AWSインフラ命名規約

フューチャー株式会社

Table of Contents

本コーディング規約は、世の中のシステム開発プロジェクトのために無償で提供致します。 ただし、掲載内容および利用に際して発生した問題、それに伴う損害については、フューチャー株式会社は一切の責務を負わないものとします。 また、掲載している情報は予告なく変更することがございますので、あらかじめご了承下さい。

# 免責事項

::: warning 有志で作成したドキュメントである

* フューチャーアーキテクトには多様なプロジェクトが存在し、それぞれの状況に合わせた開発手法が採用されている。本規約はフューチャーアーキテクトの全ての部署／プロジェクトで利用されているわけではなく、有志が観点を持ち寄って新たに整理したものである。相容れない部分があればその領域を書き換えて利用することを想定している

:::

# 前提条件

* 開発チームが 3 ～ 30 名程度で構築する規模での利用を想定している
* 本規約をそのままプロジェクトに導入することを推奨する
  + そのままの導入ができない場合は、一部を抜粋、拡張して用いられることを想定している
* AWS の構成はマネージドサービスを活用するベストプラクティスに従うものとする
  + 例えば、1 台の EC2 上で DB とアプリのように複数のサービスを稼働させるといった構成は考慮しない
* 一部のリージョンでのみ利用可能な機能は想定していない
  + 例えば、[AWS Local Zones](https://aws.amazon.com/jp/about-aws/global-infrastructure/localzones/) は考慮しない

# 名前の構成要素

各リソースの名前に用いる要素を次の一覧に示す。

| Category | Item | Name | Usage |
| --- | --- | --- | --- |
| Common | {env} | 環境 | 環境の区別 |
|  | {product} | 製品名 | 構築する製品名またはシステム名。稼働するマイクロサービス名もこれに当たる |
|  | {role} | 役割 | 役割を示す。場合によっては具体的な製品名 postgres, jenkins などを指定する |
|  | {usage} | 用途 | 利用目的やリソースの動作 (action) を示す。user\_master, fileupload など識別したい値を指定する |
|  | {target} | 対象 | 操作の対象。usage が複数の対象があり区別したいときに利用する。 |
| Network | {region} | リージョン | [リージョンコード](https://docs.aws.amazon.com/ja_jp/general/latest/gr/rande.html) の略称を用いる |
|  | {az} | アベイラビリティーゾーン | マルチ AZ 構成などで、明示的に AZ を意識する場合に用いる |
|  | {access} | アクセス修飾子 | access modifier. ネットワークでの public, private を区別したいときに利用する |
|  | {permission} | 権限 | allow または deny を指定する。Security Group での利用を想定 |
| Organization | {company} | 会社名 | 会社の特定に利用。複数の会社による構築や、運用に複数社関わる場合などに必要となる |
|  | {project} | プロジェクト | プロジェクト制でプロダクトを開発する際のプロジェクト名または、プロジェクトコード |

## 環境 ({env})

ソフトウェア開発では複数の環境を用意し、dev, stg, prod などの名前をつけて互いに完全に分離・区別する運用を行うことが多い。そういった環境分離のために AWS インフラは次のいずれか、もしくは組み合わせで設計される。

1. 環境単位で AWS アカウントを作成する
2. 環境単位で AWS リージョンを分ける
3. 命名で分ける

いずれの方法でも、 **各リソース名に環境名を付与することを推奨** する。冗長な命名となる場合もあるが、以下が理由である。

* 同一 AWS アカウントかつ同一リージョン内には、同じ名前のリソースを作成できない
* AWS リソース名のみで環境を特定できるようにする事で、誤った環境のリソースを操作してしまうミスを低減する
* メンバー間の認識齟齬を無くし生産性を高める
  + チームメンバーなどの問い合わせやトラブルシュートの際に、リソース名のみでどの環境にあるか素早く判断できるようにする
  + メンバーの役割によっては AWS アカウント構成を完全には理解できていない可能性がある

### 環境識別子

主要な環境名と識別子 (Identifier) は以下である。AWS リソースの命名には識別子を用いる。

| Name | Identifier | Memo |
| --- | --- | --- |
| Production | prod | エンドユーザーが使う環境、本番運用環境 |
| Staging | stg | 本番と同じ構成でテストするための環境 |
| User Acceptance Test | uat | ユーザーがシステムのレビュー、または操作を学習するための環境 |
| Performance Test | perf | 性能検証を行うための環境 |
| Development | dev | 開発チームが開発するための環境 |
| Local | local | ローカル環境 |

prod についてはよく用いる dev, stg と見間違えを防ぐため 4 文字にしている。

[デプロイメント環境](https://ja.wikipedia.org/wiki/%E5%B1%95%E9%96%8B%E7%92%B0%E5%A2%83) の考え方では、User Acceptance Test 環境を単にテスト環境 (Test) 呼ぶが、テストという単語は汎用的であるため複数の環境にあてはまる。したがって容易に認識齟齬が生じるため本規約では非推奨とする。

名前には必ず識別子を用いる。環境名をそのまま利用しない (例: production-example-s3bucket とは命名しない)。

理由:

* AWS リソースによっては名前の文字数の制限が厳しい (例: ELB は最大 32 文字である)
* AWS コンソールや設計ドキュメントなどで一覧化した場合に見切れる可能性を減らして可読性を高める
* 識別子に採用した単語は一般的に用いられている略称である
* その他の環境についても、環境識別子の数は通常そこまで多くならないず、またよく用いられるため、利用者にとっての認識負荷は少なく覚えるコストも低い

### 同一目的の複数環境

同一目的の環境が複数必要な場合は、識別子の末尾に連番をつける。

例:

* dev1, dev2, dev3
* stg1, stg2

## 役割 ({role})

アプリケーションを構成する要素には役割がある。それを AWS リソース名に含めることで、開発者の理解を助け、操作ミスを低減する。

主要なロール名と識別子は以下である。

| Name | Identifier | Memo |
| --- | --- | --- |
| Web Server | web | apache や nginx などの Web サーバとしての役割 |
| Web Application | app | Web アプリケーションとしての役割 |
| Web API | api | HTTP(s) API を提供する |
| Job | job | 時間やある特定のイベントをもとにバックグラウンドの処理（バッチ処理など）を行う |
| I/F | if | ファイル入出力を行う |
| DB | db | データベース |
| Cache | cache | キャッシュ |
| CI/CD | ci | CI/CD サーバ |

名前を一般化せず、プロダクト名をそのまま利用しても問題ない。例えば、Web アプリサーバに tomcat、CI/CD サーバに jenkins といった名称を使っても良い。

## 用途 ({usage})

利用目的やリソースの動作 (action) を示す。user\_master, fileupload といった形式や、認証(auth)や BFF（Backend For Frontend）など。

役割 ({role}) と合わせてリソースが一意に特定できる名称を設定する。

## リージョン ({region})

マルチリージョン構成を取り、リージョンを意識する必要のある場合に利用する。[リージョンコード](https://docs.aws.amazon.com/ja_jp/general/latest/gr/rande.html) そのものではなく略称を識別子として用いる。

| Name | Region Code | Identifier |
| --- | --- | --- |
| 米国東部 (バージニア北部) | us-east-1 | ue1 |
| 米国東部 (オハイオ) | us-east-2 | ue2 |
| 米国西部 (北カリフォルニア) | us-west-1 | uw1 |
| 米国西部 (オレゴン) | us-west-2 | uw2 |
| アジアパシフィック (東京) | ap-northeast-1 | an1 |
| アジアパシフィック (ソウル) | ap-northeast-2 | an2 |
| アジアパシフィック (大阪) | ap-northeast-3 | an3 |
| アジアパシフィック (シンガポール) | ap-southeast-1 | as1 |

シングルリージョン構成または、リージョン間のリソースの関係が疎である場合はリージョン識別子を付与しない。

## アベイラビリティゾーン ({az})

AZ 名にはリージョンコードを含めず、末尾のアルファベットだけとする。

| AZ ID | Identifier |
| --- | --- |
| ap-northeast-1a | a |
| ap-northeast-1c | c |
| ap-northeast-1d | d |

* 利用可能な文字: [a-d]{1}

## アクセス修飾子 ({access})

VPC のサブネットは、パブリックサブネットの場合インターネットに直接アクセスできる。パブリックサブネットを区別したい場合はリソース名にアクセス修飾子を付与する。

| Name | Identifier |
| --- | --- |
| パブリックサブネット | public |
| プライベートサブネット | private |

# 全体ポリシー

## 命名規約

次のように各要素を使ってケバブケース (kebab-case) で命名する。パスカルケース (PascalCase) やスネークケース (snake\_case) は利用しない。なお、サービス名自体にパスカルケースを用いることは許容する

# 命名規約の基本形  
{env}-{product}-{role}-{usage}

理由:

* ほぼ全ての AWS サービスではリソース名にハイフンを許容する。一方で、アンダースコアを許容しない WebACL のようなサービスがある
* 環境名、サービス名などの単位で区切りを明確にできる

## 利用可能な文字

利用する文字は、半角英数字とハイフンに限定する。また、 **小文字を推奨** する。

* 推奨: [a-z0-9\-]+

また、先頭文字には半角英字を用い (ハイフン、数値を先頭にしない)、ハイフンは 2 文字以上連続させないこととする。

## AWS サービス名を含めない

リソース名に AWS サービス名を含めない。

良い例：

stg-fuga-web-fileupload  
stg-fuga-web-fileupload

悪い例：

stg-fuga-web-fileupload-s3  
stg-fuga-web-fileupload-bucket

理由:

* AWS コンソールで見たときにどの AWS サービスのリソースを見ているか自明である
* [Terraform の命名規則](https://www.terraform-best-practices.com/naming#resource-and-data-source-arguments) にリソース名を繰り返さないという記載があり、整合性を持たせるため

**Resource and data source arguments** Do not repeat resource type in resource name (not partially, nor completely):

ただし、VPC エンドポイントやセキュリティグループのように、どの AWS サービスの何で利用されているかを示す場合には利用することがある。

## プロジェクト名を含めない

プロジェクト制を取っている場合、その開発チームの持ち物であることを示すためプロジェクト名をリソース名に含めたくなるが非推奨である。

理由:

* 必ずしも開発しているプロダクトと、プロジェクトの粒度・ライフサイクルは一致しない
* プロジェクトが解散すると管理主管が曖昧になる

プロジェクト名の替わりにプロダクト名を含めることとする。

## マルチクラウドを考慮し、aws 識別子を追加するかどうか

AWS だけではなく、Azure や GCP などを組み合わせたマルチクラウド運用を行っている、あるいは行う予定がある場合を考慮し、リソース名に aws といったプレフィックス/サフィックスを付与する考えもある。

本規約では、aws キーワードをリソース名に含めることは非推奨とする。

理由:

* 同一 product を異なるクラウドサービスで運用することは稀
* 一部のサービス (例えば DWH のみ Google BigQuery を利用するようなケース) だけの使用であれば、{usage} で区別すれば十分である

# サービス別の命名規約

サービスによって異なる命名規約と例を記載する。

以下ではプロダクト名を fuga とした場合の例をあげる。

## VPC

VPC に関わるリソースの命名について記載する。

| Resource Name | Naming Convention | Example | Note |
| --- | --- | --- | --- |
| VPC | {env}-{product} | stg-fuga |  |
| Subnet | {env}-{product}-{access}-{az} | stg-fuga-public-a | AZ: どこのゾーンかを識別するため |
| EIP | {env}-{product}-{usage} | stg-fuga-nat |  |
| Route Table | {env}-{product}-{access} | stg-fuga-public |  |
| Internet Gateway | {env}-{product} | stg-fuga |  |
| NAT Gateway | {env}-{product} | stg-fuga |  |
| Endpoint | {env}-{product}-{aws\_service} | stg-fuga-s3 | 様々なサービスが利用するため AWS サービス名を含めている |
| Security Group | {env}-{product}-{aws\_service}-{usage} | stg-fuga-ec2-bastion | 様々なサービスが利用するため AWS サービス名を含めている |

## API Gateway

AWS上の命名制約

* ドキュメントは確認できなかったら、リソース名は 1024 文字まで指定可能。システム上は一意である必要はない

API Gateway は [全体ポリシーの命名規約](#命名規約) に則る。管理上、一意となるように命名する

# 命名規約の基本形  
{env}-{product}-{role}-{usage}-{access}  
  
# 例  
stg-fuga-web-portal-private  
stg-fuga-web-fileupload-public

* API Gateway には複数の機能種別 (REST, HTTP) が存在するが、命名には含めない
* private/public を名前に含めることで、public は認証が入っているかなどをチェックできる

## EC2

インスタンス名の制限=タグの制限のため、名前は [Amazon EC2 リソースのタグ付け](https://docs.aws.amazon.com/ja_jp/AWSEC2/latest/UserGuide/Using_Tags.html) に従う必要がある。

# 命名規約の基本形  
{env}-{product}-{role}  
  
# 例  
stg-fuga-web

オートスケーリング、オートヒーリング構成をする場合にどこの AZ に配置するかを意識させないため、リソース名に AZ は基本的に含めない。そのような構成をしないという方針は、アンチパターンのため構成を見直すべきと考える。

## LB

AWS上の命名制約

* 最大文字数: 32
* 英数字とハイフンが利用可能 (先頭、末尾にハイフン不可)
* ALB と NLB のセット内で一意である必要がある
* 参考: [Application Load Balancer にタグを付ける](https://docs.aws.amazon.com/ja_jp/elasticloadbalancing/latest/application/load-balancer-tags.html)

LB には ALB/NLB/CLB などの種類があるが、いずれも以下の命名規約に従う。また、Internal LB に関しては、{usage} 部に含める。

# 命名規約  
{env}-{product}-{role}-{usage}-{access}  
  
# 例  
stg-fuga-web-api-public

ターゲットグループ名は、基本的には LB と同じである。

ただし、Blue/Green デプロイを行う場合は、ターゲットグループ名をユニークにし、どちら (Blue/Green) に所属しているかをわかるようにする。

# Target group name (Blue/Green) の命名規約  
{env}-{product}-{role}-{usage}-{access}-blue  
  
# 例  
stg-fuga-web-public-blue

## ECS

AWS上の命名制約

* クラスター
  + 最大文字数: 255
  + 利用可能文字種: A-z, 0-9, -, \_
  + その他制約: 先頭の文字は A-z のみ利用可能
* サービス、タスク定義
  + 最大文字数: 255
  + 利用可能文字種: A-z, 0-9, -, \_

ECS の命名規約は以下のとおりである

# クラスターの命名規約  
{env}-{product}  
  
# 例  
stg-fuga  
  
# サービスの命名規約  
{env}-{product}-{role}-{usage}  
  
# 例  
stg-fuga-api-auth  
stg-fuga-web-frontend  
  
# タスク定義の命名規約  
{env}-{product}-{role}-{usage}  
  
# 例  
stg-fuga-batch-import-address

## Lambda

### Lambda Function

AWS上の命名制約

[CreateFunction](https://docs.aws.amazon.com/ja_jp/lambda/latest/dg/API_CreateFunction.html) によると以下の制約である。

* 1 ～ 64 文字
* 利用可能文字: [a-zA-Z0-9-\_]+

Lambda は運用を経てリソース数が増えやすいサービスの一つである。そのため個別の機能名の前に {role} を含めてグルーピングしやすい名前にする。

# 命名規約  
{env}-{product}-{role}-{usage}  
  
# 例  
stg-fuga-import-userprofile  
stg-fuga-job-checkconsistency  
stg-fuga-report-successrate

もし、Scatter-Gather パターンを用いる場合は次のようにサフィックスに追加して区別する。

# 命名規約  
{env}-{product}-{role}-{usage}-scatter  
{env}-{product}-{role}-{usage}-segment  
{env}-{product}-{role}-{usage}-gather

### Lambda Layer

AWS上の命名制約

* 1 ～ 64 文字
* 利用可能文字: [a-zA-Z0-9-\_]+

Lambda Layers は実行環境が重要であるため、 {runtime} で言語バージョンを指定する。

# 命名規約  
{env}-{product}-{runtime}-{usage}  
  
# 例  
stg-fuga-python310-auth  
stg-fuga-nodejs18-frontend

## RDS/Aurora

AWS上の命名制約

[Amazon RDS の命名に関する制約](https://docs.aws.amazon.com/ja_jp/AmazonRDS/latest/UserGuide/CHAP_Limits.html#RDS_Limits.Constraints) によると以下の制約である。

* 1 ～ 63 個の英数字またはハイフンを使用する必要があります
* 1 字目は文字である必要があります
* 文字列の最後にハイフンを使用したり、ハイフンを 2 つ続けて使用したりすることはできません
* 1 つの AWS アカウント、1 つの AWS リージョンにつき、すべての DB インスタンスにおいて一意である必要があります

# クラスターの命名規約  
{env}-{product}-{role}  
  
# 例  
stg-fuga-auth  
  
# インスタンスの命名規約  
{env}-{product}-{role}-{serial}  
  
# 例  
stg-fuga-auth-01  
  
# DBパラメータグループの命名規約  
{env}-{product}-{role}  
{env}-{product}  
  
# 手動スナップショットの命名規約  
{env}-{product}-{role}-{yyyy}-{mm}-{dd}

AZ は含めない。

理由:

* RDS はマルチ AZ 構成をとることが推奨のため
* Aurora は自動でマルチ AZ 構成をとっているため

DB パラメータグループは、role 単位での設定を推奨する。product を跨いでの設定は行わない。

* 設定値の変更はパラメータグループを適用している全ての DB に変更が反映される。同じパラメータグループを複数の DB で共有し過ぎることにより意図しない DB にまで設定変更が反映されるリスクを避けるため

クラスター/インスタンスに適用する IAM ロール

* [IAM ロール](#iam-ロール) を参照

DB サブネット

* [VPC](#vpc) のサブネットを参照

## DynamoDB

AWS上の命名制約

[Amazon DynamoDB でサポートされるデータ型と命名規則](https://docs.aws.amazon.com/ja_jp/amazondynamodb/latest/developerguide/HowItWorks.NamingRulesDataTypes.html) によると以下の制約である。

* すべての名前は UTF-8 を使用してエンコードする必要があり、大文字と小文字が区別される
* 3 ～ 255 文字
* 利用可能な文字: [a-zA-Z0-9\_.-]+

DynamoDB のテーブル名には、環境、プロダクト名、用途を用いる。データは長く残り、かつ変更しにくいため会社名などの変化しやすい項目は含めない。

# DynamoDB の命名規約  
{env}-{product}-{usage}  
  
# 例  
stg-fuga-user  
stg-fuga-user-accesslog

なお、インデックス名は idx-1, idx-2 のような連番での管理を推奨する。RDB とは異なりアカウント単位での一意性は不要なため、テーブル名は含めなくても良いため、 idx\_{テーブル名}\_{連番} としなくても良い。DynamoDB は 最大で 20 のグローバルセカンダリインデックス を持つことができるが、インデックスの数は最小限に抑えることが鉄則であるため、0 埋めしない。ただし、要件上どうしても多用が避けられないことが判明している場合は idx-01, idx-02 と 0 埋めする。

## S3 Bucket

AWS上の命名制約

[Amazon S3 バケットの命名要件](https://docs.aws.amazon.com/ja_jp/awscloudtrail/latest/userguide/cloudtrail-s3-bucket-naming-requirements.html) によると以下の制約である。

* 1 ～ 63 文字
* 半角英数字、ピリオド、ダッシュのみを使用可能
* バケット名の各ラベルは、小文字または数字で始まっている必要がある
* バケット名では、アンダースコア、末尾のダッシュ、連続するピリオド、隣接するピリオドとダッシュは使用できない
* バケット名を IP アドレス (198.51.100.24) として書式設定することはできない

S3 は非常に多くの用途で用いることがあるため、利用形態に応じて規則を変えて対応する。

# 通常の命名規約  
{env}-{product}-{use}  
  
# 例  
stg-fuga-fileupload  
  
# ログを保管するバケットの命名規約  
{env}-{product}-{service}-logs  
  
# 例  
stg-fuga-alb-logs  
  
# データ授受で利用する場合の命名規約  
{env}-{product}-{use}-{dest}-if  
  
# 例  
stg-fuga-userinfo-fis-if

## Kinesis Data Streams

AWS上の命名制約

[CreateStream](https://docs.aws.amazon.com/kinesis/latest/APIReference/API_CreateStream.html#API_CreateStream_RequestSyntax) によると以下の制約である。

* 異なる AWS アカウントであれば同名が許容
* 異なるリージョンであれば同名が許容
* 1 ～ 128 文字
* [a-zA-Z0-9\_.-]+

IoT のセンシングを始めとしたイベントデータの場合は、次の命名規約を用いる。{role} には import や export など、どのような処理を行うかを規定する。

設計によっては、データ種別 (スキーマ) 毎に分離することもあるため、デバイス名やセンサー名などの発生源の名前を持たせる。

# 命名規約  
{env}-{product}-{role}-{usage}-{schema}  
  
# 例  
stg-fuga-import-iotsensor-devicetype  
stg-fuga-import-iotsensor-toggle

ジョブキューとして用いる場合は、どのジョブを利用するかが重要であるため、呼び出し用であることが明確になるように命名する。

# 命名規約  
{env}-{product}-call-{呼び出したいジョブ名}  
  
# 例  
stg-fuga-call-job-arrival-check

## SQS

AWS上の命名制約

[Amazon SQS キューとメッセージの識別子](https://docs.aws.amazon.com/ja_jp/AWSSimpleQueueService/latest/SQSDeveloperGuide/sqs-queue-message-identifiers.html) によると以下の制約である。

* 最大 80 文字
* [a-zA-Z0-9\_-]
* FIFO キューの場合には .fifo のサフィックスで終わる必要がある
* 1 つの AWS アカウント、1 つの AWS リージョンにつき一意である必要がある

1 つのキューに対し、複数のプロデューサー、コンシューマーを取りうるため、プロデューサー、コンシューマーを名前に含めることは推奨しない。

# 標準キューの命名規約  
{env}-{company}-{product}-{usage}  
  
# 例  
stg-future-fuga-processresult  
  
# FIFOキューの命名規約  
{env}-{company}-{product}-{usage}.fifo  
  
# 例  
stg-future-fuga-processresult.fifo

## Event Bridge Rule

AWS上の命名制約

* 最大 64 文字
* 数字、小文字/大文字、 . (ピリオド)、 - (ハイフン)、 \_ (アンダーバー) が使用可能
* 同じリージョン内および同じイベントバス上の別のルールと同じ名前を付けることは不可

# 命名規約  
{env}-{product}-{usage}-{source}-{target}  
  
# 例  
stg-fuga-deploy-s3-codepipeline  
stg-fuga-archive-auth0-s3  
stg-fuga-polling-schedule-lambda

※スケジュールタイプのルールの場合は {source} に schedule と記載する。

## IAM

IAM に関わるリソースの命名について記載する。IAM グループ、IAM ユーザー、IAM ロール、IAM ポリシーの 4 点について述べる。

### IAM ユーザー

IAM ユーザーについては、誰 (人またはシステム) が利用するのかを識別することを目的とする。同じユーザーを複数の人やシステムで使いまわすと、誰が操作したのかといった証跡を追えなくなってしまうため、個別に発行することを推奨する。 また、役割や権限といった情報は名前に含めない。そのような名前はユーザーに紐づけるロールが増えた際などに名前と役割や権限の実態が乖離してしまうためである。

IAM ユーザー名については全体ポリシーから外れ、アンダースコア区切りを推奨する。

理由:

* 多くのサービスでユーザー名には慣習的にアンダースコアを用いることが多いため

人が利用する IAM ユーザー：

# 命名規約  
{company}\_{username}  
  
# 例  
future\_taro\_mirai

※AWS アカウントに関与する人が単一の会社に属する人だけである場合は {company}\_ を省略しても良い。

システムが利用する IAM ユーザー：

# 命名規約  
{product}\_{usage}  
  
# 例  
fuga\_api  
fuga\_auth0

AWS サービスに権限付与する場合は IAM ロールで付与することを想定している。システムが利用する IAM ユーザーは、別のクラウドや SaaS 等への権限付与に使うことを想定している。

[全体ポリシーの命名規約](#命名規約) とは異なり、環境名 {env} を Prefix につけない理由は次である。

* ある AWS アカウントに対して、Switch Role などで別の環境にアクセスする際に混乱が生じる
* ブラウザのパスワード管理などのために ID 名を分けたいという考えもあるかもしれないが、パスワード管理アプリなどの利用を推奨する

### IAM グループ

IAM グループに IAM ユーザーを追加することで複数ユーザーの権限を一括管理できる。IAM ユーザーは複数の IAM グループに追加可能だが、所属可能なグループ数は最大で 10 という制約があるため注意が必要である。

この制約を踏まえ、各役職ごとに基本となるグループを作成し、基本グループで対応できない例外的な権限の付与を個別のグループで対応することを想定した命名としている。

また、グループ数をむやみに増やさないためにグループ名に環境名 {env} はつけない。仮に future-developer というグループが dev 環境のみにアクセスできるといったような制御をする場合でも、グループ名には dev をつけず、dev 環境にアクセス可能なポリシーをグループにアタッチする方針としている。

基本となるグループ:

# 命名規約  
{company}-{role}  
  
# 例  
future-developer  
future-maintainer

ここでの {role} はユーザーが担う役割を表す。

個別のグループ:

# 命名規約  
{target}-{usage}  
  
# 例  
bastion-access

個別のグループは Session Manager で EC2 にアクセスするグループといった使い方を想定している。

例外的に特定のユーザーにのみ権限を付与する、会社を超えて共通のグループを付与するといったユースケースも考えられる。

### IAM ロール

IAM ロールは、AWS サービスに権限を付与する目的で利用する。IAM ロールに複数の IAM ポリシーをアタッチできるため、IAM ロールの命名では細かい権限を表現することは避け、IAM ロールを誰が使うのかを明確にすることを主目的とする。

# 命名規約  
{env}-{product}-{aws\_service}-{usage}  
  
# 例  
stg-fuga-ec2-bastion  
stg-fuga-lambda-api

※場合によっては {usage} 部に詳細情報を追加しても良い

### IAM ポリシー

IAM ポリシーの命名に入る前に、ポリシーの設計方針について記載する。 ここでは、ポリシー設計方針の代表例として、以下の 2 パターンについて説明する。

* 細かく設定し再利用するパターン
* 特定のリソースに付与するポリシーを書き出すパターン

それぞれの設計方針にはメリット・デメリットがあり開発規模などで使い分けが想定されるため、それぞれの場合の命名方法について記載する。

細かく設定する場合:

{  
 "Version": "2012-10-17",  
 "Statement": [  
 {  
 "Action": "s3:\*",  
 "Effect": "Allow",  
 "Resource": "\*"  
 }  
 ]  
}

# 命名規約  
{env}-{product}-{permission}-{aws\_service}-{usage}  
  
# 例  
stg-fuga-allow-s3-full  
stg-fuga-allow-ses-send

特定のリソースに付与するポリシーを書き出す場合:

{  
 "Version": "2012-10-17",  
 "Statement": [  
 {  
 "Action": "s3:\*",  
 "Effect": "Allow",  
 "Resource": "\*"  
 },  
 {  
 "Action": ["ses:SendEmail", "ses:SendRawEmail"],  
 "Effect": "Allow",  
 "Resource": "\*"  
 },  
 {  
 "Action": "sqs:\*",  
 "Effect": "Allow",  
 "Resource": "\*"  
 }  
 ]  
}

# 命名規約  
{env}-{product}-{aws\_service}-{usage}  
  
# 例  
stg-fuga-ec2-bastion  
stg-fuga-iam-group-future-develop

IAM グループ用のポリシーを作成する例では、company を含めた future-develop といった名前を {usage} としている。

この場合は命名粒度が IAM ロールと等しくなるため、命名規約も同じ方針にしている。

予め用意されているポリシーの名前は PascalCase 形式であるが (例: AmazonS3FullAccess)、ユーザーが作成したことを明確にするため snake\_case で命名する。

# タグの命名

AWS上の命名制約

[Tag naming and usage conventions](https://docs.aws.amazon.com/mediaconnect/latest/ug/tagging-restrictions.html) によれば以下の制約である。

* 最大 50 個のタグを設定できる
* タグキーは一意でなければならない
* タグキーは最長 128 Unicode 文字、タグ値は最長 256 Unicode 文字
* 使用可能文字
  + UTF-8 で表現できる文字、数字、スペース、. : + = @ \_ / -
* タグのキーと値では大文字と小文字が区別されます
* タグに aws: プレフィックスは禁止

[AWS リソースのタグ付け](https://docs.aws.amazon.com/ja_jp/general/latest/gr/aws_tagging.html#tag-best-practices) によれば、タグ付けのベストプラクティスは以下である。

* 個人情報 (PII) などの機密情報や秘匿性の高い情報をタグに設定しない
* すべてのリソースタイプに一貫して適用する
* リソースアクセスコントロールの管理、コスト追跡、オートメーション、整理など、複数の目的に対応したタグガイドラインを考慮する
* 自動化ツールを使用してリソースタグを管理する
* タグは、多めに使用する
* 将来の変更の影響を考慮する
* AWS Organizations のタグポリシーを利用することで、組織が採用するタグ付け標準を自動的に適用する

より詳しいタグ付けのベストプラクティスも存在するが、本紙の範囲を超えるため紹介のみに留める。 <https://docs.aws.amazon.com/whitepapers/latest/tagging-best-practices/tagging-best-practices.html>

## タグキー

* 使用する文字は英数字に限定する。基本的には **パスカルケース (PascalCase)** 形式を推奨する
  + リソース作成時に自動生成される Name タグと平仄を合わせるため
* 以下の観点でタグを使い分ける
  + リソース整理
  + コスト管理
    - AWS Billing にてコスト配分タグの設定が必要
  + オートメーション
    - EC2 の自動起動停止の管理など
  + アクセス制御
    - タグ値を利用した IAM ポリシーのアクセス制御など

主要なタグ項目

| Category | Tag Key | Required | Note |
| --- | --- | --- | --- |
| Common | Env | ✅ | 環境識別子 |
|  | System | ✅ | システム名 |
|  | Name | ✅ | リソースの識別子として機能名などを設定 |
| 費用按分 | Owner | ✅ | リソースの管理主管部署。費用の負担先を想定 |
|  | Project | ✅ | 開発担当チーム。どのチームがどれくらい利用したかをトレースするために設定 |
| ツールで利用 | StartAt |  | 起動時刻。自動化ツールなどで必要があれば設定 |
|  | EndAt |  | 停止時刻 |

## タグ値

* 各タグキーごとに原則、タグ値の元となる命名規約に従う
* 元となる命名規約がない場合、以下を推奨する
  + リソースの命名規約に従う
  + 頭文字のみの略語の場合は大文字のみ
* 値の取りうるパターンが決まっている場合には、タグポリシーで値を設定する

## タグポリシー

AWS Organizations を利用している場合、タグの標準化を促進するタグポリシーの設定が可能となる。 タグポリシーにより実現できることは以下。

* タグキーの大文字小文字の組み合わせを強制する
  + 例: Name を指定した場合、 name, NAME, nAME などはタグキーとして設定できなくなる
* 任意のタグキーに対して、設定可能なタグ値を指定する
  + Env など、予め取りうるタグ値のパターンが決まっている場合に利用
* タグポリシーを適用するリソースタイプを指定

# License

[CC-By-4.0](https://creativecommons.org/licenses/by/4.0/deed.ja)