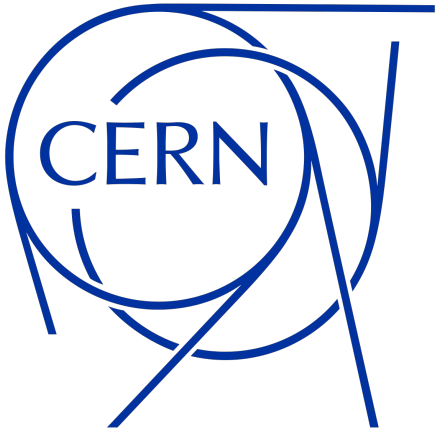


Architectural Patterns for Spring Security You Wish Your Tech Lead Knew

Cristian Schuszter, PhD

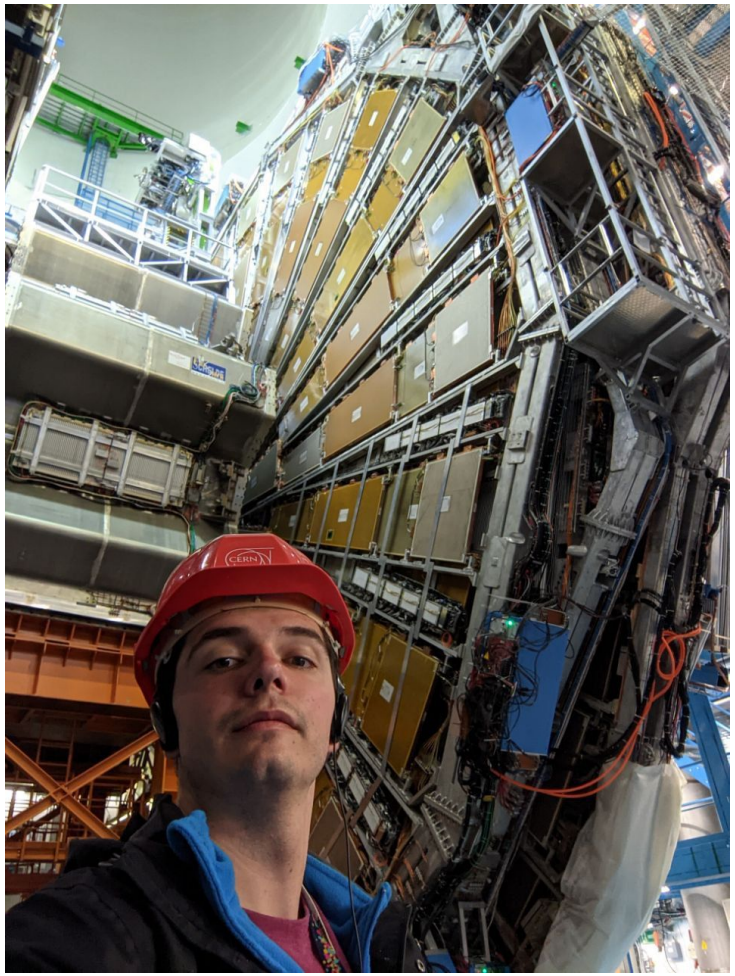
15 April 2026, Spring I/O 2026

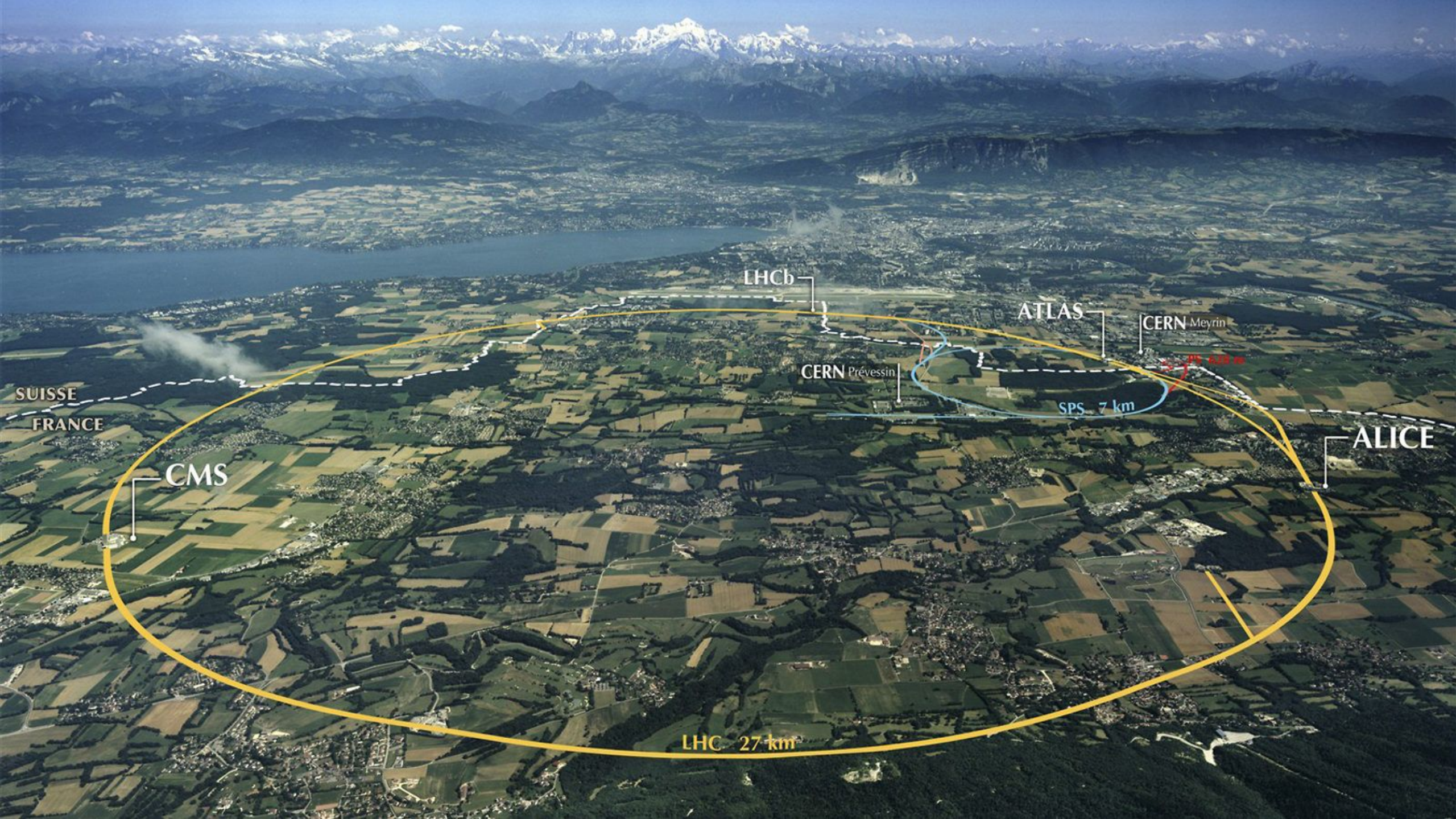
$ whoami

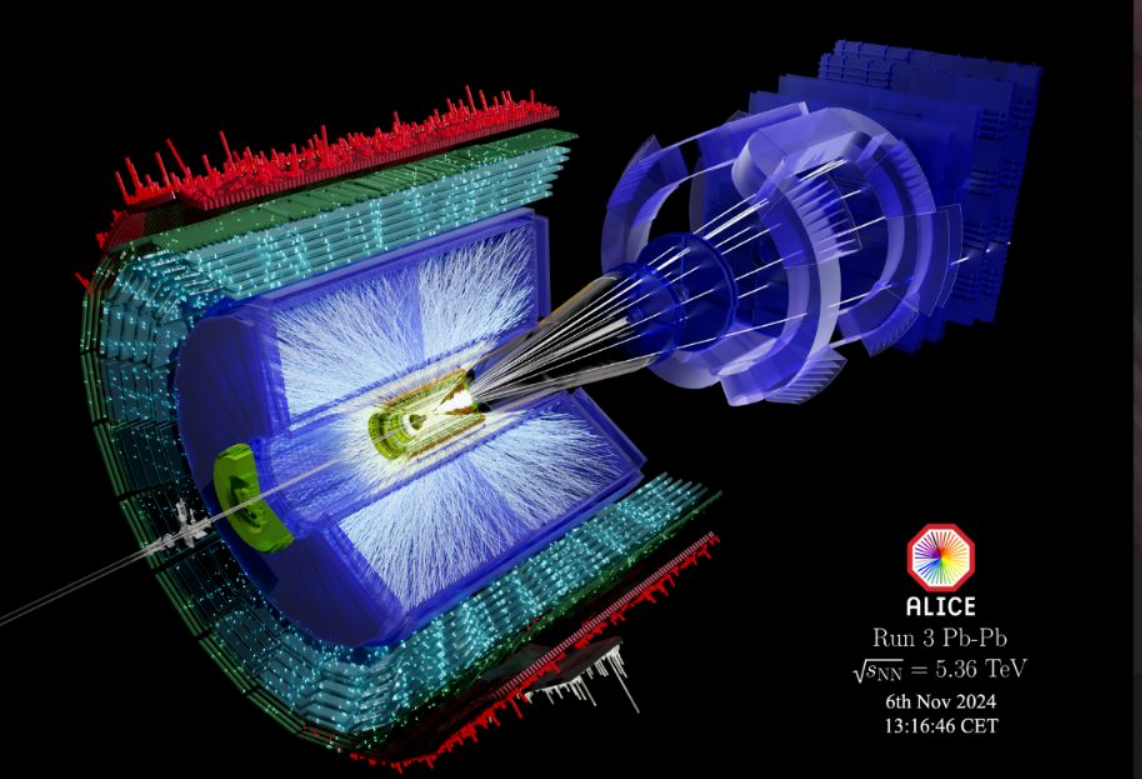
- *Cristian Schuszter*
- **Staff Software Engineer @ CERN**
- **Solution Architect**

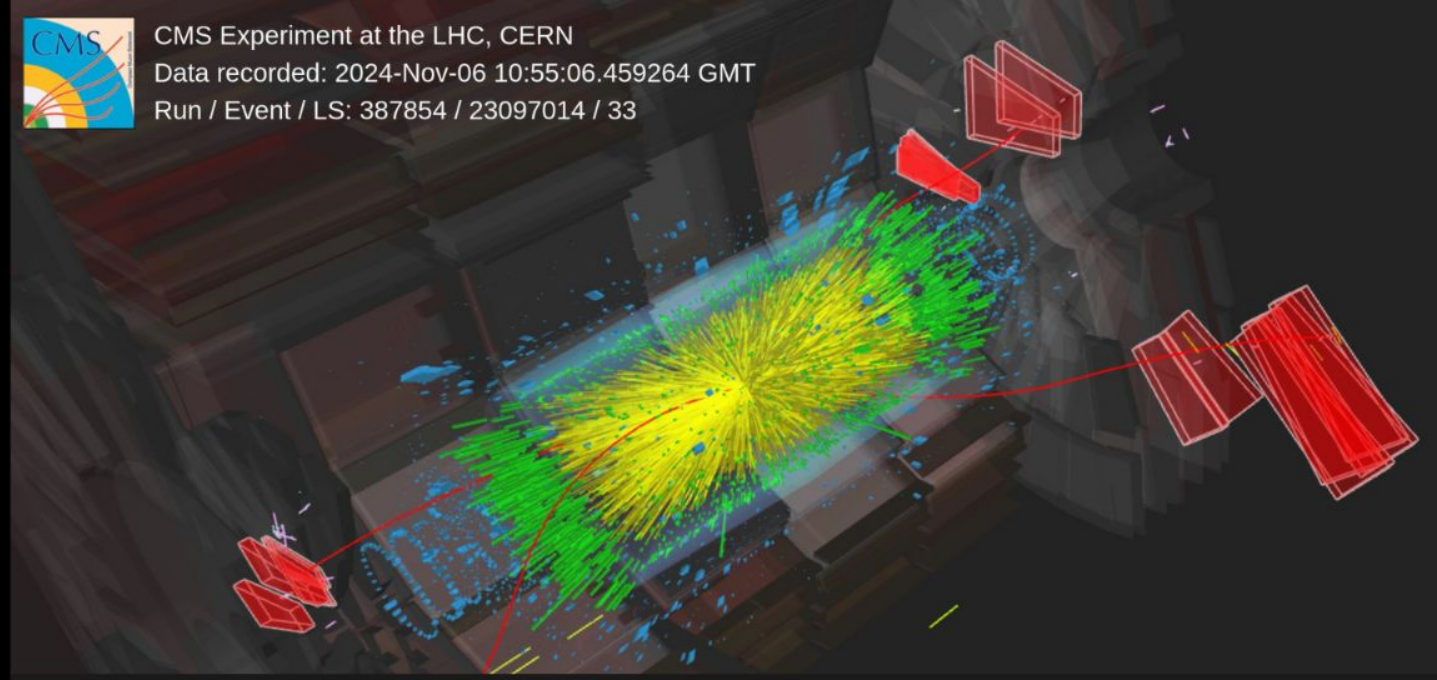
- **Been around for ~ 8 years, 12 years exp.**
  - computer security
  - SSO team
  - now working in “business computing”

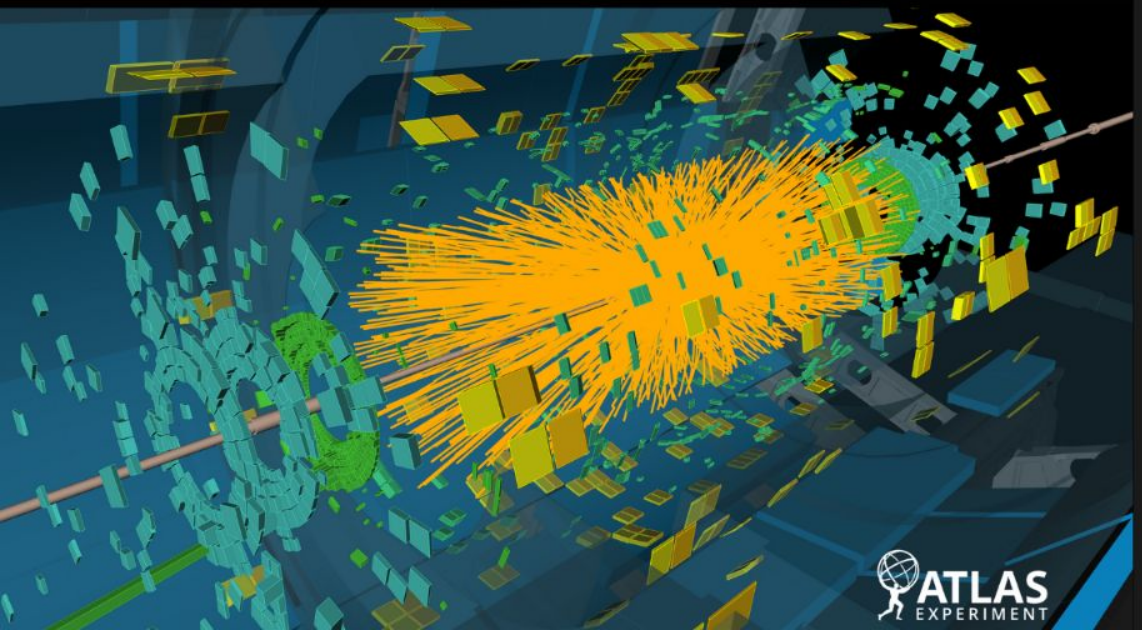
- **CERN guide in my spare time**

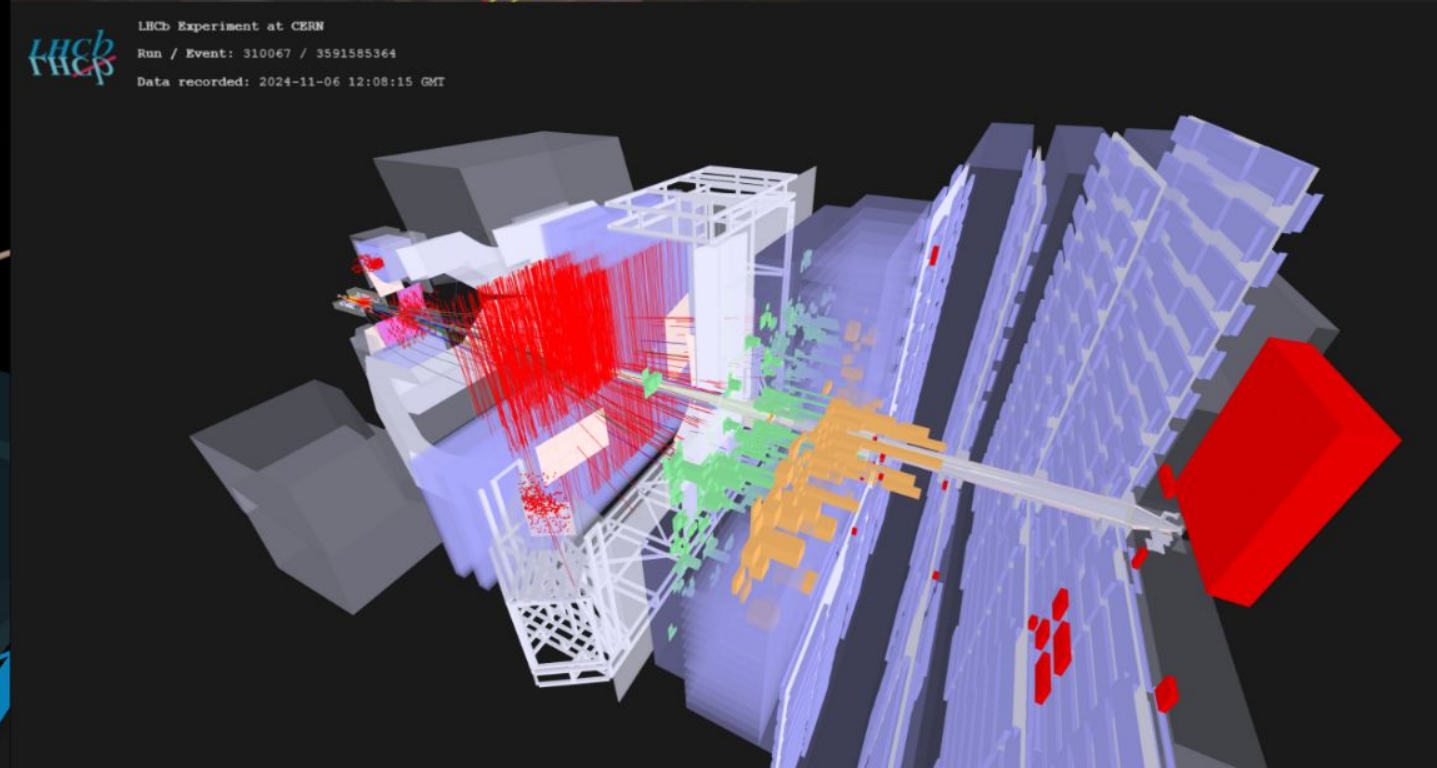



LHCb
ATLAS
CERN Meyrin
CERN Prévessin
SPS 7 km
SUISSE
FRANCE
CMS
ALICE
LHC 27 km


ALICE
Run 3 Pb-Pb
$\sqrt{s_{NN}} = 5.36$ TeV
6th Nov 2024
13:16:46 CET


CMS Experiment at the LHC, CERN
Data recorded: 2024-Nov-06 10:55:06.459264 GMT
Run / Event / LS: 387854 / 23097014 / 33


ATLAS
EXPERIMENT


LHCb Experiment at CERN
Run / Event: 310067 / 3591585364
Data recorded: 2024-11-06 12:08:15 GMT

CERN Single Sign-On

English

Sign in with a CERN account

Username

Password

Sign In

Forgot Password?

Or use another login method

Kerberos

By logging in, you agree to comply with the **CERN Computing Rules**, in particular OC5. CERN implements the measures necessary to ensure compliance.

Sign in with your email or organisation

Home Organisation - eduGAIN

Email - Guest Access

Sign in with a social account

By clicking on the buttons below, you consent to CERN's transfer of your login request to the social provider and to receive your account name, name and e-mail for authenticating you. See more details in our **Privacy Notice**.

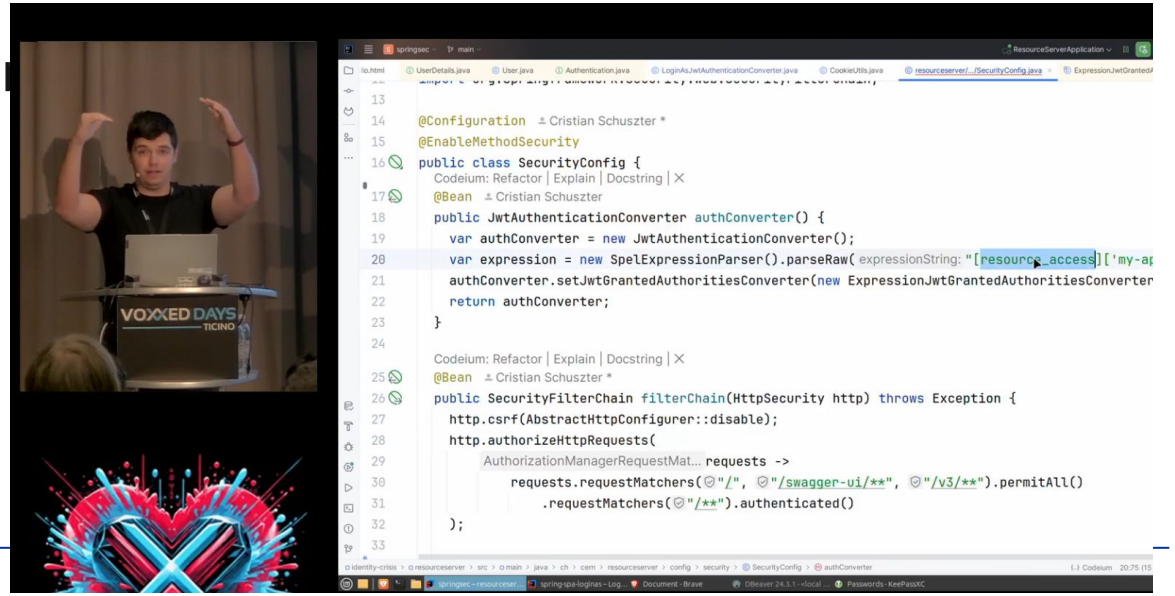
Google | GitHub

Facebook | LinkedIn

From the last episode

- OIDC vs OAuth
- Resource Server vs OAuth Client
- What type of flow is useful for you and why?
- More advanced use-cases ->

  mangling your tokens to achieve impersonation



https://youtu.be/s2hZSCuhxW4

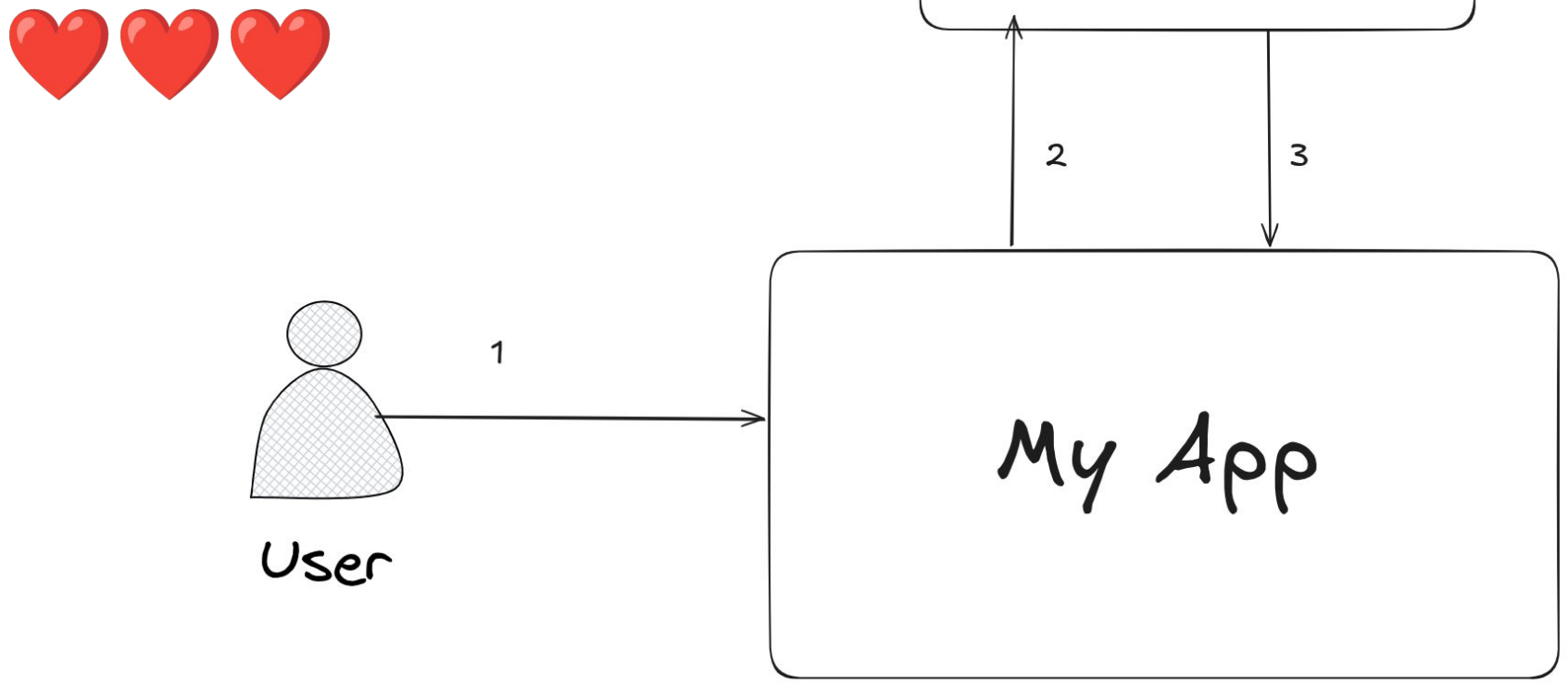
This time

- You got past your first service! Things are ramping up! What now?
- Tips & tricks for getting the best setup possible - AuthN / AuthZ
- Spring Security in a single service == solved
  - how about in an ecosystem?

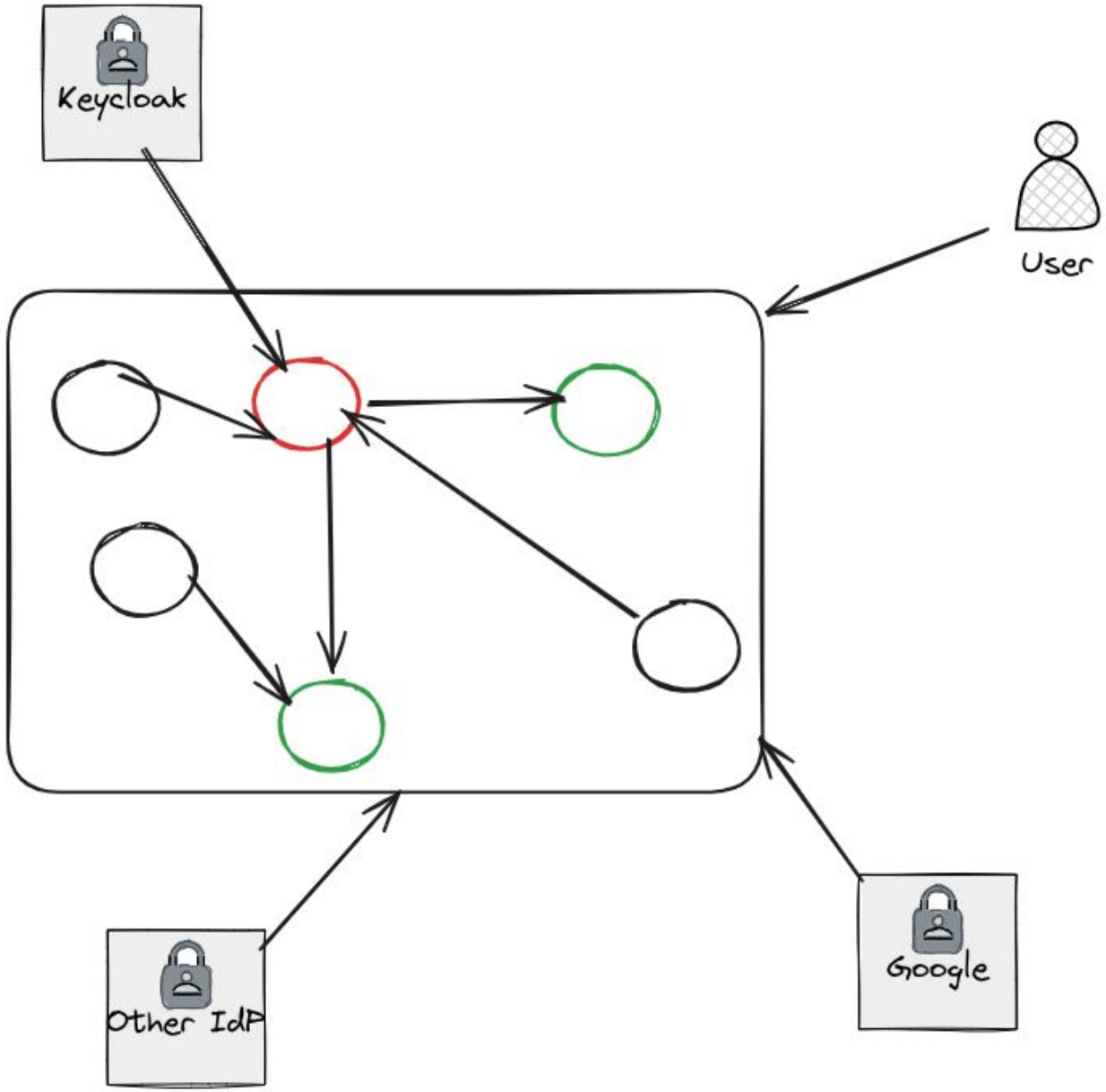
Topics

- **Scaling - the pain points**
- Multi-IdP federation concepts
- AuthN & AuthZ patterns
- Gateway integration patterns

Trouble in paradise

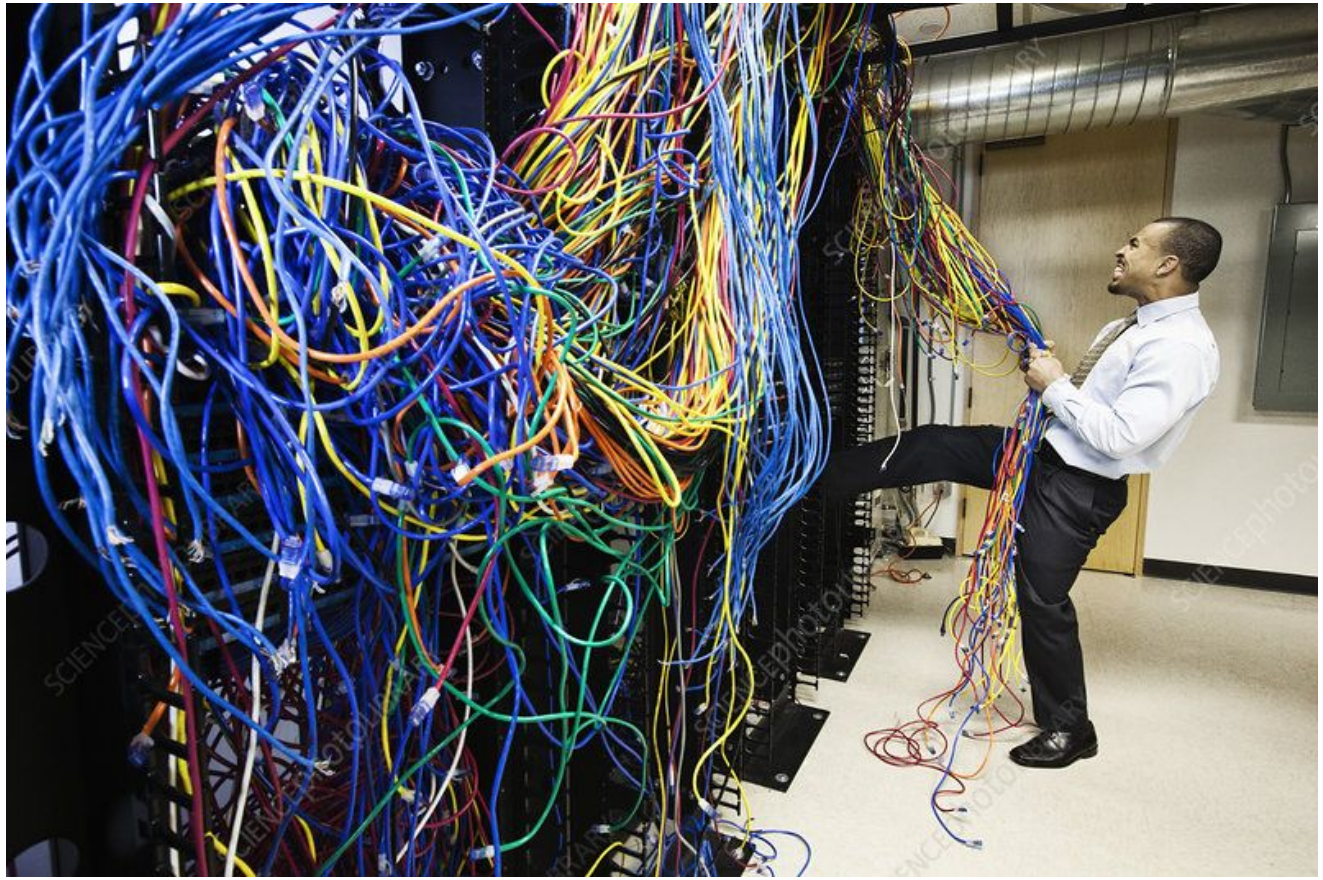


Trouble in paradise



Trouble in paradise

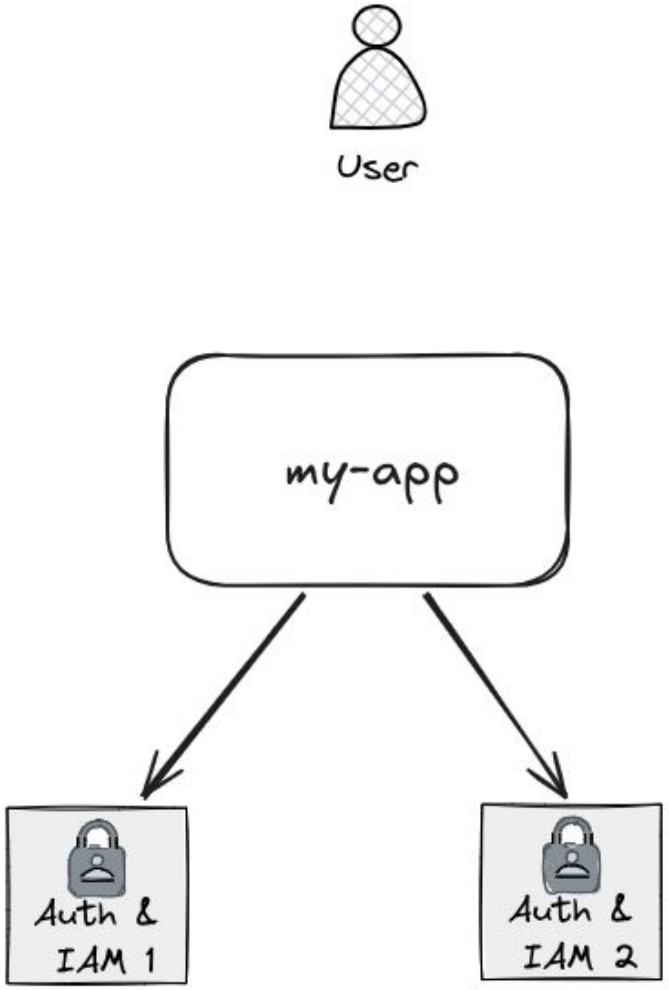
Failure to look out for: every service does its own thing



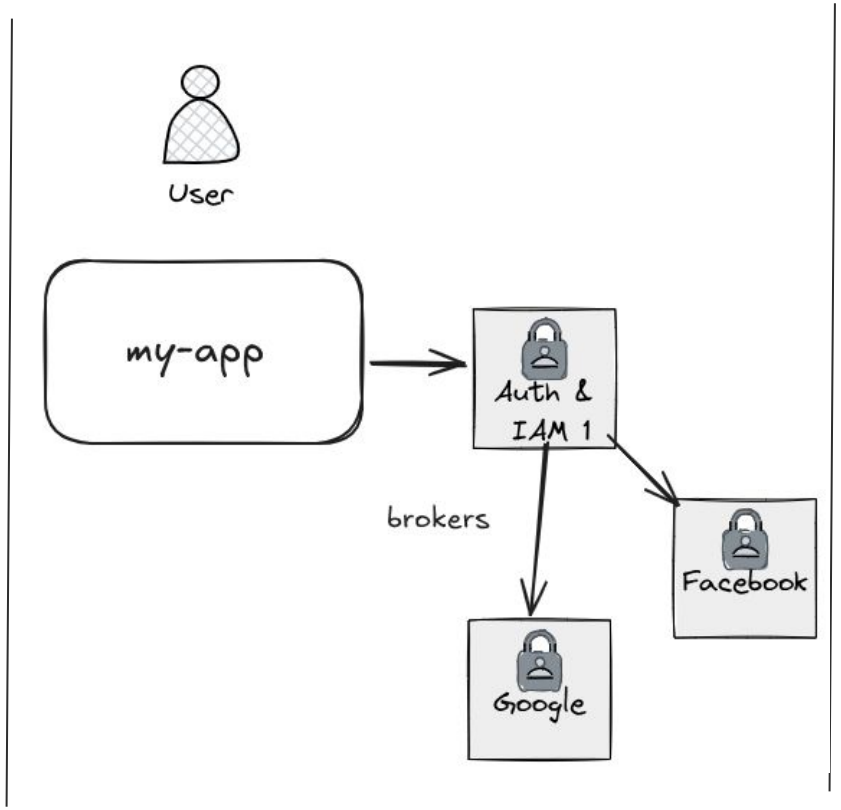
Topics

- Scaling - the pain points
- **Multi-IdP federation concepts**
- AuthN & AuthZ patterns
- Gateway integration patterns

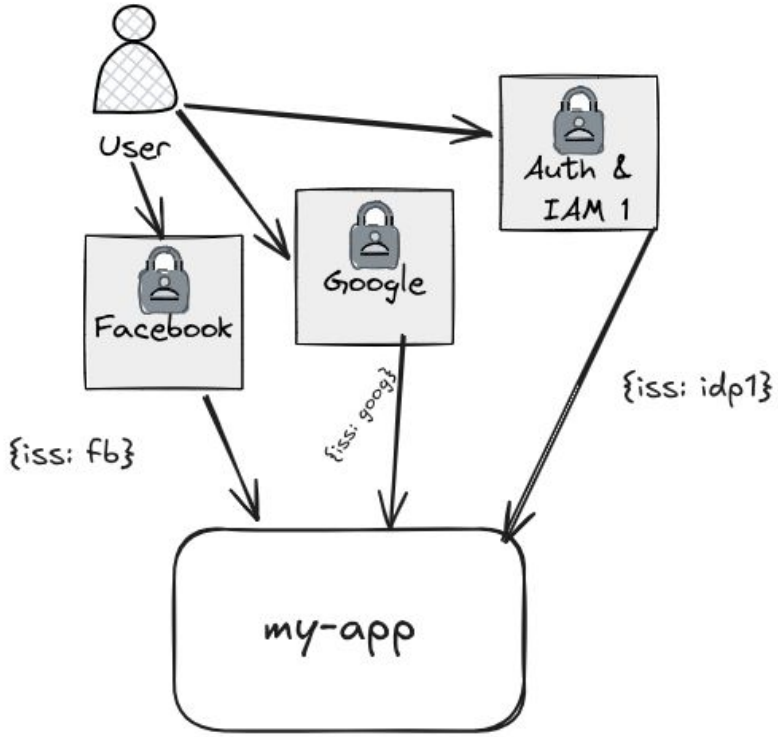
Identity provider (IdP) multi-tenancy



1. Multi-tenant client

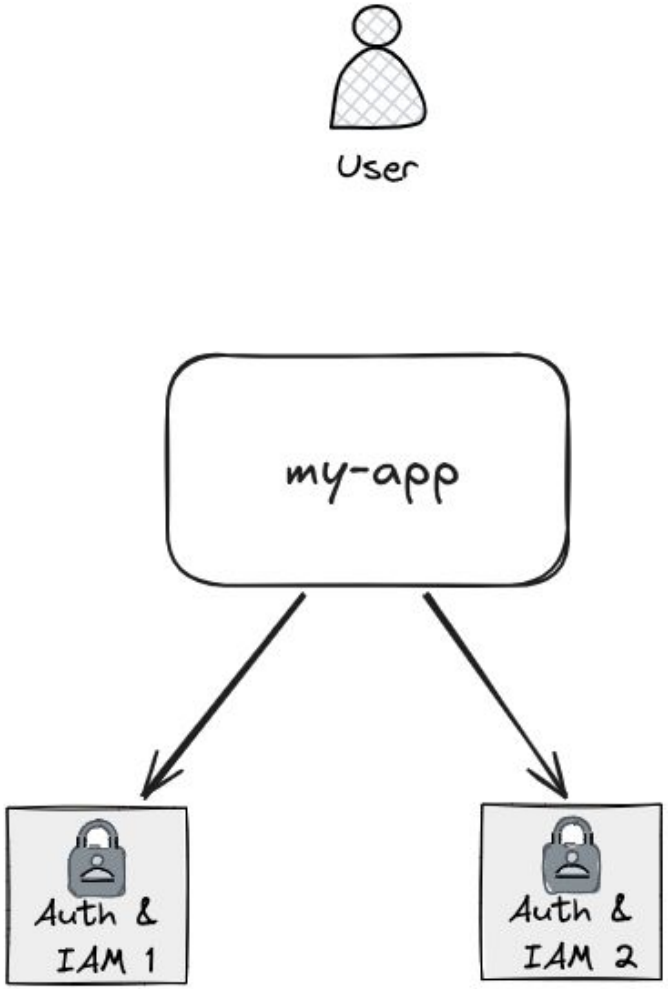


2. IdP brokering

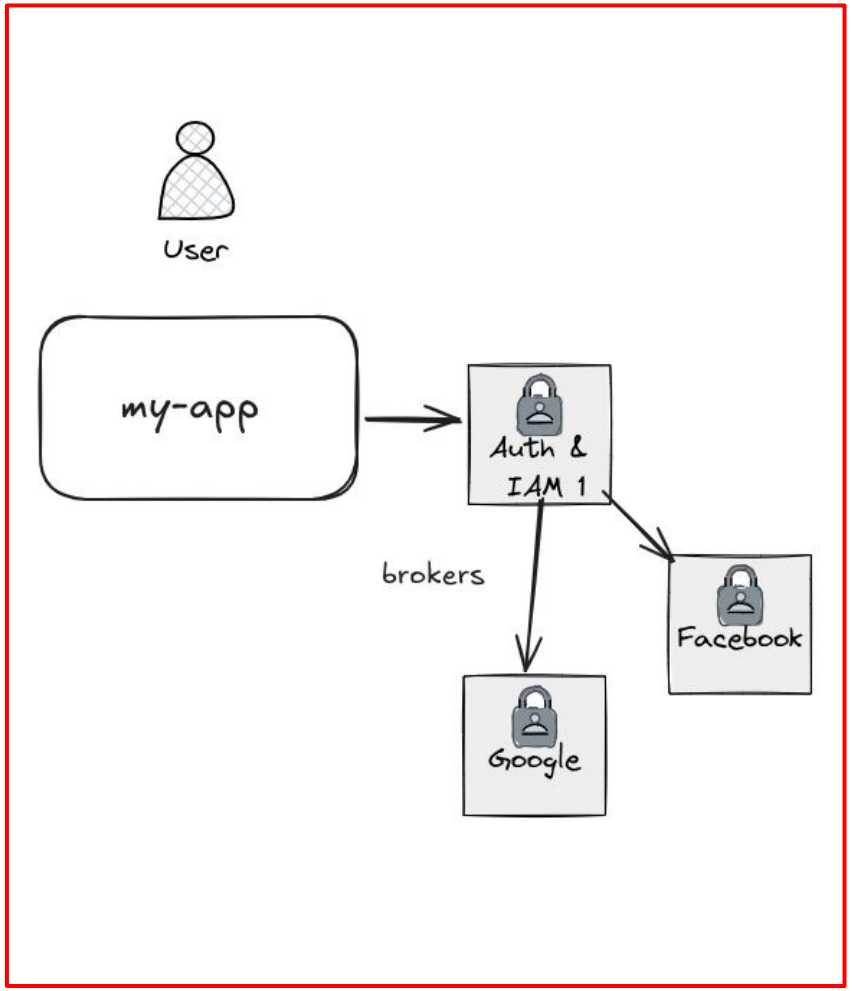


3. Multi-issuer resource server

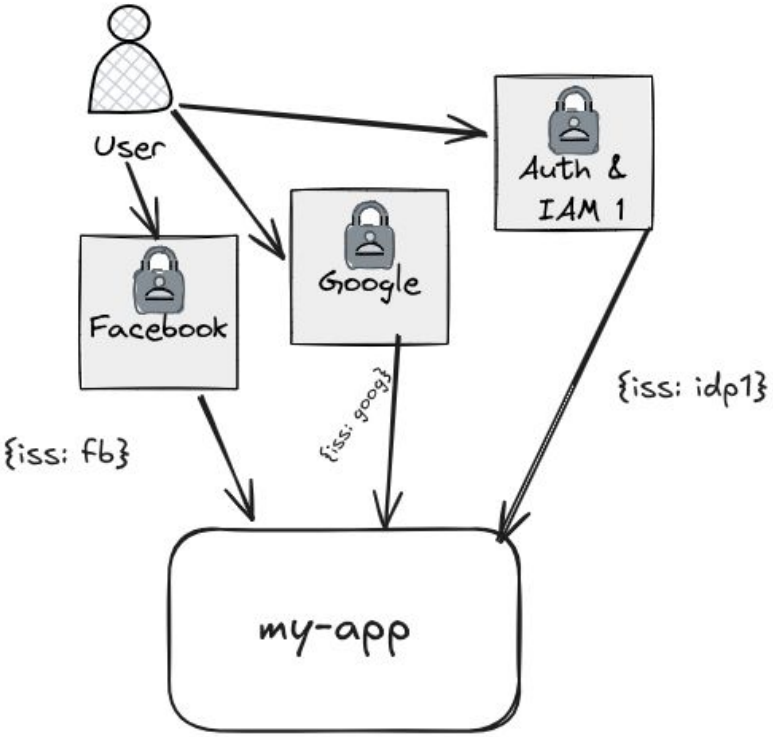
Identity provider (IdP) multi-tenancy



1. Multi-tenant client



2. IdP brokering



3. Multi-issuer resource server

CERN SSO

English

Sign in with a CERN account

Username

Password

Sign In

Forgot Password?

Or use another login method

Kerberos

By logging in, you agree to comply with the CERN Computing Rules, in particular OC5. CERN implements the measures necessary to ensure compliance.

Sign in with your email or organisation

Home Organisation - eduGAIN

Email - Guest Access

Sign in with a social account

By clicking on the buttons below, you consent to CERN's transfer of your login request to the social provider and to receive your account name, name and e-mail for authenticating you. See more details in our Privacy Notice.

Google | GitHub

Facebook | LinkedIn

Multi-tenant client (multiple IdPs)

spring-boot-starter-security-oauth2-client

+

Custom **OAuth2AuthorizationRequestResolver**

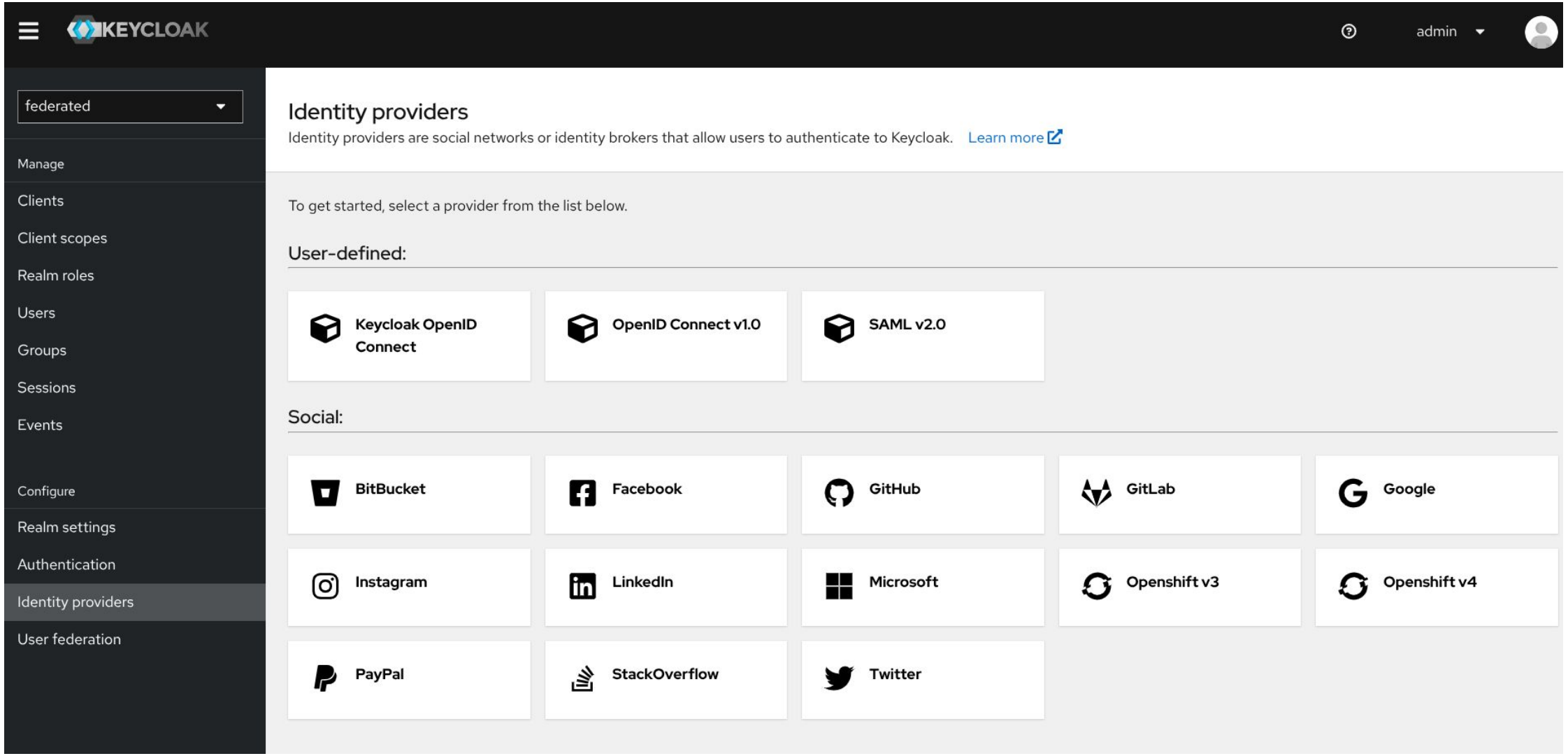
```yaml
spring:
  security:
    oauth2:
      client:
        registration:
          keycloak:
            client-id: my-app
            authorization-grant-type: authorization_code
          federation:
            client-id: federation-test
            authorization-grant-type: authorization_code
        provider:
          keycloak:
            issuer-uri: http://localhost:8080/realms/app
          federation:
            issuer-uri: http://localhost:8080/realms/federated
```

Multi-tenant resource server

```
@Bean
AuthenticationManagerResolver<HttpServletRequest> tokenAuthenticationManagerResolver
        (JwtDecoder jwtDecoder, OpaqueTokenIntrospector opaqueTokenIntrospector) {
    AuthenticationManager jwt = new ProviderManager(new
JwtAuthenticationProvider(jwtDecoder));
    AuthenticationManager opaqueToken = new ProviderManager(
            new OpaqueTokenAuthenticationProvider(opaqueTokenIntrospector));
    return (request) -> useJwt(request) ? jwt : opaqueToken;
}
```

https://docs.spring.io/spring-security/reference/servlet/oauth2/resource-server/multitenancy.html

IdP brokering



Token normalization

Design Goal: Stable AuthZ, flexible AuthN

```
{
    "jti" :
"cf241ba3-7b4c-4fac-923c-6e985f78ace5",
    "iss" : "http://localhost:8080/realms/app",
    "aud" : "account",
    "sub" :
"aec3e953-14b8-4bfc-a4eb-d47b6dc7925b",
    "typ" : "Bearer",
    "azp" : "my-app",
    "realm_access" : {
      "roles" : [ "offline_access" ]
    },
    "resource_access" : {
      "account" : {
        "roles" : [ "manage-account" ]
      },
      "my-app" : {
        "roles" : [ "normal_user" ]
      }
    },
}
```

```
{
  "aud": "api://my-api-client-id",
  "iss":
"https://login.microsoftonline.com/12345678-aaaa
-bbbb-cccc-1234567890ab/v2.0",
  "iat": 1713170000,
  "nbf": 1713170000,
  "exp": 1713173600,
  "azp": "my-client-id",
  "appid": "my-client-id",
  "appidacr": "1",
  "oid": "aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeeee",
  "sub": "aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeeee",
  "tid": "12345678-aaaa-bbbb-cccc-1234567890ab",
  "ver": "2.0",

  "scp": "user.read",

  "roles": [
    "Admin"
  ],

  "preferred_username": "user@company.com",
  "name": "John Doe"
}
```

DEMO

home.cern

Topics

- Scaling - the pain points
- Multi-IdP federation concepts
- **AuthN & AuthZ patterns**
- Gateway integration patterns

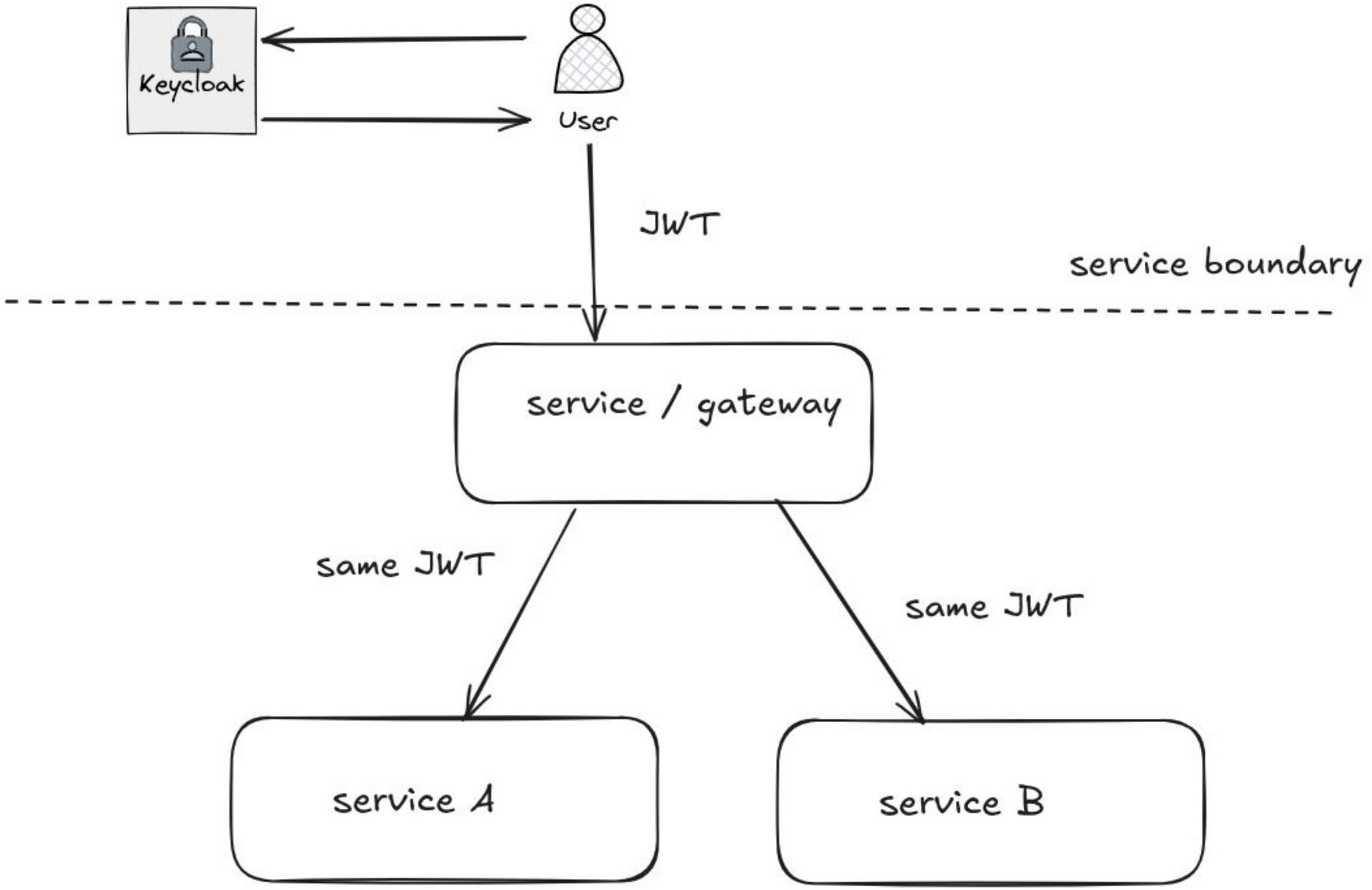
AuthN & AuthZ patterns

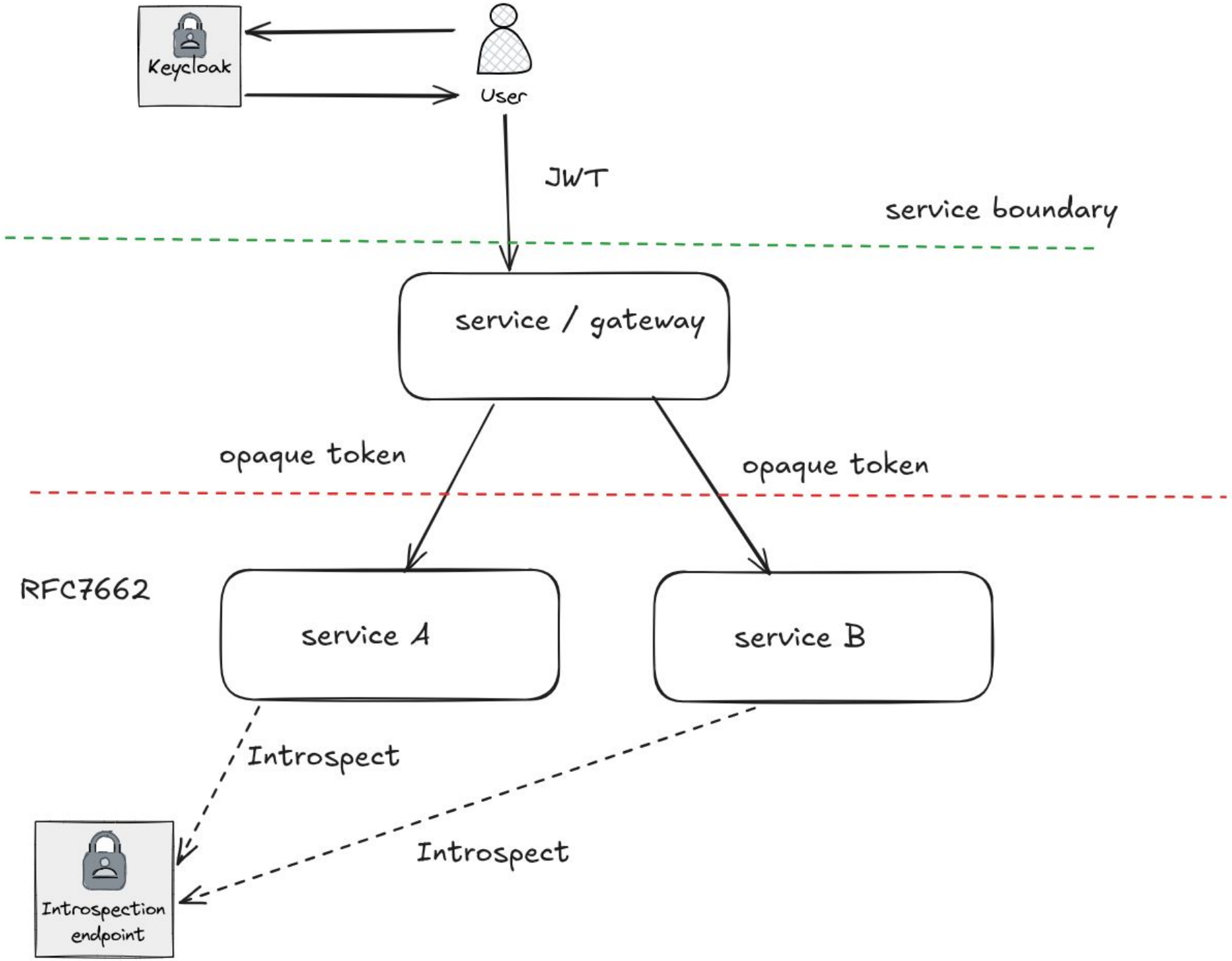
- AuthN - **who am I?** -> IdP concern
- AuthZ - **what can I do?** -> service / domain concern

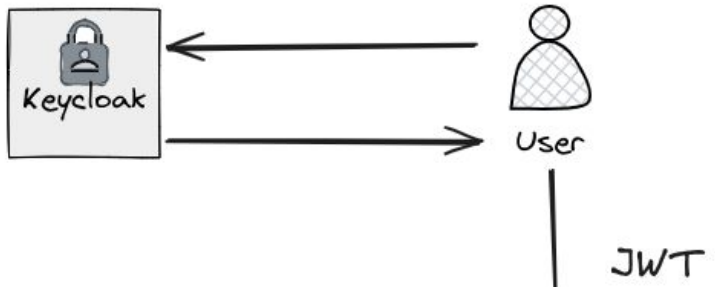
Patterns explored:

1. Validation & token distribution strategies
2. Centralized login (BFF / OAuth client)
3. Per-service / domain authorization

Edge validation vs downstream




Keycloak
User
JWT
service boundary
service / gateway
opaque token
opaque token
RFC7662
service A
service B
Introspect
Introspect
Introspection endpoint



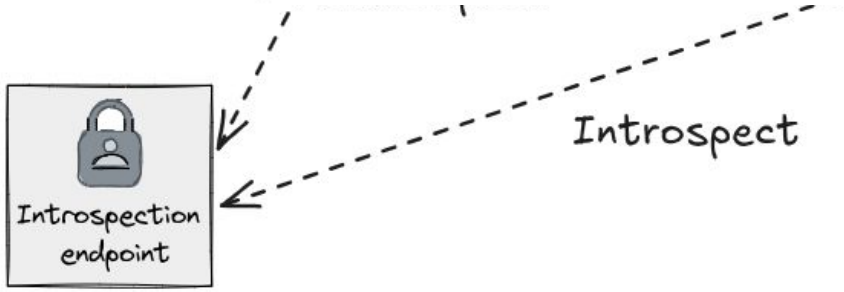
Internet Engineering Task Force (IETF)
Request for Comments: 7662
Category: Standards Track
ISSN: 2070-1721

J. Richer, Ed.
October 2015

OAuth 2.0 Token Introspection

Abstract

This specification defines a method for a protected resource to query an OAuth 2.0 authorization server to determine the active state of an OAuth 2.0 token and to determine meta-information about this token. OAuth 2.0 deployments can use this method to convey information about the authorization context of the token from the authorization server to the protected resource.



Pros / cons

JWT

Pros

- decentralized token validation
- fewer moving parts

Cons

- every service handles IdP quirks
- token bloat
- risk of leaks
- not easy to revoke

Opaque token

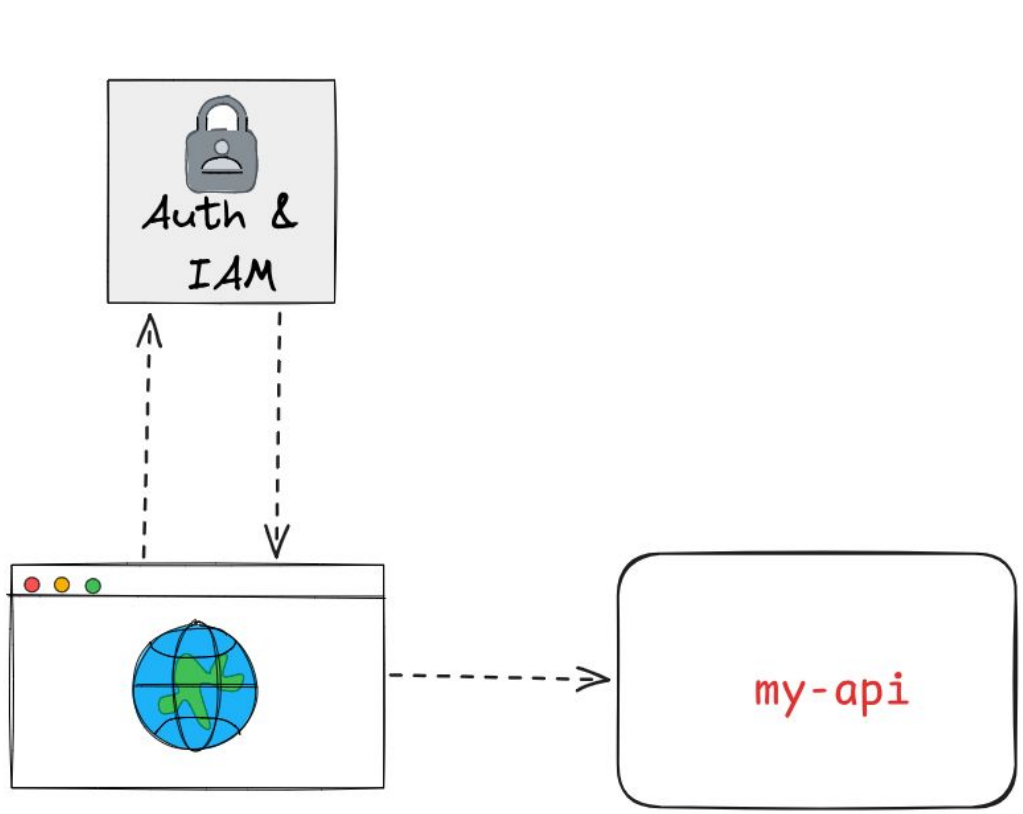
Pros

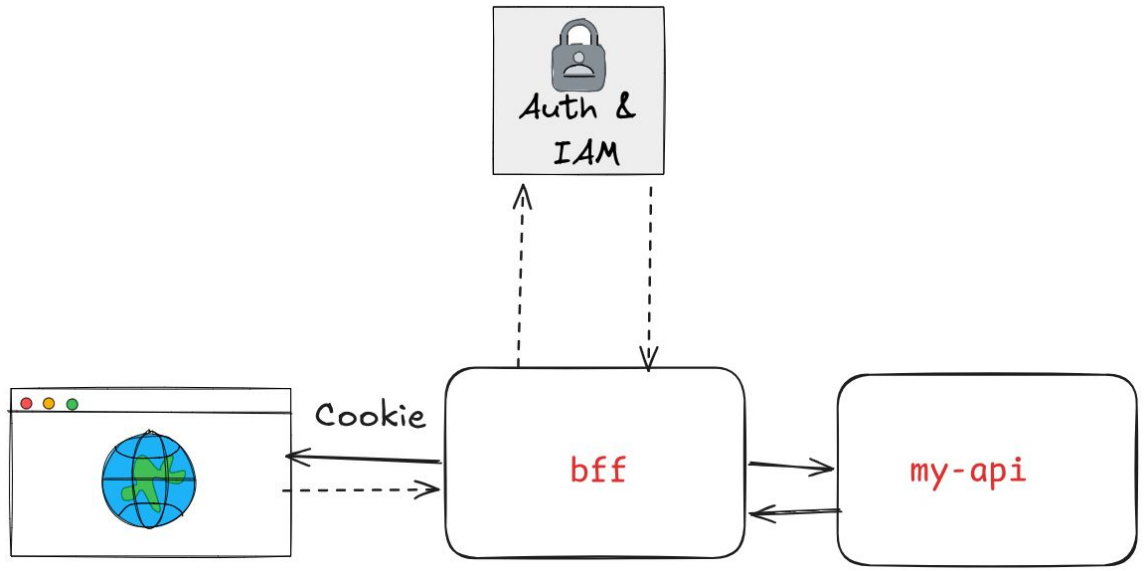
- easy to revoke
- cannot leak any data
- dynamic roles / claims
- uniform handling, regardless of # svc

Cons

- depends on an auth server
- more things that can break
- harder debugging

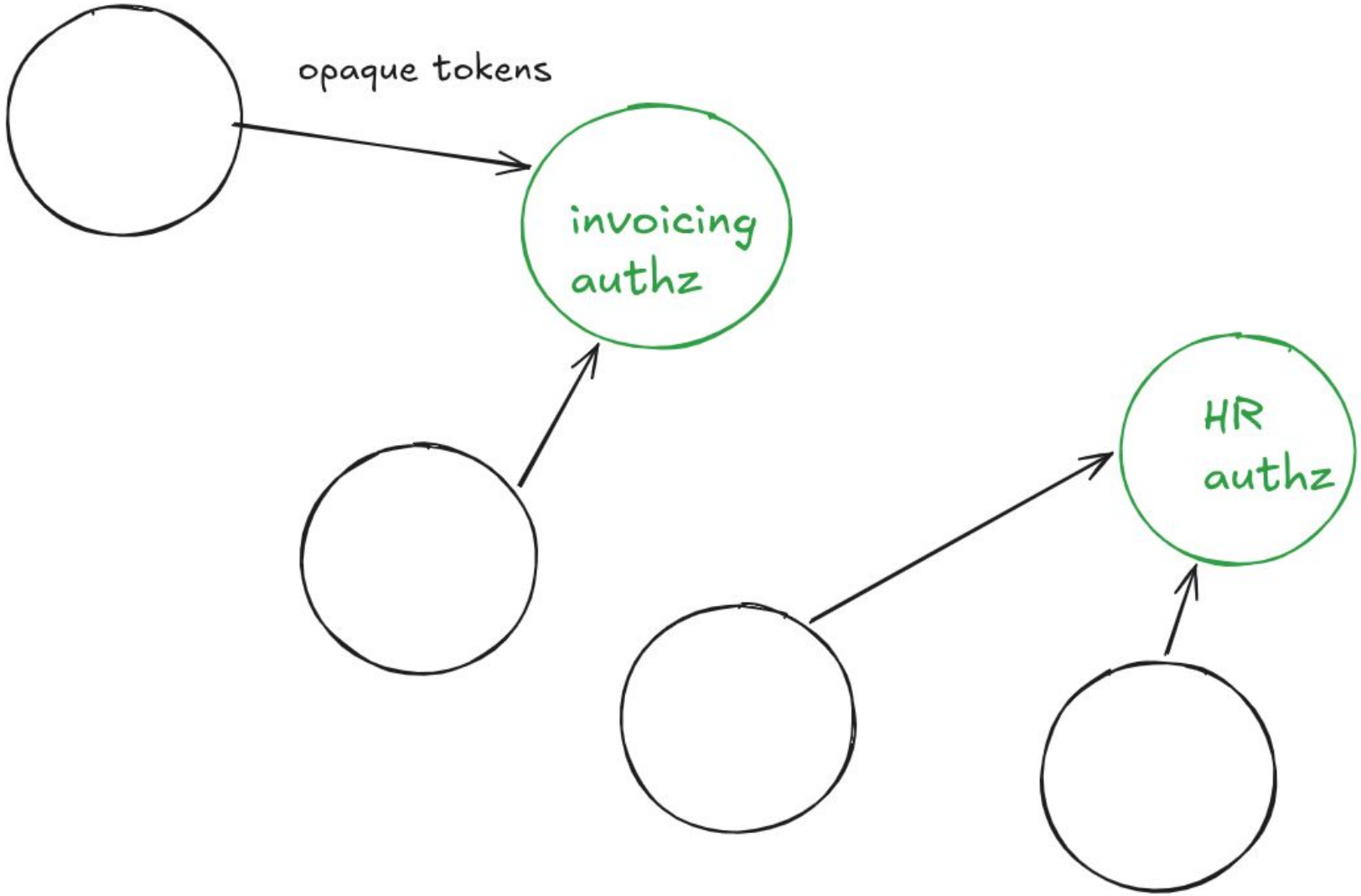
BFF (Backend-for-frontend)





Per-service authorization layer

Nothing prevents you from breaking up AuthZ services by domain



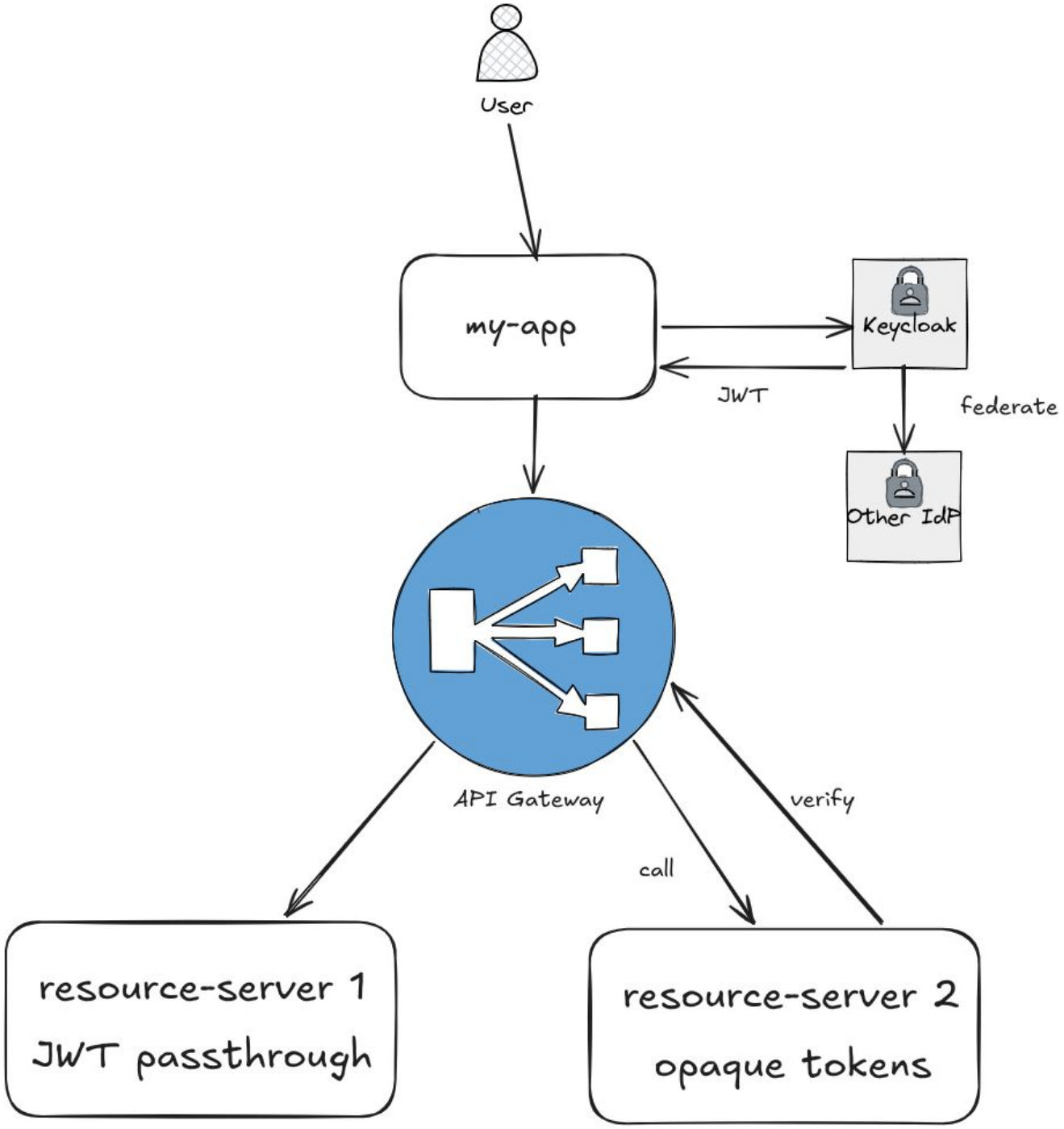
DEMO

home.cern

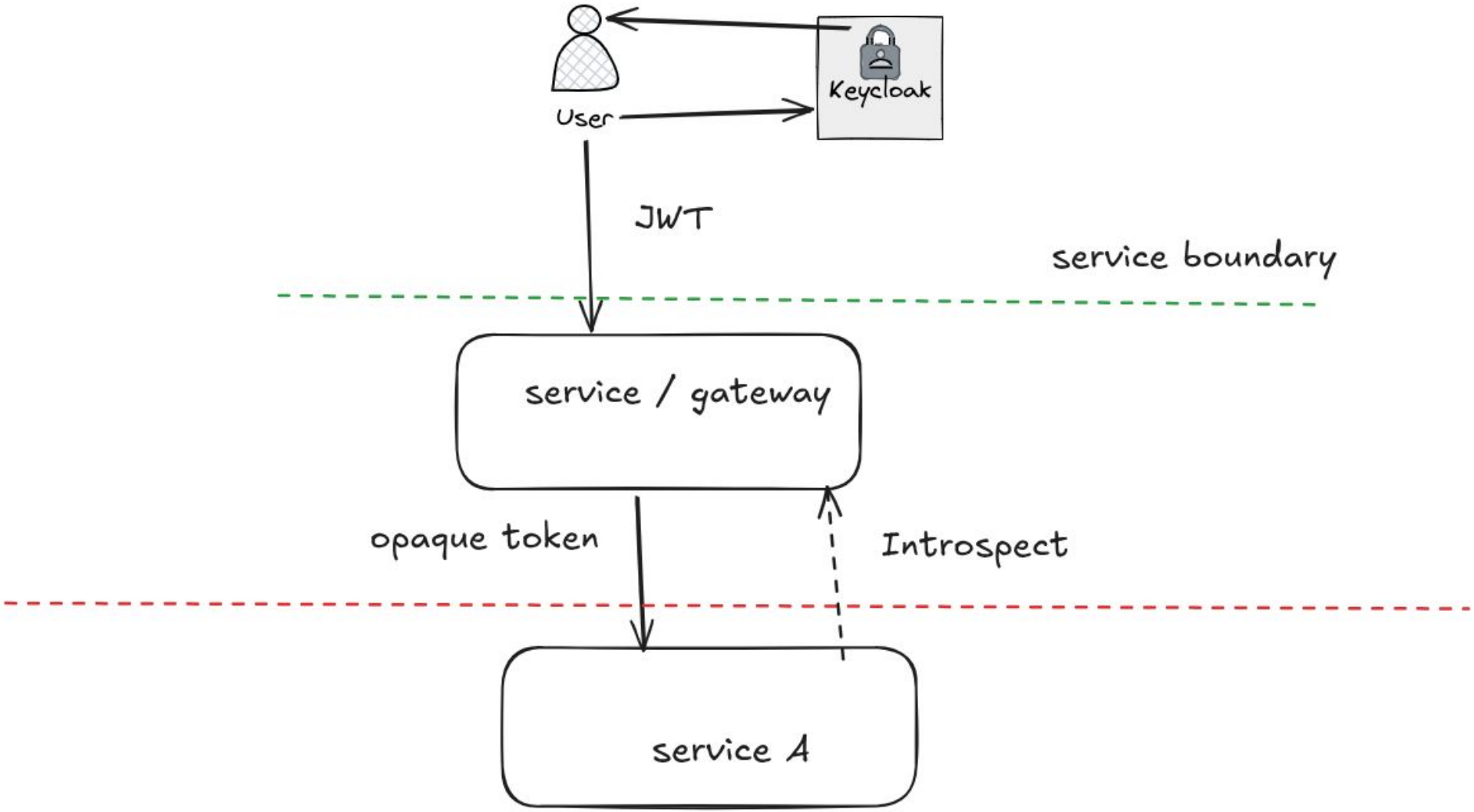
Topics

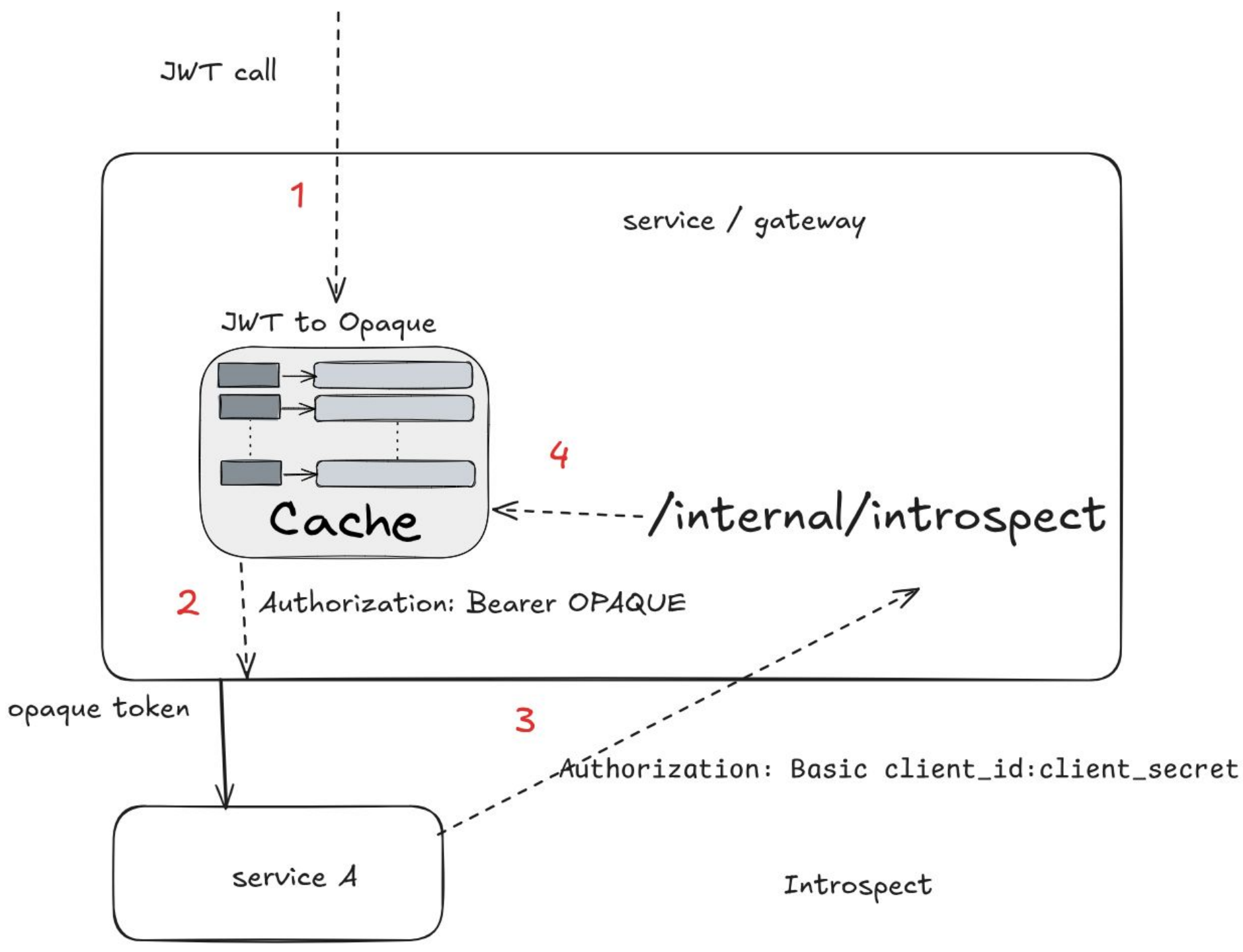
- Scaling - the pain points
- Multi-IdP federation concepts
- AuthN & AuthZ patterns
- **Gateway integration patterns**

Lab setup


User
my-app
Keycloak
JWT
federate
Other IdP
API Gateway
verify
call
resource-server 1
JWT passthrough
resource-server 2
opaque tokens

Gateway setup - opaque tokens




JWT call
1
service / gateway
JWT to Opaque
Cache
4
/internal/introspect
2
Authorization: Bearer OPAQUE
opaque token
3
Authorization: Basic client_id:client_secret
service A
Introspect

Key takeaways

Token choice

- JWT everywhere - could work
- Opaque tokens - great for sensitive environments
- Hybrid - exchange JWT at edge for Opaque tokens

Key takeaways

Pitfalls

- Make sure to set sensible TTLs
- Gateway can become single point of failure
- Take care of observability / auditing

Q&A?

@cschuszter.bsky.social

@cschuszter

https://cern.ch/arch-patterns-for-your-tech-lead

https://cern.ch/arch-patterns-slides

cristian.schuszter@cern.ch

home.cern