



AWS CloudFormation

開発・テスト・デプロイ編

山川 達也

Solutions Architect

2023/12

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 の開発をしようとされている方、すでに実施されている方

前提知識

- AWS の概要を理解していること
- CloudFormation の概要を理解していること
(AWS Blackbelt CloudFormation #1#2 基礎編 視聴済み)
- インフラ構成管理ツールのご利用経験があると望ましい（必須ではない）

ゴール

- CloudFormation の開発を進めていただく上での知識、理解を深めていただく

自己紹介

山川 達也

アマゾンウェブサービスジャパン
ソリューションアーキテクト

大手不動産業界のお客様を中心にご支援しています



好きな AWS サービス
AWS CloudFormation

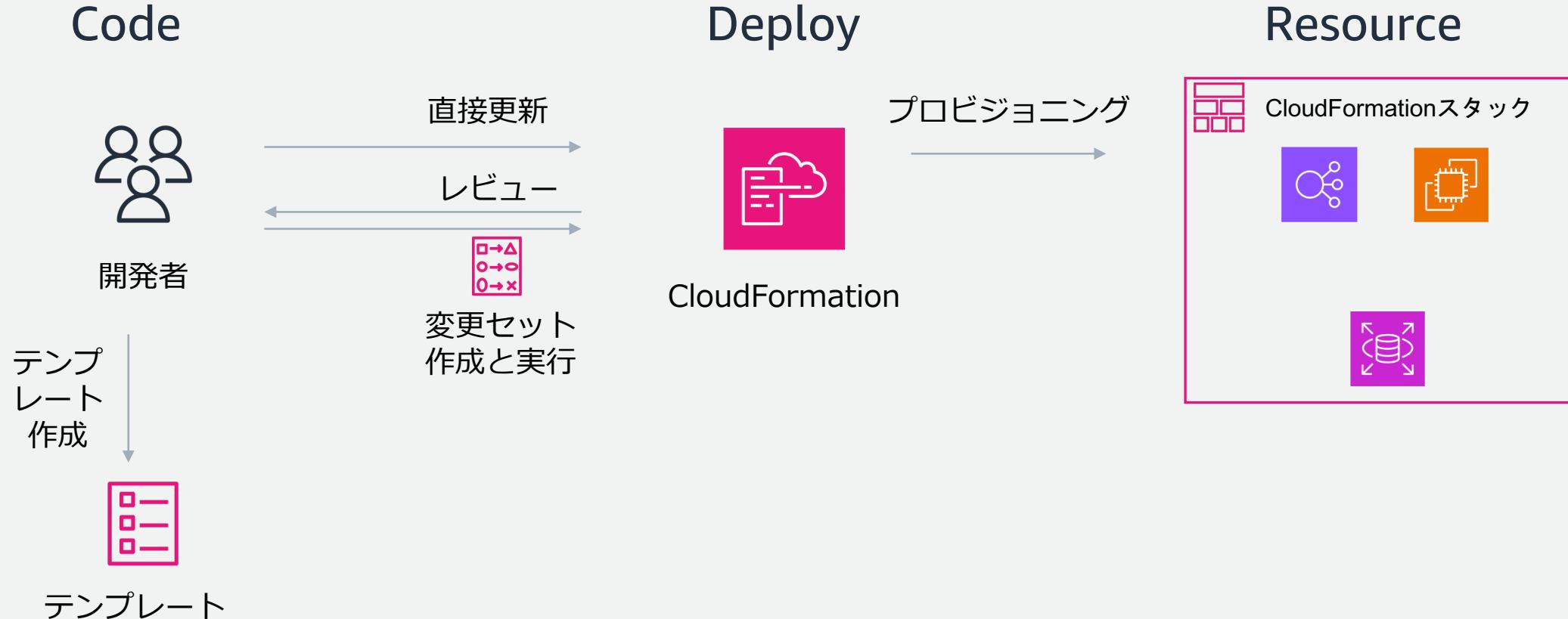
アジェンダ

- はじめに
- CloudFormation スタックとコード設計の勘所
- 開発環境の整備
- テスト
- デプロイ

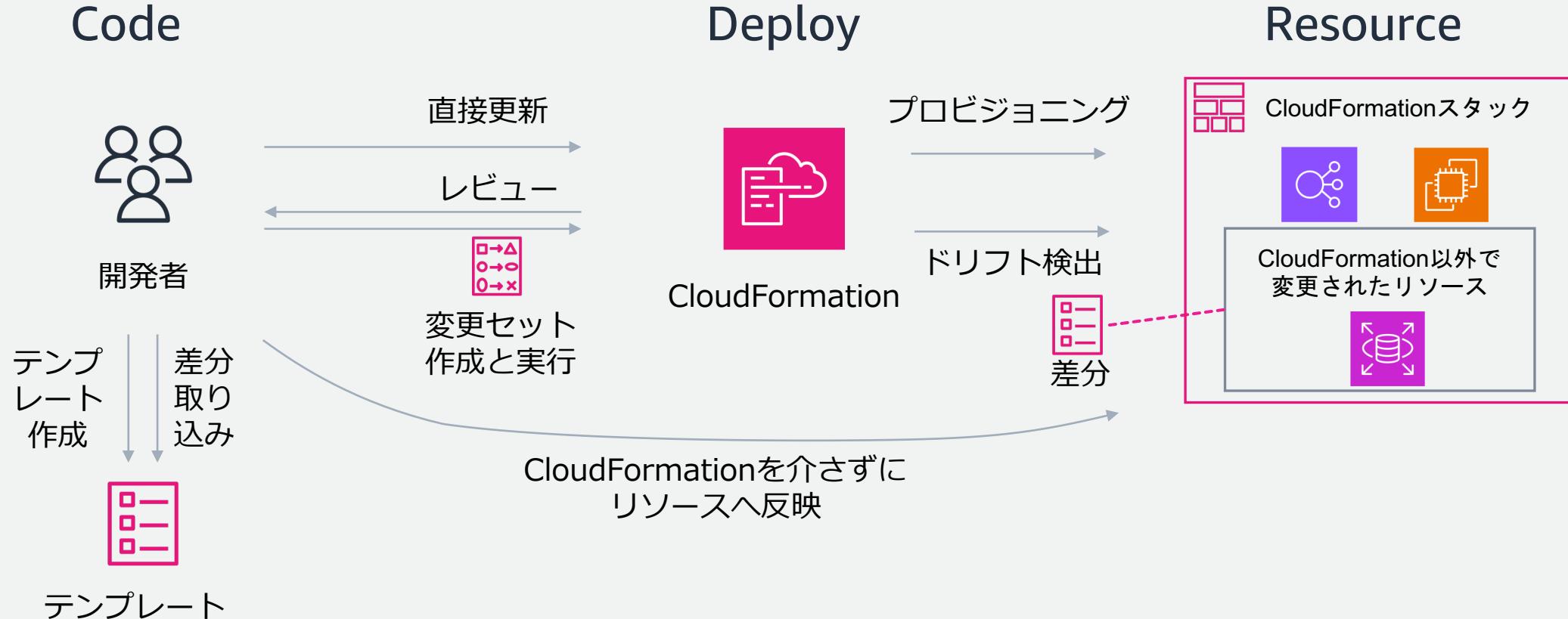
はじめに



開発者視点でのCloudFormation利用の流れ



開発者視点でのCloudFormation利用の流れ



ドリフト検出

ドリフト検出機能を利用することで、テンプレート内のリソースプロパティがリソースの実際の設定と一致するかどうかを確認することが可能

ドリフトがある状態の際は、ドリフトを解消した後にスタックを更新することを推奨

✓ SecurityGroupIngress.3 ADD - {"CidrIp": "0.0.0.0/0", "FromPort": 3389, "IpProtocol": "tcp", "ToPort": 3389}

詳細

予定 現在

```
{ "GroupDescription": "monitoring-2-WebServer", "GroupName": "monitoring-2-WebServer", "SecurityGroupIngress": [ { "CidrIp": "0.0.0.0/0", "FromPort": 3389, "IpProtocol": "tcp", "ToPort": 3389 } ], "Tags": [ { "Key": "Name", "Value": "monitoring-2-WebServer" } ] }
```

```
{ "GroupDescription": "monitoring-2-WebServer", "GroupName": "monitoring-2-WebServer", "SecurityGroupIngress": [ { "CidrIp": "0.0.0.0/0", "FromPort": 3389, "IpProtocol": "tcp", "ToPort": 3389 }, { "CidrIp": "0.0.0.0/0", "FromPort": 80, "IpProtocol": "tcp", "ToPort": 80 } ], "Tags": [ { "Key": "Name", "Value": "monitoring-2-WebServer" } ] }
```

検出されたドリフト

```
[ { "CidrIp": "0.0.0.0/0", "FromPort": 3389, "IpProtocol": "tcp", "ToPort": 3389 } ]
```



CloudFormation スタックと コード設計の勘所

CloudFormation スタックの設計について

CloudFormation スタックの設計の際に、以下の観点を考慮することで、開発や運用性の高いテンプレートの設計が可能

- CloudFormation におけるスタック分割
 - 組織やチームにあったスタック分割の進め方や方法論を整理する
- 環境ごとに再利用性可能なテンプレートの実現
 - 条件を定めて開発、検証、本番環境でテンプレートを再利用する
- 同一構成のシステム間で再利用しやすいテンプレートの作成
 - 環境依存のパラメーターをハードコーディングせずに再利用しやすいテンプレートを作成する

CloudFormation スタックの設計について

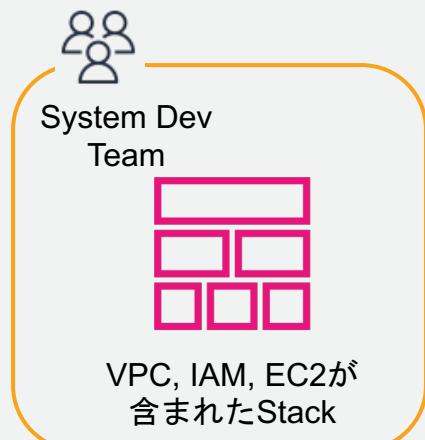
- CloudFormation 設計におけるスタック分割
- 環境ごとに再利用性可能なテンプレートの実現
- 同一構成のシステム間で再利用しやすいテンプレートの作成

CloudFormation 設計におけるスタック分割の悩み (1/2)

- 組織の体制、システム、環境、開発規模、CloudFormation のサービスクォータの上限、アカウントやリージョン跨ぎ等の理由により、スタックの分割方法について検討することがある
- スタックを分割するほど、各リソースの依存関係の管理が煩雑となるため、開発の生産性とのトレードオフ

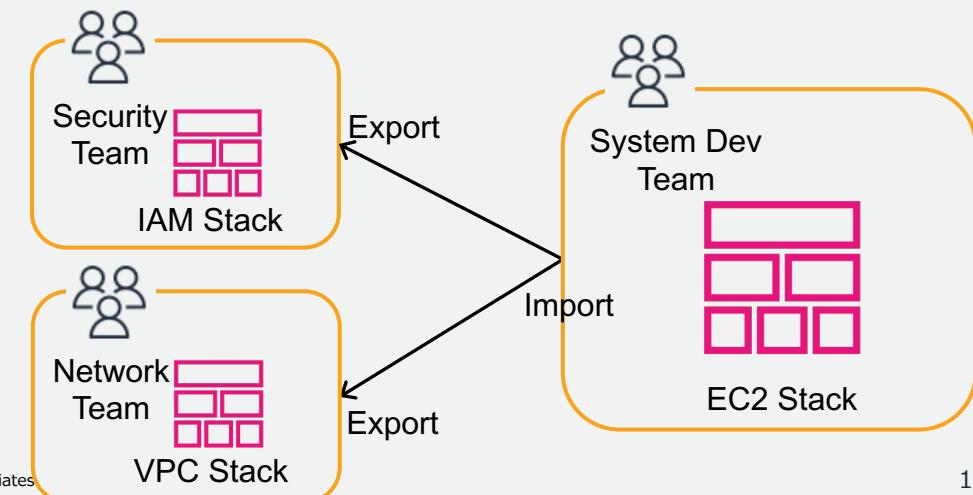
単一のスタックの場合

- CloudFormationが自動的に依存関係を解決



複数のスタックに分割した場合

- Exportされた値を別のスタックFn::ImportValueを用いてImportしなくてはならない
- 別スタックからImportされている値は変更することができない



CloudFormation 設計におけるスタック分割の悩み (2/2)

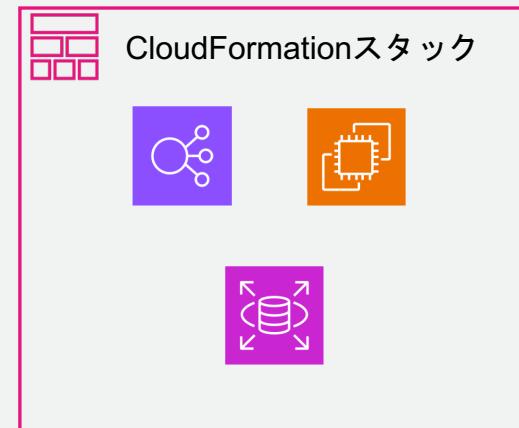
- Git リポジトリ内のすべての CloudFormation テンプレートを毎回デプロイすることで、Git リポジトリと実際の AWS リソースが同期された状態を維持できる
- 一部の CloudFormation テンプレートのみを選択的にデプロイすると、デプロイの一貫性と完全性が失われ、Git リポジトリを見ても実際のリソース構成がわからなくなるリスクがある
- さらに、スタックだけでなく Git リポジトリを複数に分割した場合、より広いスコープでデプロイ方法を検討する必要がある

Git Repository

CloudFormation テンプレート



Resource



ライフサイクルと所有権によるスタック分割

共通のライフサイクル、所有権を持つリソースでグループ化し、スタック分割を検討する手法

ライフサイクルによる分割メリット

- デプロイの影響範囲を最小化できる

ライフサイクルによる分割デメリット

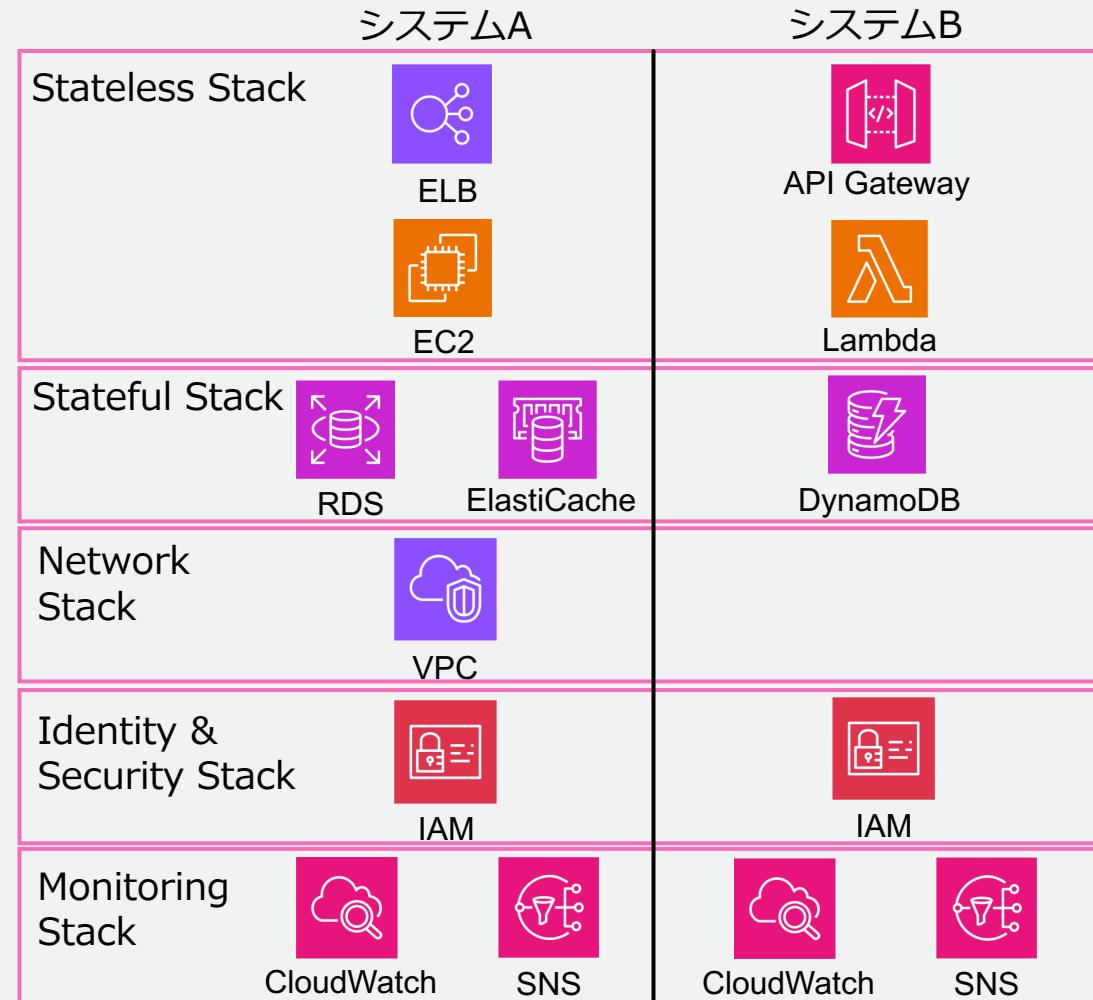
- 同一リソースにおいてもユースケースによってライフサイクルが異なる可能性があり、分類が難しい

所有権による分割メリット

- 開発者、チームによる責任範囲が明確になる

所有権による分割デメリット

- 開発におけるコミュニケーションコストが増える
- 所有者間で戦略やルールを浸透させるためのコストが増える



アプリケーション内の機能や役割によるスタック分割

クロススタック参照をできるだけ少なくすることを目標としてスタック分割を検討する手法

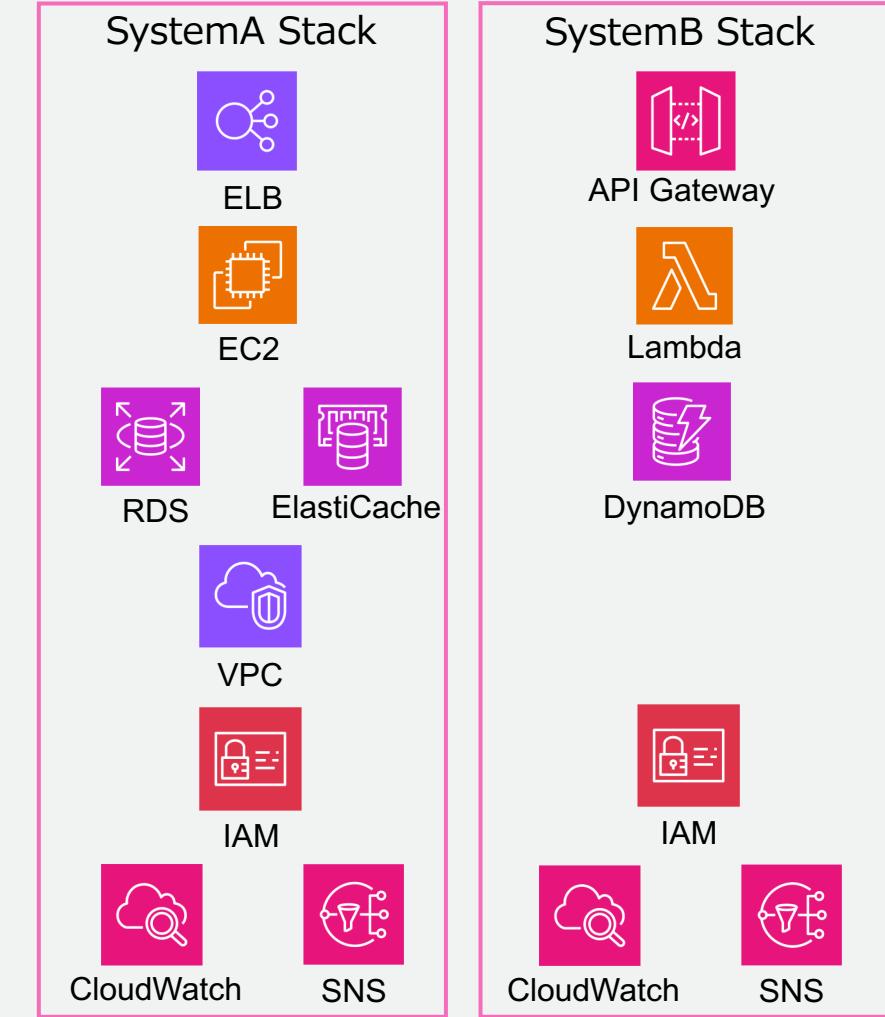
メリット

- システムに必要なリソースを1つのテンプレートに記載することで、クロススタック参照を少なくすることができ、開発・運用がしやすい
- 責任の範囲をシステムごとに設定できることから、分割の範囲を定めやすい
- 開発におけるコミュニケーションコストが減る

デメリット

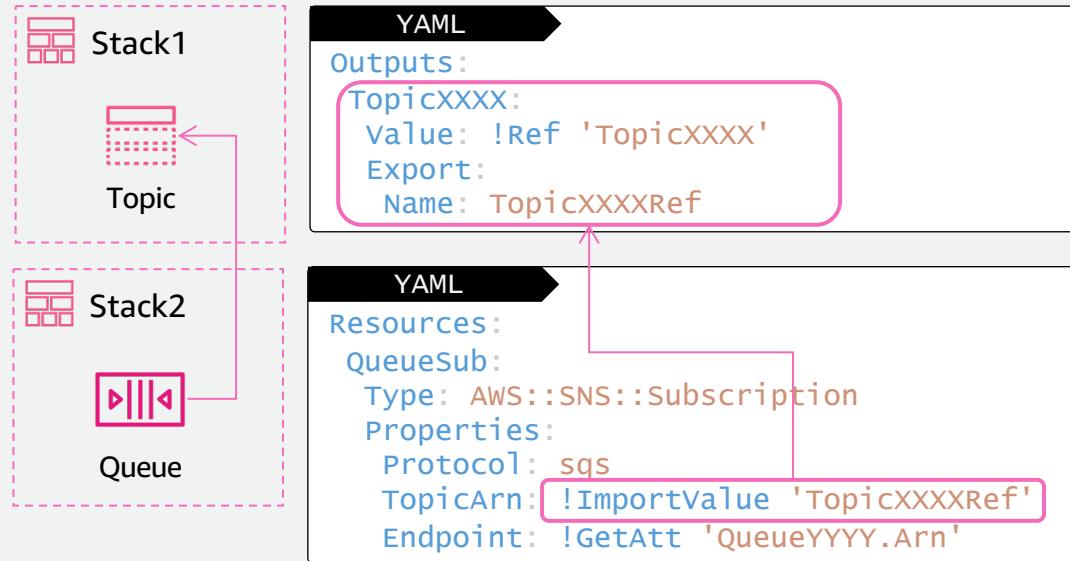
- 1つのテンプレートファイルが長くなり、可読性が低下する
- CloudFormation のリソース数やテンプレートのサイズの上限に抵触することがある

*CloudFormation のサービスクォータの上限を超えるケースにおいては、ネストされたスタックを利用した分割の検討ができる



CloudFormation における強い参照と弱い参照

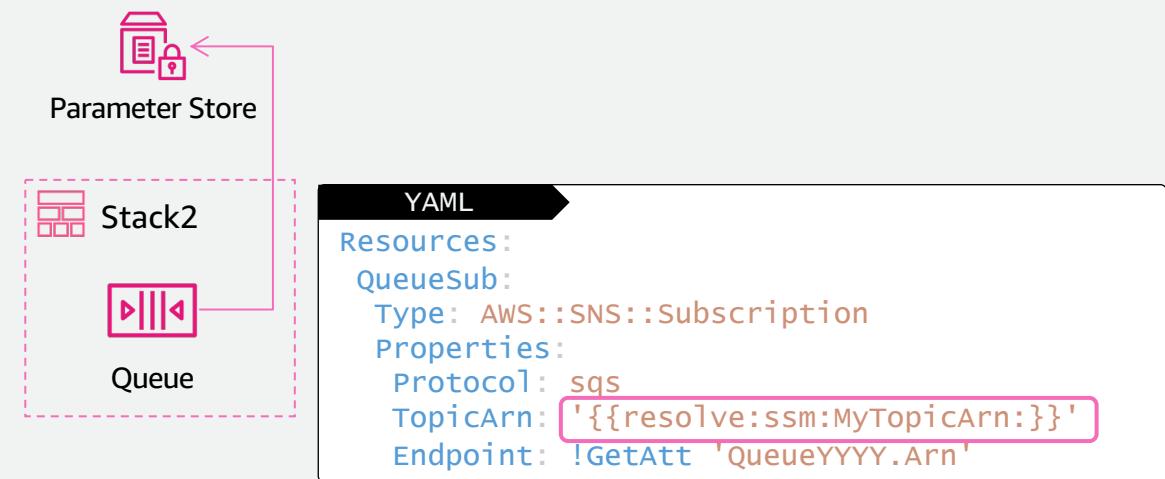
強い参照



Export した値を `!ImportValue` で参照すると
CloudFormation は参照しているスタックを記録 * し
被参照リソースが誤って削除さないようにガードする

* \$ aws cloudformation list-imports で確認可能

弱い参照



動的参照は CloudFormation がスタックのデプロイ中に
解決するが、参照しているスタックは記録しない

CloudFormation の Parameter による値渡しも同様

→ 被参照リソースを制約なく削除可能。ただし、壊れても
気づきづらい

CloudFormation スタックの設計について

- CloudFormation 設計におけるスタック分割
- 環境ごとに再利用性可能なテンプレートの実現
- 同一構成のシステム間で再利用しやすいテンプレートの作成

環境ごとに再利用可能なテンプレートの作成

- 各システムにおいて、本番と同様の構成を開発や検証環境に複製し、本番と差分の少ない環境でテストを実施することが求められる
- Parameters セクションの入力値に応じて、Mappings や Conditions により設定をコントロールすることで、テンプレートを各環境で再利用可能にすることが可能

Mappings を利用した再利用可能なテンプレートの作成

Parameters セクションで環境情報を入力し、Mappings セクションで各環境に応じた値とパラメーターをマッピングさせることで、環境ごとに設定を変更することが可能

```
Parameters:  
  EnvType:  
    Description: 'Specify the Environment type of the stack.'  
    Type: String  
    Default: Test  
    AllowedValues:  
      - Test  
      - Prod  
    ConstraintDescription: 'Specify either Test or Prod.'  
  
Mappings:  
  EnvToInstanceType:  
    Test:  
      InstanceType: t3.micro  
    Prod:  
      InstanceType: m6i.large
```

Parametersで指定したEnvType とインスタンスタイプの値をマッピング



```
Resources:  
  WebServerInstance:  
    Type: AWS::EC2::Instance  
    Properties:  
      ImageId: !Ref AmiID  
      InstanceType: !FindInMap  
        - EnvToInstanceType  
        - !Ref EnvType  
        - InstanceType
```

マッピングした値は !FindInMap 関数で取得 EnvType に応じてインスタンスを構築

Conditions を利用した再利用可能なテンプレートの作成

Parameters セクションで環境情報を入力し、Conditions セクションで条件分岐させることで環境ごとに設定を変更することが可能

```
Parameters:  
LatestAmiId:  
  Type: AWS::SSM::Parameter::Value<AWS::EC2::Image::Id>  
  Default: /aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2  
  
EnvType:  
  Description: Specify the Environment type of the stack.  
  Type: String  
  AllowedValues:  
    - test  
    - prod  
  Default: test  
  ConstraintDescription: Specify either test or prod.  
  
Conditions:  
  IsProduction: !Equals  
    - !Ref EnvType  
    - prod
```

Parameters で指定した EnvType が prod であるかどうかを判断



```
Resources:  
  EC2Instance:  
    Type: AWS::EC2::Instance  
    Properties:  
      ImageId: !Ref LatestAmiId  
      InstanceType: !If [IsProduction, m6i.large, t3.micro]
```

prod であれば m6i.large インスタンスを構築

CloudFormation スタックの設計について

- CloudFormation 設計におけるスタック分割
- 環境ごとに再利用性可能なテンプレートの実現
- 同一構成のシステム間で再利用しやすいテンプレートの実現

同一構成のシステム間で再利用しやすいテンプレートの作成

- CloudFormation では、Amazon Resource Name (ARN) をテンプレートで指定したり、クロススタック参照のためにアカウント・リージョンごとに一意な値をエクスポートする場面がある
- これらの値をハードコーディングせず、擬似パラメータを利用して記述することでアカウントやリージョンを環境ごとに書き換えることが不要となり、同一構成のシステムを構築するためのテンプレートにおいて、再利用が容易となる
- エクスポート値は、\${AWS::StackName} を含めることで一意な値を生成可能

擬似パラメータの例:

変更前: !Sub 'arn:aws:ssm:us-east-1:111122223333:parameter/MyParameter'

変更後: !Sub 'arn:aws:ssm:\${AWS::Region}:\${AWS::AccountId}:parameter/MyParameter'

開発環境の整備

CloudFormation 専用の Linter cfn-lint の導入

- CloudFormation では、cfn-lint という Linter が GitHub でパブリック公開
- Pip、Homebrew (macOS)、Docker、各IDE (Visual Studio Code, IntelliJ IDEA, Sublime, Emacs, Vim etc.) 用の Plugin など、各個人環境に合わせて導入が可能

The screenshot shows a CloudFormation YAML template on the left and its analysis results on the right. A large orange arrow points from the code to the analysis results, labeled "コードを静的解析".

CloudFormation YAML Template:

```
1 AWSTemplateFormatVersion: "2010-09-09"
2
3 Resources:
4   SqsQueue:
5     Type: AWS::SQS::Queue
6     Properties:
7       DelaySeconds: 1000
8       Tag:
9         - Key: Name
10        Value: workshop-sqs-queue
11        - Key: ProjectName
12        Value: Example
13
14 Outputs:
15   SqsQueueURL:
16     Description: The URL of your Amazon SQS Queue
17     Value: !Ref SqsQueue
18
19   SqsQueueName:
20     Description: The name of your Amazon SQS Queue
21     Value: !GetAtt SqsQueue.Name
22
```

Analysis Results (PROBLEMS tab):

- ! sqs-queue.yaml code/workspace/linting-and-testing 5
 - ✖ [cfn-lint] E3034: Value has to be between 0 and 900 at Resources/SqsQueue/Properties/DelaySeconds [Ln 11, Col 7]
 - ✖ [cfn-lint] E3002: Invalid Property Resources/SqsQueue/Properties/Tag. Did you mean Tags? [Ln 12, Col 7]
 - ✖ [cfn-lint] E1010: Invalid GetAtt SqsQueue.Name for resource SqsQueueName [Ln 25, Col 5]
 - ⓘ [cfn-lint] I3011: The default action when replacing/removing a resource is to delete it. Set explicit values for UpdateReplacePolicy / DeletionPolicy on potentially stateful resource: Resources/SqsQueue [Ln 6, Col 3]
 - ⓘ [cfn-lint] I3013: The default retention period will delete the data after a pre-defined time. Set an explicit values to avoid data loss on resource : Resources/SqsQueue/Properties [Ln 8, Col 5]

aws

© 2023, Amazon Web Services, Inc. or its affiliates.

cfn-lint の利用方法

Visual Studio Code からの実行例

検出された箇所に下線が引かれ、
PROBLEMS パネルに自動的に問題を出力

A screenshot of the Visual Studio Code interface. The main editor shows a CloudFormation template with syntax highlighting and underlined errors. The bottom right corner of the editor shows the number '22'. Below the editor is the 'PROBLEMS' tab, which is active and shows five error messages. The 'OUTPUT' tab is also visible.

```
1 AWSTemplateFormatVersion: "2010-09-09"
2
3 Resources:
4   SqsQueue:
5     Type: AWS::SQS::Queue
6     Properties:
7       DelaySeconds: 1000
8       Tag:
9         - Key: Name
10        Value: workshop-sqs-queue
11        - Key: ProjectName
12        Value: Example
13
14 Outputs:
15   SqsQueueURL:
16     Description: The URL of your Amazon SQS Queue
17     Value: !Ref SqsQueue
18
19   SqsQueueName:
20     Description: The name of your Amazon SQS Queue
21     Value: !GetAtt SqsQueue.Name
22
```

PROBLEMS 5 OUTPUT DEBUG CONSOLE TERMINAL PORTS TRANSFORMATION HUB CODE REFERENCE LOG GIT

- ! sqs-queue.yaml code/workspace/linting-and-testing 5
 - ✖ [cfn-lint] E3034: Value has to be between 0 and 900 at Resources/SqsQueue/Properties/DelaySeconds [Ln 11, Col 7]
 - ✖ [cfn-lint] E3002: Invalid Property Resources/SqsQueue/Properties/Tag. Did you mean Tags? [Ln 12, Col 7]
 - ✖ [cfn-lint] E1010: Invalid GetAtt SqsQueue.Name for resource SqsQueueName [Ln 25, Col 5]
 - ⓘ [cfn-lint] I3011: The default action when replacing/removing a resource is to delete it. Set explicit values for UpdateRe
 - ⓘ [cfn-lint] I3013: The default retention period will delete the data after a pre-defined time. Set an explicit values to avoid



cfn-lint <https://github.com/aws-cloudformation/cfn-lint>

© 2023, Amazon Web Services, Inc. or its affiliates.

CloudFormation Rain の導入

CloudFormation Rain は、テンプレートのデプロイ、削除、生成、スタックの操作等を容易に実行するコマンドラインツール

- rain deploy コマンドにより、スタックへのデプロイを実行。削除は rain rm コマンドで実行。加えられる変更の概要を表示しつつ、スタックのデプロイ中に更新ステータスを表示。
- rain fmt コマンドにより、CloudFormation を標準フォーマットにフォーマットしたり、JSON/YAML 形式へ変更
- rain build コマンドにより、リソースのテンプレートを生成可能（次ページにて生成 AI 連携について記載）
- その他、StackSets の操作、デプロイ失敗の予測、ログの表示等、様々な機能をサポート



rain <https://github.com/aws-cloudformation/rain>

© 2023, Amazon Web Services, Inc. or its affiliates.

rain deploy コマンド実行後の出力

```
Deploying template 'resources.yaml' as stack 'resources' in us-east-1.  
Stack resources: CREATE_COMPLETE  
Successfully deployed resources
```

rain build AWS::EC2::VPC > vpc.yaml
コマンド実行後の vpc.yaml の一部

```
AWSTemplateFormatVersion: "2010-09-09"  
  
Description: Template generated by rain  
  
Resources:  
MyVPC:  
  Type: AWS::EC2::VPC  
Properties:  
  CidrBlock: CHANGEME # Optional  
  EnableDnsHostnames: false # Optional  
  EnableDnsSupport: false # Optional  
  InstanceTenancy: CHANGEME # Optional  
  Ipv4IpamPoolId: CHANGEME # Optional  
  Ipv4NetmaskLength: 0 # Optional  
Tags:  
  - Key: CHANGEME  
    Value: CHANGEME
```

Rain と Amazon Bedrock の連携

Rainは、Amazon Bedrock と Claude2 モデルを使用した生成AIによるテンプレート生成をサポート。Claude は日本語に対応。

事前に Bedrock のAnthropic Claude モデルを有効化

Base models (19)		
Models	Access status	Modality
AI21 Labs		
Jurassic-2 Ultra	Available to request	Text
Jurassic-2 Mid	Available to request	Text
Amazon		
Titan Embeddings G1 - Text	Available to request	Embedding
Titan Text G1 - Lite	Available to request	Text
Titan Text G1 - Express	Available to request	Text
Titan Image Generator G1 Preview	Available to request	Image
Titan Multimodal Embeddings G1	Available to request	Embedding
Anthropic		
Claude	Access granted	Text



cfn-lint <https://github.com/aws-cloudformation/cfn-lint>

© 2023, Amazon Web Services, Inc. or its affiliates.

作成したい構成を記載の上 rain build コマンドを実行

```
$ rain build -p "1つのVPCに2つのサブネット" > vpc_genai.yaml
```

コマンド実行後の vpc_genai.yaml の一部

```
AWSTemplateFormatVersion: 2010-09-09
Description: A sample template

Resources:

  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 10.0.0.0/16
      EnableDnsHostnames: true
      Tags:
        - Key: Name
          Value: !Ref AWS::StackName

  PublicSubnet1:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId: !Ref VPC
      CidrBlock: 10.0.1.0/24
      AvailabilityZone: !Select [ 0, !GetAZs '' ]
```

Amazon CodeWhisperer の活用

- CodeWhisperer を利用すると、コメント行に記載した構成を実現するためのコードサジェストやコードの補完が可能
- CloudFormation の YAML / JSON のセキュリティスキャンに対応
- エディタに AWS Toolkit をインストールし、AWS Builder ID でサインインすることで利用を開始

```
60  # Two public subnets, where containers can have public IP addresses
61  PublicSubnetOne:
62    Type: AWS::EC2::Subnet
63    Properties:
64      AvailabilityZone:
65        Fn::Select:
66          - 0
67          - Fn::GetAZs: {Ref: 'AWS::Region'}
68    VpcId: !Ref 'VPC'
69    CidrBlock: !FindInMap ['SubnetConfig', 'PublicOne', 'CIDR']
70    MapPublicIpOnLaunch: true
71
72  Type: AWS::EC2::Subnet
Properties:
  AvailabilityZone:
    Fn::Select:
      - 1
      - Fn::GetAZs: {Ref: 'AWS::Region'}
  VpcId: !Ref 'VPC'
  CidrBlock: !FindInMap ['SubnetConfig', 'PublicTwo', 'CIDR']
  MapPublicIpOnLaunch: true
```

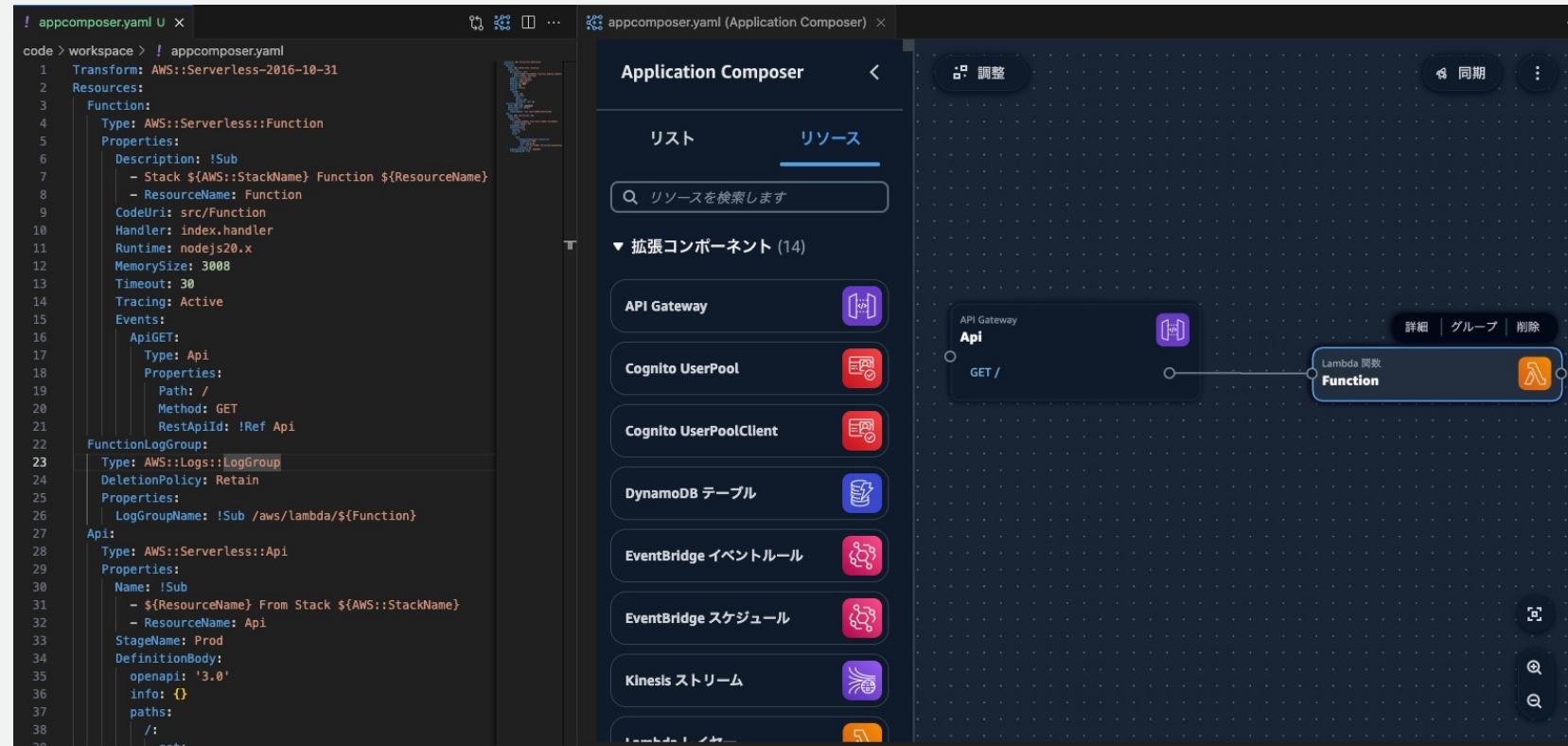
Amazon CodeWhisperer にて AI を活用した新しいコード修正、IaC サポート、および Visual Studio との統合提供を開始
<https://aws.amazon.com/jp/blogs/news/amazon-codewhisperer-offers-new-ai-powered-code-remediation-iac-support-and-integration-with-visual-studio/>



AWS Application Composer の活用

Application Composer は、Visual Studio CodeやAWS コンソールから任意のCloudFormationリソースをビジュアルキャンバスにドラッグアンドドロップし、CloudFormationテンプレートを自動生成するサービス

- VS Code の拡張機能 AWS Toolkit for Visual Studio Code として提供
- 14の拡張コンポーネントに加え、1,000を超える標準リソースをサポート
- 同期 (Sync) ボタンをワンクリックで CloudFormation スタックとしてデプロイ
- 生成 AI と連携し、リソース設定のためのコードの提案が可能
- 既存のコードの可視化に利用可能



IDE extension for AWS Application Composer enhances visual modern applications development with AI-generated IaC
<https://aws.amazon.com/jp/blogs/aws/ide-extension-for-aws-application-composer-enhances-visual-modern-applications-development-with-ai-generated-iac/>



テスト



CloudFormation の継続的テストについて

- CloudFormation のテストツールや AWS サービスを用いて、テストの効率化を目指す
- 目的や開発プロセスに応じて、下記のテストツールの CI/CD 上への組み込みを検討
- cfn-lint
- Guard
- Hooks
- AWS Control Tower プロアクティブコントロール
- AWS Configルール プロアクティブなコンプライアンス
- TaskCat

*Guard、Hooks、AWS Control Tower、AWS Configルールを用いると、組織の共通ルールを定めた上で開発全体に統制をかけることも可能

AWS CloudFormation Guard の導入

- セキュリティ、ガバナンス、ポリシーコンプライアンスの要件を設定し、AWS 環境を中心集権的に管理する際に CloudFormation Guard が利用可能
- Guard を用いて Policy as Code を実現すると、開発者のローカル環境やデリバリーパイプラインの継続的インテグレーション (CI) フェーズでルールに即しているかどうかの検証が可能



AWS CloudFormation Guard の利用方法

- CloudFormation のテンプレートに対し、値を確認するための Rule を記述した .guard ファイルを用意
- cfn-guard validate コマンドでテンプレートとルールを指定し、テストを実行

S3 Bucket 作成のテンプレート

```
AWSTemplateFormatVersion: "2010-09-09"
```

Resources:

 SampleBucket:

 Type: AWS::S3::Bucket

Properties:

 BucketEncryption:

 ServerSideEncryptionConfiguration:

 - ServerSideEncryptionByDefault:

 SSEAlgorithm: AES256

 VersioningConfiguration:

 Status: Enabled

S3 Bucket リソースタイプのチェックを行う Guard Rule
SSEAlgorithm と Versioning の値をチェック

```
AWS::S3::Bucket {  
  Properties {  
    BucketEncryption.ServerSideEncryptionConfiguration[*] {  
      ServerSideEncryptionByDefault.SSEAlgorithm == 'AES256'  
      <>BucketEncryption not configured with the AES256 algorithm>>  
    }  
    VersioningConfiguration.Status == 'Enabled'  
    <>BucketEncryption not configured with versioning enabled>>  
  }  
}
```

cfn-guard validateコマンド実行後の出力

```
example_bucket.yaml Status = PASS  
PASS rules  
example_bucket.guard/default PASS
```



Guard ルール記述時のポイント

ルール記述の際には、以下の構文を利用

構文: <query> <operator> [query|value literal] [custom message]

説明	必須	
query	ドット (.) で区切られた式で階層化し、評価対象を指定	必須
operator	queryの状態を確認するための単項演算子または二項演算子 二項演算子: == (等しい) != (等しくない) > (より大きい) >= (以上) < (より小さい) <= (以下) IN (リスト内) 単項演算子: exists empty is_string is_list is_struct not (!)	必須
query value literal	クエリや条件で利用する値リテラルを記述 string, integer(64), float(64), bool, char, regexといった値の他に、 範囲を示す r[<下限>, <上限>] といった記述も可能	二項演算子を使用する 場合は必須
custom message	関する情報を提供する文字列。メッセージはvalidateコマンドと testコマンドの詳細出力に表示され、ルール評価の理解やデバッグに役立つ	任意

*複数のルールを記述した場合、暗黙的にAND条件で判断される。構文の後ろにORを指定することが可能

構文: <query> <operator> [query|value literal] [custom message] [or|OR]



Guard ルールの記述例

単項演算子を利用したときの利用例

- S3Bucket に BucketEncryption プロパティが定義されているかどうかをチェック
- BucketEncryption プロパティが定義されていれば評価は PASS

```
Resources.S3Bucket.Properties.BucketEncryption exists
```

二項演算子を利用したときの利用例

- VolumeType プロパティに指定された値が io2、gp3 であるかどうかをチェック
- NewVolume にいずれかが指定されていれば、評価は PASS

```
Resources.NewVolume.Properties.NewVolume.VolumeType IN ['io2','gp3']
```

custom messageの利用例

- エラー時に、メッセージとして「Allowed Volume Types are io2 and gp3」を表示

```
Resources.NewVolume.Properties.VolumeType IN [ 'io2','gp3' ]  
<<Allowed Volume Types are io2 and gp3>>
```



モジュール性と再利用性を考慮した Guard ルール記述

モジュール化された Guard ルールを作成すると、再利用性が向上するだけでなく、データを検証したときに失敗したルールを特定したり、ルールをトラブルシューティングするのが簡単

- 条件を定義する `when` を利用し、S3 バケットが存在する場合にのみルールを実行するように記述
- 1つのロジックを2つのルール (Named-rule block) に分解して記述

```
AWS::S3::Bucket {
  Properties {
    BucketEncryption.ServerSideEncryptionConfiguration[*] {
      ServerSideEncryptionByDefault.SSEAlgorithm == 'AES256'
      <<BucketEncryption not configured with the AES256 algorithm>>
    }
    VersioningConfiguration.Status == 'Enabled'
    <<BucketEncryption not configured with versioning enabled>>
  }
}
```



```
let my_buckets = Resources.*[ Type == 'AWS::S3::Bucket' ]

rule validate_bucket_sse when %my_buckets !empty {
  %my_buckets.Properties {
    BucketEncryption.ServerSideEncryptionConfiguration[*] {
      ServerSideEncryptionByDefault.SSEAlgorithm == 'AES256'
      <<BucketEncryption not configured with the AES256 algorithm>>
    }
  }
}

rule validate_bucket_versioning when %my_buckets !empty {
  %my_buckets.Properties {
    VersioningConfiguration.Status == 'Enabled'
    <<BucketEncryption not configured versioning enabled>>
  }
}
```



Guard を利用したユニットテスト

Guard ではルールのテストを作成し、ルールが結果通りに機能することを検証可能

テストケースと期待する結果を記載したyamlファイル

```
- input:
  Resources:
    MyExampleBucket:
      Type: AWS::S3::Bucket
      Properties:
        BucketEncryption:
          ServerSideEncryptionConfiguration:
            - ServerSideEncryptionByDefault:
              SSEAlgorithm: AES256
  expectations:
    rules:
      validate_bucket_sse_example: PASS

- input:
  Resources:
    MyExampleBucket:
      Type: AWS::S3::Bucket
      Properties:
        VersioningConfiguration:
          Status: Suspended
  expectations:
    rules:
      validate_bucket_versioning_example: FAIL
```

Guard ルールを記載したファイル

```
let my_buckets = Resources.*[ Type == 'AWS::S3::Bucket' ]

rule validate_bucket_sse_example when %my_buckets !empty {
  %my_buckets.Properties {
    BucketEncryption.ServerSideEncryptionConfiguration[*] {
      ServerSideEncryptionByDefault.SSEAlgorithm == 'AES256'
      <<BucketEncryption not configured with the AES256 algorithm>>
    }
  }
}

rule validate_bucket_versioning_example when %my_buckets !empty {
  %my_buckets.Properties {
    VersioningConfiguration.Status == 'Enabled'
    <<BucketEncryption not configured versioning enabled>>
  }
}
```

cfn-guard testコマンド実行後の出力

```
Test Case #1
No Test expectation was set for Rule validate_bucket_versioning_example
PASS Rules:
  validate_bucket_sse_example: Expected = PASS

Test Case #2
No Test expectation was set for Rule validate_bucket_sse_example
PASS Rules:
  validate_bucket_versioning_example: Expected = FAIL
```



AWS CloudFormation Hooks の導入

- CloudFormation によるスタックやリソース作成/更新/削除の前に処理を実行するために、CloudFormation Hooks の利用が可能
- カスタムロジックを呼び出してアクションを自動化したり、リソース設定を検査することが可能
- CloudFormation Public Registry に公開されているサンプル Hooks を利用したり、CloudFormation CLIを使用してHooksを作成し、CloudFormation Private Registry に公開

*Hooksの利用方法の詳細は、「AWS Blackbelt CloudFormationレジストリ編」を参照



Guard と Hooks の違い

Guard

- CloudFormation テンプレートが特定のルールやポリシーに従っているかを検証
- CloudFormation テンプレートがデプロイされる前に、そのテンプレートが定義されたルールセットに適合しているかどうかをチェック。主にビルドステージの静的分析のフェーズで実施

Hooks

- CloudFormation のデプロイプロセス中に特定のリソースに対してカスタム検証ルールを適用し、リソースがデプロイされる前に、特定の要件やポリシーに従っているかを検証
- スタックの作成、更新、削除のプロセス中にトリガーされ、Lambda 関数を使用してリソース設定を検証し、カスタムロジックによるアクションの自動化を実施

AWS Control Tower や AWS Config ルールによるガードレールや事前テストの実現

AWS Control Tower プロアクティブコントロール

- すべての CloudFormation デプロイメントに自動的に適用され、準拠していないリソースが プロビジョニングされる前にブロック
- AWS CloudFormation Hooks によりプロビジョニングのブロックを実装

AWS Config ルール プロアクティブなコンプライアンス

- AWSリソースをプロビジョニングする前に、事前定義した AWS Config ルールに準拠しているかどうかを事前に確認する機能
- CloudFormation Hooks を使用し、実際のデプロイが行われる前に設定の事前確認が可能

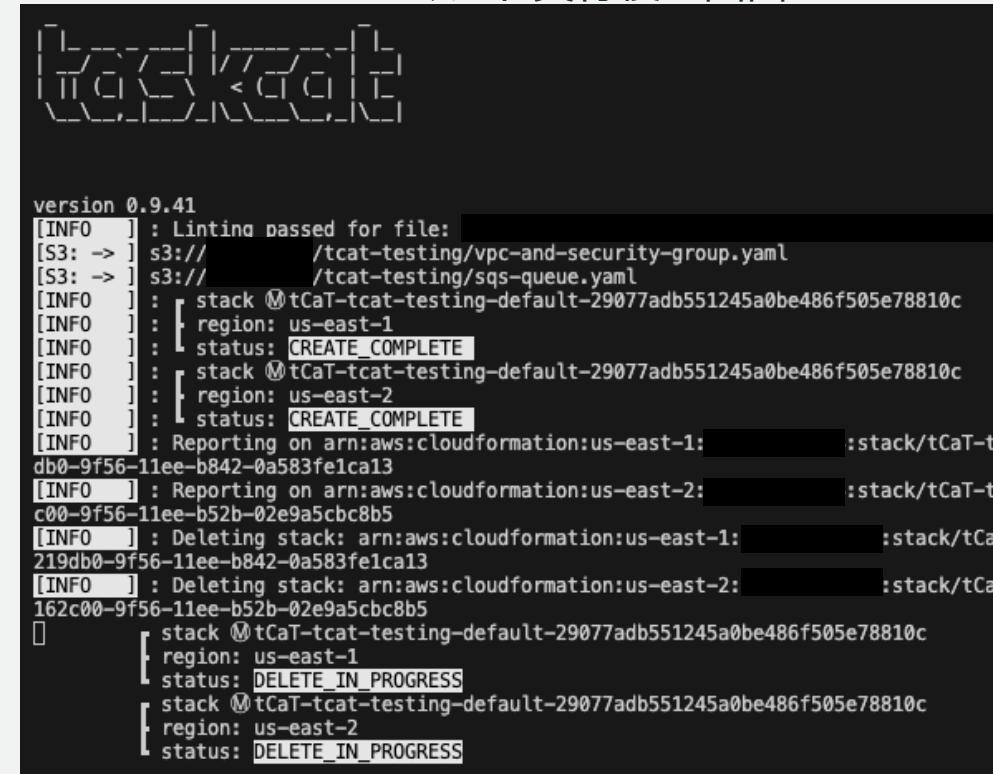
TaskCat によるインテグレーションテスト

- 指定した AWS リージョンに CloudFormation テンプレートを実行し、それらが期待通りに機能するかどうかを検証するインテグレーションテストのために TaskCat が利用可能
- 指定した AWS リージョンに対し、テンプレートからスタックを作成し、成功/失敗のレポートを出力した後にスタックを削除
- デプロイ先のリージョンを複数指定することによる並列実行も可能

テストの実行内容を記載した
.taskcat.yml ファイル

```
project:  
  name: tcat-testing  
  package_lambda: false  
  
  regions:  
    - us-east-1  
    - us-east-2  
      テストを実行したい  
      リージョンを指定  
tests:  
  default:  
    template: ./vpc-and-security-group.yaml  
  
general:  
  s3_bucket: tcat-test  
parameters:  
  VpcIpv4Cidr: 172.16.0.0/16
```

taskcat test run コマンド実行後の画面



```
version 0.9.41  
[INFO ] : Linting passed for file: /tcat-testing/vpc-and-security-group.yaml  
[S3: -> ] s3:// tcat-testing/vpc-and-security-group.yaml  
[S3: -> ] s3:// tcat-testing/sqs-queue.yaml  
[INFO ] : stack @tCat-tcat-testing-default-29077adb551245a0be486f505e78810c  
[INFO ] : [region: us-east-1  
[INFO ] : [status: CREATE_COMPLETE  
[INFO ] : stack @tCat-tcat-testing-default-29077adb551245a0be486f505e78810c  
[INFO ] : [region: us-east-2  
[INFO ] : [status: CREATE_COMPLETE  
[INFO ] : Reporting on arn:aws:cloudformation:us-east-1: :stack/tCat-t  
db0-9f56-11ee-b842-0a583fe1ca13  
[INFO ] : Reporting on arn:aws:cloudformation:us-east-2: :stack/tCat-t  
c00-9f56-11ee-b52b-02e9a5cbc8b5  
[INFO ] : Deleting stack: arn:aws:cloudformation:us-east-1: :stack/tCa  
219db0-9f56-11ee-b842-0a583fe1ca13  
[INFO ] : Deleting stack: arn:aws:cloudformation:us-east-2: :stack/tCa  
162c00-9f56-11ee-b52b-02e9a5cbc8b5  
[INFO ] : stack @tCat-tcat-testing-default-29077adb551245a0be486f505e78810c  
[INFO ] : [region: us-east-1  
[INFO ] : [status: DELETE_IN_PROGRESS  
[INFO ] : stack @tCat-tcat-testing-default-29077adb551245a0be486f505e78810c  
[INFO ] : [region: us-east-2  
[INFO ] : [status: DELETE_IN_PROGRESS
```



デプロイ



CloudFormation のデプロイ方法について

CloudFormation では、以下の方法でデプロイを検討

- マネジメントコンソールや CLI から手動でテンプレートをデプロイ
 - CI/CD の整備をするほどの頻度や規模でない・ヒューマンエラーによるリスクが少ない等、自動化の恩恵を受けづらい時に利用
- CI/CD パイプラインからデプロイ
 - テストも含めた自動化を実現したい時に利用
- Git からの同期
 - Git ベースでコード管理しており、開発用途で CD を簡易に自動化したい時に利用

CloudFormation のデプロイ方法

- マネジメントコンソールや CLI から手動でテンプレートをデプロイ
- CI/CD パイプラインからデプロイ
- Git からの同期

マネジメントコンソールや CLI から手動でテンプレートをデプロイ

- S3 経由やマネジメントコンソール上からテンプレートファイルの直接アップロードによりスタックの作成や更新が可能
- CLI で更新する場合、deploy コマンドを利用することで変更セット作成後にスタックの作成や更新が可能
- 変更セットを利用し、デプロイの前にリソースの更新や削除を把握
- deploy コマンドを利用し、変更セットを確認してから変更を反映したい場合は、`--no-execute-changeset` フラグを利用

The screenshot shows the AWS CloudFormation console interface. At the top, there's a section titled "テンプレートの指定" (Specify template) with a note: "テンプレートは、スタックのリソースおよびプロパティを表す JSON または YAML ファイルです。" (A template is a JSON or YAML file that defines the resources and properties of a stack). Below this, there are three options for specifying the template source:

- Amazon S3 URL: "テンプレートに Amazon S3 URL を指定します。"
- テンプレートファイルのアップロード: "テンプレートをコンソールに直接アップロードします。"
- Git から同期 - 新規: "Git リポジトリからテンプレートを同期します。"

Below these options is a field labeled "Amazon S3 URL" containing the value "https://". A note below it says "Amazon S3 テンプレートの URL". To the right of this field is a button labeled "デザイナーで表示" (View in Designer).

At the bottom of the screenshot, there's a "Stack Actions" dropdown menu with the following items:

- C: Create (disabled)
- 削除: Delete
- 更新: Update
- Stack Actions ▲: Stack Actions ▲ (dropdown menu)
- Stack Actions ▼: Stack Actions ▼ (dropdown menu)

The dropdown menu contains the following items:

- 削除保護を編集する: Edit deletion protection
- ドリフト結果を表示: View drift results
- ドリフトの検出: Detect drift
- 既存スタックの変更セットを作成: Create changeset for existing stack (highlighted in blue)
- Stack Actions ▼ (dropdown menu)
- Stackへのリソースのインポート: Import resources to stack

CloudFormation のデプロイ方法

- ・ マネジメントコンソールや CLI から手動でテンプレートをデプロイ
- ・ CI/CD パイプラインからデプロイ
- ・ Git からの同期

CI/CD パイプラインからデプロイ

CI/CDパイプライン経由で、テストやアプリケーションのビルド、デプロイが可能

CFn スタックをデプロイするパイプラインの例

- 1 Source → 2 Build → 3 テストスタックデプロイ → 4 本番スタックデプロイ



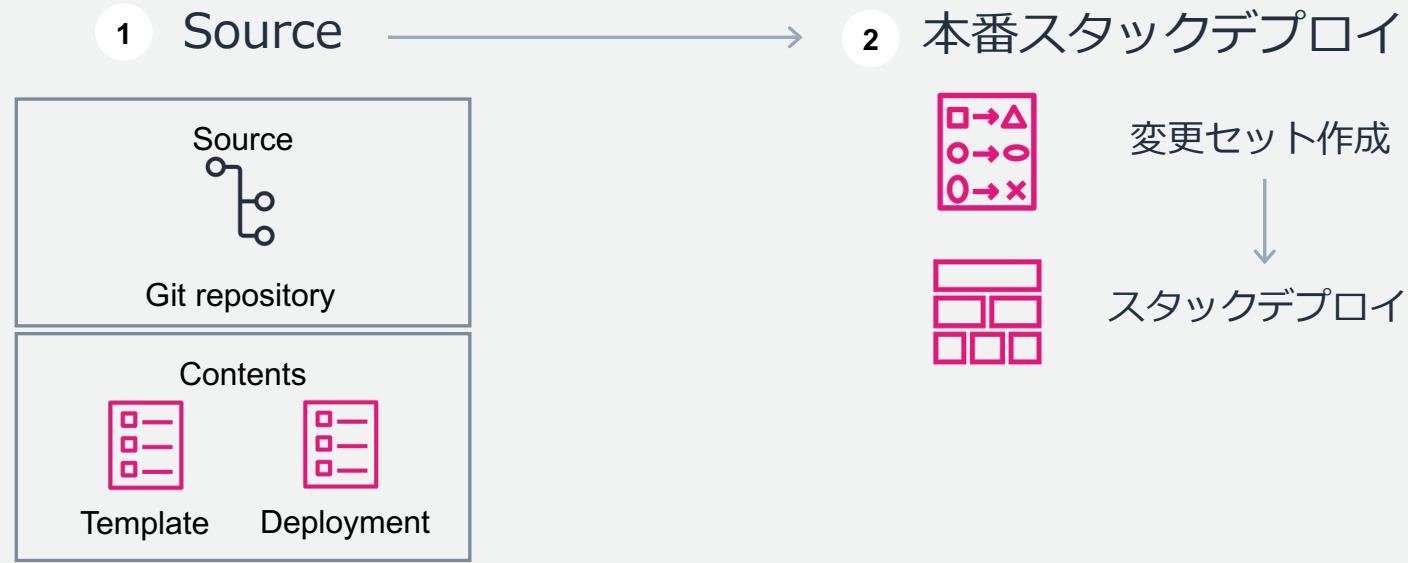
CodePipelineのCloudFormationアクションを利用すると、任意のコードは実行できないことから、よりセキュアにスタックデプロイが可能
通常は、スタックデプロイをするコンピュートリソースにAdmin相当の権限を与えることが必須



CloudFormation のデプロイ方法

- マネジメントコンソールや CLI から手動でテンプレートをデプロイ
- CI/CD パイプラインからデプロイ
- Git からの同期

Git からの同期



- Git と連携させて CloudFormation テンプレートをデプロイする際に利用
- CloudFormation テンプレートファイルと、スタックを設定するパラメータを含むスタックデプロイメントファイルの2つに変更がないかを監視し、Git リポジトリ上で更新があればスタックの作成、更新を自動的に実施
- デプロイ前に変更セットを作成
- GitHub、GitHub Enterprise、GitLab、Bitbucket のリポジトリをサポート (2023年12月現在)

Git からの同期設定 (1/4)

デベロッパー用ツールの設定より、事前に、利用している Git リポジトリへの接続を作成

The screenshot shows the AWS DevOps Tools settings interface. On the left, a sidebar lists various tools: Sources • CodeCommit, Artifacts • CodeArtifact, Builds • CodeBuild, Deploys • CodeDeploy, Pipelines • CodePipeline, and Settings (which is expanded to show Notifications and Connections). The main area is titled "Create a connection" and "Create a provider". It lists four options: Bitbucket (selected), GitHub, GitHub Enterprise Server, and GitLab. Below this, a section titled "Create a Bitbucket connection" asks for a connection name, which is currently empty. At the bottom right of this section is a large orange button labeled "Connect to Bitbucket".

Git からの同期設定 (2/4)

スタックの作成画面->テンプレートの指定より
「Gitから同期」を選択し、画面の指示に従って
値を設定

スタックの作成

前提条件 - テンプレートの準備

テンプレートの準備
各スタックはテンプレートに基づきます。テンプレートとは、スタックに含む AWS リソースに関する設定情報を含む JSON または YAML ファイルです。

テンプレートの準備完了 サンプルテンプレートを使用 デザイナーでテンプレートを作成

テンプレートの指定

テンプレートは、スタックのリソースおよびプロパティを表す JSON または YAML ファイルです。

テンプレートソース
テンプレートを選択すると、保存先となる Amazon S3 URL が生成されます。

Amazon S3 URL
テンプレートに Amazon S3 URL を指定します。 テンプレートファイルのアップロード
テンプレートをコンソールに直接アップロードします。 Git から同期 - 新規
Git リポジトリからテンプレートを同期します。

⚠️ Git プロバイダーを使用してリソースをプロジェクトするにあたり、AWS は CodeStar Connections API を使用します。これはコンプライアンスプログラムの対象外です。詳細については、「[コンプライアンスプログラムによる対象範囲内の AWS のサービス](#)」を参照してください。

Git sync は、プルリクエストを送信してスタックを Git リポジトリに接続します。この設定を作成したら、プルリクエストを Git リポジトリにマージします。

キャンセル

次へ



テンプレート定義リポジトリ 情報

CloudFormation と同期するスタックテンプレートを含む Git リポジトリを選択してください。

- リンクされている Git リポジトリを選択する
CloudFormation とリンクしている Git リポジトリを選択してください。

- Git リポジトリをリンクする
CloudFormation とリンクする Git リポジトリを選択してください。

リポジトリプロバイダーを選択する

GitHub

GitHub Enterprise Server

GitLab

Bitbucket

接続

テンプレートまたは [新しい接続を作成する](#) を含むリポジトリを含む Git アカウントへの接続を選択します。

GitHubConnect

リポジトリ

テンプレートを含むリポジトリを選択します。

cfn-gitsync

プランチ

CloudFormation は、このプランチからの変更をテンプレートに同期します。

main

デプロイファイルのパス

テンプレートのデプロイパラメータを格納するファイルのパスを指定します。

deployment.yaml

IAM ロール

この IAM ロールは、CloudFormation に Git リポジトリからスタックを更新するためのアクセス許可を提供します。「[IAM ロールの前提条件](#)」を参照してください。

- 新しい IAM ロール
アカウントにロールを作成します。

- Existing IAM role
アカウント内の既存のロールを選択してください。

ロール名

cfn-git-sync-role

文字、数字、またはハイフンのみを使用してください。最大長は 100 文字です。



Git からの同期設定 (3/4)

Git と同期ステータスが有効となると、コードリポジトリに対してPull Request を発行

The screenshot shows the AWS CloudFormation Git Sync configuration interface. At the top, there are buttons for '編集' (Edit), '最新のコミットを再試行' (Re-run latest commit), '接続解除' (Disconnect), and a close button. Below this, the configuration details are listed:

リポジトリ	デプロイファイルのパス	Git と同期
cfn-gitsync	deployment.yaml	☑ 有効
リポジトリプロバイダー	リポジトリ同期のステータス	プロビジョニングのステータス
GitHub	⌚ 開始済み	⌚ 開始済み
プランチ	リポジトリ同期のステータスのメッセージ	
main	Starting syncs for commit b0ca29af1cebe099a6a3bb47d0ce8018f740c 915	

AWS が発行した Pull Request をユーザー側で Merge

The screenshot shows a GitHub Pull Request titled "Add AWS Cloudformation Deployment file for gitsync-stack stack". The pull request has been merged by the bot "aws-connector-for-github". The commit message is: "This pull request commits a CloudFormation [stack deployment file](#) to your repository. AWS CloudFormation uses the `deployment.yaml` file to locate, track, and automatically update a stack that belongs to the `gitsync-stack` template. When this pull request is merged, AWS CloudFormation tracks changes to this repository and applies updates to the stack and parameters that are defined in the file." A note below states: "When this change is merged, AWS CloudFormation tracks and syncs changes from this repository to the stack defined in `deployment.yaml`. Make sure to review it. You can disable syncing at any time in the AWS CloudFormation console." Below the note, there are sections for "How does it work?" and "How do I update the `deployment.yaml` file?". The commit hash shown is 11fbcb0.



Git からの同期設定 (4/4)

指定した Git リポジトリのブランチに更新をかけると、変更セットを作成し、スタックを更新

Git と同期 - 新規

リンクされた Git リポジトリからスタックの更新を設定および同期します。

リポジトリ cfn-gitsync	デプロイファイルのパス deployment.yaml	Git と同期 ☑ 有効
リポジトリプロバイダー GitHub	リポジトリ同期のステータス ⌚ 成功しました	プロジェクトニングのステータス ⌚ 進行中
ブランチ main	リポジトリ同期のステータスのメッセージ Starting syncs for commit 003191efb52bf4b9823549eae06ea14480e077da	

最新の同期イベント (5)

日付	コミット ID	イベント	イベントのタイプ
2023-12-11 22:07:25 UTC+0900	003191ef	Waiting for changeset execution to finish.	⌚ CHANGESSET_EXECUTION_IN_PROGRESS
2023-12-11 22:07:24 UTC+0900	003191ef	Changeset creation succeeded.	⌚ CHANGESSET_CREATION_SUCCEEDED
2023-12-11 22:07:08 UTC+0900	003191ef	Creating a changeset.	⌚ CHANGESSET_CREATION_IN_PROGRESS
2023-12-11 22:07:08 UTC+0900	003191ef	Uploaded CloudFormation template.	⌚ TEMPLATE_BUNDLED
2023-12-11 22:07:07 UTC+0900	003191ef	Clone started	⌚ CLONE_STARTED

まとめ

- CloudFormation スタックとコード設計の勘所
 - CloudFormation 設計におけるスタック分割
 - 環境ごとに再利用性可能なテンプレートの実現
 - 同一構成のシステム間で再利用しやすいテンプレートの作成
- 開発環境の整備
 - テンプレート開発支援ツールの導入
cfn-lint / Rain / CodeWhisperer / Application Composer
- テスト
 - テストツール導入
cfn-lint / Guard / Hooks / TaskCat
AWS Control Tower プロアクティブコントロール
AWS Config ルール プロアクティブなコンプライアンス
- デプロイ
 - マネジメントコンソールや CLI から手動でテンプレートをデプロイ
 - CI/CD パイプラインからデプロイ
 - Git からの同期



Thank you!



AWS CloudFormation

Dive Deep 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 の深い機能を知りたい方

前提知識

- AWS の基本的な概要や操作を理解していること
- CloudFormation の用語 (スタック、テンプレート、変更セットなど) を理解していること

本セミナーのゴール

- カスタムリソースやマクロなど、深い機能について理解する

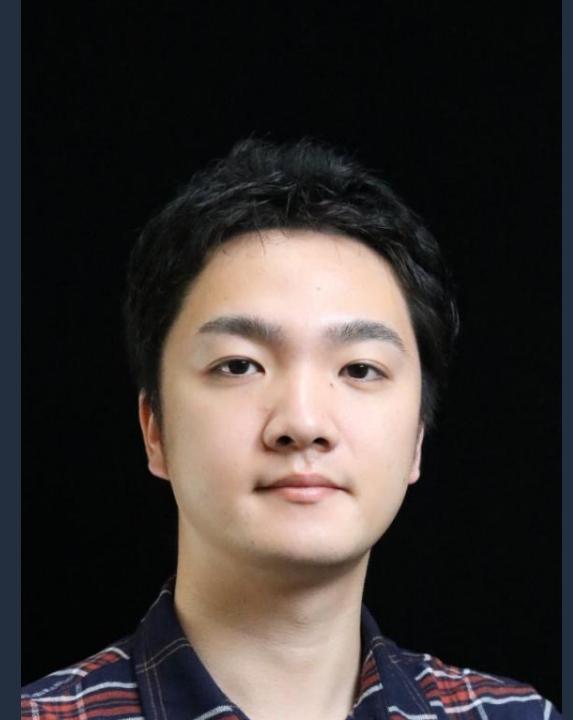
自己紹介

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



Deep な機能の使い方

Deep な機能の使い方

- ・任意の処理を追加する（カスタムリソース）
- ・スタック作成/更新時にテンプレートを加工する（マクロ）
- ・スタック作成権限とリソースの保護
- ・CodePipelineからCFnスタックをデプロイする

本資料では AWS CloudFormation を CFn と略記します

Deep な機能の使い方

- ・任意の処理を追加する（カスタムリソース）
- ・スタック作成/更新時にテンプレートを加工する（マクロ）
- ・スタック作成権限とリソースの保護
- ・CodePipelineからCFnスタックをデプロイする

任意の処理を追加する（カスタムリソース）

使い方

- ・ サービストークン (Lambda 関数、SNS トピック) を実装する
 - ・ スタックの処理に応じて Create/Update/Delete に対応するイベントが送信されるため、整合性が保たれるよう実装する
 - ・ サービストークンでの処理後、CFn にレスポンスを行うことで後続の処理が進む
- ・ Type を AWS::CloudFormation::CustomResource あるいは Custom::MyCustomResourceTypeName としたリソースを定義する

ユースケース

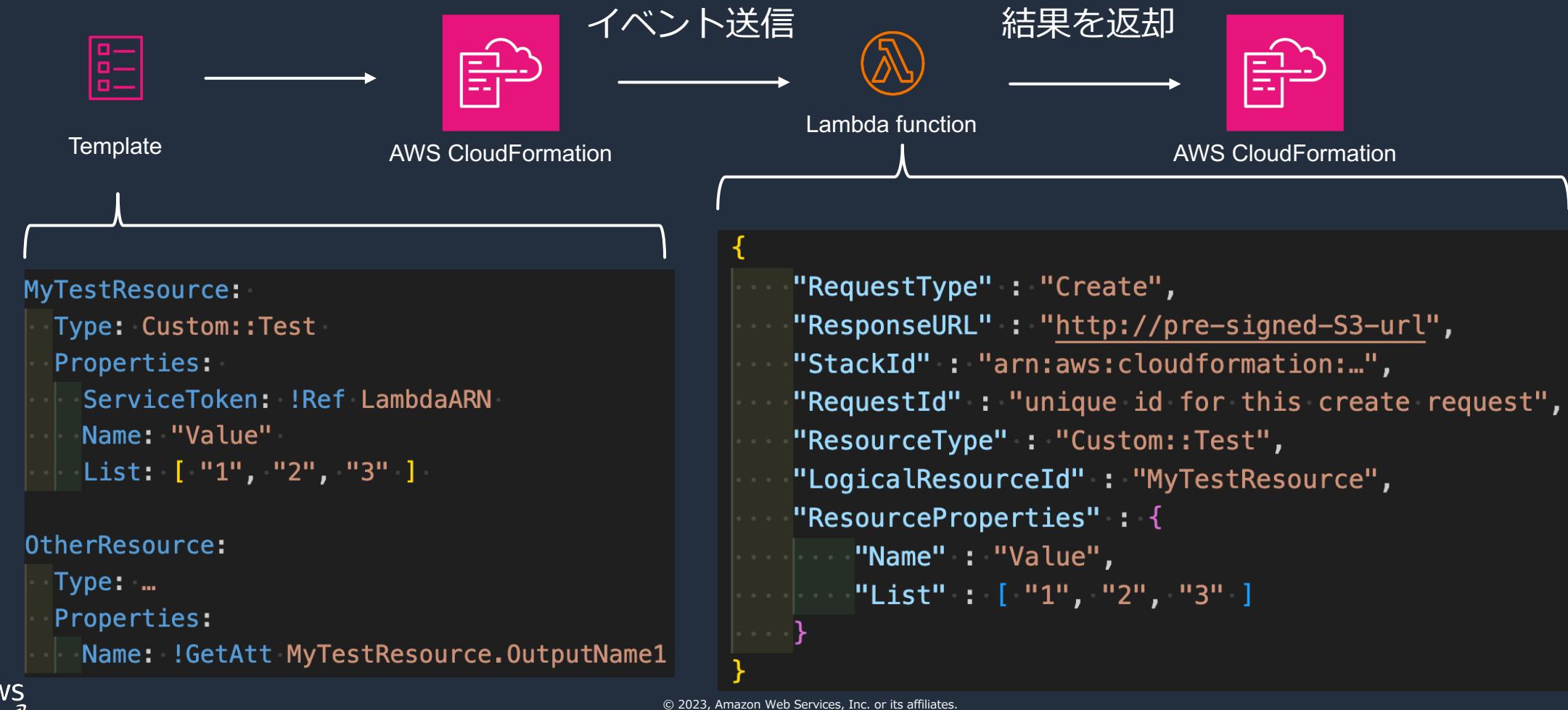
- ・ CFn で未対応のリソースを Lambda 関数から SDK (API) 経由で操作する
- ・ リソースの返り値にない値をスタック内で使用する

Tips

- ・ 実装を容易にする[ヘルパーツール](#)や[実装のベストプラクティス](#)が提供されている

任意の処理を追加する（カスタムリソース）

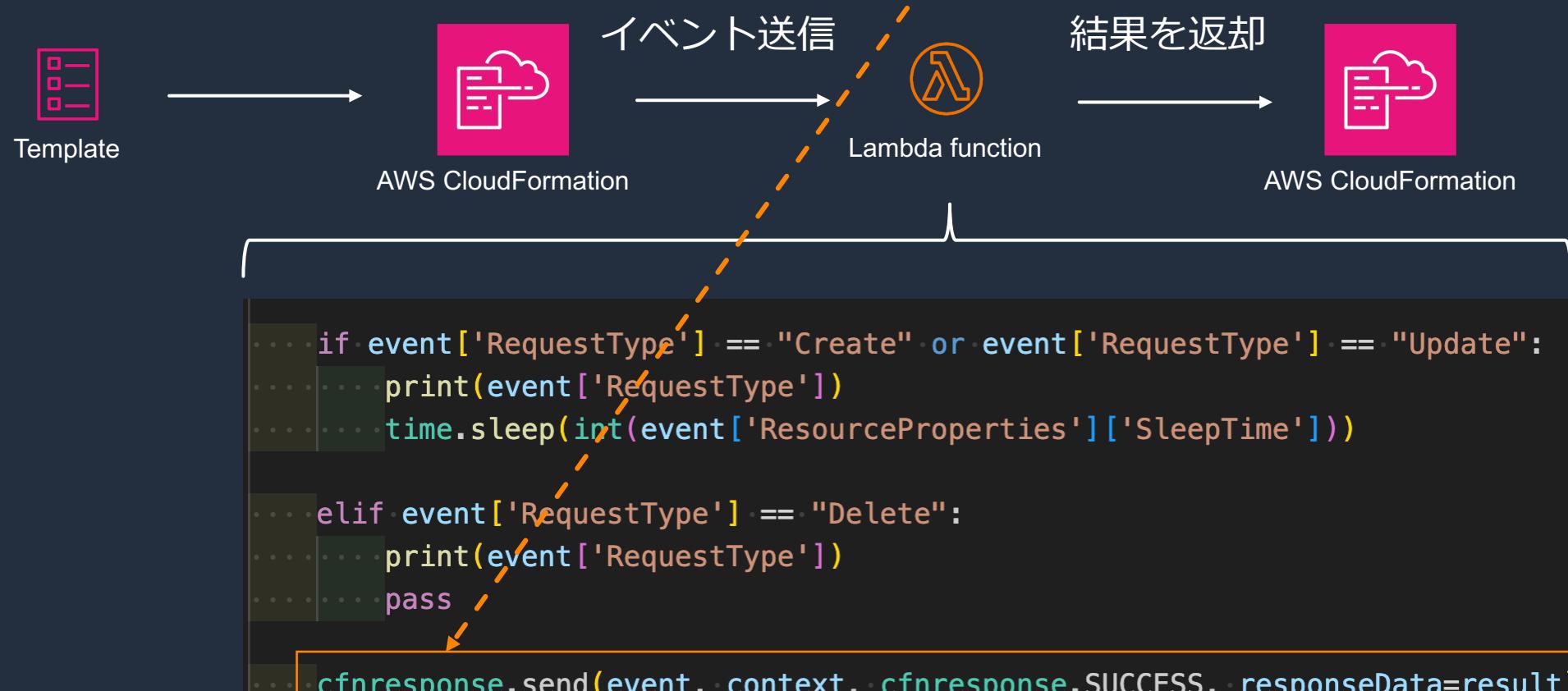
スタックの処理中にサービストークン（Lambda か SNS）を実行できる



任意の処理を追加する（カスタムリソース）

スタックの処理中にサービストークン（Lambda か SNS）を実行できる

- 送られたイベントとともに処理を実行後、CFn に結果を送信する
- CloudFormation から Lambda 関数を作成すると、cfnresponse モジュールで結果を返却できる



任意の処理を追加する（カスタムリソース）

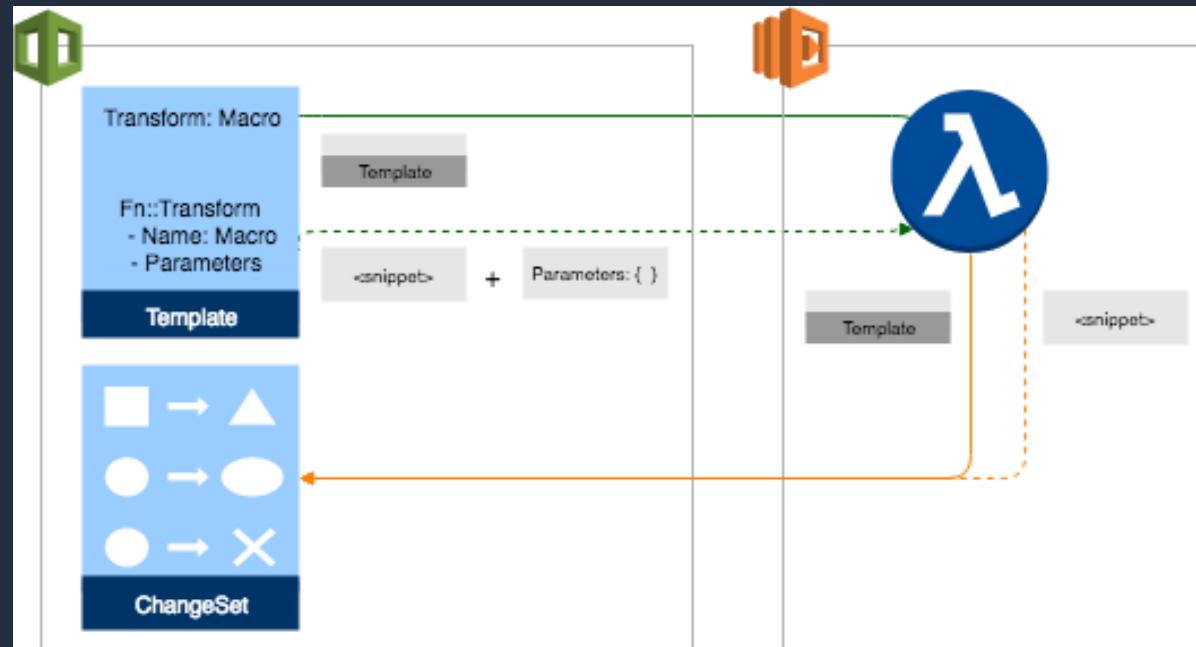


Deep な機能の使い方

- ・任意の処理を追加する（カスタムリソース）
- ・スタック作成/更新時にテンプレートを加工する（マクロ）
- ・スタック作成権限とリソースの保護
- ・CodePipelineからCFnスタックをデプロイする

スタック作成/更新時にテンプレートを加工する(マクロ)

- ・ スタック実行前にテンプレートを Lambda で加工できる
 - ・ 事前に Lambda 関数とマクロのリソースを作成することで使用可能となる
 - ・ AWS が事前に用意しているものもある
 - ・ 変更前後のテンプレートをそれぞれ Original/Processed と呼ぶ

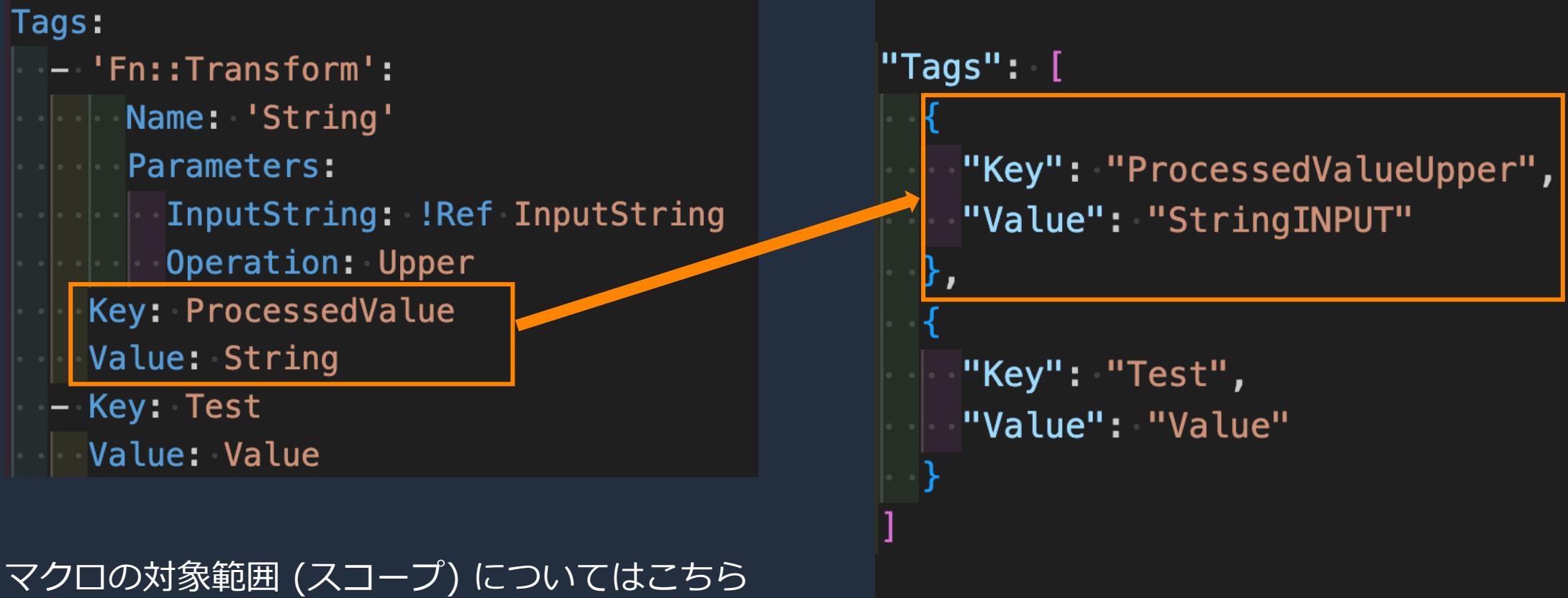


https://docs.aws.amazon.com/ja_jp/AWSCloudFormation/latest/UserGuide/template-macros.html

スタック作成/更新時にテンプレートを加工する(マクロ)

Template - Original —————>  —————> Template - Processed

Lambda function



マクロの対象範囲(スコープ)についてはこちら

<https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/template-macros.html#template-macros-use>

スタック作成/更新時にテンプレートを加工する(マクロ)

Template - Original

```
'Fn::Transform':  
  - Name: 'String'  
  - Parameters:  
    - InputString: !Ref InputString  
    - Operation: Upper  
  - Key: ProcessedValue  
  - Value: String
```

Lambda が返却する response の内容

```
{  
  - requestId: "dcd...",  
  - status: "success",  
  - fragment: {  
    - Key: "ProcessedValueUpper",  
    - Value: "StringINPUT"  
  }  
}
```



Lambda が受け取る event の内容

```
{  
  - accountId: "123...",  
  - fragment: {  
    - Value: "String",  
    - Key: "ProcessedValue"  
  },  
  - transformId: "123...::String",  
  - requestId: "dcd...",  
  - region: "ap-northeast-1",  
  - params: {  
    - InputString: "input",  
    - Operation: "Upper"  
  },  
  - templateParameterValues: {  
    - InputString: "input"  
  }  
}
```

スタック作成/更新時にテンプレートを加工する (マクロ)

使い方

- 独自マクロの場合
 - テンプレートを処理する Lambda 関数を実装する
 - 実装した Lambda 関数を指定し、AWS::CloudFormation::Macro リソースを作成する
- テンプレートの任意の箇所で Transform を指定し、どのマクロを使用するか設定する

ユースケース

- テンプレートの記述量を減らすため、共通のスニペットをマクロで挿入する (例: AWS::Include)
- 関連するリソースをまとめて抽象的なリソースとし、マクロで CFn の標準リソースとして展開する

注意

- 変更セットを使用し、実行前に変更内容を確認することを推奨する
- Processed テンプレートにもテンプレートサイズの上限が適用される
 - テンプレートサイズの上限に対応するためには、スタックの分割やネストスタックが必要となる



Stack作成/更新時にテンプレートを加工する(マクロ)

AWS が提供するマクロ

- AWS::Serverless (SAM)
 - サーバーレスアプリケーション用の簡素化記述
 - 例: AWS::Serverless::Function リソースを定義すると、関連する IAM ロールなどが展開される
- AWS::SecretsManager
 - シークレットローテーション用Lambdaを指定
- AWS::Include
 - テンプレートに S3 バケットに配置したテンプレートの一部を挿入する
- AWS::CodeDeployBlueGreen
 - ECSのBlue/Greenデプロイ定義用

スタック作成/更新時にテンプレートを加工する(マクロ)

AWS が提供するマクロ

- AWS::LanguageExtensions
 - テンプレート内でループを扱う関数 (Fn::ForEach) や、DeletionPolicy に Fn::If を使用できるようにする拡張機能
 - <https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/transform-aws-languageextensions.html>
- AWS::ServiceCatalog
 - ServiceCatalog のプロビジョニング済みの製品の Output を参照できる

一覧

- <https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/transform-reference.html>

Deep な機能の使い方

- ・任意の処理を追加する（カスタムリソース）
- ・スタック作成/更新時にテンプレートを加工する（マクロ）
- ・スタック作成権限とリソースの保護
- ・CodePipelineからCFnスタックをデプロイする

スタック作成権限とリソースの保護

アクション実行時のデフォルト (サービスロール未指定)



- Template は例として S3 Bucket を 1 つだけ定義
- CFn スタック実行 は以下の API が対象
 - CreateStack
 - UpdateStack
 - DeleteStack
 - CreateChangeSet

CFn スタック実行 IAM ユーザーやロールの権限で実リソースのアクション実行

=> PermissionsA に CFn と S3 のアクションが必要

サービスロール指定



CFn スタック実行時にサービスロールを指定可能 (iam:PassRole が必要)

=> PermissionsB には CFn のアクション、 PermissionsC には S3 のアクションと分離可能

Stack作成権限とリソースの保護

Stack削除保護

- Stackに対する削除操作をStack側で禁止する機能
 - エラー例: cannot be deleted while TerminationProtection is enabled
 - ルートStackへの削除保護はネストされたStackにも有効となる

Stackポリシー

- Stack更新時のリソースの更新を許可/拒否する機能
 - Update:Modify, Update:Replace, Update:Delete, Update:*ごとに設定可能
 - Stack自体の削除やStackからのリソースの削除時には適用されない

リソースの保護

- DeletionPolicy
 - Stack自体の削除やStackからリソースを削除するような更新時に実リソースを残すか設定できる
- https://docs.aws.amazon.com/ja_jp/AWSCloudFormation/latest/UserGuide/aws-attribute-deletionpolicy.html

Deep な機能の使い方

- ・任意の処理を追加する（カスタムリソース）
- ・スタック作成/更新時にテンプレートを加工する（マクロ）
- ・スタック作成権限とリソースの保護
- ・CodePipelineからCFnスタックをデプロイする

CodePipelineからCFnスタックをデプロイする

CFn スタックをデプロイするパイプラインの例



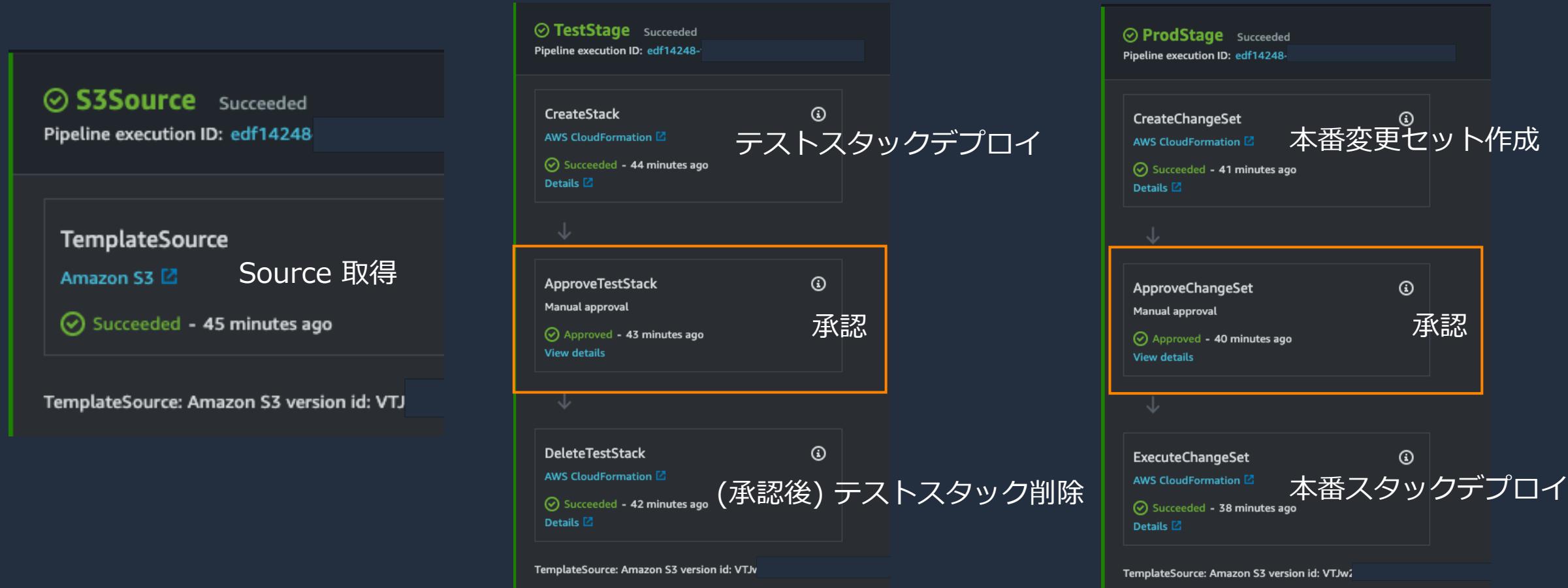
ユースケース

- 複数スタックの操作を自動化しつつ、重要な操作前には承認を挟む場合には CodePipeline を使用できる

https://docs.aws.amazon.com/ja_jp/AWSCloudFormation/latest/UserGuide/continuous-delivery-codepipeline-basic-walkthrough.html

CodePipelineからCFnスタックをデプロイする

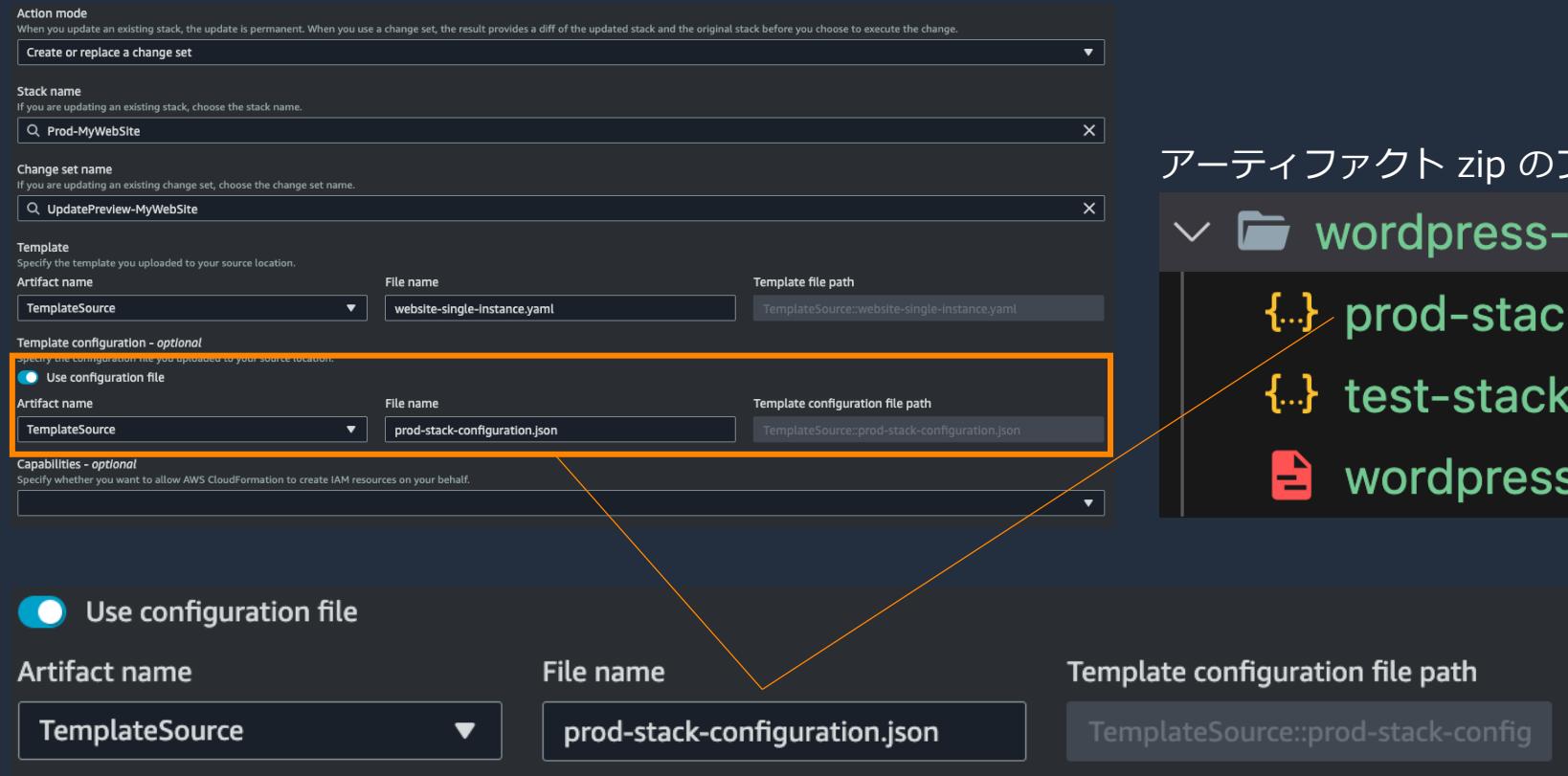
CFn スタックをデプロイするパイプラインの実行例



CodePipelineからCFnスタックをデプロイする

アーティファクトの設定を行うことで環境 (テスト、本番など) ごとに異なるファイルからパラメータを読み込むことが可能

例: テスト環境と同じテンプレートを使用しながら prod-stack-configuration.json からパラメータを読み込む



Deep な機能の使い方

- ・任意の処理を追加する（カスタムリソース）
- ・スタック作成/更新時にテンプレートを加工する（マクロ）
- ・スタック作成権限とリソースの保護
- ・CodePipelineからCFnスタックをデプロイする



Thank you!



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



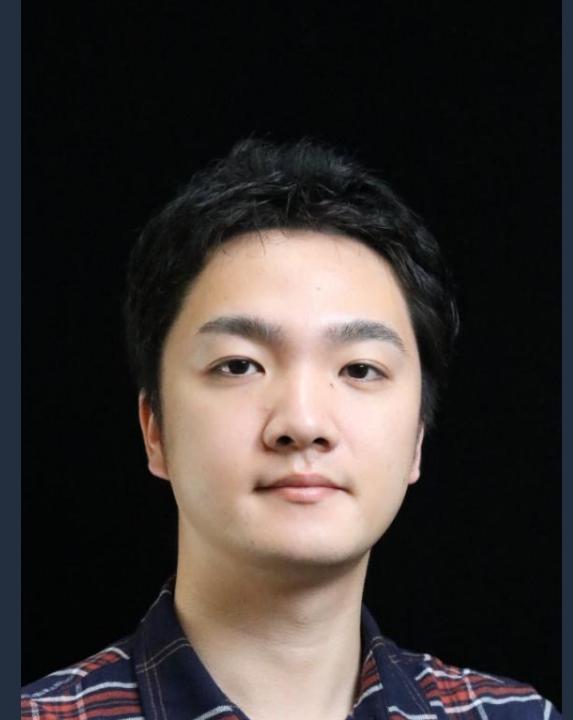
自己紹介

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

The screenshot shows the AWS CloudFormation Resource Registry interface. The top navigation bar has 'AWS::EC2::Instance' selected. Below it, there's a 'Overview' section with details like ARN, Publisher, Release date, and Registry. At the bottom, there are tabs for 'Schema' and 'Configuration', with 'Schema' being the active tab. A note at the bottom of the schema tab states: 'The schema defines the extension. It is part of the schema handler package, provided by the publisher of the extension.'

ARN	Publisher
arn:aws:cloudformation:ap-northeast-1::type/resource/AWS-EC2-Instance	AWS

Release date: 2019-11-16 00:43:51 UTC+0900
Registry: Public

Schema Configuration

The schema defines the extension. It is part of the schema handler package, provided by the publisher of the extension.

Resource types

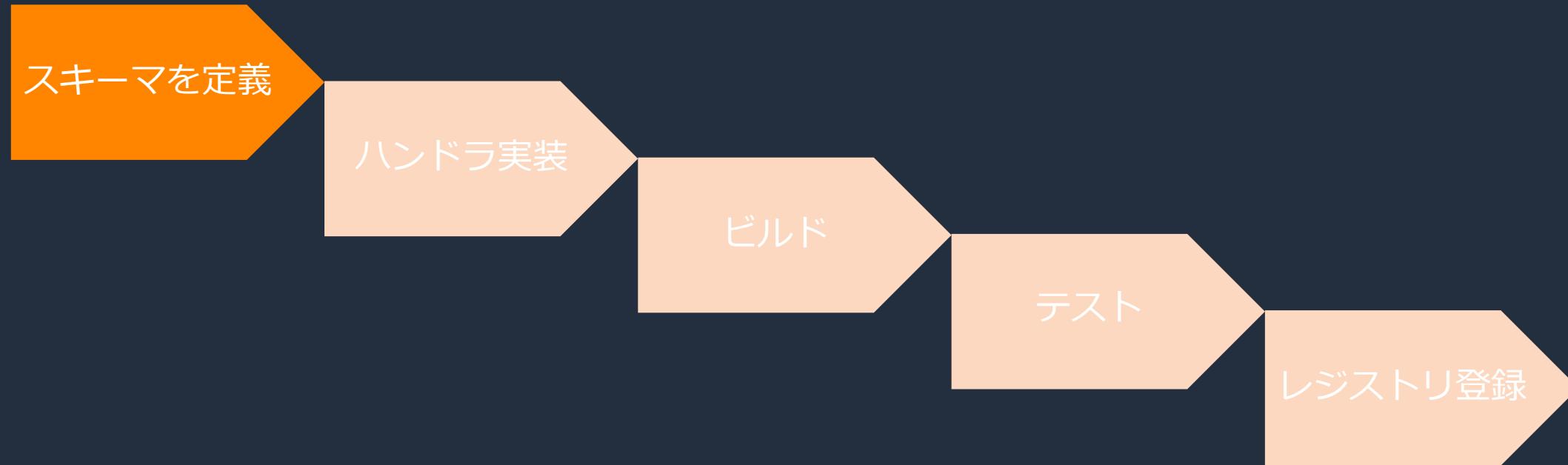
事前準備

- 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 demonstrating CloudFormation's support for AWS Lambda functions.",  
  "sourceUrl": "https://github.com/aws-cloudformation/aws-samples/tree/main/resources/ec2/importkeypair",  
  "definitions": {  
    "Tag": {  
      "description": "A key-value pair to associate with the imported key pair.",  
      "type": "object",  
      "properties": {  
        "Key": {  
          "type": "string",  
          "description": "The key name of the tag.",  
          "minLength": 1,  
          "maxLength": 128  
        },  
        "Value": {  
          "type": "string",  
          "description": "The value for the tag.",  
          "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

```
"properties": {  
    "KeyPairId": {  
        "description": "A Key Pair ID is automatically generated on creation and  
        "type": "string"  
    },  
    "KeyFingerprint": {  
        "description": "The MD5 public key fingerprint of the imported key.",  
        "type": "string"  
    },  
    "KeyName": {  
        "description": "The name of the key is a mandatory element.",  
        "type": "string",  
        "pattern": "^[\\x00-\\x7F]{1,255}$",  
        "minLength": 1,  
        "maxLength": 255  
    },  
    "KeyType": {  
        "description": "The type of the key pair.",  
        "type": "string"  
    },  
    "PublicKeyMaterial": {  
        "description": "The public key material is a mandatory element.",  
        "type": "string",  
        "pattern": "^ssh-[a-zA-Z0-9-]+ AAAA[a-zA-Z0-9\\\\+\\\\/]+=(.*?\\$)",  
        "minLength": 1  
    },  
    "Tags": {  
        "description": "An array of key-value pairs to apply to the resource.",  
        "type": "array",  
        "uniqueItems": false,  
        "insertionOrder": false,  
        "items": {  
            "$ref": "#/definitions/Tag"  
        }  
    }  
}
```

[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)

```
"Tag": {  
    "description": "A key-value p  
    "type": "object",  
    "properties": {  
        "Key": {  
            "type": "string",  
            "description": "The k  
            "minLength": 1,  
            "maxLength": 128  
        },  
        "Value": {  
            "type": "string",  
            "description": "The v  
            "minLength": 0,  
            "maxLength": 256  
        }  
    }  
},
```

Resource types

```
"additionalProperties": false,  
"required": [  
    "KeyName",  
    "PublicKeyMaterial"  
],  
"readOnlyProperties": [  
    "/properties/KeyId",  
    "/properties/KeyFingerprint",  
    "/properties/KeyType"  
],  
"writeOnlyProperties": [  
    "/properties/PublicKeyMaterial"  
],  
"primaryIdentifier": [  
    "/properties/KeyId"  
],  
"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

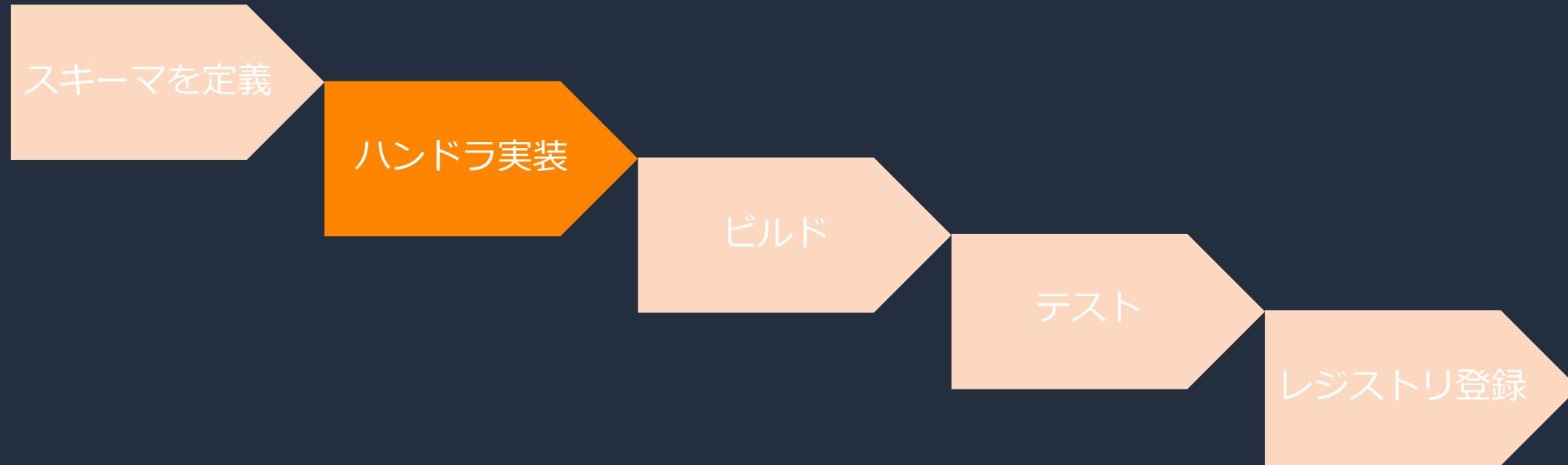
```
"handlers": {  
    "create": {  
        "permissions": [  
            "ec2:ImportKeyPair",  
            "ec2:CreateTags"  
        ]  
    },  
    "read": {  
        "permissions": [  
            "ec2:DescribeKeyPairs"  
        ]  
    },  
    "update": {  
        "permissions": [  
  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

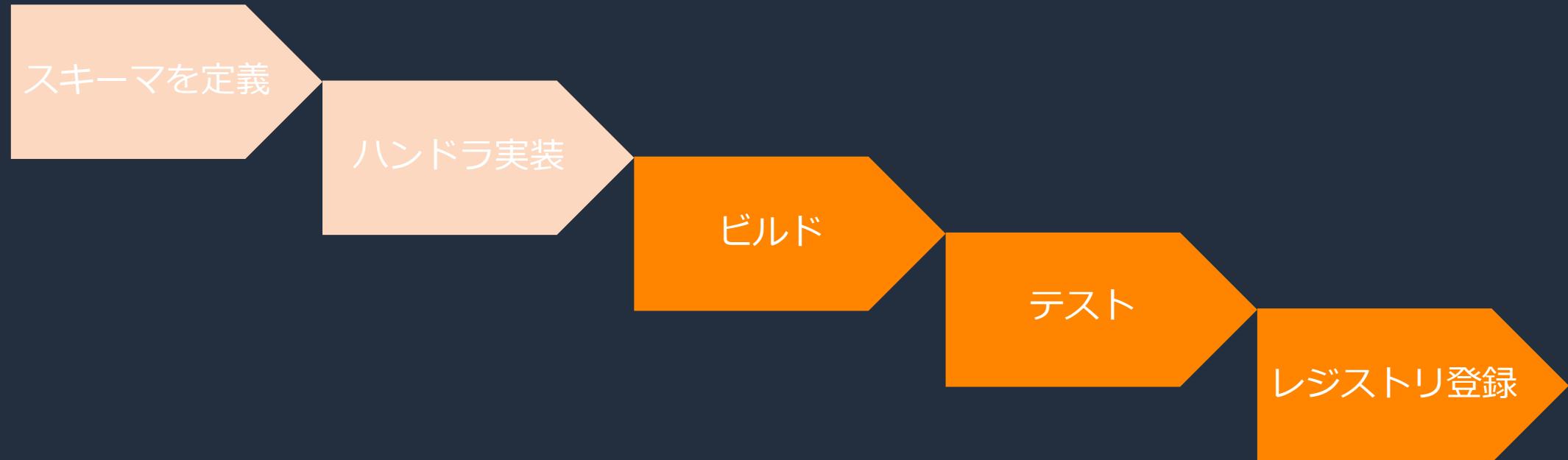
リソースの安定化 (Stabilization)

- 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

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



Resources:

```
  Sample:  
    Type: MyOrg::Sample::Sample::MODULE
```

CloudFormation テンプレートで指定する

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
```

Resources:

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

Resources:

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

Resources:

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

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

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

Private なリソースタイプ

- 明示的に有効とする必要がある

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 クラスターのログイン設定を検証
 - ・ IAM ポリシーの Condition に MFA を強制する

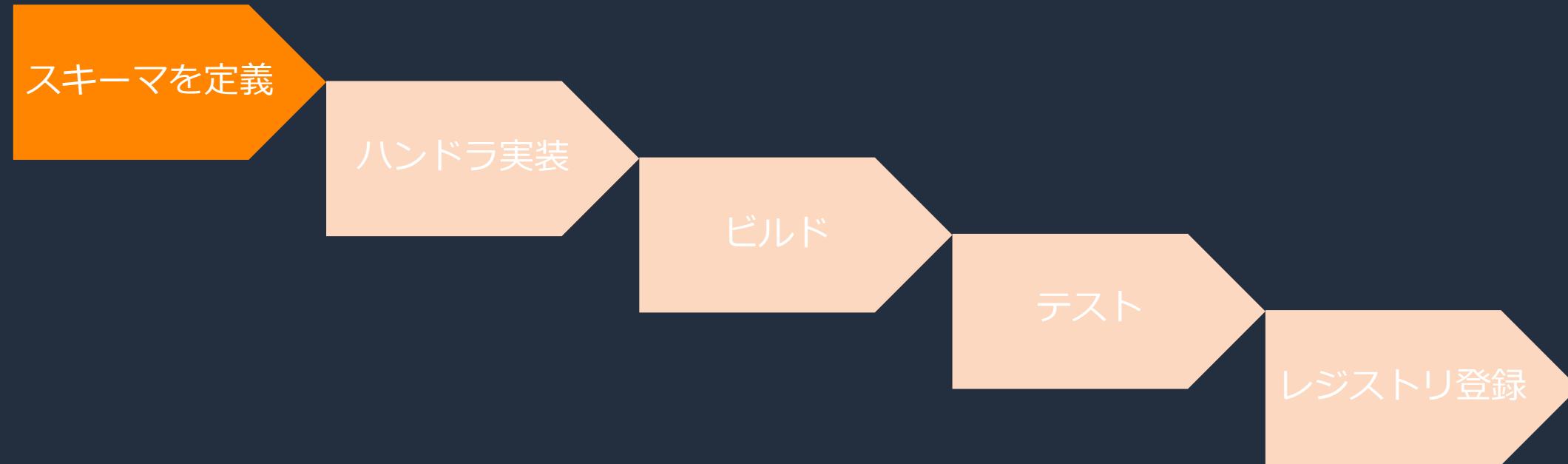
Hooks

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 compliant buckets required for the hook to fire.",  
            "type":"string"  
        },  
        "minQueues":{  
            "description":"Minimum number of compliant queues required for the hook to fire.",  
            "type":"string"  
        },  
        "encryptionAlgorithm":{  
            "description":"Encryption algorithm for the hook's payload.",  
            "default":"AES256",  
            "type":"string"  
        }  
    }  
},
```

[必須] typeConfiguration:
Hooks の設定。

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

Hooks

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

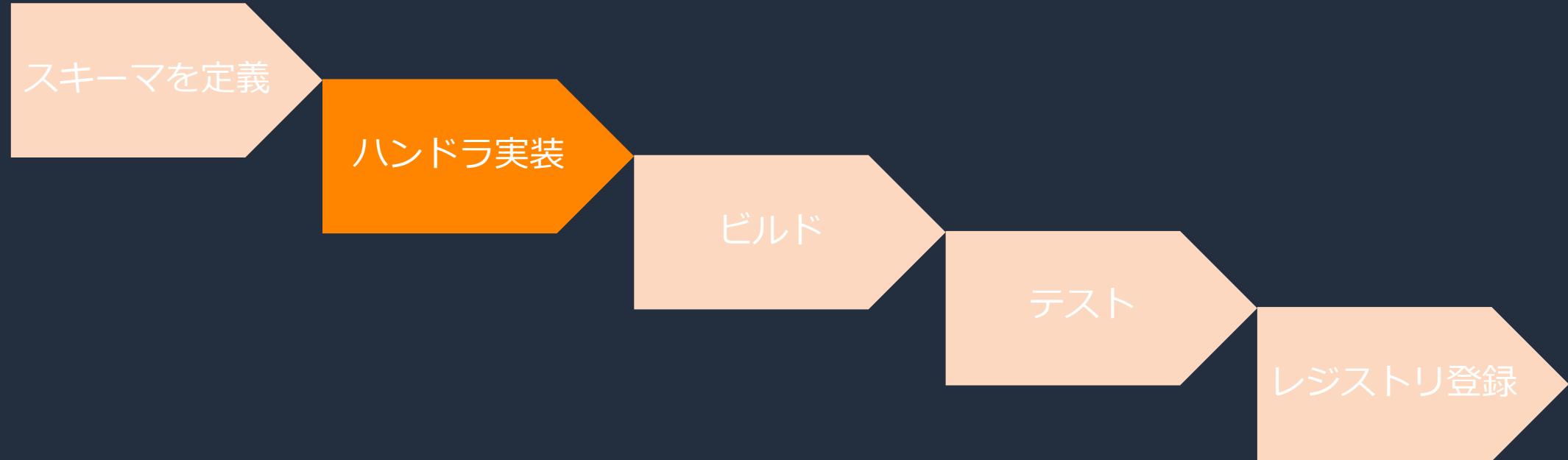
 - UpdateCleanup 例
 - テンプレートからリソースを削除しStack更新
 - 更新タイプが Replacement のリソース削除

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

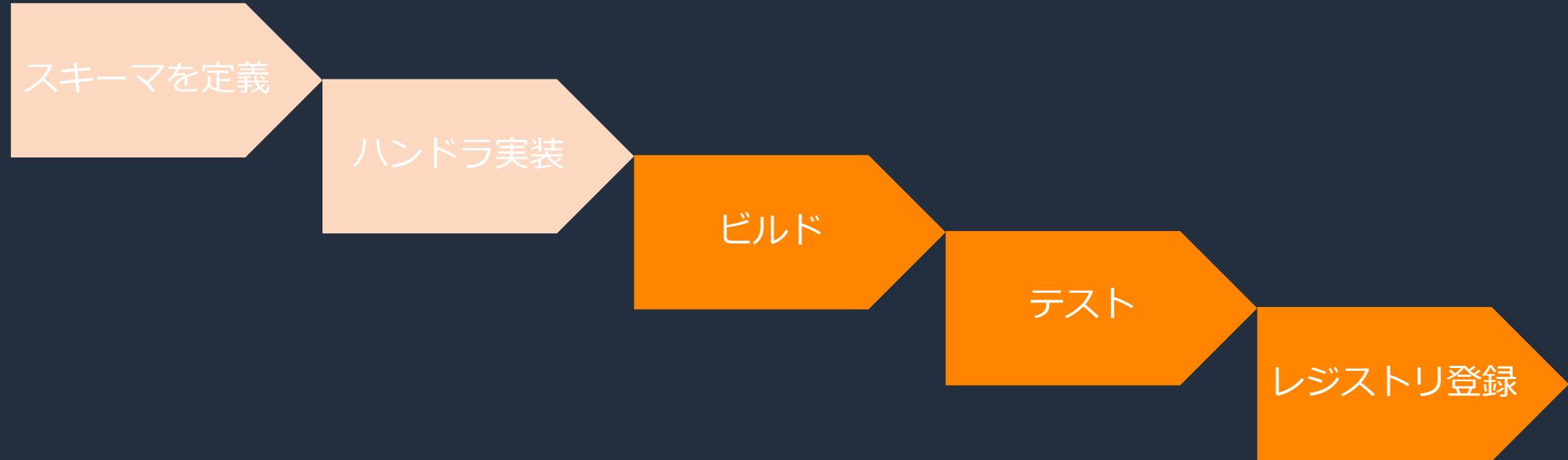
- cfn init コマンドである程度用意される
- request に含まれるリソースタイプ名や resourceProperties を元に検証できる
 - preUpdate では更新前の resourceProperties も参照可能なため更新前後の検証ができる
- Resource Type と同様に実行結果などを含む ProgressEvent を return することで処理が進む

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

Hook name	▲	Target stacks	Failure mode	Default version
<input type="radio"/> AWSSamples::EksClusterLogging::Hook	<input checked="" type="checkbox"/>	All stacks	Fail	-

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/clouformation-cli/latest/userguide/what-is-clouformation-cli.html>
 - Resource Types (Java)
 - Modules
 - Hooks (Java, Python)



Thank you!



AWS CloudFormation

よくあるユースケースと質問

木村 友則

Solutions Architect
2023/10

自己紹介

名前：木村 友則 (きむら ともなり)

所属：クラウドソリューション統括本部
ソリューションアーキテクト



好きなAWSサービス：

AWS CloudFormation, AWS CDK, AWS CLI

本セミナーの対象者

想定聴講者

- CloudFormation のユースケースやよく聞かれる疑問に興味のある方

前提知識

- AWS の基本的な概要や操作を理解していること
- CloudFormation の用語 (スタック、テンプレート、変更セットなど) を理解していること

※ 次の Black Belt Online Seminar で解説しています

AWS CloudFormation #1 基礎編

資料 https://pages.awscloud.com/rs/112-TZM-766/images/AWS-Black-Belt_2023_CloudFormation-1_0731_v1.pdf

動画 <https://youtu.be/4dyiPsYXG8I>

アジェンダ

- よくあるユースケース別の使い方
- よくある質問

※ 本資料では CloudFormation = CFn と略記することがあります

よくあるユースケース別の使いかた

よくあるユースケース

1. 手動で作った既存リソースをCFnの管理下に入れる
2. 既存のスタックを分割する(テンプレートのリファクタリング)
3. 各AWSアカウントに基本設定を展開する
4. OS上の設定も含めてサーバー構成を管理する
5. 場面に応じてパラメータの指定方法を使い分ける
6. 最新のAMI IDを取得して指定する
7. CloudFormationで作成したリソースのバックアップを取得する

よくあるユースケース

1. 手動で作った既存リソースをCFnの管理下に入れる
2. 既存のスタックを分割する(テンプレートのリファクタリング)
3. 各AWSアカウントに基本設定を展開する
4. OS上の設定も含めてサーバー構成を管理する
5. 場面に応じてパラメータの指定方法を使い分ける
6. 最新のAMI IDを取得して指定する
7. CloudFormationで作成したリソースのバックアップを取得する

手動で作った既存リソースを CFn の管理下に入れる

実現したいこと

- 手動で作った既存リソースを新規または既存のスタック管理下に入れたい

解決策

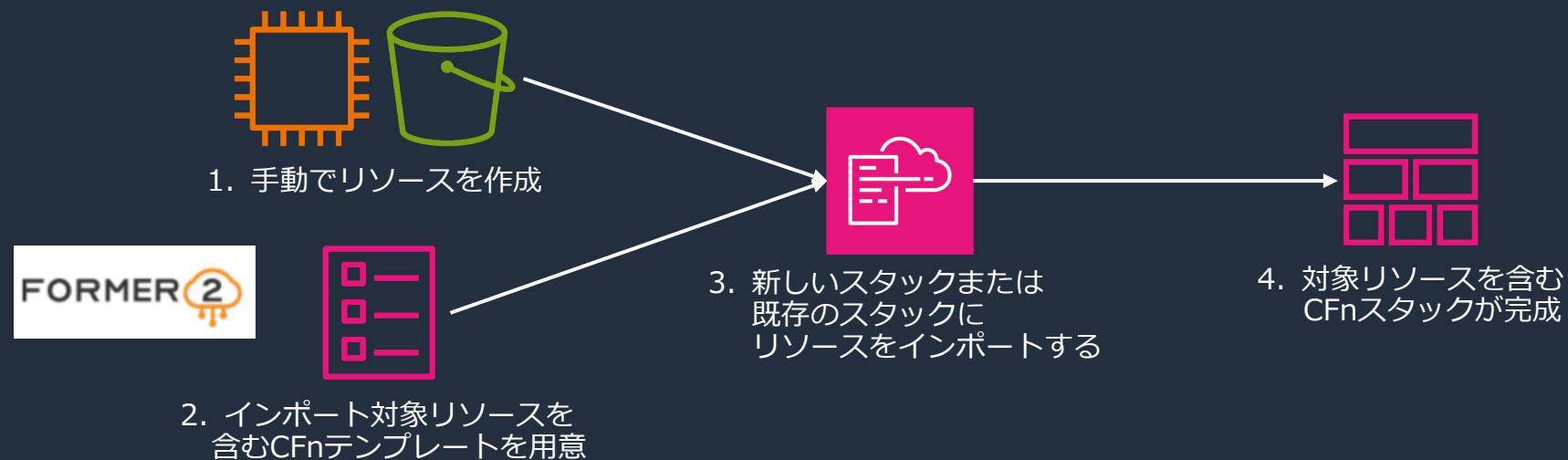
- リソースのインポート機能を利用する



リソースのインポート機能について

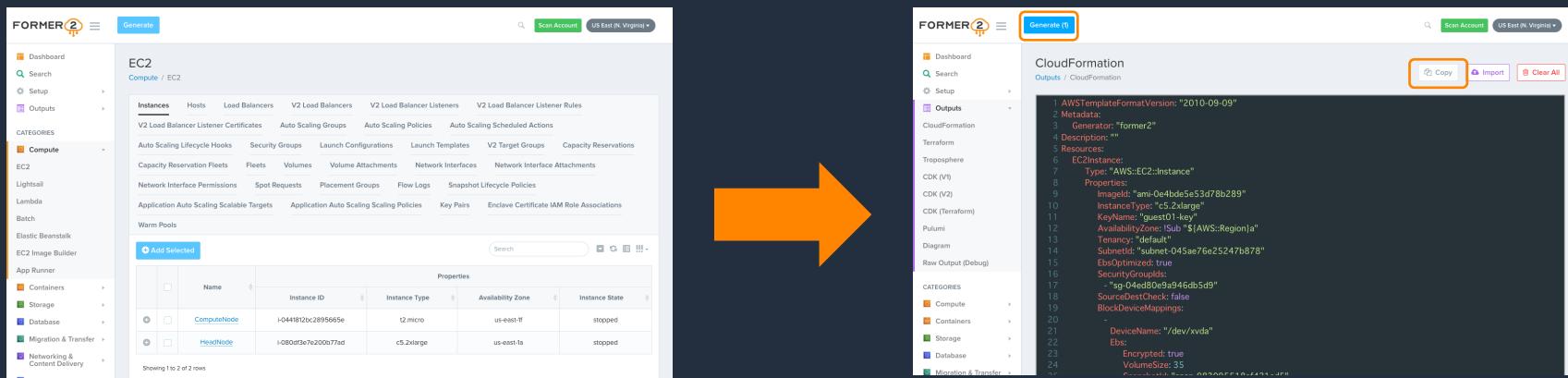
手動で作成した AWS リソースを
あとから CloudFormation スタックにインポートして管理可能

- リソースをスタックの管理下から切り離したり、別のスタック管理下に移動することも可能
- インポート対象のリソースの実際の設定と一致するようテンプレートを用意する
 - AWS 公式のツールではないが、Former2 でテンプレート作成の省力化が可能
 - <https://github.com/iann0036/former2>



Former2 を利用したインポート用テンプレートの生成

1. リソース読み取り用の IAM ユーザーを作成する
 - ReadOnlyAccess ポリシーを付与し、アクセスキーを発行する
2. Former2 を開く
 - <https://former2.com/> または ローカル環境で起動した Former2
 - ブラウザに適した拡張機能をインストール
3. 先の手順で用意したアクセスキーを適用する
4. 対象リソースを選択して、CloudFormation テンプレートを生成、コピーする



生成したテンプレートを既存スタックにインポートする

- 生成したテンプレートのResources以下にある各リソースに DeletionPolicy 属性を追加して、既存のインポート先テンプレートの Resources 以下に追記する

```
EC2Instance:  
  Type: "AWS::EC2::Instance"  
  DeletionPolicy: Retain  
  Properties:  
    ImageId: "ami-0e4bde5e53d78b289"
```

※ Retain、Delete 等、用途に応じて指定する

- CloudFormation のマネジメントコンソールでインポートを実行する



インポート先のスタックを選択し、
「Stackへのリソースのインポート」を実行する
※ 新規スタックの場合は「Stackの作成」から



対象リソースの識別子を指定する

- ドリフト検出を実行し、テンプレートとリソースが一致していることを確認、一致しない場合は修正を実施する

リソースをインポートする際の考慮事項

全てのリソースのインポートに対応しているわけではない

- 対応リソース一覧 (最新の情報は英語版をご覧ください)
- <https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/resource-import-supported-resources.html>
- 対応リソースのリクエストは Public Coverage Roadmap まで
- <https://github.com/aws-cloudformation/cloudformation-coverage-roadmap>

リソースのインポートには、識別子の指定が必須

- テンプレートの生成だけで作業は完了せず、対象リソースの量によって相応の手間もかかる

EC2 や RDS 等の依存リソースが多いものは、インポートが困難になりがち

- S3 や Lambda 等のサーバーレスなリソースは、比較的容易にインポートできる

※ インポートの実施に困難が大きい場合は、再作成もご検討ください。



参考：ローカル環境での Former2 起動方法

実行例

```
$ git clone https://github.com/iann0036/former2
$ cd former2
$ docker-compose up -d
```

※ 詳しくはドキュメントも参照ください

<https://github.com/iann0036/former2/blob/master/HOSTING.md>

よくあるユースケース

1. 手動で作った既存リソースをCFnの管理下に入れる
2. 既存のスタックを分割する(テンプレートのリファクタリング)
3. 各AWSアカウントに基本設定を展開する
4. OS上の設定も含めてサーバー構成を管理する
5. 場面に応じてパラメータの指定方法を使い分ける
6. 最新のAMI IDを取得して指定する
7. CloudFormationで作成したリソースのバックアップを取得する

肥大化したスタックを分割する（リファクタリング）

実現したいこと

- 運用過程のリソース追加によってスタックが肥大化、その結果スタック変更に時間がかかる、スタック変更の影響範囲が大きい、クオータの上限に達しそう等の顕在化した課題を解消したい

解決策

- リソースを残したままスタックを削除し、別のテンプレートでリソースのインポートを行うことでスタックをリファクタリングする（インポートに対応しているリソースに限られる）



実行手順

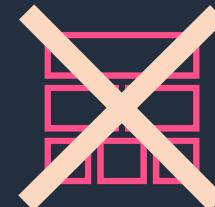
1. 既存スタックのテンプレートを編集し更新する

全てのリソースに “**DeletionPolicy: Retain**” を追記、テンプレートをスタックに適用する。Retain と指定することで、スタックを削除してもリソースは削除されない。(リソースは複数スタックに属せない)



```
EC2Instance:  
  Type: "AWS::EC2::Instance"  
  DeletionPolicy: Retain  
  Properties:  
    ImageId: "ami-0e4bde5e53d78b289"
```

2. 既存スタックを削除する

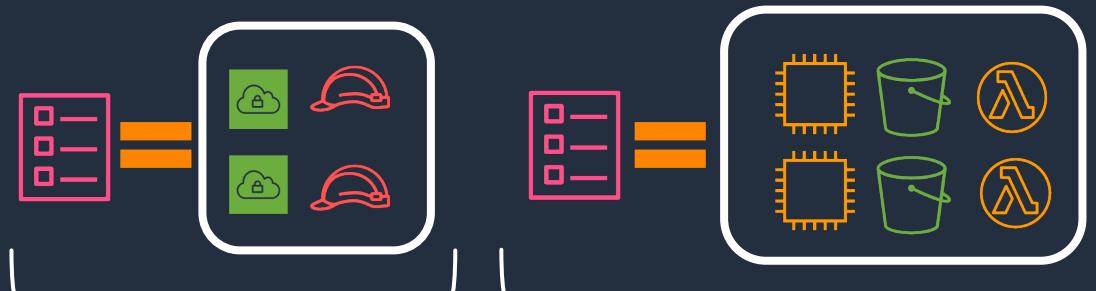


スタックは無くなる



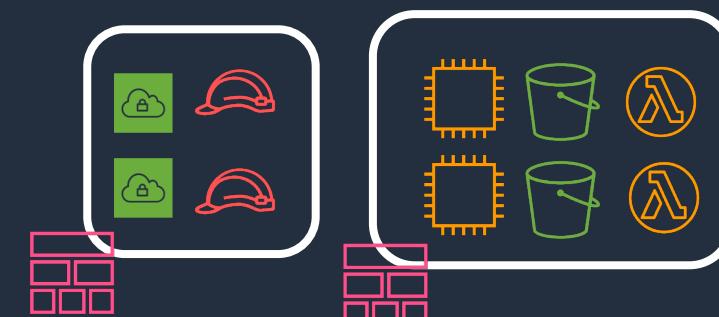
リソースは残る

3. 既存スタックのテンプレートを分割する



※ 必要に応じてクロススタックリファレンスやネストスタックを利用

4. 分割したテンプレートでリソースをインポートする



よくあるユースケース

1. 手動で作った既存リソースをCFnの管理下に入れる
2. 既存のスタックを分割する(テンプレートのリファクタリング)
3. 各AWSアカウントに基本設定を展開する
4. OS上の設定も含めてサーバー構成を管理する
5. 場面に応じてパラメータの指定方法を使い分ける
6. 最新のAMI IDを取得して指定する
7. CloudFormationで作成したリソースのバックアップを取得する

各AWSアカウント・リージョンに基本設定を展開する

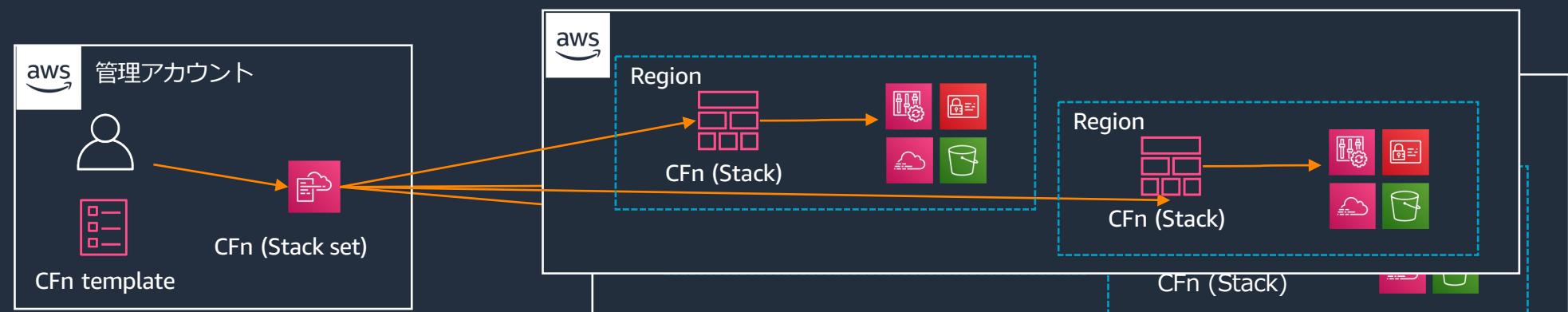
実現したいこと

- マルチアカウント・マルチリージョン環境で、基本設定を展開したい
- 基本設定として、監査ログ、構成のチェック、基本の IAM Role を設定したい

解決策

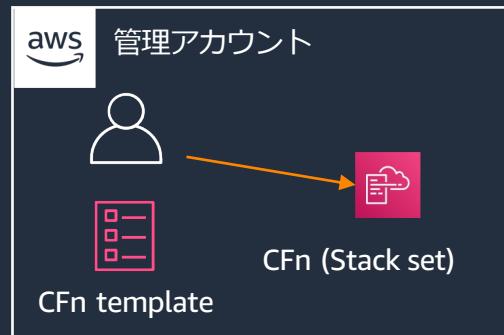
- StackSets を使って対象アカウント・リージョンに基本設定のスタックを作成する

※ AWS Organizations が利用できる状況ならば、AWS ControlTower の利用もお勧めです。
ControlTower では、例示したような基本設定の展開を StackSets を活用して実現しています。



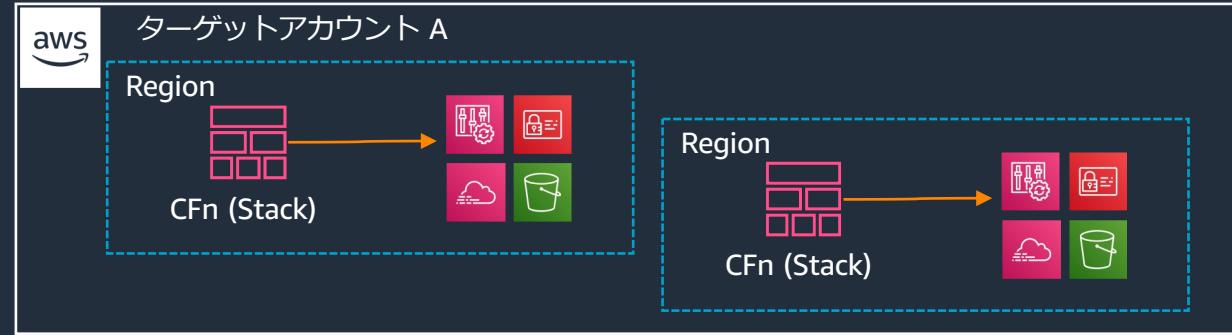
StackSets について

一回の操作で複数のアカウントやリージョンへ
スタックを作成、更新、削除できるCloudFormationの機能



スタック操作のためのアクセス許可モデル

- セルフマネージド
Organizations 不要、その代わりに
管理アカウントとターゲットアカウントの双方に
必要なIAMロールを自身が作成する
- サービスマネージド
Organizations を通じて必要なIAMロールを自動で作成する

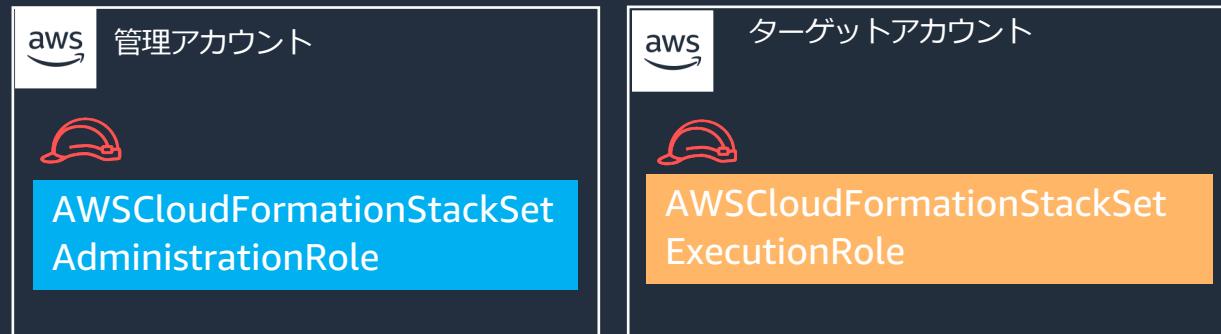


https://docs.aws.amazon.com/ja_jp/AWSCloudFormation/latest/UserGuide/what-is-cfnstacksets.html

© 2023, Amazon Web Services, Inc. or its affiliates.

実行手順 – アクセス許可モデルの選択と設定 (初回のみ)

セルフマネージド



管理アカウントとターゲットアカウントの双方に指定された名称・内容でIAMロールの作成が必要です。

IAMロール作成のための CFn テンプレートが配布されています (ドキュメント参照)。

https://docs.aws.amazon.com/ja_jp/AWSCloudFormation/latest/UserGuide/stacksets-prereqs-self-managed.html

サービスマネージド

A screenshot of the AWS Organizations CloudFormation StackSets page. The top navigation bar shows 'AWS Organizations > サービス > CloudFormation StackSets'. The main section is titled 'CloudFormation StackSets' and includes a 'コンソールに移動する' button. Below this, there's a '信頼されたアクセス' section with a status indicator 'ステータス' and a note: '信頼されたアクセスが有効'. A detailed note explains that enabling trusted access creates IAM roles in both accounts: 'CloudFormation StackSets が組織内の信頼されたサービスとして指定されます。信頼されたサービスは、組織の構造をクエリし、組織のアカウントにサービスにリンクされたロールを作成できます。サービスにリンクされたロールにより、信頼されたサービスは、信頼されたサービスのドキュメントに記載されているタスクを実行できます。信頼されたサービスは組織への変更について通知を受け、これらの通知に応じて追加のタスクを実行できます。' There is also a link '詳細はこちる'.

Organizations で CloudFormation StackSets の 信頼されたアクセスを有効化することで、以下のIAMロールが自動的に作成されます。

管理アカウント : AWSServiceRoleFor
CloudFormationStackSetsOrgAdmin (実際には一行)
ターゲットアカウント : AWSServiceRoleFor
CloudFormationStackSetsOrgMember (実際には一行)
及び stacksets-exec-*

https://docs.aws.amazon.com/ja_jp/AWSCloudFormation/latest/UserGuide/stacksets-orgs-activate-trusted-access.html

実行手順 1/2

1. テンプレートの選択

アクセス許可モデルを選んだ上で、
自分で用意したものかサンプルテンプレートを
指定します。

前提条件 - テンプレートの準備

テンプレートの準備
各スタックはテンプレートに基づきます。テンプレートとは、スタックに含む AWS リソースに関する設定情報を含む JSON または YAML ファイルです。

テンプレートの準備完了 サンプルテンプレートを使用

テンプレートの指定
テンプレートは、スタックのリソースおよびプロパティを表す JSON または YAML ファイルです。

テンプレートソース
テンプレートを選択すると、保存先となる Amazon S3 URL が生成されます。

Amazon S3 URL テンプレートファイルのアップロード

Amazon S3 URL
`https://`
Amazon S3 テンプレートの URL

S3 URL: URL を指定すると生成されます。 デザイナーで表示

2. StackSets の詳細を指定

StackSetsの名称や説明、
テンプレートのパラメータを指定します。

StackSet の詳細を指定

StackSet 名

StackSet 名
`StackSet 名`
小文字、大文字、数字、ダッシュを含める必要があります。文字で始まる必要があります。

StackSet の説明

説明を使用して、スタックセットの目的やその他の重要な情報を識別できます。

StackSet の説明
`StackSet の説明`

実行手順 2/2

3. StackSets オプションの設定

タグや実行設定を指定します。

StackSet オプションの設定

タグ
Stack のリソースに適用するタグ (キーと値のペア) を指定できます。Stack ごとに一意のタグを 50 個まで追加できます。
Stack に関連付けられたタグがありません。
 新しいタグの追加
さらに 50 のタグを追加できます

実行設定
マネージド型の実行
StackSets が競合しないオペレーションを並行して実行し、競合するオペレーションはキューに入れるかどうかを指定します。
 非アクティブ
StackSets は、一度に 1 つのオペレーションを実行します。
 アクティブ
StackSets は、競合しないオペレーションを並行して実行し、競合するオペレーションをキューに入れます。競合するオペレーションが終了すると、StackSets はリクエスト順にキューに入れられたオペレーションを開始します。

4. デプロイオプションの設定

デプロイ先 (組織・OU・アカウントID) やリージョン、自動デプロイオプション(サービススマネージドの場合)、同時実行するアカウントやリージョン等を指定します。

デプロイオプションの設定

StackSet に Stack を追加
 新しい Stack のデプロイ
 Stack を StackSet にインポート

デプロイターゲット
StackSets は、ターゲット組織または組織単位 (OU) のすべてのアカウントに Stack インスタンスをデプロイします。親 OU をターゲットとして追加すると、StackSets はターゲットとして子 OU も追加します。[詳細はこちら](#)
 組織へのデプロイ
 組織単位 (OU) へのデプロイ

自動デプロイオプション
自動デプロイ
自動デプロイが有効になっている場合、アカウントが OU に追加されると、StackSets は自動的に追加の Stack インスタンスをこのアカウントにデプロイします。アカウントが OU から削除されると、StackSets はこのアカウントの Stack インスタンスを自動的に削除します。
 有効
 無効
アカウント削除の動作
ターゲット OU からアカウントを削除する場合、アカウント内の Stack インスタンスを削除または保持する必要があります?
 Stack を削除
 Stack を保持

リージョンの指定
Stack をデプロイするリージョンを選択します。Stack は、指定した順序でこれらのリージョンにデプロイされます。StackSet の操作中に、管理者アカウントとターゲットアカウントは、アカウント自身で、むしろに連携する StackSet および Stack セットインスタンスに関するメタデータを交換するように注意!

StackSets を利用する際の考慮事項

デプロイ対象は複数のアカウントの複数のリージョンを指定できる

- Organizations があるとデプロイ対象の指定とロールの管理が楽だが必須ではない
- OUまたは個別アカウントのいずれかを指定することが多い

テンプレートとパラメータ

- テンプレートは全ての環境に同じものが利用される
- パラメータはデプロイ対象に対して一括で指定するが、特定のアカウント×リージョンに対して個別にデプロイする際のパラメータを上書きできる

StackSets 間の連携

- StackSets 同士は依存関係を指定できないため、1アカウントあたり StackSets 1つにしておくと安全

参考：基本設定の例 1/2

AWS ControlTower の設定

- ControlTower ドキュメントに記載の CFn テンプレート
https://docs.aws.amazon.com/ja_jp/controltower/latest/userguide/controls-reference.html
- ControlTower が生成するスタック
https://docs.aws.amazon.com/ja_jp/controltower/latest/userguide/account-factory-considerations.html
ControlTower 環境を作ると CFn 経由でテンプレートを参照できます

ロギングの有効化

- AWS CloudTrail および AWS Configの有効化
- AWS ConfigRulesでCloudTrailが有効であることを確認
- これらの設定は、サンプルテンプレートが StackSets から利用可能
https://docs.aws.amazon.com/ja_jp/AWSCloudFormation/latest/UserGuide/stacksets-samplitemplates.html

参考：基本設定の例 2/2

各アカウントの管理用 IAM Role

- 例: baseline-admin、baseline-abc などの名称で IAM Role を作成する
- 用途に合わせてAdministratorAccess や、ReadOnlyAccess ポリシーを持つユーザを作成
- ただし… Organizations SCP (または追加のポリシー) で以下を制限
 - Role名に baseline-* がついたロールの変更禁止 (権限昇格の禁止)
 - https://docs.aws.amazon.com/ja_jp/controltower/latest/userguide/mandatory-controls.html#iam-disallow-changes

CloudTrailおよびConfigの設定変更を禁止

- https://docs.aws.amazon.com/ja_jp/controltower/latest/userguide/mandatory-controls.html#cloudtrail-configuration-changes
- https://docs.aws.amazon.com/ja_jp/controltower/latest/userguide/mandatory-controls.html#config-disallow-changes

よくあるユースケース

1. 手動で作った既存リソースをCFnの管理下に入れる
2. 既存のスタックを分割する(テンプレートのリファクタリング)
3. 各AWSアカウントに基本設定を展開する
4. OS上の設定も含めてサーバー構成を管理する
5. 場面に応じてパラメータの指定方法を使い分ける
6. 最新のAMI IDを取得して指定する
7. CloudFormationで作成したリソースのバックアップを取得する

OS上の設定も含めてサーバー構成を管理する

実現したいこと

- CloudFormation でサーバー構成を管理するだけで無く、起動したサーバー(EC2)のOS上の各種設定についても一気通貫で管理したい

解決策

- CloudFormation と Systems Manager (以降SSM) の State Manager を併用する
- State Manager では、予め用意されている SSM ドキュメントを利用して、OS上の設定管理ツール (Ansible や PowerShell DSC 等) を定期的に実行する

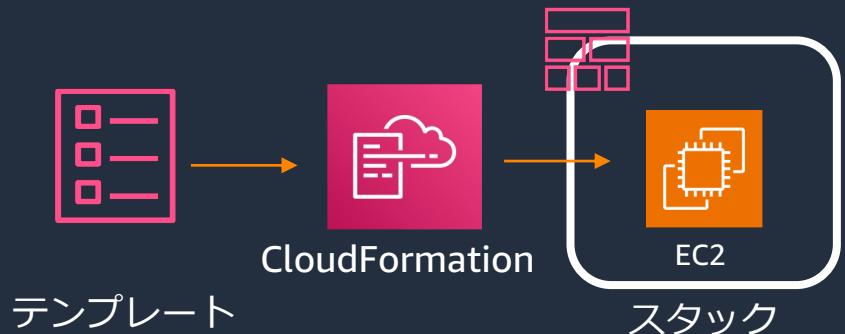
※ State Manager について

資料 https://pages.awscloud.com/rs/112-TZM-766/images/AWS-Black-Belt_2023_AWS-SystemsManager-StateManager_0630_v1.pdf

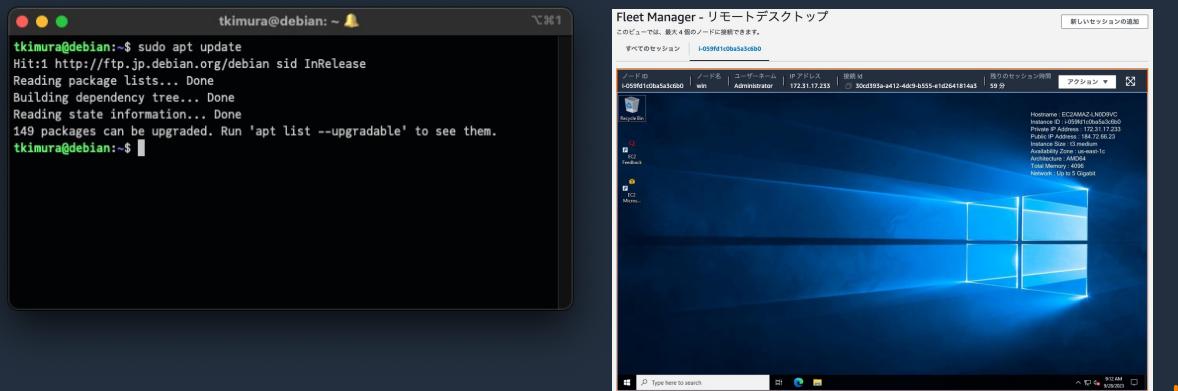
動画 <https://youtu.be/vSAbhWZFtKU>

CloudFormation を利用したサーバーの構築の流れ

1. EC2 の起動

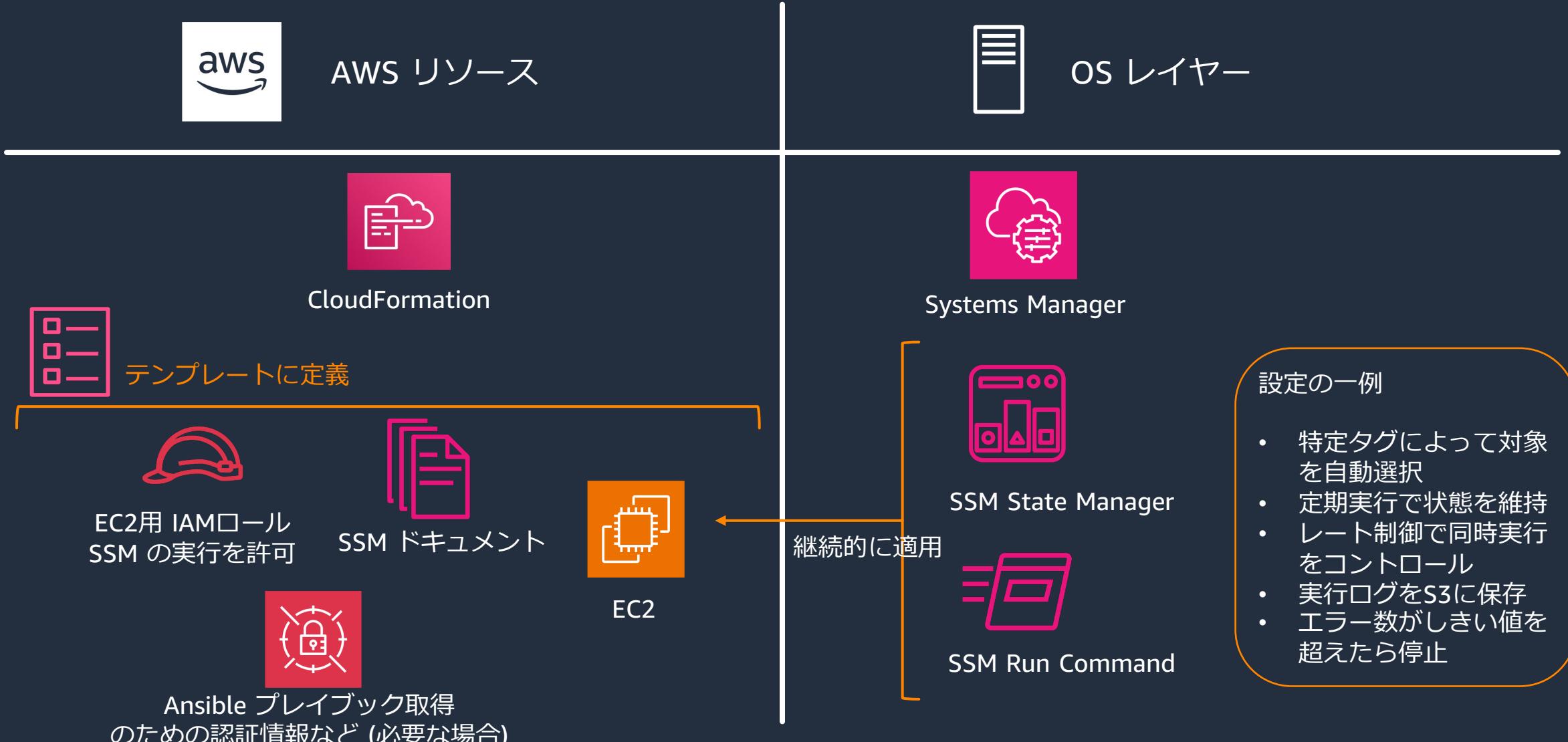


2. OS上の設定



この領域についても一気通貫で管理出来れば、個々のサーバーに対する個別オペレーションなく構築が完了出来る

実現方法



テンプレートの例

EC2を含むテンプレートに次のような内容を追加する

```
AnsibleAssociation:  
  Type: AWS::SSM::Association  
  Properties:  
    # ここではAWS-ApplyAnsiblePlaybooksを利用しています  
    Name: AWS-ApplyAnsiblePlaybooks  
    # CloudFormationは関連付けが行われるのを待ちます  
    WaitForSuccessTimeoutSeconds: 120  
  Targets:  
    - Key: InstanceIds  
      Values: [ !Ref EC2Instance ]  
  OutputLocation:  
    S3Location:  
      OutputS3BucketName: !Ref SSMAssocLogs  
      OutputS3KeyPrefix: 'logs/'
```

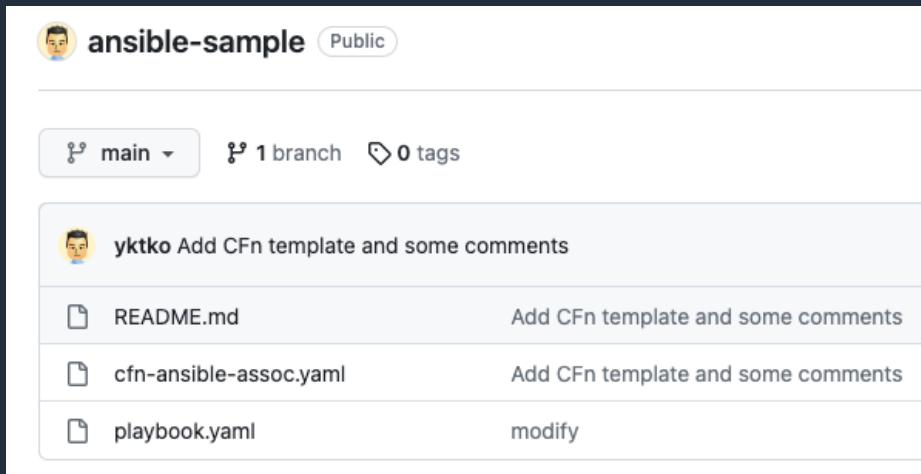
右へ続く

```
Parameters:  
  # GitHubからAnsibleプレイブックを取得します  
  SourceType:  
    - 'GitHub'  
  SourceInfo:  
    - '{ "owner": "<Insert your GitHub Owner Name>" ,  
      "repository": "<Insert your GitHub Repo>" ,  
      "path": "",  
      "getOptions": "branch:master" }'  
  # Ansibleと依存関係のインストール  
  InstallDependencies:  
    - 'True'  
  # 実行するプレイブック  
  PlaybookFile:  
    - 'playbook.yml'  
  ExtraVariables:  
    - 'SSM=True'  
  Check:  
    - 'False'
```



参考： サンプルテンプレート

<https://github.com/yktko/ansible-sample>



`cfn-ansible-assoc.yaml` を利用してスタックを作成することで、
EC2 の起動を行い、SSM State Manager 経由の
Ansible プレイブックの実行の流れを確認出来ます。

起動した EC2 には、SSM Session Manager で接続出来ます。

参考： CFn + cfn-init によるサーバー構成管理

EC2 の UserData にスクリプトを記述する方式

- 初期起動時に指定したスクリプトを実行する
- テンプレートにサーバーの起動処理をまとめて記載可能
- cfn-init はヘルパースクリプトの一つ
- 慣れ親しんだシェルスクリプトが使える
- 従来から利用されており情報が豊富

課題

- インスタンス初期起動時しか動作できず
起動後のメンテナンスは他の仕組みが必要
- CFn テンプレート内にコードを書くことで
複雑な処理や多数のサーバーの管理ではメンテナンス性に課題
- 処理中のエラーメッセージが
インスタンスのローカルディスク上のログに保持される

CFn テンプレートでの指定例

```
UserData:  
Fn::Base64: !Sub |  
#!/bin/bash -xe  
yum -y update  
yum -y install aws-cfn-bootstrap  
/opt/aws/bin/cfn-init -v ¥  
--stack ${AWS::StackName} ¥  
--resource ServerInstance ¥  
--region ${AWS::Region}
```

よくあるユースケース

1. 手動で作った既存リソースをCFnの管理下に入れる
2. 既存のスタックを分割する(テンプレートのリファクタリング)
3. 各AWSアカウントに基本設定を展開する
4. OS上の設定も含めてサーバー構成を管理する
5. 場面に応じてパラメータの指定方法を使い分ける
6. 最新のAMI IDを取得して指定する
7. CloudFormationで作成したリソースのバックアップを取得する

場面に応じてパラメータの指定方法を使い分ける

実現したいこと

- ・ パラメータによって作成するAWSリソースを変更したい
- ・ 例) 同じテンプレートを使って、開発 や 本番 環境を作り分けたい

解決策

- ・ 複数あるパラメータの指定方法を把握し、用途に応じて使い分ける

SSM Parameter Store/ Secrets Manager



指定方法 1/2

a. テンプレート直書き

EC2のAMIを指定する場合

```
Resources:  
  MyEC2Instance:  
    Type: AWS::EC2::Instance  
    Properties:  
      ImageId: "ami-79fd7eee"  
      KeyName: "testkey"
```

b. スタック設定として渡す

デプロイ先の環境名を設定する場合

```
Parameters:  
  EnvironmentTag:  
    Default: development  
    AllowedValues:  
      - development  
      - staging  
      - production  
    Type: String
```

テンプレート作成時に決定したい値の例

- 自社専用に用意したEC2のカスタムAMI
- Lambdaのランタイムバージョン
- 等々

スタック作成時に決定したい値の例

- 環境を示すタグ (各リソースのタグに指定)
- マルチAZの要否 (True/False)
- 等々

直書きすることでレビュープロセスで指摘出来る



指定方法 2/2

c. スタック作成時に外部を参照

RDSの認証情報を指定する場合

```
Resources:  
  MyDB:  
    Type: AWS::RDS::DBInstance  
    Properties:  
      Engine: MySQL  
      EngineVersion: "8.0.34"  
      MasterUsername: dbadmin  
      MasterUserPassword: '{resolve:secretsmanager:MyDBAuth:SecretString:password}'
```

テンプレートに書きたくない値、既存のParameter Store、Secrets Managerから取得したい値

- 認証情報のような機密情報
- リージョンをまたぐスタック同士での値の参照
- その他、既にParameter Store、Secrets Managerで管理されている値を参照したい場合

※ 例では別途 SSM Secrets Manager で設定済みの認証情報を使う形にしていますが、
RDS では ManageMasterUserPassword プロパティを使うことで Secrets Manager へのシークレットの作成も自動的に行われます。

https://docs.aws.amazon.com/ja_jp/AWSCloudFormation/latest/UserGuide/dynamic-references.html

よくあるユースケース

1. 手動で作った既存リソースをCFnの管理下に入れる
2. 既存のスタックを分割する(テンプレートのリファクタリング)
3. 各AWSアカウントに基本設定を展開する
4. OS上の設定も含めてサーバー構成を管理する
5. 場面に応じてパラメータの指定方法を使い分ける
6. 最新のAMI IDを取得して指定する
7. CloudFormationで作成したリソースのバックアップを取得する

最新のAMI IDを取得して指定する

実現したいこと

- 常に最新のAMI IDを利用してEC2を起動したい
- 例) 一時的に利用する環境を CloudFormation で作成する

解決策

- SSMのパブリックパラメータからAMI IDを取得して指定する



テンプレートの例

SSMパラメータタイプを利用し、パブリックパラメータからAMI IDを参照する

```
AWSTemplateFormatVersion: "2010-09-09"

Parameters:
  EC2ImageId:
    Type: AWS::SSM::Parameter::Value<AWS::EC2::Image::Id>
    Default: /aws/service/ami-amazon-linux-latest/al2023-ami-kernel-default-x86_64

Resources:
  MyEC2Instance:
    Type: AWS::EC2::Instance
    Properties:
      ImageId: !Ref EC2ImageId
      InstanceType: t3.micro
```

例では Amazon Linux を指定していますが、Windows (ami-windows-latest) についても参照可能です。

※ 注意：常に最新の AMI ID を指定できる反面、AMI ID が更新された場合は EC2 の再作成が行われます。
スタック更新の際には、変更セットで意図しない EC2 の再作成が行われないことをご確認ください。

https://docs.aws.amazon.com/ja_jp/systems-manager/latest/userguide/parameter-store-public-parameters-ami.html

© 2023, Amazon Web Services, Inc. or its affiliates.

よくあるユースケース

1. 手動で作った既存リソースをCFnの管理下に入れる
2. 既存のスタックを分割する(テンプレートのリファクタリング)
3. 各AWSアカウントに基本設定を展開する
4. OS上の設定も含めてサーバー構成を管理する
5. 場面に応じてパラメータの指定方法を使い分ける
6. 最新のAMI IDを取得して指定する
7. CloudFormationで作成したリソースのバックアップを取得する

CloudFormation で作成したリソースのバックアップを取得する

実現したいこと

- ・ スタックに含まれるデータベースやファイルシステム等のステートフルコンポーネントのバックアップを取得したい
- ・ スタック全体をまとめてバックアップ対象として設定したい

解決策

- ・ AWS Backupを利用して、バックアッププランを作成する



<https://aws.amazon.com/jp/blogs/news/new-for-aws-backup-protect-and-restore-your-cloudformation-stacks/>

© 2023, Amazon Web Services, Inc. or its affiliates.

バックアップの取得

バックアッププランの作成を行い、CloudFormation スタックを選択する

2. 特定のリソースタイプを選択 [情報](#)

このバックアップ計画で保護する特定のリソースタイプを選択します。特定のリソース ID を選択から除外することもできます。

リソースタイプを選択 ▾

リソースタイプ CloudFormation スタック名

CloudFormation 削除

CloudFormation リソースを選択

backup-sample X



- ・ バックアップには、テンプレートや各パラメータの他、AWS Backupがサポートする、RDSやS3等の全てのステートフルコンポーネントが含まれます
- ・ IAM RoleやVPC等のステートレスコンポーネントも含まれます
- ・ スケジュールや保存期間、ライフサイクルルール等、他リソース同様に指定可能

バックアップから復元

(1) バックアップから復旧ポイントを選択して復元
リソース単独またはスタックを選択可能

リスト内に複数のリソースやスタックが表示されています。選択された「cb38da6b-fc60-4666-aae4-835413840eed」について、アクションメニューで「復元」オプションが強調表示されています。

復旧ポイント ID	ステータス	リソース
composite:cc416974-a21c-49f8-adea-6c2c57657c00	完了	backup
cb38da6b-fc60-4666-aae4-835413840eed	完了	myTableName
aws-tkimurz-backup-test-20231006032458-f4195a84	完了	aws-tkimurz-backup-test
image/ami-05ec08ade35322dfd	完了	-
f2d7677d-0b46-47f2-a39d-a15d23b3a658	完了	backup-sample

リソース単独の場合、新たに名称を指定したり、既存のS3バケットに復元したり等を選択可能。

スタックの場合、スタックそのものを復元するため、データベースなどのストートフルコンポーネントは、中身が空の状態で再作成される。
そのため、データそのものは別途復元が必要となる。

(2) 作成されたスタックの変更セットを確認して実行
※ スタックを復元する場合のみ

CloudFormationコンソールで「変更セット」タブが選択されています。一覧内に「restore-sample」と名づけられた変更セットが表示され、「CREATE_COMPLETE」状態で終了しています。

名前	作成時刻	ステータス
restore-sample	2023-10-06 13:26:08 UTC+0900	CREATE_COMPLETE

CloudFormationにスタックと変更セットが作成され、変更セットの実行待ちになる。

実行すればスタックに含まれるリソースが作成される。なおスタックの復元時、リソースの物理IDが重複する場合、復元時にエラーとなります。
テンプレート作成の段階で、重複しないような対策が必要です。

よくある質問



よくある質問

1. セキュリティグループが循環参照になって作成出来ない
2. 依存関係が残っていてスタックの削除に失敗する
3. CloudFormation で注意が必要な Quota

循環参照になつてセキュリティグループが作成できない

課題

- AWS::EC2::SecurityGroup で定義しても循環参照になつてしまいスタックが作れない
 - 例1) セキュリティグループAにおいて、ソースがセキュリティグループA のインバウンドアクセスを許可したい(デフォルトセキュリティグループと同じ。セキュリティグループAが設定されているもの同士は通信OK。)
 - 例2) セキュリティグループAおよびBがあり、A はソースが B のインバウンドアクセスを許可、B はソースが A のインバウンドアクセスを許可したい

解決策

- AWS::EC2::SecurityGroup で SecurityGroup を作成した後に、AWS::EC2::SecurityGroupIngress でインバウンドルールを定義する

※ 1つのテンプレート内で表現できない場合は、別スタックにするかCLIやカスタムリソースなど手続き型の処理を行う

サンプル

課題の例1

```
sgSelfReference:  
  Type: AWS::EC2::SecurityGroup  
  Properties:  
    GroupDescription: Self Reference  
  
sgSelfReferenceAllowAll:  
  Type: AWS::EC2::SecurityGroupIngress  
  Properties:  
    IpProtocol: -1  
   GroupId: !GetAtt sgSelfReference.GroupId  
    SourceSecurityGroupId: !GetAtt sgSelfReference.GroupId
```

課題の例2

```
sgCrossReference1:  
  Type: AWS::EC2::SecurityGroup  
  Properties:  
    GroupDescription: Cross Reference - 1  
  
sgCrossReference2:  
  Type: AWS::EC2::SecurityGroup  
  Properties:  
    GroupDescription: Cross Reference - 2  
  
sgCrossReference1AllowAllFrom2:  
  Type: AWS::EC2::SecurityGroupIngress  
  Properties:  
    IpProtocol: -1  
   GroupId: !GetAtt sgCrossReference1.GroupId  
    SourceSecurityGroupId: !GetAtt sgCrossReference2.GroupId  
  
sgCrossReference2AllowAllFrom1:  
  Type: AWS::EC2::SecurityGroupIngress  
  Properties:  
    IpProtocol: -1  
   GroupId: !GetAtt sgCrossReference2.GroupId  
    SourceSecurityGroupId: !GetAtt sgCrossReference1.GroupId
```



依存関係が残っていてスタックの削除に失敗する

課題

- ・ スタックを削除するとリソースが利用中のため途中でエラーになる
 - ・ CloudFormation で作ったリソースを別スタックやスタック外で使ってしまった場合 ENI周りやセキュリティグループなど
 - ・ CloudFormation で作ったS3バケットにデータが残っている場合

解決策

- ・ 依存関係を解消してスタックを削除する
- ・ 消せないリソースを保持してスタックを削除する
- ・ どこに依存関係があるかわからない場合
 - ・ AWS Configで削除に失敗するリソースを探し、依存関係を調べる
 - ・ AWS CLI の describe- で始まるコマンドを実行して確認する

参考: VPCの依存関係を確認する方法

<https://aws.amazon.com/jp/premiumsupport/knowledge-center/troubleshoot-dependency-error-delete-vpc/>



リソース保持の例

CloudFormation で注意が必要な Quota

課題

- Quota 上限に引っかかりエラーが発生してしまう

解決策

- 予め注意が必要なところを確認しておく
https://docs.aws.amazon.com/ja_jp/AWSCloudFormation/latest/UserGuide/cloudformation-limits.html
- 注意が必要な項目
 - テンプレートで宣言できるリソースの最大数 ... 500
 - テンプレート本文の最大サイズ ... 51,200 bytes
 - S3上のテンプレート本文の最大サイズ ... 1 MB
- ネストスタックを利用してすることで上限を回避することも出来る





Thank you!

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=PLzWGOASvSx6FlwIC2X1nObr1KcMCBBIqY>



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)



AWS CloudFormation

#1 基礎編

福井 敦 (Fukui Atsushi)

Partner Sales Solutions Architect

2023/7

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 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 2023 年 07月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 をこれから利用される方、概要をお知りになりたい方

前提知識

- AWS の概要を理解していること
- YAML、JSON のファイル形式について理解していること
- インフラ構成管理ツールのご利用経験があると望ましい（必須ではない）

ゴール

- CloudFormation を扱う上で必要な用語（テンプレート、スタック、変更セット）をご理解いただくこと
- CloudFormation の利用方法（スタックの作成方法とテンプレートの記述方法）についてイメージを掴んでいただくこと

自己紹介

名前

福井 敦 (Fukui Atsushi/@hukui)

所属

技術統括本部 Partner Sales Solutions Architect

経歴

国内 SIer で運用管理/インフラ設計構築を経験

好きなAWSサービス

AWS CloudFormation, AWS Systems Manager



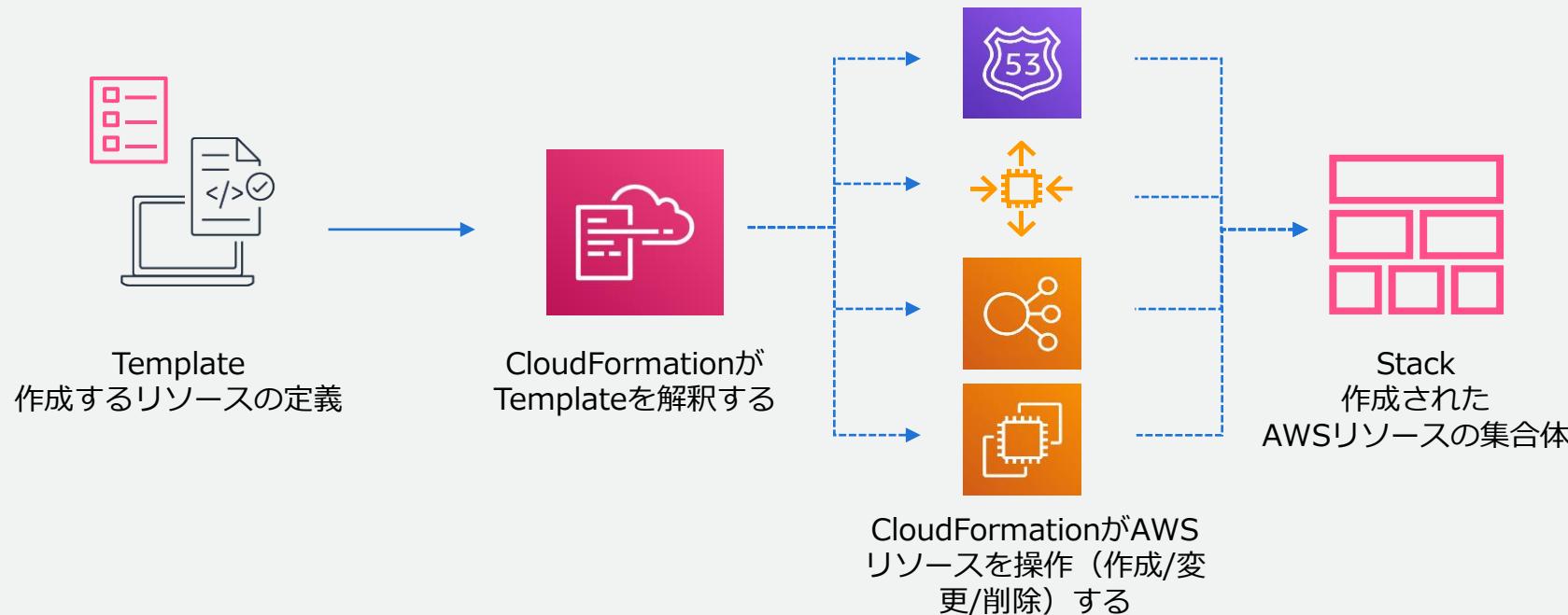
アジェンダ

1. AWS CloudFormation とは
2. CloudFormation を使った構成管理の流れ
3. テンプレートの構造
4. まとめ

AWS CloudFormation とは

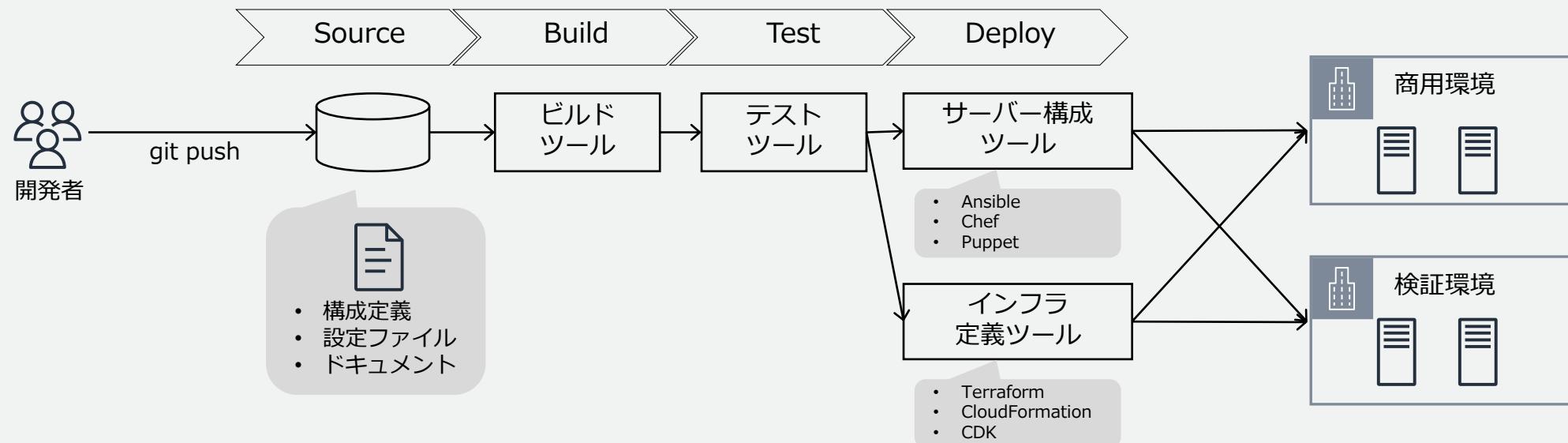
CloudFormation 概要

- あるべき状態を記載した設定ファイル（テンプレート）を元にAWSの構築を自動化できるサービス
- テンプレートにはリソースの設定情報をJSONやYAML形式で記述する
- CloudFormationは作成したリソースの集合体をスタックという単位で管理する
- 追加料金は不要（プロビジョニングされたAWSリソース分の料金のみ発生）



Infrastructure as code (IaC)

- IaC とは、インフラをコードで定義し運用するアプローチのこと
 - コードで定義、作成し、コードの変更を通じてインフラストラクチャを更新することで複製や再構築が容易になり、手動のオペレーションを排除することができる
 - Version Control System※、CI/CD を活用するソフトウェアの開発手法をインフラストラクチャに適用し、開発効率の向上を計ることができる



※ Version Control Systems。Git や Subversion といったバージョン管理のソフトウェアを指す

CloudFormation 基本機能

作成

変更

削除

- ・テンプレートに定義された構成でスタックを自動作成
- ・並列でリソースを作成し、依存関係がある場合は自動的に解決

メリット

- ✓ 構築作業を迅速化できる
- ✓ 手作業によるオペレーションミスを排除できる
- ✓ 一度テンプレートを作成すれば、検証環境の作成や別のリージョンへの展開などが容易

CloudFormation 基本機能

作成

変更

削除

- ・テンプレートで示された状態を目指し、現在の状態との差分を埋めるように働く
- ・変更セットを作ることで差分の内容とリソースへの影響を事前に確認可能
- ・スタックの作成および変更中、アプリケーションの状態に問題が発生した場合にロールバックが可能

メリット

- ✓ 幕等性、ロールバックが担保される
- ✓ 変更前後のテンプレート (=テキストファイル) の違いを見れば、インフラストラクチャに対する変更内容を確認できる

注意： 手動で行った変更は CloudFormation の管理外になります。



更新動作について

CloudFormation は、現在のスタックの状態とアップロードされたテンプレートの違いに基づいてリソースを更新する。変更のないリソースは、更新中も中断されることなく実行され、変更のあるリソースは、以下のいずれかの動作をとる。どのリソースタイプに対してどのプロパティを更新するかによって動作が異なる。各プロパティの更新動作 (Update requires) は「AWS リソースタイプのリファレンス」を参照。

中断を伴わない更新 – No interruption

リソースの使用を中断することなくリソースを更新する。物理IDに変更はない。

例：EC2のタグの変更



一時的な中断を伴う更新 – Some interruption

一部の実行を中断してリソースの更新を行う。物理IDに変更はない。

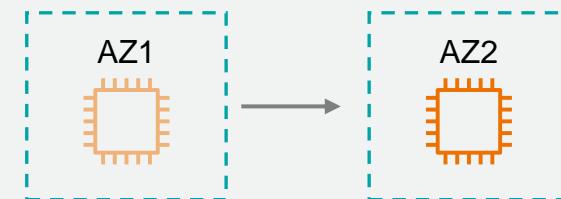
例：インスタンスタイプの変更



置換 - Replacement

リソースを再作成し、新しい物理 ID になる。まず置換先となる新しいリソースを作成してから古いリソースを削除する。

例：Availability Zone の変更



CloudFormation 基本機能

作成

変更

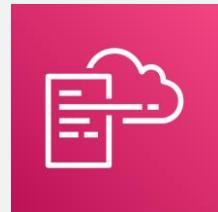
削除

- ・依存関係を解決しつつリソースを全て削除
- ・RDS のようなデータストアはスナップショットの取得 / 保持が可能

メリット

- ✓ 削除作業を迅速化できる
- ✓ 手作業によるオペレーションミスを排除できる

CloudFormation の用語



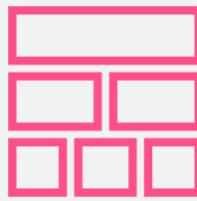
CloudFormation

スタックの作成/変更/
削除およびエラー検知
とロールバックを行う



テンプレート

リソース、属性、依存関係
についてあるべき状態を定
義するもの



スタック

テンプレートからプロ
ビジョニングされるリ
ソースの集合のこと



変更セット

テンプレートの変更前後
の差分と変更に伴う影響
(無停止変更 / 再起動 /
再作成) を事前に確認す
るもの

CloudFormation の用語 - テンプレート



- 構築したいリソースの“設計図”。どんなリソースを構築するのか
(状態)を記述

- JSON / YAML フォーマットに対応

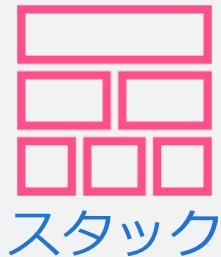
- JSON フォーマット

```
{  
    "AWSTemplateFormatVersion": "2010-09-09",  
    "Resources": {  
        "S3Bucket": {  
            "Type": "AWS::S3::Bucket",  
            "Properties": {  
                "AccessControl": "PublicRead",  
                "WebsiteConfiguration": {  
                    "IndexDocument": "index.html",  
                    "ErrorDocument": "error.html"  
                }  
            },  
            "DeletionPolicy": "Retain"  
        },  
        "BucketPolicy": {  
            "Type": "AWS::S3::BucketPolicy"  
        }  
    }  
}
```

- YAML フォーマット

```
AWSTemplateFormatVersion: 2010-09-09  
Resources:  
  S3Bucket:  
    Type: 'AWS::S3::Bucket'  
    Properties:  
      AccessControl: PublicRead  
      WebsiteConfiguration:  
        IndexDocument: index.html  
        ErrorDocument: error.html  
      DeletionPolicy: Retain  
  BucketPolicy:  
    Type: 'AWS::S3::BucketPolicy'  
    Properties:  
      PolicyDocument:  
        Id: MyPolicy  
        Version: 2012-10-17
```

CloudFormation の用語 - スタック



- テンプレートからプロビジョニングされるリソースの集合のことを
スタックと呼ぶ
 - スタック単位でリソースの管理が可能
 - スタックの削除を実行すると、スタックに紐づくリソースが削除される
 - 使用するリソースおよびリソースの構築順は、テンプレートの依存関係から
CloudFormation が自動的に決定

一例ですが、このような様々な種類のリソースを一度に作成/変更/削除が可能



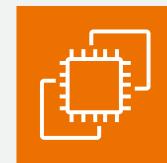
S3



DB



Web



App

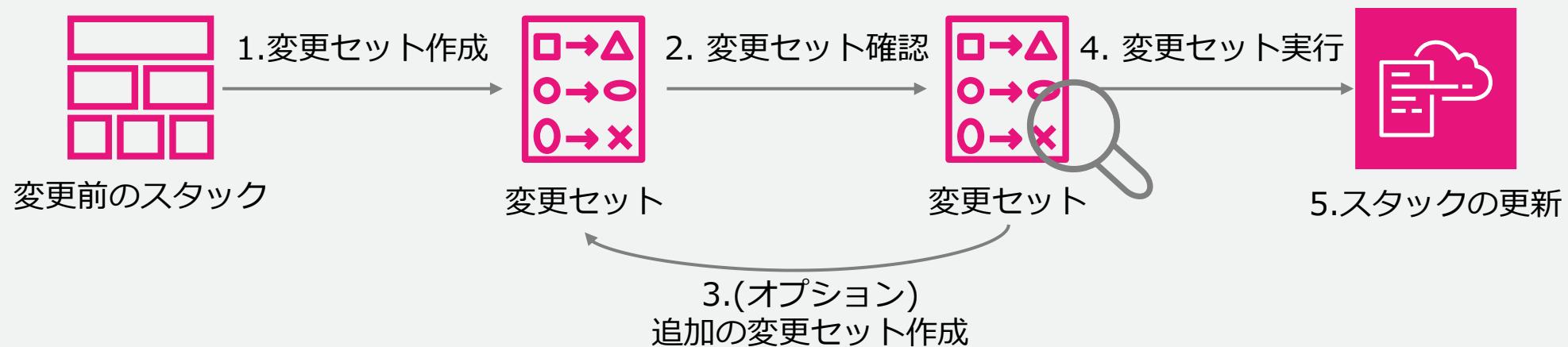


Hosted Zone

CloudFormation の用語 - 変更セット

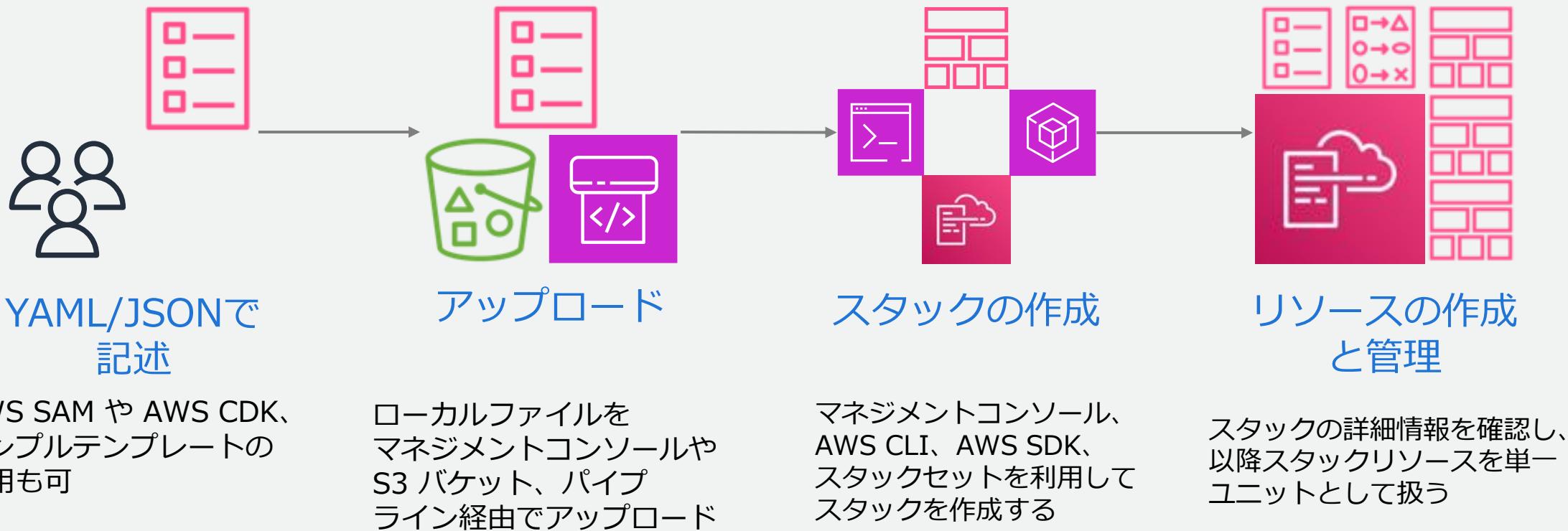


- 稼働中のリソースに与える影響をスタックの変更前に確認できる
- スタックを更新する前に、続行するか他の変更セットを作成するかを検討できる



CloudFormation を使った構成管理の流れ

CloudFormation を使った構成管理の流れ



CloudFormation を使った構成管理の流れ



- リファレンスを参考に、お好きなエディタでテンプレートを記述
 - (一例) 左図が Visual Studio Code、右図が AWS CloudFormation デザイナー
 - テンプレートの記述内容は、後述



YAML/JSONで記述



スタックの作成



The screenshot shows two side-by-side interfaces for managing CloudFormation templates.

Left Side (Visual Studio Code):

- The title bar says "CFnSample.yaml - git - Visual Studio Code".
- The editor pane displays the CloudFormation template "CFnSample.yaml".

```
1 AWSTemplateFormatVersion: "2010-09-09"
2 Description: "AWS CloudFormation #1 Sample Template"
3 Parameters:
4   EnvironmentName:
5     Default: Development
6     Type: String
7   InstanceType:
8     Default: t2.micro
9     Type: String
10  Mappings:
11    RegionMap:
12      us-west-1:
13        "AMI": "ami-0fd61683ae1a27a64"
14      us-west-2:
15        "AMI": "ami-0ae49954dfb447966"
16  Resources:
17    NewKeyPair:
18      Type: "AWS::EC2::KeyPair"
19      Properties:
20        KeyName: !Sub ${EnvironmentName} key
21    MyEC2Instance:
22      Type: AWS::EC2::Instance
23      Properties:
24        InstanceType: !Ref 'InstanceType'
```

- The status bar at the bottom indicates "Spaces: 2" and "UTF-8".

Right Side (AWS CloudFormation Designer):

- The title bar says "リソースタイプ".
- The sidebar lists various AWS services as resources: ACMPCA, APS, AccessAnalyzer, AmazonMQ, Amplify, AmplifyUIBuilder, ApiGateway, and ApiGatewayV2.
- The main canvas displays a diagram of the CloudFormation stack. It includes a "NewKeyPair" resource (represented by a key icon) and a "MyEC2Inst..." resource (represented by a square icon). A dependency arrow points from the "NewKeyPair" resource to the "MyEC2Inst..." resource.
- The bottom right corner shows a preview of the template with the title "new.template" and the language selector "JSON" (radio button selected).

https://docs.aws.amazon.com/ja_jp/AWSCloudFormation/latest/UserGuide/working-with-templates-cfn-designer.html

© 2023, Amazon Web Services, Inc. or its affiliates.

CloudFormation を使った構成管理の流れ



- AWSのマネジメントコンソールにて、前述のテンプレートをアップロード
- ローカルのファイルを選択してアップロードするか、S3にある場合はそちらを選択することも可能



https://docs.aws.amazon.com/ja_jp/AWSCloudFormation/latest/UserGuide/cfn-using-console-create-stack-template.html

© 2023, Amazon Web Services, Inc. or its affiliates.

CloudFormation を使った構成管理の流れ



- レビューページで、スタックの設定を確認
- 必要に応じてIAMリソースが作成されることを承認して、スタックを作成



スタックの作成



aws
リソースの
作成と管理

CloudFormation > スタック > スタックの作成

レビュー BlackBeltSampleStack

ステップ 1
スタックの作成

ステップ 2
スタックの詳細を指定

ステップ 3
スタックオプションの設定

ステップ 4
レビュー
BlackBeltSampleStack

テンプレート

テンプレート URL
<https://s3-ap-northeast-1.amazonaws.com/cf-templates-10jbnr11v1h23-ap-northeast-1/2023179h7Y-template1lid3zm5pnyi>

スタックの説明

The following resource(s) require capabilities: [AWS::IAM::Role]

このテンプレートには、ご利用の AWS アカウントに変更を加えるエンティティにアクセスを与える可能性を持つ Identity and Access Management (IAM) リソースが含まれています。これらのリソースを個別に作成し、それぞれに最小限必要な権限を与えるかどうか確認してください。 詳細は[こちら](#)

AWS CloudFormation によって IAM リソースが作成される場合があることを承認します。

変更セットの作成 キャンセル 戻る 送信

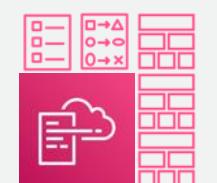
https://docs.aws.amazon.com/ja_jp/AWSCloudFormation/latest/UserGuide/cfn-using-console-create-stack-review.html

© 2023, Amazon Web Services, Inc. or its affiliates.

CloudFormation を使った構成管理の流れ



- ・ スタック作成処理が完了すると、リソースタブで確認できる
- ・ 更新する際は、再度テンプレートをアップロードして更新する



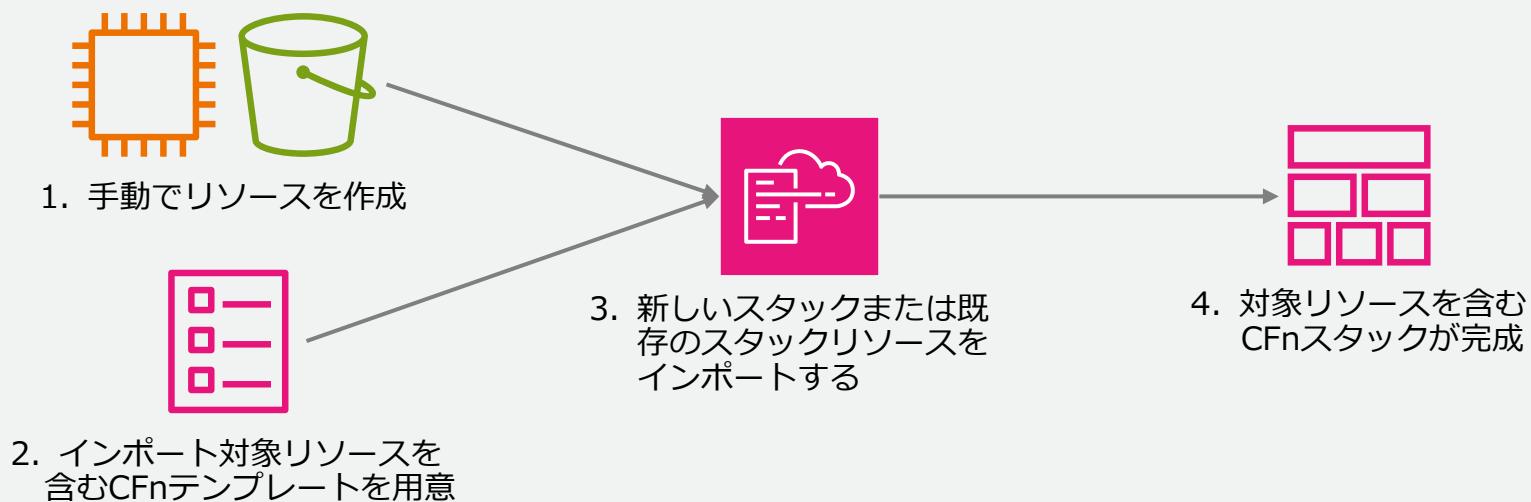
論理 ID	物理 ID	タイプ	ステータス	モジュール
MyEC2Instance	i-0c41ac428a707afaa	AWS::EC2::Instance	CREATE_COMPLETE	-
NewKeyPair	Development key	AWS::EC2::KeyPair	CREATE_COMPLETE	-

https://docs.aws.amazon.com/ja_jp/AWSCloudFormation/latest/UserGuide/cfn-using-console-create-stack-review.html

© 2023, Amazon Web Services, Inc. or its affiliates.

インポート機能について

- 手動で作成した AWS リソースをあとから CloudFormation スタックにインポートして管理可能
 - リソースをスタックの管理下から切り離したり、別のスタック管理下に移動することも可能
 - インポート対象のリソースの実際の設定と一致するようテンプレートを用意する
 - AWS 公式のツールではないが、Former2 でテンプレート作成の省力化が可能
 - <https://github.com/iann0036/former2>



テンプレートの構造

テンプレートの要素

- テンプレートは下表に示すセクションから構成される
 - Resourcesだけが必須のセクション
 - セクションの順序は任意だが、あるセクションの値が前のセクションの値を参照する場合があるため、下表の順序を推奨

順序	セクション名	説明	必須
1	AWSTemplateFormatVersion	テンプレートのバージョン	No
2	Description	テンプレートの説明文	No
3	Metadata	テンプレートに関する追加情報	No
4	Parameters	実行時にユーザ入力を求めるパラメータ	No
5	Rules	スタックの作成または更新前に入力されたパラメータを検証	No
6	Mappings	条件パラメータ値の指定に使用	No
7	Conditions	リソースが作成または設定される条件を登録	No
8	Transform	変換および拡張処理の呼出しに使用	No
9	Resources	スタックを構成するリソースを定義	Yes
10	Outputs	スタック構築後に output させる値 (DNS名やIPアドレスなど)	No

https://docs.aws.amazon.com/ja_jp/AWSCloudFormation/latest/UserGuide/template-anatomy.html

テンプレートの記述例

- 書き出しあは、**AWSTemplateFormatVersion** と **Description** で始めましょう
 - 最新のフォーマットバージョンは 2010-09-09 であり、現時点での唯一の有効な値
 - Description セクションにテンプレートに関する説明を記載

```
AWSTemplateFormatVersion: "2010-09-09"
Description: "AWS CloudFormation #1 Sample Template"
```

順序	セクション名	説明
1	AWSTemplateFormatVersion	テンプレートのバージョン
2	Description	テンプレートの説明文
3	Metadata	テンプレートに関する追加情報
4	Parameters	実行時にユーザ入力を求めるパラメータ
5	Rules	スタックの作成または更新前にパラメータを検証
6	Mappings	条件パラメータ値の指定に使用
7	Conditions	リソースが作成または設定される条件を登録
8	Transform	変換および拡張処理の呼び出しに使用
9	Resources	スタックを構成するリソースを定義
10	Outputs	スタック構築後に output させる値



テンプレートの記述例

- 必須の **Resources** セクションに作成したい EC2 インスタンスなどのリソースを定義しましょう
 - リソースタイプ毎に定められているプロパティを記述する
 - 利用可能なリソースタイプとプロパティはドキュメントを参照

順序	セクション名	説明
1	AWSTemplateFormatVersion	テンプレートのバージョン
2	Description	テンプレートの説明文
3	Metadata	テンプレートに関する追加情報
4	Parameters	実行時にユーザ入力を求めるパラメータ
5	Rules	スタックの作成または更新前にパラメータを検証
6	Mappings	条件パラメータ値の指定に使用
7	Conditions	リソースが作成または設定される条件を登録
8	Transform	変換および拡張処理の呼出しに使用
9	Resources	スタックを構成するリソースを定義
10	Outputs	スタック構築後に出力させる値

```
AWSTemplateFormatVersion: "2010-09-09"
Description: "AWS CloudFormation #1 Sample Template"
```

```
Resources:
  NewKeyPair:
    Type: 'AWS::EC2::KeyPair'
    Properties:
      KeyName: Development key
  MyEC2Instance:
    Type: AWS::EC2::Instance
    Properties:
      InstanceType: t2.micro
      ImageId: ami-0ae49954dfb447966
      KeyName: !Ref NewKeyPair
```



AWS::EC2::KeyPair

Specifies a key pair for use with an Amazon Elastic Compute Cloud instance or follows:

- To import an existing key pair include the `PublicKeyMaterial` property.
- To create a new key pair omit the `PublicKeyMaterial` property.

When you import an existing key pair, you specify the public key material for the key. CloudFormation does not create or renew the private key material when you import a key pair.

When you create a new key pair, the private key is saved to AWS Systems Manager Parameter Store, using a parameter with the following name: /AWS/CloudFormation/KeyPairs/`key-pair-name`. For more information about retrieving private key and the required permissions, see [Create a key pair using AWS CloudFormation](#) in the [Amazon EC2 User Guide](#).

When AWS CloudFormation deletes a key pair that was created or imported by a stack, it also deletes the parameter that was used to store the private key material in Parameter Store.

Syntax

To declare this entity in your AWS CloudFormation template, use the following syntax:

YAML

```
Type: AWS::EC2::KeyPair  
Properties:  
  PublicKeyMaterial: String  
  KeyName: String  
  KeyType: String  
  PublicKeyMaterialString: String  
  Tags: Tag[]
```

Properties

`KeyFormat`

The format of the key pair.

Default: pem

Required: No

Type: String

Allowed values: pem | ppk

Update requires: Replacement

`KeyName`

A unique name for the key pair.

Constraints: Up to 256 ASCII characters

Required: Yes

Type: String

Update requires: Replacement

`KeyType`

The type of key pair. Note that DSS256 SHA keys are not supported for Windows instances.

If the `PublicKeyMaterial` property is specified, the `KeyType` property is ignored, and the key type is inferred from the `PublicKeyMaterial` value.

Default: rsa

Required: No

Type: String

Allowed values: RSA1024 | RSA2048

Update requires: Replacement

`PublicKeyMaterial`

The public key material. The `PublicKeyMaterial` property is used to import a key pair; if this property is not specified, then a new key pair will be created.

Required: No

Type: String

Update requires: Replacement

`Tags`

The tags to apply to the key pair.

Required: No

Type: List of Tag

Update requires: Replacement

Return values

Ref

When you pass the logical ID of this resource to the intrinsic `Ref` function, `Ref` returns the name of the key pair.

For more information about using the `Ref` function, see [Ref](#).

Fn::GetAtt

`KeyFingerprint`

If you create the key pair using Amazon EC2:

- For RSA key pairs, the key fingerprint is the SHA-1 digest of the DER encoded private key.
- For DSS256 key pairs, the key fingerprint is the base64-encoded SHA-256 digest, which is the default for OpenSSH 4.7 and later.

If you import the key pair to Amazon EC2:

- For RSA key pairs, the key fingerprint is the MD5 public key fingerprint as specified in section 4 of RFC 4716.
- For DSS256 key pairs, the key fingerprint is the base64-encoded SHA-256 digest, which is the default for OpenSSH 4.7 and later.

`KeyId`

The ID of the key pair.

Examples

Create a new key pair and specify it when launching an instance.

The following example omits the `PublicKeyMaterial` property to create a new key pair, and specifies the key pair when launching an instance.

YAML

```
#!/bin/bash  
#!/bin/bash  
Name: MyKeyPair  
Type: AWS::EC2::KeyPair  
Properties:  
  KeyName: MyKeyPair  
  KeyType: rsa
```

AWS::EC2::KeyPair

RSS

フィルタビュー

All	▲
All	✓
JSON	
YAML	

Specifies a key pair for use with an Amazon Elastic Compute Cloud follows:

- To import an existing key pair, include the `PublicKeyMaterial` property.

Specifies a key pair for use with an Amazon Elastic Compute Cloud instance as follows:

- To import an existing key pair, include the `PublicKeyMaterial` property.
- To create a new key pair, omit the `PublicKeyMaterial` property.

When you import an existing key pair, you specify the public key material for the key. We assume that you have the private key material for the key. AWS CloudFormation does not create or return the private key material when you import a key pair.

When you create a new key pair, the private key is saved to AWS Systems Manager Parameter Store, using a parameter with the following name: `/AWS/KeyPairs/(key_name)_priv`. For more information about retrieving private key and the required permissions, see [Create a key pair using AWS CloudFormation](#) in the [Amazon EC2 User Guide](#).

When AWS CloudFormation deletes a key pair that was created or imported by a stack, it also deletes the parameter that was used to store the private key material in Parameter Store.

Syntax

To declare this entity in your AWS CloudFormation template, use the following syntax:

YAML

```
Type: AWS::EC2::KeyPair
Properties:
  KeyFormat: String
  KeyName: String
  KeyType: String
  PublicKeyMaterial: String
  Tags: Tag
```

Properties

KeyFormat
 The format of the key pair.
 Default: pem
 Required: No
 Type: String
 Allowed values: pem | ppk
 Update requires: Replacement

KeyName
 A unique name for the key pair.
 Constraints: Up to 256 ASCII characters
 Required: Yes
 Type: String
 Update requires: Replacement

KeyType
 The type of key pair. Note that DSS256 SHA keys are not supported for Windows instances.
 If the `PublicKeyMaterial` property is specified, the `KeyType` property is ignored, and the key type is inferred from the `PublicKeyMaterial` value.
 Default: rsa
 Required: No
 Type: String
 Allowed values: RSA1024 | RSA
 Update requires: Replacement

PublicKeyMaterial
 The public key material. The `PublicKeyMaterial` property is used to import a key pair; if this property is not specified, then a new key pair will be created.
 Required: No
 Type: String
 Update requires: Replacement

Tags
 The tags to apply to the key pair.
 Required: No
 Type: List of Tag
 Update requires: Replacement

Return values

Ref
 When you pass the logical ID of this resource to the intrinsic `Ref` function, `Ref` returns the name of the key pair.
 For more information about using the `Ref` function, see [Ref](#).

Fn::GetAtt

KeyFingerprint
 If you create the key pair using Amazon EC2:

- For RSA key pairs, the key fingerprint is the SHA-1 digest of the DER encoded private key.
- For DSS256 SHA key pairs, the key fingerprint is the base64-encoded SHA-256 digest, which is the default for OpenSSH, starting with OpenSSH 4.7.0.

 If you import the key pair as Amazon EC2:

- For RSA key pairs, the key fingerprint is the MD5 public key fingerprint as specified in section 4 of RFC 4716.
- For DSS256 SHA key pairs, the key fingerprint is the base64-encoded SHA-256 digest, which is the default for OpenSSH, starting with OpenSSH 4.7.0.

KeyPairId

The ID of the key pair.

Examples

Create a new key pair and specify it when launching an instance

The following example omits the `PublicKeyMaterial` property to create a new key pair, and specifies the key pair when launching an instance.

YAML

```
#!/bin/bash
NewKeyPair:
  Type: AWS::EC2::KeyPair
  Properties:
    KeyName: MyKeyPair
    PublicKeyMaterial: !Sub
```

Syntax

To declare this entity in your AWS CloudFormation template, use the following syntax:

YAML

Type: AWS::EC2::KeyPair

Properties:

KeyFormat: String

KeyName: String

KeyType: String

PublicKeyMaterial: String

Tags:

- Tag



Specifies a key pair for use with an Amazon Elastic Compute Cloud instance as follows:

- To import an existing key pair, include the `PublicKeyMaterial` property.
- To create a new key pair, omit the `PublicKeyMaterial` property.

When you import an existing key pair, you specify the public key material for the key. We assume that you have the private key material for the key. AWS CloudFormation does not store or return the private key material.

When you create a new key pair, the private key is saved to AWS Systems Manager Parameter Store, using a parameter with the following name: `/AWS/KeyPairs/(key-pair-name)/private-key`. For more information, see [AWS Systems Manager Parameter Store](#).

When AWS CloudFormation deletes a key pair that was created or imported by a stack, it also deletes the parameter that was used to store the private key material in Parameter Store.

Syntax

To declare this entity in your AWS CloudFormation template, use the following syntax:

YAML

```
Type: AWS::EC2::KeyPair
Properties:
  KeyFormat: String
  KeyName: String
  KeyType: String
  PublicKeyMaterial: String
  Tags: Tag
```

Properties

KeyFormat
The format of the key pair.
Default: pem
Required: No
Type: String
Allowed values: pem | ppk
Update requires: Replacement

KeyName
A unique name for the key pair.
Constraints: Up to 255 ASCII characters
Required: Yes
Type: String
Update requires: Replacement

KeyType
The type of key pair. Note: DSS256 SHA keys are not supported for unknown instances.
If the `PublicKeyMaterial` property is specified, the `KeyType` property is ignored, and the key type is inferred from the `PublicKeyMaterial` value.
Default: rsa
Required: No
Type: String
Allowed values: EC2SSHP256 | rsa
Update requires: Replacement

PublicKeyMaterial
The public key material. The `PublicKeyMaterial` property is used to import a key pair; if this property is not specified, then a new key pair will be created.
Required: No
Type: String
Update requires: Replacement

Tags
The tags to apply to the key pair.
Required: No
Type: List of Tag
Update requires: Replacement

Return values

Ref

When you pass the logical ID of this resource to the intrinsic `Ref` function, `Ref` returns the name of the key pair.

For more information about using the `Ref` function, see [Ref](#).

Fn::GetAtt

If you create the key pair using Amazon EC2:

- For RSA key pairs, the key fingerprint is the SHA-1 digest of the DER encoded private key.
- For DSS256 SHA key pairs, the key fingerprint is the base64-encoded SHA-256 digest, which is the default for OpenSSH, starting with OpenSSH 4.7.5.

If you import the key pair as Amazon EC2:

- For RSA key pairs, the key fingerprint is the MD5 public key fingerprint as specified in section 4 of RFC 4716.
- For DSS256 SHA key pairs, the key fingerprint is the base64-encoded SHA-256 digest, which is the default for OpenSSH, starting with OpenSSH 4.7.5.

KeyFingerprint

The ID of the key pair.

Examples

Create a new key pair and specify it when launching an instance

The following example omits the `PublicKeyMaterial` property to create a new key pair, and specifies the key pair when launching an instance.

YAML

```
AWSTestKey:
  Type: AWS::EC2::KeyPair
  Properties:
    KeyName: MyKeyPair
    KeyType: rsa
```

Properties

KeyFormat

The format of the key pair.

Default: pem

Required: No

Type: String

Allowed values: pem | ppk

Update requires: Replacement

KeyName

A unique name for the key pair.

Constraints: Up to 255 ASCII characters

Required: Yes

Type: String

Update requires: Replacement

KeyType

Specifies a key pair for use with an Amazon Elastic Compute Cloud instance as follows:

- To import an existing key pair, include the `PublicKeyMaterial` property.
- To create a new key pair, omit the `PublicKeyMaterial` property.

When you import an existing key pair, you specify the public key material for the key. We assume that you have the private key material for the key. AWS CloudFormation does not create or return the private key material.

When you create a new key pair, the private key is saved to AWS Systems Manager Parameter Store, using a parameter with the following name: `/AWS/KeyPairs/(key_name)_priv`. For more information about AWS Systems Manager Parameter Store, see [AWS Systems Manager Parameter Store](#).

When AWS CloudFormation deletes a key pair that was created or imported by a stack, it also deletes the parameter that was used to store the private key material in Parameter Store.

Syntax

To declare this entity in your AWS CloudFormation template, use the following syntax:

YAML

```
Type: AWS::EC2::KeyPair
Properties:
  KeyName: String
  PublicKeyMaterial: String
  KeyType: String
  PublicKeyMaterial: String
  Tags: Tag
```

Properties

KeyName
The name of the key pair.
Default: `key`
Required: No
Type: String
Allowed values: `pk1` | `pk2`
Update requires: Replacement

KeyType
A unique name for the key pair.
Constraints: Up to 256 ASCII characters
Required: Yes
Type: String
Update requires: Replacement

PublicKeyMaterial
The type of key pair. Note: DSS256 SHA keys are not supported for Windows instances.
If the `PublicKeyMaterial` property is specified, the `KeyType` property is ignored, and the key type is inferred from the `PublicKeyMaterial` value.
Default: `RSA`
Required: No
Type: String
Allowed values: `RSA1024` | `RSA`
Update requires: Replacement

PublicKeyMaterial
The public key material. The `PublicKeyMaterial` property is used to import a key pair; if this property is not specified, then a new key pair will be created.
Required: No
Type: String
Update requires: Replacement

Tags
The tags to apply to the key pair.
Required: No
Type: List of Tag
Update requires: Replacement

Return values

Ref
When you pass the logical ID of this resource to the intrinsic `!Ref` function, `!Ref` returns the name of the key pair.
For more information about using the `!Ref` function, see [Using the !Ref Function](#).

Fn::GetAtt

KeyFingerprint
If you create the key pair using Amazon EC2:
+ For RSA key pairs, the key fingerprint is the SHA-1 digest of the DER encoded private key.
+ For DSS256 SHA key pairs, the key fingerprint is the base64-encoded SHA-256 digest, which is the default for OpenSSH, starting with OpenSSH 4.7.15.
If you import the key pair as Amazon EC2:
+ For RSA key pairs, the key fingerprint is the MD5 public key fingerprint as specified in section 8 of RFC 4716.
+ For DSS256 SHA key pairs, the key fingerprint is the base64-encoded SHA-256 digest, which is the default for OpenSSH, starting with OpenSSH 4.7.15.

KeyPairId

The ID of the key pair.

Examples

Create a new key pair and specify it when launching an instance

The following example omits the `PublicKeyMaterial` property to create a new key pair, and specifies the key pair when launching an instance.

YAML

```
AWSTemplateFormatVersion: '2010-09-09'
Resources:
  NewKeyPair:
    Type: 'AWS::EC2::KeyPair'
    Properties:
      KeyName: MyKeyPair
  Ec2Instance:
    Type: 'AWS::EC2::Instance'
    Properties:
      ImageId: ami-02b92c281a4d3dc79
      KeyName: !Ref NewKeyPair
```

Examples

Create a new key pair and specify it when launching an instance

The following example omits the `PublicKeyMaterial` property to create a new key pair, and specifies the key pair when launching an instance.

YAML

Resources:

NewKeyPair:

Type: 'AWS::EC2::KeyPair'

Properties:

KeyName: MyKeyPair

Ec2Instance:

Type: 'AWS::EC2::Instance'

Properties:

ImageId: ami-02b92c281a4d3dc79

KeyName: !Ref NewKeyPair

テンプレートの記述例

- ・ スタック作成時、ユーザに設定値を入力させたい場合は **Parameters** セクションを使いましょう
 - ・ データ型、デフォルト値、最大最小値などのプロパティを設定可能
 - ・ 入力された値は、テンプレート中で“!Ref”という組み込み関数を使って参照

順序	セクション名	説明
1	AWSTemplateFormatVersion	テンプレートのバージョン
2	Description	テンプレートの説明文
3	Metadata	テンプレートに関する追加情報
4	Parameters	実行時にユーザ入力を求めるパラメータ
5	Rules	スタックの作成または更新前にパラメータを検証
6	Mappings	条件パラメータ値の指定に使用
7	Conditions	リソースが作成または設定される条件を登録
8	Transform	変換および拡張処理の呼び出しに使用
9	Resources	スタックを構成するリソースを定義
10	Outputs	スタック構築後に出力させる値

```
AWSTemplateFormatVersion: "2010-09-09"
Description: "AWS CloudFormation #1 Sample Template"
Parameters:
  EnvironmentName:
    Default: Development
    Type: String
  InstanceType:
    Default: t2.micro
    Type: String
```

```
Resources:
  NewKeyPair:
    Type: 'AWS::EC2::KeyPair'
    Properties:
      KeyName: !Sub ${EnvironmentName} key
  MyEC2Instance:
    Type: AWS::EC2::Instance
    Properties:
      InstanceType: !Ref 'InstanceType'
      ImageId: ami-0ae49954dfb447966
      KeyName: !Ref NewKeyPair
```



テンプレートの記述例

- リージョンや環境別に AMI を使いわけたいなど、条件によって設定値を変えたい場合は、**Mappings** セクションを使いましょう
 - キーと値のマッピングテーブルによってテンプレートの再利用性が向上
 - 組み込み関数“Find::InMap”を使って値を取得

順序	セクション名	説明
1	AWSTemplateFormatVersion	テンプレートのバージョン
2	Description	テンプレートの説明文
3	Metadata	テンプレートに関する追加情報
4	Parameters	実行時にユーザ入力を求めるパラメータ
5	Rules	スタックの作成または更新前にパラメータを検証
6	Mappings	条件パラメータ値の指定に使用
7	Conditions	リソースが作成または設定される条件を登録
8	Transform	変換および拡張処理の呼出しに使用
9	Resources	スタックを構成するリソースを定義
10	Outputs	スタック構築後に出力させる値

```
AWSTemplateFormatVersion: "2010-09-09"
Description: "AWS CloudFormation #1 Sample Template"
Parameters:
  EnvironmentName:
    Default: Development
    Type: String
  InstanceType:
    Default: t2.micro
    Type: String
Mappings:
  RegionMap:
    us-west-1:
      "AMI": "ami-0fd61683ae1a27a64"
    us-west-2:
      "AMI": "ami-0ae49954dfb447966"
Resources:
  NewKeyPair:
    Type: 'AWS::EC2::KeyPair'
    Properties:
      KeyName: !Sub ${EnvironmentName} key
  MyEC2Instance:
    Type: AWS::EC2::Instance
    Properties:
      InstanceType: !Ref 'InstanceType'
      ImageId:
        Fn::FindInMap:
          - "RegionMap"
          - Ref: "AWS::Region"
          - "AMI"
      KeyName: !Ref NewKeyPair
```



テンプレートの記述例

- インスタンス ID や IP アドレスなど、スタック構築後に確認、使用したい情報がある場合は **Outputs** セクションを使いましょう
 - Outputs (出力) のエクスポート名を介して別のスタックからリソースを参照する用途にも用いる（クロススタック参照）

順序	セクション名	説明
1	AWSTemplateFormatVersion	テンプレートのバージョン
2	Description	テンプレートの説明文
3	Metadata	テンプレートに関する追加情報
4	Parameters	実行時にユーザ入力を求めるパラメータ
5	Rules	スタックの作成または更新前にパラメータを検証
6	Mappings	条件パラメータ値の指定に使用
7	Conditions	リソースが作成または設定される条件を登録
8	Transform	変換および拡張処理の呼出しに使用
9	Resources	スタックを構成するリソースを定義
10	Outputs	スタック構築後に出力させる値

```
AWSTemplateFormatVersion: "2010-09-09"
Description: "AWS CloudFormation #1 Sample Template"
Parameters:
  EnvironmentName:
    Default: Development
    Type: String
  InstanceType:
    Default: t2.micro
    Type: String
Mappings:
  RegionMap:
    us-west-1:
      "AMI": "ami-0fd61683ae1a27a64"
    us-west-2:
      "AMI": "ami-0ae49954dfb447966"
Resources:
  NewKeyPair:
    Type: 'AWS::EC2::KeyPair'
    Properties:
      KeyName: !Sub ${EnvironmentName} key
  MyEC2Instance:
    Type: AWS::EC2::Instance
    Properties:
      InstanceType: !Ref 'InstanceType'
      ImageId:
        Fn::FindInMap:
          - "RegionMap"
          - Ref: "AWS::Region"
          - "AMI"
      KeyName: !Ref NewKeyPair
Outputs:
  MyEC2InstanceId:
    Value: !Ref MyEC2Instance
  MyEC2InstancePrivateIp:
    Value: !GetAtt MyEC2Instance.PrivateIp
```



テンプレートの要素と構成管理の流れ

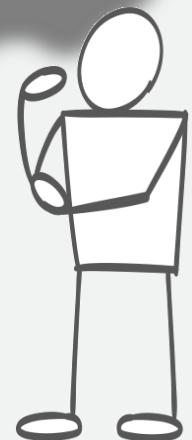
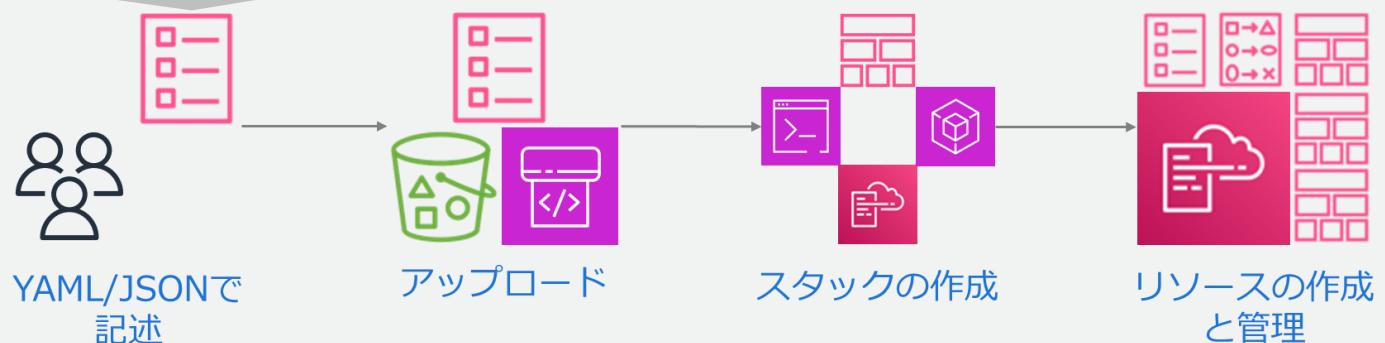
順序	セクション名	説明
1	AWSTemplateFormatVersion	テンプレートのバージョン
2	Description	テンプレートの説明文
3 Metadata	テンプレートに関する追加情報	
4	Parameters	実行時にユーザ入力を求めるパラメータ
5 Rules	スタックの作成または更新前にパラメータを検証	
6	Mappings	条件パラメータ値の指定に使用
7 Conditions	リソースが作成または設定される条件を登録	
8 Transform	変換および拡張処理の呼び出しに使用	
9	Resources	スタックを構成するリソースを定義
10	Outputs	スタック構築後に output させる値

→ **Metadata セクション**

→ **Rules セクション**

→ **Condition セクション**

→ **Transform セクション**



まとめ

まとめ

- AWS CloudFormation とは
 - **AWS CloudFormation** とは、あるべき状態を記載した設定ファイル（テンプレート）を元にAWS の構築を自動化できるサービス
 - **テンプレート**とは、構築したいリソースの“設計図”。どんなリソースを構築するのか(状態)を記述したテキストファイル
 - **スタック**とは、テンプレートからプロビジョニングされるリソースの集合
 - **変更セット**を作成すると稼働中のリソースに与える影響をスタックの変更前に確認できる
- テンプレートの構成
 - テンプレートを構成する**セクション**について記述例を通して紹介しました

参考資料

- ドキュメント
 - https://docs.aws.amazon.com/ja_jp/cloudformation/index.html#lang/ja_jp
- よくある質問
 - <https://aws.amazon.com/jp/cloudformation/faqs/>
- aws re:Post 情報センター
 - <https://repost.aws/ja/tags/knowledge-center/TAm3R3LNU3RfSX9L23YIpo3w/aws-cloudformation>
- AWS CloudFormation Workshop
 - <https://catalog.workshops.aws/cfn101/en-US>
- ロードマップ
 - <https://github.com/aws-cloudformation/cloudformation-coverage-roadmap>
- AWS 初心者向けハンズオン (AWS Hands-on for Beginners)
 - <https://aws.amazon.com/jp/events/aws-event-resource/hands-on/>
- AWS クラウドサービス活用資料集 (AWS Black Belt Online Seminar)
 - <https://aws.amazon.com/jp/events/aws-event-resource/archive/>



Thank you!

Appendix

AWSTemplateFormatVersion と Description

- AWSTemplateFormatVersion
 - 最新のフォーマットバージョンは 2010-09-09 であり、現時点で唯一の有効な値
 - 値を指定しない場合、CloudFormation は最新のフォーマットバージョンを使用する
- Description
 - Description セクションにテンプレートに関する説明やコメントをつけることができる
 - Description セクションだけを修正してスタックを更新することはできない
 - 最大1024バイトのリテラル文字列にする必要があり関数やパラメータを使うことはできない

```
AWSTemplateFormatVersion: "2010-09-09"
Description: >
  Here are some
  details about
  the template.
```

https://docs.aws.amazon.com/ja_jp/AWSCloudFormation/latest/UserGuide/format-version-structure.html
https://docs.aws.amazon.com/ja_jp/AWSCloudFormation/latest/UserGuide/template-description-structure.html

Metadata

- テンプレートに関する追加情報として JSON または YAML オブジェクトを含むことができる
 - Stack の一部として作成する EC2 インスタンスから、この Metadata を取得して初期設定に活用したりできる
 - Metadata セクションにパスワードなどの機密情報を書かないこと
 - Metadata セクションだけを修正して Stack を更新することはできない

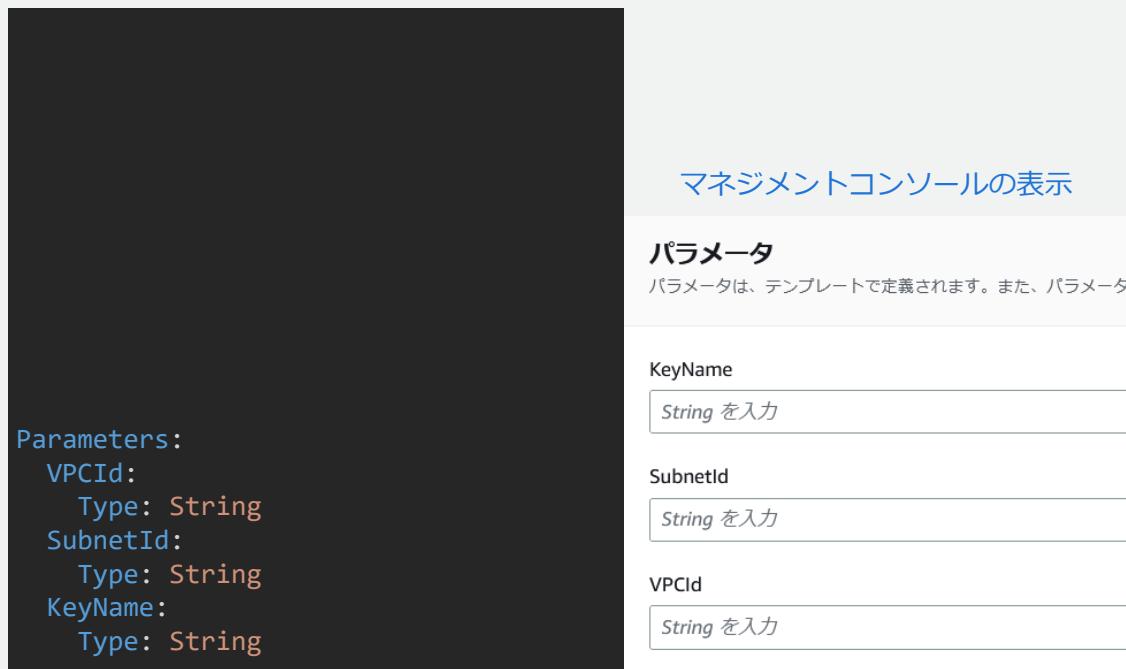
```
Metadata:  
  Instances:  
    Description: "Information about the instances"  
  Databases:  
    Description: "Information about the databases"
```

Metadata Key

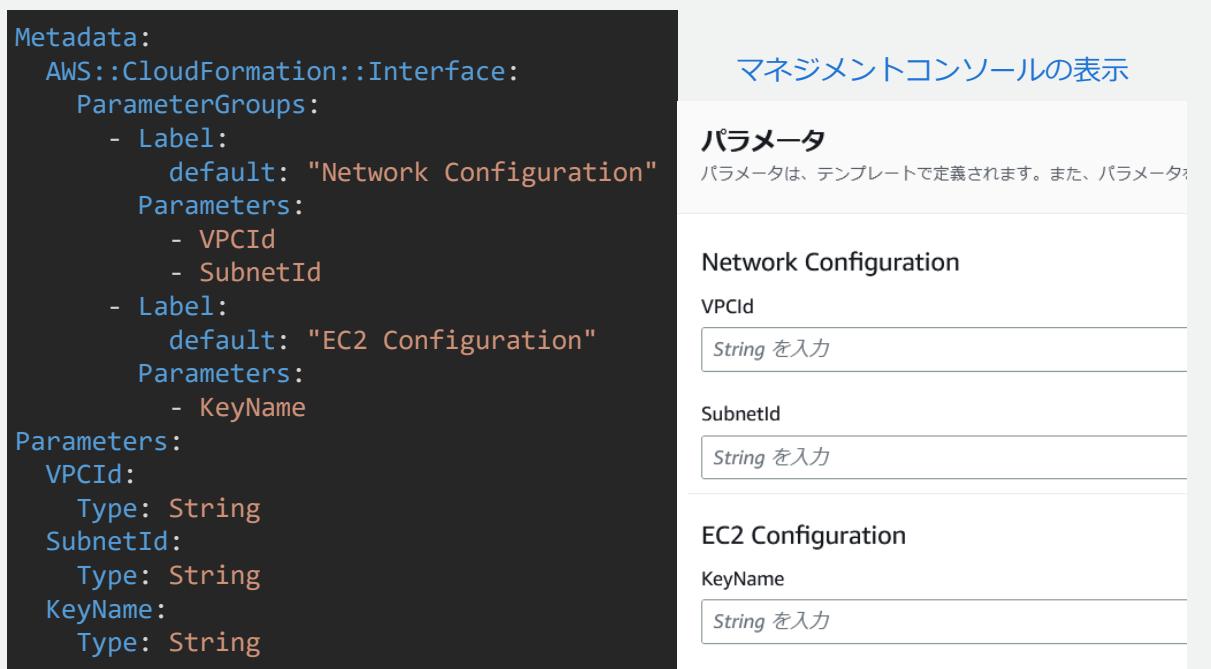
CloudFormation の一部の機能(パラメータや CloudFormation デザイナーなど)から、Metadata セクションで定義した内容を利用できる。

例えば、AWS::CloudFormation::Interface というキーを用いてパラメーターをグループ化し、順序を指定できる(マネジメントコンソールの画面上でパラメータをわかりやすく表示できる)

AWS::CloudFormation::Interface メタデータキーがないとき



AWS::CloudFormation::Interface メタデータキーがあるとき



Parameters

- パラメーターを使用すると、スタックを作成または更新するたびにユーザに指定させる値を定義できる
 - データ型、デフォルト値、最大最小値などのプロパティを設定可能
 - 入力されたパラメータの値は、テンプレート中で”!Ref”という組み込み関数を使って参照
 - テンプレートあたり最大200 個のパラメーターを指定可能

```
Parameters:  
S3NameParam:  
  Type: String  
  Default: mybucket  
  Description: Name for your AWS S3 bucket  
  MinLength: 5  
  MaxLength: 30  
Resources:  
S3Bucket:  
  Type: AWS::S3::Bucket  
  Properties:  
    AccessControl: PublicRead  
    BucketName: !Ref S3NameParam  
    DeletionPolicy: Retain
```

https://docs.aws.amazon.com/ja_jp/AWSCloudFormation/latest/UserGuide/parameters-section-structure.html

Parameters のプロパティ

プロパティ	内容
Type	データ型。String, Number, CommaDelimitedList, AWS 固有のパラメータタイプ, SSM パラメータータイプ をサポート
Default	デフォルト値。スタックの作成時に値を指定しなかった場合に使用される
NoEcho	入力されたパラメータ値をアスタリスク(*)でマスクする
AllowedValues	入力可能値の一覧指定 (例 : ["true","false"])
AllowedPattern	正規表現で入力可能パターンを指定 (例 : [a-zA-Z]*)
MaxLength	最大文字数
MinLength	最小文字数
MaxValue	最大値
MinValue	最小値
Description	パラメータの詳細説明。最大4000文字
ConstraintDescription	入力した値がAllowedPatternやMaxLengthなどの制約に引っかかった場合に表示する説明 (どのような制約があるかの説明を記述)

https://docs.aws.amazon.com/ja_jp/AWSCloudFormation/latest/UserGuide/parameters-section-structure.html



擬似パラメータ (Pseudo Parameter)

CloudFormation によって事前定義されていて、テンプレートでは宣言不要のパラメータ群。“Ref” で参照できる

擬似パラメータ名	説明
AWS::AccountId	AWSアカウントIDを取得
AWS::NotificationARNs	notificationAmazonResourceNames(ARNs)を取得
AWS::NoValue	指定されたプロパティを無視するようCloudFormationに伝える
AWS::Region	リージョン名を取得
AWS::StackId	スタックIDを取得
AWS::StackName	スタック名を取得

```
MyDB:  
  Type: AWS::RDS::DBInstance  
  Properties:  
    DBSnapshotIdentifier:  
      Fn::If:  
        - UseDBSnapshot  
        - Ref: DBSnapshotName # UseDBSnapshotがTrueのとき「DBSnapshotIdentifier」としてDBSnapshotNameの値を使う  
        - Ref: AWS::NoValue # Falseのときプロパティ「DBSnapshotIdentifier」が定義されていないものとしてDBSnapshotIdentifierを無視する  
    Tags:  
      - key:  
        Value: !Ref AWS::Region
```



組み込み関数

- 組み込み関数は、パラメータの参照や値の加工などに利用する
 - 記法は「Fn::<関数名>」。YAMLの場合は、短縮形「!<関数名>」も利用可

完全関数名	短縮形(YAMLの場合)	機能概要
Fn::Base64	!Base64	文字列をBase64エンコードする
Fn::FindInMap	!FindInMap	Mappingsのキーに対応する値を取得する
Fn::ForEach	N/A	ループ処理によって繰り返し定義する
Fn::GetAtt	!GetAtt	リソースの属性値を取得する 例) "Fn::GetAtt" : ["MyELB", "DNSName"]
Fn::GetAZs	!GetAZs	指定したリージョンのアベイラビリティゾーンのリストを取得する
Fn::ImportValue	!ImportValue	別のスタックにてエクスポートされた出力の値を取得する (クロススタック参照)
Fn::Join	!Join	文字列を結合する 例) "Fn::Join" : [":", ["a", "b"]] は 「a:b」 を返す
Fn::Select	!Select	Index値に応じた値をListから選択する 例) { "Fn::Select" : ["1", ["Jan", "Feb", "Mar", "Apr", "Jun"]] } は"Feb"を返す
Fn::Sub	!Sub	文字列内の変数を指定した値で置き換える
Ref	!Ref	指定したパラメータのキーから値、またはリソースの論理IDから物理IDを参照する

https://docs.aws.amazon.com/ja_jp/AWSCloudFormation/latest/UserGuide/intrinsic-function-reference.html



Resources

- 唯一、必須のセクション。作成、更新する Amazon EC2 インスタンスや Amazon S3 バケットなどのリソースを定義する
 - リソース毎に定められているプロパティを記述する
 - 利用可能なリソースタイプとプロパティはリファレンスを参照
 - プロパティの値には、リテラル文字列、文字列のリスト、ブール値、パラメーター、組み込み関数によって返される値が使用可能

```
Resources:  
  MyEC2Instance: #論理ID  
    Type: "AWS::EC2::Instance" #リソースタイプ  
    Properties: #リソースごとのプロパティ  
      AvailabilityZone: "us-east-1a"  
      ImageId: "ami-0ff8a91507f77f867"  
      InstanceType: !Ref InstanceType  
      KeyName: !Ref KeyName
```



https://docs.aws.amazon.com/ja_jp/AWSCloudFormation/latest/UserGuide/resources-section-structure.html

© 2023, Amazon Web Services, Inc. or its affiliates.

49

論理IDと物理ID

- 論理ID
 - テンプレート内で一意
 - テンプレートの他の部分のリソースを参照するために使用
 - !Refや!GetAttで使用
- 物理ID
 - リソースに実際に割り当てられている名前（EC2のインスタンスID、S3バケット名など）
 - AWS CloudFormationテンプレート外のリソースを識別する場合に使用

```
Resources:  
  MyEC2Instance: #論理ID  
    Type: "AWS::EC2::Instance"  
    Properties:  
      SubnetId: "subnet-xxxxxxxxxxxxxx" #ここで指定しているのは物理ID  
Outputs:  
  MyEC2PhysicalID:  
    Value: !Ref MyEC2Instance # ! Ref を使って論理IDから物理IDを取得
```

出力 -		
キー	値	説明
MyEC2PhysicalID	i-0d6c1d5278786087a	

https://docs.aws.amazon.com/ja_jp/AWSCloudFormation/latest/UserGuide/resources-section-structure.html#resources-section-resource-fields



Mappings

- キーと値のマッピングテーブルを管理できる
 - リージョンやユーザ入力パラメータによって、値が変わるものに利用
 - Mappingsを利用することでテンプレートの再利用性が向上
 - 組み込み関数"Find::InMap"を使って値を取得
 - 例) "Fn::FindInMap" :
["MapName", "Key",
"Value"]
 - 引数の
MapName,Key,Valueには"!Ref"が利用可能

```
Mappings:  
RegionMap: #RegionMap という名称でMappings を定義  
    ap-northeast-1:  
        AMIID: ami-06cd52961ce9f0d85  
        KeyPair: My-Key-Tokyo  
    ap-northeast-3:  
        AMIID: ami-06cd52961ce9f0d85  
        KeyPair: My-Key-Osaka  
Resources:  
    myEC2Instance:  
        Type: "AWS::EC2::Instance"  
        Properties: #FindInMap 関数によってRegion に合致するAMIIDの値を取得する  
            ImageId: !FindInMap [RegionMap, !Ref "AWS::Region", AMIID]  
            InstanceType: m1.small
```

https://docs.aws.amazon.com/ja_jp/AWSCloudFormation/latest/UserGuide/mappings-section-structure.html



Conditions

- Resourcesセクションなどで、“ある条件が成立しているときのみリソースを作成”といった条件ベースの制御が可能
 - Conditions セクションに条件名と成立条件を列挙
 - Fn:: If、Fn:: Equals、Fn:: Notなどの組み込み関数を使用可
 - 条件が true となるリソースが作成され、false のリソースは無視される
 - スタックの更新時、リソースが更新される前にこれらの条件が再評価される
 - Conditions セクションだけを修正してスタックを更新することはできない

```
Parameters:  
  EnvType:  
    Description: "Environment type."  
    Default: "development"  
    Type: String  
    AllowedValues: ["production", "staging", "development"]  
    ConstraintDescription: "must specify."  
Conditions:  
  # EnvTypeの値が"production"であれば、CreateProdResources条件が成立  
  CreateProdResources: {"Fn::Equals" : [{"Ref" : "EnvType"}, "production"]}  
Resources:  
  Ec2Instance:  
    Type: "AWS::EC2::Instance"  
    # CreateProdResources条件が成立した場合、EC2リソースを作成  
    Condition: "CreateProdResources"
```

https://docs.aws.amazon.com/ja_jp/AWSCloudFormation/latest/UserGuide/conditions-section-structure.html

Transform

- ・スタックや変更セットの作成時に、テンプレートを変換するための事前定義済の CloudFormation マクロを呼び出す
 - ・CloudFormation マクロには、CloudFormation によって管理されているマクロ（ AWS::Serverless や AWS::Include など）とユーザが定義するマクロがある

AWS::Serverless の例：サーバーレスアプリケーションの場合、使用するAWS SAMのバージョンを指定

```
Transform: AWS::Serverless-2016-10-31
Resources:
  MyServerlessFunctionLogicalID:
```

AWS::Include の例：メインのテンプレートとは別のS3に保存されたテンプレートスニペットを指定

```
Transform:
  Name: 'AWS::Include'
  Parameters:
    Location: 's3://MyAmazonS3BucketName/MyFileName.yaml'
```

https://docs.aws.amazon.com/ja_jp/AWSCloudFormation/latest/UserGuide/transform-section-structure.html

Outputs

- ・ スタック構築後に取得・表示したい情報を定義できる
 - ・ アクセスURLや、DBの通信先情報、作成したIAMユーザー名など、あとで使用したい情報がある場合に便利
 - ・ 出力のエクスポート名を介して別のスタックからリソースを参照できるので、易い粒度でスタックを分割できる。（クロススタック参照）
 - ・ Outputs セクションにはパスワードなどの機密情報を書かないこと

Resources:
EC2WebServer01:
Type: AWS::EC2::Instance
Properties:
ImageId: !Ref EC2AMI
InstanceType: t2.micro
Outputs:
EC2WebServer01: #出力データの名称。キー。
Value: !Ref EC2WebServer01 #出力するデータ。値。
Export:
Name: !Sub \${AWS::StackName}-EC2WebServer01 #組み込み関数を使って文字列を加工

キー	▲ 値	▼ 説明	▼ エクスポート名
EC2WebServer01	i-0bad9055e937004e5	-	handson-cfn-ec2-EC2WebServer01

https://docs.aws.amazon.com/ja_jp/AWSCloudFormation/latest/UserGuide/outputs-section-structure.html

Rules

- スタックの作成または更新時にテンプレートに渡されるパラメータまたはパラメータの組み合わせを検証できる
 - ルールは、RuleCondition と、Assertions の2つのプロパティから成り、RuleCondition が true に評価された場合に、Assertions を評価してパラメータ値が有効かどうか確認する
 - Rules セクション 固有の組み込み関数が使用できる

[Fn::And](#)

[Fn::Contains](#)

[Fn::EachMemberEquals](#)

[Fn::EachMemberIn](#)

[Fn::Equals](#)

[Fn::If](#)

[Fn::Not](#)

[Fn::Or](#)

[Fn::RefAll](#)

[Fn::ValueOf](#)

[Fn::ValueOfAll](#)

```
Rules:  
  testInstanceType:  
    RuleCondition: !Equals  
      - !Ref Environment  
      - test  
    Assertions:  
      - Assert:  
        'Fn::Contains':  
          - - a1.medium  
          - !Ref InstanceType  
    AssertDescription:  
      'For a test environment, the instance type must be a1.medium'
```





AWS CloudFormation

2 基礎編

上原 優樹 (Uehara Yuki)

Cloud Support Engineer
2023/12

自己紹介

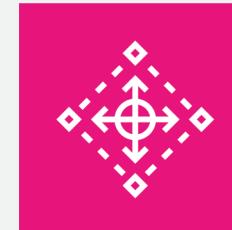
名前：上原 優樹 (Uehara Yuki)

所属：アマゾン ウェブ サービス ジャパン合同会社
技術支援本部 クラウドサポートエンジニア

好きなAWSサービス：



AWS CloudFormation



AWS Control Tower



本セミナーの対象者

想定聴講者

- CloudFormation をこれから利用される方、概要をお知りになりたい方

前提

- AWS の基本的な概要や操作を理解していること
- CloudFormation の基本的な用語 (スタック、テンプレート、変更セットなど) を理解していること

※ 次の Black Belt Online Seminar で解説しています

AWS CloudFormation #1 基礎編

資料 https://pages.awscloud.com/rs/112-TZM-766/images/AWS-Black-Belt_2023_CloudFormation-1_0731_v1.pdf

動画 <https://youtu.be/4dyiPsYXG8I>

ゴール

- CloudFormation を利用する上で必要な知識（依存関係、動的参照など）をご理解いただくこと
- CloudFormation の利用方法（ネストされたスタック、スタックセットなど）についてイメージを掴んでいただくこと

アジェンダ

1. リソースの依存関係

- DependsOn 属性による明示的な依存関係
- 組み込み関数による暗黙的な依存関係

2. 動的参照

3. ネストされたスタック

4. クロスアカウント参照

5. StackSets

※ 本資料では CloudFormation = CFn と略記することがあります

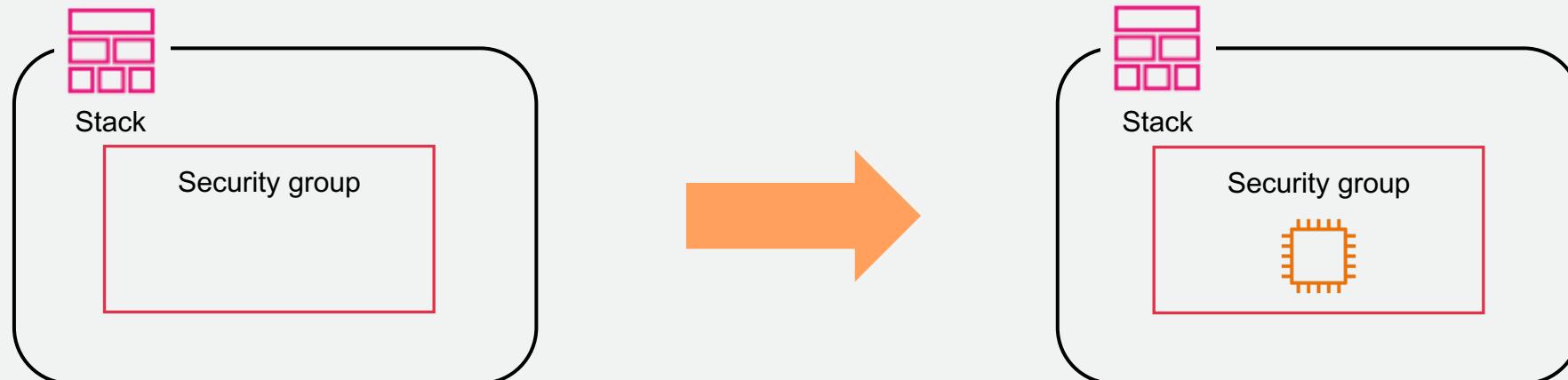


リソースの依存関係

リソースの依存関係

テンプレートに記述したリソースを作成する際、リソースの構築順は、テンプレートで定義したリソース間の**依存関係**から CloudFormation が自動的に決定する。

例：EC2 インスタンスがセキュリティグループを参照するように記述することで、最初にセキュリティグループが作成され、次に EC2 インスタンスが作成される。



テンプレートで定義したリソース間に依存関係がない場合、CloudFormation はリソースの作成を**並行**して開始する。

リソースの依存関係

1. DependsOn 属性による明示的な依存関係

DependsOn 属性を使用して依存関係を明示的に定義することで、リソースを決まった順序で処理することが可能。

DependsOn 属性が必須の場合の例

- Amazon VPC ゲートウェイのアタッチメント
- Amazon ECS サービスと Auto Scaling グループ
- AWS IAM ロールポリシー

https://docs.aws.amazon.com/ja_jp/AWSCloudFormation/latest/UserGuide/aws-attribute-dependson.html#gatewayattachment

2. 組み込み関数による暗黙的な依存関係

組み込み関数 (Fn::Ref、Fn::GetAtt、Fn::Sub) を利用し、プロパティの値として、別のリソースの属性値を参照すると、参照先のリソースが作成されてから対象のリソースを作成する。



<https://catalog.workshops.aws/cfn101/ja-JP/intermediate/templates/resource-dependencies>

© 2023, Amazon Web Services, Inc. or its affiliates.

DependsOn 属性による 明示的な依存関係

DependsOn 属性による明示的な依存関係

DependsOn 属性を使用することでリソースの依存関係を定義することが可能。

例：DependsOn 属性の値として Amazon S3 バケットリソースの論理 ID を使用することで、Amazon S3 バケットの作成が完了するのを待ってから、Amazon SNS トピックの作成を開始する。

```
Resources:  
  S3Bucket:  
    Type: AWS::S3::Bucket  
    Properties:  
      Tags:  
        - Key: Name  
        Value: Resource-dependencies-workshop  
  
  SNSTopic:  
    Type: AWS::SNS::Topic  
    DependsOn: S3Bucket  
    Properties:  
      Tags:  
        - Key: Name  
        Value: Resource-dependencies-workshop
```

論理 ID	ステータス
resource-dependencies-lab-dependson	✓ CREATE_COMPLETE
SNSTopic	✓ CREATE_COMPLETE
SNSTopic	ℹ CREATE_IN_PROGRESS
SNSTopic	ℹ CREATE_IN_PROGRESS
S3Bucket	✓ CREATE_COMPLETE
S3Bucket	ℹ CREATE_IN_PROGRESS
S3Bucket	ℹ CREATE_IN_PROGRESS
resource-dependencies-lab-dependson	ℹ CREATE_IN_PROGRESS

組み込み関数による 暗黙的な依存関係

組み込み関数による暗黙的な依存関係 (Fn::Ref)

Fn::Ref や Fn::GetAtt などの組み込み関数を使用して他のリソースを参照することで、**暗黙的な依存関係を定義することが可能**。

例：CloudFormation は SNSTopic リソースの作成が完了するのを待ってから、SNSTopicSubscription リソースの作成を開始する。

Resources:

```
SNSTopic:  
  Type: AWS::SNS::Topic  
Properties:
```

Tags:

- Key: Name
Value: Resource-dependencies-workshop

AWS::SNS::Topic は !Ref による参照で Topic の ARN を返す

SNSTopicSubscription:

```
Type: AWS::SNS::Subscription  
Properties:  
  Endpoint: !Ref EmailAddress  
  Protocol: email  
  TopicArn: !Ref SNSTopic
```

TopicArn プロパティには、サブスクライブする Topic の Amazon Resource Name (ARN) が必要



組み込み関数による暗黙的な依存関係 (Fn::Ref)

スタックを削除すると、CloudFormation は作成順と逆からリソースを削除する。

```
Resources:  
  SNSTopic:  
    Type: AWS::SNS::Topic  
    Properties:  
      Tags:  
        - Key: Name  
          Value: Resource-dependencies-workshop
```

```
SNSTopicSubscription:  
  Type: AWS::SNS::Subscription  
  Properties:  
    Endpoint: !Ref EmailAddress  
    Protocol: email  
    TopicArn: !Ref SNSTopic
```

論理 ID	ステータス
SNSTopic	ⓘ DELETE_IN_PROGRESS
SNSTopicSubscription	✔ DELETE_COMPLETE
SNSTopicSubscription	ⓘ DELETE_IN_PROGRESS
SNSTopicSubscription	ⓘ DELETE_IN_PROGRESS
resource-dependencies-with-intrinsic-functions	ⓘ DELETE_IN_PROGRESS

はじめに、最後に作成された
SNSTopicSubscription リソースを削除

組み込み関数による暗黙的な依存関係 (Fn::Ref)

スタックを削除すると、CloudFormation は作成順と逆からリソースを削除する。

```
Resources:  
  SNSTopic:  
    Type: AWS::SNS::Topic  
    Properties:  
      Tags:  
        - Key: Name  
          Value: Resource-dependencies-workshop
```

```
SNSTopicSubscription:  
  Type: AWS::SNS::Subscription  
  Properties:  
    Endpoint: !Ref EmailAddress  
    Protocol: email  
    TopicArn: !Ref SNSTopic
```

続いて最初に作成された SNSTopic リソースを削除

論理 ID	ステータス
resource-dependencies-with-intrinsic-functions	✓ DELETE_COMPLETE
SNSTopic	✓ DELETE_COMPLETE
SNSTopic	ℹ DELETE_IN_PROGRESS
SNSTopicSubscription	✓ DELETE_COMPLETE
SNSTopicSubscription	ℹ DELETE_IN_PROGRESS

組み込み関数による暗黙的な依存関係 (Fn::GetAtt)

例 : Fn::Ref 同様に参照される SecurityGroup リソースが CREATE_COMPLETE ステータスになるのを待ってから、SecurityGroupIngress の作成が開始される。

```
Resources:  
  SecurityGroup:  
    Type: AWS::EC2::SecurityGroup  
    Properties:  
      GroupDescription: Workshop Security Group  
      Tags:  
        - Key: Name  
        Value: Resource-dependencies-workshop
```

AWS::EC2::SecurityGroup は GroupID 属性を !GetAtt に渡して参照されると、セキュリティグループの ID を返す。

```
  SecurityGroupIngress:  
    Type: AWS::EC2::SecurityGroupIngress  
    Properties:  
      GroupId: !GetAtt SecurityGroup.GroupId  
      IpProtocol: tcp  
      FromPort: 80  
      ToPort: 80  
      CidrIp: 0.0.0.0/0
```

!GetAtt を使用することで、GroupId プロパティに SecurityGroup リソースの ID を指定可能。

組み込み関数による暗黙的な依存関係 (Fn::GetAtt)

スタックを削除すると、CloudFormation は作成順と逆からリソースを削除する。

```
Resources:  
  SecurityGroup:  
    Type: AWS::EC2::SecurityGroup  
    Properties:  
      GroupDescription: Workshop Security Group  
      Tags:  
        - Key: Name  
          Value: Resource-dependencies-workshop
```

```
  SecurityGroupIngress:  
    Type: AWS::EC2::SecurityGroupIngress  
    Properties:  
      GroupId: !GetAtt SecurityGroup.GroupId  
      IpProtocol: tcp  
      FromPort: 80  
      ToPort: 80  
      CidrIp: 0.0.0.0/0
```

論理 ID	ステータス
SecurityGroup	ⓘ DELETE_IN_PROGRESS
SecurityGroupIngress	✔ DELETE_COMPLETE
SecurityGroupIngress	ⓘ DELETE_IN_PROGRESS
sg-sample	ⓘ DELETE_IN_PROGRESS

はじめに、最後に作成された SecurityGroupIngress リソースを削除。

組み込み関数による暗黙的な依存関係 (Fn::GetAtt)

スタックを削除すると、CloudFormation は作成順と逆からリソースを削除する。

```
Resources:  
  SecurityGroup:  
    Type: AWS::EC2::SecurityGroup  
    Properties:  
      GroupDescription: Workshop Security Group  
      Tags:  
        - Key: Name  
        Value: Resource-dependencies-workshop
```

続いて最初に作成された
SecurityGroup リソースを削除。

```
  SecurityGroupIngress:  
    Type: AWS::EC2::SecurityGroupIngress  
    Properties:  
      GroupId: !GetAtt SecurityGroup.GroupId  
      IpProtocol: tcp  
      FromPort: 80  
      ToPort: 80  
      CidrIp: 0.0.0.0/0
```

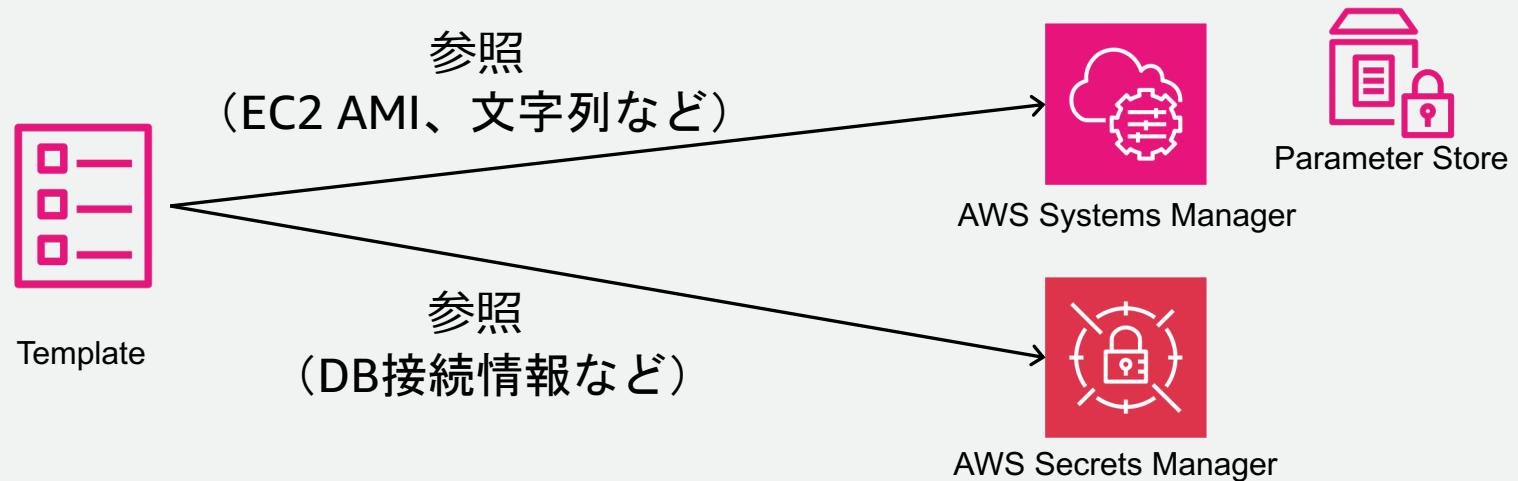
論理 ID	ステータス
sg-sample	✓ DELETE_COMPLETE
SecurityGroup	✓ DELETE_COMPLETE
SecurityGroup	ℹ DELETE_IN_PROGRESS
SecurityGroupIngress	✓ DELETE_COMPLETE
SecurityGroupIngress	ℹ DELETE_IN_PROGRESS

動的参照



動的参照

AWS Systems Manager (SSM) Parameter Store、AWS Secrets Manager を含む AWS サービスに保存されている外部値の参照が可能。



サービス	参照可能なデータ
AWS Systems Manager	<ul style="list-style-type: none">Parameter Store に格納されている String/StringList (平文で保存されているデータ)Parameter Store に格納されている SecureString (暗号化され保存されているデータ)
AWS Secrets Manager	<ul style="list-style-type: none">保存されているすべてのシークレットまたは特定のシークレット

動的参照 - SSM Parameter Store

SSM パラメータ

例：スタック操作時および変更セット操作時に SSM パラメータストアに格納されている S3AccessControl パラメータバージョン 2 の値を参照し S3 バケットのアクセス制御に設定。

MyS3Bucket:

Type: 'AWS::S3::Bucket'

Properties:

AccessControl: '{{resolve:ssm:S3AccessControl:2}}'

parameter-name version

考慮事項

- SSM にてパラメータを更新した場合、CloudFormation にパラメータの変更を反映させるために、スタックの更新にて動的な参照を含むリソースを更新する必要がある。
- バージョンを指定しない場合、AWS CloudFormation は、スタックを作成または更新するたびに最新バージョンのパラメータを使用する。
- 現時点では、クロスアカウント SSM パラメータアクセスをサポートしていない。
- 現時点では、ドリフト検出をサポートしていない。

https://docs.aws.amazon.com/ja_jp/AWSCloudFormation/latest/UserGuide/dynamic-references.html

© 2023, Amazon Web Services, Inc. or its affiliates.



動的参照 - SSM Parameter Store

SSM Secure String パラメータ

例：スタック操作時および変更セット操作時に SSM パラメータストアに格納されている安全な文字列であるバージョン 10 の値を IAM ユーザーのパスワードに設定。

MyIAMUser:

Type: AWS::IAM::User

Properties:

UserName: 'MyUserName'

LoginProfile:

Password: '{{resolve:ssm-secure:IAMUserPassword:10}}'

parameter-name version

考慮事項

- セキュアな方法で参照する必要がある機密データをパラメータとして利用する際に推奨。
- Secure String パラメータの値は保存されず、API コールの結果でも返されない。
- 現時点でサポートしているリソースプロパティに対してのみ使用可能。
- 変更セットでは安全な文字列に変換された値を比較し、実際の値の比較はしない。

https://docs.aws.amazon.com/ja_jp/AWSCloudFormation/latest/UserGuide/dynamic-references.html#dynamic-references-ssm-secure-strings

動的参照 - SSM Parameter Store

2023 年 11 月現在で動的なパラメータパターンをサポートするリソース

リソース	プロパティタイプ	プロパティ
AWS::DirectoryService::MicrosoftAD		Password
AWS::DirectoryService::SimpleAD		Password
AWS::ElastiCache::ReplicationGroup		AuthToken
<u>AWS::IAM::User</u>	LoginProfile	Password
AWS::KinesisFirehose::DeliveryStream	RedshiftDestinationConfiguration	Password
AWS::OpsWorks::App	ソース	Password
AWS::OpsWorks::Stack	CustomCookbooksSource	Password
AWS::OpsWorks::Stack	RdsDbInstances	DbPassword
AWS::RDS::DBCluster		MasterUserPassword
AWS::RDS::DBInstance		MasterUserPassword
AWS::Redshift::Cluster		MasterUserPassword

https://docs.aws.amazon.com/ja_jp/AWSCloudFormation/latest/UserGuide/dynamic-references.html#template-parameters-dynamic-patterns-resources

動的参照 - AWS Secrets Manager

リファレンスパターン

`{{resolve:secretsmanager:secret-id:secret-string:json-key:version-stage:version-id}}`

- secret-id : 必須
 - シークレット名またはシークレット ARN。
- secret-string : 必須
 - 現在サポートされている値は SecretString のみ。
- json-key
 - 値を取得するペアのキー名。指定しない場合、シークレットテキスト全体を取得する。
- version-stage
 - シークレットのバージョンのステージングラベル。
 - version-stage を利用する場合、version-id は指定できない。
- version-id
 - シークレットのバージョンの固有識別子を指定。
 - version-id を利用する場合、version-stage は指定できない。

※ version-stage、version-id を指定しない場合、デフォルトで AWSCURRENT というバージョンが指定される。

https://docs.aws.amazon.com/ja_jp/AWSCloudFormation/latest/UserGuide/dynamic-references.html#dynamic-references-secretsmanager

動的参照 - AWS Secrets Manager

例：Secrets Manager に格納されている、デフォルトのバージョンである AWSCURRENT のユーザー名とパスワードの値を参照し、それぞれのプロパティに設定。

```
MyRDSInstance:  
  Type: 'AWS::RDS::DBInstance'  
  Properties:  
    DBName: MyRDSInstance  
    AllocatedStorage: '20'  
    DBInstanceClass: db.t2.micro  
    Engine: mysql  
    MasterUsername: '{{resolve:secretsmanager:MyRDSSecret:SecretString:username}}'  
    MasterUserPassword: '{{resolve:secretsmanager:MyRDSSecret:SecretString:password}}'  
                           secret-id           json-key
```

考慮事項

- ・SSM Secure String と異なり、全てのリソースプロパティで使用可能。
- ・Secrets Manager でシークレットを更新しても、CloudFormation のシークレットは更新されないため、動的な参照を含むリソースを更新するスタック更新を実行する必要がある。



https://docs.aws.amazon.com/ja_jp/AWSCloudFormation/latest/UserGuide/dynamic-references.html#dynamic-references-secretsmanager

ネストされたスタック

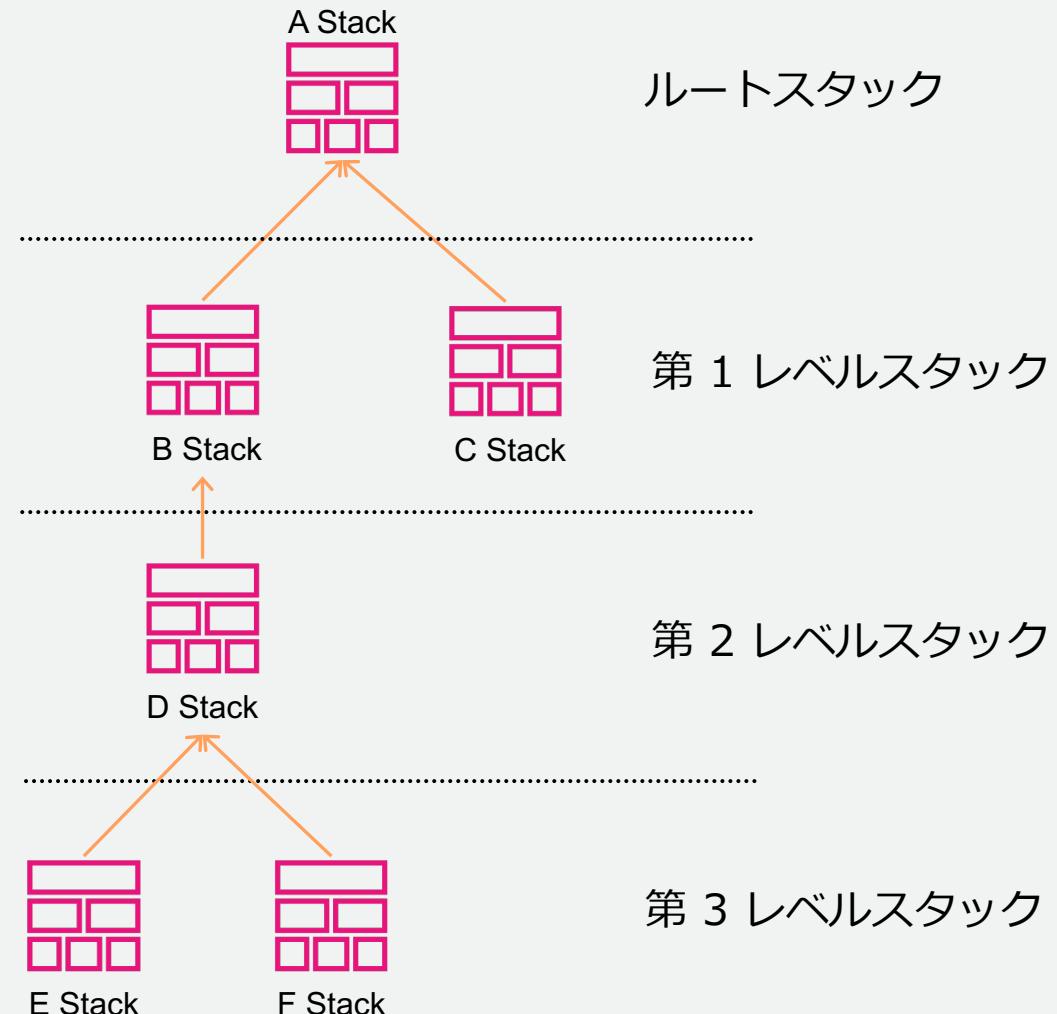
ネストされたスタック

- 大きなテンプレートを小さな専用のテンプレートに分け、**参照**することが可能。
- ネストされたスタックに対しても他のスタックをネストすることが可能。
- ルートスタックは、ネストされたすべてのスタックが最終的に属する最上位スタック。
- ネストされたスタックにはそれぞれ、直接の親スタックが存在する。

[図解]

- スタック B からみると、スタック A は親スタックであると同時にルートスタックでもある。
- スタック E からみると、スタック D が親スタック。
- スタック D からみると、スタック B が親スタック。

→ 親スタック



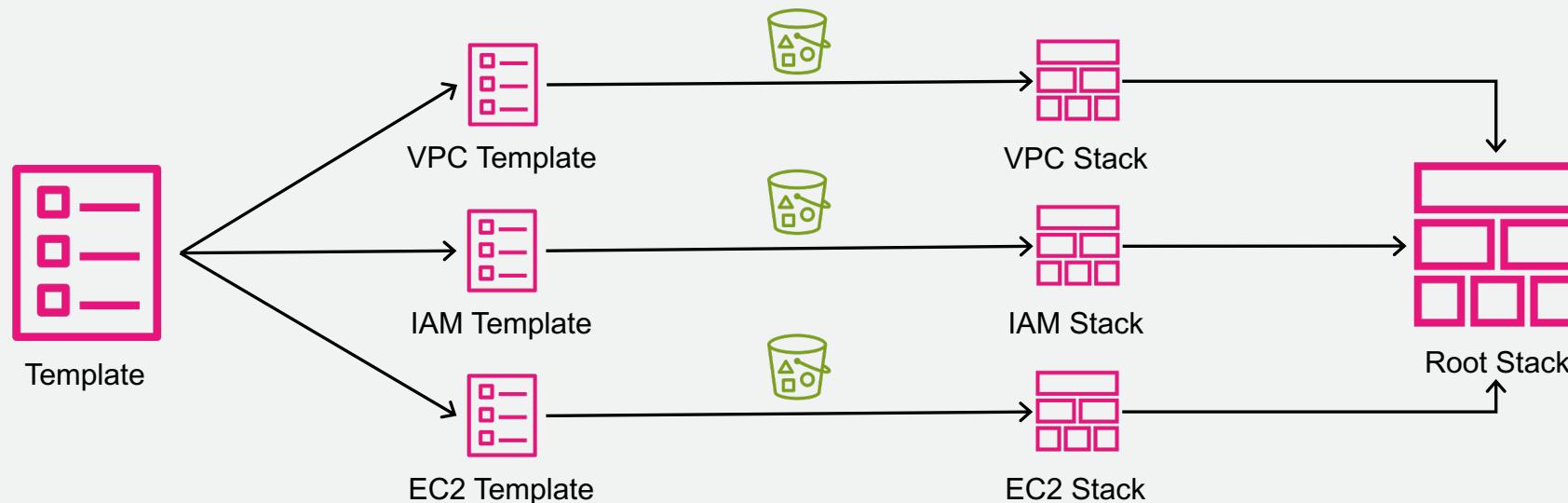
ネストされたスタック

ユースケース

- 複数のスタックに使用しているリソースの構成がある場合、テンプレートに同じ構成をコピーアンドペーストする代わりに、専用のテンプレートを再利用することが可能。
- 大きなテンプレートを小さなテンプレートに分解することでリソース制限を回避する。
- 変更セットによるリソースの変更レビューが可能。

留意点

- ネストされたスタックのテンプレートを S3 バケットに予め保存する必要がある。



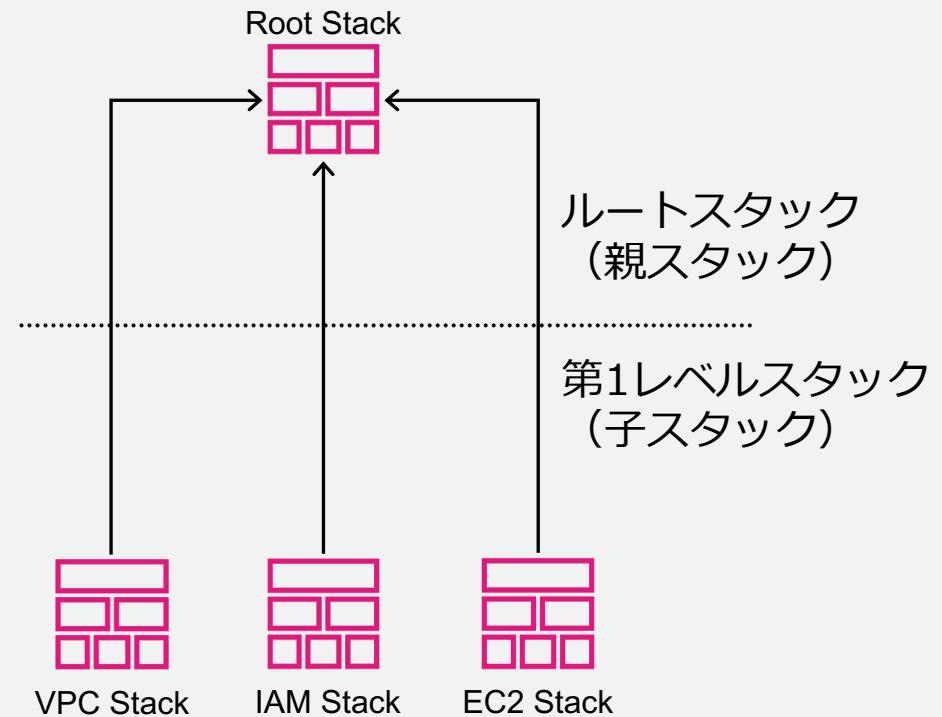
https://docs.aws.amazon.com/ja_jp/AWSCloudFormation/latest/UserGuide/using-cfn-nested-stacks.html

© 2023, Amazon Web Services, Inc. or its affiliates.

ネストされたスタック（親スタック）

AWS::CloudFormation::Stack リソースタイプにて子スタックを定義する。

```
EC2Stack:  
  Type: AWS::CloudFormation::Stack  
  Properties:  
    TemplateURL: !Sub https://${S3BucketName}.s3.amazonaws.com/ec2.yaml  
    TimeoutInMinutes: 20  
  Parameters:  
    EnvironmentType: !Ref EnvironmentType  
    VpcId: !GetAtt VpcStack.Outputs.VpcId  
    SubnetId: !GetAtt VpcStack.Outputs.PublicSubnet1
```

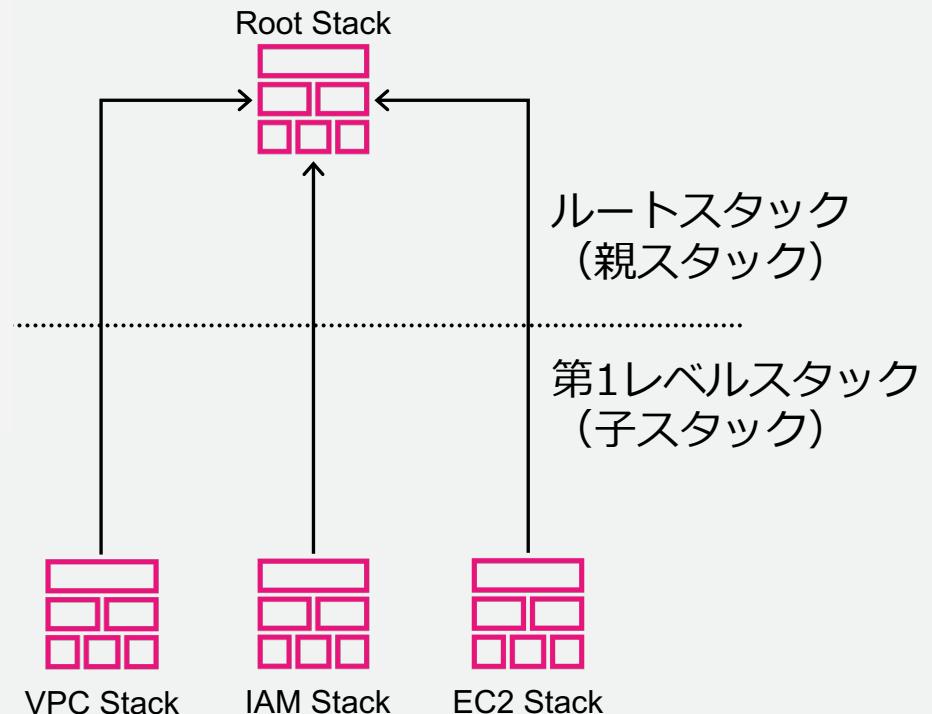


ネストされたスタック（親スタック）

TemplateURL プロパティにてネストされたスタックのテンプレートを指定する。

EC2Stack:

```
Type: AWS::CloudFormation::Stack
Properties:
  TemplateURL: !Sub https://${S3BucketName}.s3.amazonaws.com/ec2.yaml
  TimeoutInMinutes: 20
Parameters:
  EnvironmentType: !Ref EnvironmentType
  VpcId: !GetAtt VpcStack.Outputs.VpcId
  SubnetId: !GetAtt VpcStack.Outputs.PublicSubnet1
```



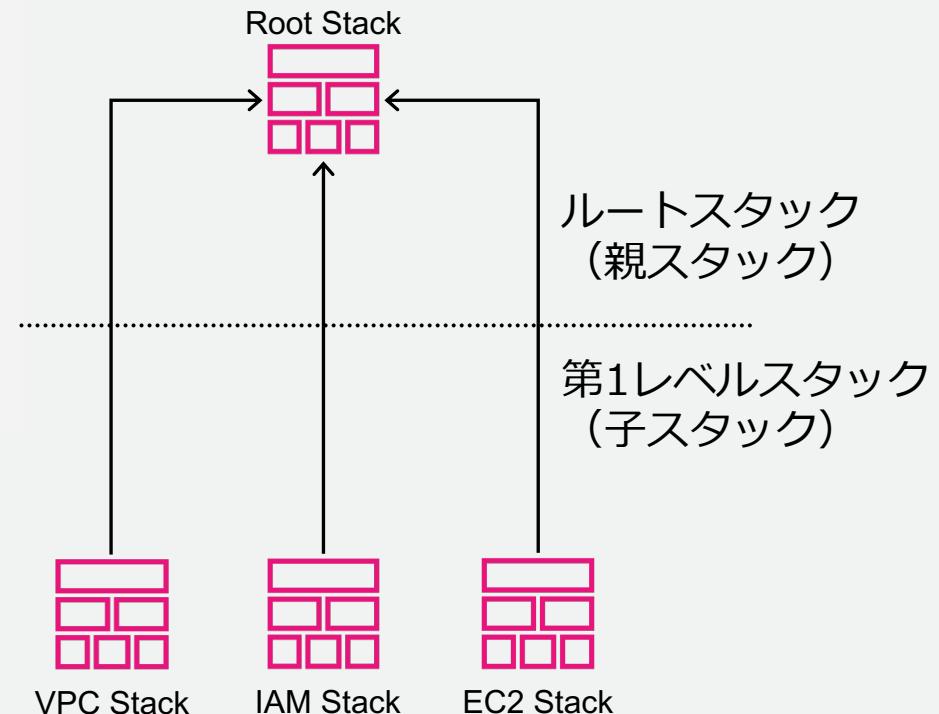
ネストされたスタック（親スタック）

Parameters プロパティにてネストされたテンプレートにパラメータを渡すことが可能。

EC2Stack:

```
Type: AWS::CloudFormation::Stack
Properties:
  TemplateURL: !Sub https://${S3BucketName}.s3.amazonaws.com/ec2.yaml
  TimeoutInMinutes: 20
Parameters:
  EnvironmentType: !Ref EnvironmentType
  VpcId: !GetAtt VpcStack.Outputs.VpcId
  SubnetId: !GetAtt VpcStack.Outputs.PublicSubnet1
```

※ 組み込み関数 Fn::GetAtt (!GetAtt) を利用することで、他のネストされたテンプレートの Output セクションで出力された値をパラメータとして利用可能。



クロススタック参照

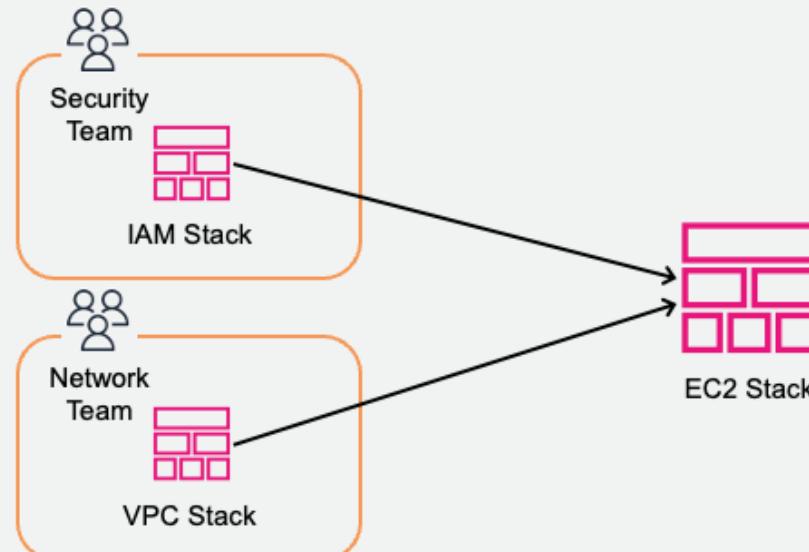
クロススタック参照

ユースケース

- あるスタックでデプロイしたリソースを、他のスタックから参照したい。
- 役割と責任を分けるために、スタックを分けて管理したい。

留意点

- スタックを分けすぎて管理が煩雑になる場合がある。
- 別スタックによって参照されている出力値を変更、削除することができない。
- スタックの出力値が参照されている場合、当該スタックを削除することができない。



https://docs.aws.amazon.com/ja_jp/AWSCloudFormation/latest/UserGuide/walkthrough-crossstackref.html

クロススタック参照

使い方: 参照されるスタック (IAM Stack)

- Output セクションの Export フィールドに出力したい値を指定する。

WebServerInstanceProfile:

Type: AWS::IAM::InstanceProfile

Properties:

Path: /

Roles:

- !Ref SSMIAMRole

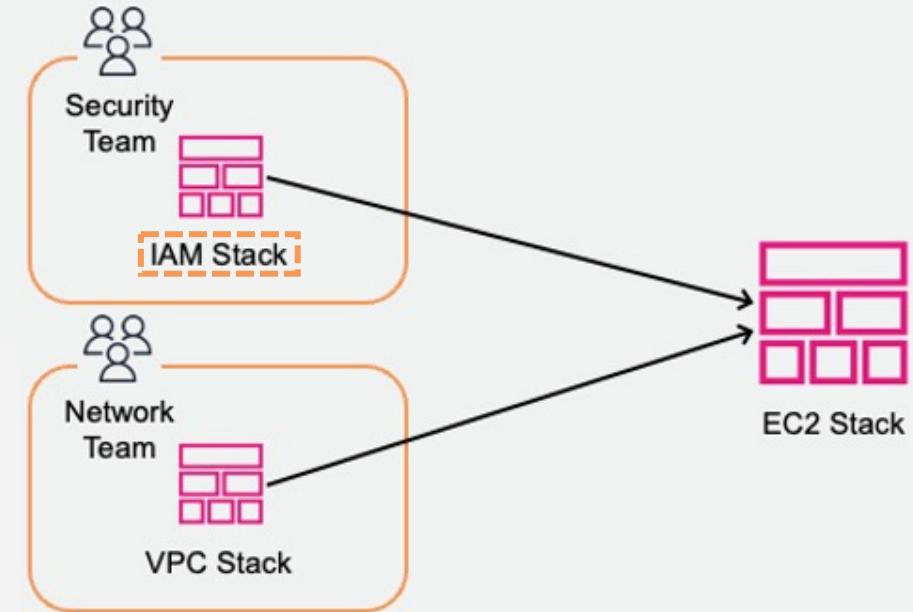
Outputs:

WebServerInstanceProfile:

Value: !Ref WebServerInstanceProfile

Export:

Name: cfn-workshop-WebServerInstanceProfile



クロススタック参照

使い方: 参照するスタック (EC2 Stack)

- 組み込み関数 Fn::ImportValue にて取得したい値を指定する。

```
WebServerInstance:
```

```
  Type: AWS::EC2::Instance
```

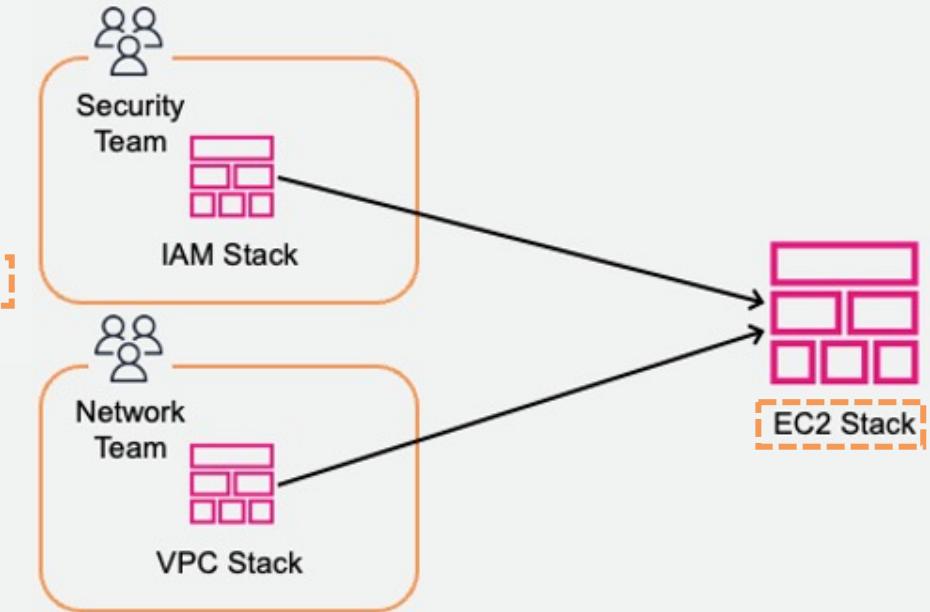
```
  {...}
```

```
Properties:
```

```
  SubnetId: !ImportValue cfn-workshop-PublicSubnet1
```

```
  IamInstanceProfile: !ImportValue cfn-workshop-WebServerInstanceProfile
```

```
  ImageId: !Ref AmiID
```



https://docs.aws.amazon.com/ja_jp/AWSCloudFormation/latest/UserGuide/walkthrough-crossstackref.html

© 2023, Amazon Web Services, Inc. or its affiliates.

ネストされたスタックとクロススタック参照の違い

ネストされたスタック

- ネストされたスタックグループ内でのみ情報共有をしたい場合に推奨。
 - 複数の子スタックのリソースを親スタックによりすべてデプロイし、管理する。
 - 子スタックのテンプレートを S3 バケットに保存する必要がある。
 - AWS::CloudFormation::Stack リソースタイプを使用することで、子スタックからテンプレートを参照可能。 (*)
 - TemplateURL プロパティにてネストされたスタックのテンプレートを指定。 (*)
 - Parameters プロパティにてネストされたテンプレートにパラメータを渡す。 (*)
- (*: 親スタック)

クロススタック参照

- ネストされたスタックグループ内に限らず、他のスタックと情報を共有したい場合に推奨。
- 他のスタックによって管理されているリソースをインポートして利用する。
- Output セクションの Export フィールドに出力したい値を指定。
- 組み込み関数 Fn::ImportValue にて取得したい値を指定。

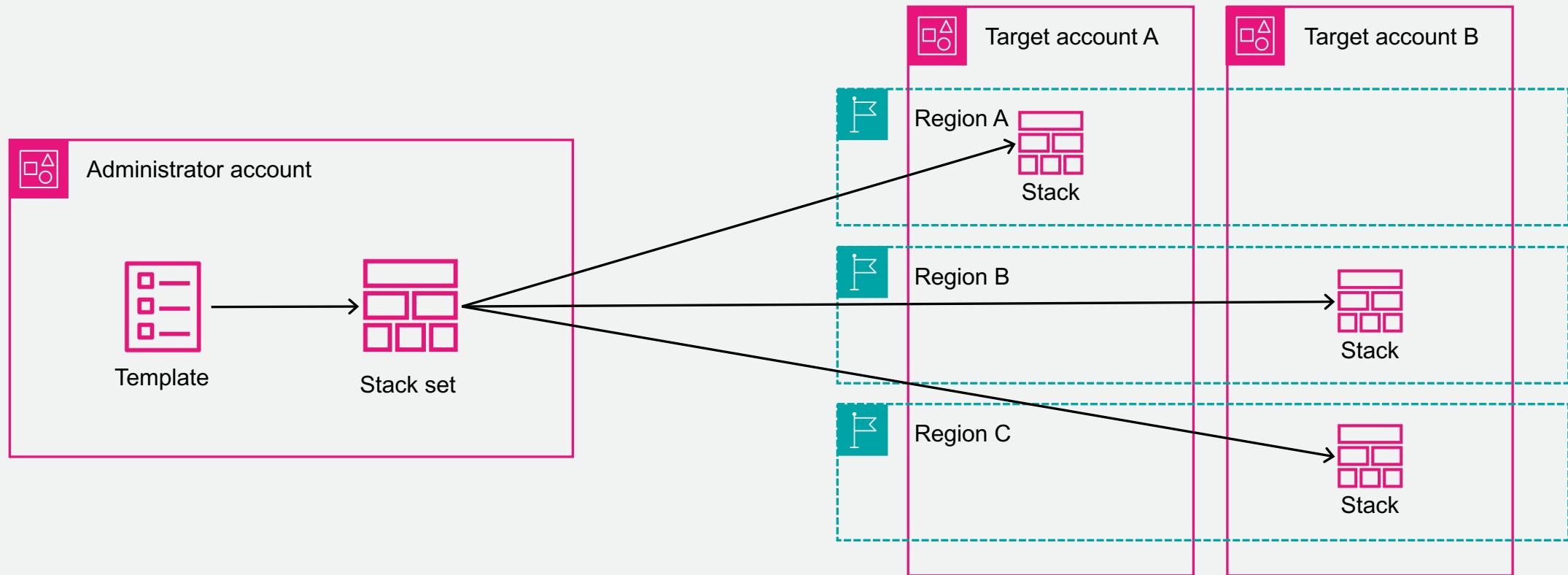


StackSets



概要

1つのテンプレートを使用して複数の AWS アカウント（ターゲットアカウント） 、複数のリージョンにスタックを作成、更新、削除することが可能。



https://docs.aws.amazon.com/ja_jp/AWSCloudFormation/latest/UserGuide/what-is-cfnstacksets.html

© 2023, Amazon Web Services, Inc. or its affiliates.

用語

- ・管理者アカウント

- スタックセットを作成する AWS アカウント

- ・ターゲットアカウント

- スタックセットの 1 つ以上のスタックを作成、更新、削除する AWS アカウント

- ・スタックセット

- スタックの作成に使用するテンプレートおよびパラメータ、スタックを作成するターゲットアカウント、デプロイするリージョンなどの定義

- ・スタックインスタンス

- ターゲットアカウントのスタックへのリファレンス
 - スタックインスタンスはスタックなしで存在可能
(何らかの理由によりスタックが作成されていない場合、スタックインスタンスに失敗理由が表示される)
 - スタックインスタンスは 1 つのスタックセットにのみ関連付けられる



オペレーション

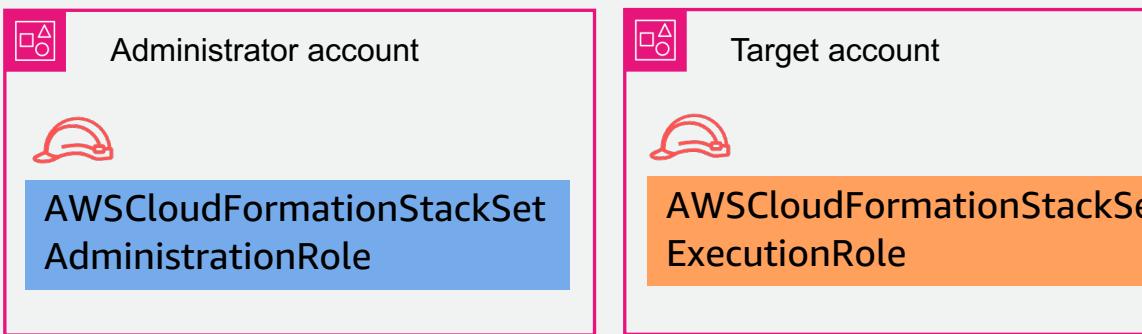
- ・ **アクセス許可モデルの選択と設定 (初回のみ)**
 - スタックセットはセルフマネージド型またはサービスマネージド型のアクセス許可が必要。
- ・ **スタックセットの作成**
 - スタックの作成に使用するテンプレート、ターゲットアカウント、デプロイする AWS リージョンの指定を行い作成。
- ・ **スタックセットの更新**
 - スタックセットを更新すると、スタックセットのスタックに変更内容がプッシュされる。
- ・ **スタックインスタンスのパラメータの上書き**
 - アカウントおよびリージョン別にスタックインスタンスのパラメータ値を上書き可能。
- ・ **スタックの削除**
 - 指定したリージョン内の指定したターゲットアカウントから、スタックを削除する。
(スタックの削除に併せて対象のスタックインスタンスも削除される。)
- ・ **スタックセットの削除**
 - スタックセット内にスタックインスタンスが存在しない場合のみ、削除可能。

https://docs.aws.amazon.com/ja_jp/AWSCloudFormation/latest/UserGuide/stacksets-concepts.html#stacksets-concepts-ops



アクセス許可モデルの選択と設定 (初回のみ)

セルフマネージド



管理アカウントとターゲットアカウントの双方に指定された名称・内容で IAM ロールを作成する必要がある。
IAM ロール作成用 CloudFormation テンプレート配布されている。

https://docs.aws.amazon.com/ja_jp/AWSCloudFormation/latest/UserGuide/stacksets-prereqs-self-managed.html

サービススマネージド

This screenshot shows the 'CloudFormation StackSets' section within the AWS Organizations console. It displays the status of trusted access for the service. A note at the bottom explains that enabling trusted access for CloudFormation StackSets makes it a trusted service within the organization.

AWS Organizations > サービス > CloudFormation StackSets

CloudFormation StackSets

コンソールに移動する

信頼されたアクセス

信頼されたアクセスが有効

ステータス

信頼されたアクセスを無効にする

信頼されたアクセスを有効にすると、CloudFormation StackSets が組織内の信頼されたサービスとして指定されます。信頼されたサービスは、組織の構造をクエリし、組織のアカウントにサービスにリンクされたロールを作成できます。サービスにリンクされたロールにより、信頼されたサービスは、信頼されたサービスのドキュメントに記載されているタスクを実行できます。信頼されたサービスは組織への変更についてに通知を受け、これらの通知に応じて追加のタスクを実行できます。

AWS Organizations で StackSets の信頼されたアクセスを有効化することで、管理アカウントとターゲットアカウントの双方に必要な IAM ロールが自動的に作成される。

https://docs.aws.amazon.com/ja_jp/AWSCloudFormation/latest/UserGuide/stacksets-orgs-activate-trusted-access.html

オペレーション

- ・ **アクセス許可モデルの選択と設定 (初回のみ)**
 - スタックセットはセルフマネージド型またはサービスマネージド型のアクセス許可が必要。
- ・ **スタックセットの作成**
 - スタックの作成に使用するテンプレート、ターゲットアカウント、デプロイする AWS リージョンの指定を行い作成。
- ・ **スタックセットの更新**
 - スタックセットを更新すると、スタックセットのスタックに変更内容がプッシュされる。
- ・ **スタックインスタンスのパラメータの上書き**
 - アカウントおよびリージョン別にスタックインスタンスのパラメータ値を上書き可能。
- ・ **スタックの削除**
 - 指定したリージョン内の指定したターゲットアカウントから、スタックを削除する。
(スタックの削除に併せて対象のスタックインスタンスも削除される。)
- ・ **スタックセットの削除**
 - スタックセット内にスタックインスタンスが存在しない場合のみ、削除可能。

https://docs.aws.amazon.com/ja_jp/AWSCloudFormation/latest/UserGuide/stacksets-concepts.html#stacksets-concepts-ops



StackSet の作成 1/2

1. テンプレートの選択

アクセス許可を指定し、自分で用意したテンプレートかサンプルテンプレートを指定。

アクセス許可

IAM ロールを選択して、CloudFormation がターゲットアカウントを管理する方法を明示的に定義します。 [詳細はこちら](#)

サービスマネージドアクセス許可
StackSets は、AWS Organizations が管理するターゲットアカウントにデプロイするために必要なアクセス許可を自動的に設定します。このオプションを使用すると、組織内のアカウントへの自動デプロイを有効にできます。

セルフサービスのアクセス許可
ターゲットアカウントにデプロイするために必要な IAM ロールを作成します。ロールを選択しない場合、CloudFormation は現在のユーザーの認証情報に基づくアクセス許可を使用します。

前提条件 - テンプレートの準備

テンプレートの準備
各スタックはテンプレートに基づきます。テンプレートとは、スタックに含む AWS リソースに関する設定情報を含む JSON または YAML ファイルです。

テンプレートの準備完了
 サンプルテンプレートを使用

テンプレートの指定

テンプレートは、スタックのリソースおよびプロパティを表す JSON または YAML ファイルです。

テンプレートソース
テンプレートを選択すると、保存先となる Amazon S3 URL が生成されます。

Amazon S3 URL
 テンプレートファイルのアップロード

Amazon S3 URL
`https://`

Amazon S3 テンプレートの URL

S3 URL: URL を指定すると生成されます。

デザイナーで表示

2. StackSet の詳細を指定

StackSet 名称や説明、テンプレートのパラメータを指定。

StackSet 名

StackSet 名

StackSet の説明

StackSet の説明

パラメータ

パラメータは、テンプレートで定義されます。また、パラメータを使用すると、スタックを作成または更新する際にカスタム値を入力できます。

SampleParameter

String を入力



StackSet の作成 2/2

3. StackSet オプションの設定

タグや実行設定を指定。

タグ

Stack のリソースに適用するタグ (キーと値のペア) を指定できます。Stack ごとに一意のタグを 50 個まで追加できます。

Stack に関するタグはありません。

新しいタグの追加

さらに 50 のタグを追加できます

実行設定

マネージド型の実行

StackSets が競合しないオペレーションを並行して実行し、競合するオペレーションはキューに入れるかどうかを指定します。

非アクティブ
StackSets は、一度に 1 つのオペレーションを実行します。

アクティブ
StackSets は、競合しないオペレーションを並行して実行し、競合するオペレーションをキューに入れます。競合するオペレーションが終了すると、StackSets はリクエスト順にキューに入れられたオペレーションを開始します。

4. デプロイオプションの設定

デプロイ先や自動デプロイオプション、同時実行するアカウントやリージョン等を指定。

デプロイターゲット

StackSets は、ターゲット組織または組織単位 (OU) のすべてのアカウントにStackインスタンスをデプロイします。親 OU をターゲットとして追加すると、StackSets はターゲットとして子 OU も追加します。[詳細はこちら](#)

組織へのデプロイ

組織単位 (OU) へのデプロイ

自動デプロイオプション

自動デプロイ
自動デプロイが有効になっている場合、アカウントが OU に追加されると、StackSets は自動的に追加のStackインスタンスをこのアカウントにデプロイします。アカウントが OU から削除されると、StackSets はこのアカウントのStackインスタンスを自動的に削除します。

有効

無効

アカウント削除の動作
ターゲット OU からアカウントを削除する場合、アカウント内のStackインスタンスを削除または保持する必要がありますか？

Stack を削除

Stack を保持

リージョンの指定

Stack をデプロイするリージョンを選択します。Stack は、指定した順序でこれらのリージョンにデプロイされます。StackSet の操作中に、管理者アカウントとターゲットアカウントは、アカウント自体、ならびに関連するStackSet およびStackインスタンスに関するメタデータを交換することに注意してください。[詳細はこちら](#)

すべてのリージョンを追加

すべてのリージョンを削除

デプロイオプション

同時にアカウントの最大数 - オプション
Stack を同時にデプロイできるリージョン別のアカウント数。数値が大きいほど、オペレーションが高速になります。

数値

障害耐性 - オプション
Stack が失敗するリージョン別のアカウント数。この数を超えると、このリージョンでのオペレーションが CloudFormation で停止されます。1 つのリージョンで停止されたオペレーションは、その他のリージョンでも続行されなくなります。数値が小さいほど、オペレーションの安全性が高くなります。

数値

リージョンの同時実行
選択して、StackSets をリージョンに複数デプロイするか、並行してデプロイします。

順次
リージョンのデプロイ順序を指定し、複数の StackSets オペレーションを同時に 1 つのリージョンにデプロイします。

並行
指定したすべてのリージョンに複数の StackSets オペレーションを並行してデプロイします。

同時に実行モード
オペレーションの実行中の同時実行レベルの動作を指定します。

厳格な障害耐性
同時に実行レベルを動的に下げて、失敗したアカウントの数が障害耐性 + 1 を超えないようにします。

ソフト障害耐性
常に指定された同時に実行レベルで実行します。

https://docs.aws.amazon.com/ja_jp/AWSCloudFormation/latest/UserGuide/stacksets-getting-started-create.html

© 2023, Amazon Web Services, Inc. or its affiliates.



自動デプロイオプション

組織単位 (OU) ヘアカウントを追加・削除した際のスタックインスタンスの挙動を設定する機能。

自動デプロイ

自動デプロイが有効になっている場合、アカウントが OU に追加されると、StackSets は自動的に追加のスタックインスタンスをこのアカウントにデプロイします。アカウントが OU から削除されると、StackSets はこのアカウントのスタックインスタンスを自動的に削除します。

有効

無効

アカウント削除の動作

ターゲット OU からアカウントを削除する場合、アカウント内のスタックインスタンスを削除または保持する必要がありますか?

スタックを削除

スタックを保持

※サービススマネージドのアクセス許可モデルの場合に設定することが可能。

※上書きされたパラメータ値[p.55、56]は、設定時に指定されたアカウントにのみ適用され、自動デプロイオプションにて今後追加されるアカウントには**適用されない**。

※自動デプロイ設定は、スタックセット作成後いつでも調整可能。



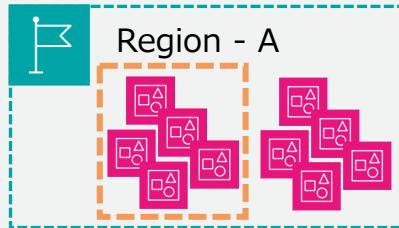
デプロイオプション

同時アカウントの最大数

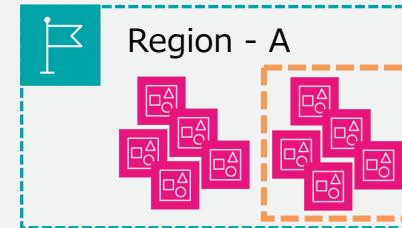
スタックセットの作成、更新、削除に適用され、リージョン別に一度にオペレーションを実行するターゲットアカウントの最大数または割合(%)を指定することが可能。

例：割合(%)：50で2つのリージョンの10個のターゲットアカウントにデプロイする場合

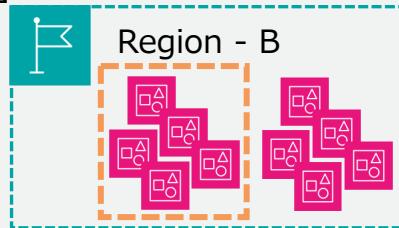
[1]. リージョン A で 5 アカウント



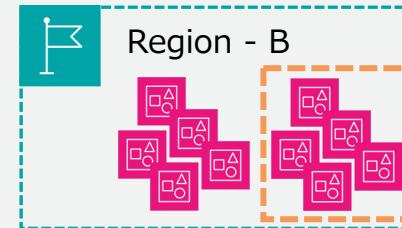
[2]. リージョン A で他の 5 アカウント



[3]. リージョン B で 5 アカウント



[4]. リージョン B で他の 5 アカウント



※指定された割合(%)が指定したアカウントの整数にならない場合は、丸められる。

例：ターゲットアカウント 10 割合(%)：25 = ターゲットアカウントの最大数 2

https://docs.aws.amazon.com/ja_jp/AWSCloudFormation/latest/UserGuide/stacksets-concepts.html#stackset-ops-options

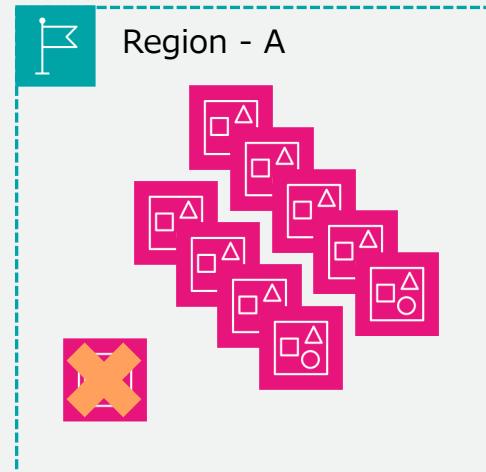
デプロイオプション

障害耐性

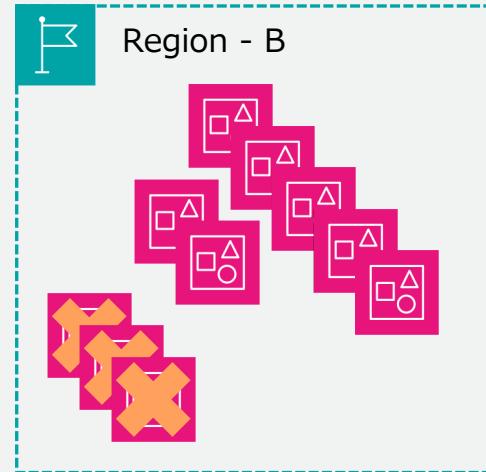
スタックセットの作成、更新、削除時に、各リージョンで発生する場合があるスタックオペレーションの失敗の最大数または割合(%)を超えるとオペレーションが自動停止する。

例：割合(%)：20で3つのリージョンの10個のターゲットアカウントにデプロイ

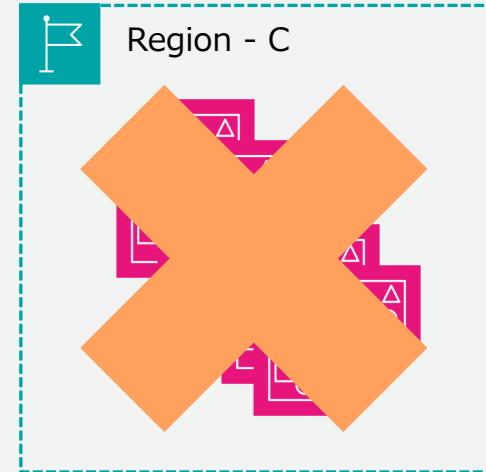
[1]. デプロイに1つ失敗



[2]. デプロイに3つ失敗



[3]. 更新オペレーション停止



※指定された割合(%)が指定したアカウントの整数にならない場合は、丸められる。

例：ターゲットアカウント10 割合(%)：25 = ターゲットアカウントの最大数 2

https://docs.aws.amazon.com/ja_jp/AWSCloudFormation/latest/UserGuide/stacksets-concepts.html#stackset-ops-options

デプロイオプション

同時実行モード

スタックセット操作中の同時実行レベルの動作を選択できるパラメータ。

厳密な障害耐性（デフォルト）

失敗したアカウントの数が**障害耐性 +1** の値を超えないように、同時実行レベルを動的に下げる。つまり、デプロイに失敗した場合、実際の同時実行数は失敗したデプロイの数に比例して減少する。そのため、デプロイが失敗するたびに同時実行性が低下するのでデプロイ速度は低下する。なお実際の同時実行数の初期値は、**同時アカウントの最大数**の値または**障害耐性 +1** の値のいずれか低い方に設定される。

ソフト障害耐性

障害耐性と実際の同時実行性を切り離し、デプロイに失敗しても同時実行レベルは一定。これにより、障害の数に関係なくスタックセットの操作を**同時アカウントの最大数**の値で設定された同時実行レベルで実行できる。

しかし、実際の障害の数を考慮せず同時にデプロイを実施するため、障害耐性で設定した数よりも、**デプロイに失敗したスタックインスタンスが多くなる可能性がある**ことに注意。

そのため、厳密な障害耐性よりもデプロイ速度を優先したい場合に推奨。



デプロイオプション

同時実行モード：厳密な障害耐性（デフォルト）

[例] 障害耐性 : 5 同時アカウントの最大数 : 10

実際の同時実行数（障害耐性 + 1）: 6

（障害耐性の値 5 + 1 が同時アカウントの最大数の値よりも低いため。）



= 実際の同時実行数



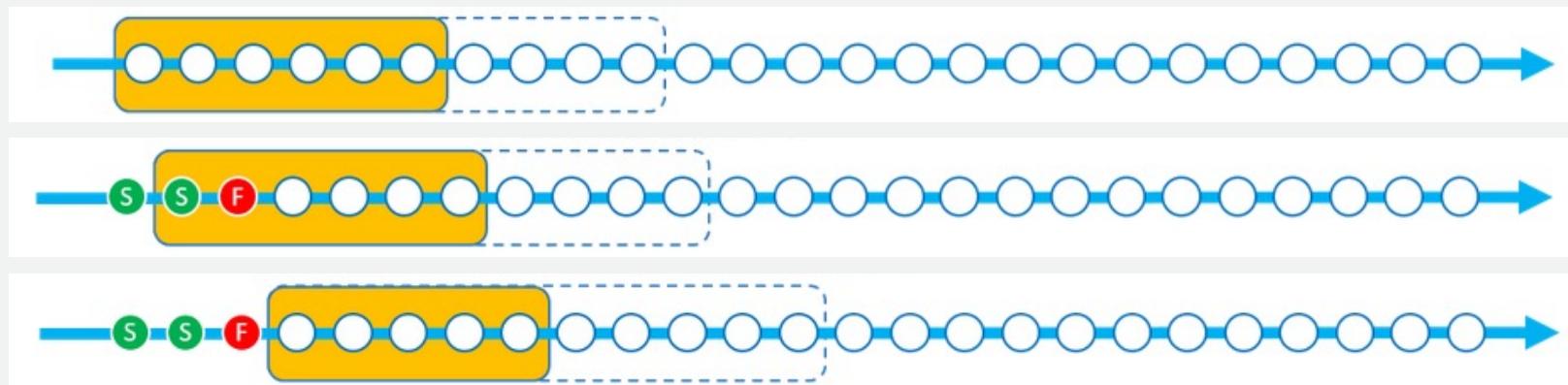
= 同時アカウントの最大数

○ = スタックセット

● = 成功したスタックセット

● = 失敗したスタックセット

(1) スタックセットが 1 つのデプロイに失敗すると、実際の同時実行数は 6 から 5 に減少する。



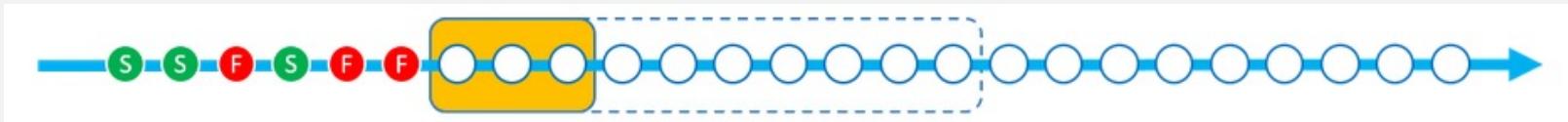
https://docs.aws.amazon.com/ja_jp/AWSCloudFormation/latest/UserGuide/concurrency-mode.html

© 2023, Amazon Web Services, Inc. or its affiliates.

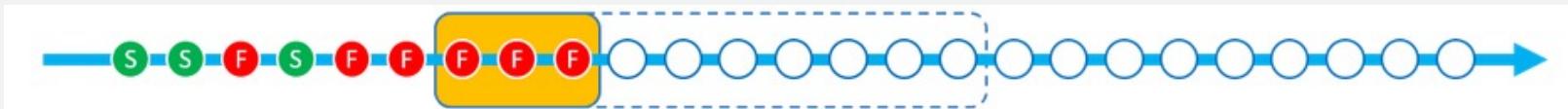
デプロイオプション

同時実行モード：厳密な障害耐性（デフォルト）

- (2) スタックセットがさらに 2 つのスタックインスタンスのデプロイに失敗すると、実際の同時実行数は 5 から 3 に減少し、失敗したスタックインスタンスの合計は 3 となる。



- (3) その後、スタックセットが 3 つのスタックインスタンスのデプロイに失敗すると、失敗したスタックインスタンスの合計は 6 となる。
その場合、失敗したスタックインスタンスの数が**障害耐性 +1** の値に等しくなるため、スタックセットは操作に失敗する。



今回の例では、スタックセットは操作を停止する前に 9 つのスタックインスタンス（3 つは成功、6 つは失敗）をデプロイを実施した。

デプロイオプション

同時実行モード：ソフト障害耐性

[例] 障害耐性 : 5 同時アカウントの最大数 : 10

実際の同時実行数 : 10



= 実際の同時実行数



= 同時アカウントの最大数

○ = スタックセット

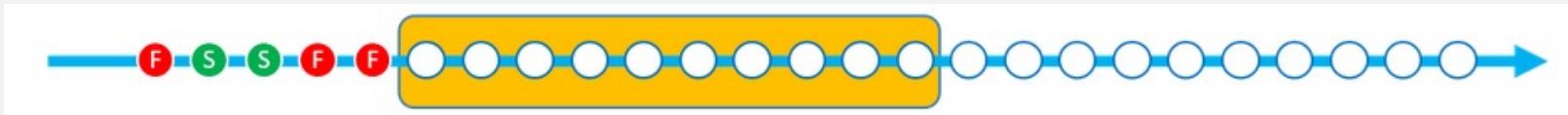
● = 成功したスタックセット

● = 失敗したスタックセット

(1) スタック操作が 1 つ失敗しても、実際の同時実行数は 10 のまま変わらない。



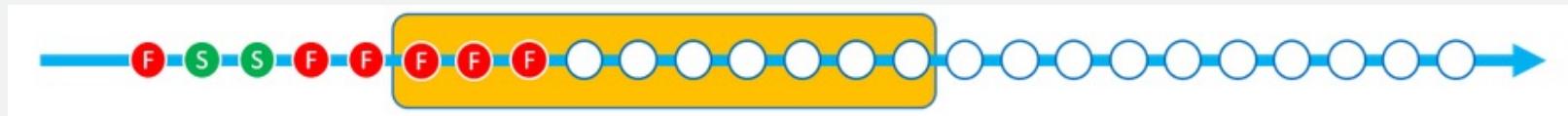
(2) さらに、スタックインスタンスに 2 回障害が発生しても、実際の同時実行数は 10 のまま変わらない。



デプロイオプション

同時実行モード：ソフト障害耐性

- (3) 6 つのスタック操作に失敗し、**障害耐性**の値である 5 に達したため、スタックセットは操作に失敗する。ただし、**同時実行キューの残りの操作が終了するまでスタックセットの操作は終了しない。**



- (4) スタックセットの操作が**障害耐性**の 5 に達したにもかかわらず、同時実行キューに実行すべき操作が 7 つ残っていたため、合計 8 つのスタックインスタンスが失敗する。



この例では、スタックセットは操作を停止する前に 15 個のスタックインスタンス (7 つは成功、8 つは失敗) をデプロイした。

オペレーション

- ・ **アクセス許可モデルの選択と設定 (初回のみ)**
 - スタックセットはセルフマネージド型またはサービスマネージド型のアクセス許可が必要。
- ・ **スタックセットの作成**
 - スタックの作成に使用するテンプレート、ターゲットアカウント、デプロイする AWS リージョンの指定を行い作成。
- ・ **スタックセットの更新**
 - スタックセットを更新すると、スタックセットのスタックに変更内容がプッシュされる。
- ・ **スタックインスタンスのパラメータの上書き**
 - アカウントおよびリージョン別にスタックインスタンスのパラメータ値を上書き可能。
- ・ **スタックの削除**
 - 指定したリージョン内の指定したターゲットアカウントから、スタックを削除する。
(スタックの削除に併せて対象のスタックインスタンスも削除される。)
- ・ **スタックセットの削除**
 - スタックセット内にスタックインスタンスが存在しない場合のみ、削除可能。

https://docs.aws.amazon.com/ja_jp/AWSCloudFormation/latest/UserGuide/stacksets-concepts.html#stacksets-concepts-ops



StackSet の更新 1/2

1. 更新する StackSet を選択

任意の StackSet を選択し、アクションから「StackSet の詳細を編集」を押下。

The screenshot shows the AWS CloudFormation console's StackSets list. A table lists nine StackSets. The first row, labeled 'test-StackSet', is selected. An 'Actions' menu is open above the table, with the 'Edit StackSet details' option highlighted in blue. Other options in the menu include 'Add stack to StackSet', 'Rewrite StackSet parameters', 'Delete stack from StackSet', 'Edit deployment automation', 'Check drift', and 'Delete StackSet'. The table columns are 'StackSet Name', 'StackSet ID', 'Status', and 'Description'.

StackSet 名	StackSet ID	ステータス	StackSet の説明
test-StackSet	test-StackSet:1c207a82-de94-49a4-b7eb-ba845bc8c47a	SERVICE_MANAGED	test stack set

2. テンプレートの選択

現在のテンプレートを使用するか、既存のテンプレートを置き換えるか選択。

3. StackSet の詳細を変更可能

StackSet の説明、テンプレートのパラメータを変更可能。

The screenshot shows the 'Edit StackSet details' configuration page. It has three main sections: 'StackSet Name' (with 'test-StackSet' entered), 'StackSet の説明' (with 'test stack set' entered), and 'Parameters' (with a note about defining parameters in the template). The 'Parameters' section is currently collapsed.

StackSet 名
StackSet 名
test-StackSet
小文字、大文字、数字、ダッシュを含める必要があります。文字で始まる必要があります。

StackSet の説明
説明を使用して、StackSet の目的やその他の重要な情報を識別できます。
StackSet の説明
test stack set

パラメータ
パラメータは、テンプレートで定義されます。また、パラメータを使用すると、Stack を作成または更新する際にカスタム値を入力できます。

StackSetの更新 2/2

3. StackSetオプションの設定

タグや実行設定を指定。

アカウントフィルタータイプ(サービススマネージド)

デプロイターゲットを個々のアカウントに制限したり、指定された組織単位(OU)を使用してターゲットアカウントを指定することが可能。



なし

アカウントフィルターは適用されません。



共通集合

指定された OU から指定された個別のアカウントにデプロイします。



差分

指定された OU から指定された個々のアカウントを除外します。



和集合

追加の個別アカウントに加えて、指定された OU にデプロイします。



4. デプロイオプションの設定

デプロイ先やリージョン、デプロイオプションやリージョン等を指定。

The screenshot shows the AWS CloudFormation StackSets update configuration interface. It includes sections for 'Account-level targets' (with 'OU' selected), 'Deployment options' (with 'Regions' selected), and 'Deployment settings' (with 'Sequential' selected). A legend on the left maps icons to target types: 'No filter' (no icon), 'Intersection' (two overlapping squares), 'Difference' (square minus circle), and 'Union' (two joined squares).

組織単位 (OU) のデプロイ
この更新は、この OU とその OU の子のすべてのアカウントにデプロイされます。

アカウントへのデプロイ
この更新は、指定した個々のアカウントにデプロイされます

AWS OU ID
Enter OU

別の OU を追加

さらに 9 の OU を追加できます

アカウントフィルタータイプ オプション
デプロイターゲットを OU 全体ではなく OU 内の特定のアカウントに設定します。

アカウントフィルタータイプの選択

- なし アカウントフィルターは選択されません。
- 共通集合 指定された OU から指定された個別のアカウントにデプロイします。
- 差分 指定された OU から指定された個々のアカウントを除外します。
- 和集合 追加の個別アカウントに加えて、指定された OU にデプロイします。

すべてのリージョンを追加 すべてのリージョンを削除

デプロイオプション

同時アカウント的最大数 - オプション
StackSetsを同時にデプロイできるリージョン別のアカウント数。数値が大きいほど、オペレーションが高速になります。

障害耐性 - オプション
StackSetsが失敗するリージョン別のアカウント数。この値を超えると、このリージョンでのオペレーションが CloudFormation で停止されます。1つのリージョンで停止されたオペレーションは、他のリージョンでも続行されなくなります。数値が小さいほど、オペレーションの安全性が高くなります。

リージョンの同時実行
選択して、StackSetsをリージョンに順次デプロイするか、並行してデプロイします。

○ 順次 リージョンのデプロイ順序を指定し、複数の StackSets オペレーションを同時に 1 つのリージョンにデプロイします。

○ 並行 指定したすべてのリージョンに複数の StackSets オペレーションを並行してデプロイします。

https://docs.aws.amazon.com/ja_jp/AWSCloudFormation/latest/UserGuide/account-level-targets.html

https://docs.aws.amazon.com/ja_jp/AWSCloudFormation/latest/UserGuide/stacksets-update.html

オペレーション

- ・ **アクセス許可モデルの選択と設定 (初回のみ)**
 - スタックセットはセルフマネージド型またはサービスマネージド型のアクセス許可が必要。
- ・ **スタックセットの作成**
 - スタックの作成に使用するテンプレート、ターゲットアカウント、デプロイする AWS リージョンの指定を行い作成。
- ・ **スタックセットの更新**
 - スタックセットを更新すると、スタックセットのスタックに変更内容がプッシュされる。
- ・ **スタックインスタンスのパラメータの上書き**
 - アカウントおよびリージョン別にスタックインスタンスのパラメータ値を上書き可能。
- ・ **スタックの削除**
 - 指定したリージョン内の指定したターゲットアカウントから、スタックを削除する。
(スタックの削除に併せて対象のスタックインスタンスも削除される。)
- ・ **スタックセットの削除**
 - スタックセット内にスタックインスタンスが存在しない場合のみ、削除可能。

https://docs.aws.amazon.com/ja_jp/AWSCloudFormation/latest/UserGuide/stacksets-concepts.html#stacksets-concepts-ops



スタックインスタンスのパラメータの上書き 1/2

1. 更新するスタックセットを選択

任意のスタックセットを選択し、アクションから「StackSet のパラメータを上書き」を押下。

The screenshot shows the AWS CloudFormation StackSets console with 9 entries. A context menu is open over the first entry, "test-StackSet". The menu items are: "StackSet にスタックを追加", "StackSet の詳細を編集", "StackSet のパラメータを上書き" (which is highlighted with a blue border), "StackSet からスタックを削除", "自動デプロイを編集", "ドリフトの検出", and "StackSet の削除". Below the table, there are filters for "ステータスのフィルター" (Active) and "アクティブ".

2. デプロイオプションの設定

デプロイ先やリージョン、デプロイオプションやリージョン等を指定。

The screenshot shows the "Override Parameters" configuration dialog. It has several sections:

- 組織単位 (OU) へのデプロイ**: "この更新は、この OU とこの OU の子のすべてのアカウントにデプロイされます。" (Details)
- アカウントへのデプロイ**: "この更新は、指定した個々のアカウントにデプロイされます。" (Details)
- AWS OU ID**: "Enter OU" input field and "削除" button.
- 別の OU を追加**: "さらに 9 の OU を追加できます" (Details)
- アカウントフィルタータイプ - オプション**: "デプロイターカークを OU 全体ではなく OU 内の特定のアカウントに設定します。" (Details)
- リージョンの指定**: "StackSet をデプロイするリージョンを選択します。StackSet は、指定した順序でこれらのリージョンにデプロイされます。" (Details)
- デプロイオプション**:
 - 同時アカウントの最大数 - オプション**: "StackSet を同時にデプロイできるリージョン別アカウント数。数値が大きいほど、オペレーションが高速になります。" (Details)
 - 障害耐性 - オプション**: "StackSet が失敗するリージョン別アカウント数。この値を超えると、このリージョンでのオペレーションが CloudFormation で停止されます。1つのリージョンで停止されたオペレーションは、他のリージョンでも実行されなくなります。数値が小さいほど、オペレーションの安全性が高くなります。" (Details)
 - リージョンの同時実行**: "選択して、StackSets をリージョンに順次デプロイするか、並行してデプロイします。"
 - 順次**: "リージョンのデプロイ順序を指定し、複数の StackSets オペレーションを同時に 1 つのリージョンにデプロイします。"
 - 並行**: "指定したすべてのリージョンに複数の StackSets オペレーションを並行してデプロイします。"
 - 同時実行モード**: "オペレーションが実行中の同時実行レベルの動作を指定します。"
 - 厳格な耐障害性**: "同時に 1 つ以上のアカウントを動かすことで、失敗したアカウントの数が耐障害性 + 1 を超えないようにします。"
 - ソフト障害耐性**: "常に指定された同時実行レベルで実行します。"

https://docs.aws.amazon.com/ja_jp/AWSCloudFormation/latest/UserGuide/stackinstances-override.html

© 2023, Amazon Web Services, Inc. or its affiliates.



スタックインスタンスのパラメータの上書き 2/2

3. 上書きの指定

任意のパラメータを選択し、アクションから「StackSet 値の上書き」を押下。



パラメータ

検索パラメータ

名前	StackSet 値
MultiRegion	false

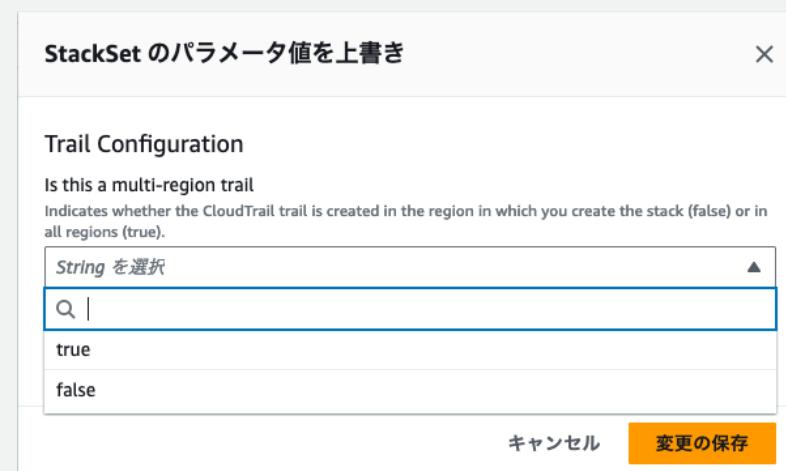
上書き値の編集 ▲

StackSet 値の上書き

StackSet 値に設定

上書きフィールドを元に戻す

上書きする内容を入力することが可能。



StackSet のパラメータ値を上書き

X

Trail Configuration

Is this a multi-region trail

Indicates whether the CloudTrail trail is created in the region in which you create the stack (false) or in all regions (true).

String を選択

検索

true

false

キャンセル 変更の保存

留意点

※上書きされたパラメータをStackSetで指定された値に戻すためには、「StackSet 値に設定」を選択する必要がある。



パラメータ

検索パラメータ

名前	StackSet 値	値の上書き
MultiRegion	false	StackSet 値に戻す

※上書きされたパラメータ値は、設定時に指定したアカウントにのみ適用される。
(今後作成されるスタックには**適用されない**。)

オペレーション

- ・ **アクセス許可モデルの選択と設定 (初回のみ)**
 - スタックセットはセルフマネージド型またはサービスマネージド型のアクセス許可が必要。
- ・ **スタックセットの作成**
 - スタックの作成に使用するテンプレート、ターゲットアカウント、デプロイする AWS リージョンの指定を行い作成。
- ・ **スタックセットの更新**
 - スタックセットを更新すると、スタックセットのスタックに変更内容がプッシュされる。
- ・ **スタックインスタンスのパラメータの上書き**
 - アカウントおよびリージョン別にスタックインスタンスのパラメータ値を上書き可能。
- ・ **スタックの削除**
 - 指定したリージョン内の指定したターゲットアカウントから、スタックを削除する。
(スタックの削除に併せて対象のスタックインスタンスも削除される。)
- ・ **スタックセットの削除**
 - スタックセット内にスタックインスタンスが存在しない場合のみ、削除可能。

https://docs.aws.amazon.com/ja_jp/AWSCloudFormation/latest/UserGuide/stacksets-concepts.html#stacksets-concepts-ops



Stackの削除

1. 削除するStackを選択

Stackセットを選択し、アクションから「StackSet からStackを削除」を押下。

The screenshot shows the AWS CloudFormation StackSets console. In the center, there is a table with two columns: 'StackSet 名' (Name) and 'StackSet ID'. The first row contains 'test-StackSet' and 'test-StackSet:1c207a82-de94-49a4-b7eb-ba845bc8c47a'. To the right of the table is a vertical menu with several options: 'StackSet にStackを追加', 'StackSet の詳細を編集', 'StackSet のパラメータを上書き', 'StackSet からStackを削除' (which is highlighted with a blue background), '自動デプロイを編集', 'ドリフトの検出', and 'StackSet の削除'. At the top of the menu, there is a button labeled 'StackSet の作成'.

2. デプロイオプションの設定

デプロイ先やリージョン、デプロイオプションやリージョン等を指定。

This screenshot shows the configuration dialog for deleting a stack from a StackSet. It consists of several sections:

- 組織単位 (OU)**: A section for specifying the Organizational Unit (OU). It includes fields for 'AWS OU ID' (with placeholder 'Enter OU') and '別の OU を追加' (Add another OU). A note says 'さきにこの OU を追加できます' (You can add this OU here).
- アカウントフィルタータイプ - オプション**: A note explaining that the target account is limited to specific accounts within the OU, not the entire account.
- リージョンの指定**: A section for specifying regions. It includes a note about the order of deployment and a link '詳細はこちる' (See details). Buttons for 'すべてのリージョンを追加' (Add all regions) and 'すべてのリージョンを削除' (Delete all regions) are present.
- デプロイオプション**: A large section containing various deployment options:
 - 同時アカウントの最大数 - オプション**: A dropdown for specifying the maximum number of accounts to deploy to simultaneously, with a value of '1' selected.
 - 障害耐性 - オプション**: A dropdown for specifying the number of regions per account, with a value of '0' selected.
 - リージョンの同時に実行**: Options for parallel execution: '順次' (Sequential), '並行' (Parallel), and '指定したすべてのリージョンに複数のStackSetsオペレーションを並行してデプロイします' (Deploy multiple StackSets operations in parallel across all specified regions).
 - 同時に実行レベル**: Options for parallel execution levels: 'オペレーション' (Operation), '最終的な耐障害性' (Final durability), and 'ソフト障害耐性' (Soft durability).

https://docs.aws.amazon.com/ja_jp/AWSCloudFormation/latest/UserGuide/stackinstances-delete.html

オペレーション

- ・ **アクセス許可モデルの選択と設定 (初回のみ)**
 - スタックセットはセルフマネージド型またはサービスマネージド型のアクセス許可が必要。
- ・ **スタックセットの作成**
 - スタックの作成に使用するテンプレート、ターゲットアカウント、デプロイする AWS リージョンの指定を行い作成。
- ・ **スタックセットの更新**
 - スタックセットを更新すると、スタックセットのスタックに変更内容がプッシュされる。
- ・ **スタックインスタンスのパラメータの上書き**
 - アカウントおよびリージョン別にスタックインスタンスのパラメータ値を上書き可能。
- ・ **スタックの削除**
 - 指定したリージョン内の指定したターゲットアカウントから、スタックを削除する。
(スタックの削除に併せて対象のスタックインスタンスも削除される。)
- ・ **スタックセットの削除**
 - スタックセット内にスタックインスタンスが存在しない場合のみ、削除可能。

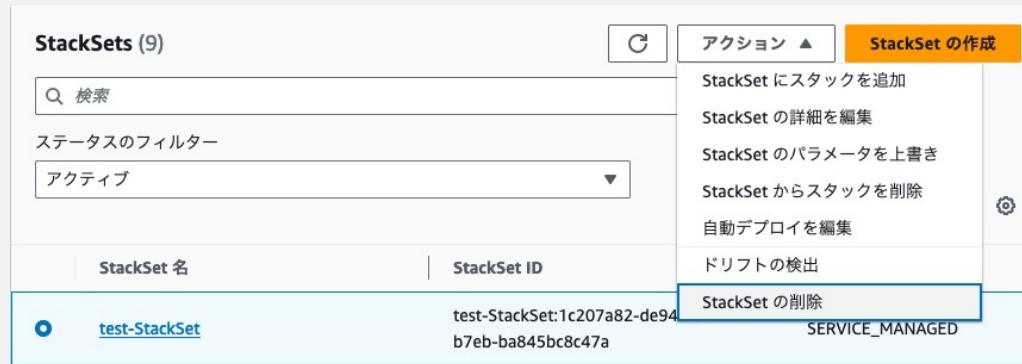
https://docs.aws.amazon.com/ja_jp/AWSCloudFormation/latest/UserGuide/stacksets-concepts.html#stacksets-concepts-ops



StackSet の削除

1. 削除するStackSetを選択

StackSetを選択し、アクションから「StackSetの削除」を押下。



StackSets (9) リスト画面。左側に検索機能とステータスフィルターがあります。右側にはアクションメニューが表示されています。リスト内に一つのStackSetが選択されている（test-StackSet）。

StackSet名	StackSet ID	作成者
test-StackSet	test-StackSet:1c207a82-de94b7eb-ba845bc8c47a	SERVICE_MANAGED

アクションメニュー:

- StackSetにStackを追加
- StackSetの詳細を編集
- StackSetのパラメータを上書き
- StackSetからStackを削除
- 自動デプロイを編集
- ドリフトの検出
- StackSetの削除

2. 削除を実行

削除ボタンを押下。



※StackSet内にStackインスタンスが存在しない場合のみ、削除可能。



Thank you!

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
- ・ 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）