AWS-33

SaaS の認証認可について改めて考える
～ アーキテクチャーパターンと実装例 ～

柴田 龍平
技術統括本部 ISV/SaaSソリューション本部 ソリューションアーキテクト
アマゾン ウェブ サービス ジャパン合同会社

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Ryuhei Shibata（柴田 龍平）

所属

アマゾンウェブサービスジャパン合同会社
技術統括本部 ISV/SaaS ソリューション本部
ソリューションアーキテクト

好きな AWS サービス

AWS Security Hub　AWS WAF　AWS Fargate

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

はじめに

想定している視聴者の方

- 一般的な Web アプリケーションの認証認可について理解されている方
- SaaS アプリケーションの設計・実装をされている方
- アプリケーションのマルチテナント化を検討されている方

このセッションでお伝えしたいこと

- SaaS の認証認可のプラクティス
- マルチテナントでの認証とアクセス制御に活用できる AWS の機能

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

アジェンダ

I. SaaS アプリケーションにおける認証認可の課題とプラクティス

II. AWS でのアイデンティティプロバイダの選択肢

III. AWS での認可とテナント分離

IV. まとめ

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

SaaS アプリケーションにおける認証認可の課題とプラクティス

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

エンドユーザーが SaaS に期待するもの

使い始めが
簡単

サブスクリプション
モデルでの支払い

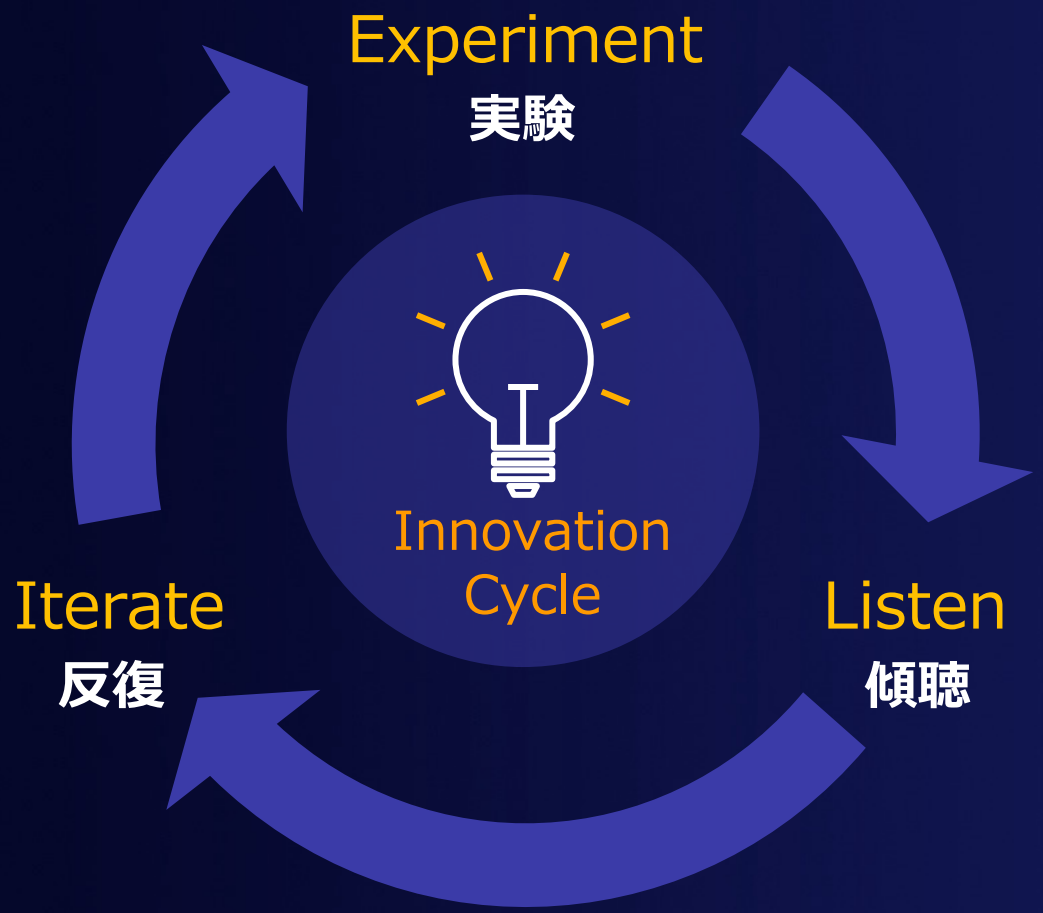
改善や革新の
速度

IT資産管理を
しなくても済む

aws

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

SaaS 開発に求められていること



Innovation Cycle を
高速に低コストで回すための
手段として

Infrastructure as Code

CI/CDパイプライン

マイクロサービス

コンテナ

サーバレス etc

ビジネスの差別化
ビジネスの価値を
最大限高める

aws
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

SaaS アプリケーションの認証認可の要件

- 高い可用性・信頼性
- 複数のサービスでの ID 共通化
- 顧客の既存の ID プロバイダとのフェデレーション
- 最小限のパフォーマンス影響
- 別テナントによるアクセスからの保護
- 開発者に負荷をかけない

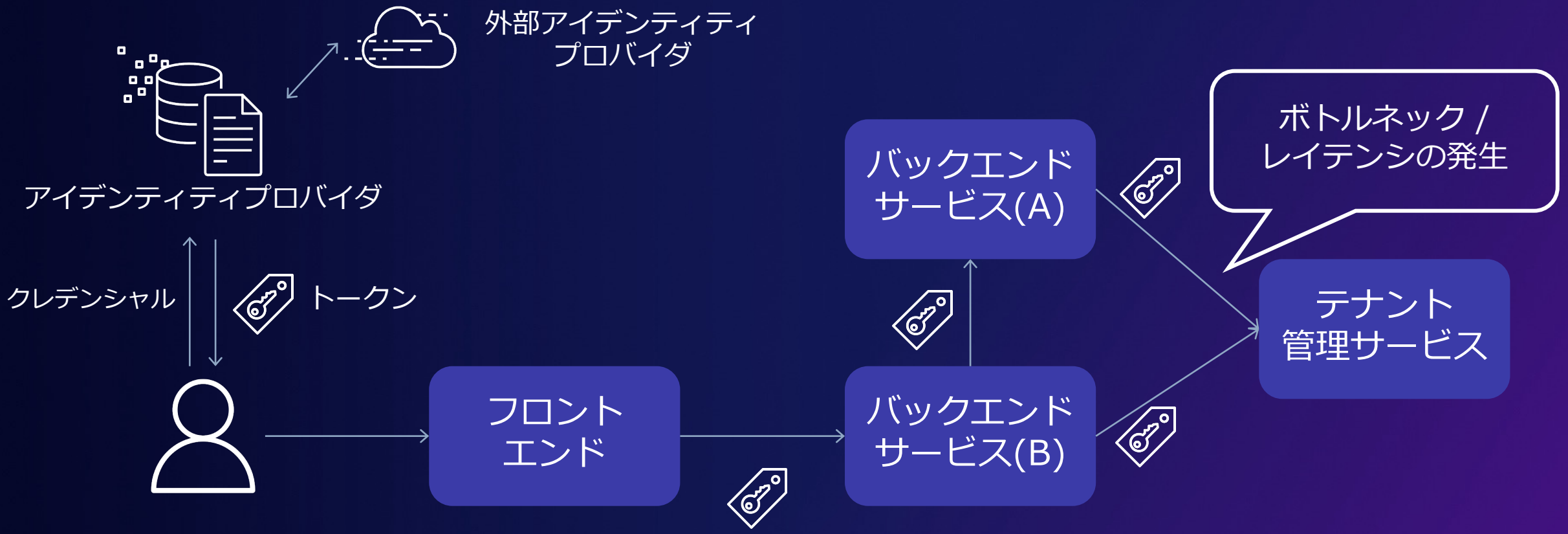
aws

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

SaaS における認証のプラクティス

(1) アイデンティティプロバイダの利用

認証を専用サービスであるアイデンティティプロバイダに移譲し
一元化されたユーザー管理 / フェデレーションなどの機能を実現する

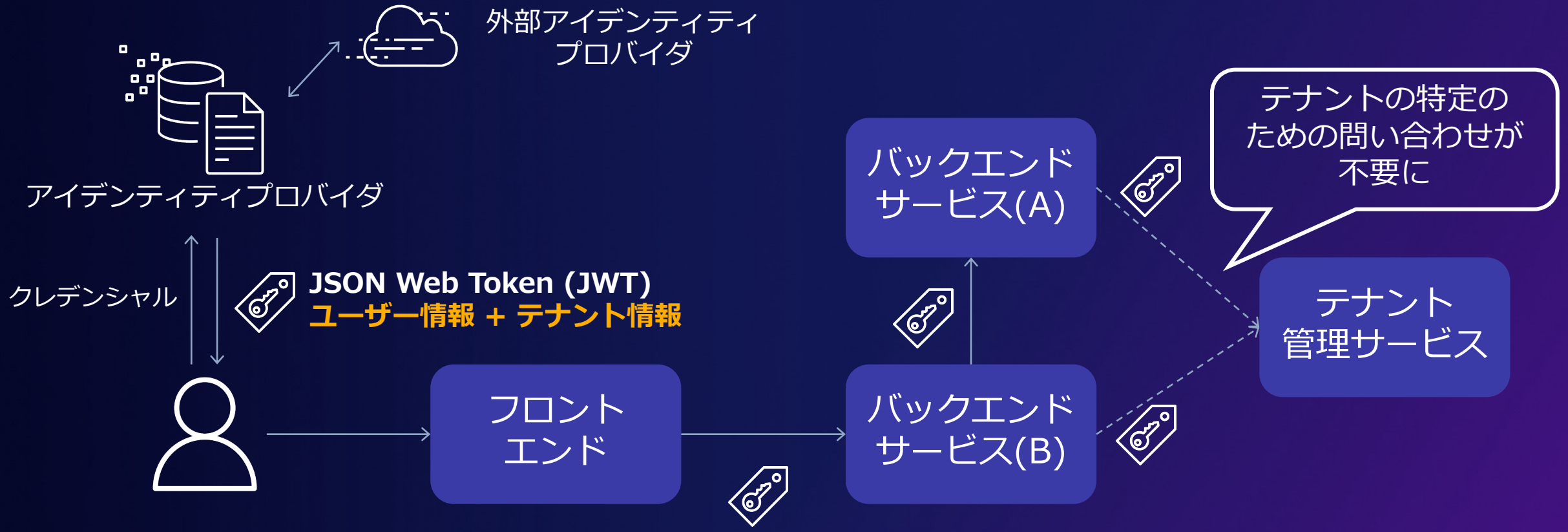


© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

SaaS における認証のプラクティス

(2) トークンによるアイデンティティの表現

ID プロバイダが発行するトークン (JSON Web Token) を活用しユーザーおよびテナントの情報を付与



© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

AWS でのアイデンティティプロバイダの選択肢

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

アイデンティティプロバイダの選択

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

運用負荷：低
AWS ネイティブな統合を
利用したい場合に選択
※下記URLのクォータは要確認

IDaaS

okta

onelogin by ONE IDENTITY

Ping Identity.

クラウドサービス
として
ユーザ認証を提供

運用負荷：低
Amazon Cognitoでは
満たせない要件に選択

パッケージを利用

OSS や商用パッケージを
Amazon EC2 / AWS Fargateで
稼働

運用負荷：中
スケールなどを自前で
管理したい場合に選択

ライブラリ等を使って独自実装

アプリの１機能
として実装

運用負荷：高
特殊な要件が
存在する場合に選択

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

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

アイデンティティプロバイダの選択

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

運用負荷：低
AWS ネイティブな統合を
利用したい場合に選択
※下記URLのクォータは要確認

IDaaS

Amazon Cognitoでは
満たせない要件に選択

パッケージを利用

スケールなどを自前で
管理したい場合に選択

ライブラリ等を使って独自実装

アプリの１機能
として実装

運用負荷：高
特殊な要件が
存在する場合に選択

Amazon Cognito にてテナントを表現するにはマルチテナントのための設計が必要

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

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Amazon Cognito でのマルチテナント設計

テナントの要件に応じて 4 つの方法が利用できる

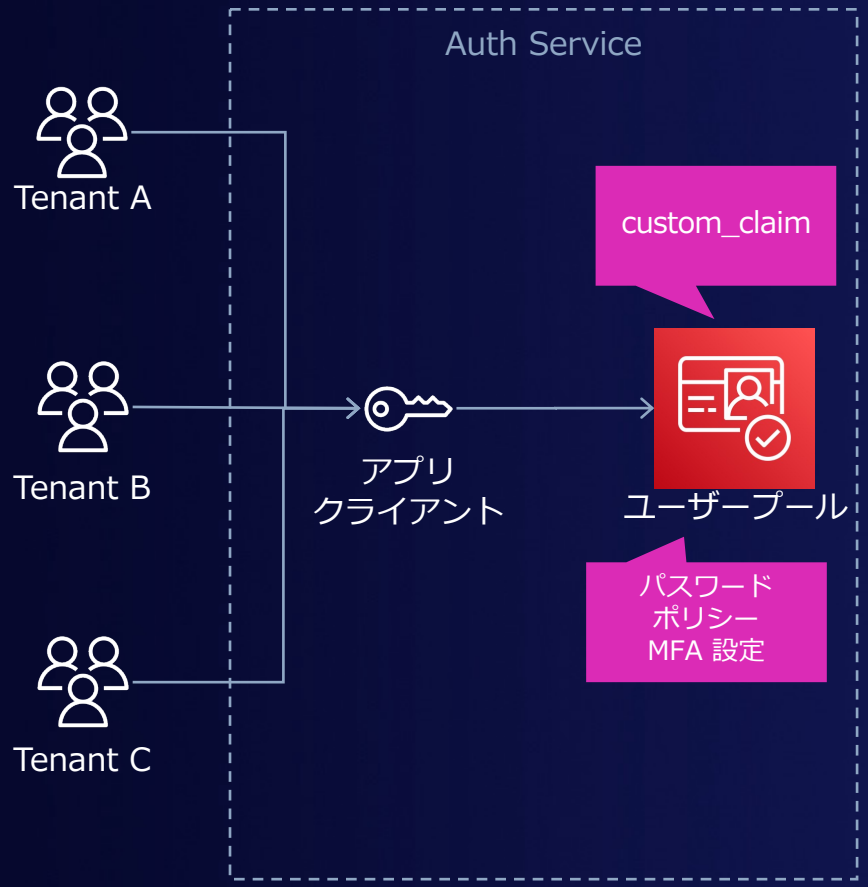
	1. カスタム属性ベース	2. グループベース	3. アプリクライアントベース	4. ユーザープールベース
実装コスト	低	低	高	高
テナント専用 IdP との連携	別の方法と組み合わせて実現	別の方法と組み合わせて実現	○	○
パスワードポリシーのカスタマイズ	全テナント共通	全テナント共通	全テナント共通	テナントごとに設定可能
スケーラビリティ	高	中	中～低	低

一長一短あるのでカスタム属性とアプリクライアントなど複数の方法を組み合わせる設計も検討する
いずれの方法もオンボーディング時にテナント情報を紐付けるための設計が必要
それぞれの方法ごとに検討する必要のあるクォータ値が異なるため、テナント数は要確認

aws
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

1. カスタム属性ベースのマルチテナント

テナントの情報をユーザーのプロファイルにカスタム属性として保存する



この方法を選択するべきシナリオ

- 認可にマネージドな機能を活用しつつ、テナント分離をしたい場合
- テナントごとのパスワードポリシーやフェデレーションなどの要件がない場合

ID トークンの例

```
{
  "sub": "aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeeee",
  "aud": "xxxxxxxxxxxxexample",
  "email_verified": true,
  "token_use": "id",
  "auth_time": 1500009400,
  "iss": "https://cognito-idp.us-east-1.amazonaws.com/us-east-1_example",
  "cognito:username": "janedoe",
  "exp": 1500013000,
  "custom:tenant_id": "tenant_A",
  …
}
```

注意すべきポイント

- **属性変更によるテナント跨ぎを防止**
  テナント識別用の属性は変更不可にする

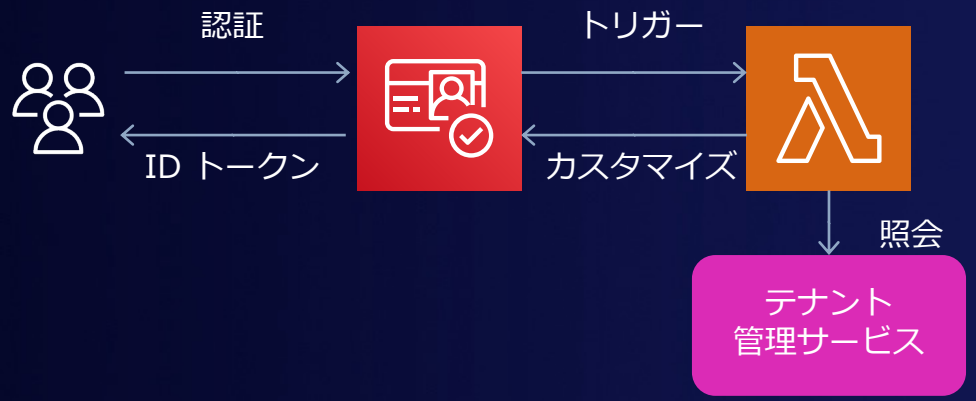
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

ID トークンへの動的なクレームの追加

トークン生成前の Lambda トリガーを利用し **ID トークン**発行時にクレームのカスタマイズが可能

カスタム属性はリクエストレートクォータの影響を受けたり検索に利用できないことから
変更や検索が発生し得る属性は別の DB に保存し動的にクレームを付与することも検討
※テナントの契約プラン、ユーザーの役割など

トークン中のクレームはログインし直すか
リフレッシュするまで更新されない点は注意

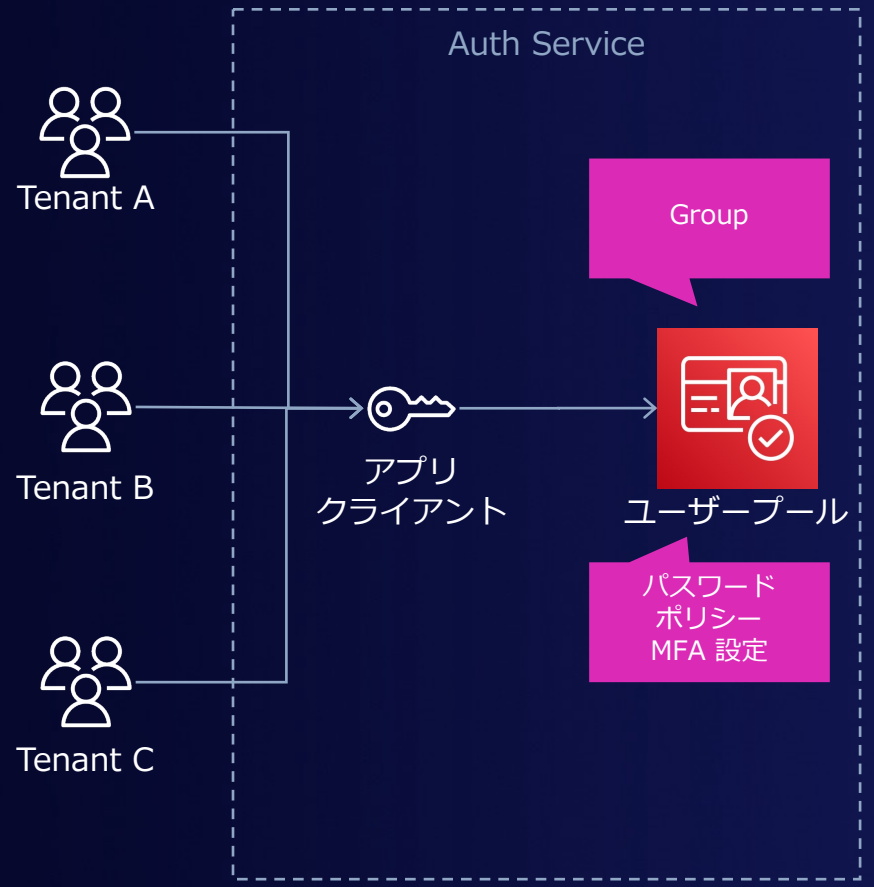



```
exports.handler = (event, context, callback) => {
    event.response = {
        "claimsOverrideDetails": {
            "claimsToAddOrOverride": {
                "attribute_key": "attribute_value"
            },
            "claimsToSuppress": ["email"],
            "groupOverrideDetails": {
                "groupsToOverride": ["group-A", "group-B", "group-C"],
                "iamRolesToOverride": ["arn:... ", "arn:...", "arn:..."],
                "preferredRole": "arn:aws:iam::XXXXXXXXXXX:role/sns_caller"
            }
        }
    };
    callback(null, event);
};
```

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

2. グループベースのマルチテナント

ユーザーテナントの紐付けをユーザープールグループとして保存する



この方法を選択するべきシナリオ

- Amazon Cognito ID プールの機能を利用してユーザーに IAM Role を割り当てたい場合
- ID トークンでなく、IAM の機能やアクセストークンで認可を行いたい場合

ID トークンの例

```
{
  "sub": "aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeeee",
  "cognito:groups": ["tenant_A"],
  "aud": "xxxxxxxxxxxxexample",
  "email_verified": true,
  "token_use": "id",
  "auth_time": 1500009400,
  "iss": "https://cognito-idp.us-east-1.amazonaws.com/us-east-1_example",
  "cognito:username": "janedoe",
  "exp": 1500013000,
...
}
```

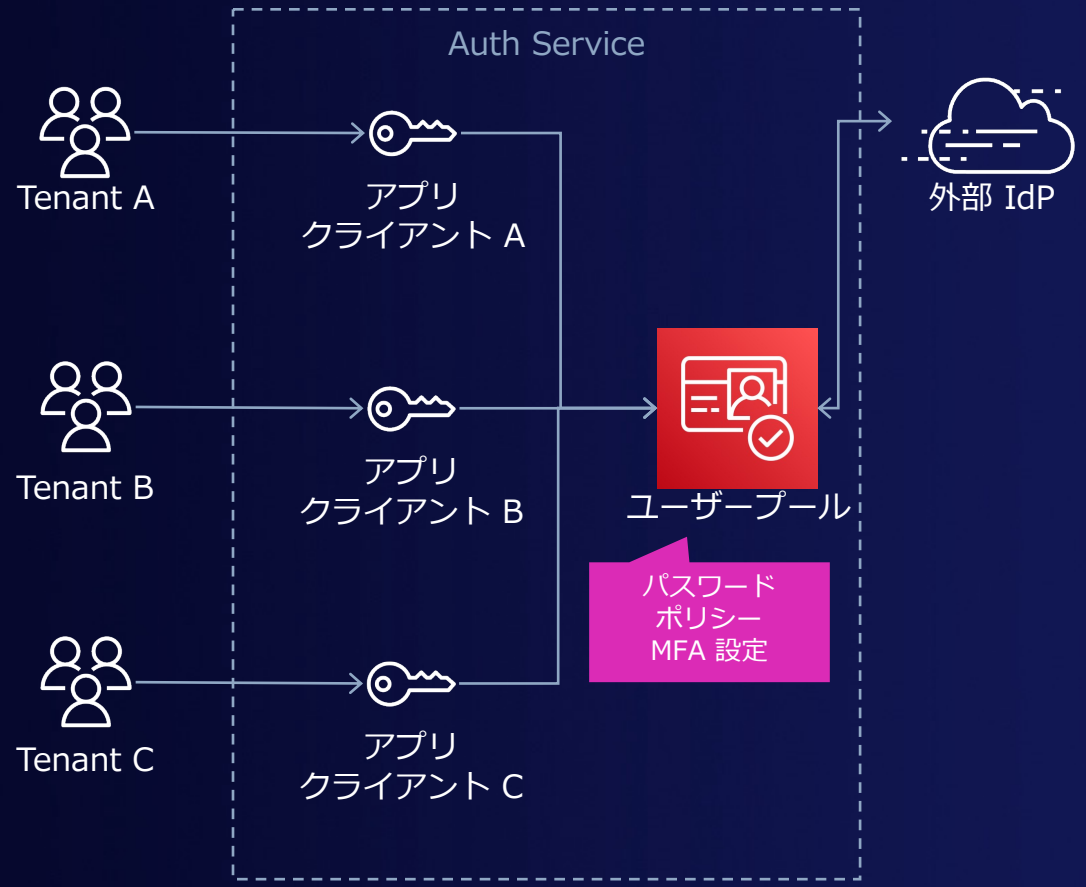
注意すべきポイント

- **ユーザープール内のデフォルトクォータ**
  グループ数は 10,000 に指定されている

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

3. アプリクライアントベースのマルチテナント

テナントごとにアプリクライアントを作成する



この方法を選択するべきシナリオ

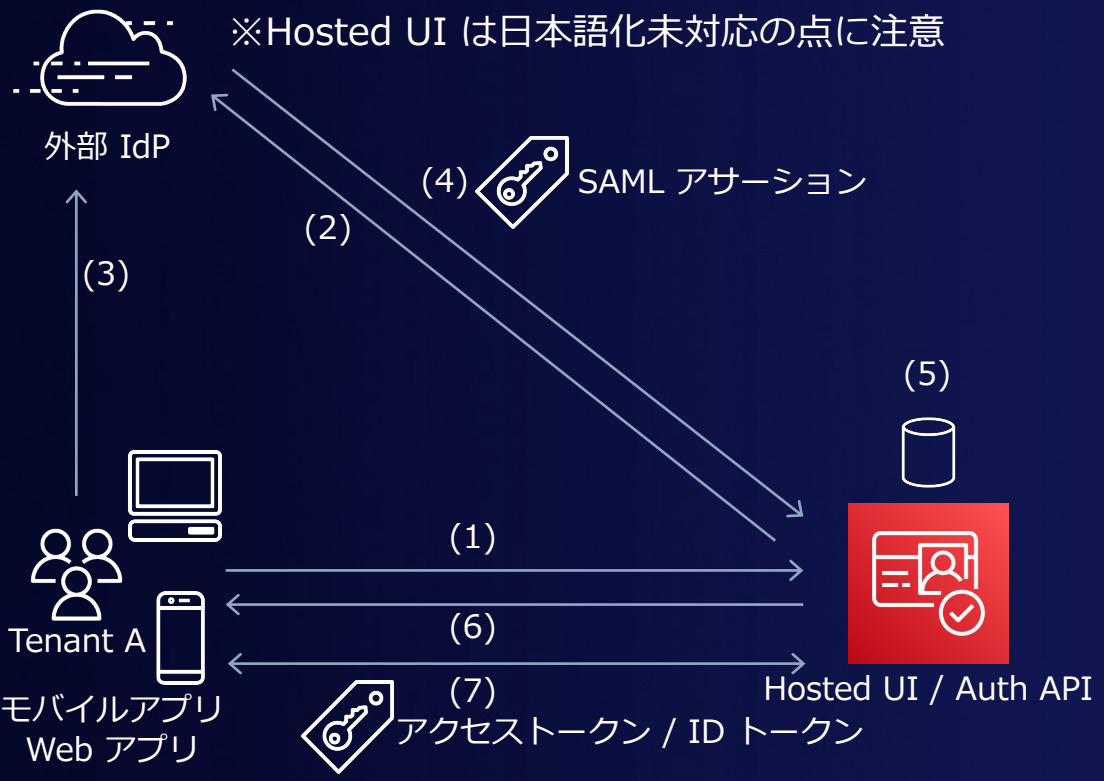
- 特定のテナントで外部 IdP を利用したフェデレーションログインを利用したい場合

注意すべきポイント

- **ユーザープールあたりのデフォルトクォータ**
  アプリクライアント数は 1,000、IdP 数は 300 までに指定されている
  通常ログインはカスタム属性ベース、フェデレーションが必要なテナントのみアプリクライアントベースなどの組み合わせも要検討
- **ID プロバイダの紐付け**
  アプリクライアントにテナント専用の ID プロバイダー以外を紐付けるとアプリクライアント ID だけではテナントを識別できない

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

外部 IdP によるフェデレーションログインの一例



(1) アプリケーションから 直接、もしくは Hosted UI 経由で
Cognito Auth API の認可エンドポイントに接続

(2) SAML 認証要求とともに IdPにリダイレクト

(3) 外部 IdP の画面経由でユーザーがログイン

(4) 外部 IdP から Cognito に SAML アサーションを返答

(5) Cognito はユーザープロファイルを作成もしくは更新

(6) Cognito からアプリケーションに認可コード付きでリダイレクト

(7) アプリケーションはトークンエンドポイントから
アクセストークンと ID トークン、リフレッシュトークンを取得

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

外部 IdP によるフェデレーションログインの一例

アプリケーションからの認可エンドポイントの呼び出し

```
let url = `https://${domain}.auth.${region}.amazoncognito.com/oauth2/authorize?`;      // Hosted UIのドメイン
url   += `response_type=code&`;                                  // 認可コードフローを利用
url   += `idp_identifier=${idpIdentifier}&`;                     // 参照する外部 IdP を識別する情報
url   += `client_id=${appClientId}&`;                            // アプリクライアントのID
url   += `redirect_uri=${redirectUrl}&`;                         // (6) でアプリケーションにリダイレクトする際のURL
url   += `state=${state}&`;                                      // CSRF 対策のため、state 値をリクエストに含めることを推奨
url   += `scope=openid+profile+aws.cognito.signin.user.admin&`;
url   += `code_challenge_method=S256&`;
url   += `code_challenge=${codeChallenge}`;                      // PKCE を用いた認可コードフローを利用

location.href = url;
```

aws
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

外部 IdP によるフェデレーションログインの一例

ID トークンの例

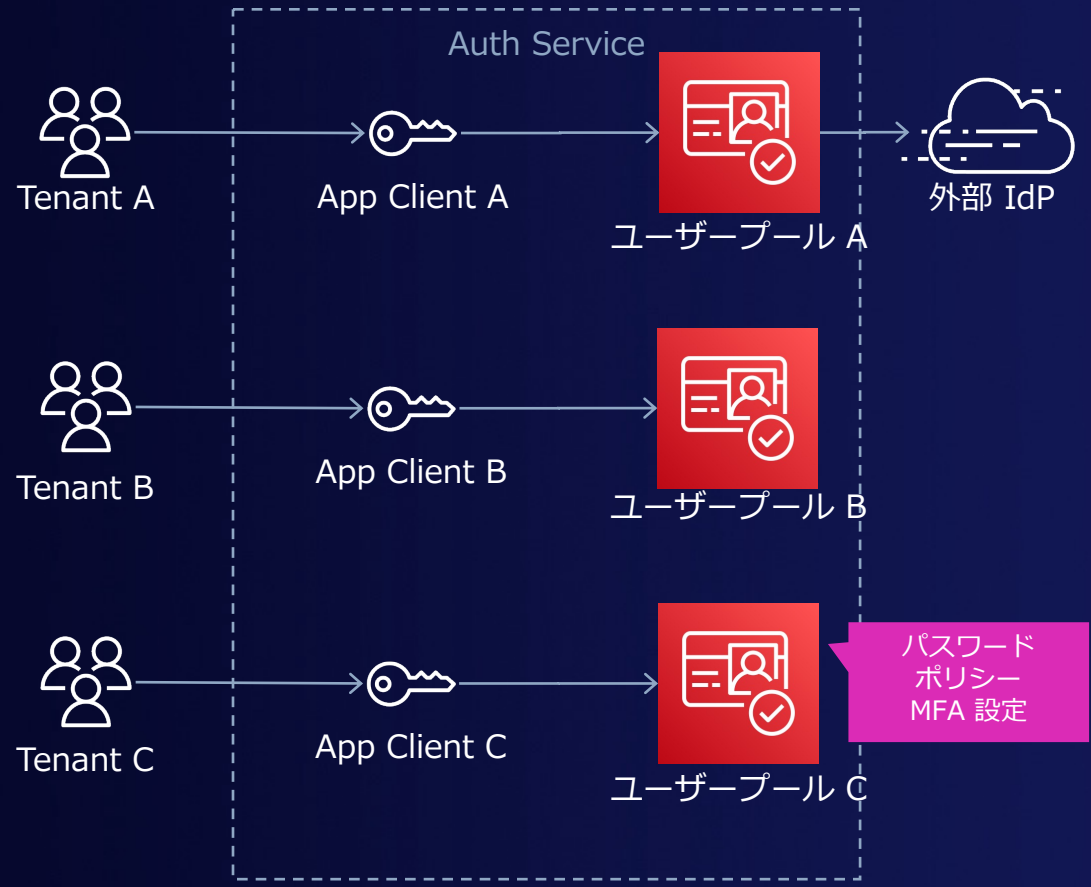
```
{
  "sub": "aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeeee",
  “aud”: "<tenant_A_app_client>",
  "email_verified": true,
  "token_use": "id",
  "auth_time": 1500009400,
  "iss": "https://cognito-idp.us-east-1.amazonaws.com/us-east-1_example",
  "cognito:username": "janedoe",
  "exp": 1500013000,
  "identities": [{
    ”providerName”: “tenant_A-idp”,
    “providerType”: “SAML”,
    “issuer”: “http://xxxxx”
    …
  }],
  …
}
```

- テナントごとにアプリクライアントを作成している場合 aud クレームからテナントを識別できる
- アプリケーションからテナントを識別しやすいように Lambda トリガーでテナントを識別するクレームを動的に付与することも検討可能

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

4. ユーザープールベースのマルチテナント

テナントごとにユーザープールを分離する



この戦略を採用するべきシナリオ

- テナントごとにパスワードポリシー、MFA 設定を変更する必要がある場合
- あるユーザーが同じメールアドレスで複数のテナントに所属したり、それぞれ異なる役割（管理者、一般ユーザー）を持つような複雑な要件がある場合

注意すべきポイント

- **アカウントごとのデフォルトクォータ**
  ユーザープール数は1,000 に指定されている全てのテナントでなく、特定のテナントのみこの方法を利用するなども検討する
- **認可の実装コスト**
  ユーザープールを分割する場合、Amazon API Gateway でのアクセス制御方法が限定される

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

実装時のセキュリティ上の留意事項

悪意のあるユーザーが別テナントを装ってアクセスできないようにする

- ユーザーとテナントのマッチングに利用する値がユーザーや管理者によって別テナントのものに変更不可能であることを保証する
- ユーザーのメールアドレスドメインを用いてテナント識別を行う場合は、当該メールアドレスがアプリケーションや外部 IdP によって検証されている場合のみ信頼する
- 認証後、アプリケーションでテナント識別に利用する tenant_id などのカスタム属性をユーザー、テナント管理者が変更できない設定（イミュータブル）にする

外部 IdP とのフェデレーションにより自動的に別テナントにログインされてしまわないようにする

- 外部 IdP を利用した Amazon Cognito セッション Cookie を持っているユーザーは、同じ IdP を利用する他のアプリクライアントでもアクセスが可能。外部 IdP を利用する場合は、あるテナントの外部 IdP を他のテナントで利用するアプリクライアントに紐付けないようにする

aws
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

アイデンティティプロバイダの選択

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

運用負荷：低
AWS ネイティブな統合を
利用したい場合に選択
※下記URLのクォータは要確認

IDaaS

okta

onelogin by ONE IDENTITY

Ping Identity.

クラウドサービス
として
ユーザ認証を提供

運用負荷：低
Amazon Cognitoでは
満たせない要件に選択

パッケージを利用

OSS や商用パッケージを
Amazon EC2 / AWS Fargateで
稼働

運用負荷：中
スケールなどを自前で
管理したい場合に選択

ライブラリ等を使って独自実装

アプリの１機能
として実装

運用負荷：高
特殊な要件が
存在する場合に選択

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

aws

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

ビジネスプロセスをもう一度見つめ直す

デジタルトランスフォーメーション: これまで以上の速度での進化が求められている

変革
ビジネスモデルと運用の再考

モダナイズ
高速な成長のための戦略

最適化
効率化によるコスト削減

技術革新
実現のためのスピード

クラウドマーケットプレイス

組織がサプライチェーンをモダナイズして
上記の目標を達成できるような支援を提供

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

クラウドマーケットプレイスを活用し DX を実現

- 慣れ親しんだソフトウェアとデータを引き続き活用しつつ企業の調達プロセスを変革
- 調達の高速化、ガバナンスの向上 IT 支出の最適化を全て 1 箇所で実現可能

慣れ親しんだソフトウェアとデータ

アプリのモダナイズのための幅広い選択肢

調達と支払いのプロセスを簡素化

ガバナンスと統制の強化

プロフェッショナルによるサポートと専門知識の提供

IT 支出の最適化

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

AWS Marketplace を活用して DX のための サードパーティソリューションを簡単に発見

aws marketplace

企業が世界中のソリューションから適切なものを
検索、サブスクライブ、デプロイ、管理する方法を変革


ソフトウェア
SaaS


データ


ML モデル


プロフェッショナル
サービス


コンテナ

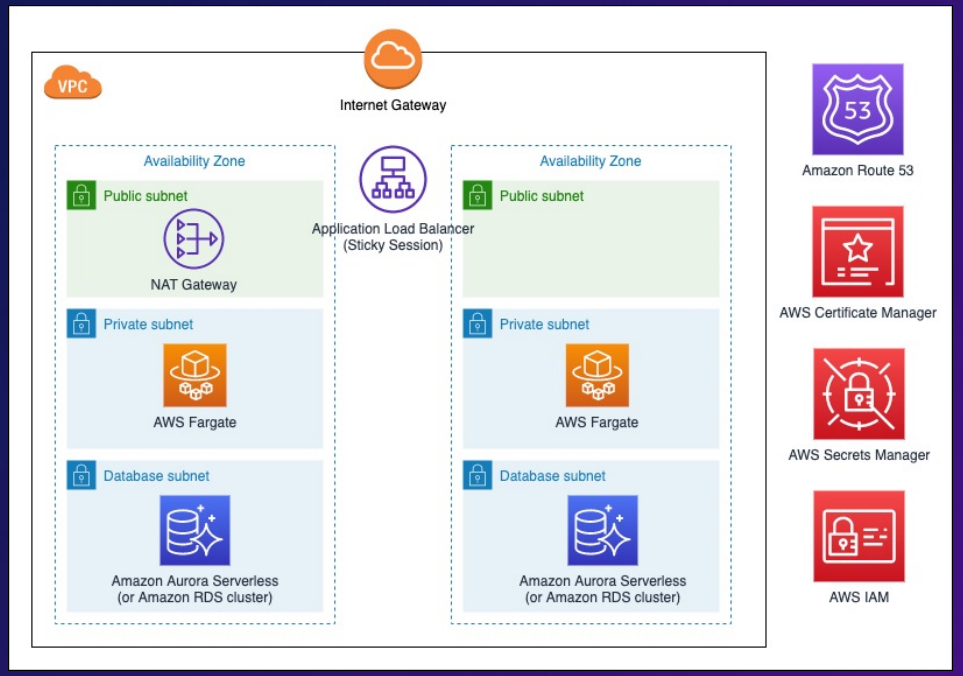
ISV やデータプロバイダー、リセラーにとっても新規顧客を獲得したり、
既存顧客をクラウド移行し収益を拡大するための戦略的なチャネルとして利用可能

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

マネージドサービス、IDaaS 以外の選択肢

マネージドサービスや IDaaS では時間あたりの認証要求数などの要件を満たせない場合、OSS やパッケージの利用も検討する

AWS Fargate や Amazon Aurora Serverless を用いて Keycloak を稼働させるサンプル実装として Keycloak on AWS ソリューションが利用可能



Keycloak on AWS Solution
https://github.com/aws-samples/keycloak-on-aws

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

SaaS アプリケーションの認証認可の要件

高い可用性・信頼性

アイデンティティプロバイダの利用

複数のサービスでの ID 共通化

顧客の既存の ID プロバイダとのフェデレーション

最小限のパフォーマンス影響

JWT によるテナントの表現

別テナントによるアクセスからの保護

開発者に負荷をかけない

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

AWS での認可とテナント分離

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

SaaS におけるレイヤー毎のアクセス制御

クライアント

ゲートウェイ

Amazon API Gateway

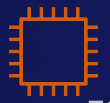
AWS AppSync

トークンを識別し
API への
アクセスを制御

- 認証済みユーザーのみ許可
- 契約プランに応じた
  API 呼び出しのみ許可
- テナント毎のスロットリング

など

コンピュート



Amazon EC2

AWS Fargate

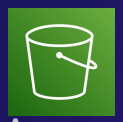
AWS Lambda

テナントを識別し
ビジネスロジックを
処理

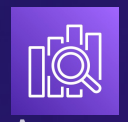
- テナントリソースの読み書き
- アプリケーション機能の制限
- **ゲートウェイがない場合は
  ゲートウェイ相当の処理**

など

ストレージ / DB



Amazon
S3



Amazon
OpenSearch Service

Amazon
DynamoDB

Amazon
Aurora

各サービスの機能で
テナント毎の
データアクセスを分離

- テナント毎の prefix にだけ
  アクセスを許可
- 行レベルセキュリティで
  テナントアクセスを分離

など

aws

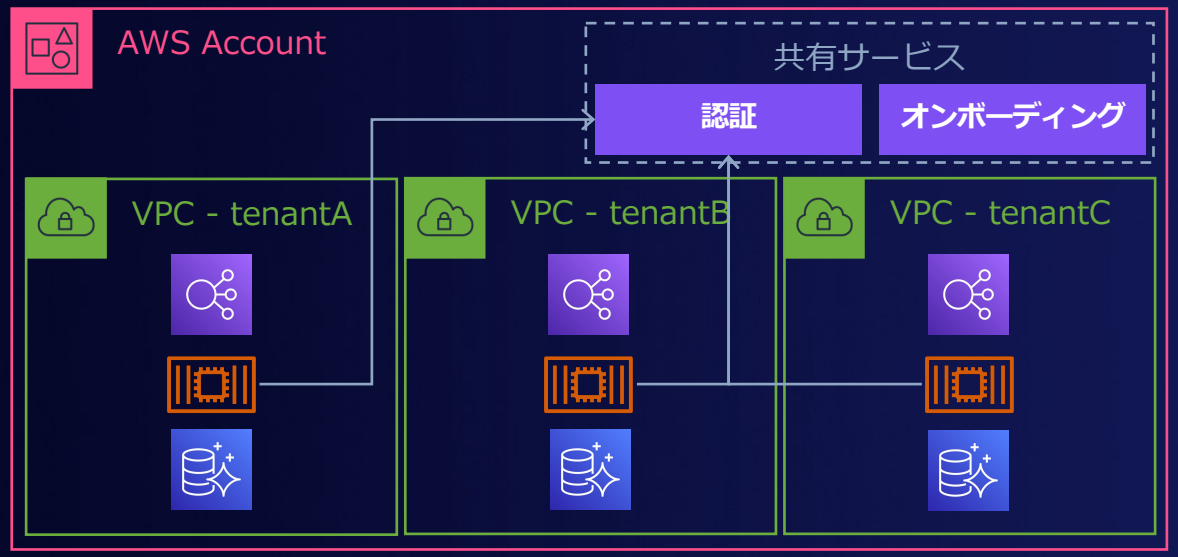
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

SaaS におけるレイヤー毎のアクセス制御

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

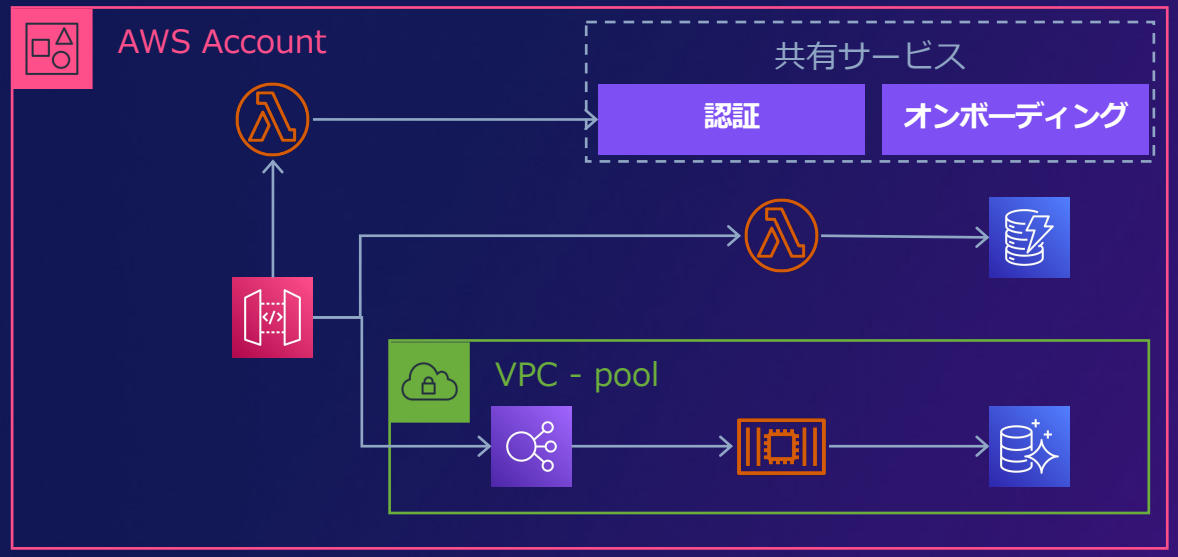
テナント分離のためのデプロイモデル

サイロモデル



テナントが専用のインフラストラクチャを保有
テナントデータの処理や保管はテナント占有リソースで実施される
サイロであっても認証やオンボーディングなどは共有サービスとして作成することが望ましい

プールモデル



テナント同士がリソースをシェアして利用
テナントデータの処理、保管はアプリケーションによって論理的に分離される

多くのサービスでは一部リソースはサイロとして分割し
一部リソースはプールにする**ブリッジモデル**を採用

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

ゲートウェイレイヤーでのアクセス制御

JWT の検証のためのメカニズム

Gateway	アクセス制御	備考
Amazon API Gateway REST API	COGNITO_USER_POOLS オーソライザー	認証済みユーザーかどうかの検証をコードなしで実施 Amazon Cognito の単一ユーザープールにのみ対応
	Lambda オーソライザー	コードによる詳細なアクセス制御が可能 複数ユーザープールや Amazon Cognito 以外の IdP にも対応 スロットリングのための API キーのソースとしても利用可能
Amazon API Gateway HTTP API	JWT オーソライザー	Open ID Connect および OAuth 2.0 に基づいた検証を実施 Amazon Cognito と共に使用する場合は対象のユーザープールおよびアプリクライアントを指定する オーソライザー毎の Audience の最大数が 50 である点に注意
	Lambda オーソライザー	ペイロード形式のバージョン 2.0 は REST API の Lambda オーソライザーとは互換性がないため注意

aws
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

ゲートウェイレイヤーでのアクセス制御

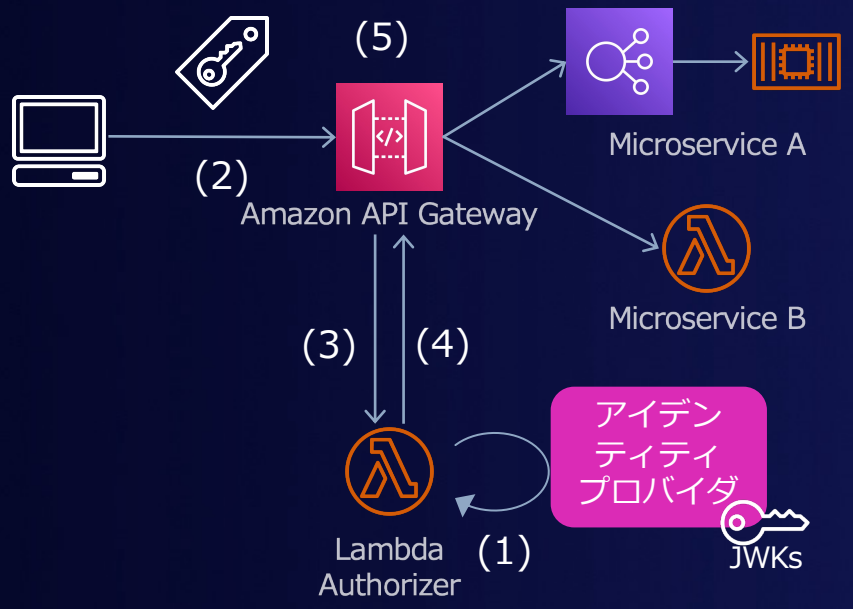
JWT の検証のためのメカニズム

Gateway	アクセス制御	備考
AWS AppSync	AWS_LAMBDA	Lambda 関数を用いてクエリ文字列や認可用のトークンによるアクセス制御が可能 アクセス可否はポリシーではなく Boolean で指定 特定のフィールドをマスクすることも可能
	OPENID_CONNECT	特定の Issuer による Open ID Connect トークンがリクエストに含まれているかの検証が可能 3rd party IdP を用いる場合に利用する Issuer およびオプションで Client ID による検証が可能
	AMAZON_COGNITO_USER_POOLS	Amazon Cognito の単一ユーザープールにのみ対応 オプションでアプリクライアントの検証が可能 ユーザープールグループによる GraphQL フィールド個別のアクセス制御が可能

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

REST API Lambda オーソライザーの利用

Lambda オーソライザーによる処理の流れ（JWT で公開鍵署名アルゴリズムを利用している場合）



(1) Lambda 関数で JWT 検証用の鍵を IdP から事前に取得する

(2) クライアントから API Gateway にトークンの付与されたリクエストが発行される

(3) トークンの検証のため Lambda関数 を実行する

(4) Lambda 関数は JWT を検証しポリシーステートメントやコンテキスト、スロットリング用の API-Key を応答する

(5) API Gatewayがポリシーを評価する
キャッシュが有効になっている場合はポリシーを一定期間キャッシュする

Amazon Cognito を利用する場合、JWT の検証については下記 URL を参照
https://docs.aws.amazon.com/ja_jp/cognito/latest/developerguide/amazon-cognito-user-pools-using-tokens-verifying-a-jwt.html

aws
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

REST API Lambda オーソライザーの出力例

Lambda 関数の出力例

```
{
  ”principalId”: ”{user_idenitifier}”,
  "policyDocument": {
    "Version": "2012-10-17",
    "Statement": [{
      "Action": "execute-api:Invoke",
      "Effect": "Allow",
      “Resource”: [                              // ユーザーのロールや テナントの契約によって細かく制御
        "arn:aws:execute-api:{region}:{account_id}:{api_id}/{stage}/GET/*”,
        "arn:aws:execute-api:{region}:{account_id}:{api_id}/{stage}/POST/{resource}"
      ],
      “Condition”: { ... }                       // テナントの属性に応じた追加の制約があれば指定
  },
  “context”: {
    "stringKey": "value"
  },
  “usageIdentifierKey”: “{api_key}”          // テナント毎のスロットリングを行う場合に利用可能
}
```

全てのアクセス制御をゲートウェイレイヤーで行えるわけではないため
コンピュートレイヤーとの役割分担を設計する必要がある点に注意

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

コンピュートレイヤーでのアクセス制御

- マイクロサービス間で JWT を受け渡すことでアクセス制御を実施する
- JWT 内のテナント ID などのクレームを利用する場合は aud クレーム等を検証することで**このサービスで利用することを想定した JWT かどうかを検証**する
- JWT の検証は共通ライブラリ、Lambda レイヤーとして実装し**開発者の労力を小さく**する

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

ストレージ / DB レイヤーでのアクセス制御

Amazon Simple Storage Service (S3)

- バケットの分割（サイロ）
- テナント専用プレフィックスによる分離（プール）

Amazon OpenSearch Service

- ドメインの分離（サイロ）
- インデックスの分離（サイロ）
- パーティションキーによる分離（プール）

Amazon DynamoDB

- テーブルの分割（サイロ）
- 行レベルセキュリティによる分割（プール）

Amazon Aurora PostgreSQL

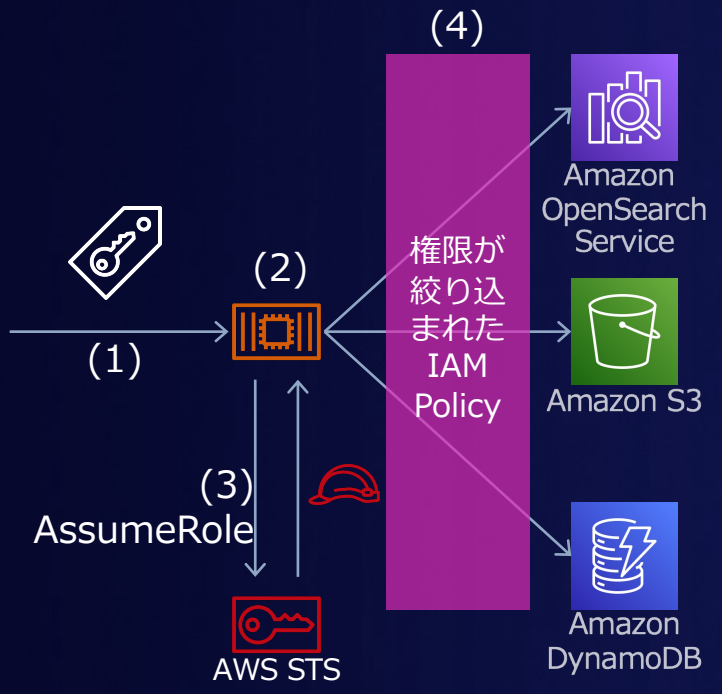
- インスタンスの分割（サイロ）
- スキーマ / テーブルの分割（ブリッジ）
- 行レベルセキュリティによる分割（プール）

アプリケーションロジックだけでなく ストレージ / DB レイヤーでの保護を追加することで**アプリケーションのリリース速度とセキュリティを両立**させることが可能

aws
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

プール分離のための IAM ポリシーの動的生成

ストレージ / DB によっては IAM ポリシーを用いてテナントごとのアクセス制御を行うが、
IAM ポリシー / ロールのリソースクォータや管理性の面から IAM ポリシーの動的作成も検討する



(1) テナント情報を含む JWT をリクエストと共に受け取る

(2) JWT の検証を行い、正規のトークンであることを確認する

(3) AWS Security Token Service (AWS STS) に動的なポリシーを引き渡し、一時クレデンシャルを要求

(4) 権限が絞り込まれたクレデンシャルを利用して、各ストレージ、DB に読み書きを実施

※ポリシーの動的生成以外にも、ABAC を用いる方法も存在する

サービスごとの具体的な実装については下記ブログ記事を参照
https://aws.amazon.com/jp/blogs/news/isolating-saas-tenants-with-dynamically-generated-iam-policies/
https://aws.amazon.com/jp/blogs/news/how-to-implement-saas-tenant-isolation-with-abac-and-aws-iam/

aws
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

SaaS アプリケーションの認証認可の要件

- 高い可用性・信頼性 — アイデンティティプロバイダの利用
- 複数のサービスでの ID 共通化
- 顧客の既存の ID プロバイダとのフェデレーション
- 最小限のパフォーマンス影響 — JWT によるテナントの表現
- 別テナントによるアクセスからの保護 — テナント分離
- 開発者に負荷をかけない — オーソライザーの機能の活用
ライブラリ、Lambdaレイヤー化

まずは、自社 SaaS において必要となる要件から着手する

aws

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

まとめ

SaaS アプリケーションの認証認可のプラクティス

- アイデンティティプロバイダの利用
- トークンによるテナント情報の表現

要件に応じて適切なアイデンティティプロバイダを選択する

- Amazon Cognitoを用いる場合は最適な方法を選択
- 要件によってはパートナーソリューションやパッケージの利用も検討

テナントの属性を用いて認可とテナント分離を実施

- レイヤーや分離モデル毎の方法が存在
- 複数レイヤーで分離を実装し、サービスデベロッパーの負荷を軽減

aws
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

参考資料

- Amazon Cognito マルチテナントアプリケーションのベストプラクティス
  https://docs.aws.amazon.com/ja_jp/cognito/latest/developerguide/multi-tenant-application-best-practices.html
- Amazon Cognito トークン生成前の Lambda トリガー
  https://docs.aws.amazon.com/ja_jp/cognito/latest/developerguide/user-pool-lambda-pre-token-generation.html
- Amazon Cognito SAML ユーザープール IdP 認証フロー
  https://docs.aws.amazon.com/ja_jp/cognito/latest/developerguide/cognito-user-pools-saml-idp-authentication.html
- 外部アイデンティティプロバイダと SAML で連携する方法
  https://aws.amazon.com/jp/premiumsupport/knowledge-center/cognito-okta-saml-identity-provider/
  https://aws.amazon.com/jp/blogs/security/how-to-set-up-amazon-cognito-for-federated-authentication-using-azure-ad/
- Amazon Cognito マルチテナンシーのセキュリティに関する推奨事項
  https://docs.aws.amazon.com/ja_jp/cognito/latest/developerguide/multi-tenancy-security-recommendations.html
- AWS Well-Architected フレームワーク SaaS レンズ
  https://docs.aws.amazon.com/ja_jp/wellarchitected/latest/saas-lens/saas-lens.html

aws
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

参考資料

- Amazon API Gateway / AWS AppSync のアクセス制御
  https://docs.aws.amazon.com/ja_jp/apigateway/latest/developerguide/apigateway-control-access-to-api.html
  https://docs.aws.amazon.com/ja_jp/apigateway/latest/developerguide/http-api-access-control.html
  https://docs.aws.amazon.com/ja_jp/appsync/latest/devguide/security-authz.html
- Amazon API Gateway Lambda オーソライザーを使用する
  https://docs.aws.amazon.com/ja_jp/apigateway/latest/developerguide/apigateway-use-lambda-authorizer.html
- SaaS テナント分離戦略（英語）
  https://d1.awsstatic.com/whitepapers/saas-tenant-isolation-strategies.pdf
- 動的に生成された IAM ポリシーで SaaS テナントを分離する
  https://aws.amazon.com/jp/blogs/news/isolating-saas-tenants-with-dynamically-generated-iam-policies/
- SaaS テナント分離を AWS IAM と ABAC で実装する方法
  https://aws.amazon.com/jp/blogs/news/how-to-implement-saas-tenant-isolation-with-abac-and-aws-iam/
- PostgreSQL の行レベルのセキュリティを備えたマルチテナントデータの分離
  https://aws.amazon.com/jp/blogs/news/multi-tenant-data-isolation-with-postgresql-row-level-security/

aws
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Thank you!

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.