

AWS Builders Online Series

T5-2

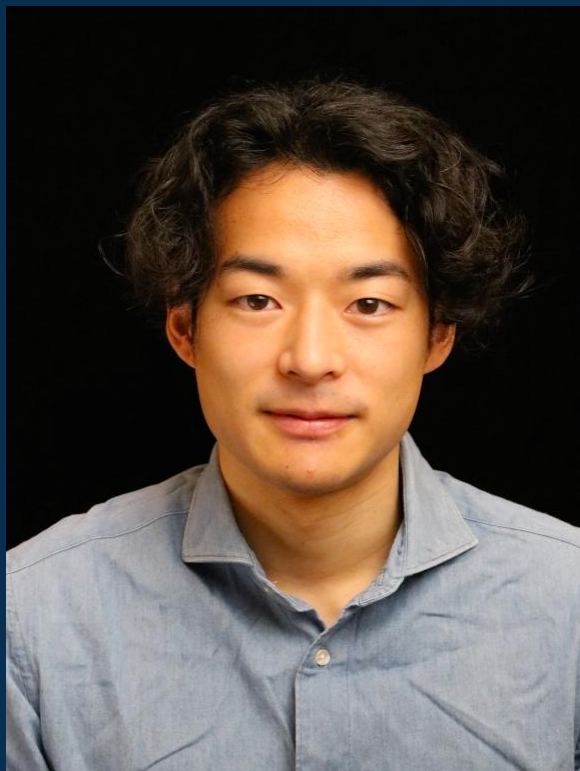
サーバーレスアプリケーションを 効率よく開発しよう！ AWS SAM とローカル開発

松本 侑也

アマゾン ウェブ サービス ジャパン合同会社
パブリックセクター 技術統括本部 ソリューションアーキテクト



自己紹介



松本 侑也（まつもと ゆうや）

パブリックセクター 技術統括本部
ソリューションアーキテクト

自治体のお客様のクラウド活用支援を担当

好きなAWSサービス

- ・ AWS SAM

本セッションで取り上げるAWSサービス・機能

本セッションで取り上げる AWS のサービス・機能は以下の通りです。詳細は、AWS クラウドサービス活用資料集 またはAWSドキュメント よりご確認ください。

- AWS SAM
- Amazon API Gateway
- AWS Lambda
- AWS Application Composer

AWS クラウドサービス活用資料集

アマゾン ウェブ サービスの公式イベントのアーカイブおよびオンデマンドコンテンツの動画や資料がご利用いただけます。

[AWS イベントお申込 »](#)

[AWS 初心者向け »](#)

[サービス別資料 »](#)

[ハンズオン資料 »](#)

<https://aws.amazon.com/jp/events/aws-event-resource/>



本セッションの対象になる方

- これからサーバーレスアプリケーションの開発を始めようとしている方
- サーバーレスアプリケーションの開発を始めていて、効率のいい開発の方法を知りたい方

サーバーレスの開発って
何から始めるのか？



もっと開発を
効率化したい

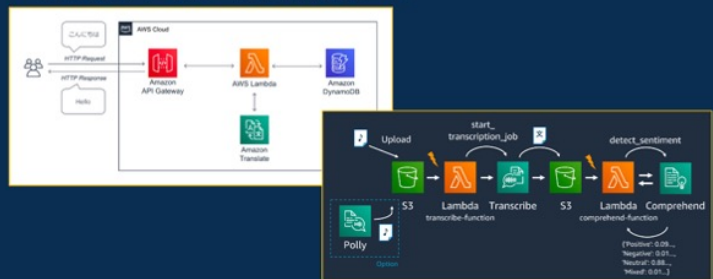


関連セッションについて

- 本セッションは、3つの関連セッションで構成しています。
- 本セッションでは、AWS Lambda/Amazon API Gatewayなどのサービス説明は行いません。サーバーレスサービスの基礎について知りたい場合、「具体的なユースケースから学ぶ、サーバーレスアプリケーションの活用方法」をご覧ください。

本セッションで学べること

- サーバーレスとは何なのか
- 具体的なユースケースと実装イメージ



具体的なユースケースから学ぶ、サーバーレスアプリケーションの活用方法

本セッションで学べること

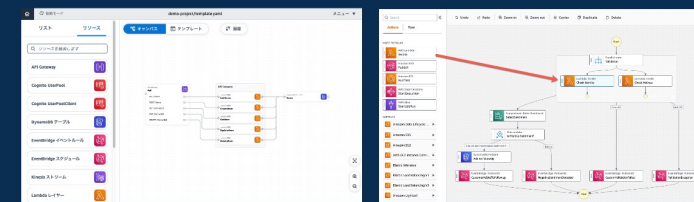
- AWS SAMによるサーバーレス開発の流れ
- サーバーレスのローカル開発の方法



サーバーレスアプリケーションを効率よく開発しよう！
AWS SAMとローカル開発

本セッションで学べること

- ビジュアルツールによるサーバーレス開発
- サーバーレスワークフロー構築の方法



ブラウザで開発するサーバーレスアプリケーション

アジェンダ

- サーバーレスアプリケーションを開発する際の課題
- AWS SAMを使ったサーバーレスアプリケーションの開発
- Demo

サーバーレスアプリケーションを 開発する際の課題

代表的なサーバーレスアプリケーションの特性

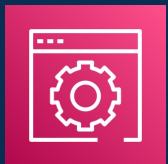
何らかの「イベント」に応じてLambda関数を起動し処理を行う性質を持つ



例) サーバーレスサービスでAPIを作成



マネジメントコンソールを使って開発する場合



AWS Management
Console

```
コードソース 情報 アップロード元 ▼
File Edit Find View Go Tools Window Test Deploy Changes not deployed
Go to Anything (⌘ P) lambda_function ×
Environment
sample-lambda
lambda_function.py
1 import json
2
3 def lambda_handler(event, context):
4     print('Hello World!!')
5     return {
6         'statusCode': 200,
7         'body': json.dumps('Hello from Lambda!')}
8
9
```

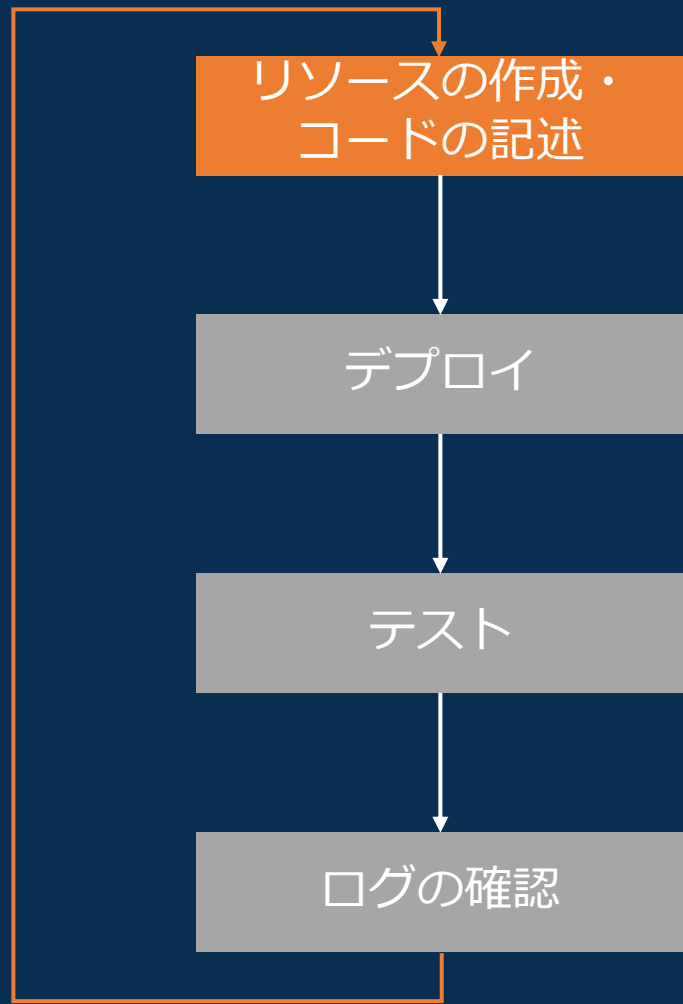
いいところ

- 直感的
- 手軽で始めやすい

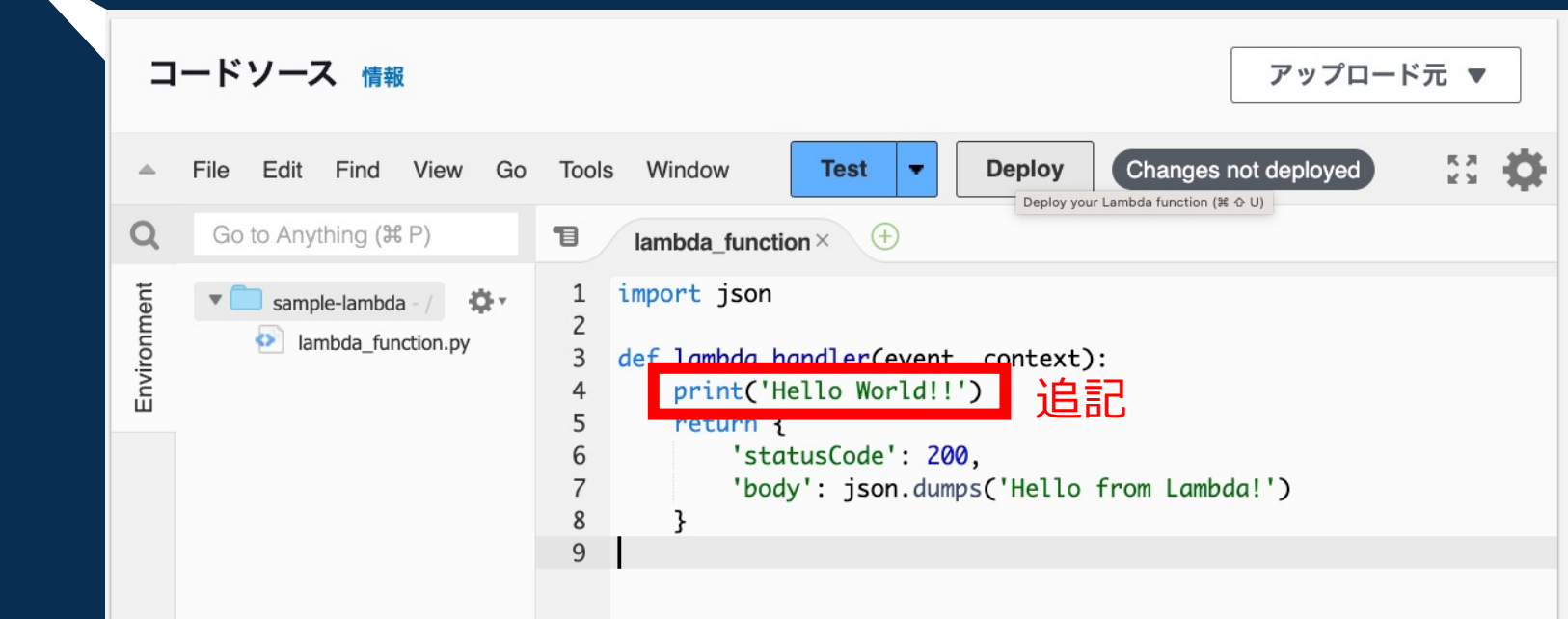
課題

- Lambda関数の数が増えると管理が煩雑に
- 環境を再現するには、手順書のようなものを作らないといけない
- 開発・検証・本番環境を再現する際にミスが生じる

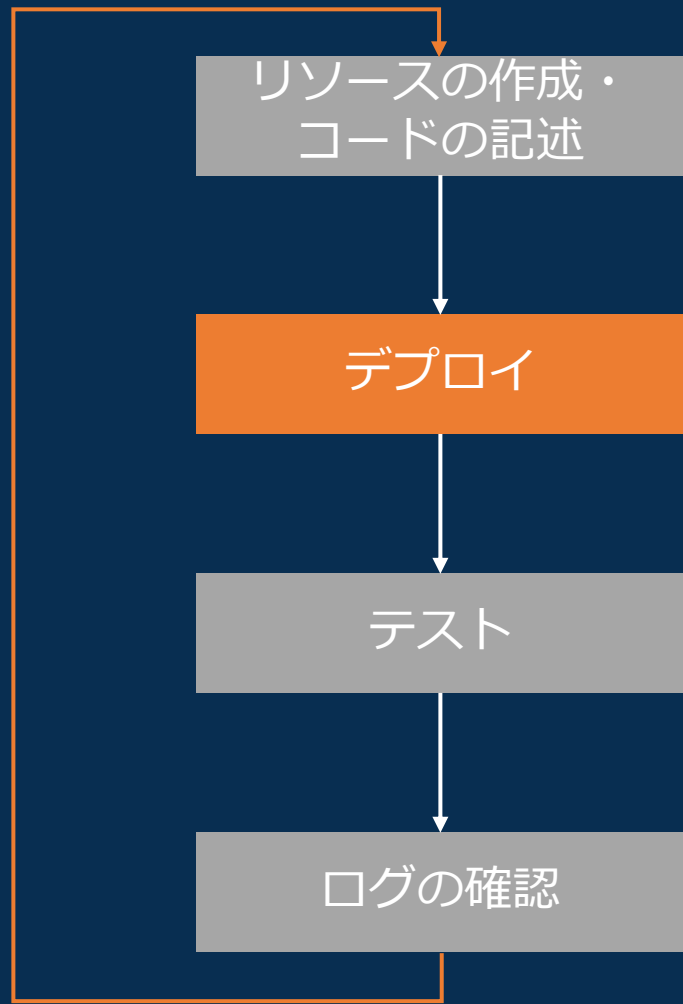
マネジメントコンソールでの開発サイクル



繰り返し



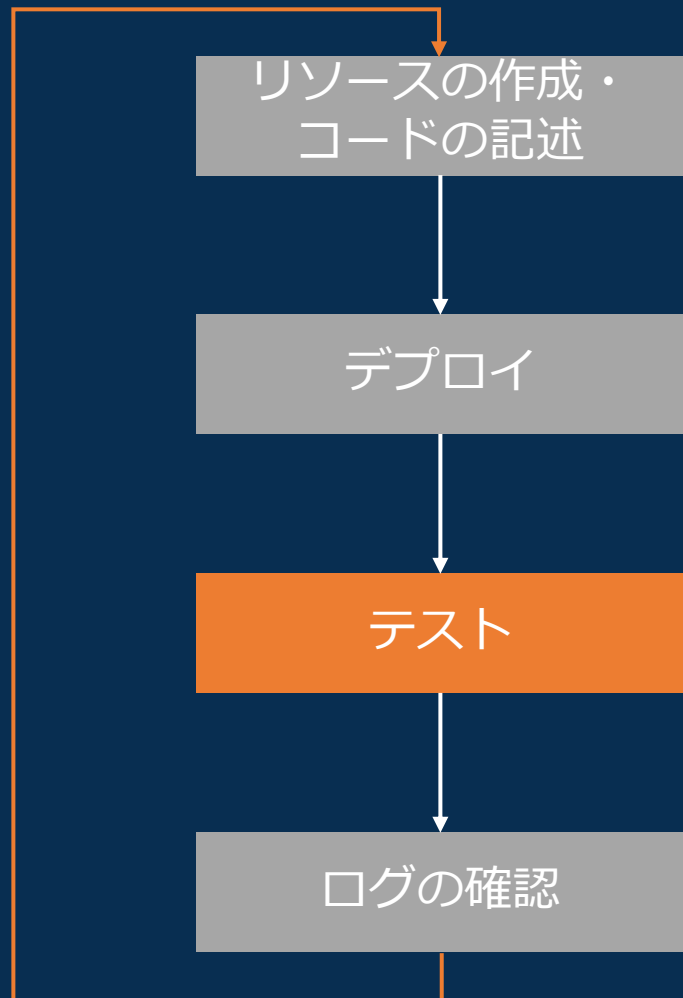
マネジメントコンソールでの開発サイクル



繰り返し



マネジメントコンソールでの開発サイクル



繰り返し

コード | **テスト** | モニタリング | 設定 | エイリアス | バージョン

Lambda関数の実行

テストイベント 情報 保存 **テスト**

イベントを保存せずに関数を呼び出すには、JSON イベントを設定し、[テスト] を選択します。

イベントアクションをテスト

新しいイベントを作成 保存されたイベントを編集

イベント名

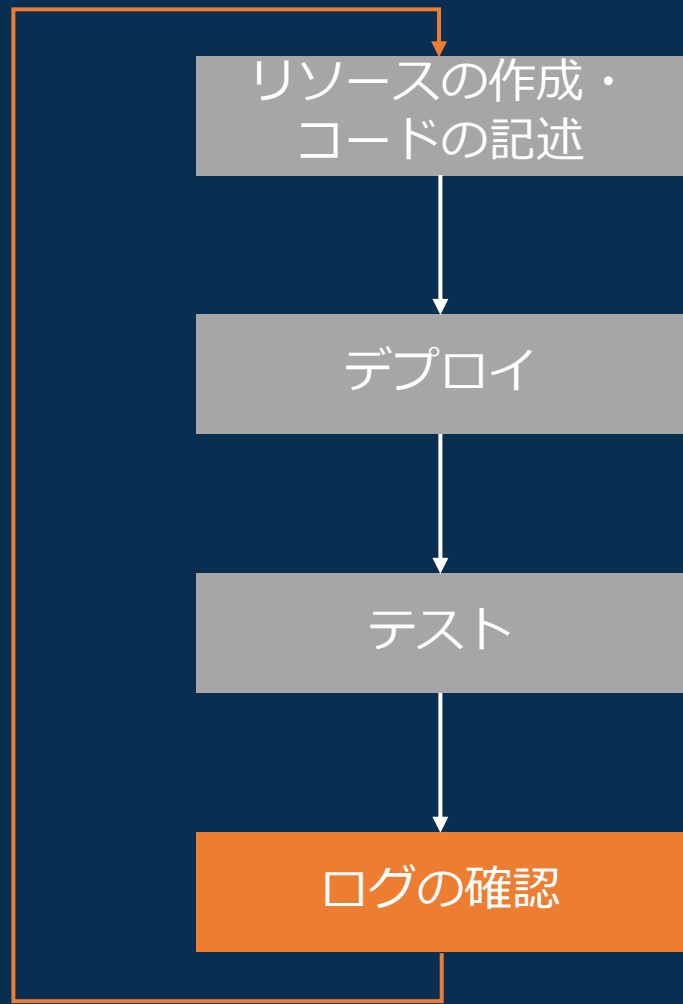
最大 25 文字 (文字、数字、ドット、ハイフン、アンダースコアのみ)。

イベント共有の設定

プライベート
このイベントは、Lambda コンソールで、イベント作成者のみが利用可能です。合計 10 個を設定できます。[詳細はこちら](#)

共有可能
このイベントは、共有可能なイベントにアクセスして使用するための許可が付与されている、同じアカウント内の IAM ユーザーが利用可能です。[詳細はこちら](#)

マネジメントコンソールでの開発サイクル



繰り返し

✓ 実行中の関数: 成功 (ログ [🔗](#))

▼ 詳細

以下の領域は、実行ログの最後の 4 KB を示しています。

```
{
  "statusCode": 200,
  "body": "\"Hello from Lambda!\""
}
```

概要

コード SHA-256 61FX4e/uhwABuKvJ2mdxoKNlqCSd7vs/pJlA82gSWRg =	リクエスト ID 30f13f2d-05f1-450c-b105-dc33d06e4ba3
初期所要時間 116.79 ms	所要時間 1.95 ms
課金期間 2 ms	設定済みリソース 128 MB
使用中の最大メモリ 35 MB	

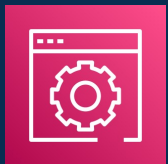
ログ出力

以下のアクションでは、コード内のロギング呼び出しを示しています。 [ここをクリックし](#)、 に対応する **変更結果確認** できます。

```
START RequestId: 30f13f2d-05f1-450c-b105-dc33d06e4ba3 Version: $LATEST
Hello World!!
END RequestId: 30f13f2d-05f1-450c-b105-dc33d06e4ba3
REPORT RequestId: 30f13f2d-05f1-450c-b105-dc33d06e4ba3 Duration: 1.95 ms Billed Duration: 2 ms
Memory Size: 128 MB Max Memory Used: 35 MB Init Duration: 116.79 ms
```

マネジメントコンソールを使って開発する場合

再掲



AWS Management
Console

いいところ

- 直感的
- 手軽で始めやすい

```
コードソース 情報 アップロード元 ▼
File Edit Find View Go Tools Window Test Deploy Changes not deployed
Go to Anything (%P)
Environment
sample-lambda
lambda_function.py
1 import json
2
3 def lambda_handler(event, context):
4     print('Hello World!!')
5     return {
6         'statusCode': 200,
7         'body': json.dumps('Hello from Lambda!')}
8
9
```

課題

- Lambda関数の数が増えると管理が煩雑に
- 環境を再現するには、手順書のようなものを作らないといけない
- 開発・検証・本番環境を再現する際にミスが生じる



AWS SAM

AWS SAMを使った サーバーレスアプリケーション の開発

AWS SAMとは

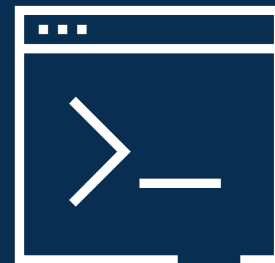
AWS上でサーバーレスアプリケーションを構築することに特化したツール



大きくSAMテンプレートとSAM CLIで構成される



SAMテンプレート



SAM CLI

- JSON/YAMLで記述
- AWS LambdaなどのAWSリソースを定義
- ターミナルから実行できるSAM関連のコマンド
- SAMテンプレートで定義されたAWSリソースのデプロイなど

SAMテンプレート

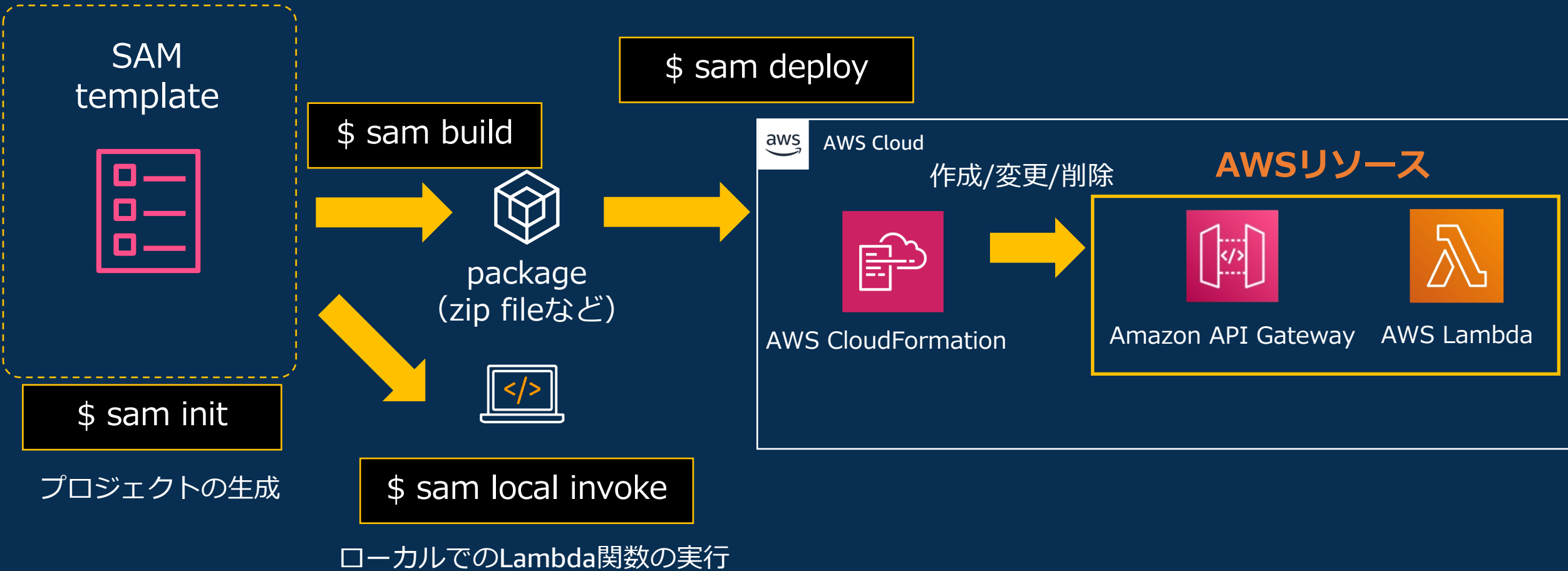
```
AWSTemplateFormatVersion: '2010-09-09'  
Transform: 'AWS::Serverless-2016-10-31'  
Description: HelloWorld  
Resources:  
  HelloWorld:  
    Type: 'AWS::Serverless::Function'  
    Properties:  
      Handler: index.handler  
      Runtime: python3.8  
      CodeUri: src/handlers/func1  
      Description: HelloWorld  
      MemorySize: 128  
      Timeout: 3  
    Events:  
      HelloApi:  
        Type: Api  
        Properties:  
          Path: /hello  
          Method: get
```

(yamlの例)



SAM CLI

ターミナルから実行できるSAM関連のコマンド

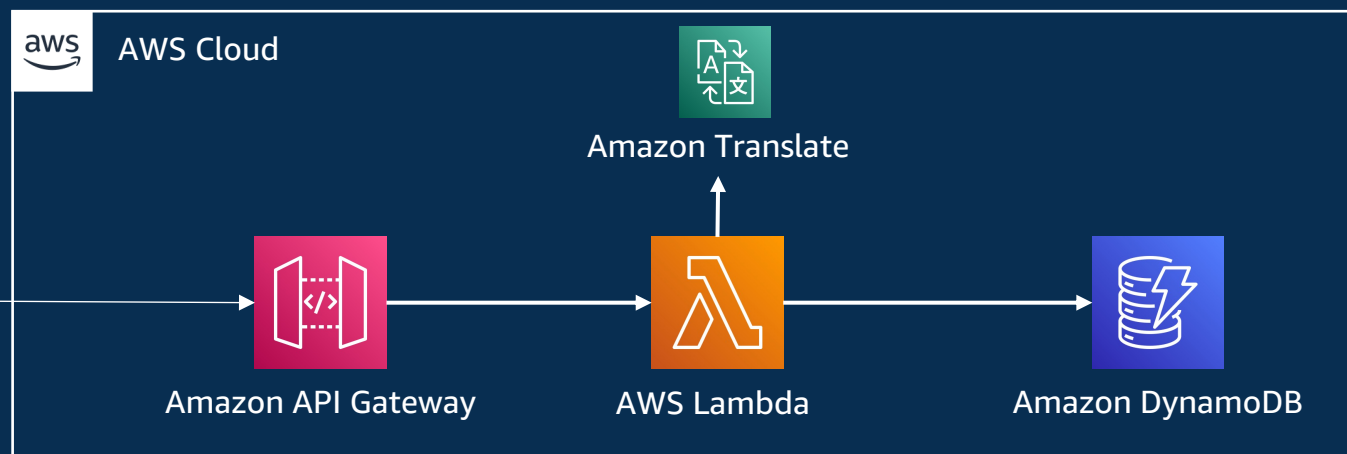


Demoで作成するサンプルAPI



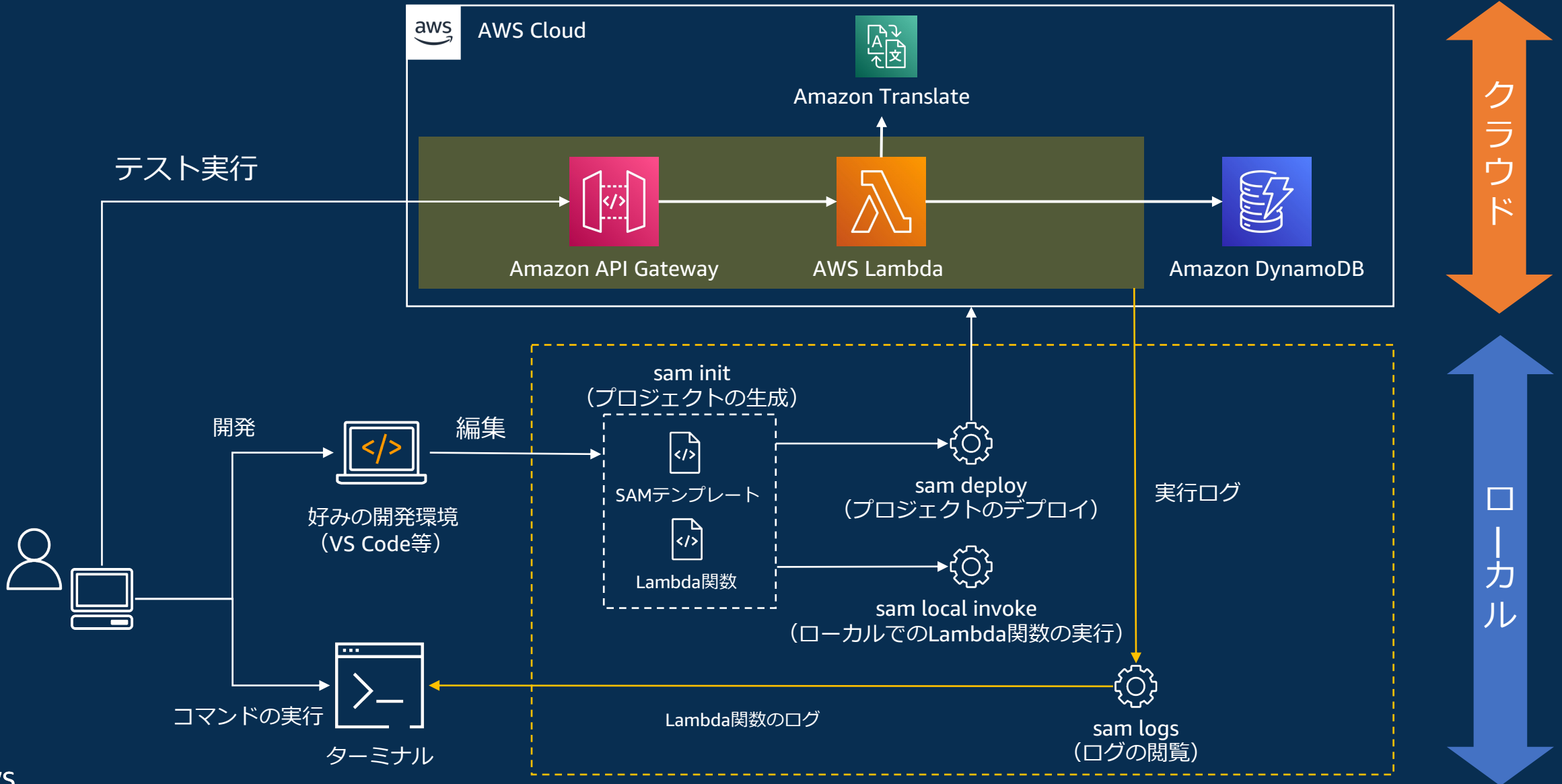
SAMテンプレート

```
AWSTemplateFormatVersion: '2010-09-09'  
Transform: 'AWS::Serverless-2016-10-31'  
Description: HelloWorld  
Resources:  
  HelloWorld:  
    Type: 'AWS::Serverless::Function'  
    Properties:  
      Handler: index.handler  
      Runtime: python3.8  
      CodeUri: src/handlers/func1  
      Description: HelloWorld  
      MemorySize: 128  
      Timeout: 3  
    Events:  
      HelloApi:  
        Type: Api  
        Properties:  
          Path: /hello  
          Method: get
```



http://xxxx/hello?input_text=こんにちは!
→Hello!

AWS SAMを使った開発の流れ

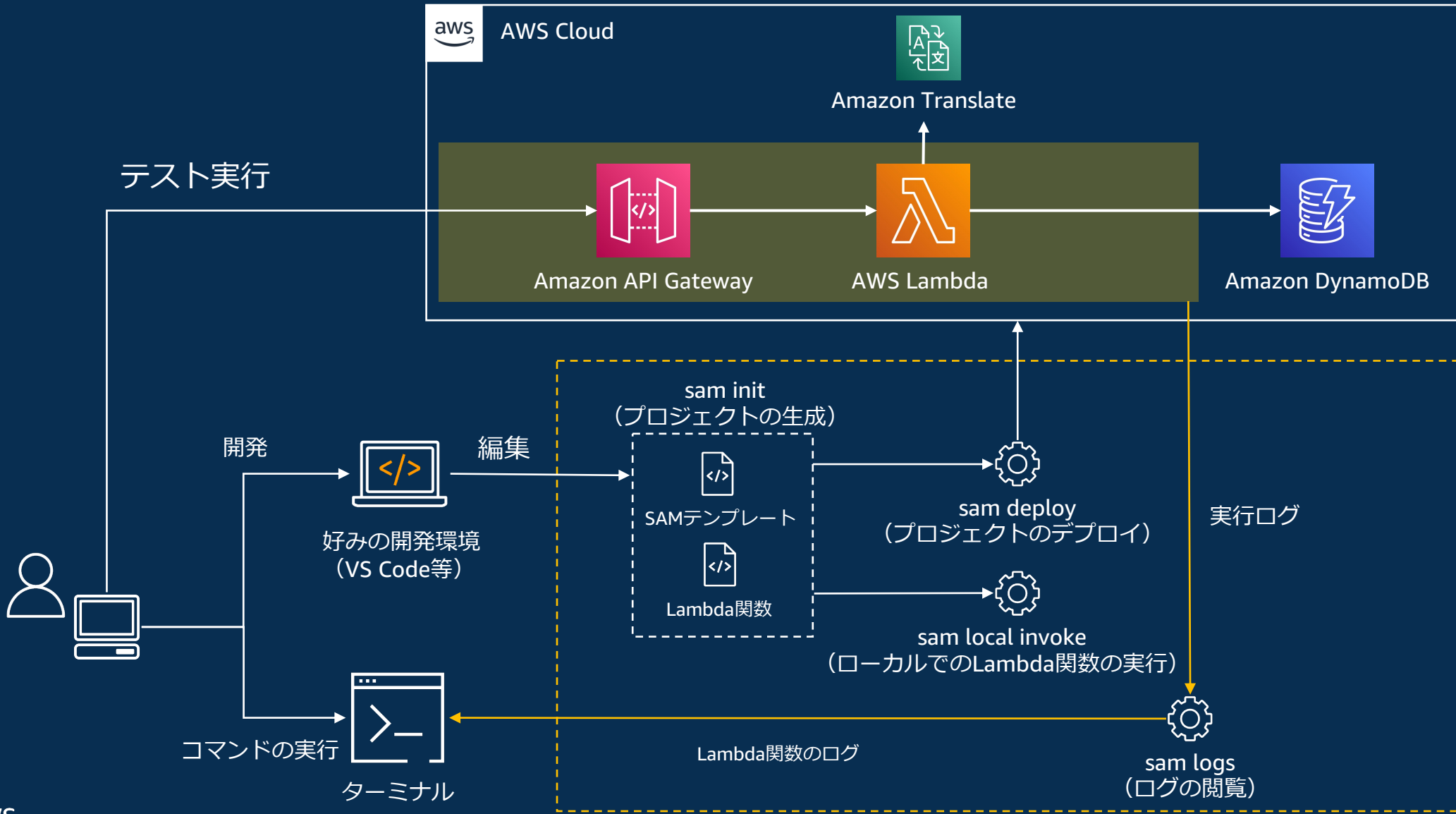


Demo



AWS SAMを使った開発の流れ

再掲



クラウド

ローカル

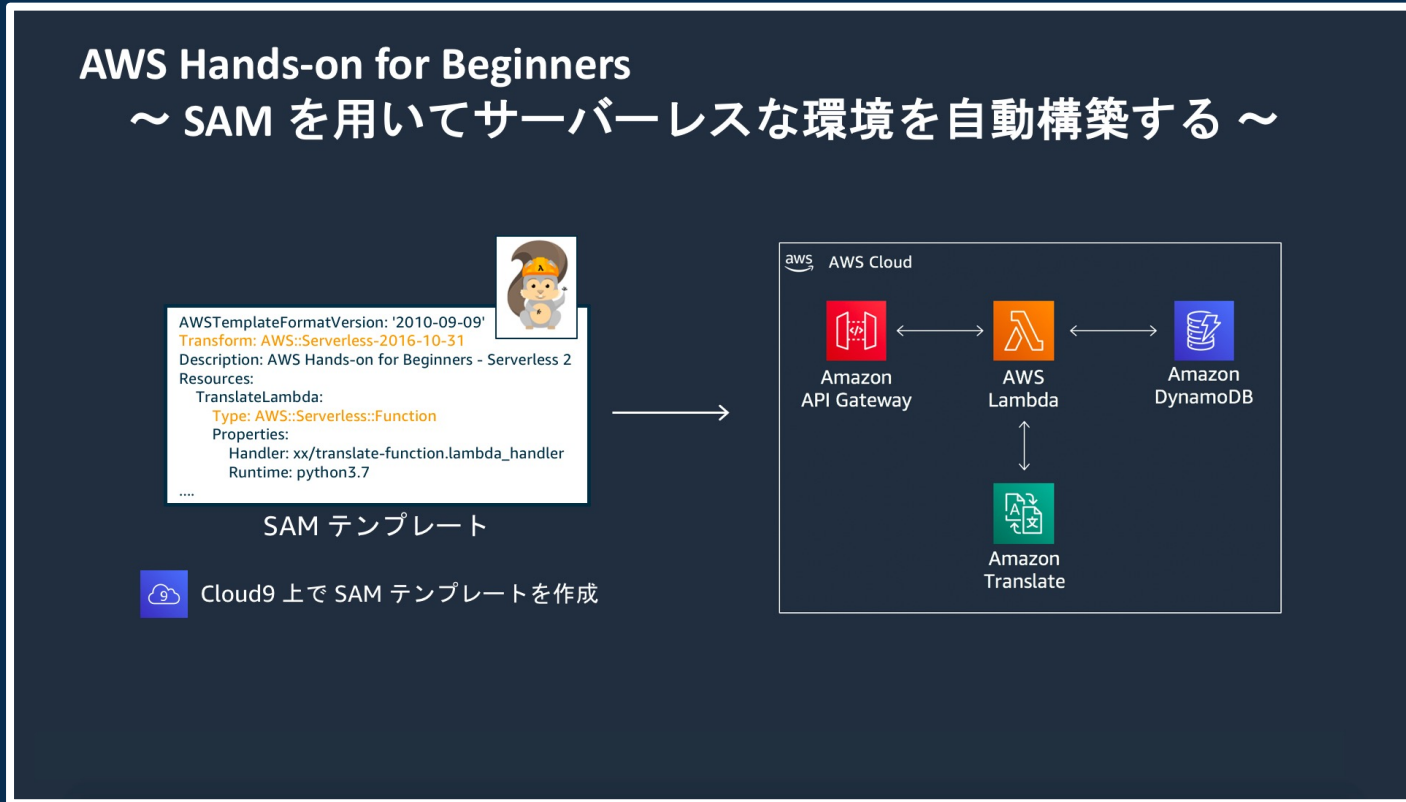


まとめ

- AWS SAM はサーバーレスアプリケーションの管理・開発を効率的にするツール
- SAM テンプレートを使うとサーバーレスアプリケーションの管理が容易に
- SAM CLI を駆使すると、サーバーレスアプリケーション開発がローカルから効率的に行える

今後に向けて

本セッションでご紹介した内容と類似したものを
具体的な画面とデモを見ながら進めることができるハンズオンです



<https://pages.awscloud.com/JAPAN-event-OE-Hands-on-for-Beginners-Serverless-2-2022-reg-event.html>

サーバーレス自己学習ガイド



<https://aws.amazon.com/jp/serverless/patterns/redirect-serverless-steps>

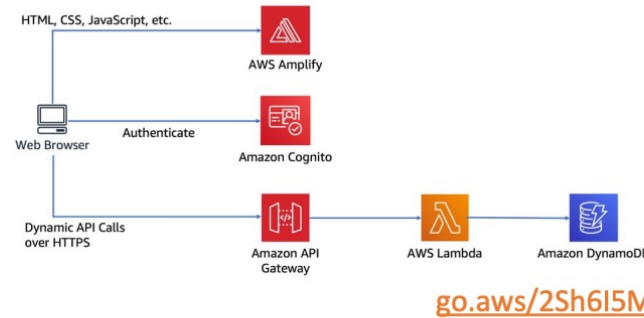
- ハンズオンから始まる6つのステップ
 - Hands-on for Beginnersも包含
- ご自分のペースで主要ポイントを学べます
- 開発作業の諸処で役立つサーバーレス技術情報サイトもご紹介



サーバーレスの始め方 (1/2)

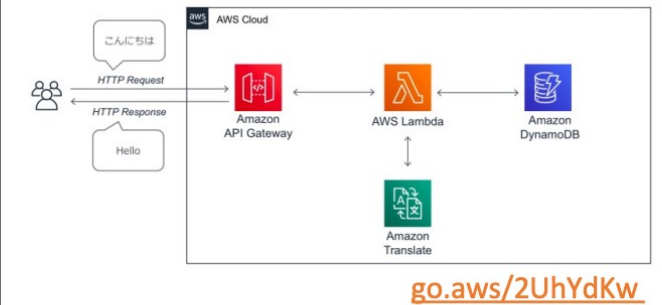
1 最初のトライ: サーバーの準備も実行環境構築も不要、いきなりアプリ開発を体験!

最初のサーバーレスWebアプリ: 手順に沿えば、多くのサーバーレスサービスに触れながら、Webアプリが作れます。



「動的 Web / モバイルバックエンド」パターン

5-10分 x 11本のハンズオンで、サーバーレスな機能APIを作りながら、少しずつサービス自体の理解を深めていきます。

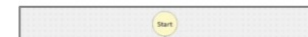


「機能API」パターン

処理フローにチャレンジ

2 処理フローや例外処理の理解

新しいビジュアルエディタでアプリケーションの処理フローを視覚的にデザインできます。



エラー処理を含む処理の流れをフローとして定義すれば、アプリケーション全体の可読性を高め、保守を容易にします。



AWS TRAINING & CERTIFICATION

600+ ある AWS Skill Builder の無料デジタルコースで学ぼう

30 以上の AWS ソリューションの中から、自分にもっとも関係のあるクラウドスキルとサービスにフォーカスし、自習用のデジタル学習プランとランプアップガイドで学ぶことができます。

自分に合ったスキルアップ方法で学ぼう

[EXPLORE.SKILLBUILDER.AWS](https://explore.skillbuilder.aws) »



あなたのクラウドスキルを AWS 認定で証明しよう

業界で認められた資格を取得して、スキルアップの一步を踏み出しましょう。AWS Certified Cloud Practitioner の取得方法と、準備に役立つ AWS のリソースをご覧ください。

[受験準備のためのリソースにアクセスしよう](#) »



AWS Builders Online Series にご参加いただきありがとうございます

楽しんでいただけましたか? ぜひアンケートにご協力ください。
本日のイベントに関するご意見/ご感想や今後のイベントについてのご希望や改善のご提案などがございましたら、ぜひお聞かせください。



aws-apj-marketing@amazon.com



twitter.com/awscloud_jp



[facebook.com/600986860012140](https://www.facebook.com/600986860012140)



<https://www.youtube.com/user/AmazonWebServicesJP>



<https://www.linkedin.com/showcase/aws-careers/>



[twitch.tv/aws](https://www.twitch.tv/aws)

Thank you!

