



アップデート紹介とちょっぴり DIVEDEEP する AWS の時間
- AWS RE:INFORCE デモ祭り

ついに GA ! Amazon Verified Permissions で アプリケーションの認可をシンプルに !

Ryuhei Shibata

Solutions Architect

Amazon Web Services Japan G.K.

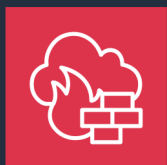
Ryuhei Shibata (柴田 龍平)

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 Network
Firewall



Amazon Cognito



AWS Fargate



Agenda

Amazon Verified Permissions とは

アプリケーションへの導入パターン

DEMO

まとめ

Amazon Verified Permissions を一般提供開始

- カスタムアプリケーション向けのスケラブルできめ細やかな認証およびアクセスコントロールを実現する新サービス
- アクセス制御はオープンソースとして公開しているポリシー言語の Cedar で定義
- アプリケーションロジックからアクセス制御を分離し、コードを変更する必要なく、パーミッションルールの変更と更新を一元管理し、運用を簡素化。アプリケーションリソースにパーミッションを実装する工数を大幅に削減
- 全リージョンで一般利用可能



アプリケーションコードでの認可の課題

```
def get_book(request):  
    log("Handling book request " + request.id)  
    book = db.query(request.bookId)  
  
    return {  
        'id': book.id,  
        'title': book.title,  
        'rating': book.rating,  
        'numReviews': book.numReviews,  
    }
```

アプリケーションコードでの認可の課題

```
def get_book(request):  
    if db.query(request.bookId).owner != request.user:  
        return 'AccessDenied'  
  
    log("Handling book request " + request.id)  
    book = db.query(request.bookId)  
  
    return {  
        'id': book.id,  
        'title': book.title,  
        'rating': book.rating,  
        'numReviews': book.numReviews,  
    }
```

アプリケーションコードでの認可の課題

```
def get_book(request):  
    if not db.query(request.user).admin:  
        if db.query(request.bookId).owner != request.user:  
            return 'AccessDenied'  
  
    log("Handling book request " + request.id)  
    book = db.query(request.bookId)  
  
    return {  
        'id': book.id,  
        'title': book.title,  
        'rating': book.rating,  
        'numReviews': book.numReviews,  
    }
```

アプリケーションコードでの認可の課題

```
def get_book(request):  
    if not db.query(request.user).admin:  
        if db.query(request.bookId).owner != request.user:  
            return 'AccessDenied'
```

```
if not request.multiFactorAuth:  
    return 'AccessDenied'
```

```
log("Handling book request " + request.id)  
book = db.query(request.bookId)
```

```
return {  
    'id': book.id,  
    'title': book.title,  
    ...
```


アプリケーションコードでの認可の課題

```
def get_book(request):  
    if not db.query(request.bookId).isPublic:  
        if not db.query(request.user).admin:  
            if db.query(request.bookId).owner != request.user:  
                return 'AccessDenied'  
  
        if not request.multiFactorAuth:  
            return 'AccessDenied'  
  
    log("Handling book request " + request.id)  
    book = db.query(request.bookId)  
  
    return {  
        'id': book.id,  
        ...
```

ビジネスの成長に伴い、
アプリケーションのいたるところに
認可ロジックが分散

ロジックが複雑化し
ルールが合っていることの
確認が困難に・・・

Amazon Verified Permissions でシンプルに！

```
def get_book(request):  
    if not client.is_authorized(...):  
        return 'AccessDenied'  
  
    log("Handling book request " + request.id)  
    book = db.query(request.bookId)  
  
    return {  
        'id': book.id,  
        'title': book.title,  
        'rating': book.rating,  
        'numReviews': book.numReviews,  
    }
```

```
permit (principal,  
        action == Action::"GetBook",  
        resource)  
when { resource.owner == principal };
```

```
permit (principal,  
        action == Action::"GetBook",  
        resource)  
when { principal.admin };
```

```
forbid (principal,  
        action == Action::"GetBook",  
        resource)  
unless { principal.multiFactorAuth };
```

ポリシーベースのアクセスコントロール



Fine-grained

リソースとユーザーの組み合わせによる
細かな制御



Real time

プリンシパルやリソースの現在の属性値
に基づくアクセス判断



Scalable

容易にメンテナンス可能なルール
アプリケーションからの独立性



User-managed access (UMA)

ユーザーによるアクセス管理

ポリシー

```
1 permit (  
3   principal in Role::"SalesTeam",  
4   action in [Action::"view", Action::"update"],  
5   resource == Photo::"vacationPhoto94.jpg"  
6 )  
when {  
   resource.accessLevel == "public" &&  
   principal.location == "USA" &&  
   context.request_client_ip == "222.222.222.222"  
10 };
```

Role based

Attribute based

Cedar

FAST, SCALABLE ACCESS CONTROL

Cedar is a language for defining permissions as policies, which describe who should have access to what. It is also a specification for evaluating those policies. Use Cedar policies to control what each user of your application is permitted to do and what resources they may access.

[Do the tutorial](#) [Try it out in playground](#)

May 10, 2023: Amazon Web Services announces the Open-Source release of the Cedar SDK. [Learn more](#)

EXPRESSIVE
Cedar is a simple yet expressive language that is purpose-built to support authorization use cases for common authorization models such

PERFORMANT
Cedar is fast and scalable. The policy structure is designed to be indexed for quick retrieval and to support fast and scalable real-time evaluation,

ANALYZABLE
Cedar is designed for analysis using Automated Reasoning. This enables analyzer tools capable of optimizing your policies and proving that your

[Privacy](#) | [Site Terms](#) | [Cookie Preferences](#) | © 2023, Amazon Web Services, Inc. or its affiliates. All rights reserved.

<https://www.cedarpolicy.com/en>

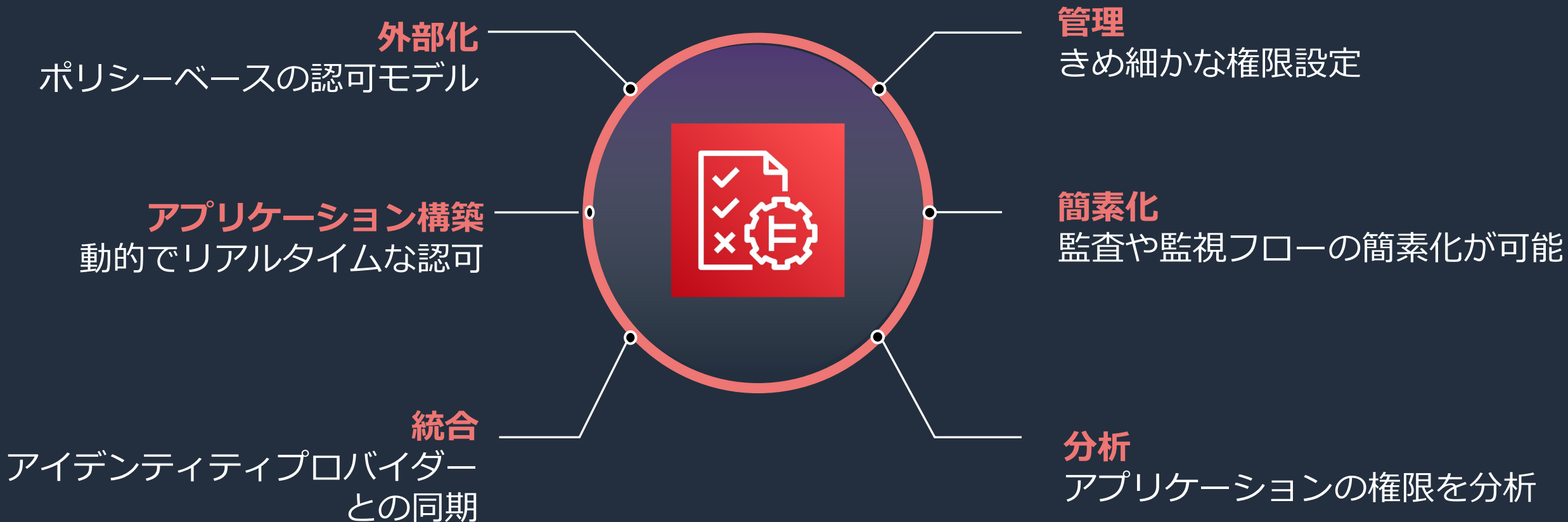
アプリケーションでの独自の認可を実現するためのオープンソース (Apache License 2.0) のポリシー言語と認可エンジン

シンプルでありながら、ロールベースアクセス制御 (RBAC) や属性ベースのアクセス制御 (ABAC) もサポート

AWS Identity and Access Management (IAM) ポリシーとの類似性

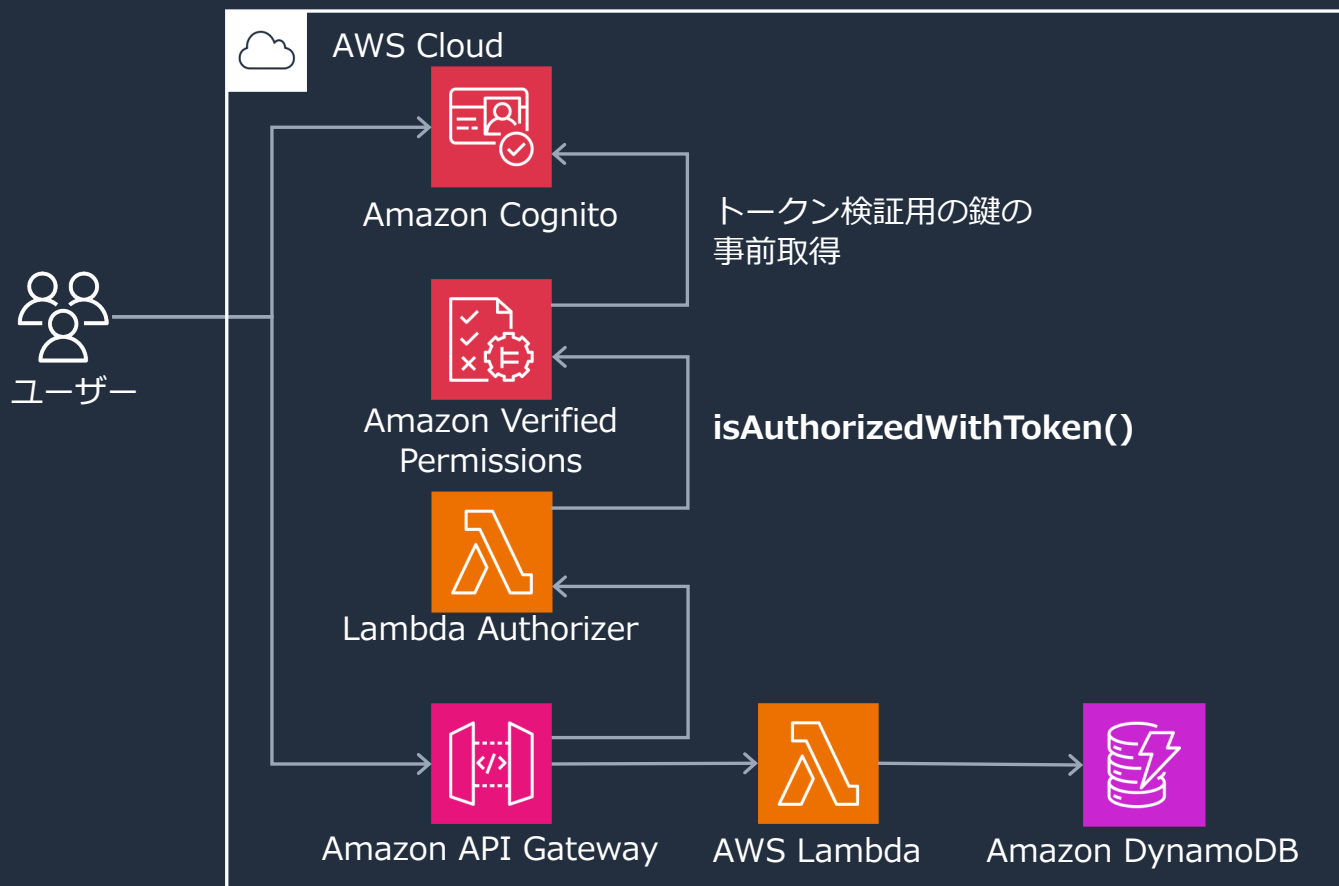
- PARC (Principal, Action, Resource, Condition) を用いたアクセス許可
- デフォルトで拒否
- 明示的な拒否は許可よりも優先される
- ポリシー間に優先順位は存在しない

Amazon Verified Permissions の特徴



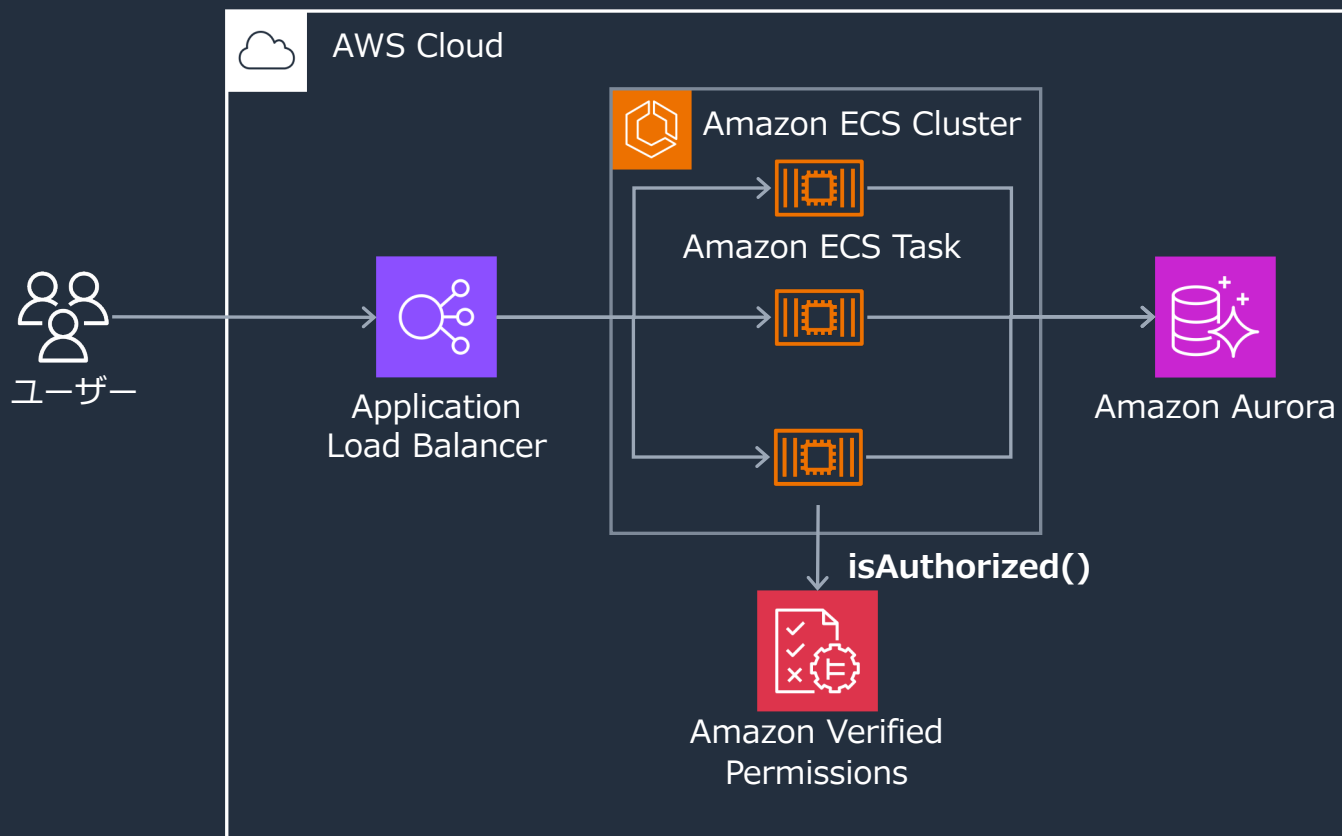
アプリケーションへの導入パターン (1)

Amazon API Gateway でのきめ細やかな認可



アプリケーションへの導入パターン (2)

新規アプリケーションの認可ロジックで Amazon Verified Permissions を利用

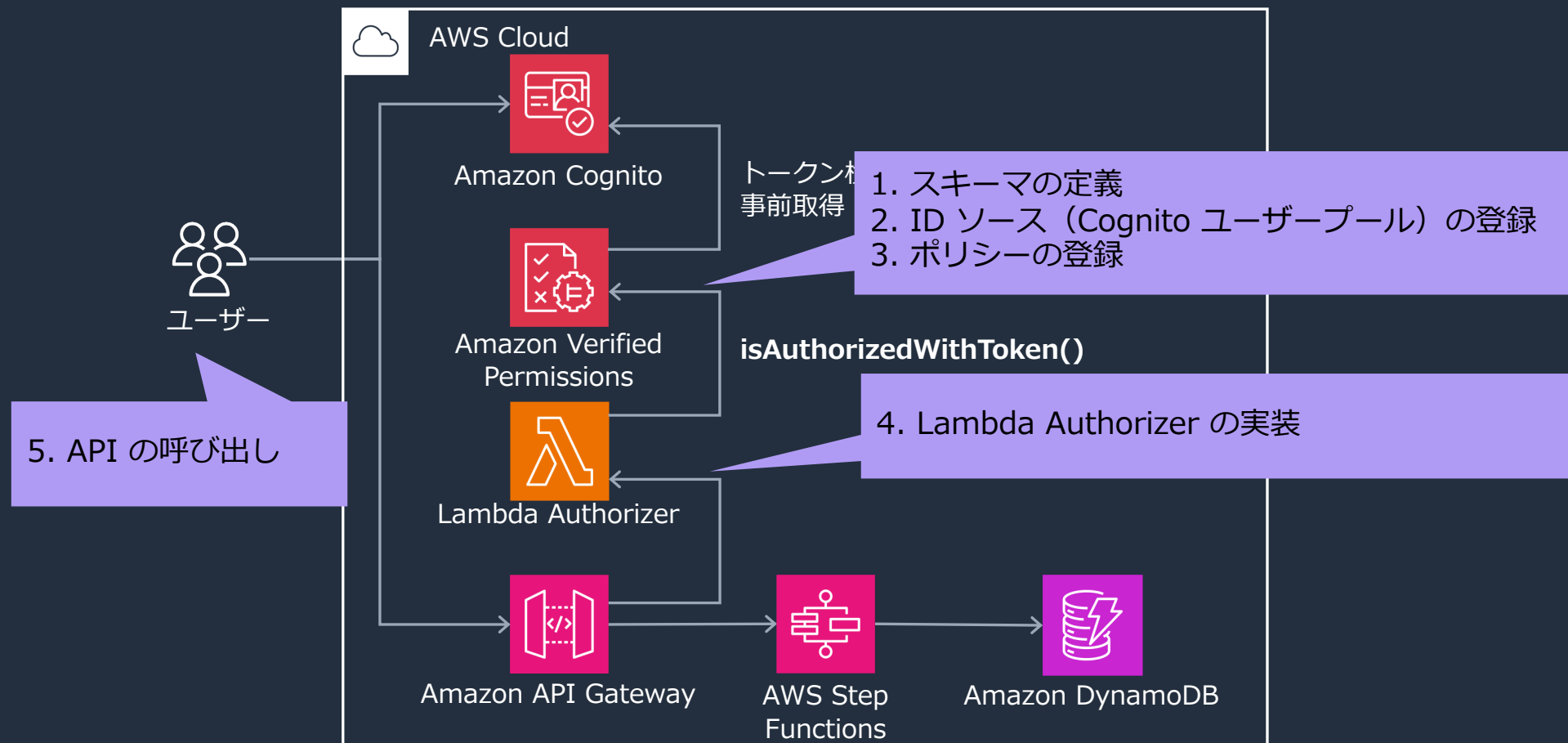


DEMO



DEMO シナリオ

Amazon API Gateway でのきめ細やかな認可



まとめ

Amazon Verified Permissions はアプリケーションのきめ細やかな認可ポリシーを一元管理し、判断結果を返すサービス

Cedar 言語を用いて RBAC や ABAC を含むポリシーを柔軟に表現

アプリケーションの各レイヤーでアクセス制御をシンプルに実装し、ビジネスロジックの変化に素早く対応



Thank you!