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。

AWS Toolkit for Visual Studio Code とは

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

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

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 Developer と Amazon CodeWhisperer

2024 年 4 月 30 日現在、Amazon CodeWhisperer は Amazon Q Developer の一部となりました。これにはインラインコードの提案と Amazon Q Developer の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 Developer サービスの詳細については、「Amazon Q Developer 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 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。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 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。
- 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。

の使用開始AWS Toolkit for Visual Studio Code

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 リージョンを変更する
- 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 では、VS Code バージョン 1.73.0 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。

- Amazon Web Services (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 [サインアップを完了する] (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、AWS Serverless Application Model などの AWS SAM (新しいサーバーレ

スアプリケーションの作成 (ローカル)) 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。アカウントは AWS を使用する必要はありませんが AWS Toolkit for Visual Studio Code、Toolkit 関数は接続なしで制限されます。

以前に別の AWS サービス (など AWS Command Line Interface) を介して AWS アカウントと認証をセットアップ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からアカウントに AWS 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、AWS 「 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。
2. 認証の設定: 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 からの による認証: 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

[AWS ツールキットへのサインイン] パネルを開くには、以下のいずれかの手順を完了してください。

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 CodeEXPLORER 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...] メニューで、[AWSへの接続] を選択して [AWS ツールキットへのサインイン] パネルを開きます。

VS Code コマンドパレットを使用して [AWS ツールキットへのサインイン] パネルを開くには

1. **Shift+Command+P** (Windows は **Ctrl+Shift+P**) 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 [AWS ツールキットへのサインイン] 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 IAM アイデンティティセンター、次の手順を実行します。

> **Note**
>
> AWS ビルダー ID または IAM アイデンティティセンターによる認証では、デフォルトのウェブブラウザで AWS 認可ポータルが起動します。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。

IAM Identity Center AWS を認証して接続する

1. [AWS ツールキットへのサインイン] パネルから、[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 アイデンティティセンターでサインイン] パネルから、組織の開始 URL を入力します。この URL は、自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 your AWS Region 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 承認リクエストのウェブサイトを開くことに同意します。
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。認可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 を認証して接続するには、次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 ツールキットへのサインイン] パネルから、[IAM 認証情報] タブを選択し、[続行] ボタンを選択して続行します。
2. 指定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。
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 CodeCatalyst の認証

Toolkit から CodeCatalyst の使用を開始するには、 AWS ビルダー ID または IAM アイデンティティセンターの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 を使用した認証と接続

1. [AWS ツールキットへのサインイン] パネルから、[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 でのサインイン] パネルの上部で、[サインインにスキップ] 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。認可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 ツールキットへのサインイン] パネルから、[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 アイデンティティセンターでサインイン] パネルから、組織の開始 URL を入力します。この URL は、自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 your AWS Region 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 承認リクエストのウェブサイトを開くことに同意します。
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。認可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 リージョンを変更する

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。デフォルトの AWS リージョンは、AWS Toolkit for Visual Studio Code から AWS アカウントに接続すると検出され、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 エクスプローラーにリージョンを追加する

AWS エクスプローラーにリージョンを追加するには、次のステップを実行します。

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. コマンドパレットから **AWS: Show or Hide Regions** を検索し、[AWS: Show or Hide Regions] を選択して、使用可能なリージョンのリストを表示します。
3. リストから、AWS エクスプローラーに追加する 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 エクスプローラーを更新します。

AWS Explorer からリージョンを非表示にする

AWSExplorer ビューでリージョンを非表示にするには、次の手順を実行します。

1. AWSExplorer から、非表示にしたいAWSリージョンを検索します。
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. [リージョンの表示または非表示] を選択して、VS Code の [AWS: Show or Hide Regions] オプションを開きます。
4. AWS エクスプローラー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、`virtualenv` を実行して `python -m venv ./.venv` 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。

の認証とアクセス AWS Toolkit for Visual Studio Code

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 アイデンティティセンターまたは 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、各認証情報タイプに関する詳細を説明します。

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「 への接続 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
- 開発者用 AWS Builder ID
- 外部認証情報プロセスの使用
- アクセスを許可するファイアウォールとゲートウェイの更新

AWS IAM アイデンティティセンター

AWS IAM アイデンティティセンター 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「Connect to 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 アカウントとの IAM 認証情報認証。

AWS ツールキットを認証して既存の 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「Connect to AWS 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ように 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) を介して IAM 認証情報をすで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 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、専用ソフトウェアを開発するとき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 アイデンティティセンター 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: 長期認証情報を使用してアクセスキーを取得する」トピックの「認証」を完了する必要があります。 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”にある 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、 は以下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”にある「」で説明credentials fileされているように、 を作成するプロセスを AWS Toolkit for Visual Studio Code 順を追って説明します。

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ストアの設定と AWS アカウントの認証情報。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。

を使用して共有認証情報ファイルを作成する AWS Toolkit for Visual Studio Code

1. **Shift+Command+P** (Windows は **Ctrl+Shift+P**) 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 [AWS ツールキットへのサインイン] パネルを開きます。
4. [AWS ツールキットへのサインイン] パネルから、[IAM 認証情報] タブを選択し、[続行] ボタンを選択して続行します。
5. 指定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
```

開発者用 AWS Builder ID

AWS Builder ID は、オプションまたは特定の AWS サービスに必須となる AWS アカウントです。AWS ビルダー ID 認証方法の詳細については、『AWS サインインユーザーガイド』の「AWS ビルダー ID によるサインイン」を参照してください。

AWS ツールキットを認証して既存の AWS ビルダー ID に接続する方法の詳細については、本ユーザーガイドの「AWS への接続 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

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 AWS Toolkit for Visual Studio Code 用に を設定できますshared config file。

shared config file 認証情報プロセスの の変更は、 AWS Toolkit for Visual Studio Code と の両方で同じです AWS Command Line Interface。外部認証情報の設定方法の詳細については、AWS Command Line Interface CC 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。

アクセスを許可するファイアウォールとゲートウェイの更新

ウェブコンテンツフィルタリングソリューションを使用して特定の AWS ドメインまたは URL エンドポイントへのアクセスをフィルタリングする場合、 AWS Toolkit for Visual Studio Code および Amazon Q で利用可能なすべてのサービスおよび機能にアクセスするには、次のエンドポイントを許可リストする必要があります。

AWS Toolkit for Visual Studio Code エンドポイント

以下は、許可を一覧表示する必要がある AWS Toolkit for Visual Studio Code 特定のエンドポイントとリファレンスのリストです。

Endpoint

```
https://idetoolkits.amazonwebservices.com/endpoints.json
```

ホストされたファイル

```
https://idetoolkits-hostedfiles.amazonaws.com/Notifications/VSCode/startup/1.x.json
https://idetoolkits-hostedfiles.amazonaws.com/Notifications/VSCode/emergency/1.x.json
```

スキーマのサポート

```
https://raw.githubusercontent.com/aws/serverless-application-model/main/samtranslator/
schema/schema.json
https://api.github.com/repos/devfile/api/releases/latest
https://raw.githubusercontent.com/devfile/api/${devfileSchemaVersion}/schemas/latest/
devfile.json
```

cSharpSamDebug インストールスクリプト

```
https://aka.ms/getvsdbgps1
https://aka.ms/getvsdbgsh
```

Amazon Q プラグインエンドポイント

以下は、許可リストに追加する必要がある Amazon Q プラグイン固有のエンドポイントとリファレンスのリストです。

```
https://idetoolkits-hostedfiles.amazonaws.com/*    (Plugin for configs)
https://idetoolkits.amazonwebservices.com/*   (Plugin for endpoints)
https://aws-toolkit-language-servers.amazonaws.com/*  (Language Server Process)
https://client-telemetry.us-east-1.amazonaws.com/ (Telemetry)
https://cognito-identity.us-east-1.amazonaws.com    (Telemetry)
https://aws-language-servers.us-east-1.amazonaws.com (Language Server Process)
```

Amazon Q Developer エンドポイント

以下は、許可リストに追加する必要がある Amazon Q Developer 固有のエンドポイントとリファレンスのリストです。

```
https://codewhisperer.us-east-1.amazonaws.com (Inline,Chat, QSDA,...)
https://q.us-east-1.amazonaws.com (Inline,Chat, QSDA....)
https://desktop-release.codewhisperer.us-east-1.amazonaws.com/ (Download url for CLI.)
https://specs.q.us-east-1.amazonaws.com (Url for autocomplete specs used by CLI)
* aws-language-servers.us-east-1.amazonaws.com (Local Workspace context)
```

Amazon Q コード変換エンドポイント

以下は、許可リストに追加する必要がある Amazon Q Code Transform 固有のエンドポイントとリファレンスのリストです。

```
https://docs.aws.amazon.com/amazonq/latest/qdeveloper-ug/security_iam_manage-access-
with-policies.html
```

認証エンドポイント

以下は、許可リストに追加する必要がある認証エンドポイントとリファレンスのリストです。

```
[Directory ID or alias].awsapps.com
* oidc.[Region].amazonaws.com
*.sso.[Region].amazonaws.com
*.sso-portal.[Region].amazonaws.com
*.aws.dev
*.awsstatic.com
*.console.aws.a2z.com
*.sso.amazonaws.com
```

アイデンティティエンドポイント

次のリストには、 AWS IAM アイデンティティセンター や AWS Builder ID など、アイデンティティに固有のエンドポイントが含まれています。

AWS IAM アイデンティティセンター

IAM アイデンティティセンターに必要なエンドポイントの詳細については、「AWS IAM アイデンティティセンター ユーザーガイド」の「IAM アイデンティティセンターを有効にする」トピックを参照してください。

エンタープライズ IAM アイデンティティセンター

```
https://[Center director id].awsapps.com/start (should be permitted to initiate auth)
https://us-east-1.signin.aws (for facilitating authentication, assuming IAM Identity
 Center is in IAD)
https://oidc.(us-east-1).amazonaws.com
https://log.sso-portal.eu-west-1.amazonaws.com.
https://portal.sso.eu-west-1.amazonaws.com
```

AWS ビルダー ID

```
https://view.awsapps.com/start (must be blocked to disable individual tier)
https://codewhisperer.us-east-1.amazonaws.com and q.us-east-1.amazonaws.com (should be
 permitted)
```

テレメトリ

以下は、許可リストに追加する必要がある Telemetry 固有のエンドポイントです。

```
https://telemetry.aws-language-servers.us-east-1.amazonaws.com/
https://client-telemetry.us-east-1.amazonaws.com
```

リファレンス

以下は、エンドポイントリファレンスのリストです。

```
idetoolkits-hostedfiles.amazonaws.com.
```

```
cognito-identity.us-east-1.amazonaws.com.
amazonwebservices.gallery.vsassets.io.
eu-west-1.prod.pr.analytics.console.aws.a2z.com.
prod.pa.cdn.uis.awsstatic.com.
portal.sso.eu-west-1.amazonaws.com.
log.sso-portal.eu-west-1.amazonaws.com.
prod.assets.shortbread.aws.dev.
prod.tools.shortbread.aws.dev.
prod.log.shortbread.aws.dev.
a.b.cdn.console.awsstatic.com.
assets.sso-portal.eu-west-1.amazonaws.com.
oidc.eu-west-1.amazonaws.com.
aws-toolkit-language-servers.amazonaws.com.
aws-language-servers.us-east-1.amazonaws.com.
idetoolkits.amazonwebservices.com.
```

AWS サービスとツールの使用

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 VS Code 向けツールキットサービスのガイドトピックとその機能の一覧です。サービスまたはツールを選択すると、その機能、設定方法、基本機能の使い方に関する詳細が表示されます。

トピック

- サーバーレスアプリケーションの使用
- Systems Manager オートメーションドキュメントの使用
- AWS Step Functions
- Threat Composer の使用
- リソースの使用

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 AWS のサービスを操作する

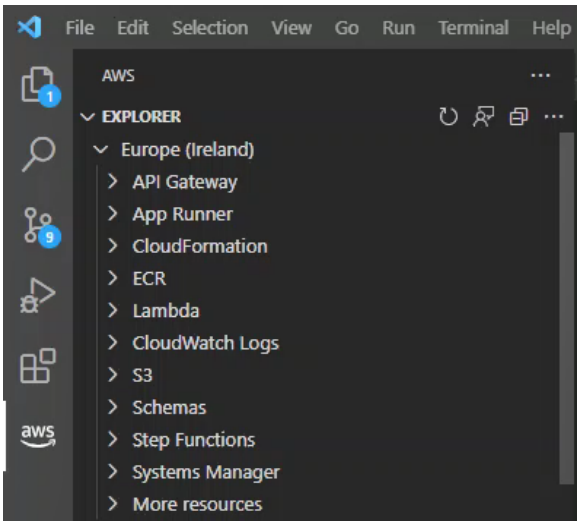
[AWS Explorer] には、AWS Toolkit for Visual Studio Code を使用するときに操作できるいくつかの AWS サービスのビューが表示されます。

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、[AWSExplorer] に項目が表示されます。AWSExplorer 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、アカウントの AWS Lambda 関数を表示するには、「AWS」で設定した認証情報に、少なくとも読み取り専用 認証とアクセス Lambda アクセス許可が含まれている必要があります。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 で直ぐに表示されない AWS サービスで操作したい場合、その他のリソースに進み、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。

  たとえば、使用可能なリソースタイプの選択から AWS Toolkit:CodeArtifact::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 に追加された後、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、JSON形式のテンプレートでリソースのプロパティと属性を記述できます。このテンプレートを保存して、AWS クラウドに新しいリソースを作ることができます。

AWS ドキュメント

AWS Toolkit for Visual Studio Code は `AWS SAM templates` の AWS Serverless Application Model `JSON Schema` をサポートし、直接 VS Code で定義、自動補完、検証を有効にすることで、テンプレートの作成エクスペリエンスを強化します。 AWSドキュメントは、すべての AWS SAM および CloudFormationリソースをサポートします。その他の詳細については、以下のリソースを参照してください。

- JSON スキーマの詳細については、JSON-Schema.org ウェブサイト の「JSON Schema」を参照してください。
- AWS SAM テンプレートの詳細については、「AWS Serverless Application Model デベロッパーガイド」の「AWS SAMテンプレート構造」トピックを参照してください。
- AWS リソースとプロパティタイプの詳細については、「CloudFormation ユーザーガイド」の「AWS リソースおよびプロパティタイプのリファレンス」を参照してください。
- AWS ツールキットで使用される AWS SAM スキーマの詳細については、AWS GitHub リポジトリの「AWS Serverless Application Model schema」を参照してください。

AWS ドキュメントの開始方法

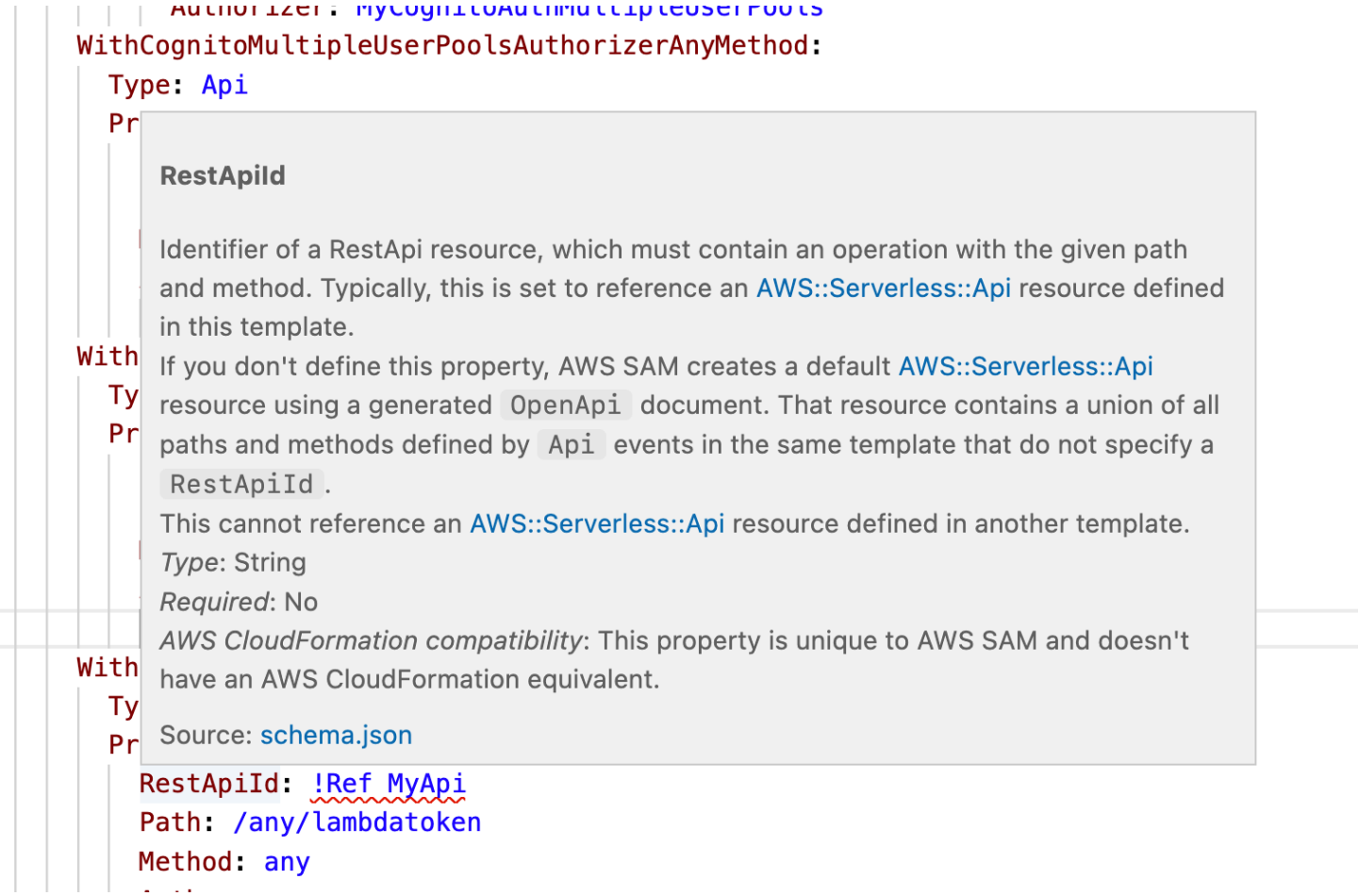
VS Code で AWS ドキュメントの使用を開始するには、IDE または VS Code Marketplace からAWS Toolkit for Visual Studio Code 拡張機能をインストールし、任意の AWS SAM テンプレートを開きます。

VS Code でのドキュメント表示、自動補完、検証

ドキュメントの表示、自動補完、検証は、 AWS ツールキットに含まれている機能です。これらの機能が VS Code でどのように表示されるかは、以下の画像を参照してください。

- 開いている AWS SAM テンプレートからドキュメントを表示するには、ドキュメントの行エントリにポインタを合わせます。
- 自動補完を使用するには、AWS SAM テンプレート内で入力を開始すると、入力内容に基づく候補がポップアップで表示されます。
- AWS SAM テンプレートは自動的にスキャンおよび検証され、エラーがハイライト表示されます。さらに、電球アイコンを選択すると追加の候補を確認できます。

これらの機能が VS Code でどのように表示されるかは、以下の画像を参照してください。



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。を通じて AWS Toolkit for Visual Studio Code、VS Code から直接 CodeCatalyst リソースを表示および管理できます。 AWS ツールキットを使用して、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 Code での Amazon CodeCatalyst リソースの操作

- 開発環境で Toolkit を使用する
- Amazon CodeCatalyst と VS コードのトラブルシューティング

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

- CodeCatalyst アカウントの作成
- AWS ツールキットと CodeCatalyst の接続

CodeCatalyst アカウントの作成

Toolkit for VS Code から CodeCatalyst に接続するには、アクティブな AWS ビルダー ID または AWS IAM アイデンティティセンター 認証情報が必要です。 AWS Builder ID、IAM Identity Center、CodeCatalyst 認証情報の詳細については、CodeCatalyst ユーザーガイド」の「CodeCatalyst でのセットアップ」セクションを参照してください。 CodeCatalyst

AWS ツールキットと CodeCatalyst の接続

AWS Toolkit を CodeCatalyst アカウントに接続するには、このユーザーガイドの「 への接続」トピックの「Amazon CodeCatalyst の認証」セクションを参照してください。 AWS

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
- 開発環境を開く
- CodeCatalyst で開発環境を作成する
- サードパーティのリポジトリから開発環境を作成する
- VS Code の CodeCatalyst コマンド

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。

AWS ツールキットを使用して CodeCatalyst アカウントから 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。

> **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。

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。
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、次の手順に従います。

> **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。

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。

CodeCatalyst で開発環境を作成する

新しい開発環境を作成するには、次の手順に従います。

> **Note**
>
> 新しい開発環境を作成する場合は、以下に留意してください。
>
> - AWS では、組織を簡素化し、開発環境の検索機能を向上させるため、エイリアスを指定することをお勧めします。
> - 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。
> - ストレージは、開発環境の作成後に変更できない唯一の設定です。
> - 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。

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。
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。
   - 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。

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。

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

AWS ツールキットに直接表示されない CodeCatalyst 関連の機能に割り当てられる追加の VS Code コマンドがあります。

CodeCatalyst に割り当てられているコマンドをコマンドパレットから一覧表示するには、次の手順に従います。

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. [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。

開発環境で Toolkit を使用する

開発環境は Amazon CodeCatalyst 用の仮想コンピューティング環境です。以下のセクションでは、AWS Toolkit for Visual Studio Codeを使用して開発環境を作成、起動、使用する方法について説明します。

開発環境の詳細については、「Amazon CodeCatalyst ユーザーガイド」の「開発環境」を参照してください。

devfiles を使用した開発環境の構成

devfile 仕様は、開発環境の設定を定義するために使用できる YAML のオープン標準形式です。すべての開発環境には devfile があります。リポジトリなしで、または devfile を含まないリポジトリから開発環境を作成すると、デフォルトがソースに自動的に適用されます。devfile は CodeCatalyst または IDE から更新できます。VS Code のローカルインスタンスでもリモートインスタンスでも devfile を更新するプロセスは同じですが、devfile をローカルで更新する場合は、更新を有効にする前に更新をソースリポジトリにプッシュする必要があります。

devfile を使用した開発環境の設定の詳細については、「Amazon CodeCatalyst ユーザーガイド」の「開発環境の設定」トピックを参照してください。

次の手順では、開発環境で実行されている Toolkit のリモートインスタンスから devfile を編集する方法を示します。

> **Important**
>
> VS Code から Devfile 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。

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。
3. VS Code エディタから devfile を更新し、変更を保存します。
4. 次回に開発環境を起動すると、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。

開発環境 AWS から への認証と接続

開発環境からすべての AWS リソースにアクセスするには、 Toolkit のリモートインスタンスを認証して AWS アカウントに接続する必要があります。Toolkit のリモートインスタンスは、開発環境の起動時に Toolkit のローカルインスタンスから継承した認証情報を使用して自動的に認証されます。

Toolkit のリモートインスタンスの認証情報を更新する手順は、Toolkit のローカルインスタンスでの認証手順と同じです。認証情報の更新、認証、および Toolkit から AWS への接続方法の詳しい手順については、このユーザーガイドの「開始方法」トピックで「AWSに接続する」セクションを参照してください。

と互換性のある各 AWS 認証方法の詳細については AWS Toolkit for Visual Studio Code、このユーザーガイドの「認証とアクセス」トピックを参照してください。

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 ツールキットを自動的にインストールし、 AWS ビルダー 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。
4. VS Code がリモート接続を閉じ、ローカルの開発インスタンスに戻ると、開発環境は正常に停止します。

開発環境設定を開く

現在の開発環境の設定を開くには、次の手順に従います。

> **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。

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. [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。

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、 AWS Toolkit からすべての CodeCatalyst 機能にアクセスすることはできません。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. AWS Explorer 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 メニューを開き(右クリック) 、AWS で呼び出す 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 での の使用 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 クラウド内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 リソースをプロビジョニングして設定する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 でソースコードリポジトリプロバイダーとしてサポートされているのは、GitHub です。

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、コンテナイメージの 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により AWS App Runner 、は次のタスクを完了できます。

- Amazon CloudWatch Logs Logs ロググループにログをプッシュします。
- Amazon CloudWatch Events ルールを作成して、Amazon Elastic Container Registry (Amazon ECR) イメージプッシュを購読します。

サービスにリンクされたロールを手動で作成する必要はありません。 AWS App Runner で、AWS マネジメントコンソール または によって呼び出される API オペレーションを使用してを作成すると AWS Toolkit for Visual Studio Code、 はこのサービスにリンクされたロールを自動的に AWS App Runner 作成します。

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によって管理されている Git リポジトリに保存する必要があります。App Runner でソースコードリポジトリプロバイダーとしてサポートされているのは、GitHub です。

GitHub リポジトリの設定については、GitHub の入門ガイドドキュメントを参照してください。

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場合、App Runner は GitHub への接続を確立します。リポジトリがプライベートである (つまり GitHub でパブリックにアクセスできない) 場合は、App Runner に接続の詳細情報を指定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 から AWSへリンクする接続を作成する必要があります。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場合は、[GitHub connections] (GitHub 接続) ページで使用可能な接続を選択します。
> 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 は現在、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すると、コマンドパレットによって提供される番号付きステップによって、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が案内されます。

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 が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 でソースコードリポジトリプロバイダーとしてサポートされているのは、GitHub です。

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する場合は、ECR アクセスロール ([AppRunnerECRAccessRole]) が必要です。このロールによって、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. 「接続を選択する」で、GitHub をリンクする接続を選択します AWS。選択可能な接続は、App Runner コンソールの [GitHub connections] (GitHub 接続) ページに表示されます。
2. [Select a remote GitHub repository] (リモート GitHub リポジトリの選択) では、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 が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 が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 が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 Logs 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、およびアクセスできます。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 が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 アプリケーションビルダー

AWS Toolkit for Visual Studio Code 向け AWS アプリケーションビルダーは、プロジェクトを視覚的に構築し、ローカルで反復処理し、アプリケーションを AWS にデプロイするためのガイドです。

以下のトピックでは、AWS Toolkit for Visual Studio Code から AWS アプリケーションビルダーを使用する方法について説明します。

トピック

- AWS Application Builder の使用

AWS Application Builder の使用

以下のセクションでは、 で AWS Application Builder にアクセスする方法について説明します AWS Toolkit for Visual Studio Code。Application Builder を使用すると、プロジェクトを視覚的に構築し、ローカルで反復処理して、 にデプロイできます AWS。Application Builder の機能と潜在的なユースケースの概要とローカル AWS Lambda エクスペリエンスについては、 AWS 「Developer YouTube video *New* AWS Lambda Local IDE Experience!」を参照してください。

AWS Application Builder エクスプローラーの使用

AWS Toolkit で Application Builder にアクセスするには、VS Code で AWS Toolkit を開き、AWS Application Builder エクスプローラーを展開します。 AWS Application Builder エクスプローラーには、VS Code エディタタブで Application Builder のチュートリアルを開くためのリンクが含まれて

おり、 AWS Application Builder 関連のリソースを含む現在の VS Code ワークスペース内のフォルダが表示されます。

Toolkit の Application Builder エクスプローラーから AWS 、project-folder-levelアクションが 4 つあります。

- [テンプレートファイルを開く] : VS Code エクスプローラーでテンプレートファイルを開きます。
- Infrastructure Composer で開く: VS Code エディタで AWS Infrastructure Composer を使用してテンプレートファイルを開きます。 AWS Infrastructure Composer の操作の詳細については、AWS Infrastructure Composer デベロッパーガイドのAWS 「Infrastructure Composer とは」トピックを参照してください。
- ビルド SAM テンプレート: AWS Toolkit でビルドダイアログのパラメータを指定を開きます。ビルドのビルドフラグを指定するか、samconfig のデフォルト値を使用するかを選択できます。AWS SAM テンプレートの詳細については、「 AWS Serverless Application Modelデベロッパーガイド」の「テンプレート構造」トピックを参照してください。
- [SAM アプリケーションのデプロイ]: VS Code で [デプロイコマンドの選択] ダイアログを開き、[デプロイ] または [同期] を選択して、デプロイ済みのアプリケーションを更新できます。 AWS SAM アプリケーションのデプロイの詳細については、「 デベロッパーガイド」の「アプリケーションとリソースのデプロイ」トピックを参照してください。 AWS Serverless Application Model

プロジェクトフォルダ内の AWS Lambda 関数の横にあるボタンアイコンから、または AWS Lambda 関数を右クリックしてアクセスできる 2 つのアクションがあります。

- [設定のローカル呼び出しおよびデバッグ]: VS Code エディタで [設定のローカル呼び出しおよびデバッグ] フォームを開きます。このフォームでは、`aws-sam` タイプの launch-configs を作成、編集、実行できます。SAM デバッグ設定の詳細については、このユーザーガイドの「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」トピックを参照してください。

> **Note**
>
> 現在、ARM64 アーキテクチャでの .NET Core アプリケーションのデバッグは VS Code ではサポートされていません。.NET Core アプリケーションをデバッグしようとすると、次のエラーが表示されます。
>
> ```
> The vsdbg debugger does not currently support the arm64
> architecture. Function will run locally without debug.
> ```

この問題の詳細については、DotNet GitHub リポジトリの 「VSCode-csharp」の問題を参照してください。

- [関数ハンドラを開く]: 関数ハンドラを含むプロジェクトファイルを開きます。

デプロイされた AWS Lambda 関数には 2 つの追加アクションがあります。

- [リモート呼び出し]: VS Code エディタで [リモート呼び出し設定] メニューを開きます。
- [ログの検索]: VS Code で [ログの検索] ダイアログを開きます。

アプリケーションビルダーのチュートリアル

Application Builder のチュートリアルは、 AWS Application Builder を使用して新しいアプリケーションを構築するプロセスを説明するstep-by-stepインタラクティブガイドです。Application Builder のチュートリアルには、 の Application Builder エクスプローラー AWS Toolkit for Visual Studio Code と VS Code Welcome タブの 2 つの場所からアクセスできます。 AWS Toolkit の Application Builder エクスプローラーから Application Builder のチュートリアルを選択すると、VS Code Editor ウィンドウの VS Code Welcome タブで Application Builder のチュートリアルが開きます。

アプリケーションビルダーのチュートリアルは、次の 5 つの主要セクションで構成されています。

1. インストール

インストールセクションでは、Application Builder やその他のオプション AWS CLI ツールに必要なツールがインストールされているかどうかを確認します。必要なツールがない場合、またはツールが古い場合は、正しいバージョンをインストールする手順が表示されます。

正しいツール AWS CLI とオプションのツールがインストールされているかどうかを確認するには、 のボタン、 AWS CLI またはテストする別のツールを選択します。ボタンを選択すると、AWS ツールキットログ が更新され、VS Code にツールのステータスを示すアラートメッセージが表示されます。ツールをインストールまたは更新する必要がある場合、アプリケーションビルダーのチュートリアルが更新され、続行するために必要な手順とリソースが表示されます。

のインストールの詳細については AWS CLI、「 AWS CLIデベロッパーガイド」の「 の最新バージョンのインストールまたは更新 AWS CLI」を参照してください。 AWS SAM CLI のインストールの詳細については、「 CLI AWS SAMデベロッパーガイド」のAWS SAM 「CLI のインストール」トピックを参照してください。

2. アプリケーションテンプレートの選択

[アプリケーションテンプレートの選択] セクションでは、テンプレートから新しいアプリケーションを構築する手順が示されます。

テンプレートを選択してアプリケーションを初期化するには、以下の手順を実行します。

1. [アプリケーションビルダーのチュートリアル] から [アプリケーションテンプレートの選択] セクションを選択すると、画面にテンプレートオプションのリストが表示されます。
2. リストからテンプレートを選択し、[プロジェクトを初期化] ボタンを選択して VS Code ダイアログを開きます。
3. VS Code ダイアログのステップを完了して、新しいアプリケーションを初期化します。
4. Toolkit は、初期化プロセス中にアプリケーションのステータスで更新を AWS ログに記録します。
5. アプリケーションビルダーエクスプローラーでアプリケーションを表示するには、[アプリケーションビルダーエクスプローラーの更新] アイコンを選択して、エクスプローラーに変更を反映させます。

3. ローカルでの反復処理

Iterate local セクションには、VS Code および AWS Toolkit エクスプローラーで利用可能な Application Builder 機能を使用して反復する方法を示すサンプルイメージが含まれています。

VS Code および AWS Toolkit エクスプローラーで使用できるすべての Application Builder 機能の詳細については、このユーザーガイドトピックにある「Application Builder エクスプローラーの使用」セクションを参照してください。

4. にデプロイする AWS

Deploy to AWS セクションには、アプリケーションをデプロイする AWS 目的で に接続する認証情報を設定する方法と、Application Builder を使用してアプリケーションをデプロイする方法の例が含まれています。

Application Builder のチュートリアルから既存の認証情報 AWS を使用して に接続するには、次のいずれかの手順を実行します。

ワークフォース: シングルサインオン AWS で にサインインします。

1. Application Builder のチュートリアルの Deploy to AWS セクションで、認証情報の設定ボタンを選択して AWS Toolkit Explorer の AWS: LOGIN メニューを開きます。

2. [AWS: ログイン] メニューから [ワークフォース] を選択し、[続行] ボタンを選択して続行します。
3. 指定されたフィールドに開始 URL を入力し、ドロップダウンメニューから AWS リージョンを選択し、[続行] ボタンを選択して続行します。
4. VS Code ポップアップウィンドウから、デフォルトのブラウザで AWS 認証サイトを開くことを確認します。
5. デフォルトのブラウザから認証手順を完了します。認証の完了が通知されたら、ブラウザウィンドウを安全に閉じることができます。

IAM 認証情報: AWS CLI ツールで使用するためのキーを保存します。

1. Application Builder のチュートリアルの Deploy to AWS セクションで、認証情報の設定ボタンを選択して AWS Toolkit Explorer の AWS: LOGIN メニューを開きます。
2. [AWS: ログイン] メニューから [IAM 認証情報] を選択し、[続行] ボタンを選択して続行します。
3. 指定されたフィールドにプロファイル名を入力し、 **`Access Key`** と **`Secret Key`** を入力して、[続行] ボタンを選択して続行します。
4. VS Code は認証のステータスを表示し、認証が完了したか、認証情報が無効だったかを通知します。

を使用してデプロイするための認証情報の設定の詳細については AWS CLI、「 AWS CLIデベロッパーガイド」の「 の設定 AWS CLI」トピックを参照してください。既存の認証情報を使用して AWS Toolkit AWS から 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「 への接続 AWS」トピックを参照してください。

AWS Infrastructure Composer

AWS Toolkit for Visual Studio Code を使用して AWS Infrastructure Composer を使用できます。AWSInfrastructure Composer は、アプリケーションアーキテクチャの設計と 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 Infrastructure Composer の詳細については、「AWS Infrastructure 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 Infrastructure Composer を使用する方法について説明します。

トピック

ツールキットでの AWS Infrastructure Composer の使用

AWS Toolkit for Visual Studio Code 向けの AWS Infrastructure 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設計できます。また、Infrastructure Composer を使用して、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。Infrastructure 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 Infrastructure Composer、開始方法、チュートリアルの詳細については、「AWS Infrastructure Composer 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 Infrastructure Composer サービスにアクセスする方法について説明します。

ツールキットから AWS Infrastructure Composer にアクセスする

ツールキットから AWS Infrastructure 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 Infrastructure 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 Infrastructure 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 Infrastructure 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 Infrastructure Composer にアクセスする

1. VS Code から、AWS Infrastructure 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 Infrastructure 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 Infrastructure 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 Infrastructure Composer**」と入力して結果が表示されたら、[AWS Infrastructure 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 Infrastructure 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、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] ウィンドウ内の **`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

- 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

CloudFormation スタックの削除

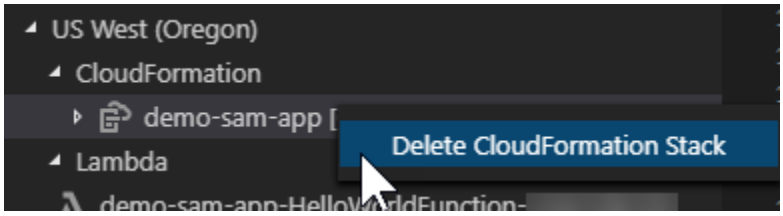
AWS Toolkit for Visual Studio Code を使用して CloudFormation スタックを削除できます。

前提条件

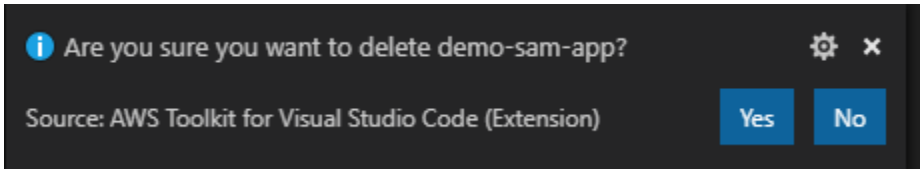
- システムが、「Toolkit for VS Code をインストールする」で指定されている前提条件を満たしていることを確認してください。
- 「認証とアクセス」で設定した認証情報に、CloudFormation サービスに対する適切な読み取り/書き込みアクセス許可が含まれていることを確認します。AWSExplorer 内、CloudFormation の下で、「CloudFormation リソースの読み込みエラー」のようなメッセージが表示される場合は、これらの認証情報にアタッチされた許可をチェックしてください。アクセス許可に変更を加えると、VS Code の AWSExplorer に影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

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 テンプレートの作成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

- CloudFormation についてすでに説明したすべての基準
- 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、SAM テンプレートスキーマは、CloudFormation テンプレートにサーバーレストランスフォームを追加してファイルを保存した後に適用されます。

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、および AWS 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

- を使用した 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ムで CloudWatch ログイベントを操作する AWS Toolkit for Visual Studio Code
- CloudWatch ロググループを検索する
- Amazon CloudWatch Logs Live Tail

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

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

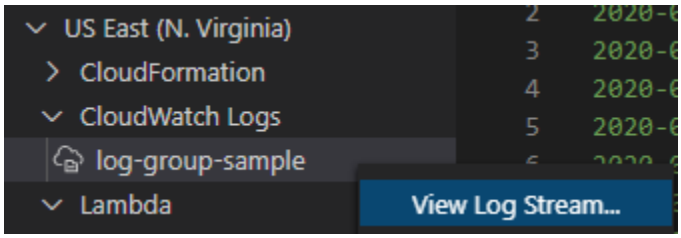
- CloudWatch Logs ノードのロググループとログストリームの表示

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。

を使用してログストリームで CloudWatch ログイベントを操作する AWS Toolkit for Visual Studio Code

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、AWS Toolkit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、AWS Toolkit 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 エディタに検索結果が表示されます。

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 Code 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 CloudWatch Logs Live Tail

Amazon CloudWatch Logs Live Tail を使用すると、特定のロググループに取り込まれる CloudWatch ログイベントをリアルタイムでストリーミングできます。

Live Tail 機能の詳細については、「Amazon CloudWatch Logs ユーザーガイド」の「CloudWatch Logs Live Tail を使用したトラブルシューティング」を参照してください。

Live Tail セッションでは、セッションの使用時間ごとに 1 分間隔でコストが発生します。料金の詳細については、「Amazon CloudWatch 料金表」の「有料利用枠」セクションの [ログ] タブを参照してください。

VS Code コマンドパレットから Live Tail セッションを開始する

VS Code コマンドパレットから Live Tail セッションを開始するには、次の手順を実行します。

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。

コマンドパレットからテーリングセッションを開始する

1. VS Code から **cmd+shift+p** (windows の場合は **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 `AWS: Tail Log Group` コマンドを入力し、選択して VS Code の [Tail 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. [ロググループのテーリング (1/3)] プロンプトから、テーリングするロググループを選択します。
4. [次からのログイベントを含める... (2/3)] プロンプトから、テーリングセッションに適用するログストリームフィルターを選択します。
5. [ログイベントフィルタパターンの指定... (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. 完了すると、結果が VS Code エディタにストリーミングされます。

> **Note**
>
> VS Code ウィンドウで実行されている Live Tail セッションが、新しく送信されたロググループのテーリングコマンドの設定と一致する場合、新しいセッションは開始されません。代わりに、既存のセッションがアクティブなテキストエディタになります。

AWS エクスプローラーから Live Tail セッションを開始する

AWS ツールキットエクスプローラーから Live Tail セッションを開始するには、次の手順を実行します。

AWS エクスプローラーからテーリングセッションを開始する

1. AWS ツールキットエクスプローラーから、[CloudWatch] を展開します。
2. テーリングする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、[ロググループのテーリング] を選択して、テーリングプロンプトを開きます。

3. プロンプトに従って続行します。
4. 結果は VS Code エディタにストリーミングされます。

Live Tail セッションの停止

実行中のテーリングセッションを停止する方法は 2 つあります。

テーリングセッションの停止

1. テーリングセッションのテキストドキュメントの下部にある Stop tailing CodeLens をクリックします。
2. テーリングセッションのテキストドキュメントを含むすべてのエディタを閉じます。

Amazon DocumentDB

AWS Toolkit for Visual Studio Codeを使用して、VS Code で直接 Amazon DocumentDB クラスターとインスタンスを管理することができます。Amazon DocumentDB (MongoDB 互換) は、MongoDB と互換性のあるデータベースをクラウド上で簡単にセットアップ、操作、スケーリングできる、高速で信頼性の高い完全マネージドデータベースサービスです。Amazon DocumentDB の詳細については、「Amazon DocumentDB デベロッパーガイド」を参照してください。

次のトピックでは、 AWS Toolkit for Visual Studio Codeで Amazon DocumentDB を使用する方法について説明します。

トピック

- ツールキットでの Amazon DocumentDB の使用

ツールキットでの Amazon DocumentDB の使用

Amazon DocumentDB (MongoDB 互換) は、MongoDB と互換性のあるデータベースをクラウド上で簡単にセットアップ、操作、スケーリングできる、高速で信頼性の高い完全マネージドデータベースサービスです。

Amazon DocumentDB、開始方法、チュートリアルの詳細については、「Amazon DocumentDB デベロッパーガイド」を参照してください。

次のセクションでは、 AWS Toolkit for Visual Studio Codeで Amazon DocumentDB を使用する方法について説明します。

AWS Toolkit から Amazon DocumentDB にアクセスする

AWS Toolkit を使用して Amazon DocumentDB にアクセスするには、次の手順を実行します。

AWS Toolkit での Amazon DocumentDB へのアクセス

1. VS Code から を開きます AWS Toolkit for Visual Studio Code。
2. Toolkit から AWS Explorer を展開します。
3. [エクスプローラー] から Amazon DocumentDB を展開し、既存の Amazon DocumentDB リソースを表示します。

インスタンスベースのクラスターの作成

Amazon DocumentDB の使用を開始するには、次の手順を実行してクラスターを作成します。

インスタンスベースのクラスターの作成

1. から AWS Toolkit for Visual Studio Code、Amazon DocumentDB のコンテキストメニューを開き (右クリック)、クラスターの作成を選択して VS Code で Amazon DocumentDB クラスターの作成ダイアログを開きます。
2. [クラスタータイプ] 画面で [インスタンスベースのクラスター] を選択します。
3. [クラスター名] 画面で、新しいクラスターの名前を指定します。
4. [エンジンバージョンの選択] 画面で、任意の Amazon DocumentDB エンジンバージョンを選択します。
5. [管理者のユーザー名とパスワード] 画面で、クラスターを保護する管理者のユーザー名とパスワードを指定します。
6. [ストレージ暗号化の指定] 画面で、クラスターを暗号化するかどうかを選択します。
7. [インスタンス数] 画面で、任意のインスタンス数を設定します。
8. [インスタンスクラスの選択] 画面で、任意のインスタンスクラスを選択し、新しいクラスターの作成に進みます。

> **Note**
>
> クラスターの作成には数分かかる場合があります。

クラスターエンドポイントのコピー

Amazon DocumentDB クラスターエンドポイントをコピーするには、次の手順を実行します。

クラスターエンドポイントのコピー

1. から AWS Toolkit for Visual Studio Code、Amazon DocumentDB を展開して Amazon DocumentDB クラスターを表示します。
2. 接続の詳細をコピーするクラスターを右クリックし、[エンドポイントのコピー] を選択してクラスターエンドポイント情報をクリップボードにコピーします。
3. これで、クラスターのエンドポイントをドキュメントに貼り付けることができます。

ブラウザで開く

AWS コンソールで Amazon DocumentDB クラスターを開き、クラスター管理機能を強化します。AWS コンソールをデフォルトのウェブブラウザで Amazon DocumentDB クラスターに開くには、次の手順を実行します。

AWS コンソールでクラスターを開く

1. から AWS Toolkit for Visual Studio Code、Amazon DocumentDB を展開して Amazon DocumentDB クラスターを表示します。
2. AWS コンソールで表示するクラスターを右クリックし、ブラウザで開くを選択します。
3. AWS コンソールがデフォルトのウェブブラウザで Amazon DocumentDB クラスターに開きます。

既存のクラスターの拡張

インスタンスを追加して Amazon DocumentDB クラスターをスケールするには、次の手順を実行します。

インスタンスを追加してクラスターを拡張する

1. から AWS Toolkit for Visual Studio Code、Amazon DocumentDB を展開して Amazon DocumentDB クラスターを表示します。
2. 展開したいクラスターを右クリックし、[インスタンスを追加する] を選択して、VS Code でAdd an Instance ダイアログを開きます。

3. プロンプトが表示されたら、新しいインスタンスの名前をテキストフィールドに入力し、**Enter** キーを押して続行します。
4. プロンプトが表示されたら、リストからインスタンスクラスを選択して続行します。
5. AWS エクスプローラーに作成ステータスが表示され、新しいインスタンスの準備ができたら更新されます。

クラスターの停止

Amazon DocumentDB クラスターを停止するには、次の手順を実行します。

> **Note**
>
> クラスターが停止している間、ほとんどのクラスター管理機能は使用できなくなります。

Amazon DocumentDB クラスターの停止

1. から AWS Toolkit for Visual Studio Code、Amazon DocumentDB を展開して Amazon DocumentDB クラスターを表示します。
2. クラスターの横にある [クラスターの停止] ボタンを選択するか、クラスターを右クリックして [クラスターの停止] を選択します。
3. プロンプトが表示されたら、[はい] を選択してクラスターを停止するか、[キャンセル] を選択して停止プロセスをキャンセルし、クラスターを実行したままにします。
4. AWS エクスプローラー にはクラスターのステータスが表示され、クラスターが停止すると更新されます。

インスタンスの再起動

インスタンスの再起動は、クラスター全体に影響を与えることなくトラブルシューティングや軽微な変更を行うのに便利です。Amazon DocumentDB インスタンスを再起動するには、次の手順を実行します。

クラスターインスタンスの再起動

1. から AWS Toolkit for Visual Studio Code、Amazon DocumentDB を展開して Amazon DocumentDB クラスターを表示します。
2. 再起動するクラスターインスタンスを右クリックし、[インスタンスを再起動] を選択します。

3. プロンプトが表示されたら、[はい] を選択してインスタンスを再起動するか、[キャンセル] を選択して再起動プロセスをキャンセルし、インスタンスを停止したままにします。
4. AWS エクスプローラー にはクラスターのステータスが表示され、インスタンスが再起動されると更新されます。

インスタンスの削除

Amazon DocumentDB クラスターインスタンスを削除するには、次の手順を実行します。

> **Note**
>
> インスタンスを削除しても、クラスター内のデータには影響しません。プライマリインスタンスを削除すると、レプリカインスタンスの 1 つが書き込み可能なインスタンスとして引き継がれます。

クラスターインスタンスの削除

1. から AWS Toolkit for Visual Studio Code、Amazon DocumentDB を展開して Amazon DocumentDB クラスターを表示します。
2. 削除するクラスターインスタンスを右クリックし、[削除] を選択して VS Code で delete-cluster-instance 確認ダイアログを開きます。
3. 確認プロンプトに従い、**`Enter`** キーを押してクラスターインスタンスを削除します。
4. AWS エクスプローラー はクラスターインスタンスのステータスを表示し、インスタンスが削除されると更新されます。

タグの表示、追加、削除

タグは、環境内のリソースを整理および追跡するために使用されます。Amazon DocumentDB クラスターに関連付けられたタグを表示または編集するには、次のいずれかの手順を実行します。

クラスタータグの表示

1. から AWS Toolkit for Visual Studio Code、Amazon DocumentDB を展開して Amazon DocumentDB クラスターを表示します。
2. タグを表示するクラスターを右クリックし、[タグ...] を選択して [**`your cluster name`** のタグ] ダイアログを開きます。

3. タグがダイアログウィンドウに表示されます。タグがクラスターに関連付けられていない場合は、[タグが割り当てられていません] というメッセージが表示されます。

クラスターへのタグの追加

1. から AWS Toolkit for Visual Studio Code、Amazon DocumentDB を展開して Amazon DocumentDB クラスターを表示します。
2. タグを追加するクラスターを右クリックし、[タグ...] を選択して [**your cluster name** のタグ] ダイアログを開きます。
3. [タグの追加...] ボタンを選択して、VS Code でAdd Tag ダイアログを開きます。
4. テキストフィールドに新しいタグを入力し、Enter キーを押して続行します。
5. テキストフィールドに値を入力し、Enter キーを押してキーと値のペアをクラスターに追加します。

クラスターからのタグの削除

1. から AWS Toolkit for Visual Studio Code、Amazon DocumentDB を展開して Amazon DocumentDB クラスターを表示します。
2. タグを削除するクラスターを右クリックし、[タグ...] を選択して [**your cluster name** のタグ] ダイアログを開きます。
3. [タグの削除...] ボタンを選択して、VS Code の Remove a tag from **your cluster name** ダイアログを開きます。
4. 表示されるリストから削除するタグを選択して、クラスターからタグを削除します。

インスタンスクラスの変更

Amazon DocumentDB クラスターインスタンスのクラスを変更するには、次の手順を実行します。

インスタンスクラスの変更

1. から AWS Toolkit for Visual Studio Code、Amazon DocumentDB を展開して Amazon DocumentDB クラスターを表示します。
2. 変更するクラスターインスタンスを右クリックし、[クラスの変更...] を選択して、VS Code で Select instance class ダイアログを開きます。
3. リストからインスタンスの新しいクラスを選択して、クラスを更新します。

4. AWS エクスプローラー はクラスターインスタンスのステータスを表示し、インスタンスが変更されると更新されます。

Amazon Elastic Compute Cloud

AWS Toolkit for Visual Studio Code の Amazon Elastic Compute Cloud を使用すると、VS Code から Amazon EC2 インスタンスを起動して接続できます。Amazon EC2 の詳細については、「Amazon Elastic Compute Cloud ユーザーガイド」の「Amazon EC2 とは」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 アプリケーションビルダーを使用する方法について説明します。

トピック

- Amazon Elastic Compute Cloud の使用
- Amazon Elastic Compute Cloud のトラブルシューティング

Amazon Elastic Compute Cloud の使用

以下のセクションでは、 AWS Toolkit for Visual Studio Codeで Amazon Elastic Compute Cloud を使用する方法について説明します。

前提条件

このユーザーガイドトピックで説明されている機能は、以下のオペレーティングシステムを搭載した Amazon EC2 インスタンスでテストされています。

- Windows 2016 以降

> **Note**
>
> この OS は、VS Code ターミナルに接続している場合にのみ動作します。完全な VS Code リモートインスタンスに接続している場合は動作しません。VS Code ターミナルとリモートインスタンスの詳細については、VS Code ドキュメントの「ターミナルの開始方法」および「VS Code リモート開発」トピックを参照してください。

- Amazon Linux 2023
- Ubuntu 22.04

Amazon EC2 インスタンスへのリモート接続を開くにはローカルにインストールされた SSH が必要ですが、Amazon EC2 インスタンスへのターミナルを開くのに SSH は必須ではありません。

Amazon EC2 インスタンスプロファイルには、次の AWS Identity and Access Management (IAM) アクセス許可が含まれている必要があります。

```
"ssmmessages:CreateControlChannel",
"ssmmessages:CreateDataChannel",
"ssmmessages:OpenControlChannel",
"ssmmessages:OpenDataChannel",
"ssm:DescribeAssociation",
"ssm:ListAssociations",
"ssm:UpdateInstanceInformation
```

> **Note**
>
> 必要なアクセス許可は、次の AWS 管理ポリシーに含まれています。
>
> - `AmazonSSMManagedInstanceCore`
> - `AmazonSSMManagedEC2InstanceDefaultPolicy`

既存の Amazon EC2 インスタンスの表示

AWS Toolkit から既存の Amazon EC2 インスタンスを表示するには、次の手順を実行します。

1. AWS Toolkit から Toolkit Explorer AWS を展開します。
2. 表示する Amazon EC2 インスタンスを含むリージョンを展開します。
3. [EC2] の見出しを展開して、既存の Amazon EC2 インスタンスを表示します。

新しい Amazon EC2 インスタンスの起動

Toolkit を使用して新しい Amazon EC2 インスタンスを作成するには、3 つの方法があります AWS 。

各ワークフローでは、 AWS コンソールで [インスタンスの起動] ウィザードを開きます。[インスタンス起動] ウィザードから新しい Amazon EC2 インスタンスを起動する方法の詳細については、

「Amazon Elastic Compute Cloud ユーザーガイド」のコンソールトピックの「インスタンス起動ウィザードを使用して EC2 インスタンスを起動する」を参照してください。新しい Amazon EC2 インスタンスを起動するには、次のいずれかの手順を実行します。

VS Code コマンドパレットからの新しい Amazon EC2 インスタンスの起動

1. VS Code から **command + shift + P (Windows: ctrl + shift + P)** を押して VS Code コマンドパレットを開きます。
2. VS Code コマンドパレットから **AWS: Launch EC2** コマンドを検索し、リストに入力するときにそれを選択して、VS Code で Launch EC2 instance [Select Region] プロンプトを開きます。
3. EC2 インスタンスの起動 リージョンの選択 プロンプトで、新しいインスタンスを起動するリージョンを選択し、デフォルトのウェブブラウザで AWS コンソールを開くことを確認します。
4. デフォルトのウェブブラウザの AWS コンソールから、認証プロセスを完了してインスタンスの起動ウィザードに進みます。
5. [インスタンスの起動] ウィザードで必要なセクションを完了し、[インスタンスの起動] ボタンを選択して新しい Amazon EC2 インスタンスを起動します。
6. AWS Explorer が更新され、新しい Amazon EC2 インスタンスが表示されます。

AWS Explorer から新しい Amazon EC2 インスタンスを起動する

1. AWS Toolkit Explorer を展開し、新しい Amazon EC2 インスタンスを作成するリージョンを展開します。
2. [EC2] の見出しを展開するか見出しにカーソルを合わせ、+ (EC2 インスタンスの起動) アイコンを選択します。
3. プロンプトが表示されたら、デフォルトのウェブブラウザで AWS コンソールを開くことを確認します。
4. ウェブブラウザの AWS コンソールから、認証プロセスを完了してインスタンスの起動ウィザードに進みます。
5. [インスタンスの起動] ウィザードで必要なセクションを完了し、[インスタンスの起動] ボタンを選択して新しい Amazon EC2 インスタンスを起動します。
6. AWS Explorer が更新され、新しい Amazon EC2 インスタンスが表示されます。

コンテキスト (右クリック) メニューからの新しい Amazon EC2 インスタンスの起動

1. AWS Toolkit Explorer を展開し、新しい Amazon EC2 インスタンスを作成するリージョンを展開します。
2. [EC2] の見出しを右クリックし、[EC2 インスタンスの起動]を選択します。
3. プロンプトが表示されたら、デフォルトのウェブブラウザで AWS コンソールを開くことを確認します。
4. ウェブブラウザの AWS コンソールから、認証プロセスを完了してインスタンスの起動ウィザードに進みます。
5. [インスタンスの起動] ウィザードで必要なセクションを完了し、[インスタンスの起動] ボタンを選択して新しい Amazon EC2 インスタンスを起動します。
6. AWS Explorer が更新され、新しい Amazon EC2 インスタンスが表示されます。

VS Code から Amazon EC2 インスタンスへの接続

VS Code から Amazon EC2 インスタンスに接続するには、3 つの方法があります。VS Code を EC2 インスタンスに接続するには、次のいずれかの手順を実行します。

コマンドパレットから VS Code を Amazon EC2 インスタンスに接続

1. VS Code から **command + shift + P (Windows: ctrl + shift + P)** を押して VS Code コマンドパレットを開きます。
2. VS Code コマンドパレットから **AWS: Connect VS Code to EC2 instance...** コマンドを検索し、リストに入力するときにそれを選択して、[Select EC2 Instance] プロンプトを開きます。
3. Select EC2 Instance プロンプトで、接続するインスタンスを含むリージョンを選択し、接続するインスタンスを選択します。
4. VS Code は、接続の確立中にステータスを表示します。
5. 接続が完了すると、新しいウィンドウが開き、Amazon EC2 インスタンスが表示されます。

AWS Explorer から VS Code を Amazon EC2 インスタンスに接続する。

1. AWS Toolkit Explorer を展開し、接続先の Amazon EC2 インスタンスを含むリージョンを展開します。
2. Amazon EC2 インスタンスにカーソルを合わせ、 (VS Code を EC2 インスタンスに接続する) アイコンを選択します。

> **Note**
>
> AWS Explorer の EC2 サービス見出しから (VS Code を EC2 インスタンスに接続) アイコンを選択することもできます。 EC2

3. VS Code は、接続の確立中にステータスを表示します。
4. 接続が完了すると、新しいウィンドウが開き、Amazon EC2 インスタンスが表示されます。

右クリックメニューから VS Code を Amazon EC2 インスタンスに接続

1. AWS Toolkit Explorer を展開し、接続先の Amazon EC2 インスタンスを含むリージョンを展開します。
2. 接続する Amazon EC2 インスタンスを右クリックし、[VS Code を EC2 インスタンスに接続] を選択します。

> **Note**
>
> AWS Explorer で EC2 サービスの見出しを右クリックし、Connect VS Code to EC2 インスタンスを選択することもできます。

3. VS Code は、接続の確立中にステータスを表示します。
4. 接続が完了すると、新しいウィンドウが開き、Amazon EC2 インスタンスが表示されます。

ターミナルを開いて Amazon EC2 インスタンスに接続

VS Code ターミナルから Amazon EC2 インスタンスに接続するには、3 つの方法があります。

コマンドパレットから VS Code を Amazon EC2 インスタンスに接続

1. VS Code から **`command + shift + P (Windows: ctrl + shift + P)`** を押して VS Code コマンドパレットを開きます。
2. VS Code コマンドパレットから **`AWS:Open terminal to EC2 instance...`** コマンドを検索し、リストに入力するときにそれを選択して、[Select EC2 Instance] プロンプトを開きます。
3. Select EC2 Instance プロンプトで、ターミナルで開くインスタンスを含むリージョンを選択し、インスタンスを選択します。
4. VS Code は、接続の確立中にステータスを表示します。

5. VS Code ターミナルが開き、接続が完了すると新しいセッションが表示されます。

AWS Explorer から VS Code ターミナルで Amazon EC2 インスタンスを開きます。

1. AWS Toolkit Explorer を展開し、接続先の Amazon EC2 インスタンスを含むリージョンを展開します。
2. Amazon EC2 インスタンスにカーソルを合わせ、 (ターミナルを開いてEC2 インスタンスに接続...) アイコンを選択します。

> **Note**
>
> AWS Explorer の EC2 サービス見出しから (Open terminal to EC2 instance...) アイコンを選択することもできます。 EC2

3. VS Code は、接続の確立中にステータスを表示します。
4. VS Code ターミナルが開き、接続が完了すると新しいセッションが表示されます。

右クリックメニューから VS Code ターミナルで Amazon EC2 インスタンスを開く

1. AWS Toolkit Explorer を展開し、VS Code ターミナルで開く Amazon EC2 インスタンスを含むリージョンを展開します。
2. ターミナルで開く Amazon EC2 インスタンスを右クリックし、[ターミナルを開いて EC2 インスタンスに接続...]を選択します。

> **Note**
>
> AWS Explorer で EC2 サービスの見出しを右クリックし、EC2 インスタンスへのターミナルを開くを選択することもできます。

3. VS Code は、接続の確立中にステータスを表示します。
4. VS Code ターミナルが開き、接続が完了すると新しいセッションが表示されます。

Amazon EC2 インスタンスの起動または再起動

Amazon EC2 インスタンスを起動または再起動するには、3 つの方法があります。

コマンドパレットからの Amazon EC2 インスタンスの再起動

1. VS Code から **command + shift + P (Windows: ctrl + shift + P)** を押して VS Code コマンドパレットを開きます。
2. VS Code コマンドパレットから **AWS: Reboot EC2 instance** コマンドを検索し、リストに入力するときにそれを選択して、[Select EC2 Instance] プロンプトを開きます。

> **Note**
>
> 実行されていないインスタンスを起動するには、 **AWS: Start EC2 instance** コマンドを選択する必要があります。**AWS: Reboot EC2 instance** コマンドは、現在実行中のインスタンスのみを再起動します。

3. Select EC2 Instance プロンプトで、起動または再起動するインスタンスを含むリージョンを選択します。
4. VS Code は、インスタンスの再起動中にステータスを表示します。
5. AWS Explorer が更新され、再起動が完了したときにインスタンスが実行されていることが示されます。

AWS Explorer から Amazon EC2 インスタンスを起動または再起動する

1. AWS Toolkit Explorer を展開し、起動または再起動する Amazon EC2 インスタンスを含むリージョンを展開します。
2. Amazon EC2 インスタンスにカーソルを合わせ、 (EC2 インスタンスの再起動) アイコンを選択します。

> **Note**
>
> インスタンスが停止している場合、選択できるのは (EC2 インスタンスの起動) アイコンのみです。

3. VS Code は、インスタンスの再起動中にステータスを表示します。
4. AWS Explorer が更新され、再起動が完了したときにインスタンスが実行されていることが示されます。

右クリックメニューからの Amazon EC2 インスタンスを起動または再起動

1. AWS Toolkit Explorer を展開し、起動または再起動する Amazon EC2 インスタンスを含むリージョンを展開します。
2. 接続する Amazon EC2 インスタンスを右クリックし、[EC2 インスタンスの再起動]を選択します。

> **Note**
>
> インスタンスが停止している場合、選択できるのは [EC2 インスタンスの起動] アイコンのみです。

3. VS Code は、インスタンスの再起動中にステータスを表示します。
4. AWS Explorer が更新され、再起動が完了したときにインスタンスが実行されていることが示されます。

Amazon EC2 インスタンスの停止

Amazon EC2 インスタンスを停止する方法は 3 つあります。

コマンドパレットからの Amazon EC2 インスタンスの停止

1. VS Code から **command + shift + P (Windows: ctrl + shift + P)** を押して VS Code コマンドパレットを開きます。
2. VS Code コマンドパレットから **AWS: Stop EC2 instance** コマンドを検索し、リストに入力するときにそれを選択して、[Select EC2 Instance] プロンプトを開きます。
3. Select EC2 Instance プロンプトで、停止するインスタンスを含むリージョンを選択します。
4. VS Code は、インスタンスの停止中にステータスを表示します。
5. AWS Explorer が更新され、インスタンスが停止されたことが示されます。

AWS Explorer からの Amazon EC2 インスタンスの停止

1. AWS Toolkit Explorer を展開し、停止する Amazon EC2 インスタンスを含むリージョンを展開します。
2. Amazon EC2 インスタンスにカーソルを合わせ、 (EC2 インスタンスの停止) アイコンを選択します。

3. VS Code は、インスタンスの停止中にステータスを表示します。
4. AWS Explorer が更新され、インスタンスが停止したことが示されます。

右クリックメニューからの Amazon EC2 インスタンスの停止

1. AWS Toolkit Explorer を展開し、停止する Amazon EC2 インスタンスを含むリージョンを展開します。
2. 接続する Amazon EC2 インスタンスを右クリックし、[EC2 インスタンスの再起動]を選択します。
3. VS Code は、インスタンスの停止中にステータスを表示します。
4. AWS Explorer が更新され、インスタンスが停止したことが示されます。

インスタンス ID のコピー

インスタンス ID をコピーするには、次の手順を実行します。

1. ID をコピーするインスタンスを右クリックします。
2. [インスタンス IDのコピー] を選択します。
3. インスタンス ID がローカルクリップボードにコピーされます。

名前のコピー

インスタンス名をコピーするには、次の手順を実行します。

1. 名前をコピーするインスタンスを右クリックします。
2. [インスタンス名のコピー] を選択します。
3. インスタンス名がローカルクリップボードにコピーされます。

ARN のコピー

インスタンスの ARN をコピーするには、次の手順を実行します。

1. ARN をコピーするインスタンスを右クリックします。
2. [インスタンスの ARN のコピー] を選択します。
3. インスタンスの ARN がローカルクリップボードにコピーされます。

Amazon Elastic Compute Cloud のトラブルシューティング

以下のセクションでは、 AWS Toolkit for Visual Studio Codeで Amazon Elastic Compute Cloud を使用する際に発生する可能性のある既知の問題のトラブルシューティング方法について説明します。Amazon EC2 サービス固有の問題のトラブルシューティングの詳細については、「Amazon Elastic Compute Cloud ユーザーガイド」の「Amazon EC2 インスタンスに関する問題をトラブルシューティングする」トピックを参照してください。

一般的なデバッグ

何らかの理由でリモート接続の問題が発生した場合は、まず AWS コンソールから AWS Systems Manager 接続を確立できるかどうかを確認します。

AWS コンソールから Systems Manager を介して Amazon EC2 インスタンスに接続するには、次の手順を実行します。

1. ウェブブラウザから、 AWS コンソールに移動します。
2. AWS コンソール EC2 のランディングに進むには、認証を完了します。
3. Amazon EC2 ナビゲーションペインで [インスタンス] を選択します。
4. 接続するインスタンスの横にあるチェックボックスをオンにします。
5. [接続] ボタンを選択して、新しいブラウザタブで [インスタンスへの接続] 画面を開きます。

> **Note**
>
> インスタンスは、実行中の場合にのみ接続できます。[接続] ボタンを選択できない場合は、インスタンスが実行されていることを確認します。

6. [インスタンスへの接続] 画面で [Session Manager] タブを選択し、[接続] ボタンを選択して、現在のブラウザタブで Systems Manager 接続を開きます。

> **Note**
>
> インスタンスを起動したばかりで、Systems Manager に接続するオプションが表示されない場合は、オプションが利用可能になるまで数分待つ必要がある場合があります。

ターゲットインスタンスが実行されていない

ターミナルまたはリモート接続から Amazon EC2 インスタンスに接続するには、インスタンスが実行されている必要があります。Toolkit からインスタンスに接続しようとする前に AWS 、 AWS Explorer、 AWS マネジメントコンソールまたは からインスタンスを起動します AWS Command Line Interface。

ターゲットインスタンスに IAM ロールがないか、IAM ロール のアクセス許可が不適切

Amazon EC2 インスタンスに接続するには、適切なアクセス許可がアタッチされた IAM ロールが必要です。IAM ロールがアタッチされていないインスタンスに接続しようとすると、VS Code によって通知されます。

IAM ロールがあるものの、必要なアクセス許可がないインスタンスに接続しようとすると、必要な最小限のアクションをインラインポリシーとして既存の IAM ロールに追加するように求められます。インラインポリシーを更新すると、インスタンスに接続されます。IAM ロール、アクセス許可、およびインスタンスへのロールのアタッチの詳細については、「Amazon Elastic Compute Cloud ユーザーガイド」の「Amazon EC2 の IAM ロール」トピックと、「AWS 「Systems Manager ユーザーガイド」の「ステップ 2: Session Manager のインスタンスアクセス許可の確認または追加」トピックを参照してください。

次の例には、最低限必要なアクションが含まれています。

```
"ssmmessages:CreateControlChannel",
"ssmmessages:CreateDataChannel",
"ssmmessages:OpenControlChannel",
"ssmmessages:OpenDataChannel",
"ssm:DescribeAssociation",
"ssm:ListAssociations",
"ssm:UpdateInstanceInformation
```

> **Note**
>
> 必要なアクセス許可は、次の AWS 管理ポリシーに含まれています。
>
> - `AmazonSSMManagedEC2InstanceDefaultPolicy`

- `AmazonSSMManagedInstanceCore`

ターゲットインスタンスで Systems Manager エージェントが実行されていない

この問題は、さまざまな理由で発生する可能性があります。この問題を解決するには、まずインスタンスを再起動し、接続を再試行します。または、SSM 以外の接続方法を使用して初期接続を手動で開始します。Systems Manager の詳細については、「AWS Systems Manager」の「Systems Manager エージェントの使用」トピックを参照してください。

起動時に、Amazon EC2 ステータスは実行中であることを示すが、接続されない

インスタンスの新しい IAM ロールを開始または作成したばかりで、接続を確立できない場合は、数分待ってから接続を再試行してください。

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 を統合し、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。

トピック

- Amazon Elastic Container Registry の使用
- Amazon ECR を使用した App Runner サービスの作成

Amazon Elastic Container Registry の使用

Amazon Elastic Container Registry (Amazon ECR) サービスには、VS Code の AWS Explorer から直接アクセスし、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できます。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。

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 ユーザーの作成

Amazon ECR などの AWS サービスにアクセスするには、認証情報を指定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。代わりに、 AWS Identity and Access Management (IAM) を使用して IAM ユーザーを作成し、そのユーザーを管理権限を持つ IAM グループに追加します。その後、特別な URL と IAM ユーザーの認証情報 AWS を使用して にアクセスできます。

にサインアップした AWS が、自分で IAM ユーザーを作成しなかった場合は、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	目的	方法	以下の操作も可能
IAM アイデンティティセンター内 (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ラクティス」を参照してください。	AWS IAM アイデンティティセンター 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 を使用する AWS CLI ように を設定 AWS IAM アイデンティティセンターして、プログラムによるアクセスを設定します。
IAM 内 (非推奨)	長期認証情報を使用して AWSにアクセスします。	IAM ユーザーガイドの「緊急アクセス用の IAM ユーザーを作成する」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を管理する」の手順に従って、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 コンソールからサインアウトし、次の URL を使用します。次の URL では、your_aws_account_id はハイフンのない AWS アカウント番号です (たとえば、 AWS アカウント番号が の場合1234-5678-9012、 AWS アカウント ID は です123456789012)。

```
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「your_user_name @ your_aws_account_id」が表示されます。

サインインページの URL に AWS アカウント ID を含めない場合は、アカウント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。

CLI AWS バージョン 2 のインストールと設定

CLI バージョン AWS 2 ユーザーガイドから任意のオペレーティングシステムを選択して、CLI バージョン 2 AWS をインストールおよび設定します。

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 メニューを展開します。
3. AWS アカウントに関連付けられているデフォルトの AWS リージョンを見つけます。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リ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 CLI で AWS Explorer を使用すると、以下を実行できます。

- 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. CLI AWS 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。

のインスタンスで Docker を認証したら AWS、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
```

タグ付けされた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 メニューに表示されます。

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。最初の方法は AWS Explorer を使用することです。

1. AWS Explorer から ECR メニューを展開します。
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。

CLI AWS を使用してイメージを削除する

- 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。最初の方法は AWS Explorer を使用することです。

1. AWS Explorer から ECR メニューを展開します。
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。

CLI から Amazon ECR AWS 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 ECR を使用した App Runner サービスの作成

次のトピックでは、 で Amazon Elastic Container Registry (Amazon ECR) ノードから AWS App Runner サービスを作成して起動する方法について説明します AWS Toolkit for Visual Studio Code。AWS App Runner および Amazon ECR サービスの詳細については、AWS App Runner「」および「Amazon ECR ユーザーガイド」を参照してください。

前提条件

Toolkit で Amazon ECR AWS App Runner から を作成して起動する前に AWS 、以下を完了する必要があります。これらの手順を完了する方法の詳細については、本ユーザーガイドの「Amazon Elastic Container Registry の使用」トピックを参照してください。

1. dockerfile を作成します。
2. dockerfile からイメージをビルドします。
3. 新しいレポジトリを作成します。
4. リポジトリにプッシュするイメージにタグを付けます。

既存の Amazon ECR リポジトリからの AWS App Runner サービスの作成

次の手順では、 AWS Toolkit で既存の Amazon ECR リポジトリから AWS App Runner サービスを作成する方法について説明します。

1. AWS Explorer から、 AWS App Runner サービスを作成する Amazon ECR リポジトリを含むリージョンを展開します。
2. Amazon ECR サービスノードを展開して、Amazon ECR リポジトリを表示します。
3. AWS App Runner サービスを作成する Amazon ECR リポジトリまたはリポジトリイメージのコンテキストメニューを開きます (右クリック) 。
4. コンテキストメニューから App Runner Service の作成を選択し、VS Code で AWS App Runner 作成ウィザードを開きます。
5. [Enter a port for the new service (1/5)] で、使用するポート番号を入力し、 **Enter** を押して続行します。
6. [Configure environment variables (2/5)] から [Use file...] を選択してローカルファイルを参照するか、[Skip] を選択してこのステップをスキップします。
7. [Select a role to pull from ECR (3/5)] で、リストから既存の IAM ロールを選択します。

   > **Note**
   >
   > Amazon ECR プライベートレジストリから AWS App Runner サービスを作成するには、AppRunnerECRAccessRole アクセスロールが必要です。リストに有効なロールがない場合は、[+ (Create Role...)] アイコンを選択して AppRunnerECRAccessRole を自動的に作成し、レジストリに割り当てます。

8. [Name your service (4/5)] で、新しいサービスの名前を入力し、 **Enter** を押して続行します。
9. [Select instance configuration (5/5)] で、リストから **vCPU** および **Memory** 設定を選択して新しいサービスを作成します。
10. AWS Explorer から App Runner サービスノードを展開して AWS App Runner 、リソースを表示します。新しいサービスが正常に作成されると、ステータスは自動的に [実行中] に更新されます。

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。をインストールすると AWS Toolkit for Visual Studio Code、インストールには Amazon ECS タスク定義ファイルの IntelliSense 機能が含まれます。

前提条件

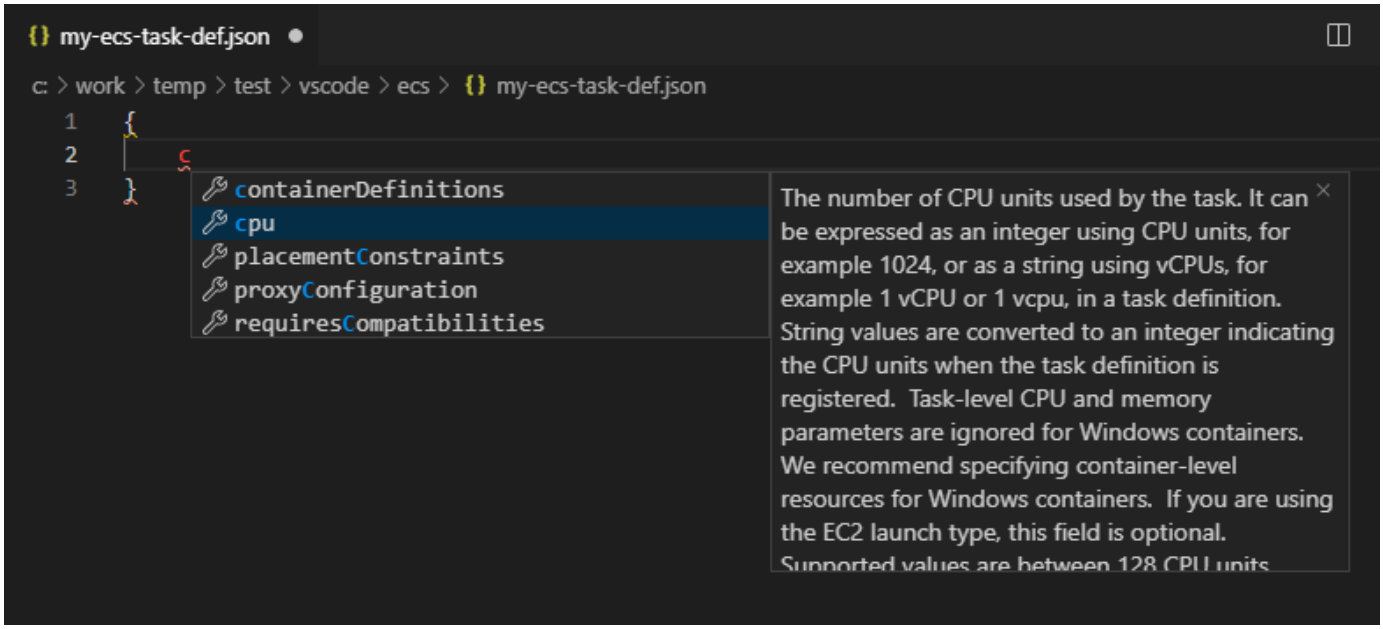
- システムが、「Toolkit for VS Code をインストールする」で指定されている前提条件を満たしていることを確認してください。

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 アカウント内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どうかに応じて AWS 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 スキーマの使用

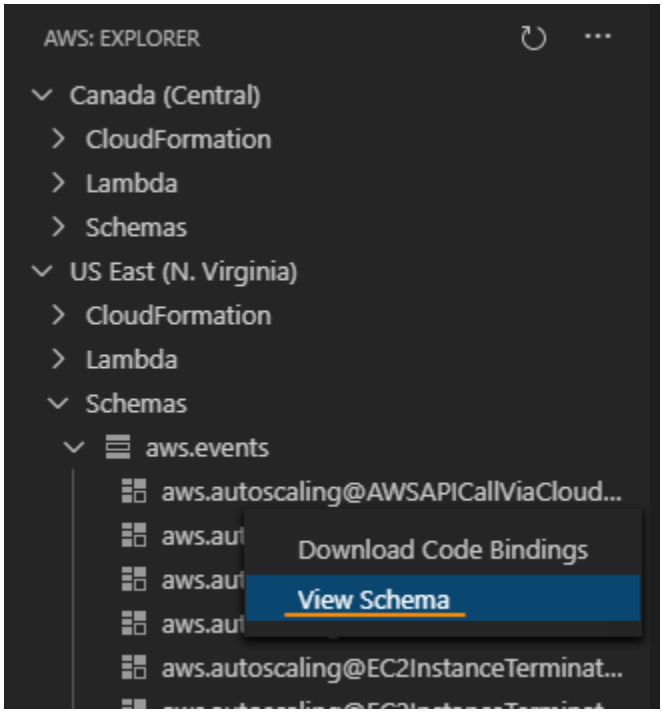
AWS Toolkit for Visual Studio Code (VS Code) を使用して、Amazon EventBridge スキーマでさまざまなオペレーションを実行できます。

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 EventBridge ユーザーガイド」の「Amazon EventBridge スキーマ」を参照してください。

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。たとえば、 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。



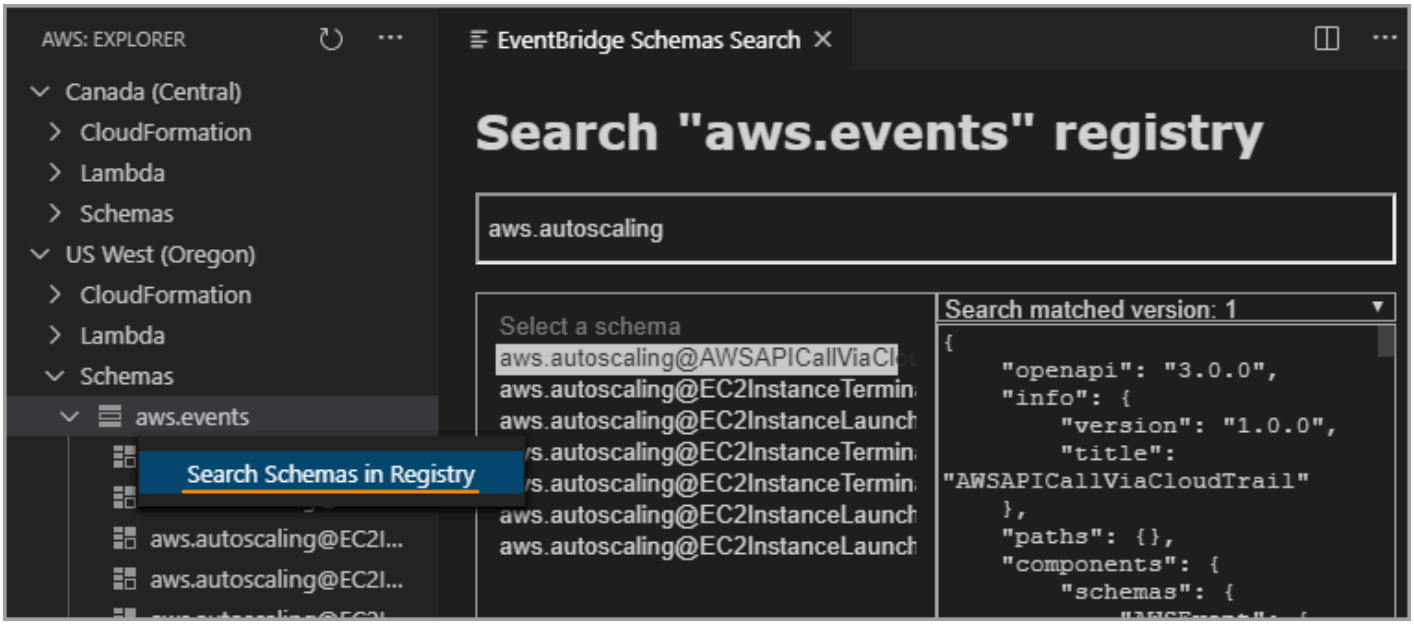
使用可能なスキーマの検索

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 スキーマの検索] 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、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。 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ポリシー検証の検出結果には、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。

カスタムポリシーチェックを実行して、セキュリティ標準に基づいて新しいアクセスをチェック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「ポリシーを検証するためのチェック」トピックを参照してください。

以下のトピックでは、 AWS Toolkit for Visual Studio Code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。

トピック

- IAM Access Analyzer AWS の使用

IAM Access Analyzer AWS の使用

以下のセクションでは、 AWS Toolkit for Visual Studio Codeで IAM ポリシー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。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」および「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

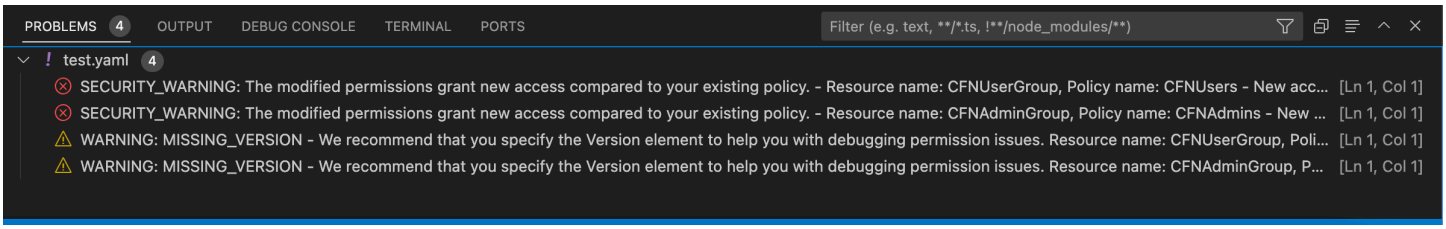
ツールキットから IAM Access Analyzer ポリシーチェックを使用する前に、次の前提条件を満たす必要があります。

- Python バージョン 3.6 以降をインストールします。
- Python CLI ツールで必要であり、IAM ポリシーチェックウィンドウで指定されている CloudFormation用の IAM Policy Validatorまたは Terraform 用の IAM Policy Validatorのいずれかをインストールします。
- AWS ロールの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、 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。チェックの検出結果は VS

Code の [Problems Panel] に表示されます。次の図は、VS Code の [Problems Panel] を示しています。



IAM Access Analyzer には 4 種類のチェックがあります。

- ポリシーの検証
- CheckAccessNotGranted
- CheckNoNewAccess
- CheckNoPublicAccess

次のセクションでは、各種類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 ロールの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、 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。
> ファイルパス参照は通常、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 となります。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。
> 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 の料金ガイドを参照してください。

ポリシー検証の実行

ポリシーの検証チェックは、ポリシーの検証とも呼ばれ、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 AWS エディタで 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 ポリシーチェックを開くには、**CRTL+Shift+P** を押して VS Code コマンドパレットを開き、**IAM Policy Checks** を検索し、クリックして VS Code エディタの [IAM Policy Checks] ペインを開きます。
3. [IAM Policy Checks] 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. [Validate Policies] セクションから、[Run Policy Validation] 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 [Problems Panel] から、ポリシーチェックの検出結果を確認します。
6. ポリシーを更新し、この手順を繰り返します。ポリシーチェックの結果にセキュリティ警告やエラーが表示されなくなるまで、ポリシーの検証チェックを再実行します。

CheckAccessNotGranted の実行

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**
>
> ファイルパス参照は通常、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 となります。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。少なくとも 1 つのアクションまたはリソースを指定し、次の例にしたがってファイルを構成する必要があります。
>
> ```
> {"actions": ["action1", "action2", "action3"], "resources":
> ["resource1", "resource2", "resource3"]}
> ```

1. VS Code から、VS Code AWS エディタで 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 ポリシーチェックを開くには、**CRTL+Shift+P** を押して VS Code コマンドパレットを開き、**IAM Policy Checks** を検索し、クリックして VS Code エディタの [IAM Policy Checks] ペインを開きます。

3. [IAM Policy Checks] 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. [Custom Policy Checks] セクションで、CheckAccessNotGranted を選択します。
5. テキスト入力フィールドに、アクションとリソースの ARN を含むカンマ区切りのリストを入力できます。少なくとも 1 つのアクションまたはリソースを指定する必要があります。
6. [Run Custom Policy Check] ボタンを選択します。
7. VS Code [Problems Panel] から、ポリシーチェックの検出結果を確認します。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、この手順を繰り返して、PASS が返されるまで CheckAccessNotGranted チェックを再実行します。

CheckNoNewAccess の実行

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していないかを確認するカスタムポリシーチェックです。

1. VS Code から、VS Code AWS エディタで 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 ポリシーチェックを開くには、**CRTL+Shift+P** を押して VS Code コマンドパレットを開き、**IAM Policy Checks** を検索し、クリックして VS Code エディタの [IAM Policy Checks] ペインを開きます。
3. [IAM Policy Checks] 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. [Custom Policy Checks] セクションで、[CheckNoNewAccess] を選択します。
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。
6. 参照ドキュメントのタイプに一致する参照ポリシータイプを選択します。
7. [Run Custom Policy Check] ボタンを選択します。
8. VS Code [Problems Panel] から、ポリシーチェックの検出結果を確認します。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、この手順を繰り返して、PASS が返されるまで CheckNoNewAccess チェックを再実行します。

CheckNoPublicAccess の実行

CheckNoPublicAccess は、テンプレート内のサポートされているリソースタイプに対して、ポリシーがパブリックアクセスを付与していないかを検証するためのカスタムポリシーチェックです。

サポートされているリソースタイプの詳細については、cloudformation-iam-policy-validator および terraform-iam-policy-validator GitHub リポジトリを参照してください。

1. VS Code から、VS Code AWS エディタで 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 ポリシーチェックを開くには、**CRTL+Shift+P** を押して VS Code コマンドパレットを開き、**IAM Policy Checks** を検索し、クリックして VS Code エディタの [IAM Policy Checks] ペインを開きます。
3. [IAM Policy Checks] 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. [Custom Policy Checks] セクションで、[CheckNoPublicAccess] を選択します。
5. [Run Custom Policy Check] ボタンを選択します。
6. VS Code [Problems Panel] から、ポリシーチェックの検出結果を確認します。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。
7. ポリシーを更新し、この手順を繰り返して、`PASS` が返されるまで CheckNoNewAccess チェックを再実行します。

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

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

AWS Toolkit for Visual Studio Code は AWS Lambda 関数の包括的なサポートを提供するため、VS Code から直接ビルド、テスト、デプロイできます。

Lambda は、200 を超える AWS サービスと software-as-a-service (SaaS) アプリケーションのイベントに応じてコードを自動的に実行する、完全マネージド型のイベント駆動型コンピューティングサービスです。AWS Lambda サービスの詳細については、「AWS Lambda デベロッパーガイド」を参照してください。

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

トピック

AWS Lambda 関数の使用

AWS Toolkit for Visual Studio Code を使用すると、ローカル VS Code 環境で AWS Lambda 関数を操作できます。Toolkit を使用すると AWS 、IDE を離れることなく、Lambda 関数を作成、編集、テスト、デバッグ、デプロイできます。 AWS Lambda サービスの詳細については、 AWS Lambdaデベロッパーガイドを参照してください。

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

> **Note**
>
> を使用して Lambda 関数を既に作成している場合は AWS マネジメントコンソール、 Toolkit から呼び出すことができます。さらに、 から Lambda 関数を VS Code に開くことができます。詳細については AWS Lambda console、このユーザーガイドのAWS Lambda console IDE へトピックを参照してください。VS Code で新しい Lambda 関数を作成するには、このユーザーガイドの「新しいサーバーレスアプリケーションの作成 (ローカル)」トピックで説明されているステップに従ってください。

前提条件

AWS Toolkit で AWS Lambda サービスを使用するには、次の条件を満たす必要があります。

- の最新バージョン AWS Toolkit for Visual Studio Code がインストールされ、認証情報を使用してセットアップされます AWS 。
- AWS Identity and Access Management (IAM) 管理のアクセス許可とポリシーは、 AWS Lambda サービスと連携するように設定されています。アクセス許可を設定し、互換性のある AWS 管理ポリシーを作成する方法の詳細については、「 AWS Lambda デベロッパーガイド」のAWS Identity and Access Management 「 for AWS Lambda 」トピックを参照してください。
- 既存の AWS Lambda 関数がある、または作成方法に精通している。Lambda 関数を作成する方法については、 AWS Lambda デベロッパーガイドの「最初の Lambda 関数を作成する」トピックを参照してください。

Lambda 関数の呼び出し

AWS アカウントから VS Code に Lambda 関数を呼び出すには、次の手順を実行します。

1. から AWS Toolkit for Visual Studio Code、 AWS エクスプローラーを展開します。
2. AWS エクスプローラーから、Lambda を展開して Lambda リソースを表示します。
3. 呼び出す Lambda 関数のコンテキストメニューを開き (右クリック)、[クラウドで呼び出し] を選択するか、[クラウドで呼び出し] アイコンを選択して、VS Code で [Remote invoke configuration] メニューを開きます。
4. [Remote invoke configuration] メニューから、ペイロード設定を指定し、イベントに必要な情報を追加します。

   > **Note**
   >
   > 最初の呼び出しプロセスは、 AWS エクスプローラーのクラウドで呼び出しを選択するとすぐに実行を開始できます。VS Code ターミナルの OUTPUT タブに出力が表示されます。

5. [Remote Invoke] ボタンを選択して関数を呼び出します。VS Code ターミナルの OUTPUT タブに出力が表示されます。

Lambda 関数の削除

Lambda 関数を削除するには、次の手順を実行します。

> **Warning**
>
> この手順は、CloudFormation に関連付けられている Lambda 関数の削除には使用しないでください。これらの関数は、 CloudFormation スタックを通じて削除する必要があります。

1. から AWS Toolkit for Visual Studio Code、 AWS エクスプローラーを展開します。
2. AWS エクスプローラーから、Lambda を展開して Lambda リソースを表示します。
3. 削除する Lambda 関数を右クリックし、[削除] を選択します。
4. プロンプトが表示されたら、関数を削除することに同意します。

関数が削除されると、 AWS エクスプローラーに表示されなくなります。

Lambda 関数のダウンロード

リモート Lambda 関数から VS Code ワークスペースにコードをダウンロードして、編集とデバッグを行うことができます。

> **Note**
>
> Lambda 関数をダウンロードするには、アクセス可能なフォルダを持つ VS Code ワークスペースで作業する必要があり、 AWS Toolkit は Node.js および Python ランタイムを使用する Lambda 関数でのみこの機能をサポートします。

1. から AWS Toolkit for Visual Studio Code、 AWS エクスプローラーを展開します。
2. AWS エクスプローラーから、Lambda を展開して Lambda リソースを表示します。
3. ダウンロードする Lambda 関数を右クリックし、[ダウンロード] を選択します。
4. Lambda 関数が VS Code エディタで開き、ダウンロードが完了すると AWS エクスプローラーに表示されます。 AWS Toolkit は、VS Code 実行パネルにも起動設定を作成し、 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るようにします AWS Serverless Application Model。の使用の詳細については AWS SAM、「」を参照してください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)”。

新しい Lambda 関数の更新のデプロイ

ローカルマシンの指定されていない一時的な場所から、新しい Lambda 関数に更新をデプロイできます。

> **Note**
>
> Lambda ファイルにデプロイされていない変更がある場合、VS Code エディタと AWS エクスプローラーで変更されたファイルの横に表示される [M] アイコンによって通知されます。

VS Code エディタからのデプロイ

1. VS Code エディタで Lambda 関数からファイルを開き、ファイルを変更します。
2. VS Code のメインメニューから手動で保存するか、 **option+s** (Mac) または **ctrl+s** (Windows) を押します。
3. VS Code に、クラウドへの変更のデプロイを求めるプロンプトが表示されます。Deploy ボタンを選択してデプロイに同意します。
4. VS Code にデプロイの最新ステータスが表示され、プロセスが完了すると通知します。

AWS Explorer からのデプロイ

1. VS Code エディタで Lambda 関数からファイルを開き、ファイルを変更します。
2. Toolkit から AWS 、 AWS エクスプローラーを展開します。
3. AWS エクスプローラーから、変更をデプロイする Lambda 関数を使用して AWS リージョンを展開します。
4. AWS リージョンから Lambda を展開し、変更をデプロイする関数に移動します。
5. 関数の横にあるクイックメニューから、[コードの保存とデプロイ] アイコンを選択します。
6. VS Code にデプロイの最新ステータスが表示され、プロセスが完了すると通知します。

既存の Lambda 関数の更新のアップロード

次の手順では、既存の Lambda 関数に加えられたローカルでの変更をアップロードする方法について説明します。この機能は、Lambda がサポートするすべてのランタイムによるアップロードをサポートします。

> **Warning**
>
> Lambda 関数をアップロードする前に、次の点に注意してください。
>
> - この方法でコードを更新しても、デプロイに AWS SAM CLI を使用したり、CloudFormation スタックを作成したりすることはありません。
> - Toolkit AWS はコードを検証しません。クラウドに変更をアップロードする前にコードを検証し、関数をテストしてください。

Zip アーカイブのアップロード

1. から AWS Toolkit for Visual Studio Code、 AWS エクスプローラーを展開します。
2. AWS エクスプローラーから、Lambda を展開して Lambda リソースを表示します。
3. 変更をアップロードする Lambda 関数を右クリックし、[Lambda のアップロード...] を選択して [アップロードタイプの選択] メニューを開きます。
4. [ZIP アーカイブ] を選択して、ローカルディレクトリ内の `ZIP Archive` を見つけます。
5. プロンプトが表示されたら、アップロードの内容を確認して、選択した `ZIP Archive` のアップロードを開始します。
6. VS Code にアップロードのステータスが表示され、アップロードプロセスが完了すると通知されます。

ビルドせずにディレクトリをアップロードする

1. から AWS Toolkit for Visual Studio Code、 AWS エクスプローラーを展開します。
2. AWS エクスプローラーから、Lambda を展開して Lambda リソースを表示します。
3. 変更をアップロードする Lambda 関数を右クリックし、[Lambda のアップロード...] を選択して [アップロードタイプの選択] メニューを開きます。
4. [ディレクトリ] を選択して、[ディレクトリのビルド] 画面に進みます。
5. [ディレクトリのビルド] 画面で、[いいえ] を選択してアップロードするローカルディレクトリを選択します。
6. プロンプトが表示されたら、アップロードの内容を確認して、選択したディレクトリをアップロードします。
7. VS Code にアップロードのステータスが表示され、アップロードプロセスが完了すると通知されます。

ビルドでディレクトリをアップロードする

> **Note**
>
> 以下の点に注意してください。
>
> - この手順には CLI AWS Serverless Application Model が必要です。
> - Toolkit は、アップロード前に一致するハンドラーを検出できないこと AWS を通知します。
> - Lambda 関数にアタッチされているハンドラーを変更するには、 AWS Lambda console または を使用します AWS Command Line Interface。

1. から AWS Toolkit for Visual Studio Code、 AWS エクスプローラーを展開します。
2. AWS エクスプローラーから、Lambda を展開して Lambda リソースを表示します。
3. 変更をアップロードする Lambda 関数を右クリックし、[Lambda のアップロード...] を選択して [アップロードタイプの選択] メニューを開きます。
4. [ディレクトリ] を選択して、[ディレクトリのビルド] 画面に進みます。
5. [ディレクトリのビルド] 画面で [はい] を選択し、アップロードするローカルディレクトリを選択します。
6. プロンプトが表示されたら、アップロードの内容を確認して、選択したディレクトリのビルドとアップロードを開始します。
7. VS Code にアップロードのステータスが表示され、アップロードプロセスが完了すると通知されます。

Lambda 関数を AWS SAM プロジェクトに変換する

Lambda 関数を AWS SAM スタックに変換するには、次の手順を実行します。

> **Warning**
>
> 現在、Lambda 関数を AWS SAM プロジェクトに変換する場合にサポートされているのは一部のリソースのみです。変換後に不足しているリソースを見つけるには、Lambda コンソールを確認し、 AWS SAM テンプレートに手動で追加します。サポートされているリソースとサポートされていないリソースの詳細については、「 AWS CloudFormation デベロッパーガイド」の「リソースタイプのサポート」トピックを参照してください。

1. Toolkit から AWS 、 AWS エクスプローラーを展開します。
2. AWS エクスプローラーから、 AWS SAM プロジェクトに変換する Lambda 関数を使用して AWS リージョンを展開します。
3. AWS リージョンから Lambda を展開し、 AWS SAM スタックに変換する関数に移動します。
4. Lambda 関数の横にあるクイックメニューから、[SAM アプリケーションに変換] アイコンを選択してローカルファイルシステムを参照し、新しい AWS SAM プロジェクトの場所を指定します。
5. AWS Toolkit が Lambda 関数の AWS SAM プロジェクトへの変換を開始する場所を指定すると、VS Code はプロセスのステータスに関する更新を提供します。

   > **Note**
   >
   > このプロセスには数分かかることがあります。

6. VS Code からプロンプトが表示されたら、スタック名を入力し、**Enter** キーを押して続行します。
7. VS Code はプロジェクトのステータスを引き続き更新し、プロセスが完了すると に通知し、新しい AWS SAM プロジェクトを VS Code ワークスペースとして開きます。

AWS Lambda console IDE へ

IDE AWS Lambda console への 機能を使用すると、 から VS Code AWS Lambda console に AWS Lambda 関数をダウンロードできます。VS Code で Lambda 関数を使用すると、 AWS Serverless Application Model (AWS SAM) や などの他のローカル開発オプションにアクセスできます AWS Cloud Development Kit (AWS CDK)。

詳細については AWS Lambda、「 AWS Lambdaデベロッパーガイド」を参照してください。 AWS Toolkit で Lambda 関数の使用を開始するには、このユーザーガイドのAWS Lambda 「関数の使用」トピックを参照してください。以下のセクションでは、ワークフローを Lambda コンソールから VS Code に移動する方法について説明します。Lambda コンソールの使用を開始する方法を含む、Lambda 関数を Lambda コンソールから VS Code に移動する方法の詳細については、「AWS Lambda デベロッパーガイド」の「VS Code を使用した Lambda 関数のローカル開発」トピックを参照してください。

コンソールからローカル開発への移行

VS Code で Lambda 関数を Lambda コンソールから開くには、次の手順を実行します。

1. ブラウザで Lambda コンソールを開きます。
2. Lambda コンソールから、VS Code で開く関数を選択します。
3. 関数ビューから、[コードソース] タブに移動します。
4. [コードソース] タブから、[VS Code で開く] を選択します。

VS Code での Lambda 関数の使用

Lambda コンソールを介して VS Code で Lambda 関数が開いた場合、以下のことが起こります。

- VS Code がローカルマシンで自動的に起動します。
- Lambda 関数が VS Code ワークスペースとして開きます。
- Lambda `handler file` が VS Code エディタで開きます。

> **Note**
>
> ワークスペースに正しく設定された `handler file` がない場合、VS Code エディタでファイルが開きます。

Lambda コンソールを使用して VS Code で Lambda 関数を開くと、フル言語サポート、ローカルテスト、リモートデバッグ、デプロイサポート、依存関係管理を使用して関数コードを編集する機能など、既存の AWS Toolkit Lambda のすべての機能にアクセスできます。Toolkit でサポートされている Lambda 機能の詳細については AWS 、このユーザーガイドAWS Lambdaのサービス目次を参照してください。

AWS Lambda LocalStack をサポートする

AWS Toolkit for Visual Studio Codeでの LocalStack のサポートによって、サーバーレスアプリケーションを構築、テスト、デバッグします。LocalStack は、サーバーレスアプリケーションのローカルテストを可能にする AWS クラウドエミュレータです。

詳細については AWS Lambda、「 AWS Lambdaデベロッパーガイド」を参照してください。LocalStack の詳細については、LocalStack のウェブサイトを参照してください。

前提条件

以下は、VS Code で LocalStack を使用するための前提条件です。

> **Note**
>
> LocalStack CLI はセットアッププロセス中にインストールされますが、別のバージョンの LocalStack CLI を使用する場合、最低でもバージョン 4.8.0 が必要です。

- 無料および有料の LocalStack 階層で使用できるすべての機能にアクセスするには、LocalStack ウェブアプリケーションアカウントが必要です。LocalStack Community エディションは、 アカウントなしで使用できます。
- VS Code で LocalStack を使用するには Docker が必要です。Docker の LocalStack 要件の詳細については、LocalStack ドキュメントの「LocalStack Docker イメージ」トピックを参照してください。
- 推奨: AWS Command Line Interface (AWS CLI) は、シミュレートされたクラウド環境でのサービスの使用を支援します。

LocalStack のインストール

LocalStack の無料および有料階層バージョンをインストールするには、次の手順を実行します。

> **Note**
>
> LocalStack Community エディションをセットアアップする方法については、このトピックの「LocalStack のセットアップ」セクションの「LocalStack Community」コンテンツを参照してください。

1. Toolkit から AWS 、APPLICATION BUILDER エクスプローラーを展開します。
2. [チュートリアルを選択] ボタンを選択して、VS Code エディタの [Get started building your application] のチュートリアルタブタブを開きます。
3. チュートリアルから [Install LocalStack] を選択して、VS Code で LocalStack のインストールプロセスを開始します。

LocalStack のセットアップ

VS Code の LocalStack 拡張機能をインストールすると、セットアップが必要な場合に次のいずれかのインジケータが表示されることがあります。

- デフォルトでは IDE の左下隅にある VS Code ステータスバーで、LocalStack のステータスが赤になります。
- VS Code に LocalStack を設定するようにプロンプトが表示されます。

LocalStack のセットアップと設定には、使用している LocalStack のバージョンに応じて 2 種類あります。以下のタブ付きセクションでは、各 LocalStack セットアッププロセスについて説明します。

> **Note**
>
> LocalStack の無料および有料階層バージョンには LocalStack 認証トークンが必要です。LocalStack の料金の詳細については、料金ガイドである「プランの選択」を参照してください。

LocalStack の無料階層と有料階層

LocalStack をセットアップする方法は 2 つあります。

- VS Code の [Setup LocalStack to get started] プロンプトで、[Setup] ボタンを選択します。
- VS Code ステータスバーから、LocalStack ステータスアイコンを選択して [Setup LocalStack to get started] プロンプトを開き、[Setup] ボタンを選択します。

セットアップ中、システムは次のステップを実行します。

1. LocalStack CLI をインストールします。
2. LocalStack アカウントがあるかどうかをチェックします。
3. LocalStack アカウントをお持ちの場合、システムはデフォルトのウェブブラウザで認証プロセスをガイドします。同様に、LocalStack アカウントがない場合、システムは認証プロセスの前にアカウントのセットアップをガイドします。

LocalStack のセットアップが完了すると、VS Code ステータスバーの LocalStack ステータスが更新されます。

> **Note**
>
> LocalStack のプロファイルを作成していない場合は、LocalStack のセットアッププロセスの一環として新しい AWS プロファイルが自動的に作成されます。

LocalStack Community

LocalStack Community エディションは無料で使用でき、アカウントにサインアップする必要はありません。これは、ライセンスを必要としない Docker イメージから実行されます。LocalStack Community Edition の詳細については、「LocalStack Community イメージ」ドキュメントを参照してください。以下のセクションでは、VS Code で LocalStack Community エディションを使用するために必要な前提条件と基本的なセットアップについて説明します。

新しいインスタンスの起動

LocalStack Community の新しいインスタンスを起動するには、次の手順を実行します。

> **Note**
>
> 次の例では、ポート 4566 で LocalStack のコンテナインスタンスを起動します。異なるポート値を指定する場合は、「 AWS CLI と AWS Toolkit の設定」セクションにある手順で指定されたポート値を更新する必要があります。

1. VS Code から、**ctrl + `(backtick)** を押して VS Code ターミナルを開きます。
2. ターミナルウィンドウで、以下を入力します。

   Mac:

   ```
   docker run -d --name localstack_main \
   >> -p 4566:4566 \
   >> -v /var/run/docker.sock:/var/run/docker.sock \
   >> localstack/localstack
   ```

   Windows:

   ```
   docker run -d --name localstack_main `
   >> -p 4566:4566 `
   >> -v /var/run/docker.sock:/var/run/docker.sock `
   ```

```
>> localstack/localstack
```

3. プロセスが完了すると、ターミナルで Docker インスタンスのステータスが更新されます。

LocalStack のこのコンテナ化されたインスタンスでは、ダウンロードプロセス中に指定した AWS サービスにアクセスできます。

LocalStack と Docker 用の CLI の設定。

Docker で LocalStack と連携するように AWS CLI と AWS Toolkit を設定するには、次の手順を実行して新しいプロファイルを設定します。

1. VS Code から、**ctrl + `(backtick)** を押して VS Code ターミナルを開きます。
2. ターミナルウィンドウで、以下を入力します。

```
~/.aws/credentials
[localstack]
aws_access_key_id = test
aws_secret_access_key = test
~/.aws/config
[profile localstack]
region = us-east-1
output = json
endpoint_url = http://localhost:4566 [default localstack endpoint]
```

3. AWS Toolkit は LocalStack プロファイルを検出し、接続ステータスメニューを更新します。

セットアップ後、ステータスバーのプロファイルセクションから LocalStack AWS プロファイルを選択すると、LocalStack リソースが AWS エクスプローラーに表示されます。さらに、VS Code ターミナルの [Output] タブで LocalStack ログを表示できます。

VS Code での LocalStack の開始

次のいずれかの方法で LocalStack の使用を開始できます。

VS Code ステータスバーからの LocalStack の開始

1. VS Code からステータスバーに移動し、[Start LocalStack] ボタンを選択して LocalStack を起動します。
2. LocalStack が正常に起動すると、VS Code ステータスバーが更新されます。

VS Code コマンドパレット からの LocalStack の開始

1. VS Code で **Cmd + Shift + P** (Mac) または **Control + Shift + P** (Windows) 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、検索バーに **Start LocalStack** と入力し、結果に表示されたら選択します。
3. LocalStack が正常に起動すると、VS Code ステータスバーが更新されます。

VS Code ターミナルからの LocalStack の開始

1. VS Code から、**ctrl + `(backtick)** を押して VS Code ターミナルを開きます。
2. VS Code ターミナルから、**localstack start** CLI コマンドを入力します。
3. LocalStack が正常に起動すると、VS Code ステータスバーが更新されます。

サンプルサーバーレスアプリケーションのビルド

VS Code で LocalStack の使用を開始するには、サンプルサーバーレスアプリケーションが必要です。 AWS アカウントに既存のアプリケーションがある場合は、LocalStack を使用してローカルにデプロイすることも、 AWS Serverless Land を使用して新しいアプリケーションを作成することもできます。

AWS ツールキットで Serverless Land を使用してアプリケーションを作成する方法の詳細については、本ユーザーガイドの「AWS Serverless Land の使用」トピックを参照してください。Serverless Land の詳細については、Serverless Land ウェブアプリケーションのメインランディングページを参照してください。

LocalStack を使用した Lambda 関数のテストとデバッグ

LocalStack VS Code 拡張機能での Lambda 関数のテストとデバッグは、 AWS クラウドにデプロイされた関数の操作に似ています。主な違いは、LocalStack を使用して関数をデプロイおよびデバッグするために、 AWS ツールキットインスタンスを LocalStack アカウントで認証する必要があることです。

> **Note**
>
> このセクションで説明するテストおよびデバッグ機能は、LocalStack Community エディションでは使用できません。

VS Code で LocalStack を使用するには、 AWS ツールキットの LocalStack プロファイルに接続します。LocalStack プロファイルがアクティブな場合、VS Code ステータスバーには AWS: profile:localstack (カスタムエンドポイント) とチェックマークが表示されます。

AWS Toolkit での Lambda 関数の使用の詳細については、このユーザーガイドのAWS Lambda 関数の使用トピックを参照してください。

AWS Lambda リモートデバッグ

AWS Toolkit for Visual Studio Code を使用すると、クラウドで実行されている AWS Lambda 関数を VS Code で直接デバッグできます。 AWS Lambda リモートデバッグを使用すると、既存の開発ワークフローを変更することなく、実行中の関数の検査、ブレークポイントの設定、変数の検査、ステップスルーデバッグを行うことができます。

以下のセクションでは、 AWS Toolkit for Visual Studio Codeで Lambda リモートデバッグを使用する方法について説明します。

Lambda リモートデバッグの仕組み

Toolkit は、追加の Lambda デバッグレイヤーを使用して Lambda 関数を一時的に変更し、Lambda 呼び出しタイムアウト制限を 900 秒に延長することで、リモートデバッグ AWS を有効にします。AWS IoT セキュアトンネリングを使用して、ローカルデバッガーと Lambda ランタイム環境の間に安全な接続が確立されます。この接続により、ローカルコードブレークポイントを使用して、関数がリモートで実行される際にステップ実行できます。デバッグセッションが完了すると、すべての一時的な変更は自動的に元の設定に戻ります。

開始方法

ランタイムのサポート

Lambda リモートデバッグは、次のランタイムをサポートしています。

- Python (Amazon Linux 2023)
- Java
- Typescript/JavaScript/Node.js (Amazon Linux 2023)

> **Note**
>
> Lambda マネージドインスタンスと OCI イメージ関数タイプは、Lambda リモートデバッグではサポートされていません。

前提条件

開始するには、以下の前提条件を満たしておく必要があります。

- Toolkit で有効な AWS 認証情報を設定する必要があります AWS 。 AWS Toolkit のインストールと認証情報の設定の詳細については、このユーザーガイドの「開始方法」トピックを参照してください。
- Lambda 関数が AWS アカウントにデプロイされました。Lambda 関数のデプロイの詳細については、「AWS Lambda デベロッパーガイド」の「最初の Lambda 関数の作成」トピックを参照してください。
- 関数をデバッグするには、適切な AWS Identity and Access Management (IAM) ポリシーとアクセス許可が必要です。Lambda のアクセス許可の詳細については、「AWS Lambda デベロッパーガイド」の「AWS LambdaのAWS 管理ポリシー」トピックを参照してください。以下は、 AWS ツールキットで Lambda リモートデバッグを使用するために必要な最低限のアクセス許可を含むポリシーの例です。

  > **Note**
  >
  > リモートデバッグは、 AWS AWS IoT セキュアトンネリングを通じて有効になります。これにより、ローカルデバッガーは Lambda ランタイム環境への安全な接続を確立できます。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "lambda:ListFunctions",
        "lambda:GetFunction",
        "lambda:GetFunctionConfiguration",
```

```
        "lambda:GetLayerVersion",
        "lambda:UpdateFunctionConfiguration",
        "lambda:InvokeFunction",
        "lambda:PublishVersion",
        "lambda:DeleteFunction",
        "iot:OpenTunnel",
        "iot:RotateTunnelAccessToken",
        "iot:ListTunnels"
      ],
      "Resource": "*"
    }
  ]
}
```

Lambda リモートデバッグへのアクセス

AWS Toolkit で Lambda リモートデバッグにアクセスするには、 AWS エクスプローラーまたは Application Builder エクスプローラーの 2 つの主要なパスがあります。 AWS エクスプローラーから、ノードを介して AWS Lambda Lambda リモートデバッグにアクセスできます。アプリケーションビルダーエクスプローラーから、ローカル AWS SAM プロジェクトを通じて Lambda リモートデバッグにアクセスできます。

AWS エクスプローラーからの Lambda リモートデバッグへのアクセス

1. VS Code から Toolkit AWS 拡張機能を開きます。
2. Toolkit から AWS 、 AWS エクスプローラーを展開します。
3. エクスプローラーから、[Lambda] ノードを展開します。
4. デバッグする関数に移動し、コンテキストメニューから [リモート呼び出し] アイコンを選択して [リモート呼び出し設定] 画面を開きます。

アプリケーションビルダーエクスプローラーからの Lambda リモートデバッグへのアクセス。

1. VS Code から Toolkit AWS 拡張機能を開きます。
2. AWS Toolkit から、Application Builder Explorer を展開します。
3. エクスプローラーから、デバッグする Lambda プロジェクトを含む AWS SAM プロジェクトを展開します。
4. デバッグする、デプロイされた Lambda 関数を展開します。

5. リモート関数に移動し、コンテキストメニューから [リモート呼び出し] アイコンを選択して [リモート呼び出し設定] 画面を開きます。

Lambda リモートデバッグの使用

以下のセクションでは、 AWS Toolkit for Visual Studio Codeで Lambda リモートデバッグを使用する方法について説明します。

> **Note**
>
> Lambda 関数には、関数コードとアタッチされたすべてのレイヤーに対して、5 つのレイヤーと合計 250 MB の制限があります。Lambda リモートデバッグを実行するには、少なくとも 1 つの空きレイヤーが必要です。

デバッグセッションの設定

開始する前に、次の手順を実行してデバッグセッションを設定します。

1. AWS エクスプローラーからの Lambda リモートデバッグへのアクセスまたは前のセクションにある Application Builder エクスプローラーからの Lambda リモートデバッグへのアクセスの手順を完了して、リモート呼び出し設定メニューを開きます。
2. [リモート呼び出し設定] メニューから、[リモートデバッグ] チェックボックスを選択してリモートデバッグプロパティを表示します。
3. ローカルハンドラーファイルへのローカルルートパスを指定します。

   > **Note**
   >
   > ローカルルートパスは、デプロイされた Lambda 関数に一致するソースコードの場所です。アプリケーションビルダーエクスプローラーでデプロイされた関数から作業している場合、ローカルルートパスが自動的に検出されます。
   > ローカルに保存されたソースコードがない場合は、[リモートコードのダウンロード] ボタンを選択して Lambda 関数のソースコードを取得します。これにより、VS Code エディタで `handler file` が開きます。

4. [Payload] セクションで、テストイベントデータを取得する場所を指定します。

ブレークポイントの設定とデバッグ

ブレークポイントを設定し、次の手順を実行してデバッグを開始します。

1. VS Code エディタの `handler file` から、余白部分をクリックして、デバッグを一時停止する行番号にブレークポイントを設定します。
2. ブレークポイントに問題がなければ、[リモート呼び出し設定] メニューに戻り、設定が正しく構成されていることを確認します。次に [リモート呼び出し] ボタンを選択してデバッグを開始します。
3. AWS Toolkit は、Lambda 関数をデバッグ機能で更新し、デバッグセッションのセキュアトンネルを確立し、指定されたペイロードで関数を呼び出し、ブレークポイントに達するとプロセスを一時停止します。
4. ブレークポイントの一時停止時に、[実行およびデバッグ] ペインを使用して [変数]、[呼び出しスタック]、[ブレークポイント] を表示します。

関数の更新とテスト

クイックデプロイでコードを変更し、変更をテストするには、次の手順を実行します。

1. デバッグセッションをアクティブにした状態で、VS Code エディタで `handler file` を変更します。
2. 変更内容を保存します (**`Command+S on macOS`**、**`Ctrl+S on Windows`**)。
3. プロンプトが表示されたら、変更をデプロイすることに同意します。Toolkit AWS は、変更されたコードで Lambda 関数を更新します。
4. 新しいブレークポイントを設定し、[リモート呼び出し] ボタンを再度選択して、変更のデバッグとテストを続行します。

> **Note**
>
> または、VS Code デバッグコントロールで [Attach debugger] オプションの選択を解除し、[リモート呼び出し] ボタンを選択してデバッグなしで関数を実行できます。

デバッグセッションの終了

次の各オプションにより、リモートデバッグセッションが終了し、プロジェクトからデバッグレイヤーが削除されます。

- [リモート呼び出し設定] 画面から [デバッグセットアップを削除] を選択します。
- VS Code のデバッグコントロールで [disconnect] アイコンを選択します。
- VS Code エディタで `handler file` を閉じます。

> **Note**
>
> 以下の情報を記録します。
>
> - Lambda デバッグレイヤーは、60 秒間非アクティブ状態が続くと自動的に削除されます。最後の呼び出しの完了時からカウントが開始されます。
> - デバッグプロセス中に infrastructure-as-code (IaC) マネージド (AWS SAM、 AWS CDK、Terraform) 関数にコード変更を加えた場合は、ローカルプロジェクトに保存し、ソースコントロールリポジトリの更新を検討してください。保存されていない変更は、IaC 関数が再デプロイされると上書きされます。
> - デバッグのみを目的として一時的な変更を加えた場合は、ソースコントロールから関数を再デプロイして、本番コードと一致するようにすることを推奨します。

ソースマップを使用した TypeScript Lambda 関数のデバッグ

以下のセクションでは、ソースマップを使用して TypeScript Lambda 関数をデバッグする方法について説明します。

前提条件

TypeScript Lambda 関数をデバッグするには、次の前提条件を満たす必要があります。

- TypeScript は、ソースマップオプションを有効にしてコンパイルする必要があります。詳細については、VS Code ドキュメントの「JavaScript のソースマップのサポート」トピックを参照してください。
- インラインソースマップはサポートされていません。ソースマップを保存するには、別の `.js.map` ファイルを使用する必要があります。

設定

AWS Toolkit で TypeScript Lambda 関数の Lambda リモートデバッグを設定するには、次の手順を実行します。

1. Toolkit から AWS 、 AWS エクスプローラーを展開します。
2. エクスプローラーから、[Lambda] ノードを展開します。
3. TypeScript を設定する関数に移動し、コンテキストメニューから [リモート呼び出し] アイコンを選択して [リモート呼び出し設定] 画面を開きます。
4. リモートデバッグを有効にするには、[リモートデバッグ] チェックボックスをオンにします。
5. `TypeScript handler file` ディレクトリを指すように、ローカルルートパスを設定します。

> **Note**
>
> `TypeScript handler file` は、デバッグブレークポイントを設定する場所です。

6. [リモートデバッグの追加設定] を展開します。
7. [ソースマップ] チェックボックスをオンにして、ソースマッピングを有効にします。
8. [Out files] フィールドを Lambda 関数のコピーのローカルディレクトリに設定します。

   Example

   `app.js` と `app.map` が `.aws-sam/build/HelloWorldFunction` にある場合は、[Out files] の場所を `/Users/`**`user`**`/`**`project`**`/aws-sam/build/`**`HelloWorldFunction`**`/*` にします。

> **Note**
>
> Out file パスは絶対パスである必要があります。
> AWS SAM および AWS CDK プロジェクトの場合、 AWS Toolkit はソースマップの自動検出をサポートします。これらのプロジェクトで Out files フィールドが空のままの場合、ツールキットは自動的にソースマップの場所を検出しようとします。

9. 設定に問題がなければ、[リモート呼び出し] ボタンを選択して TypeScript 関数のデバッグを開始します。

トラブルシューティングと高度なユースケース

デバッグセッションが失敗した場合は、以下の手順を実行してトラブルシューティングプロセスを開始します。

1. AWS Toolkit を最新バージョンに更新します。

2. [リモート呼び出し設定]ウェブビューを閉じ、再度開くことで、ウェブビューを更新します。
3. VS Code を完全に閉じ、再度開くことで、VS Code を再起動します。
4. VS Code コマンドパレットを開き、コマンド **`AWS: Reset Lambda Remote Debugging Snapshot`** を入力します。結果に表示されたら選択肢、Lambda リモートデバッグスナップショットをリセットします。
5. 問題をトラブルシューティングできない場合は、AWS Toolkit for Visual Studio Code GitHub Issues に問題を送信してください。

高度なユースケース: コード署名設定

リモートデバッグでは、Lambda 関数にデバッグレイヤーをアタッチする必要があります。関数でコード署名設定が有効で強制されている場合、 AWS Toolkit はデバッグレイヤーを関数に自動的にアタッチできません。

コード署名設定の問題を解決するには、2 つのオプションがあります。

- コード署名を一時的に削除する。
- 署名付きデバッグレイヤーを使用する。

コード署名の一時的な削除

`UntrustedArtifactOnDeployment : Warn` を設定してコード署名設定を更新し、デバッグプロセスの完了後に再度有効にして `Enforced` に戻します。

詳細については、「AWS Lambda API リファレンス」の UpdateCodeSigningConfig リファレンスを参照してください。

署名付きデバッグレイヤーの使用

1. Toolkit の Lambda リモートデバッグから AWS 、リモートデバッグの追加設定セクションを展開します。
2. [リモートデバッグの追加設定] セクションで、[レイヤーオーバーライド] フィールドからリージョンレイヤー ARN をコピーします。
3. から AWS CLI、次のコマンドを使用してレイヤーバージョン をダウンロードし`aws lambda get-layer-version-by-arn --arn` **`layer-arn`**、 Layer-arn をレイヤー ARN に置き換えます。署名付きデバッグレイヤーをダウンロードする詳細な手順については、「AWS CLI コマンドリファレンス」の get-layer-version-by-arn リファレンスを参照してください。

4. コード署名設定でレイヤーに署名し、アカウントに公開します。署名と公開のガイダンスについては、「 AWS Serverless Application Model デベロッパーガイド」のAWS SAM 「アプリケーション用のコード署名の設定」トピックを参照してください。
5. レイヤーに署名してアカウントに公開したら、Lambda リモートデバッグの [リモートデバッグの追加設定] セクションに戻り、新しいレイヤー ARN を [レイヤーオーバーライド] フィールドに入力します。プロセスが完了すると、Lambda リモートデバッグはデフォルトのレイヤーの代わりに署名付きレイヤーを使用します。

高度なユースケース: SnapStart またはプロビジョニングされた同時実行による関数のデバッグ

SnapStart またはプロビジョニングされた同時実行で設定された Lambda 関数の場合、新しいバージョンの公開にはかなりの時間がかかります。デバッグワークフローを高速化するには、新しい$LATESTバージョンを発行するのではなく、関数のバージョンのみを更新するように Lambda リモートデバッグを設定できます。

1. リモート呼び出し設定画面から、リモートデバッグの追加設定を展開します。
2. Publish version オプションの選択を解除します。
3. AWS Toolkit は関数$LATESTのバージョンのみを更新し、それを使用してデバッグするようになりました。

> **Note**
>
> $LATEST バージョンでのデバッグの副作用として、障害のないデバッグ環境を確保するために、$LATESTバージョンを呼び出す可能性のある他のトラフィックを回避する必要があります。

サポート対象のリージョン

次のエラーは、リージョンでリモートデバッグがサポートされていない場合に発生します。

```
Region ${region} doesn't support remote debugging yet
```

以下は、サポートされているリージョンの一覧です。

- ap-east-1
- ap-northeast-1

- ap-northeast-2
- ap-south-1
- ap-southeast-1
- ap-southeast-2
- ca-central-1
- eu-central-1
- eu-north-1
- eu-west-1
- eu-west-2
- eu-west-3
- me-central-1
- me-south-1
- sa-east-1
- us–east–1
- us-east-2
- us-west-1
- us-west-2

Lambda RequestEntityTooLargeException

Lambda 関数には、関数コードとアタッチされたすべてのレイヤーに対して、5 つのレイヤーと合計 250 MB の制限があります。リモートデバッグレイヤーは約 40 MB のサイズがあるため、大きな関数パッケージまたは複数のレイヤーがある場合、関数がこの制限を超える可能性があります。詳細については、「AWS Lambda デベロッパーガイド」の「Lambda: InvalidParameterValueException または RequestEntityTooLargeException」トピックセクションを参照してください。

次のリストでは、このエラーをトラブルシューティングして修正する方法について説明します。

- 関数のサイズを小さくする: 関数コードを最適化し、不要な依存関係を削除します。
- 未使用のレイヤーを削除する: デバッグ中に、必須ではないレイヤーを一時的に削除します。
- 外部依存関係を使用する: 大きな依存関係を Amazon S3 などの外部ストレージに移動し、ランタイムにロードします。

Java デバッグのトラブルシューティング

Java Lambda 関数をデバッグするには、Lambda 関数のランタイムバージョンと一致する同じ Java バージョンがローカルにインストールされている必要があります。

たとえば、Java 25 関数をデバッグする場合、 AWS Toolkit が実行されているローカル環境に Java 25 がインストールされている必要があります。Java 21 以前のバージョンがローカルにインストールされている Java 25 関数をデバッグしようとすると、リモートデバッグは設定したブレークポイントで停止できません。

デバッグセッションを開始する前に、ローカル Java バージョンが Lambda 関数のランタイムバージョンと一致していることを確認してください。

IoT セキュアトンネリングのクォータを超過

以下は、Lambda リモートデバッグの AWS IoT セキュアトンネリング接続の日次制限に達したときに発生するトンネルクォータ超過エラーの例です。

```
Error creating/reusing tunnel: LimitExceededException: Exceeded quota of Lambda
 debugging tunnels
```

AWS IoT セキュアトンネリング接続には次のクォータがあります。

- 無料階層の IoT セキュアトンネリングには、1 日あたり 10 個の接続が割り当てられます。
- 各トンネルは、1 つの VS Code インスタンスを最大 12 時間サポートします。
- クォータは AWS アカウントごと、1 日あたりに適用されます。

AWS IoT セキュアトンネリングエラーが発生した場合は、毎日のクォータのリセットを待つか、 サポートに連絡して AWS クォータ制限の引き上げをリクエストしてください。 AWS サポート連絡先情報については、AWS サポート連絡先ポータルを参照してください。 AWS IoT セキュアトンネリングの詳細については、「 AWS IoT デベロッパーガイド」のAWS IoT 「セキュアトンネリング」トピックを参照してください。

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、ペタバイトスケール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 から Amazon Redshift を操作する

Toolkit for VS Code から Amazon Redshift を操作する

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。

開始方法

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 Serverless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は、以下の手順で 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確認してください。

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「サンプルの Amazon Redshift クラスターを作成する」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. [プロビジョニングされたクラスター ダッシュボード] で [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 データをロード] 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 Serverless 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 コンソールで [Amazon Redshift サーバーレスを試す] ボタンを選択し、[Amazon Redshift サーバーレスの使用を開始する] 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] 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 秘密マネジャー
- 一時的な認証情報

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 エクスプローラー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ある AWS リージョン を展開します。
2. [Redshift] を展開してデータウェアハウスを選択し、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] ダイアログで [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 エクスプローラーに表示されます。

AWS Secrets Manager を使用してデータウェアハウスに接続する

> **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、AWS Secrets Manager ユーザーガイドの「AWS Secrets Manager データベースシークレットを作成する」を参照してください。

1. Toolkit エクスプローラーから、データ ウェアハウスがある AWS リージョン を展開します。
2. [Redshift] を展開して、データウェアハウスを選択し、VS Code の [接続タイプの選択] ダイアログを開きます。
3. [接続タイプを選択] ダイアログで [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ある AWS を展開します。
2. [Redshift] を展開して、データウェアハウスを選択し、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する

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、 データウェアハウスがある AWS リージョン 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ます。[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ある 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「Toolkit 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エ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エ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。からの Amazon S3 S3 リソースの使用の詳細については、このユーザーガイドのS3 リソースの使用」トピック AWS Toolkit for Visual Studio Code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、拡張機能設定を選択します。
3. [設定] ページで、[AWS > S3: ページ当たりの最大項目数] 設定までスクロールダウンします。
4. [さらにロード] が表示される前に、デフォルト値を表示したい S3 項目数に変更します。

> **Note**
>
> 有効値は 3 ~ 1000 の数値です。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 メニューを開いて(右クリック) 、[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 メニューを開き(右クリック) 、[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 メニューを開きます(右クリック) 。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

を使用して AWS Toolkit for Visual Studio Code 、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 メニューを開き(右クリック) 、[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。

VS Code 向け Amazon SageMaker Unified Studio

次世代 Amazon SageMaker の一部である、Amazon SageMaker Unified Studio は、AWS データ、分析、人工知能 (AI)、機械学習 (ML) サービスを組み合わせた統合開発エクスペリエンスです。単一の

インターフェースからワークフローを構築、デプロイ、実行、モニタリングできます。VS Code IDE と Amazon SageMaker Unifed Studio の統合設定の詳細については、「Amazon SageMaker Unified Studio ユーザーガイド」の「VS Code での Amazon SageMaker Unified Studio 統合の設定」を参照してください。

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
- AWS Serverless Land の使用
- 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

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

前提条件

を作成または操作する前に AWS サーバーレスアプリケーション、次の前提条件を満たす必要があります。

> **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) をインストールします。 AWS SAM CLI のインストール方法の詳細と手順については、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 からツールチェーンを設定する方法の詳細については、このユーザーガイドの「ツールチェーンの設定」トピック AWS Toolkit for Visual Studio Code 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

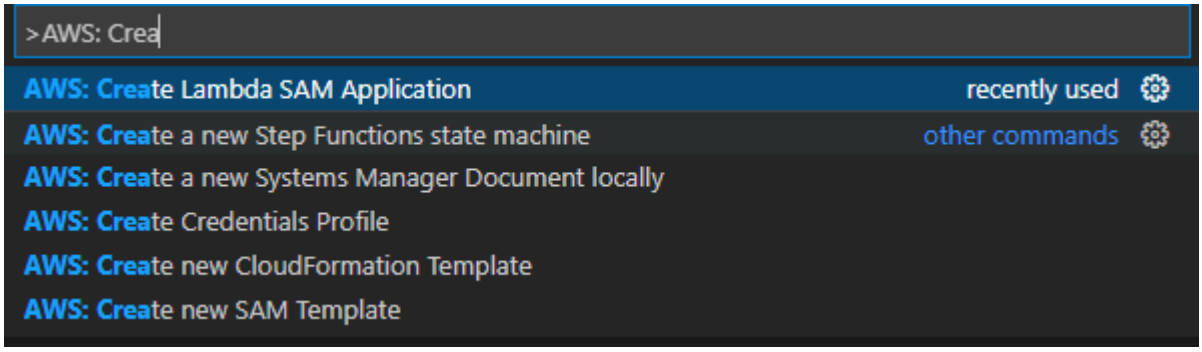
Toolkit for VS Code には、サーバーレスアプリケーションをデプロイして実行するために必要な AWS Identity and Access Management (IAM) のアクセス許可を含める認証情報プロファイルが必要です。次のサービスへの適切な読み取り/書き込みアクセスが必要です: CloudFormation、IAM、Lambda、Amazon API Gateway、Amazon Simple Storage Service (Amazon S3)、Amazon Elastic Container Registry (Amazon ECR)。

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)

この手順では、 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 AWS SAM。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あり、構築、ローカルテスト、変更、 AWS クラウドへのデプロイを行うことができます。

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 エディタの右下隅に エ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。`Image` および `Zip` パッケージタイプの相違点の詳細については、「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。

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)
- python3.12 (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-based環境でアプリケーションを実行できるようにするには、 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、3.12
> - 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。
   `template.yaml` ファイルの CodeLens インジケーターを使用して、デバッグ設定を追加します。
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"` – 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`"template"`になるように、 値が であることを確認します。
   - `"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名前と一致することを確認します。この場合はタイプ AWS::Serverless::Function の HelloWorldFunction です。

テンプレートベースのデバッグ用の launch.json ファイルの設定。

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。

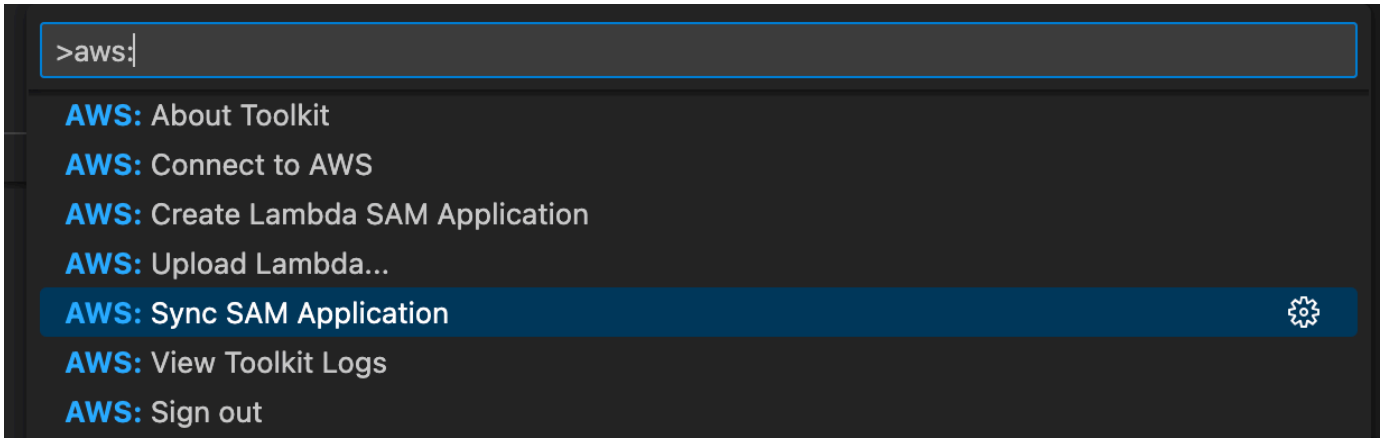
   デバッグセッションが開始すると、デバッグコンソールパネルには、デバッグ出力が表示され、Lambda 関数によって返される値が表示されます。(AWS SAM アプリケーションをデバッグする場合、AWS ツールキット が 出力 パネル内の 出力 チャンネルとして選択されています。)。

AWS SAM アプリケーションの同期

は AWS SAM CLI コマンド AWS Toolkit for Visual Studio Code を実行してsam sync、サーバーレスアプリケーションを にデプロイします AWS クラウド。 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 sam syncから AWS クラウド を使用してサーバーレスアプリケーションを 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。



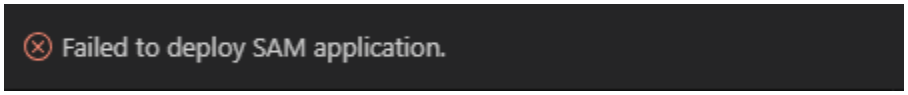
3. サーバーレスアプリケーションを同期する AWS リージョンを選択します。
4. デプロイに使用する `template.yaml` ファイルを選択します。
5. アプリケーションのデプロイ先となる既存の Amazon S3 バケットを選択するか、新しい Amazon S3 バケット名を入力します。

> **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**
>
> 同期の詳細については、 コマンドパレットから AWS Toolkit for Visual Studio Code 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 を検索し**`AWS: View AWS Toolkits Logs`**、リストに入力するときに選択します。

デプロイが完了したら、アプリケーションが [AWS Explorer] 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AWS Lambda 関数の使用トピックを参照してください。

からのサーバーレスアプリケーションの削除 AWS クラウド

サーバーレスアプリケーションを削除するには、以前に AWS クラウドにデプロイした CloudFormation スタックを削除します。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. [AWS Toolkit 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、CloudFormationを拡張します。
3. 削除するサーバーレスアプリケーションに対応する CloudFormation スタックの名前のコンテキスト (右クリック) メニューを開き、 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 CloudFormation リストからスタック名を削除します。

AWS Serverless Land の使用

AWS の Serverless Land AWS Toolkit for Visual Studio Code は、イベント駆動型アーキテクチャの構築を支援する機能のコレクションです。以下のトピックセクションでは、 AWS Toolkit

で Serverless Land を使用する方法について説明します。Serverless Land の詳細については、Serverless Land ウェブアプリケーションを参照してください。

サーバーレスランドへのアクセス

AWS Toolkit で Serverless Land にアクセスするには、主に 3 つのエントリポイントがあります。

- VS Code コマンドパレット
- AWS Toolkit Explorer
- AWS Toolkit Application Builder エクスプローラー

VS Code コマンドパレットから Serverless Land を開く

VS Code コマンドパレットから Serverless Land を開くには、次のステップを完了してください。

1. VS Code で **option+shift+p** (Mac) または **control+shift+p** (Windows) を押して、コマンドパレットを開きます。
2. VS Code コマンドパレットで、検索バーに **AWS Create application with Serverless template** を入力します。
3. リストに表示されたら、 [AWS: Create application with Serverless template] を選択します。
4. プロセスが完了すると、VS Code に Serverless Land ウィザードの [Select a Pattern for you application (1/5)] 画面が開きます。

AWS Toolkit Explorer から Serverless Land を開きます。

AWS Toolkit Explorer から Serverless Land を開くには、次の手順を実行します。

1. AWS Toolkit Explorer から、Serverless Land を開くリージョンを展開します。
2. Lambda ノードのコンテキストメニューを開きます (右クリック) 。
3. コンテキストメニューから [Serverless テンプレートを使用してアプリケーションを作成] を選択します。
4. プロセスが完了すると、VS Code に Serverless Land ウィザードの [Select a Pattern for you application (1/5)] 画面が開きます。

アプリケーションビルダーエクスプローラーからサーバーレスランドを開く

AWS Toolkit Application Builder エクスプローラーから Serverless Land を開くには、次の手順を実行します。

1. AWS Toolkit Explorer から、Application Builder エクスプローラーに移動します。
2. アプリケーションビルダーエクスプローラーを右クリックし、コンテキストメニューから [Serverless テンプレートを使用してアプリケーションを作成] を選択します。
3. プロセスが完了すると、VS Code に Serverless Land ウィザードの [Select a Pattern for you application (1/5)] 画面が開きます。

Serverless テンプレートでのアプリケーションの作成

Serverless テンプレートを使用してアプリケーションを作成するには、次の手順を実行します。

1. Serverless Land ウィザードの [Select a Pattern for you application (1/5)] 画面で、アプリケーションのベースとなるパターンを選択します。

   > **Note**
   >
   > 特定のパターンのプレビューと詳細を表示するには、表示するパターンの横にある [Open in Serverless Land] アイコンを選択します。Serverless Land のパターンがデフォルトのウェブブラウザで開きます。

2. [Select Runtime (2/5)] 画面で、プロジェクトのランタイムを選択します。
3. [Select IaC (3/5)] 画面で、プロジェクトの IaC オプションを選択します。
4. [Select a project location (4/5)] 画面で、プロジェクトを保存する場所を選択します。
5. [Enter Project Name (5/5)] 画面で、新しいアプリケーションの名前を入力します。
6. 手順が完了すると、新しいアプリケーションが VS Code エクスプローラーに表示され、プロジェクト `readme.md` が VS Code エディタで開きます。

   > **Note**
   >
   > 新しいアプリケーションが作成されると、アプリケーションタイプに固有の追加のアクションが `readme.md` ファイルに表示されます。さらに、 AWS Serverless Application Model (AWS SAM) アプリケーションは、ローカルテスト、デバッグなどのために AWS Application Builder で開くことができます。

AWS Toolkit での Application Builder の使用の詳細については、このユーザーガイドのAWS 「Application Builder エクスプローラーの使用」トピックを参照してください。

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ときは、Lambda 関数のみを実行およびデバッグし、AWS SAM テンプレートが定義する他のリソースを除外することを選択できます。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。

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
	• 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、3.12)	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。 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。
   Lambda 関数ハンドラーの CodeLens インジケータ内の [デバッグ設定の追加] オプションにアクセスします。
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。

Lambda 関数を直接呼び出す launch.json ファイルの設定。

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 Toolkit は出力パネルの出力チャネルとして選択されます) 。

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。

設定

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 を使用すると、Lambda 関数を直接呼び出すか、 AWS SAM テンプレートを使用して呼び出すためのプロパティを検索して定義できます。また、 "lambda" (関数の実行方法)"sam"、(CLI がアプリケーションを構築する方法)、 "aws" (AWS 接続情報の提供方法) AWS SAM の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。CLI AWS SAM を使用してローカルでビルドおよびデバッグ`aws-sam`するには、常に に設定します。
`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 フィールドの値を設定します。 • 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 テンプレートにあります。は、 AWS SAM テンプレートPackageType: Image で AWS Toolkit で定義された関数をイメージベースの Lambda 関数として暗黙的に認識することに注意してください。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`) と `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。作業ディレクトリが Dockerfile または CloudFormation テンプレートファイルの WorkingDirectory パラメータで変更された場合、デバッガーがローカルで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、少なくとも 1 つのpathMapping エントリを指定する必要があります。 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 CLI AWS SAM 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します。 AWS SAM CLI コマンドの動作は、 launch.json ファイル"sam"の設定のプロパティを使用して設定できます。

AWS SAM CLI (**"sam"**) プロパティ

プロパティ	説明	デフォルトの値
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	パラメータを使用して AWS SAM テンプレートをカスタマイズし、顧客値を入力しま	"parameters":{}

プロパティ	説明	デフォルトの値
	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	デフォルトの値
`credentials`	認証情報ファイルから特定のプロファイル (など`profile:default`) を選択して、 AWS 認証情報を取得します。	既存の共有設定ファイルまたは共有 AWS 認証情報ファイルが Toolkit for VS Code に提供する認証情報。 AWSAWS
`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ているデフォルトの 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: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

- WSL を使用すると、ウェブビュー (「Invoke on AWS」フォームなど) が壊れます。
- 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

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 (「Invoke on AWS」フォームなど) が壊れます。

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 では、 上のインフラストラクチャを可視化して制御できます AWS。Systems Manager は統合されたユーザーインターフェイスを提供するため、複数の AWS サービスの運用データを表示し、 AWS リソース全体の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、 を使用してオートメーション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 AWS Toolkit for Visual Studio Code。

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。

JSON

```
{
    "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。ファイルの名前を付けて VS Code に保存することはできますが、ファイルの名前は表示されません AWS。

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 AWS する前に、 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 の Systems Manager で、開く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 AWS する前に に発行する必要があります。

Systems Manager Automation ドキュメントの編集

Automation ドキュメントを所有している場合は、 AWS Explorer の Systems Manager ドキュメントの Owned by Me カテゴリに表示されます。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することも AWS、VS Code AWS から に以前に公開した新規または更新されたドキュメントを所有することもできます。

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 マネジメントコンソールで行えること以上のことができます。例えば、次のようになります。

- 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、公開できます AWS。

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、自動化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、クイック更新を選択して、その 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。

既存の Automation ドキュメントの更新を に発行すると AWS、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 の Systems Manager で、削除する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 に公開されたら AWS、それを実行して、 AWS アカウントでユーザーに代わってタスクを実行できます。Automation ドキュメントを実行するには、 AWS マネジメントコンソール、APIs、、 AWS CLIまたは を使用します AWS Tools for PowerShell。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。

または、 AWS SDKs 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。 APIs

> **Note**
>
> 自動化ドキュメントを実行すると、 に新しいリソースが作成され AWS 、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 マネジメントコンソール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、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アクセス許可があることを確認します。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 に発行しましたが AWS、 に表示されません AWS マネジメントコンソール。

で参照しているのと同じ 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ます AWS マネジメントコンソール。

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。Billing AWS 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。

AWS Step Functions

を使用すると AWS Step Functions、ワークフロー (ステートマシンとも呼ばれます) を作成して、分散アプリケーションの構築、プロセスの自動化、マイクロサービスのオーケストレーション、データ

と機械学習パイプラインの作成を行うことができます。以下のトピックでは、 AWS Step Functions で を使用する方法について説明します AWS Toolkit for Visual Studio Code。 AWS Step Functions サービスの詳細については、 AWS Step Functionsデベロッパーガイドを参照してください。

トピック

- の使用 AWS Step Functions
- AWS Step Functions Workflow Studio の使用

の使用 AWS Step Functions

以下のセクションでは、 AWS Toolkit でステートマシン定義を含むファイルを操作する AWS Step Functions Amazon State Language (ASL)方法について説明します。 AWS Step Functions ステートマシンの詳細については、「 AWS Step Functionsデベロッパーガイド」の「Step Functions でステートマシンについて学ぶ」トピックを参照してください。

Step Functions ステートマシンの表示

AWS Toolkit Explorer でステートマシン定義を含む既存のASLファイルを表示するには、次の手順を実行します。

1. AWS Toolkit Explorer から、表示するASLファイルを含むリージョンを展開します。
2. [Step Functions] の見出しを展開します。
3. ASL ファイルは AWS Explorer に表示されます。

Step Functions ステートマシンの作成

Toolkit では AWS 、ファイルから新しい Step Functions ステートマシンを作成することも、テンプレートを使用することもできます。次の手順では、 ファイルから Step Functions ステートマシンを作成する方法について説明します。テンプレートから SFN; ステートマシンを作成する方法の詳細については、以下にある本ユーザーガイドトピックの「ステートマシンテンプレート」セクションを参照してください。

> **Note**
>
> VS Code で Step Functions を使用するには、ステートマシン定義を含む Amazon State Language(ASL) ファイルの拡張子が asl.json、asl.yml、または .asl.yaml で終わる必要があります。

デフォルトでは、関連する Step Functions ファイルが Workflow Studio で開きます。 AWS Toolkit を使用した Workflow Studio での作業の詳細については、このユーザーガイドの「Working with Workflow Studio」トピックを参照してください。

1. VS Code のワークスペースから、新しいファイルを作成します。
2. ファイルに名前を付け、ファイル拡張子を `asl.json`、`asl.yml`、または `.asl.yaml` として指定します。
3. 作成時に、 AWS Toolkit は AWS Step Functions Workflow Studio で新しいファイルを開きます。
4. Workflow Studio で、ユーティリティメニューから [保存] ボタンを選択して、新しい ASL ファイルを保存します。

テンプレートからの Step Functions ステートマシンの作成

Toolkit では AWS 、テンプレートから Step Functions ステートマシンを作成できます。テンプレートプロセスは、ステートマシン定義を含む ASL ファイルが作成されるため、プロジェクトの開始点とすることができます。次の手順では、 AWS Toolkit のテンプレートから Step Functions ステートマシンを作成する方法について説明します。

1. AWS Toolkit Explorer から、Step Functions ステートマシンを作成するリージョンを展開します。
2. [Step Functions] のコンテキストメニューを開き (右クリック)、[新しい Step Functions ステートマシンの作成] を選択して、VS Code で [Select a starter template(1/2)] ウィザードを開きます。
3. [Select a starter template(1/2)] ウィザードで、Step Functions ステートマシンのテンプレートタイプを選択して続行します。
4. [Select template format(2/2)] 画面で、テンプレート形式に YAML または JSON を選択します。
5. ステートマシン定義を含む新しい ASL ファイルが VS Code エディタで開きます。

Step Functions ステートマシンのダウンロード

リモートで保存された Step Functions ステートマシンを VS Code のローカルインスタンスにダウンロードするには、次の手順を実行します。

1. AWS Toolkit Explorer から、ダウンロードする Step Functions ステートマシンを含むリージョンを展開します。

2. [Step Functions] を展開し、ダウンロードする Step Functions ステートマシンを右クリックして、[定義のダウンロード...] を選択します。
3. Step Functions ステートマシンをローカルに保存する場所を指定して、続行します。
4. 手順が完了すると、Step Functions ステートマシンが Workflow Studio で開きます。

Step Functions ステートマシンへの変更の保存

次の手順では、Step Functions ステートマシンに加えた変更を保存する方法について説明します。

> **Note**
>
> Workflow Studio で行われた編集はローカルファイルに同期されますが、VS Code エディタまたは Workflow Studio で作業を保存するまでは保存されません。Workflow Studio が開いているときにローカルファイルが変更および保存され、ASL ファイルにエラーが検出されない場合、保存が完了すると Workflow Studio に [成功] 通知が送信されます。ただし、ローカルファイルに無効な JSON または YAML が含まれている状態で保存しようとすると、ローカルファイルは同期に失敗し、Workflow Studio に [警告] 通知が送信されます。

1. Workflow Studio のステートマシン定義を含む、開いた状態の ASL ファイルから、ユーティリティボタンに移動します。
2. [保存] ボタンを選択します。
3. VS Code は、ファイルが保存されたときに通知します。

Step Functions ステートマシンの実行

次の手順では、 AWS Toolkit で Step Functions ステートマシンを実行する方法について説明します。

1. AWS Toolkit Explorer から、実行する Step Functions ステートマシンを含むリージョンを展開します。
2. [Step Functions] を展開し、実行する Step Functions ステートマシンを右クリックします。
3. コンテキストメニューから、[実行の開始] を選択して起動プロセスを開始します。
4. 起動のステータスは、VS Code の [AWS Toolkit Output] ウィンドウに表示されます。

コードスニペットの使用

コードスニペットは、使用しているコードに基づいて自動で生成される提案です。ツールキットの Step Functions でコードスニペットを使用するには、次の手順を実行します。

> **Note**
>
> VS Code で Step Functions コードスニペットを使用するには、ステートマシン定義を含む ASL ファイルの拡張子が `.asl.json`、`.asl.yml`、または `.asl.yaml` で終わる必要があります。
> デフォルトでは、関連する Step Functions ファイルが Workflow Studio で開きます。

1. VS Code から、変更する、または新しい ASL ファイルを作成するステートマシン定義を含む ASL ファイルを開きます。
2. Workflow Studio で、現在が [設計] モードの場合は [コード] モードに切り替えます。
3. Workflow Studio のコードエディタで、`"States"` プロパティにカーソルを置きます。
4. **`control + space`** を押してコードスニペットメニューを開きます。**`control + space`** を押すことで、`"State"` `"Type"` に基づく追加のプロパティにアクセスできます。
5. リストから目的のコードスニペットを選択します。

コード検証

Workflow Studio で Step Functions を使用すると、コード検証によってエラーがアクティブに識別され、次の提案が行われます。

- 不足しているプロパティ
- 不正な値
- 非ターミナル状態
- 存在しない参照先ステート

AWS Step Functions Workflow Studio の使用

以下のセクションでは、 で AWS Step Functions Workflow Studio を使用する方法について説明します AWS Toolkit for Visual Studio Code。 AWS Step Functions Workflow Studio の詳細については、

「 デベロAWS Step Functionsッパーガイド」の「ワークフローの開発」トピックを参照してください。

Workflow Studio を開く

次のリストでは、VS Code で Workflow Studio を開くために使用できるさまざまなパスについて説明します。

> **Note**
>
> VS Code で Workflow Studio を使用するには、ステートマシン定義を含む Amazon State Language(ASL) ファイルの拡張子が asl.json、asl.yml、asl.yaml で終わる必要があります。 AWS Toolkit で新しいステートマシン定義をダウンロードまたは作成する方法の詳細については、このユーザーガイドの「 の使用 AWS Step Functions」トピックの「ステートマシンのダウンロード」セクションと「ステートマシンの作成」セクションを参照してください。

- AWS Explorer から、ステートマシン定義を含むASLファイルのコンテキストメニューを開き (右クリック) 、Workflow Studio で開くを選択します。
- ステートマシン定義を含む、開いた状態の ASL ファイルから、VS Code エディタウィンドウのタブの横にある [Workflow Studio で開く] アイコンを選択します。
- ステートマシン定義を含む、開いた状態の ASL ファイルから、ファイルの上部にある Open with Workflow Studio の CodeLens コマンドを選択します。
- ステートマシン定義を含む ASL ファイルを閉じてから再度開くと、デフォルトの Workflow Studio を手動で無効にしない限り、Workflow Studio でファイルが自動的に再度開きます。

設計モードとコードモード

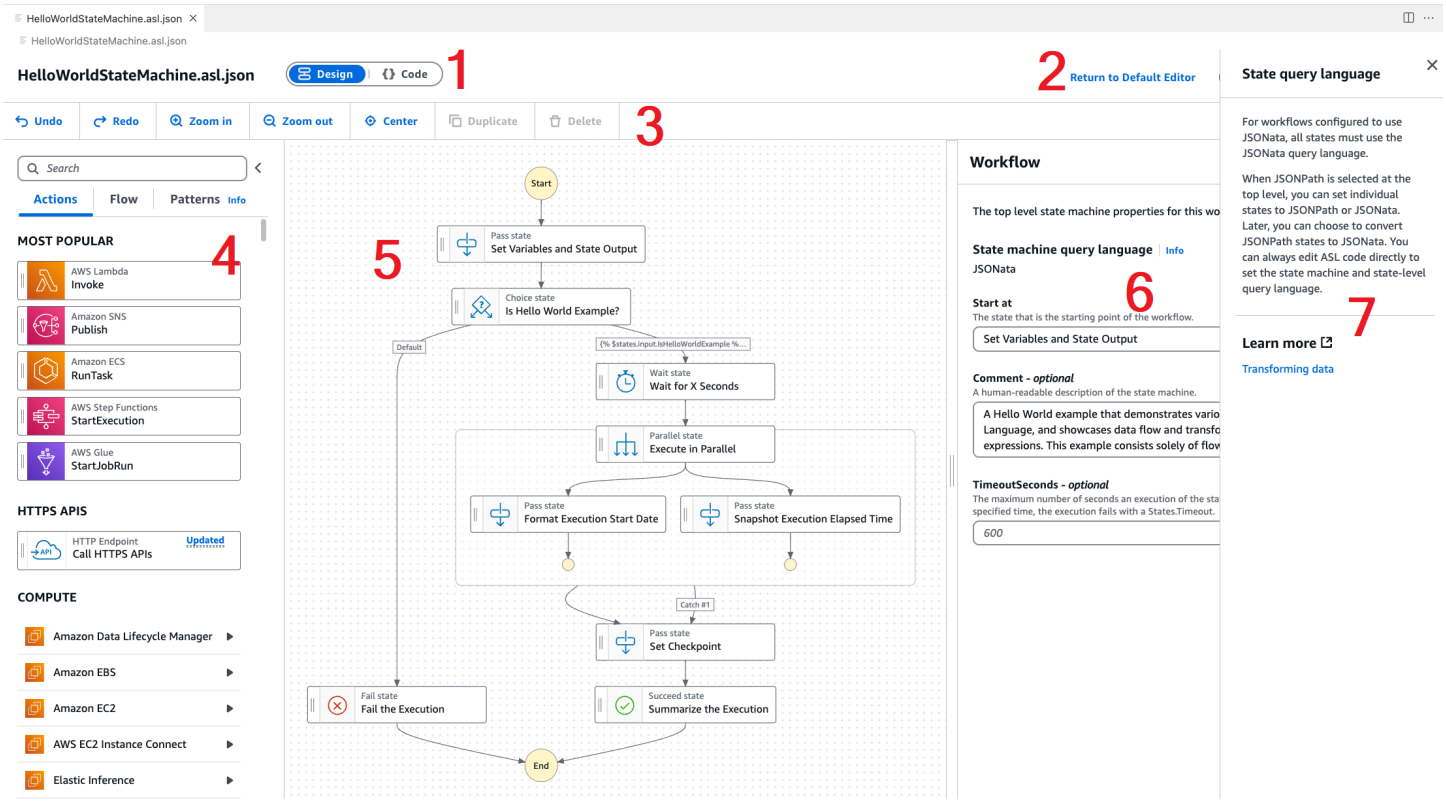
Workflow Studio には、ステートマシン定義を含む ASL ファイルを操作するためのモードが 2 つあります。[設計] モードと [コード] モードです。[設計] モードには、プロトタイプを作成する際にワークフローを可視化するグラフィカルインターフェイスがあります。[コード] モードには、ワークフローの ASL 定義を表示、書き込み、編集できるコードエディタが統合されています。

> **Note**
>
> 設計モードとコードモードの各 UI セクションの詳細については、「AWS Step Functions デベロッパーガイド」の「Workflow Studio の使用」トピックを参照してください。たとえ

ば、Config モードなど、すべての Workflow Studio 機能が AWS Toolkit で使用できるわけではありません。

[設計] モードの UI には、次の図でラベル付きで説明されているように、7 つの主要セクションがあります。

1. モードボタン: [設計] モードと [コード] モードを切り替えるためのボタン。
2. ユーティリティボタン: Workflow Studio の終了、ワークフローの保存、JSON ファイルや YAML ファイル への ASL 定義のエクスポートなどのタスクを実行するための一連のボタン。
3. デザインツールバー : 取り消し、削除、ズームコントールなどの一般的なアクションを実行する一連のボタンを含むツールバー。
4. ステートブラウザ: ワークフローキャンバスにステートをドラッグアンドドロップできるブラウザ。ステートはタブに整理され、アクション、フロー、パターンとして定義されます。
5. キャンバスとワークフローグラフ: 設定のステートを削除、再編成、選択できるワークフローの視覚的なレンダリング。
6. Inspector パネル: キャンバスで選択したステートのプロパティを表示および編集。キャンバスワークフローグラフで選択されたステートに応じて、設定、入力/出力、変数、エラー処理のステート固有のオプションがタブに入力されます。
7. 情報リンク: ヘルプが必要なときにコンテキスト情報を含むパネルを表示。これらのパネルには、「AWS Step Functions デベロッパーガイド」の関連トピックへのリンクも含まれています。



設計中の単一ステートテストの使用

Workflow Studio のテストステート UI から、ステートマシンの個々のステートをテストできます。これには、状態入力の提供、変数の設定、 と AWS SAM CloudFormation 定義の両方の置換を行う機能が含まれます。

コードとしてのインフラストラクチャ (IaC)、リソース定義、データの変換の詳細については、「AWS Step Functionsデベロッパーガイド」の「Step Functions ワークフローを構築 AWS SAM するための の使用」および「Step Functions トピックの JSONata を使用したデータの変換」を参照してください。

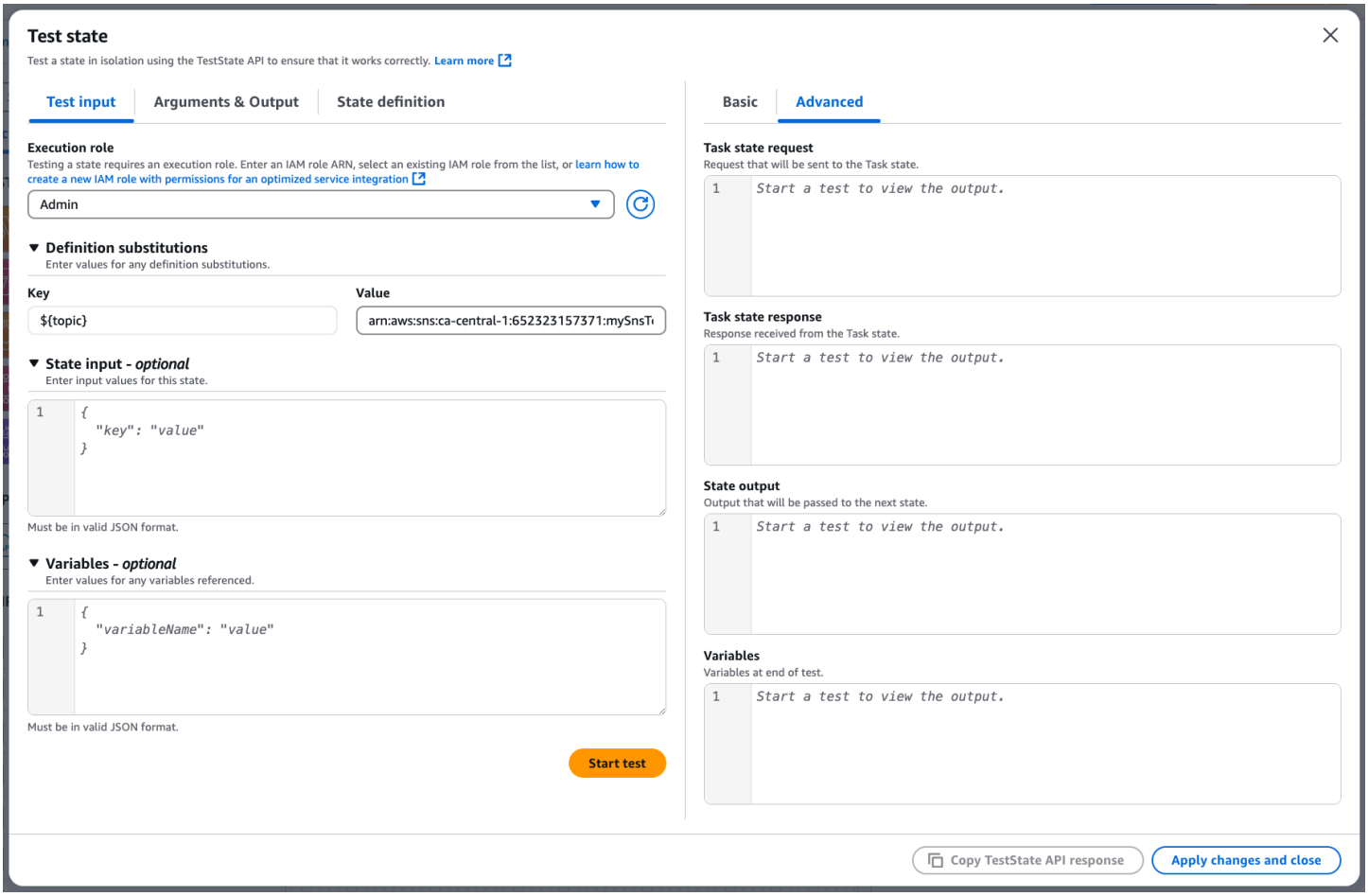
次の手順では、Workflow Studio でテストステート UI を開く方法について説明します。

テストステート UI を開く

1. Workflow Studio の [設計] モードタブからキャンバスに移動し、ステートを選択して [Inspector] パネルで開きます。
2. [Inspector] パネルから、[テストステート] ボタンを選択します。
3. VS Code で [Test state] UI が開きます。

テストステート UI には、[Test input]、[Arguments & Output]、[State definition] の 3 つのメインタブがあります。テスト入力タブには、状態入力を提供したり、変数を設定したり、 AWS SAM または CloudFormation テンプレートから定義の置換を指定したりできる 3 つの追加フィールドがあります。[State definition] タブでは、ワークフローを調整して再テストできます。テストの実行が完了したら、ステートマシン定義に変更を適用して保存できます。

次のスクリーンショットは、トピックリソース定義を含むテストステート UI を示しています。



デフォルトで Workflow Studio を無効にする

デフォルトでは、Workflow Studio はステートマシン定義を含む ASL ファイルのデフォルトのエディタです。ローカル `.vscode` ディレクトリの `settings.json` ファイルを変更することで、デフォルト設定を無効にすることができます。Workflow Studio をデフォルトで無効にしても、このトピックにある「Workflow Studio を開く」セクションに記載されている方法でアクセスできます。

VS Code から `settings.json` ファイルを編集するには、次の手順を実行します。

1. VS Code で **option+shift+p** (Mac) 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. VS Code コマンドパレットで検索フィールドに **Open User Settings (JSON)** を入力し、リストに表示されたらそのオプションを選択します。
3. エディタの `settings.json` から、ファイルに次の変更を追加します。

```
                {
                    "workbench.editorAssociations": {
                    // Use all the following overrides or a specific one for a
certain file type
                    "*.asl.json": "default",
                    "*.asl.yaml": "default",
                    "*.asl.yml": "default"
                    }
                }
```

4. 変更を `settings.json` に保存し、VS Code を更新または再起動します。

Threat Composer の使用

AWS Toolkit for Visual Studio Code で Threat Composer ツールを使用できます。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 の GitHub リポジトリを参照してください。

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

トピック

- ツールキットからの Threat Composer の使用

ツールキットからの Threat Composer の使用

Threat Composer を使用すると、Threat Composer 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 の 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、AWS Toolkit for Visual Studio Code で Threat Composer ツールにアクセスする方法について説明します。

ツールキットからの Threat Composer へのアクセス

ツールキットから 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 が自動的に開き、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がレンダリングされます。

新しい Threat Composer 脅威モデルの作成

1. VS Code のメインメニューで、[File] を展開し、[New File] を選択します。
2. [New File] ダイアログで、[Threat Composer File...] を選択します。
3. プロンプトが表示されたら `file name` を入力し、 **`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 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 **`Threat Composer`** と入力して、結果に表示されたら [Create New Threat Composer File] を選択します。
3. プロンプトが表示されたら `file name` を入力し、 **`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 にデフォルトでリストされている AWS サービスへのアクセスに加えて、 リソースに移動し、数百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。では AWS、リソースは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、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の現在の AWS リージョンで利用可能な既存のリソースのリストを表示します。
- 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と、リソースをモデル化するためのスキーマ (JSON および YAML 形式) を説明する AWS 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**
>
> * 現在のリリースでは、リソースを作成、編集、削除する AWS Toolkit for Visual Studio Code 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。また、実験的な機能は、予告 AWS Toolkit for Visual Studio Code なしに から削除される場合があります。
> 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

サービスに関連付けられた AWS リソースにアクセスするには、特定のアクセス AWS Identity and Access Management 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。

サービスリソースのアクセス許可に加えて、IAM エンティティには、Toolkit for VS Code が代わりに AWS Cloud Control API オペレーションを呼び出すことを許可する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。

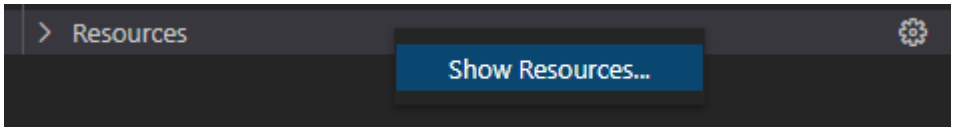
アクセス許可を付与する最も簡単な方法は、 Toolkit インターフェイスを使用してこれらの API オペレーションを呼び出す IAM エンティティに管理 AWS ポリシー PowerUserAccess をアタッチすることです。このマネージドポリシーでは、API オペレーションの呼び出しなど、アプリケーション開発タスクを実行するために一定の範囲のアクセス許可が付与されます。

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。

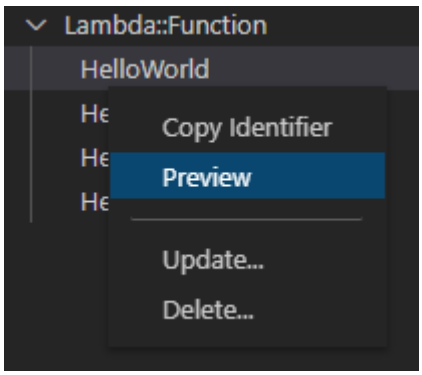
   使用可能なリソースのリストが、リソースタイプの下に表示されます。

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。(このバージョンの ???では、リソースの削除は現在 です) AWS Toolkit for Visual Studio Code。

> **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 CloudFormation テンプレートでリソースへの参照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

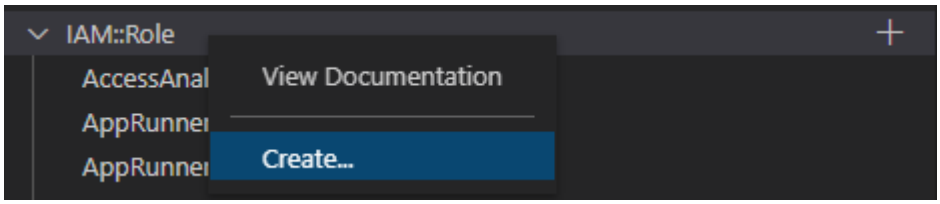
> **Important**
>
> リソースの作成と更新は、このバージョンの AWS Toolkit for Visual Studio Codeでは現在 ??? となっています。

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 AI 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。

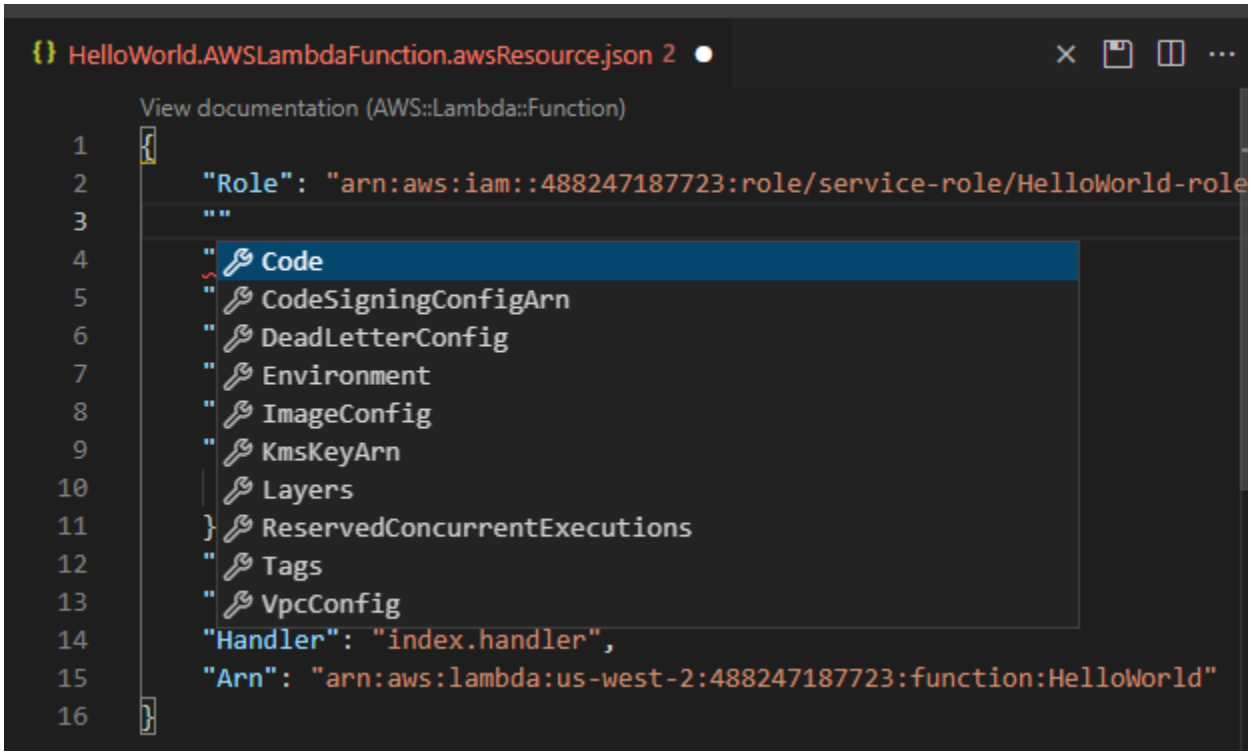
1. 右クリックして、作成するリソースのリソースタイプを追加し、[リソース] をクリックし、[リソースを表示] を選択します。
2. [リソース] の下でリソースタイプを追加した後、プラス (「+」) アイコンを選択して、新しいエディタでテンプレートファイルを開きます。

   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、保存アイコンを選択してテンプレートを検証し、リソースをリモート AWS クラウドに保存します。

   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。

のトラブルシューティング AWS Toolkit for Visual Studio Code

以下のセクションでは、 AWS Toolkit for Visual Studio Code および ツールキットからの AWS サービスの使用に関する一般的なトラブルシューティング情報について説明します。Toolkit での SAM 問題のトラブルシューティングに特に関連する問題については AWS 、このユーザーガイドの「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」トピックを参照してください。

トピック

- トラブルシューティングのベストプラクティス
- プロファイル ... が設定ファイルで見つかりませんでした
- SAM json スキーマ: template.yaml ファイルのスキーマを変更できません

トラブルシューティングのベストプラクティス

以下は、 AWS Toolkit for Visual Studio Code 問題のトラブルシューティング時に推奨されるベストプラクティスです。への貢献方法の詳細については AWS Toolkit for Visual Studio Code、AWS Toolkit for Visual Studio Code GitHub リポジトリの「 への貢献 AWS Toolkit for Visual Studio Code」トピックを参照してください。

- レポートを送信する前に、問題またはエラーの再現を試してください。
- 再現プロセス中に、各ステップ、設定、エラーメッセージを詳細にメモしてください。
- AWS Toolkit デバッグログを収集します。Toolkit AWS デバッグログの検索方法の詳細については、このユーザーガイドトピックにある AWS ログの検索方法の手順を参照してください。
- 未解決のリクエストや既知の解決策がないかを確認するか、 AWS Toolkit for Visual Studio Code GitHub リポジトリのAWS Toolkit for Visual Studio Code 「問題」セクションで未解決の問題を報告します。

> **Note**
>
> 次の手順では、 AWS Toolkit デバッグログを表示する方法について説明します。Amazon Q デバッグログを表示する手順は、VS Code コマンドパレットから [Amazon Q: ログの表示] を選択することを除いて同じです。

AWS Toolkit for Visual Studio Code デバッグログを見つける方法

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、検索フィールドに **AWS View Logs** を入力します。
2. AWS ログの表示を選択して、VS Code ターミナル出力ウィンドウで AWS Toolkit ログを開きます。
3. [VS Code ターミナル出力] ウィンドウから 歯車アイコンメニューを展開し、[デバッグ] を選択します。
4. 歯車アイコンメニューを再度展開し、[デフォルトとして設定] を選択します。
5. **Cmd + Shift + P** または **Ctrl + Shift + P** (Windows) を押してコマンドパレットを再度開き、**Reload Window** を検索し、[デベロッパー: ウィンドウの再ロード] を選択します。
6. VS Code が再ロードされ、VS Code ターミナル出力ウィンドウに、更新された AWS Toolkit Debug ログが表示されます。

プロファイル ... が設定ファイルで見つかりませんでした

問題

> **Note**
>
> この問題は `~/.aws/config` ファイルにのみ適用され、 `~/.aws/credentials` ファイルには適用されません。 AWS 設定ファイルと AWS 認証情報ファイルの詳細については、AWS SDK およびツールリファレンスガイドの「共有設定ファイルと認証情報ファイル」トピックを参照してください。

認証情報を選択する場合、 AWS Toolkit ログには、 という構造でメッセージが表示されますProfile *name* could not be found in shared credentials file。

以下は、 AWS Toolkit ログでこのエラーがどのように表示されるかの例です。

```
        2023-08-08 18:20:45 [ERROR]: _aws.auth.reauthenticate: Error: Unable to
authenticate connection
        -> CredentialsProviderError: Profile vscode-prod-readonly could not be found
in shared credentials file.
```

解決策

プロファイルが ~/.aws/config に既に存在する場合は、[profile で始まっていることを確認します。以下は、正しい構造のユーザープロファイルの例です。

```
[profile example]
region=us-west-2
credential_process=...
```

以下は、誤った構造のユーザープロファイルの例です。

```
[example]
region=us-west-2
credential_process=...
```

SAM json スキーマ: template.yaml ファイルのスキーマを変更できません

問題

SAM template.yaml で別の JSON スキーマを手動で選択できない

解決策

vscode-yaml バージョン 1.11 以降に更新すると、YAML ファイルの先頭に **yaml-language-server** モードラインを追加して、URI でスキーマを強制的に使用できます。詳細については、GitHub リポジトリ「Redhat developer」の「yaml language server」トピックの「Using inlined schema」を参照してください。次は、**yaml-language-server** モードラインの例です。

```
# yaml-language-server: $schema=https://raw.githubusercontent.com/aws/
serverless-application-model/main/samtranslator/schema/schema.json
```

AWS Toolkit for Visual Studio Code のセキュリティ

トピック

でのデータ保護AWS Toolkit for Visual Studio Code

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があります。ユーザーは、このインフラストラクチャでホストされるコンテンツに対する管理を維持する責任があります。また、使用する「AWS のサービス」のセキュリティ設定と管理タスクもユーザーの責任となります。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に関する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」 認証情報を保護し、「AWS IAM アイデンティティセンター」 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権限のみが各ユーザーに付与されます。また、次の方法でデータを保護することもお勧めします:

- 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 が必須で、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ギングを設定します。CloudTrail 証跡を使用して AWS アクティビティをキャプチャする方法については、「AWS CloudTrail ユーザーガイド」の「CloudTrail 証跡の使用」を参照してください。
- AWS のサービス 内のすべてのデフォルトセキュリティコントロールに加え、AWS 暗号化ソリューションを使用します。
- Amazon Macie などの高度な管理された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」 にアクセスする際に FIPS 140-3 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) 140-3」を参照してください。

お客様の E メールアドレスなどの極秘または機密情報は、タグ、または名前フィールドなどの自由形式のテキスト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。タグ、または名前に使用される自由記述のテキストフィールドに入力したデータは、請求または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

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
Amazon SageMaker Unified Studio	Amazon SageMaker Unified Studio サービスとの統合。	2025 年 9 月 18 日
LocalStack	LocalStack の起動をサポートするために新しいユーザーガイドトピックを追加しました。	2025 年 9 月 11 日
AWS Lambda 関数を操作する	ツールキットの更新された Lambda 関数を含めるようにユーザーガイドトピックを更新しました。	2025 年 7 月 17 日
AWS Lambda リモートデバッグ	AWS Toolkit for Visual Studio Code ユーザーガイドに新しい AWS Lambda リモートデバッグトピックを追加しました。	2025 年 7 月 17 日
AWS Lambda コンソールから IDE へ	AWS Toolkit for Visual Studio Code ユーザーガイドに新しい AWS Lambda console から IDE のトピックを追加しました。	2025 年 7 月 17 日
AWS Step Functions コンテンツの更新と Workflow Studio のサポートの追加	機能の起動をサポートするために、AWS Step Functions Workflow Studio の既存のコンテンツ AWS Step Functions	2025 年 3 月 6 日

	とユーザーガイドトピックを更新しました。	
AWS Serverless Land	AWS アプリケーションビルダー TOC に新しい AWS Serverless Land のトピックを追加しました。	2025 年 3 月 6 日
アクセスを許可するファイアウォールとゲートウェイの更新	AWS Toolkit for Visual Studio Code と VS コード拡張機能用の Amazon Q 内のすべてのサービスと機能にアクセスするために許可リストに追加する必要があるエンドポイントとリソースのリスト。	2025 年 2 月 28 日
Amazon ECR App Runner のサポート	AWS ツールキットの Amazon Elastic Container Registry ノードから AWS App Runner サービスを起動するための、ドキュメントサポートを追加しました。	2025 年 2 月 6 日
– Amazon DocumentDB	AWS Toolkit for Visual Studio Code ユーザーガイドに新しい Amazon DocumentDB のトピックを追加しました。	2025 年 2 月 6 日
EC2 のサポート	Amazon Elastic Compute Cloud サービスのサポートをツールキットに追加しました。	2025 年 1 月 31 日
AWS ドキュメント	AWS ドキュメントの新しいユーザーガイドトピックを追加しました。	2025 年 1 月 20 日

Amazon CloudWatch Logs Live Tail	AWS Toolkit for Visual Studio Code に Amazon CloudWatch Logs Live Tail 機能をサポートする新しいサブトピックを追加しました。	2024 年 12 月 15 日
AWS アプリケーションビルダー	AWS Toolkit for Visual Studio Code ユーザーガイドに新しい AWS アプリケーションビルダーのトピックを追加しました。	2024 年 10 月 30 日
Infrastructure Composer	AWS Application Composer は AWS Infrastructure Composer になりました。	2024 年 10 月 3 日
AWS Identity and Access Management (IAM) Access Analyzer の更新	IAM Access Analyzer のコンテンツを更新し、新しい API リファレンスを追加しました。	2024 年 7 月 10 日
AWS Identity and Access Management (IAM) Access Analyzer	IAM Access Analyzer の新しいユーザーガイドトピックを追加しました。	2024 年 5 月 23 日
AWS 承認フローへの接続が更新されました	承認フローが更新され、認証プロセスの変更と、Amazon Q が AWS Toolkit for Visual Studio Code から分離したことが反映されました。	2024 年 4 月 30 日
VS コードの Amazon Q 拡張機能	2024 年 4 月 30 日以降、CodeWhisperer は Amazon Q の一部となり、Amazon Q は VS Code の拡張機能として利用可能になりました。	2024 年 4 月 30 日

開発環境での仮想プライベートクラウドのサポート	開発環境で VPC をサポートするための UI の変更に関するコンテンツを更新しました。	2024 年 1 月 21 日
Infrastructure Composer	AWS Toolkit for Visual Studio Code ユーザーガイドに新しい Infrastructure Composer のトピックを追加しました。	2023 年 11 月 28 日
CodeCatalyst 向けの SSO サポート	CodeCatalyst 環境と開発環境向けの IAM アイデンティティセンターのサポートをカバーするように、コンテンツを更新しました。	2023 年 11 月 17 日
VS Code とツールキットのダウンロードリンクを追加しました	コンテンツを更新し、VS Code と AWS Toolkit for Visual Studio Code のダウンロードリンクを追加しました。	2023 年 11 月 1 日
Amazon Redshift トピック	AWS Toolkit for Visual Studio Code ユーザーガイドに新しい Amazon Redshift のトピックを追加しました。	2023 年 10 月 17 日
AWS 承認フローへの接続が更新されました	サービス固有の認証方法に焦点を当てるように承認フローが更新されました。	2023 年 9 月 29 日
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

UI のマイナーアップデート	UI との整合性を高めるため、「マシン状態をプレビュー」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 の 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
の Amazon ECR サービスの概要AWS Toolkit for Visual Studio Code	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
Node.js セクションの形式アップデート	お客様からのフィードバックに応じて、Node.js/TypeScript の形式をアップデートしました。	2021 年 8 月 12 日
App Runner サポート	AWS App Runner のサポートを AWS Toolkit for Visual Studio Code に追加しました。	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
のYAML サポートAWS Step Functions	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
用ユーザーガイドAWS Toolkit for Visual Studio Code	一般的な可用性のリリース	2019 年 7 月 11 日

用ユーザーガイドAWS Toolkit for Visual Studio Code	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
のインストールAWS Toolkit for Visual Studio Code	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。