参考. ABAC の実装例-Amazon Web Services 編

[1]Amazon Web Services（以下、「AWS」という。）は代表的な Infrastructure as a Service（IaaS）であり、アクセス制御機能をその一部として実装している。本文書では、AWS に対して適用できる属性を用いた ABAC 実装例を紹介する。なお、本ドキュメントは2023年3月段階の仕様である。

1. ユーザーからサービスへの ABAC（サブジェクト側での制御）

この例（図 1）では、サーバー管理を担当する従業員のアカウント名がそれぞれ Red と Blue とし、それぞれが管理するサーバーに対してのみ、サーバーの起動および停止といった処理を行えるよう制御をする。また、本例ではその制御をサブジェクトで実施する。表 1 に一般的なアクセス制御のコンポーネントと AWS の機能をマッピングする。

[11] デフォルトでタグは任意に作成・変更できるため、適切な権限管理を設定する必要がある

図 1

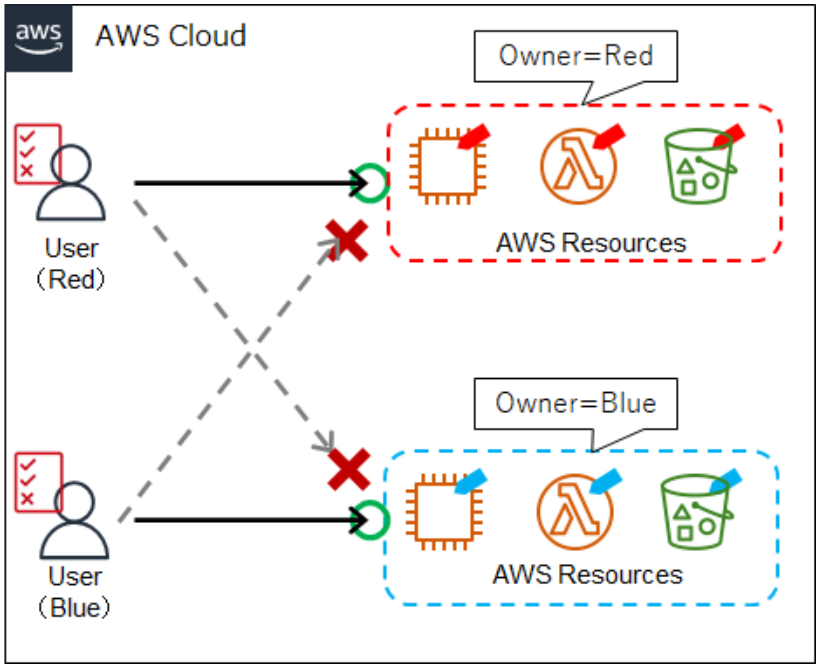

表 1

アクセス制御のコンポーネント	例で該当する機能
ユースケース	サーバーの起動および停止を、サーバー管理者のみに限定する
サブジェクト	AWS IAM User（ユーザー）
オブジェクト	AWS EC2 Instance（サーバー）
属性	AWS IAM User につけられたユーザー名、AWS EC2 Instances に付加的に設定されたタグ
PDP	AWS IAM Policy
PEP	AWS IAM User

具体的には、各 Red と Blue という AWS IAM User に AWS IAM Policy というアクセス制御におけるポリシーを定義する（図 2）。本ポリシーを付与された AWS IAM User は、自身のユーザー名が「Owner」属性の値として設定されたサーバーにのみ、起動および停止する処理を実施できる。AWS では各リソースに対してタグとよばれる情報を定義できる。タグは、キーとオプションという形で構成されており、これを属性と属性

値として扱うことで ABAC を実装できる。そのため、図1に示す通り、AWS EC2 Instance に適切なタグを設定する必要がある。

図 2

```
{
        "Version": "2012-10-17",
        "Statement": [{
                "Effect": "Allow",
                "Action": ["ec2:StartInstances", "ec2:StopInstances"],
                "Resource": "arn:aws:ec2:*:*:instance/*",
                "Condition": {
                        "StringEquals":{
                        "aws:ResourceTag/Owner": "${aws:username}"}
                }
        }]
}
```

2. ユーザーからユーザーへの ABAC（サブジェクト側での制御）

図 3 の例では、特定プロジェクトに関わるアカウントのみが同プロジェクトに所属する別アカウントに対する操作を制御する例である。

図 3

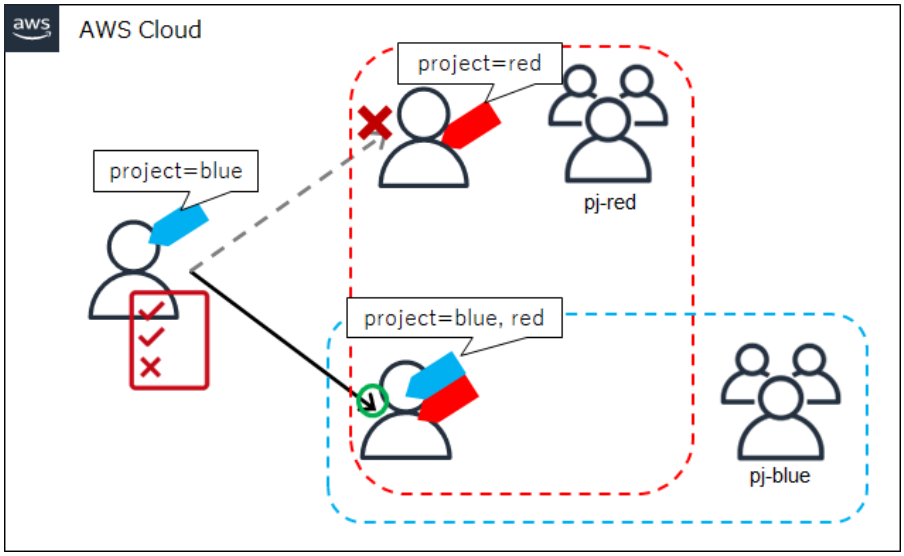

表 2

アクセス制御のコンポーネント	例で該当する機能
ユースケース	同一プロジェクトに所属するユーザー同士のみが互いのアカウント情報を参照可能とする
サブジェクト	AWS IAM User（ユーザー）
オブジェクト	AWS IAM User（ユーザー）
属性	AWS IAM User に付加的に設定された配属先プロジェクト属性
PDP	AWS IAM Policy
PEP	AWS IAM User

AWS IAM User であるユーザー同士がサブジェクトおよびオブジェクトになる例である。これら AWS IAM User には配属されたプロジェクトを示すタグ「Project」が付加されている。図 4 では同一プロジェクトである場合にのみ、AWS IAM User の所属グループを取得することを許可する設定を構成している。

図 4

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Sid": "VisualEditor0",
         "Effect": "Allow",
         "Action": "iam:ListGroupsForUser",
         "Resource": "arn:aws:iam::111111111111:user/*",
         "Condition": {
            "StringLike": {
               "aws:ResourceTag/project": "*${aws:PrincipalTag/project}*"
            }
         }
      },
      {
         "Effect": "Allow",
         "Action": "iam:ListUsers",
         "Resource": "*"
      }
   ]
}
```

3. 処理内の環境情報を使った ABAC（オブジェクト側での制御）

図 5 の例では、特定の環境条件を満たすファイルの保管処理を許可する。具体的にはアップロードされたファイルが指定された暗号鍵を用いて暗号化されているかを確認する。これは ABAC における環境情報を使う例である。また、その際のアクセス制御はリソース側にて実施される。

図 5

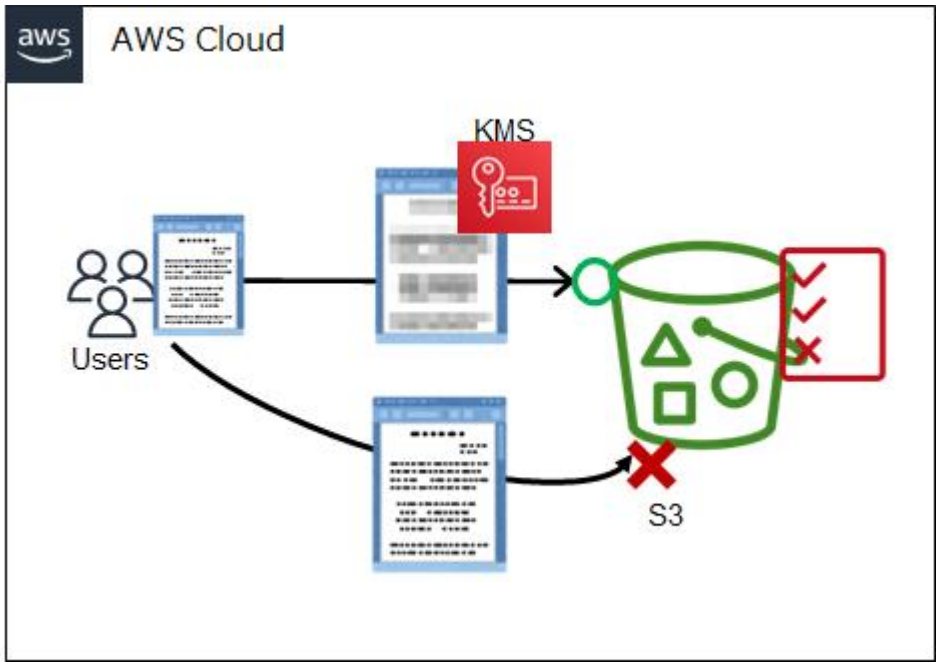

表 3

アクセス制御のコンポーネント	例で該当する機能
ユースケース	ファイルストレージへのファイル保管時に、指定した暗号鍵による暗号化を強制したい
サブジェクト	AWS IAM User（ユーザー）
オブジェクト	AWS S3 Bucket（ファイルストレージ）
属性	ファイル保管処理に関する環境情報。特に暗号化に利用するために指定される暗号鍵の識別子
PDP	AWS S3 Bucket Policy
PEP	AWS S3 Bucket

AWS のファイルストレージサービスであるS3 Bucket では、Bucket Policy を設定することで S3 Bucket に対するアクセスを制御することが可能である。図 6 では、S3 Bucket に対するアップロードを制御するポリシーである。多くのコンプライアンス要件で見られるファイル保管時の暗号化を、図 6 のポリシーで強制的に準拠できる。具体的には、アップロード時に特定の暗号鍵（AWS KMS）を指定していない限り、保管処理に失敗する。これは実際の処理におけるサブジェクトとオブジェクトそのものの属性には着目しておらず、その処理をとりまく環境情報である暗号化状況を使った制御である。

図 6

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Effect": "Allow",
         "Principal": {
            "AWS": "arn:aws:iam::111111111111:user/*"
         },
         "Action": "s3:PutObject",
         "Resource": "arn:aws:s3:::digital-cho-test-s3/*",
         "Condition": {
            "StringEquals": {
               "s3:x-amz-server-side-encryption-aws-kms-key-id": “xxxx-xxxx-xxxx"
            }
         }
      }
   ]
}
```

4. 外部 IdP 上ユーザーから AWS サービスへの ABAC（SAML の活用）

クラウド・バイ・デフォルト原則を前提にする業務環境では、より強固なアクセス制御を実現するうえで、ID フェデレーションとアサーションが重要になる。そのような要件を効率的に実現するために、外部の ID プロバイダー（IdP）から連携された属性情報を別システムでアクセス制御に活用する例が図 7 となる。

この構成では、属性情報や認証情報等の機械判読可能な情報が、異なる管理領域にあるシステム間で授受される。そのため、予め両システム間で信頼関係を構築しつつ、属性スキーマが標準化されていなければならない。

図 7

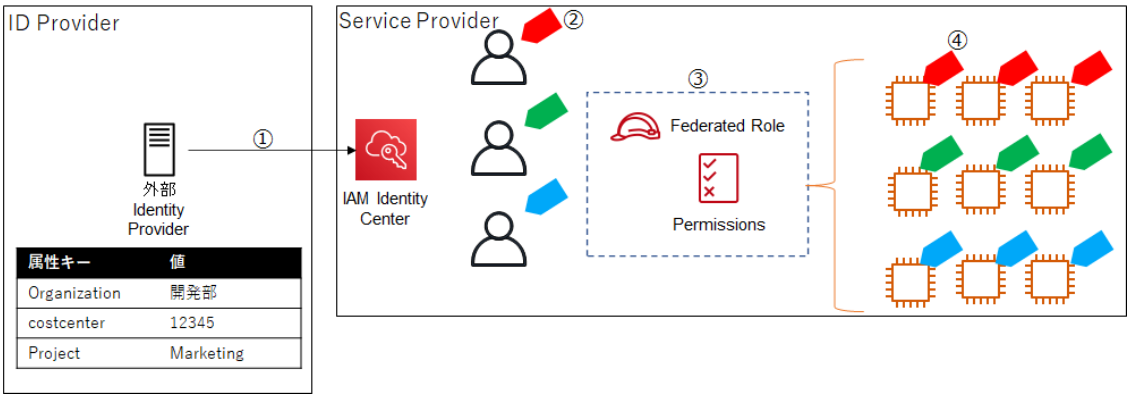

表 4

アクセス制御のコンポーネント	例で該当する機能
ユースケース	強固な ID フェデレーション及びアサーションによる、効率的なアクセス制御をする
サブジェクト	①外部IdP のアカウント ②,③ AWS Identity Center（AWS 側の IdP アカウント） ④AWS IAM Role
オブジェクト	①AWS Identity Center（AWS 側の IdP アカウント） ②,③AWS IAM Role ④AWS の各種リソース
属性	外部 IdP の属性、AWS の各種リソースに設定されたタグ

PDP	①AWS Identity Center の Trust Relationship 設定及び Permission ②,③AWS Identity Center の割り当て設定および AWS IAM Role の Trust Relationship 設定 ④AWS IAM Policy または各種 AWS リソースの Policy、あるいはその組み合わせ
PEP	① AWS Identity Center ②,③AWS Identity Center および AWS IAM Role ④AWS IAM Role あるいは AWS リソース、あるいはその組み合わせ

まず①にて、外部 IdP のアカウントから AWS の認証統合機能である IAM Identity Center に SAML Single-Sign On をする。この際、図 6 の SAML アサーションが AWS に連携される。具体的には、属性名「https://aws.amazon.com/SAML/Attributes/PrincipalTag:Project」および「https://aws.amazon.com/SAML/Attributes/PrincipalTag:CostCenter」の値として「Marketing」と「12345」が設定さている。これらが AWS Identity Center を介して(②)、AWS アカウント内のセッションにおけるタグ情報として連携される。このタグは AWS IAM Role のタグとして引き継がれ(③)、その Role に紐づけられた図 7 の AWS IAM Policy と照合される。この AWS IAM Policy は、外部 IdP のけるアカウントが属していたコストセンターに管理されている AWS EC2 へのアクションのみが許可されることを示している。

図 8

```
<Attribute Name="https://aws.amazon.com/SAML/Attributes/PrincipalTag:Project">
  <AttributeValue>Marketing</AttributeValue>
</Attribute>
<Attribute Name="https://aws.amazon.com/SAML/Attributes/PrincipalTag:CostCenter">
  <AttributeValue>12345</AttributeValue>
</Attribute>
```

図 9

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ec2:DescribeInstances"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "ec2:StartInstances",
        "ec2:StopInstances"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "ec2:ResourceTag/costcenter":"${aws:PrincipalTag/costcenter}"
        }
      }
    }
  ]
}
```

5. 外部サービスから AWS サービスへの ABAC（OIDC の活用）

アクセス制御はユーザーアカウントが実行する処理に限定されない。本例はサービス間での処理における ABAC 適用事例となる。具体的には、GitHub Action という Continuous Integration（CI） / Continuous Delivery（CD）サービスと AWS サービス間のアクセス制御となる。AWS を含む Web API による管理機能が基本となる IaaS/PaaS では Terraform や CloudFormation といったツール（Infrastructure as Code）を管理手法として採用することで、多くの構成・変更業務の自動化が可能となる。これらのツールで記述されたファイルはソースコードとして GitHub などのバージョン管理ツールに格納される。ファイルへの変更は適切な承認を経てビルドされ、インフラ環境にデリバリされる(CI/CD)。この例では GitHub Action がそれらの機能を担ったサブジェクトとして、オブジェクトである AWS の API およびリソースに処理を試みる。その際のアクセス制御は、GitHub 上の OpenID Connect（OIDC） Provider によって発行された属性情報によって行われる。なお、本例は「4. 外部 IdP 上ユーザーから AWS サービスへの ABAC（SAML の活用）」と同じく、異なる管理領域にあるシステム間の処理になるため、事前に信頼関係を結び、OIDCによる連携が可能でなければならない。

図 10

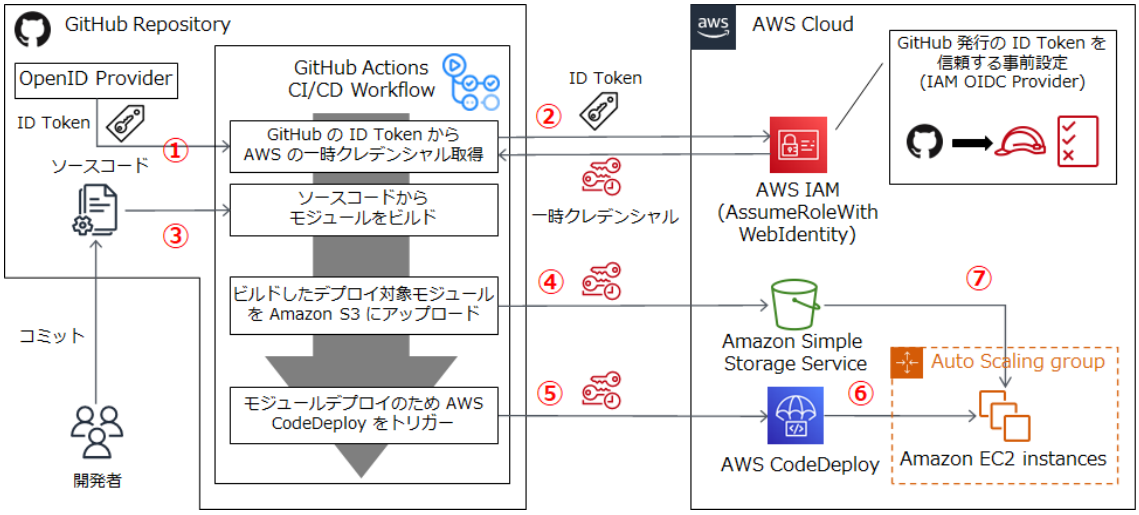

表 5

アクセス制御のコンポーネント	例で該当する機能
ユースケース	CICD サービスから AWS リソースの管理作業を自動で実施する
サブジェクト	外部サービス（GitHub Action）

オブジェクト	AWS IAM Role（②）、各種 AWS リソース（⑤, ⑥）
属性	サービスが管理する IdP によって発行された ID Token のクレーム
PDP	AWS IAM　及び AWS IAM Policy
PEP	AWS IAM

GitHub Action の挙動は workflow ファイル（図 11）内で定義される。この workflow が実行時に GitHub の OpenID Connect に準拠した IdP に ID トークンの発行を依頼する（図 11-①）。GitHub Action の workflow は発行された ID トークン（図 12）を AWS に送付し、特定の AWS IAM Role として処理するための一次トークンの発行を依頼する（図 11-②）。AWS は ID トークンの sub 属性とその値を読取り、AWS IAM Role 側の信頼関係設定のポリシー（図 13）と照合する。照合結果に問題がなければ、AWS リソースを操作する API リクエストに必要な一時的なクレデンシャルが発行される。
GitHub Action はそれを用いて、AWS IAM Role に許可された範囲の権限で、AWS の各種リソースに対して処理を実行する（図 11-④、図 11-⑤）。

図 11

```yaml
name: deliver IaC

on:
  push:
    tags:
      - v**
permissions:
  id-token: write
  contents: read
jobs:
  apply:
    name: build and push
    runs-on: ubuntu-latest
    steps:
      - name: Checkout
        uses: actions/checkout@v3
      - name: Configure aws credentials
        uses: aws-actions/configure-aws-credentials@v1-node16
        with:
          aws-region: ap-northeast-1
          role-to-assume: arn:aws:iam::xxxxxxxx:role/infra-delivery-role
          role-session-name: tf-apply-github-action
      - name: Deliver
        run: |
           terraform pan -out=/tmp/tfplan && ¥
           aws s3 cp /tmp/tfplan s3://delivery-bucket
```

図 12

GitHub が生成する ID Token

```json
{
  "jti": "6cd92056-fccc-407e-a8db-2d7a5e10f1a2",
  "sub": "repo:example/github-action:ref:refs/heads/main",
  "aud": "https://github.com/example/github-action",
  "ref": "refs/heads/main",
  "sha": "284d6fb79903f2fe545f608f4d32cb97179bcc01",
  "repository": "example/github-action",
  "repository_owner": "example",
  "actor": "example",
  (snip...)
  "iss": "https://vstoken.actions.githubusercontent.com",
  "exp": 1669726800,
  "iat": 1669723200
}
```

図 13

対応するIAMポリシー(信頼ポリシー)例

```
{
 "Version": "2008-10-17",
 "Statement": [{
  "Effect": "Allow",
  "Principal": {
   "Federated": "{IAM_OIDC_Provider_ARN}"
  },
  "Action": "sts:AssumeRoleWithWebIdentity",
  "Condition": {
   "StringLike": {
    "vstoken.actions.githubusercontent.com:sub": "repo:
example/*:ref:refs/heads/main"
   }
  }
 }]
}
```

組織exampleの全てのリポジトリのmain
ブランチからの GitHub Actions 連携を許可

6. 参考資料

タグを使用した AWS リソースへのアクセスの制御:
https://docs.aws.amazon.com/ja_jp/IAM/latest/UserGuide/access_tags.html

Using server-side encryption with Amazon S3-managed encryption keys (SSE-S3)
:
https://docs.aws.amazon.com/AmazonS3/latest/userguide/UsingServerSideEncryption.html

認証レスポンスの SAML アサーションを設定する:
https://docs.aws.amazon.com/ja_jp/IAM/latest/UserGuide/id_roles_providers_create_saml_assertions.html

Using environments for deployment:
https://docs.github.com/en/actions/deployment/targeting-different-environments/using-environments-for-deployment