イチから始めるサーバーレスアプリ開発

サーバーレスアプリケーション開発

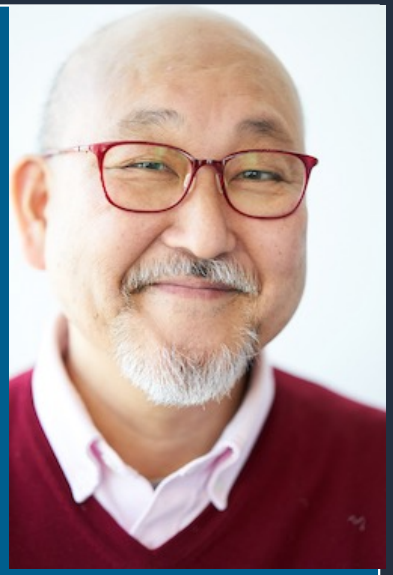
~ AWS Step Functions Workflow Studioも使ってみよう ~

Atsushi Fukui
Senior Solutions Architect, Serverless Specialist

Amazon Web Services Japan K.K.
2021/09/09

自己紹介

- **名前**
  - 福井 厚（ふくい あつし）fatsushi@
- **所属**
  - アマゾン ウェブ サービス ジャパン株式会社
  - 技術統括本部レディネスソリューション本部
  - シニアソリューションアーキテクト
サーバーレス スペシャリスト
- **関心領域**
  - ソフトウェア アーキテクチャ、オブジェクト指向設計、アジャイル開発
- **好きなAWSサービス**
  - サーバーレステクノロジー全般、 AWS Code シリーズ、AWS Amplify



aws

Agenda

- 開発者のためのAWS Developer Tools
- AWS Lambda関数のビルドとデプロイ
- Serverless Application Model(SAM)
- AWS Toolkit + SAM
- SAM CLI
- CI/CD for Serverless / Serverless Application Pipeline
- AWS Step Functions Workflow Studio
- まとめ

aws

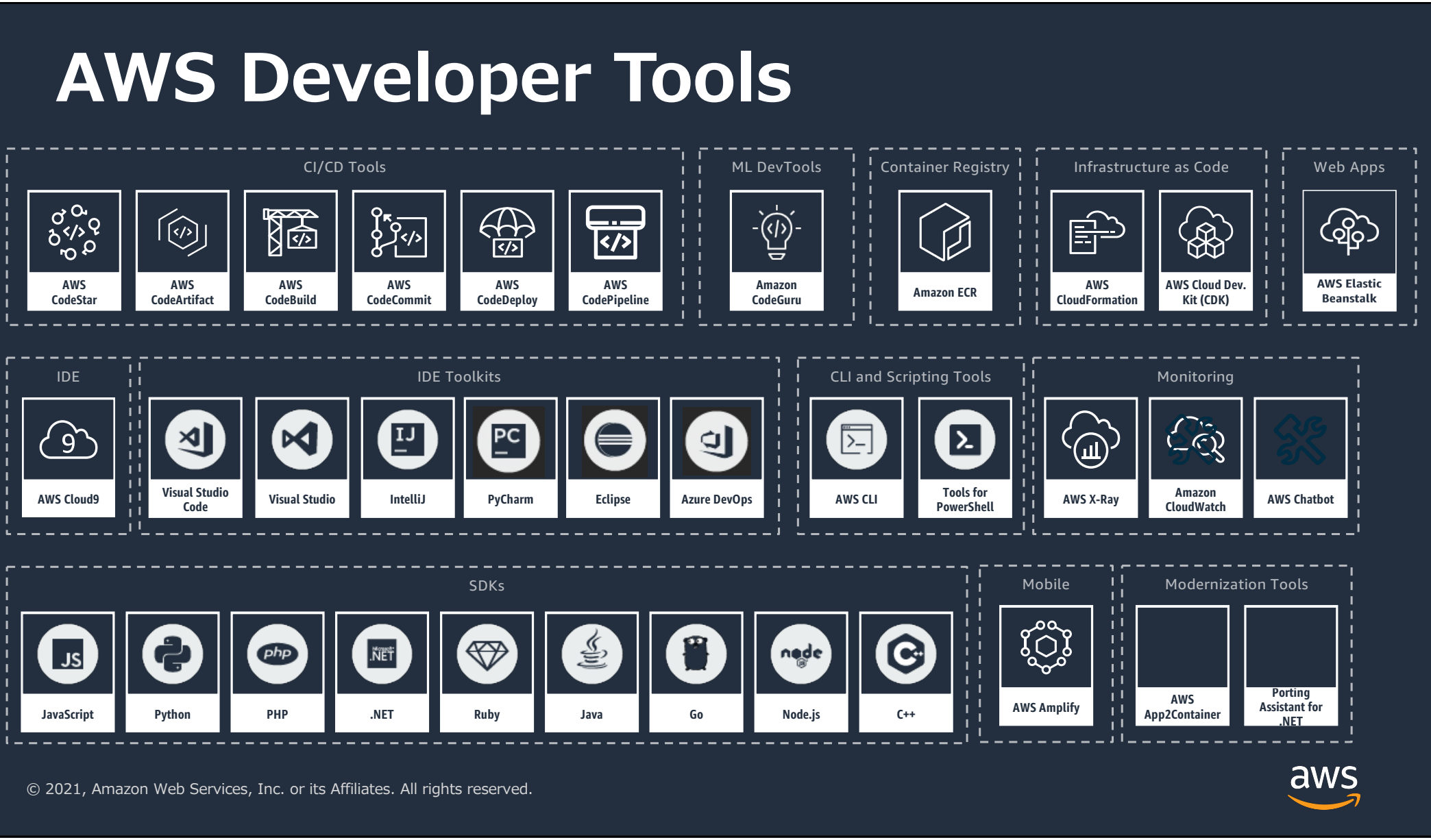
開発者のための
AWS Developer Tools

AWS Developer Tools

- **モダンアプリケーション開発をサポート:** 開発チームに対してモダンなソフトウェアやベストプラクティスに従った高品質なソフトウェアをより高速にデリバリーできるように支援
- **学習曲線を短縮:** 開発者が慣れ親しんだ環境、言語、ツールをサポート
- **開発者とチームを繋げる:** ソフトウェア開発におけるチームのコラボレーションを支援

aws

AWS Developer Tools



AWS Lambda関数のビルドとデプロイ

AWS Lambda関数の作成手順

- Lambda関数のコードを記述
- Lambda関数をビルドしてパッケージ化
  - Zip形式とコンテナ形式が選択できる
- ビルドしたパッケージをAWS環境にデプロイ
- 指定したイベントでLambda関数が実行される

aws

Lambda関数実装例

```
def lambda_handler(event, context):
    """ Unit TestしやすいLambda関数のサンプル
    """
    id = get_parameter(event, "id")
    is_takeout = get_parameter(event, "is_takeout")

    item = get_item(id)

    tax = calc_tax(item, is_takeout)

    return {
        "statusCode": 200,
        "body": json.dumps({
            'id': item['id'],
            'item_name': item['item_name'],
            'price': item['price'],
            'tax': tax
        }),
    }
```


aws

Lambda関数実装例

```
def lambda_handler(event, context):
    """ Unit TestしやすいLambda関数のサンプル
    """
    id = get_parameter(event, "id")
    is_takeout = get_parameter(event, "is_takeout")

    item = get_item(id)


    tax = calc_tax(item, is_takeout)

    return {
        "statusCode": 200,
        "body": json.dumps({
            'id': item['id'],
            'item_name': item['item_name'],
            'price': item['price'],
            'tax': tax
        }),
    }
```

Lambdaを呼び出す側からパラメータを取得する処理

関数が実行するロジックの記述

呼び出し側に結果を返す処理

では、どうやってビルドしてデプロイするのか？

Serverless Application Model (SAM)

SAM：コードによる実行環境の構築

- マネージメントコンソールやAWS CLIでも可能だが…
  - マネージメントコンソールによる関数作成の課題
    - 手元のソースコードとのマッピングができない
    - 手作業によるミスの可能性
    - 関数の増加に伴い管理が困難に
- Infrastructure as Codeによる自動化のメリット
  - 本番環境と同じ環境を検証環境に構築できる
  - リポジトリによるバージョン管理が可能
  - ミスなく迅速にリリース可能

aws

LambdaのデプロイにはAWS Serverless Application Model (SAM)が利用可能

- AWS上のサーバーレスアプリケーションを構築するためのオープンソース フレームワーク
- 関数、API、データベース、イベントソースマッピングを表現する簡易な文法を使用
- デプロイ時にSAMの文法をAWS CloudFormationの文法に変換、展開
- すべてのAWS CloudFormationリソースタイプをサポート

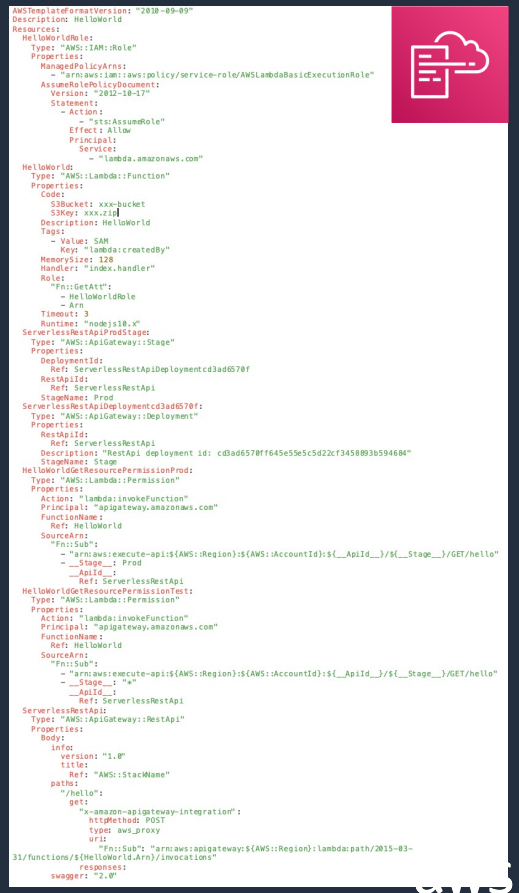
https://aws.amazon.com/serverless/sam/

aws

SAMでより簡潔にサーバーレス アプリを定義できる

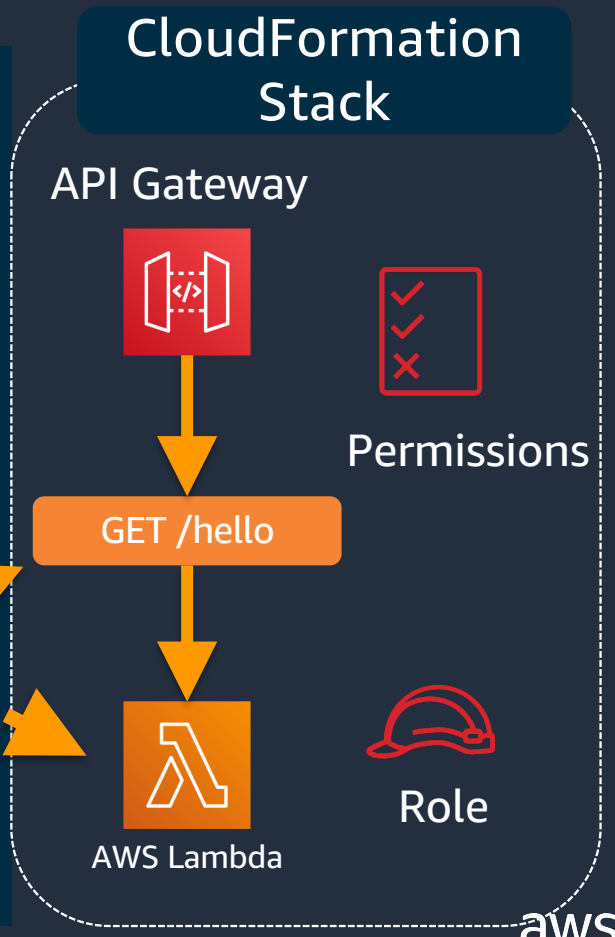

```yaml
AWSTemplateFormatVersion: '2010-09-09'
Transform: 'AWS::Serverless-2016-10-31'
Description: HelloWorld
Resources:
  HelloWorld:
    Type: 'AWS::Serverless::Function'
    Properties:
      Handler: index.handler
      Runtime: python3.8
      CodeUri: src/handlers/func1
      Description: HelloWorld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

AWS CloudFormation



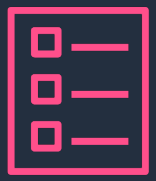
SAMでより簡潔にサーバーレス アプリを定義できる


```yaml
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
```



SAMによるServerless構築の流れ

SAM
template



ロード

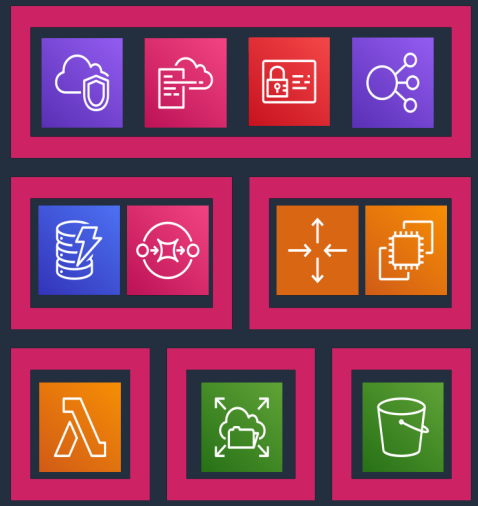
SAM

トランスパイル

CloudFormation

作成/変更/削除

CloudFormation
Stack



リソースの定義
パラメータの定義

スタックの作成/変更/削除
エラー検知とロールバック

AWSリソース

aws

SAMを使いやすく

IDE + AWS Toolkit

SAM CLI

AWS Toolkit

AWS Toolkit

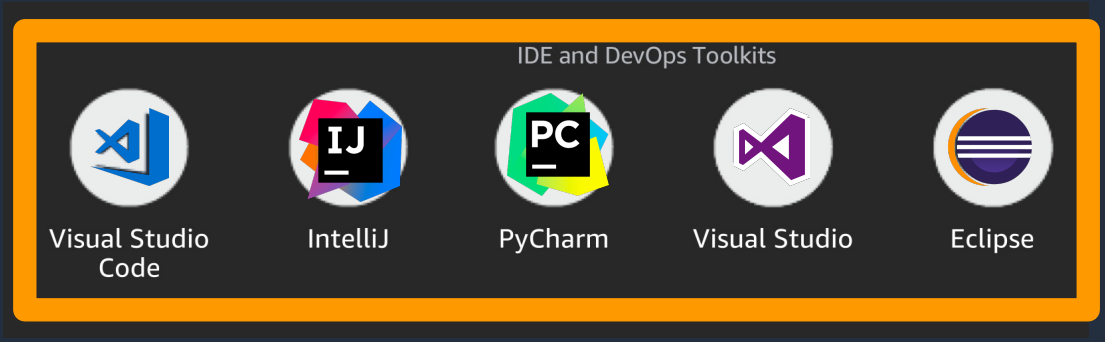
- オープンソースプラグイン
- AWS上でのアプリケーションの作成、デバッグ、デプロイを容易に
- ステップ実行によるデバッグ、およびIDE からのデプロイを含む、サーバーレスアプリケーションの統合開発環境を提供

AWS Toolkit

aws

AWS Toolkit

AWS Toolkit 対応 IDE



- 雛形からプロジェクトを作成
- ステップ実行などのデバッグ支援
- IDE/エディタからのデプロイ

AWS Toolkit for Visual Studio Code

aws

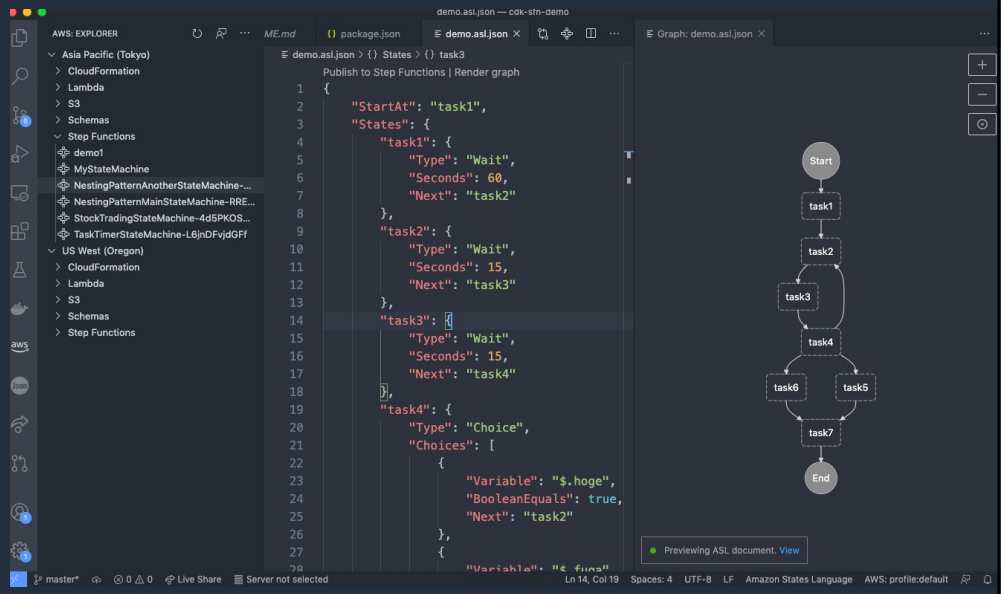
AWS Toolkit for Visual Studio Codeとは

- Visual Studio Code用のオープンソースプラグイン
  https://github.com/aws/aws-toolkit-vscode
- AWS上でのアプリケーションの作成、デバッグ、デプロイを容易に
- サーバーレスアプリケーションテンプレート
- サーバーレスアプリケーションをローカルでデバッグ
- IDEからサーバーレスアプリケーションをデプロイ
- 詳細はUser Guideを参照
  https://docs.aws.amazon.com/ja_jp/toolkit-for-vscode/latest/userguide/welcome.html

aws

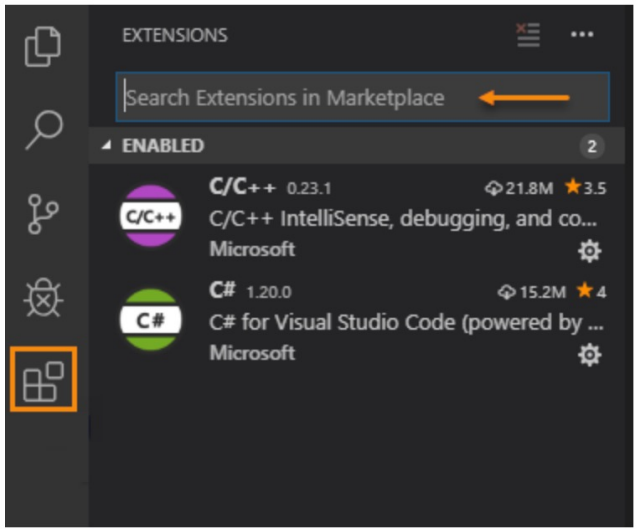
AWS Step Functionsをサポート New!

- AWS Toolkit を使用してStep Functions ステートマシンの開発を加速
- ステートマシンのビルド時にステートマシンを
  リアルタイムで視覚化
- Amazon States Language の言語
  パーサーがステートマシンの定義の
  構文解析、エラーを強調表示
- ステートマシンを作成、更新、
  実行可能



VS Code Toolkit のインストール

1. VS Code エディタを起動します。
2. VS Code エディタの側にある **[Activity Bar (アクティビティっバー)]** で、**[Extensions (拡張機能)]** アイコンを選択します。これにより、**[Extensions (拡張機能)]** ビューが開き、VS Code Marketplace にアクセスできます。



- https://github.com/aws/aws-toolkit-vscode

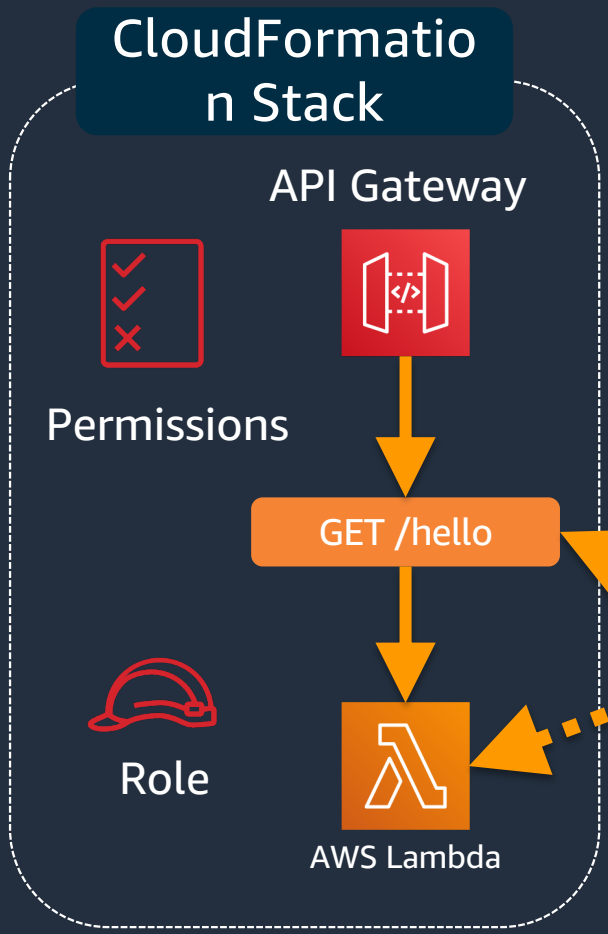
Demo — AWS Toolkit for Visual Studio Code

AWS Lambda関数のデプロイ デモ

- 新規SAMアプリケーションの作成
- SAMテンプレートの確認
- SAMアプリケーションのデプロイ
- デプロイ結果の確認

aws

Deployされた アーキテクチャとの対比




```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
Descript
    Sem

Resources:
    HelloWorldFunction:
        Type: AWS::Serverless::Function
        Properties:
            CodeUri: hello_world/
            Handler: app.lambda_handler
            Runtime: python3.8
            Events:
                HelloWorld:
                    Type: Api
                    Properties:
                        Path: /hello
                        Method: get
```

アーキテクチャとの対応

参考情報：Debug Panel からのデバッグ

- Running and debugging Lambda functions directly from code https://docs.aws.amazon.com/toolkit-for-vscode/latest/userguide/serverless-apps-run-debug-no-template.html

aws

SAM CLI

SAMを使いやすくしよう！

IDE + AWS Toolkit

SAM CLI

SAM CLI

- SAMをコマンドラインで実行
- スクリプトによる自動化が可能
- 自動化により手作業によるミスを防ぐ

AWS SAM CLI

1) アプリケーションのひな形を生成

```
$ sam init
```

2) ローカル環境でビルド

```
$ cd sam-app
$ sam build
```

3) ビルド後にAWS へデプロイ

```
$ sam deploy --guided
```

※sam deploy の際に必要な S3 バケットは自動生成

SAM CLI

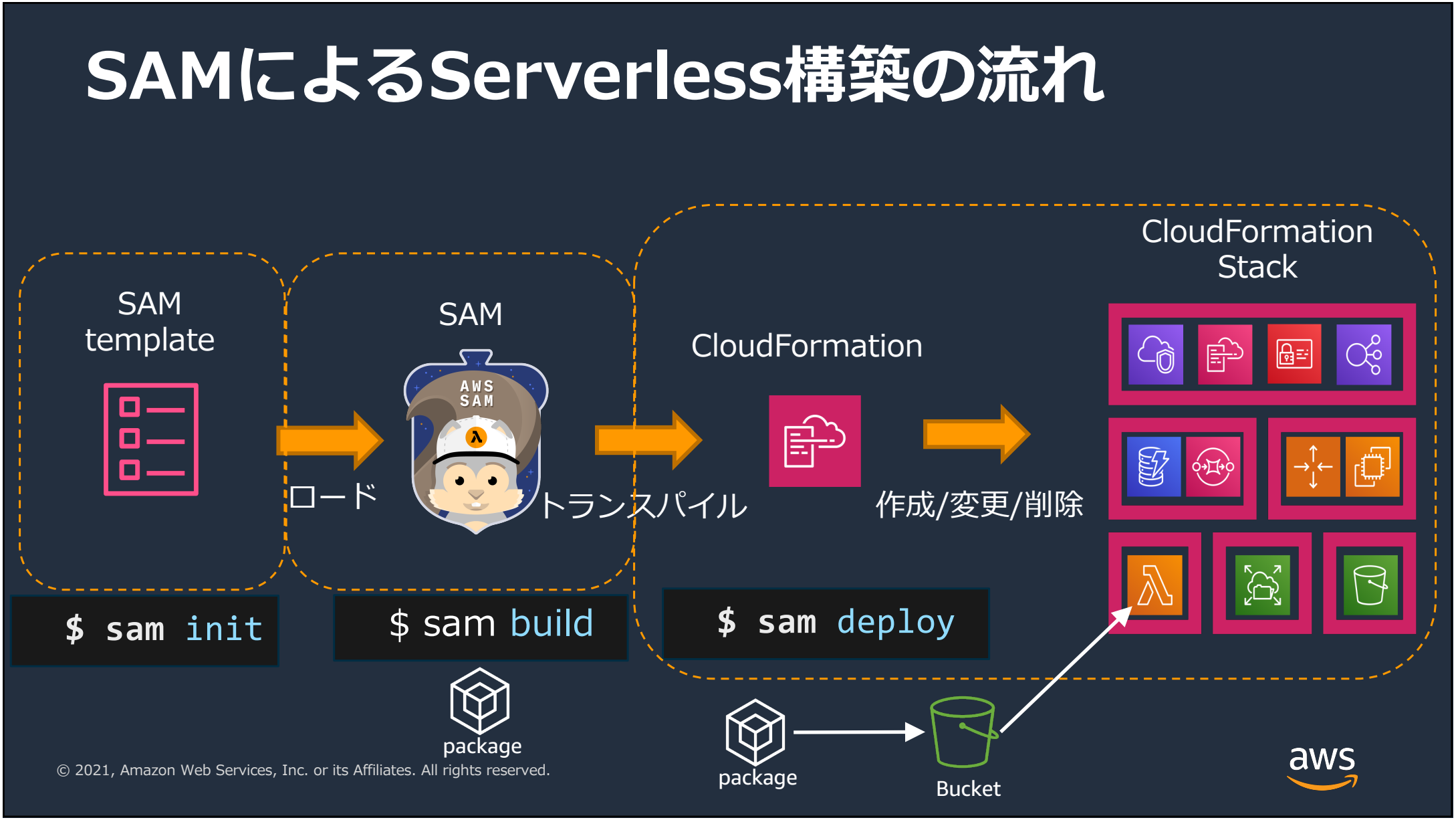
Demo — SAM CLI

SAM CLIのデモ

- 新規SAMアプリケーションの作成
- プロジェクトファイルの構成確認
- SAMテンプレートの確認
- SAMアプリケーションのデプロイ
- デプロイ結果の確認

aws

SAMによるServerless構築の流れ



CI/CD for Serverless

Serverless Application Pipeline

aws

Serverless Application Pipeline

- AWS Lambdaのマネジメントコンソールからアプリケーションの CI/CD環境を構築
  - 構築方法を選択
    - サンプルアプリケーション/AWS Serverless Application Repoitory / ーから作成
  - コードリポジトリを選択
    - CodeCommit / GitHub
- CloudFormationスタックの生成と実行
  - CI/CD環境用スタック
  - Lambda関数デプロイ用スタック
- パイプラインを使用した継続的インテグレーション/デプロイメント

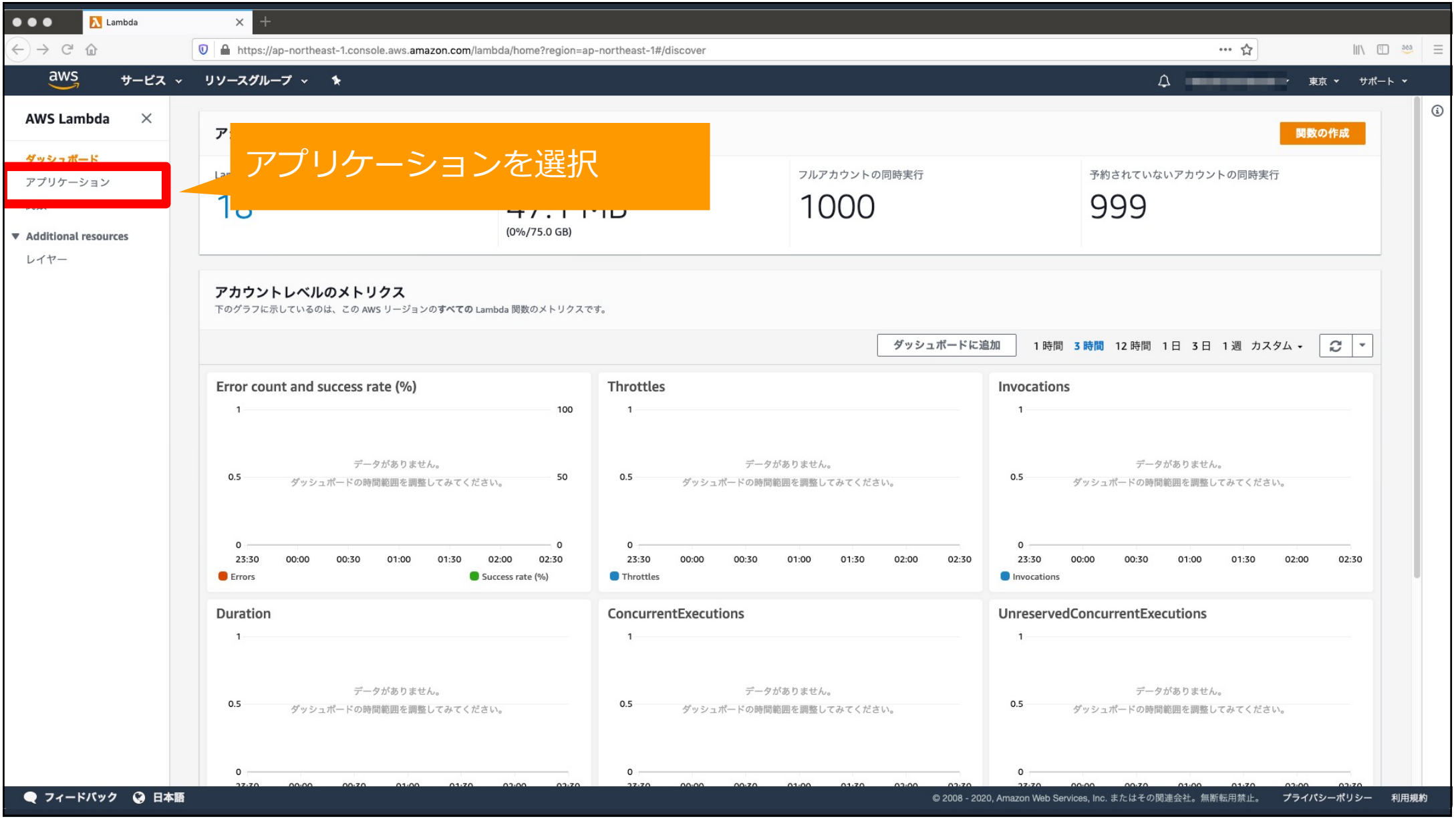
aws

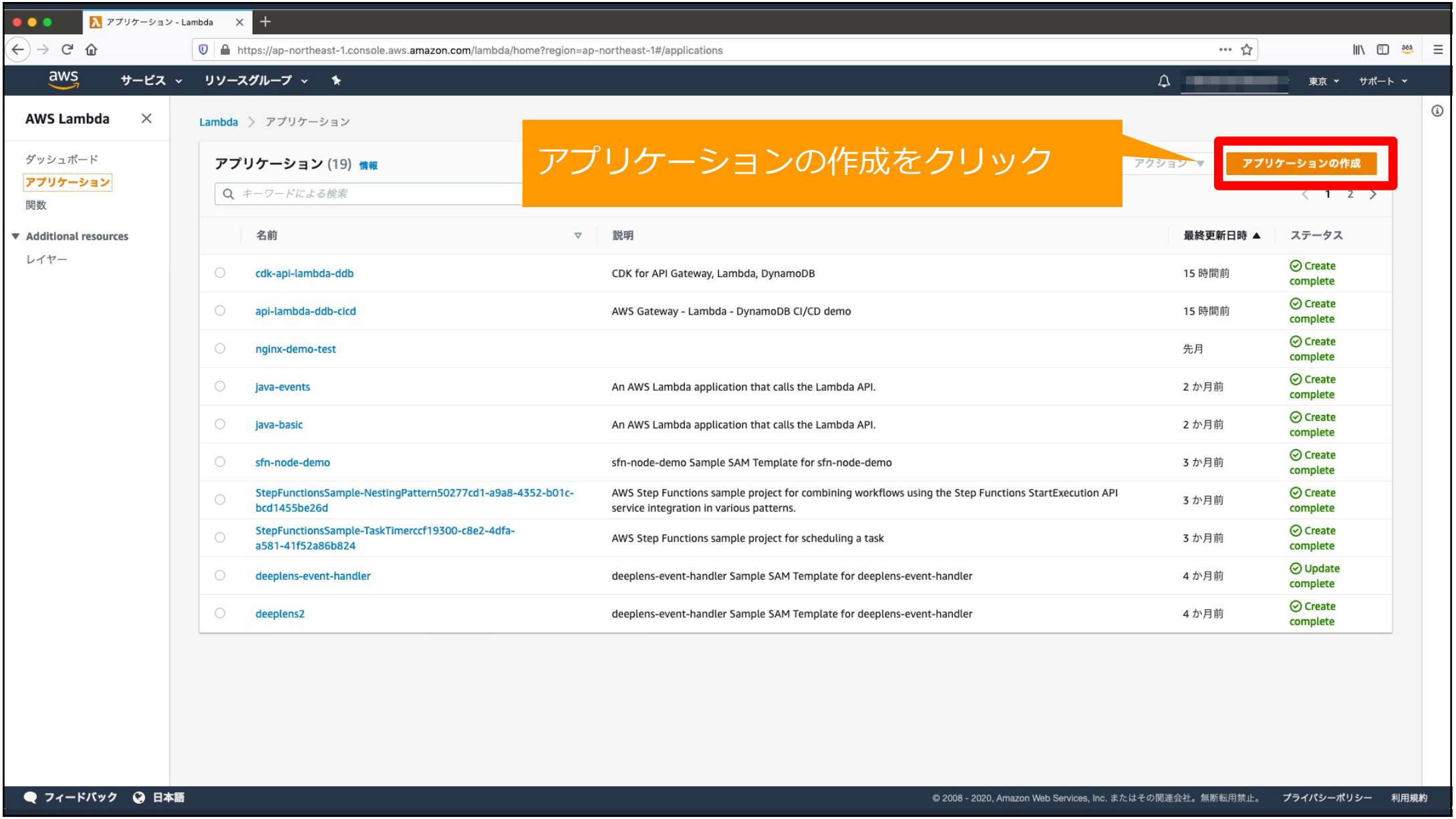
Demo —
Serverless Application Pipeline

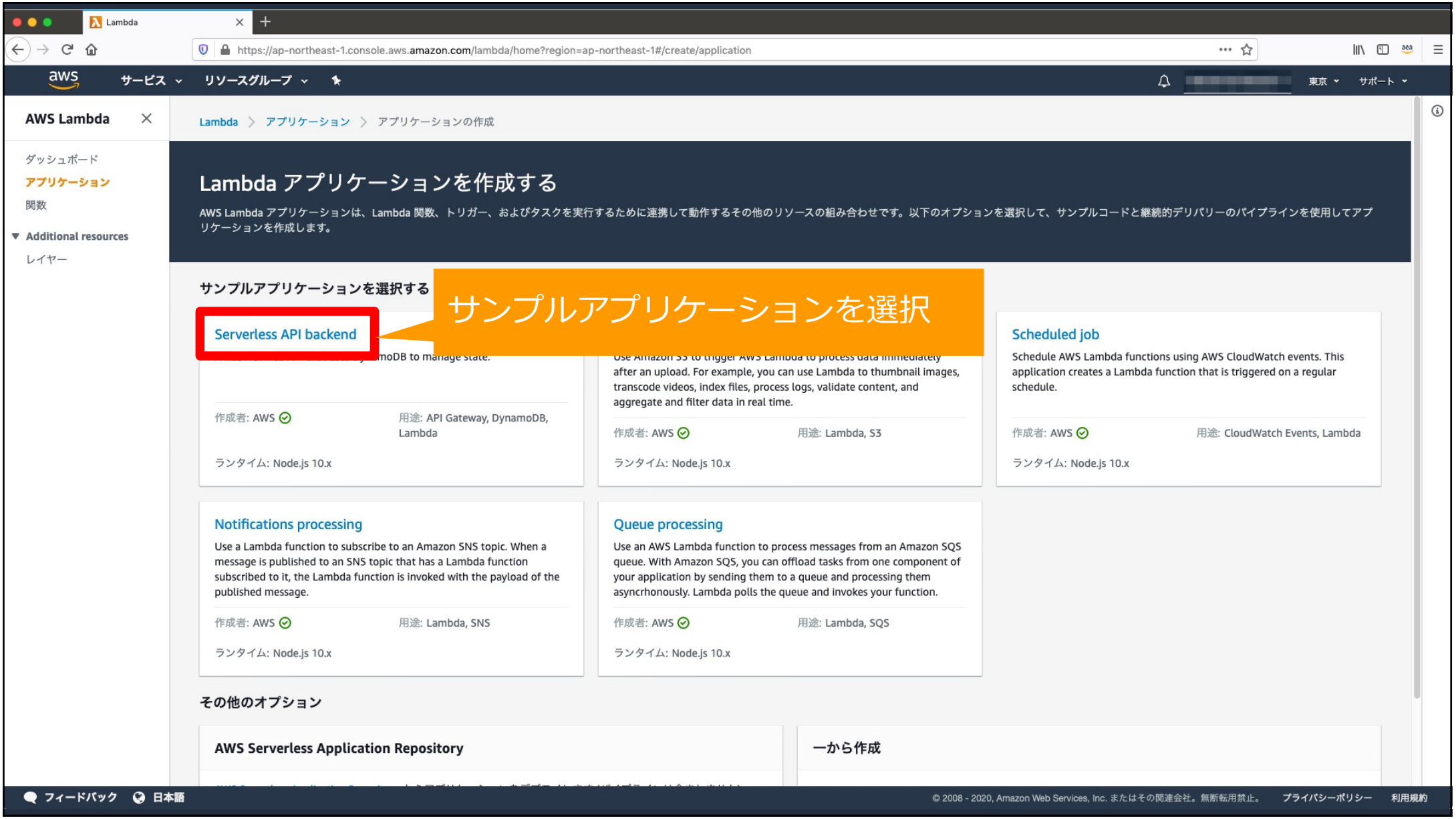
aws

Serverless Application Pipelineのデモ

- サンプルアプリケーションの選択
- CI/CD環境の構築
- コードリポジトリの確認
- パイプラインの確認
- モニタリングの確認


Lambda
https://ap-northeast-1.console.aws.amazon.com/lambda/home?region=ap-northeast-1#/discover
aws
サービス
リソースグループ
東京
サポート
AWS Lambda
ダッシュボード
アプリケーション
Additional resources
レイヤー
アプリケーションを選択
関数の作成
フルアカウントの同時実行
1000
予約されていないアカウントの同時実行
999
(0%/75.0 GB)
アカウントレベルのメトリクス
下のグラフに示しているのは、この AWS リージョンのすべての Lambda 関数のメトリクスです。
ダッシュボードに追加
1 時間
3 時間
12 時間
1 日
3 日
1 週
カスタム
Error count and success rate (%)
Throttles
Invocations
Duration
ConcurrentExecutions
UnreservedConcurrentExecutions
データがありません。
ダッシュボードの時間範囲を調整してみてください。
Errors
Success rate (%)
フィードバック
日本語
© 2008 - 2020, Amazon Web Services, Inc. またはその関連会社。無断転用禁止。
プライバシーポリシー
利用規約


アプリケーション - Lambda
https://ap-northeast-1.console.aws.amazon.com/lambda/home?region=ap-northeast-1#/applications
aws
サービス
リソースグループ
東京
サポート
AWS Lambda
ダッシュボード
アプリケーション
関数
Additional resources
レイヤー
Lambda > アプリケーション
アプリケーションの作成をクリック
アプリケーション (19) 情報
キーワードによる検索
アクション
アプリケーションの作成
1 2
名前
説明
最終更新日時
ステータス
cdk-api-lambda-ddb
CDK for API Gateway, Lambda, DynamoDB
15 時間前
Create complete
api-lambda-ddb-cicd
AWS Gateway - Lambda - DynamoDB CI/CD demo
15 時間前
Create complete
nginx-demo-test
先月
Create complete
java-events
An AWS Lambda application that calls the Lambda API.
2 か月前
Create complete
java-basic
An AWS Lambda application that calls the Lambda API.
2 か月前
Create complete
sfn-node-demo
sfn-node-demo Sample SAM Template for sfn-node-demo
3 か月前
Create complete
StepFunctionsSample-NestingPattern50277cd1-a9a8-4352-b01c-bcd1455be26d
AWS Step Functions sample project for combining workflows using the Step Functions StartExecution API service integration in various patterns.
3 か月前
Create complete
StepFunctionsSample-TaskTimerccf19300-c8e2-4dfa-a581-41f52a86b824
AWS Step Functions sample project for scheduling a task
3 か月前
Create complete
deeplens-event-handler
deeplens-event-handler Sample SAM Template for deeplens-event-handler
4 か月前
Update complete
deeplens2
deeplens-event-handler Sample SAM Template for deeplens-event-handler
4 か月前
Create complete
フィードバック
日本語
© 2008 - 2020, Amazon Web Services, Inc. またはその関連会社。無断転用禁止。
プライバシーポリシー
利用規約


Lambda
https://ap-northeast-1.console.aws.amazon.com/lambda/home?region=ap-northeast-1#/create/application
aws
サービス
リソースグループ
東京
サポート
AWS Lambda
ダッシュボード
アプリケーション
関数
Additional resources
レイヤー
Lambda > アプリケーション > アプリケーションの作成
Lambda アプリケーションを作成する
AWS Lambda アプリケーションは、Lambda 関数、トリガー、およびタスクを実行するために連携して動作するその他のリソースの組み合わせです。以下のオプションを選択して、サンプルコードと継続的デリバリーのパイプラインを使用してアプリケーションを作成します。
サンプルアプリケーションを選択する
サンプルアプリケーションを選択
Serverless API backend
作成者: AWS
用途: API Gateway, DynamoDB, Lambda
ランタイム: Node.js 10.x
after an upload. For example, you can use Lambda to thumbnail images, transcode videos, index files, process logs, validate content, and aggregate and filter data in real time.
作成者: AWS
用途: Lambda, S3
ランタイム: Node.js 10.x
Scheduled job
Schedule AWS Lambda functions using AWS CloudWatch events. This application creates a Lambda function that is triggered on a regular schedule.
作成者: AWS
用途: CloudWatch Events, Lambda
ランタイム: Node.js 10.x
Notifications processing
Use a Lambda function to subscribe to an Amazon SNS topic. When a message is published to an SNS topic that has a Lambda function subscribed to it, the Lambda function is invoked with the payload of the published message.
作成者: AWS
用途: Lambda, SNS
ランタイム: Node.js 10.x
Queue processing
Use an AWS Lambda function to process messages from an Amazon SQS queue. With Amazon SQS, you can offload tasks from one component of your application by sending them to a queue and processing them asynchronously. Lambda polls the queue and invokes your function.
作成者: AWS
用途: Lambda, SQS
ランタイム: Node.js 10.x
その他のオプション
AWS Serverless Application Repository
一から作成
フィードバック
日本語
© 2008 - 2020, Amazon Web Services, Inc. またはその関連会社。無断転用禁止。
プライバシーポリシー
利用規約

Lambda

https://ap-northeast-1.console.aws.amazon.com/lambda/home?region=ap-northeast-1#/create/application/view?applicationId=web-backend

aws サービス リソースグループ 東京 サポート

AWS Lambda

ダッシュボード
アプリケーション
関数
Additional resources
レイヤー

Serverless API backend

A RESTful web API that uses DynamoDB to manage state.

作成者: AWS

用途: API Gateway, DynamoDB, Lambda

ランタイム: Node.js 10.x

アーキテクチャ

ソースコード

使用したサービス

ソース管理
CodeCommit または GitHub

継続的デリバリー
CodePipeline

アプリケーションのリソース
API Gateway
DynamoDB
Lambda

構築およびテスト
CodeBuild

デプロイ
AWS CloudFormation

開発ワークフロー

次のページでアプリケーションを設定して、開始します。

1 選択
このサンプルアプリケーションを使用するか、前のページに戻ります。

2 作成
設定を構成し、アプリケーションのリソースを作成します。

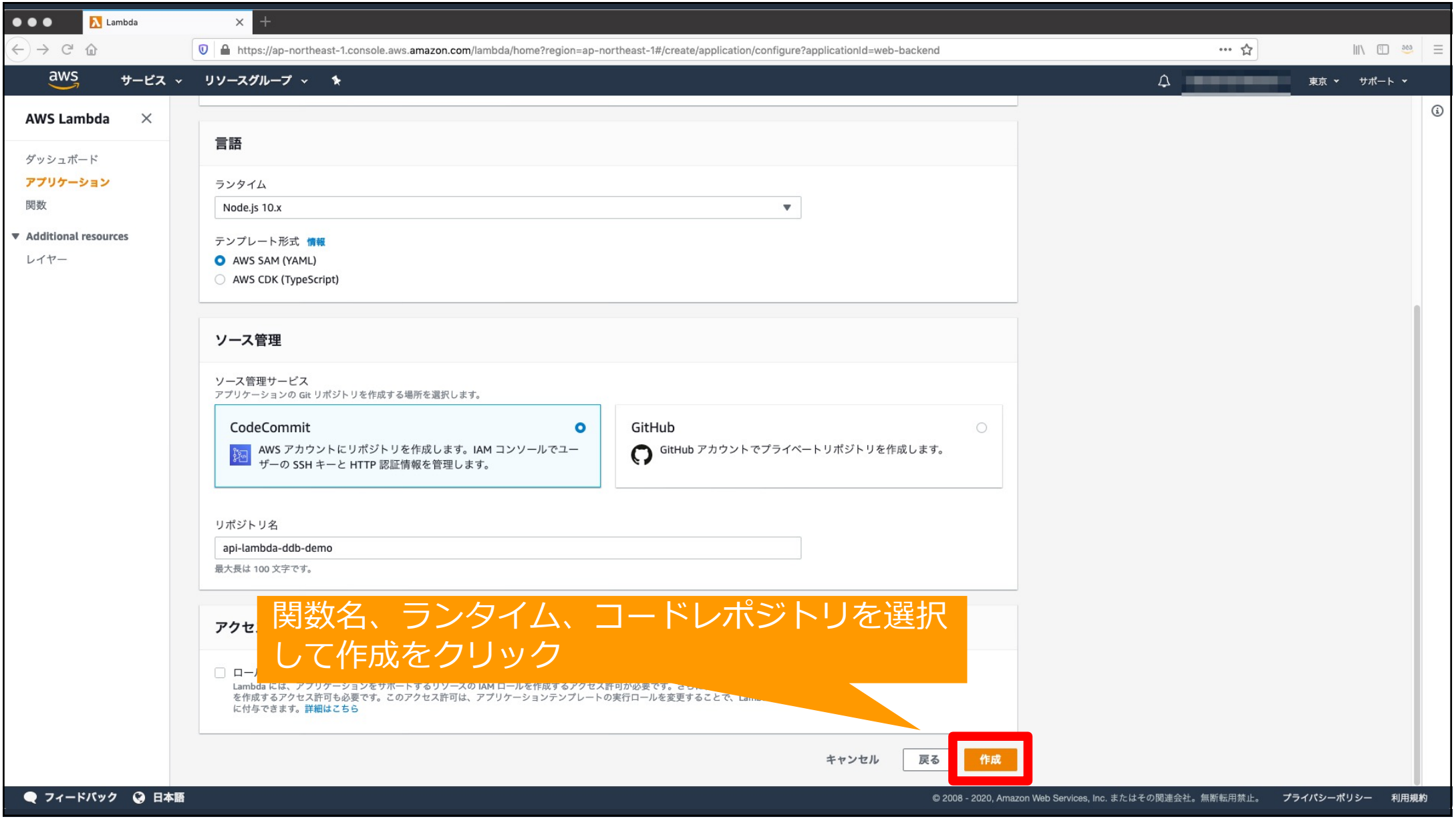
3 クローン作成
ローカル開発用のアプリケーション
アップツールのクローンを作成します。

ラインをトリガーします。

次へをクリック

戻る 次へ

フィードバック 日本語


Lambda
https://ap-northeast-1.console.aws.amazon.com/lambda/home?region=ap-northeast-1#/create/application/configure?applicationId=web-backend
aws
サービス
リソースグループ
東京
サポート
AWS Lambda
ダッシュボード
アプリケーション
関数
Additional resources
レイヤー
言語
ランタイム
Node.js 10.x
テンプレート形式 情報
AWS SAM (YAML)
AWS CDK (TypeScript)
ソース管理
ソース管理サービス
アプリケーションの Git リポジトリを作成する場所を選択します。
CodeCommit
AWS アカウントにリポジトリを作成します。IAM コンソールでユーザーの SSH キーと HTTP 認証情報を管理します。
GitHub
GitHub アカウントでプライベートリポジトリを作成します。
リポジトリ名
api-lambda-ddb-demo
最大長は 100 文字です。
関数名、ランタイム、コードレポジトリを選択して作成をクリック
キャンセル
戻る
作成
フィードバック
日本語
© 2008 - 2020, Amazon Web Services, Inc. またはその関連会社。無断転用禁止。
プライバシーポリシー
利用規約

api-lambda-ddb-cicd アプリケ

https://ap-northeast-1.console.aws.amazon.com/lambda/home?region=ap-northeast-1#/applications/api-lambda-ddb-cicd

aws サービス リソースグループ 東京 サポート

AWS Lambda

ダッシュボード
アプリケーション
関数

Additional resources
レイヤー

Lambda > アプリケーション > api-lambda-ddb-cicd

api-lambda-ddb-cicd

概要 | コード | デプロイ | モニタリング

開始方法

サーバーレスアプリケーション

アプリケーションリポジトリには、アプリケーションを定義する関数コード、テンプレート、およびビルド設定が含まれています。その構造の概要と開始手順については、 README を表示します。**Code タブには、ローカル開発およびテスト用のソースリポジトリとツールに関する情報が表示されます。アプリケーションからメトリクスを表示してログを分析するには、 Monitoring を選択します。**

継続的デリバリーインフラストラクチャ

マスターブランチに変更をコミットして AWS CodeCommit または GitHub にプッシュすると、アプリケーションのパイプラインによって変更が自動的にビルド、パッケージ化、デプロイされます。アプリケーションの継続的デリバリーを有効にするサポートリソースのリストについては、以下を参照してください。 **デプロイで**、、ソース、ビルド、デプロイの各リソースを接続するパイプラインの詳細を確認できます。

AWS リソースとサービスのアクセス許可を追加する

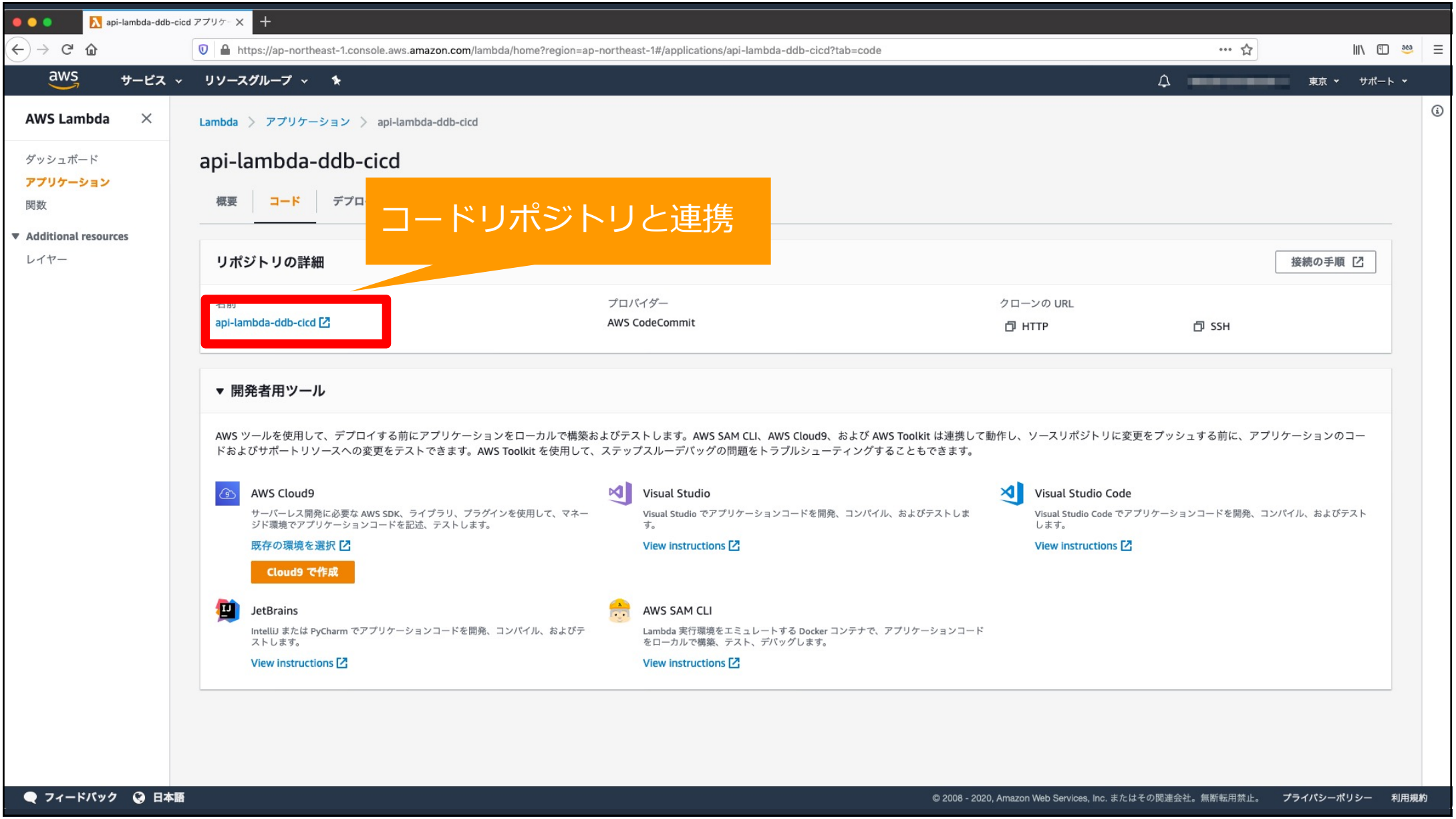
アプリケーションに AWS リソースとトリガーを追加するには、アプリケーションのソースコードに含まれている AWS SAM テンプレートでそれらを定義します。テンプレートは、アプリケーションの関数の実行ロールも定義します。このロールを拡張して、コードからより多くのサービスやリソースにアクセスできます。実行ロールは、Amazon SQS などのサービスから読み取る Lambda イベントソースマッピングへのアクセスも提供します。 詳細

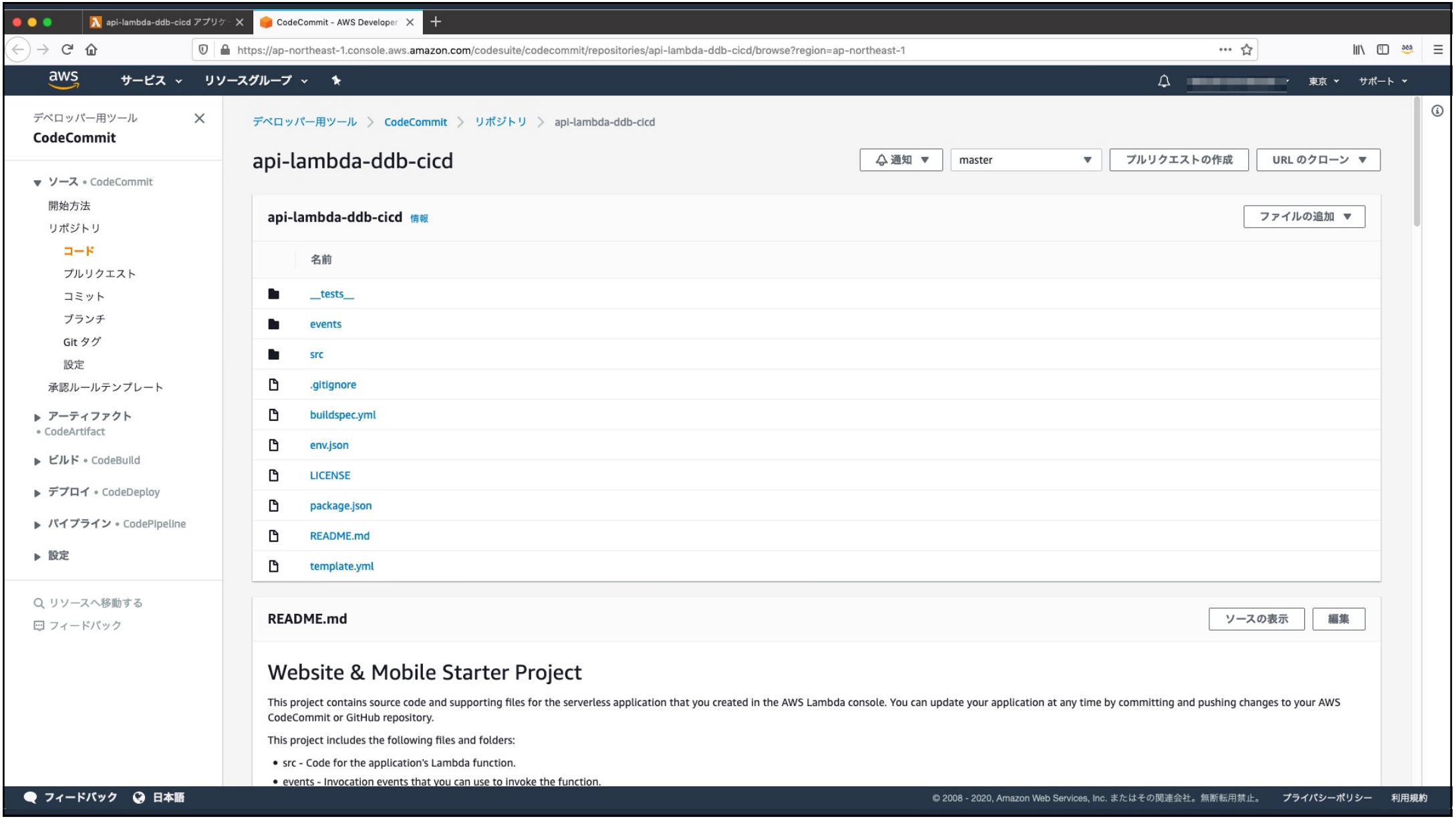
API エンドポイント

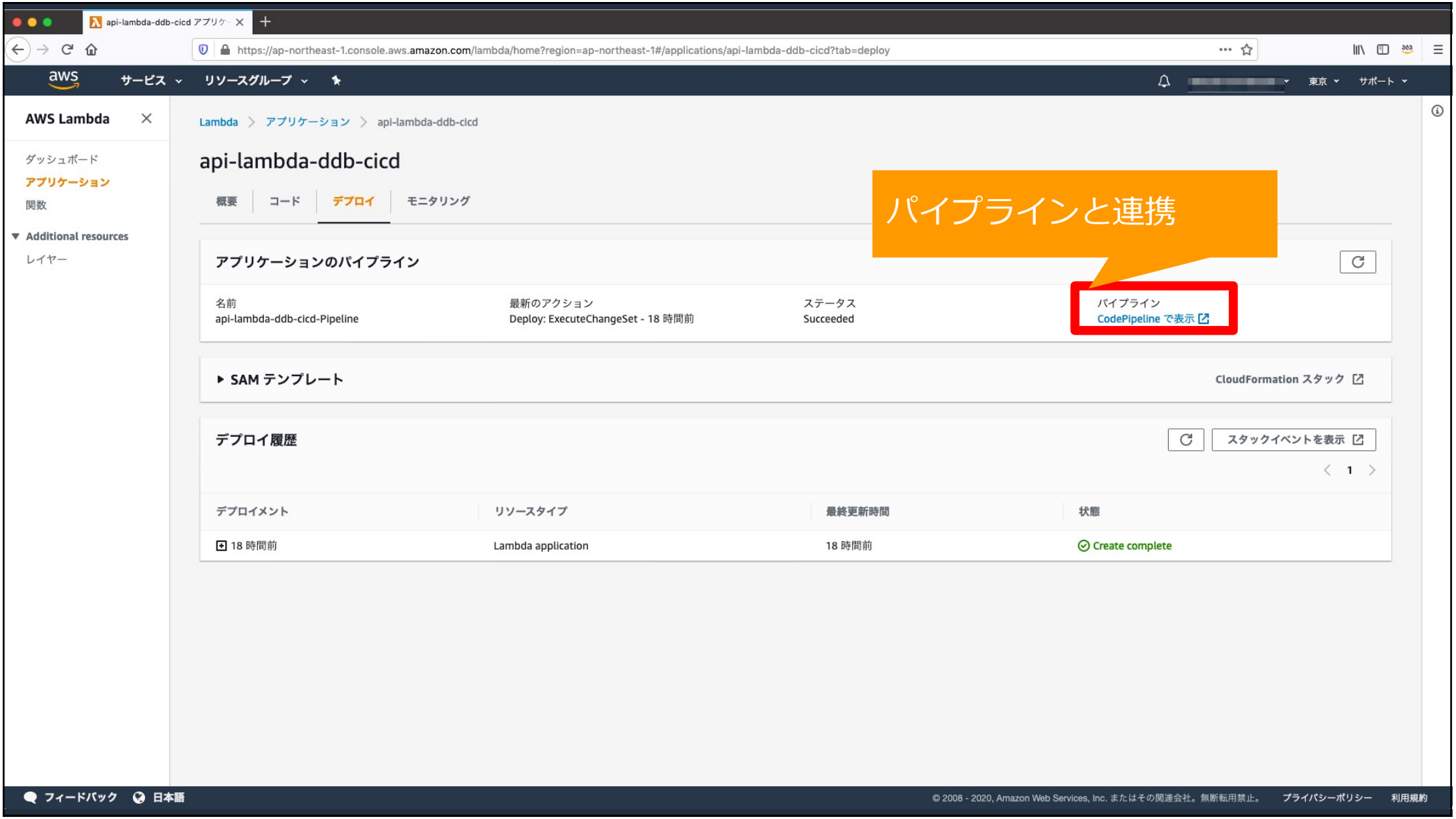
エンドポイント
https://nn6y2e3ay8.execute-api.ap-northeast-1.amazonaws.com/Prod

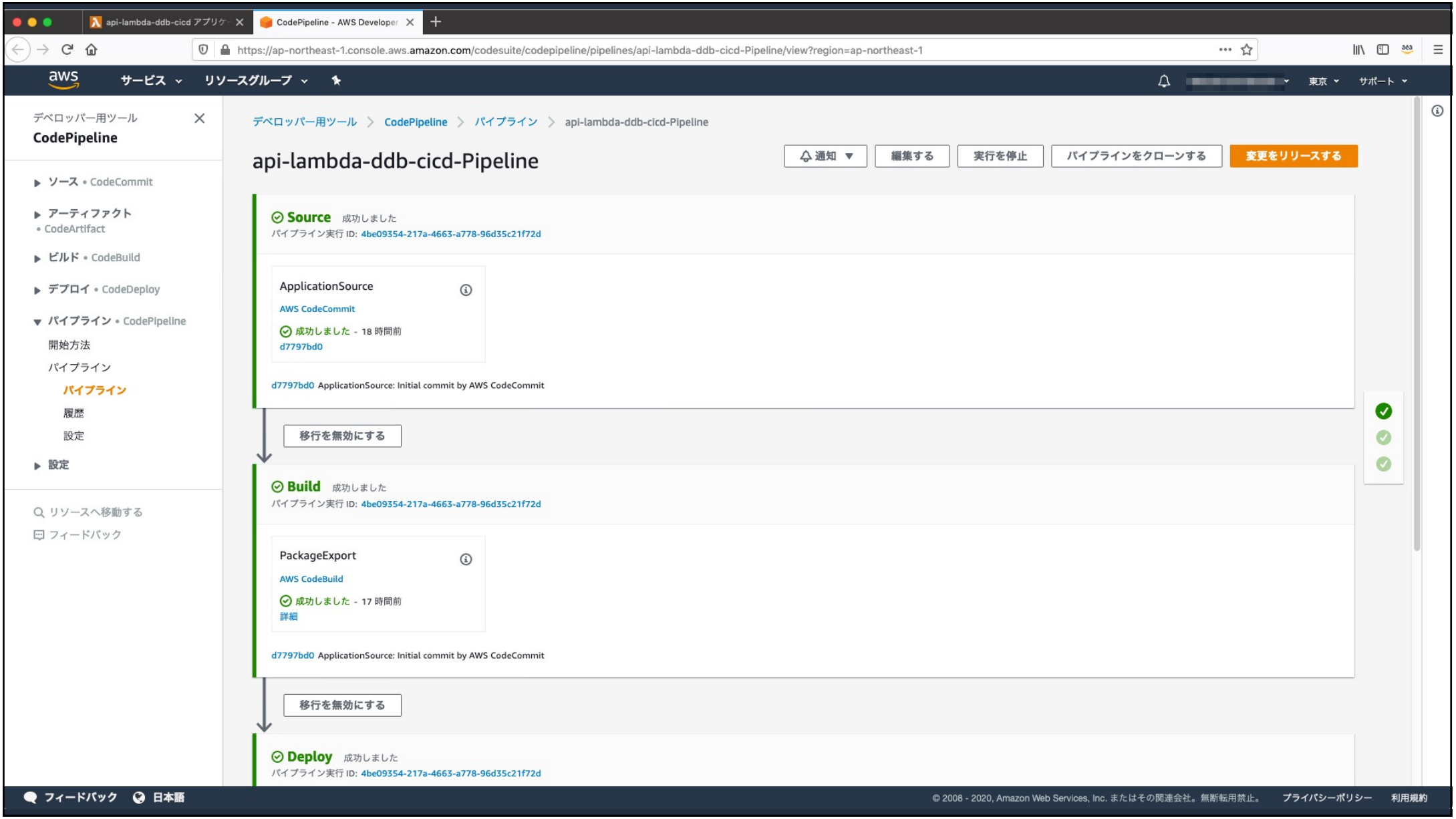
フィードバック 日本語

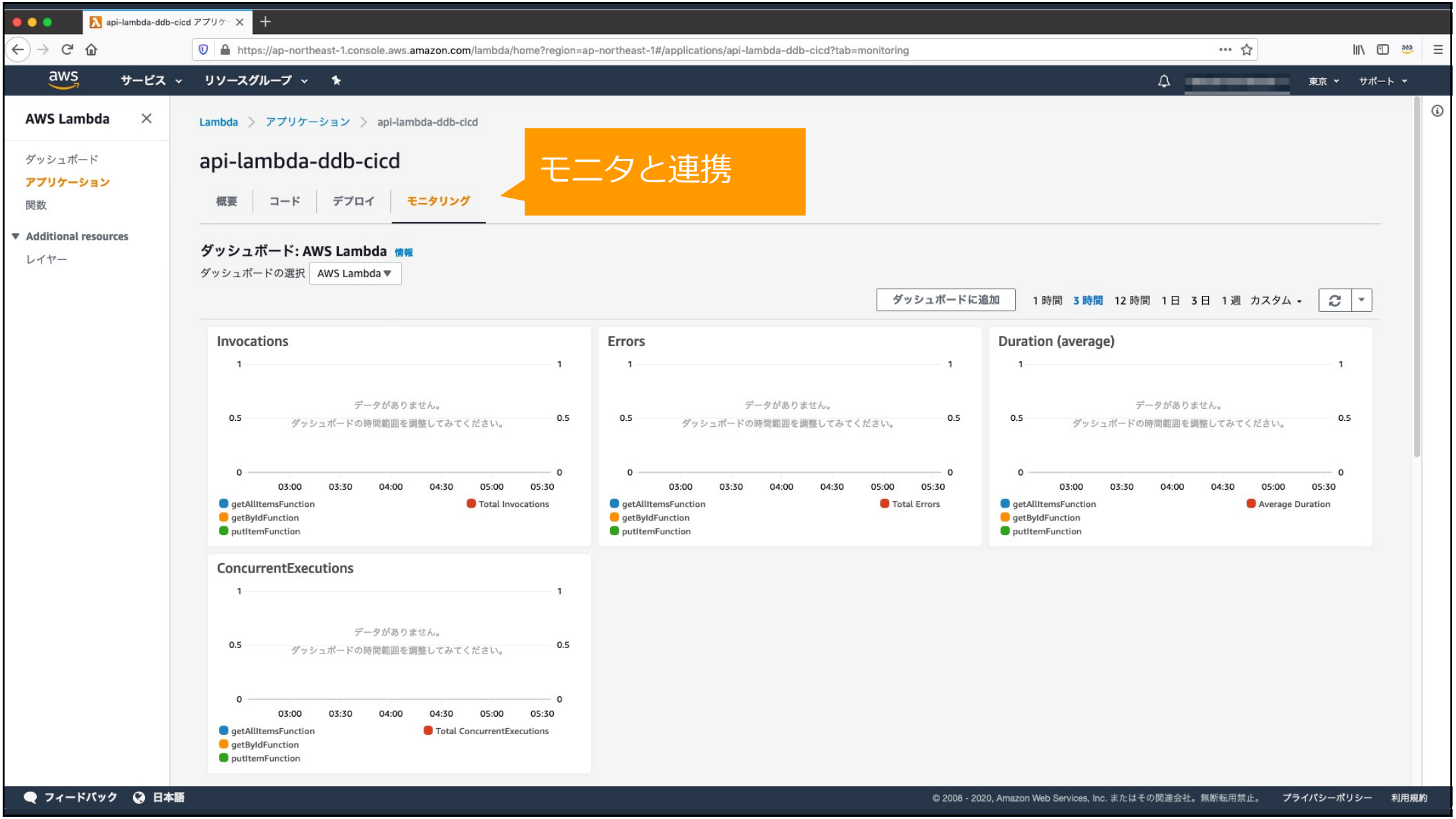
プライバシーポリシー 利用規約


api-lambda-ddb-cicd アプリケ
https://ap-northeast-1.console.aws.amazon.com/lambda/home?region=ap-northeast-1#/applications/api-lambda-ddb-cicd?tab=code
aws
サービス
リソースグループ
東京
サポート
AWS Lambda
ダッシュボード
アプリケーション
関数
Additional resources
レイヤー
Lambda > アプリケーション > api-lambda-ddb-cicd
api-lambda-ddb-cicd
概要
コード
デプロ
コードリポジトリと連携
リポジトリの詳細
接続の手順
名前
api-lambda-ddb-cicd
プロバイダー
AWS CodeCommit
クローンの URL
HTTP
SSH
開発者用ツール
AWS ツールを使用して、デプロイする前にアプリケーションをローカルで構築およびテストします。AWS SAM CLI、AWS Cloud9、および AWS Toolkit は連携して動作し、ソースリポジトリに変更をプッシュする前に、アプリケーションのコードおよびサポートリソースへの変更をテストできます。AWS Toolkit を使用して、ステップスルーデバッグの問題をトラブルシューティングすることもできます。
AWS Cloud9
サーバーレス開発に必要な AWS SDK、ライブラリ、プラグインを使用して、マネージド環境でアプリケーションコードを記述、テストします。
既存の環境を選択
Cloud9 で作成
Visual Studio
Visual Studio でアプリケーションコードを開発、コンパイル、およびテストします。
View instructions
Visual Studio Code
Visual Studio Code でアプリケーションコードを開発、コンパイル、およびテストします。
View instructions
JetBrains
IntelliJ または PyCharm でアプリケーションコードを開発、コンパイル、およびテストします。
View instructions
AWS SAM CLI
Lambda 実行環境をエミュレートする Docker コンテナで、アプリケーションコードをローカルで構築、テスト、デバッグします。
View instructions
フィードバック
日本語
© 2008 - 2020, Amazon Web Services, Inc. またはその関連会社。無断転用禁止。
プライバシーポリシー
利用規約


api-lambda-ddb-cicd アプリケ
CodeCommit - AWS Developer
https://ap-northeast-1.console.aws.amazon.com/codesuite/codecommit/repositories/api-lambda-ddb-cicd/browse?region=ap-northeast-1
aws
サービス
リソースグループ
東京
サポート
デベロッパー用ツール
CodeCommit
ソース • CodeCommit
開始方法
リポジトリ
コード
プルリクエスト
コミット
ブランチ
Git タグ
設定
承認ルールテンプレート
アーティファクト
• CodeArtifact
ビルド • CodeBuild
デプロイ • CodeDeploy
パイプライン • CodePipeline
設定
リソースへ移動する
フィードバック
デベロッパー用ツール > CodeCommit > リポジトリ > api-lambda-ddb-cicd
api-lambda-ddb-cicd
通知
master
プルリクエストの作成
URL のクローン
api-lambda-ddb-cicd 情報
ファイルの追加
名前
__tests__
events
src
.gitignore
buildspec.yml
env.json
LICENSE
package.json
README.md
template.yml
README.md
ソースの表示
編集
Website & Mobile Starter Project
This project contains source code and supporting files for the serverless application that you created in the AWS Lambda console. You can update your application at any time by committing and pushing changes to your AWS CodeCommit or GitHub repository.
This project includes the following files and folders:
• src - Code for the application's Lambda function.
• events - Invocation events that you can use to invoke the function.
フィードバック
日本語
© 2008 - 2020, Amazon Web Services, Inc. またはその関連会社。無断転用禁止。
プライバシーポリシー
利用規約


api-lambda-ddb-cicd アプリケ
https://ap-northeast-1.console.aws.amazon.com/lambda/home?region=ap-northeast-1#/applications/api-lambda-ddb-cicd?tab=deploy
aws
サービス
リソースグループ
東京
サポート
AWS Lambda
ダッシュボード
アプリケーション
関数
Additional resources
レイヤー
Lambda > アプリケーション > api-lambda-ddb-cicd
api-lambda-ddb-cicd
概要
コード
デプロイ
モニタリング
パイプラインと連携
アプリケーションのパイプライン
名前
api-lambda-ddb-cicd-Pipeline
最新のアクション
Deploy: ExecuteChangeSet - 18 時間前
ステータス
Succeeded
パイプライン
CodePipeline で表示
SAM テンプレート
CloudFormation スタック
デプロイ履歴
スタックイベントを表示
1
デプロイメント
リソースタイプ
最終更新時間
状態
18 時間前
Lambda application
18 時間前
Create complete
フィードバック
日本語
© 2008 - 2020, Amazon Web Services, Inc. またはその関連会社。無断転用禁止。
プライバシーポリシー
利用規約


api-lambda-ddb-cicd アプリケ
CodePipeline - AWS Developer
https://ap-northeast-1.console.aws.amazon.com/codesuite/codepipeline/pipelines/api-lambda-ddb-cicd-Pipeline/view?region=ap-northeast-1
aws
サービス
リソースグループ
東京
サポート
デベロッパー用ツール
CodePipeline
ソース • CodeCommit
アーティファクト
• CodeArtifact
ビルド • CodeBuild
デプロイ • CodeDeploy
パイプライン • CodePipeline
開始方法
パイプライン
パイプライン
履歴
設定
設定
リソースへ移動する
フィードバック
デベロッパー用ツール > CodePipeline > パイプライン > api-lambda-ddb-cicd-Pipeline
api-lambda-ddb-cicd-Pipeline
通知
編集する
実行を停止
パイプラインをクローンする
変更をリリースする
Source 成功しました
パイプライン実行 ID: 4be09354-217a-4663-a778-96d35c21f72d
ApplicationSource
AWS CodeCommit
成功しました - 18 時間前
d7797bd0
d7797bd0 ApplicationSource: Initial commit by AWS CodeCommit
移行を無効にする
Build 成功しました
パイプライン実行 ID: 4be09354-217a-4663-a778-96d35c21f72d
PackageExport
AWS CodeBuild
成功しました - 17 時間前
詳細
d7797bd0 ApplicationSource: Initial commit by AWS CodeCommit
移行を無効にする
Deploy 成功しました
パイプライン実行 ID: 4be09354-217a-4663-a778-96d35c21f72d
フィードバック
日本語
© 2008 - 2020, Amazon Web Services, Inc. またはその関連会社。無断転用禁止。
プライバシーポリシー
利用規約


api-lambda-ddb-cicd アプリケ
https://ap-northeast-1.console.aws.amazon.com/lambda/home?region=ap-northeast-1#/applications/api-lambda-ddb-cicd?tab=monitoring
aws
サービス
リソースグループ
東京
サポート
AWS Lambda
ダッシュボード
アプリケーション
関数
Additional resources
レイヤー
Lambda > アプリケーション > api-lambda-ddb-cicd
api-lambda-ddb-cicd
モニタと連携
概要
コード
デプロイ
モニタリング
ダッシュボード: AWS Lambda 情報
ダッシュボードの選択 AWS Lambda
ダッシュボードに追加
1 時間 3 時間 12 時間 1 日 3 日 1 週 カスタム
Invocations
Errors
Duration (average)
ConcurrentExecutions
データがありません。
ダッシュボードの時間範囲を調整してみてください。
1
0.5
0
03:00
03:30
04:00
04:30
05:00
05:30
getAllItemsFunction
getByIdFunction
putItemFunction
Total Invocations
Total Errors
Average Duration
Total ConcurrentExecutions
フィードバック
日本語
© 2008 - 2020, Amazon Web Services, Inc. またはその関連会社。無断転用禁止。
プライバシーポリシー
利用規約

CloudFormation Stack の確認

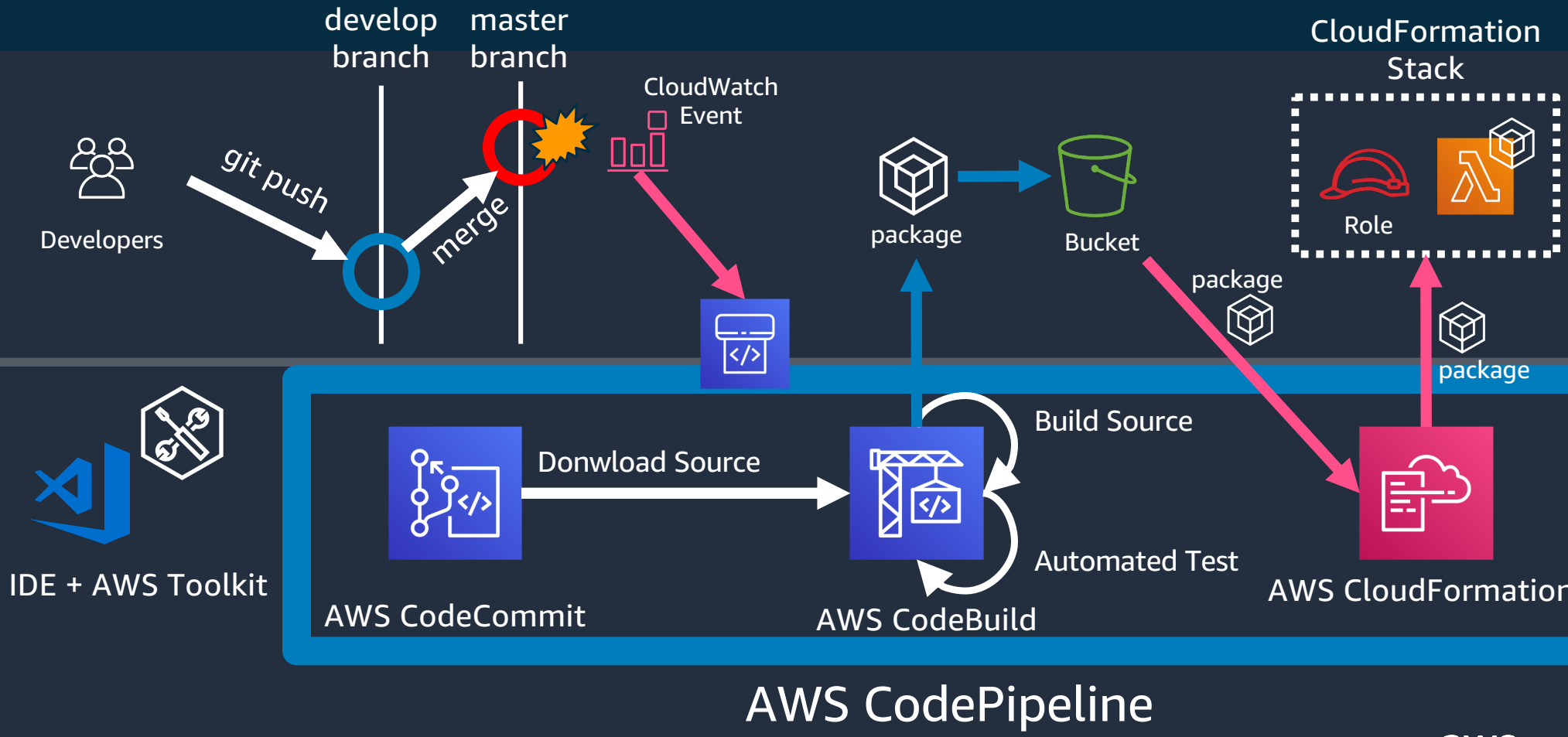
上位Stack

seminar-app
アプリケーションに必要なリソースを生成

下位Stack

serverlessrepo-seminar-app-toolchain
パイプラインに必要なツールリソースを生成

Lambda関数修正後のmergeでパイプライン起動！



ここまでのまとめ

- AWS Lambdaの開発にはServerless Application Model (SAM) の利用が可能
- SAM CLIを利用して手作業によるミスを防ぎ自動化を促進
- Serverless Application Pipelineを利用してCI/CD環境を容易に構築可能

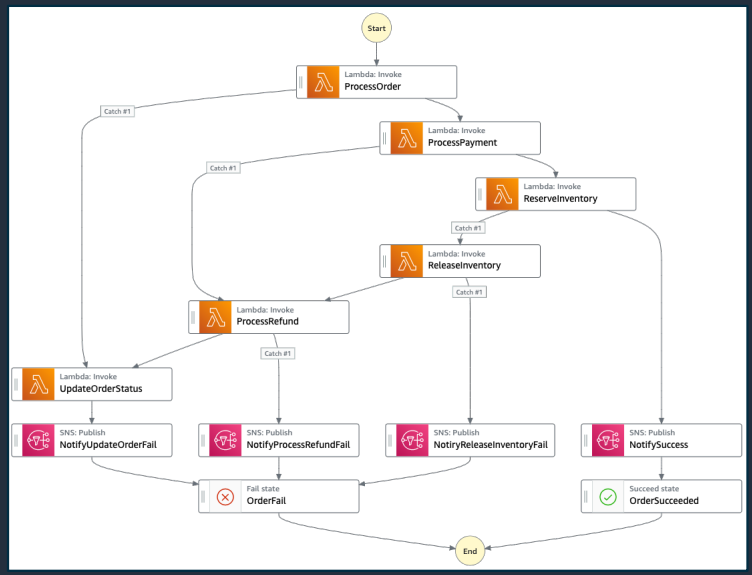
aws

AWS Step Functions Workflow Studio

おさらい：AWS Step Functionsとは

AWSのフルマネージドなステートマシン

- 弾力性のあるワークフローオートメーション
- 組み込みのエラーハンドリング
- AWSサービスとの強力な統合
  独自のサービスとの統合サポート
- 実行履歴の監査とビジュアルモニタリング

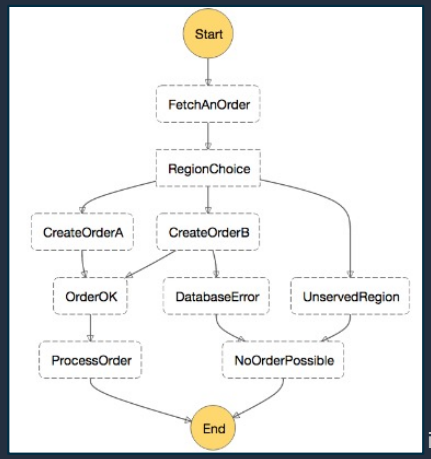


おさらい：AWS Step Functionsとは

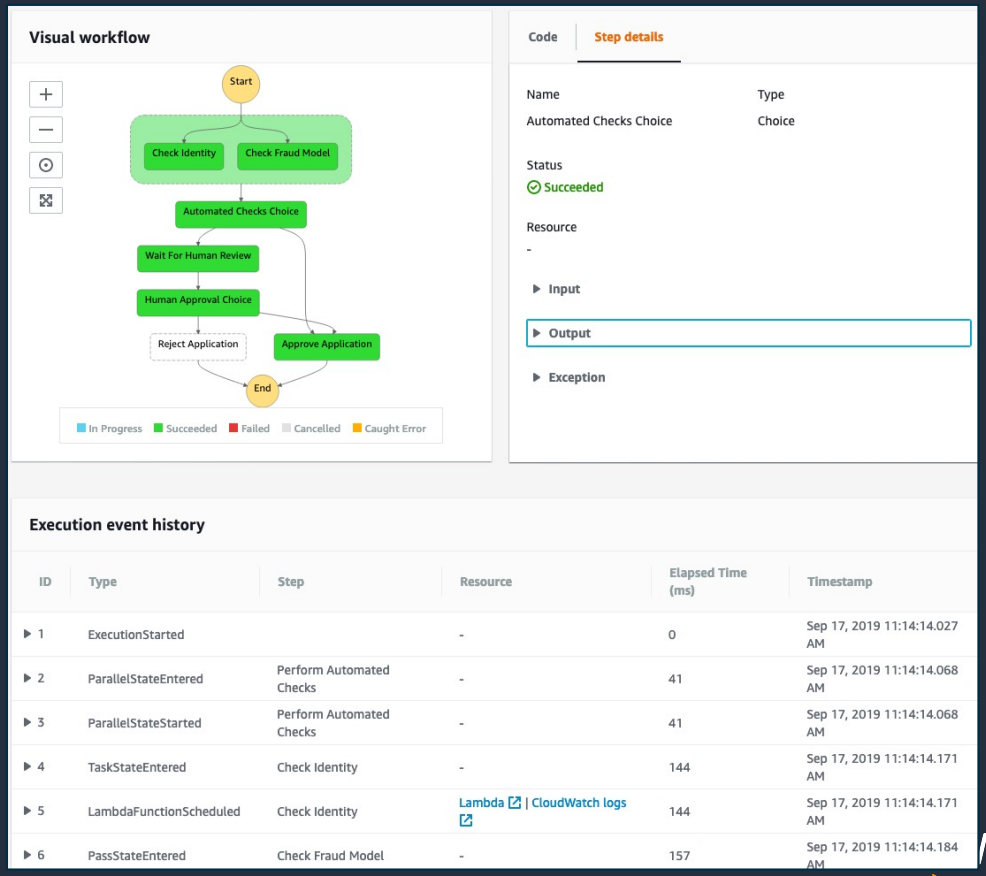
JSONで定義(Amazon States Language)

```
{
    "Comment": "Manage opening an account",
    "StartAt": "Perform Automated Checks",
    "States": {
        "Perform Automated Checks": {
            "Type": "Parallel",
            "Branches": [{
                    "StartAt": "Check Identity",
                    "States": {
                        "Check Identity": {
                            "Type": "Task",
                            "Parameters": {
```

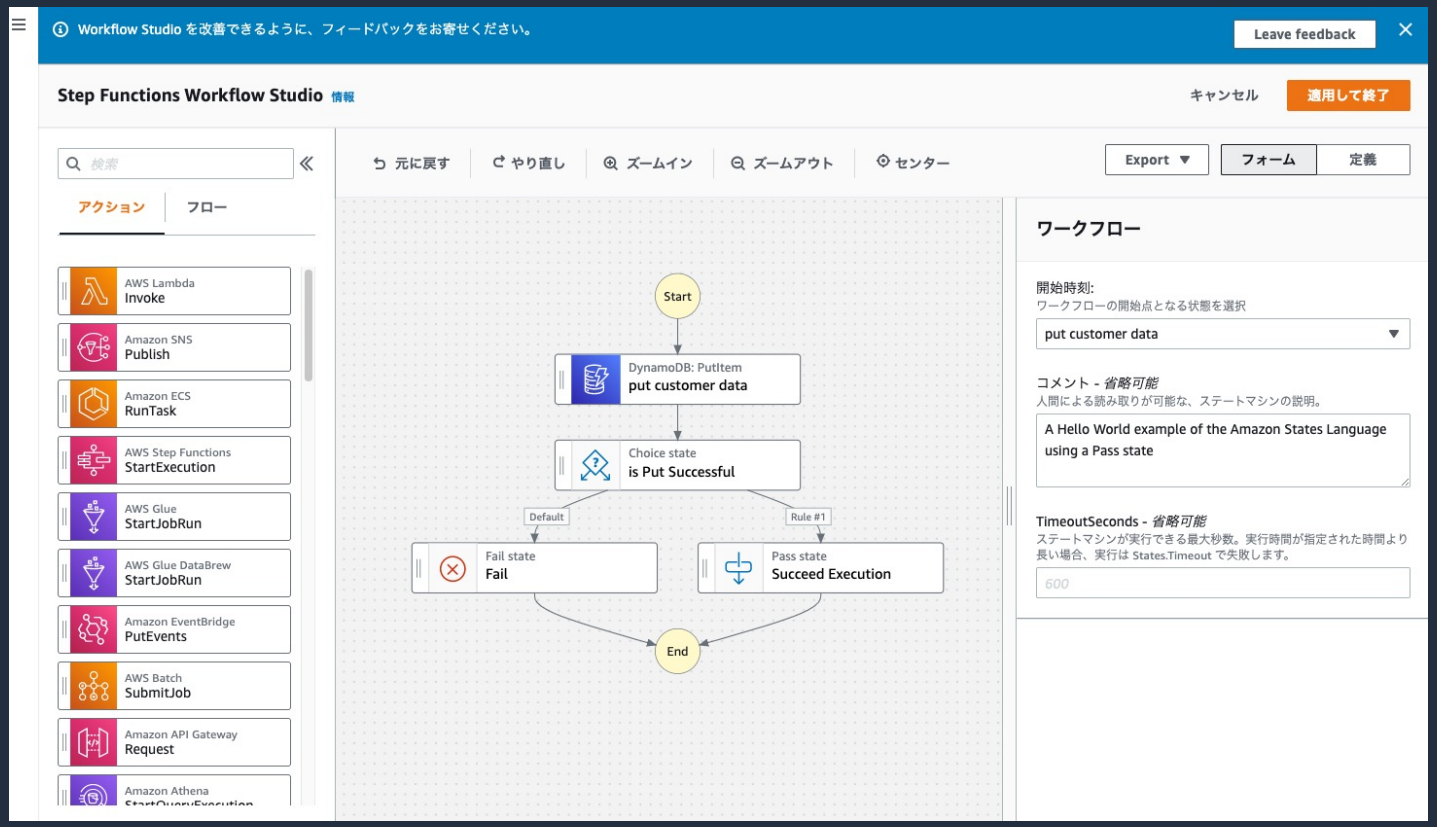
コンソールで視覚化



実行結果をモニタリング



Step Functions Workflow Studio New!



- GUIによる直感的な操作で素早くワークフローを構築可能
- 作成したワークフローからASLを自動生成
- 作成済みのワークフローの編集も可能

aws

Step Functions Workflow Studio Demo

aws

Workflow Studio Demo

- AWS Step Functionsの同期型Express Workflowを作成
- 作成したワークフローをAPI Gatewayから直接呼び出し
  - Amazon DynamoDBのテーブルを作成
  - Workflow StudioでSynchronous Express Workflowを作成
  - AWS Step Functionsでテストを実行
  - Amazon API Gateway HTTP APIからWorkflowを起動する設定を実施
  - Curlコマンドを実行し結果を確認

aws

まとめ

まとめ

- SAMを利用してサーバーレス環境を容易に構築
- 統合開発環境（IDE）からSAMを実行することが可能
- SAM CLIを利用することで自動化が容易に
- Serverless CI/CDの機能も提供
- AWS Step FunctionsのWorkflow Studioによって
  ワークフローの構築を、より直感的に

aws