AUTHLETE

Authleteで体験する認可サーバー & リソースサーバーの構築（AWS編）

株式会社 Authlete

Welcome!

- AWS 上に認可サーバーと
  リソースサーバーを構築し
  動作をご紹介する勉強会です

- サーバーとクライアントとの間でやりとりされる OAuth のメッセージを確認しながら、サーバーが何をするのかを理解していきます

- 弊社サービス「Authlete（オースリート）」を用いることにより、短時間で効率的な認可サーバーとリソースサーバーの構築を目指します

タイムテーブル

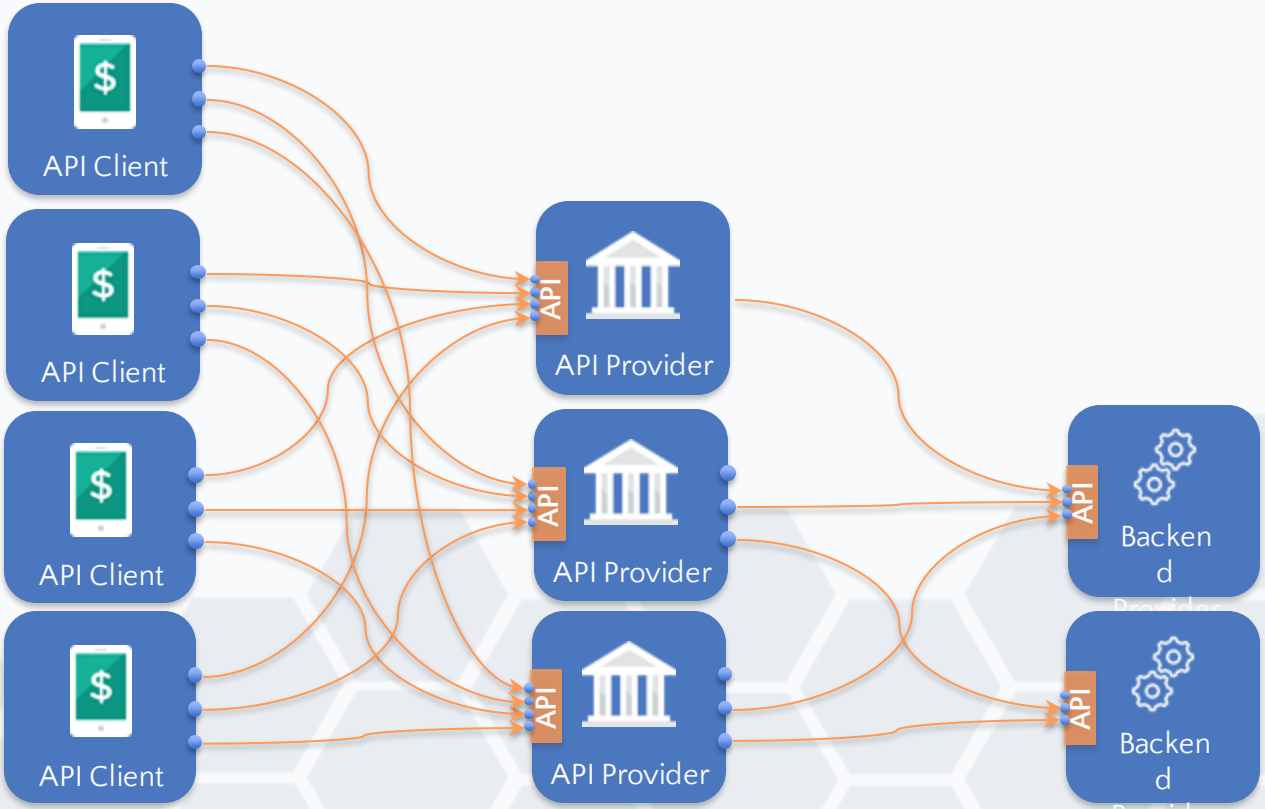
時間	内容
16:00 - 16:30	Authlete: API 認可エンジン のご紹介
16:30 - 17:30	認可サーバー&リソースサーバーの構築・動作デモ
17:30 - 18:00	Q & A

AUTHLETE

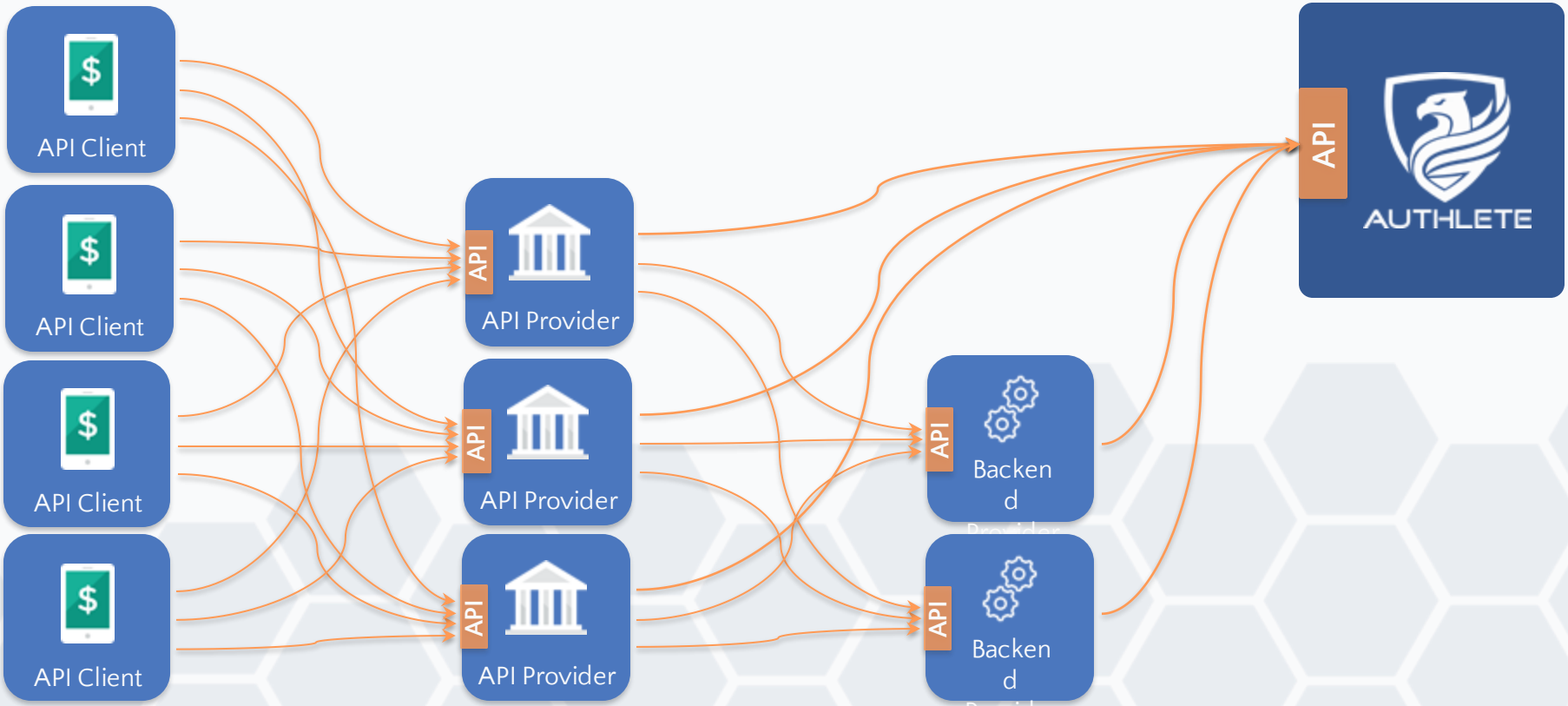
Authlete: API 認可エンジンのご紹介

株式会社 Authlete

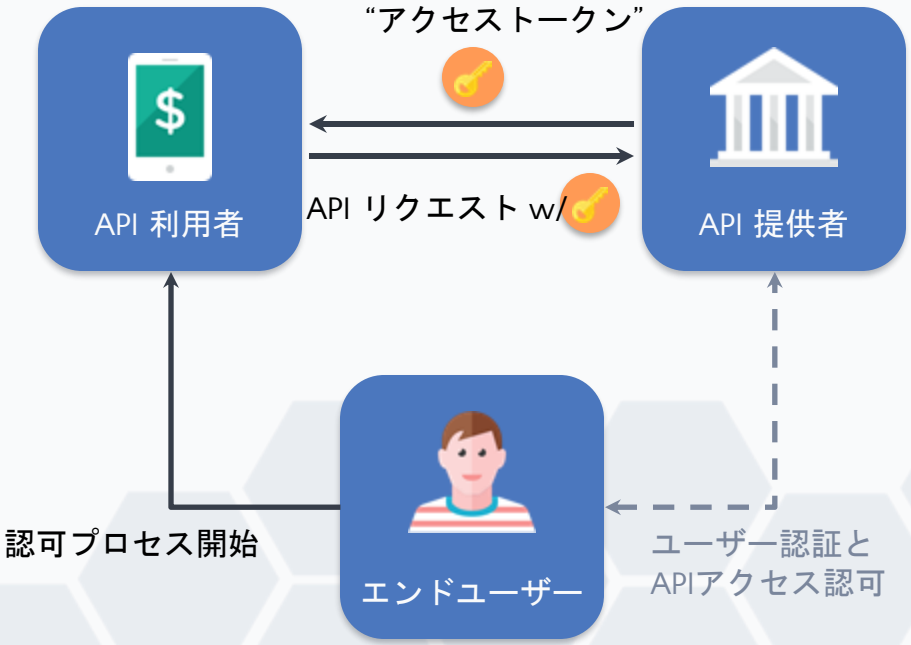
API エコシステムの多層化・分業化



Authlete は「API 認可」に特化したサービス



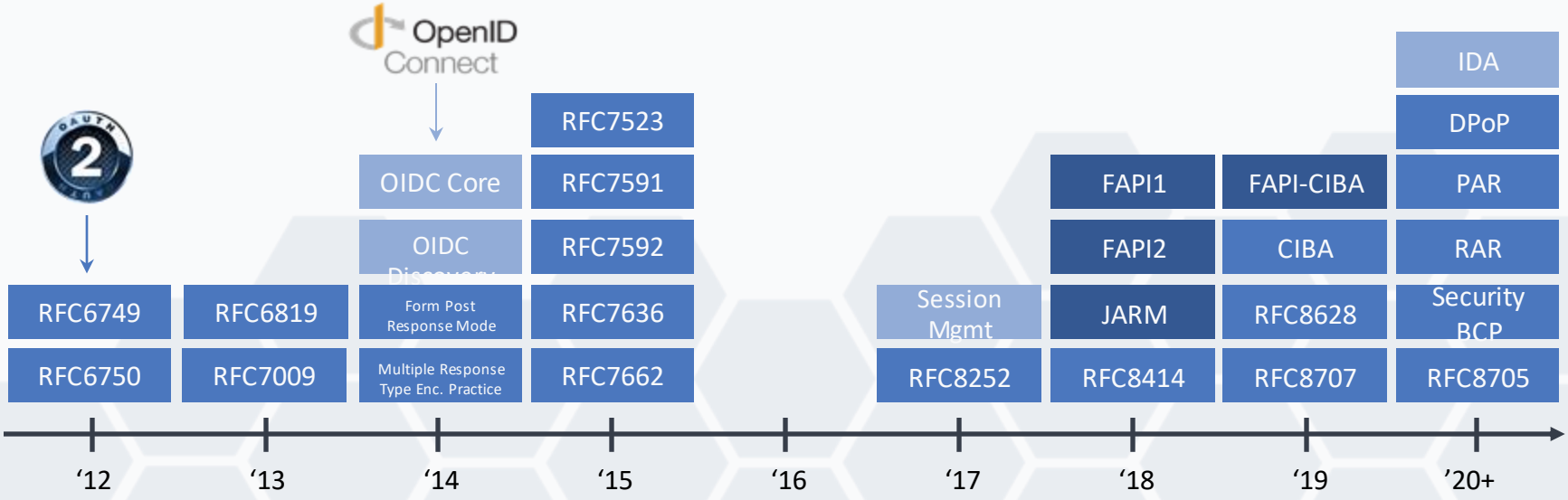
API 認可はオープン API のセキュリティの要



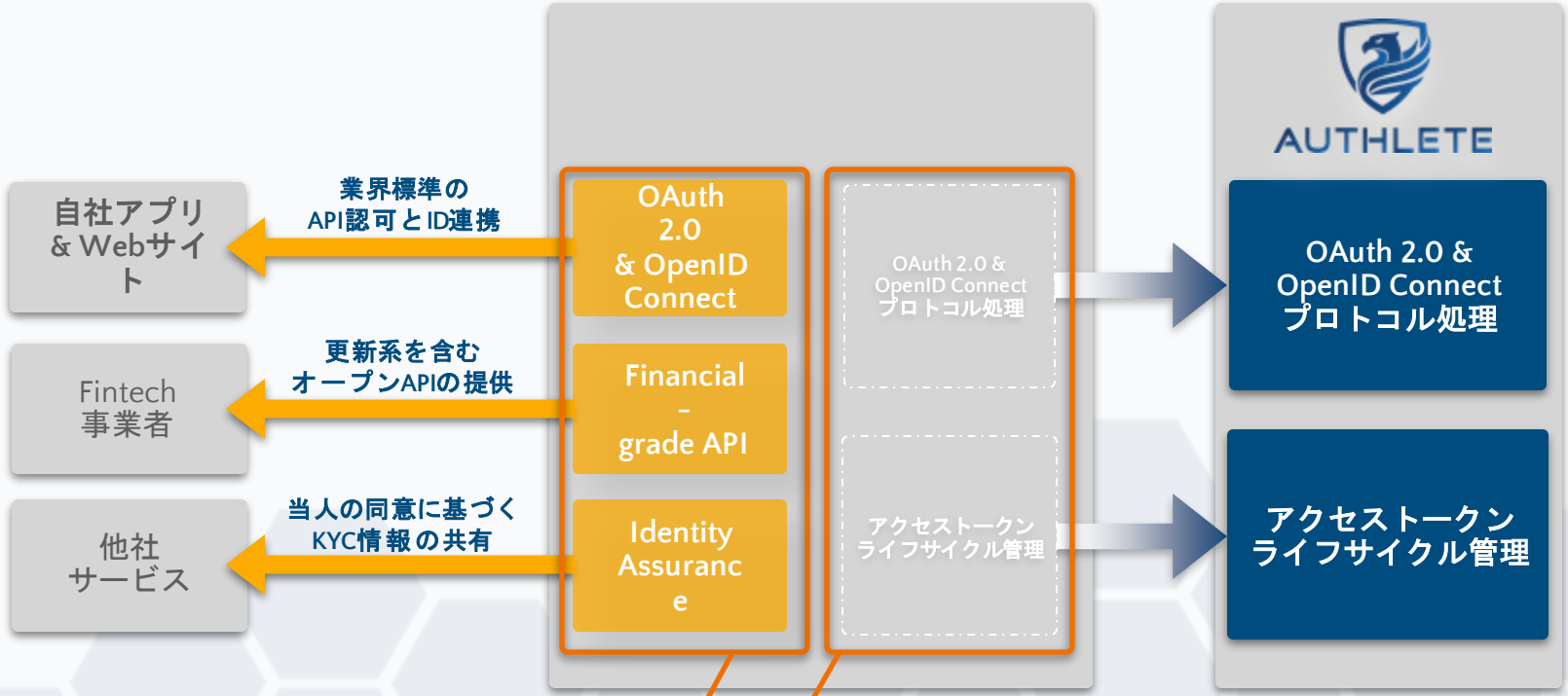
- API アクセスを
  エンドユーザーが
  認可
- “OAuth 2.0”
  - アクセストークン
    による API 認可

OAuth/OIDC を API 基盤に組み込むには?

- 最新仕様・プラクティスを正しく理解し実装する必要がある



OAuth/OIDC 実装を Authlete に外部化

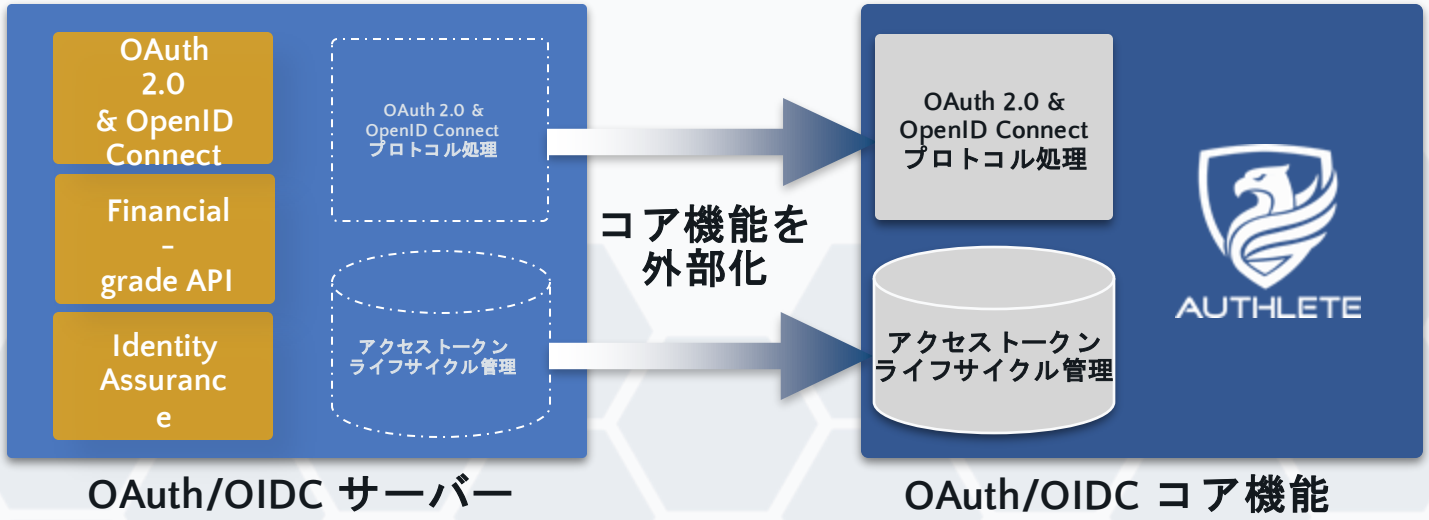


最先端の業界標準仕様に準拠したセキュアなAPI提供を実現

OAuth 2.0 & OpenID Connect に関する複雑な実装・運用を外部化
特定ソリューションに依存せず自由にUX設計が可能

Authlete が OAuth/OIDC サーバーのコアを代替

- 複雑な OAuth/OIDC 機能の実装工数が不要に

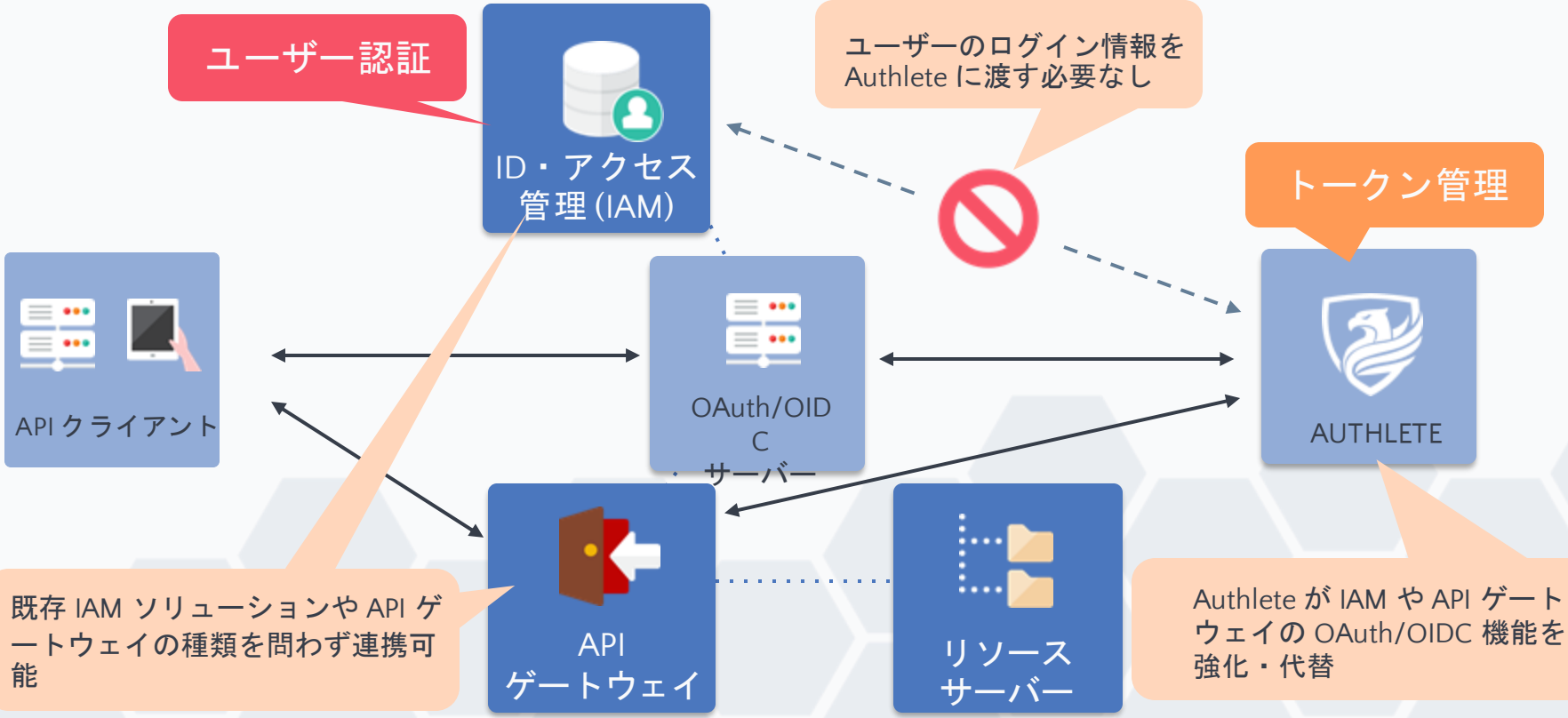


開発者ファーストのソリューション

- OAuth/OIDC のコア機能を API として疎結合化
- ランタイムや SDK へのロックインを排除

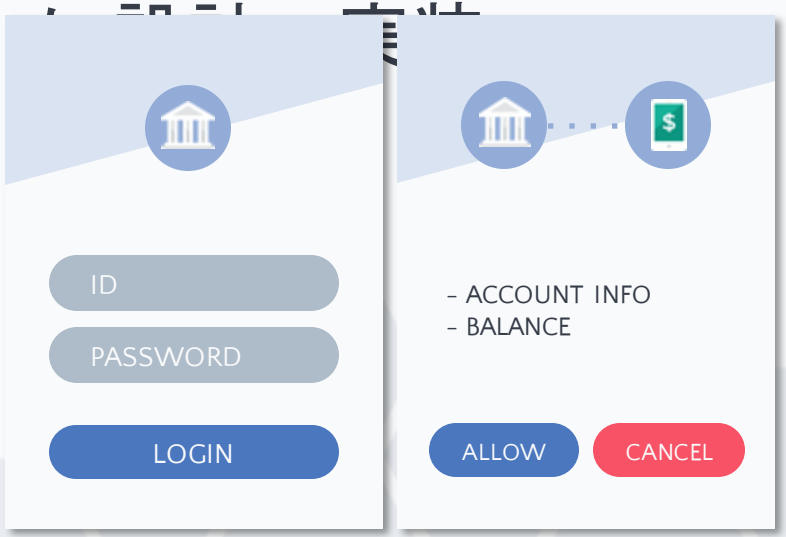
- OAuth 2.0 / OIDC プロトコルをスクラッチから実装する必要なし
- 実装のためのプログラミング言語・フレームワークを自由に選択可能
- OAuth/OIDC サーバーを数週間から数ヶ月で構築

既存のサービス基盤を補完・強化

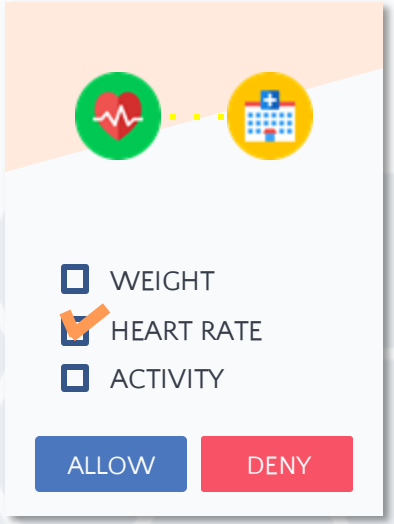


ユーザー同意の UI/UX を自在に設計

- Authlete をバックエンドに用いて、OAuth/OIDC サーバーのフロントエンドを自由

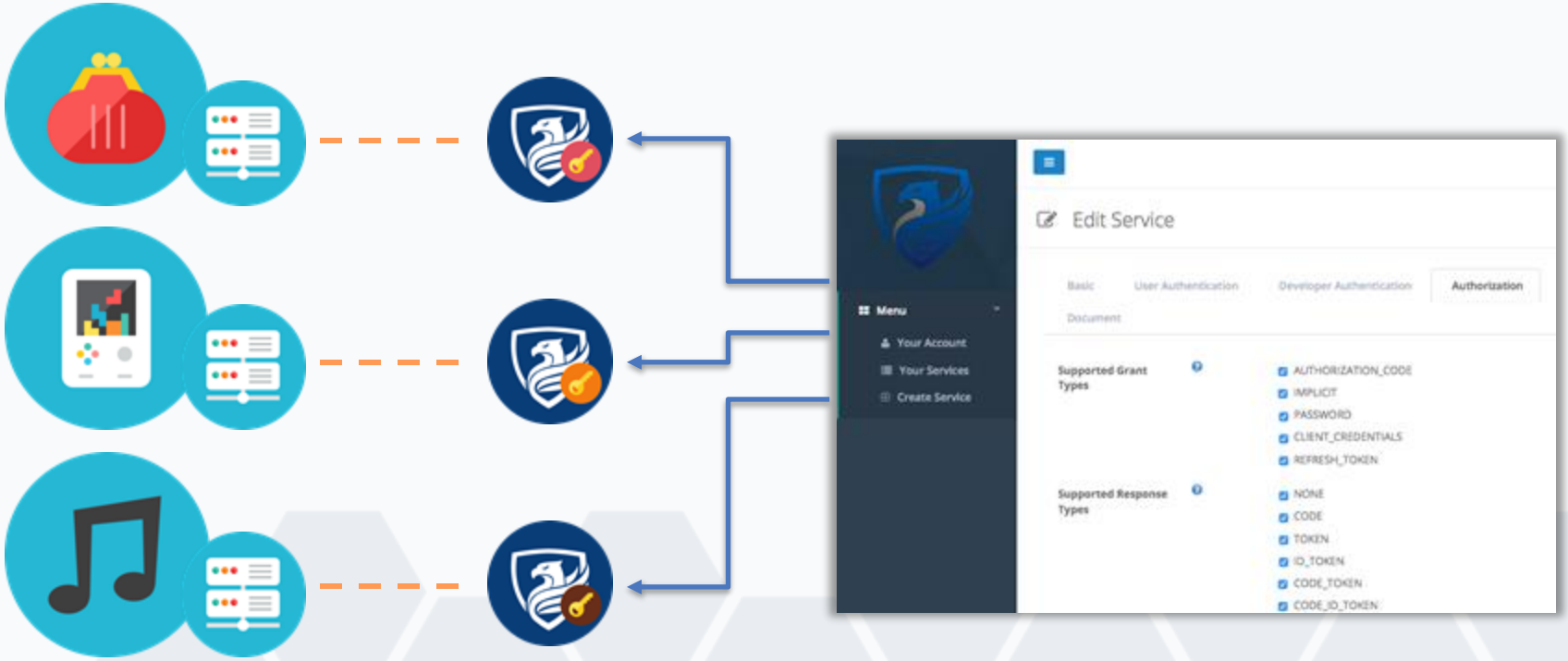


例：認証画面と認可画面を分離する



例：エンドユーザーにスコープを選択させる

事業ドメインごとにバックエンドを生成

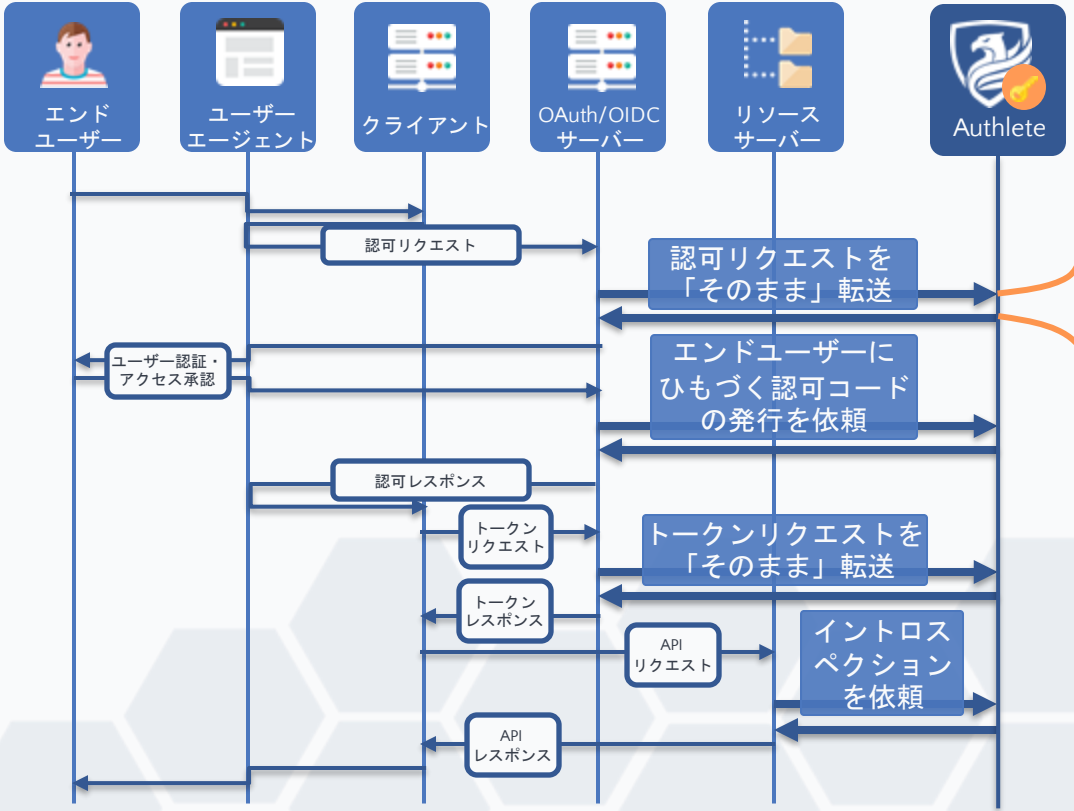


サービス・用途別の
OAuth/OIDC サーバー

サーバー毎に独立した
Authlete サービス

- ✔ 認可サーバー管理コンソール
- ✔ クライアント管理コンソール

Authlete と OAuth/OIDC サーバーの連携フロー



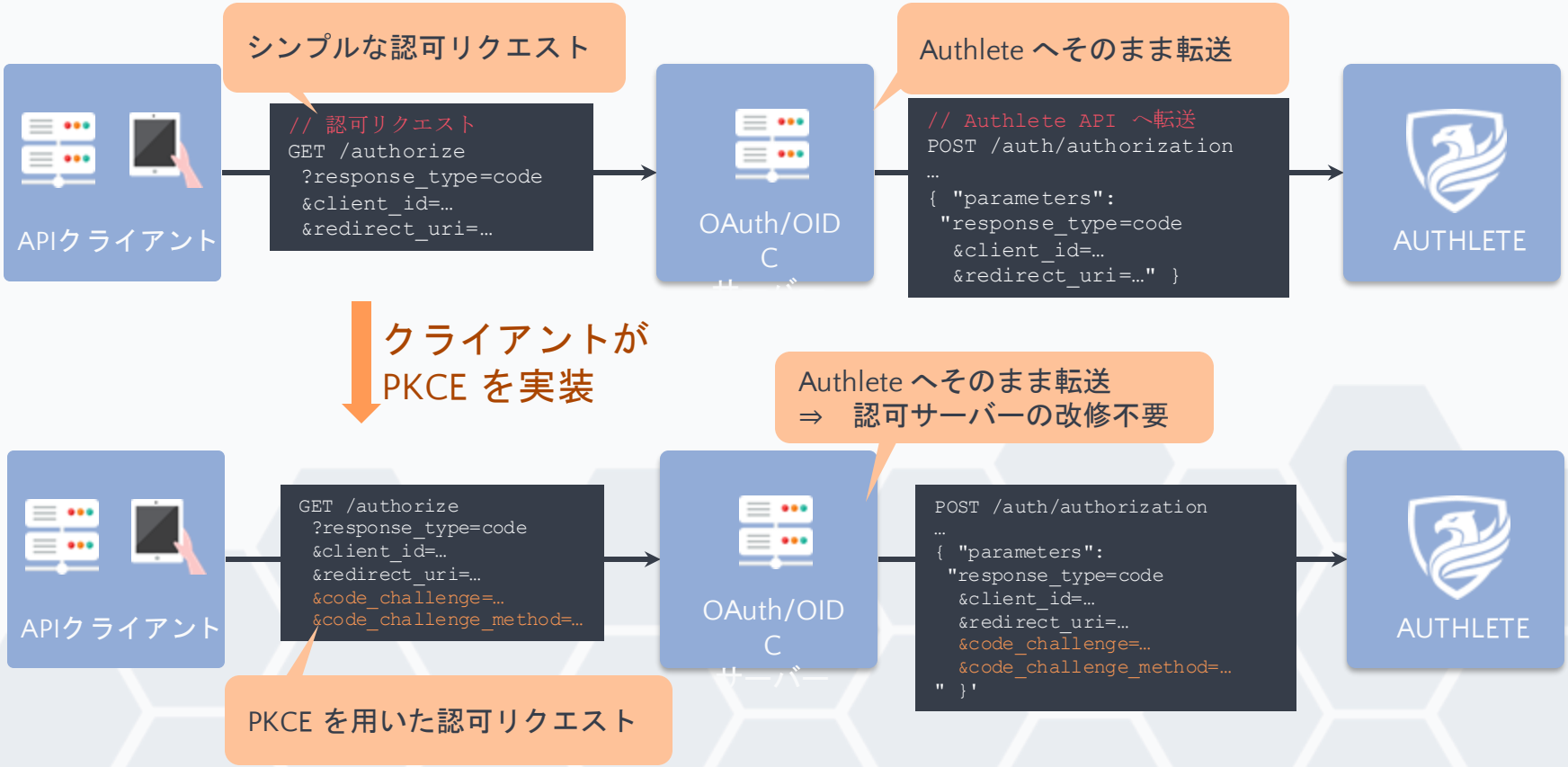
POST /auth/authorization

```
{ "parameters":
"response_type=code&client_id=5
7297408867&redirect_uri=https%3
A%2F%2Fapi.authlete.com%2Fapi%2
Fmock%2Fredirection%2F101672402
35" }'
```

OAuth/OIDC サーバーが次に
することを Authlete API が指示

```
{
  "action": "INTERACTION",
  "ticket": "c4iy3TWUzV9axH-9Q"
  ...
}
```

OAuth/OIDC の最新仕様に「知らぬ間に」対応



OAuth/OIDC の仕様から実装までコミット

金融、ヘルスケア、HR、情報銀行、
テレコム、EC、エンタメ、SI...

Authlete 導入企業からの
フィードバック

OAuth/OIDC
Experts

OAuth/OIDC ワーキング
グループにて仕様策定に貢献

OpenID Connect

OAuth/OIDC 仕様を
業界最速実装

AUTHLETE

複雑な OAuth/OIDC 仕様を
使いやすい API として提供

AUTHLETE

料金体系

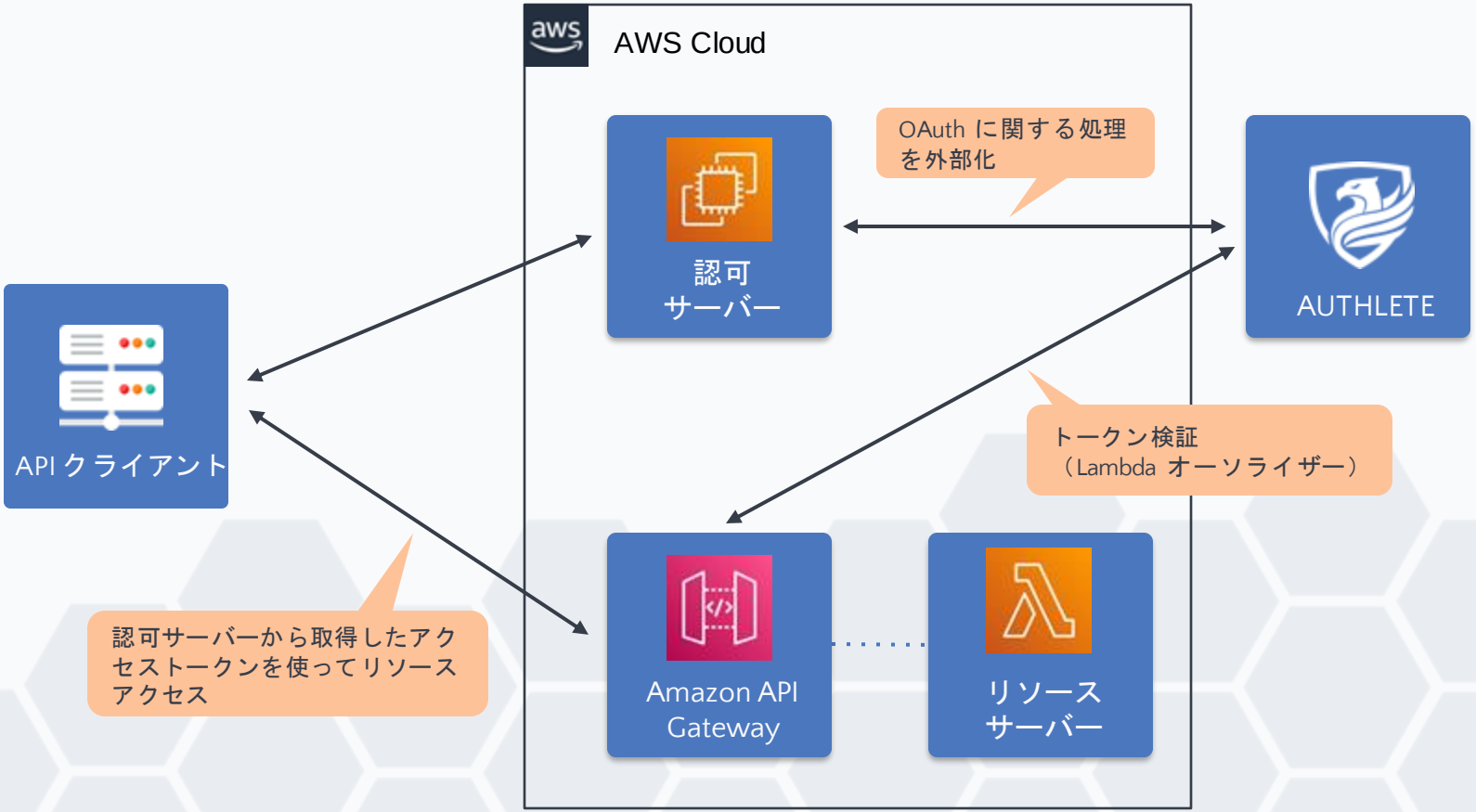
Free	Business	Enterprise
30日間 無料	¥98,000/月 から（税抜）	要見積り
マルチテナント型クラウド		専有クラウド or オンプレミス

AUTHLETE

認可サーバー&リソースサーバーの構築・動作デモ

株式会社 Authlete

全体構成

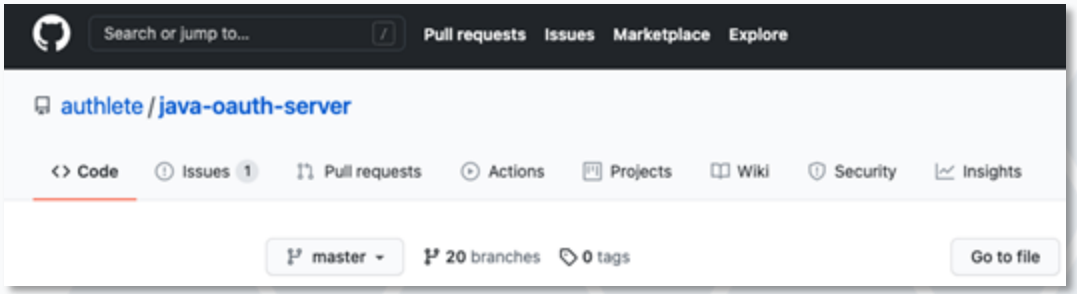

aws
AWS Cloud
認可
サーバー
OAuth に関する処理
を外部化
AUTHLETE
API クライアント
トークン検証
（Lambda オーソライザー）
認可サーバーから取得したアク
セストークンを使ってリソース
アクセス
Amazon API
Gateway
リソース
サーバー

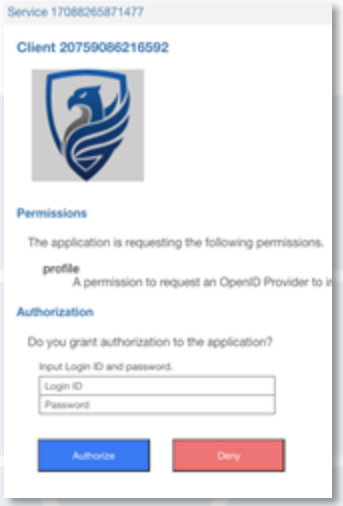
API クライアント

- Node.js (Express+EJS) で動作するWeb アプリケーション

認可サーバー（EC2）

- java-oauth-server
  - https://github.com/authlete/java-oauth-server
- Authlete 社が公開しているリファレンス実装





Amazon API Gateway

- Lambda オーソライザーを設定
  - Lambda 関数を使用したAPI へのアクセス制御
- Authlete 社 代表 川崎のブログ記事の実装
  - https://qiita.com/TakahikoKawasaki/items/b372ab49da0a9aedb76a

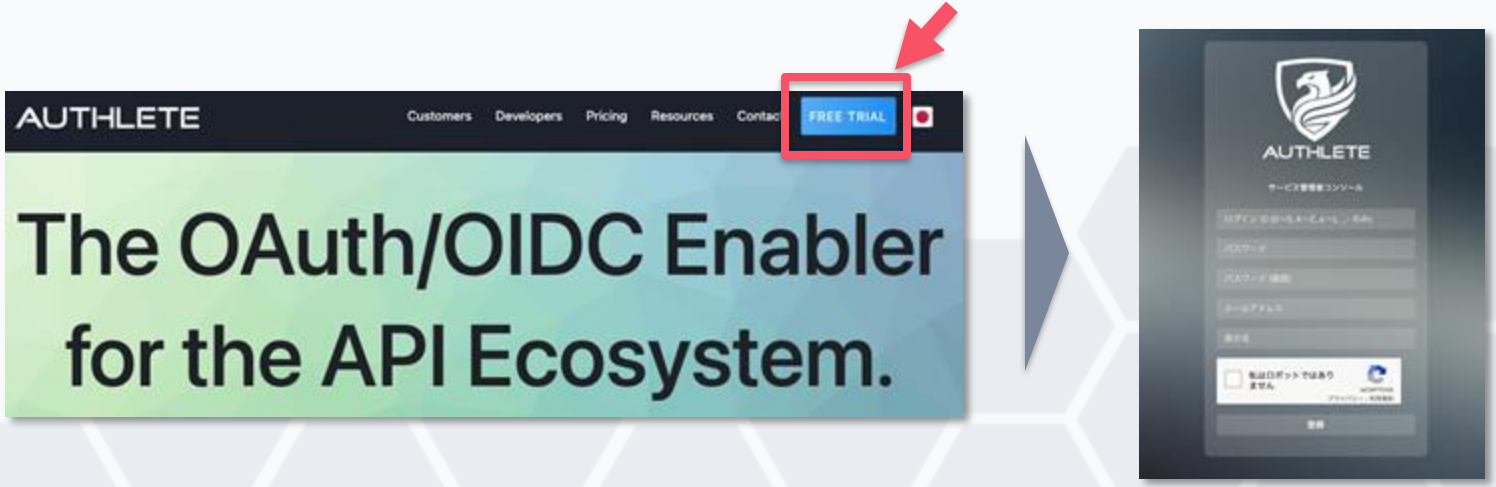
リソースサーバー（AWS Lambda）

- Node.js で動作するユーザー情報を返す関数

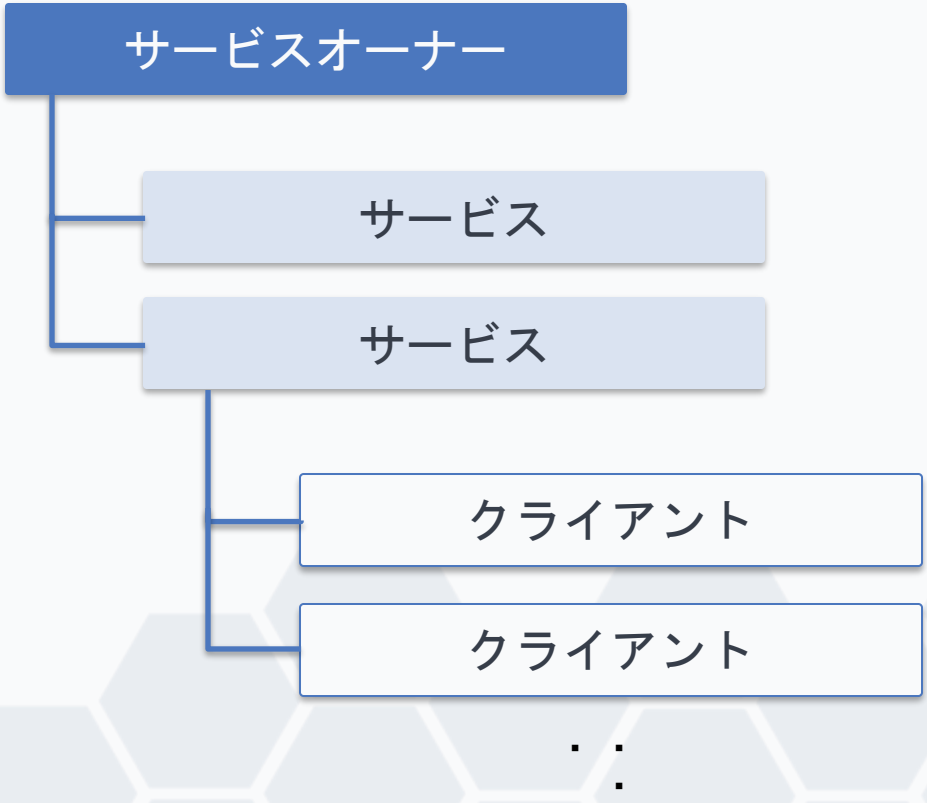
```
exports.handler = async (event) => {
  const users = [
    {
      subject: '1001',
      name: 'John Flibble Smith',
      email: 'john@example.com',
      address: 'USA Flibble',
      phoneNumber: '+1 (425) 555-1212',
    },
    {
      subject: '1002',
      name: 'Jane Smith',
      email: 'jane@example.com',
      address: 'Chile',
      phoneNumber: '+56 (2) 687 2400',
    },
    {
      subject: '1003',
      name: 'Max Meier',
      email: 'max@example.com',
      address: 'Germany',
      phoneNumber: '+49 (30) 210 94-0',
    },
  ];

  const result = users.find((user) => user.subject === event.principalId);
  return result;
};
```

Authlete 環境

- Authlete 社の Web サイトからアクセスできる共有環境
  - 30日間 無料で利用可能（Free Plan）

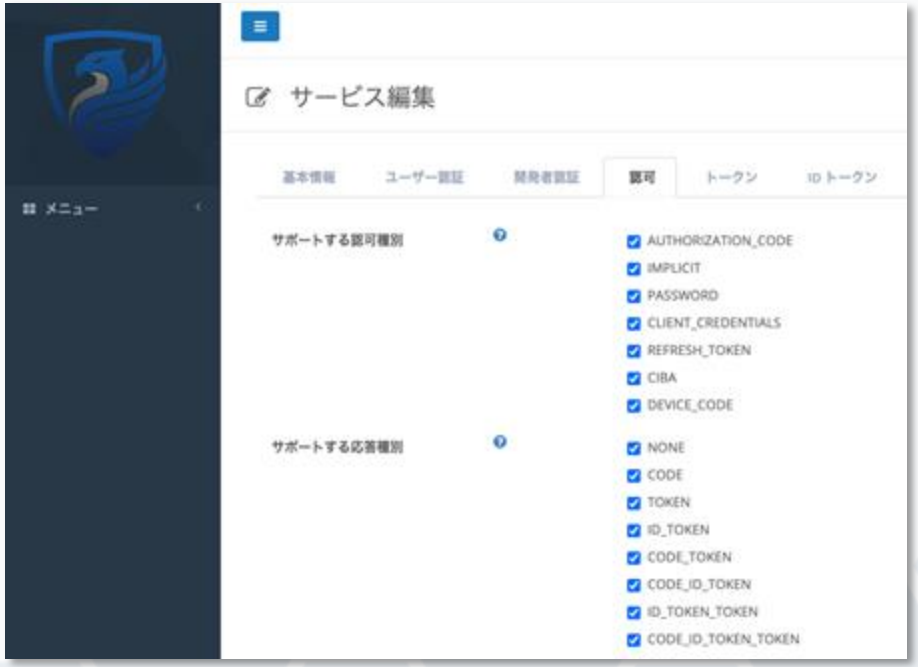


Authlete 環境の構成



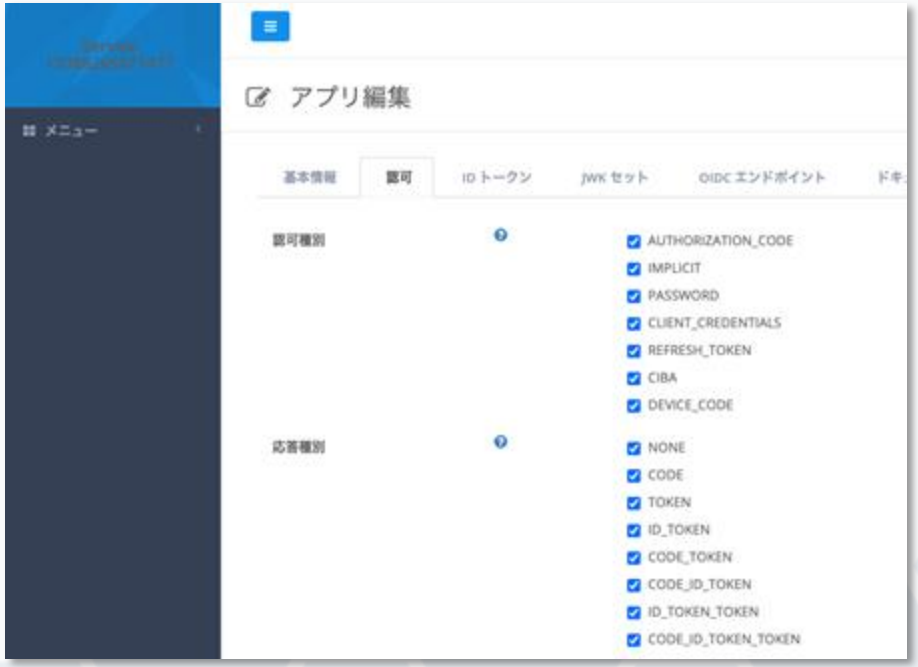
- サービスオーナー
  - Authlete ユーザーアカウント
- サービス
  - 認可サーバーの単位
  - 一つのサービスオーナーに複数のサービスがぶら下がる
- クライアント
  - 認可サーバーを利用するクライアント
  - 一つのサービスに複数のクライアントがぶら下がる

サービスオーナーコンソール



- サービスの設定変更を行うためのコンソール画面
- サービスオーナーアカウントでログイン

クライアントコンソール



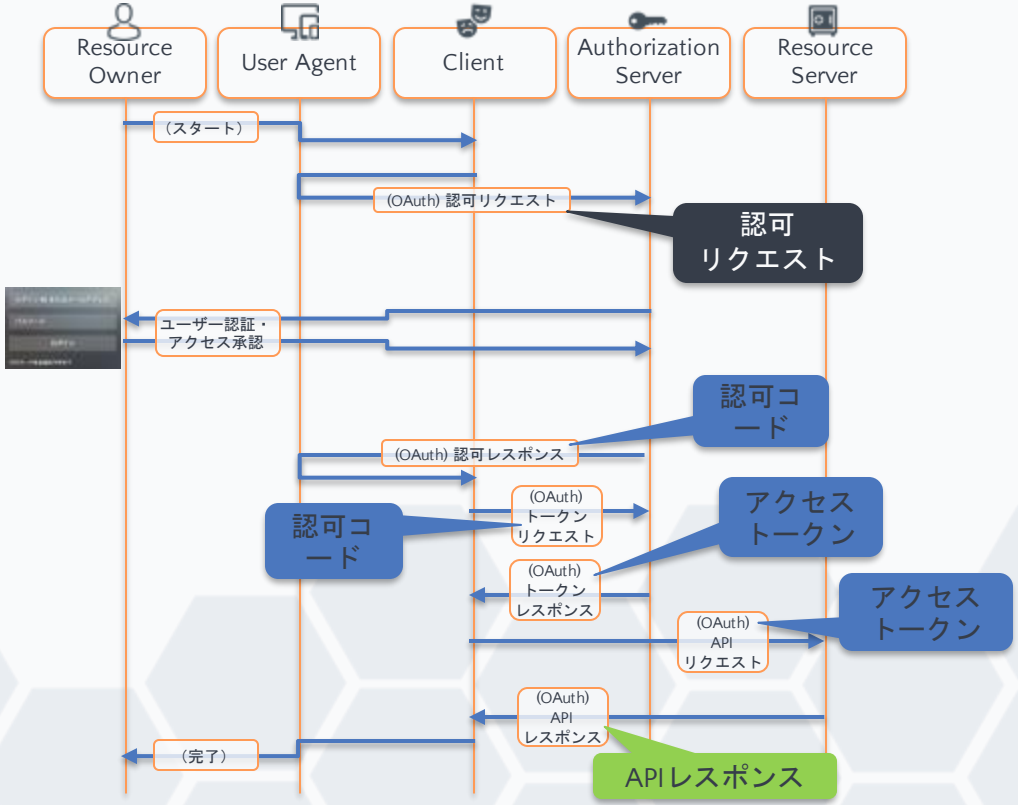
- クライアントの設定変更を行うためのコンソール画面
- サービスの API キーとシークレットでログイン

構築手順

1. Authlete 環境の設定
2. Amazon EC2 上で java-oauth-server の起動
3. AWS Lambda 関数の設定
4. Amazon API Gateway の設定
5. クライアント Web アプリケーションの起動

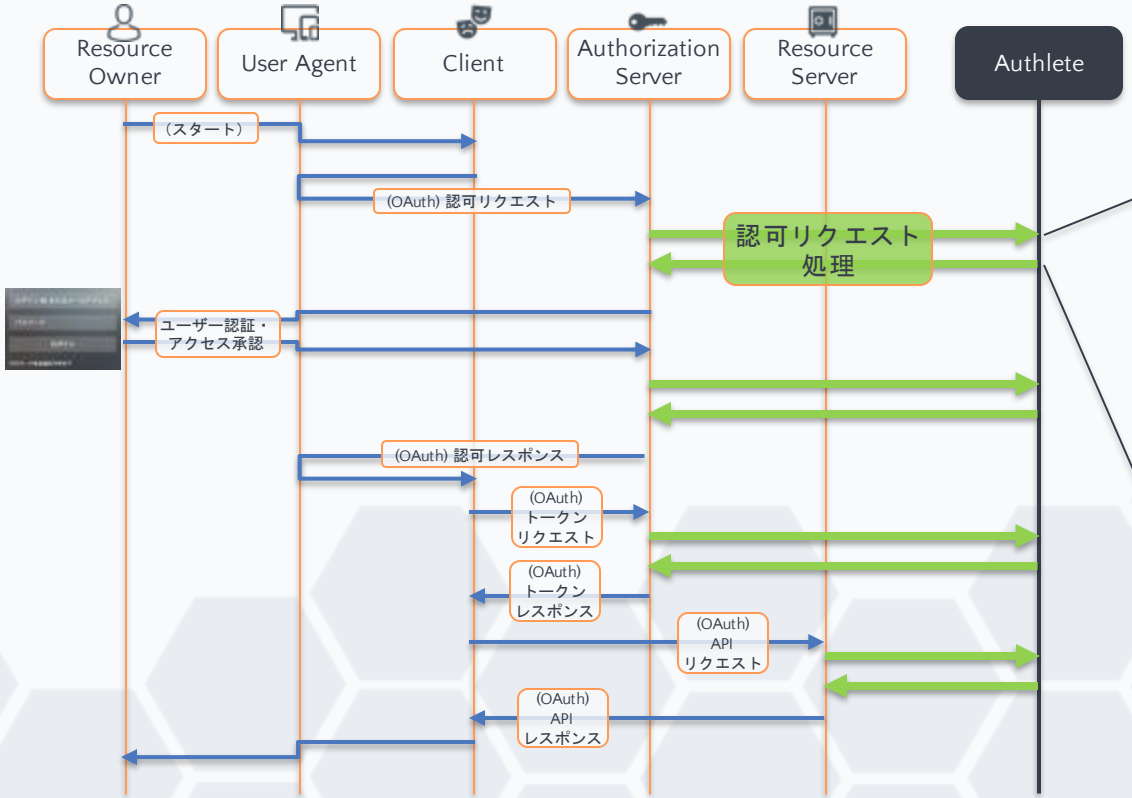
“OAuth 2.0” の基本的なシーケンス

(Authorization Code Grant Flow / Bearer Token)



- 認可リクエストを受け付けたら認可コードを返す
- 認可コードをもらったらアクセストークンを返す
- アクセストークンつきのAPIリクエストを受け付けてレスポンスを返す
- …もしかしてかんたん?

1. 認可リクエスト処理



POST /auth/authorization

```
{ "parameters":
"response_type=code&client_id=57297408867&red
irect_uri=https%3A%2F%2Fapi.authlete.com%2Fap
i%2Fmock%2Fredirection%2F10167240235" }'
```

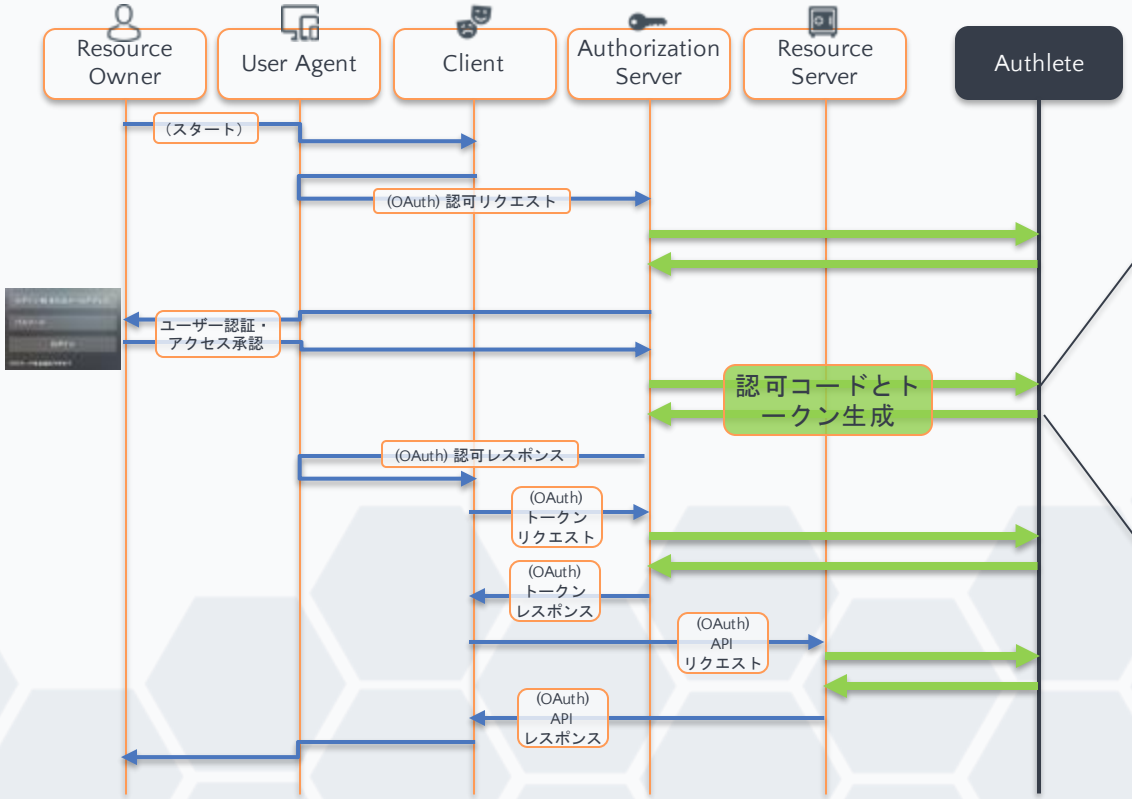
認可リクエストのクエリパラメーターを「そのまま」送信

「チケット」と「認証・認可処理に必要な情報」を返却

```
{
  "action": "INTERACTION",
  "client": {
    (クライアントに関する情報)
  },
  "service": {
    (Authlete サービスに関する情報)
  },
  "ticket":
"c4iy3TWGn74UMO7ihRl0ZS8OEUzV9axBlBbJbqxH-9Q"
  […]
}
```

AUTHLETE

2. 認可コードとトークン生成



POST /auth/authorization/issue

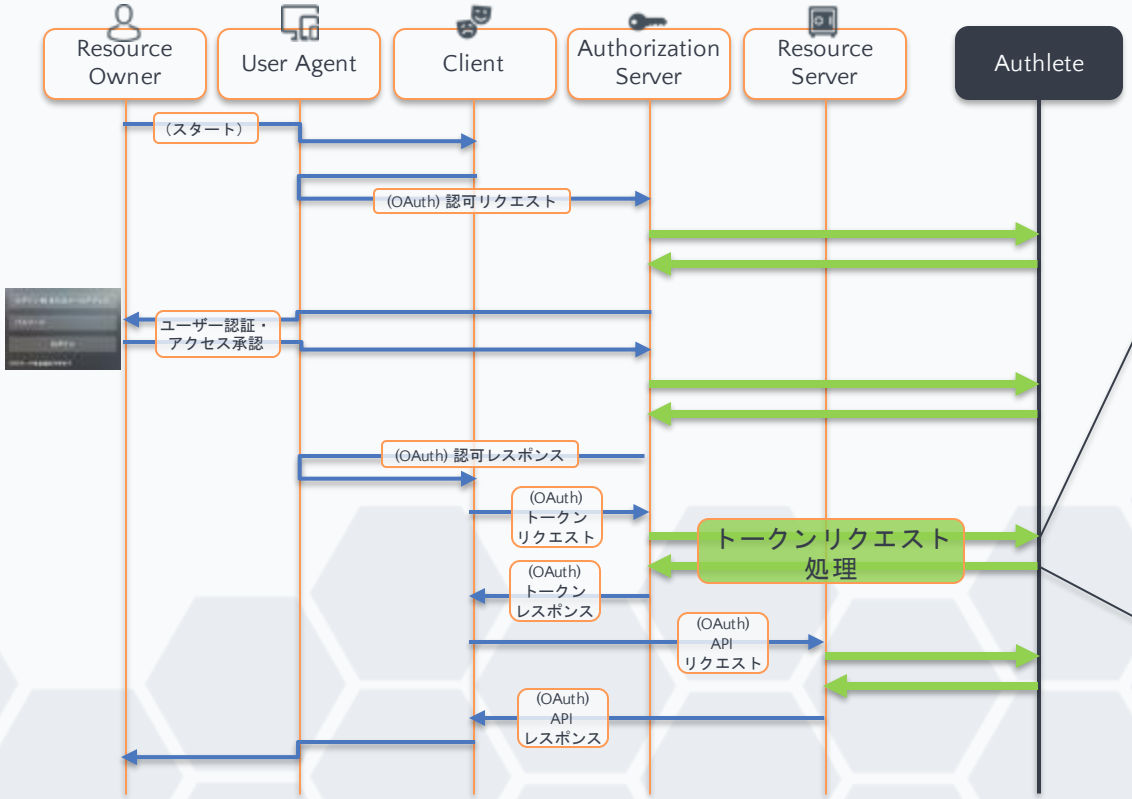
```
{
  "ticket": "FFgB9gwb_WXh6g1u-UQ8ZI-d_k4B-o-cm7RkVzI8Vnc",
  "subject": "78yM7DpDNgUTF7h4"
}
```

「チケット」と
「ユーザー識別子」を送信

「クライアントに返却する
レスポンス」を返却

```
{
 "action": "LOCATION",
  "responseContent":
"https://api.authlete.com/api/mock/redirection/10167240235?code=_VSzpgug3OpdUymcbLYJjH1v9tE4hSPaxX_pSkxUUyk"
  […]
}
```

3. トークンリクエスト処理



POST /auth/token

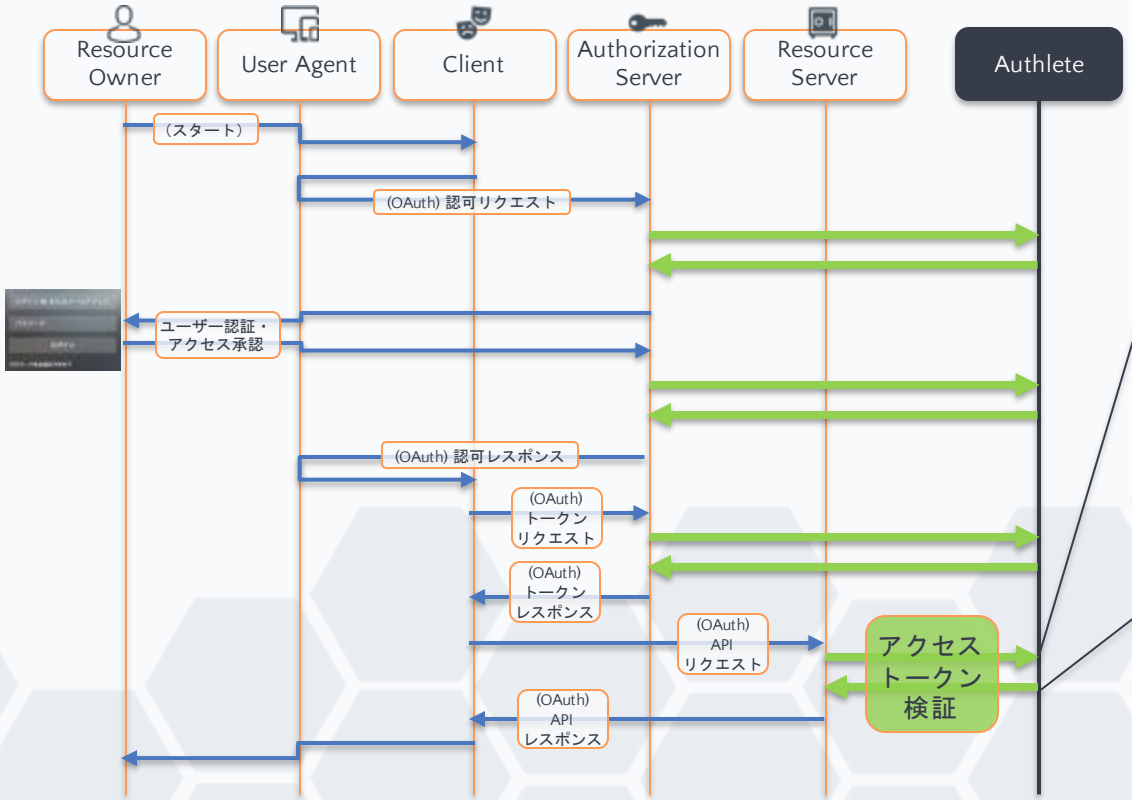
```
{ "parameters":
"grant_type=authorization_code&code=DkKMvKzrR
AsWErsWCChNX_gydqqgA55AW2OJlXmNTQI&redirect_u
ri=https%3A%2F%2Fapi.authlete.com%2Fapi%2Fmoc
k%2Fredirection%2F10167240235",
  "clientId": 57297408867,
  "clientSecret": "[…]", […] }
```

トークンリクエストの
クエリパラメーターを
「そのまま」送信

「クライアントに返却する
レスポンス」を返却

```
{
  "responseContent":
"{¥"scope¥":null,¥"expires_in¥":86400,¥"token
_type¥":¥"Bearer¥",¥"refresh_token¥":¥"sdQqY9
Tbhsq6ZsWm1rZLgW4A3yxIk6RcgmmexHZ9BXB¥",¥"acc
ess_token¥":¥"rt5bUJsGfS17YCSmYGtgelMtokTLdoC
mBe4VUFCk1YZ¥"}",
  […]
}
```

4. アクセストークン検証



POST /auth/introspection

```
{ "token": "VFGsNK-
5sXiqterdaR7b5QbRX9VTwVCQB87jbr2_xAI" }
```

トークンの値を送信

トークンにひもづく
詳細な情報を返却


```
{
  "type": "introspectionResponse",
  "resultCode": "A056001",
  "resultMessage": "[A056001] The access
token is valid.",
  "action": "OK",
  "clientId": 57297408867,
  "clientIdAliasUsed": false,
  "existent": true,
  "expiresAt": 1511252965000,
  "refreshable": true,
  "responseContent": "Bearer
error=¥"invalid_request¥"",
  "subject": "78yM7DpDNgUTF7h4",
  "sufficient": true,
  "usable": true
}
```

まとめ

- AWS 上に認可サーバーとリソースサーバーを構築し動作確認をしました
- Authlete を利用すると複雑な OAuth/OIDC 機能を自前で実装する必要がなくなります
- Authlete の使い方のイメージが伝わっていれば幸いです。

AUTHLETE

Thank You

www.authlete.com