

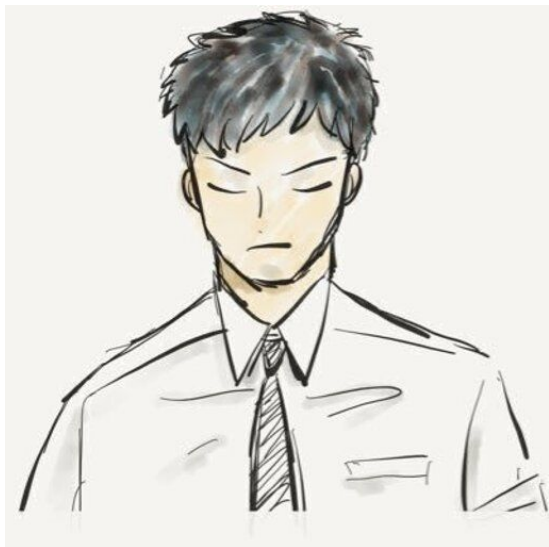
JAWS-UG 京都支部 × KyotoLT

AWSを活用したマルチテナントSaaSの設計と実践

CDKハンズオン

株式会社シーズ 中村 勇太





株式会社シーズ - クラウドソリューション事業部

中村 勇太 Yuta Nakamura

FROM



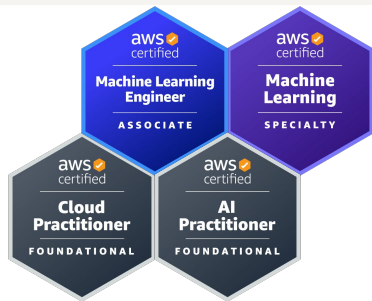
経歴

2024年3月 大阪電気通信大学 卒業

2024年4月 株式会社シーズ 入社

好きなAWSサービス

Amazon Lightsail / AWS Step Functions



目次

ハンズオン環境構築

IaC / CDKについて

CDKハンズオン

SaaSハンズオン環境構築（先にデプロイだけします！）



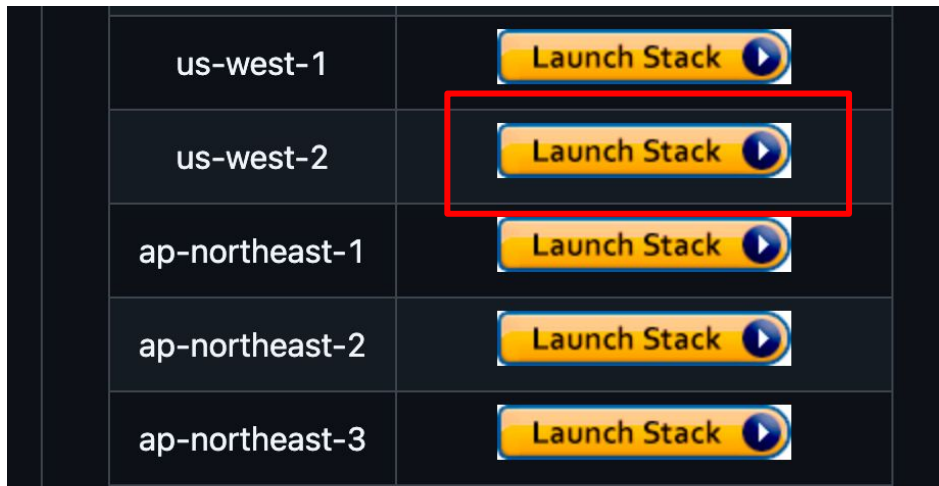
ハンズオン環境構築



ハンズオン環境構築 1/2

<https://github.com/aws-samples/sagemaker-studio-code-editor-template>

us-west-2のLaunch Stackをクリックする



参考: <https://qiita.com/moritalous/items/656793cd97e1e967f045>

ハンズオン環境構築 1/2

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。

AutoStopIdleTimeInMinutes

Idle time (in minutes) before auto-stop of Code Editor, disabled if 0 (Must be 60 or greater, or 0.)

360

EbsSizeInGb

EBS volume size of Code Editor (This parameter cannot be less than 20 GB.)

20

360にしてください

機能

① The following resource(s) require capabilities: [AWS::IAM::Role]

このテンプレートには、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デプロイする前に、IAM リソースの権限を確認してください。カスタム名が、ご利用の AWS アカウント内で一意のものであることを確認してください。 [詳細はこちら](#)

☒ AWS CloudFormation によって IAM リソースがカスタム名で作成される場合があることを承認します。

スタックのクイック作成

テンプレート

テンプレート URL
<https://es-assets-prod-us-east-1-nrt-2-cb4b4649d0e0794s3.ap-northeast-1.amazonaws.com/9748a536-3a71-4ff0e-afcd-ecf160e3487/cloudformation/CodeEditorStack.template>

スタックの説明
AWS CloudFormation Template for SageMaker Studio Code Editor

スタック名を提供

スタック名

CodeEditorStack

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。

AutoStopIdleTimeInMinutes

Idle time (in minutes) before auto-stop of Code Editor, disabled if 0 (Must be 60 or greater, or 0.)

360

EbsSizeInGb

EBS volume size of Code Editor (This parameter cannot be less than 20 GB.)

20

InstanceType

Instance type of Code Editor (Fast launch is supported on ml.i3.medium, ml.i3.large, ml.m5.large, ml.m5.xlarge, and ml.c5.large.)

ml.i3.medium

UseDefaultVpc

Whether to use the default VPC (true) or create a new one (false)

true

タグ - オプション

タグ(キーと値のペア)は AWS リソースにメタデータを適用するために使用され、それらのリソースの整理、識別、分類に役立ちます。スタックごとに最大 50 個の固有のタグをスタック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

アクセス許可 - オプション

CloudFormation が引き受けられることができる既存の AWS Identity and Access Management (IAM) サービスロールを指定します。

IAM ロール - オプション

スタックに追加されるすべての CloudFormation IAM ロールを選択します。

IAM ロール名

Sample role name

詳細オプション

通知オプションやスタックポリシーなど、スタックのオプションを追加設定することができます。 [詳細はこちら](#)

▶ スタックポリシー - オプション

スタックの更新中の意図しない更新から保護するリソースを定義します。

▶ ロールバック設定 - オプション

スタックの作成時および更新時にモニタリングする CloudFormation のアラームを指定します。オペレーションでアラームのしきい値を超過した場合、CloudFormation ではスタックの作成を中止し、ロールバックを実行します。

▶ 通知オプション - オプション

スタックイベントに関する通知が送信される新機または既存の Amazon Simple Notification Service トピックを指定します。

▶ スタックの作成オプション - オプション

スタックの作成時に使用するテンプレートのバージョンを指定します。

機能

① The following resource(s) require capabilities: [AWS::IAM::Role]

このテンプレートには、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デプロイする前に、IAM リソースの権限を確認してください。カスタム名が、ご利用の AWS アカウント内で一意のものであることを確認してください。 [詳細はこちら](#)

☒ AWS CloudFormation によって IAM リソースがカスタム名で作成される場合があることを承認します。



laC / CDKとは



laCとは 1/2

Infrastructure as Code (laC)とは、インフラストラクチャの構成をコードとして管理するアプローチです。

手動でのインフラ構築やGUIツールを使った設定に代わり、テキストファイルやコードを使ってインフラの構成を定義します。



laCとは 2/2

一般的に知られているEC2を作成する方法に、コンソールにログインして作成する方法がありますが、laCを用いて作成することもできます。

インスタンスを起動 情報

Amazon EC2 では、AWS クラウドで実行される仮想マシン (インスタンス) を作成できます。以下の簡単なステップに従ってすばやく開始できます。

名前とタグ 情報

名前

例: My Web Server

さらにタグを追加

▼ アプリケーションおよび OS イメージ (Amazon マシンイメージ) 情報

AMI は、インスタンスの起動に必要なソフトウェア設定 (オペレーティングシステム、アプリケーションサーバー、アプリケーション) を含むテンプレートです。お探しのものが以下に表示されない場合は、AMI を検索または参照してください。

🔍 何千ものアプリケーションイメージと OS イメージを含むカタログ全体を検索します。

クイックスタート



その他の AMI を開
発する
AWS、Marketplace、
コミュニティからの
AMI を含む

Amazon マシンイメージ (AMI)

Amazon Linux 2023 AMI
ami-08ea1604e9f1115d (64 ビット, uefi, preferred) / ami-0ba1327d947a471fd (64 ビット (Arm), uefi)
仮想化: hvm ENA 有効: true ルートデバイスタイプ: ebs

無料利用枠の対象

説明

Amazon Linux 2023 は、5 年間の長期サポートを備えた、最新の汎用 Linux ベースの OS です。AWS 向けに最適化されており、クラウドアプリケーションを開発および実行するための安全で安定した高性能な実行環境を提供するように設計されています。

Amazon Linux 2023 AMI 2023.7.20250331.0 arm64 HVM kernel-6.1

| アーキテクチャ | ブートモード | AMI ID | 発行日 | ユーザー名 |
|----------------|--------|-----------------------|------------|----------|
| 64 ビット (Arm) ▼ | uefi | ami-0ba1327d947a471fd | 2023-03-29 | ec2-user |

検証済みプロバイダー

```
{
  "version": "tree-0.1",
  "tree": {
    "id": "App",
    "path": "",
    "children": {
      "CdkSampleStack": {
        "id": "CdkSampleStack",
        "path": "CdkSampleStack",
        "children": {
          "Vpc": {
            "id": "Vpc",
            "path": "CdkSampleStack/Vpc",
            "children": {
              "Resource": {
                "id": "Resource",
                "path": "CdkSampleStack/Vpc/Resource",
                "attributes": {
                  "aws:cdk:cloudformation:type": "AWS::EC2::VPC",
                  "aws:cdk:cloudformation:props": {
```

laCのメリット

- 再現性：同じコードから常に同じ環境が構築される
- バージョン管理：GitなどでlaCコードを管理が可能になる
- 自動化：デプロイメントプロセスの自動化が容易になる
- ドキュメント化：laCコード自体がドキュメントとなる
- スケーラビリティ：環境の複製や拡張が容易



AWSにおけるIaC

AWS CloudFormation(CFn)

YAML/JSONでAWSリソースを記述、幅広くリソースを管理できる。

AWS SAM

CFnをベースに、AWS Lambda中心にサーバレスな構成を記述できる。

AWS CDK

コンストラクトによる部品化に加えて条件分岐、ループといったプログラミングの恩恵を享受できる。



MEET SAM.

https://d2908q01vomqb2.cloudfront.net/da4b9237baccdf19c0760cab7aec4a8359010b0/2017/08/09/aws_sam_introduction-1024x286.png



CDKとは

AWS CDK (Cloud Development Kit) は、プログラミング言語 (TypeScript、Python、Java、C#など) を使ってAWSリソースを定義できるオープンソースのIaCフレームワークです。

CDKで書かれたコードは最終的にCFnに変換されてデプロイされます。



CDKのメリット

- プログラミング言語の利用：馴染みのある言語でAWSリソースを定義
- 型安全性：コンパイル時にエラーを検出（TypeScriptなど）
- 抽象化：高レベルの構成要素（Construct）を使用可能
- 再利用性：コンポーネントの再利用が容易
- IDE統合：コード補完やリファクタリング、AIエージェントの機能が使える



CDK ハンズオン

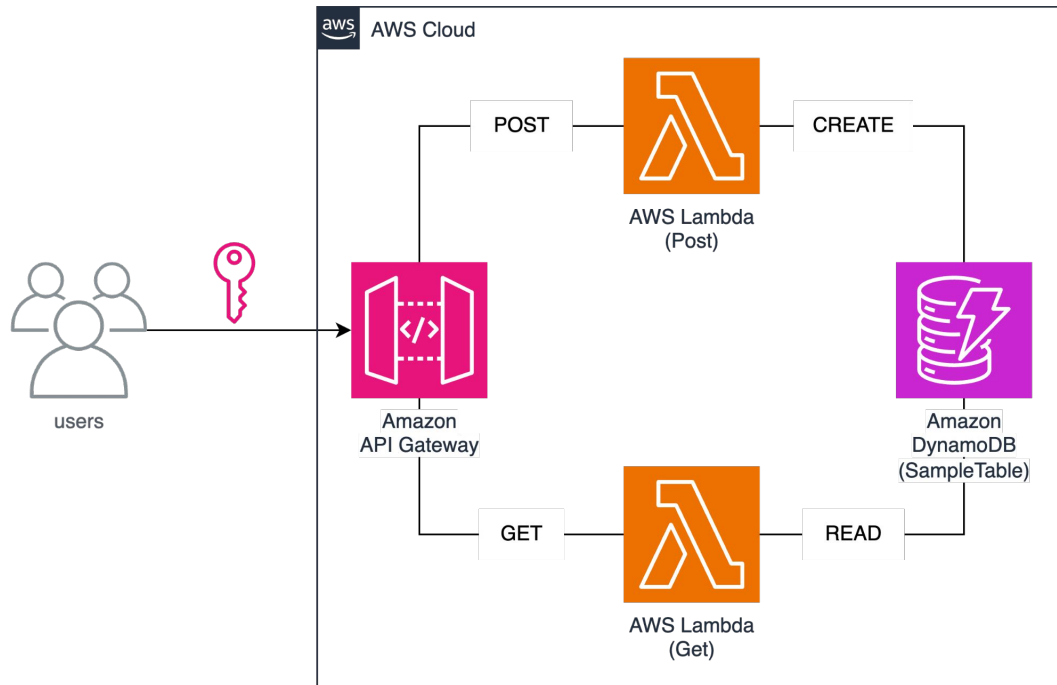


CDK触ったことある人 🙋



CDKハンズオンの構成

シンプルなCreateとReadができる
サーバーレスなアプリケーション



ハンズオン環境へのアクセス 1/2

Amazon SageMaker AIのコンソールにアクセス

The screenshot displays the Amazon SageMaker AI console interface. On the left sidebar, under the 'Applications and IDEs' section, the 'Studio' option is highlighted with a red rectangular box. The main content area features a dark header with the text 'Amazon SageMaker' and 'SageMaker Studio 機械学習向けの初の完全統合開発環境 (IDE)'. To the right, a white card titled '今すぐ始める' (Get started now) contains a dropdown menu labeled 'ユーザープロフィールを選択' (Select user profile) with 'DefaultUser' selected, and an orange button labeled 'Studio を開く' (Open Studio), both of which are also enclosed in a red rectangular box. At the bottom of the console, the text '新着情報' (New information) is visible.

Amazon SageMaker AI

開始方法

What's new 37

▼ Applications and IDEs

Studio

Canvas

RStudio

Notebooks

Partner AI Apps 新規

Amazon SageMaker

SageMaker Studio

機械学習向けの初の完全統合開発環境 (IDE)。

今すぐ始める

ユーザープロフィールを選択

DefaultUser

Studio を開く

新着情報

ハンズオン環境へのアクセス 2/2

Applications (6)

JupyterLab

RStudio

Canvas

Code Editor

Studio CL...

MLflow

Partner AI Apps

New

Home

Running instances

Compute

Provide feedback

About

Code Editor, based on Code-OSS, Visual Studio Code Open Source, enables you to write, test, debug and run your analytics and machine learning code. It is fully integrated with SageMaker Studio and supports IDE extensions available in the Open VSX Extension Registry.

[Learn more about Code Editor](#)

Search...

Filter spaces: Running

| Name | Application | Status | Type | Last modified | Action |
|---------|-------------|---------|---------|----------------|---------------------------------|
| default | Code Editor | Running | Private | 36 minutes ago | <div>Stop</div> <div>Open</div> |

1 results

Results are cached

Refresh

Go to page 1

Page 1 of 1

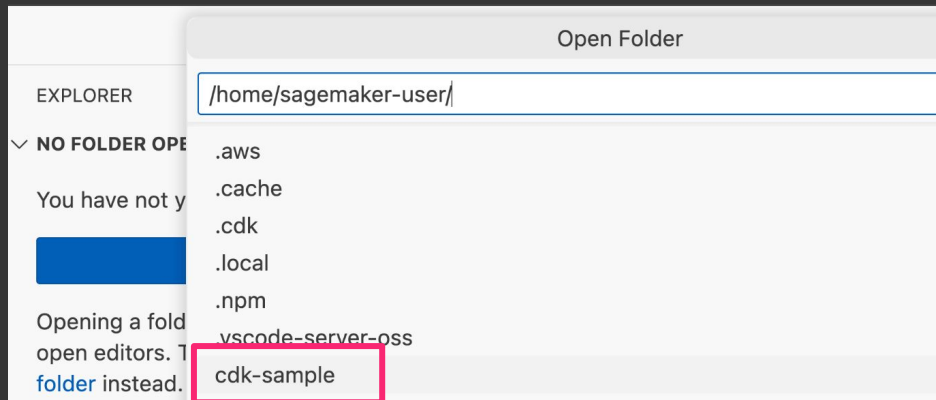
CDKハンズオンのコードの用意

```
> git clone https://github.com/papi-tokei/cdk-sample.git
```

Code Editorでフォルダを開く→

```
> cd cdk-sample
```

```
> npm i
```



リポジトリのファイル構成

> npx cdk bootstrap // CDK初めてだよという方は実行してください



リポジトリのファイル構成

```
> npx cdk bootstrap // CDK初めてだよという方は実行してください
```

さっそく見てみましょう

(git stashすればなんとかなります)



cdk-sample-stack.ts L11-16 - DynamoDB作成

```
// DynamoDBのテーブルを作成する
const table = new dynamodb.Table(this, 'SampleTable', {
  partitionKey: { name: 'id', type: dynamodb.AttributeType.STRING },
  tableName: 'SampleTable',
  removalPolicy: cdk RemovalPolicy.DESTROY,
});
```

cdk-sample-stack.ts L18-26, L38-40 - Lambda(Get)作成

```
// Lambda関数を作成する(GET用)
const getLambda = new lambda.Function(this, 'GetLambda', {
  runtime: lambda.Runtime.PYTHON_3_9,
  handler: 'get.handler',
  code: lambda.Code.fromAsset('lambda'),
  environment: {
    TABLE_NAME: table.tableName,
  },
});
=== 略 ===
// DynamoDBテーブルへのアクセス権限を Lambda関数に付与する
// GET Lambdaに対してテーブルの読み取り権限を付与
table.grantReadData(getLambda);
```

cdk-sample-stack.ts L44-51 - API Gateway作成 1/3

```
// API Gatewayを作成する
const api = new apigateway.RestApi(this, 'SampleApi', {
  restApiName: 'Sample Service',
  apiKeySourceType: apigateway.ApiKeySourceType.HEADER,
});

// APIキーを作成する
const apiKey = api.addApiKey('ApiKey');
```


cdk-sample-stack.ts L53-64 - API Gateway作成 2/3

```
// 使用量プランを作成し、APIキーと紐づける
const usagePlan = api.addUsagePlan('UsagePlan', {
  name: 'BasicUsagePlan',
  throttle: {
    rateLimit: 10,
    burstLimit: 2,
  },
});
usagePlan.addApiKey(apiKey);
usagePlan.addApiStage({
  stage: api.deploymentStage,
});
```

cdk-sample-stack.ts L66-70 - API Gateway作成 3/3

```
// API GatewayとLambda関数を紐づける
const getIntegration = new apigateway.LambdaIntegration(getLambda);
api.root.addMethod('GET', getIntegration, {
  apiKeyRequired: true, // Require API key
});
```

デプロイ

```
> npx cdk synth // CFnテンプレートを生成  
> npx cdk bootstrap // cdkが初めての方は実行してください(再掲)  
> npx cdk deploy // デプロイ
```

この内容でデプロイしてOKか聞かれますので y で返答してください。

コンソールから確認

テーブル (1) 情報

Q テーブルの検索

任意のタグキー

| <input type="checkbox"/> | 名前 ▲ | 状態 ▼ | パーティションキー ▼ | ソートキー ▼ | インデックス |
|--------------------------|-----------------------------|---------|-------------|---------|--------|
| <input type="checkbox"/> | SampleTable | 🟢 アクティブ | id (S) | - | |

関数 (33)

Q 属性によるフィルター、またはキーワードによる検索

| <input type="checkbox"/> | 関数名 ▼ | 説明 |
|--------------------------|--|----|
| <input type="checkbox"/> | CdkSampleStack-GetLambda3B1776D4-sJKLFplUQTuq | - |
| <input type="checkbox"/> | CdkSampleStack-PostLambda0EF1C7F9-lwuX4lCp5abT | - |

API Gateway

API

カスタムドメイン名
ドメイン名アクセスの関連付け
VPC リンク

使用量プラン

API キー

クライアント証明書
設定

API (1/1)

Q API の検索

| <input type="radio"/> | 名前 ▲ | 説明 ▼ |
|-----------------------|--------------------------------|------|
| <input type="radio"/> | Sample Service | |



API キーのチェック

API Gateway



API

カスタムドメイン名

ドメイン名アクセスの関連付け

VPC リンク

使用量プラン

API キー

クライアント証明書

設定

API キー (1) 情報

アクション ▼

🔍 API キーの検索

| | 名前 ▼ | ステータス ▼ | ID ▼ | API キー |
|-----------------------|---|---------|------------|--|
| <input type="radio"/> | CdkSam-Sampl-UUADcBWY9MhG | 🟢 アクティブ | pdxcrh84ld |  |

⚠️ このAPIキーはこの後使用します ⚠️



APIエンドポイントのチェック



CdkSampleStack



Deployment time: 1.69s

Outputs:

CdkSampleStack.SampleApiEndpoint -
<https://xxxxxxxx.execute-api.ap-northeast-1.amazonaws.com/prod/>

Stack ARN:

arn:aws:cloudformation:ap-northeast-1:123456789012:stack/CdkSampleStack/xxxxxxxx-xxxx-xxxxxxx



動作チェック

```
> curl -X POST '<ここにエンドポイントを入れる>' \  
  -H 'Content-Type: application/json' \  
  -H 'x-api-key: <ここにAPIキーを入れる>' \  
  -d '{"name":"TARO", "age":55}'
```

```
> curl -X GET '<ここにエンドポイントを入れる>' \  
  -H 'x-api-key: <ここにAPIキーを入れる>'
```

EX) Lambdaのアーキテクチャを Graviton(ARM64)に 1/2

```
// Lambda関数を作成する (GET用)
const getLambda = new lambda.Function(this, 'GetLambda', {
  runtime: lambda.Runtime.PYTHON_3_9,
  handler: 'get.handler',
  architecture: lambda.Architecture.ARM_64, // ここを追加
  code: lambda.Code.fromAsset('lambda'),
  environment: {
    TABLE_NAME: table.tableName,
  },
});
```

GravitonはAmazonが独自設計したARMベースのプロセッサ
高いパフォーマンスと低コスト、低消費電力が特徴

EX) Lambdaのアーキテクチャを Graviton(ARM64)に 2/2

```
> npx cdk diff // 手元のCFnとデプロイされたCFnの差分を表示
```

Resources

```
[~] AWS::Lambda::Function GetLambda GetLambdaXXXXXX
```

```
└─ [+] Architectures
```

```
    └─ ["arm64"]
```

```
[~] AWS::Lambda::Function PostLambda PostLambdaXXXXXX
```

```
└─ [+] Architectures
```

```
    └─ ["arm64"]
```

CDKハンズオン環境の削除

```
> npx cdk destroy
```

削除していいか聞かれますので y で返答してください。

SaaS ハンズオン環境構築



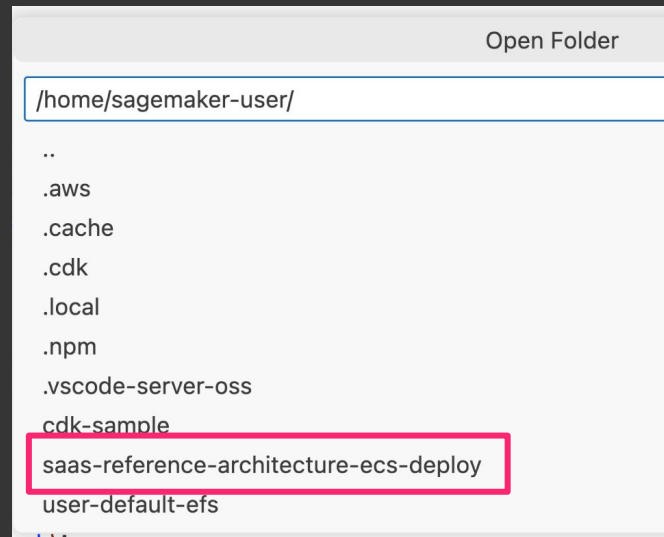
CDKハンズオンのコードを用意

```
> git clone https://github.com/papi-tokei/saas-reference-architecture-ecs-deploy.git
```

Code Editor上でフォルダを開く →

```
> cd saas-reference-architecture-ecs-deploy
```

```
> npm i
```



SaaSハンズオン環境構築 1/4

```
lib/saas-reference-architecture-ecs-deploy-stack.ts L28-34
  build: {
    commands: [
      'cd scripts',
      './build-application.sh',
      './install.sh <ここにメールアドレスを入れる >'
    ],
  },
```

⚠ 必ず受信できるメールアドレスを入力してください ⚠

SaaSハンズオン環境構築 2/4

```
> npx cdk deploy
```

この内容でデプロイしてOKか聞かれますので y で返答してください。

デプロイ中にまとめ

座学編から一部抜粋

オンボーディング/オフボーディング

オンボーディングとは、新規顧客がサービスを使うことができるようになるまでの一連の作業のことです。

オンボーディングはできる限り自動化できることを目指す必要があります。

SaaSハンズオン環境構築 3/4

デベロッパー用ツール

CodeBuild

ソース • CodeCommit

アーティファクト • CodeArtifact

ビルド • CodeBuild

開始方法

ビルドプロジェクト

ビルドプロジェクト

設定

ビルド履歴

レポートグループ

レポート履歴

コンピューティングフリー

新しい

アカウントメトリクス

関連する統合

Jenkins

GitHub アクション

新しい

GitLab ランナー

新しい

デベロッパー用ツール > CodeBuild > ビルドプロジェクト > SaaSDeploymentBuildProject5-EQq3tXxF8342

SaaSDeploymentBuildProject5-EQq3tXxF8342

アクション トリガーの作成 編集 クローンを作成 デバッグビルド 上書きでビルドを開始する **ビルドを開始**

設定

ソースプロバイダ

プライマリリポジトリ

アーティファクトのアップロード場所

サービスロール

GitHub

papi-tokei/saas-reference-architecture-ecs

-

パブリックビルド

無効

ビルド履歴 バッチ履歴 プロジェクトの詳細 ビルドのトリガー メトリクス

ビルド履歴

ビルドの停止 アーティファクトの表示 ログの表示 ビルドの削除 ビルドの再試行

ビルドの実行

ステータス

ビルド番号

ソースバージョン

送信者

期間

完了済み

| | | | | | | | |
|--------------------------|---|----|---|------|--------|-------------|------|
| <input type="checkbox"/> | SaaSDeploymentBuildProject5-EQq3tXxF8342:caf32f41-335c-4471-bb23-df478c0f3c3d | 成功 | 1 | main | hiroya | 37 分間 12 秒間 | 7 分前 |
|--------------------------|---|----|---|------|--------|-------------|------|

SaaSハンズオン環境構築 4/4

aws

サービス

検索

[Alt+S]

デベロッパー用ツール

CodeBuild

ソース • CodeCommit

アーティファクト

CodeArtifact

ビルド • CodeBuild

開始方法

ビルドプロジェクト

ビルド履歴

レポートグループ

レポート履歴

コンピューティングフリー

ト

新しい

アカウントメトリクス

デベロッパー用ツール > CodeBuild > ビルドプロジェクト

ビルドプロジェクト 情報

リフレッシュ

アクション ▼

トリガーの作成

詳細を表示

検索 SaaSDeploymentBuildProject

| | 名前 ▼ | ソースプロバイダ | リポジトリ | 最新のビルドステータス | 説 |
|--|--|----------|--|-------------|---|
| | SaaSDeploymentBuildProject5-EQq3tXxF8342 | GitHub | papi-tokei/saas-reference-architecture-ecs | 成功 | - |



ありがとうございました！

コメントやワイワイは
#jawsug
でお待ちしております



[付録] cdk deployの裏で何が動いているのか

