[AWS Black Belt Online Seminar]
Amazon Cognito

サービスカットシリーズ

Solutions Architect 辻 義一
2020/06/30

AWS 公式 Webinar
https://amzn.to/JPWebinar

過去資料
https://amzn.to/JPArchive

自己紹介

辻　義一（つじ　よしかず）

- 西日本担当 ソリューションアーキテクト
- 簡単な経歴
  - 大阪生まれの大阪育ち。
  - 独立系SIerでインフラエンジニア。
- AWSのすきな所

  ～ 安い、早い、おもしろい ～

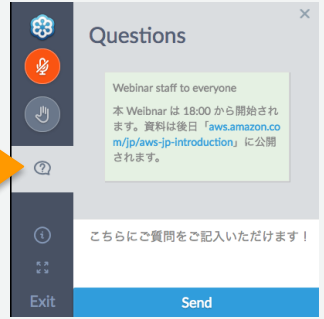
AWS Black Belt Online Seminar とは

「サービス別」「ソリューション別」「業種別」のそれぞれのテーマに分かれて、アマゾン ウェブ サービス ジャパン株式会社が主催するオンラインセミナーシリーズです。

質問を投げることができます！

- 書き込んだ質問は、主催者にしか見えません
- 今後のロードマップに関するご質問は
  お答えできませんのでご了承下さい

① 吹き出しをクリック
② 質問を入力
③ Sendをクリック



Twitter ハッシュタグは以下をご利用ください
#awsblackbelt

内容についての注意点

- 本資料では2020年6月30日時点のサービス内容および価格についてご説明しています。最新の情報はAWS公式ウェブサイト(http://aws.amazon.com)にてご確認ください。

- 資料作成には十分注意しておりますが、資料内の価格とAWS公式ウェブサイト記載の価格に相違があった場合、AWS公式ウェブサイトの価格を優先とさせていただきます。

- 価格は税抜表記となっています。日本居住者のお客様には別途消費税をご請求させていただきます。

- AWS does not offer binding price quotes. AWS pricing is publicly available and is subject to change in accordance with the AWS Customer Agreement available at http://aws.amazon.com/agreement/. Any pricing information included in this document is provided only as an estimate of usage charges for AWS services based on certain information that you have provided. Monthly charges will be based on your actual use of AWS services, and may vary from the estimates provided.

Agenda

- AWS におけるアプリ ユーザ認証の主な選択肢
- Cognito
- Cognito ユーザプール
  - 構成要素
  - サインインの実装
  - サインイン後の実装
  - 各種機能

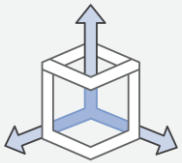
まず要件を整理

アプリ

- モバイル / SPA Web / 非SPA Web
- サーバ環境 (EC2 / ECS,EKS / Lambda)
- 新規 / 既存
- 単一 / 複数

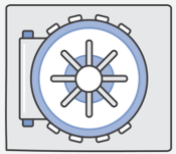
ユーザ

- ユーザ自身で作成 / 管理者が作成
- マルチテナント / シングルテナント
- 一般コンシューマ / 企業従業員 / 自社社員



スケーラビリティ

- ユーザ数
- 認証リクエスト数



セキュリティ

- 認証方法
- リスクに応じた制御
- 外部 ID プロバイダ対応

など

AWS におけるアプリ ユーザ認証の主な選択肢

アプリの特性、対象ユーザ、認証機能要件、非機能要件などに応じて選択する。

Amazon Cognito

モバイル・Web アプリ向けの ユーザ認証を提供

AWS Single Sign-On (SSO)

社員に SAML 対応 アプリや AWS 管理者 へのログインを提供

IDaaS を利用

AWS パートナー のサービス

Auth0 okta onelogin

クラウドサービス として ユーザ認証を提供

パッケージを利用

on EC2

OSS、プロプライエタリ のパッケージで実現

独自あるいは ライブラリ を使って実装

アプリの 1 機能 として実装

アプリと同様に仮想マシン上、コンテナ上、Lambda上で実装が可能

ユーザ認証をマネージドで提供

AWS におけるアプリ ユーザ認証の主な選択肢

アプリの特性、対象ユーザ、認証機能要件、非機能要件などに応じて選択する。

Amazon Cognito

モバイル・Web
アプリ向けの
ユーザ認証を提供

AWS Single Sign-On (SSO)

社員にSAML対応
アプリやAWSへの
ログインを提供

IDaaS を利用

AWS パートナー
のサービス

Auth0 okta
onelogin

クラウドサービス
として
ユーザ認証を提供

パッケージを利用

on EC2

OSS、プロプライエタリ
のパッケージで実現

独自あるいは ライブラリ を使って実装

アプリの 1 機能
として実装

アプリと同様に仮想マシン上、コンテナ上、Lambda上で実装が可能

ユーザ認証をマネージドで提供

Amazon Cognito

API ベースで実装されるモバイルアプリや Web アプリに
ユーザ認証機能を提供するサービス

ユーザプール

- 独自のユーザディレクトリに加え、
外部 ID プロバイダでのログインに基づき、
アプリへのアクセスに利用できるトークンを提供

ID プール

（フェデレーティッド アイデンティティ）

- Cognito ユーザプールに加え、
外部 ID プロバイダでのログインに基づき、
AWS にアクセスできるクレデンシャルを提供

Cognito の ユーザプール、ID プール 以外のサービス

Cognito Sync

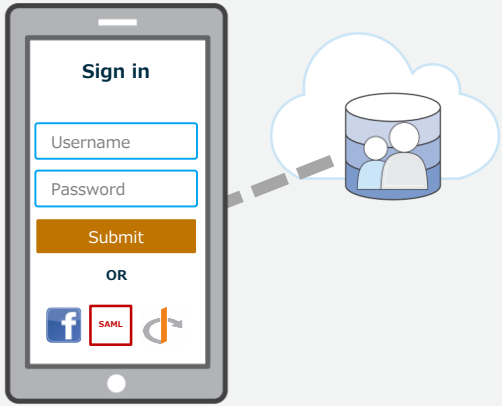
モバイルアプリとクラウド間のデータ同期を実現

Cognito Sync は現在もサービス提供を継続していますが、同様の機能を含むより高機能な新サービス

AWS AppSync

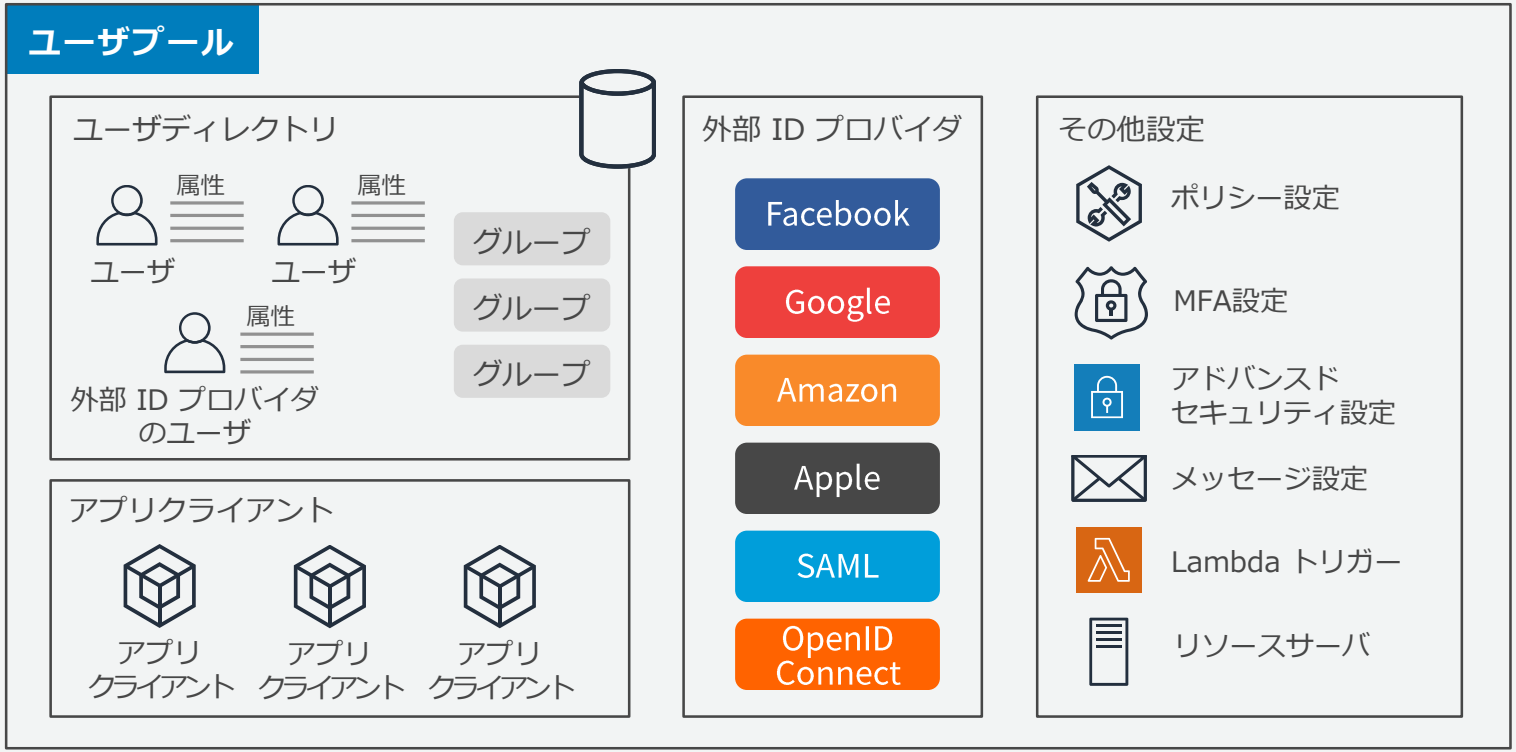
が提供されています。新規の開発では **AppSync** の利用を推奨します。

ユーザプール



ユーザプールの主な構成要素

リージョン毎に必要に応じて複数のユーザプールの作成が可能。



ユーザ

ユーザは管理者が作成することも、ユーザ自身にサインアップを許可することも可能。
外部 ID プロバイダのユーザについては初回サインイン時に情報が自動的に登録される。

- Cognito 固有ユーザ情報

Username	Enabled
UserStatus	Groups
PreferredMfaSetting	UserMFASettingList
roles	prefered_role

etc

- 標準属性 (OIDC仕様)

email	email_verified
phone_number	phone_number_verified
preferred_username	sub
name	updated_at

etc

- カスタム属性

- サインインに使用したデバイス情報
(有効時のみ記録)

デバイスキー
名前
最後の IP
SDK
最後の利用日時

カスタム属性を活用すると、ユーザについての任意の情報を保存することが可能だが、頻繁に参照・変更・検索する情報については DB に保存する方が望ましい。

ユーザがサインインに使用できる情報

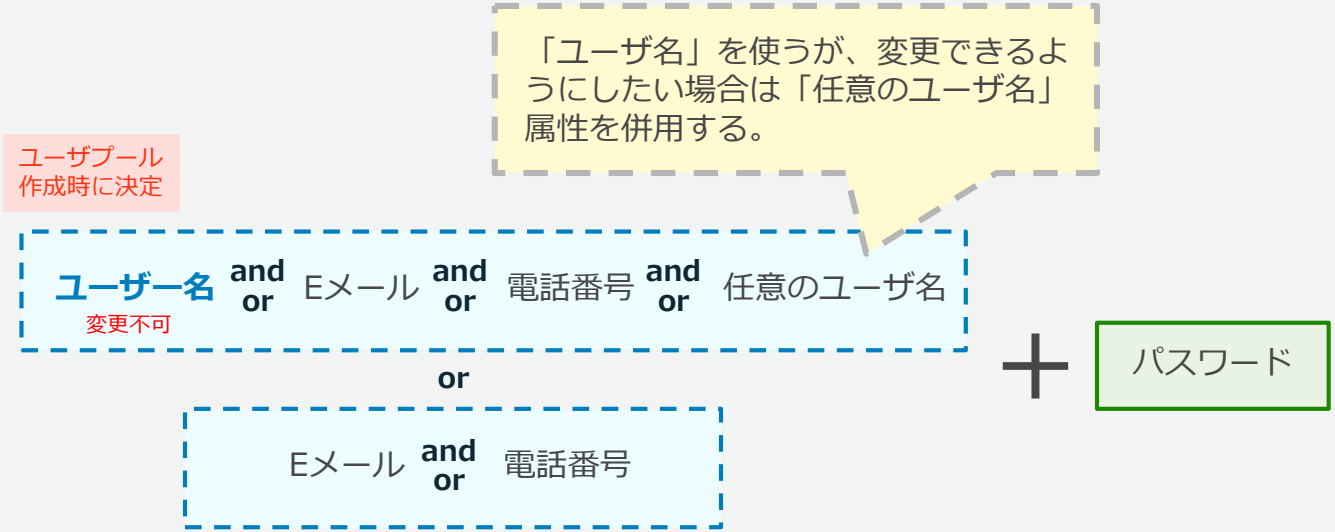
サインイン ユーザ名

＋

パスワード

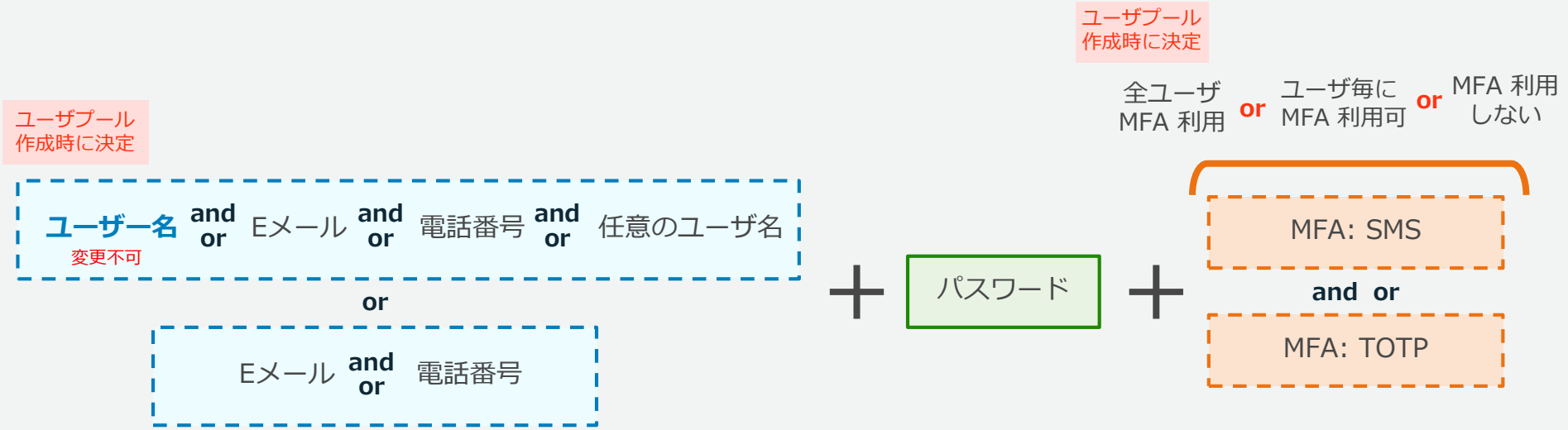
ユーザがサインインに使用できる情報

ユーザプール作成時に、ユーザがサインイン時に何を使用できるかを設定する。
作成後の変更はできないため、検討してからユーザプールの作成を行う。



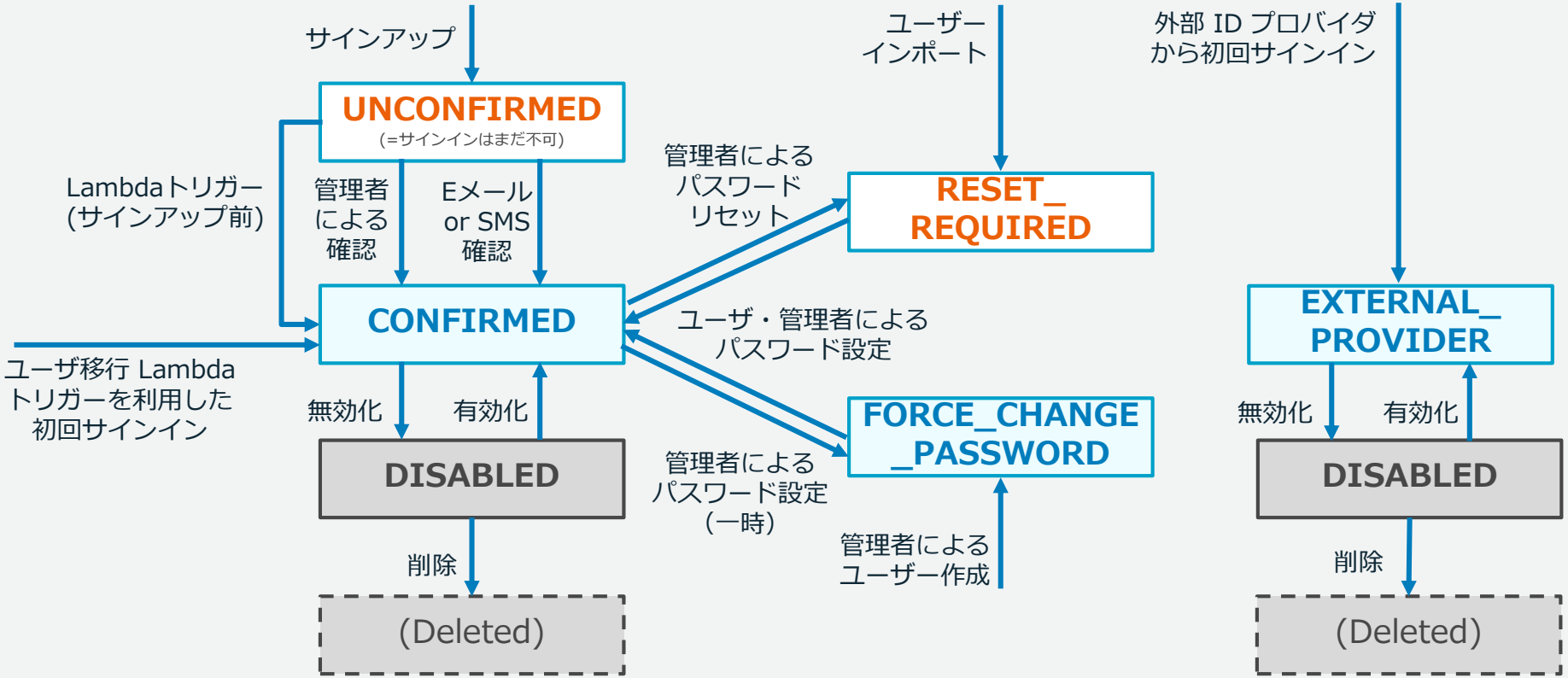
ユーザがサインインに使用できる情報

ユーザプール作成時に、ユーザがサインイン時に何を使用できるかを設定する。
作成後の変更はできないため、検討してからユーザプールの作成を行う。



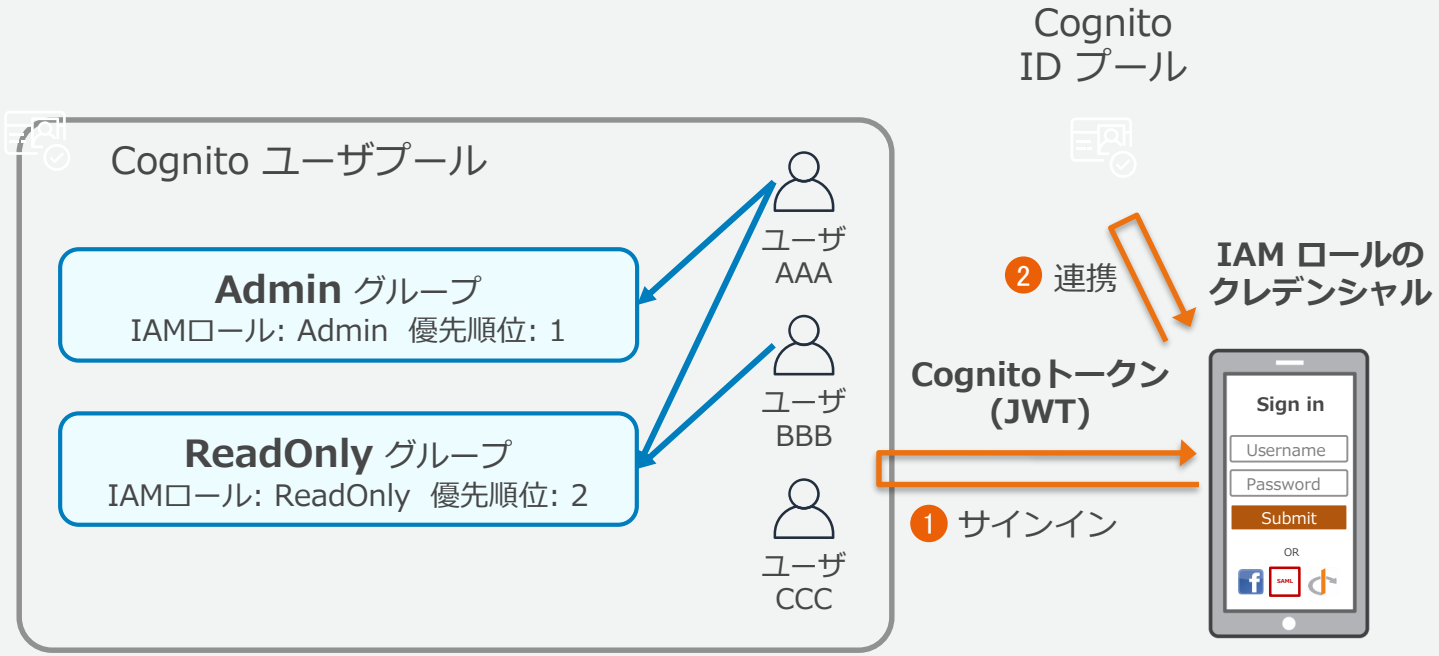
ユーザのステータス

ユーザは以下のようにステータスが管理される。



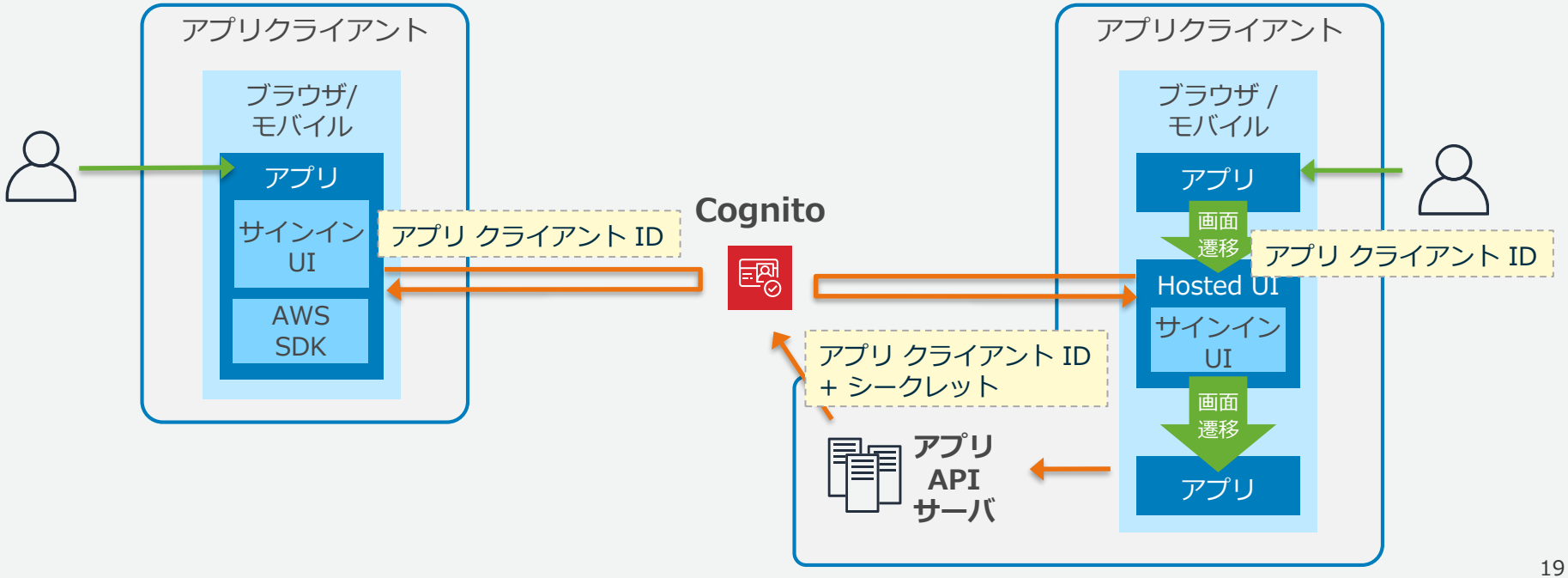
グループ

ユーザは複数のグループに入れることが可能。グループは属性としてアプリケーションから確認できる事に加え、Cognito ID プールとの連携では利用できる IAM ロールやその優先順位の指定に使用できる。AppSync にもグループに応じた権限設定機能がある。



アプリ クライアント

API 呼び出し、Hosted UI の利用、OAuth による認可などユーザプールにアクセスするアプリケーションをそれぞれ登録し、利用できる機能や権限を設定する。
登録されたアプリケーションは IAM クレデンシャルを利用せずにアクセスできる。



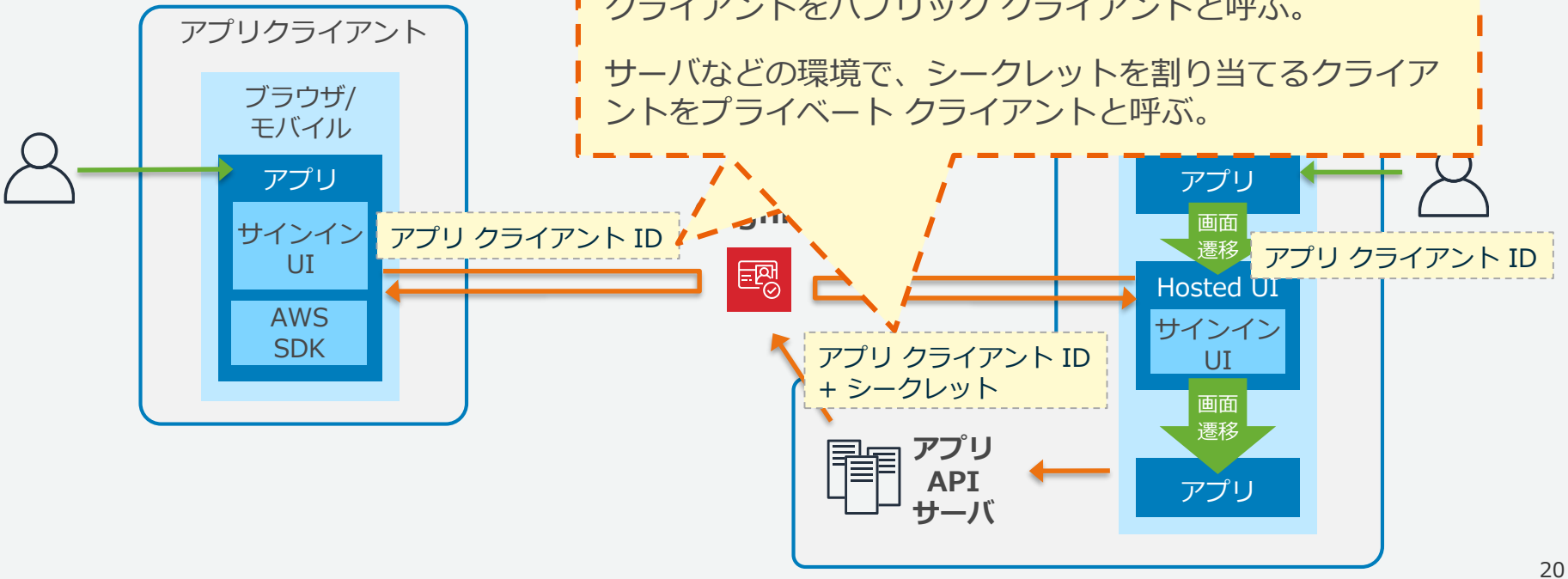
アプリ クライアント

API 呼び出し、Hosted UI の利用
アプリケーションをそれぞれ登録
登録されたアプリケーションは I

アプリクライアント ID、シークレットでアプリケーションを識別する。

ブラウザ上の JavaScript などシークレットを保護できない環境の場合は、シークレットを割り当てない。そのようなクライアントをパブリック クライアントと呼ぶ。

サーバなどの環境で、シークレットを割り当てるクライアントをプライベート クライアントと呼ぶ。



アプリ クライアント

アプリ クライアント毎に以下の設定があり、マネージメントコンソールでは対象の API によって設定ページが分かれている。

- 全般設定 – アプリクライアント
  - クライアント名
  - **クライアント ID**
  - **クライアント シークレット**
  - 更新トークンの有効期限
  - 有効な認証フロー
  - 読み書きできる属性
- アプリの統合 – アプリクライアント設定
  - 有効な外部 ID プロバイダー
  - コールバック URL
  - サインアウト URL
  - 許可する OAuth フロー
  - 許可する OAuth スコープ

アプリ クライアント

アプリ クライアント毎に以下の設定があり、マネージメントコンソールでは対象の API によって設定ページが分かれている。

■ 全般設定 – アプリクライアント

- クライアント名
- **クライアント ID**
- **クライアント シークレット**
- 更新トークンの有効期限
- 有効な認証フロー
- 読み書きできる属性

Cognito Identity Provider API

■ アプリの統合 – アプリクライアント設定

- 有効な外部 ID プロバイダー
- コールバック URL
- サインアウト URL
- 許可する OAuth フロー
- 許可する OAuth スコープ

Cognito Auth API や Hosted UI

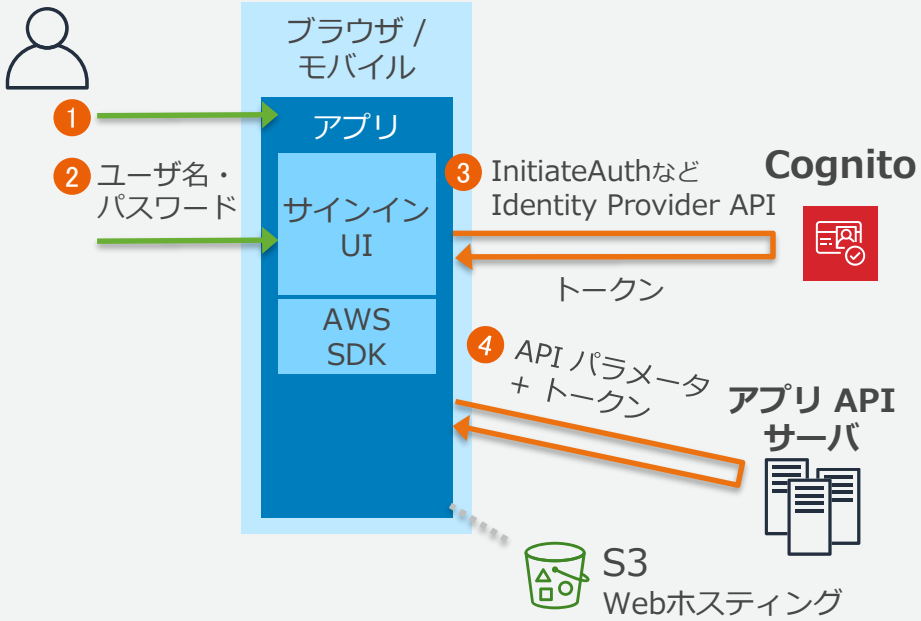
ユーザ
プール

サインインの実装

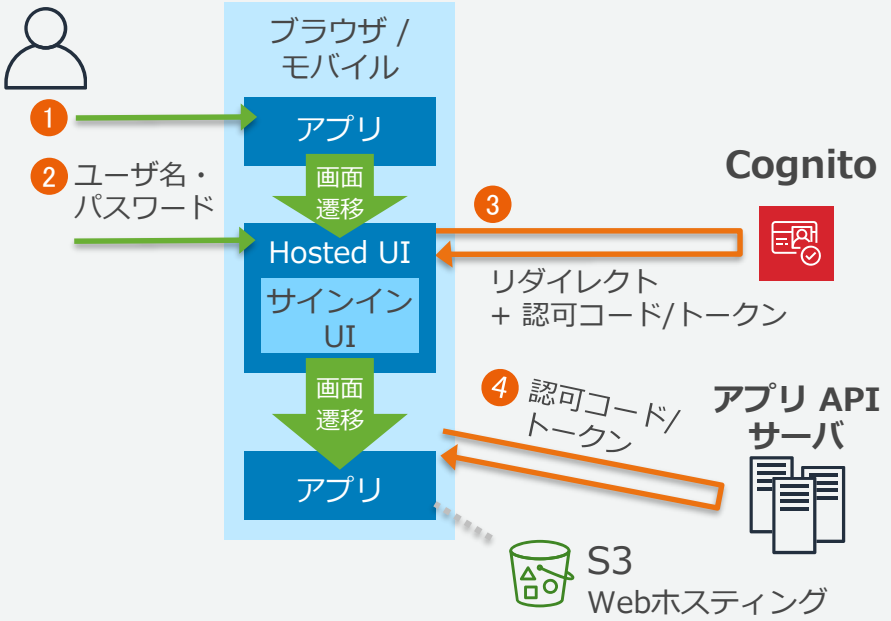
利用できる API は 2 セット

OAuth / OIDC の
IdP の機能を一部実装

❶ Cognito Identity Provider API を利用

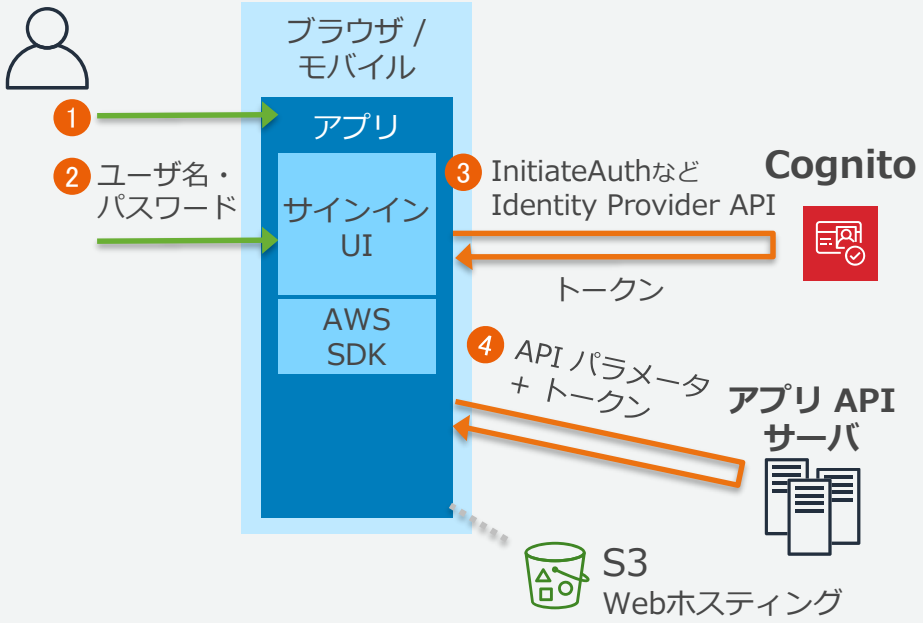


❷ Cognito Auth API と Hosted UI を利用

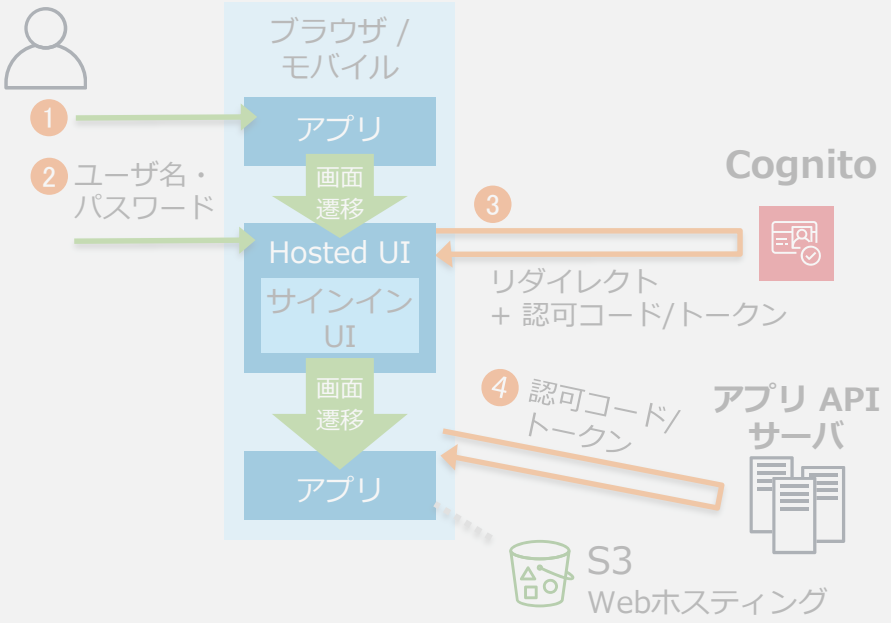


利用できる API は 2 セット

1 Cognito Identity Provider API を利用

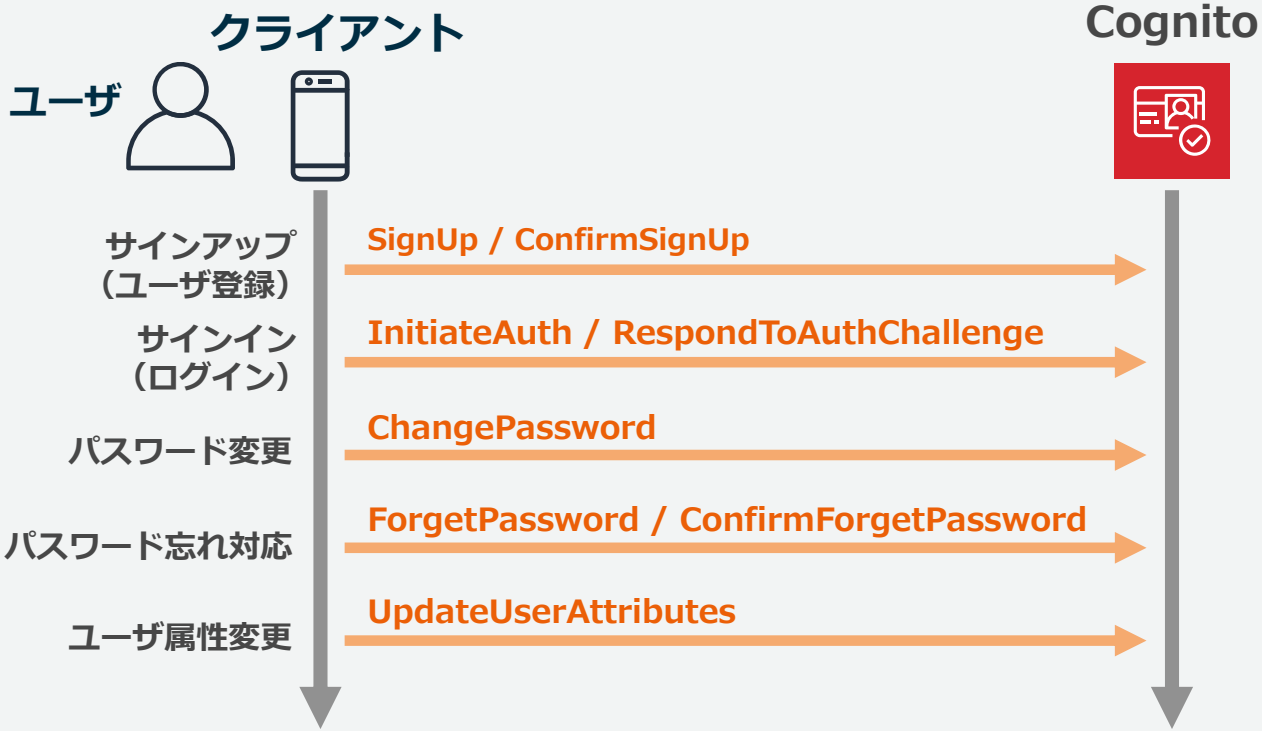


2 Cognito Auth API と Hosted UI を利用



❶ Cognito Identity Provider API を利用

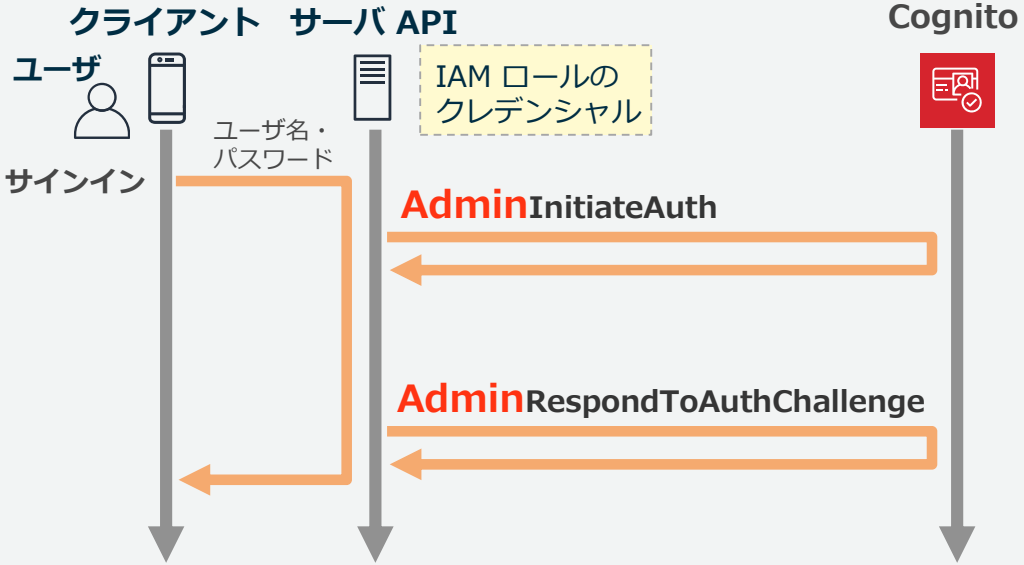
サインインを始めとし、各種操作に対応したユーザプール独自の API が用意されている。
ユーザがアプリケーションに入力した内容を API に送って、各種操作を実現する。



❶ 呼び出し元によって異なる API あり

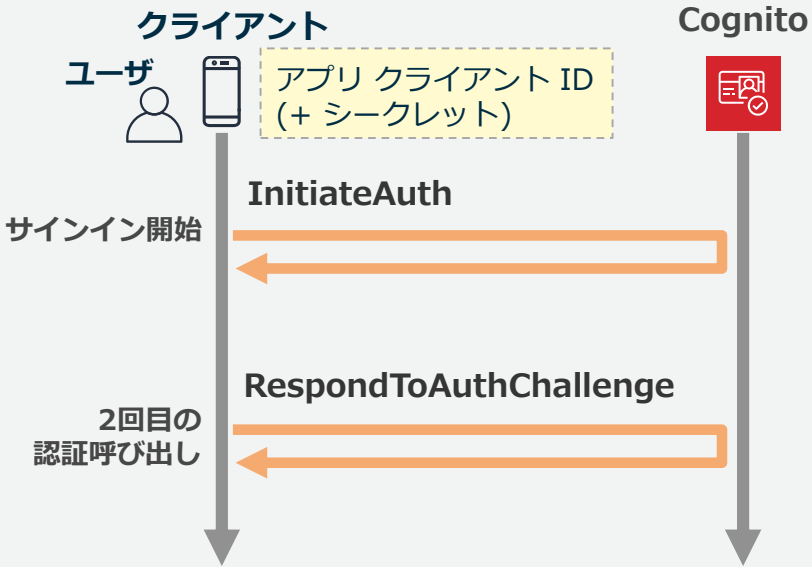
管理者としてサーバ側から呼び出す前提の Admin API と、
ユーザとしてサーバ側からもクライアン側からも呼び出せる API がある。

■ Admin API



非SPA の Web でも利用できる。
機能追加などの自由度が高い。

■ 非Admin API

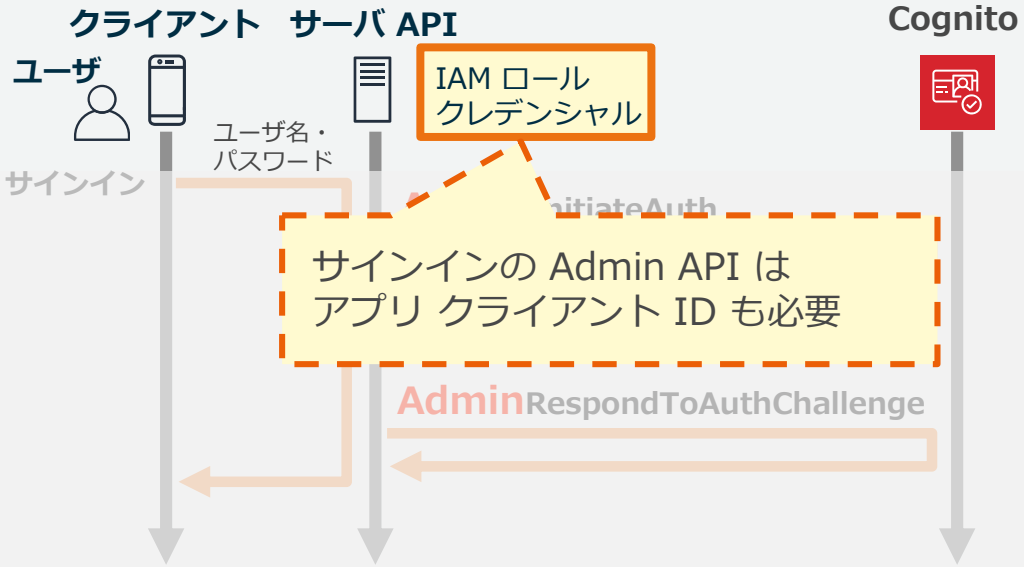


クライアントからも呼び出せるため、
シンプルな実装にできる。

❶ 呼び出し元によって異なる API あり

管理者としてサーバ側から呼び出す前提の Admin API と、
ユーザとしてサーバ側からもクライアン側からも呼び出せる API がある。

■ Admin API



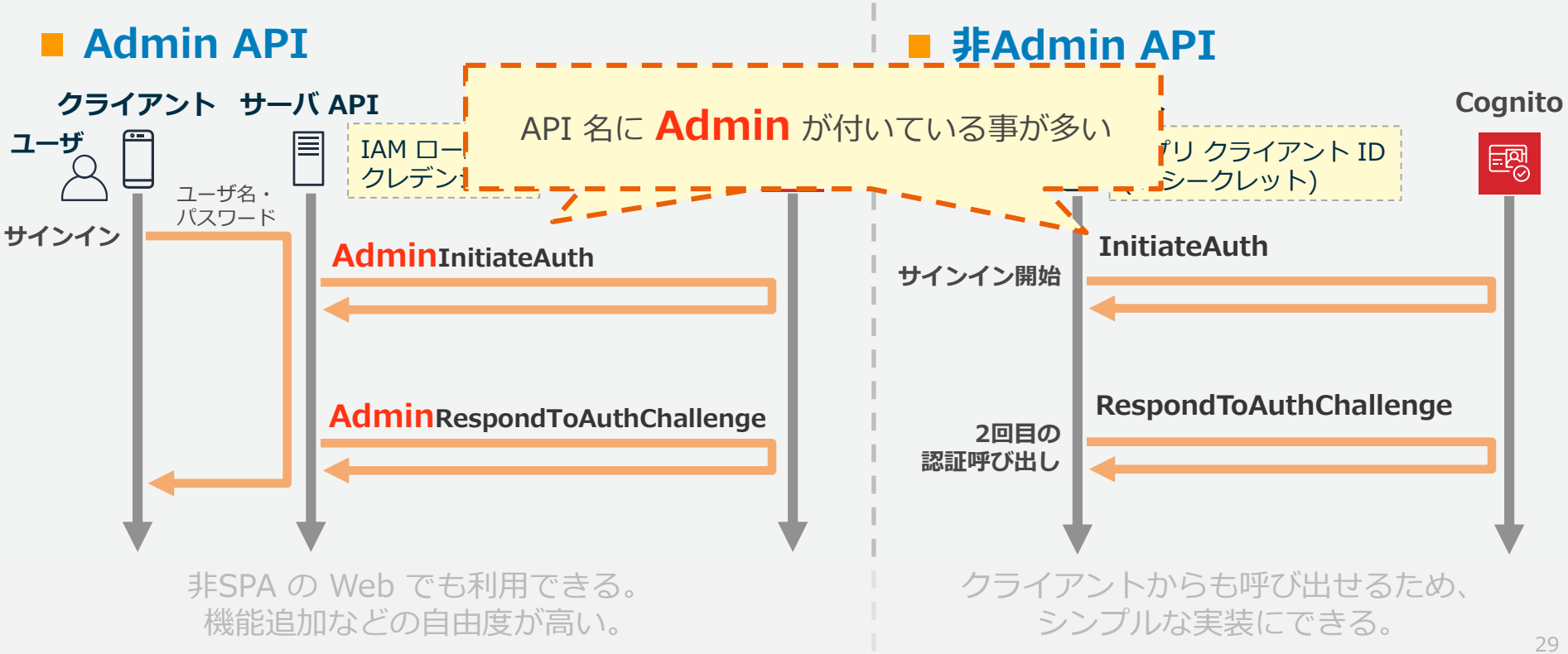
非SPA の Web でも利用できる。
機能追加などの自由度が高い。

■ 非Admin API

クライアントからも呼び出せるため、
シンプルな実装にできる。

❶ 呼び出し元によって異なる API あり

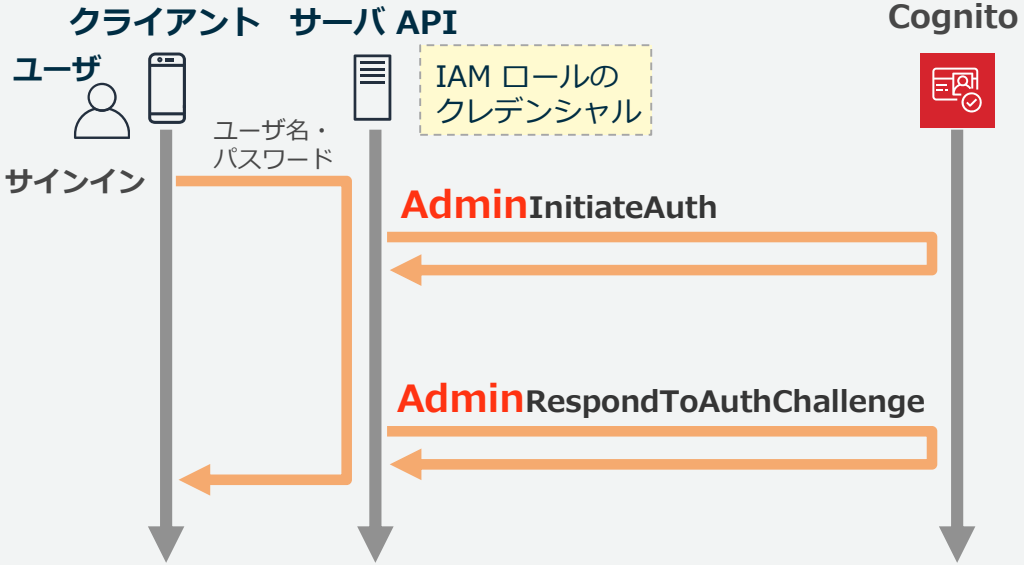
管理者としてサーバ側から呼び出す前提の Admin API と、
ユーザとしてサーバ側からもクライアン側からも呼び出せる API がある。



❶ 呼び出し元によって異なる API あり

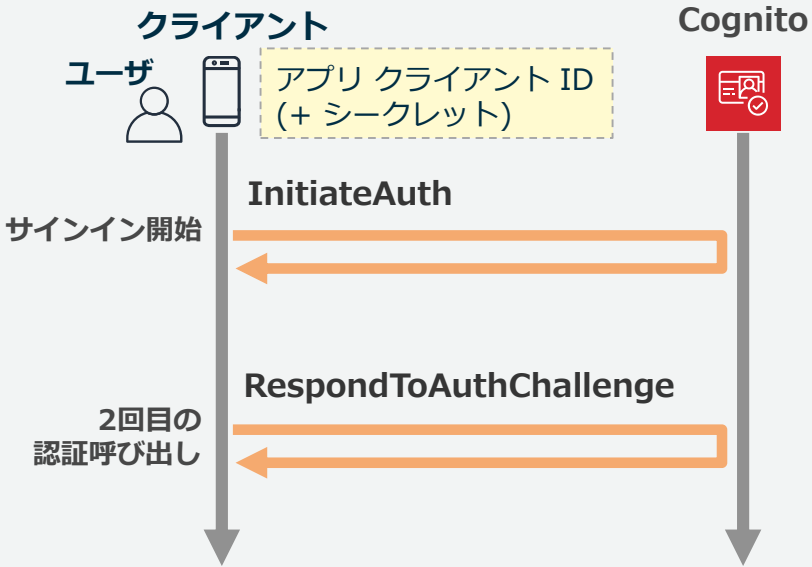
管理者としてサーバ側から呼び出す前提の Admin API と、
ユーザとしてサーバ側からもクライアン側からも呼び出せる API がある。

■ Admin API



非SPA の Web でも利用できる。
機能追加などの自由度が高い。

■ 非Admin API



クライアントからも呼び出せるため、
シンプルな実装にできる。

❶ サインインは認証フローを指定して API を呼び出す

アプリ クライアントは以下の認証フローのうち許可されているフローを利用する。
フローによってはサインインに複数回の呼び出しが必要となる。

種類	説明	API 呼び出し回数	Admin API からの 呼び出し可否	非 Admin API からの 呼び出し可否
USER_SRP_AUTH	SRPプロトコルに基づいた方法で、パスワードを基にしたチャレンジレスポンスで認証する。サーバ側でも元のパスワードは不明な状態になるなど**安全な方法**。	MFAなし: 2回 MFAあり: 3回	○	○
CUSTOM_AUTH	認証時に Lambda ファンクションがトリガーされ、ファンクション次第で追加の認証を行って認証などができる。	1回以上	○	○
USER_PASSWORD_AUTH	SRPプロトコルを使用せず、パスワード自体を送って認証する。クライアントからは**USER_SRP_AUTH の利用を推奨。**	MFAなし: 1回 MFAあり: 2回	○	○
ADMIN_USER_PASSWORD_AUTH	**USER_PASSWORD_AUTH** と似ているがサーバ用 API からのみ呼び出せる。	MFAなし: 1回 MFAあり: 2回	○	✗
REFRESH_TOKEN_AUTH	更新トークンから新しいトークンをもらう。	1回	○	○

❶ サインインは認証フローを指定して API を呼び出す

アプリ クライアントは以下の認証フローのうち許可されているフローを利用する。
フローによってはサインインに複数回の呼び出しが必要となる。

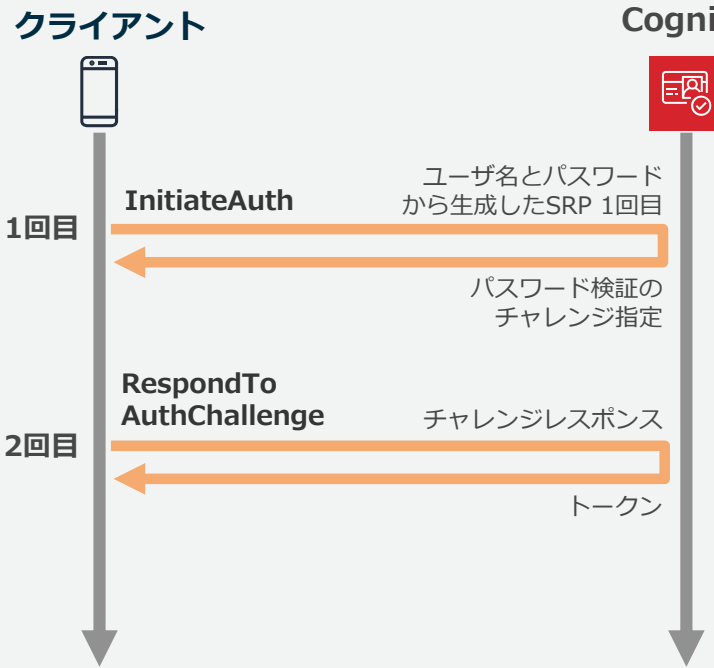
種類	説明	API 呼び出し回数	Admin API からの 呼び出し可否	非 Admin API からの 呼び出し可否
USER_SRP_AUTH	SRPプロトコルに基づいた方法で、パスワードを基にしたチャレンジレスポンスで認証する。サーバ側でも元のパスワードは不明な状態になるなど**安全な方法**。	MFAなし: 2回 MFAあり: 3回	○	○
CUSTOM_AUTH	認証時に Lambda ファンクションがトリガーされ、ファンクション次第で追加の認証を行って認証などができる。	1回以上	○	○
USER_PASSWORD_AUTH	SRPプロトコルを使用せず、パスワード自体を送って認証する。クライアントからは **USER_SRP_AUTH の利用を推奨。**	MFAなし: 1回 MFAあり: 2回	○	○
ADMIN_USER_PASSWORD_AUTH	**USER_PASSWORD_AUTH** と似ているがサーバ用 API からのみ呼び出せる。	MFAなし: 1回 MFAあり: 2回	○	X
REFRESH_TOKEN_AUTH	更新トークンから新しいトークンをもらう。	1回	○	○

❶ USER_SRP_AUTH – SRP プロトコルに基づいて認証

認証フローを指定して InitiateAuth を呼び出す。レスポンスで次に送るべきチャレンジが返ってくる。

```
cognito.initiateAuth({
  AuthFlow: 'USER_SRP_AUTH'
  ClientId: 'xxxxxxxxxxxxxxxx',
  AuthParameters: {
    CHALLENGE_NAME: ‘SRP_A’ ,
    SRP_A: 'xxxxxxxxxxxxxxx',
    USERNAME: 'user001',
})
```

```
cognito.respondToAuthChallenge({
  ChallengeName: ‘PASSWORD_VERIFIER’,
  ClientId: 'xxxxxxxxxxxxxxxx',
  ChallengeResponses: {
    TIMESTAMP: ‘xxxxxxxxx’,
    USERNAME: ‘user001’,
    PASSWORD_CLAIM_SECRET_BLOCK: …,
    PASSWORD_CLAIM_SIGNATURE: …,
})
```




```
{
  ChallengeName: ‘PASSWORD_VERIFIER’,
  ChallengeParameters: {SALT: ……},
  ……
}
```

```
{
  AuthenticationResult: {
    AccessToken: ……,
    ExpiresIn: ……,
    TokenType: ‘Bearer’,
    RefreshToken: ……,
    IdToken: ……,
  },
  ……
}
```

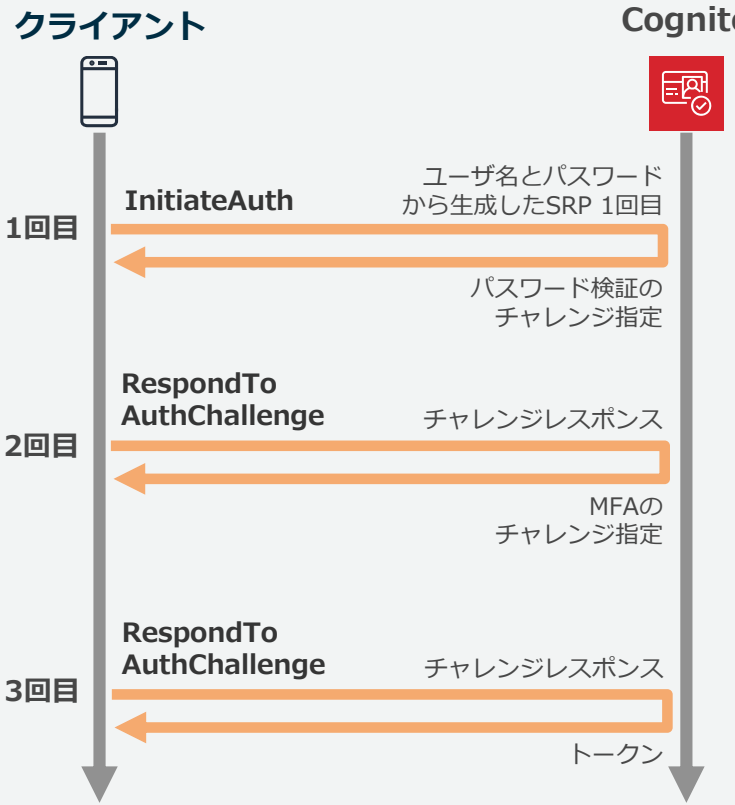
❶ USER_SRP_AUTH – SRP プロトコルに基づいて認証

ユーザに MFA が設定されていると、パスワード検証後にチャレンジとして指定される。

```
cognito.initiateAuth({
  AuthFlow: 'USER_SRP_AUTH'
  ClientId: 'xxxxxxxxxxxxxxxx',
  AuthParameters: {
    CHALLENGE_NAME: 'SRP_A',
    SRP_A: 'xxxxxxxxxxxxxxx',
    USERNAME: 'user001',
})
```

```
cognito.respondToAuthChallenge({
  ChallengeName: 'PASSWORD_VERIFIER',
  ClientId: 'xxxxxxxxxxxxxxxx',
  ChallengeResponses: {
    TIMESTAMP: 'xxxxxxxxx',
    USERNAME: 'user001',
    PASSWORD_CLAIM_SECRET_BLOCK: …,
    PASSWORD_CLAIM_SIGNATURE: …,
})
```

```
cognito.respondToAuthChallenge({
  ChallengeName: 'SOFTWARE_TOKEN_MFA',
  ClientId: 'xxxxxxxxxxxxxxxx',
  ChallengeResponses: {
    USERNAME: 'user001',
    SOFTWARE_TOKEN_MFA_CODE: '123456'
})
```



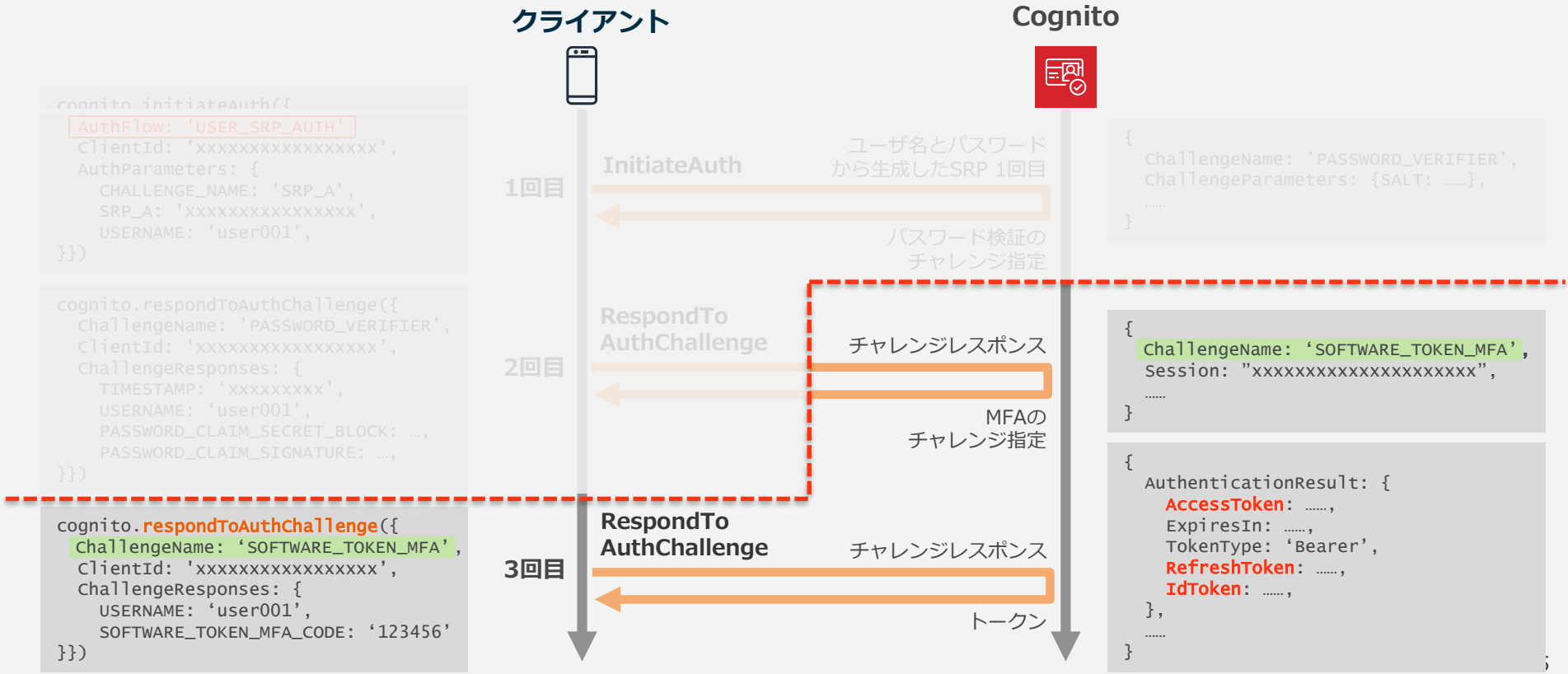

```
{
  ChallengeName: 'PASSWORD_VERIFIER',
  ChallengeParameters: {SALT: ……},
  ……
}
```

```
{
  ChallengeName: 'SOFTWARE_TOKEN_MFA',
  Session: "xxxxxxxxxxxxxxxxxxxx",
  ……
}
```

```
{
  AuthenticationResult: {
    AccessToken: ……,
    ExpiresIn: ……,
    TokenType: 'Bearer',
    RefreshToken: ……,
    IdToken: ……,
  },
  ……
}
```

❶ USER_SRP_AUTH – SRP プロトコルに基づいて認証

ユーザに MFA が設定されていると、パスワード検証後にチャレンジとして指定される。



❶ サインインは認証フローを指定して API を呼び出す

サインインには複数回の API 呼び出しが必要な場合があり、以下の認証フローに沿って呼び出す。どの認証フローを許可するかはアプリ クライアントごとに指定する。

種類	説明	API 呼び出し回数	Admin API からの呼び出し可否	非 Admin API からの呼び出し可否
USER_SRP_AUTH	SRPプロトコルに基づいた方法で、パスワードを基にしたチャレンジレスポンスで認証する。サーバ側でも元のパスワードは不明な状態になるなど**安全な方法**。	MFAなし: 2回 MFAあり: 3回	○	○
CUSTOM_AUTH	認証時に Lambda ファンクションがトリガーされ、ファンクション次第で追加の認証を行って認証などができる。	1回以上	○	○
USER_PASSWORD_AUTH	SRPプロトコルを使用せず、パスワード自体を送って認証する。クライアントからは**USER_SRP_AUTH の利用を推奨。**	MFAなし: 1回 MFAあり: 2回	○	○
ADMIN_USER_PASSWORD_AUTH	**USER_PASSWORD_AUTH** と似ているがサーバ用 API からのみ呼び出せる。	MFAなし: 1回 MFAあり: 2回	○	X
REFRESH_TOKEN_AUTH	更新トークンから新しいトークンをもらう。	1回	○	○

❶ CUSTOM_AUTH – カスタム認証

カスタム認証の場合は Lambda ファンクションで、次のチャレンジの指定や
チャレンジレスポンスの検証を行うことで、カスタマイズができる。

- 実現できる内容の例
  - Cognito とは別方式の MFA 認証を行う
  - 月 1 回は 秘密の質問を追加で確認する
  - 人からのアクセスである事を検証するため CAPTCHA を確認する
  - パスワードレス認証を行う

- 3 種類の Lambda ファンクションのトリガーで実現する


DefineAuthChallenge 認証チャレンジの定義


CreateAuthChallenge 認証チャレンジの作成


VerifyAuthChallengeResponse 認証チャレンジレスポンスの確認

1 CUSTOM_AUTH – カスタム認証

例：SRP パスワード認証に加え、追加のワンタイムパスワード入力を求める場合

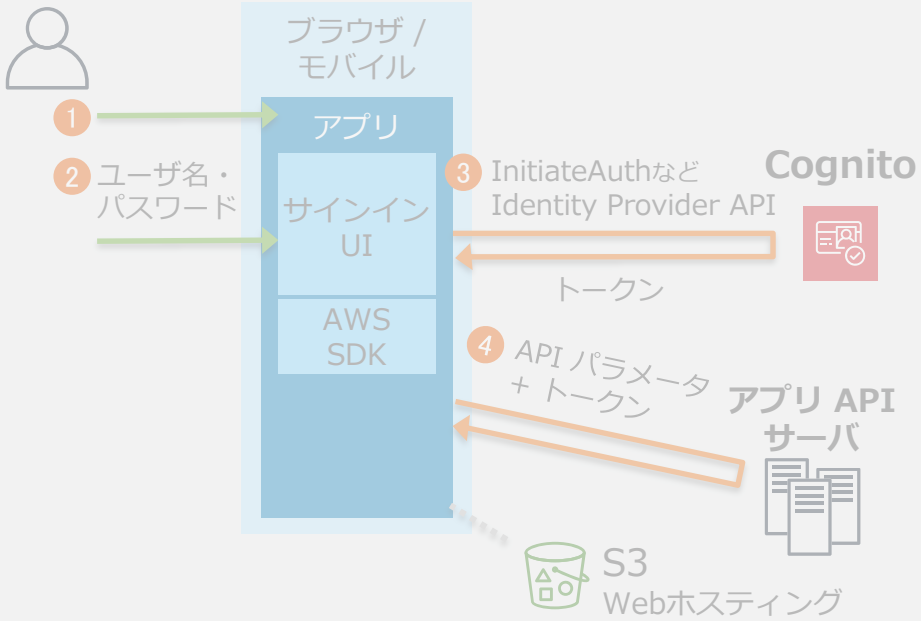
❶ サインインは認証フローを指定して API を呼び出す

サインインには複数回の API 呼び出しが必要な場合があり、以下の認証フローに沿って呼び出す。どの認証フローを許可するかはアプリ クライアントごとに指定する。

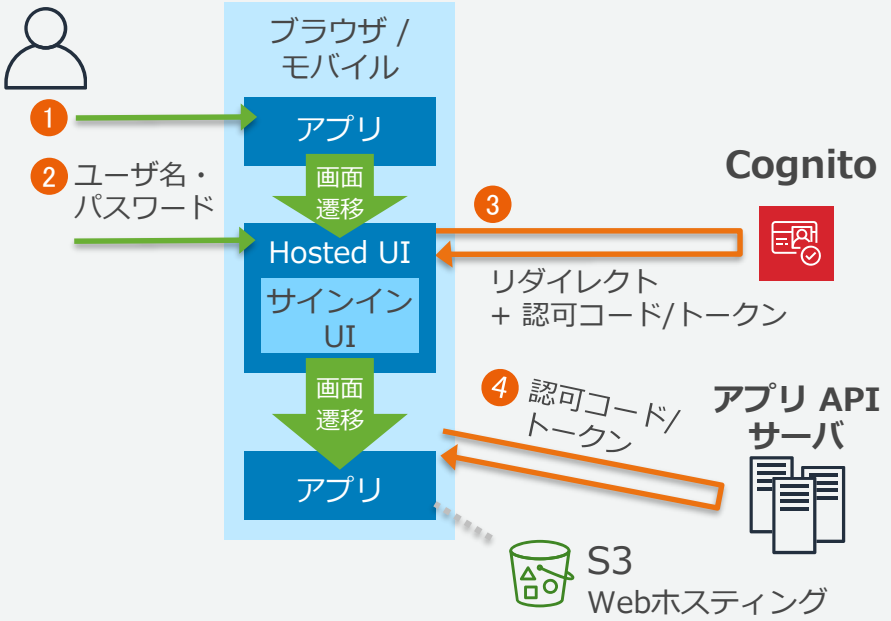
種類	説明	API 呼び出し回数	Admin API からの 呼び出し可否	非 Admin API からの 呼び出し可否
USER_SRP_AUTH	SRPプロトコルに基づいた方法で、パスワードを基にしたチャレンジレスポンスで認証する。サーバ側でも元のパスワードは不明な状態になるなど**安全な方法**。	MFAなし: 2回 MFAあり: 3回	○	○
CUSTOM_AUTH	認証時に Lambda ファンクションがトリガーされ、ファンクション次第で追加の認証を行って認証などができる。	1回以上	○	○
USER_PASSWORD_AUTH	SRPプロトコルを使用せず、パスワード自体を送って認証する。クライアントからは **USER_SRP_AUTH の利用を推奨。**	MFAなし: 1回 MFAあり: 2回	○	○
ADMIN_USER_PASSWORD_AUTH	**USER_PASSWORD_AUTH** と似ているがサーバ用 API からのみ呼び出せる。	MFAなし: 1回 MFAあり: 2回	○	X
REFRESH_TOKEN_AUTH	更新トークンから新しい ID トークン、アクセストークンをもらう。	1回	○	○

利用できる API は 2 セット

1 Cognito Identity Provider API を利用

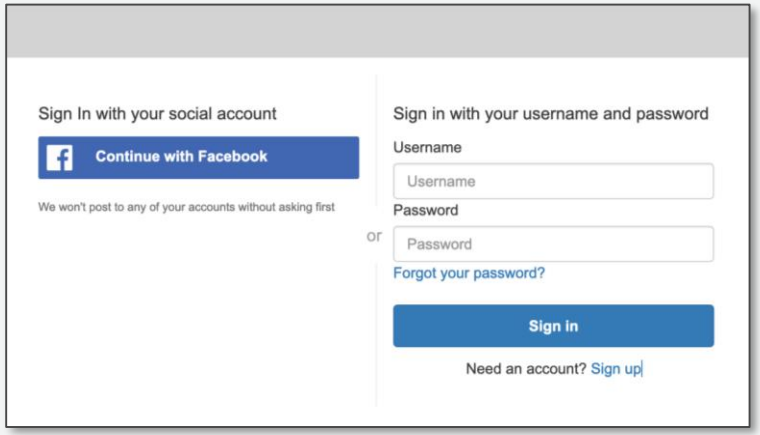


2 Cognito Auth API と Hosted UI を利用



❷ Cognito Auth API と Hosted UI を利用

Cognito がマネージドサービスの一部として、サインアップ、サインイン、パスワード忘れの対応が行える Web の UI を提供する。

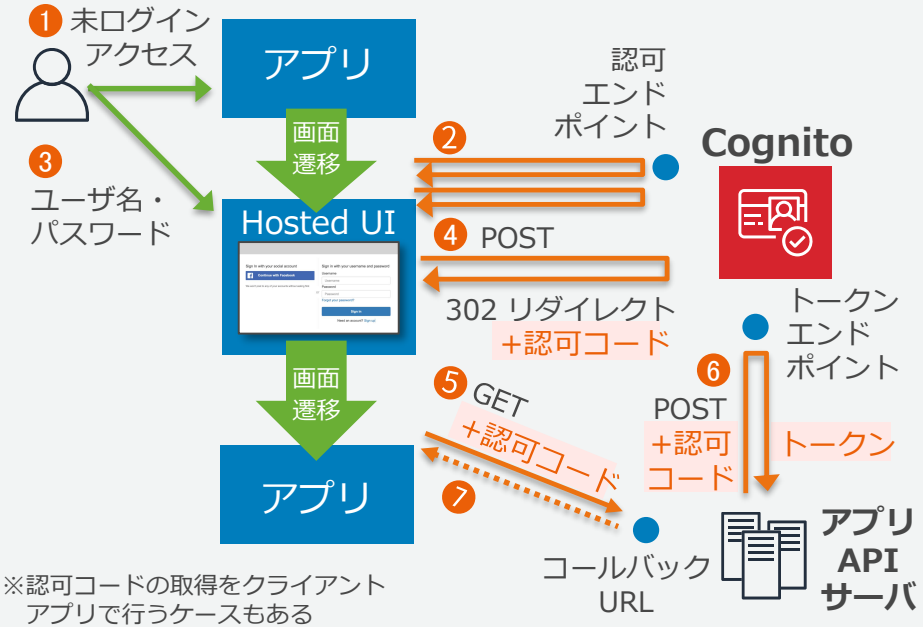


- 多要素認証（MFA）に対応。
  カスタム認証フローのような仕組みはなし。
- ドメイン名、UI のロゴ、CSS のカスタマイズが可能だが、日本語対応は不可。
- 外部 ID プロバイダーを使ったサインインには Auth API が必須。
- OAuth を使ったサードパーティアプリへの認可には Auth API と Hosted UI が必須。

❷ 認証は OAuth フローに沿って

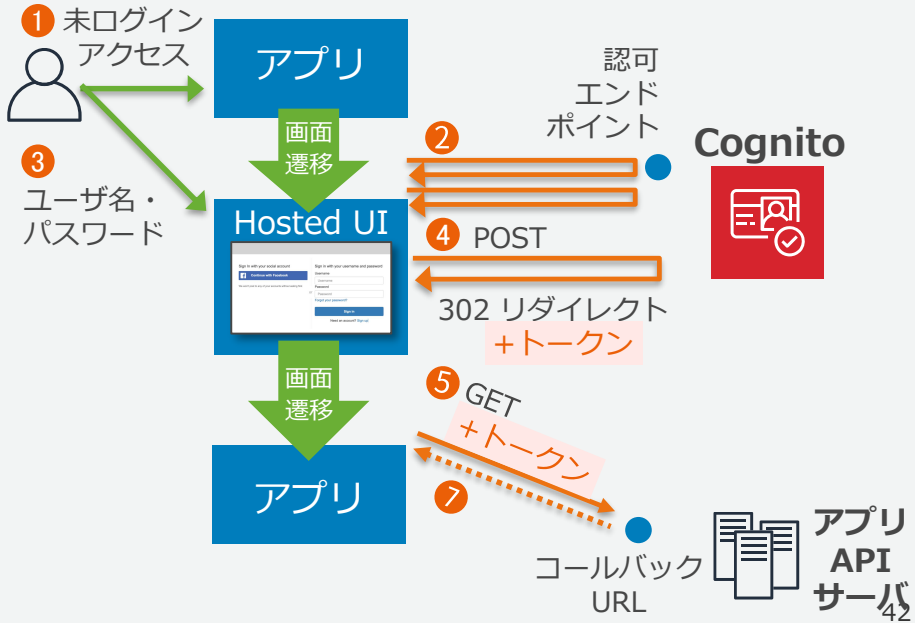
認証は OAuth 2.0 で定義されているフローに沿って行い、認証後アプリには**認可コード**あるいは**トークン**を URL パラメータなどに付けてリダイレクトされる。

■ Authorization Code Grant
(認可コード許可)



※認可コードの取得をクライアントアプリで行うケースもある

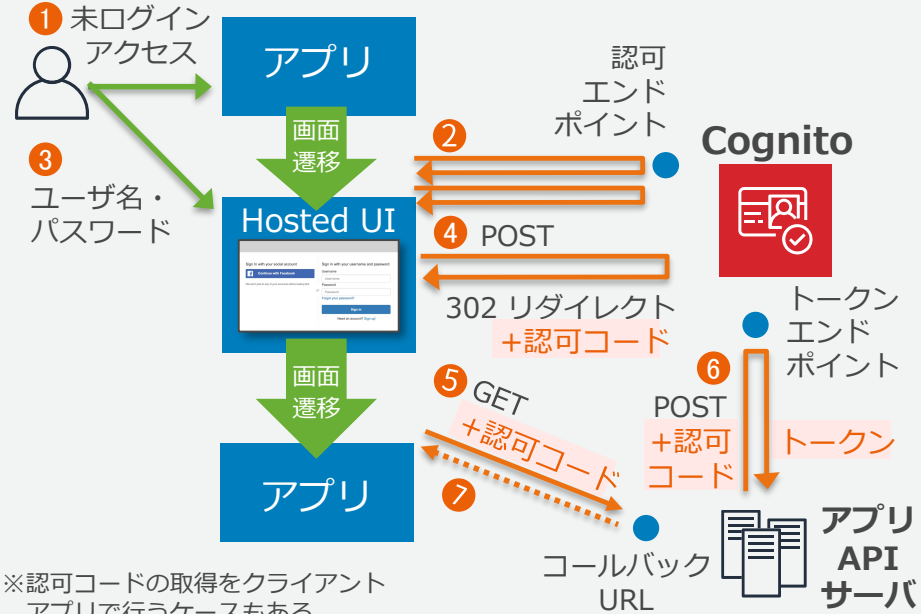
■ Implicit Grant
(暗黙的許可)



❷ 認証は OAuth フローに沿って

認証は OAuth 2.0 で定義されているフローに沿って行い、認証後アプリには**認可コード**あるいは**トークン**を URL パラメータなどに付けてリダイレクトされる。

■ Authorization Code Grant
(認可コード許可)



※認可コードの取得をクライアントアプリで行うケースもある

■ Implicit Grant
(暗黙的許可)

Authorization Code Grant の方がセキュアなため、極力 Authorization Code Grant の利用を推奨。

❷ エンドポイントへのアクセス

■ Authorization Code Grant

認可エンドポイント

GET https://<指定したプレフィックス>.auth.
{region}.amazoncognito.com/oauth2/authorize?
response_type=**code** ient_id=<client id>&
redirect_uri=<callback url>&state=XXXXXX&
code_challenge_method=S256&code_challenge=XXXXXXXXX
XXXXXXXXXXXXXXXXX

認証後にリダイレクトされる URL

GET https://<callback url>?
code=XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX
state=XXXXXX

認可コード

トークンエンドポイント

POST https://<指定したプレフィックス>.auth.
{region}.amazoncognito.com/oauth2/token

```
grant_type: authorization_code
client_id: <client id>
code: XXXXXXXXXXXXXXXXXXXXXXXXX
redirect_url: <callback url>
code_verifier: XXXXXXXXXXXXXXXX
```

■ Implicit Grant

認可エンドポイント

GET https://<指定したプレフィックス>.auth.
{region}.amazoncognito.com/oauth2/authorize?
response_type=**token** ent_id=<client id>&
redirect_uri=<callback url>&state=XXXXXX&
code_challenge_method=S256&code_challenge=XXXXXXXXX
XXXXXXXXXXXXXXXXX

認証後にリダイレクトされる URL

GET https://<callback url>#
id_token=XXXXX.....XXXXXXX &
access_token=XXXXX.....XXXXXXX
expires_in=3600&token_type=Bearer&
state=XXXXXX

トークン

❷ エンドポイントへのアクセス

■ Authorization Code Grant

認可エンドポイント

GET https://<指定したプレフィック
{region}.amazoncognito.com/oauth2
response_type=code&client_id=<cli
redirect_uri=**<callback url>**
code_challenge_method=S256&code_challenge=XXXXXXXXX
XXXXXXXXXXXXXXXXXX

指定したURLに
リダイレクトされる

認証後にリダイレクトされる URL

GET https:// **<ca <callback url>**
code=XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX &
state=XXXXXX

トークンエンドポイント

POST https://<指定したプレフィックス>.auth.
{region}.amazoncognito.com/oauth2/token

```
grant_type: authorization_code
client_id: <client id>
code: XXXXXXXXXXXXXXXXXXXXXXXXXX
redirect_url: <callback url>
code_verifier: XXXXXXXXXXXXXXXX
```

■ Implicit Grant

認可エンドポイント

GET https://<指定したプレフィック
{region}.amazoncognito.com/oauth2
response_type=token&client_id=<cl
redirect_uri=**<callback url>**
code_challenge_method=S256&code_challenge=XXXXXXXXX
XXXXXXXXXXXXXXXXXX

指定したURLに
リダイレクトされる

認証後にリダイレクトされる URL

GET https:// **<ca <callback url>**
id_token=XXXXX.....XXXXXXX &
access_token=XXXXX.....XXXXXXX &
expires_in=3600&token_type=Bearer&
state=XXXXXX

アプリ クライアントに設定したした
コールバック URL とも完全一致する必要がある。

❷ エンドポイントへのアクセス

■ Authorization Code Grant

認可エンドポイント

GET https://<指定したプレフィックス>.auth.
{region}.amazoncognito.com/oauth
response_type=code&client_id=<cl
redirect_uri=<callback url>& state=XX **state=XXXXXX**
code_challenge_method=S256&code_challenge=XXXXXXXXX
XXXXXXXXXXXXXXXXXX

呼び出し時に付加

認証後にリダイレクトされる URL

GET https://**<callb**
code=XXXXXXXX
state=XXXXXX

同じ値を付けて
リダイレクトされてきた事を確認

state パラメータ
CSRF 対策

自ら呼び出してから、認可エンドポイントから
リダイレクトされて戻ってきたのかを確認する。

■ Implicit Grant

認可エンドポイント

GET https://<指定したプレフィックス>.auth.
{region}.amazoncognito.com/oau
response_type=token&client_id=
redirect_uri=<callback url>& state=XX **state=XXXXXX**
code_challenge_method=S256&code_challenge=XXXXXXXXX
XXXXXXXXXXXXXXXXXX

呼び出し時に付加

認証後にリダイレクトされる URL

GET https://**<callback url>**#
id_token=XXXXX.....XXXXXXX &
access_token=XXXXX.....XXXXXXX &
expires_in=3600&token_type=Bearer&
state=XXXXXX

同じ値を付けて
リダイレクトされてきた事を確認

❷ エンドポイントへのアクセス

■ Authorization Code Grant

認可エンドポイント

GET https://<指定したプレフィックス>.auth.
{region}.amazoncognito.com/oauth2/authorize?
response_type=code&client_id=<client id>&
redirect_uri=<callback url>&state=XXXXXX &
code_challenge_method=S256&code_challenge=XXX
XXXXXXXXXXXXXXXXXXXXXXXX

認証後にリダイレクトされ

GET https://**<callback url>**?
code=XXXXXXXX-XXXX-XXXX-XXXX-
state=XXXXXX

呼び出し側で
決めた任意の値
code_verifier
のハッシュ値を
呼び出し時に付加

トークンエンドポイント

POST https://<指定したプレフィックス>.auth.
{region}.amazoncognito.com/oauth2

grant_type: authorization_code
client_id: <client id>
code: XXXXXXXXXXXXXXXXXXXXXXXX
redirect_url: <callback url>
code_verifier: XXXXXXXXXXXXXXXX

呼び出し側で
決めた元の値を
パラメータに

■ Implicit Grant

認可エンドポイント

GET https://<指定したプレフィックス>.auth.
{region}.amazoncognito.com/oauth2/authorize?
response_type=token&client_id=<client id>&
redirect_uri=<callback url>&state=XXXXXX&
code_challenge_method=S256&code_challenge=XXXXXXXXX

PKCE (RFC7636)
認可コードの横取り対策

認可コードを知った悪意のある第三者がシークレットを設定されていないアプリクライアントの振りをして、認可コードからトークンを取得するのを防ぐ。

❷ エンドポイントへのアクセス

■ Authorization Code Grant

認可エンドポイント

GET https://<指定したプレフィックス>.auth.
{region}.amazoncognito.com/oauth2/authorize?
response_type=code&client_id=<client id>&
redirect_uri=<callback url>&state=XXXXXX&
code_challenge_method=S256&code_challenge=XXXXXXXXX
XXXXXXXXXXXXXXXXXX

&identity_provider=Facebook

認証後にリダイレクトされる URL

GET https://
code=XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX &
state=XXXXXX

トークンエンドポイント

POST https://<指定したプレフィックス>.au
{region}.amazoncognito.com/oauth2/token

grant_type: authorization_code
client_id: <client id>
code: XXXXXXXXXXXXXXXXXXXXXXXXX
redirect_url: <callback url>
code_verifier: XXXXXXXXXXXXXXXX

■ Implicit Grant

認可エンドポイント

GET https://<指定したプレフィックス>.auth.
{region}.amazoncognito.com/oauth2/authorize?
response_type=code&client_id=<client id>&
redirect_uri=<callback url>&state=XXXXXX&
code_challenge_method=S256&code_challenge=XXXXXXXXX
XXXXXXXXXXXXXXXXXX

&identity_provider=Facebook

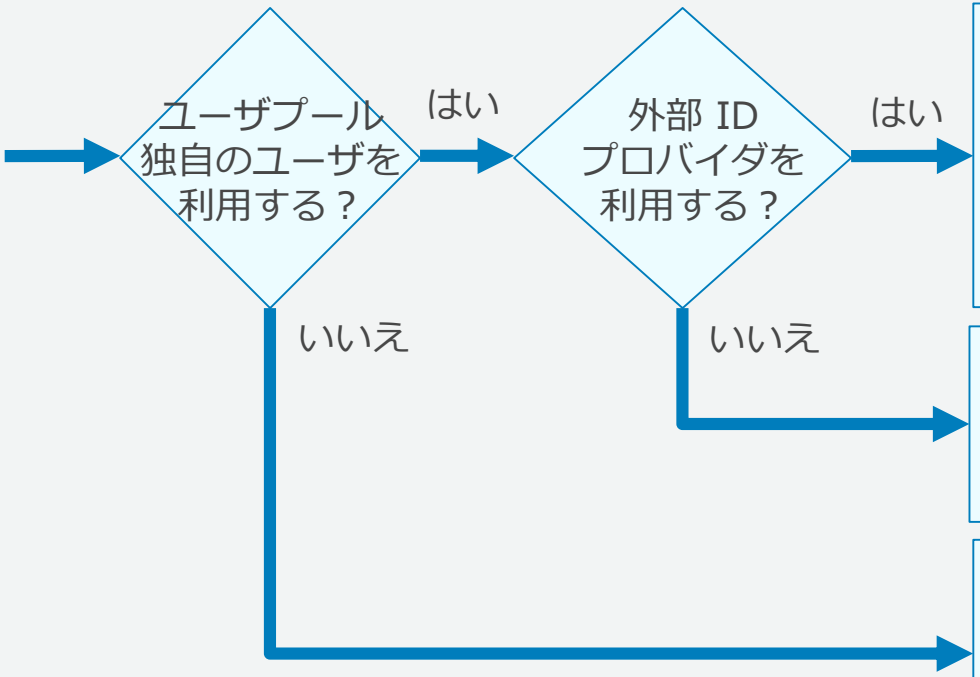
認証後にリダイレクトされる URL

GET https://
id_token=XXXXX.....XXXXXXX&

ID プロバイダを明示指定することで、
Hosted UI を表示させずに直ぐに
外部 ID プロバイダのログイン UI に
リダイレクトさせれる。

要件に応じた API 選択

日本語対応を前提に考えると、以下のように実装する事がおすすめ。



1 Cognito Identity Provider API &
2 Cognito Auth API

サインインの UI をアプリで用意する。その UI にはユーザ名・パスワードの入力フォームに加え、外部 ID プロバイダにサインインするための認可エンドポイントへのリンクの載せる。

1 Cognito Identity Provider API

ユーザ名・パスワードの入力フォームのある UI をアプリで用意する。

2 Cognito Auth API

外部 Id プロバイダにサインインするための認可エンドポイントへのリンクのみを載せた UI を用意する。

実装には SDK や Library を利用する

API 呼び出しを直接実装する必要はなく、SDK や Library を使って実装できる。

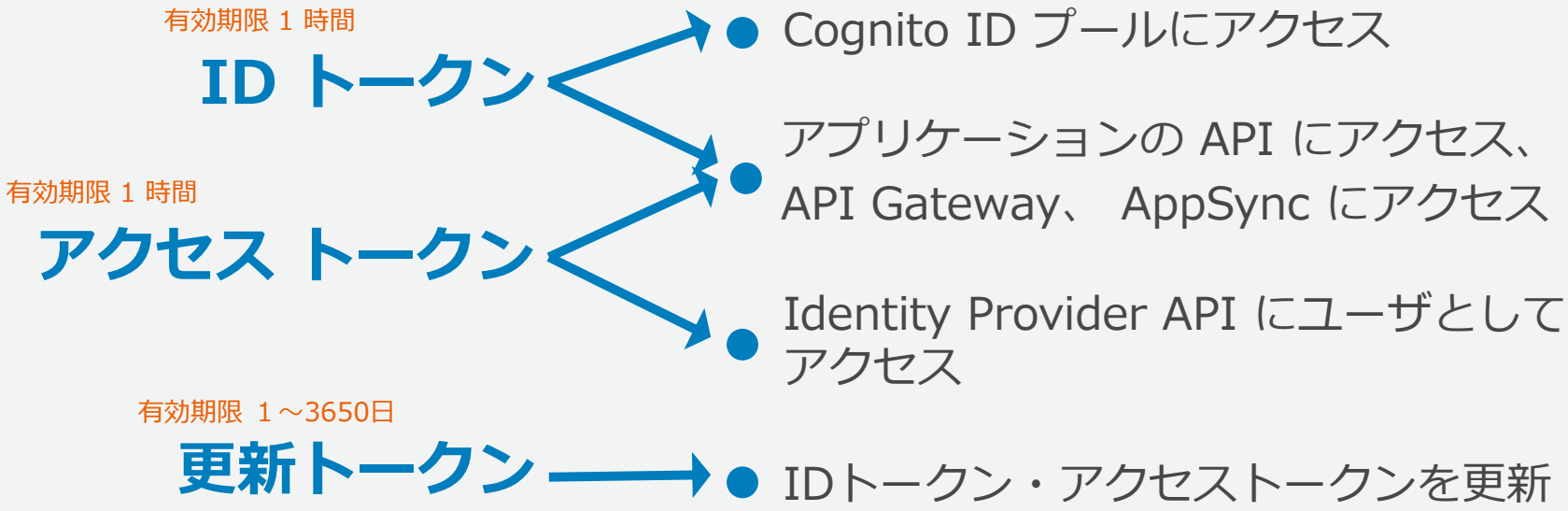
SDK	動作環境	概要	①Identity Provider API を利用	②Auth API とHosted UI を利用	Identity Provider API を利用する SDK 独自の サインイン UI
AWS SDK for JavaScript などの 各種言語向け SDK	サーバやブラウザ	低レベルな ライブラリ	○	ー	ー
Amazon Cognito Identity SDK for JavaScript	ブラウザ	ブラウザ向けの ライブラリ	○ 非Admin API	ー	ー
AWS Mobile SDK	iOS, Android	モバイル向け ライブラリ	○ 非Admin API	△ 日本語化不可	△ 日本語化不可
AWS Amplify Library	ブラウザ (JavaScript, React, Angular, Vue)、 モバイル (Android, iOS, React Native, Ionic)	アプリ開発者が 直感的に利用できる 宣言型インターフェース のライブラリ Mobile SDK や Cognito Identity SDK も内部で使用している。	○	△ 日本語化不可 動作環境によって対応状況に差異あり	△ Android, iOS以外は i18nモジュールで日本語化可能 動作環境によって対応状況に差異あり

ユーザ
プール

サインイン後の実装

トークンの種類と用途

サインイン時に発行されるトークンは 3 種類。含まれている情報が異なるため、用途が異なっている。サインインの方法によっては更新トークンは発行されない。



サインイン後に発行されるトークン

各トークンは OAuth / OIDC と同様の JWT 形式となっている。

エンコードされた ID トークン

eyJraWQiOiJhTFZyKzBXUnB6Q2FIUzhYSFdSWF
wveDZ1ZEhKRFlrSStcL1NwNmJwaG4rRjg9Iiwi
YWxnIjoiUlMyNTYifQ.eyJzdWIiOiIwNWExNzF
lOS0xOWFkLTRlYWMtYWY2MC1lZWIyMTg3MDAxO
DkiLCJhdWQiOiJlcmtqczN0NzQ4YnBwaGZmanB
ia2RucTZiIiwiZW1haWxfdmVyaWZpZWQiOnRyd
WUsImV2ZW50X2lkIjoiNWNlODRhOWItODA1NS0
0YmFiLTgzN2ItNWYxMWFmOTE4Zjc2IiwidG9rZ
W5fdXNlIjoiaWQiLCJhdXRoX3RpbWUiOjE1OTI
yNDM4NDcsImlzcyI6Imh0dHBzOlwvXC9jb2dua
XRvLWlkcC5hcC1ub3J0aGVhc3QtMS5hbWF6b25
hd3MuY29tXC9hcC1ub3J0aGVhc3QtMV8zbnoxW
jdNbHQiLCJjb2duaXRvOnVzZXJuYW1lIjoiQ29
nbml0b1VzZXJQb29sVXNlclRlc3RVc2VyLURaU
Etabm5sZnR3YyIsImV4cCI6MTU5MjI0NzQ0Nyw
iaWF0IjoxNTkyMjQzODQ3LCJlbWFpbCI6InRzd
WppeSt1c2VyMDAxQGFtYXpvbi5jby5qcCJ9.Uj
iyt4nCVxovpyH4vs0xybpUD07rpzTR8OzHWsjr
ckuP67bc975vnuJX_Tb9yJW-Gd1eQLCcp8Sra8
2LxV6h12QVGTicZPm6qRHRFPMM55AE9P1SASjP
86IdZG5wspiPXgg4pnq4UJIHQlfU-v8TeG0-7X
Qf5whqcxlIRfwx7Q_9keyuDp6V_617g_FxVsYL
MyPcyJEDpDfRPCUO_KgcveHt3G47uGM8qMTZaP
udqgZSsqoCXsnr1pP_dWMVyAq5urkJQ6lb1EQB
jl2fOh8hC7aaZg1sY_FxPb_sZghzv_Px4OCrt1
WuyAvajsLIDlAYwctF_TXYWX-m864eUElcVw

デコード後の ID トークン

- **ヘッダ**


```
{
  "kid":"aLVr+0WRpzCaHS8XHWRX¥/x6udHJDYkI.....",
  "alg":"RS256"
}
```


- **ペイロード**


```
{
  "sub":"c29641fa-0b5b-475b-a588-d9be34f3534d",
  "cognito:groups":["test001"],
  "email_verified":true,
  "iss":"https://cognito-idp.ap-northeast-1.....",
  "cognito:username":"XXXXXXXXXXXXXXXXXXXXXXXXXX",
  "given_name":"Yoshikazu",
  "aud":"3ff54d7l89o1atsub76hnp7br3",
  "event_id":"05b18935-2f15-426b-8d73-58f2082...",
  "token_use":"id",
  "custom:org":"Making History Company",
  "auth_time":1592810488,
  "exp":1592814088,
  "iat":1592810488,
  "email":xxxxxxxxxxxxx
}
```


- **署名**

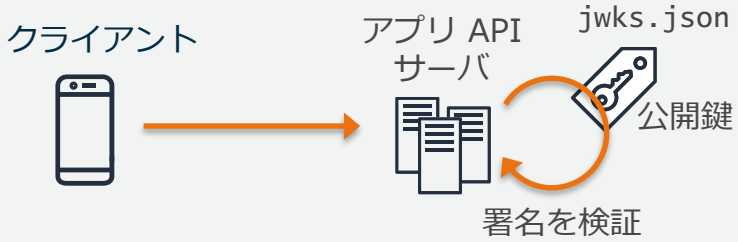
ヘッダとペイロードを RSA-SHA256 で署名

- ID トークンについては、カスタム属性含むユーザの属性が含まれる。
- アクセストークンには一部のユーザ情報に加えスコープなどが含まれている。
- 署名は以下の公開鍵で検証できる。

```
https://cognito-idp.{region}.amazonaws.com/
{userPoolId}/.well-known/jwks.json
```

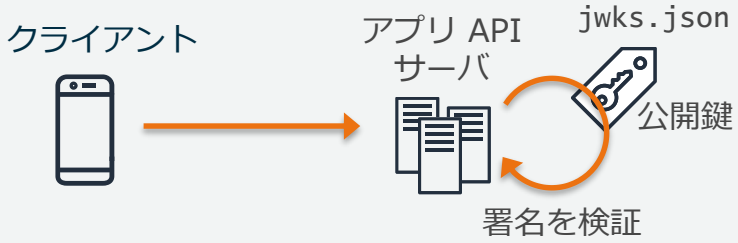
トークンをアプリケーションの API へのアクセスに利用

サーバでトークンを検証

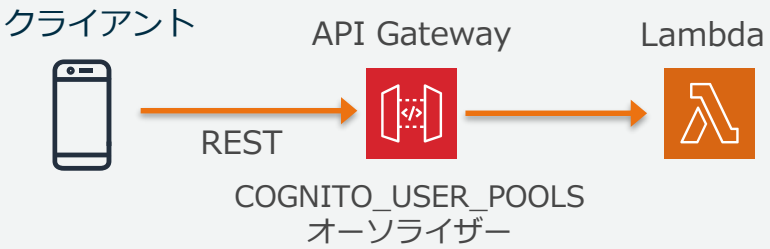


トークンをアプリケーションの API へのアクセスに利用

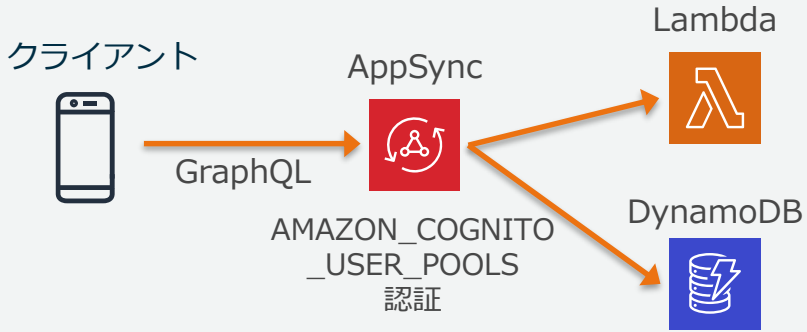
サーバでトークンを検証



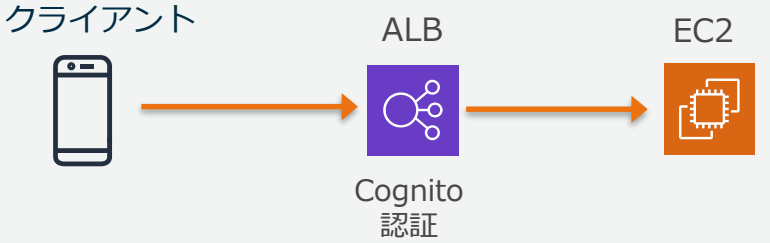
API Gateway 経由でアクセス



AppSync 経由でアクセス

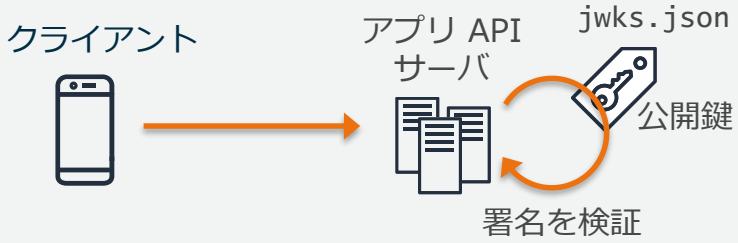


ALB 経由でアクセス

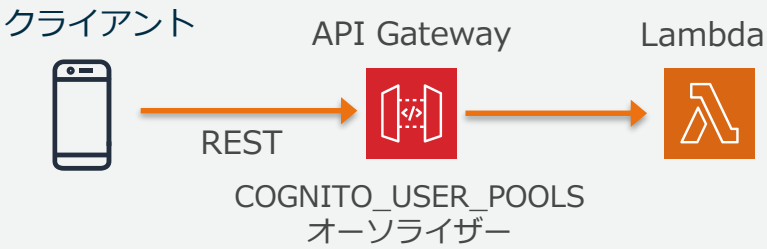


トークンをアプリケーションの API アクセスに利用

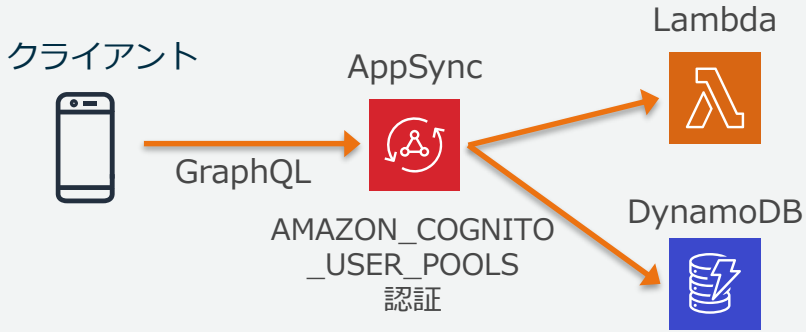
サーバでトークンを検証



API Gateway 経由でアクセス



AppSync 経由でアクセス



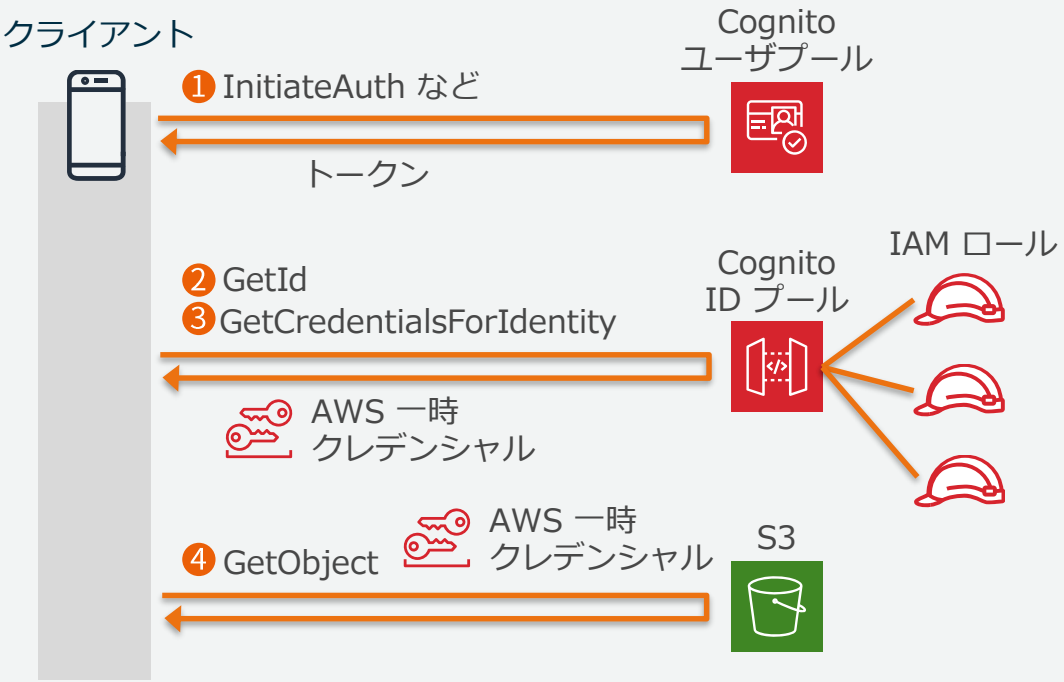
ALB 経由でアクセス

Cognito Auth API & Hosted UI 前提

API ベースでは無い Web アプリへのアクセスに
Cognito のサインインを条件にできる。
ただし、Hosted UIの日本語対応不可に注意。

トークンを AWS の API アクセスに利用

ユーザプールで発行された ID トークンを Cognito ID プールで、AWS の一時クレデンシャルを発行できる。



- クレデンシャルが発行される IAM ロールを指定できる。
  - ID プールで指定した認証ずみユーザ用のロール
  - ユーザプールのグループに基づくロール
  - ルールで指定されたロール
- クレデンシャルには cognito-identity.amazonaws.com:sub などの変数が設定されており、S3 などのアクセス条件に使用できる。

サインアウト

共用端末などで、ユーザ本人以外が同じクライアントを使ってアプリケーションを利用することを防ぐために、サインアウトを実装する。
どのようにトークンを使っているかによって、行うべき対処が異なる。

- 共通

トークンをアプリや SDK で クライアントから破棄

アクセストークン・ID トークン自体は 最大 1 時間の有効期限内は利用可能

- Identity Provider API
  - Identity Provider API を ユーザとしての呼び出し (更新トークンの利用を含む)

Identity Provider API グローバルサインアウトで無効化

- Auth API
  - Hosted UI からの再サインイン

Auth API ログアウトエンドポイントで無効化

サインアウト

共用端末などで、ユーザ本人以外が同じクライアントを使ってアプリケーションを利用することを防ぐために、サインアウトを実装する。
どのようにトークンを使っているかによって、行うべき対処が異なる。

- 共通

トークンをアプリや SDK で クライアントから破棄

アクセストークン・ID トークン自体は 最大 1 時間の有効期限内は利用可能

- Identity Provider API
  - Identity Provider API を ユーザとしての呼び出し (更新トークンの利用を含む)
- Auth API
  - Hosted UI からの再サインイン

OAuth によるサードパーティ アプリへの API アクセス認可など、トークンの破棄が難しい場合は、リスクに応じて追加の対処を検討する。

ユーザプール

各種機能

主なユーザ操作

※IdP API = Identity Provider API

ユーザへ提供が可能な主な操作		実装方法		
		① IdP API Admin	① IdP API 非Admin	② Auth API & Hosted UI
サインアップ	ユーザ名、Email、電話番号を使用したサインアップ	(CreateUser)	SignUp	Hosted UI
Email あるいは電話番号の確認	アカウントを確認するための Email アドレスや電話番号の検証への対応	Admin ConfirmSignup	Confirm SignUp	Hosted UI
サインイン	ユーザ名、パスワード、MFA などを使ってサインインしてトークンの入手	Admin InitiateAuth	InitiateAuth	Hosted UI
パスワード忘れ対応	確認済みのメールアドレス、電話番号に確認コードを送り、確認できた場合にパスワードを変更 (AccountRecoverySetting優先順位をカスタマイズ可)	-	Forgot Password	Hosted UI
パスワード変更	新しいパスワードを設定	(Admin SetPassword)	Change Password	-
ユーザ属性変更	代替ユーザ名、メールアドレスや住所などの変更	Update UserAttributes	AdminUpdate UserAttributes	-
サインアウト	サインアウトし、再度サインインしないとアプリを使えないようにする	アプリや SDK でトークンを破棄		
		AdminUser GlobalSignOut	GlobalSignOut	Logout Endpoint を呼び出し

主なユーザ操作

※IdP API = Identity Provider API

ユーザへ提供が可能な主な操作

		実装方法		
		① IdP API Admin	① IdP API 非Admin	② Auth API & Hosted UI
サインアップ	ユーザ名、Email、電話番号を使用したサインアップ	(CreateUser)	SignUp	Hosted UI
Email あるいは電話番号の確認	アカウントを確認するための Email アドレスや電話番号の検証への対応	Admin ConfirmSignup	Confirm SignUp	Hosted UI
サインイン	ユーザ名、パスワード、MFA などを使ってサインインしてトークンの入手	Admin InitiateAuth	InitiateAuth	Hosted UI
パスワード忘れ対応	確認済みのメールアドレス、電話番号に確認コードを送り、確認できた場合にパスワードを変更 (AccountRecoverySetting優先順位をカスタマイズ可)	-	Forgot Password	Hosted UI
パスワード変更	新しいパスワードを設定	(Admin SetPassword)	Change Password	-
ユーザ属性変更	代替ユーザ名、メールアドレスや住所などの変	Update	AdminUpdate	
サインアウト	サインアウトし、再度サインインしないとアクセスできないようにする	AdminUser GlobalSignOut	GlobalSignOut	Logout Endpoint を呼び出し

パスワード変更のように Auth API だけでは実現できない操作あり

主な管理者操作

※IdP API = Identity Provider API

管理者が行える主な操作。サーバサイドで実装しユーザから行えるようにする事も考えられる。

		実装方法		
		① IdP API Admin	① IdP API 非Admin	② Auth API & Hosted UI
パスワードのリセット	基本的にはパスワードの忘れ対応と同じ、ユーザにて確認操作が必要	AdminReset UserPassword	-	-
パスワードの強制変更	パスワードを一時的な設定、あるいは恒久的な設定として変更	AdminSet UserPassword	-	-
ユーザを 外部 ID プロバイダとリンク	ユーザプール上の独自ユーザと外部 ID プロバイダのユーザをリンクし同じユーザとして扱えるようにする	Admin LinkProvider ForUser	-	-
ユーザの無効化	ユーザを無効化して使えないようする	Admin DisableUser	-	-
グループの作成	グループを作成し、IAM ロールや優先順位を割り当てる	CreateGroup	-	-
ユーザの検索	標準属性について完全一致、前方部分一致するユーザをリストできる	ListUsers	-	-
ユーザのインポート	CSV ファイルを使用してユーザをインポートする	CreateUser ImportJob	-	-

サインインに利用できる外部 ID プロバイダ

Auth API を使うと、外部 ID プロバイダでのサインインに基づいて、Cognito ユーザプールにサインインできる。ユーザにとっては、ユーザ登録や新たなパスワードの記憶などの手間を省くことができる。

- Facebook
- Google
- Login with Amazon
- Sign In with Apple
- SAML (SP-Initiated限定)
- OpenID Connect (OIDC)

SAML や OIDC を使うと数多くの ID プロバイダに対応できる

セキュリティの強化

様々なセキュリティを強化する機能が用意されている。

- 電話番号と Email の検証
- 多要素認証（MFA）の利用
  - SMS テキストメッセージ
  - TOTP ソフトウェアトークン
- クライアント デバイスの追跡
- アドバンスドセキュリティの機能
  - 侵害された認証情報が使われていないか確認
  - アダプティブ認証
    リスクに応じて MFA の利用を求めたり、ブロックしたりする

メッセージのカスタマイズ

メッセージを日本語にするなど、ユーザに送られるメッセージをカスタマイズが可能。

- 管理者が作成したユーザへの招待メッセージ
  - SMS: 本文
  - Email: サブジェクト、本文
- メールアドレス・電話番号の検証メッセージ
  - SMS: 本文
  - Email: サブジェクト、本文、検証方法
- SMS を使った MFA の認証メッセージ

Lambda トリガーを使った
メッセージのカスタマイズを使うと、
状況に応じて動的なメッセージを
送ること事が可能。

AWS Lambda – トリガーを用いたカスタマイズ

サインインフローを変更したり、エクスペリエンスを改善する事が可能。

カテゴリ	オペレーション	説明
カスタム認証フロー	認証チャレンジの定義	カスタム認証フロー内で次のチャレンジを決定する。
	認証チャレンジの作成	カスタム認証フロー内でチャレンジの作成する。
	認証チャレンジレスポンスの確認	カスタム認証フロー内でレスポンスが正しいか決定する。
サインイン	サインイン前	サインインの許可・拒否をカスタムで決定する。
	サインイン後	分析のためにイベントログを記録できる。
	トークン生成前	トークンのクレームを追加・更新・抑制する。
サインアップ	サインアップ前	サインアップの許可・拒否をカスタムで決定する。
	確認後	Welcomeメッセージなどのカスタマイズや分析用にイベントログを記録する。
	ユーザー移行	既存のユーザーディレクトリからユーザープールにユーザーを移行する。
メッセージ	カスタムメッセージ	メッセージをカスタマイズする。 （ユーザのロケールに合わせた言語で通知するなど）

Amazon Pinpoint – 分析

ユーザプールのユーザ利用状況を Amazon Pinpoint に提供し、アプリケーションや Cognito についての利用状況を分析でき、キャンペーンに利用できる情報を追加する。

- サインアップ
- サインイン
- 失敗した認証
- 1日当たりのアクティブユーザ（DAU）
- 1月当たりのアクティブユーザ（MAU）

Amazon CloudTrail – ログ記録

Cognito API 呼び出しは CloudTrail にログが記録され、監査やトラブルシューティングに使用できる。

ログ記録される API

✅ Cognito Identity Provider API

ログ記録されない API

❌ Cognito Auth API や Hosted UI

サインインしたユーザの情報なども
ログ記録するには、Lambda
トリガー内で出力する事で実現できる


```
{
    "eventVersion": "1.05",
    "userIdentity": {
        "type": "Unknown",
        "principalId": "Anonymous"
    },
    "eventTime": "2020-06-22T07:21:28Z",
    "eventSource": "cognito-idp.amazonaws.com",
    "eventName": "InitiateAuth",
    "awsRegion": "ap-northeast-1",
    "sourceIPAddress": "XXX.XXX.XXX.XXX",
    "userAgent": "Mozilla/5.0 (Macintosh; Intel Mac OS X……",
    "requestParameters": {
        "authParameters": {
            "USERNAME": "******",
            "SRP_A": "******"
        },
        "clientId": "3ff54d7l89o1atsu
        "clientMetadata": {},
        "authFlow": "USER_SRP_AUTH"
    },
    "responseElements": {
        "challengeName": "PASSWORD_VERIFIER",
        "challengeParameters": {
            "SALT": "******",
            "SECRET_BLOCK": "******",
            "USER_ID_FOR_SRP": "******",
            "USERNAME": "******",
            "SRP_B": "******"
```


サインインしたユーザの情報やセキュリティ情報は記録されない。

Amazon CloudWatch – モニタリング

CloudWatch メトリックにイベントの発生が記録される。
ユーザによる使用状況やスロットリングの発生を確認できる。

一部の API を対象にユーザプール、アプリクライアントごとに以下が記録される。

- サインアップの成功/スロットリング
- サインインの成功/スロットリング
- トークン更新の成功/スロットリング
- フェデレーションの成功/スロットリング

クォータ

API 呼び出しレートのクォータを越えた呼び出しは、スロットリングされて一時的なエラーが返される。リソースの数量などのクォータを超える作成や設定を行う API 呼び出しは、失敗が返される。

- Identity Provider API 呼び出しレートのソフト制限（デフォルト）

API 名	回/秒
AdminInitiateAuth	30
AdminRespondToAuthChallenge	20
AdminAddUserToGroup AdminRemoveUserFromGroup AdminListGroupsForUser	各10
SignUp、InitiateAuth、ForgotPassword などのユーザー認証操作	各10
ListUsers	25
上記にない API	各5

- リソースの数量、文字数、有効期限などの制限

https://docs.aws.amazon.com/ja_jp/cognito/latest/developerguide/limits.html

机上でのクォータ確認に加えて、
実際に想定されている構成、
アクセス量でのテストを行う。

テストは、徐々にアクセス量を増やしながらテストを行う。エラーやスロットリングが多く発生した場合は、それ以上の量でのテストは止める。

クォータを超えるアクセスが予想される場合やスロットリング発生については、上限緩和申請やサポートへの相談をする。

最後に ～ ユーザプールの特徴

1

サーバーレスで
認証とユーザ管理

サーバについて心配することなく、サインアップ、サインイン機能を実現

2

サードパーティとの
連携が充実

外部 ID プロバイダを使ったサインイン機能に加え、OAuth を使ってサードパーティアプリへの認可も実現

3

強力なセキュリティ機能

電話番号やEmailアドレスの検証や多要素認証を実現

アダプティブ認証も実現

参考：直近のアップデート (2020)

日付	内容
2020.04.07	Amazon Cognito アイデンティティプールが「Apple でサインイン」をサポート開始 https://aws.amazon.com/jp/about-aws/whats-new/2020/04/amazon-cognito-identity-pools-now-supports-sign-in-with-apple/
2020.02.12	Amazon Cognito ユーザープールサービスがユーザーエイリアスの大文字と小文字を区別しない設定の提供を開始 https://aws.amazon.com/jp/about-aws/whats-new/2020/02/amazon-cognito-user-pools-service-now-supports-case-insensitivity-for-user-aliases/
2020.02.05	Amazon Cognito User Pools サービスで、AWS CloudTrail を使用して、すべての API 呼び出しのログ記録ができるようになりました https://aws.amazon.com/jp/about-aws/whats-new/2020/02/amazon-cognito-user-pools-now-supports-logging-for-all-api-calls-with-aws-cloudtrail/
2020.01.10	Amazon Cognito での CloudWatch 使用メトリクスのサポートを開始 https://aws.amazon.com/jp/about-aws/whats-new/2020/01/amazon-cognito-supports-cloudwatch-usage-metrics/

参考：直近のアップデート (2019)

日付	内容
2019.11.26	Amazon Cognito がアカウント復旧メソッドの優先順位付けに対応 https://aws.amazon.com/jp/about-aws/whats-new/2019/11/amazon-cognito-supports-account-recovery-method-prioritization/
2019.11.20	Amazon Cognito が Apple でのサインインをサポート開始 https://aws.amazon.com/jp/about-aws/whats-new/2019/11/amazon-cognito-now-supports-sign-in-with-apple/
2019.10.31	Amplify CLI が Amazon Cognito ユーザープールグループを作成し、グループに対するきめ細かいアクセス権を設定し、ユーザーのマネジメント機能をアプリケーションに追加することを可能に https://aws.amazon.com/jp/about-aws/whats-new/2019/10/amplify-cli-enables-creating-amazon-cognito-user-pool-groups-configuring-fine-grained-permissions-adding-user-management-capabilities/
2019.10.07	Amazon Cognito で CloudFormation のサポートを強化 https://aws.amazon.com/jp/about-aws/whats-new/2019/10/amazon-cognito-increases-cloudformation-support/
2019.07.09	Amplify Framework で Auth カテゴリと Storage カテゴリのイベントに対する AWS Lambda トリガーの追加のサポートが開始 https://aws.amazon.com/jp/about-aws/whats-new/2019/07/amplify-framework-now-supports-adding-aws-lambda-triggers-for-events-auth-storage-categories/
2019.05.06	Amazon Cognito は、管理者向けユーザーパスワードリセット用 API を提供開始いたします。 https://aws.amazon.com/jp/about-aws/whats-new/2019/05/amazon-cognito-launches-enhanced-user-password-reset-api-for-administrators/
2019.04.05	Amplify フレームワークにおけるモバイルおよびウェブアプリケーション向けの OAuth 2.0 フロー、ホストされた UI、AR/VR シーンの設定が簡素化されます https://aws.amazon.com/jp/about-aws/whats-new/2019/04/amplify-framework-simplifies-configuring-oauth-2-0-flows--hosted/
2019.01.22	Amazon Cognito で 99.9% のサービスレベルアグリーメント (SLA) を発表 https://aws.amazon.com/jp/about-aws/whats-new/2019/01/amazon-cognito-announces-99-9-service-level-agreement/

参考: API リファレンス

4 種類の API が提供されており、それぞれ API リファレンスがある。

- Cognito ユーザプール
  - Cognito Identity Provider API (英語)
    https://docs.aws.amazon.com/cognito-user-identity-pools/latest/APIReference/Welcome.html
  - Cognito Auth API (英語)
    (OAuth 2.0/Open ID Connect 1.0)
    https://docs.aws.amazon.com/ja_jp/cognito/latest/developerguide/cognito-userpools-server-contract-reference.html
- Cognito ID プール - Cognito Federated Identity API (英語)
  https://docs.aws.amazon.com/cognitoidentity/latest/APIReference/Welcome.html
- Cognito Sync - Cognito Sync API (英語)
  https://docs.aws.amazon.com/cognitosync/latest/APIReference/Welcome.html

参考：リンク集

ハンズオンコンテンツ

- Using Amazon Cognito for serverless consumer apps (英語)
  https://serverless-idm.awssecworkshops.com/
- Amplify for iOS (英語)
  https://amplify-ios-workshop.go-aws.com/

ブログ

- Cognito の OAuth 機能を理解する (英語)
  https://aws.amazon.com/jp/blogs/mobile/understanding-amazon-cognito-user-pool-oauth-2-0-grants/
- Cognito と ADFS を連携させる (英語)
  https://aws.amazon.com/jp/blogs/mobile/building-adfs-federation-for-your-web-app-using-amazon-cognito-user-pools/
- サーバレス JavaScript アプリケーションで SAML (日本語, 英語)
  https://aws.amazon.com/jp/blogs/news/saml-for-your-serverless-javascript-application-part-i/
  https://aws.amazon.com/jp/blogs/compute/saml-for-your-serverless-javascript-application-part-ii/
- Cognito グループ、きめ細かなルールベースのアクセス制御 (英語)
  https://aws.amazon.com/jp/blogs/news/new-amazon-cognito-groups-and-fine-grained-role-based-access-control-2/
- ユーザを Cognito に移行する (英語)
  https://aws.amazon.com/jp/blogs/mobile/migrating-users-to-amazon-cognito-user-pools/
- パスワードレス認証 (英語)
  https://aws.amazon.com/jp/blogs/mobile/implementing-passwordless-email-authentication-with-amazon-cognito/

参考：リンク集

サポート ナレッジセンター

- Amazon Cognito のユーザープールで記憶済みデバイスを使用するには、どうすれば良いですか?
  https://aws.amazon.com/jp/premiumsupport/knowledge-center/cognito-user-pool-remembered-devices/
- Amazon Cognito のカスタムスコープを使用して API Gateway API へのアクセスを許可する方法を教えてください。
  https://aws.amazon.com/jp/premiumsupport/knowledge-center/cognito-custom-scopes-api-gateway/
- Amazon Cognito ユーザープールを使用してサードパーティの SAML ID プロバイダーを設定する方法を教えてください。
  https://aws.amazon.com/jp/premiumsupport/knowledge-center/cognito-third-party-saml-idp/
- Amazon Cognito JSON ウェブトークンの署名を復号して検証する方法を教えてください。
  https://aws.amazon.com/jp/premiumsupport/knowledge-center/decode-verify-cognito-json-token/

規格

- OAuth 2.0 (日本語)
  https://openid-foundation-japan.github.io/rfc6749.ja.html
- OAuth 2.0 Security Best Current Practice (英語)
  https://tools.ietf.org/id/draft-ietf-oauth-security-topics-13.htm
- 書籍：OAuth徹底入門 セキュアな認可システムを適用するための原則と実践 (日本語)
  https://www.amazon.co.jp/dp/B07L5M7DXS/ref=cm_sw_r_tw_dp_U_x_N0l8EbKFY86Q7

Q&A

お答えできなかったご質問については
AWS Japan Blog 「https://aws.amazon.com/jp/blogs/news/」にて
後日掲載します。

AWS の日本語資料の場所「AWS 資料」で検索

https://amzn.to/JPArchive

AWS Well-Architected 個別技術相談会

毎週”W-A個別技術相談会”を実施中

- AWSのソリューションアーキテクト(SA)に対策などを相談することも可能
- 申込みはイベント告知サイトから

(https://aws.amazon.com/jp/about-aws/events/)

AWS イベント で[検索]

ご視聴ありがとうございました

AWS 公式 Webinar
https://amzn.to/JPWebinar

過去資料
https://amzn.to/JPArchive