Distributed API Management in a Hybrid Cloud Environment

Thomas Siegrist, Head of Integration, SBB
Christian Sanabria, Principal Architect, ipt
Chris Eberle, Senior Solution Architect, Red Hat
Storyline

1. Overview / Motivation (TS)
2. Architecture / Implementation (CS)
3. Collaboration / Evolution (CE)
4. Key Takeaways / Roadmap (TS)
5. Q&A (all)
in a dematerialized world

classic industries must embed their physical assets in an ecosystem of software and services in order to create new customer journeys
APIs @ Swiss Federal Railways (SBB)

The outside goal - APIs represent physical products

- Train Schedule API
- Booking API
- Customer Info API
- ...
APIs @ Swiss Federal Railways (SBB)

The inside goal - we want to keep the overview of the application mesh

1. 2 Openshift clusters
   a. OSE SBB on private IaaS
   b. OSE AWS on public IaaS
   c. ... more will follow

2. 900 applications with all kinds of SLAs
3. 3000 containers
4. 50 Mio. requests per day
APIs @ Swiss Federal Railways (SBB)

The inside goal - every application exposes it’s assets over a managed API
implementation
3scale Architecture
From a very high altitude

- Distributable and scalable API GW
- Centralized API Mgmt and Dev Portal
- Shared API GW for common use cases
- Dedicated API GW for special needs
- Provides authentication / authorization
- OAuth as standard token based security
Developer Portal

What you see is what you get
Implementation Details
3scale product

- APIcast Template
  - Standalone Gateway
- APIcast Customizations
  - Frontside CORS
  - Backside 2-Way-SSL
  - Token Cache
- Developer Portal Customizations
  - apis.json Backend
  - Customized CMS
RHSSO
A byproduct

- Simple and lightweight Identity Broker
- Easy integration of common external IdPs
- Easy integration of SAML and OAuth based IdPs
- Provides single point of entry for OAuth based SSO
- Strategic SSO solution for SBB
Automation
From config to API

- 3scale
- Keycloak
- Route53
- Sandbox
- SBB CFF FFS Developer Portal
- Spring Boot
- Nexus
- Jenkins
Monitoring & Logging

Big Brother is watching you
Implementation Details

3scale utilities

- Automation Microservice
  - The whole API with one request

- Synchronization Microservice
  - Keep 3scale and RHSSO in sync

- Developer Portal Backend
  - Provide a marketplace view for APIs

- Platform Automation
  - 3scale and RHSSO in one click
NginxWAF
A byproduct

- Highly distributable and scalable WAFs
- Provides threat protection
- Shared WAF for common use cases
- Dedicated WAFs for special needs
collaboration
Out of the box

Developer Portal:
- API Provider Branded
- API Description
- Signup
- Active Docs (OAS)

Admin Portal:
- Dashboard
- Developer / Application / Key Management
- CMS
- Analytics
- Billing

API Manager:
- System
- Config
- Authorize & Report Traffic

API Gateway:
- API Request
- Authorized API Request

API Backend:

Developer Apps:

API Backend:

Authorized API Request:

Authorized API Request:

Authorized API Request:
Decoupling
Caching policies

- API Manager
  - System
    - Config
    - Backend
  - Developer Portal
  - Admin Portal

- API Request
  - Authorized API Request
- Developer Apps
- API Gateway
- API Backend
Decoupling
Java Based Gateway

API Request
Authorized API Request

Developer Apps
API Gateway
API Backend

Developer Portal
API Manager
System
Config
Backend
Admin Portal

See: https://github.com/SchweizerischeBundesbahnen/apim-adapter
Redis HA
On OpenShift, on AWS

See: [https://github.com/openlab-red/redis-ha](https://github.com/openlab-red/redis-ha)
System Database

From MySQL to Oracle DB

Diagram:
- Developer Portal → API Manager → System → Config → Backend
- API Manager → Admin Portal
- Developer Apps → API Request → API Gateway → Authorized API Request → API Backend
Developer Portal

Angular Based Frontend

See: https://github.com/SchweizerischeBundesbahnen/api-devportal
key takeaways
Key takeaways

Highly scalable and distributed means accepting some tradeoffs

Scaling API Management in a distributed world means ...

- every dependency is called asynchronously
- gateways seek maximum independence in runtime

We prioritize A & P over C in CAP theorem, means ...

- some loss of analytics (low %)
- some traffic exceeding rate limits

Like this we will be able to scale up from now 3’000 up to 100’000s of requests per second.
Roadmap

Next steps

- rollout asynchronous gateways in two options
  - as a gateway
  - as a spring-boot library

- onboard the most frequented APIs

- implement support for streaming APIs
questions