# From Good to Great:

## How API Gateways Can Level-Up Your Software

PRESENTATION BY:

DANIEL MIKUSA
dan@mikusa.com
Lead Software Engineer @ 7SIGNAL, Inc.

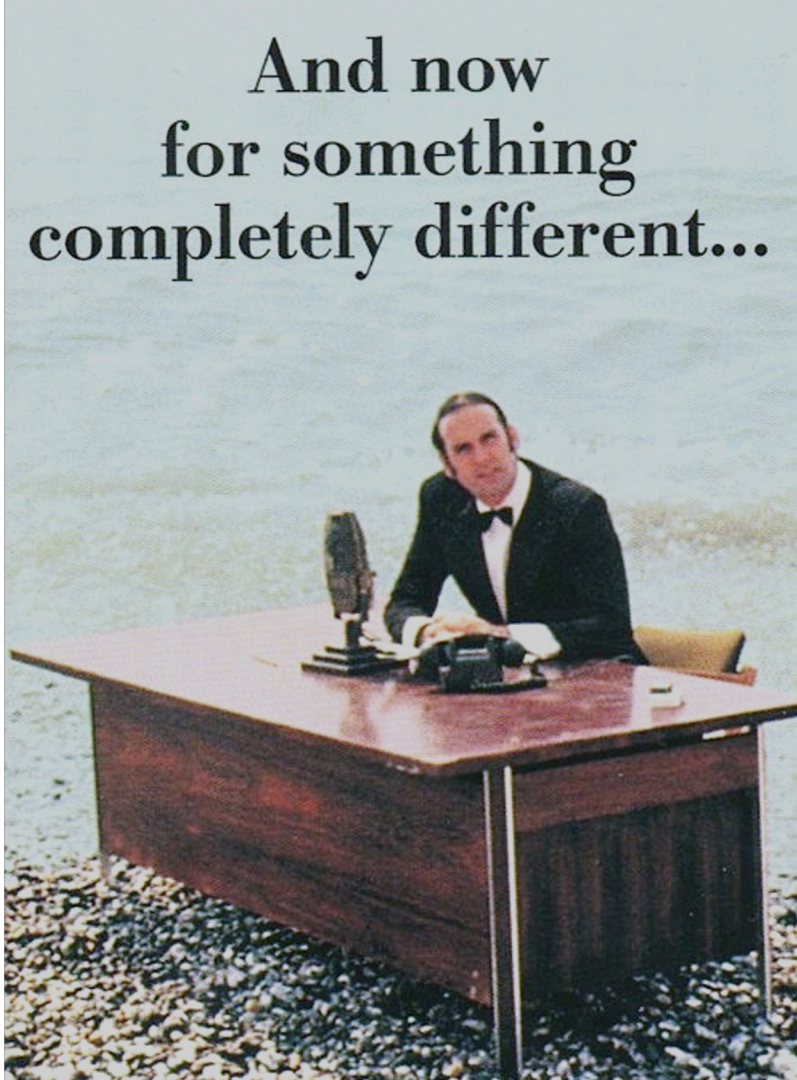
# How API Gateways Can Level-Up Your Software

01. Intro to APIs

02. API Gateway Patterns

03. Demo

Survey



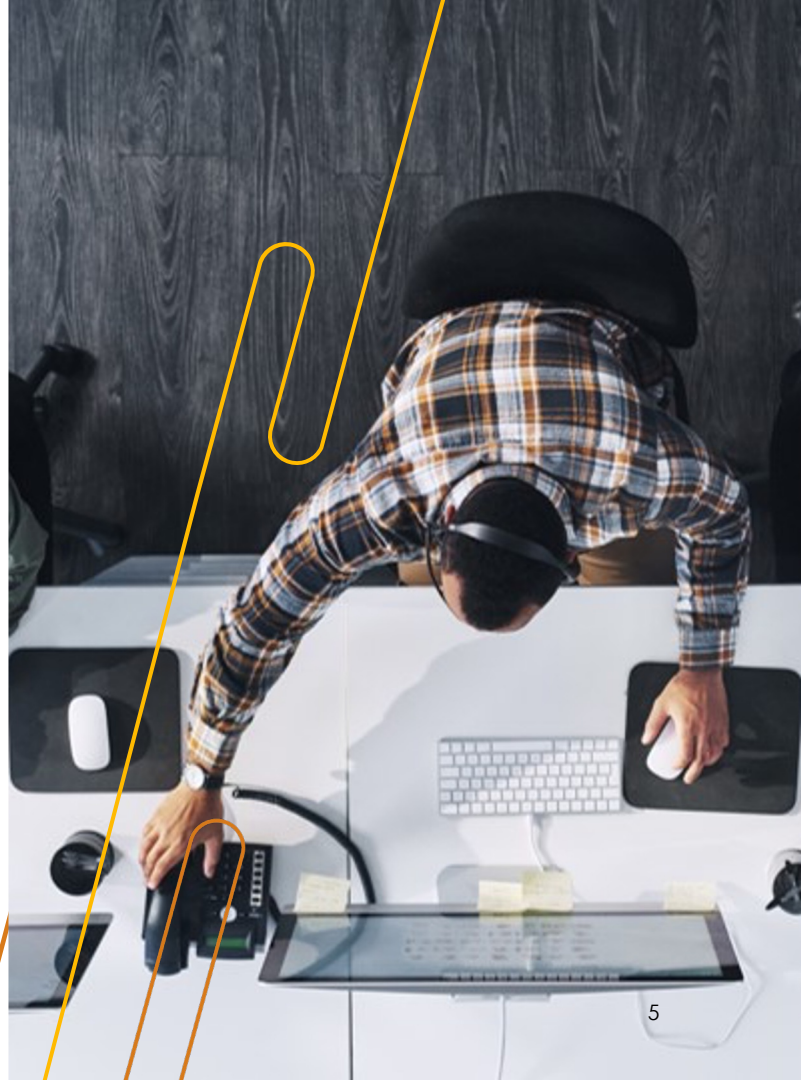
# Part One

## We will cover these topics

- What is an API?
- Types of API?
- Consumer Benefits
- Service Provider Benefits
- Discovery

# API Basics:
# What is an API?

- A way for two different pieces of software to communicate
  - What software?
  - How does it communicate?
  - What does it look like?
  - Why?
- More specifically, a Remote APIs
  - What software?
  - How does it communicate?
  - What does it look like?
  - Why?



# API Basics :
## API Philosophy

By Resource

By Operation

# API Basics:
# Types of APIS

- History of Remote APIs
  - IPC
  - Raw TCP/IP & UDP
  - CORBA
  - HTTP
  - XML RPC
  - SOAP
- Today's Landscape
  - REST
  - GraphQL
  - gRPC
  - Message-based Systems

# Consumers:
# Why use APIs?

- Less work, so you can deliver results faster
- Less work takes less time. Time is money, so saving time saves the company money
- Specialized providers deliver high quality results, or features that would be impossible to develop internally
- Different, possibly better, pricing models
- Mash-ups. Taking multiple APIs and combining them together for new results.

# Consumers:
# Why use APIs?

- Customizable. Clients can implement behaviors not otherwise possible through a UI.
- Automation. With an API, a user can automate tasks that would otherwise be tedious and time consuming.
- Stability. The API is a contract, so you can develop against it knowing your investment will work for years to come.
- Many providers. There are providers for so many different services, and in some cases providers compete which can help with costs.

# Service Providers:
# Why provide APIs?

- UI work is time consuming, APIs skip the UI so we can deliver functionality faster
- Exposing an API enables customers to build their own features, which saves development time
- APIs drive engagement with our software. The more integrated a customer is with our software, the more likely they are to renew.
- APIs can drive new business models, new ways of selling services and products
- APIs enable Integration with 3rd party services to drive more visibility for our product and more customers

# Service Providers:
# Why provide APIs?

- Flexibility. An API is basically a facade, what happens behind the API can change without the customer being impacted
- APIs are composable. We can incrementally add features and functionality without impacting users.
- APIs provide logical boundaries around which to organize the development team and projects.

# Discovery:
# How do you find APIs?

- Many public APIs can be found online
- Most companies advertise their API or SDK as part of their product offerings
- Public API Marketplaces

- Many companies also have internal API
- With smaller companies or only a few APIs this can be organic or word-of-mouth
- With scale, a company may create their own API marketplace where employees can get access to services

# Progress

01. Intro to APIs

02. API Gateway Patterns

03. Demo

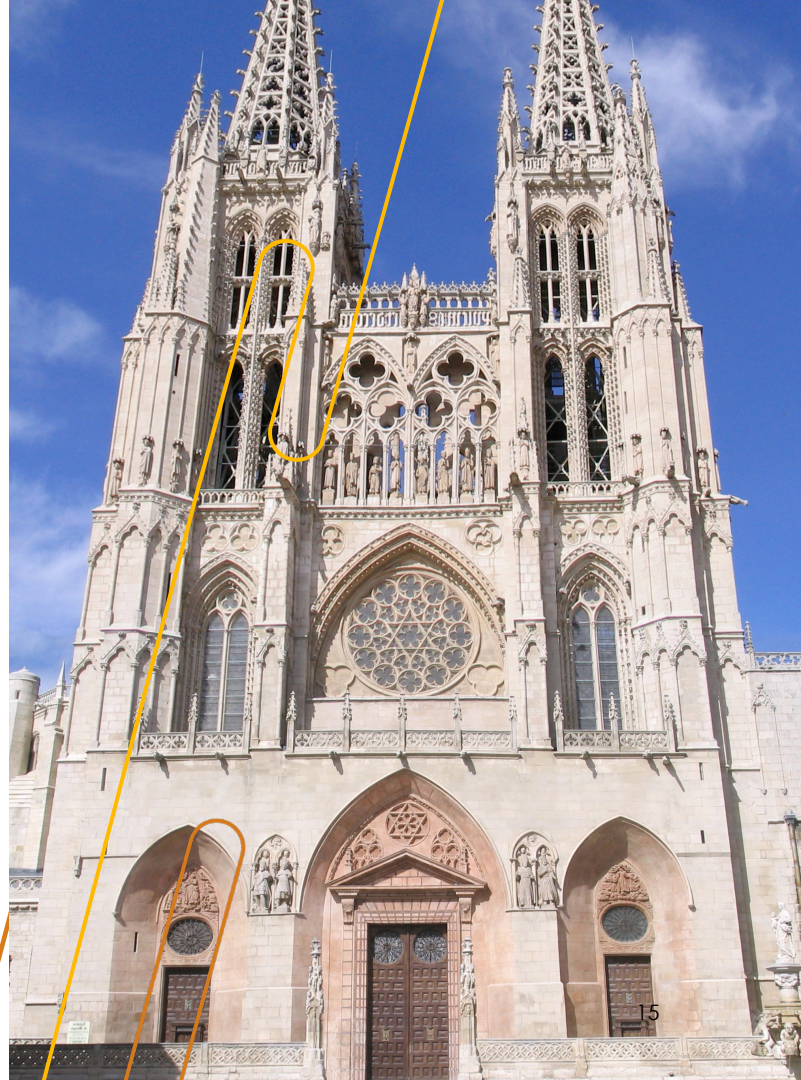
# Part Two

## We will cover these patterns

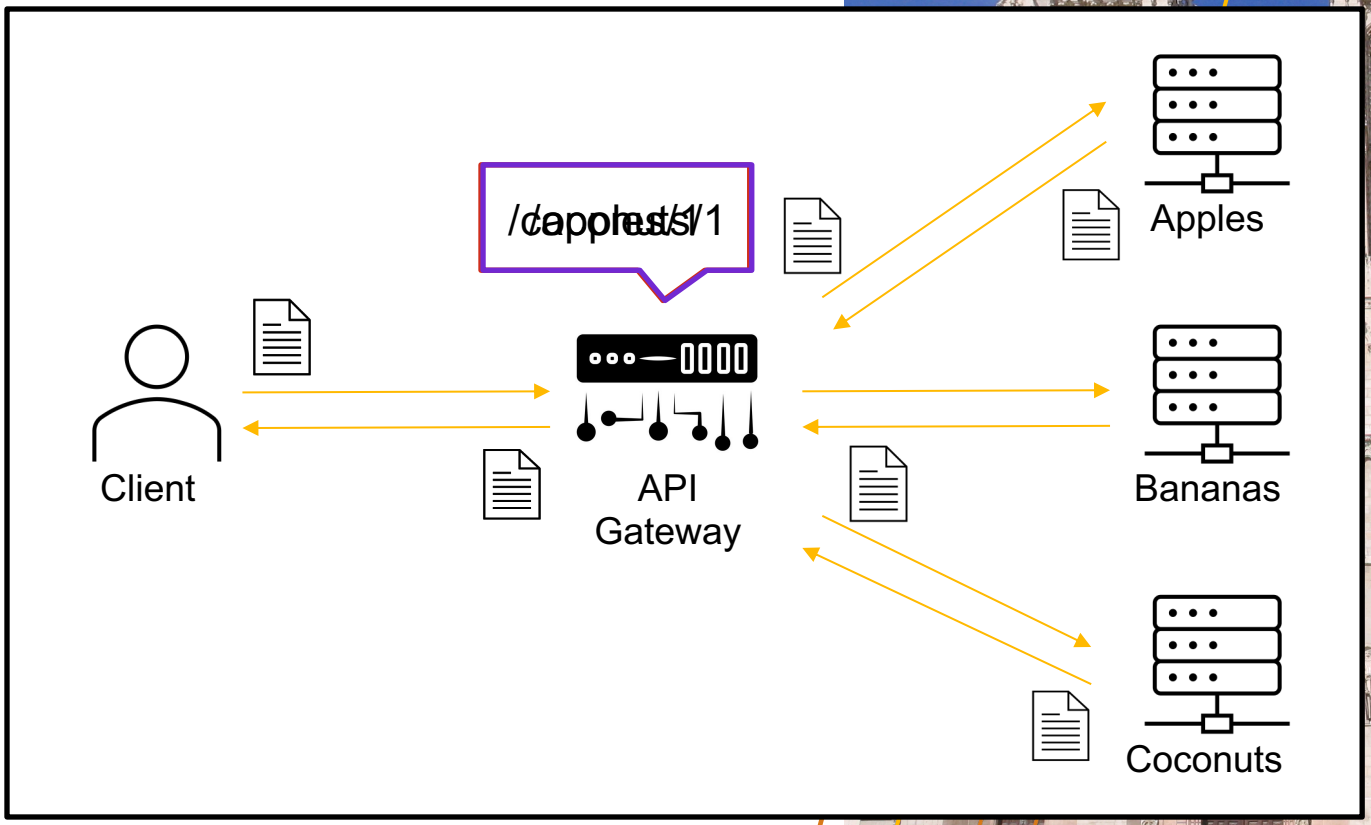
- Façade
- Smart Load balancer
- Spy
- Bouncer
- Transformer
- Circuit Breaker
- Cache

# Patterns:
# Façade

- Modify outward appearance of your APIs
- Great for:
  - Unifying multiple APIs
  - Managing API versions more easily
  - Helping with CORS
  - Facilitating easy microservices
    - Provides flexibility in how your service is decomposed
    - Moderates between services and clients
  - Enabling polyglot backend development
  - Deconstructing monolith applications
  - Providing common functionality in the Gateway



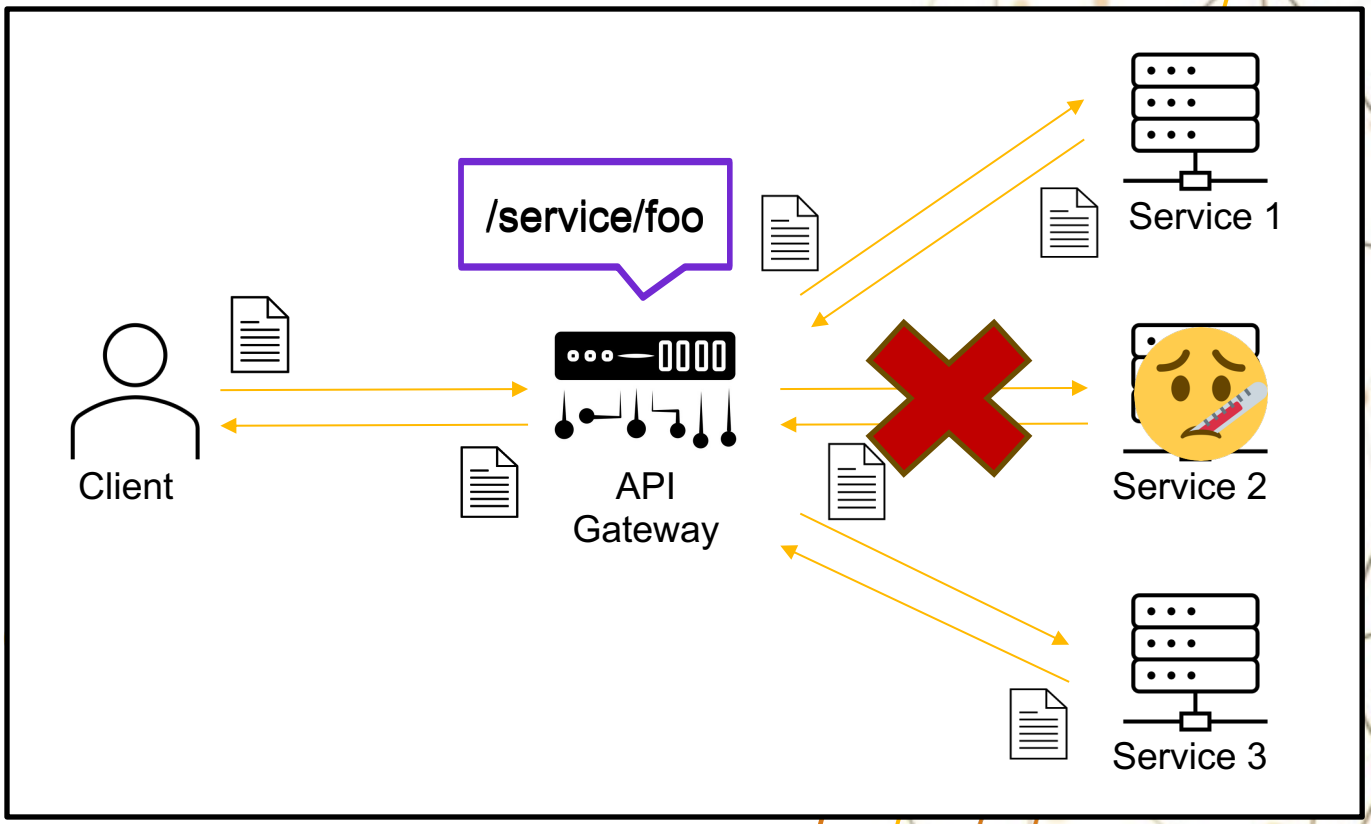
# Patterns: Façade



# Patterns:
# Smart Load Balancer

- Balance and route traffic in today's dynamic world
- Great for:
  - Most traditional load balancing and reverse proxy needs
  - Load balancing in dynamic environments
  - Client-side load balancing
  - Routing in more customized and context aware situations

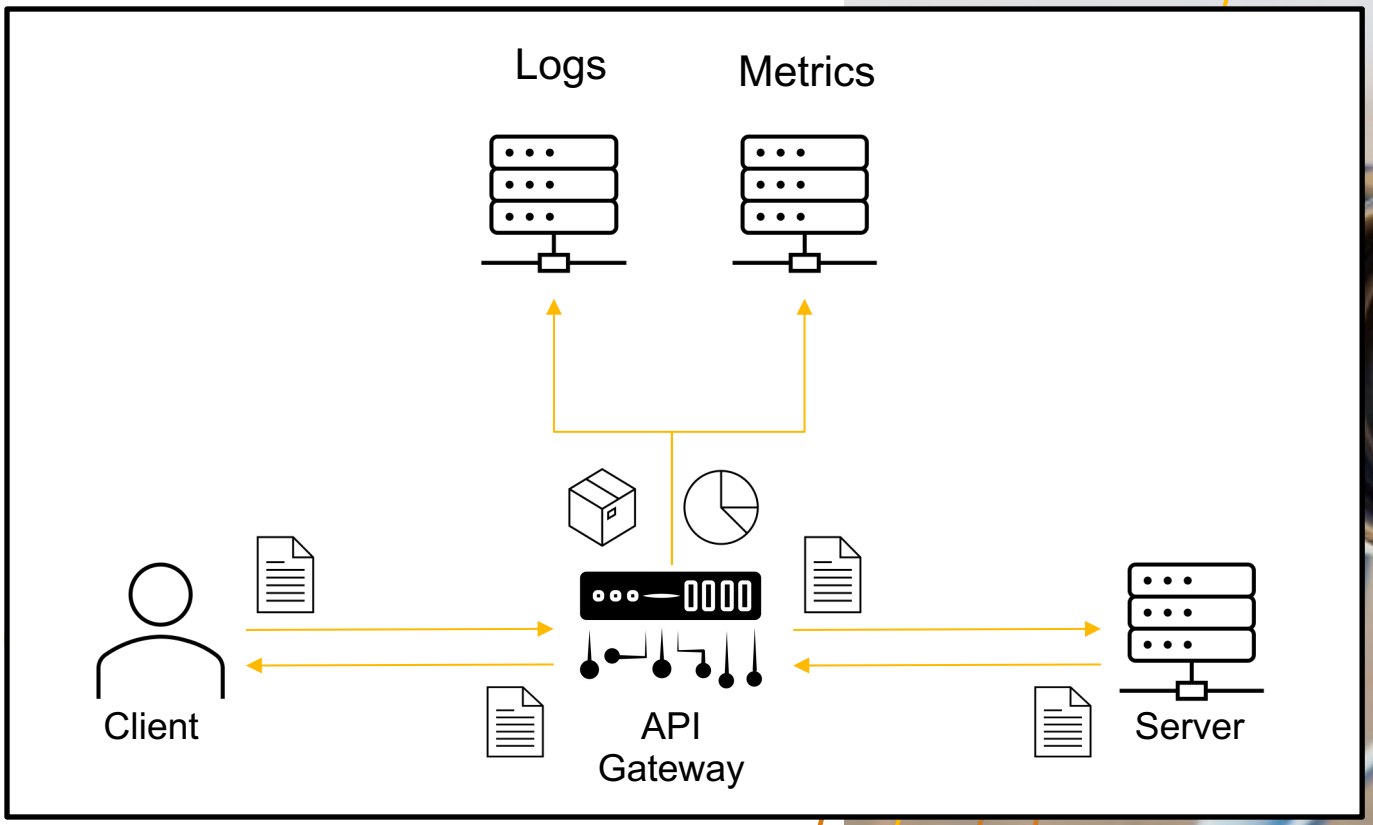
# Patterns: Smart Load Balancer



# Patterns:
# Spy

- Spy or introspect traffic coming through the Gateway
- Great for:
  - Centralized monitoring and metrics across all services
  - Logging information about requests, like Audit logging
  - Capturing data for comprehensive troubleshooting

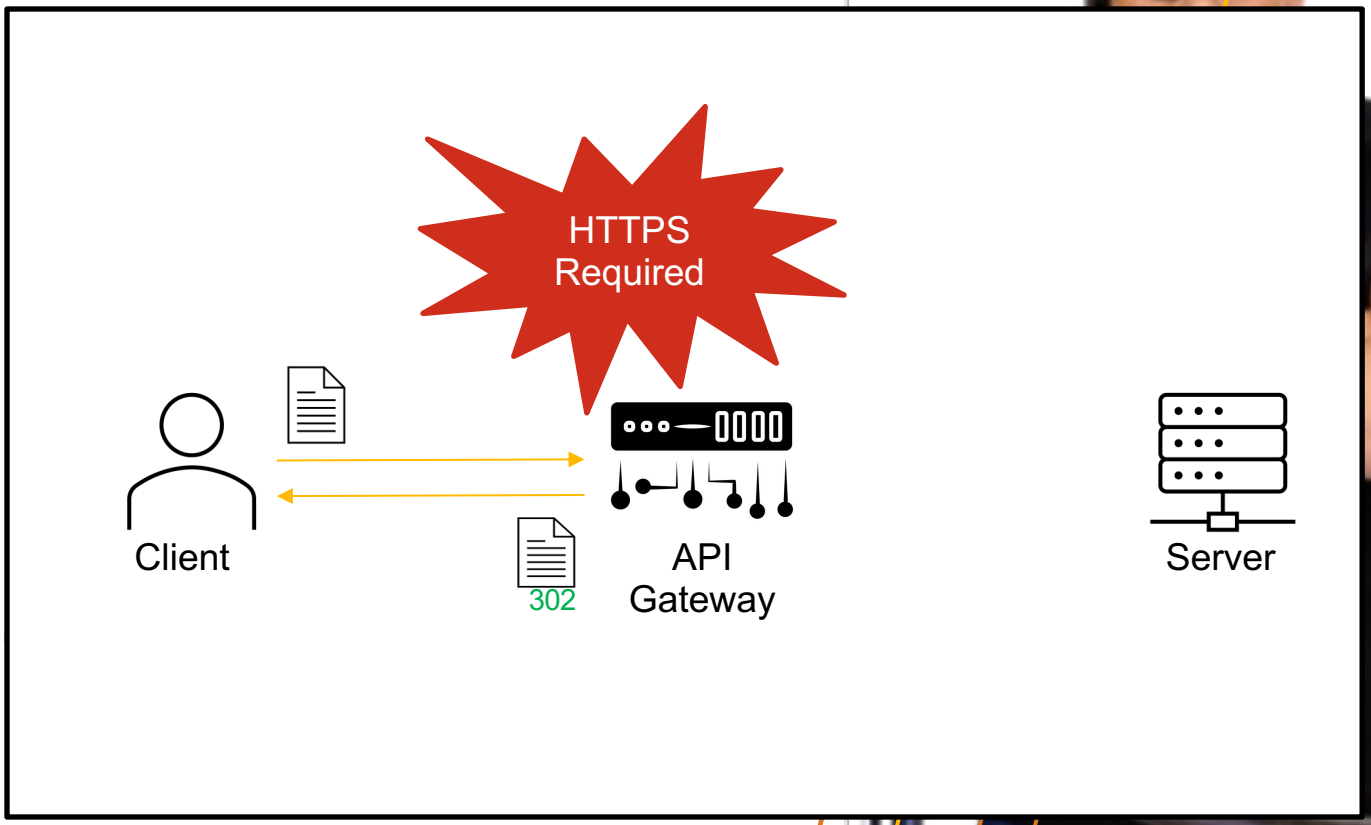
# Patterns: Spy



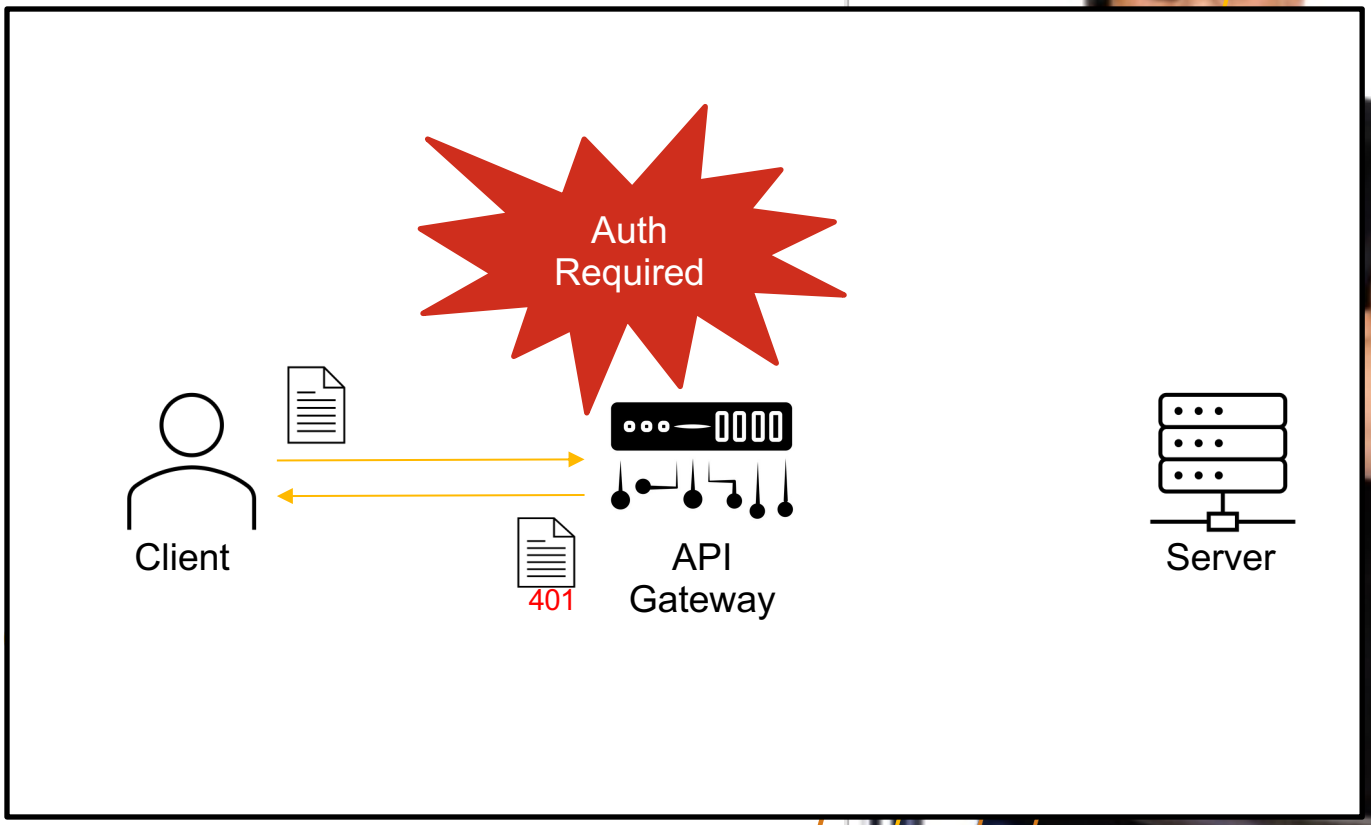
# Patterns:
# Bouncer

- Security checkpoint for all of your services
- Great for:
  - Enforce HTTPS traffic
  - Providing client filtering, for example by IP address
  - Consistently applying required headers (CSP, Referrer Policy)
  - Limiting request/Body sizes
  - Preventing direct access to services (reduce public footprint)
  - Provides an avenue for CVE mitigation
  - Effectively perform responsibilities of OAuth2 Resource Server
  - Apply RBAC rules to endpoints, often with OAuth2 scopes
  - Rate Limiting clients

# Patterns: Bouncer



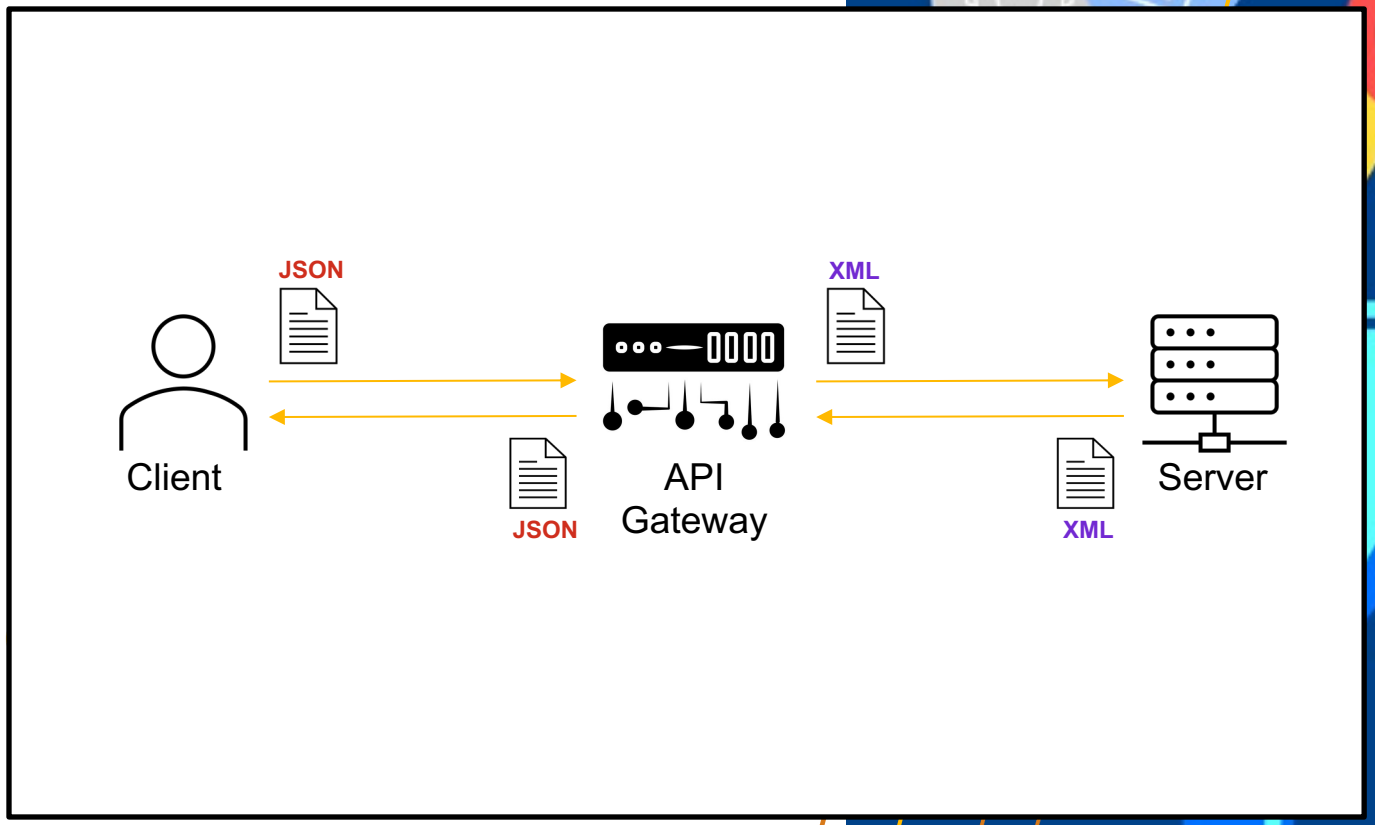
# Patterns: Bouncer



# Patterns:
# Transformer

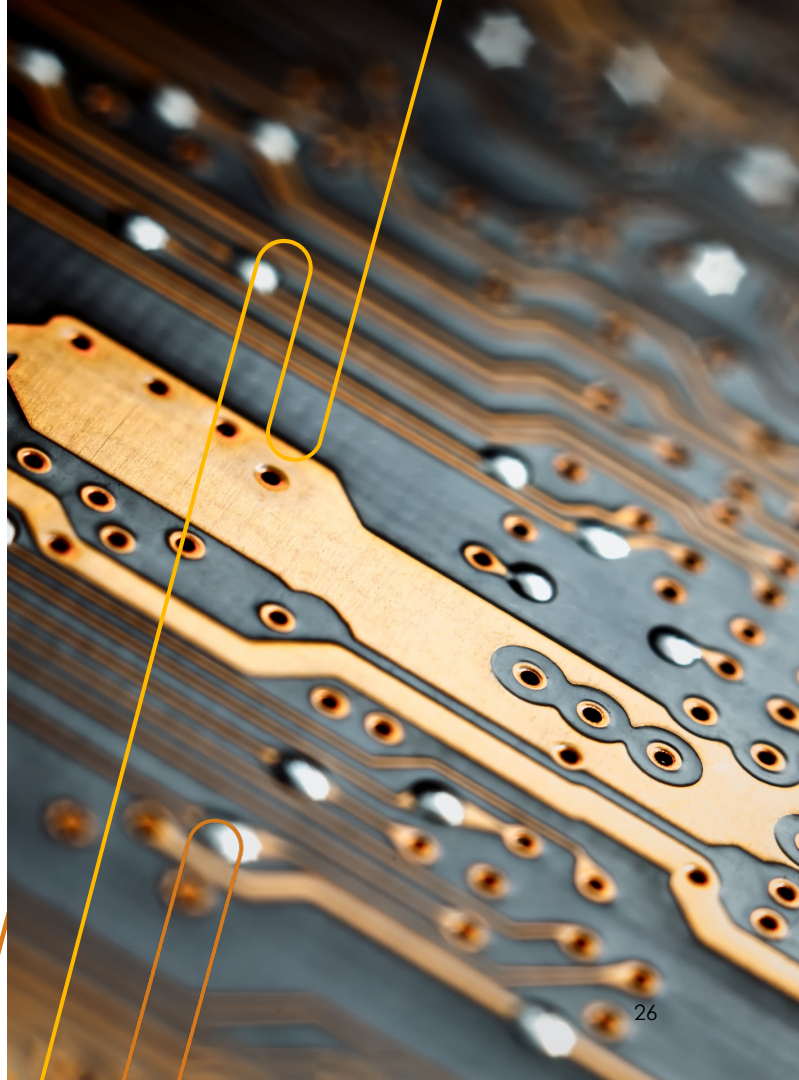
- Adjust and modify your data on-the-fly
- Great for:
  - Making client-specific APIs, common with IoT devices
  - Transitioning between API versions
  - Translate between protocols (REST to gRPC or REST to AMQP)
  - Providing consistent request/response conventions
  - Hiding that weird old service

# Patterns: Transformer

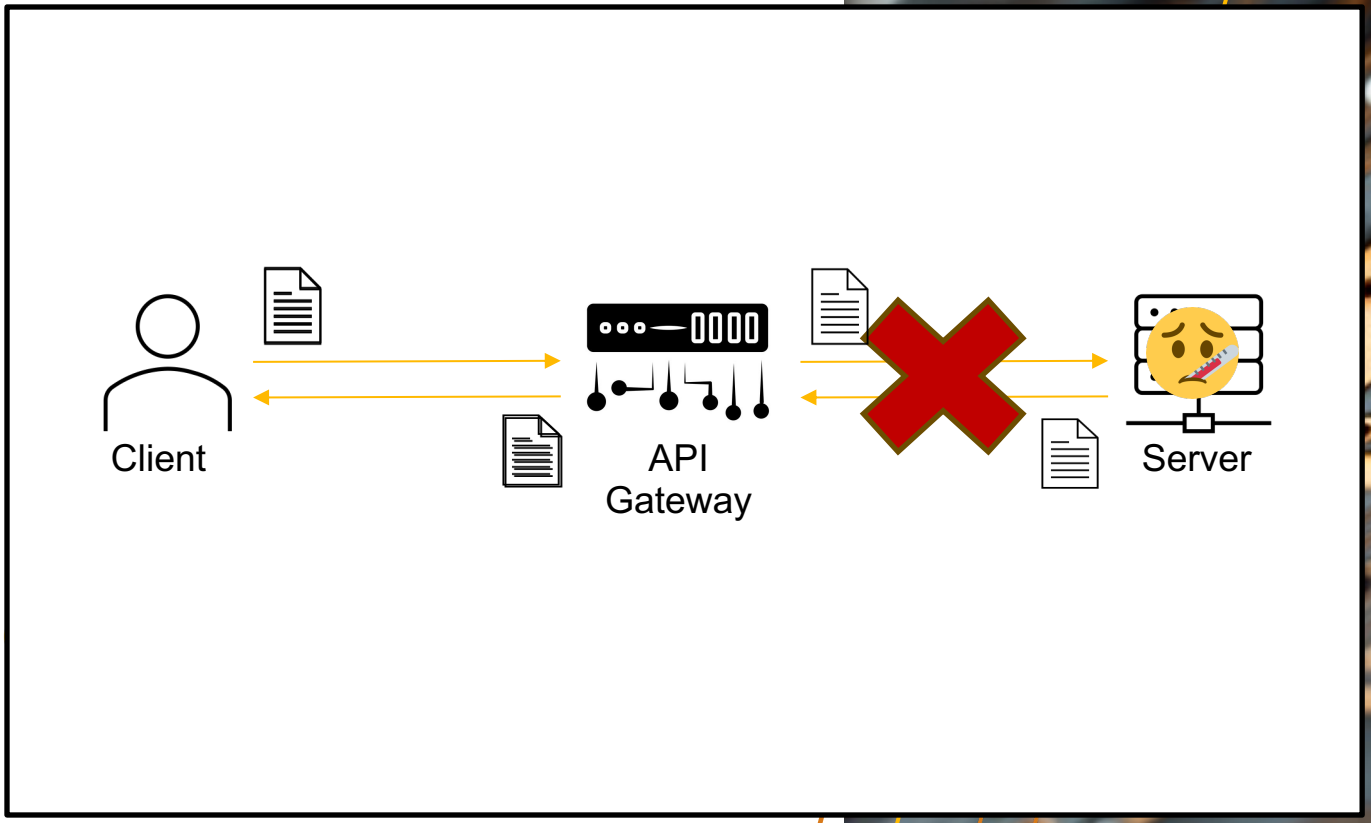


# Patterns:
# Circuit Breaker

- Automatic resiliency for your services
- Great for:
  - Deflecting traffic when there are service issues
  - Providing fallback behavior when services are overloaded
  - Defining consistent retry behavior for services



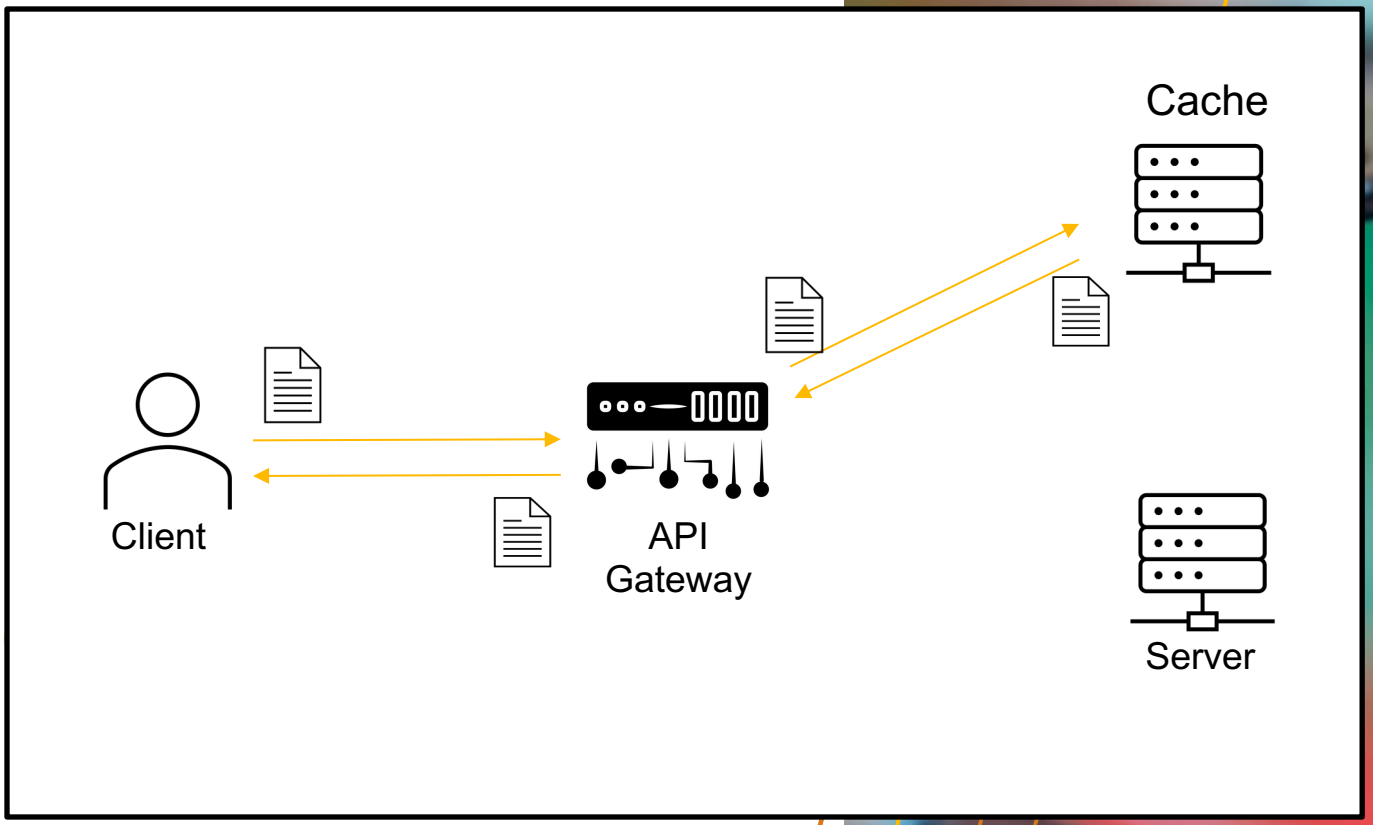
# Patterns: Circuit Breaker



# Patterns:
# Cache

- Reused previous responses to decrease response time
- Great for:
  - Making slow services seem faster
  - Reducing load on services
  - Reducing volume to potentially expensive services

# Patterns: Cache



# Progress

01. Intro to APIs

02. API Gateway Patterns

03. Demo

# Part Three

## Spring Cloud Gateway Demo

# Demo

# Questions?