AWS Black Belt Online Seminar

AWS CloudFormation

CloudFormation レジストリ編

山本 一生

Cloud Support Engineer
2023/10

AWS Black Belt Online Seminar とは

- 「サービス別」「ソリューション別」「業種別」などのテーマに分け、アマゾン ウェブ サービス ジャパン合同会社が提供するオンラインセミナーシリーズです
- AWS の技術担当者が、 AWS の各サービスやソリューションについてテーマごとに動画を公開します
- 以下の URL より、過去のセミナー含めた資料などをダウンロードすることができます
  - https://aws.amazon.com/jp/aws-jp-introduction/aws-jp-webinar-service-cut/
  - https://www.youtube.com/playlist?list=PLzWGOASvSx6FIwIC2X1nObr1KcMCBBlqY

ご感想は X (Twitter) へ！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

- 本資料では資料作成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。AWS のサービスは常にアップデートを続けているため、最新の情報は AWS 公式ウェブサイト (https://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
- 技術的な内容に関しましては、有料の AWS サポート窓口へお問い合わせください
- 料金面でのお問い合わせに関しましては、カスタマーサポート窓口へお問い合わせください (マネジメントコンソールへのログインが必要です)

本セミナーの対象者

想定聴講者

- CloudFormation レジストリに興味のある方

前提知識

- AWS の基本的な概要や CloudFormation の用語 (スタック、テンプレートなど) を理解していること
- Java や Python などプログラミング言語の基本的な知識を有すること

前提知識の補足

- AWS CloudFormation #1 基礎編
  - https://www.youtube.com/watch?v=4dyiPsYXG8I
  - https://pages.awscloud.com/rs/112-TZM-766/images/AWS-Black-Belt_2023_CloudFormation-1_0731_v1.pdf

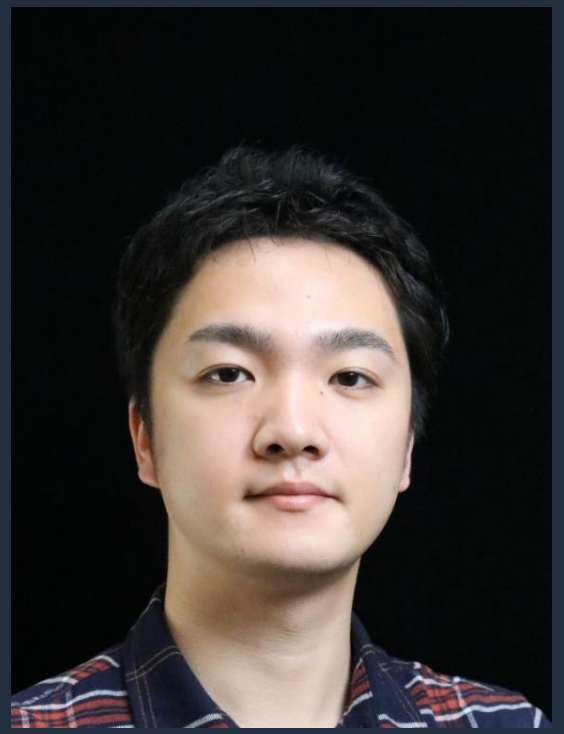
自己紹介

名前：山本 一生 (やまもと かずき)

所属：Cloud Support Engineer

経歴：SaaS 提供企業で開発業務を経験

好きなAWSサービス：AWS CloudFormation, Amazon EKS



CloudFormation レジストリの使い方

CloudFormation レジストリの使い方

- CloudFormation レジストリの概要
- Resource types
- Modules
- Hooks
- 資料紹介

CloudFormation レジストリの概要

CloudFormation の拡張機能を管理する機能

- 使用できる機能
  - Resource types
    - Create, Read, Update, Delete, List (CRUDL) の動作を全て実装した独自のリソースを定義できる
  - Modules
    - 複数のリソースや設定を一つの新たなリソースタイプとして定義できる
  - Hooks
    - CloudFormation が対象のリソースを作成、更新する前にカスタムロジックを実行できる
- 公開範囲
  - Public – AWS (例: AWS::RDS::DBInstance) や 3rd-party が公開
  - Private – 有効にしたアカウントでのみ使用できる

CloudFormation レジストリの概要

ユースケース

- AWS 外の API を通して操作できるリソースを CloudFormation 管理とする
  - Resource Types: CRUDL それぞれに対応する API を実行する実装を行う
- 特定の設定を含めたリソースを組織全体で使用する
  - Modules: 必要な設定やリソースを含めたモジュールを作成し共有する
- 必要な設定がないリソースがある場合、スタックを失敗させる
  - Hooks: 特定のリソースの操作前に Hooks で検証を行う

追加資料

AWS CloudFormation のパブリックレジストリの紹介
https://aws.amazon.com/jp/blogs/news/introducing-a-public-registry-for-aws-cloudformation/

CloudFormation レジストリの使い方

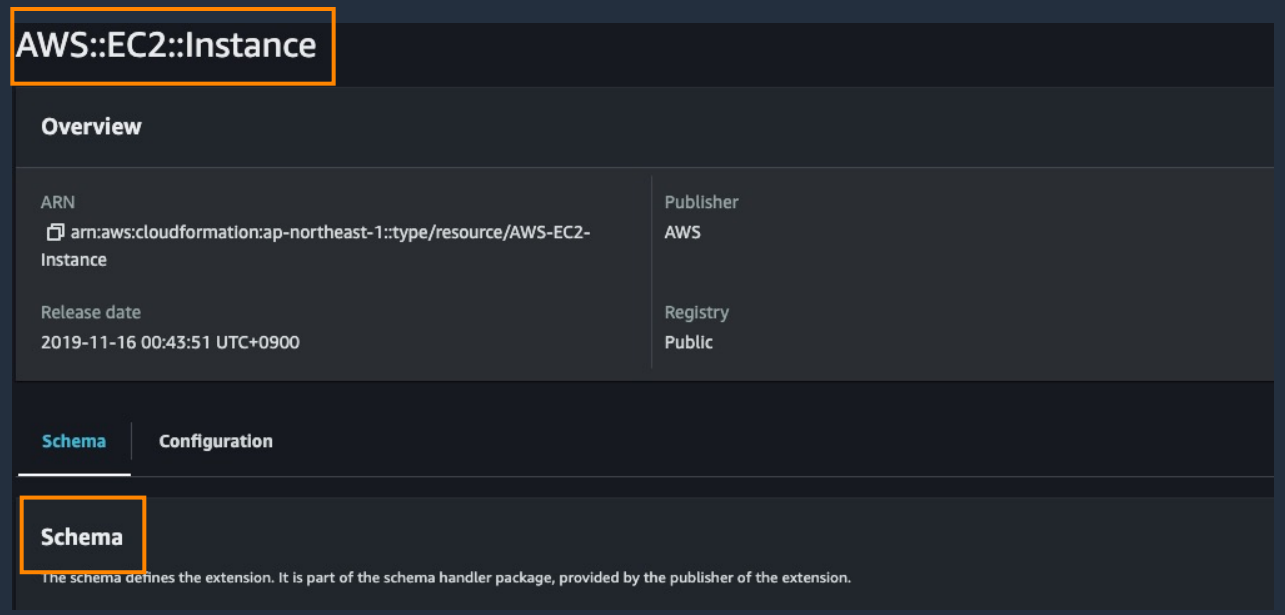
- CloudFormation レジストリの概要
- Resource types
- Modules
- Hooks
- 資料紹介

Resource types

作成したリソース定義を登録することで他の CloudFormation の機能と統合

- テンプレート上で指定するのみでそのリソースを使用できる
- ドリフト検出やリソースインポートが可能
- AWS Config による構成情報の追跡が可能

AWS が公開しているリソースタイプの例



Resource types

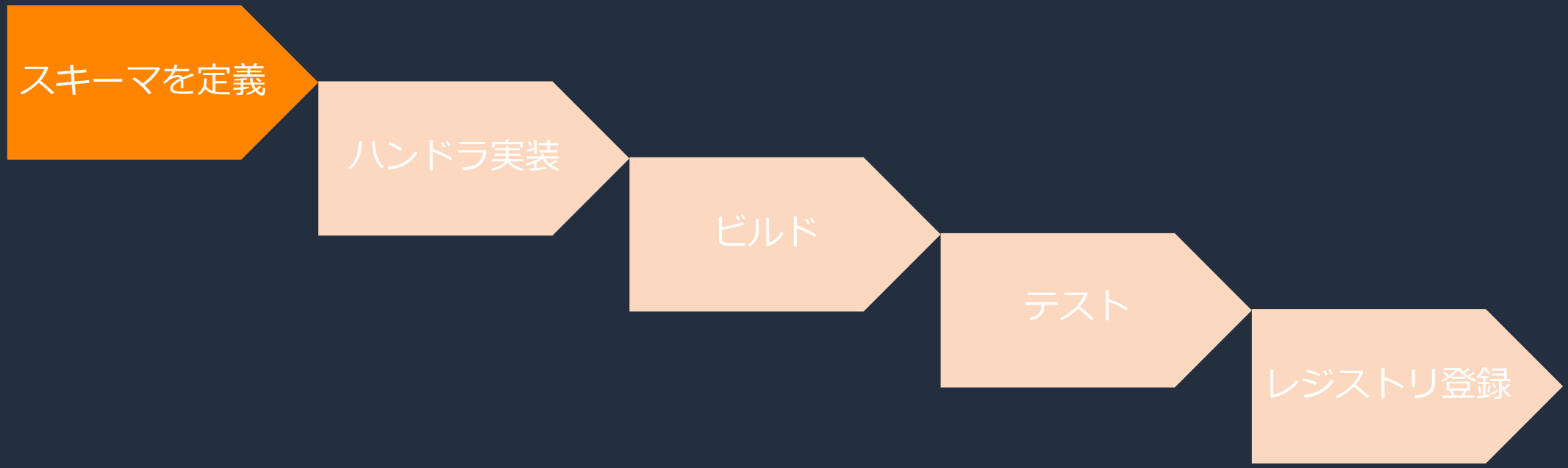
事前準備

- Python3, pip
  - 各言語向けプラグインのインストール用
- SAM CLI, AWS CLI, Docker
  - ローカルでのテスト用
- cloudformation-cli (cfn コマンド)
- 各言語向けプラグイン
  - Java
  - Python
  - Go
  - TypeScript

https://docs.aws.amazon.com/cloudformation-cli/latest/userguide/what-is-cloudformation-cli.html#resource-type-setup

Resource types

実装の流れ



Resource types

スキーマ: リソースタイプの詳細を定義した JSON ファイル。cfn init コマンドで用意される。

```
{
    "typeName": "AWSSamples::EC2::ImportKeyPair",
    "description": "Sample resource schema demonstratin
    "sourceUrl": "https://github.com/aws-cloudformation
    "definitions": {
        "Tag": {
            "description": "A key-value pair to associa
            "type": "object",
            "properties": {
                "Key": {
                    "type": "string",
                    "description": "The key name of the
                    "minLength": 1,
                    "maxLength": 128
                },
                "Value": {
                    "type": "string",
                    "description": "The value for the t
                    "minLength": 0,
                    "maxLength": 256
                }
            },
            "required": [
                "Key",
                "Value"
            ],
            "additionalProperties": false
        }
```

[必須] typeName:

リソースタイプ名 (テンプレートの Type に指定)

[必須] description:

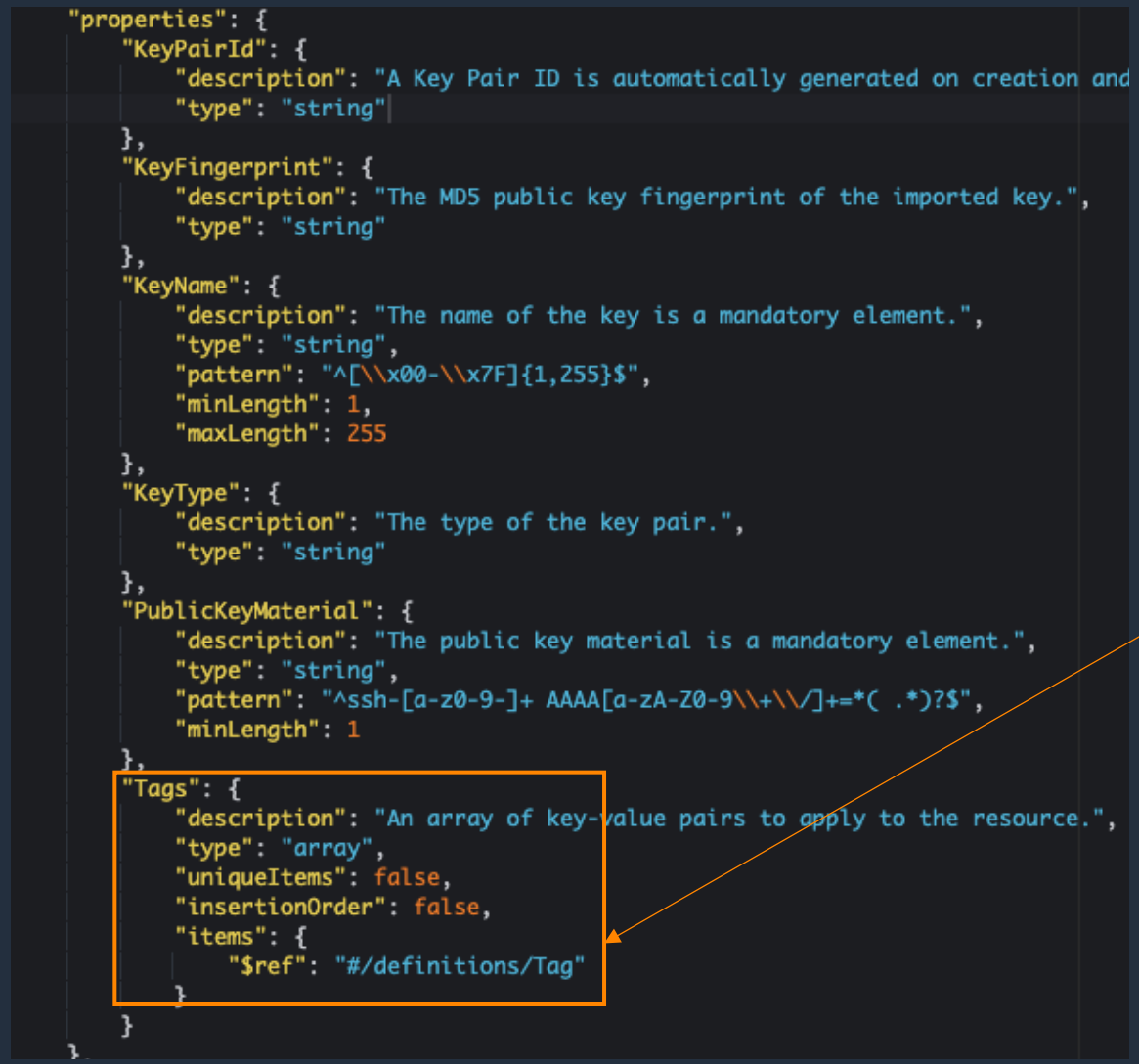
リソースの説明

definitions:

properties で使用できるスキーマ

https://docs.aws.amazon.com/cloudformation-cli/latest/userguide/resource-type-schema.html

Resource types



[必須] properties:

テンプレートの Properties に指定する値の定義

- Pattern や minLength などで制約
- Definitions のスキーマを参照可能
- Required や createOnlyProperties を設定可能 (後述)

Resource types

```
"additionalProperties": false,
"required": [
    "KeyName",
    "PublicKeyMaterial"
],
"readOnlyProperties": [
    "/properties/KeyPairId",
    "/properties/KeyFingerprint",
    "/properties/KeyType"
],
"writeOnlyProperties": [
    "/properties/PublicKeyMaterial"
],
"primaryIdentifier": [
    "/properties/KeyPairId"
],
"createOnlyProperties": [
    "/properties/KeyName",
    "/properties/PublicKeyMaterial"
],
"tagging": {
    "taggable": true,
    "tagOnCreate": true,
    "tagUpdatable": true,
    "cloudFormationSystemTags": false,
    "tagProperty": "/properties/Tags"
},
```

必須プロパティのリスト

作成時のみ設定できるプロパティのリスト

- 変更するとリソースが再作成される
- リソース置き換えの動作は replacementStrategy で指定
  - デフォルト動作は作成し削除 (create_then_delete)

Resource types

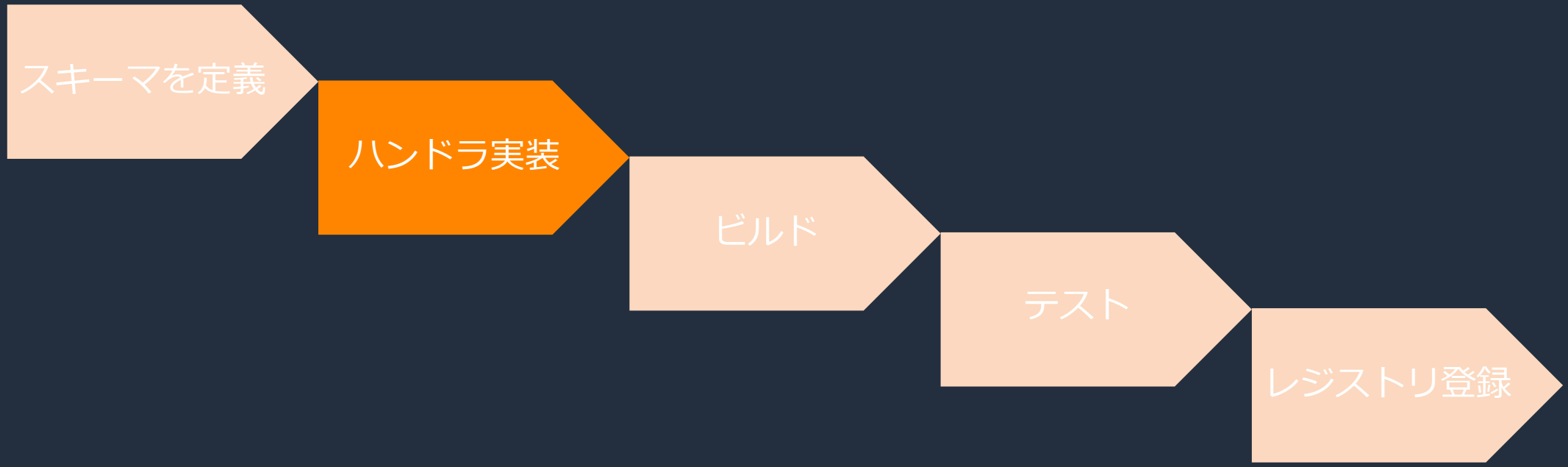
```
"handlers": {
    "create": {
        "permissions": [
            "ec2:ImportKeyPair",
            "ec2:CreateTags"
        ]
    },
    "read": {
        "permissions": [
            "ec2:DescribeKeyPairs"
        ]
    },
    "update": {
        "permissions": [
            "ec2:CreateTags",
            "ec2:DeleteTags"
        ]
    },
    "delete": {
        "permissions": [
            "ec2:DeleteKeyPair",
            "ec2:DescribeKeyPairs"
        ]
    },
    "list": {
        "permissions": [
            "ec2:DescribeKeyPairs"
        ]
    }
```

各操作 (Create, Read, Update, Delete, List) に必要なアクション

- 必須
  - Create
  - Read
  - Delete

Resource types

実装の流れ



Resource types

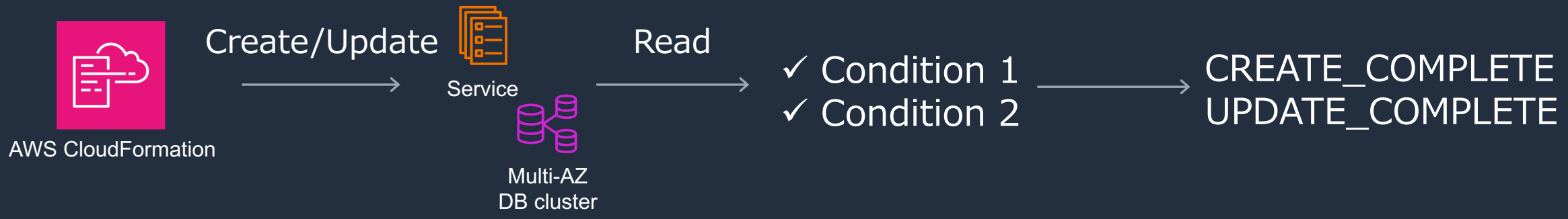
ハンドラ	用途
Create	リソースの**作成**時に実行
Read	CloudFormation が作成したリソースの詳細を取得 リソースの**安定化**の判定やドリフト検出で使用
Update	リソースの**更新**時に実行*
Delete	リソースの**削除**時に実行
List	(必要な場合) リソースの一覧を取得

* Update ハンドラがないとリソースを更新できず全て再作成となる

- cfn init コマンドである程度用意される
- 各ハンドラで操作の結果やエラーコードなどを含む ProgressEvent を return することで処理が進む
- AWS API を実行する時は通常のリソースと同様サービスロールか実行 IAM ユーザー/ロールの権限を使用できる

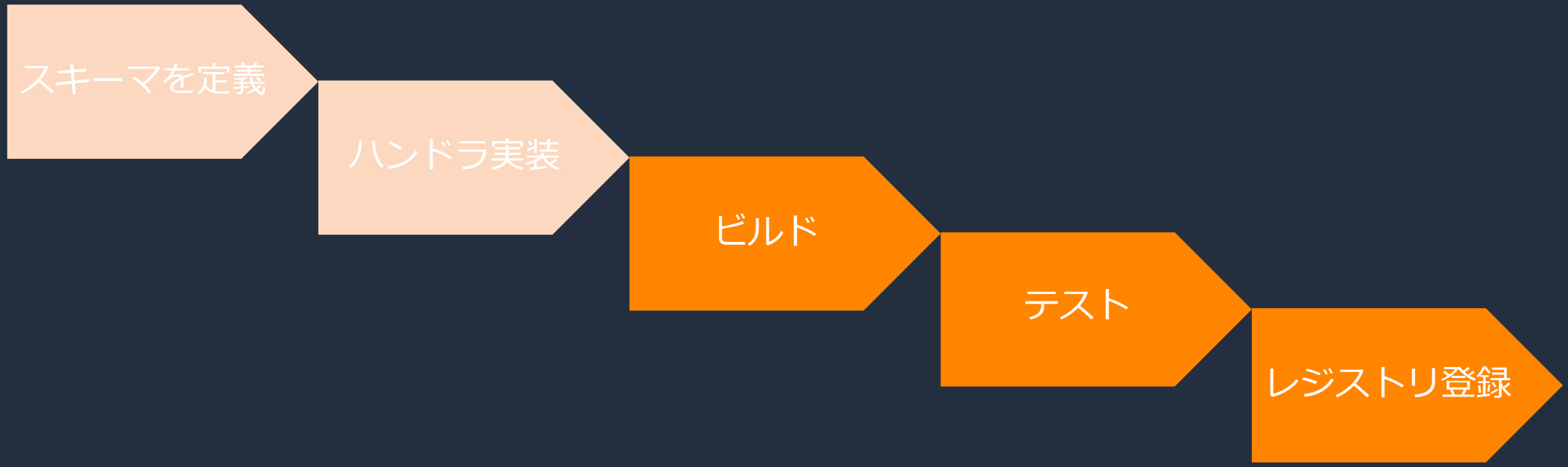
リソースの安定化 (Stabilization)

- リソース作成や更新後 (Create, Update ハンドラー実行後) にリソースが一定のステータスとなるまで待機する機能
- Resource type でも実装可能



Resource types

実装の流れ



Resource types

- ビルド
  - 必要に応じてハンドラをビルドする
- テスト
  - Docker を使用できる環境であればハンドラをテスト実行できる
    - sam local start-api と cfn test コマンドを使用する
      - 実行環境上で Lambda 関数のモックをコンテナで立ち上げる
- レジストリ登録
  - cfn submit コマンドを実行することで登録される
  - CloudFormation から通常のリソースと同じように使用できる

CloudFormation レジストリの使い方

- CloudFormation レジストリの概要
- Resource types
- Modules
- Hooks
- 資料紹介

Modules

複数のリソースを独自のリソースタイプにまとめる機能

- Type 名例: organization::service::usecase::MODULE (MODULE は固定)
- Module に Module を含めることができる

使い方

1. cfn init
2. fragments フォルダにテンプレート作成 (YAML/JSON)
3. cfn submit (テンプレートの検証やレジストリへの登録)
4. CloudFormation テンプレートで指定しスタック作成

Modules

```yaml
Parameters:
  Days:
    Type: Number
    Default: 1

Resources:
  TestLog:
    Type: "AWS::Logs::LogGroup"
    UpdateReplacePolicy: "Delete"
    DeletionPolicy: "Delete"
    Properties:
      RetentionInDays: !Ref Days
```

cfn submit する fragments/sample.yaml

```yaml
Resources:
  Sample:
    Type: MyOrg::Sample::Sample::MODULE
```

CloudFormation テンプレートで指定する

```yaml
Hooks: {
}
Resources:
  SampleTestLog:
    Type: AWS::Logs::LogGroup
    DeletionPolicy: Delete
    UpdateReplacePolicy: Delete
    Metadata:
      AWS::Cloudformation::Module:
        TypeHierarchy: MyOrg::Sample::Sample::MODULE
        LogicalIdHierarchy: Sample
    Properties:
      RetentionInDays: 1
Rules: {
}
Conditions: {
```

CloudFormation が Module を展開する

テンプレートで指定可能なタイプまとめ

```
Resources:
  CustomResource:
    Type: Custom::Test
```

カスタムリソース
- 処理中に Lambda や SNS を実行できる
- Lambda や SNS を実装する必要がある

```
Resources:
  NativeResource:
    Type: AWS::RDS::DBInstance
```

AWS が公開している Public なリソースタイプ
- ネイティブリソースとも呼ばれる
- 追加設定なしで使用できる

```
Resources:
  PrivateResourceTypeResource:
    Type: AWSSamples::EC2::ImportKeyPair
```

Private なリソースタイプ
- 明示的に有効とする必要がある

```
Resources:
  ModuleResource:
    Type: MyOrg::Sample::Sample::MODULE
```

Module
- 明示的に有効とする必要がある

CloudFormation レジストリの使い方

- CloudFormation レジストリの概要
- Resource types
- Modules
- Hooks
- 資料紹介

Hooks

CloudFormation によるリソース作成や更新の前に処理を実行

- 対象のスタックやリソースを設定可能
- エラーとしてスタックの操作を止めるか警告のみとするか設定可能
- AWSSamples として公開している Hooks も存在

ユースケース

- セキュリティの強化
  - EKS クラスターのロギング設定を検証
  - IAM ポリシーの Condition に MFA を強制する

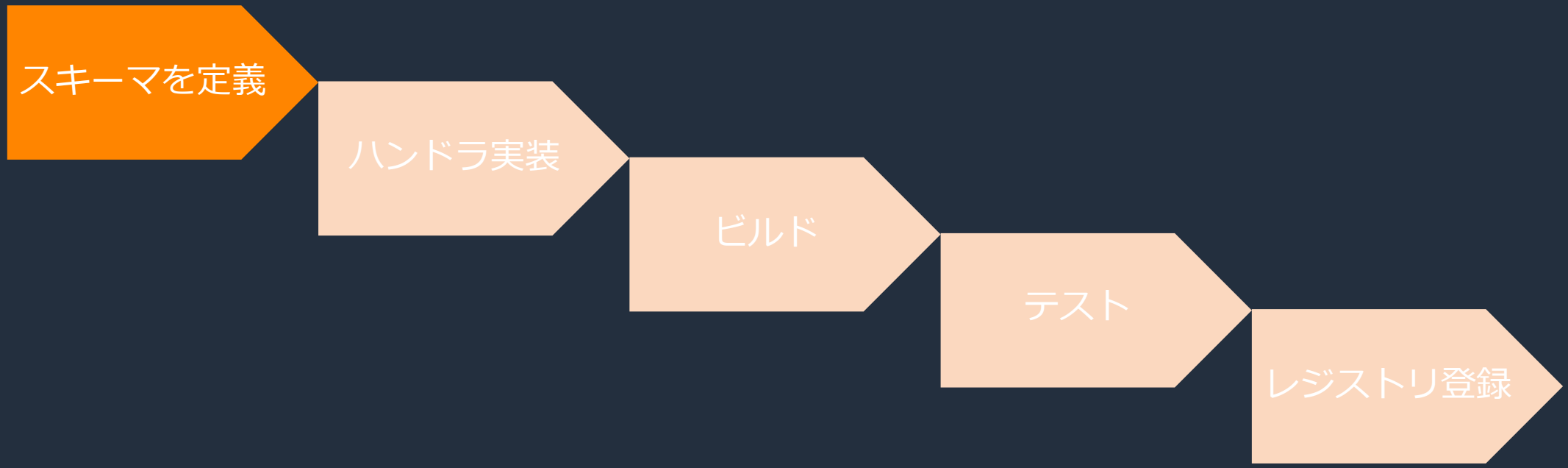
Hooks

事前準備

- Python3, pip
  - 各言語向けプラグインのインストール用
- SAM CLI, AWS CLI, Docker
  - ローカルでのテスト用
- cloudformation-cli (cfn コマンド)
- 各言語向けプラグイン
  - Java
  - Python

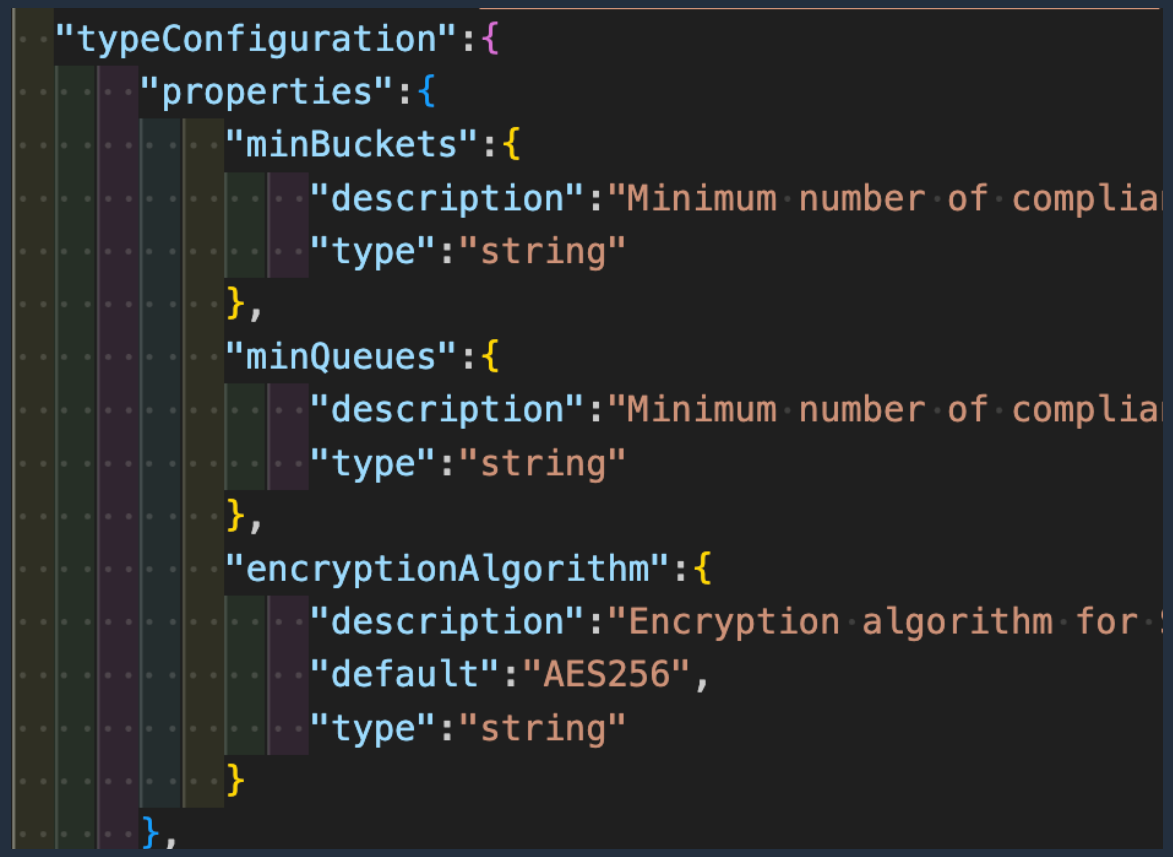
Hooks

実装の流れ



Hooks

スキーマ: リソースの詳細を定義した JSON ファイル。cfn init コマンドで用意される。



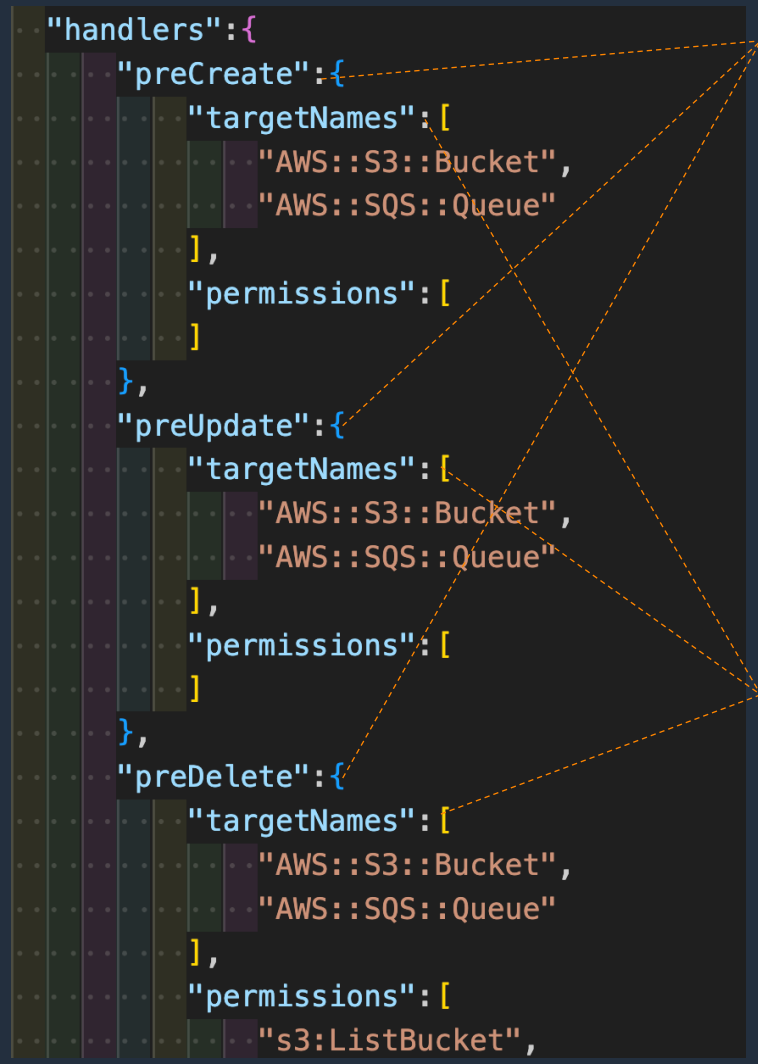
```
"typeConfiguration":{
    "properties":{
        "minBuckets":{
            "description":"Minimum number of complia
            "type":"string"
        },
        "minQueues":{
            "description":"Minimum number of complia
            "type":"string"
        },
        "encryptionAlgorithm":{
            "description":"Encryption algorithm for
            "default":"AES256",
            "type":"string"
        }
    },
```

[必須] typeConfiguration:

Hooks の設定。

https://docs.aws.amazon.com/cloudformation-cli/latest/userguide/hooks-structure.html#hook-configuration-schema

Hooks



```
"handlers":{
    "preCreate":{
        "targetNames":[
            "AWS::S3::Bucket",
            "AWS::SQS::Queue"
        ],
        "permissions":[
        ]
    },
    "preUpdate":{
        "targetNames":[
            "AWS::S3::Bucket",
            "AWS::SQS::Queue"
        ],
        "permissions":[
        ]
    },
    "preDelete":{
        "targetNames":[
            "AWS::S3::Bucket",
            "AWS::SQS::Queue"
        ],
        "permissions":[
            "s3:ListBucket",
```

各ハンドラに必要なアクション (最低一つのハンドラが必須)

- preCreate
  - リソース作成前に実行
- preUpdate
  - リソース更新前に実行
- preDelete
  - リソース削除前に実行

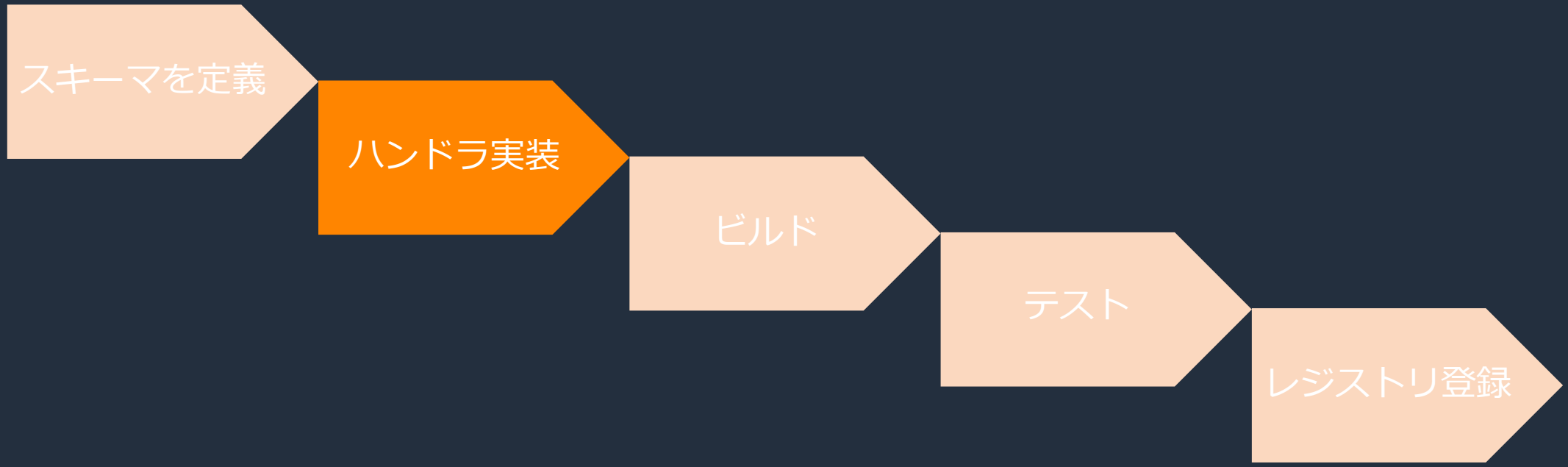
  ※ UpdateCleanup 前には実行されない
    - UpdateCleanup 例
      - テンプレートからリソースを削除しスタック更新
      - 更新タイプが Replacement のリソース削除

対象のリソースタイプ

- ハンドラーが実行されるリソースタイプのリスト
  - AWS::S3::* のようにワイルドカードを使用可能

Hooks

実装の流れ



Hooks

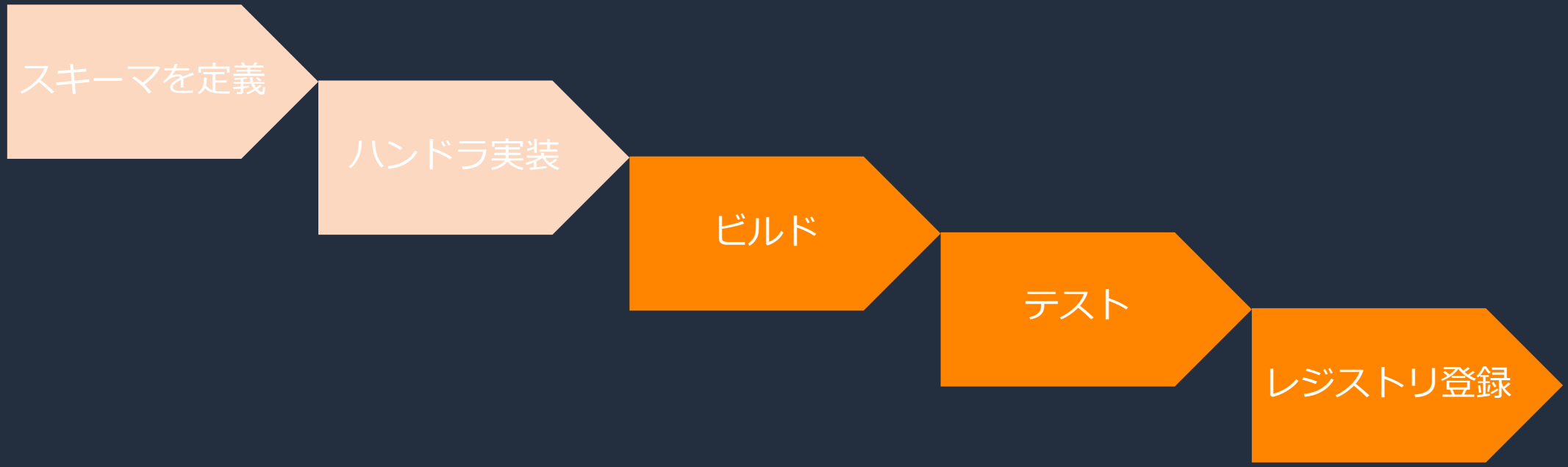
- cfn init コマンドである程度用意される
- request に含まれるリソースタイプ名や resourceProperties を元に検証できる
  - preUpdate では更新前の resourceProperties も参照可能なため更新前後の検証ができる
- Resource Type と同様に実行結果などを含む ProgressEvent を return することで処理が進む

ハンドラ	用途
preCreate	リソースの**作成前**に実行*
preUpdate	リソースの**更新前**に実行*
preDelete	リソースの**削除前**に実行*

* スキーマで定義されたいずれか一つのハンドラは必須

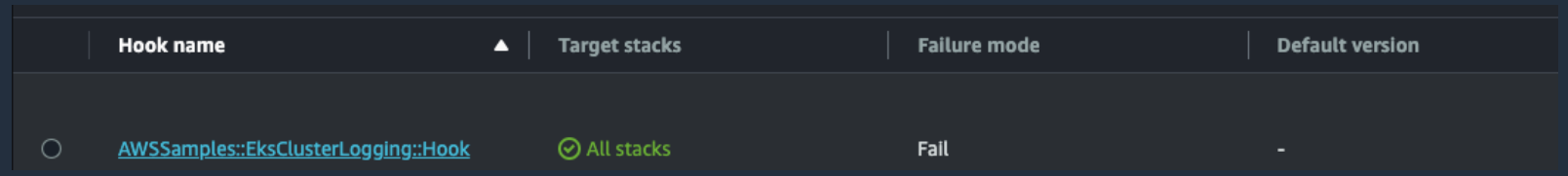
Hooks

実装の流れ



Hooks

- ビルド
  - 必要に応じてハンドラをビルドする
- テスト
  - Docker を使用できる環境であればハンドラをテスト実行できる
    - sam local start-api と cfn test コマンドを使用する
      - 実行環境上で Lambda 関数のモックをコンテナで立ち上げる
- レジストリ登録
  - cfn submit コマンドを実行することで登録される
  - Hook を Activate することで使用可能となる (画像はサンプル Hook を有効とした例)
    - Hook の設定で対象のスタックや失敗時の動作 (Fail/Warn) などを設定できる



CloudFormation レジストリの使い方

- CloudFormation レジストリの概要
- Resource types
- Modules
- Hooks
- 資料紹介

資料紹介

実際に CloudFormation レジストリを試してみたい場合はこちら

- AWS CloudFormation Workshop advanced
  - https://catalog.workshops.aws/cfn101/en-US/advanced
    - Resource Types (Python)
    - Modules

- CloudFormation Command Line Interface
  - https://docs.aws.amazon.com/cloudformation-cli/latest/userguide/what-is-cloudformation-cli.html
    - Resource Types (Java)
    - Modules
    - Hooks (Java, Python)

Thank you!