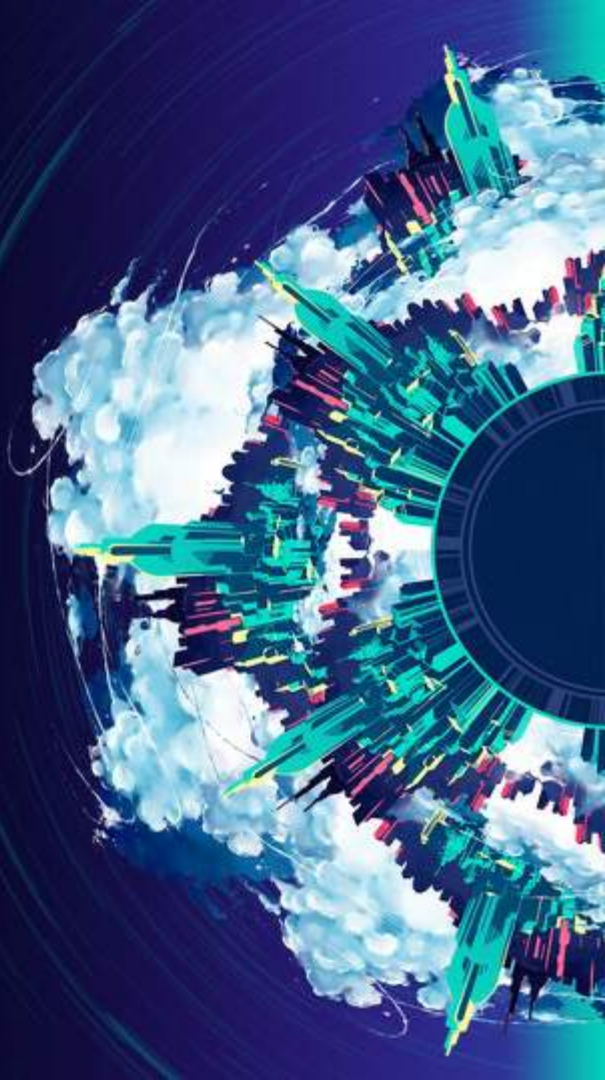
API Days workshop

# API Gateway & IdP, a Match Made in Microservices Heaven



## About me

Yaara Letz

Consulting &&/|| Engineer @Tyk

London, UK

## About Tyk

API Management Platform:

- API Gateway
- Management Dashboard
- Developer Portal

Tyk lets you easily manage your APIs:

- Protect and secure APIs
- Publish and document
- Developer self-service enrolment
- Microservice architecture support

Offices in London and Singapore

## Today's talk

- Privacy and Data breaches
- Keyless APIs
- Options for microservice Authentication and Authorization
- Our authentication and authorisation methods
- Identity Management and API management integration
- Demos
- Q & A

## Privacy, regulation and Data breaches

Privacy - Personal information is collected via Emails, Social media, Smart speakers

GDPR - General Data Protection Regulation

Data breaches **– invest in security today to avoid to avoid reputational and financial**

**consequences tomorrow**

## Story time - Domino's Pizza - Keeping your data safe

## How does it apply to APIs?

- APIs are vulnerable too
- Facebook

  Graph API of early version of FB’s API

  50 million users had their profiles harvested by Cambridge Analytica
- Uber

  Sensitive information of 57 million users and drivers has been compromised
- Steps to take:

  - Not storing personal data and verification dev tools could have saved Uber

  - Authorisation – proper authorisation and versioning enforcement could have helped FB

## When to apply authentication and authorisation?

Almost always!

Consumers can negatively affect your API, whether intentional or not

Degraded API performance affects all consumers

## Keyless APIs –What, When, Why, How

1. What is a keyless API?
2. When do I make APIs keyless?
3. Is it dangerous? Why?

## Answer 1 - Keyless APIs - unofficial definition

- An unprotected API – open for everyone
- No need to supply any key/token/password/identifying details when call it
- It’s consumer/caller is unauthenticated and **usually** unauthorised and unlimited

## Answer 2 - When do people use keyless APIs

All are more of the same:

- When I want everyone to access my APIs
- When I want barriers-free access
- Drive extremely easy adoption and usage of my APIs
- When I'm busy and it's internal usage (hint: bad decision)
- When I want everyone to access my APIs freely
- Even Google have some limited keyless access

## Answer 3 – Yes, keyless API is dangerous, but Why?

- Back door - Give the world direct access to your backend services i.e. your code
- Risky in case of a bug -
  - Might expose your services
  - or worse – expose your clients data
- Usually no rate limit and quota are enforced - risk overloading your API service
- Not danger, but loss for the business:
  - Harder to get clients' details without some kind of identification and mutual contract
  - No way to segment your API traffic by users

➔ Solution: Use an API gateway ☺

## Always use an API gateway

- Another protective layer
- Prevent overload by rate limiting the access per API
- Rate limit by IP address
- Quota per API and/or IP or other recognisable headers
- Whitelist and blacklist endpoints of the API
- Redirect keyless APIs to special services in the DMZ or other locations
- Restrict access only to the gateway –

  - Can easily block network access to anyone but the gateway

  - Client-side Authentication (Mutual TLS)

  - Authentication and authorisation of the gateway as a client in the realm

## Conclusion for keyless APIs

- Limit keyless access only to specific APIs
- Open only the necessary endpoints
- Use an API gateway with URL redirection
- Make sure to gather/aggregate analytics for headers such as User-Agent.

## Why Microservices – a brief overview

Benefits:

- Lightweight simplicity
- Flexibility
- Compatible with modern approaches
- Reduced risk

## Options for microservice Authentication and Authorization

1. Internally within each microservice
2. Externally by a gateway
3. Combination of internal and external

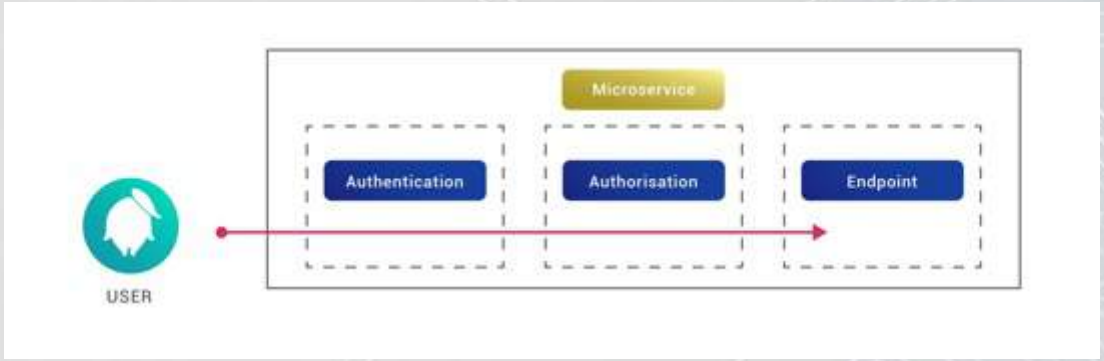
## Microservices - internal approach

Pros:

- Gives fined grained control
- Self-reliance

Cons:

- More development effort
- Larger microservices
- Copy/paste code



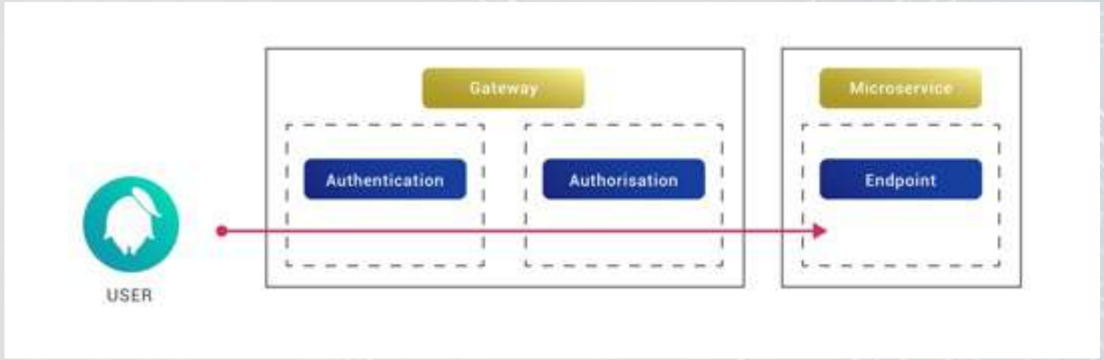
## Microservices - external approach

Pros:

- Centrally handled
- Enforce one method
- Less development effort
- Smaller microservices

Cons:

- Vulnerability
- Less control



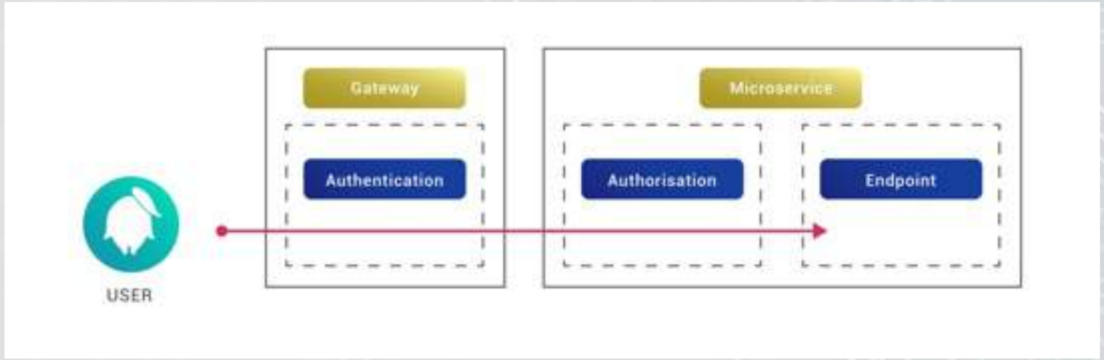
## Microservices - combined approach

Pros:

- Relieves burden of authentication
- Gives control of authorisation

Cons:

- More development effort
- Larger microservices



## Choosing the right authentication Methods

- ? Regulations industry
- ? Consistency
- ? Scenarios
- ? Reusing  with IdP

## Authentication Methods On The Gateway

- **Basic Auth** – Require migration of users and their passwords
- **Bearer token** –specific to the gateway.
- **OAuth2.0 flows** – Still specific but with short lived token and refresh token
- **OIDC –** No need to update the IdP except for creating a client_id
- **Generic JWT –** Require ability to update the JWT
- **Certificate authentication–** create a Key based on a provided client certificate
- **Custom Auth** – using plugin to extend Tyk for specific scenarios

## Identity providers

Factors contributing for to the Identity Providers market

- Regular data breaches
- Increasing regulation
- Authentication flow complexity

Identify providers will take care for you for

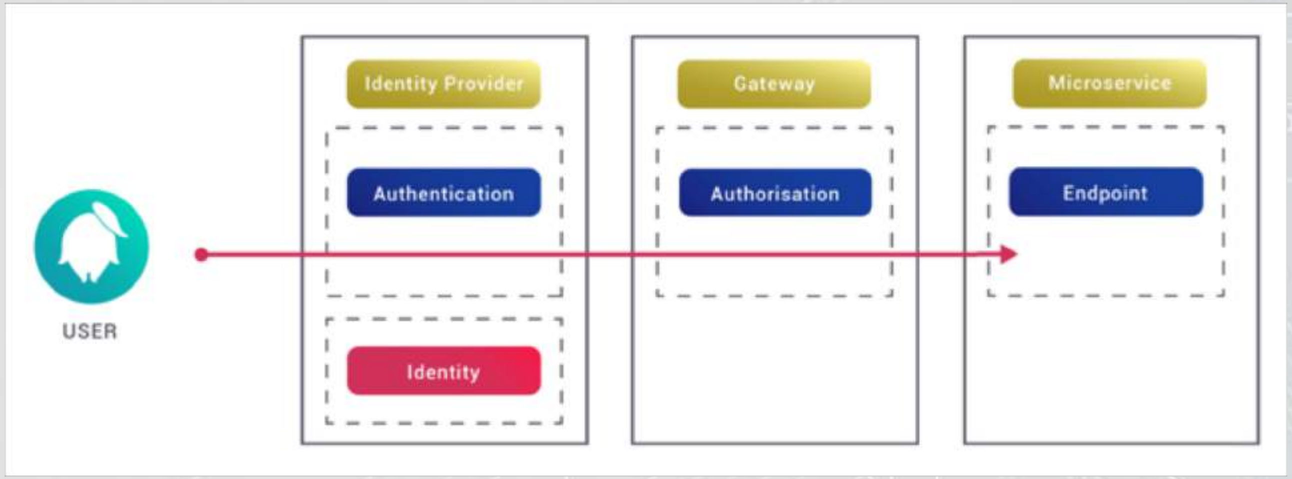
- Authentication –many standards and protocols
- Single Sign-On  - in the intranet and internet
- Multi-factor authentication –emails OTP, SMS, USB key etc.
- Management multiple user’s identities

## Focus on identity providers

System for managing identity

Provides authentication as a service

Utilise as a central hub for all of your identity needs



## Demos

Three demos:

1. Securing an API with a generic JWT
2. Securing an API with OIDC flow
3. Login as admin user into Tyk Dashboard using Okta as my Identity provider

Using the Tyk Identity Broker

- Enables authentication against different platforms

Demo 1: Securing an API

## Using a generic JWT

Demo 2: Generate API token

## OIDC with Okta

Demo 2: Dashboard login

## OpenId Connect with Okta

## Conclusion

There is no single best solution

Pick the right architecture to suit you need, but favour external

Use an API gateway to simplify authentication and authorisation

Choose an authentication method to suit your situation and your consumers, but be consistent

Use an identity provider to simplify identity management and authentication

## Q & A

twitter.com/tyk_io

@yaarale

github.com/TykTechnologies

facebook.com/Tyk.API.Management

Tyk