aws

ユーザーガイド

AWS Toolkit for VS Code

AWS Toolkit for VS Code: ユーザーガイド

Table of Contents

AWS Toolkit for Visual Studio Code

これは、AWS Toolkit for VS Code のユーザーガイドです。AWS Toolkit for Visual Studio をお探しの場合は、「AWS Toolkit for Visual Studio用ユーザーガイド」を参照してください。

とは AWS Toolkit for Visual Studio Code

Toolkit for VS Code は、Visual Studio コード (VS Code) エディタのオープンソースの拡張機能です。この拡張機能により、開発者は Amazon Web Services (AWS) を使用するサーバーレスアプリケーションの開発、ローカルでのデバッグ、デプロイが簡単になります。

トピック

- AWS Toolkit for Visual Studio Code の使用開始
- AWS サービスとツールの操作

関連情報

ツールキットのソースコードにアクセスしたり、現在未解決の問題を表示したりするには、次のリソースを使用します。

- ソースコード
- 問題追跡

Visual Studio Code エディタの詳細については、https://code.visualstudio.com/ を参照してください。

Amazon Q デベロッパーと Amazon CodeWhisperer

2024 年 4 月 30 日現在、Amazon CodeWhisperer は Amazon Q Developer の一部になりました。これにはインラインコードの提案と Amazon Q Developer セキュリティスキャンが含まれます。VS Code Marketplace から Amazon Q Developer IDE 拡張機能をダウンロードして開始します。

Amazon Q デベロッパーサービスの詳細については、「Amazon Q デベロッパーユーザーガイド」を参照してください。Amazon Q のプランと料金の詳細については、Amazon Q の料金ガイドを参照してください。

Toolkit for VS Code をダウンロードする

IDE の VS Code Marketplace を通じて AWS Toolkit for Visual Studio Code をダウンロード、インストール、セットアップできます。詳細な手順については、このユーザー ガイドの「はじめに」トピックの「ダウンロードとインストール」セクションを参照してください。

VS Code Marketplace からのツールキットのダウンロード

あるいは、Web ブラウザから VS Code Marketplace に移動して、AWS Toolkit for Visual Studio Code インストール ファイルをダウンロードすることもできます。

AWSからのその他の IDE ツールキット

AWS Toolkit for Visual Studio Codeに加えて、JetBrainsおよびVisual AWS Studio のIDE ツールキットも提供しています。

AWS Toolkit for JetBrains リンク

- JetBrains Marketplace から、AWS Toolkit for JetBrainsダウンロードするには、このリンクをクリックしてください。
- AWS Toolkit for JetBrains についての詳細は、AWS Toolkit for JetBrainsユーザーガイド を参照してください。

Toolkit for Visual Studio のリンク

- Visual Studio Marketplace から Toolkit for Visual Studio をダウンロードするには、このリンクをクリックしてください。
- Toolkit for Visual Studio ユーザーガイドを参照してください。

AWS Toolkit for Visual Studio Code の使用開始

AWS Toolkit for Visual Studio Code により、AWS サービスとリソースを VS Code 統合開発環境 (IDE) から直接利用できるようになります。

使用開始の参考になるように、以下のトピックではAWS Toolkit for Visual Studio Codeをセットアップ、インストール、および設定する方法について説明します。

トピック

- AWS Toolkit for Visual Studio Code のインストール
- への接続 AWS
- AWS リージョンの変更
- ツールチェーンの設定

AWS Toolkit for Visual Studio Code のインストール

前提条件

VS Code からAWS Toolkit for Visual Studio Codeの作業を開始するには、次の条件を満たす必要があります。AWS Toolkit for Visual Studio Codeから利用できるすべてのAWSサービスとリソースへのアクセス方法の詳細については、このガイドのthe section called “(オプション) 前提条件”セクションを参照してください。

- VS Code には Windows、macOS、または Linux オペレーティングシステムが必要です。
- AWS Toolkit for Visual Studio Codeには、VS Code バージョン 1.42.0 またはそれ以降のバージョンで作業する必要があります。

VS Code の詳細や VS Code の最新バージョンのダウンロードについては、VS Code のダウンロード Web サイトを参照してください。

AWS Toolkit for Visual Studio Codeのダウンロードとインストール

IDE の VS Code Marketplace から、AWS Toolkit for Visual Studio Code をダウンロード、インストール、セットアップできます。あるいは、Web ブラウザから VS Code Marketplace に移動し

て、AWS Toolkit for Visual Studio Code インストール ファイルをダウンロードすることもできます。

VS Code IDE Marketplace からAWS Toolkit for Visual Studio Codeをインストールする

1. 次のリンクを使用して、VS Code IDE AWS Toolkit for Visual Studio Code で拡張機能を開きます：VS Code Marketplaceを開く。

   > **Note**
   >
   > VS Code がマシン上でまだ実行されていない場合、VS Code のロード中にこの操作に少し時間がかかる場合があります。

2. VS Code Marketplace のAWS Toolkit for Visual Studio Code拡張機能から、[インストール] を選択してインストールプロセスを開始します。
3. プロンプトが表示されたら、VS Code を再起動してインストールプロセスを完了します。

(オプション) 前提条件

AWS Toolkit for Visual Studio Code の特定の機能を使用する前に、以下のものが必要です。

- アマゾンウェブサービス (AWS) アカウント： AWSアカウントはAWS Toolkit for Visual Studio Codeを使用するための必須条件ではありませんが、アカウントがないと機能が大幅に制限されます。AWSアカウントを取得するには、AWSホームページにアクセスしてください。[AWS アカウントを作成する]、または [サインアップを完成する] (以前にサイトにアクセスしたことがある場合) を選択します。
- コード開発 – 使用する言語の関連する SDK。以下のリンクからダウンロードするか、お好みのパッケージマネージャーを使用できます。
  - .NET SDK: https://dotnet.microsoft.com/download
  - Node.js SDK: https://nodejs.org/en/download
  - Python SDK: https://www.python.org/downloads
  - Java SDK: https://aws.amazon.com/corretto/
  - Go SDK: https://golang.org/doc/install
- AWS SAM CLI – この AWS CLI ツールを使用することで、サーバーレスアプリケーションをローカルで開発、テスト、および分析しやすくなります。これは、ツールキットのインストールには必要ありません。ただし、新しいサーバーレスアプリケーションの作成 (ローカル) などの AWS

Serverless Application Model (AWS SAM) 機能に必要なため、次に説明する Docker とともにインストールすることをお勧めします。

詳細については、「AWS Serverless Application Model デベロッパーガイド」の「AWS SAM SDK for JavaScript のインストール」を参照してください。

- Docker – このオープンソースのソフトウェアコンテナプラットフォームは、AWS SAM CLI で必要です。詳細およびダウンロード手順については、「Docker」を参照してください。
- パッケージマネージャー — パッケージマネージャーであり、アプリケーションコードをダウンロードして共有できます。
  - .NET:NuGet
  - Node.js:npm
  - Python:pip
  - Java:Gradle または Maven

への接続 AWS

ほとんどの Amazon Web Services (AWS) リソースは、 AWS アカウントを通じて管理されます。 AWS アカウントは を使用する必要はありませんが AWS Toolkit for Visual Studio Code、 ツールキットの機能は接続なしで制限されます。

以前に AWS アカウントと別の AWS サービス (など AWS Command Line Interface) による認証を設定したことがある場合、 は認証情報 AWS Toolkit for Visual Studio Code を自動的に検出します。

前提条件

を初めて使用する場合、 AWS またはアカウントを作成していない場合は、 を AWS Toolkit for Visual Studio Code AWS アカウントに接続するための 3 つの主なステップがあります。

1. AWS アカウントにサインアップする: サインアップポータル AWS からアカウントにサインアップできます。 AWS新しい AWS アカウントの設定の詳細については、「セットアップユーザーガイド」の「概要」トピックを参照してください。 AWS
2. 認証のセットアップ: から AWS アカウントで認証するには、主に 3 つの方法があります AWS Toolkit for Visual Studio Code。これらの方法の詳細については、本ユーザーガイドの「認証とアクセス」トピックを参照してください。
3. Toolkit AWS から を使用して認証する: このユーザーガイドの以下のセクションの手順を完了することで、 Toolkit から AWS アカウントに接続できます。

サインインパネルを開く

Toolkit AWS サインインパネルを開くには、次のいずれかの手順を実行します。

AWS Explorer から AWS Toolkit サインインパネルを開くには：

1. から AWS Toolkit for Visual Studio Code、EXPLORER を展開します。
2. ... アイコンを選択して、その他のアクション... メニューを展開します。
3. その他のアクション メニューから に接続 AWS を選択して Toolkit AWS サインイン パネルを開きます。

VS Code コマンドパレットを使用して AWS Toolkit サインインパネルを開くには：

1. **Shift+Command+P** (**Ctrl+Shift+P** Windows) を押してコマンドパレットを開きます。
2. 検索フィールドに**AWS: Add a New Connection**「」と入力します。
3. **AWS: Add a New Connection** を選択して Toolkit AWS サインインパネルを開きます。

Toolkit AWS から に接続する

SSO による認証と接続

AWS を使用して を認証して接続するには AWS AWS IAM Identity Center、次の手順を実行します。

> **Note**
>
> AWS Builder ID または IAM Identity Center による認証は、デフォルトのウェブブラウザで AWS 認証ポータルを起動します。認証情報の有効期限が切れるたびに、このプロセスを繰り返して、 AWS アカウントと 間の接続を更新する必要があります AWS Toolkit for Visual Studio Code。

AWS IAM Identity Center で認証して接続する

1. AWS Toolkit のサインインパネルでワークフォースタブを選択し、続行ボタンを選択して続行します。
2. 「IAM Identity Center でサインイン」パネルから、組織の「開始 URL」を入力します。この URL は、会社の管理者またはヘルプデスクから提供されます。

3. ドロップダウンメニューから AWS リージョンを選択します。これは、ID ディレクトリをホストする AWS リージョンです。
4. 続行 ボタンを選択し、デフォルトのウェブブラウザでAWS 認証リクエストのウェブサイトを開くことを確認します。
5. デフォルトのウェブブラウザのプロンプトに従います。認証プロセスが完了すると、ブラウザを閉じて VS Code に戻っても安全であることが通知されます。

IAM 認証情報で認証して接続する

IAM 認証情報 AWS を使用して で認証および接続するには、以下の手順を実行します。

IAM 認証情報で認証して接続する

1. AWS Toolkit のサインインパネルで IAM 認証情報 を選択し、続行ボタンを選択して続行します。
2. 表示されたフィールドに**Secret Key** AWS アカウントの **Profile Name**、**Access Key**、および を入力し、続行 ボタンを選択して設定ファイルにプロファイルを追加し、 Toolkit を AWS アカウントに接続します。
3. 認証が完了して接続が確立されると、Toolkit の AWS Explorer が更新され、 AWS のサービスとリソースが表示されます。

Amazon の認証 CodeCatalyst

Toolkit CodeCatalyst から の使用を開始するには、 AWS Builder ID または IAM Identity Center 認証情報を使用して認証し、接続します。

以下の手順では、 AWS アカウントを使用して Toolkit の認証と接続を行う方法について説明します。

AWS Builder ID を使用して認証して接続する

1. AWS Toolkit のサインインパネルでワークフォースタブを選択し、続行ボタンを選択して続行します。
2. SSO でサインインパネルの上部にある「スキップしてサインイン」リンクを選択します。
3. デフォルトのウェブブラウザのプロンプトに従います。認証プロセスが完了すると、ブラウザを閉じて VS Code に戻っても安全であることが通知されます。

IAM アイデンティティセンターで認証して接続する

1. AWS Toolkit のサインインパネルでワークフォースタブを選択し、続行ボタンを選択して続行します。
2. 「IAM Identity Center でサインイン」パネルから、組織の「開始 URL」を入力します。この URL は、会社の管理者またはヘルプデスクから提供されます。
3. ドロップダウンメニューから AWS リージョンを選択します。これは、ID ディレクトリをホストする AWS リージョンです。
4. 続行 ボタンを選択し、デフォルトのウェブブラウザでAWS 認証リクエストのウェブサイトを開くことを確認します。
5. デフォルトのウェブブラウザのプロンプトに従います。認証プロセスが完了すると、ブラウザを閉じて VS Code に戻っても安全であることが通知されます。

AWS リージョンの変更

AWS リージョンは、 AWS リソースが管理される場所を指定します。から AWS アカウントに接続すると、デフォルトの AWS リージョンが検出され AWS Toolkit for Visual Studio Code、 AWS Explorer に自動的に表示されます。

次のセクションでは、AWS Explorerでリージョンを追加または非表示にする方法について説明します。

AWS Explorer にリージョンを追加する

Explorer にリージョンを追加するには、以下の手順を実行します AWS 。

1. VS Code から、メイン メニューの [表示] を展開し、[コマンド パレット] を選択して、コマンド パレット を開きます。または、次のショートカットキーを使用します。
   - Windows および Linux — **`Ctrl+Shift+P`**を押します。
   - MacOS — **`Shift+Command+P`**を押します。
2. コマンドパレット で、 を検索**`AWS: Show or Hide Regions`**して選択しますAWS。使用可能なリージョンのリストを表示するには、リージョンを表示または非表示にします。
3. リストから、AWS Explorer に追加する AWS リージョンを選択します。
4. OK ボタンを選択して選択内容を確認し、AWS Explorer を更新します。

AWS Explorer からリージョンを非表示にする

AWS Explorer ビューからリージョンを非表示にするには、以下の手順を実行します。

1. AWS Explorer から、非表示にする AWS リージョンを見つけます。
2. 非表示にするリージョンのコンテキスト メニューを開きます (右クリック)。
3. リージョンを表示または非表示 を選択して、VS Code でAWSリージョンを表示または非表示オプションを開きます。
4. AWS Explorer ビューで非表示にするリージョンの選択を解除します。

ツールチェーンの設定

AWS Toolkit for Visual Studio Codeは、すべてのAWSサービスで複数の言語をサポートしています。以下のセクションでは、さまざまな言語用にツールチェーンの設定方法について説明します。

.NET Core 用ツールチェーンを設定する

1. AWS Toolkit for VS Code がインストール済み であることを確認します。
2. C# 拡張機能をインストールします。この拡張機能により、VS Code が .NET Core アプリケーションをデバッグできるようにします。
3. AWS Serverless Application Model (AWS SAM) アプリケーションを開くか、アプリケーションを作成します。
4. `template.yaml` が含まれているフォルダを開きます。

Node.js 用のツールチェーンを設定する

1. AWS Toolkit for VS Code がインストール済み であることを確認します。
2. AWS SAM アプリケーションを開くか、アプリケーションを作成します。
3. `template.yaml` が含まれているフォルダを開きます。

> **Note**
>
> TypeScript Lambda 関数をソース コードから直接デバッグする場合 (起動設定に `"target": "code"` がある)、TypeScript コンパイラをグローバルにインストールするか、プロジェクトの `package.json` にインストールする必要があります。

Python 用のツールチェーンを設定する

1. AWS Toolkit for VS Code がインストール済み であることを確認します。
2. Visual Studio Code の Python 拡張機能をインストールします。この拡張機能により、VS Code は Python アプリケーションをデバッグできます。
3. AWS SAM アプリケーションを開くか、アプリケーションを作成します。
4. `template.yaml` が含まれているフォルダを開きます。
5. アプリケーションのルートにあるターミナルを開き、`python -m venv ./.venv` を実行して `virtualenv` を設定します。

   > **Note**
   >
   > システムごとに `virtualenv` を 1 回のみ設定する必要があります。

6. 次のいずれかを実行して `virtualenv` をアクティブ化します。

   - Bash shell: `./.venv/Scripts/activate`
   - PowerShell: `./.venv/Scripts/Activate.ps1`

Java 用のツールチェーンを構成する

1. AWS Toolkit for VS Code がインストール済み であることを確認します。
2. Java 拡張および Java 11 をインストールします。この拡張機能により、VS Code は Java 関数を認識できるようになります。
3. Java デバッガー拡張 をインストールします。この拡張機能により、VS Code は Java アプリケーションをデバッグできます。
4. AWS SAM アプリケーションを開くか、アプリケーションを作成します。
5. `template.yaml` が含まれているフォルダを開きます。

Go 用のツールチェーンを設定する

1. AWS Toolkit for VS Code がインストール済み であることを確認します。
2. Go Lambda 関数のデバッグには Go 1.14 以上が必要です。
3. Go 拡張機能をインストールします。

> **Note**
>
> Go1.15+ ランタイムをデバッグするには、バージョン 0.25.0 以上が必要です。

4. コマンドパレット を使用して Go ツールをインストールします:
   a. コマンドパレットから、`Go: Install/Update Tools` を選択します。
   b. チェックボックスのセットから、`dlv` および `gopls` を選択します。
5. AWS SAM アプリケーションを開くか、アプリケーションを作成します。
6. `template.yaml` が含まれているフォルダを開きます。

ツールチェーンの使用

ツールチェーンを設定したら、このツールを使用して AWS SAM アプリケーションを 実行またはデバッグ します。

AWS Toolkit for Visual Studio Codeに対する認証とアクセスコントロール

の使用を開始 AWS するために、 で認証する必要はありません AWS Toolkit for Visual Studio Code。ただし、ほとんどの AWS リソースは AWS アカウントを通じて管理されます。すべての AWS Toolkit for Visual Studio Code サービスと機能にアクセスするには、、 AWS Builder ID AWS IAM Identity Center、または IAM 認証情報を使用して認証する必要があります。

以下のトピックでは、各認証情報タイプに関する追加の詳細について説明します。

既存の認証情報 AWS Toolkit for Visual Studio Code を使用して AWS で に接続する方法の詳細については、このユーザーガイドの「 に接続する AWS」トピックを参照してください。

トピック

- AWS IAM アイデンティティセンター
- AWS IAM 認証情報
- AWS デベロッパー向けの Builder ID
- 外部認証情報プロセスの使用

AWS IAM アイデンティティセンター

AWS IAM Identity Center は、 AWS アカウント認証を管理するための推奨されるベストプラクティスです。

ソフトウェア開発キット (SDK) に IAM アイデンティティーセンター を設定する方法の詳細については、「AWS SDK およびツール リファレンス ガイド」の「IAM アイデンティティーセンター 認証」セクションを参照してください。

AWS ツールキットを認証して既存の IAM Identity Center 認証情報に接続する方法の詳細については、このユーザーガイドの「 に接続する AWS」トピックを参照してください。

AWS IAM 認証情報

AWS ローカルに保存されたアクセスキーによる AWS アカウントでの IAM 認証情報認証。

AWS ツールキットを認証して既存の AWS IAM 認証情報に接続する方法の詳細については、このユーザーガイドの「 に接続する AWS」トピックを参照してください。

以下のセクションでは、 から AWS アカウントで認証するために IAM 認証情報を設定する方法について説明します AWS Toolkit for Visual Studio Code。

> **Important**
>
> AWS アカウントで認証するように IAM 認証情報を設定する前に、次の点に注意してください。
>
> - 別の AWS サービス (など AWS CLI) を使用して IAM 認証情報を既に設定している場合、はそれらの認証情報 AWS Toolkit for Visual Studio Code を自動的に検出し、VS Code で使用できるようにします。
> - AWS では、IAM Identity Center 認証の使用を推奨しています。 AWS IAM のベストプラクティスの詳細については、 AWS Identity and Access Management ユーザーガイドの「IAM でのセキュリティのベストプラクティス」セクションを参照してください。
> - セキュリティリスクを避けるため、専用ソフトウェアの開発や実際のデータを扱うときは、IAM ユーザーを認証に使用しないでください。代わりに、AWS IAM Identity Center ユーザーガイドの「IAM アイデンティティセンターとは?」に記載している、アイデンティティプロバイダーによるフェデレーションを使用してください。

「IAM ユーザーの作成」

AWS アカウントで認証 AWS Toolkit for Visual Studio Code するように を設定する前に、「 SDK およびツールリファレンスガイド」の「ステップ 1: IAM ユーザーを作成する」および「ステップ 2: 長期的な認証情報を使用して認証する」トピックでアクセスキーを取得する」を完了する必要があります。 https://docs.aws.amazon.com/sdkref/latest/guide/access-iam-users.html AWS SDKs

> **Note**
>
> AWS SDK およびツールリファレンスガイド の「ステップ 3: 共有認証情報ファイルの更新」はオプションです。
> ステップ 3 を完了すると、 は以下に示す the section called “から共有認証情報ファイルを作成する AWS Toolkit for Visual Studio Code” 中に認証情報 AWS Toolkit for Visual Studio Code を自動的に検出します。

ステップ 3 を完了していない場合は、 で以下の the section called “から共有認証情報ファイルを作成する AWS Toolkit for Visual Studio Code” で説明`credentials file`されているように、 を作成するプロセス AWS Toolkit for Visual Studio Code について説明します。

から共有認証情報ファイルを作成する AWS Toolkit for Visual Studio Code

共有 設定ファイルと共有認証情報ファイルには、 AWS アカウントの設定と認証情報が保存されます。共有の設定と認証情報の詳細については、「AWS Command Line Interface ユーザーガイド」の「構成設定の保存先」セクションを参照してください。

を使用した共有認証情報ファイルの作成 AWS Toolkit for Visual Studio Code

1. **Shift+Command+P** (**Ctrl+Shift+P** Windows) を押してコマンドパレットを開きます。
2. 検索フィールドに**AWS: Add a New Connection**「」と入力します。
3. **AWS: Add a New Connection** を選択して Toolkit AWS サインインパネルを開きます。
4. AWS Toolkit のサインインパネルで、IAM 認証情報 を選択し、続行ボタンを選択して続行します。
5. 表示されたフィールドに**Secret Key** AWS アカウントの **Profile Name**、**Access Key**、および を入力し、続行 ボタンを選択してプロファイルを設定ファイルに追加し、 Toolkit を AWS アカウントに接続します。
6. 認証が完了して接続が確立されると、Toolkit の AWS Explorer が更新され、 AWS のサービスとリソースが表示されます。

> **Note**
>
> この例では、*[Profile_Name]* に構文エラーがあり、認証が失敗すると仮定します。
>
> ```
> [Profile_Name]
> xaws_access_key_id= AKIAI44QH8DHBEXAMPLE
> xaws_secret_access_key= wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
> ```
>
> 認証に失敗した場合に生成されるログメッセージの例を以下に示します。

```
2022-11-02 22:01:54 [ERROR]: Profile [Profile_Name] is not a valid Credential
 Profile: not supported by the Toolkit
2022-11-02 22:01:54 [WARN]: Shared Credentials Profile [Profile_Name] is not
 valid. It will not be used by the toolkit.
```

認証情報プロファイルを追加する

設定ファイルには複数の認証情報を追加できます。これを行うには、コマンドパレット を開き、[AWS Toolkit 作成認証情報プロファイル] を選択します。これにより、認証情報ファイルが開きます。次の例に示すように、このページでは、最初のプロファイルの下に新しいプロファイルを追加できます。

```
# Amazon Web Services Credentials File used by AWS CLI, SDKs, and tools
# This file was created by the AWS Toolkit for Visual Studio Code extension.
#
# Your AWS credentials are represented by access keys associated with IAM users.
# For information about how to create and manage AWS access keys for a user, see:
# https://docs.aws.amazon.com/IAM/latest/UserGuide/id_credentials_access-keys.html
#
# This credential file can store multiple access keys by placing each one in a
# named "profile". For information about how to change the access keys in a
# profile or to add a new profile with a different access key, see:
# https://docs.aws.amazon.com/cli/latest/userguide/cli-config-files.html
#
[Profile1_Name]
# The access key and secret key pair identify your account and grant access to AWS.
aws_access_key_id = AKIAIOSFODNN7EXAMPLE
# Treat your secret key like a password. Never share your secret key with anyone. Do
# not post it in online forums, or store it in a source control system. If your secret
# key is ever disclosed, immediately use IAM to delete the access key and secret key
# and create a new key pair. Then, update this file with the replacement key details.
aws_secret_access_key = wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
[Profile2_Name]
aws_access_key_id = AKIAI44QH8DHBEXAMPLE
aws_secret_access_key = je7MtGbClwBF/2Zp9Utk/h3yCo8nvbEXAMPLEKEY
```

AWS デベロッパー向けの Builder ID

AWS Builder ID は、特定の AWS サービスに対してオプションまたは必須の追加 AWS アカウントです。 AWS Builder ID 認証方法の詳細については、「 サインインユーザーガイド」の AWS 「 Builder ID でAWS サインイン」トピックを参照してください。

AWS ツールキットを認証して既存の AWS Builder ID に接続する方法の詳細については、このユーザーガイドの「 に接続する AWS」トピックを参照してください。

外部認証情報プロセスの使用

`shared config file` を変更することで、AWS によって直接サポートされていない認証情報プロセス用に AWS Toolkit for Visual Studio Code を構成できます。

認証情報プロセス用の `shared config file` の変更は、AWS Toolkit for Visual Studio Code と AWS Command Line Interface の両方で同じです。外部認証情報の設定方法の詳細については、AWS Command Line InterfaceCC ユーザーガイドの「外部プロセスを使用した認証情報のソーシング」トピックを参照してください。

AWS サービスとツールの操作

AWS Toolkit for Visual Studio Code は、 AWS サービス、ツール、リソースを VS Code で直接利用できるようにします。以下は、各 Toolkit for VS Code サービスとその機能について説明するガイドトピックのリストです。サービスまたはツールを選択すると、その機能、設定方法、基本的な機能の操作方法の詳細が表示されます。

トピック

実験的な機能の使用

正式にリリースされる前に、実験的な機能は AWS Toolkit for Visual Studio Code の機能への早期アクセスを提供します。

> **Warning**
>
> 実験的な機能は引き続きテストおよび更新されるため、ユーザビリティに問題がある可能性があります。実験的な特徴は、通知なしに AWS Toolkit for Visual Studio Code から削除される可能性があります。

VS Code IDE の [設定] ペインの [AWSツールキット] セクションで、特定の AWS サービスへの実験的な機能を有効にできます。

1. VS Code で AWS 設定を編集するには、[ファイル]、[環境設定]、[設定] の順に選択します。
2. [設定] ペインで、[拡張機能] を展開し [AWSツールキット] を選択します。
3. AWS: 実験で、リリース前にアクセスする実験的機能のチェックボックスをオンにします。実験的な機能をオフにするには、該当するチェックボックスをオフにします。

AWS Explorer での AWS サービスの使用

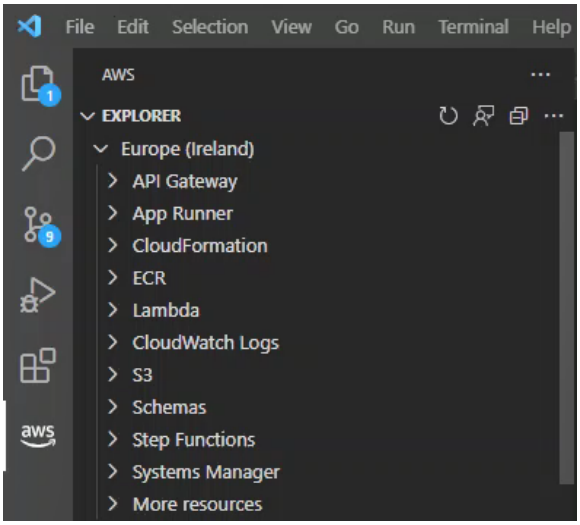
AWS Explorer では、 の使用時に操作できるいくつかの AWS サービスを表示できます AWS Toolkit for Visual Studio Code。

このセクションでは、VS Code で AWS Explorer にアクセスして使用する方法について説明します。ここでは、システムに Toolkit for VS Code がインストーおよび設定済みであることを前提としています。

いくつかの重要なポイント:

- ツールキットがインストールされ、正しく設定されている場合は、[AWS Explorer] に項目が表示されます。AWS Explorer を見るには、 アクティビティバー で AWS アイコンを選択します。

  例:



- 特定の機能には、特定の AWS アクセス許可が必要です。例えば、 AWS アカウントの関数を表示するには AWS Lambda 、 で設定した認証情報に、少なくとも読み取り専用の Lambda アクセス許可が含まれている認証とアクセス必要があります。各機能に必要なアクセス許可の詳細については、以下のトピックを参照してください。
- AWS Explorer にすぐに表示されない AWS サービスを操作する場合は、その他のリソースに移動し、インターフェイスに追加できる数百のリソースから選択できます。

  例えば、使用可能なリソースタイプの選択から AWS ToolkitCodeArtifact::Repository を選択できます。このリソースタイプがその他のリソース に追加されると、エントリを展開して、独自のプロパティと属性を持つ異なる CodeArtifact リポジトリを作成するリソースのリストを表示できます。さらに、リソースのプロパティと属性を JSON 形式のテンプレートで記述できます。このテンプレートは、 AWS クラウドに新しいリソースを作成するために保存できます。

VS Code 用 Amazon CodeCatalyst

Amazon CodeCatalyst とは?

Amazon CodeCatalyst は、ソフトウェア開発チーム向けのクラウドベースのコラボレーションスペースです。AWS Toolkit for Visual Studio Codeを使用することで、VS Code から直接 CodeCatalyst リソースを表示し管理することができます。また、AWS Toolkit を使用して VS Code を実行する開発環境の仮想コンピューティング環境を起動することで、クラウド内で直接作業することもできます。CodeCatalyst サービスの詳細については、「Amazon CodeCatalyst」ユーザーガイドを参照してください。

次のトピックでは、Toolkit for VS CodeCatalyst を操作する方法について説明します。

トピック

Toolkit for VS Codeの使用を始めるには

VS CodeCatalyst の使用を始めるには、次のステップを実行します。

トピック

VS Code をインストールする

開始する前に、最新バージョンの VS Code を使用していることを確認します。VS Code の最新バージョンをダウンロードするには、VS Code ウェブサイトの「VS Code のダウンロード」を参照してください。

Toolkit for VS Code をインストールする

Toolkit for VS Code 拡張機能の最新バージョンをインストールして、VToolkit for VS Code を CodeCatalyst アカウントに接続します。Toolkit をインストールするには VS Code 拡張機能: Marketplace から直接 AWS Toolkit をインストールします。

VS Code 拡張機能: Marketplace を開き、Toolkit for VS Code インストールするには、次の手順を実行します。

1. メイン画面の左端にある VS Code アクティビティバーから、拡張機能アイコンを選択し、「拡張機能:Marketplace」ビューを開きます。

> **Note**
>
> VS Code コマンドパレットを使用して [拡張機能:Marketplace] ビューを開くこともできます。これを行うには、`Command+Shift+P` (macOS) または `Ctrl+Shift+P` (Windows) を押してコマンドパレットを開きます。次に、`Extensions: Focus on Marketplace View`表示された検索フィールドに入力し、「拡張機能:Marketplace ビューにフォーカス」を選択してMarketplaceビューを開きます。

2. 「拡張機能:マーケットプレイス」検索バーから、`AWS Toolkit`と入力します。
3. 入力すると検索結果が絞り込まれます。AWSToolkit の拡張機能が使用可能な場合は、[インストール] を選択して AWS Toolkit の最新バージョンのインストールを開始します。

CodeCatalyst アカウントの作成

AWS Toolkit の最新バージョンのインストールに加えて、CodeCatalyst を Toolkit for VS Code に接続するには、アクティブな AWS Builder ID が必要です。アクティブな AWS Builder ID または CodeCatalyst アカウントがない場合は、「CodeCatalyst」ユーザーガイドの「CodeCatalyst の設定」セクションを参照してください。

> **Note**
>
> AWS Builder ID は、 IAM 認証情報とは異なります。IAM 認証情報は、AWS Toolkit からアクセスできるほとんどの AWS のサービス で必要になります。対照的に、新しい CodeCatalyst アカウントを作成してAWS Toolkit に接続するには、AWS Builder ID が必要です。

AWS Toolkit と CodeCatalyst の接続

AWS Toolkit を CodeCatalyst アカウントに接続するには、次の手順を実行します。

1. Toolkit for VS Code から、デベロッパーツールエクスプローラを展開します。
2. デベロッパーツールエクスプローラーから CodeCatalyst を展開し、[開始] を選択します。
3. VS Code のプロンプトが表示されたら、お好みの Web ブラウザーで AWS Builder ID ポータルを開いて続行します。

> **Note**
>
> AWSBuilder ID をお持ちでない場合は、提供されているリンクを使用してBuilder ID を作成してください。次に、次の手順を実行して、AWS Builder ID の認証情報を認証して Toolkit for VS Code に接続します。

4. AWSBuilder ID ポータルから、手順に従って AWSBuilder ID を認証します。プロンプトが表示されたら、Toolkit for VS Code にデータへのアクセスを[許可する] を選択します。
5. AWS Builder ID の認証情報が Toolkit for VS Code に正常に接続されると、ブラウザーが自動的に更新され、ブラウザーを閉じて Toolkit for VS Code に戻っても安全であることが示されます。

> **Note**
>
> Toolkit for VS Code が IAM 認証情報ですでに認証されている場合、VS Code は次のプロンプトを開きます。
> これまで使用してきたツールの中には、AWSBuilder ID では動作しないものがあります。AWS認証情報を使用している間、AWS Builder ID をバックグラウンドで使用し続けますか?
>
> - AWS Builder ID を使用して CodeCatalyst サービスを認証し、IAM 資格情報を使用して他の Toolkit for VS Code サービスを認証する場合は、[はい、他のサービスで IAM 資格情報を使用しながら、CodeCatalyst で AWS Builder ID を使用し続けます] を選択します。
> - AWSBuilder ID のみを使用してすべての Toolkit for VS Code サービスの認証を続行するには、[いいえ、すべてを AWSBuilder ID による認証に切り替える] を選択します。

6. AWS Builder ID が Toolkit for VS Code に正常に接続されると、デベロッパーツールエクスプローラーは CodeCatalyst の機能を更新し、 AWSBuilder ID 接続ステータスを表示します。

VS Code での Amazon CodeCatalyst リソースの操作

次のセクションでは、Toolkit for VS CodeCatalyst のリソース管理機能の概要について説明します。

開発環境の詳細と CodeCatalyst からアクセスする方法については、「Amazon CodeCatalyst」ユーザーガイドの「開発環境」セクションを参照してください。

次のセクションでは、VS Code から開発環境を作成、開き、操作する方法について説明します。

トピック

リポジトリのクローンを作成する

CodeCatalyst はクラウドベースのサービスで、CodeCatalyst プロジェクトで作業するにはクラウドに接続する必要があります。プロジェクトをローカルで行いたい場合は、CodeCatalyst リポジトリをローカルマシンに複製して、次にクラウドに接続したときに CodeCatalyst プロジェクトとオンラインで同期できます。

CodeCatalyst アカウントから Toolkit を使用して VS Code にリポジトリをクローンするには、次の手順を実行します。AWS

1. Toolkit for VS Code から、デベロッパーツールエクスプローラを展開します。
2. CodeCatalyst を展開し、「クローン・リポジトリー」を選択します。
3. 「CodeCatalyst リポジトリの選択」ダイアログで、複製するリポジトリを検索して選択し、「クローンするフォルダーの選択」ダイアログを開きます。
4. 「リポジトリの場所を選択」を選択してプロンプトを閉じ、リポジトリのクローニングを開始します。

   > **Note**
   >
   > サードパーティーのサービスからリポジトリをクローニングする場合、そのサービスの認証情報を使用して認証するように求められることがあります。
   > リポジトリがクローンされている間、VS Code はクローニングリポジトリのステータスウィンドウに進行状況を表示します。リポジトリがクローンされたら、クローンされたリポジトリを開きますか? メッセージが表示されます。

5. ダイアログウィンドウから次のいずれかを選択してクローニングプロセスを完了します。

   - 現在の VS Code ウィンドウでリポジトリを開くには、「開く」を選択します。

- リポジトリを新しい VS Code ウィンドウで開くには、[新しいウィンドウで開く] を選択します。
- リポジトリを開かずにクローニングプロセスを完了するには、ダイアログウィンドウを閉じます。

開発環境を開く

VS Code で既存の開発環境を開くには

1. Toolkit for VS Code から、デベロッパーツールエクスプローラを展開します。
2. CodeCatalyst を展開し、「開発環境を開く」を選択して VS Code の「CodeCatalyst 開発環境を選択」ダイアログを開きます。
3. CodeCatalyst 開発環境を選択ダイアログから、開きたい開発環境を選択します。

> **Note**
>
> 開発環境を選択すると、開発環境を開いて VS Code を CodeCatalyst に接続するプロセスが開始されます。このプロセス中、VS Code は CodeCatalyst のステータスウィンドウに進行状況の更新を表示します。プロセスが完了すると、ステータスウィンドウが更新されます。
>
> - Dev Environment が開かないと、ステータスが更新され、プロセスが失敗した理由に関する情報と、プロセスログを開くためのリンクが表示されます。
> - プロセスが成功すると、VS Code の新しいウィンドウに開発環境が開きます。

CodeCatalyst で開発環境を作成する

新しい開発環境を作成する方法

1. Toolkit for VS Code から、デベロッパーツールエクスプローラを展開します。
2. CodeCatalyst を展開し、「開発環境を作成」オプションを選択して VS Code の「CodeCatalyst 開発環境の作成」メニューを開きます。
3. 「ソースコードセクションから、以下のいずれかのオプションを選択します。

   - 既存の CodeCatalyst リポジトリを使用する:既存の CodeCatalyst リポジトリから開発環境を作成します。CodeCatalyst プロジェクトとブランチを選択する必要があります。

- 空の開発環境を作成:空の開発環境を作成します。

4. (オプション) Alias セクションから、開発環境の代替名を入力します。

> **Note**
>
> 必須ではありませんが、エイリアスを指定することをおすすめします。エイリアスを指定すると Dev Environment の整理と検索が簡単になるからです。

5. (オプション)「開発環境設定」セクションから、特定のニーズに合わせて以下の設定を変更します。

- コンピューティング:「コンピューティングの編集」を選択して、システムに割り当てられる処理能力と RAM の量を変更します。
- タイムアウト:「タイムアウトの編集」を選択すると、開発環境が停止するまでに許容されるシステムアイドル時間を変更できます。

> **Note**
>
> 開発環境は作業を永続的に保存します。これは、作業内容を失うことなく開発環境を停止できることを意味します。開発環境を停止することで、開発環境を稼働させ続けるために必要なコストを削減できます。

- ストレージ:「ストレージサイズの編集」を選択して、システムに割り当てるストレージ容量を変更します。

> **Note**
>
> ストレージは、開発環境の作成後に変更できない唯一の設定です。

6. [開発環境の作成] を選択して、新しいクラウド開発環境を作成します。

> **Note**
>
> VS Code は、開発環境の作成の進行状況をステータスウィンドウに表示します。開発環境が作成されると、VS Code は開発環境を新しいウィンドウで開き、「このフォルダ内のファイルの作成者を信頼しますか?」というプロンプトも表示されます。利用規約に同意して、開発環境で作業を続行してください。

サードパーティのリポジトリから開発環境を作成する

リポジトリをソースとしてリンクすることで、サードパーティのリポジトリから開発環境を作成できます。

ソースとしてサードパーティのリポジトリにリンクすると、CodeCatalyst のプロジェクトレベルで処理されます。サードパーティのリポジトリを開発環境に接続する方法の手順や詳細については、「Amazon CodeCatalyst ユーザーガイド」の「ソースリポジトリをリンクする」トピックを参照してください。

VS Code の CodeCatalyst コマンド

CodeCatalyst 関連の機能や機能には、 Toolkit には直接表示されない VS Code コマンドが他にもあります。AWS

CodeCatalyst に割り当てられているコマンドのリストをコマンドパレットから表示するには:

1. Toolkit for VS Code から、デベロッパーツールエクスプローラを展開します。
2. [Show CodeCatalyst Commands] を選択して、`CodeCatalyst` の検索が事前に入力されたコマンド パレットを開きます。
3. リストから CodeCatalyst コマンドを選択してアクティブにします。

開発環境と VS コードでの作業

Toolkit for VS Code から、開発環境として知られる VS Code 仮想コンピューティング環境を起動できます。開発環境はカスタマイズ可能なクラウド開発環境であり、スペース内のさまざまなチームメンバー間でコピーや共有をすることができます。

以下のセクションでは、開発環境をローカルおよびリモートで構成する方法と、開発環境で起動したときに Toolkit for VS Code からアクセスできる基本機能について説明します。開発環境の詳細については、CodeCatalyst ガイドの「開発環境」を参照してください。

トピック

- devfiles を使用した開発環境の構成
- AWSIAM 認証情報を開発環境に接続する
- 開発環境でToolkit for VS Code の使用

devfiles を使用した開発環境の構成

開発環境の devfile で、開発環境にインストールされるプロジェクト固有のツールとアプリケーションライブラリを定義できます。devfile 仕様は、YAML ドキュメントに書き込むオープンスタンダードです。

VS Code では、devfile をローカルでもリモートでも編集できます。このプロセスは VS Code 互換のファイルを作成または編集するのと似ています。プロジェクトのソース リポジトリから CodeCatalyst で Devfile を直接編集することもできます。CodeCatalyst から開発ファイルを編集する方法については、『CodeCatalyst ガイド』の「開発環境の設定」セクションを参照してください。

> **Important**
>
> DevfileからVS Code を編集する場合は、次の点に注意してください。
>
> - devfile の名前または devfile コンポーネント名を変更すると、ルートディレクトリの内容が置き換えられます。以前のコンテンツはすべて失われ、回復できません。
> - ルートフォルダに devfile がない状態で Dev Environment を作成したり、ソースリポジトリに関連付けられていない Dev Environment を作成したりすると、開発環境の作成時にデフォルトの構成設定を含む devfile が生成されます。
> - Devfileを定義して設定する方法については、devfile.io Web サイトの「コマンドの追加」ドキュメントを参照してください。

VS Code のローカルインスタンスでDevfileを編集するには:

1. VS Code エクスプローラからdevfile.yamlファイルを見つけて VS Code エディタで開きます。
2. ドキュメントに変更を加えて保存します。
3. 変更をコミットし、その変更をソースリポジトリにプッシュします。
4. 次回 Dev Environment を起動すると、Devfileで定義されている仕様に合わせて設定が更新されます。

AWSIAM 認証情報を開発環境に接続する

開発環境内のすべての Toolkit for VS Code サービスにアクセスするには、AWS IAM 認証情報を開発環境に接続する必要があります。開発環境に IAM 認証情報を接続するには、以下を実行します。

1. 開発環境の Toolkit for VS Code EXPLORER から、「リソースを表示するには IAM 認証情報を選択」を選択し、「接続の切り替え」ダイアログを開きます。
2. 使用する IAM 認証情報をリストから選択するか、[Add New Connection] を選択して新しい認証情報を追加します。
3. 以下のオプションから選択し、プロンプトに従って新しい接続を作成します。

   - 個人用のメールを使用してサインアップし、AWS Builder ID でサインイン:個人用の新しい AWS Builder ID を作成します。このオプションは、CodeCatalyst などの AWS Builder ID 専用サービスの使用のみをサポートします。
   - AWSIAM ID センターを使用して接続: 会社の IAM アクセスポータルにサインインできます。このオプションは IAM Identity Center へのアクセス権を持つ会社のみが利用できます。このアカウントからアクセスできるAWSリソースとサービスは、アカウントを所有する会社によって管理されます。
   - IAM 認証情報を使用する: Toolkit for VS Code から AWS サービスとリソースを操作できるようにします。このオプションだけでは、CodeCatalyst などの AWS Builder ID 専用サービスの使用はサポートされません。
4. IAM 認証情報が開発環境の Toolkit for VS Code に正常に接続されると、AWSEXPLORER が AWSサービスとリソースで更新されます。

開発環境でToolkit for VS Code の使用

VS Code で開発環境を開くか作成したら、VS Code のローカルインスタンスから行うのと同様に、VS Code 用 Toolkit から作業できます。VS Code を実行する開発環境は、AWS Toolkit を自動的にインストールし、AWS Builder ID で接続するように設定されています。

開発環境の停止

現在の開発環境を停止するには:

1. Toolkit for VS Code から、デベロッパーツールエクスプローラを展開します。
2. CodeCatalyst を展開し、「開発環境を停止」を選択します。

3. VS Code のプロンプトが表示されたら、開発環境を停止することを確認します。開発環境が正常に停止すると、VS Code はリモート接続を閉じ、ローカルの開発インスタンスに戻ります。

開発環境設定を開く

現在の開発環境の設定を開くには:

1. Toolkit for VS Code から、デベロッパーツールエクスプローラを展開します。
2. CodeCatalyst を展開し、[設定を開く] を選択して、現在の開発環境の [開発環境設定] ビューを開きます。
3. [Dev Environment Settings] (開発環境設定) ビューで、以下のセクションに開発環境のオプションが表示されます。

   - Alias (エイリアス): 開発環境に割り当てられているエイリアスを表示および変更します。
   - Status (ステータス) : 現在の開発環境のステータス、割り当てられているプロジェクトを表示し、開発環境を停止します。
   - [Devfile:] 開発環境の `Devfile` の名前と場所を表示します。`Devfile`を開くには、[エディタで開く]ボタンを選択します。
   - Compute Settings (コンピューティング設定): 開発環境のサイズとデフォルトのタイムアウトまでの長さを変更します。

> **Note**
>
> 開発環境を作成した後に割り当てたストレージ容量を変更することはできません。

開発環境の設定

開発環境`devfile`を開き、現在の開発環境を設定するには、次のステップを実行します。devfile の編集について詳しくは、本ガイドの「devfiles による開発環境の設定」セクションに記載されているリソースを参照してください。

1. Toolkit for VS Code から、デベロッパーツールエクスプローラを展開します。
2. CodeCatalyst を展開し、「開発ファイルを開く」を選択すると、現在の開発環境内の新しいエディタウィンドウで `devfile.yaml` を開きます。

Amazon CodeCatalyst と VS コードのトラブルシューティング

以下のトピックでは、Amazon CodeCatalyst と VS Code を使用する際に発生する可能性のある技術的問題について説明します。

トピック

VS コードバージョン

ご使用のバージョンの VS Code では、システムに `vscode://` URI のハンドラーを設定することが想定されます。このハンドラーがないと、 Toolkit から CodeCatalyst のすべての機能にアクセスすることはできません。AWSたとえば、VS Code Insider から起動すると エラーが発生します。これは、VS Code Insider は `vscode-insiders://` URI を処理し、`vscode://` URI は処理しないためです。

Amazon CodeCatalyst の使用許可

AWS Toolkit for Visual Studio Code から CodeCatalyst を操作するためのファイル権限要件は次のとおりです。

- `~/.ssh/config`ファイルに対する独自のアクセス権限を`read`および`write`に設定します。他のすべてのユーザーに対して `write` 権限を制限します。
- `~/.ssh/id_dsa` および `~/.ssh/id_rsa` ファイルのアクセス許可を `read` のみに設定します。他のすべてのユーザーに対して `read`、`write`、および `execute` 権限を制限します。
- `globals.context.globalStorageUri.fsPath`ファイルは書き込み可能な場所になければなりません。

Toolkit for VS Code から接続する

AWS Toolkit for Visual Studio Code から開発環境に接続しようとすると、次のエラーを受け取った場合

`~/.ssh/config` には古い可能性のある `aws-devenv-*` セクションがあります。

- 設定を開く...ボタンを選択して、VS Code Editorで `~/.ssh/config`ファイルを開きます。

- エディタから、`Host aws-devenv-*`セクションの内容を選択して削除します。
- `~/.ssh/config` の `Host aws-devenv-*` に加えた変更を保存します。ファイルを閉じます。
- Toolkit for VS Code から接続し直します。

Amazon API Gateway の使用

AWS Toolkit for Visual Studio Code を使用して接続されている AWS アカウントで、リモート API Gateway リソースを参照して実行できます。

> **Note**
>
> この機能は、デバッグをサポートしていません。

リモート API Gateway リソースを参照して実行するには

1. AWSExplorer で、API Gateway を選択し、メニューを拡張します。リモート API Gateway リソースが一覧表示されます。
2. 呼び出す API Gateway リソースを見つけ、そのコンテキスト (右クリック) メニューを開き、呼び出すAWS を選択してください。
3. パラメータフォームで、呼び出しパラメータを指定します。
4. リモート API Gateway リソースを実行するには、呼び出しを選択します。結果は、VS コード出力ビューに表示されます。

AWS App Runner で を使用する AWS Toolkit for Visual Studio Code

AWS App Runner は、ソースコードまたはコンテナイメージから AWS クラウドのスケーラブルで安全なウェブアプリケーションに直接デプロイするための、高速でシンプルで費用対効果の高い方法を提供します。これを使用すると、新しいテクノロジーを学習したり、使用するコンピューティングサービスを決定したり、 AWS リソースのプロビジョニングと設定方法を知っている必要はありません。

を使用して AWS App Runner 、ソースイメージまたはソースコード に基づいてサービスを作成および管理できます。ソースイメージを使用する場合は、イメージリポジトリに保存されているパブリッ

クまたはプライベートコンテナイメージを選択できます。App Runner は以下のイメージリポジトリプロバイダーをサポートしています。

- Amazon Elastic Container Registry (Amazon ECR): AWS アカウントにプライベートイメージを保存します。
- Amazon Elastic Container Registry Public (Amazon ECR Public): パブリックに読み取り可能なイメージを保存します。

ソースコードオプションを選択した場合、サポートされているリポジトリプロバイダーによって管理されているソースコードリポジトリからデプロイできます。現在、App Runner はソースコードリポジトリプロバイダーGitHubとして をサポートしています。

前提条件

を使用して App Runner を操作するには、以下 AWS Toolkit for Visual Studio Code が必要です。

- AWS アカウント
- AWS Toolkit for Visual Studio Code その機能の バージョン AWS App Runner

これらのコア要件に加えて、関連するすべての IAM ユーザーが App Runner サービスを操作するアクセス許可を持っている必要があります。また、コンテナイメージ URI や GitHub リポジトリへの接続など、サービスソースに関する特定の情報を取得する必要があります。この情報は、App Runner サービスを作成するときに必要です。

App Runner の IAM アクセス許可の設定

App Runner に必要なアクセス許可を付与する最も簡単な方法は、既存の AWS 管理ポリシーを関連する AWS Identity and Access Management (IAM) エンティティ、特にユーザーまたはグループにアタッチすることです。App Runner には、IAM ユーザーにアタッチできる 2 つのマネージドポリシーが用意されています。

- `AWSAppRunnerFullAccess`: ユーザーがすべての App Runner アクションを実行できるようにします。
- `AWSAppRunnerReadOnlyAccess`: App Runner リソースの詳細をリストおよび表示できます。

また、サービスソースとして Amazon Elastic Container Registry (Amazon ECR) からプライベートリポジトリを選択した場合は、App Runner サービス用に次のアクセスロールを作成する必要があります。

- `AWSAppRunnerServicePolicyForECRAccess`: アプリランナーは、アカウント内の Amazon Elastic Container Registry (Amazon ECR) イメージにアクセスできるようにします。

VS Codeのコマンドパレットでサービスインスタンスを設定するときに、このロールを自動的に作成できます。

> **Note**
>
> AWSServiceRoleForAppRunner サービスにリンクされたロールは AWS App Runner 、 が以下のタスクを完了できるようにします。
>
> - Amazon CloudWatch Logs ロググループにログをプッシュします。
> - Amazon Elastic Container Registry (Amazon ECR) イメージプッシュをサブスクライブする Amazon CloudWatch Events ルールを作成します。
>
> サービスリンクロールを手動で作成する必要はありません。 AWS App Runner で を作成するか、 によって呼び出される API オペレーション AWS Management Console を使用して を作成すると AWS Toolkit for Visual Studio Code、 によってこのサービスにリンクされたロールが自動的に AWS App Runner 作成されます。

詳細については、AWS App Runner デベロッパーガイドの「App Runner の Identity and Access Management」を参照してください。

App Runner のサービスソースの取得

AWS App Runner を使用して、ソースイメージまたはソースコードからサービスをデプロイできます。

Source image

ソースイメージからデプロイする場合は、プライベートまたはパブリックイメージレジストリからその AWS イメージのリポジトリへのリンクを取得できます。

- Amazon ECR プライベートレジストリ: Amazon ECR コンソール (https://console.aws.amazon.com/ecr/repositories) を使用するプライベートリポジトリの URI をコピーします。
- Amazon ECR パブリックレジストリ: Amazon ECR Public Gallery (https://gallery.ecr.aws/) を使用するパブリックリポジトリの URI をコピーします。

> **Note**
>
> プライベート Amazon ECR リポジトリの URI を Toolkit for VS Code の AWS Explorer から直接取得するには、次の手順を実行します。
>
> - AWS Explorer を開き、ECR ノードを展開して、その AWS リージョンのリポジトリのリストを表示します。
> - リポジトリを右クリックし、[Copy Repository URI] (リポジトリ URI をコピー) を選択して、リンクをクリップボードにコピーします。

イメージリポジトリの URI は、VS Code の コマンドパレット でサービスインスタンスを設定するときに指定します。

詳細については、「AWS App Runner デベロッパーガイド」の「ソースイメージに基づいた App Runner サービス」を参照してください。

Source code

ソースコードを AWS App Runner サービスにデプロイするには、そのコードを、サポートされているリポジトリプロバイダーが管理する Git リポジトリに保存する必要があります。App Runner は、1 つのソースコードリポジトリプロバイダー をサポートしていますGitHub。

GitHub リポジトリの設定については、「 の開始方法のドキュメント」を参照してください GitHub。

GitHub リポジトリから App Runner サービスにソースコードをデプロイするために、App Runner は への接続を確立します GitHub。リポジトリがプライベート (でパブリックにアクセスできない GitHub) の場合は、App Runner に接続の詳細を提供する必要があります。

> **Important**
>
> GitHub 接続を作成するには、App Runner コンソール (https://console.aws.amazon.com/apprunner) を使用して、 GitHub にリンクする接続を作成する必要があります AWS。VS Code の コマンドパレット を使用してサービスインスタンスを設定するときに、接続ページで使用可能なGitHub接続を選択できます。
> 詳細については、AWS App Runner デベロッパーガイドの「App Runner 接続の管理」を参照してください。

App Runner サービスインスタンスは、コードの構築と実行を許可するマネージドランタイムを提供します。 AWS App Runner 現在、 は次のランタイムをサポートしています。

- Python マネージドランタイム
- Node.js マネージランタイム

サービス設定の一環として、App Runner サービスがサービスをビルドして開始する方法に関する情報を指定します。この情報は、[Command Palette] (コマンドパレット) を使用して入力するか、YAML 形式の App Runner 設定ファイルを指定します。このファイルの値は、App Runner にサービスを構築して開始し、ランタイムコンテキストを提供する方法を指示します。これには、関連するネットワーク設定と環境変数が含まれます。 設定ファイルの名前は `apprunner.yaml` です。設定ファイルは、アプリケーションのリポジトリのルートディレクトリに自動的に追加されます。

料金

アプリケーションが使用するコンピューティングリソースとメモリリソースに対して課金されます。また、デプロイを自動化する場合は、1 か月間のすべての自動デプロイを提供する各アプリケーションに対して設定された月額料金も支払います。ソースコードからデプロイする場合は、App Runner がソースコードからコンテナをビルドするのにかかる時間に対して、ビルド料金を追加で支払います。

詳細については、「AWS App Runner の料金」を参照してください。

トピック

- App Runner サービスの作成

- App Runner サービスの管理

App Runner サービスの作成

Toolkit for VS Code で App Runner サービスを作成するには、AWS Explorer および VS Code の コマンドパレット を使用します。特定の AWS リージョンでサービスを作成することを選択した場合、 コマンドパレットによって提供される番号付きのステップが、アプリケーションが実行されるサービスインスタンスを設定するプロセスをガイドします。

App Runner サービスを作成する前に、前提条件 を満たしてください。これには、関連する IAM 権限の提供と、デプロイする特定のソースリポジトリの確認が含まれます。

App Runner サービスを作成するには

1. Explorer をまだ開いていない場合は AWS 、開きます。
2. App Runner ノードを右クリックして、[Create Service] (サービスの作成) を選択します。

   コマンドパレット ディスプレイ。
3. [Select a source code location type] (ソースコードの場所タイプを選択する) では、[ECR] または [Repository] (リポジトリ) を選択します。

   [ECR] を選択した場合は、Amazon Elastic Container Registry が管理するリポジトリ内のコンテナイメージを指定します。[Repository] (リポジトリ) を選択した場合は、サポートされているリポジトリプロバイダーが管理するソースコードリポジトリを指定します。現在、App Runner はソースコードリポジトリプロバイダーGitHubとして をサポートしています。

ECR からのデプロイ

1. [Select or enter an image repository] (イメージリポジトリを選択または入力する) では、Amazon ECR プライベートレジストリまたは Amazon ECR Public Gallery によって管理されるイメージリポジトリの URL を選択または入力します。

   > **Note**
   >
   > Amazon ECR Public Gallery からリポジトリを指定する場合は、App Runner が ECR パブリックリポジトリ内のイメージの自動デプロイをサポートしていないため、自動デプロイがオフになっていることを確認してください。

自動デプロイはデフォルトでオフになっています。この場合、コマンドパレットヘッダーのアイコンには斜線が表示されます。自動デプロイをオンにすると、このオプションには追加料金がかかることを示すメッセージが表示されます。

2. コマンドパレットのステップに No tags found (タグが見つかりません) と報告された場合は、タグ付けされたコンテナイメージを含むリポジトリを選択するステップに戻る必要があります。
3. Amazon ECR プライベートレジストリを使用している場合は、ECR アクセスロールAppRunnerである ECR AccessRoleが必要です。これにより、App Runner はアカウント内の Amazon Elastic Container Registry (Amazon ECR) イメージにアクセスできます。コマンドパレットヘッダーの「+」アイコンを選択して、このロールを自動的に作成します。(イメージが一般公開されている Amazon ECR Public にイメージが保存されている場合は、アクセスロールは必要ありません)。
4. [Port] (ポート) には、サービスが使用する IP ポートを入力します (例: ポート `8000`)。
5. [Configure environment variables] (環境変数の設定) では、サービスインスタンスの動作のカスタマイズに使用する環境変数を記述したファイルを指定できます。このステップはスキップすることもできます。
6. [Name your service] (サービスの名前) では、一意の名前 (スペースは使用できません) を入力し、Enter を押します。
7. [Select instance configuration] (インスタンス設定の選択) では、サービスインスタンスの CPU ユニット数とメモリ (GB) を選択します。

   サービスの作成時に、そのステータスは [Creating] (作成中) から [Running] (実行中) に変わります。
8. サービスの実行が開始されたら、サービスを右クリックし、[Copy Service URL] (サービス URL のコピー) を選択します。
9. デプロイ済みのアプリケーションにアクセスするには、コピーした URL をウェブブラウザのアドレスバーに貼り付けます。

リモートリポジトリからのデプロイ

1. 接続の選択 で、 にリンク GitHubする接続を選択します AWS。選択可能な接続は、App Runner コンソールGitHub の接続ページに表示されます。
2. リモート GitHub リポジトリを選択 で、リモートリポジトリの URL を選択または入力します。

   Visual Studio Code のソース管理管理 (SCM) で既に構成されているリモートリポジトリを選択できます。リストにない場合は、リポジトリへのリンクを貼り付けることもできます。

3. [Select a branch] (ブランチの選択) では、デプロイするソースコードの Git ブランチを選択します。
4. [Choose configuration source] (設定ソースの選択) では、ランタイム設定の定義方法を指定します。

   [Use configuration file] (設定ファイルを使用) を選択すると、サービスインスタンスは `apprunner.yaml` 設定ファイルで定義された設定を使用します。このファイルは、アプリケーションのリポジトリのルートディレクトリにあります。

   [Configure all settings here] (ここですべて設定する) を選択した場合は、[コマンドペイン]を使用して、以下の項目を指定します。

   - [Runtime] (ランタイム): [Python 3] または [Nodejs 12] を選択します。
   - [Build command] (ビルドコマンド): サービスインスタンスのランタイム環境でアプリケーションをビルドするコマンドを入力します。
   - [Start command] (開始コマンド): サービスインスタンスのランタイム環境でアプリケーションを開始するコマンドを入力します。
5. [Port] (ポート) には、サービスが使用する IP ポートを入力します (例: ポート `8000`)。
6. [Configure environment variables] (環境変数の設定) では、サービスインスタンスの動作のカスタマイズに使用する環境変数を記述したファイルを指定できます。このステップはスキップすることもできます。
7. [Name your service] (サービスの名前) では、一意の名前 (スペースは使用できません) を入力し、Enter を押します。
8. [Select instance configuration] (インスタンス設定の選択) では、サービスインスタンスの CPU ユニット数とメモリ (GB) を選択します。

   サービスの作成時に、そのステータスは [Creating] (作成中) から [Running] (実行中) に変わります。
9. サービスの実行が開始されたら、サービスを右クリックし、[Copy Service URL] (サービス URL のコピー) を選択します。
10. デプロイ済みのアプリケーションにアクセスするには、コピーした URL をウェブブラウザのアドレスバーに貼り付けます。

> **Note**
>
> App Runner サービスの作成に失敗すると、Explorer でのサービスのステータス表示が [Create failed] (作成に失敗しました) になります。トラブルシューティングのヒントについては、App Runner デベロッパーガイド の「サービスの作成に失敗した場合」を参照してください。

App Runner サービスの管理

App Runner サービスを作成したら、 AWS Explorer ペインを使用して以下のアクティビティを実行して管理できます。

- App Runner サービスの一時停止と再開
- App Runner サービスの展開
- App Runner のログストリームの表示
- App Runner サービスの削除

App Runner サービスの一時停止と再開

ウェブアプリケーションを一時的に無効にしてコードの実行を停止する必要がある場合は、 AWS App Runner サービスを一時停止できます。App Runner は、サービスのコンピューティング容量をゼロに削減します。アプリケーションを再度実行する準備ができたら、App Runner サービスを再開します。App Runner は、新しいコンピューティングキャパシティをプロビジョニングし、アプリケーションをデプロイして、アプリケーションを実行します。

> **Important**
>
> App Runner が稼働しているときにのみ課金されます。したがって、コストを管理するために、必要に応じてアプリケーションを一時停止および再開できます。これは、開発およびテストシナリオで特に役立ちます。

App Runner サービスを一時停止するには

1. Explorer をまだ開いていない場合は AWS 、開きます。
2. [App Runner] を展開して、サービスのリストを表示します。

3. サービスを右クリックし、[Pause] (一時停止) を選択します。
4. 表示されるダイアログボックスで、[Confirm] (確認) を選択します。

   サービスが一時停止している間に、サービスのステータスは [Running] (実行中) から [Pausing] (一時停止中) に変わり、その後 [Paused] (一時停止) に変わります。

App Runner サービスを再開するには

1. Explorer をまだ開いていない場合は AWS 、開きます。
2. [App Runner] を展開して、サービスのリストを表示します。
3. サービスを右クリックし、[Resume] (再開) を選択します。

   サービスが再開している間に、サービスのステータスは [Resuming] (再開中) から [Running] (実行中) に変わります。

App Runner サービスの展開

サービスの手動デプロイオプションを選択した場合は、サービスへの各デプロイを明示的に開始する必要があります。

1. Explorer をまだ開いていない場合は AWS 、開きます。
2. [App Runner] を展開して、サービスのリストを表示します。
3. サービスを右クリックし、[Start Deployment] (デプロイの開始) を選択します。
4. アプリケーションがデプロイされている間に、サービスのステータスは [[Deploying] (デプロイ中) から [Running] (実行中) に変わります。
5. アプリケーションが正常にデプロイされたことを確認するには、同じサービスを右クリックし、[Copy Service URL] (サービス URL のコピー) を選択します。
6. デプロイされたウェブアプリケーションにアクセスするには、コピーした URL をウェブブラウザのアドレスバーに貼り付けます。

App Runner のログストリームの表示

CloudWatch ログを使用して、App Runner などのサービスのログストリームをモニタリング、保存、およびアクセスします。ログストリームは、同じソースを共有する一連のログイベントです。

1. App Runner を展開して、サービスインスタンスのリストを表示します。

2. 特定のサービスインスタンスを展開して、ロググループのリストを表示します。ロググループは、保持、監視、アクセス制御について同じ設定を共有するログストリームのグループです。
3. ロググループを右クリックし、[View Log Streams] (ログストリームの表示) を選択します。
4. コマンドパレット から、グループからログストリームを選択します。

   VS Code エディタには、ストリームを構成するログイベントのリストが表示されます。古いイベントまたは新しいイベントをエディタにロードするかを選択できます。

App Runner サービスの削除

> **Important**
>
> App Runner サービスを削除すると、そのサービスは完全に削除され、保存されたデータは削除されます。サービスを再作成する必要がある場合は、App Runner がソースを再度取得し、コードリポジトリの場合はビルドする必要があります。ウェブアプリケーションは、新しい App Runner ドメインを取得します。

1. Explorer をまだ開いていない場合は AWS 、開きます。
2. [App Runner] を展開して、サービスのリストを表示します。
3. サービスを右クリックし、[Delete Service] (サービスの削除) を選択します。
4. コマンドパレット で、[削除] を入力し、[入力] を押して確認します。

   削除されるサービスのステータスには、[Deleting] (削除中) が表示され、その後このサービスはリストから削除されます。

AWS Application Composer

AWS Toolkit for Visual Studio Code を使用して AWS Application Composer サービスを操作できます。AWSApplication Composer は、アプリケーションアーキテクチャの設計と AWS CloudFormation インフラストラクチャの視覚化を支援する、AWS アプリケーション用のビジュアルビルダーです。

AWS Application Composer サービスの詳細については、「AWS Application Composer ユーザーガイド」を参照してください。

以下のトピックでは、AWS Toolkit for Visual Studio Code から AWS Application Composer を使用する方法について説明します。

トピック

Toolkit から AWS Application Composer を使用する

AWS Toolkit for Visual Studio Code 用の AWS Application Composer では、インタラクティブなキャンバスを使用してアプリケーションを視覚的にデザインできます。また、Application Composer を使用して、AWS CloudFormation および AWS Serverless Application Model (AWS SAM) テンプレートを視覚化および変更することもできます。Application Composer での作業中、変更内容は永続的に保存されるため、VS Code エディタでのファイルの直接編集と、インタラクティブなキャンバスの使用との間でシームレスに切り替えることができます。

AWS Application Composer サービスの詳細、開始方法、チュートリアルについては、「AWS Application Composer サービスユーザーガイド」を参照してください。

以下のセクションでは、AWS Toolkit for Visual Studio Code から AWS Application Composer サービスにアクセスする方法について説明します。

Toolkit から AWS Application Composer にアクセスする

Toolkit から AWS Application Composer にアクセスするには、主に 3 つの方法があります。

既存のテンプレートから AWS Application Composer にアクセスする

1. VS Code から、VS Code エディタで既存のテンプレートファイルを開きます。
2. エディタウィンドウで、右上隅にある [AWS Application Composer] ボタンをクリックします。
3. VS Code エディタウィンドウで AWS Application Composer が開き、テンプレートファイルが表示されます。

コンテキストメニュー (右クリック) から AWS Application Composer にアクセスする

1. VS Code から、AWS Application Composer で開くテンプレートファイルを右クリックします。
2. コンテキストメニューで、[App Composer で開く] オプションを選択します。
3. 新しい VS Code エディタウィンドウで AWS Application Composer が開き、テンプレートファイルが表示されます。

コマンドパレットから AWS Application Composer にアクセスする

1. VS Code で **Cmd + Shift + P** または **Ctrl + Shift + P** (Windows) を押して、コマンドパレットを開きます。
2. 検索フィールドに「**AWS Application Composer**」と入力して結果が表示されたら、[AWSApplication Composer] を選択します。
3. 開くテンプレートファイルを選択すると、新しい VS Code エディタウィンドウで AWS Application Composer が開き、テンプレートファイルが表示されます。

VSコードの AWS CDK

これはプレビューリリースの機能に関するプレリリースドキュメントです。このドキュメントは変更される可能性があります。

AWS CDK サービス を使用すると、AWS Cloud Development Kit (AWS CDK) アプリケーションまたは アプリ を操作できます。詳細については、「AWS Cloud Development Kit (AWS CDK)デベロッパーガイド」の「AWS CDK」を参照できます。

AWS CDK アプリケーションは コンストラクト として知られるビルディングブロックで構成されており、AWS CloudFormation スタックや AWS リソースの定義を含んでいます。AWS CDK Explorer を使用して、AWS CDK コンストラクトで定義されている スタック および リソース を視覚化できます。この視覚化は、Visual Studio Code (VS Code) エディター内の [デベロッパーツール] ペインのツリービューで提供されます。

このセクションでは、VS Code エディターで、AWS CDK　にアクセスして使用する方法について説明します。ここでは、システムに Toolkit for VS Code が　インストールと設定済みであることを前提としています。

トピック

- AWS CDK アプリケーションの使用

AWS CDK アプリケーションの使用

これはプレビューリリースの機能に関するプレリリースドキュメントです。このドキュメントは変更される可能性があります。

視覚化するために AWS Toolkit for VS Code にある AWS CDK Explorer を使用し、AWS CDK アプリケーションで作業を行います。

前提条件

- システムが、「Toolkit for VS Code のツールキットをインストールする」で指定されている前提条件を満たしていることを確認してください。
- 「AWS Cloud Development Kit (AWS CDK)デベロッパーガイド」の「AWS CDKの使用開始」の最初の数セクションで記載されているように、AWS CDK コマンドラインインターフェイスをインストールします。

> **Important**
>
> AWS CDK のバージョンは 1.17.0 以降である必要があります。コマンドラインで **`cdk --version`** を使用して、実行しているバージョンを確認します。

AWS CDK アプリケーションを視覚化する

AWS Toolkit for VS Code AWS CDK Explorer を使用して、アプリケーションの CDK コンストラクトに保存されているスタック と リソースを管理できます。AWS CDK Explorer は、**`cdk synth`** コマンドの実行時に作成される `tree.json` ファイルで定義された情報を使用して、リソースをツリービューに表示します。デフォルトでは、`tree.json` ファイルはアプリケーションの `cdk.out` ディレクトリにあります。

Toolkit AWS CDK Explorer の使用を開始するには、CDK アプリケーションを作成します。

1. AWS CDK デベロッパーガイド で、Hello World のチュートリアル の最初のいくつかのステップを完了します。

   > **Important**
   >
   > 「スタックのデプロイ」ステップに達したら、操作を中止してこのガイドに戻ってください。

> **Note**
>
> チュートリアルで提供されているコマンド、例えば オペレーティングシステムのコマンドライン上、または VS Code エディタ内の ターミナル 内の **`mkdir`** および **`cdk init`** を実行できます。

2. CDK チュートリアルの必要なステップを完了したら、 VS Code エディタで作成した CDK コンテンツを開きます。
3. AWS ナビゲーションペインで、CDK (プレビュー) の見出しを展開します。CDK アプリケーションとその関連リソースが CDK Explorer のツリービューに表示されます。

重要な注意事項

- VS Code エディタに CDK アプリをロードするときに、一度に複数のフォルダをロードすることができます。各フォルダには、前のイメージに示すように、複数の CDK アプリを含めることができます。AWS CDK エクスプローラは、プロジェクトのルートディレクトリとその直下のサブディレクトリ内でアプリを検索します。

- チュートリアルの最初のいくつかのステップを実行すると、最後に実行するコマンドは **`cdk synth`** であることに気付きます。これにより、`tree.json` ファイルが生成されます。例えば、リソースの追加など、CDK アプリの側面を変更する場合は、そのコマンドを再度実行して、変更がツリービューに反映されていることを確認する必要があります。

AWS CDK アプリでのその他のオペレーションの実行

VS Code エディタを使用して、オペレーティングシステムのコマンドラインやその他のツールを使用する場合と同様に、CDK アプリで他のオペレーションを実行できます。例えば、エディタでコードファイルを更新し、VS Code ターミナル ウィンドウを使用してアプリをデプロイできます。

これらのタイプのアクションを試すには、VS コードエディタを使用して、「AWS CDKデベロッパーガイド」の「Hello World のチュートリアル」を継続します。AWS アカウントに予測外の費用が発生しないように、最後のステップ「アプリのリソースの破棄」を必ず実行してください。

AWS CloudFormation スタックの操作

AWS Toolkit for Visual Studio Code は、AWS CloudFormation をサポートしています。Toolkit for VS Code を使用して、スタックの削除など、AWS CloudFormation で特定のタスクを実行できます。

トピック

- AWS CloudFormation スタックの削除
- を使用して AWS CloudFormation テンプレートを作成する AWS Toolkit for Visual Studio Code

AWS CloudFormation スタックの削除

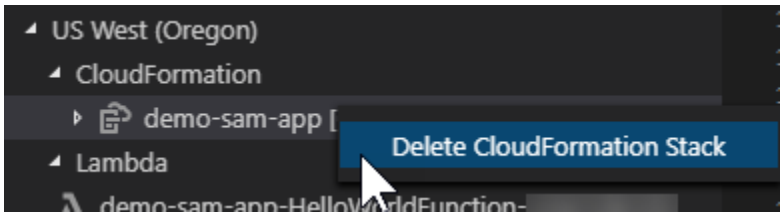
AWS Toolkit for Visual Studio Code を使用して AWS CloudFormation スタックを削除できます。

前提条件

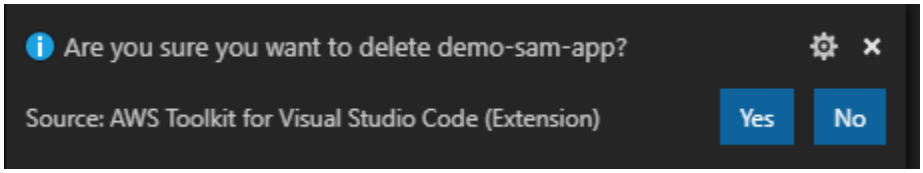
- システムが、「Toolkit for VS Code をインストールする」で指定されている前提条件を満たしていることを確認してください。
- 「認証とアクセス」で設定した認証情報に、AWS CloudFormation サービスに対する適切な読み取り/書き込みアクセス許可が含まれていることを確認します。AWSExplorer 内、CloudFormation の下で、「CloudFormation リソースの読み込みエラー」のようなメッセージが表示される場合は、これらの認証情報にアタッチされた許可をチェックしてください。アクセス許可に変更を加えると、VS Code の AWSExplorer に影響するまで数分かかります。

CloudFormation スタックを削除する

1. [AWS Explorer] で、削除する AWS CloudFormation スタックのコンテキストメニューを開きます。



2. [Delete CloudFormation Stack] (CloudFormation スタックを削除する) を選択します。
3. 表示されるメッセージで、[はい] を選択して削除を確認します。



スタックが削除されると、AWS Explorer に表示されなくなります。

を使用して AWS CloudFormation テンプレートを作成する AWS Toolkit for Visual Studio Code

AWS Toolkit for Visual Studio Code は、 AWS CloudFormation および SAM テンプレートの記述に役立ちます。

前提条件

Toolkit for VS Code

- Toolkit for VS Code から CloudFormation サービスにアクセスする前に、ユーザーガイド「Toolkit for VS Code のインストール」で説明されている要件を満たす必要があります。
- で作成した認証情報には、 AWS CloudFormation サービスへの適切な読み取り/書き込みアクセスが含まれている認証とアクセス必要があります。

> **Note**
>
> CloudFormation サービスに CloudFormation 「リソースのロード中にエラーが発生しました」というメッセージが表示された場合は、それらの認証情報にアタッチしたアクセス許可を確認してください。また、権限を変更すると AWS Explorer で更新されるまでに数分かかる場合があることにも注意してください。

CloudFormation テンプレートの前提条件

- Redhat Developer YAML VS コードエクステンションをインストールして有効にします。
- Redhat Developer YAML VS Code エクステンションを使用するときは、インターネットに接続する必要があります。これは、JSON スキーマをマシンにダウンロードしてキャッシュするために使用されるからです。

YAML スキーマサポートを使用した CloudFormation テンプレートの記述

このツールキットは YAML 言語サポートと JSON スキーマを使用して、 CloudFormation および SAM テンプレートの記述プロセスを効率化します。構文検証やオートコンプリートなどの機能により、処理が速くなるだけでなく、テンプレートの品質も向上します。テンプレートのスキーマを選択する際には、以下のベストプラクティスが推奨されます。

CloudFormation テンプレート

- ファイルには .yaml または .yml という拡張子が付いています。
- ファイルには最上位 `AWSTemplateFormatVersion` または Resources ノードがあります。

SAM テンプレート

- についてすでに説明されているすべての基準 CloudFormation
- このファイルには、`AWS::Serverless`で始まる値を含む最上位の Transform ノードがあります。

スキーマはファイルの変更時に適用されます。たとえば、サーバーレス変換をテンプレートに追加してファイルを保存した後、SAM CloudFormation テンプレートスキーマが適用されます。

IAM ポリシー構文

YAML エクステンションはテンプレートに型検証を自動的に適用します。これにより、特定のプロパティの型が無効なエントリが強調表示されます。強調表示されたエントリの上にカーソルを置くと、拡張機能に修正アクションが表示されます。

オートコンプリート

新しいフィールド、列挙値、またはその他のリソースタイプを追加する場合、Ctrl + space を入力することで YAML 拡張機能のオートコンプリート機能を起動できます。

AWS Toolkit for Visual Studio Code ツールキットを使って CloudWatch Logs を使用

Amazon CloudWatch Logs により、使用中のすべてのシステム、アプリケーション、AWS のサービスからのログを、スケーラビリティに優れた 1 つのサービスで一元管理することができます。これ

により、ログを簡単に表示したり、特定のエラーコードやパターンを検索したり、特定のフィールドに基づいてフィルタリングしたり、将来の分析のために安全にアーカイブしたりできます。詳細については、「Amazon CloudWatch ユーザーガイド」の「Amazon CloudWatch Logs とは」を参照してください。

次のトピックでは、AWS Toolkit for Visual Studio Codeを使用して AWS アカウント内の CloudWatch Logs を操作する方法について説明します。

トピック

- AWS Toolkit for Visual Studio Code を使用して CloudWatch ロググループとログストリームの表示
- AWS Toolkit for Visual Studio Code を使用して、ログストリーミングで CloudWatch ログイベントを操作
- CloudWatch ロググループを検索する

AWS Toolkit for Visual Studio Code を使用して CloudWatch ロググループとログストリームの表示

ログストリームは、同じソースを共有する一連のログイベントです。CloudWatch Logs のログのソースごとに、別個のログストリームとなります。

ロググループは、保持、モニタリング、アクセス制御について同じ設定を共有するログストリームのグループです。ロググループを定義して、各グループに入れるストリームを指定できます。1 つのロググループに属することができるログストリーミングの数に制限はありません。

詳細については、「Amazon CloudWatch ユーザーガイド」の「ロググループとログストリーミングを操作する」を参照してください。

トピック

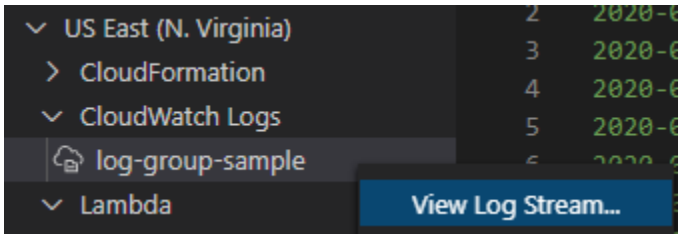
- CloudWatch Logs ノードのロググループとログストリームの表示

CloudWatch Logs ノードのロググループとログストリームの表示

1. VS Codeで、AWS Explorer を開くには 表示、Explorer を選択します。
2. CloudWatch Logs ノードをクリックして、ロググループのリストを展開します。

   現在のAWSリージョンのロググループは CloudWatch Logs ノードの下に表示されます。

3. ロググループのログストリーミングを表示するには、ロググループの名前を右クリックした後、ログストリーミングの表示を選択します。



4. コマンドパレットから、表示するログストリームをグループから選択します。

> **Note**
>
> コマンドパレットは、各ストリームの最後のイベントのタイムスタンプを表示します。

[ログストリーム] エディタ を起動して、ストリームのログイベントを表示します。

AWS Toolkit for Visual Studio Code を使用して、ログストリーミングで CloudWatch ログイベントを操作

Log Steam エディタを開いた後、各ストリームのログイベントにアクセスできます。ログイベントは、モニタリングされているアプリケーションまたはリソースによって記録されたアクティビティのレコードです。

トピック

- ログストリーム情報の表示とコピー
- ログストリームエディタのコンテンツをローカルファイルに保存します。

ログストリーム情報の表示とコピー

ログストリーミングを開くと、[ログストリーミング] エディタには、そのストリーミングのログイベントのシーケンスが表示されます。

1. 表示するログストリーミングを探す場合、[ログストリーミング] エディタを開きます (CloudWatch ロググループとログストリームの表示 を参照)。

   イベントをリストする各行にはタイムスタンプがつき、ログに記録された時刻を示します。

2. 次のオプションを使用して、ストリームのイベントに関する情報を表示およびコピーできます。

   - イベントを時間別に表示: 新しいイベントのロードまたは古いイベントのロードを選択して、最新ログイベントと古いログイベントを表示。

     > **Note**
     >
     > [ログストリーミング] エディタは最新の 10,000 行のログイベントまたは 1 MB のログデータ (どちらか小さい方) のバッチを最初にロードします。新しいイベントをロードを選択すると、最後のバッチをロードした後にログに記録されたイベントをエディタが表示します。古いイベントをロードを選択すると、現在表示されているイベントの前に発生したイベントのバッチをエディタが表示します。

   - ログイベントのコピー: コピーするイベントを選択した後に右クリックしてメニューから [コピー] を選択します。
   - ログストリーミング名をコピー: [ログストリーミング] エディタのタブを右クリックし、[ログストリーミング名のコピー] を選択します。

     > **Note**
     >
     > コマンド パレットを使用して、AWSToolkitでログストリーム名をコピー を実行することもできます。

ログストリームエディタのコンテンツをローカルファイルに保存します。

CloudWatch ログストリーミングエディタのコンテンツを `log` ローカルマシン上のファイルにダウンロードできます。

> **Note**
>
> このオプションで、ログストリーミングエディタに現在表示されているログイベントのみをアーカイブに保存する許可がでます。例えば、ログストリーミングの合計サイズが 5 MB で、エディタに 2 MB しかロードされていない場合、保存されたファイルには 2 MB のログデータしか含まれません。保存するデータをさらに表示するには、エディタで新しいイベントをロードまたは古いイベントをロードを選択します。

1. コピーするログストリーミングを検索するには、[ログストリーミング] エディタを開きます (CloudWatch ロググループとログストリームの表示 参照)。
2. ログストリームの名前を表示するタブの横にある 保存 アイコンを選択します。

> **Note**
>
> また、コマンドパレットを使用して、AWSToolkit で現在のログストリームコンテンツを保存を実行できます。

3. ダイアログボックスを使用して、ログファイルのダウンロードフォルダを選択または作成し、[Save] をクリックします。

CloudWatch ロググループを検索する

[ロググループの検索] を使用して、ロググループ内のすべてのログストリームを検索できます。

Amazon CloudWatch Logs サービスの詳細については、Amazon CloudWatch ユーザーガイドの「ロググループとログストリームを操作」トピックを参照してください。

VS Code コマンドパレットからのロググループの検索

VS Code コマンドパレットからロググループを検索するには、次のステップを完了してください。

Amazon CloudWatch Logs のフィルターとパターンの詳細については、Amazon CloudWatch ユーザーガイドの「フィルターとパターンの構文」セクションを参照してください。

1. VS Code から **cmd+shift+p** (windows:**ctrl+shift+p**) を押してコマンドパレットを開きます。
2. コマンドパレットからコマンド**AWS: Search Log Group**を入力して選択し、VS Code の検索ロググループダイアログを開き、プロンプトに従って続行します。

> **Note**
>
> 最初のプロンプトから、次の手順に進む前にAWSリージョンを切り替えることができます。

3. ロググループの選択(1/3) プロンプトから、検索するロググループを選択します。
4. 「時間フィルターの選択 (2/3)」プロンプトから、検索に適用する時間フィルターを選択します。

5. ロググループの検索... (3/3) プロンプトで、表示されたフィールドに検索パターンを入力し、**Enter**キーを押して検索を続行するか、**ESC**キーを押して検索をキャンセルします。
6. 検索が完了すると、VS Code エディタで検索結果が開きます。

AWSExplorer からロググループを検索する

AWS Toolkit for Visual Studio CodeExplorer からロググループを検索するには、次のステップを完了してください。

1. AWS Toolkit for Visual Studio CodeExplorer から CloudWatch を展開します。
2. 検索する検索ロググループのコンテキストメニュー (右クリック) を開き、[ロググループの検索] を選択して、検索プロンプトを開きます。
3. プロンプトに従い、時間枠を選択して続行します。
4. プロンプトが表示されたら、表示されたフィールドに検索パターンを入力し、**Enter**キーを押して続行するか、**ESC**キーを押して検索をキャンセルします。
5. 検索が完了すると、VS コードエディタに検索結果が表示されます。

検索ログ結果の処理

CloudWatch ロググループの検索が正常に完了すると、検索結果が VS Code エディタで開きます。次の手順では、検索ログ結果を操作する方法を示します。

> **Note**
>
> 1 つのログストリームを表示する場合、以下の機能は現在アクティブなログストリームの結果に限定されます。

検索ロググループの結果を保存します。

検索ロググループの結果をローカルに保存するには、次のステップを完了してください。

1. 検索ロググループの結果から、VS Code エディタの右上隅にある [ログをファイルに保存] アイコンボタンを選択します。
2. [名前を付けて保存] プロンプトから、ファイルを保存する名前と場所を指定します。
3. 「OK」 を選択すると、ファイルはローカルマシンに保存されます。

時間範囲、時間範囲を変更します。

検索ロググループの結果に表示される時間範囲を変更するには、次の手順を実行します。

1. 検索ロググループの結果から [日付で検索...] を選択します。 VS Code エディタの右上隅にあるアイコンボタン。
2. [時間フィルターの選択] プロンプトから、検索ログ結果の新しい時間範囲を選択します。
3. 「時間フィルターの選択」 プロンプトが閉じると、結果が更新されます。

検索パターンの変更

検索ロググループの結果に表示される検索パターンを変更するには、次の手順を実行します。

1. 検索ロググループの結果から、「パターンで検索...」を選択します。 VS Code エディタの右上隅にあるアイコンボタン。
2. ロググループの選択 プロンプトから、表示されたフィールドに新しい検索パターンを入力します。
3. **Enter**キーを押してプロンプトを閉じ、結果を新しい検索パターンで更新します。

Amazon Elastic Container Registry の使用

Amazon Elastic Container Registry (Amazon ECR) は、セキュリティ、スケーラビリティ備えた AWS マネージドコンテナイメージレジストリサービスです。VS Code エクスプローラーのツールキットからいくつかの Amazon ECR サービス関数にアクセスできます。

- リポジトリの作成
- リポジトリまたはタグ付きイメージの AWS App Runner サービスの作成。
- イメージタグおよびリポジトリの URI または ARN にアクセスする。
- イメージタグとリポジトリを削除する。

AWS CLI およびその他のプラットフォームと VS Code で統合し、VS Code コンソールを使用して Amazon ECR の全機能にアクセスすることもできます。

Amazon ECR の詳細については、Amazon Elastic Container Registry ユーザーガイドの「Amazon ECR とは?」を参照してください。

前提条件

VS Code Explorer から Amazon ECR サービスにアクセスするには、これらのステップを完了する必要があります。

IAM ユーザーを作成する

AWS サービスにアクセスする前に、Amazon ECR などのサービスは、認証情報を提供する必要があります。これにより、サービスのリソースにアクセスする権限の有無が確認されます。ルート AWS アカウントの認証情報を使用して AWS に直接アクセスすることはお勧めしません。その代わりに、AWS Identity and Access Management (IAM) を使用して IAM ユーザーを作成し、このユーザーを管理権限のある IAM グループに追加します。これで、特別な URL と IAM ユーザーの認証情報を使って AWS にアクセスできます。

AWS にサインアップし、ご自分の IAM ユーザーを作成していない場合は、IAM コンソールを使用して作成できます。

管理者ユーザーを作成するには、以下のいずれかのオプションを選択します。

管理者を管理する方法を 1 つ選択します	To	By	以下の操作も可能
IAM Identity Center 内 (推奨)	短期の認証情報を使用して AWS にアクセスします。 これはセキュリティのベストプラクティスと一致しています。ベストプラクティスの詳細については、IAM ユーザーガイドの「IAM でのセキュリティのベストプ	AWS IAM Identity Center ユーザーガイドの「開始方法」の手順に従います。	AWS Command Line Interface ユーザーガイドの「AWS IAM Identity Center を使用するための AWS CLI の設定」に従って、プログラムによるアクセスを設定します。

管理者を管理する方法を 1 つ選択します	To	By	以下の操作も可能
	ラクティス」を参照してください。		
IAM 内 (非推奨)	長期認証情報を使用して AWS にアクセスする。	IAM ユーザーガイドの「最初の IAM 管理者のユーザーおよびグループの作成」の手順に従います。	IAM ユーザーガイドの「IAM ユーザーのアクセスキーの管理」に従って、プログラムによるアクセスを設定します。

この新しい IAM ユーザーとしてサインインするには、AWS からサインアウトし、次の URL を使用します。以下の URL で、your_aws_account_id はハイフンを除いた AWS アカウント番号です (例えば AWS アカウント番号が 1234-5678-9012 であれば、AWS アカウント ID は 123456789012 です)。

```
https://your_aws_account_id.signin.aws.amazon.com/console/
```

作成した IAM ユーザー名とパスワードを入力します。サインインすると、ナビゲーションバーには「your_user_name @ your_aws_account_id」と表示されます。

サインインページの URL に AWS のアカウント ID を含めない場合は、アカウントのエイリアスを作成できます。IAM ダッシュボードから [カスタマイズ] を選択し、[アカウントエイリアス] を入力します。これは、会社名です。詳細については、「IAM ユーザーガイド」の「AWS アカウント ID とそのエイリアス」を参照してください。

アカウントエイリアスを作成した後、サインインするには、次の URL を使用します。

```
https://your_account_alias.signin.aws.amazon.com/console/
```

アカウントの IAM ユーザーのサインインリンクを確認するには、IAM コンソールを開き、ダッシュボードの [IAM users sign-in link] の下を確認します。

IAM の詳細については、「AWS Identity and Access Management ユーザーガイド」を参照してください。

Docker をインストールして構成する

Docker をインストールして構成するには、Docker エンジンのインストール から好ましいオペレーティングシステムを選択し、指示に従います。

AWS CLI バージョン 2 のインストールおよび設定

AWS CLI バージョン 2 ユーザーガイドのインストール、更新、アンインストールから好ましいオペレーティングシステムを選択して、AWS CLI バージョン 2 をインストールして設定します。

トピック

- VS Code での Amazon Elastic コンテナレジストリサービスの操作

VS Code での Amazon Elastic コンテナレジストリサービスの操作

Amazon Elastic Container Registry (Amazon ECR) サービスには、VS Code AWS から直接アクセスでき、それを使用してプログラムイメージを Amazon ECR リポジトリにプッシュします。開始するには、次の手順を実行する必要があります。

1. イメージの構築に必要な情報を含む Dockerfile を作成します。
2. その Dockerfile からイメージをビルドし、処理のためにイメージにタグを付けます。
3. Amazon ECR インスタンス内にリポジトリを作成します。
4. リポジトリにタグ付けされたイメージをプッシュします。

セクション

- 前提条件
- 1. Dockerfile の作成
- 2. Dockerfile からイメージをビルドする
- 3. 新規レポジトリを作成します
- 4. イメージのプッシュ、プル、削除

前提条件

Toolkit for VS Code の Amazon ECR サービス機能を使用するには、こうした前提条件を満たす必要があります。

1. Dockerfile の作成

Docker は Dockerfile というファイルを使用して、リモートリポジトリにプッシュおよび保存できるイメージを定義します。ECR リポジトリにイメージをアップロードする前に、Dockerfile を作成し、その Dockerfile からイメージをビルドする必要があります。

Dockerfile の作成

1. Toolkit for VS Code Explorer を使用して、Dockerfile を保存するディレクトリに移動します。
2. Dockerfile という名前の新しいファイルを作成します。

> **Note**
>
> VS Code は、ファイルタイプまたはファイル拡張子を選択するように促す場合があります。この問題が発生した場合は、プレーンテキストを選択します。Vs Code には「dockerfile」拡張子があります。ただし、使用することは推奨されていません。これは、拡張機能が特定のバージョンの Docker または他の関連アプリケーションと競合する可能性があるためです。

VS Code を使用して Dockerfile を編集する

Dockerfile にファイル拡張子がある場合は、そのファイルのコンテキスト (右クリック) メニューを開き、ファイル拡張子を削除します。

Dockerfile からファイル拡張子を削除したら、次の操作を行います。

1. 空の Dockerfile を VS Code で直接開きます。
2. 次の例の内容を Dockerfile にコピーします。

   Example Dockerfile イメージテンプレート

```
FROM ubuntu:18.04
```

```
# Install dependencies
RUN apt-get update && \
 apt-get -y install apache2

# Install apache and write hello world message
RUN echo 'Hello World!' > /var/www/html/index.html

# Configure apache
RUN echo '. /etc/apache2/envvars' > /root/run_apache.sh && \
 echo 'mkdir -p /var/run/apache2' >> /root/run_apache.sh && \
 echo 'mkdir -p /var/lock/apache2' >> /root/run_apache.sh && \
 echo '/usr/sbin/apache2 -D FOREGROUND' >> /root/run_apache.sh && \
 chmod 755 /root/run_apache.sh

EXPOSE 80

CMD /root/run_apache.sh
```

これは Ubuntu 18.04 イメージを使用する Dockerfile です。実行命令は、パッケージキャッシュを更新します。ウェブサーバー用のいくつかのソフトウェアがインストールされてから、「Hello World!」 ウェブサーバーのドキュメントルートに書き込まれます。EXPOSE の命令はコンテナ上のポート 80 を公開し、CMD の命令はウェブサーバーを起動します。

3. Dockerfile を保存します。

> **Important**
>
> Dockerfile の名前に拡張子が付いていないことを確認してください。拡張子を持つ Dockerfile は、Docker の特定のバージョンやその他の関連アプリケーションと競合する可能性があります。

2. Dockerfile からイメージをビルドする

作成した Dockerfile には、プログラムのイメージを構築するために必要な情報が含まれています。そのイメージを Amazon ECR インスタンスにプッシュする前に、まずイメージをビルドする必要があります。

Dockerfile からイメージを作成する

1. Docker CLI または Docker のインスタンスと統合された CLI を使用して、Dockerfile を含むディレクトリに移動します。
2. Docker ビルドコマンドを実行して、Dockerfile で定義されているイメージをビルドします。

```
docker build -t hello-world .
```

3. docker images コマンドを実行して、イメージが正しく作成されたことを確認します。

```
docker images --filter reference=hello-world
```

Example 出力例:

```
REPOSITORY          TAG                 IMAGE ID            CREATED
 SIZE
hello-world         latest              e9ffedc8c286        4 minutes ago
 241MB
```

4.

> **Note**
>
> この手順は、イメージの作成やプッシュには必要ありませんが、プログラムイメージの実行時の動作を確認できます。

新しいビルトイメージを実行するには、Docker 実行 コマンドを使用します。

```
docker run -t -i -p 80:80 hello-world
```

-p オプションは、前の例で指定され、コンテナ上の 80 ポート からホストシステム 80 ポート にエキスポートされます。。Docker をローカルに実行している場合は、ブラウザで http://localhost:80 を参照します。プログラムが正常に実行された場合、「Hello World!」 ステートメントが表示されます。

Docker run コマンドの詳細については、Docker ウェブサイトの「Docker run reference」を参照してください。

3. 新規レポジトリを作成します

Amazon ECR インスタンスにイメージをアップロードするには、保存できる新しいリポジトリを作成します。

Amazon ECR リポジトリを作成します。

1. VSコードアクティビティバー から、AWS Toolkit アイコン を選択します。
2. AWS Explorer メニューを拡張します。
3. デフォルトに関連付けられているデフォルトの AWS リージョン AWS を見つけます。次に、それを選択して、VS Code の Toolkit を介したサービスのリストを表示します。
4. ECR + オプションを選択し、リポジトリの新規作成プロセスを開始します。
5. プロンプトに従ってプロセスを完了します。
6. 完了したら、AWS Explorer メニューの ECR セクションから新しいレポジトリにアクセスできます。

4. イメージのプッシュ、プル、削除

Dockerfile からイメージを構築してリポジトリを作成したら、イメージを Amazon ECR リポジトリにプッシュできます。さらに、Docker と AWS Explorer および AWS CLI を使用して、以下のことが可能です。

- イメージをリポジトリからプルします。
- リポジトリに保存されているイメージを削除します。
- リポジトリを削除します。

デフォルトレジストリで Docker を認証する

Amazon ECR インスタンスと Docker インスタンス間でデータを交換するには、認証が必要です。レジストリで Docker を認証するには

1. AWS CLI のインスタンスに接続されているコマンドラインオペレーティングシステムを開きます。

2. get-login-password を使用して、プライベート ECR レジストリを認証するメソッドです。

```
aws ecr get-login-password --region region | docker login --username AWS --
password-stdin AWS_account_id.dkr.ecr.region.amazonaws.com
```

> **Important**
>
> 上記のコマンドでは、AWS アカウント固有の情報に **region** および **AWS_account_id** の両方を更新する必要があります。

リポジトリにプッシュするイメージにタグを付けます。

AWS のインスタンスで Docker を認証したら、イメージをリポジトリにプッシュします。

1. Docker イメージコマンドを使用して、ローカルに保存したイメージを表示し、タグ付けするイメージを特定します。

```
docker images
```

Example 出力例:

```
REPOSITORY          TAG                 IMAGE ID            CREATED
 SIZE
hello-world         latest              e9ffedc8c286        4 minutes ago
 241MB
```

2. Docker コマンドを使用してイメージをビルドします。

```
docker tag hello-world:latest AWS_account_id.dkr.ecr.region.amazonaws.com/hello-
world:latest
```

3. リポジトリに Docker タグコマンドでタグ付けされたイメージをプッシュします。

```
docker push AWS_account_id.dkr.ecr.region.amazonaws.com/hello-world:latest
```

Example 出力例:

```
The push refers to a repository [AWS_account_id.dkr.ecr.region.amazonaws.com/hello-
world] (len: 1)
e9ae3c220b23: Pushed
a6785352b25c: Pushed
0998bf8fb9e9: Pushed
0a85502c06c9: Pushed
latest: digest:
 sha256:215d7e4121b30157d8839e81c4e0912606fca105775bb0636b95aed25f52c89b size: 6774
```

タグ付きイメージがリポジトリに正常にアップロードされると、AWS Explorer メニューが見えます。

Amazon ECR からイメージをプルする

- イメージは、Docker タグコマンドのローカルインスタンスにプルできます。

```
docker pull AWS_account_id.dkr.ecr.region.amazonaws.com/hello-world:latest
```

Example 出力例:

```
The push refers to a repository [AWS_account_id.dkr.ecr.region.amazonaws.com/hello-
world] (len: 1)
e9ae3c220b23: Pushed
a6785352b25c: Pushed
0998bf8fb9e9: Pushed
0a85502c06c9: Pushed
latest: digest:
 sha256:215d7e4121b30157d8839e81c4e0912606fca105775bb0636b95aed25f52c89b size: 6774
```

Amazon ECR リポジトリからイメージを削除する

VS Code からイメージを削除する方法は 2 つあります。1 つ目の方法は、AWS Explorer を使用することです。

1. AWS Explorer から、ECR メニューを拡張します。
2. イメージを削除するリポジトリを展開します。
3. コンテキストメニュー (右クリック) を開いて、削除するイメージに関連付けられているイメージタグを選択します。
4. ・ ウmタグの削除... オプションを選択して、そのタグに関連付けられているすべての保存されたイメージを削除します。

AWS CLIを使用して、イメージを削除します。

- AWS ecr batch-delete-image コマンドを使用して、リポジトリからイメージを削除することもできます。

```
AWS ecr batch-delete-image \
      --repository-name hello-world \
      --image-ids imageTag=latest
```

Example 出力例:

```
{
    "failures": [],
    "imageIds": [
        {
            "imageTag": "latest",
            "imageDigest":
 "sha256:215d7e4121b30157d8839e81c4e0912606fca105775bb0636b95aed25f52c89b"
        }
    ]
}
```

Amazon ECR インスタンスからリポジトリを削除する

VS Code からリポジトリを削除する方法は 2 つあります。1 つ目の方法は、AWS Explorer を使用することです。

1. AWS Explorer から、ECR メニューを拡張します。

2. コンテキスト (右クリック) メニューを開き、削除するリポジトリを選択します。
3. リポジトリの削除... オプションを選択して、レポジトリを選択します。

AWS CLI から Amazon ECR リポジトリを削除する

- リポジトリは、AWS ecr delete-repository コマンドで削除できます。

> **Note**
>
> デフォルトでは、イメージを含むリポジトリを削除することはできません。ただし、--force フラグはこれを許可します。

```
    AWS ecr delete-repository \
  --repository-name hello-world \
  --force
```

Example 出力例:

```
{
    "failures": [],
    "imageIds": [
        {
            "imageTag": "latest",
            "imageDigest":
 "sha256:215d7e4121b30157d8839e81c4e0912606fca105775bb0636b95aed25f52c89b"
        }
    ]
}
```

Amazon Elastic Container Service を使用する

AWS Toolkit for Visual Studio Code は、Amazon Elastic Container Service (Amazon ECS) にいくつかのサポートを提供しています。Toolkit for VS Code は、タスク定義の作成など Amazon ECS 関連の特定の作業に役立ちます。

トピック

- Amazon ECS タスク定義ファイルでの IntelliSense の使用
- の Amazon Elastic Container Service Exec AWS Toolkit for Visual Studio Code

Amazon ECS タスク定義ファイルでの IntelliSense の使用

Amazon Elastic Container Service (Amazon ECS) を使用するときに実行することの 1 つは、「Amazon Elastic Container Service デベロッパーガイド」の「タスク定義の作成」に記載されているように、タスクの定義を作成することです。AWS Toolkit for Visual Studio Code をインストールすると、Amazon ECS タスク定義ファイル用に IntelliSense 機能が追加されます。

前提条件

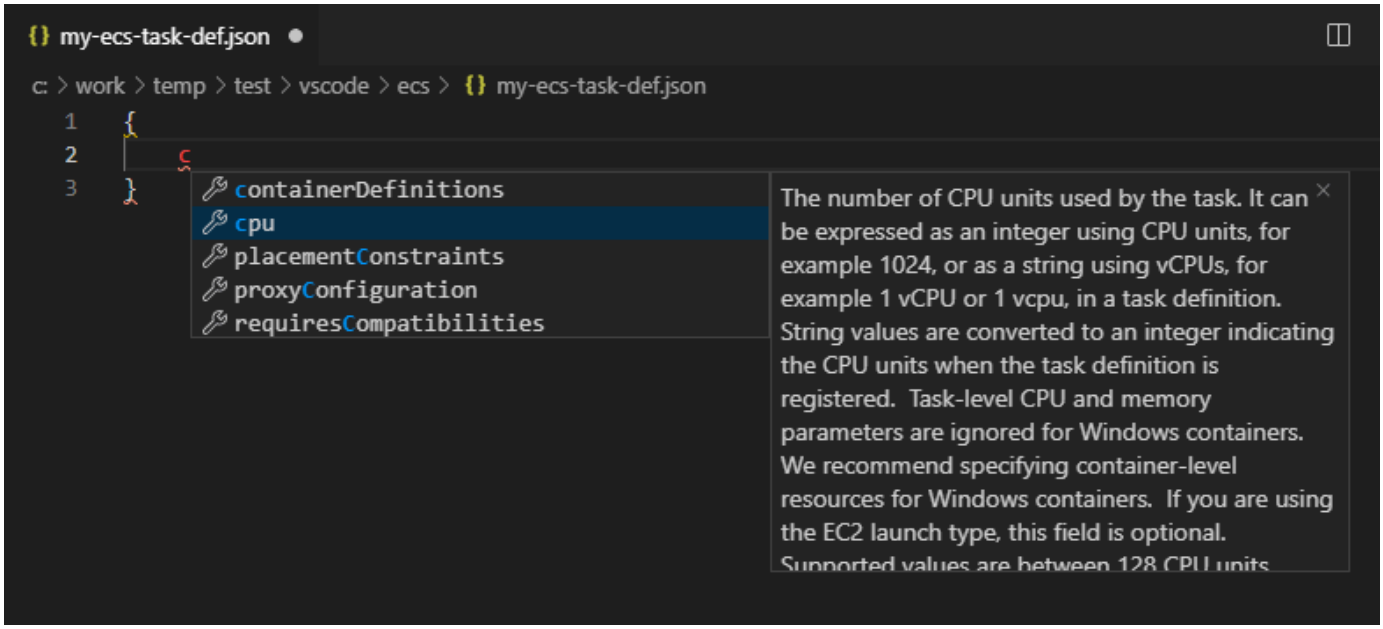
- システムが、「Toolkit for VS Code をインストールする」で指定されている前提条件を満たしていることを確認してください。

Amazon ECS タスク定義ファイルで IntelliSense を使用する

以下の例では、Amazon ECS タスク定義ファイルで IntelliSense を活用する方法を示しています。

1. Amazon ECS タスク定義の JSON ファイルを作成します。ファイルの名前には、末尾に `ecs-task-def.json` が必要ですが、先頭に追加の文字を含めることができます。

   この例では、`my-ecs-task-def.json` という名前のファイルを作成します。
2. VS Code エディタでファイルを開き、最初の中かっこを入力します。
3. 定義に `cpu` を追加する場合と同様に、文字「c」を入力します。表示される IntelliSense ダイアログを確認します。これは以下のようになっています。



の Amazon Elastic Container Service Exec AWS Toolkit for Visual Studio Code

Amazon ECS Exec 機能を使用して AWS Toolkit for Visual Studio Code、 を使用して Amazon Elastic Container Service (Amazon ECS) コンテナで単一のコマンドを発行できます。

> **Important**
>
> Amazon ECS Exec を有効または無効にすると、 AWS アカウントのリソースの状態が変更されます。これには、サービスの停止と再起動が含まれます。さらに、Amazon ECS Exec が有効になっている間にリソースの状態を変更すると、予期しない結果が生じる可能性があります。Amazon ECS の詳細については、デベロッパーガイドの「Amazon ECS Exec を使用してデバッグする」を参照してください。

Amazon Exec の前提条件

Amazon ECS の機能を使用する前に、いくつかの前提条件を満たす必要があります。

Amazon ECS の要件

タスクが Amazon EC2 でホストされているか でホストされているかに応じて AWS Fargate (Fargate)、Amazon ECS Exec のバージョン要件は異なります。

- Amazon EC2 を使用している場合は、2021 年 1 月 20 日以降にリリースされた Amazon ECS 最適化 AMI を、エージェントバージョン 1.50.2 以上で使用する必要があります。補足情報はデベロッパーガイド「Amazon ECS に最適化された AMI」に記載されています。
- を使用している場合は AWS Fargate、プラットフォームバージョン 1.4.0 以降を使用する必要があります。Fargate の要件に関する補足情報は、デベロッパーガイド「AWS Fargate プラットフォームバージョン」に記載されています。

AWS アカウント設定と IAM アクセス許可

Amazon ECS Exec 機能を使用するには、 AWS アカウントに関連付けられた既存の Amazon ECS クラスターが必要です。Amazon ECS Exec は Systems Manager を使用してクラスター内のコンテナとの接続を確立します。SSM サービスと通信するには、特定のタスクの IAM ロールのアクセス許可が必要です。

Amazon ECS Exec に固有の IAM ロールとポリシーの情報については、「ECS Exec に必要な IAM アクセス許可 デベロッパーガイド」に記載されています。

Amazon ECS Exec の操作

Toolkit for VS Code の AWS Explorer から直接 Amazon ECS Exec を有効または無効にできます。Amazon ECS Exec を有効にすると、Amazon ECS メニューからコンテナを選択し、それらに対してコマンドを実行できます。

Amazon ECS Exec の有効化

1. AWS Explorer から、Amazon ECS メニューを見つけて展開します。
2. 変更するサービスを含むクラスターを拡張します。
3. サービスのコンテキストメニュー (右クリック) を開き、[Enable Command Execution] (コマンドの実行を有効にする) を選択します

> **Important**
>
> これによりサービスの新規デプロイが開始されます。完了まで数分かかることがあります。詳細については、このセクションの冒頭にある注意事項を参照してください。

Amazon ECS Exec の無効化

1. AWS Explorer から、Amazon ECS メニューを見つけて展開します。
2. 必要なサービスを収容するクラスターを展開します。
3. サービスのコンテキストメニュー (右クリック) を開き、[Disable Commance Execution] (コマンド実行を無効にする) を選択します

> **Important**
>
> これによりサービスの新規デプロイが開始されます。完了まで数分かかることがあります。詳細については、このセクションの冒頭にある注意事項を参照してください。

コンテナに対するコマンドの実行

AWS Explorer を使用してコンテナに対してコマンドを実行するには、Amazon ECS Exec を有効にする必要があります。有効になっていない場合は、このセクションの「ECS Exec を有効にする」の手順を参照してください。

1. AWS Explorer から、Amazon ECS メニューを見つけて展開します。
2. 必要なサービスを収容するクラスターを展開します。
3. サービスを展開して、関連するコンテナを一覧表示します。
4. コンテナのコンテキストメニュー (右クリック) を開き、[Run Command in Container] (コンテナでコマンドを実行) を選択します
5. 実行中のタスクのリストが表示されたプロンプトが開くので、目的のタスク ARN を選択します。

   > **Note**
   >
   > そのサービスに対して実行中のタスクが 1 つだけの場合、そのタスクは自動的に選択され、このステップはスキップされます。

6. プロンプトが表示されたら、実行するコマンドを入力し、Enter キーを押して処理します。

Amazon EventBridge スキーマの使用

AWS Toolkit for Visual Studio Code (VS Code) は Amazon EventBridge にサポートを提供します。Toolkit for VS Code を使用すると、EventBridge の特定の側面 (スキーマなど) を操作できます。

トピック

- Amazon EventBridge Schemas の使用

Amazon EventBridge Schemas の使用

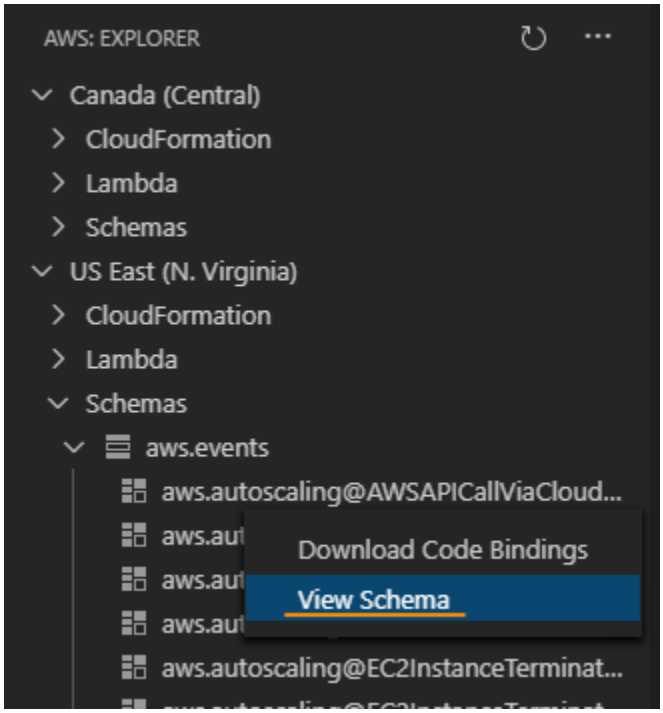
AWS Toolkit for Visual Studio Code (VS Code) を使用して、Amazon EventBridge スキーマ に対してさまざまなオペレーションを実行できます。

前提条件

- システムが、「Toolkit for VS Code をインストールする」で指定されている前提条件を満たしていることを確認してください。
- 使用する EventBridge スキーマは、 AWS アカウントで利用できる必要があります。そうでない場合は、スキーマを作成またはアップロードします。「Amazon ユーザーガイド」の「Amazon EventBridge Schemas」を参照してください。 EventBridge

使用可能なスキーマの表示

1. AWS Explorer で、[スキーマ] を展開します。
2. 表示するスキーマを含むレジストリの名前を展開します。例えば、 AWS が提供するスキーマの多くは aws.events レジストリにあります。
3. エディタでスキーマを表示するには、スキーマのコンテキストメニューを開き、[スキーマの表示] を選択します。



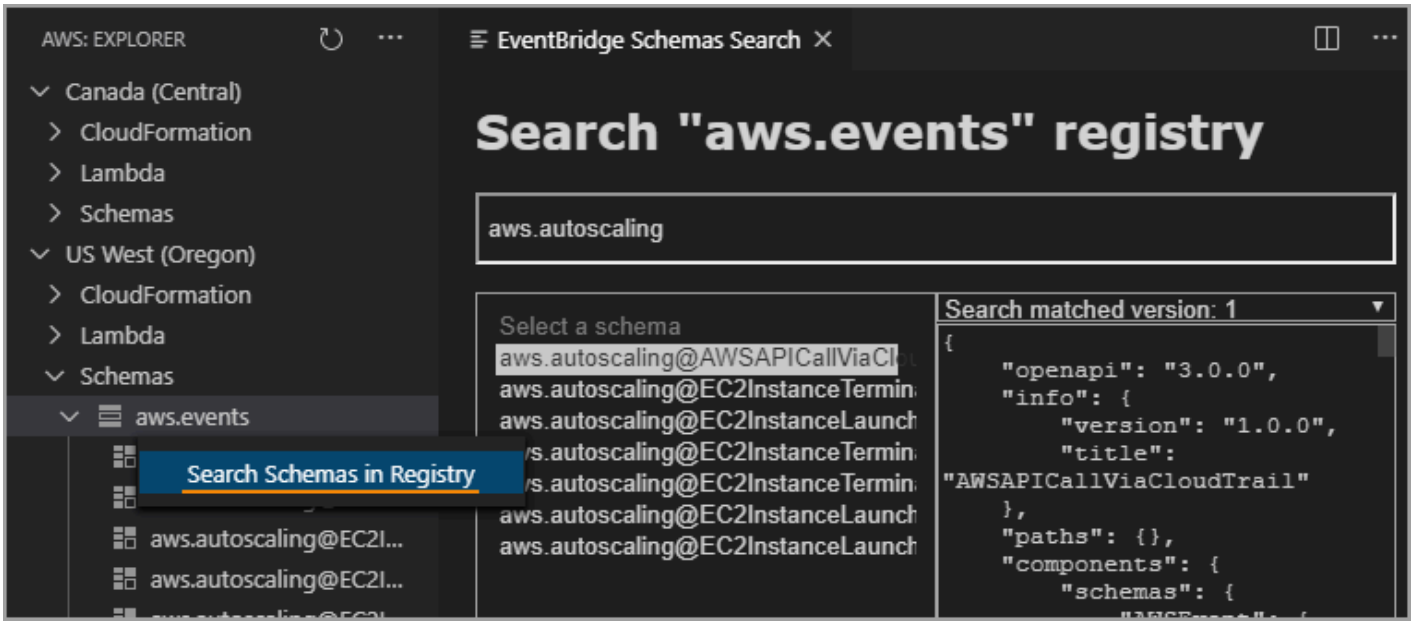
使用可能なスキーマの検索

AWS Explorer で、次のいずれかの操作を行います。

- 検索するスキーマのタイトルの入力を開始します。AWS Explorer で、一致を含むスキーマタイトルがハイライト表示されます (ハイライト表示されたタイトルを表示するには、レジストリを展開する必要があります)。

- [スキーマ] のコンテキストメニューを開き、[スキーマの検索] を選択します。または、[スキーマ] を展開し、検索するスキーマを含むレジストリのコンテキストメニューを開いて、[Search Schemas in Registry (レジストリのスキーマの検索)] を選択します。EventBridge スキーマ検索ダイアログボックスで、検索するスキーマのタイトルの入力を開始します。一致部分を含むスキーマタイトルがダイアログボックスに表示されます。

  ダイアログボックスにスキーマを表示するには、スキーマのタイトルを選択します。



使用可能なスキーマのコードの生成

1. AWS Explorer で、[スキーマ] を展開します。
2. コードを生成するスキーマを含むレジストリの名前を展開します。
3. スキーマのタイトルを右クリックし、[Download code bindings (コードバインドのダウンロード)] を選択します。
4. 表示されたウィザードのページで、以下を選択します。
   - スキーマのバージョン
   - コードのバインド言語
   - 生成されたコードを保存するローカル開発マシン上のワークスペースフォルダ

AWS IAM Access Analyzer

の AWS IAM Access Analyzer を使用して、テンプレート、Terraform プラン、および JSON ポリシードキュメントで作成された IAM ポリシーに対して Identity and Access Management (IAM) Access Analyzer ポリシーチェックを実行できます AWS Toolkit for Visual Studio Code。 AWS CloudFormation

IAM Access Analyzer ポリシーチェックには、ポリシーの検証とカスタムポリシーチェックが含まれます。ポリシーの検証は、「 AWS Identity and Access Managementユーザーガイド」

の「IAM JSON ポリシー言語の文法」および「IAM トピックのセキュリティのベストプラクティス」で説明されている標準に従って IAM ポリシーを検証するのに役立ちます。 AWS https://docs.aws.amazon.com/IAM/latest/UserGuide/best-practices.htmlポリシー検証の結果には、セキュリティ警告、エラー、一般的な警告、ポリシーの提案が含まれます。

セキュリティ標準に基づいて、新しいアクセスのカスタムポリシーチェックを実行することもできます。料金は、新しいアクセスに対する各カスタムポリシーチェックに関連付けられます。料金の詳細については、AWS 「IAM Access Analyzer の料金」サイトを参照してください。IAM Access Analyzer ポリシーチェックの詳細については、「 AWS Identity and Access Managementユーザーガイド」の「ポリシーの検証の確認」トピックを参照してください。

以下のトピックでは、 で IAM Access Analyzer ポリシーチェックを使用する方法について説明します AWS Toolkit for Visual Studio Code。

トピック

- AWS IAM Access Analyzer の使用

AWS IAM Access Analyzer の使用

以下のセクションでは、 で IAM ポリシーの検証とカスタムポリシーチェックを実行する方法について説明します AWS Toolkit for Visual Studio Code。詳細については、「 AWS Identity and Access Management ユーザーガイド」の「IAM Access Analyzer ポリシーの検証」と「IAM Access Analyzer カスタムポリシーチェック」を参照してください。

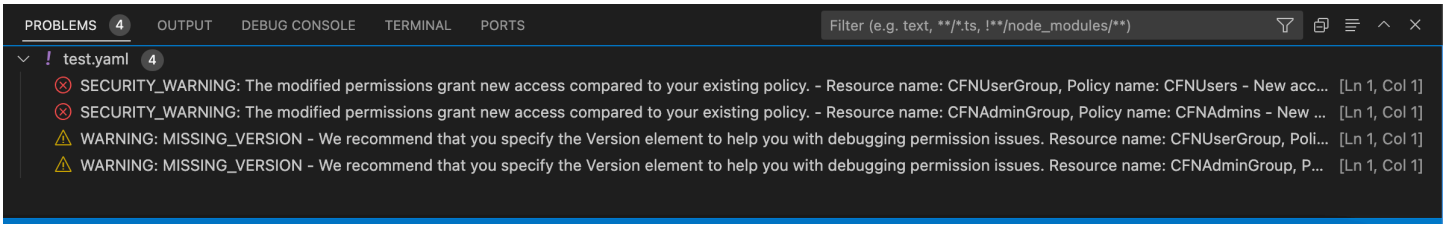
前提条件

Toolkit から IAM Access Analyzer ポリシーチェックを使用する前に、次の前提条件を満たす必要があります。

- Python バージョン 3.6 以降をインストールします。
- Python CLI ツールで必要で、IAM ポリシー AWS CloudFormationチェックウィンドウで指定されている の IAM ポリシー検証ツールまたは Terraform の IAM ポリシー検証ツールをインストールします。
- AWS ロール認証情報を設定します。

IAM Access Analyzerのポリシーチェック

を使用して、 AWS CloudFormation テンプレート、Terraform プラン、JSON ポリシードキュメントのポリシーチェックを実行できます AWS Toolkit for Visual Studio Code。チェック結果は VS Code Problems パネル に表示されます。次の図は、VS Code Problems パネル を示しています。



IAM Access Analyzer には、次の 3 種類のチェックがあります。

- ポリシーの検証
- CheckAccessNotGranted
- CheckNoNewAccess

以下のセクションでは、各タイプのチェックを実行する方法について説明します。

> **Note**
>
> 任意のタイプのチェックを実行する前に、 AWS ロール認証情報を設定します。サポートされているファイルには、 AWS CloudFormation テンプレート、Terraform プラン、JSON ポリシードキュメントのドキュメントタイプが含まれます。

検証ポリシーの実行

ポリシー検証とも呼ばれるポリシー検証チェックは、IAM ポリシーの文法と AWS ベストプラクティスに照らしてポリシーを検証します。詳細については、「 AWS Identity and Access Managementユーザーガイド」の「IAM JSON ポリシー言語の文法」と AWS 「IAM トピックのセキュリティのベストプラクティス」を参照してください。

1. VS Code から、VS Code エディタで AWS IAM ポリシーを含むサポートされているファイルを開きます。
2. IAM Access Analyzer のポリシーチェックを開くには、 を押して VS Code コマンドパレットを開き**CRTL+Shift+P**、 を検索し**IAM Policy Checks**、 をクリックして VS Code エディタの IAM ポリシーチェックペインを開きます。

3. IAM ポリシーチェックペインで、ドロップダウンメニューからドキュメントタイプを選択します。
4. ポリシーの検証セクションで、ポリシー検証の実行ボタンを選択して、ポリシーの検証チェックを実行します。
5. VS Code の問題パネルから、ポリシーチェックの結果を確認します。
6. ポリシーを更新してこの手順を繰り返し、ポリシーチェックの結果にセキュリティ警告またはエラーが表示されなくなるまで、ポリシー検証チェックを再実行します。

実行中 CheckAccessNotGranted

CheckAccessNotGranted は、特定の IAM アクションがポリシーで許可されていないことを確認するカスタムポリシーチェックです。

> **Note**
>
> 料金は、各カスタムポリシーチェックに関連付けられます。カスタムポリシーチェックの料金の詳細については、AWS IAM Access Analyzer 料金ガイドを参照してください。

1. VS Code から、VS Code エディタで AWS IAM ポリシーを含むサポートされているファイルを開きます。
2. IAM Access Analyzer のポリシーチェックを開くには、 を押して VS Code コマンドパレットを開き**CRTL+Shift+P**、 を検索し**IAM Policy Checks**、 をクリックして VS Code エディタの IAM ポリシーチェックペインを開きます。
3. IAM ポリシーチェックペインで、ドロップダウンメニューからドキュメントタイプを選択します。
4. カスタムポリシーチェックセクションで、 を選択しますCheckAccessNotGranted。
5. テキスト入力フィールドに、アクションのカンマ区切りリストを入力します。または、アクションのリストを指すファイルパスを指定することもできます。

> **Note**
>
> ファイルパスは通常、管理者またはセキュリティチームによって提供され、システムファイルパスまたは Amazon S3 バケット URI にすることができます。Amazon S3 バ

ケット URI を使用するには、現在のロールが Amazon S3 バケットにアクセスできる必要があります。

6. カスタムポリシーの実行チェックボタンを選択します。
7. VS Code の問題パネルから、ポリシーチェックの結果を確認します。カスタムポリシーチェックは、 PASSまたは FAILの結果を返します。
8. ポリシーを更新してこの手順を繰り返し、 が返されるまで CheckAccessNotGranted チェックを再実行しますPASS。

実行中 CheckNoNewAccess

CheckNoNewAccess は、ポリシーが参照ポリシーと比較して新しいアクセスを許可するかどうかを確認するカスタムポリシーチェックです。

> **Note**
>
> 料金は、各カスタムポリシーチェックに関連付けられます。カスタムポリシーチェックの料金の詳細については、AWS IAM Access Analyzer 料金ガイドを参照してください。

1. VS Code から、VS Code エディタで AWS IAM ポリシーを含むサポートされているファイルを開きます。
2. IAM Access Analyzer のポリシーチェックを開くには、 を押して VS Code コマンドパレットを開き**CRTL+Shift+P**、 を検索し**IAM Policy Checks**、 をクリックして VS Code エディタの IAM ポリシーチェックペインを開きます。
3. IAM ポリシーチェックペインで、ドロップダウンメニューからドキュメントタイプを選択します。
4. カスタムポリシーチェックセクションで、 を選択しますCheckNoNewAccess。
5. 参照 JSON ポリシードキュメントを入力します。または、JSON ポリシードキュメントを参照するファイルパスを指定することもできます。

> **Note**
>
> ファイルパスは通常、管理者またはセキュリティチームによって提供され、システムファイルパスまたは Amazon S3 バケット URI にすることができます。Amazon S3 バ

ケット URI を使用するには、現在のロールが Amazon S3 バケットにアクセスできる必要があります。

6. リファレンスドキュメントのタイプに一致するリファレンスポリシータイプを選択します。
7. カスタムポリシーの実行チェックボタンを選択します。
8. VS Code の問題パネルから、ポリシーチェックの結果を確認します。カスタムポリシーチェックは、 PASSまたは FAILの結果を返します。
9. ポリシーを更新してこの手順を繰り返し、 が返されるまで CheckNoNewAccess チェックを再実行しますPASS。

AWS Toolkit for Visual Studio Code で AWS IoT を使用する

AWS IoT の AWS Toolkit for Visual Studio Code は AWS IoT サービスで操作を許可し、VS Code でのワークフローの中断を最小限に抑えます。このユーザガイドは、AWS IoT のサービス機能を使用して開始に役立つことを目的としており、AWS Toolkit for Visual Studio Code で使用できます。AWS IoT サービスに関する追加情報については、「デベロッパーガイド」の「AWS IoT とは?」を参照してください。

AWS IoT の前提条件

Toolkit for VS Code から AWS IoT の使用を開始するには、AWS アカウントと VS Code は、以下のガイドの要件を満たしています。

- AWS IoT サービスに固有の AWS アカウント要件および AWS ユーザー権限については、「デベロッパーガイド」の「Core AWS IoT で使用開始」を参照してください。
- Toolkit for VS Code 固有の要件については、「Toolkit for VS Code を設定する ユーザーガイド」を参照してください。

AWS IoT モノ

AWS IoT はデバイスと AWS ローカルサービスとリソースに接続する。モノ と呼ばれるオブジェクトを使用して、AWS IoT にデバイスを接続できます。"モノ" とは、特定のデバイスまたは論理エンティティを表します。物理的なデバイスやセンサー (電球や壁のスイッチなど) は、モノとして扱うことができます。AWS IoT モノに関する追加情報については、「デベロッパーガイド」の「AWS IoT によるデバイスの管理」を参照してください。

AWS IoT モノの管理

Toolkit for VS Code には、AWS IoT モノ管理をより効率的にする機能がいくつかあります。VS Code コードキットを使用して、AWS IoT モノを管理できる方法があります。

- Create a thing
- Attach a certificate to a thing
- Detach a certificate from a thing
- Delete a thing

モノを作成するには

1. AWS Explorer から、[IoT] サービス見出しを展開し、[モノ] をコンテキストメニュー (右クリック) から選択します。
2. [モノを作成する] を選択し、コンテキストメニューからダイアログボックスを開きます。
3. プロンプトに従って、IoT モノの名前を [モノの名前] フィールドに入力します。
4. 完了すると、特定した名前に続く [モノアイコン] は [モノ] セクションに表示されます。

モノに証明書をアタッチするには

1. AWS Explorer から、[IoT] サービスセクションを展開します。
2. [モノ] サブセクションで、証明書を添付する [モノ] を検索します。
3. [モノ] を右クリックして、コンテキストメニューから [証明書の添付] を選択して、証明書のリストを含む入力セレクタを開きます。
4. リストから [証明書 ID] を選択します。これはモノにアタッチする証明書に対応します。
5. これが完了すると、添付したモノのアイテムとして、証明書に AWS Explorer エクスプローラーからアクセスできます。

モノから証明書をデタッチするには

1. AWS Explorer から、[IoT] サービスセクションを展開します。
2. [モノ] サブセクションで、証明書からデタッチしたい [モノ] を検索します。
3. [モノ] を右クリックして、コンテキストメニューから [証明書のデタッチ] を選択します。

4. これが完了すると、デタッチされた証明書は AWS Explorer の下に表示されなくなりますが、まだ [証明書] サブセクションからアクセス可能です。

モノを削除するには

1. AWS Explorer から、[IoT] サービスセクションを展開します。
2. [モノ] サブセクションで、削除したい [モノ] を検索します。
3. モノを右クリックして、コンテキストメニューの [モノの削除] を選択して削除します。
4. これが完了すると、削除されたモノは [モノ] サブセクションから利用できなくなります。

> **Note**
>
> 注意: 削除できるのは、証明書が添付されていないモノのみです。

AWS IoT 証明書

証明書は、ユーザーの AWS IoT サービスとデバイス間のセキュアな接続を作成するための一般的な方法です。X.509 証明書は、X.509 パブリックキーインフラストラクチャ規格を使用して、パブリックキーと証明書内の ID を関連付けるための、デジタル証明書です。AWS IoT 証明書に関する追加情報については、「デベロッパーガイド」の「認証 (IoT)」を参照してください。

証明書の管理

VS Code ツールキットには、AWS IoT 証明書を、AWS Explorer からさまざまな方法で管理できます。

- Create a certificate
- Change a certificate status
- Attach a policy to a certificate
- Delete a certificate

AWS IoT 証明書を作成するには

X.509 証明書を使用して、AWS IoT のインスタンスに接続できます。

1. AWS Explorer から、[IoT] サービス見出しを展開し、[証明書] をコンテキストメニュー (右クリック) から選択します。
2. [証明書を作成する] を選択し、コンテキストメニューからダイアログボックスを開きます。
3. ローカルファイルシステム内のディレクトリを選択して、RSA key pair と X.509 証明書を保存します。

> **Note**
>
> - デフォルトのファイル名には、証明書 ID がプレフィックスとして含まれます。
> - X.509 証明書だけが AWS IoT サービスを通して AWSアカウント に管理されます。
> - RSA key pair は 1 回だけ発行でき、プロンプトが表示されたら、ファイルシステム内の安全な場所に保存してください。
> - この時点で証明書またはkey pair いずれかがファイルシステムに保存できない場合は、AWS Toolkit は AWS アカウント証明書を削除します。

証明書のステータスを変更するには

個々の証明書のステータスは、AWS Explorer のID の横に表示されま、アクティブ、非アクティブ、または失効に設定できます。

> **Note**
>
> - デバイスを AWS IoT サービスに接続する前に、証明書は アクティブ ステータスが必要です。
> - 非アクティブ 証明書は、以前に非アクティブ化されているか、デフォルトで非アクティブであるかにかかわらず、アクティブ化することができます。
> - 失効した 証明書は復活できません。

1. AWS Explorer から、[IoT] サービスセクションを拡張します。
2. [証明書] サブセクションで、修正する証明書を見つけます。
3. 証明書を右クリックして、その証明書で使用可能なステータス変更オプションを表示するコンテキストメニューを開きます。

- 証明書に 非アクティブ ステータスががある場合、[アクティブ] を選択してステータスを アクティブ に変更します。
- 証明書に アクティブ ステータスがある場合、[非アクティブ] を選択してステータスを非アクティブ に変更します。
- 証明書に アクティブ または 非アクティブ ステータスがある場合は、[取り消す] を選択し、ステータスを失効しましたに変更します。

> **Note**
>
> これらのステータス変更の各アクションは、[モノ] サブセクションに表示されている間にアタッチされている証明書を選択した場合にも使用できます。

IoT ポリシーを証明書にアタッチするには

1. AWS Explorer から、[IoT] サービスセクションを展開します。
2. [証明書] サブセクションで、修正する証明書を見つけます。
3. 証明書ををを右クリックして、コンテキストメニューから [ポリシーのアタッチ] を選択し、使用可能なポリシーのリストを含む入力セレクタを開きます。
4. 証明書にアタッチするには、ポリシーを選択します。
5. これが完了すると、選択したポリシーがサブメニュー項目として証明書に追加されます。

証明書から IoT ポリシーをデタッチするには

1. AWS Explorer から、[IoT] サービスセクションを展開します。
2. [証明書] サブセクションで、修正する証明書を見つけます。
3. 証明書を展開し、デタッチするポリシーを見つけます。
4. ポリシーを右クリックして、コンテキストメニューから [デタッチ] を選択します。
5. これが完了すると、ポリシーは、証明書からアクセスできるアイテムではなくなりますが、まだ [ポリシー] サブセクションからアクセス可能です。

証明書を削除するには

1. AWS Explorer から、[IoT] サービス見出しを拡張します。
2. [証明書] サブセクションで、削除する証明書を見つけます。
3. 証明書を右クリックして、コンテキストメニューから [証明書の削除] を選択します。

> **Note**
>
> モノにアタッチされている証明書や、アクティブなステータスの証明書は削除できません。ポリシーがアタッチされた証明書を削除できます。

AWS IoT ポリシー

AWS IoT コアポリシーは、JSON ドキュメントを使用して定義され、それぞれに 1 つ以上のポリシーステートメントが含まれています。AWS IoT、AWS、およびデバイスが操作する方法をポリシーが定義します。ポリシードキュメントの作成方法の詳細については、「デベロッパーガイド」の「IoT ポリシー」を参照してください。

> **Note**
>
> 名前付きポリシーは、バージョン管理されているため、ロールバックできます。AWS Explorer では、IoT ポリシーはIoT サービスのサブセクションである ポリシー の下に記載されます。ポリシーを展開すると、ポリシーバージョンを表示できます。デフォルトバージョンはアスタリスクで示されます。

ポリシーの管理

Toolkit for VS Code は、AWS IoT サービスポリシーを管理するいくつかの方法を提供します。次の方法で、VS Code の AWS Explorer から直接ポリシーの管理や変更できます。

- Create a policy
- Upload a new policy version
- Edit a policy version
- Change the policy version defualt
- Change the policy version defualt

AWS IoT ポリシーを作成するには

> **Note**
>
> AWS Explorer から新しいポリシーを作成できますが、ポリシーを定義する JSON ドキュメントがファイルシステムにすでに存在している必要があります。

1. AWS Explorer から、[IoT] サービスセクションを展開します。
2. [ポリシー] サブセクションを右クリックして、[ドキュメントからポリシーを作成] を選択し、[ポリシー名] 入力フィールドを開きます。
3. 名前を入力し、プロンプトに従って、ファイルシステムから JSON ドキュメントを選択するように求めるダイアログを開きます。
4. ポリシー定義を含む JSON ファイルを選択すると、ポリシーはこれが完了したら AWS エクスプローラーで利用できます。

新しい AWS IoT ポリシーバージョンをアップロードするには

ポリシーに新しいバージョンを作成するには、JSON ドキュメントをポリシーにアップロードします。

> **Note**
>
> 新しいバージョンを作成するには、AWS Explorer を使用して新しい JSON ドキュメントがファイルシステム上に存在している必要があります。

1. AWS Explorer から、[IoT] サービスセクションを展開します。
2. ポリシーサブセクションを展開すると、AWS IoT ポリシーが表示されます。
3. 更新するポリシーを右クリックして、[ドキュメントから新しいバージョンを作成する] を選択します。
4. ダイアログボックスが開いたら、ポリシー定義の更新を含む JSON ファイルを選択します。
5. 新しいバージョンには、AWS　Explorer の次のポリシーからアクセスできます。

AWS IoT ポリシーバージョンを編集するには

ポリシードキュメントは VS Code を使用して開き、編集できます。ドキュメントの編集が終了したら、そのドキュメントをファイルシステムに保存できます。次に、AWS Explorer から AWS IoT サービスにアップロードすることができます。

1. AWS Explorer から、[IoT] サービスセクションを展開します。
2. ポリシー を拡張し、更新するポリシーを見つけ、ドキュメントからポリシーを作成 を更新してポリシー名 入力フィールドを開きます。
3. 更新するポリシーを展開し、編集するポリシーバージョンを右クリックします。
4. コンテキストメニューから [表示] を選択して、VS Code でポリシーバージョンを開きます
5. ポリシードキュメントが開かれたら、必要な変更を加えて保存します。

> **Note**
>
> この時点で、ポリシーに加えた変更は、ローカルファイルシステムにのみ保存されます。バージョンを更新し、AWS Explorer で追跡するには、Upload a new policy version に記載されている手順を繰り返します。

新しいポリシーバージョンのデフォルトを選択するには

1. AWS Explorer から、[IoT] サービスセクションを展開します。
2. ポリシー を拡張し、更新するポリシーを見つけます。
3. 更新するポリシーを展開し、設定するポリシーバージョン右クリックして、[デフォルトとして設定] を選択します。
4. これが完了すると、選択した新しいデフォルトバージョンの横に星が表示されます。

ポリシーを削除するには

> **Note**
>
> ポリシーまたはポリシーバージョンを削除する前に、満たす必要がある条件があります。
>
> - 証明書にアタッチされているポリシーは削除できません。

- デフォルト以外のバージョンがある場合は、ポリシーを削除できません。
- 新しいデフォルトバージョンを選択するか、ポリシー全体が削除されない限り、ポリシーのデフォルトバージョンを削除することはできません。
- ポリシー全体を削除する前に、そのポリシーのデフォルト以外のバージョンをすべて削除する必要があります。

1. AWS Explorer から、[IoT] サービスセクションを展開します。
2. ポリシー を拡張し、更新するポリシーを見つけます。
3. 更新するポリシーを展開し、削除するポリシーバージョンを右クリックして、[削除] を選択します。
4. バージョンが削除されると、Explorer からは表示されなくなります。
5. ポリシーに残っている唯一のバージョンがデフォルトの場合、親ポリシーを右クリックして、[削除] を選択して削除します。

AWS Lambda 関数を操作する

AWS Toolkit for Visual Studio Code では、AWS Lambda 関数がサポートされています。ツールキットを使うと、サーバーレスアプリケーション のパートとなる Lambda 関数のコードを作成できます。さらに、Lambda 関数をローカルまたは AWS で呼び出すことができます。

Lambda は、カスタムコードによって生成されるイベント、またはさまざまな AWS サービス (Amazon Simple Storage Service (Amazon S3)、Amazon DynamoDB、Amazon Kinesis、Amazon Simple Notification Service (Amazon SNS)、Amazon Cognito など) から生成されるイベントに応じてコードを実行する、フルマネージドコンピューティングサービスです。

トピック

- リモート Lambda 関数の操作

リモート Lambda 関数の操作

このトピックで後述するように、Toolkit for VS Code, を使用して AWS Lambda 関数をさまざまな方法で操作できます。

Lambda の詳細については、「AWS Lambda デベロッパーガイドhttps://docs.aws.amazon.com/lambda/latest/dg/」を参照してください。

> **Note**
>
> AWS Management Console を使用するか、別の方法ですでに Lambda 関数を作成している場合は、ツールキットから呼び出すことができます。AWS Lambda にデプロイできる新しい関数を作成するには (VS Codeを使用)、まず、サーバーレスアプリケーションを作成する必要があります。

トピック

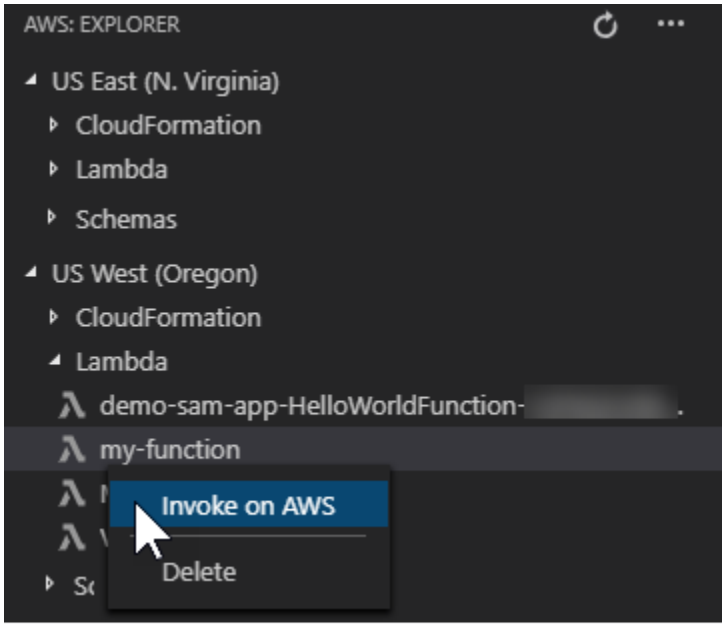
前提条件

- システムが、「Toolkit for VS Code をインストールする」で指定されている前提条件を満たしていることを確認してください。
- 「認証とアクセス」で設定した認証情報に、AWS Lambda サービスに対する適切な読み取り/書き込みアクセス許可が含まれていることを確認します。[AWS Explorer] では、Lambdaの下に「Lambda リソースのロードエラー」のようなメッセージが表示される場合は、こういった認証情報に添付されている許可をチェックしてください。アクセス許可に変更を加えると、VS Code の AWSExplorer に影響するまで数分かかります。

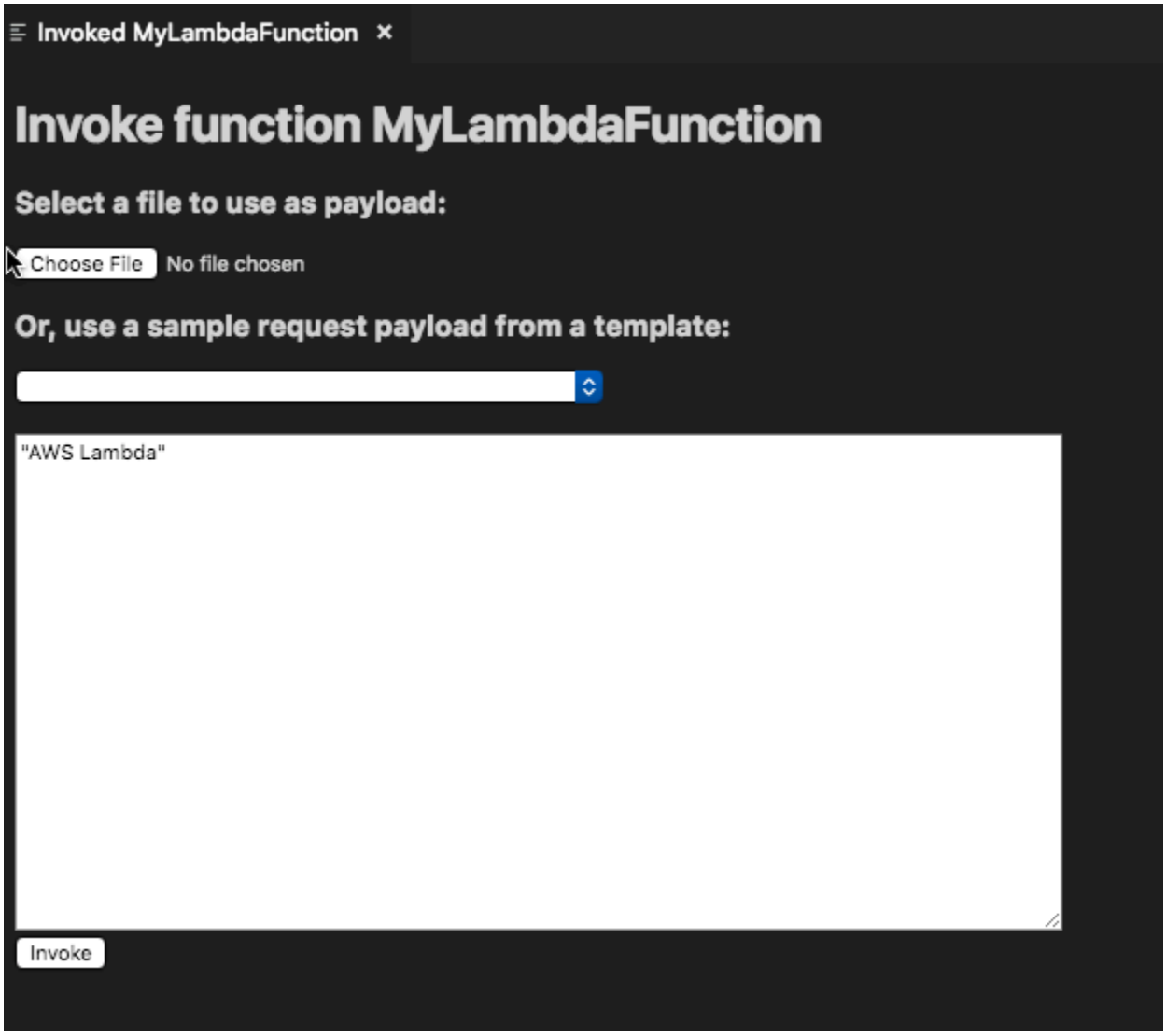
Lambda 関数を呼び出す

AWS で Toolkit for VS Codeを使って Lambda 関数を呼び出すことができます。

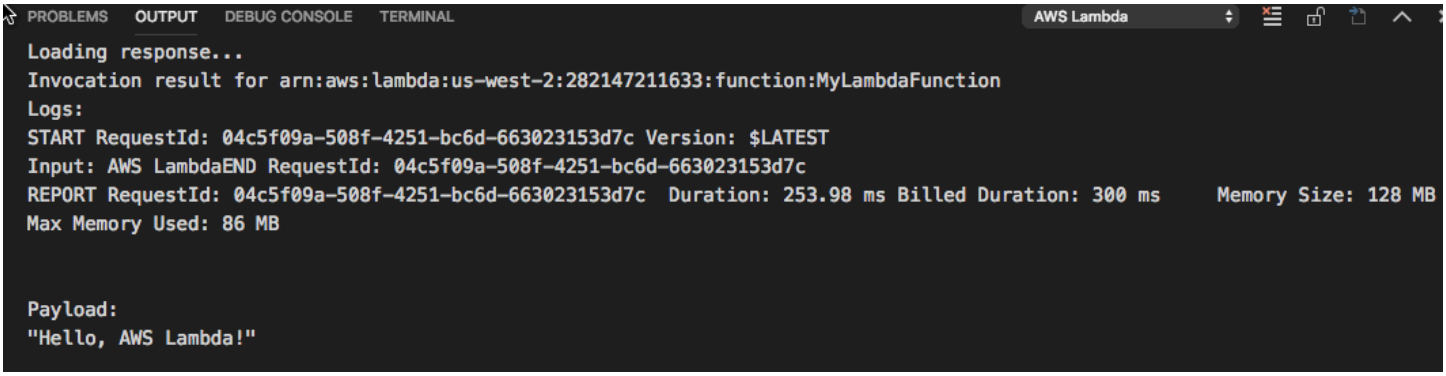
1. AWS Explorer で、呼び出したい Lambda 関数の名前を選択し、コンテキストメニューを開きます。



2. [AWS での呼び出し] を選択します。
3. 開いた呼び出しウィンドウで、Lambda 関数に必要な入力を入力します。例えば、Lambda 関数では、テキストボックスに示すように、入力として文字列を要求する場合があります。



Lambda 関数の出力が表示されます。これは、他のプロジェクトで VS Code を使用する場合と同様です。

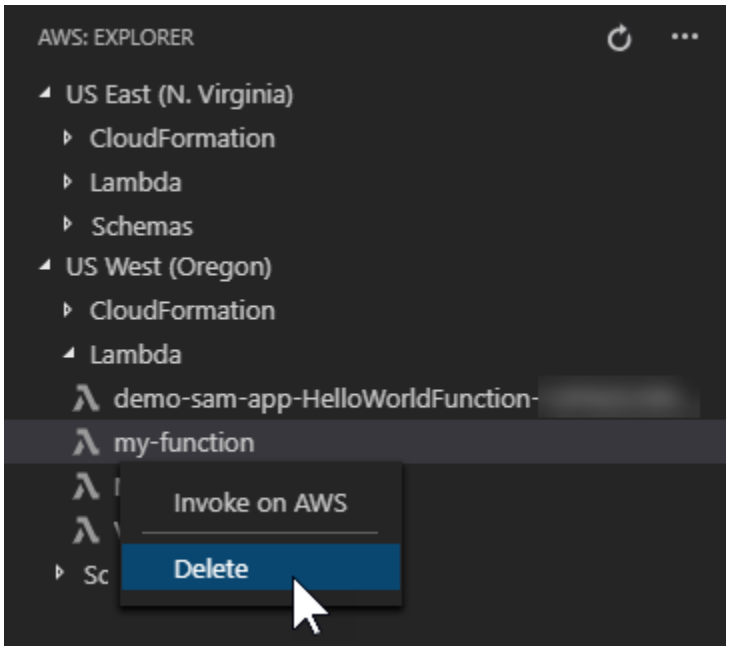


Lambda 関数を削除する

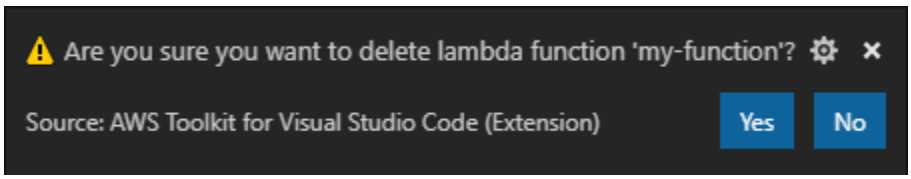
同じコンテキストメニューを使用して Lambda 関数を削除することもできます。

> **Warning**
>
> AWS CloudFormation と関連づけられている Lambda 関数 (このガイドで前述したサーバーレスアプリケーションの作成時に作成した Lambda 関数など) は、この手順を使用して削除しないでください。これらの関数は、AWS CloudFormation スタックを通じて削除する必要があります。

1. AWS Explorer で、削除したい Lambda 関数の名前を選択し、コンテキストメニューを開きます。



2. [Delete] (削除) をクリックします。
3. 表示されるメッセージで、[はい] を選択して削除を確認します。



関数が削除されると、[AWS Explorer] に表示されなくなります。

Lambda 関数をインポートする

リモート Lambda 関数から VS Code ワークスペースにコードをインポートして、編集とデバッグを行うことができます。

> **Note**
>
> ツールキットは、サポートされている Node.js および Python ランタイムを使用した Lambda 関数のインポートのみをサポートします。

1. AWS Explorer で、インポートしたい Lambda 関数の名前を選択し、コンテキストメニューを開きます。
2. [インポート...] を選択します。
3. Lambda コードのインポート先のフォルダーを選択します。現在のワークスペースの外にあるフォルダは、現在のワークスペースに追加されます。
4. ダウンロード後、ツールキットはコードをワークスペースに追加し、Lambda ハンドラーコードを含むファイルを開きます。ツールキットは、起動設定を作成し、これは [VS Code Run] パネルに表示されるため、AWS Serverless Application Model を使用して Lambda 関数をローカルで実行およびデバッグできます。AWS SAM の使用の詳細については、「the section called “テンプレートからのサーバーレスアプリケーションの実行とデバッグ (ローカル)”」を参照してください。

Lambda 関数のアップデート

既存の Lambda 関数をローカルコードで更新できます。この方法でコードを更新しても、デプロイのため AWS SAM CLI を使用することにはならず、AWS CloudFormation スタックは作成されません。この機能は、Lambda でサポートされている任意のランタイムを使って Lambda 関数をアップロードできます。

> **Warning**
>
> ツールキットは、コードが動作するかどうかを確認できません。本番 Lambda 関数を更新する前に、コードが動作することを確認してください。

1. AWS Explorer で、インポートしたい Lambda 関数の名前を選択し、コンテキストメニューを開きます。
2. Lambda のアップロード...を選択
3. Lambda 関数をアップロードするため、3 つのオプションから選択します。オプションには以下が含まれます。

   事前に作成された .zip アーカイブをアップロード

   - Quick Pick メニューから [ZIP アーカイブ] を選択します。
   - ファイルシステムから.zip ファイルを選択し、モーダルダイアログでアップロードを確認します。こうして、.zip ファイルそのままアップロードされ、デプロイ後にすぐに Lambda が更新されます。

   ディレクトリをそのままアップロード

   - [クイックピック] メニューから ディレクトリ を選択します。
   - 自分のファイルシステムからディレクトリを選択します。
   - ディレクトリを構築するように求められたら [なし] を選択し、モーダルダイアログでアップロードを確認します。これにより、ディレクトリがそのままアップロードされ、デプロイ後にすぐに Lambda が更新されます。

   ディレクトリの構築とアップロード

   > **Note**
   >
   > これには AWS SAM CLI が必要です。

   - [クイックピック] メニューから [ディレクトリ] を選択します。
   - 自分のファイルシステムからディレクトリを選択します。
   - ディレクトリを構築するように求められたら [はい] を選択して、モーダルダイアログでアップロードを確認します。こうして AWS SAM CLI `sam build` コマンドを使用してディレクトリにコードを構築し、デプロイの後、すぐに Lambda を更新します。

> **Note**
>
> ツールキットは、アップロード前に一致するハンドラを検出できない場合に警告します。Lambda 関数に関連付けられているハンドラを変更する場合は、AWS Management Console または AWS CLI から追加できます。

Toolkit for VS Code for VS Code

Amazon Redshift は、 クラウド内でのフルマネージド型、ペタバイト規模のデータウェアハウスサービスです。Amazon Redshift サービスの詳細については、Amazon Redshift ユーザーガイドの目次を参照してください。

以下のトピックでは、AWS Toolkit for Visual Studio Codeから Amazon Redshift を使用する方法について説明しています。

トピック

- Toolkit for VS Code

Toolkit for VS Code

次のセクションでは、AWS Toolkit for Visual Studio Code から Amazon Redshift の使用を開始する方法について説明します。

Amazon Redshift サービスの詳細については、Amazon Redshift ユーザーガイドのトピックを参照してください。

はじめに

AWS Toolkit for Visual Studio Codeから Amazon Redshift を使用する前に、次の要件を満たす必要があります。

1. Toolkit からAWSアカウントに接続されている。Toolkit から AWS アカウントへの接続に関する補足情報については、本ユーザーガイドの「AWS に接続する」トピックを参照してください。
2. プロビジョニングされたデータ ウェアハウスまたはサーバーレス データウェアハウスが作成されている。

Amazon Redshift サーバーレスまたは Amazon Redshift プロビジョニングクラスターをまだ作成していない場合、以下の手順では、AWSコンソールからサンプルデータセットを使用してデータウェアハウスを作成する方法について説明します。

プロビジョニングされたデータウェアハウスを作成する

Amazon Redshift プロビジョニングクラスターデータウェアハウスの作成に関する詳細については、Amazon Redshift 入門ユーザーガイドの「サンプル Amazon Redshift クラスターの作成」トピックを参照してください。

1. 好みのインターネットブラウザから AWS マネジメントコンソールにサインインし、Amazon Redshift コンソール (https://console.aws.amazon.com/redshift/) を開きます。
2. Amazon Redshift コンソールで、[プロビジョニングされたクラスターダッシュボード]に移動します。
3. プロビジョニング済みクラスターダッシュボードから [クラスターの作成] ボタンを選択し、[クラスターの作成] ペインを開きます。
4. 「クラスター設定」セクションの必須フィールドに入力します。
5. [サンプル データ] セクションで、[サンプル データのロード] ボックスを選択して、**public** スキーマを使用してサンプル データセット **Tickit** をデフォルトのデータベース **Dev** にロードします。
6. 「データベース設定」セクションで、「管理者ユーザー名」フィールドと「管理者ユーザーのパスワード」フィールドに値を入力します。
7. [クラスターの作成] を選択して、プロビジョニングされたデータウェアハウスを作成します。

サーバーレスデータウェアハウスを作成する

Amazon Redshift サーバーレスデータウェアハウスの作成に関する詳細については、Amazon Redshift 入門ユーザーガイドの「Amazon Redshift サーバーレスによるデータウェアハウスの作成」セクションを参照してください。

1. 好みのインターネットブラウザから AWS マネジメントコンソールにサインインし、Amazon Redshift コンソール https://console.aws.amazon.com/redshift/ を開きます。
2. Amazon Redshift コンソールから、[Amazon Redshift サーバーレスを試す] ボタンを選択して [Amazon Redshift サーバーレスを始める] ウィンドウを開きます。
3. [設定] セクションで、[デフォルト設定を使用する] ラジアルを選択します。

4. [Amazon Redshift サーバーレスの使用を開始する] ペインの下部で、[設定を保存] を選択して、デフォルトのワークグループ、名前空間、認証情報、および暗号化設定を使用してサーバーレスデータ ウェアハウスを作成します。

Toolkit からデータウェアハウスに接続する

Toolkit からデータベースに接続するには、次の 3 つの方法があります。

- データベースユーザー名とパスワード
- AWS Secrets Manager
- テンポラリ認証情報

Toolkit から既存のプロビジョニング済みクラスターまたはサーバーレスデータウェアハウスにあるデータベースに接続するには、次の手順を実行します。

> **Important**
>
> このユーザーガイドのトピックの「前提条件」セクションの手順を完了しても、データウェアハウスが Toolkit Explorer に表示されない場合は、エクスプローラーの正しいAWSリージョンから作業していることを確認してください。

データベースのユーザー名とパスワードを使用した方法でデータウェアハウスに接続する

1. ツールキットエクスプローラーから、データウェアハウスが存在する AWS リージョン を展開します。。
2. Redshift を展開し、データウェアハウスを選択して VS Code の [接続タイプの選択] ダイアログを開きます。
3. [接続タイプの選択] ダイアログから [データベースのユーザー名とパスワード] を選択し、各プロンプトに必要な情報を入力します。
4. Toolkit がデータウェアハウスに接続して手順が完了すると、使用可能なデータベース、テーブル、スキーマが Toolkit Explorer に表示されます。

AWS Secrets Managerを使用してデータウェアハウスに接続します。

> **Note**
>
> この手順を完了するには、AWSシークレットマネージャーのデータベースシークレットが必要です。データベースシークレットの設定方法については、「AWSSecrets Manager ユーザーガイド」の「AWS Secrets Managerデータベースシークレットの作成」を参照してください。

1. ツールキット エクスプローラーから、データ ウェアハウスが存在する AWS リージョン を展開します。
2. Redshift を展開し、データウェアハウスを選択して VS Code の [接続タイプを選択] ダイアログを開きます。
3. 「接続タイプを選択」 ダイアログから「Secrets Manager」を選択し、各プロンプトに必要な情報を入力します。
4. Toolkit がデータウェアハウスに接続して手順が完了すると、使用可能なデータベース、テーブル、スキーマが Toolkit エクスプローラーに表示されます。

一時的な認証情報を使用してデータウェアハウスに接続する

1. Toolkit エクスプローラーから、データウェアハウスが存在するAWSリージョンを展開します。
2. Redshift を展開し、データウェアハウスを選択して VS Code の [接続タイプの選択] ダイアログを開きます。
3. [接続タイプの選択] ダイアログから [一時的な認証情報] を選択し、各プロンプトに必要な情報を入力します。
4. Toolkit がデータウェアハウスに接続して手順が完了すると、使用可能なデータベース、テーブル、スキーマが Toolkit エクスプローラーに表示されます。

データウェアハウスへの接続を編集します。

データウェアハウスへの接続を編集して、接続するデータベースを変更できます。

1. Toolkit エクスプローラーから、AWS リージョンデータウェアハウスが存在する場所を展開します。

2. Redshift を展開し、接続しているデータウェアハウスを右クリックして [接続を編集] を選択し、接続するデータベースの名前を指定します。
3. Toolkit がデータウェアハウスに接続して手順が完了すると、使用可能なデータベース、テーブル、スキーマが Toolkit エクスプローラーに表示されます。

データウェアハウスへの接続を削除する

1. Toolkit エクスプローラーから、データウェアハウスが存在するAWS リージョンを展開します。
2. Redshift を展開し、削除する接続のあるデータウェアハウスを右クリックして、「接続を削除」を選択します。これにより、使用可能なデータベース、テーブル、スキーマが Toolkit エクスプローラーから削除されます。
3. データウェアハウスに再接続するには、[クリックして接続] を選択し、各プロンプトに必要な情報を入力します。デフォルトでは、再接続では以前の認証方法を使用してデータウェアハウスに接続します。別の方法を使用するには、認証プロンプトが表示されるまでダイアログの [戻る] 矢印を選択します。

SQL ステートメントの実行

以下の手順では、AWS Toolkit for Visual Studio Codeからデータベースに SQL ステートメントを作成および実行する方法について説明します。

> **Note**
>
> 以下の各手順のステップを完了するには、まずこのユーザーガイドのトピックにある「ツールキットからデータウェアハウスに接続する」セクションを完了する必要があります。

1. Toolkit エクスプローラーから Redshift を展開し、クエリするデータベースを含むデータウェアハウスを展開します。
2. Create-Notebook を選択してノートブックをローカルに保存するファイル名と場所を指定し、OK を選択して VS Code エディタでノートブックを開きます。
3. VS Code エディタから、このノートブックに保存する SQL ステートメントを入力します。
4. 「すべてを実行」ボタンを選択して、入力した SQL ステートメントを実行します。
5. SQL ステートメントの出力は、入力したステートメントの下に表示されます。

ノートブックへの Markdown の追加

1. VS Code エディタのノートブックから [Markdown] ボタンを選択し、ノートブックに Markdown セルを追加します。
2. 表示されたセルにマークダウンを入力します。
3. Markdown セルは Markdown セルの右上隅にある工ディターツールを使用して編集できます。

ノートブックへのコードの追加

1. VS Code Editor のノートブックから [コード] ボタンを選択し、ノートブックにコードセルを追加します。
2. 表示されたセルにコードを入力します。
3. コードセルの右上隅にあるセル工ディターツールから適切なボタンを選択すると、コードセルの上または下でコードを実行できます。

Amazon S3 での使用

Amazon SSimple Storage Service (Amazon S3) は、スケーラブルなデータストレージサービスです。AWS Toolkit for Visual Studio Codeにより、Amazon S3 のオブジェクトとリソースを VS Code から直接管理できます。

DynamoDB サービスに関する詳細については、「Amazon S3 ユーザーガイド」を参照してください。

次の項目では、AWS Toolkit for Visual Studio Codeから Amazon S3 オブジェクトとリソースを使用する方法について説明します。

トピック

- Amazon S3 リソースでの作業
- Amazon S3 オブジェクトの操作

Amazon S3 リソースでの作業

AWS Toolkit for Visual Studio Codeから Amazon S3 を使用して、Amazon S3 バケットやその他のリソースを表示、管理、編集できます。

次のセクションでは、AWS Toolkit for Visual Studio Codeの Amazon S3 リソースを操作する方法について説明します。AWS Toolkit for Visual Studio Codeにある Amazon S3 オブジェクト (フォルダやファイルなど) の操作については、このユーザーガイドの「S3 オブジェクトの操作」トピックを参照してください。

Amazon S3 バケットの作成

1. ツールキット エクスプローラーから、S3 サービスのコンテキスト (右クリック) メニューを開き、[バケットの作成...] を選択します。または、[バケットを作成] アイコンを選択して [バケットを作成] ダイアログボックスを開きます。
2. [バケット名] フィールドに、バケットの有効な名前を入力します。

   Enter を押してバケットを作成し、このダイアログボックスを閉じます。すると、新しいバケットがツールキットの S3 サービスの下に表示されます。

> **Note**
>
> Amazon S3 ではバケットをパブリックにアクセス可能な URL として使用できるので、グローバルに一意なバケット名を選択する必要があります。使用する名前で別のアカウントがすでにバケットを作成している場合は、別の名前を使用する必要があります。
> バケットを作成できない場合は、[出力]タブの [AWS Toolkit ログ]をチェックできます。
> 無効なバケット名を使用しようとすると、`BucketAlreadyExists`エラーが発生します。
> 詳細については、「Amazon Simple Storage Service ユーザーガイド」の「バケットの制約と制限」を参照してください。

Amazon S3 バケットにフォルダを追加

S3 バケットの内容は、オブジェクトをフォルダにグループ化することで整理できます。フォルダ内にフォルダを作成することもできます。

1. Toolkit エクスプローラーから、[S3] サービスを展開して S3 リソースのリストを表示します。
2. [フォルダを作成アイコン]を選択し、[フォルダを作成] ダイアログボックスを開きます。または、バケットまたはフォルダーのコンテキスト (右クリック) メニューを開き、[フォルダーを作成] を選択します。

3. 「フォルダ名」フィールドに値を入力し、Enter キーを押してフォルダを作成し、ダイアログボックスを閉じます。新しいフォルダは、ツールキットメニューの対応する S3 リソースの下に表示されます。

Amazon S3 バケットの削除

S3 バケットを削除すると、それに含まれるフォルダーとオブジェクトも削除されます。そのため、バケットを削除しようとすると、削除を確認するメッセージが表示されます。

1. ツールキットのメインメニューから、[Amazon S3] サービスを展開して S3 リソースのリストを表示します。
2. バケットまたはフォルダーのコンテキスト (右クリック) メニューを開き、[S3 バケットを削除] を選択します。
3. プロンプトが表示されたら、テキストフィールドにバケット名を入力する。Enter を押してバケットを削除し、確認プロンプトを閉じます。

> **Note**
>
> バケットにオブジェクトが含まれている場合は、削除される前に空になります。大量のリソースまたはオブジェクトを一度に削除しようとすると、削除までに少し時間がかかることがあります。削除すると、正常に削除されたことを知らせる通知が届きます。

Amazon S3 オブジェクトの操作

S3 リソースバケットに保存されているファイル、フォルダ、その他のデータは S3 オブジェクトと呼ばれます。

次のセクションでは、AWS Toolkit for Visual Studio Code から Amazon S3 オブジェクトを使用する方法について説明します。AWS Toolkit for Visual Studio Codeからの Amazon S3 リソース (S3 バケットなど) の操作の詳細については、このユーザーガイドの「S3 リソースの使用」トピックを参照してください。

オブジェクトの割りの一覧表示

多数の Amazon S3 オブジェクトとフォルダを操作している場合は、のページ割りを使用してページに表示する項目の数を指定できます。

1. VS Code アクティビティバーに移動し、[拡張機能] を選択します。
2. AWS Toolkit 拡張機能から、設定アイコンを選択し、拡張機能の設定 を選択してください。
3. 設定 ページで、AWS > S3 までスクロールダウンします。ページ 設定ごとの最大項目。
4. 「さらにロード」が表示される前に表示したい S3 アイテムの数にデフォルト値を変更します。

> **Note**
>
> 有効な値には 3 ~ 1000 の数が含まれます。この設定は、一度に表示されるオブジェクトまたはフォルダの数にのみ適用されます。作成したすべてのバケットが一度に表示されます。デフォルトでは、AWS アカウントにつき最大で 100 個のバケットを作成できます。

5. [設定] ページを閉じて、変更を確認します。

JSON 形式のファイルの設定を更新するには、[設定] ページの右上にある [設定を開く (JSON)] を選択します。

Amazon S3 オブジェクトのアップロードとダウンロード

AWS Toolkit for Visual Studio Code から、ローカルに保存されているファイルを Amazon S3 バケットにアップロードしたり、リモートの Amazon S3 オブジェクトをローカル システムにダウンロードしたりできます。

Toolkit を使用して、ファイルをアップロードする

1. Toolkit エクスプローラーから、[Amazon S3] サービスを展開して S3 リソースのリストを表示します。
2. バケットまたはフォルダの横にある ファイルをアップロードアイコンを選択し、ファイルをアップロードダイアログを開きます。または、コンテキスト (右クリック) メニューを開いて、[ファイルをアップロード] を選択することもできます。

> **Note**
>
> オブジェクトの親フォルダまたはリソースにファイルをアップロードするには、任意の S3 オブジェクトのコンテキスト (右クリック) メニューを開き、[親にアップロード]を選択します。

3. システムのファイルマネージャを使用してファイルを選択し、[ファイルをアップロード] を選択してダイアログを閉じてファイルをアップロードします。

コマンドパレットを使用してファイルをアップロードする

ツールキットインターフェイスまたはコマンドパレットを使用して、バケットにファイルをアップロードできます。

1. アップロードするファイルを選択するには、VS Codeで、ファイルのタブを選択します。
2. Ctrl+Shift+P を押して コマンドパレットを表示します。
3. コマンドパレットで、 フレーズ`upload file`を入力して 推奨されたコマンドを表示します。
4. AWS：ファイルをアップロードコマンドを選択し、AWS：ファイルをアップロードダイアログを開きます。
5. プロンプトが表示されたら、アップロードするファイルを選択し、そのファイルをアップロードするバケットを選択します。
6. アップロードを確認してダイアログを閉じ、アップロードプロセスを開始します。アップロードが完了すると、オブジェクトサイズ、最終変更日、パスなどのメタデータとともにオブジェクトがツールキットメニューに表示されます。

Amazon S3 オブジェクトのダウンロード

1. Toolkit Explorer から、S3 サービスを展開します。
2. バケットまたはフォルダから、ダウンロードするオブジェクトのコンテキスト (右クリック) メニューを開きます。次に、[名前をつけてダウンロード] を選択して [名前をつけてダウンロード] ダイアログボックスを開きます。または、オブジェクトの近くにある「名前をつけてダウンロード」アイコンを選択します。
3. システムのファイル マネージャーを使用して、宛先フォルダーを選択し、ファイル名を入力し、[ダウンロード] を選択してダイアログを閉じ、ダウンロードを開始します。

リモートオブジェクトの編集

AWS Toolkit for Visual Studio Codeを使用して、リモート Amazon S3 リソースに保存されている Amazon S3 オブジェクトを編集できます。

1. Toolkit Explorer から、S3 サービスを展開します。
2. 編集するファイルを含む S3 リソースを展開します。

3. ファイルを編集するには、鉛筆アイコン (ファイルの編集) を選択します。
4. 読み取り専用モードで開いているファイルを編集するには、VS Code Editor でファイルを表示し、UI の右上隅にある鉛筆アイコンを選択します。

> **Note**
>
> - VS Code を再起動または終了すると、IDE は S3 リソースから切断されます。接続を切断したときにリモート S3 ファイルが編集されていた場合、編集は停止します。編集を再開するには、VS Code を再起動し、編集タブを再度開く必要があります。
> - [ファイルを編集] ボタンは、UI の右上隅にあります。VS Code Editor で読み取り専用ファイルをアクティブに表示している場合にのみ表示されます。
> - テキスト以外のファイルは読み取り専用モードでは開くことができません。これらは常に編集モードで開きます。
> - 編集専用モードから読み取り専用モードに戻すことはできず、その逆しかできません。

Amazon S3 オブジェクトのパスのコピー

以下の手順では、AWS Toolkit for Visual Studio Codeから Amazon S3 オブジェクトのパスをコピーする方法について説明します。

1. ツールキット エクスプローラーから、S3 サービスを展開します。
2. パスをコピーするオブジェクトを含むリソースバケットを展開します。
3. パスをコピーするオブジェクトのコンテキスト (右クリック) メニューを開き、[パスをコピー] を選択してオブジェクトパスをローカルクリップボードにコピーします。

Amazon S3 オブジェクトの署名付き URL を生成します。

署名付き URL 機能を使用してダウンロードの期限付きのアクセス許可を付与することにより、プライベート Amazon S3 オブジェクトを共有できます。詳細については、「署名付き URL を使用したオブジェクトの共有」を参照してください。

1. Toolkit Explorer から、S3 サービスを展開します。
2. バケットまたはフォルダから、共有するオブジェクトのコンテキスト (右クリック) メニューを開きます。次に、「署名済み URL を生成」を選択してコマンドパレットを開きます。

3. コマンドパレットから、URL を使用してオブジェクトにアクセスできる時間を分単位で入力します。次に Enter キーを押して確認し、ダイアログを閉じます。
4. 署名済み URL が生成されると、ローカルクリップボードにコピーされたオブジェクトの署名済み URL が VS Code ステータスバーに表示されます。

Amazon S3 オブジェクトの削除

オブジェクトがバージョン管理されていないバケット内にある場合は、それを完全に削除できます。バージョニングが有効になっているバケットの場合、削除リクエストによってそのオブジェクトは完全に削除されません。代わりに、Amazon S3 はバケットに削除マーカーを挿入します。詳細については、「オブジェクトのバージョンを削除」を参照してください。

1. Toolkit エクスプローラーから、S3 サービスを展開して S3 リソースのリストを表示します。
2. 削除するオブジェクトのコンテキスト (右クリック) メニューを開き、[削除] を選択して確認ダイアログを開きます。
3. [削除] を選択して、S3 オブジェクトの削除を確認します。次に、ダイアログを閉じます。

サーバーレスアプリケーションの使用

AWS Toolkit for Visual Studio Codeは、AWS サーバーレスアプリケーション をサポートしています。以下のトピックでは、AWS Toolkit for Visual Studio Codeから AWS Serverless Application Model (AWS SAM) アプリケーションの作成と操作を開始する方法について説明します。

トピック

- サーバーレスアプリケーション入門
- コードから Lambda 関数を直接実行およびデバッグ
- ローカル Amazon API Gateway リソースの実行とデバッグ
- サーバーレスアプリケーションのデバッグ用設定オプション
- サーバーレスアプリケーションのトラブルシューティング

サーバーレスアプリケーション入門

次のセクションでは、AWS Serverless Application Model (AWS SAM) および EE スタックを使用して、AWS Toolkit for Visual Studio Code から AWS サーバーレスアプリケーション の作成を開始する方法について説明します。

前提条件

AWS サーバーレスアプリケーションを作成するには、次の前提条件を設定する必要があります。

> **Note**
>
> 次の操作では、変更が完了する前に VS Code を終了または再起動する必要がある場合があります。

- AWS SAM コマンドラインインターフェイス (CLI) のインストール AWS SAM CLI のインストール方法に関する追加情報と手順については、この AWS Serverless Application Model ユーザー ガイドの「AWS SAM CLI のインストール」トピックを参照してください。
- AWS 設定ファイルから、デフォルトの AWS リージョンを特定します。構成ファイルの詳細については、「AWS Command Line Interface ユーザー ガイド」の「構成と認証情報ファイルの設定」トピックを参照してください。
- 言語 SDK をインストールし、ツールチェーンを設定する AWS Toolkit for Visual Studio Code からツールチェーンを構成する方法の詳細については、このユーザー ガイドの「ツールチェーンの構成」トピックを参照してください。
- VS Code マーケットプレイスから YAML 言語サポートエクステンションをインストールします。これは、AWS SAMテンプレートファイルの CodeLens 機能にアクセスできるようにするために必要です。CodeLens に関する追加情報については、VS Code ドキュメントの「CodeLens」セクションを参照してください。

サーバーレスアプリケーションの IAM アクセス許可

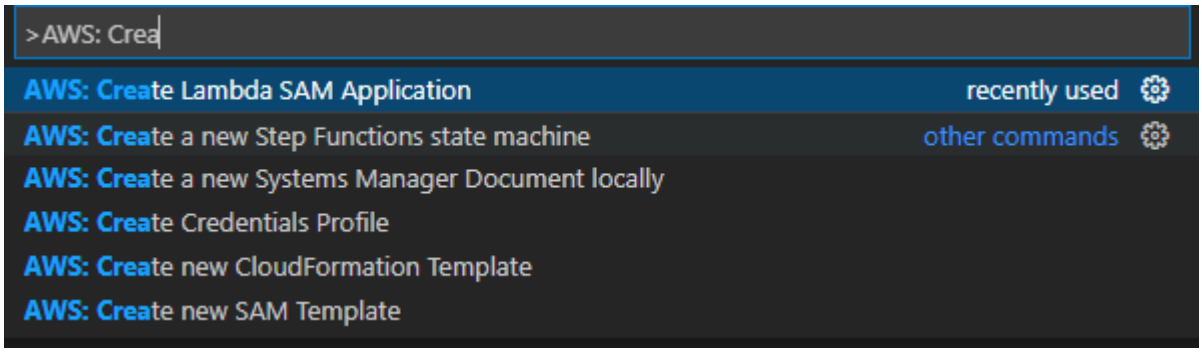
Toolkit for VS Code には、サーバーレスアプリケーションをデプロイして実行するために必要な AWS Identity and Access Management (IAM) のアクセス許可を含める認証情報プロファイルが必要です。以下のサービスへの適切な読み取り/書き込みアクセスが必要です: AWS CloudFormation、IAM、Lambda、Amazon API Gateway、Amazon Simple Storage Service (Amazon S3)、および Amazon Elastic Container Registry (Amazon ECR)。

サーバーレスアプリケーションのデプロイと実行に必要な認証の設定に関する追加情報については、「AWS Serverless Application Model開発者ガイド」の「リソースへのアクセスと権限の管理」を参照してください。認証情報の設定方法の詳細については、本ユーザーガイドのAWS IAM 認証情報を参照してください。

新しいサーバーレスアプリケーションの作成 (ローカル)

これは、AWS SAM を使用して、Toolkit for VS Code でサーバーレスアプリケーションを作成する方法を示します。この手順の出力は、サンプルサーバーレスアプリケーションを含む開発ホスト上のローカルディレクトリです。サーバーレスアプリケーションをビルド、ローカルでテスト、変更、および AWS クラウドにデプロイできます。

1. [Command Palette (コマンドパレット)] を開くには、[表示]、[Command Palette (コマンドパレット)] の順に選択し、[AWS] と入力します。
2. [AWS: Lambda SAM アプリケーションの作成] を選択します。



> **Note**
>
> AWS SAM CLI がインストールされていない場合は、 VS Code工ディタの右下隅に工ラーが表示されます。この問題が発生した場合は、すべての仮定条件と前提条件を満たしていることを確認します。

3. AWS SAM アプリケーションのランタイムを選択します。

> **Note**
>
> 「(イメージ」でランタイムの 1 つを選択した場合、アプリケーションはパッケージタイプ `Image` です。「(イメージ)」を使わずにランタイムの 1 つを選択した場合、アプリケーションのタイプは `Zip` です。との違いの詳細については、`Image` および `Zip` パッケージタイプのちがいについては、「AWS Lambdaデベロッパーガイド」の「Lambda デプロイパッケージ」を参照してください。

4. 選択したランタイムによっては、SAM アプリケーションの依存関係マネージャとランタイムアーキテクチャを選択するよう求められることがあります。

Dependency Manager

[Gradle] または [Maven] を選択します。

> **Note**
> このビルド自動化ツールの選択は Java ランタイムでのみ使用できます。

Architecture

[x86_64] または [arm64] を選択します。

デフォルトの x86_64 ベースの環境ではなく、ARM64 ベースのエミュレート環境でサーバーレスアプリケーションを実行するオプションは、次のランタイムで使用できます。

- nodejs12.x (ZIP とイメージ)
- nodejs14.x (ZIP とイメージ)
- python3.8 (ZIP とイメージ)
- python3.9 (ZIP とイメージ)
- python3.10 (ZIP とイメージ)
- python3.11 (ZIP とイメージ)
- java8.al2 と Gradle (ZIP とイメージ)
- java8.al2 と Maven (ZIP のみ)
- java11 と Gradle (ZIP とイメージ)
- java11 と Maven (ZIP のみ)

> **Important**
> ARM64 ベースの環境でアプリケーションを実行するために、AWS CLI バージョン 1.33.0 以降をインストールする必要があります。詳細については、「前提条件」を参照してください。

5. 新しいプロジェクトの場所を選択します。既存のワークスペースフォルダが開いている場合は、既存の別のフォルダを選択するか、新しいフォルダを作成して選択します。この例では、[There

are no workspace folders open] (開いているワークスペースフォルダはありません) を選択して、MY-SAM-APP という名前のフォルダを作成します。

6. 新しいプロジェクトの名前を入力します。この例では my-sam-app-nodejs を使用します。入力 を押した後、Toolkit for VS Code でプロジェクトが作成されるまで少し時間がかかります。

プロジェクトが作成されると、アプリケーションは現在のワークスペースに追加されます。[Explorer] ウィンドウに表示されます。

サーバーレスアプリケーションを開く (ローカル)

ローカル開発ホストでサーバーレスアプリケーションを開くには、アプリケーションのテンプレートファイルを含むフォルダを開きます。

1. [ファイル] メニューで、[フォルダを開く...] を選択します。
2. [Open Folder] (フォルダを開く) ダイアログボックスで、開きたいサーバーレスアプリケーションフォルダに移動します。
3. [Select Folder] (フォルダの選択) ボタンを選択します。

アプリケーションのフォルダを開くと、[Explorer] (エクスプローラ) ウィンドウに追加されます。

テンプレートからのサーバーレスアプリケーションの実行とデバッグ (ローカル)

Toolkit for VS Code を使用して、サーバーレスアプリケーションをデバッグし、開発環境でローカルで実行する方法を設定します。

VS Code CodeLens 機能を使用してデバッグ動作の設定を開始し、適格な Lambda 関数を識別できます。CodeLens を使用すると、コンテンツに応じたソースコードとのやり取りが可能になります。CodeLens 機能にアクセスできることを確認する方法については、このトピックの前半の 前提条件 セクションをレビューします。

> **Note**
>
> この例では、JavaScript を使用するアプリケーションをデバッグします。しかし、次の言語とランタイムを備えた Toolkit for VS Code では、デバッグの特徴を使用できます。
>
> - C# — .NET Core 2.1, 3.1; .NET 5.0
> - JavaScript/TypeScript — Node.js 12.x14.x
> - Python – 3.6, 3.7, 3.8, 3.9, 3.10, 3.11

- Java — 8, 8.al2, 11
- Go – 1.x

言語の選択は、CodeLens が適格な Lambda ハンドラーを検出する方法にも影響します。詳細については、「コードから Lambda 関数を直接実行およびデバッグ」を参照してください。

この手順では、このトピックの前半の 新しいサーバーレスアプリケーションの作成 (ローカル) セクションで作成したアプリケーションのサンプルを使用します。

1. VS Code のファイルエクスプローラでアプリケーションファイルを表示するには、[View] (表示)、[Explorer] (エクスプローラ) を選択します。
2. アプリケーションフォルダ (例:my-sample-app) で `template.yaml` ファイルを開きます。

   > **Note**
   >
   > `template.yaml` と違う名前のテンプレートを使用する場合、CodeLens インジケーターは YAML ファイルでは自動的には使用できません。つまり、デバッグ設定をマニュアルで追加する必要があります。

3. `template.yaml` のエディタでは、サーバーレスリソースを定義するテンプレートの `Resources` セクションに移動します。この場合、これはタイプ `AWS::Serverless::Function` の `HelloWorldFunction` リソースです。

   このリソースの CodeLens インジケータで、[Add Debug Configuration] (デバッグ設定の追加) を選択します。
4. [Command Palette] (コマンドパレット) で、AWS SAM アプリケーションが実行するランタイムを選択します。
5. `launch.json` ファイルのエディタでは、次の設定プロパティの値を編集または確認します。

   - `"name"` – 読みやすい名前を入力して、[実行] ビューの [設定] ドロップダウンフィールドに表示します。
   - `"target"` – 値が `"template"` であるため、AWS SAM テンプレートがデバッグセッションのエントリポイントになるように確認します。
   - `"templatePath"` – `template.yaml` ファイルに相対パスまたは絶対パスを入力。

- "logicalId" – 名前が AWS SAM テンプレートの リソース セクションで指定されたものに一致するように確認します。この場合はタイプ AWS::Serverless::Function の HelloWorldFunction です。

launch.json ファイル内のこれらと他の入力に関する詳細は、「サーバーレスアプリケーションのデバッグ用設定オプション」を参照してください。

6. デバッグ設定に問題がなければ、launch.json を保存します。次に、デバッグをスタートするため、RUN ビューにある緑色の [再生] ボタンを選択します。

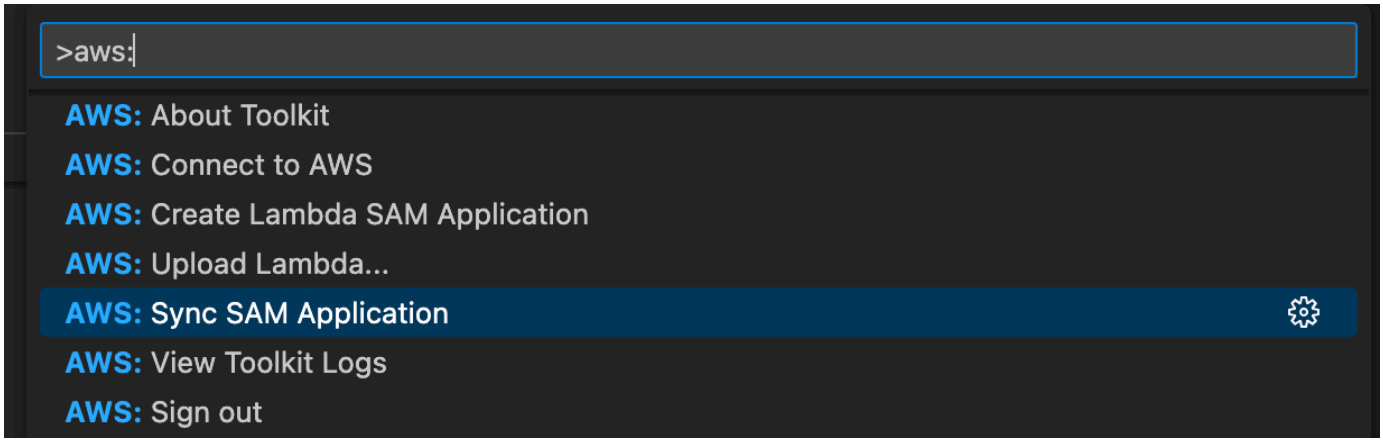
   デバッグセッションが開始すると、デバッグコンソールパネルには、デバッグ出力が表示され、Lambda 関数によって返される値が表示されます。(AWS SAM アプリケーションをデバッグする場合、AWS ツールキット が 出力 パネル内の 出力 チャンネルとして選択されています。)。

AWS SAMアプリケーションの同期

AWS Toolkit for Visual Studio Code は AWS SAM CLI コマンド sam sync を実行して、サーバーレス アプリケーションを AWS クラウド にデプロイします。AWS SAM 同期の詳細については、AWS Serverless Application Model デベロッパーガイドの AWS SAM CLI コマンド リファレンス トピックを参照してください。

次の手順では、Toolkit for VS Code から sam sync を使用してサーバーレス アプリケーションを AWS クラウド にデプロイする方法について説明します。

1. VS Code のメインメニューから、[表示] を展開し、[コマンドパレット] を選択してコマンド パレットを開きます。
2. コマンド パレットで AWS を検索し、[Sync SAM アプリケーションを同期] を選択して同期の設定を開始します。



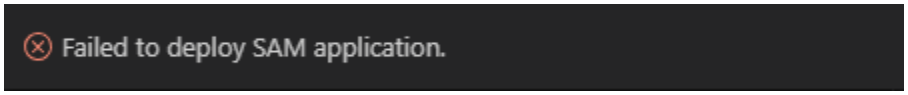
3. サーバーレスアプリケーションを同期するAWSリージョンを選択します。
4. デプロイに使用する `template.yaml` ファイルを選択します。
5. アプリケーションのデプロイ先となる既存の Amazon S3 バケットを選択するか、新しい Amazon S3 バケット名を入力します。

> **Important**
>
> Amazon S3 バケットは、以下の要件を満たしている必要があります。
>
> - バケットは、同期先のリージョンにある必要があります。
> - Amazon S3 バケット名は、Amazon S3 内のどの既存バケット名とも重複しないグローバルに一意な名前である必要があります。

6. サーバーレスアプリケーションに、パッケージタイプの `Image` を伴う関数が含まれているのであれば、このデプロイで使用できる Amazon ECR リポジトリの名前を入力します。リポジトリは、デプロイ先のリージョンにある必要があります。
7. 以前のデプロイのリストからデプロイスタックを選択するか、新しいスタック名を入力して新しいデプロイスタックを作成します。次に、同期プロセスを開始します。

> **Note**
>
> 以前のデプロイで使用されたスタックは、ワークスペースとリージョンごとにリコールされます。

8. 同期プロセス中、デプロイのステータスが VS Code の [ターミナル] タブにキャプチャされます。ターミナルタブで同期が成功したことを確認します。エラーが発生すると、通知が届きます。



> **Note**
>
> 同期に関するその他の詳細については、AWS Toolkit for Visual Studio Codeコマンドパレットからログにアクセスできます。

コマンドパレットから AWS Toolkit for Visual Studio Code ログにアクセスするには、[表示] を展開し、[コマンドパレット] を選択してから **`AWS: View AWS Toolkits Logs`** を検索し、リストに表示されたらそれを選択します。

デプロイが完了したら、アプリケーションが [AWSExplorer] に表示されます。アプリケーションの一部として作成した Lambda 関数の呼び出し方法の詳細については、このユーザーガイドのリモート Lambda 関数の操作トピックを参照してください。

AWS クラウド からのサーバーレスアプリケーションの削除

サーバーレスアプリケーションを削除すると、AWS クラウドに以前、デプロイした AWS CloudFormation スタックも削除されます。この手順を実行しても、ローカルホストからアプリケーションディレクトリは削除されないことにご注意ください。

1. AWS Explorer を開きます。
2. [AWSToolkit Explorer] ウィンドウで、削除したいデプロイされたアプリケーションを含むリージョンを展開し、AWS CloudFormationを拡張します。
3. 削除したいサーバーレスアプリケーションに対応する AWS CloudFormation スタック名のコンテキスト (右クリック) メニューを開き、AWS CloudFormation スタックの削除 を選択します。
4. 選択したスタックを削除したいことを確認する場合は、[はい] を選択します。

スタックの削除が成功すれば、Toolkit for VS Code は、AWS Explorer内の AWS CloudFormation リストからスタック名を削除します。

コードから Lambda 関数を直接実行およびデバッグ

AWS SAM アプリケーションをテストする場合、Lambda 関数だけを実行およびデバッグだけを選択し、AWS SAM テンプレートで定義されている他のリソースを除外することができます。このアプ

ローチでは、CodeLens 機能を使用して、直接呼び出すことができるソースコード内の Lambda 関数ハンドラを識別します。

CodeLens によって検出される Lambda ハンドラーは、アプリケーションで使用している言語とランタイムによって異なります。

言語/ランタイム	CodeLens インジケーターによって識別される Lambda 関数の基準
C# (dotnetcore2.1, 3.1; .NET 5.0)	関数には以下の特徴があります。 • パブリッククラスのパブリック関数です。 • 1 つまたは 2 つのパラメータがあります。2 つのパラメータを使用する場合、2 番目のパラメータは `ILambdaContext` インターフェイスを実装する必要があります。 • VS Code ワークスペースフォルダー内の親フォルダーに `*.csproj` ファイルがあります。 ms-dotnettools.csharp 拡張機能 (または C# の言語シンボルを提供する拡張機能) がインストールされ、有効になっています。
JavaScript/TypeScript (Node.js 12.x, 14.x)	関数には以下の特徴があります。 • 最大 3 つのパラメータがあるエクスポートされた関数です。 • VS Code ワークスペースフォルダー内の親フォルダーに `package.json` ファイルがあります。
Python (3.7, 3.8, 3.9, 3.10, 3.11)	関数には以下の特徴があります。 • 上位レベルの関数です。 • VS Code ワークスペースフォルダー内の親フォルダーに `requirements.txt` ファイルがあります。

言語/ランタイム	CodeLens インジケーターによって識別される Lambda 関数の基準
	ms-python.python 拡張機能 (または Python の言語シンボルを提供する拡張機能) がインストールされ、有効になっています。

言語/ランタイム	CodeLens インジケーターによって識別される Lambda 関数の基準

Java (8, 8.al2, 11)	関数には以下の特徴があります。
	• これは、パブリックで非抽象クラスのパブリック関数です。
	• 1 つ、2 つ、または 3 つのパラメータがあります。
	  • 1 つのパラメータ: パラメータは何でもかまいません。
	  • 2 つのパラメータ: パラメーターは `java.io.InputStream` と `java.io.OutputStream` 、または、最後のパラメータは `com.amazonaws.services.lambda.runtime.Context` である必要があります。
	  • 3 つのパラメータ: パラメータは、`java.io.InputStream` と `java.io.OutputStream` 、そして最後のパラメータは `com.amazonaws.services.lambda.runtime.Context` にする必要があります。
	• VS Code ワークスペースフォルダー内の親フォルダーに `build.gradle` (Gradle) や `pom.xml` (Maven) ファイル があります。
	
	redhat.java 拡張機能 (または Java の言語シンボルを提供する拡張機能) がインストールされ、有効になっています。この拡張機能には、どの Java ランタイムを使用していても Java 11 が必要です。

言語/ランタイム	CodeLens インジケーターによって識別される Lambda 関数の基準
	vscjava.vscode-java-debug 拡張機能 (または Java デバッガを提供する拡張機能) がインストールされ、有効になっています。
Go (1.x)	関数には以下の特徴があります。 • 上位レベルの関数です。 • 0 から 2 までの引数を取ります。2 つの引数がある場合は、最初の引数が `context.Context` を実装する必要があります。 • 0 から 2 までの引数を返します。0 以上の引数がある場合は、最後の引数が `error` を実装する必要があります。 • VS Code ワークスペースフォルダー内に `go.mod` ファイルがあります。 golang.go 拡張機能 がインストールされ、構成され、有効になっています。

サーバーレスアプリケーションをアプリケーションコードから直接実行およびデバッグ

1. VS Code のファイルエクスプローラでアプリケーションファイルを表示するには、[View] (表示)、[Explorer] (エクスプローラ) を選択します。
2. アプリケーションフォルダ (my-sample-app など) から、関数フォルダー (この場合、hello-world) を拡張し、`app.js` ファイルを開きます。
3. 適格な Lambda 関数 ハンドラーを識別する CodeLens インジケータで、`Add Debug Configuration` を選択します。
4. [Command Palette] (コマンドパレット) で、AWS SAM アプリケーションが実行するランタイムを選択します。
5. `launch.json` ファイルのエディタでは、次の設定プロパティの値を編集または確認します。

- "name" – 読みやすい名前を入力して、[実行] ビューの [設定] ドロップダウンフィールドに表示します。
- "target" – 値が "code" で、Lambda 関数ハンドラを直接呼び出せることを確認します。
- "lambdaHandler" – 関数を呼び出すために Lambda が呼び出すコード内のメソッドの名前を入力します。例えば、JavaScript のアプリケーションでは、デフォルトは app.lambdaHandler です。
- "projectRoot" – Lambda 関数を含むアプリケーションファイルにパスを入力します。
- "runtime" – Lambda 実行環境の有効なランタイムを入力または確認します。例えば、"nodejs.12x"。
- "payload" – 以下のいずれかのオプションを選択して、Lambda 関数に入力として提供するイベントペイロードを定義します。
  - "json": イベントペイロードを定義する JSON 形式のキーバリューペア。
  - "path": イベントペイロードとして使用されるファイルへのパス。

  以下の例では、"json" オプションは、ペイロードを定義します。

launch.json ファイル内のこれらと他の入力に関する詳細は、「サーバーレスアプリケーションのデバッグ用設定オプション」を参照してください。

6. デバッグ設定が満足なものであれば、[RUN] の横の緑の再生矢印を選択して、デバッグをスタートします。

   デバッグセッションが開始すると、デバッグコンソールパネルには、デバッグ出力が表示され、Lambda 関数が返す値が表示されます。(AWS SAM アプリケーションをデバッグする場合、AWS ツールキットが [出力] パネル内の [出力] チャンネルとして選択されています)。

ローカル Amazon API Gateway リソースの実行とデバッグ

invokeTarget.target=api と共に VS Codeの type=aws-sam 起動設定を実行することによって、template.yaml に指定されている AWS SAM API Gateway のローカルリソースを実行またはデバッグすることができます。

> **Note**
>
> API Gateway は、REST と HTTP の 2 種類の API をサポートしています。しかし、AWS Toolkit for Visual Studio Code がある API Gateway の特徴は、REST API のみをサポートします。時に、HTTP API は「API Gateway V2 API」と呼ばれます。

ローカル API Gateway リソースを実行およびデバッグ

1. 以下のいずれかのアプローチを選択し、AWS SAM API Gateway リソース用起動 Config を作成:
   - オプション 1:AWS SAM プロジェクトにあるハンドラーのソースコード (.js、.cs、または.py ファイル) にアクセスし、Lambda ハンドラーの上で移動して、デバッグ設定の追加 CodeLens を選択します。次に、メニューで API イベントとマークされた項目を選択します。
   - オプション 2 `launch.json` を編集して、次の構文を使用して新しい起動設定を作成します。

```
{
  "type": "aws-sam",
  "request": "direct-invoke",
  "name": "myConfig",
  "invokeTarget": {
    "target": "api",
    "templatePath": "n12/template.yaml",
    "logicalId": "HelloWorldFunction"
  },
  "api": {
    "path": "/hello",
    "httpMethod": "post",
    "payload": {
      "json": {}
    }
  },
  "sam": {},
  "aws": {}
}
```

2. VS Code [Run] (実行) パネルは、起動設定 (上の例では `myConfig` という名前) を選択します。
3. (オプション) Lambda プロジェクトコードにブレークポイントを追加します。

4. [F5] をタイプするか、または [Run] (実行) パネルの [Play] (再生) を選択します。
5. 出力ペインで、結果を表示します。

構成

invokeTarget.target プロパティ値 api を使用すると、ツールキットは起動設定の検証と動作を変更して、api フィールドをサポートします。

```
{
  "type": "aws-sam",
  "request": "direct-invoke",
  "name": "myConfig",
  "invokeTarget": {
    "target": "api",
    "templatePath": "n12/template.yaml",
    "logicalId": "HelloWorldFunction"
  },
  "api": {
    "path": "/hello",
    "httpMethod": "post",
    "payload": {
      "json": {}
    },
    "querystring": "abc=def&qrs=tuv",
    "headers": {
        "cookie": "name=value; name2=value2; name3=value3"
    }
  },
  "sam": {},
  "aws": {}
}
```

例の値を、次のように置き換えます。

invokeTarget.logicalId

API リソース。

パス

起動設定が要求する API パス (例:"path": "/hello")。

invokeTarget.templatePath によって指定される template.yaml から解決された有効な API パスでなければなりません。

httpMethod

「delete」(削除)、「get」(取得)、「head」(率いる、ヘッド)、「option」(オプションを与える)、「patch」(パッチ)、「post」(転記、投稿)、「put」(プット、つける) のいずれかの動詞とすることができます。

payload

リクエストで送信する lambda.payload フィールドと同じ構造とルールを持つ JSON ペイロード (HTTP 本文)。

payload.pathは JSON ペイロードを含むファイルを指します。

payload.jsonは JSON ペイロードをインラインで指定します。

headers

名前と値のペアのオプションのマップ。以下の例のようにリクエストに含める HTTP ヘッダーを指定するため使用します。

```
"headers": {
     "accept-encoding": "deflate, gzip;q=1.0, *;q=0.5",
     "accept-language": "fr-CH, fr;q=0.9, en;q=0.8, de;q=0.7, *;q=0.5",
     "cookie": "name=value; name2=value2; name3=value3",
     "user-agent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6)
 AppleWebKit/537.36 (KHTML, like Gecko) Chrome/86.0.4240.198 Safari/537.36",
}
```

querystring

リクエストの querystring を設定するオプションの文字列 (例: "querystring": "abc=def&ghi=jkl")。

AWS

AWS 接続情報を提供する方法。詳細については、サーバーレスアプリケーションのデバッグ用設定オプション セクションの [AWS 接続 (aws) プロパティ]テーブルを参照してください。

SAM

AWS SAM CLIがアプリケーションを構築する方法 詳細については、サーバーレスアプリケーションのデバッグ用設定オプション セクションの [AWS SAM CLI (sam) プロパティ] テーブルを参照してください。

サーバーレスアプリケーションのデバッグ用設定オプション

`launch.json` ファイルを開いてデバッグ設定を編集する場合は、表示するために VS Code IntelliSense 機能を使用して、有効なプロパティを表示して自動的に完了できます。エディタで IntelliSense をトリガーするには、Ctrl + スペースバーを押します。

IntelliSense では、Lambda 関数を直接呼び出すために、または AWS SAM テンプレート によりプロパティを検索し、定義できます。"lambda" (関数の実行方法) 、"sam" (AWS SAM CLIがアプリケーションを構築する方法) 、および "aws" (AWS 接続情報の提供方法) に関するプロパティを定義することもできます。

AWS SAM: 直接 Lambda ハンドラー呼び出し/テンプレートベースの Lambda 呼び出し

プロパティ	説明
`type`	起動設定を管理する拡張機能を指定します。常に `aws-sam` に設定して AWS SAM CLI を使用して、ローカルで構築およびデバッグを行います。
`name`	起動設定のデバッグリストに表示される読みやすい名前を指定します。
`request`	指定された拡張子 (`aws-sam`) が実行する構成の種類を指定します。常に `direct-invoke` に設定され、Lambda 関数をスタートします。
`invokeTarget`	リソースを呼び出すためのエントリポイントを指定します。

プロパティ	説明
	Lambda 関数を直接呼び出すには、次のinvokeTarget フィールドに値を設定: • target – code に設定します。 • lambdaHandler – 呼び出す予定の Lambda 関数ハンドラの名前。 • projectRoot – Lambda 関数ハンドラーを含むアプリケーションファイルのパス。 • architecture — ローカル SAM Lambda アプリケーションが実行されるエミュレート環境のプロセッサアーキテクチャ。特定のランタイムでは、デフォルトの x86_64 アーキテクチャの代わりに arm64 を選択できます。詳細については、「新しいサーバーレスアプリケーションの作成 (ローカル)」を参照してください。 AWS SAM テンプレートを使用して Lambda リソースを呼び出すには、次の invokeTarget フィールドに値を設定: • target – template に設定します。 • templatePath – AWS SAM テンプレートファイルへのパス。 • logicalId – 呼び出す リソース名 AWS::Lambda::Function またはAWS::Serverless::Function 。リソース名は、YAML 形式の AWS SAM テンプレートで検索できます。AWS Toolkit は暗黙に、イメージベース Lambda 関数として AWS SAM テンプレート内の PackageType: Image で定義された関数を認識します。詳細については、「AWS Lambda デベロッパーガイド」の「Lambda デプロイパッケージ」を参照してください。

Lambda (**"lambda"**) のプロパティ

プロパティ	説明

environmentVariables	オペレーショナルパラメータを Lambda 関数に渡します。例えば、Amazon S3 バケットに書き込む場合、書き込み先のバケット名はハードコーディングせずに、環境可変として設定します。
	
	Note
	サーバーレスアプリケーションの環境変数を指定する場合は、AWS SAM テンプレート(template.yaml) および launch.json ファイルの両方に設定を追加する必要があります。
	AWS SAM テンプレートの環境変数の書式設定の例
	`Resources:`
	`HelloWorldFunction:`
	`Type: AWS::Serverless::Function`
	`Properties:`
	`CodeUri: hello-world/`
	`Handler: app.lambdaHandlerN10`
	`Runtime: nodejs10.x`
	`Environment:`
	`Variables:`
	`SAMPLE1: Default Sample 1 Value`
	
	launch.json ファイルの環境変数の書式設定の例
	`"environmentVariables": {`
	`"SAMPLE1": "My sample 1 value"`
	`}`

payload	入力として Lambda 関数に提供されるイベントペイロード用に 2 つのオプションを提供します。 • "json": イベントペイロードを定義する JSON 形式のキーバリューのペア。

プロパティ	説明
	• "path": イベントペイロードとして使用されるファイルへのパス。
memoryMB	呼び出された Lambda 関数の実行のために提供されたメモリのメガバイト (MB) を指定します。
runtime	Lambda 関数で使用するランタイムを指定します。詳細については、「AWS Lambda ランタイム」を参照してください。
timeoutSec	デバッグセッションがタイムアウトするまでの許可される時間を秒単位で設定します。

プロパティ	説明
pathMappings	ローカルコードがコンテナ内のどこで実行されるかを指定します。 デフォルトでは、Toolkit for VS Code が localRoot をローカルワークスペースの Lambda 関数のコードルート、および remoteRoot から Lambda で実行されるコードのデフォルトの作業ディレクトリの /var/task に設定します。作業ディレクトリを Dockerfile や AWS CloudFormation テンプレートファイルの WorkingDirectory パラメータで変更する場合、少なくとも 1 つの pathMapping エントリがローカルに設定されたブレークポイントを Lambda コンテナで実行されているコードに正常にマッピングできるように、指定される必要があります。 launch.json ファイルの pathMappings の書式設定の例 `"pathMappings": [ { "localRoot": " ${workspaceFolder}/sam-app/HelloWorldFunction ", "remoteRoot": " /var/task " } ]` 注意: • .NET イメージベースの Lambda 関数の場合、remoteRoot エントリはビルドディレクトリである必要があります。 • Node.js ベースの Lambda 関数の場合は、指定できるパスマッピングエントリは 1 つだけです。

Toolkit for VS Code は AWS SAM CLI を使用して、サーバーレスアプリケーションをローカルで構築およびデバッグできます。launch.json ファイル内の "sam" 設定のプロパティを使って AWS SAM CLI コマンドの動作を設定できます。

AWS SAM CLI (**"sam"**) プロパティ

プロパティ	説明	デフォルト値
buildArguments	sam build コマンドが Lambda ソースコードを構築する方法を設定します。構築オプションを表示するには、「AWS Serverless Application Model デベロッパーガイド」の「sam build」を参照してください。	空の文字列
containerBuild	Lambda　のようなDocker コンテナ内部に関数を構築するかどうかを示します。	false
dockerNetwork	Lambda Docker コンテナが接続する既存の Docker ネットワークの名前または ID を、デフォルトのブリッジネットワークとともに指定します。指定されていない場合、Lambda コンテナはデフォルトのブリッジ Docker ネットワークのみに接続します。	空の文字列
localArguments	追加のローカル呼び出し引数を指定します。	空の文字列
skipNewImageCheck	コマンドが Lambda ランタイム用の最新 Docker イメージのプルダウンをスキップするかどうかを指定します。	false
template	AWS SAM パラメータを使用して顧客の値を入力することで、SAM テンプレートをカ	"parameters":{}

プロパティ	説明	デフォルト値
	スタマイズします。詳細については、「AWS CloudFormation ユーザーガイド」の「パラメータ」を参照してください。	

AWS接続 (**"aws"**) プロパティ

プロパティ	説明	デフォルト値
`credentials`	認証情報ファイルから特定のプロファイルを選択 (例えば、`profile:default`) して、AWS 認証情報を取得します。	既存の共有 AWS Config ファイルまたは共有 AWS 認証情報ファイル が Toolkit for VS Code に提供する AWS 認証情報。
`region`	サービスの AWS リージョン (us-east-1 など) を設定します。	アクティブな認証情報プロファイルに関連付けられたデフォルトの AWSリージョン。

例: テンプレートの起動設定

AWS SAM テンプレートターゲットに関して、以下に起動設定ファイルの例を示します。

```
{
    "configurations": [
        {
            "type": "aws-sam",
            "request": "direct-invoke",
            "name": "my-example:HelloWorldFunction",
            "invokeTarget": {
                "target": "template",
                "templatePath": "template.yaml",
                "logicalId": "HelloWorldFunction"
            },
            "lambda": {
                "payload": {},
```

```
                "environmentVariables": {}
            }
        }
    ]
}
```

例: コード起動設定

Lambda 関数ターゲットの起動設定ファイルの例を次に示します。

```
{
    "configurations": [
        {
            "type": "aws-sam",
            "request": "direct-invoke",
            "name": "my-example:app.lambda_handler (python3.7)",
            "invokeTarget": {
                "target": "code",
                "projectRoot": "hello_world",
                "lambdaHandler": "app.lambda_handler"
            },
            "lambda": {
                "runtime": "python3.7",
                "payload": {},
                "environmentVariables": {}
            }
        }
    ]
}
```

サーバーレスアプリケーションのトラブルシューティング

このトピックでは、Toolkit for VS Code を使用してサーバーレスアプリケーションを作成するときに発生する可能性のある、一般的なエラーと問題を解決する方法について詳しく説明します。

トピック

- SAM 起動設定で samconfig.toml をどのように使用できますか？
- エラー:「RuntimeError: コンテナは存在しません」
- エラー:「docker.errors.APIError: 500 サーバーエラー..。プルレート制限に達しました。」
- エラー:「500 Server Error: Mounting C:\Users\...」

- WSL を使用するとウェブビュー (例えば、「AWS の呼び出し」形式) が壊れる
- TypeScript アプリケーションをデバッグするが、ブレークポイントが機能しない

SAM 起動設定で samconfig.toml をどのように使用できますか？

起動設定の `sam.localArguments` プロパティ内の `--config-file` 引数を設定して、SAM CLI samconfig.toml の場所を指定します。例えば、samconfig.toml ファイルがワークスペースの最上位にある場合は、次のようにします。

```
"sam": {
    "localArguments": ["--config-file", "${workspaceFolder}/samconfig.toml"],
}
```

エラー:「RuntimeError: コンテナは存在しません」

システムに Docker コンテナ用の十分なディスク領域がない場合、`sam build` コマンドでこのエラーが表示されることがあります。システムストレージに空き容量が 1～2 GB しかない場合は、`sam build` はビルドの開始前にシステムストレージが完全にいっぱいでなくても、処理中に失敗することがあります。詳細については、this GitHub issue を参照してください。

エラー:「docker.errors.APIError: 500 サーバーエラー..。プルレート制限に達しました。」

Docker Hub は、匿名ユーザーが行うことができるリクエストを制限します。システムが制限に達すると、Docker が失敗し、VS Code の OUTPUT ビューにこのエラーが表示されます。

```
docker.errors.APIError: 500 Server Error: Internal Server Error ("toomanyrequests: You
 have
reached your pull rate limit. You may increase the limit by authenticating and
 upgrading:
https://www.docker.com/increase-rate-limit")
```

システムの Docker サービス は Docker Hub 認証情報で認証されていることを確認してください。

エラー:「500 Server Error: Mounting C:\Users\...」

AWS SAM アプリケーションをデバッグする場合に、Windows ユーザーにこの Docker マウントエラーが表示されることがあります。

```
Fetching lambci/lambda:nodejs10.x Docker container image......
2019-07-12 13:36:58 Mounting C:\Users\<username>\AppData\Local\Temp\ ... as /var/
task:ro,delegated inside runtime container
Traceback (most recent call last):
...
requests.exceptions.HTTPError: 500 Server Error: Internal Server Error ...
```

共有ドライブの認証情報を更新してみてください (Docker 設定において)。

WSL を使用するとウェブビュー (例えば、「AWS の呼び出し」形式) が壊れる

これは、Cisco VPN のユーザにとって既知の VS コードの問題です。詳細については、this GitHub issue を参照してください。

回避策は、この WSL トラッキングの問題で提案されています。

TypeScript アプリケーションをデバッグするが、ブレークポイントが機能しない

これは、コンパイルされた JavaScript ファイルをソース TypeScript ファイルにリンクするソースマップがない場合に発生します。この問題を修正するには、`tsconfig.json` ファイルを開き、次のオプションと値が `"inlineSourceMap": true` に設定されていることを確認します。

Systems Manager オートメーションドキュメントの使用

AWS Systems Manager は、AWS 上のインフラを可視化し、コントロールすることができます。Systems Manager を使用すると、統合ユーザーインターフェイスで AWS のさまざまなサービスの運用データを確認でき、AWS リソース全体に関わる運用タスクを自動化できます。

システムマネージャのドキュメントは、マネージドインスタンスで Systems Manager が実行するアクションを定義します。オートメーションドキュメントは、Amazon Machine Image (AMI) の作成や更新など、一般的なメンテナンスやデプロイメントタスクを実行する際に使用する、Systems Manager キュメントを使用します。このトピックでは、AWS Toolkit for Visual Studio Code を使用して自動化ドキュメントを作成、編集、公開、削除する方法について説明します。

トピック

- 仮定条件と前提条件
- Systems Manager Automation ドキュメントの IAM アクセス許可
- Systems Manager の自動化ドキュメントの新規作成
- 既存のSystems Manager 自動化ドキュメントを開く

- Systems Manager Automation ドキュメントの編集
- Systems Manager Automation ドキュメントの公開
- Systems Manager Automation ドキュメントの削除
- Systems Manager Automation ドキュメントの実行
- Toolkit for VS Code の Systems Manager Automation ドキュメントのトラブルシューティング

仮定条件と前提条件

開始する前に、以下を確認してください。

- Visual Studio Code と、AWS Toolkit for Visual Studio Code の最新バージョンをインストールしています。詳細については、「AWS Toolkit for Visual Studio Code のインストール」を参照してください。
- Systems Manager に精通しています 詳細については、「AWS Systems Manager ユーザーガイド」を参照してください。
- Systems Manager オートメーションのユースケースに精通しています。詳細については、「AWS Systems Manager ユーザーガイド」の「AWS Systems Manager オートメーション」を参照してください。

Systems Manager Automation ドキュメントの IAM アクセス許可

Toolkit for VS Code には、Systems Manager Automation ドキュメントを作成、編集、公開、削除するために必要な AWS Identity and Access Management (IAM) のアクセス許可を含む、認証情報プロファイルがある必要があります。次のポリシードキュメントは、プリンシパルポリシーで使用できる必要な IAM アクセス権限を定義します。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ssm:ListDocuments",
                "ssm:ListDocumentVersions",
                "ssm:DescribeDocument",
                "ssm:GetDocument",
                "ssm:CreateDocument",
```

```
                "ssm:UpdateDocument",
                "ssm:UpdateDocumentDefaultVersion",
                "ssm:DeleteDocument"
            ],
            "Resource": "*"
        }
    ]
}
```

IAM ポリシーの更新方法については、「IAM ユーザーガイド」の「IAM ポリシーの作成 」を参照してください。認証情報プロファイルの設定方法については、「AWS IAM 認証情報」を参照してください。

Systems Manager の自動化ドキュメントの新規作成

JSON または YAML Visual Studio コードで新しいオートメーションドキュメントを作成できます。新しいオートメーションドキュメントを作成すると、そのドキュメントはタイトルのないファイルに表示されます。ファイルに名前を付けて VS Code に保存することはできますが、ファイル名は AWS には表示されません。

自動化ドキュメントの作成

1. VSコードを開きます。
2. [表示]で、[コマンドパレット]を選択して、[コマンドパレット]を開きます。
3. コマンドパレットで、「AWS Toolkit が新しいSystems Manager ドキュメントをローカルで作成する」と入力します。
4. Hello World の例については、スターターテンプレートの 1 つを選択します。
5. [JSON] または [YAML] のいずれかを選択します。

   新しいオートメーションドキュメントが作成されます。

> **Note**
>
> VS Code の新しいオートメーションドキュメントは、自動的に AWS には表示されません。実行するには、ドキュメントを AWS に発行する必要があります。

既存のSystems Manager 自動化ドキュメントを開く

AWS Explorer を使用して、既存の Systems Manager Automation ドキュメントを検索します。既存のオートメーションドキュメントを開くと、VS Code に無題のファイルとして表示されます。

オートメーションドキュメントを開くには

1. VSコードを開きます。
2. 左側のナビゲーションで、[AWS] を選択して AWS Explorer を開きます。
3. AWS Explorer で、Systems Manager のために、開きたいドキュメントのダウンロードアイコンを選択し、ドキュメントのバージョンを選択します。ファイルは、そのバージョンの形式で開きます。それ以外の場合は、[JSON としてダウンロードする] または [YAML としてダウンロードする] を選択します。

> **Note**
>
> オートメーションドキュメントを VS Code にファイルとしてローカルに保存しても、そのドキュメントは AWS に表示されません。実行する前に AWS に公開する必要があります。

Systems Manager Automation ドキュメントの編集

オートメーションドキュメントを所有している場合、それらは AWS Explorer 内の Systems Manager のドキュメントの [私が所有] カテゴリに表示されます。AWS にすでに存在するオートメーションドキュメントを所有でき、VS コードから AWS に以前公開した新規ドキュメントまたは更新されたドキュメントを所有できます。

VS Code で編集用にオートメーションドキュメントを開くと、AWS Management Console で行えること以上のことができます。例:

- JSON および YAML 形式両方にスキーマ検証があります。
- ドキュメントエディタには、オートメーションステップタイプの作成に使用できるスニペットがあります。
- JSON および YAML のさまざまなオプションにオートコンプリートサポートがあります。

バージョンの使用

Systems Manager オートメーションドキュメントでは、変更管理にバージョンが使用されます。VS Code では、オートメーションドキュメントのデフォルトバージョンを選択できます。

デフォルトバージョンを設定するには

- AWS Explorer で、デフォルトバージョンを設定するドキュメントに移動し、ドキュメントのコンテキスト (右クリック) メニューを開き、[デフォルトバージョンの設定] を選択します。

> **Note**
>
> 選択したドキュメントに 1 つのバージョンしかない場合は、デフォルトを変更することはできません。

Systems Manager Automation ドキュメントの公開

VS Code でオートメーションドキュメントを編集したら、そのドキュメントを AWS に公開できます。

オートメーションドキュメントを公開するには

1. 既存のSystems Manager 自動化ドキュメントを開く で説明されている手順を使用して、公開する自動化ドキュメントを開きます。
2. 公開したい変更を行います。詳細については、「Systems Manager Automation ドキュメントの編集」を参照してください。
3. 開いているファイルの右上にある [Upload] (アップロード) アイコンを選択します。
4. 公開ワークフロー]ダイアログボックスで、公開したいオートメーションドキュメントを公開する AWS リージョンを選択します。
5. 新しいドキュメントを公開する場合は、[Quick Create] を選択します。それ以外の場合は、AWS リージョンの既存のオートメーションドキュメントを更新するに クイックアップデート を選択します。
6. このオートメーションドキュメントの名前を入力します。

AWS に既存のオートメーションドキュメントに更新をパブリッシュすると、新しいバージョンがドキュメントに追加されます。

Systems Manager Automation ドキュメントの削除

VS Code でオートメーションドキュメントを削除できます。オートメーションドキュメントを削除すると、ドキュメントとドキュメントのすべてのバージョンが削除されます。

> **Important**
>
> - 削除は破壊的なアクションで、元に戻すことはできません。
> - すでに実行されているオートメーション ドキュメントを削除しても、開始時に作成または変更された AWS リソースは削除されません。

オートメーションドキュメントを削除するには

1. VSコードを開きます。
2. 左側のナビゲーションで、[AWS] を選択して AWS Explorer を開きます。
3. AWS Explorer で、Systems Manager のために、削除したいドキュメントのコンテキスト (右クリック) メニューを開き、[ドキュメントの削除] を選択します。

Systems Manager Automation ドキュメントの実行

オートメーションドキュメントが AWS に公開されたら、それを実行して、ユーザーに代わって AWS アカウントでタスクを実行することができます。オートメーションドキュメントを実行するには、AWS Management Console、Systems Manager API、AWS CLI、または AWS Tools for PowerShell を使用します。オートメーションドキュメントの実行方法については、「AWS Systems Manager ユーザーガイド」の「シンプルなオートメーションを実行する」を参照してください。

Systems Manager API を備えた AWS SDK のいずれかを使用してオートメーションドキュメントを実行する場合は、「AWS SDK リファレンス」を参照してください。

> **Note**
>
> オートメーションドキュメントを実行すると、AWS に新しいリソースを作成でき、請求コストが発生する可能性があります。オートメーションドキュメントを開始する前に、アカウントにどのようなリソースが作成されるかを把握することを強くお勧めします。

Toolkit for VS Code の Systems Manager Automation ドキュメントのトラブルシューティング

VS Code にオートメーションドキュメントを保存したけれど、AWS Management Console にそれが見えません。

VS Code にオートメーションドキュメントを保存しても、オートメーションドキュメントは AWS に発行されません。Automation ドキュメントの公開の詳細については、「Systems Manager Automation ドキュメントの公開」を参照してください。

オートメーションドキュメントの公開がパーミッションエラーで失敗しました。

AWS 認証情報プロファイルには、オートメーションドキュメントを発行するために必要な権限があることを確認してください。アクセス許可ポリシーの例については、「Systems Manager Automation ドキュメントの IAM アクセス許可」を参照してください。

オートメーションドキュメントを AWS に公開しましたが、AWS Management Console にそれが見えません。

AWS Management Console で閲覧している同じ AWS リージョンに、ドキュメントを公開していることを確認してください。

オートメーションドキュメントを削除しましたが、そのドキュメントが作成したリソースに対して課金されます。

オートメーションドキュメントを削除しても、作成または変更したリソースは削除されません。AWS Billing Management Console から作成した AWS リソースを特定し、料金を調べて、そこから削除するリソースを選択できます。

の使用 AWS Step Functions

AWS Toolkit for Visual Studio Code (VS Code) は、 のサポートを提供しますAWS Step Functions。VS Code の Toolkit を使用して、Step Functions ステートマシンを作成、更新、および実行できます。

トピック

- の使用 AWS Step Functions

の使用 AWS Step Functions

AWS Toolkit for Visual Studio Code (VS Code) を使用して、ステートマシン でさまざまなオペレーションを実行できます。

トピック

前提条件

- システムが「Toolkit for VS Code をインストールする」に指定されている前提条件を満たしていることを確認し、ツールキットをインストールします。
- AWS Explorer を開く前に、認証情報が設定済みであることを確認します。

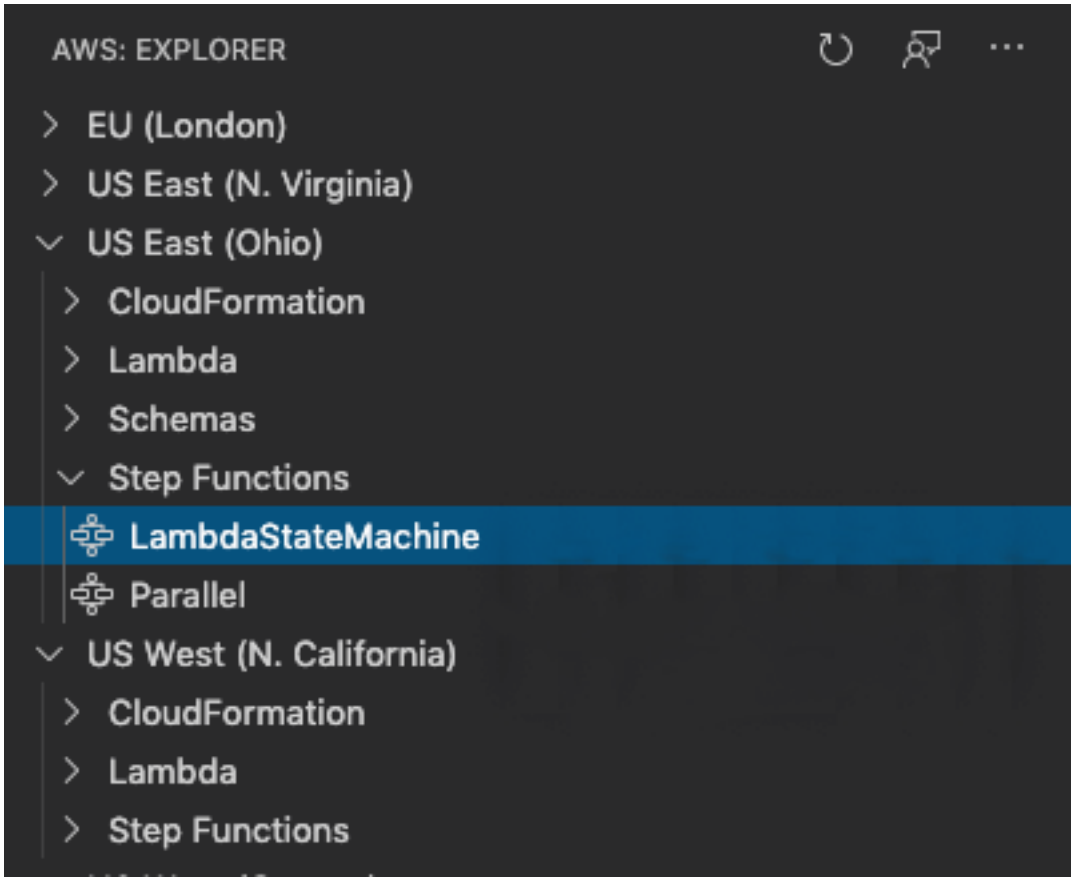
VS Code でのステートマシンの使用

VS Code を使用して、リモートステートマシンとやり取りし、JSON または YAML 形式でローカルでステートマシンを開発できます。ステートマシンの作成や更新、既存のステートマシンの一覧表示、実行およびダウンロードを行うことができます。VS Code を使用して、テンプレートから新しいステートマシンを作成したり、ステートマシンを可視化したり、コードスニペット、コード補完、コード検証を行ったりすることもできます。

既存のステートマシンの一覧表示

作成済みのステートマシンがある場合は、それらを一覧表示できます。

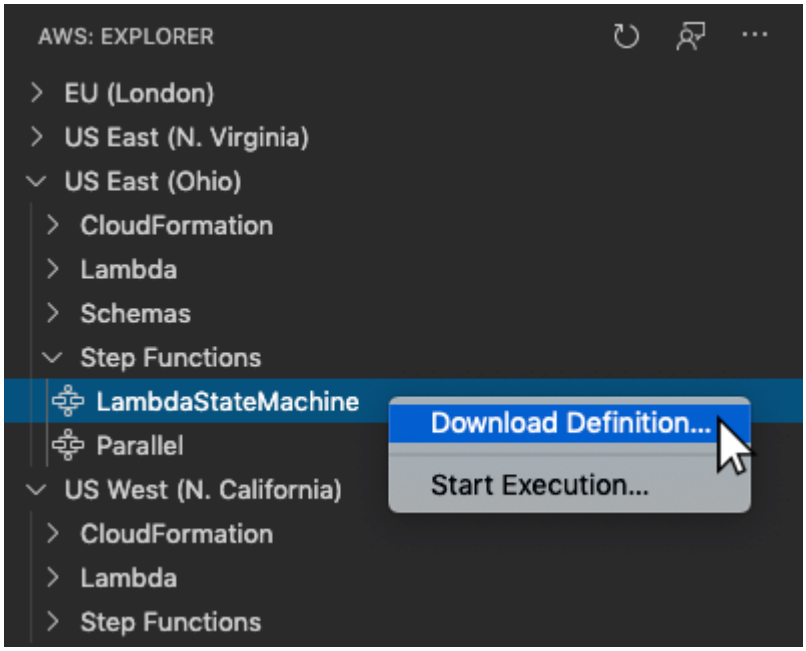
1. AWS Explorer を開きます。
2. Step Functions 選択
3. アカウント内のすべてのステートマシンが一覧表示されていることを確認します。



ステートマシンのダウンロード

ステートマシンをダウンロードするには

1. AWS Explorer で、ダウンロードするステートマシンを右クリックします。
2. [Download] を選択し、ステートマシンのダウンロード先を選択します。
3. 正常にダウンロードされたことを確認します。

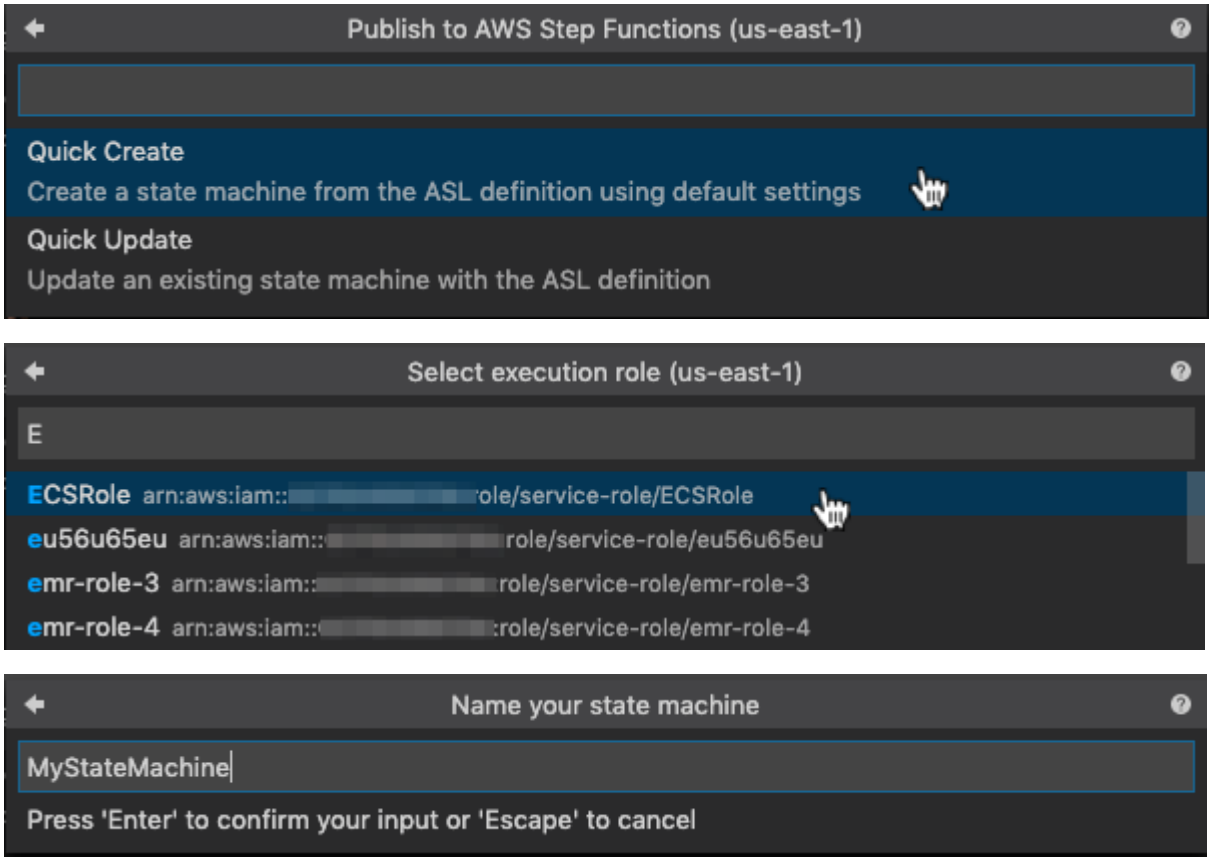


ステートマシンを作成する

新しいステートマシンは、自分で作成するか、テンプレートを使用して作成できます。テンプレートからステートマシンを作成する方法の詳細については、「ステートマシンのテンプレート」セクションを参照してください。新しいステートマシンを作成するには

1. ステートマシン定義を含む新しい Amazon ステート言語 (ASL) ファイルを作成します。右下のメニューを使用して、それを Amazon ステート言語として設定します。
2. [ステップ関数に公開] を選択します。

3. [クイック作成] を選択し、ロールを選択して、ステートマシンに名前を付けます。

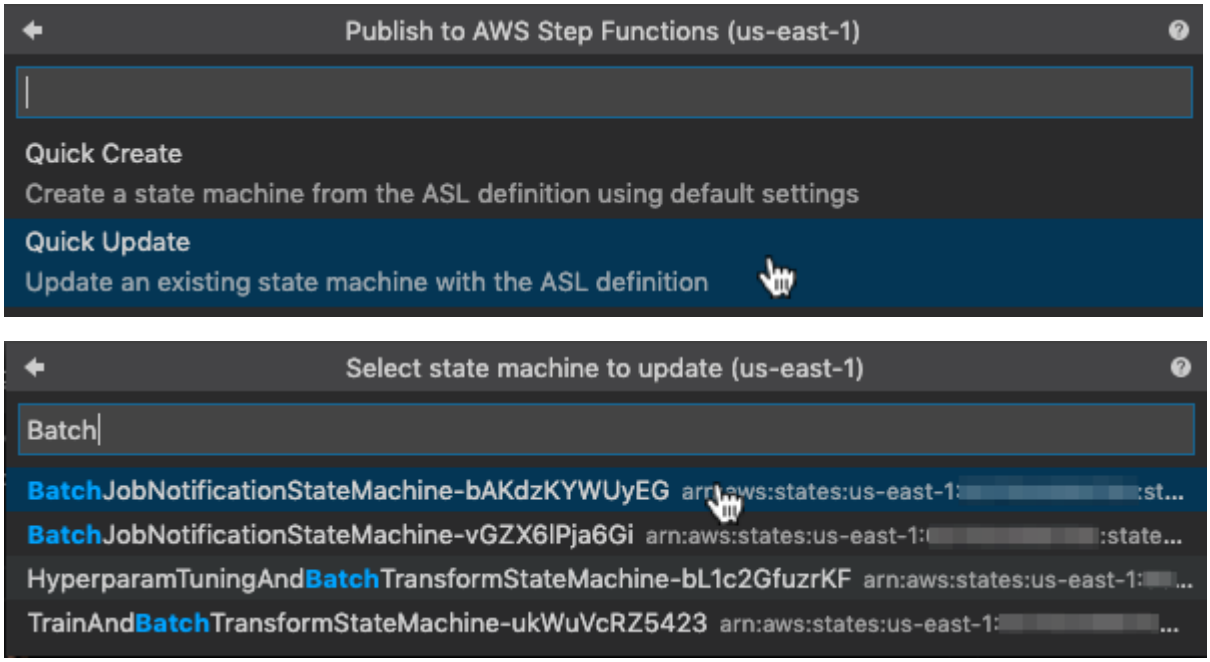


ステートマシンの更新

ステートマシンを更新するには

1. ステートマシン定義を含む ASL ファイルを編集します。
2. [ステップ関数に公開] を選択します。

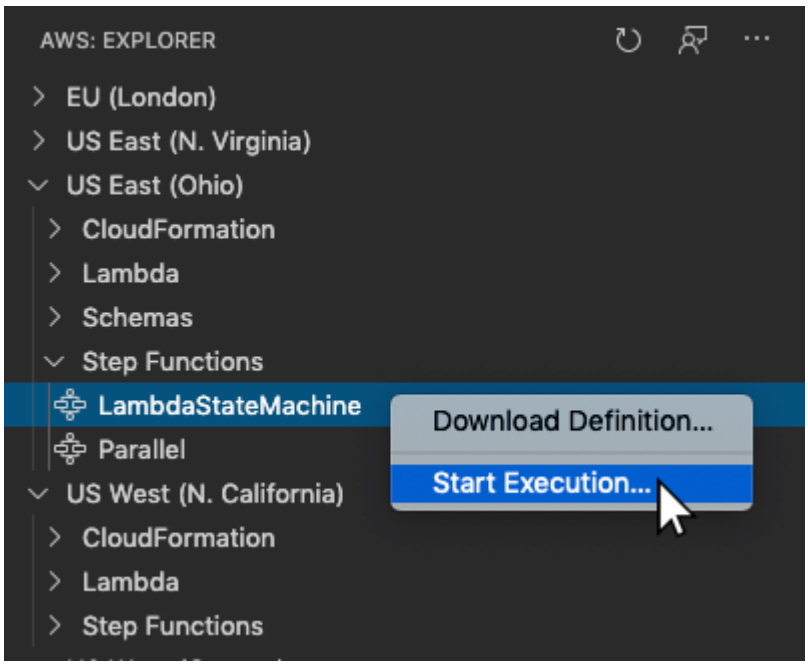
3. [Quick Update] を選択し、更新するステートマシンを選択します。

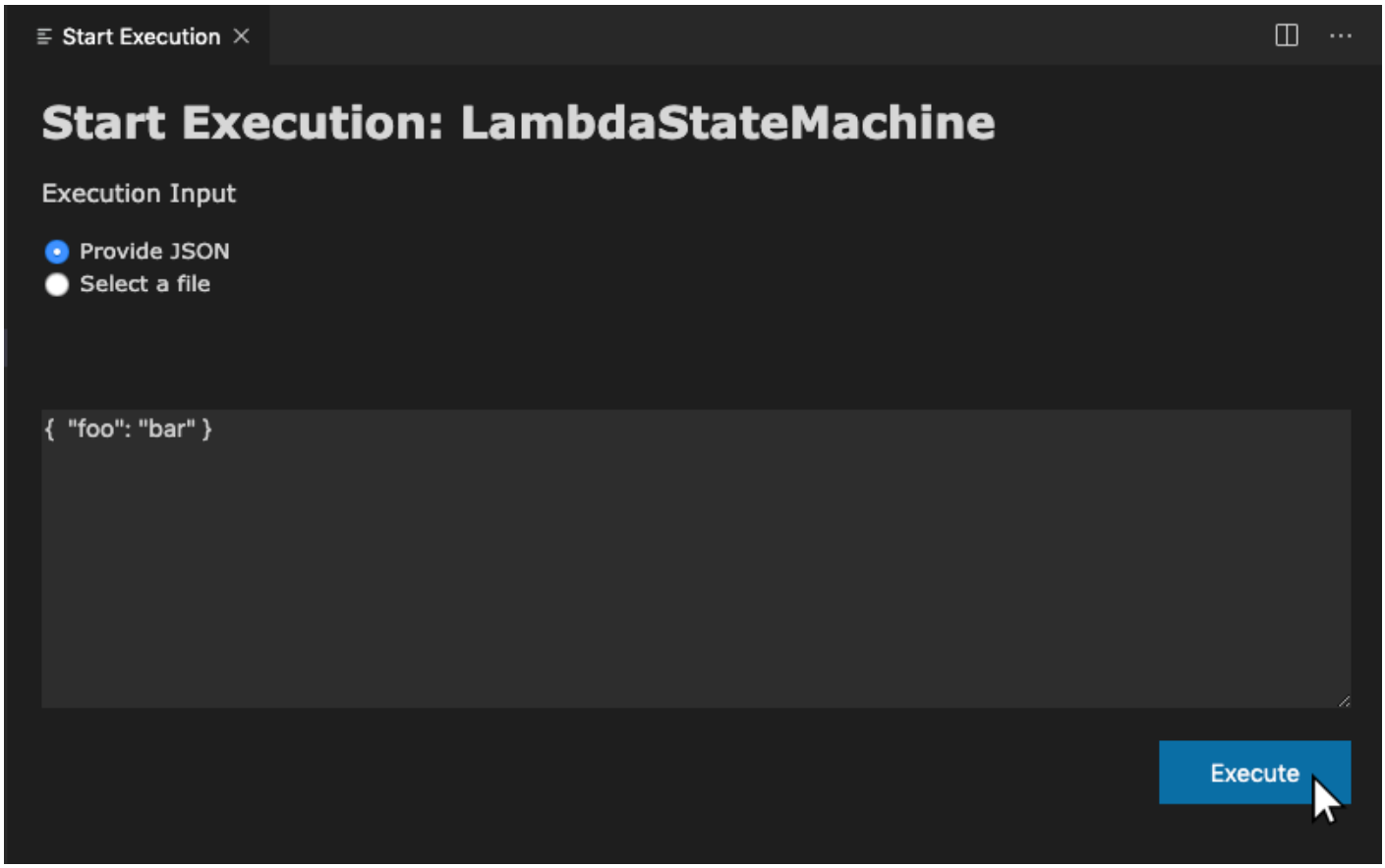


ステートマシンを実行します

ステートマシンを実行するには:

1. AWS Explorer で、実行するステートマシンを右クリックします。
2. ステートマシンの入力を指定します。ファイルからの入力と、テキストボックスへの入力の両方を試すことができます。
3. ステートマシンを開始し、正常に実行されていることを確認します。

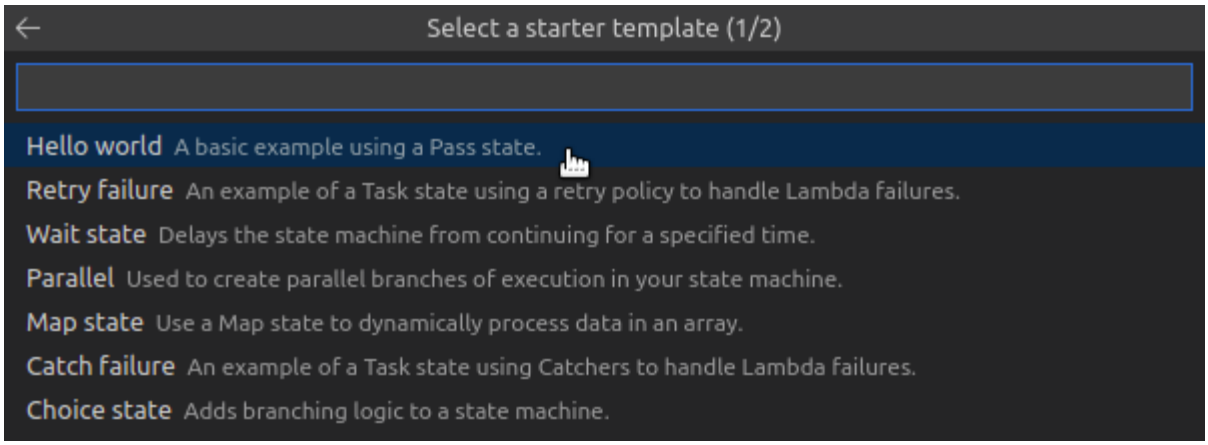

AWS: EXPLORER
EU (London)
US East (N. Virginia)
US East (Ohio)
CloudFormation
Lambda
Schemas
Step Functions
LambdaStateMachine
Parallel
US West (N. California)
CloudFormation
Lambda
Step Functions
Download Definition...
Start Execution...


Start Execution
Start Execution: LambdaStateMachine
Execution Input
Provide JSON
Select a file
{ "foo": "bar" }
Execute

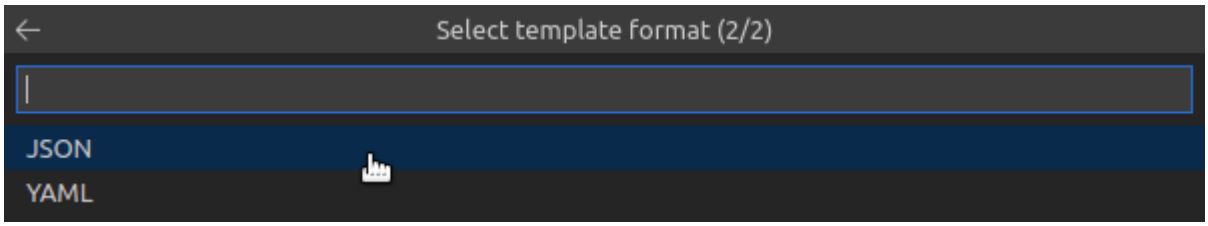
ステートマシンのテンプレートを記します

ステートマシンを作成する場合、テンプレートから作成することを選択できます。テンプレートには、よく使用されるいくつかのステートが設定されたサンプルのステートマシン定義が含まれており、ここから作成を開始できます。ステートマシンテンプレートを使用するには

1. VS Code で コマンドパレット を開きます。
2. [AWS Toolkit が新しいステップ関数ステートマシンを作成] を選択します。
3. 使用するテンプレートを選択します。



4. JSON または YAML テンプレート形式のどちらを使用するかを選択します。



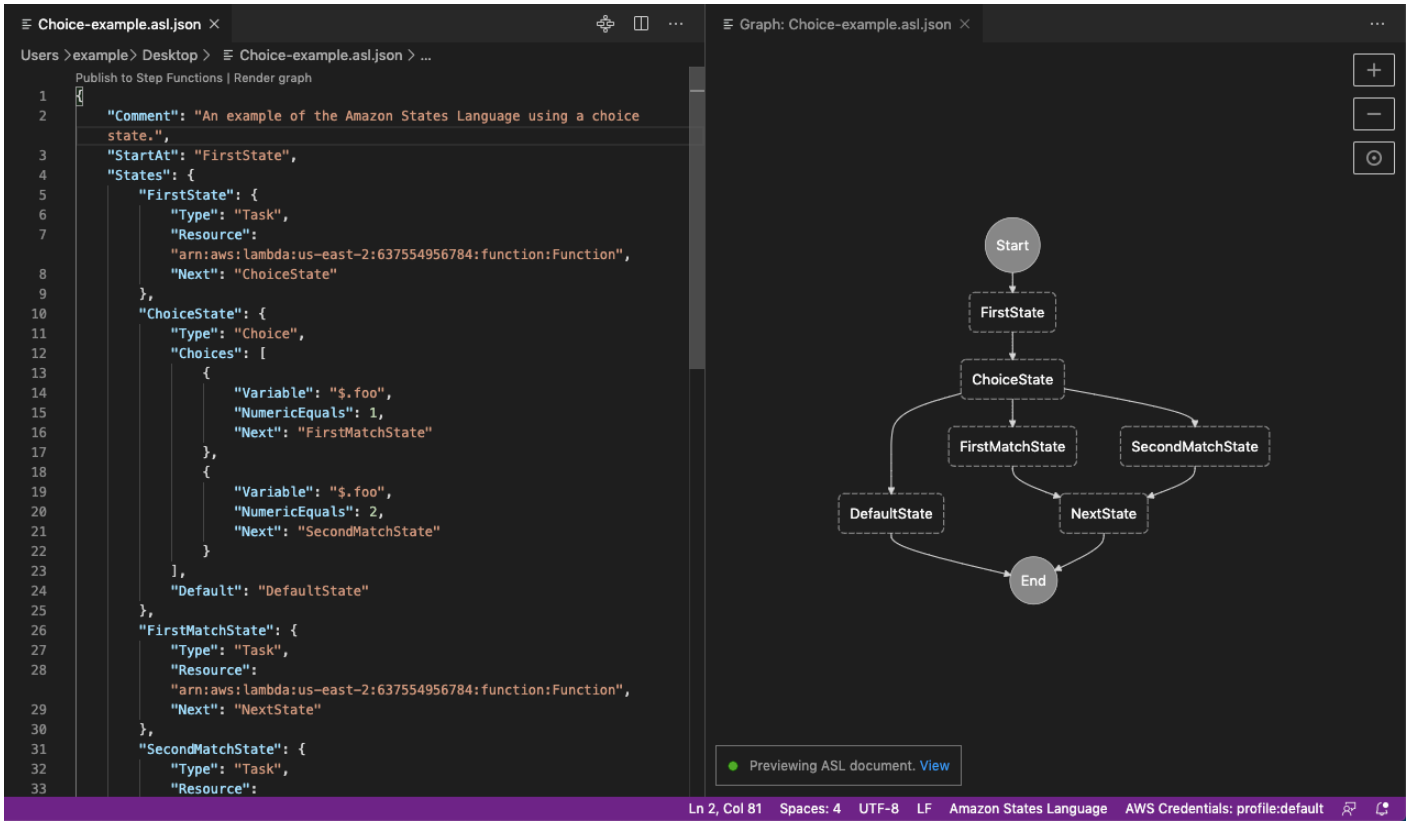
ステートマシンのグラフ可視化を記します

グラフ可視化を使用すると、ステートマシンの外観をグラフ形式で確認できます。グラフ可視化を作成すると、別のタブが開いて、ステートマシン JSON または YAML の可視化が表示されます。記述しているステートマシンの定義を、その可視化と並行して比較できます。ステートマシン定義を変更すると、可視化が更新されます。

> **Note**
>
> ステートマシン定義の可視化を作成するには、その定義をアクティブなエディタで開いておく必要があります。定義ファイルを閉じたり、名前を変更したりすると、可視化が閉じます。

ステートマシンのグラフ可視化を作成するには

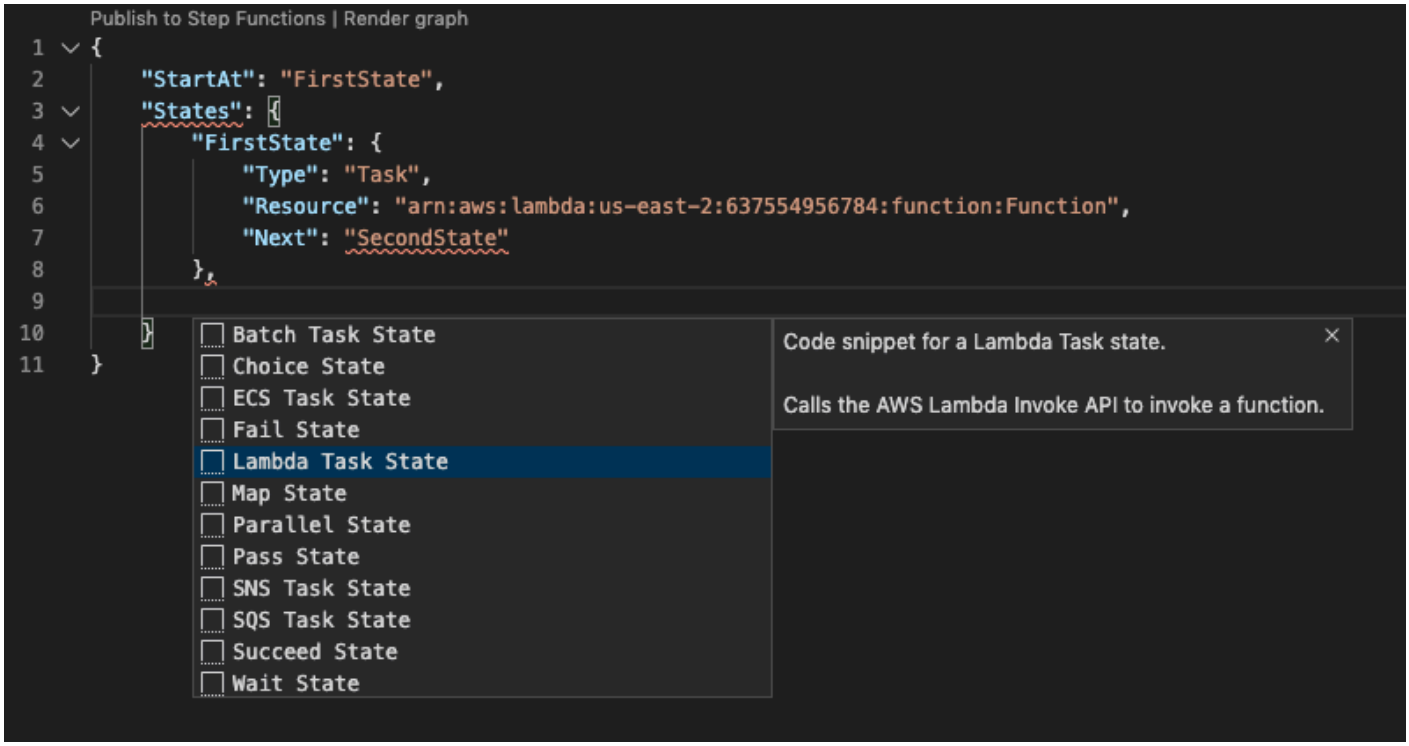
1. ステートマシンを定義します。
2. VS コードで コマンドパレット を開きます。
3. 可視化を作成するには、右上にある可視化ボタンを使用するか、[AWS グラフをレンダリング を選択します。



コードスニペット

コードスニペットを使用すると、コードの短いセクションを挿入できます。コードスニペットを使用するには

1. ファイルを開き、JSON 形式に .asl.json 拡張子を付けて、または YAML 形式に .asl.yaml 拡張子を付けて保存します。
2. States プロパティを使用して新しいステートマシンを作成します。
3. [States] 内にカーソルを置きます。
4. キーの組み合わせ Control + Space を使用して、必要なコードスニペットを選択します。
5. Tab を使用して、コードスニペットの変数とパラメータをスキャンします。
6. 関連するステート内にカーソルを置いて、Retry スニペットと Catch スニペットをテストします。

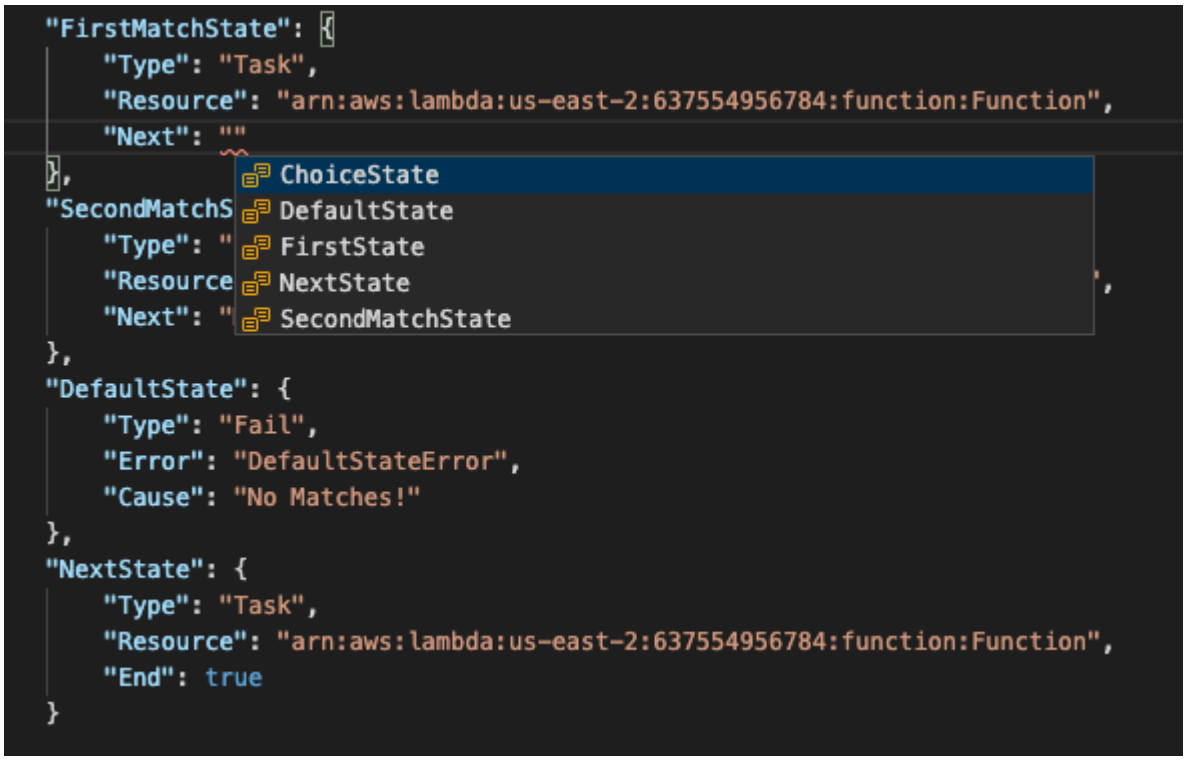


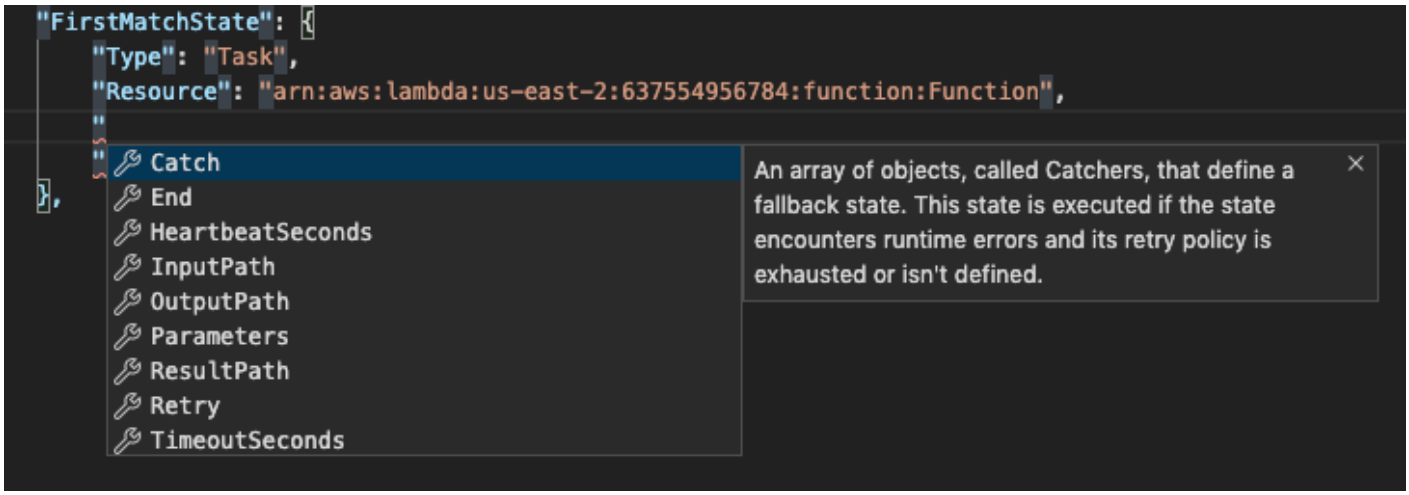
コードの補完と検証

コード補完の動作を確認するには

1. 複数のステートを作成します。
2. Next 、、StartAtまたは Default プロパティの後にカーソルを置きます。
3. キーの組み合わせ Control + Space を使用して、使用可能な補完を一覧表示します。追加のプロパティは、再度 Control + Space を使用してアクセスすることができ、State の Type に基づいて行われます。
4. 作業を進めるに従って、以下に関するコード検証が実行されます。

- 不足しているプロパティ
- 不正な値
- 存在しない最終ステート
- 存在しない参照先ステート





Threat Composer の使用

を使用して AWS Toolkit for Visual Studio Code 、Threat Composer ツールを操作できます。Threat Composer は、脅威モデリングプロセスを簡素化できる脅威モデリングツールです。

Threat Composer ツールの詳細については、「Threat Composer GitHub リポジトリ」を参照してください。

以下のトピックでは、 で Threat Composer を使用する方法について説明します AWS Toolkit for Visual Studio Code。

トピック

- Toolkit からの Threat Composer の使用

Toolkit からの Threat Composer の使用

Threat Composer を使用すると、Threat Composer の脅威モデルを VS Code で直接作成、表示、編集できます。Threat Composer ツールの詳細については、「Threat Composer GitHub リポジトリ」を参照してください。

以下のセクションでは、 で Threat Composer ツールにアクセスする方法について説明します AWS Toolkit for Visual Studio Code。

Toolkit から Threat Composer にアクセスする

Toolkit から Threat Composer にアクセスするには、主に 3 つの方法があります。

既存の脅威モデルを介した Threat Composer へのアクセス

Threat Composer を開くには、VS Code で既存の脅威モデルファイル (拡張子 `.tc.json`) を開きます。Threat Composer は、VS Code エディタウィンドウで脅威モデルファイルの視覚化を自動的に開き、レンダリングします。

新しい Threat Composer 脅威モデルの作成

1. VS Code のメインメニューから、ファイル を展開し、新しいファイル を選択します。
2. 「新しいファイル」ダイアログで、「脅威コンポーザーファイル」を選択します。
3. プロンプトが表示されたら、 を入力し`file name`、 **`enter`**キーを押して Threat Composer を開き、新しい VS Code エディタウィンドウで空の脅威モデルファイルの視覚化を作成します。

コマンドパレットからの新しい Threat Composer 脅威モデルの作成

1. VS Code から、 **`Cmd + Shift + P`**または **`Ctrl + Shift + P`** (Windows) を押してコマンドパレットを開きます。

2. 検索フィールドに「Create New Threat Composer File when it populates in the results**Threat Composer**」と入力し、選択します。
3. プロンプトが表示されたら、 を入力し`file name`、 **enter**キーを押して Threat Composer を開き、新しい VS Code エディタウィンドウで空の脅威モデルファイルの視覚化を作成します。

リソースの使用

AWS Explorer にデフォルトで表示されている AWS サービスへのアクセスに加えて、[Resources] (リソース) を選択して、数百ものリソースから選択してインターフェイスに追加することもできます。AWS では、リソースはユーザーが操作できるエンティティです。追加できるリソースには、Amazon AppFlow、Amazon Kinesis Data Streams、AWS IAM ロール、Amazon VPC、および Amazon CloudFront ディストリビューションなどがあります。

選択したら、リソース に進み、リソースタイプを展開して、そのタイプで使用可能なリソースを一覧表示します。例えば、`AWS Toolkit:Lambda::Function` リソースタイプを選択すると、さまざまな関数、そのプロパティ、および属性を定義するリソースにアクセスできます。

リソースタイプを [Resources] (リソース) に追加すると、次の方法でリソースタイプとそのリソースを操作できます。

- このリソースタイプについて、現在の AWS リージョンで使用可能な既存のリソースを一覧表示します。
- リソースを記述する JSON ファイルの読み取り専用バージョンを表示します。
- リソースのリソース識別子をコピーします。
- リソースタイプの目的およびリソースをモデリングするためのスキーマ (JSON および YAML 形式) について説明する AWS のドキュメントを表示します。
- スキーマに準拠する JSON 形式のテンプレートを編集して保存して、新しいリソースを作成します。*
- 既存のリソースを更新または削除します。*

> **Important**
>
> * 現在の AWS Toolkit for Visual Studio Code では、リソースの作成、編集、削除のオプションは実験的な機能です。実験的な機能は引き続きテストおよび更新されるため、ユーザビリティに問題がある可能性があります。実験的な特徴は、AWS Toolkit for Visual Studio Code から予告なしに削除されるかもしれません。

リソースに実験的な機能を使用できるようにするには、VS Code IDE の [設定] ペインを開き、[拡張機能] を展開して [AWSツールキット] を選択します。
[AWS: Toolkit Experiments] の下から [JSONResourceModification] を選択して、リソースを作成、更新、削除できるようにします。
詳細については、「実験的な機能の使用」を参照してください。

リソースにアクセスするための IAM アクセス許可

AWS サービスに関連付けられたリソースにアクセスするには、特定の AWS Identity and Access Management アクセス許可が必要です。例えば、IAM エンティティ (ユーザーまたはロールなど) は、`AWS Toolkit:Lambda::Function` リソースにアクセスするために Lambda アクセス許可を必要とします。

サービスリソースへのアクセス許可だけでなく、IAM エンティティには自身に代わって Toolkit for VS Code が AWS クラウドコントロール API オペレーションを呼び出すためのアクセス許可が必要です。Cloud Control API オペレーションでは、IAM ユーザーまたはロールがリモートリソースにアクセスして更新できます。

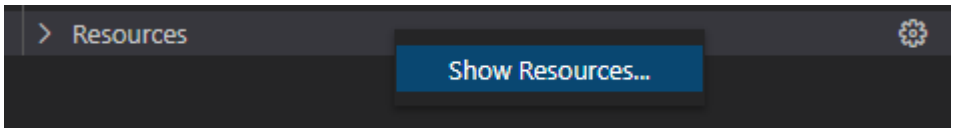
アクセス許可を付与する場合、ツールキットインターフェイスを使用して API オペレーションを呼び出す IAM エンティティに AWS マネージドポリシー (PowerUserAccess) をアタッチするのが最も簡単な方法です。このマネージドポリシーでは、API オペレーションの呼び出しなど、アプリケーション開発タスクを実行するために一定の範囲のアクセス許可が付与されます。

リモートリソースで使用可能な API オペレーションを定義する特定のアクセス許可については、「AWS クラウドコントロール API ユーザーガイド」を参照してください。

既存のリソースの追加と操作

1. [AWSエクスプローラ] をクリックして、リソース 右クリックして [リソースを表示] を選択します。

   ペインには、選択可能なリソースタイプのリストが表示されます。

2. 選択ペインで、[AWS Explorer] に追加するリソースタイプを選択し、[戻る] をクリックするかまたは [OK] を選択して確認します。

選択したリソースタイプは、[リソース] に表示されます

> **Note**
>
> リソースタイプを [AWS エクスプローラー] にすでに追加済みの場合は、そのタイプのチェックボックスをオフにすると、[OK] をクリックした後 [リソース] の下のリストに表示されなくなります。現在選択されているリソースタイプのみが、[AWS エクスプローラー] に表示されます。

3. リソースタイプに既に存在するリソースを表示するには、そのタイプのエントリを展開します。

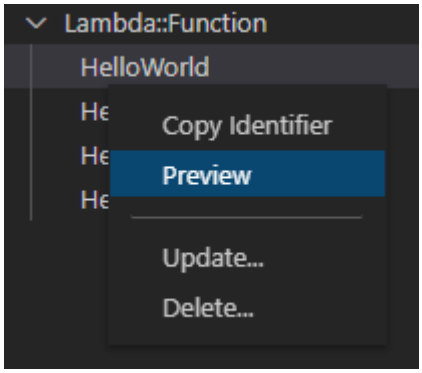
   使用可能なリソースのリストが、リソースタイプの下に表示されます。

4. 特定のリソースを操作するには、リソースの名前を右クリックし、次のいずれかを選択します。

   - リソース識別子をコピー: 特定のリソースの識別子をクリップボードにコピーします。(例えば、`AWS Toolkit:DynamoDB::Table` リソースは `TableName` プロパティを使用して識別できます。)
   - [Preview] (プレビュー): リソースを記述する JSON 形式のテンプレートの読み取り専用バージョンを表示します。

     リソーステンプレートが表示されたら、エディタタブの右側にある [更新] アイコンを選択して修正できます。リソースを更新するには、??? を有効にする必要があります。
   - 更新: VS Code エディタでリソースの JSON 形式のテンプレートを編集します。詳細については、「リソースの作成と編集」を参照してください。
   - 削除: 表示されたダイアログボックスで削除を確認して、リソースを削除します。(リソースの削除は、AWS Toolkit for Visual Studio Code のこのバージョンでは現在 ??? です。)

   > **Warning**
   >
   > リソースを削除すると、そのリソースを使用する AWS CloudFormation スタックは更新に失敗します。この更新の失敗を修正するには、リソースを再作成するか、スタックの AWS CloudFormation テンプレートテンプレートにリソースへのリファレンスを削除する必要があります。詳細情報は、この ナレッジセンターの記事 を参照してください。



リソースの作成と編集

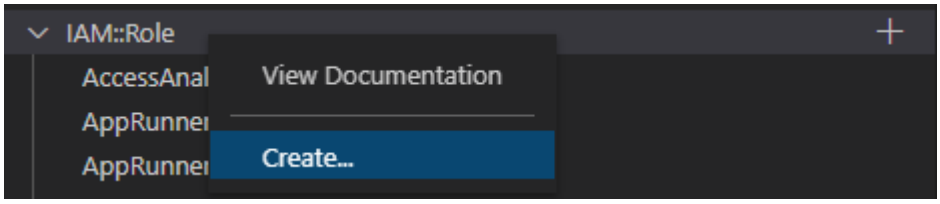
> **Important**
>
> リソースの作成と更新は、このバージョンの AWS Toolkit for Visual Studio Code では現在 ??? となっています。

新しいリソースを作成するには、[リソース] リストにリソースタイプを追加し、リソース、そのプロパティ、および属性を定義する JSON 形式のテンプレートを編集します。

例えば、`AWS Toolkit:SageMaker::UserProfile` リソースタイプに属しているリソースは、Amazon SageMaker Studio のユーザープロファイルを作成するテンプレートで定義されます。このユーザープロファイルリソースを定義するテンプレートは、`AWS Toolkit:SageMaker::UserProfile` のリソースタイプスキーマに準拠している必要があります。例えば、プロパティが見つからないか正しくないためにテンプレートがスキーマに準拠していない場合は、リソースを作成または更新することはできません。

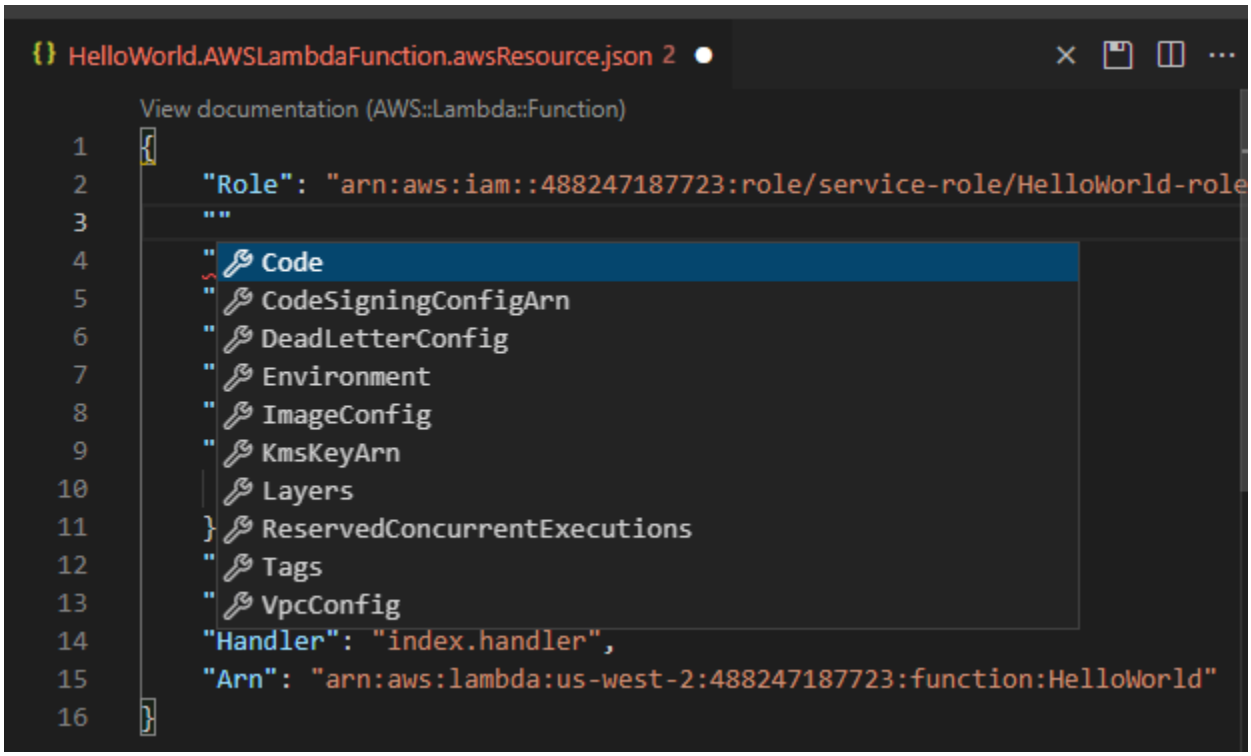
1. 右クリックして、作成するリソースのリソースタイプを追加し、[リソース] をクリックし、[リソースを表示] を選択します。
2. [リソース] の下でリソースタイプを追加した後、プラス (「+」) アイコンを選択して、新しいエディタでテンプレートファイルを開きます。

   また、リソースタイプの名前を右クリックして、[作成] をクリックしてください。[ドキュメントの表示] を選択して、リソースのモデル化方法に関する情報にアクセスすることもできます。



3. エディタで、リソーステンプレートを構成するプロパティの定義を開始します。オートコンプリート機能では、テンプレートのスキーマに適合するプロパティ名が提案されます。プロパティタイプにカーソルを合わせると、そのプロパティの使用目的の説明がペインに表示されます。スキーマの詳細については、[ドキュメントの表示] を選択してください。

   リソーススキーマに適合しないテキストは、波状の赤い下線で示されます。



4. リソースの宣言が完了したら、[保存] アイコンを使用して、テンプレートを検証し、AWS クラウドにリソースを保存します。

   テンプレートがスキーマに従ってリソースを定義している場合は、リソースが作成されたことを確認するメッセージが表示されます。(リソースが既に存在する場合は、リソースが更新されたことを確認するメッセージが表示されます。)

   リソースが作成されると、リソースタイプの見出しの下のリストに追加されます。

5. ファイルにエラーが含まれている場合は、リソースを作成または更新できなかったことを示すメッセージが表示されます。[ログを表示する] を選択して、修正する必要があるテンプレート要素を特定します。

AWS Toolkit for Visual Studio Code のセキュリティ

トピック

AWS Toolkit for Visual Studio Code でのデータ保護

AWS 責任共有モデルは、AWS Toolkit for Visual Studio Code のデータ保護に適用されます。このモデルで説明されているように、AWS は、AWS クラウド のすべてを実行するグローバルインフラストラクチャを保護する責任を負います。顧客は、このインフラストラクチャでホストされているコンテンツに対する管理を維持する責任があります。このコンテンツには、使用される AWS のサービスのセキュリティ構成と管理タスクが含まれます。データプライバシーの詳細については、「データプライバシーのよくある質問」を参照してください。欧州でのデータ保護の詳細については、AWS セキュリティブログに投稿された「AWS 責任共有モデルおよび GDPR」のブログ記事を参照してください。

データを保護するため、AWS アカウント の認証情報を保護し、AWS IAM Identity Center または AWS Identity and Access Management (IAM) を使用して個々のユーザーをセットアップすることをお勧めします。この方法により、それぞれのジョブを遂行するために必要なアクセス許可のみを各ユーザーに付与できます。また、次の方法でデータを保護することをお勧めします。

- 各アカウントで多要素認証 (MFA) を使用します。
- SSL/TLS を使用して AWS リソースと通信します。TLS 1.2 および TLS 1.3 をお勧めします。
- AWS CloudTrail で API とユーザーアクティビティログをセットアップします。
- AWS のサービス 内でデフォルトである、すべてのセキュリティ管理に加え、AWS の暗号化ソリューションを使用します。
- Amazon Macie などの高度なマネージドセキュリティサービスを使用します。これらは、Amazon S3 に保存されている機密データの検出と保護を支援します。
- コマンドラインインターフェイスまたは API を使用して AWS にアクセスするときに FIPS 140-2 検証済みの暗号化モジュールが必要な場合は、FIPS エンドポイントを使用します。利用可能な FIPS エンドポイントの詳細については、連邦情報処理規格 (FIPS) 140-2 を参照してください。

お客様の E メールアドレスなどの機密情報やセンシティブ情報は、タグや [Name] フィールドなどの自由形式のフィールドに配置しないことを強くお勧めします。これには、コンソール、API、AWS

CLI、AWS SDK を使用して AWS Toolkit for Visual Studio Code または他の AWS のサービス を操作する場合が含まれます。名前に使用する自由記述のテキストフィールドやタグに入力したデータは、課金や診断ログに使用される場合があります。外部サーバーに URL を提供する場合、そのサーバーへのリクエストを検証できるように、認証情報を URL に含めないことを強くお勧めします。

AWS Toolkit for Visual Studio Code ユーザーガイドのドキュメント履歴

ドキュメントの主要な最終更新日: 2021 年 10 月 14 日

次の表に、「AWS Toolkit for Visual Studio Code」の各リリースにおける重要な変更点を示します。このドキュメントの更新に関する通知については、RSS フィードを購読してください。

変更	説明	日付
作成済みユーザーガイド: CloudFormation テンプレートの作成	Toolkit for VS Code を使用して CloudFormation テンプレートを作成する方法を説明する新しいユーザーガイドを作成しました	2021 年 12 月 17 日
マイナーなUIの更新	UI との整合性を高めるため、「マシン状態のプレビュー」の既存のテキストを「グラフをレンダリング」に更新しました。	2021 年 12 月 14 日
Amazon Elastic Container Service Exec のユーザーガイドを作成しました	これは Amazon ECS Exec の概要です。	2021 年 12 月 13 日
VS Code Service AWS IoT Toolkit のユーザーガイドを作成しました	このユーザーガイドは、Toolkit for VS Code の AWS IoT サービスを使用して開始することをサポートする目的で作成されました。	2021 年 11 月 22 日
実験的機能のサポート	AWS サービスの実験的機能をオンにするサポートが追加されました。	2021 年 10 月 14 日

AWS リソースのサポート	リソースを作成、編集、および削除するインターフェイスオプションと、リソースタイプにアクセスするためのサポートが追加されました。	2021 年 10 月 14 日
AWS Toolkit for Visual Studio Code の Amazon ECR サービスの概要	VS Code でアクセス可能な Amazon ECR サービスの特徴と機能の概要とウォークスルーを追加しました。	2021 年 10 月 14 日
ARM64 環境のサポート	arm64 ベースのエミュレート環境と x86_64 ベースの環境で、サーバーレスアプリケーションを実行できるようになりました。	2021 年 10 月 1 日
AWS サーバーレスアプリケーション	AWS SAM ARM64 プラットフォーム上のアプリケーションを実行するサポートが追加されました。	2021 年 9 月 30 日
フォーマットアップデート Node.js セクション	お客様からのフィードバックに応じて、Node.js/TypeScript のフォーマットを更新しました。	2021 年 8 月 12 日
App Runner サポート	AWS App Runner のサポートが AWS Toolkit for Visual Studio Code に追加されました。	2021 年 8 月 11 日
Go 関数のデバッグ	ローカル Go 関数のデバッグのサポートが追加されました。	2021 年 5 月 10 日

Java 関数のデバッグ	ローカル Java 関数のデバッグのサポートが追加されました。	2021 年 4 月 22 日
AWS Step Functions のYAML サポート	AWS Step Functions の YAML のサポートが追加されました。	2021 年 3 月 4 日
Amazon API Gateway リソースのデバッグ	ローカル Amazon API Gateway リソースのデバッグのサポートが追加されました。	2020 年 12 月 1 日
Amazon API Gateway	Amazon API Gateway のサポートの追加。	2020 年 12 月 1 日
AWS サーバーレスアプリケーション	Lambda コンテナイメージのサーバーレスアプリケーションでのサポートが追加されました。	2020 年 12 月 1 日
AWS Systems Manager のサポート	Systems Manager Automation ドキュメントのサポートが追加されました。	2020 年 9 月 30 日
CloudWatch Logs	CloudWatch Logs のサポートを追加	2020 年 8 月 24 日
Amazon S3	Amazon S3 の追加されたサポート	2020 年 7 月 30 日
AWS Step Functions のサポート	AWS Step Functions のサポートが追加されました。	2020 年 3 月 31 日
セキュリティコンテンツ	セキュリティコンテンツを追加しました。	2020 年 2 月 6 日
Amazon EventBridge スキーマの使用	Amazon EventBridge スキーマのサポートが追加されました	2019年12月1日

AWS CDK	AWS CDKサービスのプレビューリリース。	2019 年 11 月 25 日
外部認証情報プロセスの使用	外部認証情報プロセスによる AWS 認証情報の取得に関する情報を追加しました。	2019 年 9 月 25 日
タスク定義ファイルでの IntelliSense の使用	Amazon ECS タスク定義ファイルを使用するための IntelliSense のサポートが追加されました。	2019 年 9 月 24 日
AWS Toolkit for Visual Studio Code 用ユーザーガイド	一般的な可用性のリリース	2019 年 7 月 11 日
AWS Toolkit for Visual Studio Code 用ユーザーガイド	わかりやすさと使いやすさのためにドキュメントの構造を更新しました。	2019 年 7 月 3 日
AWS Toolkit for Visual Studio Code のインストール	さまざまなツールチェーンをサポートするための言語 SDK のインストールに関する情報を追加しました。	2019 年 6 月 12 日
ツールチェーンを設定する	さまざまなツールチェーンの設定に関する情報を追加しました。	2019 年 6 月 12 日
初回リリース	AWS Toolkit for Visual Studio Code ユーザーガイドの初回リリース。	2019 年 3 月 28 日

翻訳は機械翻訳により提供されています。提供された翻訳内容と英語版の間で齟齬、不一致または矛盾がある場合、英語版が優先します。