コンピュータセキュリティシンポジウム 2019
OSSセキュリティ技術ワークショップ 2019

オープンAPIに求められる認可認証OSSセキュリティ技術

中村　雄一

日立製作所　OSSソリューションセンタ

Contents

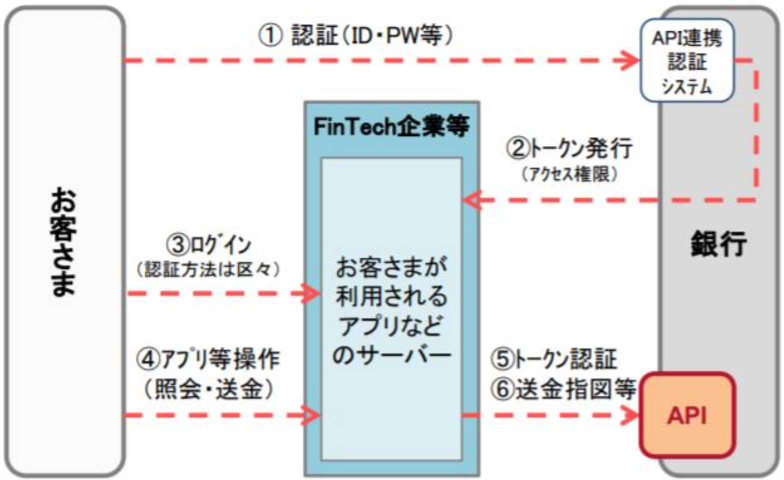
1. 背景と要件
2. 要件を満たすためのOSSの活用

日本における金融業界でのオープンAPI

- 2017年6月に銀行法改正を機に、銀行業界でAPI公開が加速
  - Fintech企業との連携を促進 -> 最近では自社アプリとの連携にもAPI。
- 114行のうち83%が2020/6までにAPI公開と回答(*)

(*) 2017年12月全銀協調査による

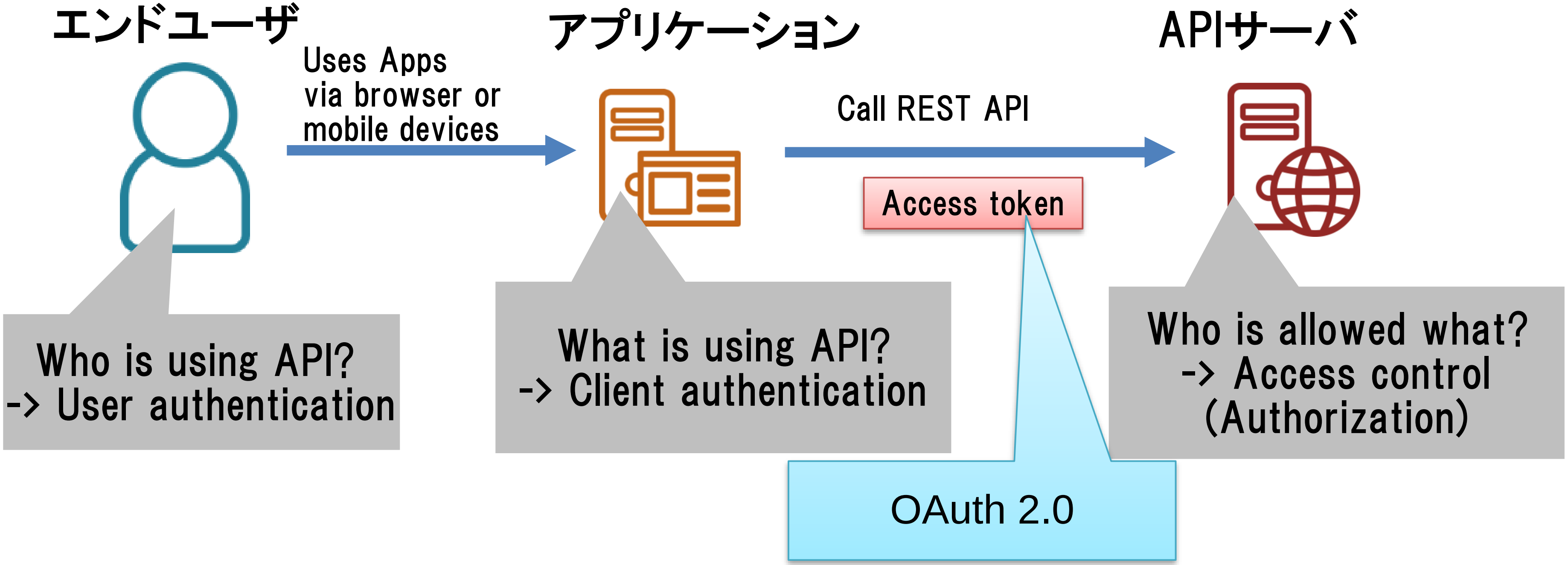
- 認可に関わるプロトコルOAuth 2.0が基本とみなされている
  - 従来は認証情報をFintech企業に渡す必要があったが、OAuth2.0により渡す必要がなくなった



【図表１】オープン API の基本的な仕組み（OAuth2.0）

銀協：「オープン API のあり方に関する検討会報告書」より引用
tps://www.zenginkyo.or.jp/fileadmin/res/news/news290713_1.pdf

OAuth 2.0による認可・認証



エンドユーザ

Uses Apps
via browser or
mobile devices

アプリケーション

Call REST API

Access token

APIサーバ

Who is using API?
-> User authentication

What is using API?
-> Client authentication

Who is allowed what?
-> Access control
(Authorization)

OAuth 2.0

＊ OAuth 2.0 (RFC6749)はトークンをどう発行するかしか記載がない
それ以外については、他の標準や自作が必要

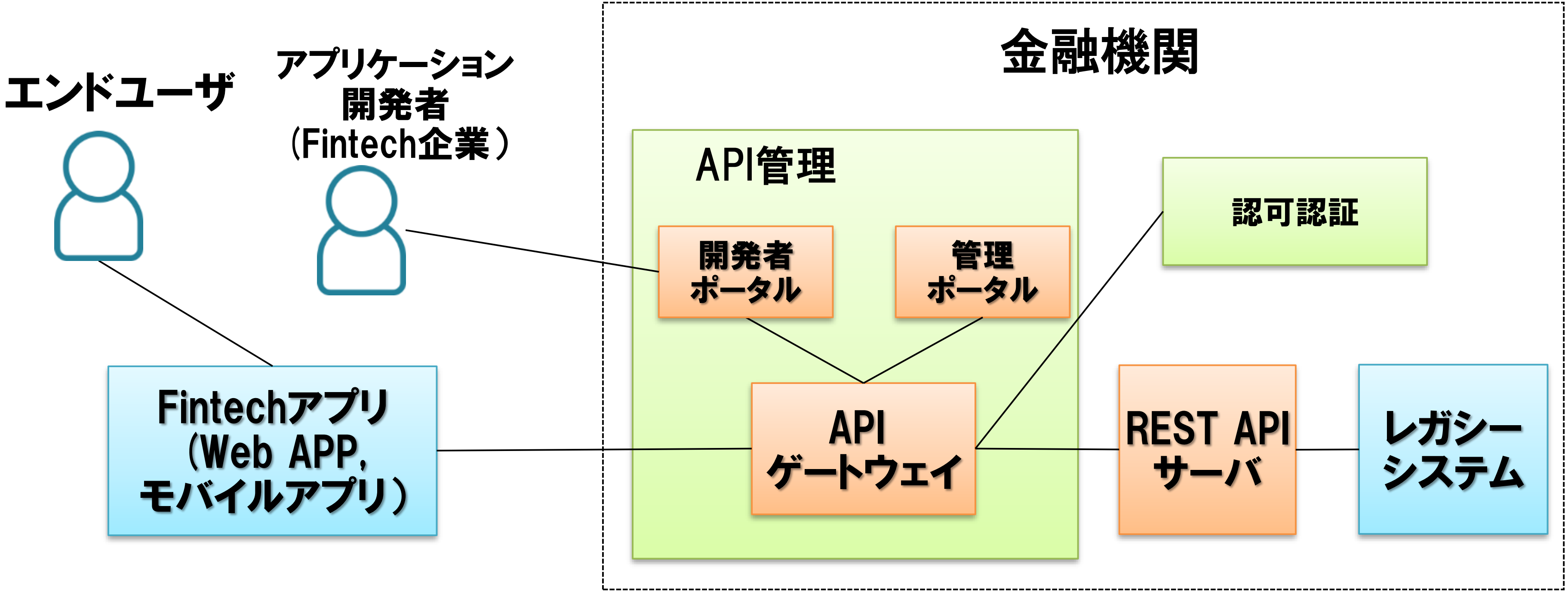
高セキュリティなオープンAPIを実現するための要件

#	分類	概要
1	認証	• OAuthのフローの中での様々な認証への対応 • OAuthの上のOpenID Connectへの対応
2	アクセス制御	• トークン中の属性に応じたアクセス制御 • バックエンド保護のため流量制御との組み合わせ
3	トークンの管理	• エンドユーザおよび管理者からのトークン失効インタフェース • ポリシに基づいたトークン失効
4	最新の標準への対応	• Financial-grade API (FAPI)仕様への対応 • WebAuthnへの対応

Contents

1. 背景と要件
2. <u>要件を満たすためのOSSの活用</u>

オープンAPIシステム（金融業界の例）



- 共通機能を担うAPI Management製品が一般に使われる
  - 流量制御、開発者ポータル、分析機能等
- 認可認証をAPI管理と統合必要

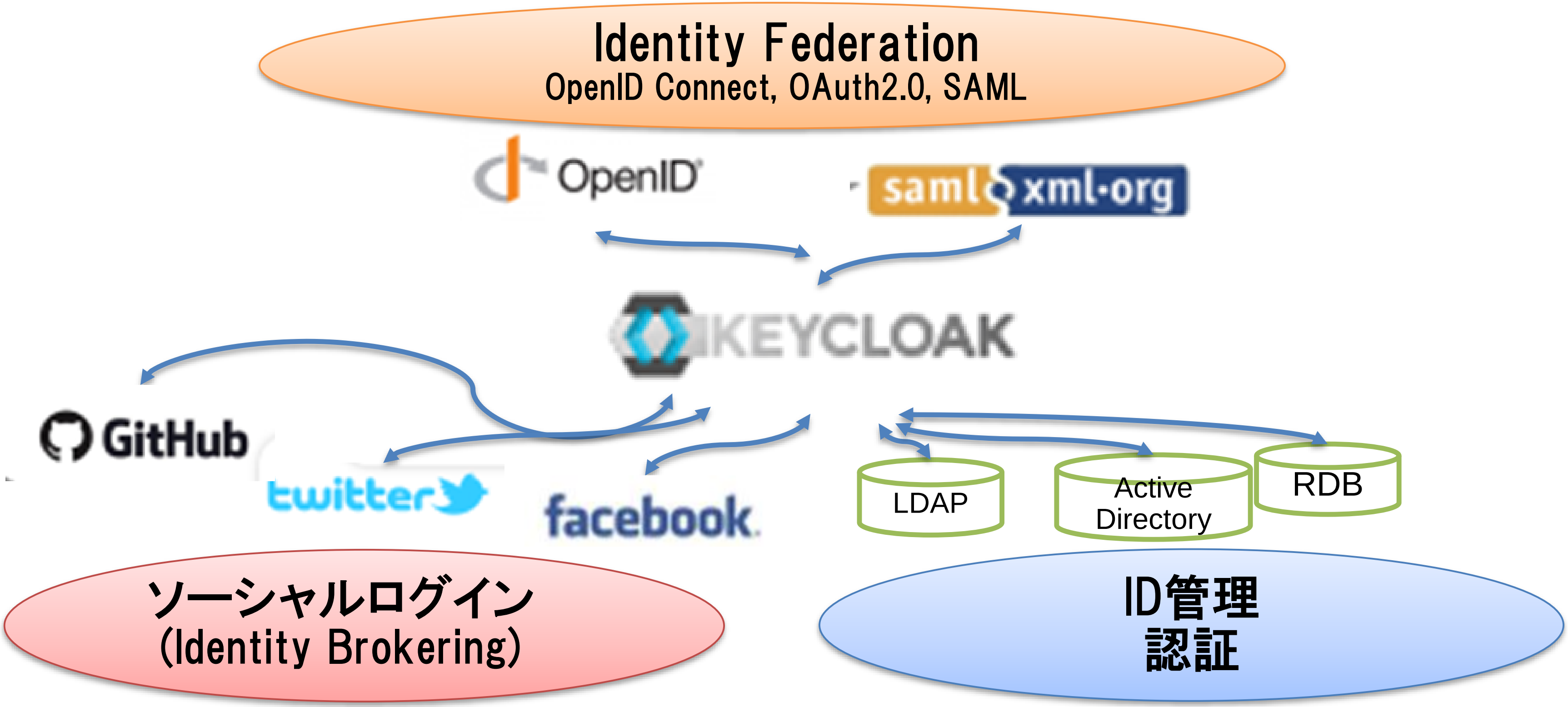
関連するOSS

	OSS
API管理	Kong
	3scale
	WSO2
	tyk
認可認証	Keycloak
	Gluu
	OpenAM

- 今回は機能のカバー範囲・コミュニティ活性度から“3scale” と “Keycloak”に着目

Keycloakとは

ID管理に関連するRed Hat社を中心に開発されているOSS: www.keycloak.org

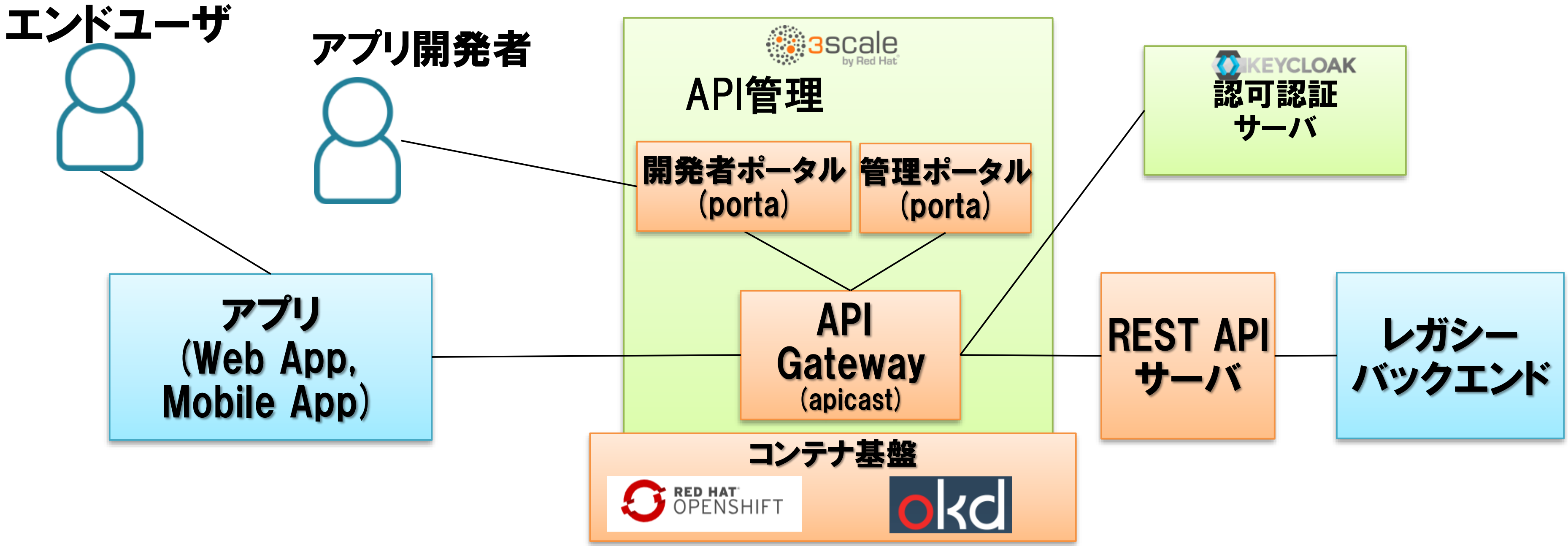


Identity Federation
OpenID Connect, OAuth2.0, SAML

ソーシャルログイン
(Identity Brokering)

ID管理
認証

3scaleとは



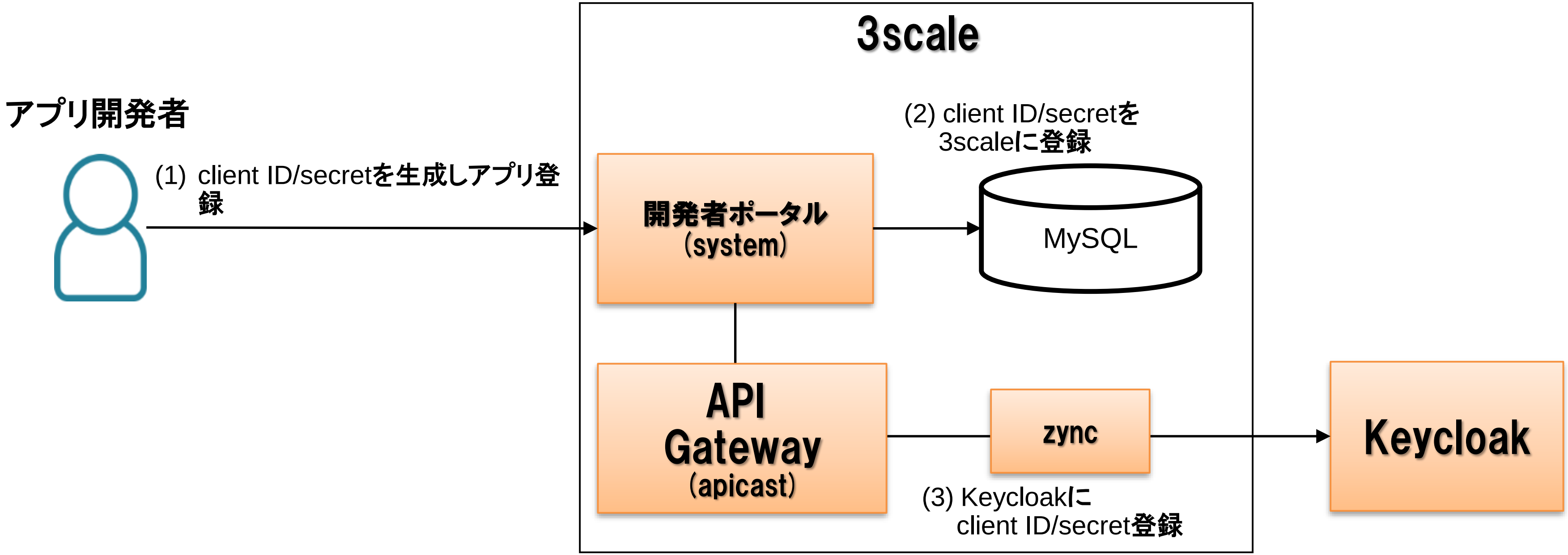
- OSSのAPI管理の実装、Red Hat社メインで開発 (https://github.com/3scale)
- API管理のフル機能を保有
- **クラウドネィティブ：コンテナ基盤のOpenShift(okdプロジェクトで開発)の上で動作**
- Keycloakと連携してOAuth2/OIDCに対応した認可認証

高セキュリティなオープンAPIを実現するための要件

#	分類	概要
1	認証	• OAuthのフローの中での様々な認証への対応 • OAuthの上のOpenID Connectへの対応
2	アクセス制御	• トークン中の属性に応じたアクセス制御 • バックエンド保護のため流量制御との組み合わせ
3	トークンの管理	• エンドユーザおよび管理者からのトークン失効インタフェース • ポリシに基づいたトークン失効
4	最新の標準への対応	• Financial-grade API (FAPI)仕様への対応 • WebAuthnへの対応

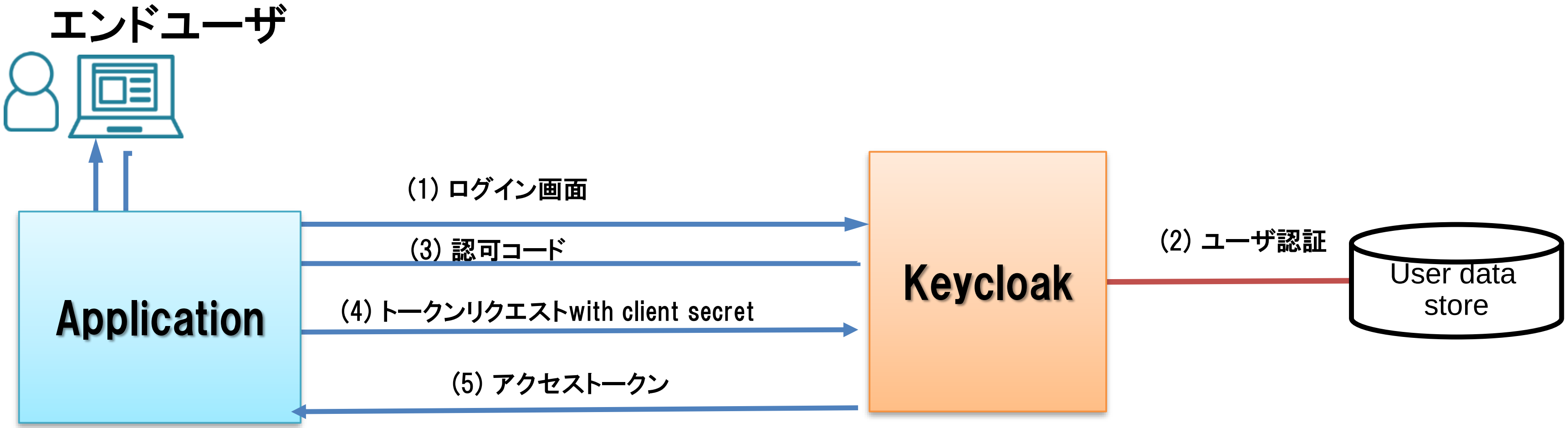
3scale+Keycloakを活用し、OSSコミュニティとも連携して実現

認証：アプリの登録


3scale
アプリ開発者
(1) client ID/secretを生成しアプリ登録
開発者ポータル (system)
(2) client ID/secretを3scaleに登録
MySQL
API Gateway (apicast)
zync
(3) Keycloakに client ID/secret登録
Keycloak

認証：アクセストークン発行

OAuth 2.0のAuthorization code flowでアクセストークン発行



エンドユーザ

Application

Keycloak

User data store

(1) ログイン画面

(2) ユーザ認証

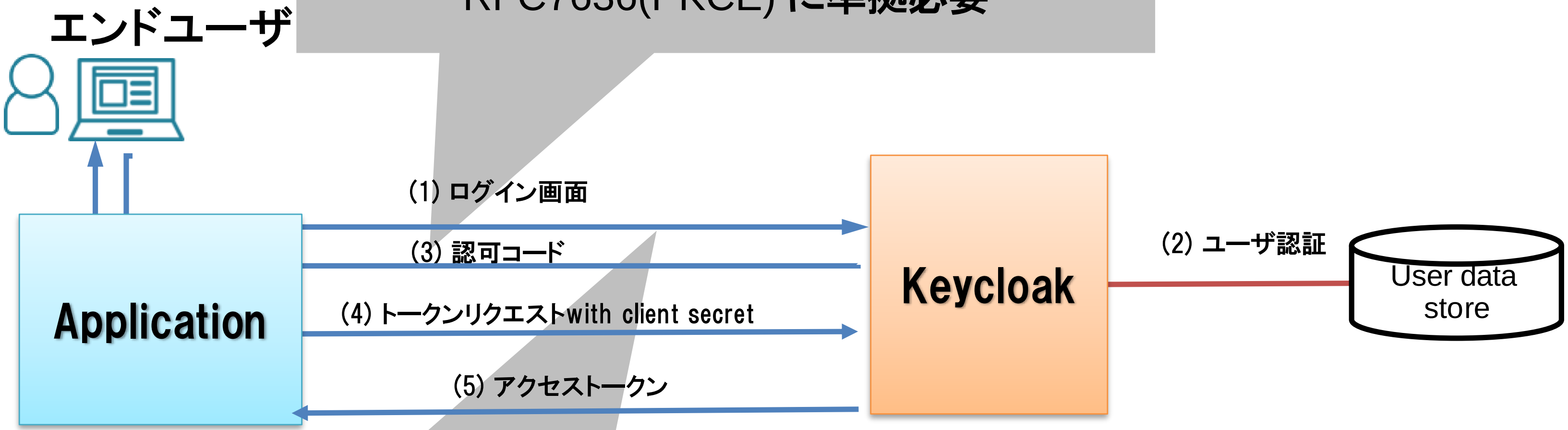
(3) 認可コード

(4) トークンリクエストwith client secret

(5) アクセストークン

認証: 課題

認可コード保護のため、
RFC7636(PKCE) に準拠必要



Keycloakによってログイン画面が生成。
画面はテンプレートでカスタマイズできるが、
高度にカスタマイズしたい場合は不十分

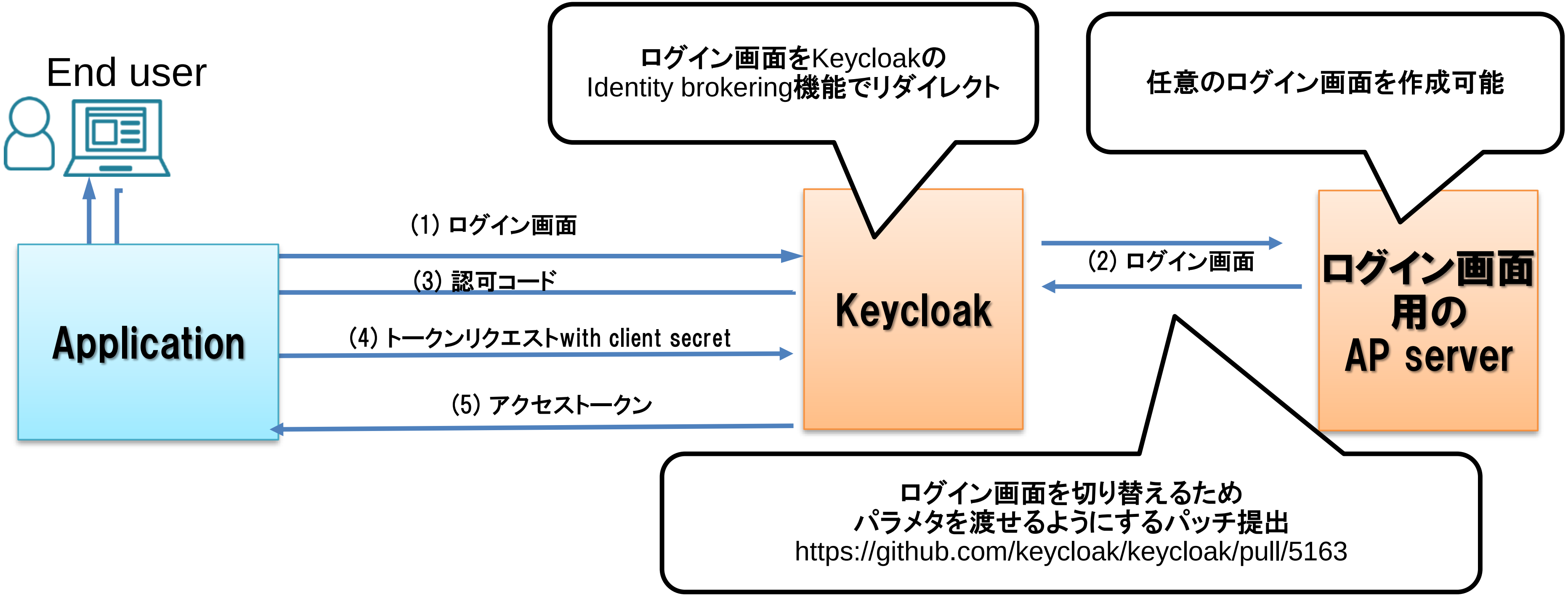
KeycloakでのPKCEサポート

- KeycloakはPKCEをサポートしていなかった
  -> パッチを提出することで対応

https://github.com/keycloak/keycloak/pull/3831

- Keycloak 3.1.0からPKCEサポート
  - その後、いくつか関連パッチも提出され、様々なケースで使えるようになっている

高度にカスタマイズされたログイン画面の実現



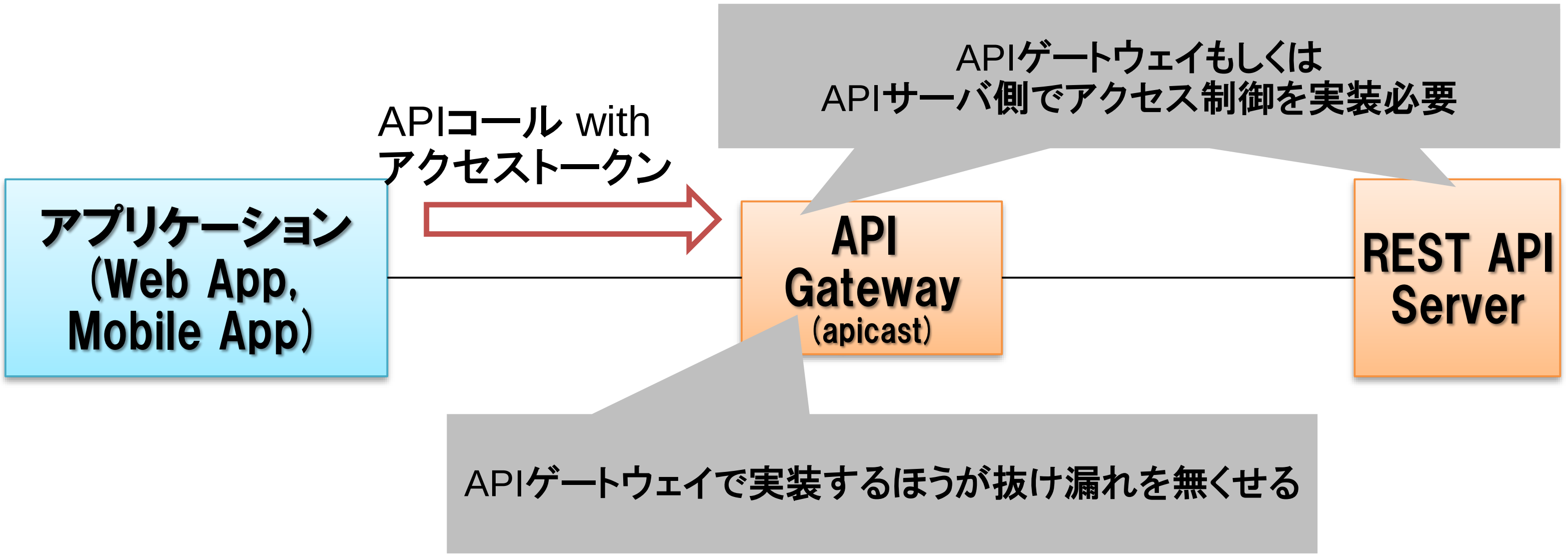
ログイン画面をKeycloakの
Identity brokering機能でリダイレクト

任意のログイン画面を作成可能

ログイン画面を切り替えるため
パラメタを渡せるようにするパッチ提出
https://github.com/keycloak/keycloak/pull/5163

アクセス制御

- Keycloakから発行されたトークンをどう使ってアクセス制御するかは、Keycloakの範囲外



3scaleのAPIゲートウェイ(apicast)ではアクセストークンでのアクセス制御を実装していなかった
→ OSSコミュニティと共に開発！

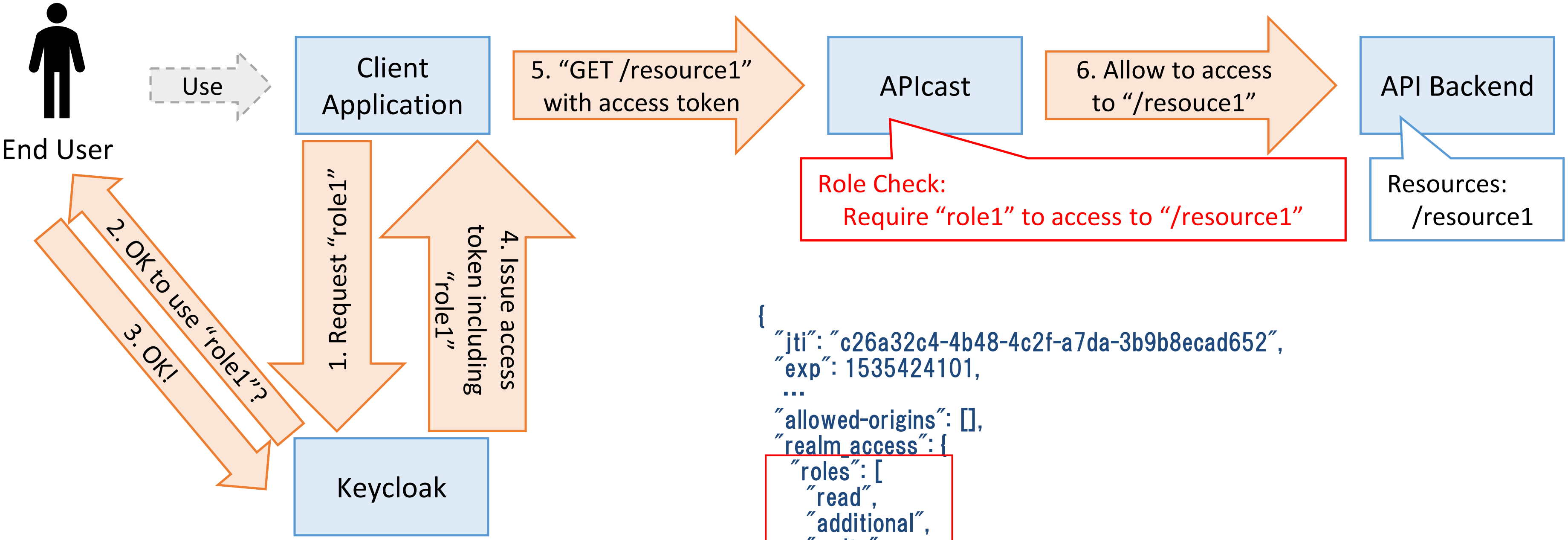
Keycloakのアクセストークン

- アクセストークンのフォーマットは任意で、認可サーバの実装次第。
- Keycloakでは、JWT(JSON Web Token)という形式。JSON+署名

```
{
  ″jti″: ″c26a32c4-4b48-4c2f-a7da-3b9b8ecad652″,
  ″exp″: 1535424101,
  ″nbf″: 0,
  ″iat″: 1535423801,
  ″iss″: ″http://localhost:8080/auth/realms/provider″,
  ″aud″: ″broker″,
  ″sub″: ″e4b11e2e-9136-409b-8720-57463c627c10″,
  ″typ″: ″Bearer″,
  ″azp″: ″broker″,
  ″auth_time″: 0,
  ″session_state″: ″ac1767e2-2e30-4d44-b6f3-b77935a7a0bc″,
  ″acr″: ″1″,
  ″allowed-origins″: [],
  ″realm_access″: {
    ″roles″: [
      ″read″,
      ″additional″,
      ″write″
    ]
  },
″name″: ″Takashi Mogi″,
  ″preferred_username″: ″mogi″,
  ″given_name″: ″Takashi″,
  ″family_name″: ″Mogi″,
  ″email″: ″mogi@example.com″
}
```

APIゲートウェイでのアクセス制御実装

- Apicastで実装した拡張
- アクセストークン中の“role”フィールドとURLの対応付けをチェック
- パッチ提出し、3scale 2.3に採用. https://github.com/3scale/apicast/pull/773



End User

Use

Client Application

5. “GET /resource1” with access token

APIcast

6. Allow to access to “/resouce1”

API Backend

Role Check:
Require “role1” to access to “/resource1”

Resources:
/resource1

1. Request “role1”

4. Issue access token including “role1”

2. OK to use “role1”?

3. OK!

Keycloak

```
{
  "jti": "c26a32c4-4b48-4c2f-a7da-3b9b8ecad652",
  "exp": 1535424101,
  …
  "allowed-origins": [],
  "realm_access": {
    "roles": [
      "read",
      "additional",
      "write"
    ]
  },
```

トークンの管理

Keycloakそのものにはアクセストークンの失効機能はあり

- 管理者を契機としたトークン失効
  -> 管理コンソールから失効できる
- ポリシーベースの失効
  -> 管理コンソールでタイムアウトなど設定可能

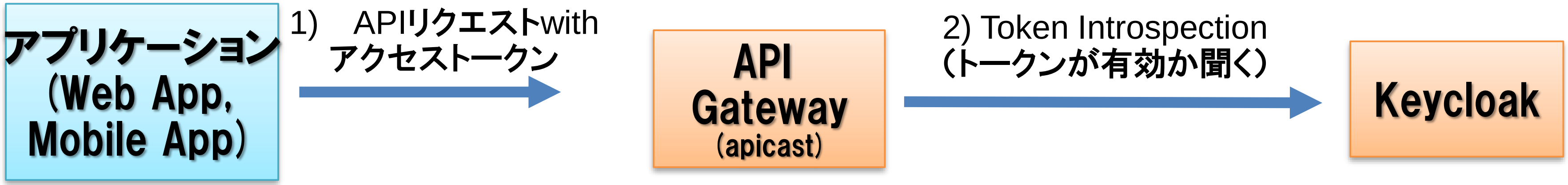
- エンドユーザの意思によるトークン失効
  - ログアウトエンドポイントで失効
    /auth/realms/＜realm＞/protocol/openid-connect/logout

→ トークンが有効か無効かの最新情報は認可サーバしか知らない。
そのため、APIがコールされたときに認可サーバに聞く必要がある。

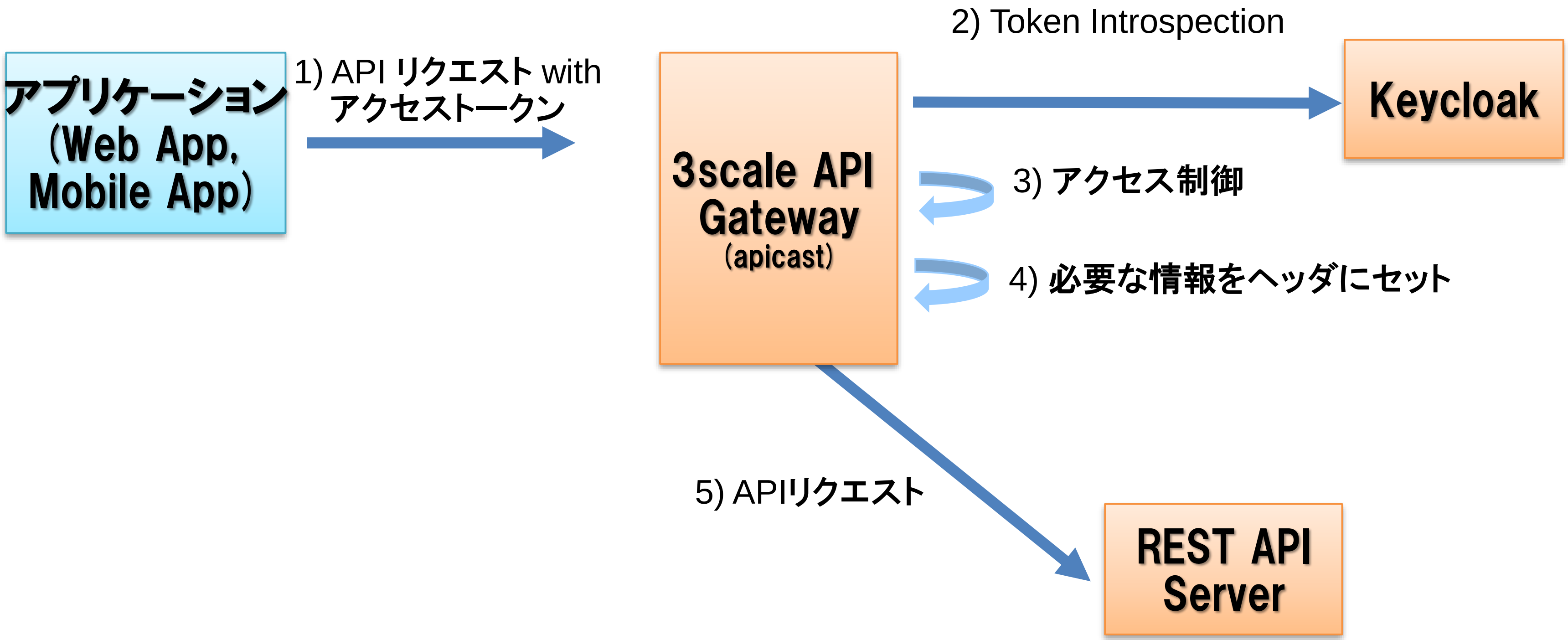
聞くためのインタフェースが「Token IntrospectionエンドポイントRFC7662) 」

APIゲートウェイでのToken Introspection

- APIサーバ側で毎度Token Introspectionをコールするのは煩雑
- APIゲートウェイでToken Introspectionをコールできるようにapicastを拡張、パッチ提出。3scale 2.3にて採用。



3scale + KeycloakでのAPIコールフロー



アプリケーション
(Web App,
Mobile App)

1) API リクエスト with
アクセストークン

3scale API
Gateway
(apicast)

2) Token Introspection

Keycloak

3) アクセス制御

4) 必要な情報をヘッダにセット

5) APIリクエスト

REST API
Server

最新の標準への対応

- 最新のOAuth標準： Financial Grade API(FAPI)への対応
  Keycloakで主要項目は対応

- 最新の認証標準： WebAuthn
  webauthn4jと連携し、keycloak側で対応を進めています

まとめ

- OAuthはオープンAPIシステムにおいてのキー技術

- OAuth周辺での要件
  - 認証、アクセス制御、トークン管理、最新の標準

- OSS (3scale + Keycloak)にてオープンAPIシステムを実現
  - 足りない部分はOSSコミュニティと共に改善
    - 3scale: アクセス制御、token introspection
    - Keycloak: FAPIやWebAuthn

- 実際の商用システムでも使われています

他社所有商標に関する表示

- Red Hat is a registered trademark of Red Hat, Inc. in the United States and other countries.
- OpenShift is a registered trademark of Red Hat, Inc. in the United States and other countries.
- WSO2は米国およびその他の国におけるWSO2の登録商標です。
- OpenIDは米国およびその他の国におけるOpenID Foundationの登録商標です。
- Githubは米国およびその他の国におけるGithub, Inc. の登録商標です。
- Twitterは米国およびその他の国におけるTwitter, Inc. の登録商標です。
- Facebookは米国およびその他の国におけるFacebook, Inc. の登録商標です。
- その他記載の会社名、製品名などは、それぞれの会社の商標もしくは登録商標です。

HITACHI
Inspire the Next