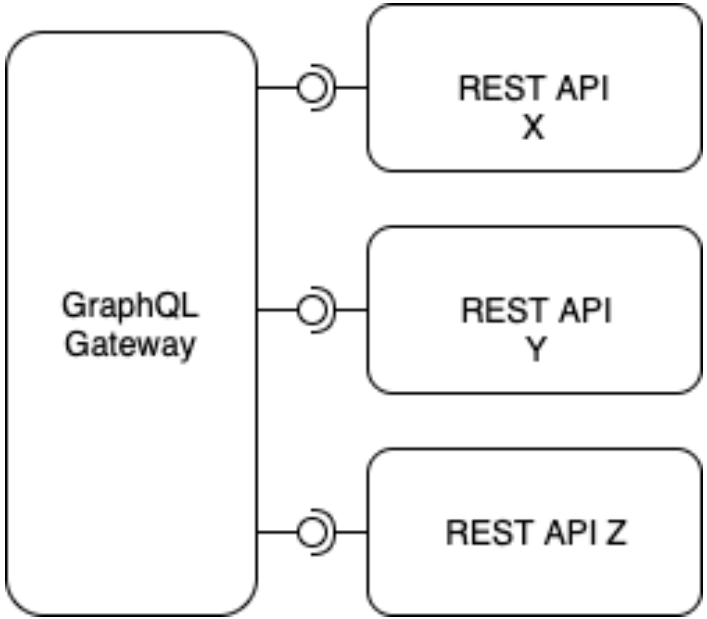


API Gateways – The Hipsters

GraphQL is a good fit for “living” in the API Gateway

- To integrate GraphQL into an existing infrastructure, you either need to build it directly into your services, or build an intermediate layer that resolves a GraphQL Query to different data sources – **a GraphQL Gateway**

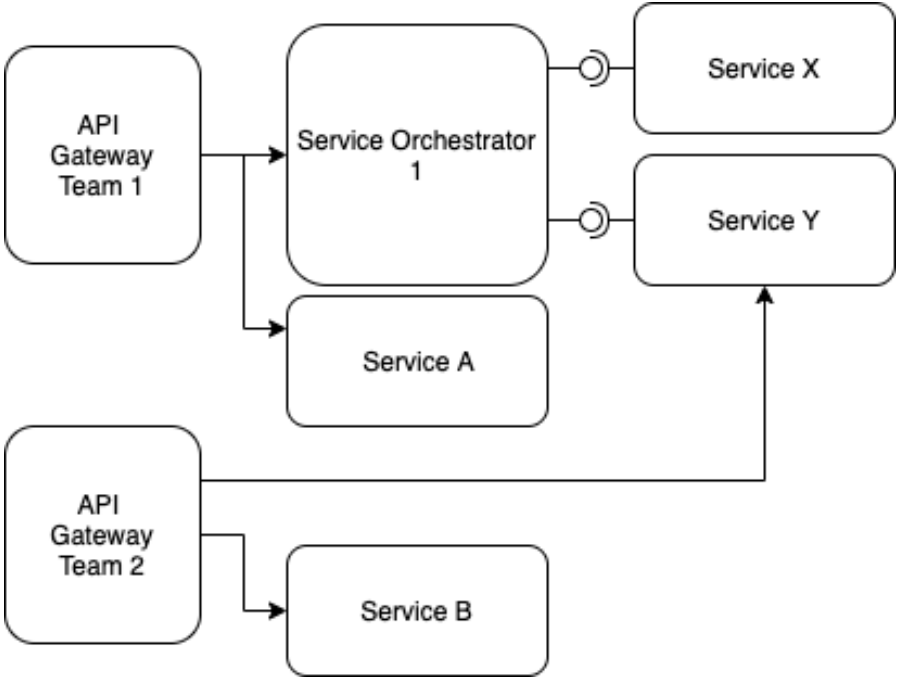
- Idea is to federate multiple data sources into a single API
- A new variation of BFF pattern



API Gateways – Good Patterns

Backend for Frontend

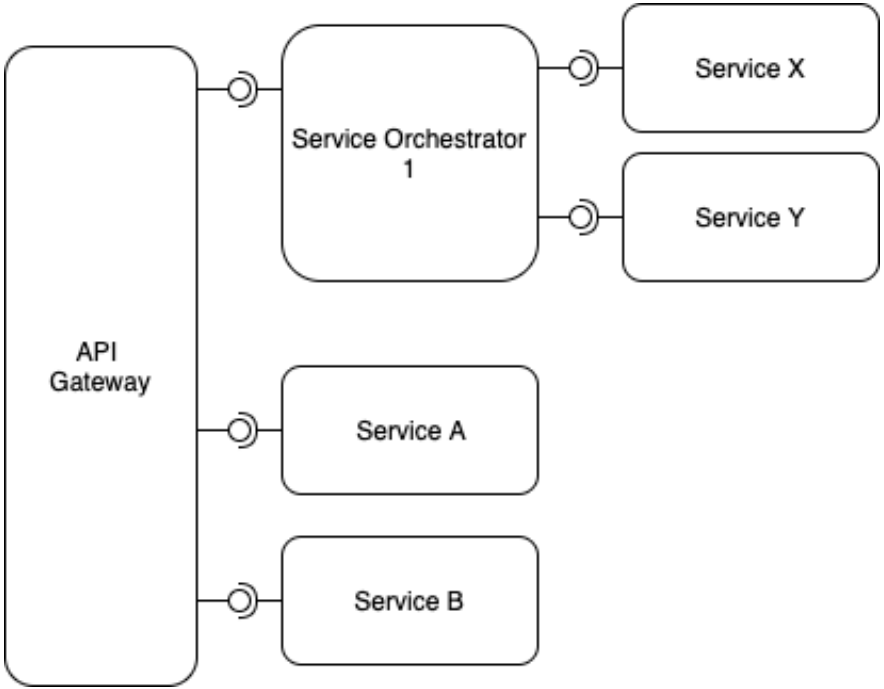
- If you scale your team, give teams their own API Gateway so they can slice and dice the backend APIs to their needs



API Gateways – Good Patterns

Orchestrator Services

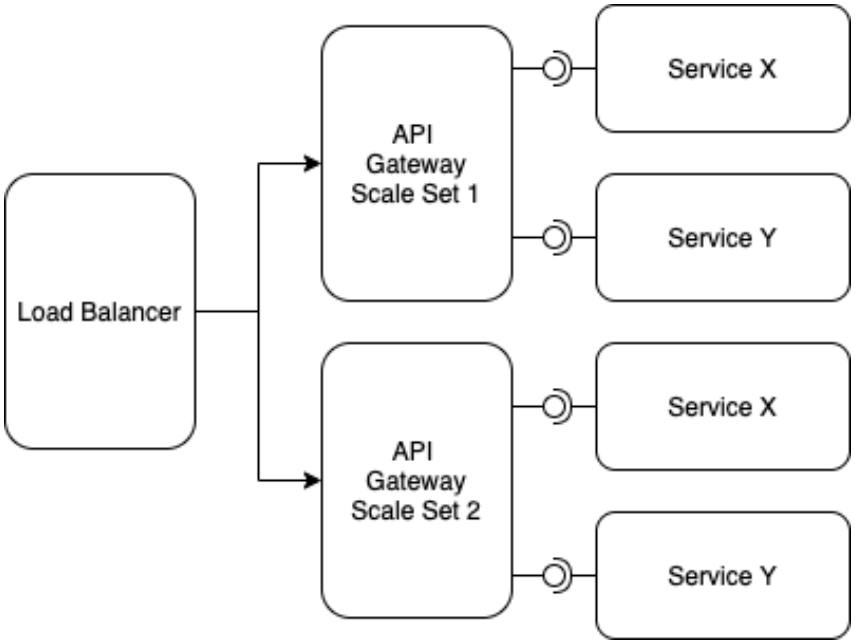
- Refrain from putting logic into the API Gateway – Because of Change Cycles, have Orchestrator Services



API Gateways – Good Patterns

Put a Load Balancer Infront

- API Gateways need to scale themselves. Even though they can serve as a load balancer, the inbound traffic needs to be managed, too



API Gateways – Bad Patterns

Centralisation, Single Point of Failure, Overambitious API Gateway, Fake Service Mesh

- Refrain from tendencies that lead to dangerous concentrations of centralization – and to loss of flexibility therefore due to more complex change and deployment procedures. **BFFs are your BFF**
- Avoid having your API Gateway as a single point of failure – explore ways how to scale out early. Ideally **keep it stateless**
- Don’t put too much logic, orchestration etc.. into the API Gateway – this again leads to hazardous levels of centralization
- Don’t put an API Gateway in front of every service – this will lead to a counterfeit service mesh, and it will get expensive as you consume more compute resources

API Gateways

Market Overview

API Gateway Features

Look out for the following features…

- Authentication
- Authorisation
- Access Control
- Load Balancing
- Routing
- Rate Limiting
- Caching
- Manage API Clients
- Metrics Collection
- Request Logging
- Health Checks
- Circuit Breaker
- Support your Protocols
- Plugin Architecture / custom adapters
- CORS Headers
- Simple Orchestration

You want your API Gateway to be….

- Easy to Scale out
- Declarative way to configure Resources and Routes
- Support standards like OpenAPI
- Integrate with your Monitoring

Some are more a swiss army knife, some more a toolbox

- Functions like Routing, Auth, etc.. Can be available in a L7 Load Balancer
- Maybe you need more a toolbox or a framework
- Products on the market are usually more a swiss army knife, with some being more of a toolbox

API Gateways – Market Overview

API Gateway Products that are available for free in some shape or form

- **“Standard” Gateways:**
  - Spring Cloud
  - Kong
  - Tyk
  - Express Gateway
  - KrakenD
  - Ambassador
- **GraphQL Gateways:**
  - Apollo Gateway
  - Sqoop

API Gateways – Enterprisey Vendors

They usually cost a lot of money… Therefore I am going to skip them

- **Cloud Provider Gateways:**
  - Amazon API Gateway
  - APIGee (now Google Cloud)
  - They are usually very well integrated within the Cloud Vendor Ecosystem
- Other Vendors
  - Dell Boomi
  - Mulesoft
  - 3Scale
  - Axway
- They all fall firmly in the Swiss Army Knife Category

(1) Here is the footnote

Spring Cloud Gateway

Based on Spring Framework

The Good

Proven Java Technology and APIs

Very easy to add in your Java Project

The Bad

It‘s more a Framework (maybe that‘s not so bad ;)

All the configuration is in Java Code

Conclusion: Great Toolbox and best option to use if you are firmly located in the Spring world

Kong

Based on proven NginX Technology, Open Source sind 2015

The Good

Proven Technology Nginx

One of the oldest products in the field

Everything can be controled through Control API

Cloud Function Adapters available

The Bad

Questionable Scale Out Model – needs Postgres or Cassandra

Control API is Imperative

Declarative Configuration added as an Afterthought and mixed up with DB-Less Operation Model, makes Infrastructure-as-Code cumbersome

Need to pay for UI

Need to pay for a lot of features

Custom plugins written in Lua – a rather obscure language

Conclusion: Hasn‘t aged well in my opinion – complex scaling and operating model, need paid tier fast

Tyk

They have the cutest Mascot

The Good

Everything can be controled through Control API

UI included

Built in Go

You can develop Plugins in Go and Javascript (JSVM)

Tyk.io

The Bad

Requires a Redis to run and potentially a MongoDB for Dashboard

Not built „declarative first“, makes Infrastructure-as-Code cumbersome

Scaling out requires paid version (i.e. >1 node)

Something like Tyk Pump between „Dashboard“ and Gateway – complex setup

No protocol transformation out of the box

Conclusion: Gets expensive fast, full operating model is complex

Express Gateway

Based on NodeJS Express

The Good

Proven technology NodeJS and Express

Build declarative first

It‘s just JavaScript

Easy to customise

Needs no database – completely stateless

The Bad

It‘s JavaScript – I guess that can put off some people

Relatively new technology

Conclusion: A real option, easy to customise and build your own gateway

KrakenD

Very Nimble Option build from Scratch in Go

The Good

Needs no database – completely stateless, scales very well horizontally

Completely declarative

Got some nice adapters, called Backends, for RabbitMQ, Kafka, Lambdas and more

It doesn‘t do stuff other systems do really well – like Monitoring and Service Discovery. Instead it pre-integrates with standard solutions

Benchmarks extremely well compared with Kong and Tyk

Can aggregate multiple API Calls

The Bad

Relatively New Technology

Conclusion: Yes it‘s a Swiss Army Knife, but more of the small, practical sort

Ambassador

Based on Envoy Proxy, exclusive to Kubernetes

The Good

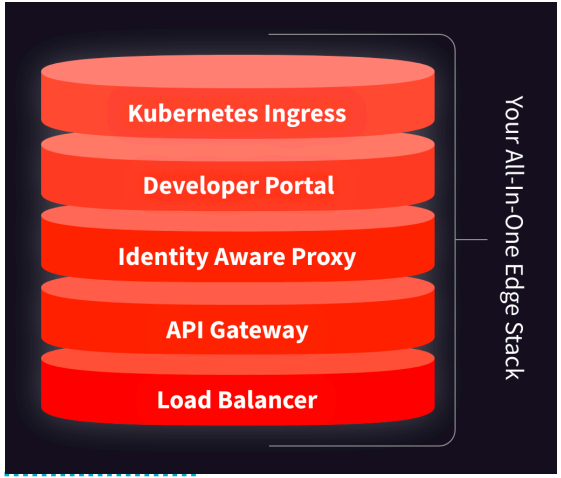
Completely declarative Configuration

Based on Envoy means it has already great support for advanced features like Canary Routing and Shadowing

The Bad

It looks like they are trying to build another One-size-fits-all solution

Runs only in Kubernetes



Conclusion: Powerful reverse proxy with added functionality– worth trying if you run K8S

Apollo Gateway

Federates other GraphQL APIs

The Good

More a Framework kind of thing

Made by early GraphQL Adopters

The Bad

Only GraphQL backends supported

In praxis, federating GraphQL APIs can turn out very complex

Conclusion: Not sure why someone will need that

Sqoop

Based on Envoy Proxy, exclusive to Kubernetes, Attempt to federate all Sort of APIs

The Good

Adapters for gRPC, Serverless – not just REST

Based on Envoy means it has already great support for advanced features like Canary Routing and Shadowing

The Bad

Only for Kubernetes

I don‘t see how this can work purely with Configuration and Go Templates

Very new technology

Conclusion: I don‘t have an opinion yet – worth trying if you run K8S and want to have GraphQL

By the way, it really exists…

- https://www.amazon.com/Wenger-16999-Swiss-Knife-Giant/dp/B001DZTJRQ

Tools & Home Improvement › Power & Hand Tools › Hand Tools › Knives › Pocket Knives & Folding Knives

Wenger 16999 Swiss Army Knife Giant

by Wenger

572 ratings | 282 answered questions

Available from these sellers.

- 87 implements
- 141 functions
- Perfect for the collector
- Featured by major media outlets

› See more product details

Get in touch to learn more

Sebastian Weikart

Engineering Lead

sebastian.weikart@finleap.com

Thank you for your attention