AWS Black Belt Online Seminar

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

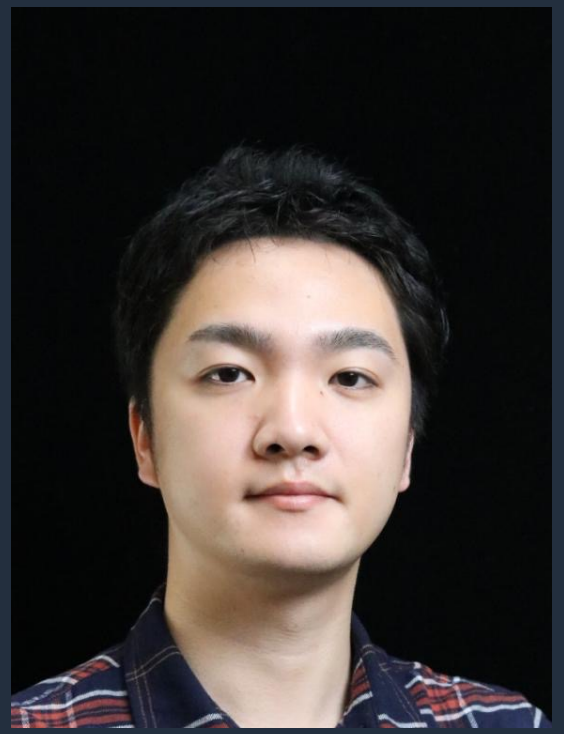
自己紹介

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


```yaml
MyTestResource:
  Type: Custom::Test
  Properties:
    ServiceToken: !Ref LambdaARN
    Name: "Value"
    List: [ "1", "2", "3" ]

OtherResource:
  Type: …
  Properties:
    Name: !GetAtt MyTestResource.OutputName1
```

```json
{
    "RequestType" : "Create",
    "ResponseURL" : "http://pre-signed-S3-url",
    "StackId" : "arn:aws:cloudformation:…",
    "RequestId" : "unique id for this create request",
    "ResourceType" : "Custom::Test",
    "LogicalResourceId" : "MyTestResource",
    "ResourceProperties" : {
        "Name" : "Value",
        "List" : [ "1", "2", "3" ]
    }
}
```

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

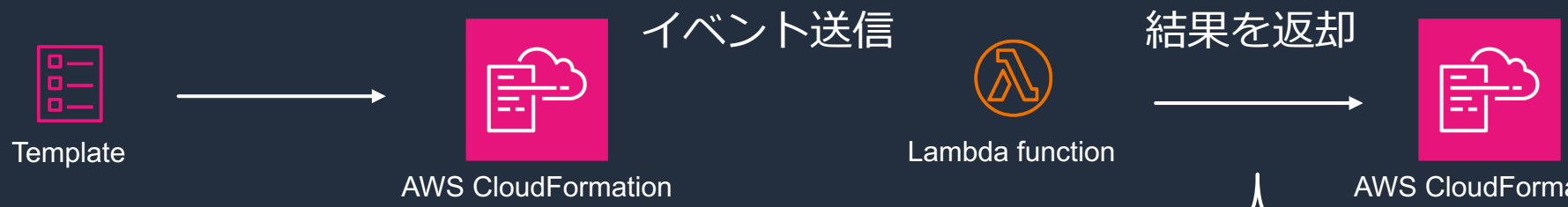
- 送られたイベントをもとに処理を実行後、CFn に結果を送信する
- CloudFormation から Lambda 関数を作成すると、cfnresponse モジュールで結果を返却できる


```
    if event['RequestType'] == "Create" or event['RequestType'] == "Update":
        print(event['RequestType'])
        time.sleep(int(event['ResourceProperties']['SleepTime']))

    elif event['RequestType'] == "Delete":
        print(event['RequestType'])
        pass

    cfnresponse.send(event, context, cfnresponse.SUCCESS, responseData=result)
```

任意の処理を追加する（カスタムリソース）




```
MyTestResource:
  Type: Custom::Test
  Properties:
    ServiceToken: !Ref LambdaARN
    Name: "Value"
    List: [ "1", "2", "3" ]

OtherResource:
  Type: …
  Properties:
    Name: !GetAtt MyTestResource.OutputName1
```

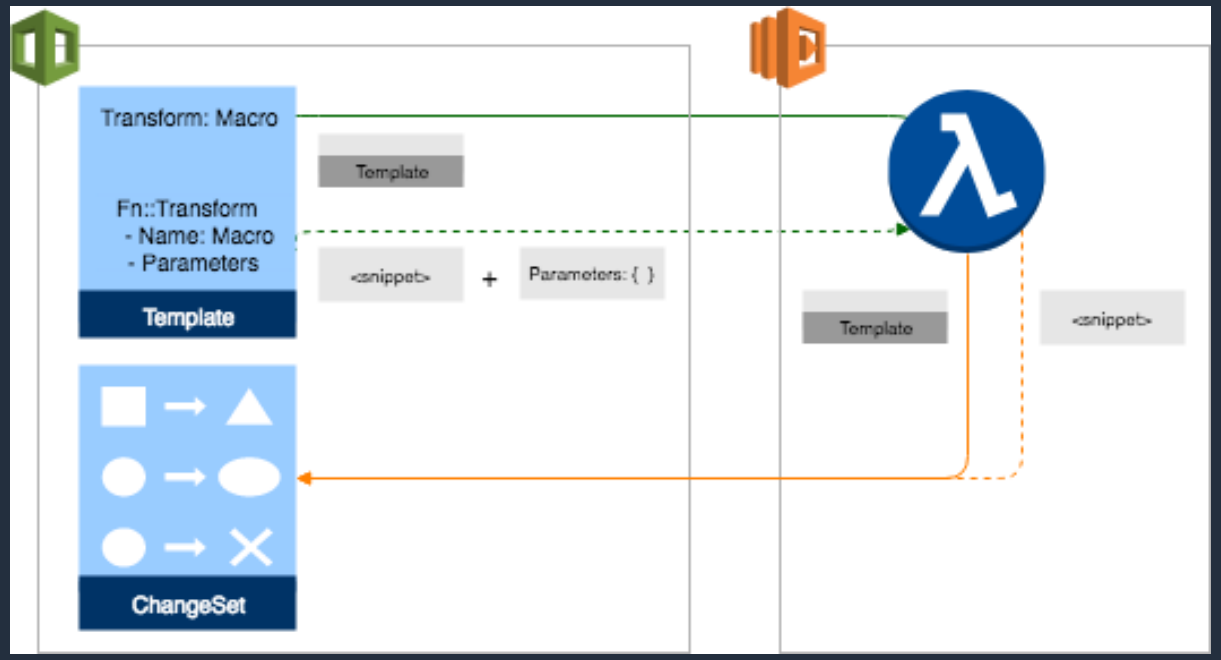
```
{
  "Status" : "SUCCESS",
  "PhysicalResourceId" : "TestResource1",
  "StackId" : "arn:aws:cloudformation:…",
  "RequestId" : "unique id for this create request",
  "LogicalResourceId" : "MyTestResource",
  "Data" : {
    "OutputName1" : "Value1"
  }
}
```

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)

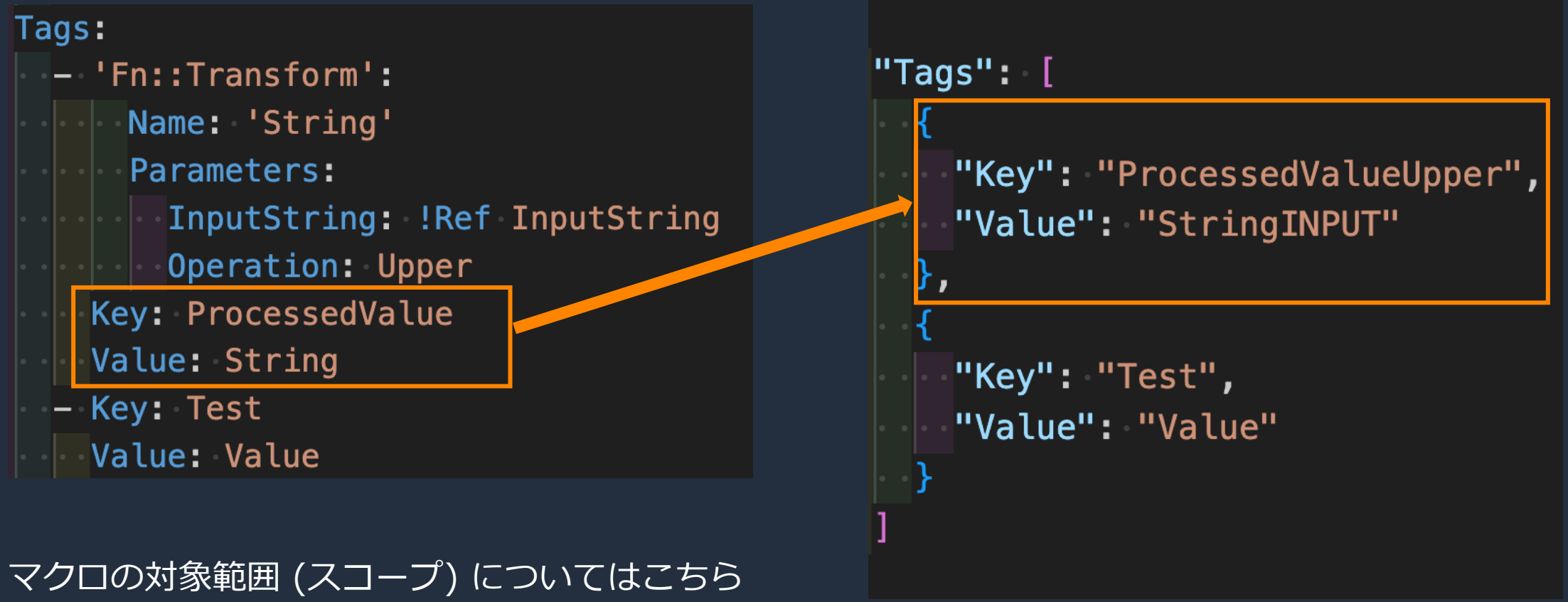
- スタック実行前にテンプレートを Lambda で加工できる
  - 事前に Lambda 関数とマクロのリソースを作成することで使用可能となる
  - AWS が事前に用意しているものもある
  - 変更前後のテンプレートをそれぞれ Original/Processed と呼ぶ



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 → Lambda function → Template - Processed



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"
  }
}
```

Lambda が返却する response の内容

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

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

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::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

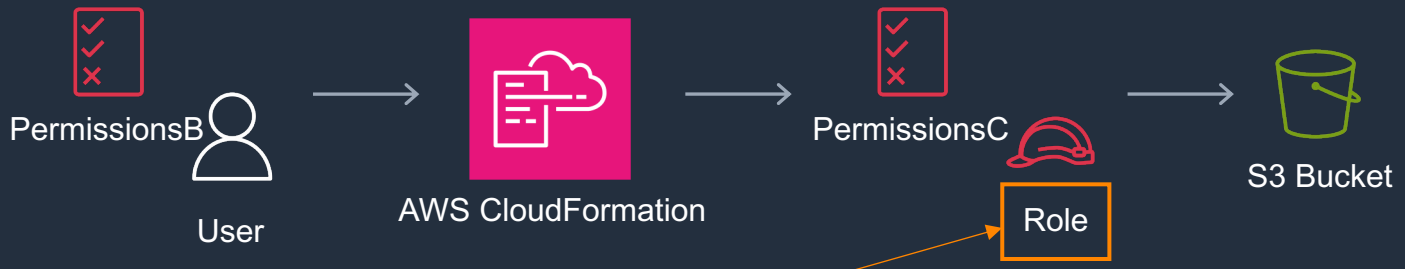
スタック作成権限とリソースの保護

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

スタック作成権限とリソースの保護

スタック削除保護

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

スタックポリシー

- スタック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
- スタック自体の削除やスタック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
- スタック自体の削除やスタック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

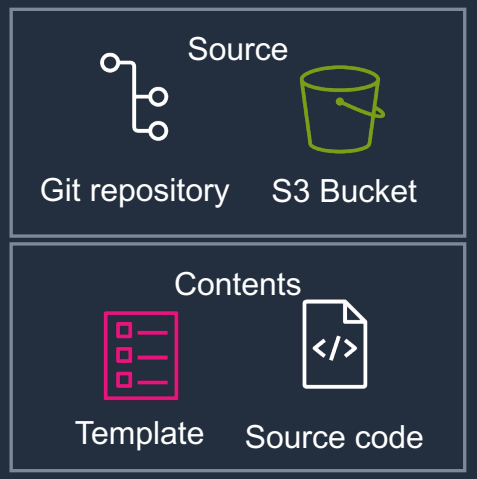
Deep な機能の使い方

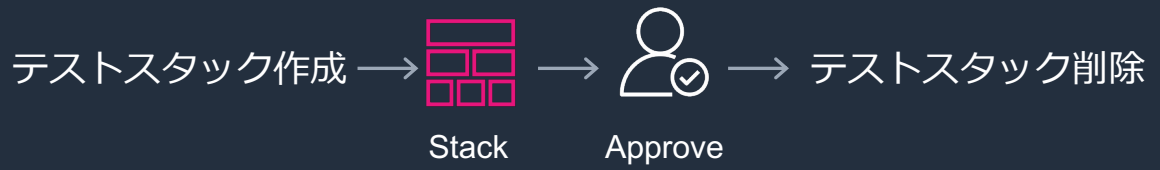
- 任意の処理を追加する（カスタムリソース）
- スタック作成/更新時にテンプレートを加工する (マクロ)
- スタック作成権限とリソースの保護
- CodePipelineからCFnスタックをデプロイする

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

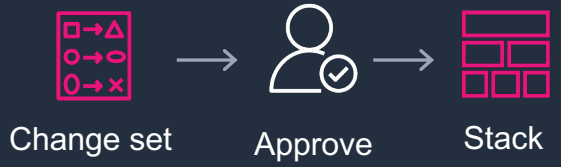
1 Source → 2 Build → 3 テストスタックデプロイ →





- パラメータ (設定) ファイル作成
- Lambda 関数のソースコードのビルド
- コンテナイメージのビルド
- その他

4 本番スタックデプロイ



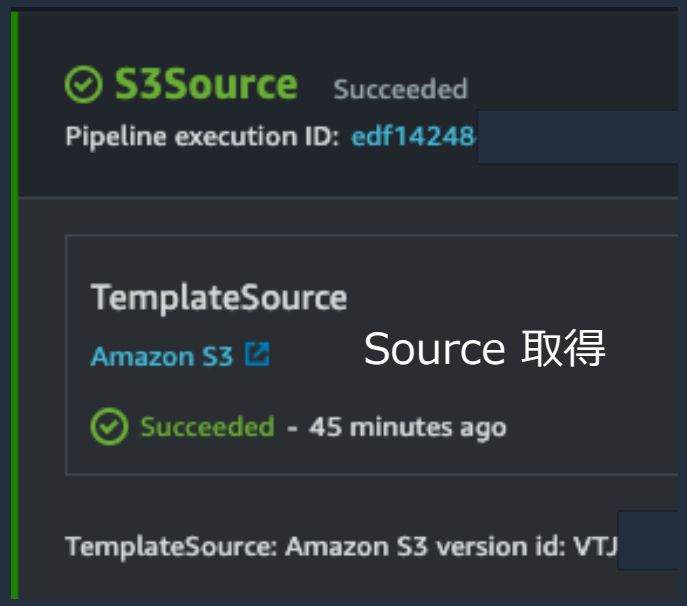
ユースケース

- 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

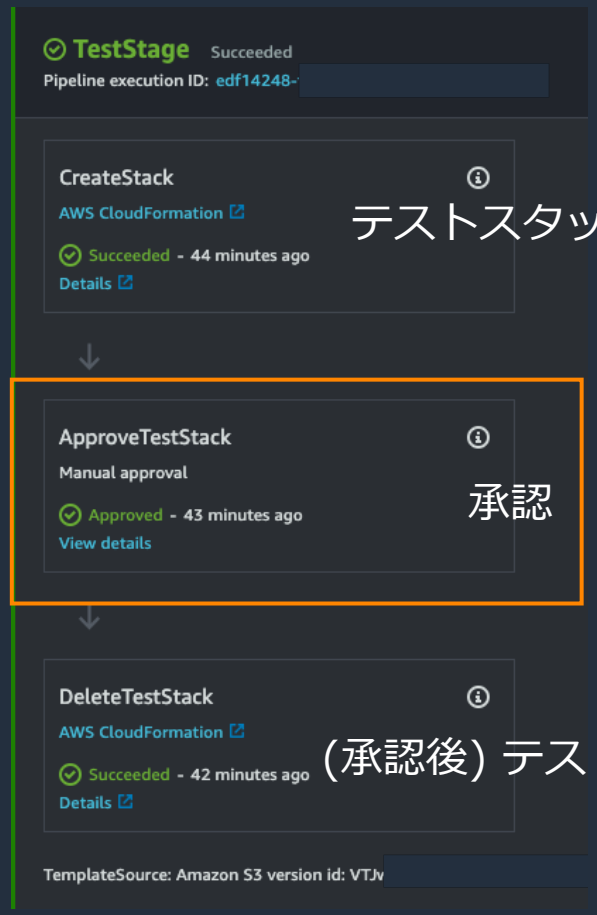
https://docs.aws.amazon.com/ja_jp/AWSCloudFormation/latest/UserGuide/continuous-delivery-codepipeline-basic-walkthrough.html

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



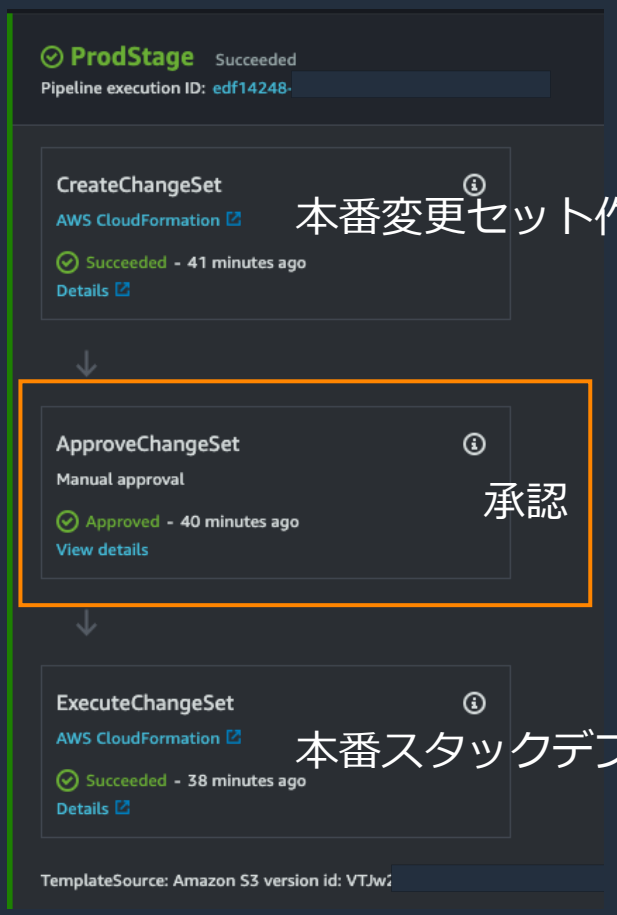
Source 取得



テストスタックデプロイ

承認

(承認後) テストスタック削除



本番変更セット作成

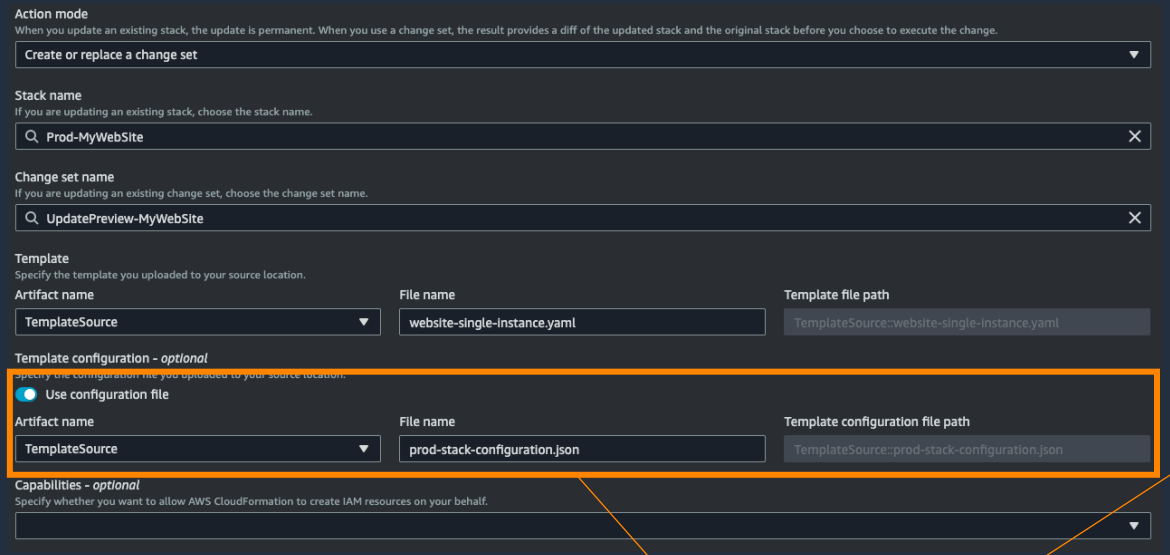
承認

本番スタックデプロイ

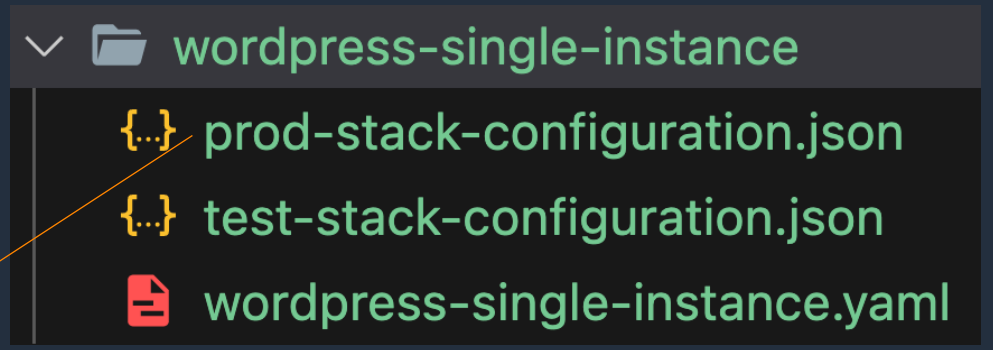
CodePipelineからCFnスタックをデプロイする

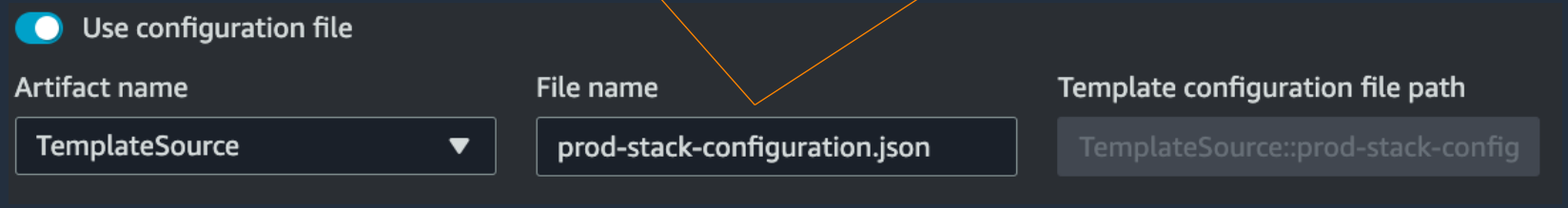
アーティファクトの設定を行うことで環境 (テスト、本番など) ごとに異なるファイルからパラメータを読み込むことが可能

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



アーティファクト zip のファイル一覧





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!