ユーザーガイド

AWS Amazon Q を使用したツールキット

AWS Amazon Q を使用したツールキット: ユーザーガイド

Table of Contents

AWS Amazon Q を使用したツールキット

これは、AWS Toolkit for Visual Studio with Amazon Q のユーザーガイドです。AWS Toolkit for VS Code を探している場合は、「AWS Toolkit for Visual Studio Codeユーザーガイド」を参照してください。

Amazon Q を使用した AWS Toolkit for Visual Studio とは

AWS Toolkit for Visual Studio with Amazon Q は、Amazon Web Services を使用する .NET アプリケーションの開発、デバッグ、デプロイを容易にする Visual Studio IDE の拡張機能です。 AWS Toolkit with Amazon Q は、Visual Studio バージョン 2022 以降でサポートされています。キットをダウンロードしてインストールする方法の詳細については、このユーザーガイドの「インストールとセットアップ」のトピックを参照してください。

> **Note**
>
> Toolkit for Visual Studio は Visual Studio 2008、2010、2012、2013、2015、2017、2019 の各バージョン用にも提供されています。ただし、これらのバージョンはサポートされていません。詳細については、このユーザーガイドの「インストールとセットアップ」のトピックを参照してください。

AWS Toolkit with Amazon Q には、開発エクスペリエンスを向上させるための以下の機能が含まれています。

AWS Explorer

AWS Explorer ツールウィンドウは IDE の表示メニューでアクセスでき、Visual Studio の AWS サービスとやり取りできます。サポートされている AWS サービスと機能のリストについては、このユーザーガイドのAWS 「サービスの使用」トピックを参照してください。

Amazon Q

Visual Studio の Amazon Q Developer とチャットして、 でのビルドについて質問 AWS したり、ソフトウェア開発の支援を受けたりします。Amazon Q は、コーディングの概念とコードスニペットについて説明できるほか、コードやユニットテストを生成し、デバッグやリファクタリングを通じてコードを改善できます。

Toolkit for Visual Studio の Amazon Q をインストールしてセットアップするには、このユーザーガイドの「開始方法」のトピックを参照してください。Amazon Q Developer の操作の詳細については、「Amazon Q Developer User Guide」」の「Amazon Q Developer in IDE」のトピック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。

関連情報

問題を開いたり、現在開いている問題を表示したりするには、https://github.com/aws/aws-toolkit-visual-studio/issues にアクセスしてください。

Visual Studio の詳細については、https://visualstudio.microsoft.com/vs/ にアクセスしてください。

Amazon Q

Amazon Q とは

2024 年 4 月 30 日現在、Amazon CodeWhisperer は Amazon Q Developer の一部となり、これにはインラインコードの提案とセキュリティスキャンが含まれます。

AWS Toolkit for Visual Studio での Amazon Q Developer の使用の詳細については、「Amazon Q Developer ユーザーガイド」の「Amazon Q Developer in IDE」のトピック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 Visual Studio のダウンロード

Toolkit for Visual Studio は IDE の Visual Studio Marketplace からダウンロード、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「開始方法」トピックの「AWS Toolkit for Visual Studio のインストール」セクションを参照してください。

Visual Studio Marketplace からツールキットをダウンロードする

ウェブブラウザで AWS Visual Studio ダウンロードサイトを開いて、Toolkit for Visual Studio のインストールファイルをダウンロードします。

AWS のその他の IDE ツールキット

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

AWS Toolkit for Visual Studio Code へのリンク

- このリンクにアクセスして、VS Code Marketplace から AWS Toolkit for Visual Studio Code をダウンロードできます。
- AWS Toolkit for Visual Studio Code の詳細については、「AWS Toolkit for Visual Studio Code ユーザーガイド」を参照してください。

AWS Toolkit for JetBrains へのリンク

- このリンクにアクセスして、JetBrains Marketplace から AWS Toolkit for JetBrains をダウンロードできます。
- AWS Toolkit for JetBrains の詳細については、「AWS Toolkit for JetBrains ユーザーガイド」を参照してください。

開始方法

AWS Toolkit for Visual Studio は、AWS のサービスとリソースを Visual Studio 統合開発環境 (IDE) から直接利用できるようにします。

使用開始をサポートするために、以下のトピックでは AWS Toolkit for Visual Studio のインストール、セットアップ、設定方法について説明します。

トピック

のインストールとセットアップ AWS Toolkit for Visual Studio

次のトピックでは、 AWS Toolkit for Visual Studioのダウンロード、インストール、セットアップ、アンインストールの方法について説明します。

トピック

前提条件

サポートされているバージョンの AWS Toolkit for Visual Studioをセットアップするための前提条件を次に示します。

- Visual Studio 19 以降のリリース
- Windows 10 以降のリリース
- Windows と Visual Studio への管理者アクセス
- アクティブな AWS IAM 認証情報

> **Note**
>
> サポートされていないバージョンの AWS Toolkit for Visual Studio は、Visual Studio 2008、2010、2012、2013、2015、および 2017 で使用できます。サポートされていないバージョンをダウンロードするには、AWS Toolkit for Visual Studio のランディングページに移動し、ダウンロードリンクのリストから目的のバージョンを選択します。
> IAM 認証情報の詳細を確認したり、アカウントにサインアップしたりするには、AWS コンソールゲートウェイにアクセスしてください。

のインストール AWS Toolkit for Visual Studio

をインストールするには AWS Toolkit for Visual Studio、次の手順で Visual Studio のバージョンを検索し、必要なステップを完了します。のすべてのバージョンのダウンロードリンク AWS Toolkit for Visual Studio は、AWS Toolkit for Visual Studioランディングページにあります。

> **Note**
>
> のインストール中に問題が発生した場合は AWS Toolkit for Visual Studio、このガイドの「インストールに関する問題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。

AWS Toolkit for Visual Studio for Visual Studio 2022 のインストール

Visual Studio から AWS Toolkit for Visual Studio 2022 をインストールするには、次の手順を実行します。

1. [Main menu] から [Extensions] に移動し、[Manage Extensions] を選択します。
2. 検索ボックスで AWSを検索します。
3. 該当するバージョンの Visual Studio 2022 の [Download] ボタンを選択し、インストールプロンプトに従います。

> **Note**
>
> インストールプロセスを完了するには、Visual Studio を手動で閉じて再起動しなければならない場合があります。

4. ダウンロードとインストールが完了したら、表示メニューから AWS Explorer AWS Toolkit for Visual Studio を選択して を開くことができます。

AWS Toolkit for Visual Studio for Visual Studio 2019 のインストール

Visual Studio から AWS Toolkit for Visual Studio 2019 をインストールするには、次の手順を実行します。

1. [Main menu] から [Extensions] に移動し、[Manage Extensions] を選択します。
2. 検索ボックスで AWSを検索します。
3. Visual Studio 2017 と 2019 の [Download] ボタンを選択し、プロンプトに従います。

> **Note**
>
> インストールプロセスを完了するには、Visual Studio を手動で閉じて再起動しなければならない場合があります。

4. ダウンロードとインストールが完了したら、表示メニューから AWS Explorer AWS Toolkit for Visual Studio を選択して を開くことができます。

のアンインストール AWS Toolkit for Visual Studio

をアンインストールするには AWS Toolkit for Visual Studio、次の手順から Visual Studio のバージョンを検索し、必要なステップを完了します。

AWS Toolkit for Visual Studio for Visual Studio 2022 のアンインストール

Visual Studio から AWS Toolkit for Visual Studio 2022 をアンインストールするには、次の手順を実行します。

1. [Main menu] から [Extensions] に移動し、[Manage Extensions] を選択します。
2. [Manage Extensions] ナビゲーションメニューから、[Installed] 見出しを展開します。
3. AWS Toolkit for Visual Studio 2022 拡張機能を探し、[Uninstall] ボタンを選択します。

> **Note**
>
> ナビゲーションメニューのインストール済みセクションから が表示され AWS Toolkit for Visual Studio ない場合は、Visual Studio を再起動する必要がある場合があります。

4. 画面上のプロンプトに従ってアンインストールプロセスを完了します。

AWS Toolkit for Visual Studio for Visual Studio 2019 のアンインストール

Visual Studio から AWS Toolkit for Visual Studio 2019 をアンインストールするには、次の手順を実行します。

1. [Main menu] から [ツール] に移動し、[Manage Extensions] を選択します。
2. [Manage Extensions] ナビゲーションメニューから、[Installed] 見出しを展開します。
3. AWS Toolkit for Visual Studio 2019 拡張機能を探し、[Uninstall] ボタンを選択します。
4. 画面上のプロンプトに従ってアンインストールプロセスを完了します。

AWS Toolkit for Visual Studio for Visual Studio 2017 のアンインストール

Visual Studio で AWS Toolkit for Visual Studio 2017 をアンインストールするには、次の手順を実行します。

1. [Main] メニューから [ツール] に移動し、[Extensions and Updates] を選択します。
2. [Extensions and Updates] ナビゲーションメニューから、[Installed] 見出しを展開します。
3. AWS Toolkit for Visual Studio 2017 拡張機能を探し、[Uninstall]ボタンを選択します。
4. 画面上のプロンプトに従ってアンインストールプロセスを完了します。

AWS Toolkit for Visual Studio for Visual Studio 2013 または 2015 のアンインストール

AWS Toolkit for Visual Studio 2013 または 2015 をアンインストールするには、次の手順を実行します。

1. Windows のコントロールパネルから [プログラムと機能] を開きます。

> **Note**
>
> Windows のコマンドプロンプトまたは Windows の [実行] ダイアログから `appwiz.cpl` を実行すると、[プログラムと機能] をすぐに開くことができます。

2. インストールされたプログラムのリストから、[AWS Tools for Windows] のコンテキスト (右クリック) メニューを開きます。
3. [Uninstall] を選択し、プロンプトに従ってアンインストールプロセスを完了します。

> **Note**
>
> [Samples] ディレクトリはアンインストールプロセル中に削除されません。このディレクトリは、サンプルを変更した場合に保持されます。このディレクトリは手動で削除する必要があります。

への接続 AWS

以下のセクションでは、Amazon Q で AWS Toolkit for Visual Studio の使用を開始する方法について説明します。拡張機能のインストール後に Visual Studio を初めて起動すると、開始方法がエディタウィンドウに表示されます。[開始方法] タブから、次のアクションを実行できます。

- Amazon Q と AWS Toolkit を有効または無効にします。
- 新しい認証情報を追加して認証します。
- 既存の認証情報を使用して認証します。
- Amazon Q と AWS Toolkit の使用を開始するのに役立つドキュメントとチュートリアルにアクセスします。

前提条件

Amazon Q と AWS Toolkit の使用を開始するには、 AWS 認証情報を使用して認証する必要があります。以前に別の AWS ツールやサービス (など AWS Command Line Interface) を使用して 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、 AWS Toolkit によって認証情報が自動的に検出されます。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、サインアッププポータルから AWS アカウントにサインアップできます。 AWS新しい AWS アカウントの設定の詳細については、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。

Toolkit AWS から に接続する

Toolkit から AWS アカウントに接続するには AWS 、「開始方法」タブをいつでも開き、以下を完了します。

Visual Studio [開始方法] タブを開く

1. Visual Studio のメインメニューから [拡張機能] を展開し、[AWS Toolkit] サブメニューを展開します。
2. [開始方法] を選択します。
3. Visual Studio エディタウィンドウで [開始方法] タブが開きます。

[開始方法] タブには、2 つの主要なセクションがあります。

- 機能: このセクションでは、Amazon Q や AWS Toolkit などの機能を有効または無効にできます。
- ドキュメントとチュートリアル: 有効にした機能へのリファレンスのコレクションです。

> **Note**
>
> [ドキュメントとチュートリアル] セクションは、1 つ以上の機能が有効になっている場合にのみ表示されます。

Amazon Q Developer

[開始方法]タブの Amazon Q セクションから、Amazon Q を有効または無効にしたり、新しい接続を追加したり、別の AWS 接続に切り替えることができます。これらのアクションを表示またはアクセスする前に、Amazon Q を有効にする必要があります。Amazon Q を有効にするには、[有効化]ボタンをクリックします。

Amazon Q を無効にすると、Amazon Q のすべての機能と関数が Visual Studio から完全に削除されます。Amazon Q を有効にすると、[開始方法] タブで [Amazon Q の認証のセットアップ] が自動的に開きます。続行するには、プロフェッショナル層にアクセスするための AWS IAM アイデンティティセンター 認証情報、または無料利用枠にアクセスするための AWS ビルダー ID を使用して認証する必要があります。各階層オプションの詳細については、「Amazon Q Developer ユーザーガイド」の「Amazon Q Developer のサービス階層について」のトピックを参照してください。

続行するには、次のいずれかの手順を実行します。

IAM Identity Center によるプロフェッショナル階層認証

> **Note**
>
> プロフェッショナル階層での認証に必要な [プロファイル名]、[開始 URL]、[プロファイルリージョン]、または [SSO リージョン] フィールドは、通常、会社または組織の管理者によって提供されます。IAM アイデンティティセンター認証情報の詳細については、「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とは」のトピックを参照してください。

1. Amazon Q での開始方法: AWS ツールキット画面から、Amazon Q タイルのサインインボタンを選択して、Amazon Q 画面のセットアップ認証に移動します。
2. [Amazon Q の認証のセットアップ] 画面から、[プロファイル階層]セクションに移動し、必須フィールドに入力して、[接続] ボタンを選択します。
3. デフォルトのウェブブラウザで Authorize AWS リクエストポータルを開くことを確認します。
4. Authorize AWS リクエストポータルに必要なステップを完了すると、ブラウザを閉じて Visual Studio に戻ることが安全であると通知されます。
5. [開始方法] タブで Amazon Q が更新され、プロセスが完了したときに IAM アイデンティティセンターに接続していることが示されます。

AWS Builder ID を使用した無料利用枠認証

> **Note**
>
> AWS ビルダー ID の詳細については、「 サインインユーザーガイド」の AWS 「ビルダー ID でAWS サインイン」トピックを参照してください。

1. Amazon Q での開始方法: AWS ツールキット画面から、Amazon Q タイルのサインインボタンを選択して、Amazon Q 画面のセットアップ認証に移動します。
2. [Amazon Q の認証のセットアップ] 画面から、[無料利用枠] セクションに移動し、[サインアップ] または [サインイン] ボタンを選択します。
3. デフォルトのウェブブラウザで Authorize AWS リクエストポータルを開くことを確認します。
4. Authorize AWS リクエストポータルに必要なステップを完了すると、ブラウザを閉じて Visual Studio に戻ることが安全であると通知されます。

5. 「開始方法」タブで、Amazon Q が更新され、プロセスが完了したときに AWS Builder ID で接続されていることが表示されます。

IAM Identity Center または AWS Builder ID 認証情報を使用して認証したら、Visual Studio の Amazon Q にアクセスできます。さらに、[開始方法] タブで次のアクションを実行できます。

- サインアウト: すべての Amazon Q 機能から現在の認証情報の接続を切断します。Amazon Q は引き続き有効ですが、ほとんどの機能は機能しません。
- Amazon Q を無効にする: Visual Studio のすべての Amazon Q 機能を完全に無効にします。

AWS ツールキット

AWS ツールキットの開始方法タブの AWS ツールキットセクションから、 AWS ツールキットを有効または無効にしたり、新しい接続を追加したり、別の AWS 接続に切り替えることができます。これらのアクションを表示またはアクセスする前に、 AWS ツールキットを有効にする必要があります。Toolkit を有効にするには AWS 、有効化ボタンをクリックします。

Toolkit を有効にすると、 AWS Toolkit のセットアップ認証が Toolkit の開始方法タブに自動的にロードされます AWS 。 AWS 続行するには、[AWS IAM アイデンティティセンター] 認証情報または [IAM ユーザーロール] 認証情報を使用して認証する必要があります。

> **Note**
>
> IAM アイデンティティセンター認証情報の詳細については、「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とは」のトピックを参照してください。IAM ユーザーロールの認証情報の詳細については、「AWS SDK およびツールリファレンスガイド」の「AWS アクセスキー: 長期認証情報」のトピックを参照してください。

IAM アイデンティティセンターで認証して接続する

1. Amazon Q での開始方法: AWS ツールキット画面から、 AWS ツールキットタイルのサインインボタンを選択して、 AWS ツールキットの認証のセットアップ画面に移動します。
2. AWS ツールキットの認証のセットアップ画面で、プロファイルタイプのドロップダウンメニューから IAM アイデンティティセンター (シングルサインオンの後継) を選択します。

3. [既存のプロファイルから選択するか、新しく追加する] ドロップダウンメニューから、既存のプロファイルを選択するか、[新しいプロファイルを追加] を選択して新しいプロファイル情報を追加します。

> **Note**
>
> 既存のプロファイルを選択した場合は、ステップ 7 に進みます。

4. [プロファイル名] フィールドに、認証に使用する IAM アイデンティティセンターアカウントに関連付けられた **`profile name`** を入力します。
5. [開始 URL] テキストフィールドに、IAM アイデンティティセンターの認証情報にアタッチされている **`Start URL`** を入力します。
6. [プロファイルのリージョン (デフォルトは us-east-1)] ドロップダウンメニューから、認証に使用する IAM アイデンティティセンターユーザープロファイルで定義されている [プロファイルのリージョン] を選択します。
7. [SSO リージョン (デフォルトは us-east-1)] ドロップダウンメニューから、IAM アイデンティティセンターの認証情報で定義されている [SSO リージョン] を選択します。
8. [接続] ボタンを選択すると、デフォルトのウェブブラウザで [AWS Authorize リクエスト] サイトが開きます。
9. 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 Visual Studio に戻っても安全であることが通知されます。
10. [開始方法] タブの [AWS Toolkit] セクションが更新され、プロセスが完了したときに IAM アイデンティティセンターに接続していることが示されます。

IAM ユーザーロール認証情報を使用して認証および接続する

1. Amazon Q での開始方法: AWS ツールキット画面から、 AWS ツールキットタイルのサインインボタンを選択して、 AWS ツールキットの認証のセットアップ画面に移動します。
2. AWS ツールキットの認証のセットアップ画面で、プロファイルタイプのドロップダウンメニューから IAM ユーザーロールを選択します。
3. [既存のプロファイルから選択するか、新しく追加する] ドロップダウンメニューで、**`Add new profile`** を選択します。

> **Note**
>
> リストから既存のプロファイル名を選択する場合は、ステップ 8 に進みます。

4. [プロファイル名] テキストフィールドに、新しいプロファイルの名前を入力します。
5. [アクセスキー ID] テキストフィールドに、認証するプロファイルの **Access Key ID** を入力します。
6. [シークレットキー] テキストフィールドに、認証するプロファイルの **Secret Key** を入力します。
7. [ストレージの場所 (デフォルトでは共有認証情報ファイル)] ドロップダウンメニューから、認証情報を [共有認証情報] ファイルに保存するか、[.NET暗号化ストア] に保存するかを指定します。
8. [プロファイルリージョン (デフォルトは us-east-1)] ドロップダウンメニューから、認証するプロファイルにアタッチされている [パーティション] と [プロファイルリージョン] を選択します。
9. このプロファイルを AWS ストレージの場所に追加したり、認証したりするには、接続ボタンを選択します AWS。
10. [開始方法] タブの [AWS Toolkit] セクションが更新され、プロセスが完了すると、IAMユーザーロールの認証情報に接続したことが表示されます。

IAM Identity Center または IAM ユーザーロールの認証情報を使用して認証したら、Toolkit for Visual Studio の AWS Explorer にアクセスできます。また、[開始方法] タブから、[サインアウト] および [AWS Toolkit for Visual Studio with Amazon Q の無効化] ができます。

ドキュメントとチュートリアル

ドキュメントとチュートリアルセクションは、 AWS サービスおよび機能の設定に基づいてドキュメントとチュートリアルの提案で自動的に更新されます。これらの参照は、少なくとも 1 つの機能が有効になっている場合にのみ表示されます。

AWS Toolkit for Visual Studio のインストールに関する問題のトラブルシューティング

以下の情報によって、AWS Toolkit for Visual Studio のセットアップ中によく発生する問題を解決できることがわかっています。

AWS Toolkit for Visual Studio のインストール中にエラーが発生した場合、またはインストールが完了したかどうか不明な場合は、以下の各セクションの情報を確認してください。

Visual Studio のための管理者のアクセス許可

AWS Toolkit for Visual Studio 拡張機能には、すべての AWS サービスと機能にアクセスできるようにするための管理者のアクセス許可が必要です。

ローカル管理者のアクセス許可がある場合、管理者のアクセス許可が Visual Studio インスタンスに直接適用されない可能性があります。

管理者のアクセス許可で Visual Studio をローカルで起動するには、以下の手順を実行します。

1. Visual Studio アプリケーションランチャー (アイコン) を Windows から探します。
2. Visual Studio アイコンのコンテキスト (右クリック) メニューを開きます。
3. コンテキストメニューから [Run as administrator] を選択します。

管理者のアクセス許可で Visual Studio をリモートで起動するには、以下の手順を実行します。

1. Visual Studio のリモートインスタンスへの接続に使用しているアプリケーションのアプリケーションランチャーを Windows から探します。
2. アプリケーションのコンテキスト (右クリック) メニューを開きます。
3. コンテキストメニューから [Run as administrator] を選択します。

> **Note**
>
> プログラムをローカルで起動するか、リモートで接続するかにかかわらず、Windows では管理者認証情報の確認を求められることがあります。

インストールログの取得

上記の「管理者のアクセス許可」セクションの手順を完了し、管理者のアクセス許可で Visual Studio を実行しているか Visual Studio に接続していることが確認できた場合は、インストールログファイルを取得することでその他の問題を診断できます。

`.vsix` ファイルから AWS Toolkit for Visual Studio を手動でインストールし、インストールログファイルを生成するには、次の手順を実行します。

1. AWS Toolkit for Visual Studio のランディングページから [ダウンロード] リンクをクリックして、インストールする AWS Toolkit for Visual Studio のバージョンの `.vsix` ファイルを保存します。
2. Visual Studio のメインメニューから [ツール] ヘッダーを展開し、[Command Line] サブメニューを展開して、[Visual Studio Developer Command Prompt] を選択します。
3. [Visual Studio Developer Command Prompt] から、`vsixinstaller` コマンドを以下の形式で入力します。

   ```
   vsixinstaller /logFile:[file path to log file] [file path to Toolkit installation file]
   ```
4. `[file path to log file]` を、インストールログを作成したいディレクトリのファイル名とフルファイルパスに置き換えます。ファイルパスとファイル名を指定した `vsixinstaller` コマンドの例は、以下のようになります。

   ```
   vsixinstaller /logFile:C:\Users\Documents\install-log.txt [file path to AWSToolkitPackage.vsix]
   ```
5. `[file path to Toolkit installation file]` を、`AWSToolkitPackage.vsix` が置かれているディレクトリのフルファイルパスに置き換えます。

   Toolkit インストールファイルへのフルファイルパスを含む `vsixinstaller` コマンドの例は、以下のようになります。

   ```
   vsixinstaller /logFile:[file path to log file] C:\Users\Downloads\AWSToolkitPackage.vsix
   ```
6. ファイル名とパスが正しいことを確認し、`vsixinstaller` コマンドを実行します。

   完全な `vsixinstaller` コマンドの例は、以下のようになります。

   ```
   vsixinstaller /logFile:C:\Users\Documents\install-log.txt C:\Users\Downloads\AWSToolkitPackage.vsix
   ```

さまざまな Visual Studio 拡張機能のインストール

インストールログファイルを入手しても、インストールプロセスが失敗する理由がわからない場合は、その他の Visual Studio 拡張機能をインストールできるかどうかを確認してください。さまざまな Visual Studio 拡張機能をインストールすると、インストールの問題についてさらに詳しく知ることができます。どの Visual Studio 拡張機能もインストールできない場合、AWS Toolkit for Visual

Studio ではなく Visual Studio に関する問題のトラブルシューティングを行う必要があるかもしれません。

サポートへのお問い合わせ

このガイドに含まれるすべてのセクションを確認した後に、さらにリソースやサポートが必要な場合は、AWS Toolkit for Visual Studio Github Issues サイトで過去の問題を確認したり、新しい問題を作成したりできます。

迅速な解決のために以下をお試しください。

- 過去および現在の問題をチェックして、他のユーザーが同じような状況に見舞われていないか確認します。
- 問題に対処するために行った各手順について詳細にメモしておきます。
- AWS Toolkit for Visual Studio またはその他の拡張機能をインストールして取得したログファイルをすべて保存します。
- AWS Toolkit for Visual Studio のインストールログファイルを新しい問題に添付します。

プロファイルとウィンドウのバインド

Toolkit for Visual Studio のプロファイルとウィンドウのバインド

Toolkit for Visual Studio の発行ツール、ウィザード、およびその他の機能を使用するときは、次の点に注意してください。

- AWS Explorer ウィンドウは一度に 1 つのプロファイルおよびリージョンにバインドされます。AWS Explorer から開いたウィンドウはデフォルトで、バインドされたプロファイルとリージョンになります。
- 新しいウィンドウを開くと、AWS Explorer のインスタンスを使用して別のプロファイルまたはリージョンに切り替えることができます。
- Toolkit for Visual Studio の発行ツールと機能は、AWS Explorer で設定されたプロファイルとリージョンに自動的にデフォルト設定されます。
- 発行ツール、ウィザード、または機能で新しいプロファイルまたはリージョンを指定した場合、それ以降に作成されたすべてのリソースは、新しいプロファイルとリージョンの設定を引き続き使用します。
- 複数の Visual Studio のインスタンスが開いている場合は、各インスタンスが異なるプロファイルとリージョンにバインドされていることがあります。

- AWS Explorer では、最後に指定されたプロファイルとリージョンが保存され、最後に閉じた Visual Studio インスタンスの値が保持されます。

認証とアクセス

Amazon Q で AWS Toolkit for Visual Studio の使用を開始する AWS には、 で認証する必要はありません。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。Amazon Q のサービスと機能を使用してすべての AWS Toolkit for Visual Studio にアクセスするには、少なくとも 2 種類のアカウント認証が必要です。

1. AWS アカウントの AWS Identity and Access Management (IAM) またはAWS IAM アイデンティティセンター認証。ほとんどの AWS サービスとリソースは、IAM および IAM Identity Center を通じて管理されます。
2. AWS Builder ID は、他の特定の AWS サービスではオプションです。

以下のトピックには、各認証情報の種類と認証方法の詳細と設定手順が記載されています。

トピック

AWS の IAM Identity Center 認証情報 AWS Toolkit for Visual Studio

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 (SDKs「IAM Identity Center 認証AWS SDKs」セクションを参照してください。 AWS Toolkit for Visual Studio

からの IAM Identity Center による認証 AWS Toolkit for Visual Studio

IAM Identity Center プロファイルを `credentials`または `config` ファイルに追加 AWS Toolkit for Visual Studio して から IAM Identity Center で認証するには、次の手順を実行します。

1. 任意のテキストエディタで、 <hone-directory>\.aws\credentials ファイルに保存されている AWS 認証情報を開きます。
2. credentials file セクションの [default] から、名前付きの IAM アイデンティティセンタープロファイルのテンプレートを追加します。以下はテンプレートの例です。

> **Important**
>
> credential ファイルにエントリを作成するときは、credential ファイルの命名規則と矛盾するプロファイルという単語を使用しないでください。
> config ファイル内に名前付きプロファイルを設定する場合にのみ profile_ プレフィックスを含めます。

```
[sso-user-1]
sso_start_url = https://example.com/start
sso_region = us-east-2
sso_account_id = 123456789011
sso_role_name = readOnly
region = us-west-2
```

- **sso_start_url**: 組織の IAM アイデンティティセンターユーザーポータルを指す URL。
- **sso_region**: IAM Identity Center ポータルホストを含む AWS リージョン。これは、デフォルトの regionパラメータで後述する AWS リージョンとは異なる場合があります。
- **sso_account_id**: この IAM Identity Center ユーザーに付与するアクセス許可を持つ IAM ロールを含む AWS アカウント ID。
- **sso_role_name**: IAM アイデンティティセンターを経由して認証情報を得るために、このプロファイルを使用するときのユーザーのアクセス許可を定義する IAM ロールの名前。
- **region**: この IAM Identity Center ユーザーがサインインするデフォルトの AWS リージョン。

> **Note**
>
> aws configure sso コマンド AWS CLI を実行して、IAM Identity Center 対応プロファイルを に追加することもできます。このコマンドを実行したら、IAM Identity Center の開始

URL (`sso_start_url`) と、IAM Identity Center ディレクトリをホストする AWS リージョン (`region`) の値を指定します。
詳細については、 AWS Command Line Interface ユーザーガイドのAWS 「シングルサインオンを使用するように AWS CLI を設定する」を参照してください。

IAM アイデンティティセンターでサインインする

IAM アイデンティティセンタープロファイルでサインインする場合は、`credential file` で指定された `sso_start_url` でデフォルトのブラウザが起動されます。で AWS リソースにアクセスする前に、IAM Identity Center の口グインを確認する必要があります AWS Toolkit for Visual Studio。認証情報の有効期限が切れた場合は、新しい一時的な認証情報を取得するために接続プロセスを繰り返す必要があります。

AWS IAM 認証情報

AWS IAM 認証情報は、ローカルに保存されたアクセスキーを介して AWS アカウントで認証されます。

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。

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 を自動的に検出します。
- AWS では、 AWS IAM アイデンティティセンター 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、 などの ID プロバイダーとのフェデレーションを使用します AWS IAM アイデンティティセンター。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とは?」を参照してください。

「IAM ユーザーの作成」

AWS アカウントで認証 AWS Toolkit for Visual Studio 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た認証」トピックの「アクセスキーの取得」を完了する必要があります。 https://docs.aws.amazon.com/singlesignon/latest/userguide/what-is.html AWS SDKs

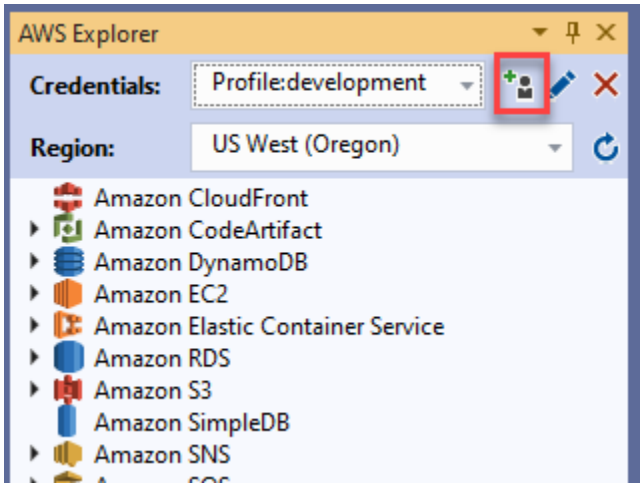
> **Note**
>
> 「ステップ 3: 共有認証情報ファイルの更新」はオプションです。
> ステップ 3 を完了すると、 は から認証情報 AWS Toolkit for Visual Studio を自動的に検出しますcredentials file。
> ステップ 3 を完了していない場合、 は、以下の セクションから認証情報ファイル AWS Toolkit for Visual Studioを作成するで説明credentials fileされているように、 を作成するプロセスを AWS Toolkit for Visual Studio 順を追って説明します。

認証情報ファイルの作成

AWS Toolkit for Visual Studioでユーザーを追加する、あるいは `credentials file` を作成するには、以下の手順を実行します。

> **Note**
>
> ツールキットで新しいユーザープロファイルを追加する場合:
>
> - `credentials file` が既に存在する場合は、新しいユーザー情報が既存のファイルに追加されます。
> - `credentials file` が存在しない場合は、新しいファイルが作成されます。

1. AWS Explorer から新しいアカウントプロファイルアイコンを選択し、新しいアカウントプロファイルダイアログを開きます。



2. [新しいアカウントプロファイル] ダイアログの必須フィールドに入力し、[OK] ボタンを選択して IAM ユーザーを作成します。

ツールキットで IAM ユーザー認証情報を編集する

ツールキットで IAM ユーザー認証情報を編集するには、以下のステップを実行します。

1. AWS Explorer の認証情報ドロップダウンから、編集する IAM ユーザー認証情報を選択します。
2. [プロファイルを編集] アイコンをクリックして、[プロファイルを編集] ダイアログを開きます。
3. [プロファイルを編集] ダイアログで更新を完了し、[OK] ボタンを選択して変更を保存します。

ツールキットで IAM ユーザー認証情報を削除するには、以下のステップを実行します。

1. AWS Explorer の認証情報ドロップダウンから、削除する IAM ユーザー認証情報を選択します。
2. [プロファイルを削除] アイコンをクリックして、[プロファイルを削除] プロンプトを開きます。
3. `Credentials file` からプロファイルを削除することを確認します。

> **Important**
>
> [プロファイルを編集] ダイアログで IAM アイデンティティセンターや多要素認証 (MFA) など、高度なアクセス機能をサポートするプロファイルは、 AWS Toolkit for Visual Studioから編集できません。これらのタイプのプロファイルを変更するには、テキストエディタを使用して `credentials file` を編集する必要があります。

テキストエディタで IAM ユーザー認証情報を編集する

を使用して IAM ユーザーを管理するだけでなく AWS Toolkit for Visual Studio、任意のテキストエディタcredential filesから編集することもできます。Windows の場合、credential file のデフォルトの場所は C:\Users*USERNAME*\.aws\credentials です。

credential files の場所と構造の詳細については、「AWS SDK とツールのリファレンスガイド」の「Shared config and credentials files」セクションを参照してください。

AWS Command Line Interface (AWS CLI) からの IAM ユーザーの作成

AWS CLI は、 コマンド credentials fileを使用して で IAM ユーザーを作成するために使用できるもう 1 つのツールですaws configure。

から IAM ユーザーを作成する方法の詳細については、AWS CLI 「 ユーザーガイド」の「トピックの設定 AWS CLI AWS CLI 」を参照してください。

Toolkit for Visual Studio は、次の構成プロパティをサポートしています。

```
aws_access_key_id
aws_secret_access_key
aws_session_token
credential_process
credential_source
external_id
mfa_serial
role_arn
role_session_name
source_profile
sso_account_id
sso_region
sso_role_name
sso_start_url
```

AWS ビルダー ID

AWS ビルダー ID は、Amazon CodeCatalyst でサードパーティーリポジトリのクローンを作成するなど、特定のサービスまたは機能を使用するために必要な追加の AWS 認証方法です。

AWS ビルダー ID 認証方法の詳細については、「 サインインユーザーガイド」の AWS 「ビルダー ID でAWS サインイン」トピックを参照してください。

CodeCatalyst のリポジトリのクローン作成の詳細については AWS Toolkit for Visual Studio、このユーザーガイドの「Amazon CodeCatalyst の使用」トピックを参照してください。

Toolkit for Visual Studio での多要素認証 (MFA)

多要素認証 (MFA) は AWS 、アカウントの追加のセキュリティです。MFA では、ウェブサイト AWS またはサービスにアクセスするときに、 AWS サポートされている MFA メカニズムからサインイン認証情報と一意の認証を提供する必要があります。

AWS は、MFA 認証用の仮想デバイスとハードウェアデバイスの両方をサポートしています。以下にスマートフォンアプリケーションを通じて有効化された仮想 MFA デバイスの例を示します。MFAデバイスオプションの詳細については、IAM ユーザーガイドの「AWSでの多要素認証 (MFA) の使用」を参照してください。

ステップ 1: IAM ユーザーにアクセス権を委任するための IAM ロールの作成

以下の手順では、IAM ユーザーにアクセス許可を割り当てるロールの委任を設定する方法について説明します。ロールの委任の詳細については、「AWS Identity and Access Management ユーザーガイド」の「IAM ユーザーにアクセス許可を委任するロールの作成」トピックを参照してください。

1. https://console.aws.amazon.com/iam/home の IAM コンソールに移動します。
2. ナビゲーションバーで [Roles] (ロール) を選択した後、[Create Role] (ロールの作成) を選択します。
3. [ロールの作成] ページで、[もう 1 つの AWS アカウント] を選択します。
4. 必要な [Account ID] (アカウント ID) を入力し、[Require MFA] (MFA が必要な) チェックボックスをオンにします。

   > **Note**
   >
   > 12 桁のアカウント ID 番号を確認するには、コンソールのナビゲーションバーで [Support]、[Support Center] の順に選択します。

5. [Next: Permissions] (次のステップ: 許可) を選択します。
6. 既存のポリシーをロールにアタッチするか、新しいポリシーを作成します。このページで選択するポリシーによって、IAM ユーザーが Toolkit でアクセスできる AWS サービスが決まります。

7. ポリシーをアタッチしたら、ロールに IAM タグを追加するオプションのために、[Next: Tags] (次へ:タグ) を選択します。[Next: Review] (次へ: 確認) を選択して続行します。
8. [Review] (確認) ページで、必須の [Role name] (ロール名) (例えば、toolkit-role) を入力します。オプションの [Role description] (ロールの説明) 値を追加することもできます。
9. [ロールの作成] を選択してください。
10. 確認メッセージ (「ロール toolkit-role が作成されました」など) が表示されたら、メッセージ内のロールの名前を選択します。
11. [Summary] (概要) ページで、[Copy] (コピー) アイコンを選択して [Role ARN] (ロールの ARN) にコピーし、ファイルに貼り付けます。この ARN は、ロールを引き受けるように IAM ユーザーを設定するときに必要です。

ステップ 2: ロールのアクセス許可を引き受ける IAM ユーザーの作成

このステップでは、インラインポリシーを追加できるように、アクセス許可のない IAM ユーザーを作成します。

1. https://console.aws.amazon.com/iam/home の IAM コンソールに移動します。
2. ナビゲーションバーで、 [Users] (ユーザー)、[Add user] (ユーザーを追加する) の順に選択します。
3. [ユーザーの追加] ページで必要な [ユーザー名] (例えば、toolkit-user) を入力し、[プログラムによるアクセス] チェックボックスをオンにします。
4. [Next: Permissions] (次へ: アクセス許可)、[Next: Tags] (次へ: タグ)、[Next: Review] (次へ: 確認) の順に選択して、次ページに移動します。ユーザーがロールの権限を引き受けるため、この段階では権限を追加しません。
5. [Review] (確認) ページでは、[This user has no permissions] (このユーザーにはアクセス許可がありません) という旨が通知されます。[ユーザーの作成] を選択します。
6. [Success] (成功) ページで、[Download .csv] (.csv をダウンロード) を選択して、アクセスキー ID およびシークレットアクセスキーを含むファイルをダウンロードします。(認証情報ファイルのプロファイルを定義する場合は、両方とも必要です。)
7. [閉じる] を選択してください。

ステップ 3: IAM ユーザーがロールを引き受けることを許可するポリシーを追加する

次の手順では、ユーザーがロール (およびそのロールのアクセス許可) を引き受けることを許可するインラインポリシーを作成します。

1. IAM コンソールの [Users] (ユーザー) ページで、作成した IAM ユーザー (例えば、toolkit-user) を選択します。
2. [Summary] (概要) ページの[Permissions] (アクセス許可) タブで [Add inline policy] (インラインポリシーの追加 を選択します。
3. [ポリシーの作成] ページで、[サービスを選択] を選択し、[サービスを探す] で [STS] を入力し、結果から [STS] を選択します。
4. [Actions] (アクション) に AssumeRole という語を入力し始めます。表示されたらAssumeRoleチェックボックスをオンにします。
5. [Resource] (リソース) セクションで、[Specific] (特定) が選択されていることを確認し、アクセスを制限するため [Add ARN] (ARN の追加) を選択します。
6. [Add ARN(s)] (ARN を追加する) ダイアログボックスの [Specify ARN for role] (ロールの ARN を指定します) にステップ 1 で作成したロールの ARN を追加します。

   ロールの ARN を追加すると、そのロールに関連付けられた信頼済みアカウントおよびロール名が [Account] (アカウント) および [Role name with path] (パス付きのロール名) に表示されます。
7. [Add] (追加) を選択します。
8. [ポリシーの作成] ページに戻り、[リクエスト条件の指定 (オプション)] を選択し、[MFA が必要] チェックボックスをオンにしてから、確認するため [閉じる] を選択します。
9. [Review poilicy] (ポリシーの確認) を選択します。
10. [Review poilicy] (ポリシーの確認) ページで、ポリシーの [Name] (名前) を入力してから [Create policy] (ポリシーの作成) を選択します。

    [Permissions] (アクセス許可) タブには、IAM ユーザーに直接アタッチされた新しいインラインポリシーが表示されます。

ステップ 4: IAM ユーザーの仮想 MFA デバイスの管理

1. 仮想 MFA アプリケーションをスマートフォンにダウンロードしてインストールします。

サポートされているアプリケーションのリストについては、「多要素認証」リソースページを参照してください。

2. IAM コンソールで、ナビゲーションバーから、ユーザーを選択し、ロールを引き受けるユーザーを選択します (今回は toolkit-user)。
3. [Summary] (概要) ページで [Security credentials] (セキュリティ認証情報) タブを選択し、[Assigned MFA device] (割り当てられた MFA デバイス) に対して [Manage] (管理) を選択します。
4. [Manage MFA device] (MFA デバイスの管理) ペーンで、[Virtual MFA device] (仮想 MFA デバイス)、[Continue] (続行) の順に選択します。
5. [Set up virtual MFA device] (仮想 MFA デバイスの設定) ペインで [Show QR code] (QR コードを表示する) を選択してからスマートフォンにインストールした仮想 MFA アプリケーションを使用してコードをスキャンします。
6. QR コードをスキャンすると、仮想 MFA アプリケーションはワンタイムの MFA コードを生成します。[MFA code 1] (MFA コード 1) および [MFA code 2] (MFA コード 2) に 2 つの連続した MFAコードを入力します。
7. MFA の割り当てを選択します。
8. ユーザーの [Security credentials] (セキュリティ認証情報) タブに戻り、新しく[Assigned MFA device] (割り当てられた MFA デバイス) のARNをコピーします。

   ARN には 12 桁のアカウント ID が含まれ、形式は次の `arn:aws:iam::123456789012:mfa/toolkit-user` のようになります。この ARN は、次のステップで MFA プロファイルを定義するときに必要になります。

ステップ 5: MFA を許可するプロファイルの作成

次の手順では、Toolkit for Visual Studio から AWS サービスにアクセスするときに MFA を許可するプロファイルを作成します。

作成したプロファイルには、前の手順でコピーして保存した 3 つの情報が含まれています。

- IAM ユーザーのアクセスキー (アクセスキー ID およびシークレットアクセスキー)
- IAM ユーザーにアクセス許可を委任しているロールの ARN
- IAM ユーザーに割り当てられている仮想 MFA デバイスの ARN

認証情報を含む AWS 共有 AWS 認証情報ファイルまたは SDK ストアで、次のエントリを追加します。

```
[toolkit-user]
aws_access_key_id = AKIAIOSFODNN7EXAMPLE
aws_secret_access_key = wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY

[mfa]
source_profile = toolkit-user
role_arn = arn:aws:iam::111111111111:role/toolkit-role
mfa_serial = arn:aws:iam::111111111111:mfa/toolkit-user
```

この例では、次の 2 つのプロファイルが定義されています：

- [toolkit-user] プロファイルには、ステップ 2 で IAM ユーザーを作成したときに生成され、保存されたアクセスキーおよびシークレットアクセスキーが含まれます。
- [mfa] プロファイルは、多要素認証のサポート方法を定義します。3 つのエントリがあります：

  ◦ source_profile: このプロファイル内のこの role_arn 設定で指定されたロールを継承するために使用される認証情報のプロファイルを指定します。この場合は、toolkit-user プロファイルです。

  ◦role_arn: このプロファイルを使用してリクエストされた操作の実行に使用する IAM ロールの Amazon リソースネーム (ARN) を指定します。この場合、ステップ 1 で作成したロールの ARN です。

  ◦mfa_serial: ロールを引き受けるときに使用する必要がある MFA デバイスの ID もしくはシリアル番号を指定します。この場合、ステップ 3 で設定した仮想デバイスの ARN です。

外部認証情報の設定

AWSで直接サポートされていない認証情報を生成または参照する方法がある場合、credential_process 設定を含むプロファイルを共有認証ファイルに追加することができます。この設定は、コマンドに使用する認証情報を生成もしくは取得するために実行する外部のコマンドを指定します。例えば、config ファイルに次のように似たエントリを含めることができます。

```
[profile developer]
credential_process = /opt/bin/awscreds-custom --username helen
```

外部認証情報の使用および関連するセキュリティリスクの詳細については、「AWS Command Line Interface ユーザーガイド」の「外部プロセスを使用した認証情報の調達」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 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する必要があります。 AWS Toolkit with Amazon Q のファイアウォールおよびプロキシ設定のトラブルシューティング方法の詳細については、このユーザーガイドの「トラブルシューティング」トピックの「ファイアウォールおよびプロキシ設定」セクションを参照してください。Amazon Q の企業プロキシの設定の詳細については、「Amazon Q Developer ユーザーガイド」の「Amazon Q での企業プロキシの設定」のトピックを参照してください。

AWS Toolkit for Visual Studio 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 特定のエンドポイントとリファレンスのリストです。

エンドポイント

```
https://idetoolkits-hostedfiles.amazonaws.com/*
https://idetoolkits.amazonwebservices.com/*
http://vstoolkit.amazonwebservices.com/*
https://aws-vs-toolkit.s3.amazonaws.com/*
https://raw.githubusercontent.com/aws/aws-toolkit-visual-studio/main/version.json
https://aws-toolkit-language-servers.amazonaws.com/*
```

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 auto-complete 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 のサービスを使用する

以下のトピックでは、AWS Toolkit for Visual Studio with Amazon Q から AWS のサービスの使用を開始する方法について説明します。

トピック

Amazon Q を使用した AWS Toolkit for Visual Studio の 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。Amazon Q で AWS Toolkit for Visual Studio を使用すると、Amazon Q で AWS Toolkit

for Visual Studio から直接 CodeCatalyst リソースを表示および管理できます。CodeCatalyst 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。

以下のトピックでは、 AWS Toolkit for Visual Studio を CodeCatalyst で Amazon Q に接続する方法と、 AWS Toolkit for Visual Studio を Amazon Q で使用して CodeCatalyst を操作する方法について説明します。

トピック

Amazon Q を使用した Amazon CodeCatalyst と AWS Toolkit for Visual Studio の開始方法

AWS Toolkit for Visual Studio with Amazon Q から Amazon CodeCatalyst の使用を開始するには、次の手順を実行します。

トピック

Amazon Q を使用した AWS Toolkit for Visual Studio のインストール

AWS Toolkit for Visual Studio を Amazon Q と CodeCatalyst アカウントを統合する前に、Amazon Q で AWS Toolkit for Visual Studio の最新バージョンを使用していることを確認してください。Amazon Q で AWS Toolkit for Visual Studio の最新バージョンをインストールおよびセットアップする方法の詳細については、このユーザーガイドの「Amazon Q で AWS Toolkit for Visual Studio をセットアップする」セクションを参照してください。

CodeCatalyst アカウントと AWS ビルダー ID の作成

Amazon Q で AWS Toolkit for Visual Studio の最新バージョンをインストールするだけでなく、Amazon Q で AWS Toolkit for Visual Studio に接続するには、アクティブな AWS Builder ID と

CodeCatalyst アカウントが必要です。アクティブな AWS Builder ID または CodeCatalyst アカウントがない場合は、CodeCatalyst ユーザーガイド」の「CodeCatalyst のセットアップ」セクションを参照してください。 CodeCatalyst

> **Note**
>
> AWS Builder ID は AWS 認証情報とは異なります。 AWS Builder ID でサインアップして認証する方法については、このユーザーガイドの「Authentication and access: AWS Builder ID」トピックを参照してください。
> AWS Builder IDs「 AWS 全般のリファレンスユーザーガイド」のAWS 「Builder ID」トピックを参照してください。

AWS Toolkit for Visual Studio と Amazon Q with CodeCatalyst の接続

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

1. Visual Studio の [Git] メニュー項目から、[Clone Repository...] を選択します。
2. [Browse a Repository」 セクションから、プロバイダーとして [Amazon CodeCatalyst] を選択します。
3. 接続セクションから、 AWS ビルダー ID で接続 を選択して、任意のウェブブラウザで CodeCatalyst コンソールを開きます。
4. ブラウザから、提供されたフィールドに AWS Builder ID を入力し、手順に従って続行します。
5. プロンプトが表示されたら、Amazon Q を使用した AWS Toolkit for Visual Studio と CodeCatalyst アカウント間の接続の確認を許可するを選択します。接続プロセスが完了すると、CodeCatalyst にブラウザを閉じても問題ないことを示す確認メッセージが表示されます。

AWS Toolkit for Visual Studio から Amazon Q で Amazon CodeCatalyst リソースを使用する

以下のセクションでは、Amazon Q で AWS Toolkit for Visual Studio で使用できる Amazon CodeCatalyst リソース管理機能の概要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

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あるため、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。ローカルでプロジェクトで作業するには、CodeCatalyst リポジトリをローカルマシンにクローンして、次にクラウドに接続したときに CodeCatalyst プロジェクトと同期します。

リポジトリをローカルマシンにクローンするには、以下の手順を実行します。

1. Visual Studio の [Git] メニュー項目から、[Clone Repository...] を選択します。
2. [Browse a Repository」 セクションから、プロバイダーとして [Amazon CodeCatalyst] を選択します。

> **Note**
>
> Connection セクションに`Not Connected`メッセージが表示された場合は、続行する前に、このユーザーガイドの「Authentication and access: AWS Builder ID」セクションのステップを完了してください。

3. リポジトリのクローン元の [Space] と [Project] を選択します。
4. [Repositories] セクションから、クローンするリポジトリを選択します。
5. [Path] セクションから、リポジトリのクローン先のフォルダーを選択します。

> **Note**
>
> クローンを正常に作成するには、このフォルダーは最初は空でなければなりません。

6. [Clone] を選択してリポジトリのクローンを開始します。
7. リポジトリのクローンが作成されると、Visual Studio はクローンされたソリューションをロードします。

> **Note**
>
> Visual Studio が、クローンされたリポジトリでソリューションを開かない場合は、[Source Control] メニューの [Git Global Settings] にある [Automatically load the solution when opening a Git repository] 設定から、Visual Studio のオプションを調整できます。

トラブルシューティング

以下は、Amazon Q で AWS Toolkit for Visual Studio から Amazon CodeCatalyst を使用する際の既知の問題に対処するためのトラブルシューティングトピックです。

トピック

認証情報

CodeCatalyst から Git ベースのリポジトリをクローンしようとしたときに認証情報を求めるダイアログが表示される場合、[AWS CodeCommit 認証情報ヘルパー] がグローバルに設定されており、CodeCatalyst に干渉している可能性があります。 AWS CodeCommit 認証情報ヘルパーの詳細については、AWS CodeCommit ユーザーガイド」の「CLI 認証情報ヘルパーを使用した Windows AWS 上の CodeCommit リポジトリへの HTTPS 接続のセットアップ手順」セクションを参照してください。 AWS CodeCommit

[AWS CodeCommit 認証情報ヘルパー] の処理を CodeCommit URL のみに制限するには、次の手順を実行します。

1. 以下の場所にあるグローバル Git 設定ファイルを開きます。`%userprofile%\.gitconfig`
2. ファイル内で以下のセクションを見つけます。

```
[credential]
    helper = !aws codecommit credential-helper $@
    UseHttpPath = true
```

3. セクションを以下のように変更します。

```
[credential "https://git-codecommit.*.amazonaws.com"]
    helper = !aws codecommit credential-helper $@
    UseHttpPath = true
```

4. 変更を保存し、手順を完了してリポジトリをクローンします。

Amazon CloudWatch Logs の Visual Studio への統合

AWS Toolkit for Visual Studio with Amazon Q を使用して Amazon CloudWatch Logs を統合することで、IDE を離れることなく CloudWatch Logs リソースをモニタリング、保存、およびアクセスできます。CloudWatch サービスのセットアップと CloudWatch Logs 機能の操作方法の詳細については、以下のトピックから選択してください。

トピック

CloudWatch Logs の Visual Studio への統合をセットアップ

AWS Toolkit with Amazon Q を使用して Amazon CloudWatch Logs の統合を行うには、AWS アカウントが必要です。AWS サインインサイトから新しい AWS アカウントを作成できます。AWS Toolkit with Amazon Q を使用して利用できる CloudWatch Logs 機能のほとんどが、アクティブな AWS 認証情報を使用してアクセスできます。特定の機能のために求められる追加設定の要件は、「CloudWatch Logs の操作」ガイドの関連セクションに記載されています。

CloudWatch ログのセットアップに関する詳細やオプションについては、「Amazon CloudWatch Logs ガイド」の「セットアップ」セクションを参照してください。

Visual Studio 上での CloudWatch Logs の操作

Amazon CloudWatch Logs の統合により、 AWS Toolkit for Visual Studio から Amazon Q を使用して CloudWatch Logs をモニタリング、保存、アクセスできます。IDE を離れることなく CloudWatch Logs 機能にアクセスできるため、CloudWatch Logs の開発プロセスが簡素化され、ワークフローの中断が軽減され、効率が向上します。以下のトピックでは、CloudWatch Logs の統合の基本的な機能と関数の使用方法について説明します。

トピック

CloudWatch ロググループ

`log group` 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 `log streams` のグループです。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。

ロググループの表示

`View Log Groups` 機能によって CloudWatch ロググループエクスプローラーにロググループの一覧が表示されます。

View Log Groups 機能にアクセスして CloudWatch ロググループエクスプローラーを開くには、以下のステップを実行します。

1. AWS Explorer から Amazon Amazon CloudWatchを展開します。
2. [ロググループ] をダブルクリックするか、コンテキスト (右クリック) メニューを開いて [表示] を選択し、[CloudWatch ロググループエクスプローラー] を開きます。

> **Note**
>
> CloudWatch ロググループエクスプローラーは、ソリューションエクスプローラーと同じウィンドウの場所で開きます。

ロググループのフィルタリング

個々のアカウントには、何千もの異なるロググループを含めることができます。特定のグループを簡単に検索するには、以下の `filtering` 機能を使用してください。

1. [CloudWatch ロググループエクスプローラー] で、ウィンドウの上部にある検索バーにカーソルを置きます。
2. 探しているロググループに関連するプレフィックスを入力し始めます。
3. [CloudWatch ロググループエクスプローラー] が、前のステップで指定した検索語と一致する結果を表示するように自動的に更新されます。

ロググループの削除

特定のロググループを削除するには、以下の手順を参照してください。

1. [CloudWatch ロググループエクスプローラー] で、削除するロググループを右クリックします。
2. プロンプトが表示されたら、現在選択されているロググループを削除することを確認します。
3. [はい] ボタンをクリックすると、選択したロググループが削除され、[CloudWatch ロググループエクスプローラー] が更新されます。

ロググループの更新

[CloudWatch ロググループエクスプローラー] に表示されている現在のロググループのリストを更新するには、[ツールバー] にある [更新] アイコンをクリックします。

ロググループの ARN をコピー

特定のロググループの ARN をコピーするには、以下の手順を実行します。

1. [CloudWatch ロググループエクスプローラー] で、ARN のコピー元のロググループを右クリックします。
2. メニューから [ARN をコピー] オプションを選択します。
3. ARN がローカルクリップボードにコピーされ、貼り付けできる状態になりました。

CloudWatch ログストリーム

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。

> **Note**
>
> ログストリームを表示するときは、次のプロパティについて注意してください。
>
> - デフォルトでは、ログストリームは最新のイベントのタイムスタンプでソートされます。
> - ログストリームに関連する列は、列ヘッダーにある [キャレット] を切り替えることで、昇順または降順に並べ替えることができます。
> - フィルタリングされたエントリは、[ログストリーム名] でのみソートできます。

ログストリームの表示

1. [CloudWatch ロググループエクスプローラー] で、ロググループをダブルクリックするか、ロググループを右クリックしてコンテキストメニューから [ログストリームを表示] を選択します。

2. [ドキュメント] ウィンドウに新しいタブが開き、ロググループに関連するログストリームのリストが表示されます。

ログストリームのフィルタリング

1. [ドキュメント] ウィンドウの [ログストリーム] タブで、検索バーにカーソルを置きます。
2. 探しているログストリームに関連するプレフィックスを入力し始めます。
3. 入力すると、現在の表示が自動的に更新され、入力によってログストリームがフィルタリングされます。

ログストリームの更新

[ドキュメント] ウィンドウに表示されている現在のログストリームのリストを更新するには、[検索バー] の横の [ツールバー] にある [更新] アイコンをクリックします。

ログストリームの ARN をコピー

特定のログストリームの ARN をコピーするには、以下の手順を実行します。

1. [ドキュメント] ウィンドウの [ログストリーム] タブから、ARN のコピー元のログストリームを右クリックします。
2. メニューから [ARN をコピー] オプションを選択します。
3. ARN がローカルクリップボードにコピーされ、貼り付けできる状態になりました。

ログストリームのダウンロード

[ログストリームをエクスポート] 機能は、選択したログストリームをダウンロードしてローカルに保存し、追加の処理のためにカスタムツールやソフトウェアからアクセスできるようにします。

1. [ドキュメント] ウィンドウの [ログストリーム] タブから、ダウンロードするログストリームを右クリックします。
2. [ログストリームをエクスポート] を選択して、[テキストファイルにエクスポート] ダイアログを開きます。
3. ファイルをローカルに保存する場所を選択し、表示されるテキストフィールドで名前を指定します。
4. [OK] を選択してダウンロードを確定します。ダウンロードのステータスは [Visual Studio タスクステータスセンター] に表示されます。

CloudWatch ログイベント

ログイベントは、CloudWatch で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。

ログイベントのアクション

ログイベントはテーブルとして表示されます。デフォルトでは、イベントは最も古いイベントから最新のイベントまでソートされます。

以下のアクションは Visual Studio 上のログイベントに関連付けられています。

- ﾃキスト折り返しモード: イベントをクリックすることで、テキストの折り返しを切り替えることができます。
- テキスト折り返しボタン: このボタンは `document window` **`toolbar`** にあり、すべてのエントリのテキスト折り返しのオン/オフを切り替えることができます。
- メッセージをクリップボードにコピー: コピーするメッセージを選択し、選択したメッセージを右クリックして [コピー] (キーボードショートカット `Ctrl + C`) を選択します。

ログイベントの表示

1. [ドキュメント] ウィンドウから、ログストリームのリストを含むタブを選択します。
2. ログストリームをダブルクリックするか、ログストリームを右クリックして、メニューから [ログストリームを表示] を選択します。
3. [ドキュメント] ウィンドウに新しい [ログイベント] タブが開き、選択したログストリームに関連するログイベントのテーブルが表示されます。

ログイベントのフィルタリング

ログイベントをフィルタリングする方法は、コンテンツ、時間範囲、またはその両方の 3 つあります。ログイベントをコンテンツと時間範囲の両方でフィルタリングするには、まずコンテンツまたは時間範囲のいずれかでメッセージをフィルタリングし、次にその結果をもう一方の方法でフィルタリングします。

ログイベントをコンテンツでフィルタリングするには、以下の手順を実行します。

1. [ドキュメント] ウィンドウの [ログイベント] タブから、ウィンドウ上部にある検索バーにカーソルを置きます。

2. 検索するログイベントに関連する用語または語句を入力し始めます。
3. 入力すると、現在の表示でログイベントのフィルタリングが自動的に開始されます。

> **Note**
>
> フィルターパターンでは大文字と小文字が区別されます。英数字以外の文字を二重引用符 (* "" *) で囲み、完全に一致する用語や語句を囲むことで、検索結果の精度を高めることができます。フィルタパターンの詳細については、「Amazon CloudWatch ガイド」の「フィルタとパターンの構文」トピックを参照してください。

特定の時間範囲に生成されたログイベントを表示するには、以下の手順を実行します。

1. [ドキュメント] ウィンドウの [ログイベント] タブから、[ツールバー] にある [カレンダー] アイコンをクリックします。
2. 表示されるフィールドを使用して、検索する時間範囲を指定します。
3. フィルタリングされた結果は、日付と時刻の制約を指定すると自動的に更新されます。

> **Note**
>
> [フィルターをクリア] オプションを使用すると、現在の日付と時刻のフィルター選択がすべてクリアされます。

ログイベントの更新

[ログイベント] タブに表示されている現在のログイベントのリストを更新するには、[ツールバー] にある [更新] アイコンをクリックします。

CloudWatch Logs へのその他のアクセス

他の AWS サービスやリソースに関連付けられた CloudWatch Logs には、Visual Studio の AWS Toolkit から直接アクセスできます。

Lambda

Lambda 関数に関連付けられているログストリームを表示するには、以下の手順を実行します。

> **Note**
>
> Lambda 実行ロールには、CloudWatch Logs にログを送信するための適切なアクセス許可が必要です。CloudWatch Logs に必要な Lambda のアクセス許可の詳細については、「https://docs.aws.amazon.com/lambda/latest/dg/monitoring-cloudwatchlogs.html#monitoring-cloudwatchlogs-prereqs」を参照してください。

1. AWS Toolkit Explorer から Lambda を展開します。
2. 表示したい関数を右クリックして [ログを表示] を選択すると、[ドキュメント] ウィンドウに関連するログストリームが表示されます。

Lambda 統合 `function view` を使用してログストリームを表示するには、以下の手順を実行します。

1. AWS Toolkit Explorer から Lambda を展開します。
2. 表示したい関数を右クリックして [関数を表示] を選択すると、[ドキュメント] ウィンドウに関数ビューが表示されます。
3. `function view` から [ログ] タブに切り替えると、選択した Lambda 関数に関連するログストリームが表示されます。

ECS

ECS タスクコンテナに関連付けられているログリソースを表示するには、以下の手順を実行します。

> **Note**
>
> Amazon ECS サービスが CloudWatch にログを送信するには、特定の Amazon ECS タスクの各コンテナが必要な設定を満たしていなければなりません。必要なセットアップと設定の詳細については、AWS 「 ログドライバーを使用する」ガイドを参照してください。

1. AWS Toolkit Explorer から、Amazon ECS を展開します。
2. 表示する Amazon ECS クラスターを選択すると、[ドキュメント] ウィンドウに新しい [ECS クラスター] タブが開きます。

3. [ECS クラスター] タブの左側にあるナビゲーションメニューから [タスク] を選択して、クラスターに関連するすべてのタスクを一覧表示します。
4. [タスク] 画面からタスクを選択し、左下隅にある [ログを表示] のリンクをクリックします。

> **Note**
>
> この表示には、クラスターに含まれるすべてのタスクが一覧表示されます。`View Logs` リンクは、必要なログ設定を満たす各タスクにのみ表示されます。
>
> - タスクが 1 つのコンテナにのみ関連付けられている場合、[ログを表示] リンクをクリックするとそのコンテナのログストリームが開きます。
> - タスクが複数のコンテナに関連付けられている場合、[ログを表示] リンクをクリックすると [ECS タスクの CloudWatch Logs を表示] ダイアログが開きます。[コンテナ] ドロップダウンメニューから、ログを表示したいコンテナを選択し、[OK] を選択します。

5. [ドキュメント] ウィンドウに新しいタブが開き、選択したコンテナに関連するログストリームが表示されます。

Amazon EC2 インスタンスの管理

AWS Explorer では、Amazon Machine Image (AMI) および Amazon Elastic Compute Cloud (Amazon EC2) インスタンスの詳しいビューが提供されています。これらのビューで、AMI から Amazon EC2 インスタンスを起動、そのインスタンスに接続、インスタンスを停止または終了できます。すべて Visual Studio 開発環境から可能です。インスタンスビューを使用して、使用するインスタンスから AMI を作成できます。詳細については、「Amazon EC2 インスタンスから AMI を作成する」を参照してください。

Amazon マシンイメージビューと Amazon EC2 インスタンスビュー

AWS Explorer から、Amazon Machine Images (AMI) と Amazon EC2 インスタンスのビューを表示できます。AWS Explorer で、Amazon EC2 ノードを展開します。

AMI のビューを表示するには、最初の、[AMIs (AMI)] サブノードでコンテキスト (右クリック) メニューを開き、[View (新規)] を選択します。

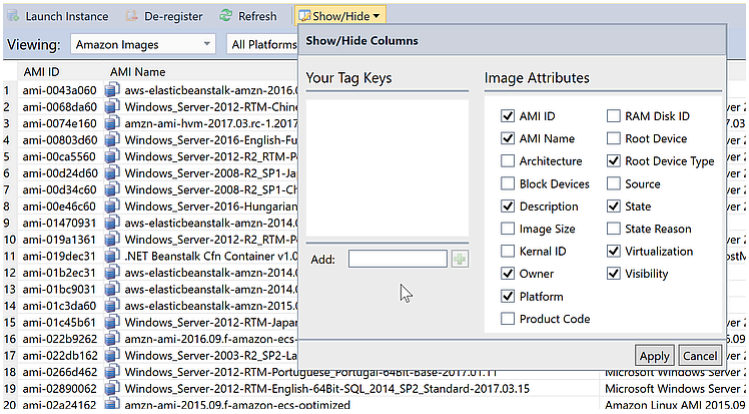
Amazon EC2 インスタンスビューを表示するには、[Instances (インスタンス)] ノードで、コンテキスト (右クリック) メニューを開き、[View (ビュー)] を選択します。

いずれのビューも、該当するノードをダブルクリックして、表示することもできます。

- ビューのスコープは、AWS Explorer で指定したリージョン (例えば、米国西部 (北カリフォルニア) リージョン) になります。
- クリックしてドラッグすると列をソートすることができます。列で値をソートするには、列見出しをクリックします。
- [Viewing (表示する)] のフィルタボックスとドロップダウンリストを使用してビューを設定できます。初期ビューには任意のプラットフォームタイプ (Windows もしくは Linux) の AMI が表示されます。これらは AWS Explorer で指定されたアカウントが所有しているものです。

列の表示/非表示

[Show/Hide (表示/非表示)] ドロップダウンをビューの上部で選択して、表示する列の種類を設定することもできます。ビューを閉じて、再度開いた場合でも、表示する列の選択は保持されます。



AMI とインスタンスビューの [Show/Hide Columns (列の表示/非表示)] UI

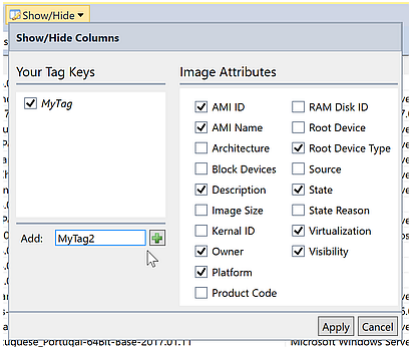
AMI、インスタンス、ボリュームのタグ付け

[Show/Hide] (表示/非表示) ドロップダウンリストを使用して、AMI、Amazon EC2 インスタンス、自分のボリュームにタグを追加することもできます。タグとは名前と値のペアで、インスタンス、ボリューム、AMI にメタデータをアタッチできます。タグ名の限定範囲は、お客様のアカウントと、AMI とインスタンスごとの両方があります。例えば、AMI とインスタンスに同じタグ名を使用した場合、競合にはなりません。タグ名では大文字と小文字が区別されません。

タグの詳細についてはLinux インスタンス用 Amazon EC2 ユーザーガイドの「タグの使用」を参照してください。

タグを追加するには:

1. [Add (追加)] ボックスにタグの名前を入力します。緑のプラス記号 (+) ボタンを選択し、[Apply (適用)] を選択します。



AMI または Amazon EC2 インスタンスにタグを追加します

新しいタグはイタリック体で表示されます。そのタグに関連付けられている値がないことを示します。

リストビューでは、新しい列としてタグ名が表示されます。1 つ以上の値がタグに関連付けられているとき、タグは AWS マネジメントコンソール に表示されます。

2. タグの値を追加するには、該当するタグの列でセルをダブルクリックし、値を入力します。タグの値を削除するには、セルをダブルクリックして、値のテキストを削除します。

[Show/Hide (表示/非表示)] ドロップダウンリストでタグをクリアすると、ビューから対応する列が消えます。タグは、AMI、インスタンス、またはボリュームに関連付けられた任意のタグ値とともに保持されます。

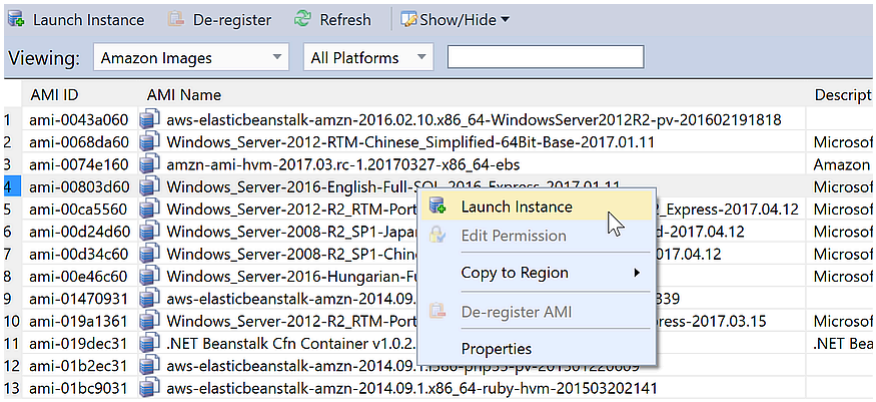
> **Note**
>
> [Show/Hide] (表示/非表示) ドロップダウンリストで、関連付けられた値がないタグをクリアする場合は、AWS Toolkit によりタグが完全に削除されます。このリストビューや [Show/Hide (表示/非表示)] ドロップダウンリストに表示されなくなります。再度そのタグを使用するには、[Show/Hide (表示/非表示)] ダイアログボックスを使用して再作成します。

Amazon EC2 インスタンスを起動する

AWS Explorer では、Amazon EC2 インスタンスを起動するために必要なすべての機能を提供します。このセクションでは、Amazon Machine Image (AMI) を選択し、設定を行い、Amazon EC2 インスタンスとして起動します。

Windows Server の Amazon EC2 インスタンスを起動するには

1. AMI ビューの上部にある、左側のドロップダウンリストから [Amazon Images] を選択します。右側のドロップダウンリストで、[Windows] を選択します。フィルタボックスに、Elastic Block Storage に対する ebs を入力してください。ビューが更新されるまでに数分かかることがあります。
2. リストで AMI を選択し、コンテキスト (右クリック) メニューを開き、[Launch Instance (起動インスタンス)] を選択します。



AMI リスト

3. [Launch New Amazon EC2 Instance (新しい Amazon EC2 インスタンスを起動する)] ダイアログボックスで、アプリケーション用に AMI を設定します。

インスタンスタイプ

起動する EC2 インスタンスのタイプを選択します。インスタンスのタイプと料金については、EC2 料金表に関するページにアクセスしてください。

名前:

インスタンスの名前を入力します。256 文字を超える名前は使用できません。

キーペア

キーペアは Windows パスワードを取得するために使用されます。これは Remote Desktop Protocol (RDP) を使用して EC2 インスタンスにログインするために使用します。プライベー

トキーにアクセスするためのキーペアを選択します。またはキーペアを作成するオプションを選択します。Toolkit でキーペアを作成した場合、Toolkit でプライベートキーを保存できます。

ツールキットに保存されているキーペアは暗号化されます。それらは `%LOCALAPPDATA%\AWSToolkit\keypairs` (通常: `C:\Users\<user>\AppData\Local\AWSToolkit\keypairs`) にあります。暗号化されたキーペアを `.pem` ファイルにエクスポートできます。

a. Visual Studio の [View] (ビュー) を選択し、[AWS Explorer] をクリックします。

b. [Amazon EC2] をクリックし、[Key Pairs (キーペア)] を選択します。

c. キーペアが一覧表示され、Toolkit で作成/管理されているものは、[Stored in AWSToolkit (AWSToolkit で保存)] とマークされます。

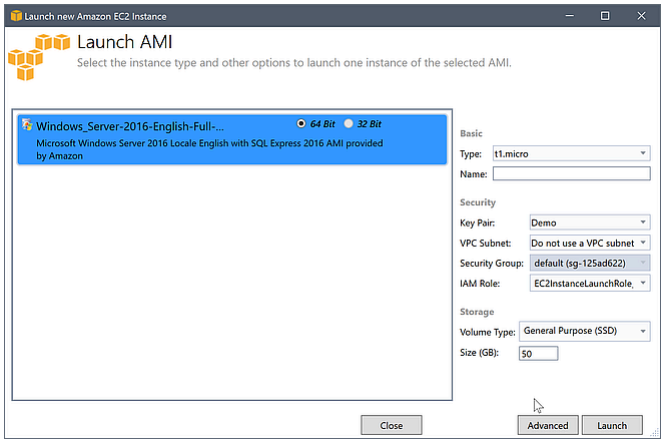
d. 作成したキーペアを右クリックし、[Export Private Key (プライベートキーのエクスポート)] を選択します。プライベートキーが復号され、指定した場所に保存されます。

セキュリティグループ

セキュリティグループを使用して、EC2 インスタンスに許可するネットワークトラフィックのタイプを制御します。セキュリティグループではポート 3389 上の受信トラフィックを有効にして、EC2 インスタンスへの RDP 接続を許可する必要があります。Toolkit を使用してセキュリティグループを作成する方法については、「AWS Explorer からセキュリティグループを管理する」を参照してください。

インスタンスプロファイル

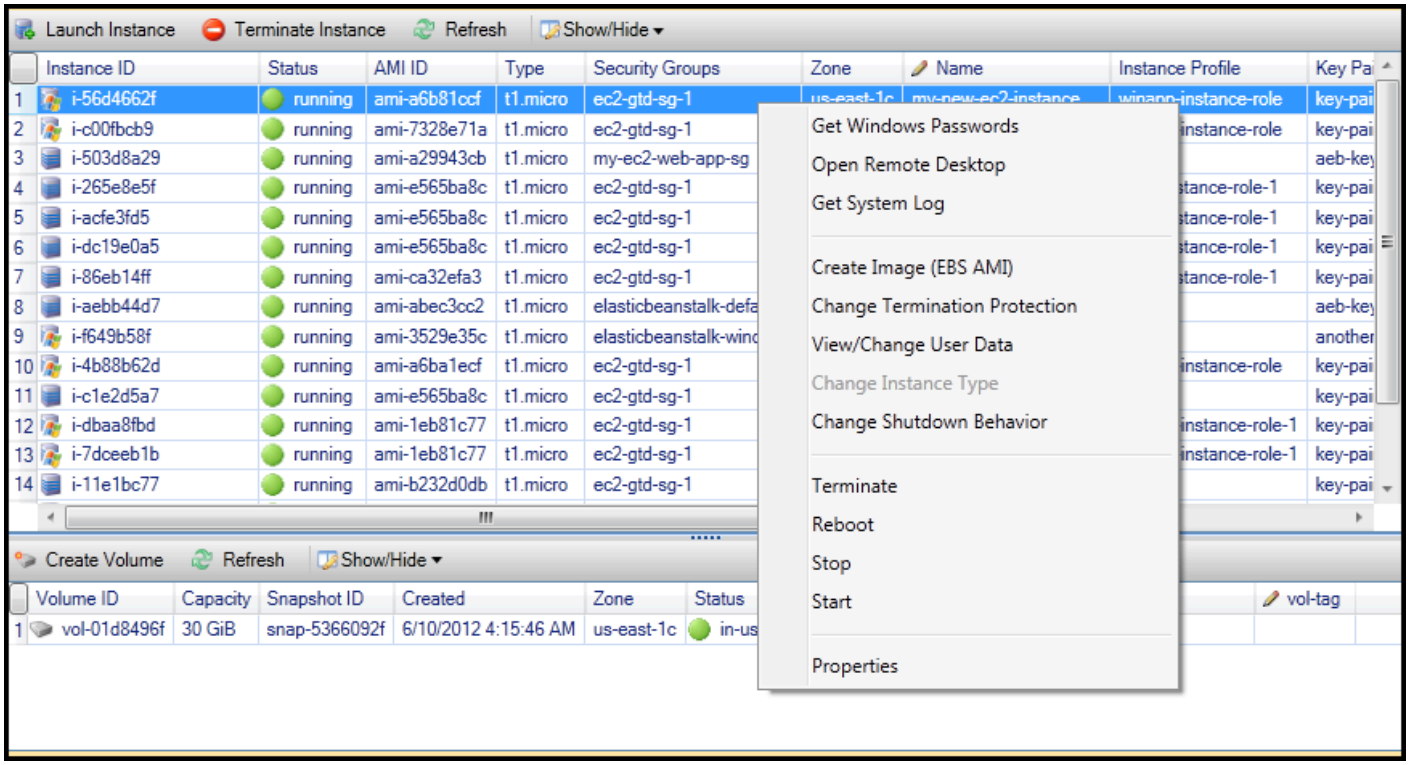
インスタンスプロファイルは IAM ロールの論理コンテナです。インスタンスプロファイルを選択するとき、対応する IAM ロールを EC2 インスタンスに関連付けます。IAM ロールには、Amazon Web Services とアカウントリソースへのアクセスを指定するポリシーを設定します。EC2 インスタンスを IAM ロールに関連付けると、インスタンスで実行されるアプリケーションソフトウェアには、IAM ロールに指定したアクセス許可が付与されます。これにより、アプリケーションソフトウェアはそれ自体の AWS 認証情報を指定しなくても実行可能になり、より安全になります。IAM ロールの詳細については、「IAM ユーザーガイド」を参照してください。



EC2 の [Launch AMI (AMI の起動)] ダイアログボックス

4. [Launch] (起動する) を選択します。

AWS Explorer で Amazon EC2 の [Instances] (インスタンス) サブノードのコンテキスト (右クリック) メニューを開いて、[View] (表示) を選択します。AWS Toolkit では、アクティブなアカウントに関連付けられている Amazon EC2 インスタンスのリストが表示されます。新しいインスタンスを表示するには、[Refresh (更新)] を選択する必要があります。最初にインスタンスが表示されたときは、インスタンスが保留状態であることがありますが、しばらくすると、実行状態に移行します。



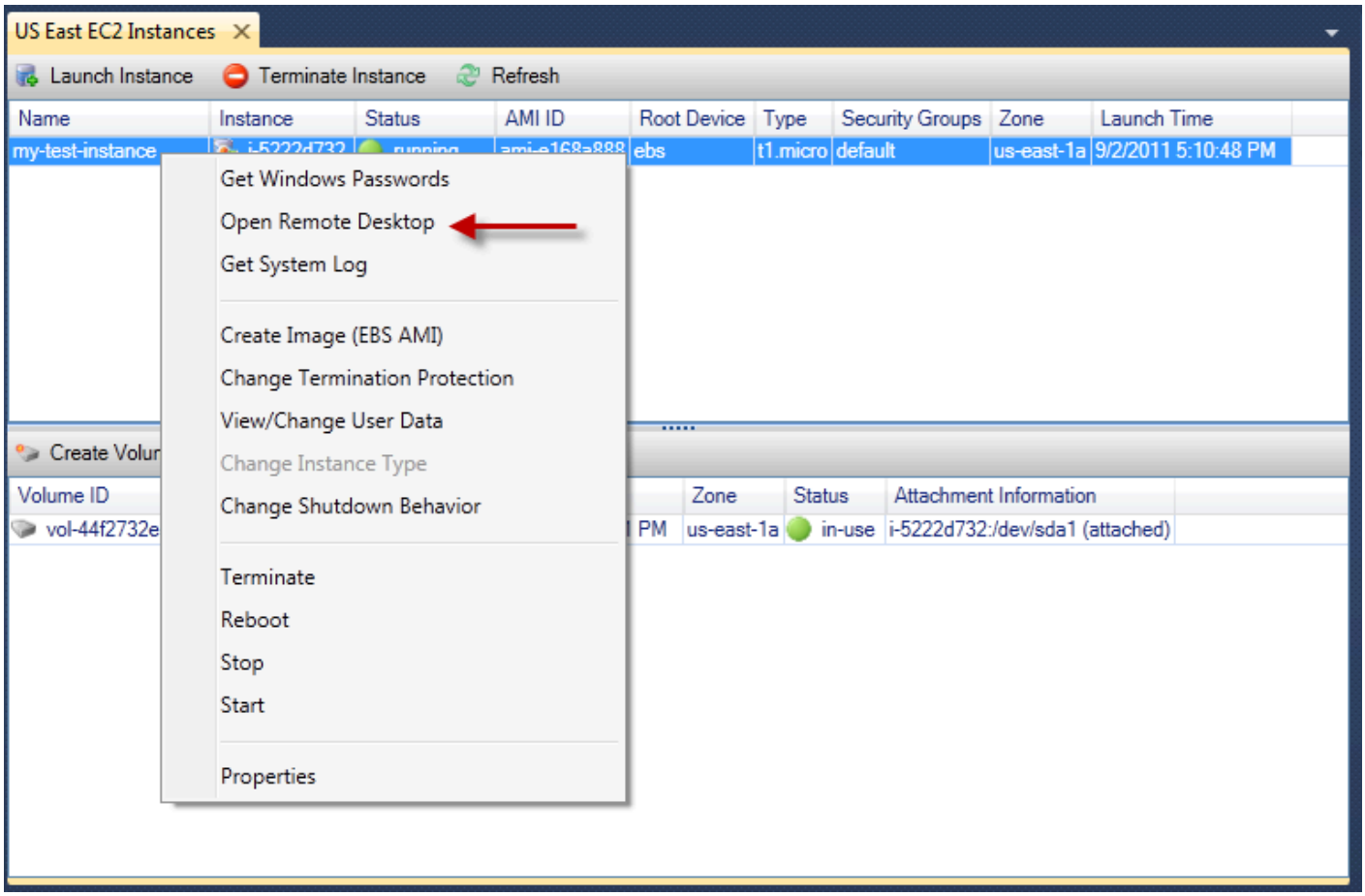
Amazon EC2 インスタンスへの接続

Windows リモートデスクトップを使用して、Windows Server インスタンスに接続します。認証については、AWS Toolkit でインスタンスの管理者パスワードを取得できます。またはインスタンスに関連付けられた格納キーペアを単に使用できます。次の手順では、保存されたキーペアを使用します。

Windows リモートデスクトップを使用して、Windows Server インスタンスに接続するには

1. EC2 インスタンスのリストで、接続する対象の Windows Server インスタンスを右クリックします。コンテキストメニューから [Open Remote Desktop (リモートデスクトップを開く)] を選択します。

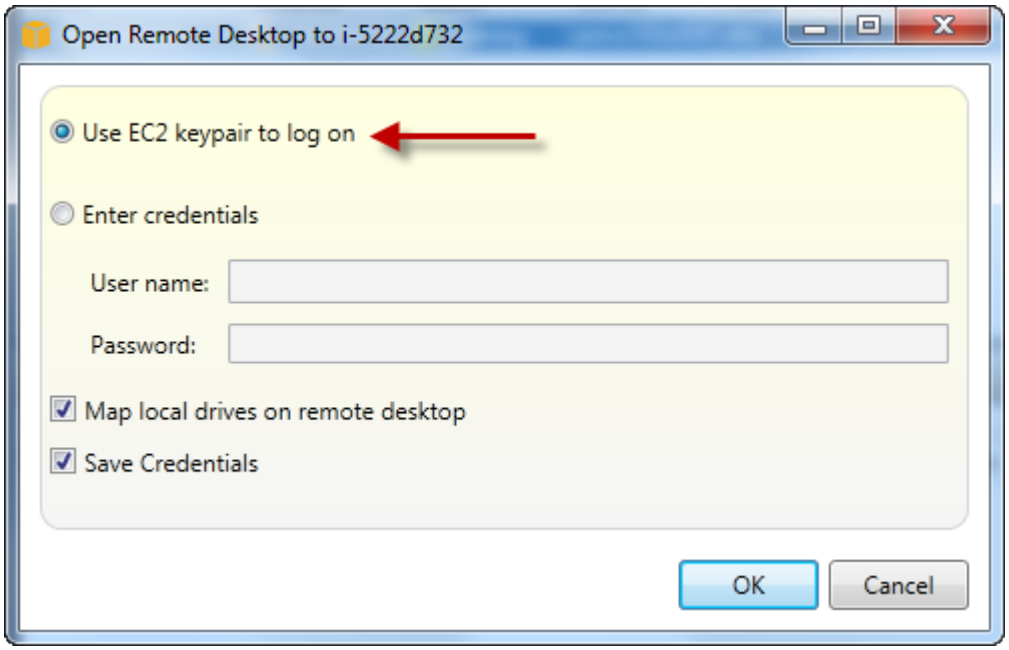
   管理者パスワードを使用して認証する場合は、[Get Windows Passwords (Windows のパスワードを取得)] を選択します。



   EC2 インスタンスのコンテキストメニュー

2. [Open Remoe Desktop (リモートデスクトップを開く)] ダイアログボックスで、[Use EC2 keypair to log on (EC2 キーペアを使用してログオンする)] を選択し、[OK] を選択します。

キーペアが AWS Toolkit に保存されていない場合は、プライベートキーを含む PEM ファイルを指定します。

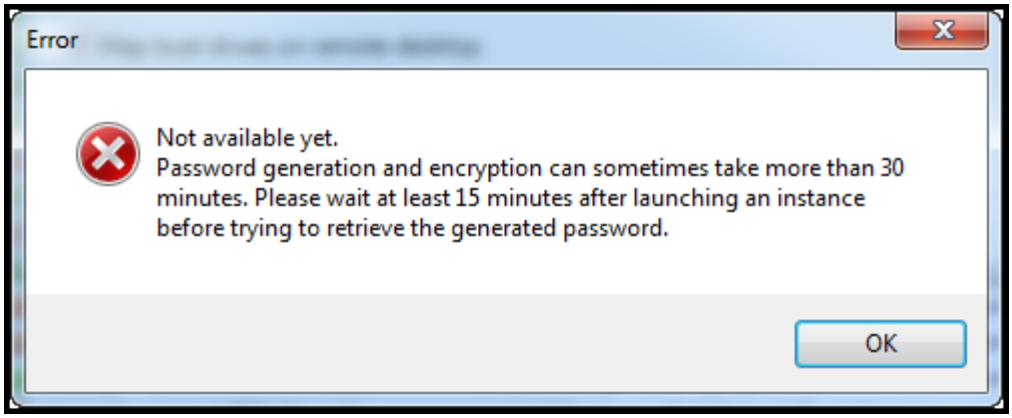


[Open Remote Desktop (リモートデスクトップを開く)] ダイアログボックス

3. リモートデスクトップウィンドウが開きます。キーペアで認証されているため、サインインする必要はありません。Amazon EC2 インスタンスで管理者として実行しています。

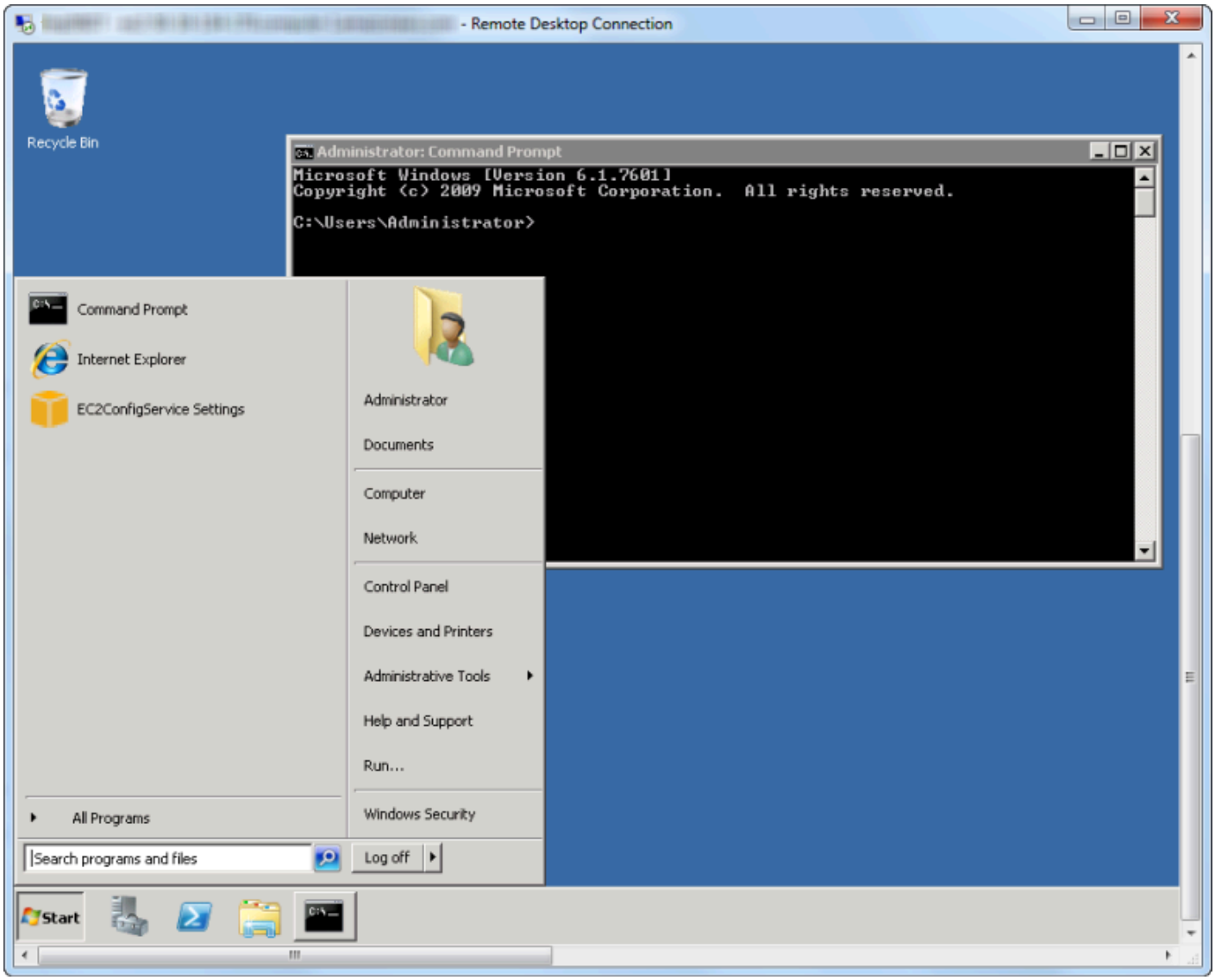
   EC2 インスタンスが最近開始された場合のみ、接続できない可能性があります。次の 2 つの理由が考えられます。

   - リモートデスクトップサービスがまだ稼働していません。数分後にもう一度お試しください。
   - パスワード情報がまだインスタンスに転送されていません。この場合、次のようなメッセージボックスが表示されます。



パスワードはまだ使用できません

次の画面例では、ユーザーがリモートデスクトップを介して管理者として接続されています。



リモートデスクトップ

Amazon EC2 インスタンスを削除する

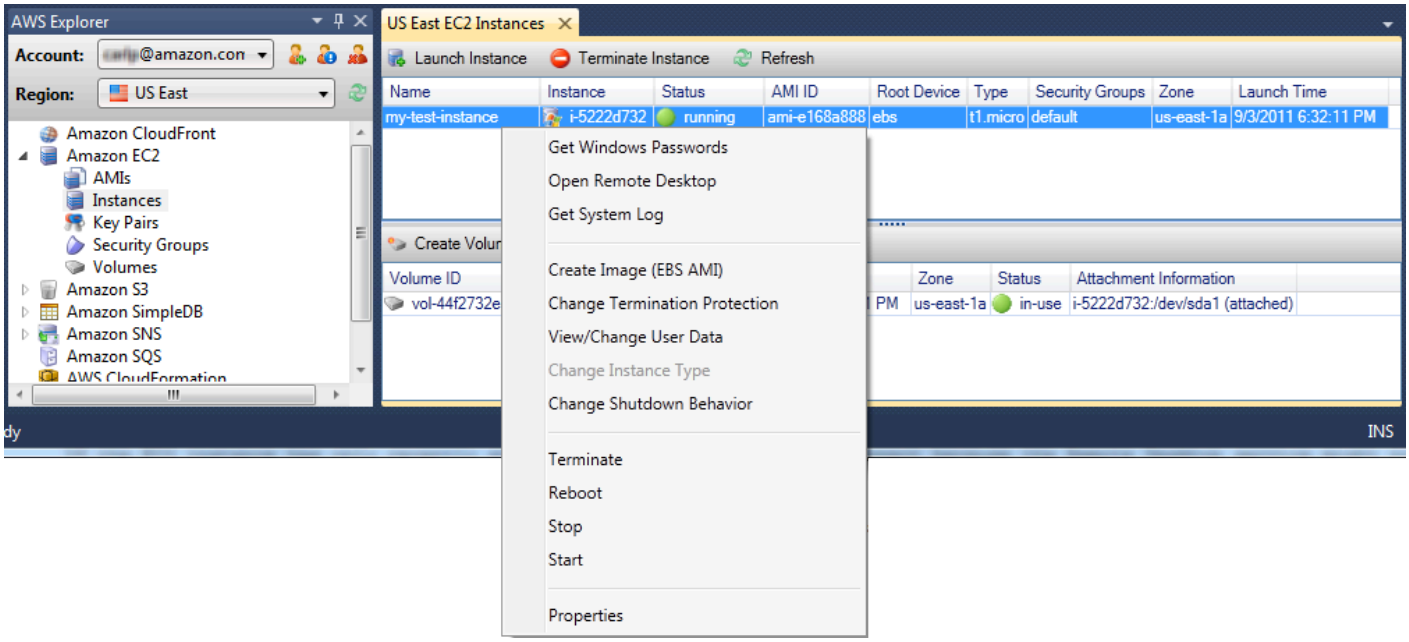
AWS Toolkit を使用して、Visual Studio から実行中の Amazon EC2 インスタンスを停止または終了できます。インスタンスを停止するには、EC2 インスタンスが Amazon EBS ボリュームを使用している必要があります。EC2 インスタンスが Amazon EBS ボリュームを使用していない場合、インスタンスを削除するオプションのみが表示されます。

インスタンスを停止した場合でも、EBS ボリュームに保存されているデータは保持されます。インスタンスを終了した場合は、インスタンスのローカルストレージデバイスに保存されているデータはすべて失われます。いずれの場合も、停止または終了すると、EC2 インスタンスへの課金は継続されません。ただし、インスタンスを停止した場合は、インスタンスが停止した後も保持される EBS ストレージに対しては、引き続きお客様への課金が発生することになります。

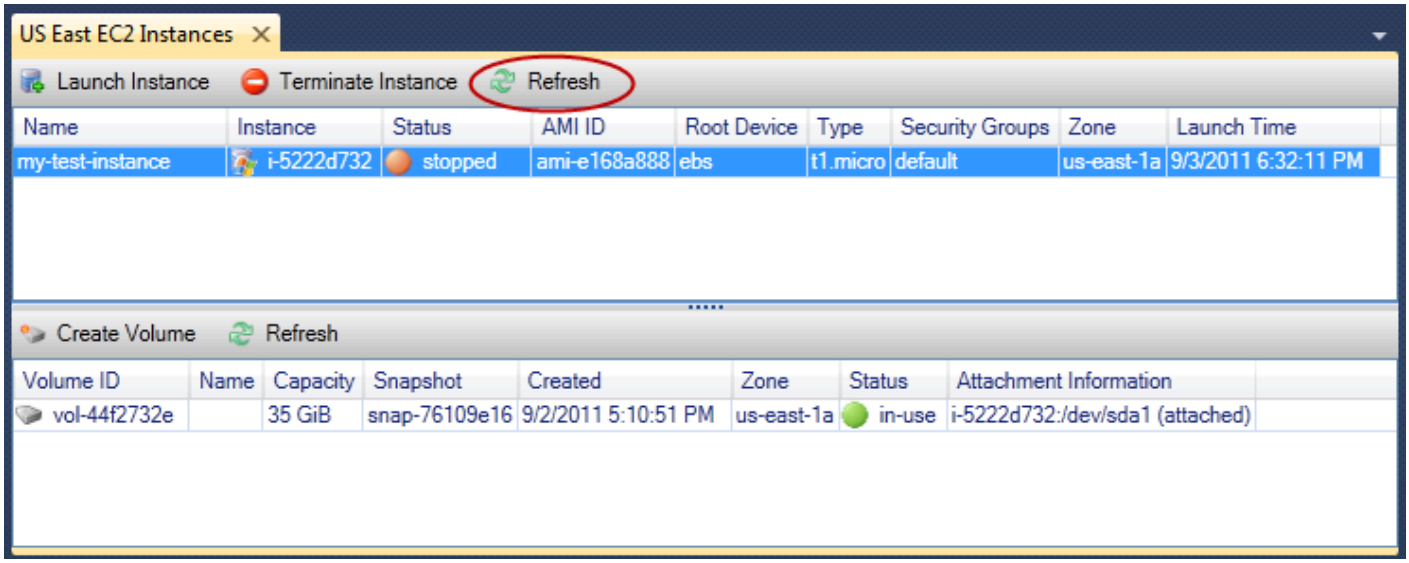
インスタンスを削除するもう 1 つの可能な方法は、インスタンスに接続しているリモートデスクトップを使用することです。Windows の [スタート] メニューから [シャットダウン] を使用します。このシナリオではインスタンスの停止または終了のいずれにも設定できます。

Amazon EC2 インスタンスを停止するには

1. AWS Explorer で Amazon EC2 ノードを展開し、[Instances] (インスタンス) のコンテキスト (右クリック) メニューを開いて、[View] (表示) を選択します。[Instances (インスタンス)] リストで、停止するインスタンスを右クリックして、コンテキストメニューから [Stop (停止)] を選択します。[Yes (はい)] をクリックしてインスタンスの停止を確定します。

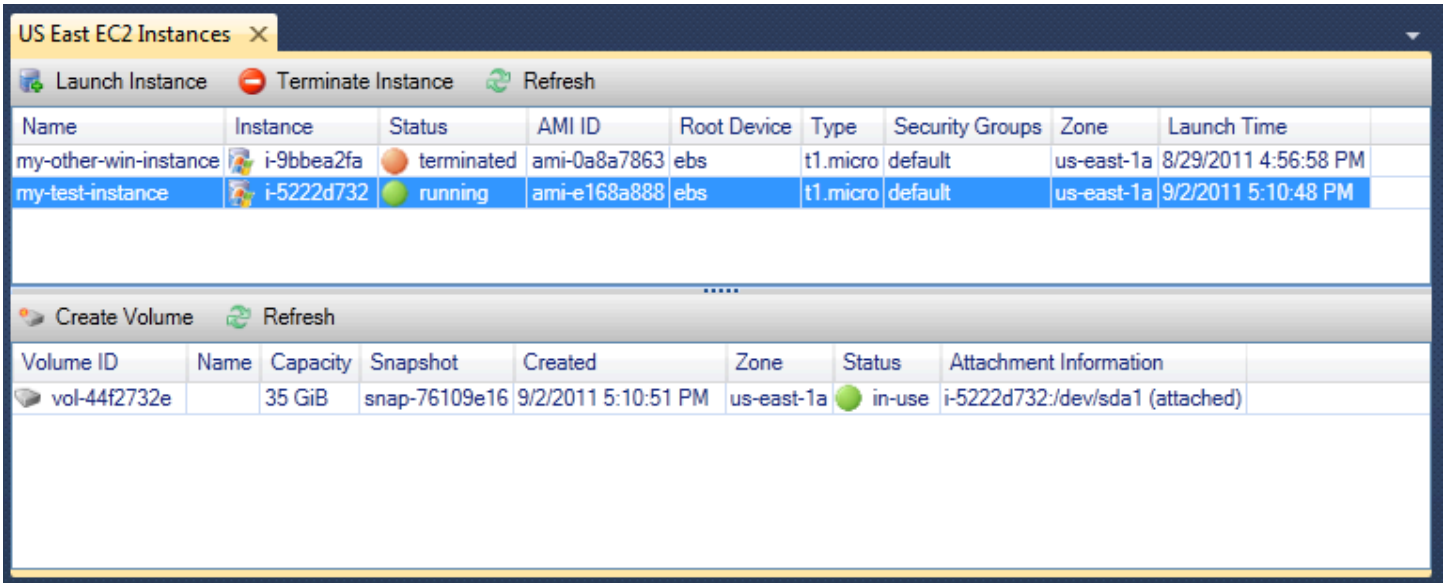


2. [Instances (インスタンス)] リストの上部で、Amazon EC2 インスタンスのステータスの変更を確認するには、[Refresh (更新)] を選択します。インスタンスを終了ではなく停止したため、インスタンスと関連付けられた EBS ボリュームはアクティブのままです。



終了したインスタンスを引き続き表示

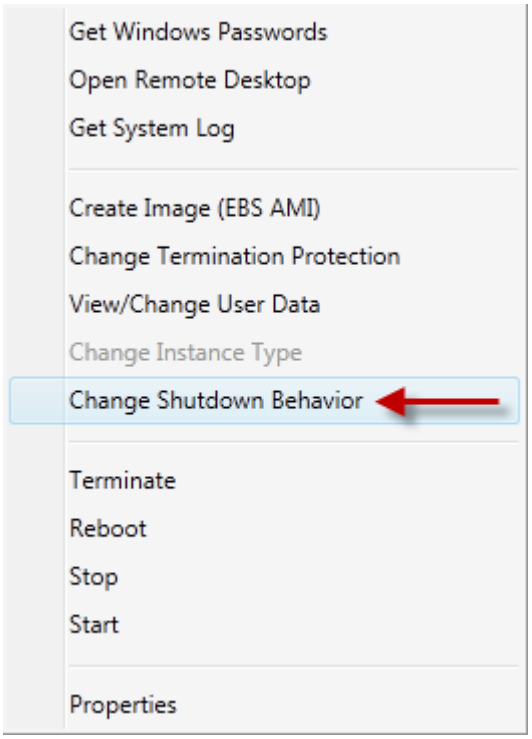
インスタンスを終了した場合、実行中や停止したインスタンスとともに [Instance (インスタンス)] リストに引き続き表示されます。最終的に、AWS がこれらのインスタンスを再利用すると、リストから消去されます。終了状態のインスタンスに対して課金されることはありません。



シャットダウン時の EC2 インスタンスの動作を指定するには

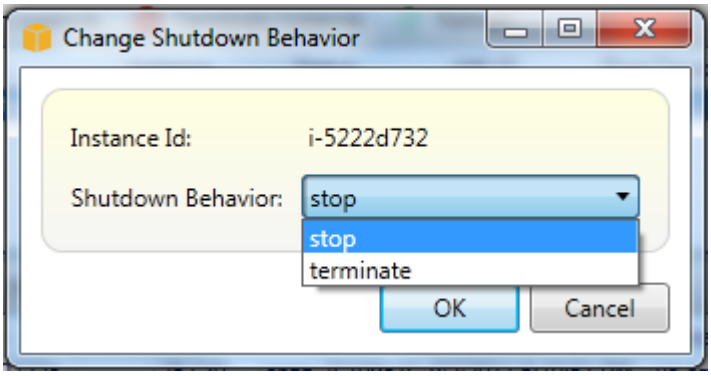
[スタート] メニューで [シャットダウン] が選択された場合、AWS Toolkit により、Amazon EC2 インスタンスを停止するか終了するかどうかを指定できます。

1. [Instances (インスタンス)] リストで、Amazon EC2 インスタンスを右クリックし、[Change shutdown behavior (シャットダウン動作の変更)] を選択します。



[Change Shutdown Behavior (シャットダウン動作の変更)] メニューアイテム

2. [Change Shutdown Behavior (シャットダウン動作の変更)] ダイアログボックスで、[Shutdown Behavior (シャットダウン動作)] ドロップダウンリストから [Stop (停止)] または [Terminate (終了)] を選択します。



Amazon ECS インスタンスの管理

AWS Explorer は、Amazon Elastic Container Service (Amazon ECS) クラスターとコンテナリポジトリの詳細ビューを提供します。Visual Studio 開発環境内からクラスターとコンテナの詳細を作成、削除、管理できます。

サービスのプロパティの変更

サービスの詳細、イベント、プロパティはクラスタービューで表示できます。

1. AWS Explorer で、管理するクラスターのコンテキスト (右クリック) メニューを開き、表示を選択します。
2. [ECS Cluster (ECS クラスター)] ビューで、左側の [Services (サービス)] をクリックしてから、詳細ビューの [Details (詳細)] タブをクリックします。[Events (イベント)] をクリックしてイベントメッセージを表示できます。また、[Deployments (デプロイ)] をクリックしてデプロイステータスを表示できます。
3. [Edit (編集)] をクリックします。目的のタスク数と、最小および最大ヘルス率を変更できます。
4. 変更を受け入れるには、[Save (保存)] をクリックします。既存の値に戻すには、[Cancel (キャンセル)] をクリックします。

タスクの停止

クラスタービューで、タスクの現在のステータスを表示し、1 つ以上のタスクを停止できます。

タスクを停止するには

1. AWS Explorer で、停止するタスクを含むクラスターのコンテキスト (右クリック) メニューを開き、表示を選択します。
2. [ECS Cluster (ECS クラスター)] ビューで、左側の [Tasks (タスク)] をクリックします。
3. [Desired Task Status (必要なタスクのステータス)] が `Running` に設定されていることを確認します。停止する個々のタスクを選択し、[Stop (停止)] をクリックします。または、[Stop All (すべてを停止)] をクリックし、実行中のすべてのタスクを選択して停止します。
4. [Stop Tasks (タスクの停止)] ダイアログボックスで、[Yes (はい)] を選択します。

サービスの削除

クラスタービューで、クラスターからサービスを削除できます。

クラスターサービスを削除するには

1. AWS Explorer で、削除するサービスがあるクラスターのコンテキスト (右クリック) メニューを開き、表示を選択します。
2. [ECS Cluster (ECS クラスター)] ビューで、左側の [Services (サービス)] をクリックしてから、[Delete (削除)] をクリックします。
3. [Delete Cluster (クラスターの削除)] ダイアログボックスで、クラスターにロードバランサーとターゲットグループがある場合にクラスターからそれらを削除するかどうかを選択できます。それらのバランサーとグループはサービスが削除されると使用されなくなります。
4. [Delete Cluster (クラスターの削除)] ダイアログボックスで、[OK] を選択します。クラスターは削除されると、 AWS Explorer から削除されます。

クラスターの削除

AWS Explorer から Amazon Elastic Container Service クラスターを削除できます。

クラスターを削除するには

1. AWS Explorer で、Amazon ECS のクラスターノードで削除するクラスターのコンテキスト (右クリック) メニューを開き、削除を選択します。
2. [Delete Cluster (クラスターの削除)] ダイアログボックスで、[OK] を選択します。クラスターは削除されると、 AWS Explorer から削除されます。

リポジトリの作成

AWS Explorer から Amazon Elastic Container Registry リポジトリを作成できます。

リポジトリを作成するには

1. AWS Explorer で、Amazon ECS のリポジトリノードのコンテキスト (右クリック) メニューを開き、リポジトリの作成を選択します。
2. [Create Repository (リポジトリの作成)] ダイアログボックスでリポジトリ名を指定し、[OK] を選択します。

リポジトリの削除

AWS Explorer から Amazon Elastic Container Registry リポジトリを削除できます。

リポジトリを削除するには

1. AWS Explorer で、Amazon ECS のリポジトリノードのコンテキスト (右クリック) メニューを開き、リポジトリの削除を選択します。
2. [Delete Repository (リポジトリの削除)] ダイアログボックスで、リポジトリを削除できます。この場所では、リポジトリはイメージを含んでいても削除されます。それ以外の場所では、リポジトリは空の場合にのみ削除されます。[Yes (はい)] をクリックします。

AWS Explorer からセキュリティグループを管理する

Toolkit for Visual Studio を使用すると、Amazon Elastic Compute Cloud (Amazon EC2) インスタンスと CloudFormation で使用するセキュリティグループを作成および設定できます。Amazon EC2 インスタンスを起動する際、またはアプリケーションを CloudFormation にデプロイする際には、Amazon EC2 インスタンスに関連付けるセキュリティグループを指定します。(CloudFormation のデプロイにより Amazon EC2 インスタンスが作成されます。)

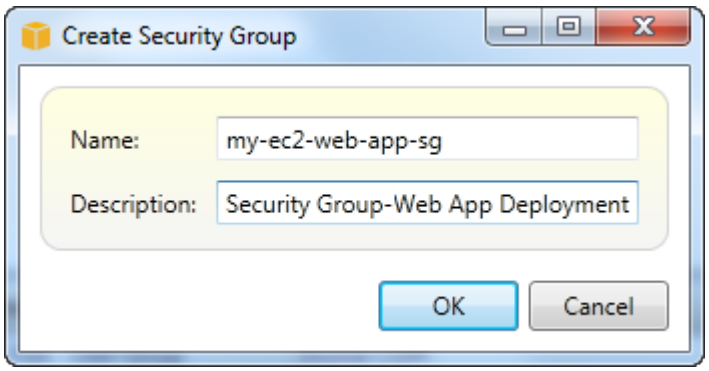
セキュリティグループは受信ネットワークトラフィックに対するファイアウォールのような役割を果たします。セキュリティグループでは、Amazon EC2 インスタンスに許可する受信ネットワークトラフィックのタイプを指定します。また、特定の IP アドレスまたは指定したユーザーまたは他のセキュリティグループからのみの受信トラフィックを許可するように指定することもできます。

セキュリティグループを作成する

このセクションでは、セキュリティグループを作成します。セキュリティグループが作成されたときは、セキュリティグループでいずれのアクセス許可も設定されていません。アクセス許可の設定はこの後のオペレーションで扱います。

セキュリティグループを作成するには

1. AWS Explorer で、Amazon EC2 ノードの下で、[Security Groups] (セキュリティグループ) ノードのコンテキスト (右クリック) メニューを開いて、[View] (表示) を選択します。
2. [EC2 Security Groups (EC2 セキュリティグループ)] タブで、[Create Security Group (セキュリティグループの作成)] をクリックします。
3. [Create Security Group (セキュリティグループの作成)] ダイアログボックスで、セキュリティグループ名と説明を入力し、[OK] を選択します。

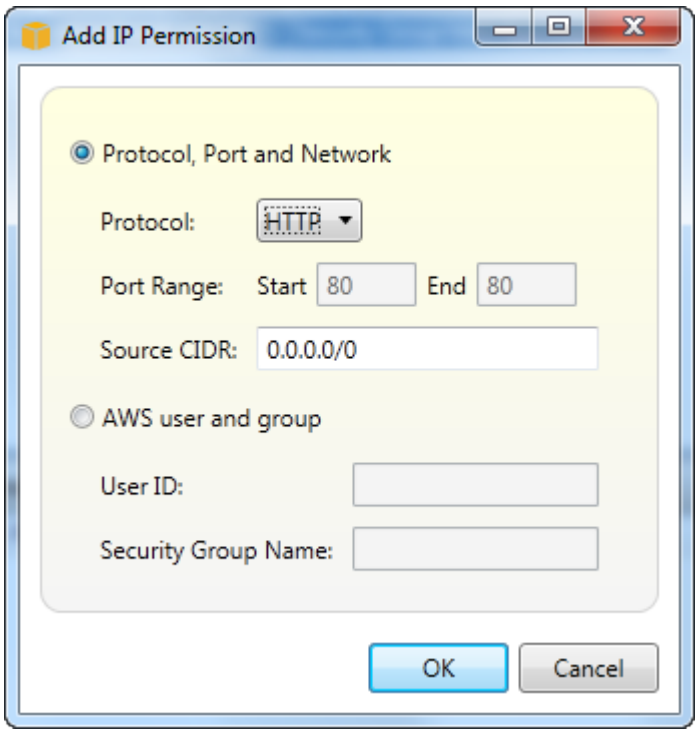


セキュリティグループにアクセス許可を追加する

このセクションでは、セキュリティグループにアクセス許可を追加して、HTTP および HTTPS プロトコルを使用したウェブトラフィックを許可します。また、他のコンピュータが Windows リモートデスクトッププロトコル (RDP) を使用して接続することを許可します。

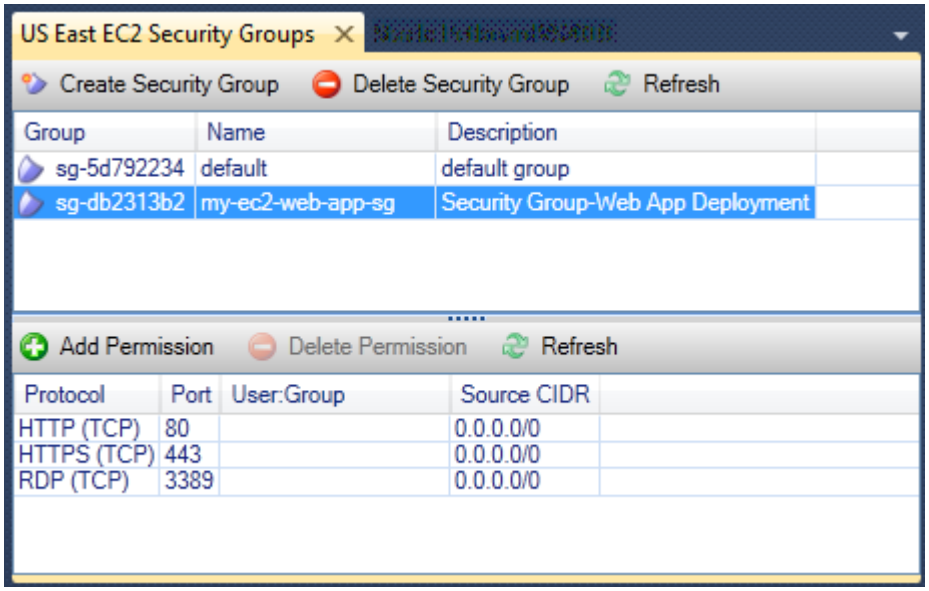
セキュリティグループにアクセス許可を追加するには

1. [EC2 Security Groups (EC2 セキュリティグループ)] タブで、セキュリティグループを選択し、[Add Permission (アクセス許可の追加)] ボタンを選択します。
2. [Add IP Permission (アクセス許可の追加)] ダイアログボックスで、[Protocol, Port and Network (プロトコル、ポート、ネットワーク)] ラジオボタンを選択し、[Protocol (プロトコル)] ドロップダウンリストで、[HTTP] を選択します。ポート範囲は HTTP のデフォルトポートであるポート 80 に自動的に調整されます。[Source CIDR (ソース CIDR)] フィールドのデフォルトは 0.0.0.0/0 となります。これは、任意の外部 IP アドレスからの HTTP ネットワークトラフィックが許可されることを指定します。[OK] を選択してください。



このセキュリティグループ用にポート 80 (HTTP) を開く

3. このプロセスを HTTPS および RDP に繰り返します。セキュリティグループのアクセス許可は以下のようになります。



また、ユーザー ID とセキュリティグループ名を指定することで、セキュリティグループにアクセス許可を設定することもできます。この場合、このセキュリティグループの Amazon EC2 インスタンスは、指定したセキュリティグループの Amazon EC2 インスタンスからのすべての受信ネットワー

クトラフィックを許可します。また、セキュリティグループ名を特定するためにユーザー ID も指定しなければなりません。セキュリティグループ名はすべての AWS 間で一意であるとは限らないためです。セキュリティグループの詳細については、EC2 のドキュメントを参照してください。

Amazon EC2 インスタンスからの AMI の作成

AWS Toolkit for Visual Studioを使用して Amazon マシンイメージ (AMI) を作成できます。AMI の詳細については、「Windows インスタンス用 Amazon Elastic Compute Cloud ユーザーガイド」の「Amazon マシンイメージ (AMI)」のトピックを参照してください。

既存の Amazon EC2 インスタンスから AMI を作成するには、次の手順を実行します。

既存の Amazon EC2 インスタンスから AMI を作成する

1. AWS Toolkit Explorer から Amazon Amazon EC2 を展開し、インスタンスを選択して既存のインスタンスのリストを表示します。
2. AMI のベースとして使用するインスタンスを右クリックし、[イメージの作成 (ABS AMI)] を選択して [イメージの作成] ダイアログウィンドウを開きます。
3. [イメージの作成] ダイアログウィンドウで、指定されたフィールドにイメージの名前と説明を追加し、[OK] ボタンを選択して続行します。
4. イメージが作成されると、Visual Studio で [イメージが作成されました] 確認ウィンドウが開きます。[OK] ボタンを選択して続行します。

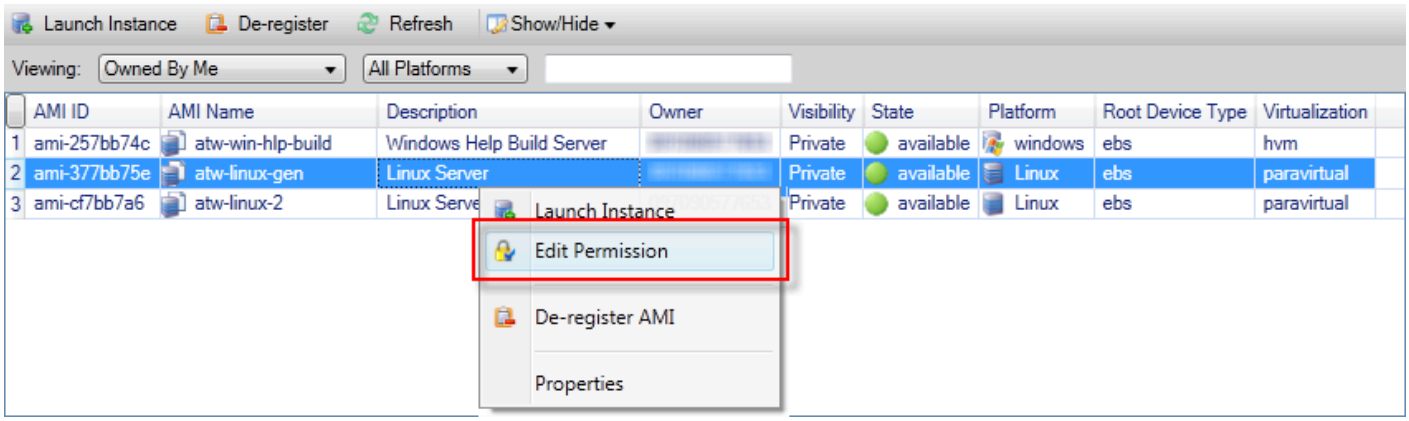
Toolkit で新しい AMI を表示するには AWS 、Amazon EC2 を展開し、AMIs をダブルクリックして、既存の AMIs のリストを表示する Visual Studio Editor ペインのウィンドウを開きます。リストに新しい AMI が表示されない場合は、AMI ウィンドウの上部にある[更新]ボタンを選択します。

Amazon マシンイメージに起動許可を設定する

AWS Explorer AMIs ビューから Amazon マシンイメージ (AMIs) に起動許可を設定できます。[Set AMI Permissions (AMI アクセス許可の設定)] ダイアログボックスを使用して、AMI から許可をコピーします。

AMI にアクセス許可を設定するには

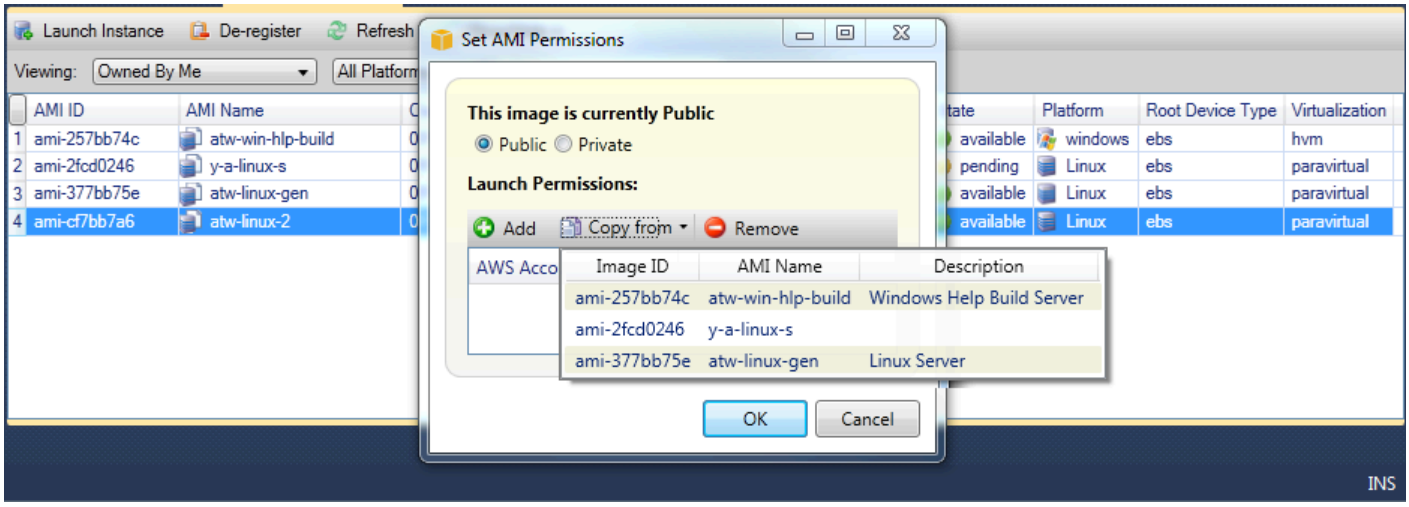
1. AWS Explorer AMIs ビューで、AMI のコンテキスト (右クリック) メニューを開き、アクセス許可の編集を選択します。



2. [Set AMI Permissions (AMI アクセス許可の設定)] ダイアログボックスに、選択できる 3 つのオプションがあります。

   - 起動許可を付与するには、追加を選択し、起動許可を付与する AWS ユーザーのアカウント番号を入力します。
   - 起動許可を削除するには、起動許可を削除する AWS ユーザーのアカウント番号を選択し、削除を選択します。
   - 1 つの AMI から別の AMI に許可をコピーするには、リストから AMI を選択し、[Copy from (コピー)] を選択します。AMI の起動許可を持っているユーザーを選択すると、現在の AMI の起動許可が与えられます。このプロセスを [Copy-from (コピー)] リストの他の AMI で繰り返すことができ、これにより複数の AMI からターゲットの AMI へ許可をコピーできます。

     [Copy-from] (コピー元) リストには、 AWS Explorer で [AMI] ビューが表示されたときにアクティブであったアカウントが所有する AMI のみが含まれます。その結果、アクティブアカウントによって所有されている他の AMI がない場合には、[Copy-from (コピー)] リストに AMI が表示されない場合があります。



[Copy AMI permissions (AMI アクセス許可のコピー)] ダイアログボックス

Amazon Virtual Private Cloud (VPC)

Amazon Virtual Private Cloud (Amazon VPC) を使用すると、定義した仮想ネットワーク内で Amazon Web Services リソースを起動できます。この仮想ネットワークは、お客様自身のデータセンターで運用されていた従来のネットワークに似ていますが、 AWSのスケーラブルなインフラストラクチャを使用できるというメリットがあります。詳細については、Amazon VPC ユーザーガイドをご覧ください。

Toolkit for Visual Studio により、デベロッパーは VPC 機能にアクセスできます。この機能は、AWS マネジメントコンソール で公開されているものと似ていますが、Visual Studio の開発環境から実行します。 AWS Explorer の Amazon VPC ノードには、次の領域のサブノードが含まれます。

- VPC
- サブネット
- Elastic IP
- インターネットゲートウェイ
- ネットワーク ACL
- ルートテーブル
- セキュリティグループ

を使用したデプロイ用のパブリック/プライベート VPC の作成 AWS Elastic Beanstalk

このセクションでは、パブリックサブネットとプライベートサブネットの両方が含まれる Amazon VPC を作成する方法を説明します。パブリックサブネットは、ネットワークアドレス変換 (NAT) を実行する Amazon EC2 インスタンスを使用し、プライベートサブネットのインスタンスからパブリックのインターネットと通信できるようにします。2 つのサブネットは同じアベイラビリティーゾーン (AZ) に存在している必要があります。

これは、 AWS Elastic Beanstalk 環境を VPC にデプロイするために必要な最小限の VPC 設定です。このシナリオでは、アプリケーションをホストする Amazon EC2 インスタンスがプライベートサブネット内に存在し、受信トラフィックをアプリケーションにルーティングする Elastic Load Balancing ロードバランサーはパブリックサブネット内にあります。

ネットワークアドレス変換 (NAT) の詳細については、Amazon Virtual Private Cloud ユーザーガイドの「NAT インスタンス」を参照してください。VPC を使用するためにデプロイを構成する方法の例については、「Elastic Beanstalk へのデプロイ」を参照してください。

パブリック/プライベートサブネット VPC を作成するには

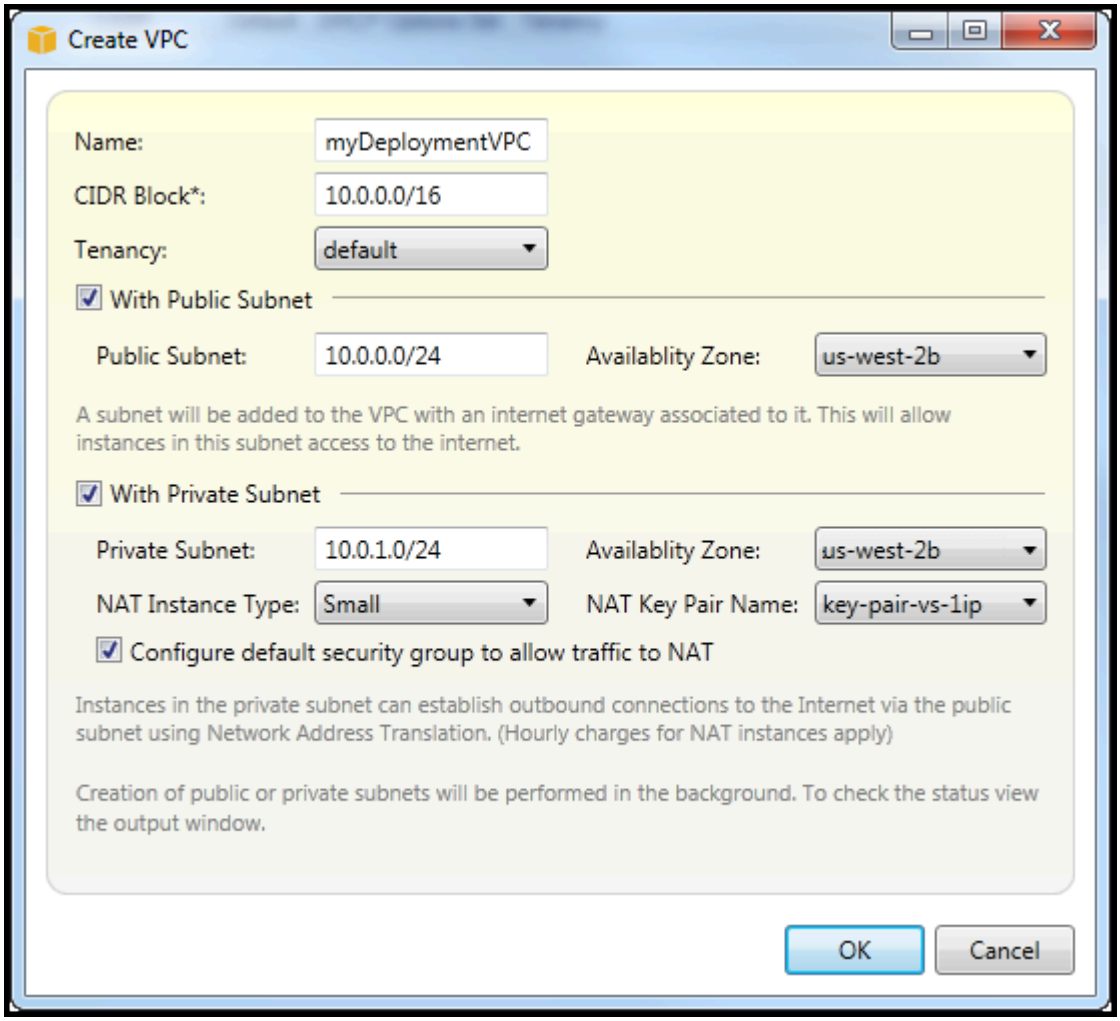
1. AWS Explorer の Amazon VPC ノードで、VPCsサブノードを開き、VPC の作成を選択します。

2. VPC を次のように設定します。
   - 使用する VPC の名前を入力します。
   - [With Public Subnet (パブリックサブネットを使用)] と [With Private Subnet (プライベートサブネットの使用)] チェックボックスをオンにします。
   - 各サブネットの [Availability Zone (アベイラビリティーゾーン)] ドロップダウンリストから、アベイラビリティーゾーンを選択します。両方のサブネットに必ず同じ AZ を使用してください。
   - プライベートサブネットでは、[NAT Key Pair Name (NAT キーペア名)] にキーペアを入力します。このキーペアは Amazon EC2 インスタンス用で、このインスタンスがプライベートサブネットからパブリックインターネットへのネットワークアドレス変換を実行します。
   - [Configure default security group to allow traffic to NAT (NAT へのトラフィックを許可するようにデフォルトセキュリティグループを設定する)] チェックボックスをオンにします。

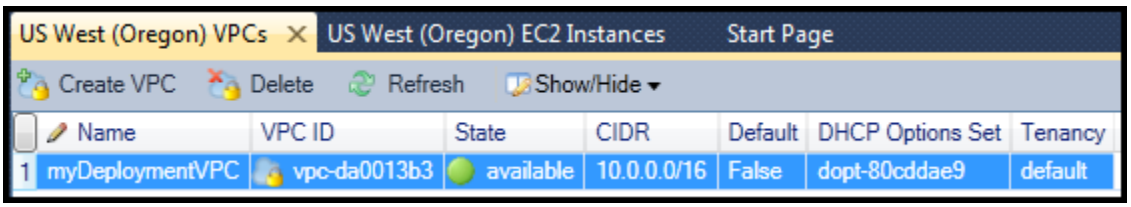
   使用する VPC の名前を入力します。[With Public Subnet (パブリックサブネットを使用)] と [With Private Subnet (プライベートサブネットの使用)] チェックボックスをオンにします。各サブネットの [Availability Zone (アベイラビリティーゾーン)] ドロップダウンリストから、アベイラビリティーゾーンを選択します。両方のサブネットに必ず同じ AZ を使用してください。プライベートサブネットでは、[NAT Key Pair Name (NAT キーペア名)] にキーペアを入力します。このキー

ペアは Amazon EC2 インスタンス用で、このインスタンスがプライベートサブネットからパブリックインターネットへのネットワークアドレス変換を実行します。[Configure default security group to allow traffic to NAT (NAT へのトラフィックを許可するようにデフォルトセキュリティグループを設定する)] チェックボックスをオンにします。

[OK] を選択してください。

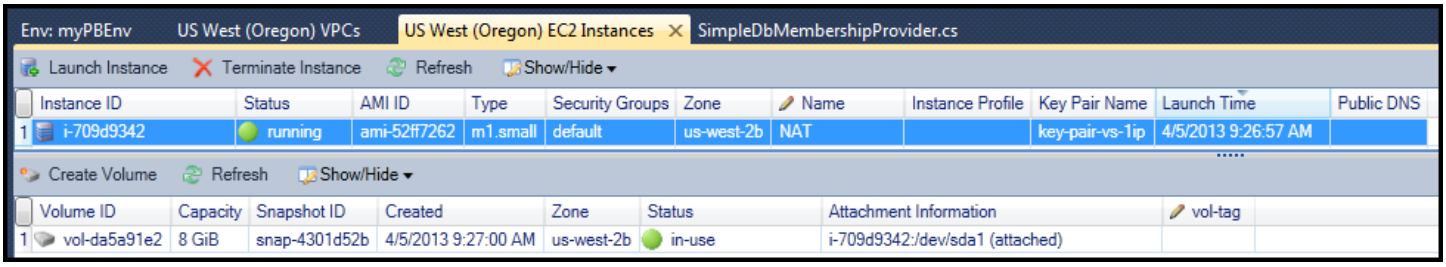


Explorer の VPC タブで新しい VPCs AWS を表示できます。



NAT インスタンスが起動するまでに数分かかることがあります。使用可能な場合は、 AWS Explorer で Amazon EC2 ノードを展開し、Instances サブノードを開くことで表示できます。

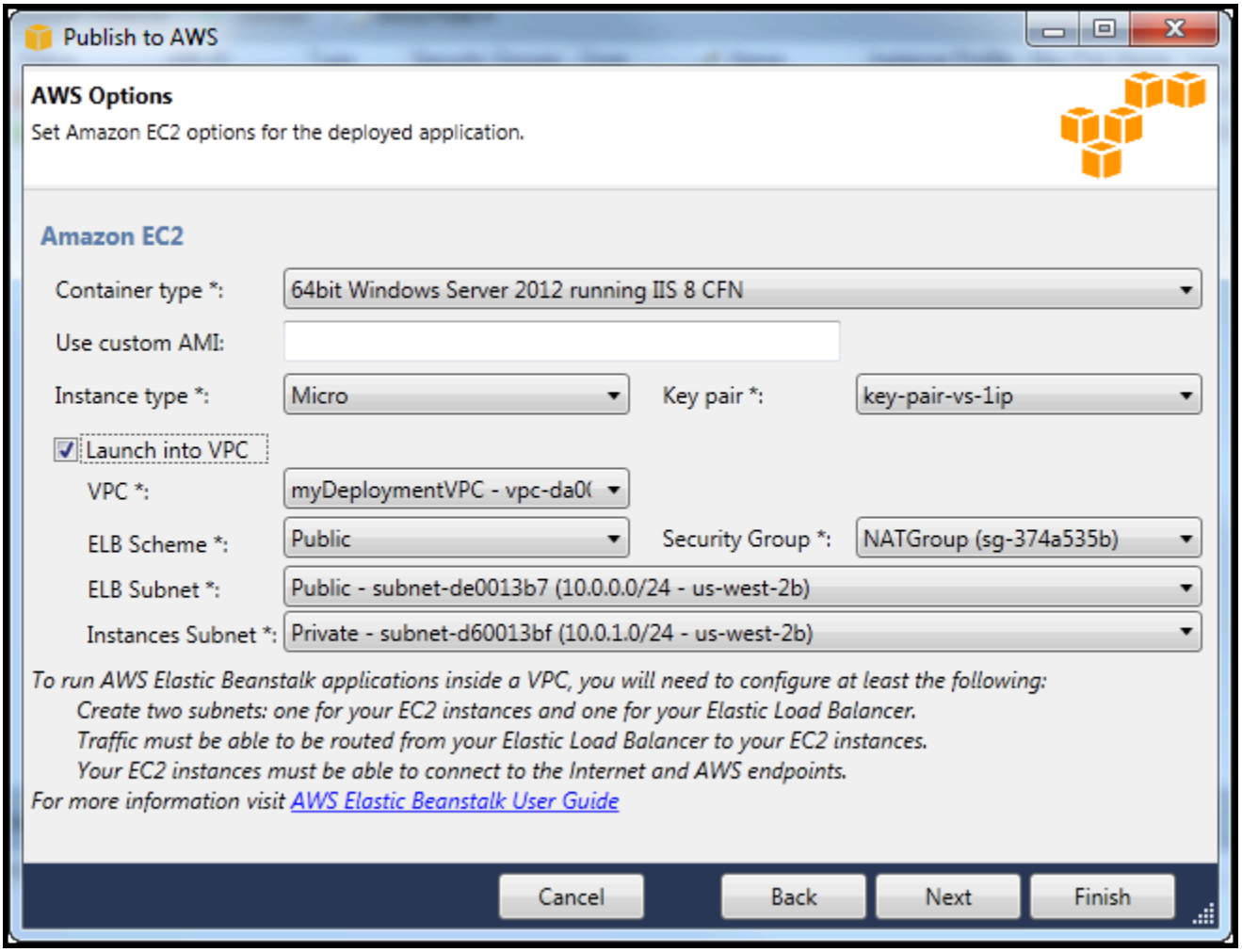
Amazon Elastic Block Store (Amazon EBS) ボリュームは、NAT インスタンスに対して自動的に作成されます。Amazon EBS の詳細については、「Amazon EC2 Linux インスタンス用 ユーザーガイド」の「Amazon Elastic Block Store (Amazon EBS)」のトピックを参照してください。



アプリケーションを AWS Elastic Beanstalk 環境にデプロイし、VPC で環境を起動することを選択した場合、Toolkit は VPC の設定情報を Publish to Amazon Web Services ダイアログボックスに入力します。

Toolkit によりダイアログボックスに入力される情報は、Toolkit で作成された VPC からの情報だけで、 AWS マネジメントコンソールを使用して作成された VPC のものは含まれません。これは、Toolkit が VPC を作成したとき、情報にアクセスできるように VPC のコンポーネントにタグ付けしているからです。

次のデプロイウィザードのスクリーンショットは、Toolkit で作成した VPC の値が入ったダイアログボックスの例を示します。



VPC を削除するには

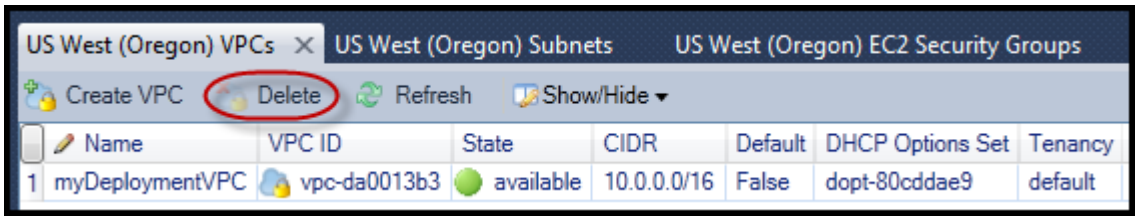
VPC を削除するには、まず VPC 内のすべての Amazon EC2 インスタンスを終了する必要があります。

1. VPC 内の AWS Elastic Beanstalk 環境にアプリケーションをデプロイした場合は、環境を削除します。これにより、Elastic Load Balancing ロードバランサーとともにユーザーのアプリケーションをホストするすべての Amazon EC2 インスタンスが終了します。

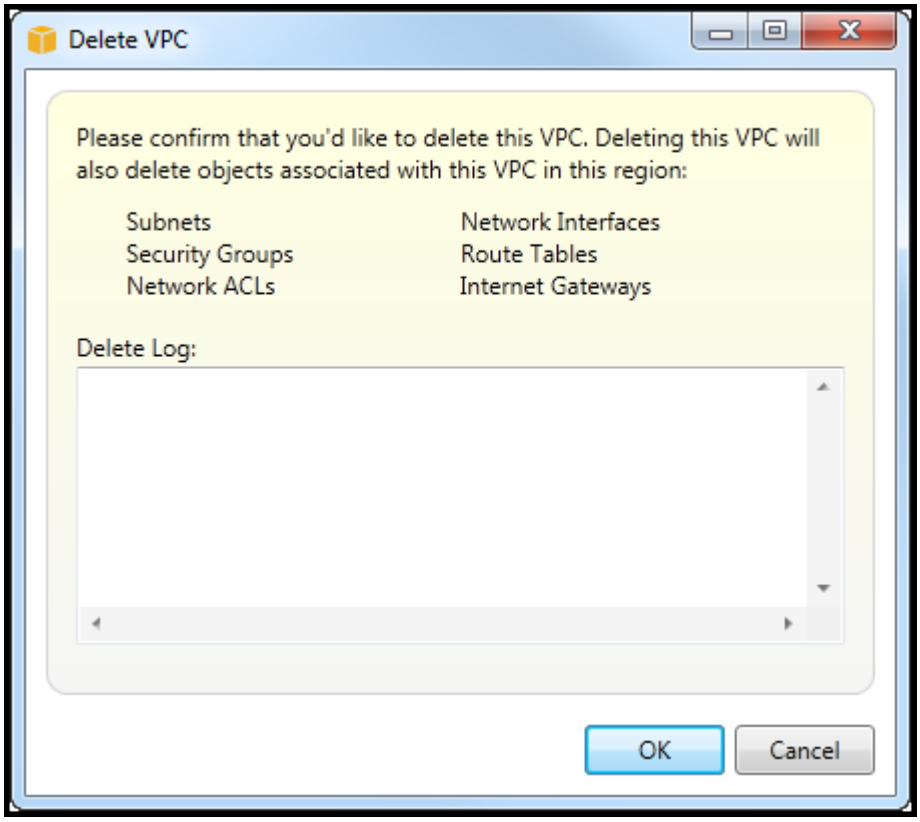
   環境を削除しないで、アプリケーションのホスティングしているインスタンスを直接終了しようとすると、Auto Scaling サービスは、削除されたインスタンスを置き換える新しいインスタンスを自動的に作成します。詳細については、Auto Scaling デベロッパーガイドを参照してください。
2. VPC 向け NAT インスタンスを削除します。

VPC を削除するために、NAT インスタンスに関連付けられている Amazon EBS ボリュームを削除する必要はありません。ただし、このボリュームを削除しない場合は、NAT インスタンスおよび VPC を削除した場合でも、引き続きお客様への課金が発生することになります。

3. [VPC] タブで、[Delete (削除)] リンクを選択して VPC を削除します。



4. [Delete VPC (VPC の削除)] ダイアログボックスで、[OK] を選択します。



CloudFormation Template Editor for Visual Studio の使用

Toolkit for Visual Studioには、Visual Studio 用の CloudFormation テンプレートエディタおよび CloudFormation テンプレートプロジェクトが含まれています。以下の機能がサポートされています。

- 提供された CloudFormation テンプレートプロジェクトのタイプを使った、新しいテンプレートの作成 (空または既存のスタックまたはサンプルテンプレートからコピーする)。
- 自動 JSON 検証、オートコンプリート、コードの折りたたみ、構文の強調表示を使った、テンプレートの編集。
- テンプレートでの、組み込み関数の自動候補およびフィールド値のリソース参照パラメータの自動候補。
- Visual Studio からテンプレートの一般的なアクションを実行するためのメニュー項目。

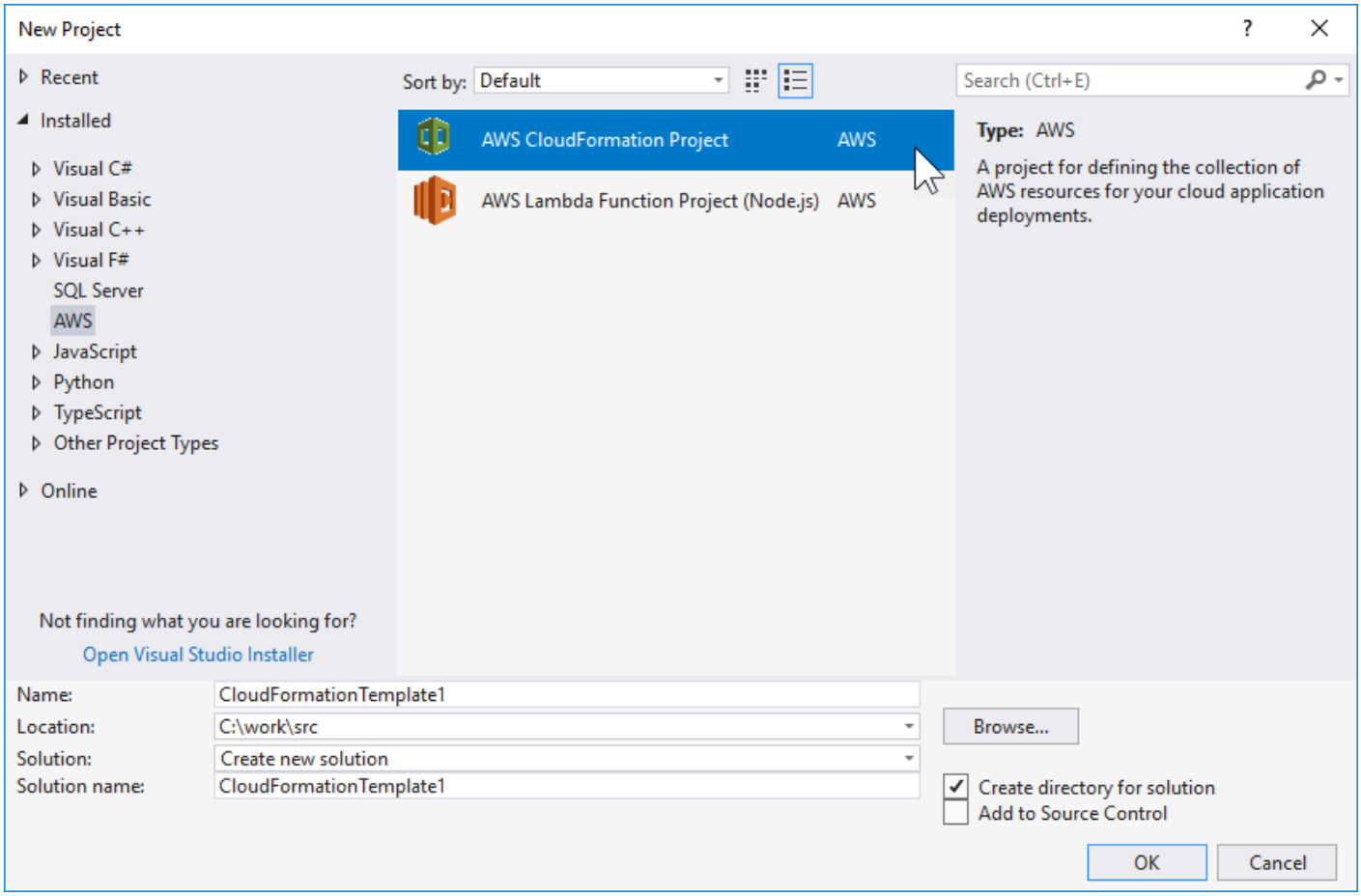
トピック

Visual Studio での CloudFormation テンプレートプロジェクトの作成

テンプレートプロジェクトを作成するには

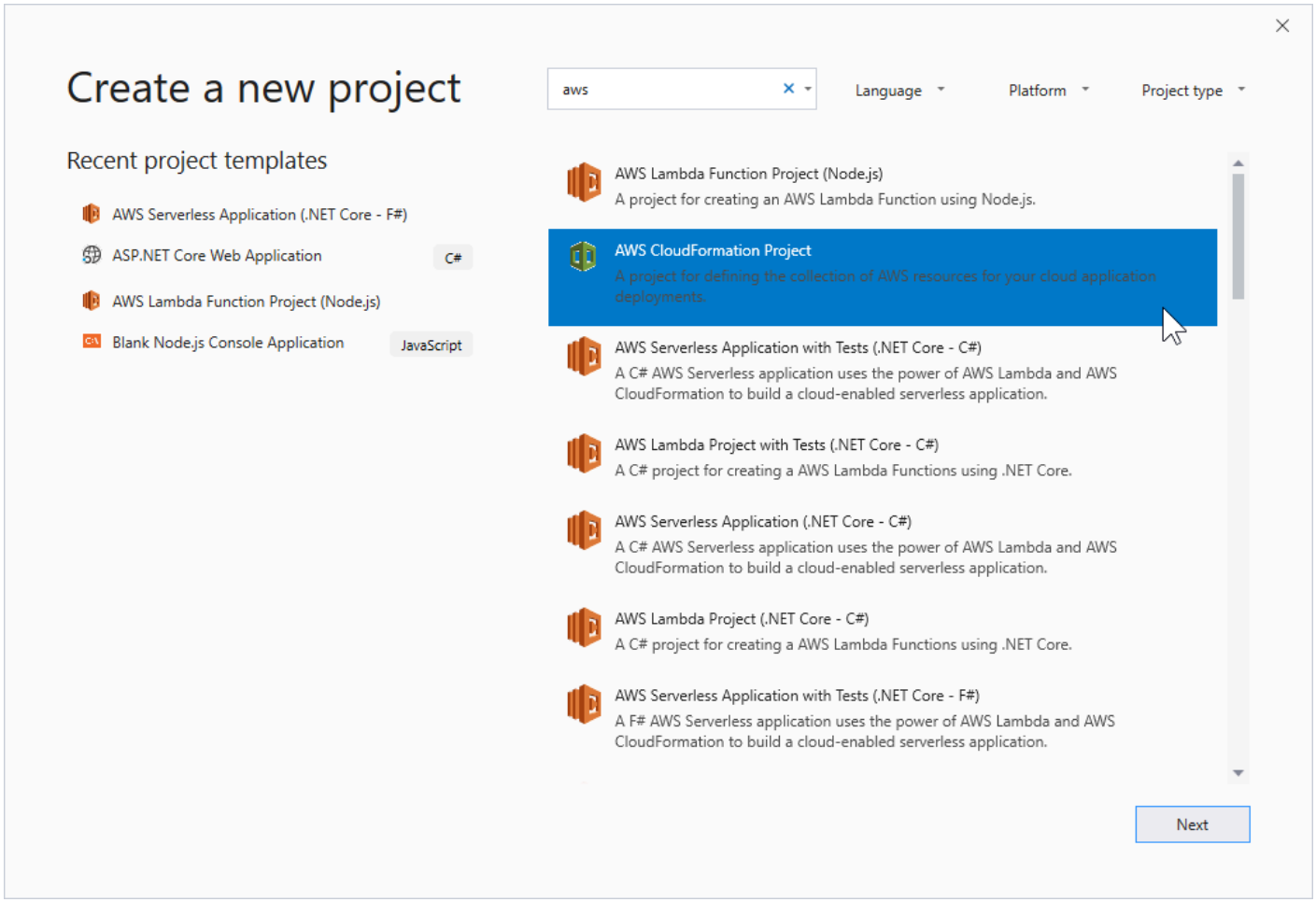
1. Visual Studio で [File (ファイル)]、[New (新規)] の順に選択し、[Project (プロジェクト)] を選択します。
2. Visual Studio 2017 の場合:

   [New Project] (新しいプロジェクト) ダイアログボックスで、[Installed] (インストール済み) を展開し、AWS を選択します。



Visual Studio 2019 の場合:

[新しいプロジェクト] ダイアログボックスで、[言語]、[プラットフォーム]、および [プロジェクトタイプ] のドロップダウンボックスが [すべて...] に設定されていることを確認して、[検索] フィールドに aws を入力します。



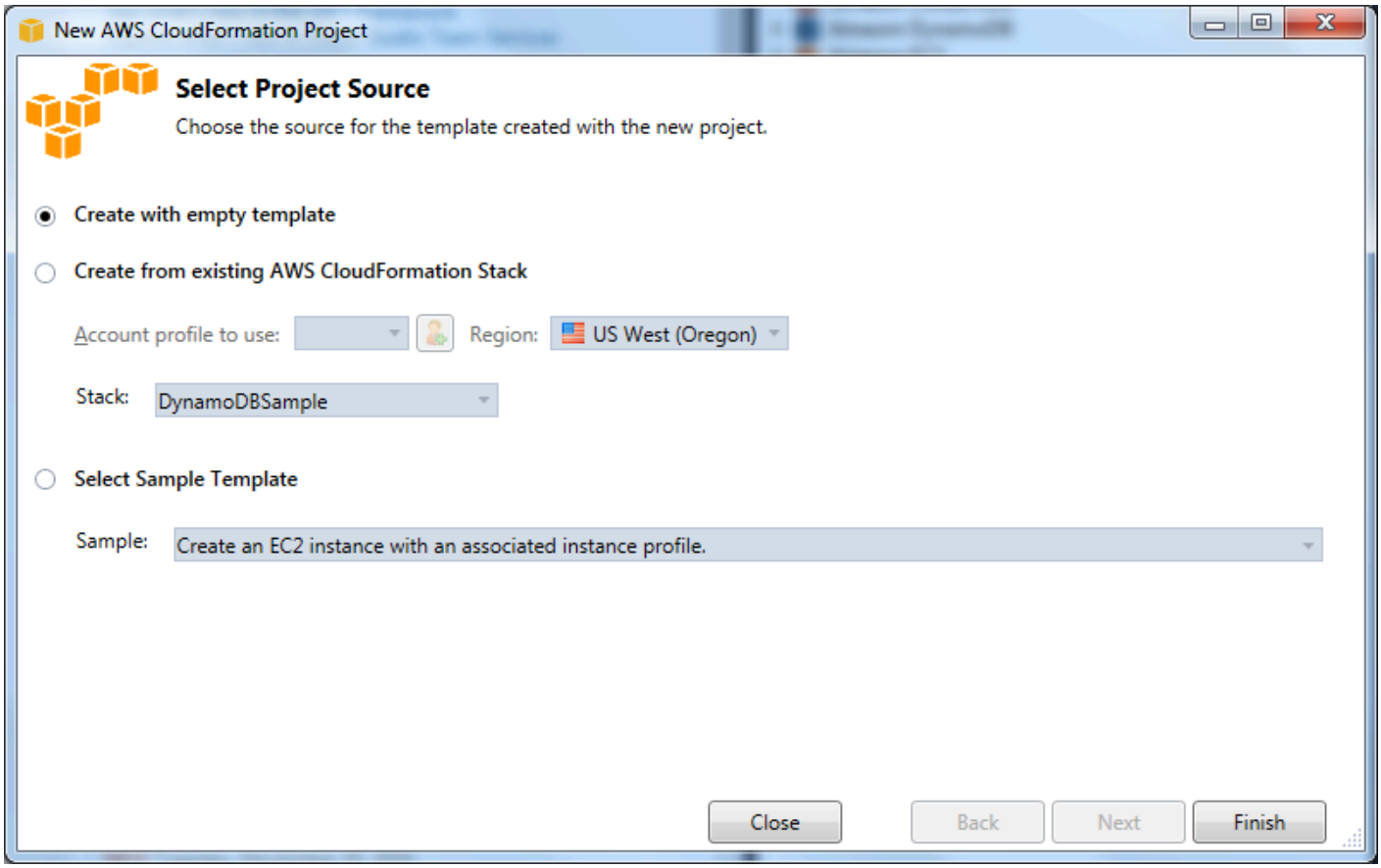
3. [AWS CloudFormation Project] (CloudFormation プロジェクト) テンプレートを選択します。
4. Visual Studio 2017 の場合:

   テンプレートプロジェクトの目的の [名前]、[場所] などを入力し、[OK] をクリックします。

   Visual Studio 2019 の場合:

   [次へ] をクリックします。次のダイアログで、テンプレートプロジェクトの目的の [名前]、[場所] などを入力し、[作成] をクリックします。
5. [Select Project Source (プロジェクトソースの選択)] ページで、作成するテンプレートのソースを選択します。
   - [Create with empty template (空のテンプレートを使用して作成)] では、新しい空の CloudFormation テンプレートが生成されます。
   - 既存の AWS |CFN| スタックから作成する は、 AWS アカウント内の既存のスタックからテンプレートを生成します。(スタックのステータスは `CREATE_COMPLETE` である必要はありません。)

- [Select sample template (サンプルテンプレートを選択)] では、 CloudFormation サンプルテンプレートの 1 つからテンプレートを生成します。

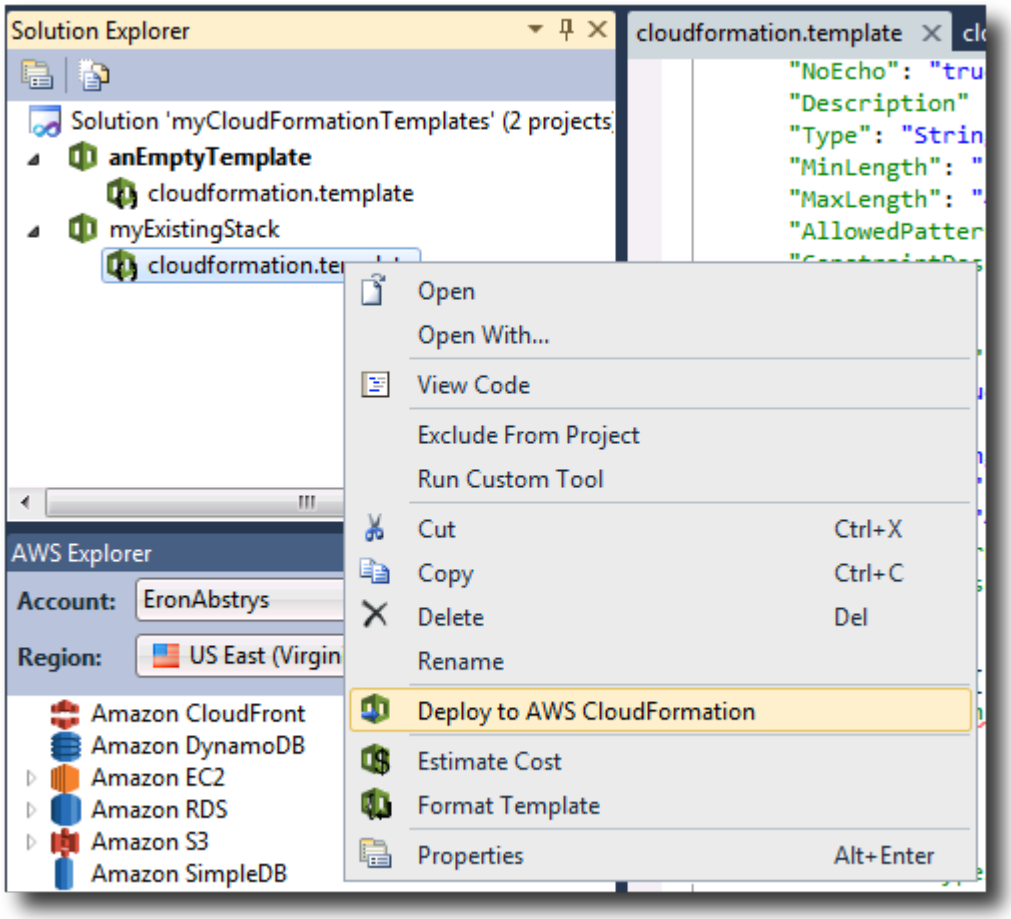


6. CloudFormation テンプレートプロジェクトの作成を完了するには、完了を選択します。

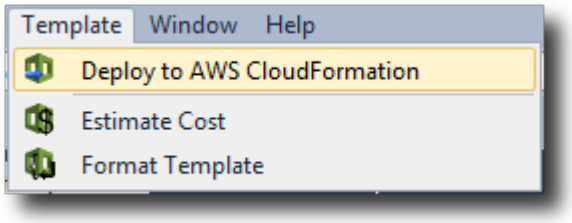
Visual Studio での CloudFormation テンプレートのデプロイ

CFN テンプレートをデプロイするには

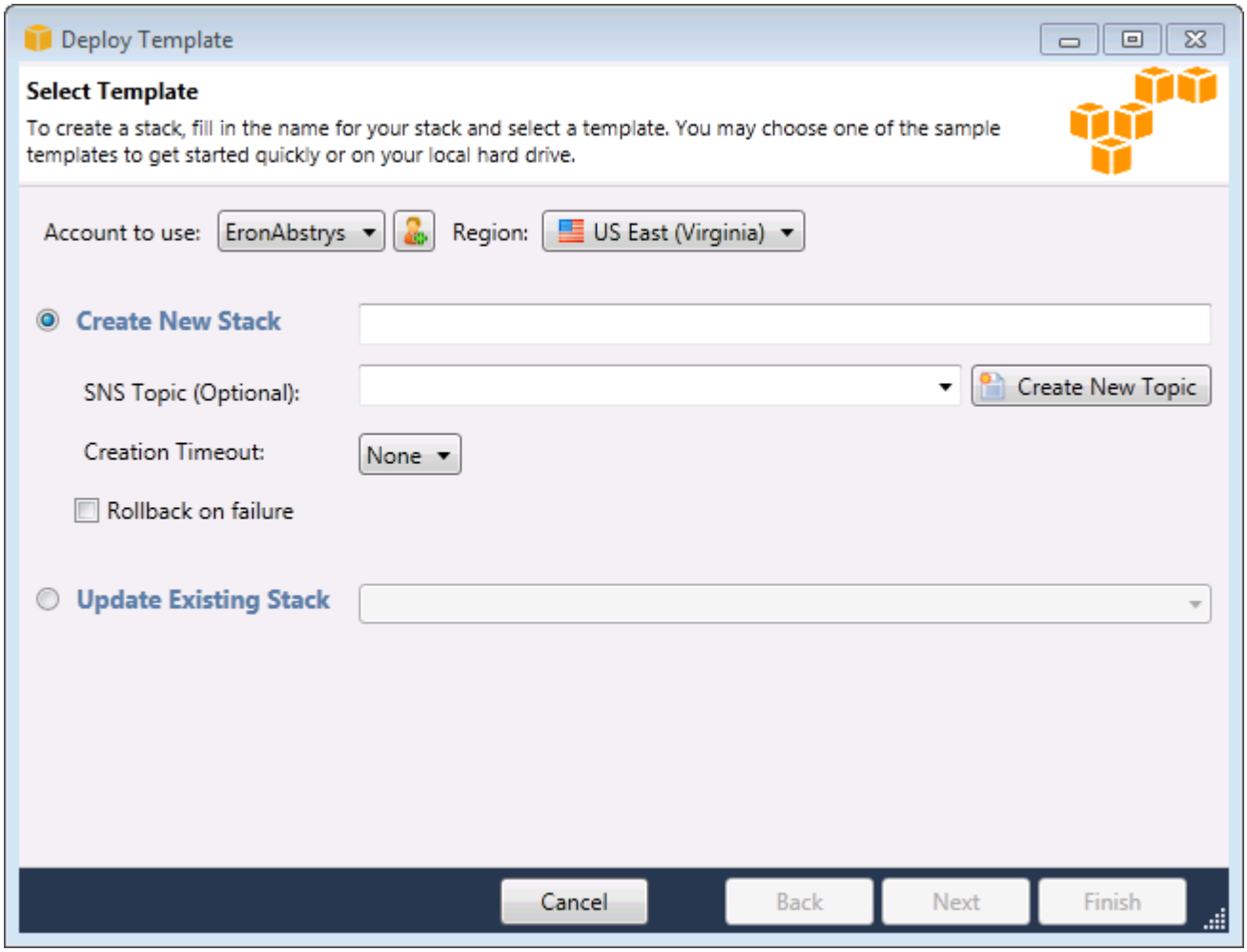
1. Solution Explorer でデプロイしたいテンプレートのコンテキスト (右クリック) メニューを開き、[Deploy to AWS CloudFormation CloudFormation] (CloudFormation へのデプロイ) を選択します。



また、現在編集中のテンプレートをデプロイするには、編集中の [Template] (テンプレート) メニューで、[Deploy to AWS CloudFormation CloudFormation] (CloudFormation へのデプロイ) を選択します。



2. [Deploy Template] (テンプレートのデプロイ) ページで、スタック起動に使用する AWS アカウント アカウントと起動するリージョンを選択します。

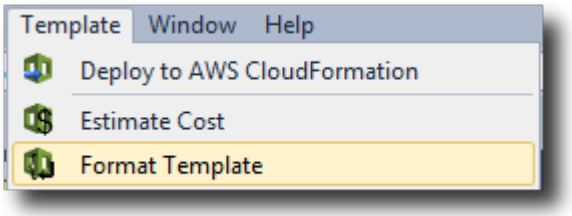


3. [Create New Stack (新しいスタックの作成)] を選択してスタックの名前を入力します。
4. 以下のオプションのいずれかを選択します (いずれも選択しない場合もあります)。
   - スタックの進行状況についての通知を受信するには、[SNS Topic (SNS トピック)] ドロップダウンリストで SNS トピックを選択します。[Create New Topic (新しいトピックの作成)] を選択し、ボックスに E メールアドレスを入力することで、SNS トピックを作成することもできます。
   - [Creation Timeout (タイムアウトの作成)] を使用して、スタックの作成を失敗と CloudFormation 判断するまでの時間 (および、[Rollback on failure (失敗時のロールバック)] オプションの選択が外れていない場合はロールバックされるまでの時間) を指定できます。
   - 失敗した際にスタックをロールバック (つまり削除) する場合は、[Rollback on failure (失敗時のロールバック)] を使用します。デバッグ目的で、起動に失敗したスタックをアクティブのままにする場合は、このチェックボックスをオフのままにします。
5. [Finish (完了)] をクリックして、スタックを起動します。

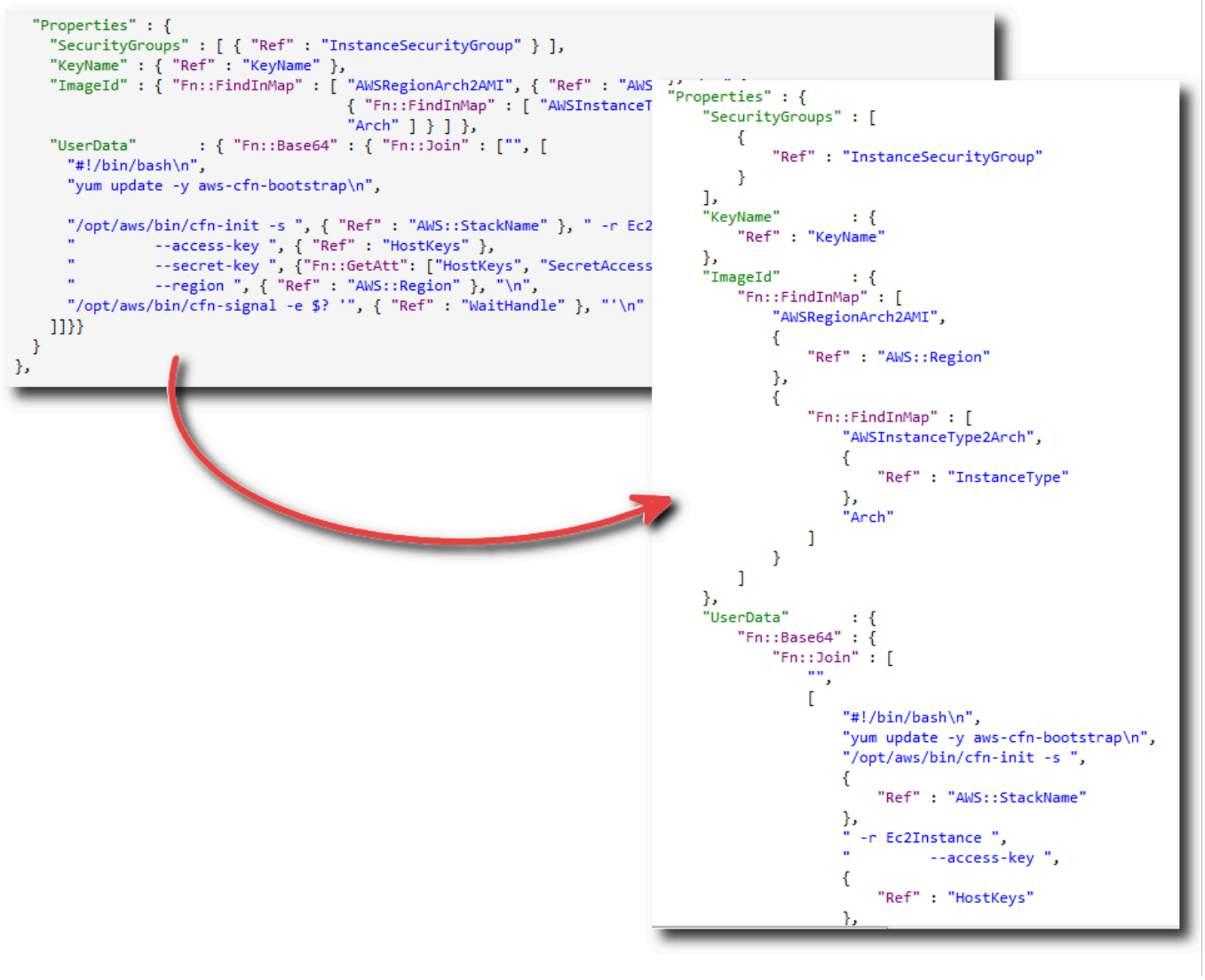
Visual Studio での CloudFormation テンプレートのフォーマット

- Solution Explorer でテンプレートのコンテキスト (右クリック) メニューを開いて、[Format Template (フォーマットテンプレート)] を選択します。

  また、現在編集中のテンプレートをフォーマットするには、編集中の [Template (テンプレート)] メニューで、[Format Template (フォーマットテンプレート)] を選択します。



  ユーザーの JSON コードがフォーマットされ、構造が明確に表示されます。



AWS Explorer からの Amazon S3 の使用

Amazon Simple Storage Service (Amazon S3) を使用すると、インターネット接続を介してデータの保存と取得が可能になります。Amazon S3 に保存したすべてのデータは、デフォルトではアカウントに関連付けられ、自分のみがアクセスできます。Toolkit for Visual Studio を使用すると、Amazon S3 にデータを保存して、そのデータを表示、管理、取得、配布することができます。

Amazon S3 では、バケットの概念を使用しており、これはファイルシステムまたは論理ドライブに類似したものと考えることができます。バケットにはフォルダとオブジェクトを含めることができます。フォルダはディレクトリに類似しています。オブジェクトはファイルに類似しています。このセクションでは、これらの概念を使用して、Toolkit for Visual Studio によって公開されている Amazon S3 の機能を紹介します。

> **Note**
>
> このツールを使用するには、IAM ポリシーで `s3:GetBucketAcl`、`s3:GetBucket` および `s3:ListBucket` のアクションについてアクセス許可を付与する必要があります。詳細については、「IAM AWS ポリシーの概要」を参照してください。

Amazon S3 バケットを作成する

バケットは Amazon S3 で最も基本的なストレージの単位です。

S3 バケットを作成するには

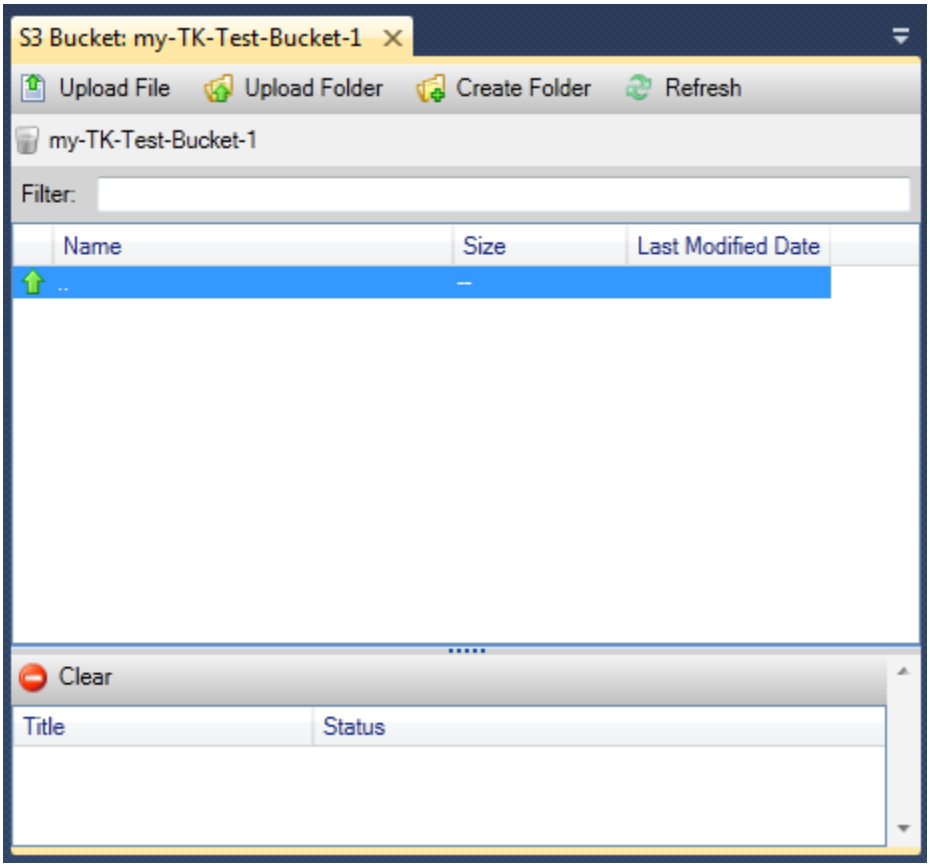
1. AWS Explorer で、Amazon S3 ノードのコンテキスト (右クリック) メニューを開き、バケットの作成を選択します。
2. [Create Bucket (バケットの作成)] ダイアログで、バケットの名前を入力します。バケット名は、AWSにわたって一意である必要があります。その他の制約については、Amazon S3 ドキュメントを参照してください。
3. [OK] を選択してください。

AWS Explorer からの Amazon S3 バケットの管理

AWS Explorer では、Amazon S3 バケットのコンテキスト (右クリック) メニューを開くと、次のオペレーションを使用できます。

参照

バケットに含まれるオブジェクトのビューを表示します。ここでは、フォルダを作成したり、ローカルコンピュータからファイルまたはディレクトリ全体とフォルダをアップロードすることができます。下のペインには、アップロードプロセスについてのステータスメッセージが表示されます。これらのメッセージをクリアするには、[Clear (クリア)] アイコンを選択します。 AWS Explorer でバケット名をダブルクリックして、バケットのこのビューにアクセスすることもできます。



プロパティ

ダイアログボックスが表示されます。ここでは、次のことを実行できます。

- Amazon S3 アクセス許可を次の範囲に設定します。
  - バケット所有者。
  - AWS上で認証されたすべてのユーザー。
  - インターネットにアクセスできるすべてのユーザー。
- バケットのログ記録を有効にします。
- Amazon Simple Notification Service (Amazon SNS) を使用して通知をセットアップして、低冗長化ストレージ (RRS) を使用している場合に、データ損失が発生したときに通知されるようにします。RRS は、標準のストレージよりも耐久性とコストが低い、Amazon S3 のストレージオプションです。詳細については、「S3 のよくある質問」を参照してください。
- バケットのデータを使用して静的ウェブサイトを作成します。

Policy

バケットの AWS Identity and Access Management (IAM) ポリシーを設定できます。詳細については、IAM ドキュメントならびに IAM および S3 のユースケースを参照してください。

署名付き URL の作成

バケットのコンテンツへのアクセスを提供するために配布できる、時間制限のある URL を生成します。詳細については、「署名付き URL の作成方法」を参照してください。

マルチパートアップロードの表示

マルチパートアップロードを表示します。Amazon S3 は、大規模なオブジェクトをいくつかに分割してアップロードして、アップロードのプロセスを効率的にすることができます。詳細については、S3 ドキュメントのマルチパートアップロードの説明を参照してください。

[Delete] (削除)

バケットを削除します。空のバケットのみを削除できます。

ファイルとフォルダを Amazon S3 にアップロードする

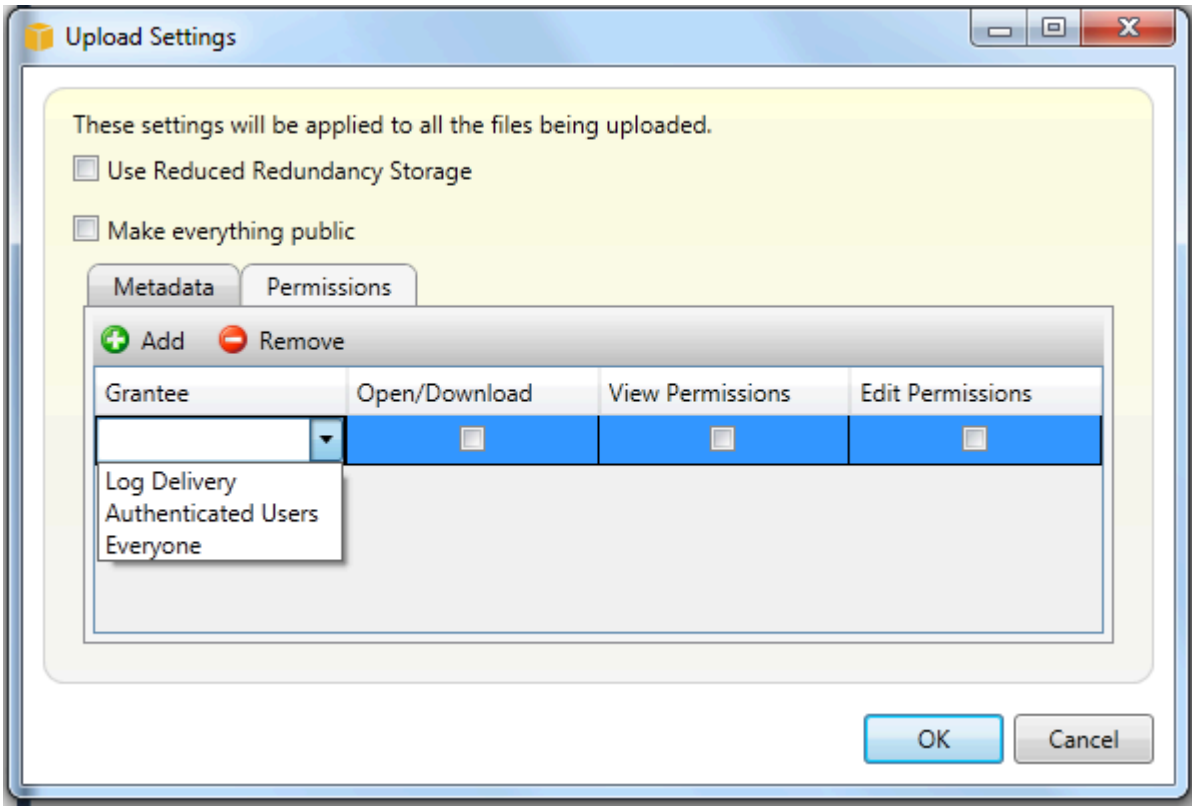
AWS Explorer を使用して、ローカルコンピュータから任意のバケットにファイルまたはフォルダ全体を転送できます。

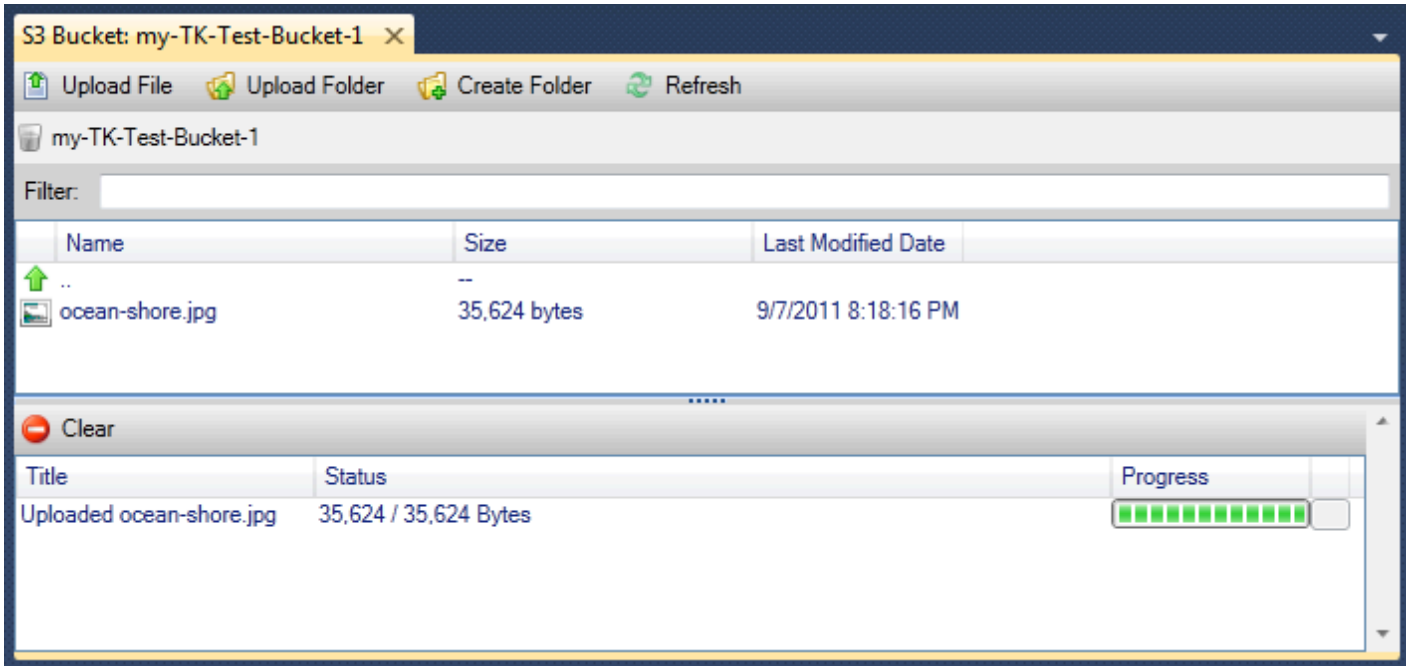
> **Note**
>
> Amazon S3 バケットに既に存在するファイルまたはフォルダと同じ名前を持つファイルまたはフォルダをアップロードすると、アップロードされたファイルは既存のファイルを警告なしに上書きします。

S3 にファイルをアップロードするには、以下を行います。

1. AWS Explorer で Amazon Amazon S3ノードを展開し、バケットをダブルクリックするか、バケットのコンテキスト (右クリック) メニューを開き、参照を選択します。
2. バケットの [Browse (参照)] ビューで、[Upload File (ファイルのアップロード)] または [Upload Folder (フォルダのアップロード)] を選択します。
3. [File-Open] ダイアログボックスで、アップロードするファイルに移動して、それを選択し、[Open (開く)] を選択します。フォルダをアップロードする場合は、そのフォルダに移動して選択し、[Open (開く)] を選択します。

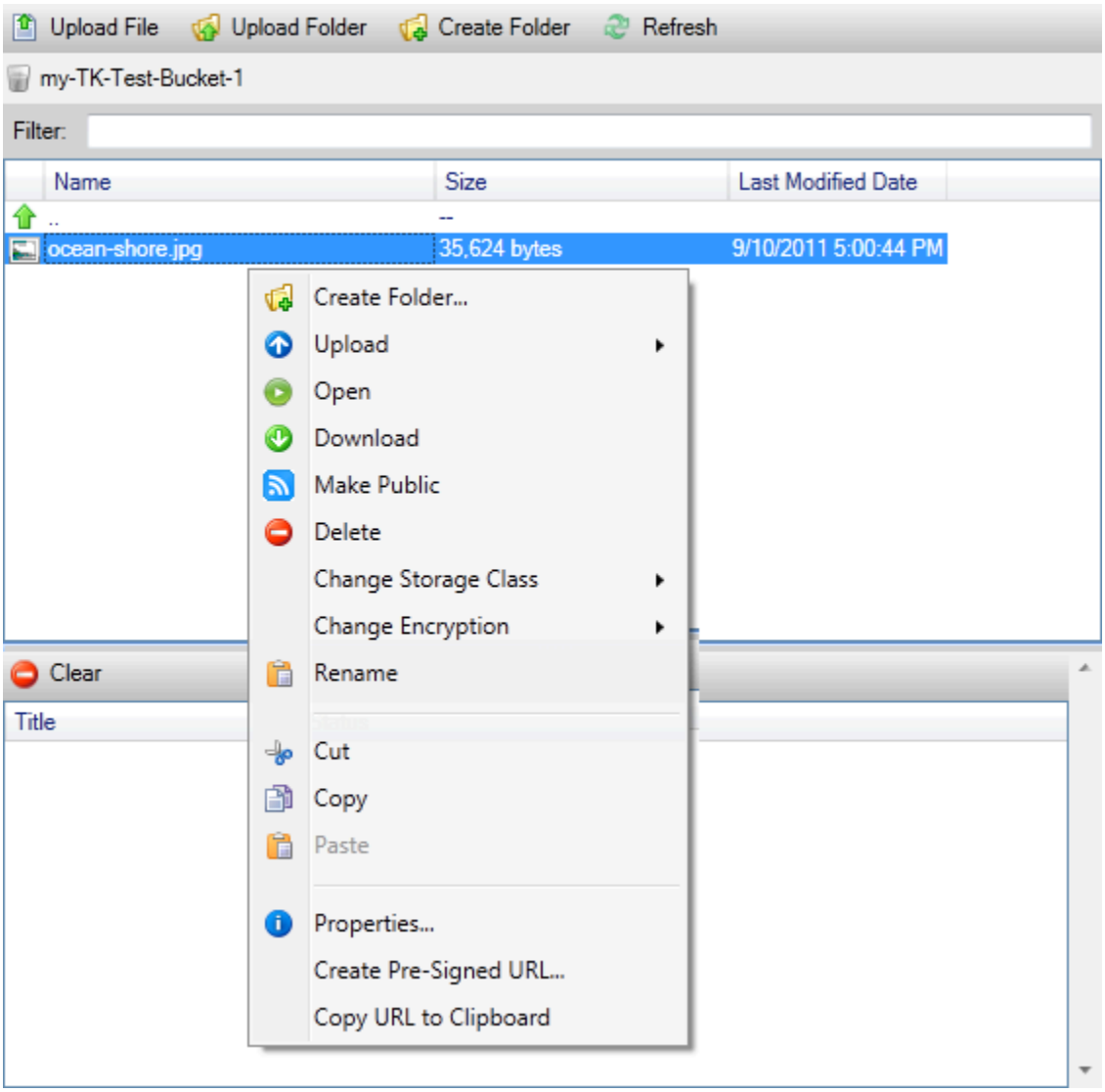
[Upload Settings (アップロード設定)] ダイアログボックスを使用すると、アップロードするファイルやフォルダに、メタデータとアクセス権限を設定できます。[Make everything public (すべてを公開する)] チェックボックスを選択すると、[Everyone (全員)] に [Open/Download (開く/ダウンロード)] のアクセス権限を設定することと同じになります。アップロードされたファイルに低冗長化ストレージを使用するオプションを選択することができます。





AWS Toolkit for Visual Studio の Amazon S3 ファイルオペレーション

Amazon S3 ビューでファイルを選択して、コンテキスト (右クリック) メニューを開くと、ファイルにさまざまなオペレーションを実行できます。



フォルダを作成する

現在のバケット内にフォルダを作成することができます。([Create Folder (フォルダの作成)] のリンクを選択した場合と同じです。)

アップロード

ファイルまたはフォルダをアップロードすることができます。([Upload File (ファイルのアップロード)] または [Upload Folder (フォルダのアップロード)] のリンクを選択した場合と同じです。)

開く

選択したファイルをデフォルトのブラウザで開きます。ファイルの種類とデフォルトのブラウザの機能によっては、ファイルが表示されない場合があります。ブラウザによってダウンロードが行われる場合もあります。

ダウンロード

[Folder-Tree] ダイアログボックスが開き、選択したファイルをダウンロードできます。

公開する

選択したファイルについて [Everyone]] (全員) に [Open/Download] (開く/ダウンロード) のアクセス許可を設定します。([Upload Settings] のダイアログボックスで [Make everything public] のチェックボックスを選択するのと同じです。)

[Delete] (削除)

選択されたファイルまたはフォルダを削除します。ファイルまたはフォルダを選択し、`Delete` を押して削除することもできます。

ストレージクラスの変更

ストレージクラスをスタンダードまたは低冗長化ストレージ (RRS) に設定します。現在のストレージクラスの設定を表示するには、[Properties (プロパティ)] を選択します。

暗号化の変更

ファイルのサーバー側の暗号化を設定できます。現在の暗号化の設定を表示するには、[Properties (プロパティ)] を選択します。

[Rename] (名前の変更)

ファイルの名前を変更することができます。フォルダ名を変更することはできません。

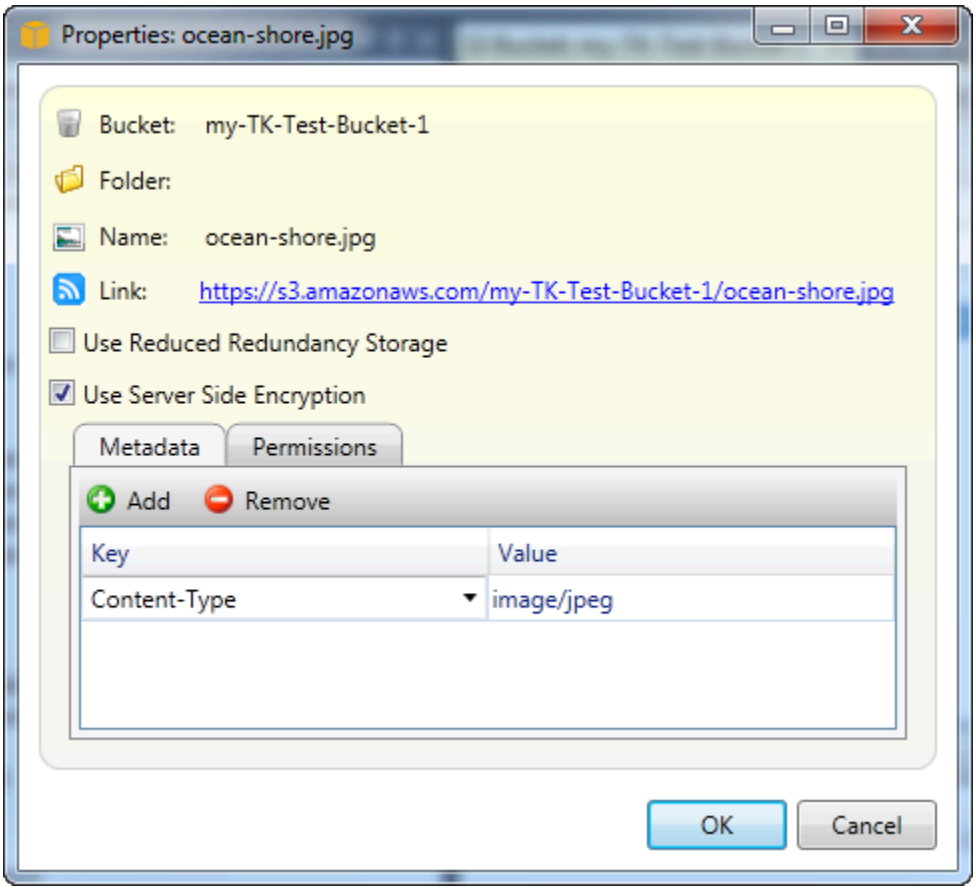
切り取り | コピー | 貼り付け

フォルダ間またはバケット間で、ファイルまたはフォルダの切り取り、コピー、貼り付けを行うことができます。

プロパティ

表示されるダイアログボックスを使って、ファイルにメタデータとアクセス許可を設定できます。ファイルのストレージを低冗長化ストレージ (RRS) またはスタンダードに切り替えることができます。ファイルにサーバー側の暗号化を設定できます。このダイアログボックスには、ファイルへの https リンクも表示されます。このリンクを選択すると、Toolkit for Visual Studio はデフォルトのブ

ラウザでファイルを開きます。ファイルについて [Everyone] (全員) に [Open/Download] (開く/ダウンロード) のアクセス許可を設定すると、このリンクを使って他のユーザーがファイルにアクセスできます。このリンクを配布するのではなく、署名付き URL を作成して配布することをお勧めします。



署名付き URL の作成

Amazon S3 に保存されているコンテンツに他のユーザーがアクセスできるように配布する、時間制限のある署名付き URL を作成します。

署名付き URL の作成方法

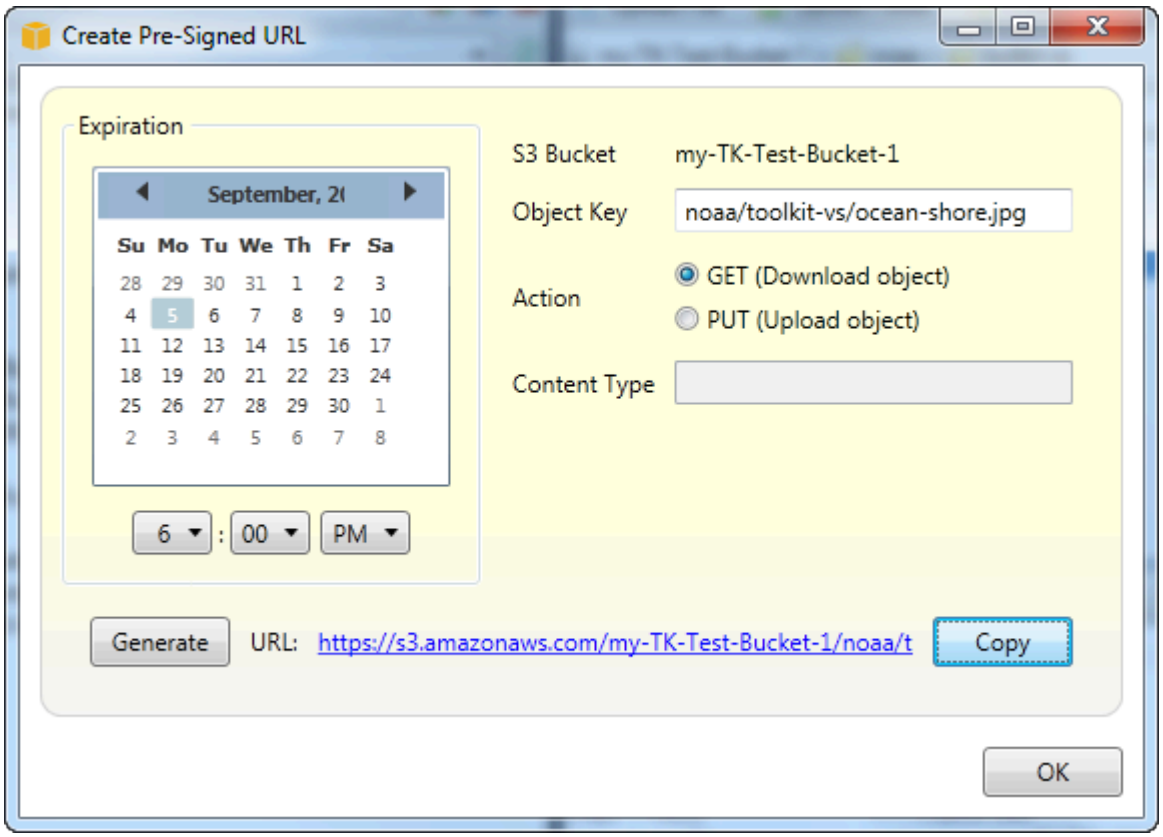
バケットまたはバケット内のファイルに署名付き URL を作成できます。他のユーザーはこの URL を使用して、バケットまたはファイルにアクセスできます。この URL は、URL の作成時に指定した有効期限になると、失効します。

署名付き URL を作成するには

1. [Create Pre-Signed URL (署名付き URL の作成)] ダイアログボックスで、URL の有効期限の日時を設定します。デフォルト設定では、現在の時間から 1 時間後に設定されます。

2. [生成] ボタンを選択します。

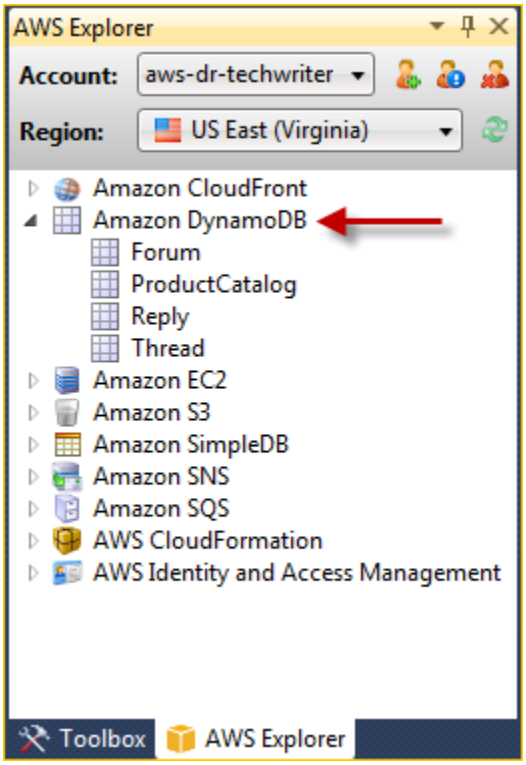
3. URL をクリップボードにコピーするには、[コピー] を選択します。



AWS Explorer から DynamoDB を使用する

Amazon DynamoDB は、拡張性と可用性に優れた、費用効果の高い、高速な非リレーショナルデータベースサービスです。DynamoDB により、データストレージに対して低いレイテンシーと予測可能なパフォーマンスを維持しながら、従来の拡張性の限界を排除できます。Toolkit for Visual Studio には、開発の場面でDynamoDB の操作を行う機能が用意されています。DynamoDB の詳細については、Amazon Web Services ウェブサイトの「DynamoDB」を参照してください。

Toolkit for Visual Studio では、AWS Explorer は、アクティブな AWS アカウント に関連付けられているすべての DynamoDB テーブルが表示されます。



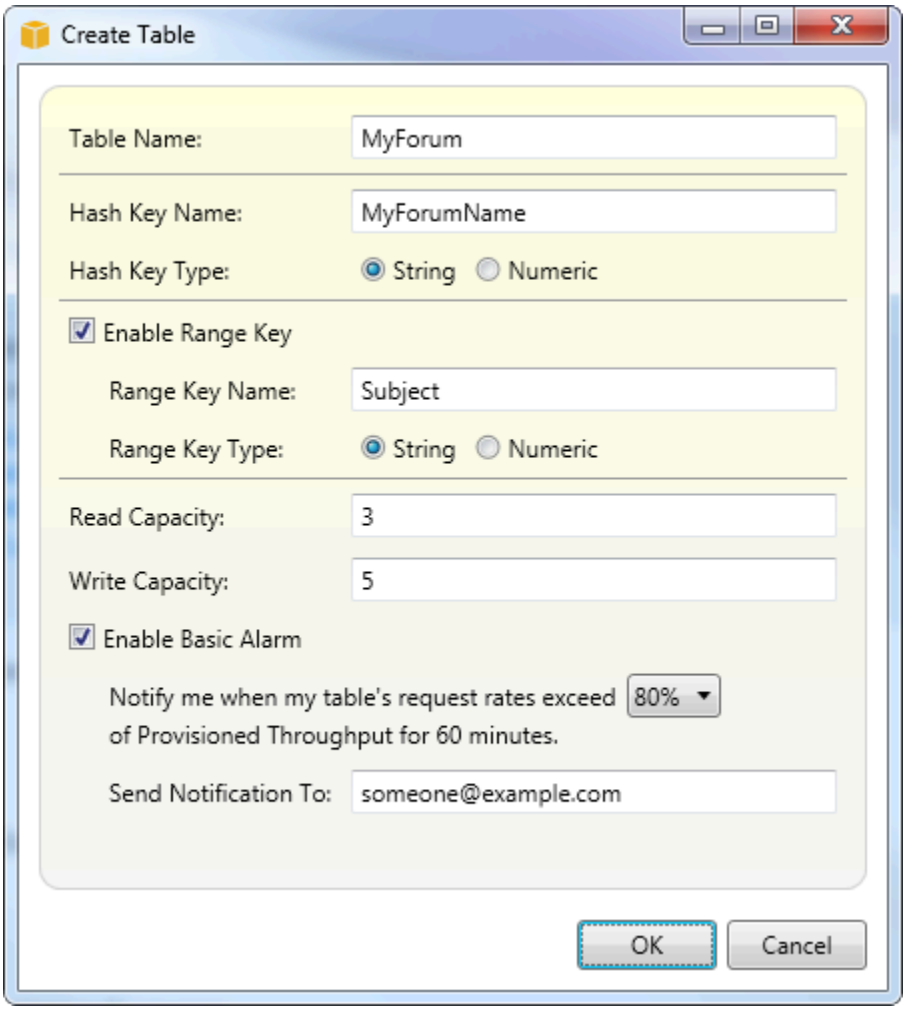
DynamoDB テーブルの作成

Toolkit for Visual Studio を使用して DynamoDB 表を作成できます。

AWS Explorer でテーブルを作成するには

1. AWS Explorer で、[Amazon DynamoDB] のコンテキスト (右クリック) メニューを開いて、[Create Table] (テーブルの作成) を選択します。
2. [Create Table (テーブルの作成)] ウィザードで、[Table Name (テーブル名)] に、テーブルの名前を入力します。
3. [Hash Key Name] (ハッシュキー名) フィールドにプライマリハッシュキー属性を入力し、[Hash Key Type] (ハッシュキータイプ) ボタンから、ハッシュキータイプを選択します。DynamoDBによって、プライマリキー属性を使用するハッシュインデックス (ソートなし) が生成され、必要に応じて、レンジプライマリキー属性を使用するレンジインデックス (ソートあり) が生成されます。プライマリハッシュキー属性の詳細については、Amazon DynamoDB デベロッパーガイドの「プライマリキー」を参照してください。
4. (オプション) [Enable Range Key (レンジキーを有効にする)] を選択します。[Range Key Name (レンジキー名)] フィールドにレンジキー属性を入力し、[Range Key Type (レンジキータイプ)] ボタンでレンジキータイプを選択します。

5. [Read Capacity (読み込みキャパシティー)] フィールドで、読み込みキャパシティーユニットの数を入力します。[Write Capacity (書き込みキャパシティー)] フィールドで、書き込みキャパシティーユニットの数を入力します。読み込みキャパシティーユニット数として 3 以上、書き込みキャパシティーユニット数として 5 以上の値を指定する必要があります。読み込みおよび書き込みキャパシティーユニットに関する情報は、「プロビジョニングされたスループット」を参照してください。
6. (オプション) [Enable Basic Alarm (基本アラームを有効にする)] をクリックすると、テーブルのリクエスト率が高すぎるときに警告を発します。プロビジョニングされたスループットの割合 (60 分ごと) のしきい値を選択します。この値を超えるとアラートが送信されます。[Send Notifications To (通知の送信先)] に E メールアドレスを入力します。
7. [OK] をクリックすると、テーブルが作成されます。

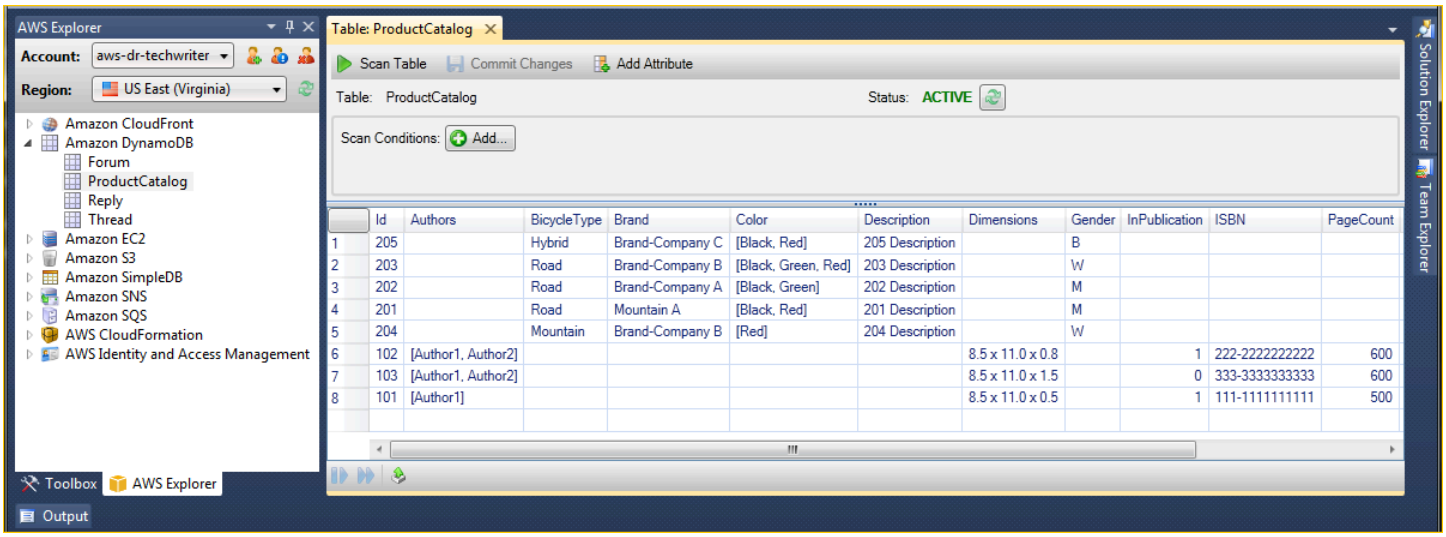


DynamoDB テーブルの詳細については、「データモデルのコンセプト - テーブル、項目、属性」を参照してください。

DynamoDB テーブルをグリッドとして表示する

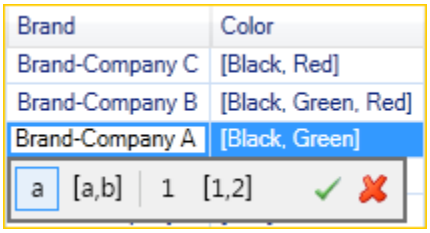
いずれかの DynamoDB テーブルのグリッドビューを開くには、AWS Explorer でテーブルに対応するサブノードをダブルクリックします。グリッドビューで、テーブルに格納されている項目、属性、値を表示できます。各行は、テーブル内の項目に対応しています。各列は、テーブル内の属性に対応しています。グリッドビューの各セルには、その項目のその属性に関連付けられている値が表示されます。

属性の値は、文字列または数字です。文字列または数字のセットになる場合もあります。セット値は、角かっこで囲まれたカンマ区切りのリストとして表示されます。

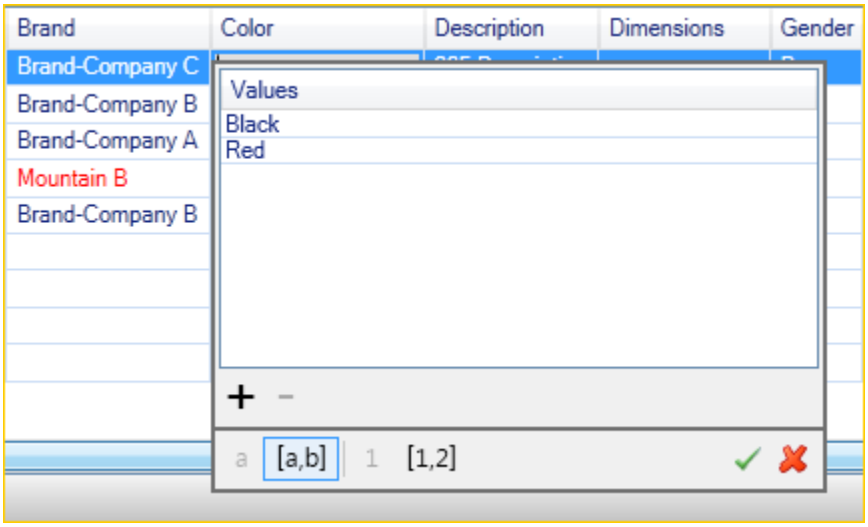


属性と値を編集および追加する

セルをダブルクリックすることで、項目の対応する属性の値を編集できます。セット値の属性の場合は、セットの個々の値を追加したり削除したりできます。



属性の値を変更するだけでなく、一部制限はありますが、属性の値の形式も変更できます。例えば、任意の数値を文字列値に変換できます。文字列値があり、その内容が「125」などの数字である場合は、セル工ディタにより値の形式を文字列から数字に変換できます。また、単一の値をセット値に変換できます。ただし一般的に、セット値から 1 つの値に変換することはできません。その例外として、セット値の値が実際は 1 つしかない場合は 1 つの値に変換できます。

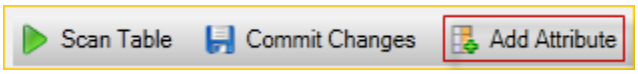


属性値の編集後、変更を確定するには、緑のチェックマークを選択します。変更を破棄する場合は、赤の [X] をクリックします。

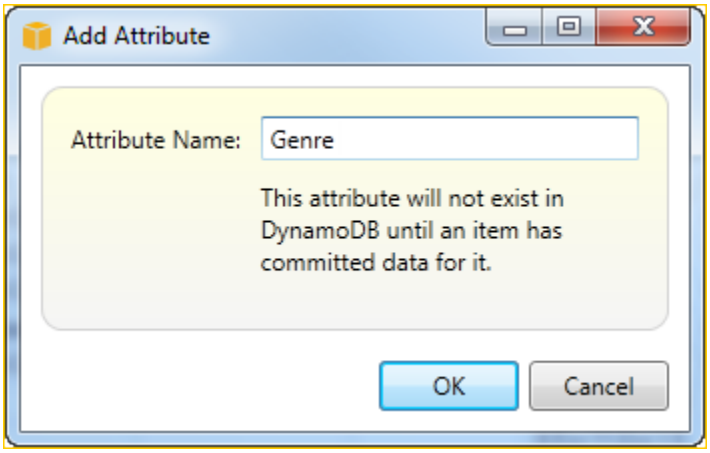
変更の確定後、属性値は赤で表示されます。これは、属性が更新されましたことを示しますが、新しい値が Dynamo DB データベースにまだ書き戻されていないことを示します。DynamoDB に変更を書き戻すには、[Commit Changes] (変更をコミット) を選択します。変更を破棄するには、[Scan Table (テーブルスキャン)] をクリックし、スキャン前に変更をコミットするかどうか Toolkit によって尋ねられたら、[No] をクリックします。

属性の追加

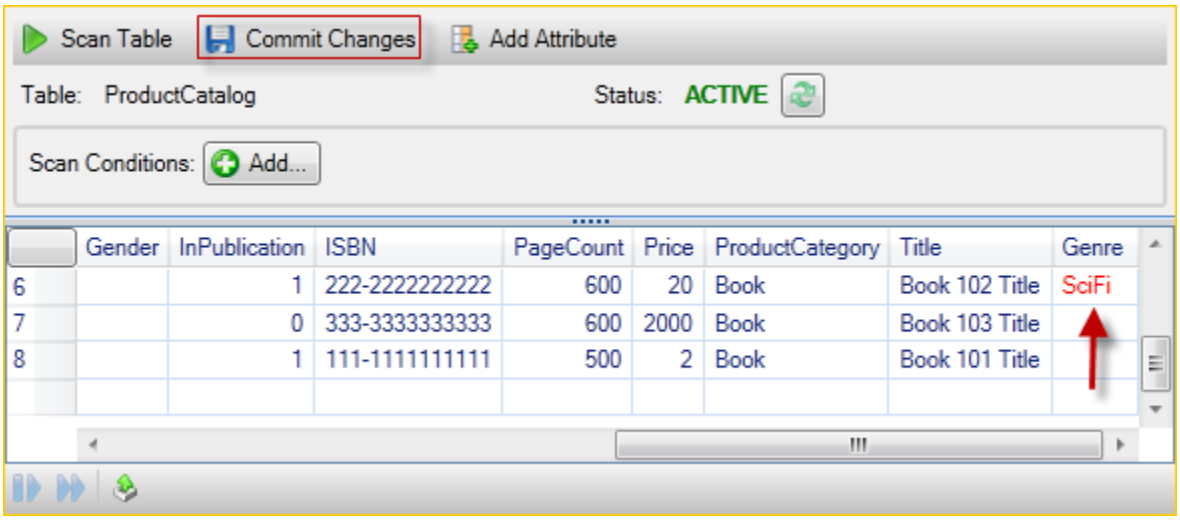
グリッドビューで、テーブルに属性を追加することもできます。新しい属性を追加するには、[Add Attribute (属性を追加)] を選択します。



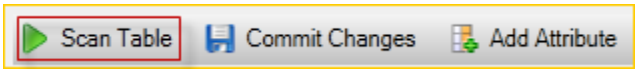
[Add Attribute (属性を追加)] ダイアログボックスに使用する属性の名前を入力して、[OK] をクリックします。



新しい属性をテーブルの一部にするには、少なくとも 1 つの項目に値を追加し、[Commit Changes (変更をコミット)] ボタンをクリックします。新しい属性を破棄するには、[Commit Changes (変更をコミット)] を選択せずにテーブルのグリッドビューを閉じるだけです。



DynamoDB テーブルをスキャンする



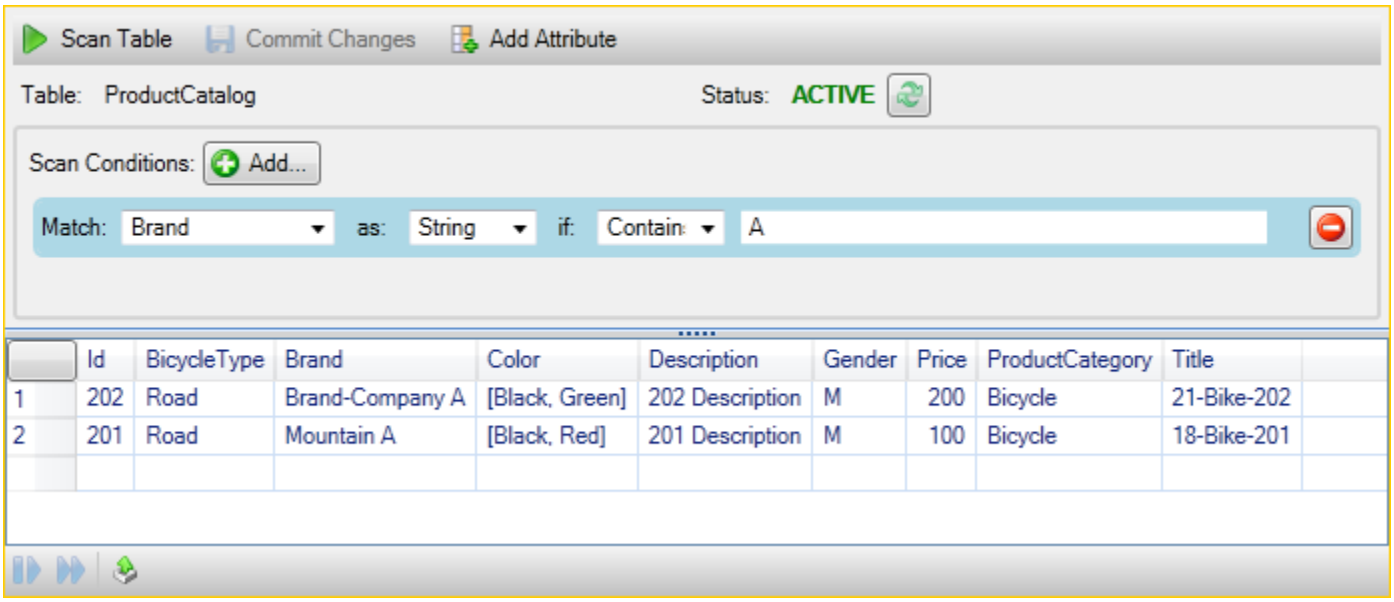
Toolkit から、 DynamoDB テーブルのスキャンを実行できます。スキャンでは、定義した一連の条件に一致したテーブルの項目がすべて返されます。スキャンはワークロードの高いオペレーションであるため、テーブルに対する優先度の高い本稼働トラフィックが中断されないように、注意して使用する必要があります。スキャンオペレーションの使用の詳細については、Amazon DynamoDB デベロッパーガイドを参照してください。

AWS Explorer から DynamoDB テーブルのスキャンを実行するには

1. グリッドビューで、[Scan Conditions: Add (スキャン条件: 追加)] ボタンをクリックします。

2. スキャン句のエディタで、一致条件の対象属性、属性値の解釈方法 (文字列、数値、セット値)、一致の方法 ([Begins With]、[Contains] など)、一致すべきリテラル値を選択します。
3. 検索での必要に応じてスキャン句を追加します。スキャンによってすべてのスキャン句の条件に一致する項目のみが返されます。スキャンでは、文字列値に対する一致を調べるときに大文字と小文字が区別されます。
4. グリッドビューの上部にあるボタンバーで、[Scan Table (テーブルスキャン)] を選択します。

スキャン句を削除するには、各フレーズの右側にある赤色のボタン (-) を選択します。



すべての項目が表示されたテーブルのビューに戻るには、すべてのスキャン句を削除し、[Scan Table(テーブルスキャン)] を再度選択します。

スキャン結果をページ分割する

ビューの下部には 3 つのボタンがあります。

最初の 2 つの青色のボタンをクリックすると、スキャン結果がページ分割されます。最初のボタンでは結果の次のページが表示されます。2 番目のボタンでは、結果の 10 ページ先が表示されます。この場合、1 ページは 1 MB に相当します。

スキャン結果を CSV にエクスポートする

3 番目のボタンをクリックすると、現在のスキャンの結果が CSV ファイルにエクスポートされます。

Visual Studio Team Explorer と AWS CodeCommit の併用

Team Explorer 内で AWS Identity and Access Management (IAM) ユーザーアカウントを使用して、Git 認証情報を作成すると、それを使用してリポジトリを作成したりリポジトリのクローンを作成できます。

AWS CodeCommit の認証情報の種類

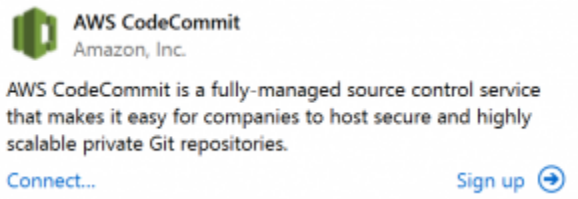
AWS Toolkit for Visual Studio のユーザーのほとんどは、アクセスキーとシークレットキーを含む AWS 認証情報プロファイルの設定を認識しています。これらの認証情報プロファイルは Toolkit for Visual Studio でサービス API を呼び出すために使用されます。例えば、Amazon S3 バケットを AWS Explorer で一覧表示したり、Amazon EC2 インスタンスを起動したりすることができます。AWS CodeCommit と Team Explorer の統合も、これらの認証情報プロファイルを使用します。ただし、Git 自体を使用するには、追加の認証情報が必要となります。具体的には、HTTPS 接続のための Git 認証情報が必要となります。これらの認証情報 (ユーザー名とパスワード) については、「AWS CodeCommit ユーザーガイド」の「Git 認証情報を使用する HTTPS ユーザー用のセットアップ」を参照してください。

AWS CodeCommit の Git 認証情報は、ユーザーアカウントに対してのみ作成できます。ルートアカウントに作成することはできません。サービスに対してこれらの認証情報のセットを 2 つまで作成できます。認証情報のセットを非アクティブとしてマークすることができますが、非アクティブなセットは引き続き 2 セットの制限にカウントされます。認証情報はいつでも削除と再作成を行うことができます。AWS CodeCommit を Visual Studio 内から使用する場合 (例えば、リポジトリの作成や一覧表示を行う場合など) は、従来の AWS 認証情報を使ってサービス自体を使用します。AWS CodeCommit でホストされている実際の Git リポジトリを使用する場合は、Git の認証情報を使用します。

AWS CodeCommit のサポートの一部として、Toolkit for Visual Studio によって、これらの Git 認証情報は自動的に作成されて、管理され、AWS 認証情報プロファイルに関連付けられます。Team Explorer 内で Git オペレーションを実行するために適切な認証情報のセットが設定されているかどうかを心配する必要はありません。AWS 認証情報プロファイルを使用して Team Explorer に接続すると、Git リモート使用している場合は常に関連付けられている Git 認証情報が自動的に使用されます。

を に接続するAWS CodeCommit

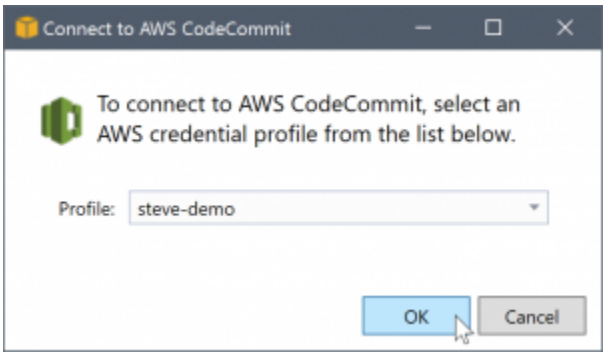
Visual Studio 2015 以降で Team Explorer ウィンドウを開くと、[Manage Connections] の [Hosted Service Providers] セクションに AWS CodeCommit エントリが表示されます。



[Sign up] (サインアップ) を選択すると、ブラウザウィンドウで Amazon Web Services ホームページが開きます。[Connect] (接続) を選択したときに発生する動作は、Toolkit for Visual Studio で AWS アクセス権とシークレットキーを持つ認証プロファイルが見つかり、ユーザーに代わって AWS の呼び出しを実行できるかどうかによって変わります。Toolkit for Visual Studio でローカルに保存された認証情報が見つからない場合には、IDE に表示される新しい [ご利用開始にあたって] のページを使って認証情報プロファイルが設定されている場合もあります。あるいは、Toolkit for Visual Studio、AWS Tools for Windows PowerShell、または AWS CLI を使用したことがあり、Toolkit for Visual Studio で使用できる AWS 認証情報プロファイルが既にあるという場合もあります。

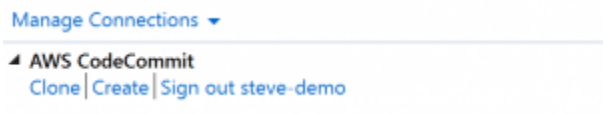
[Connect] (接続) を選択すると、Toolkit for Visual Studio で、接続で使用する認証情報プロファイルの検索が始まります。Toolkit for Visual Studio で認証情報プロファイルが見つからない場合は、ダイアログボックスが表示され、そこで AWS アカウント のアクセスキーとシークレットキーを入力できます。ルート認証情報ではなく、IAM ユーザーアカウントを使用することをお勧めします。また、前に説明したように、最終的に必要となる Git 認証情報は IAM ユーザーにのみ作成できます。アクセスキーとシークレットキーが用意され、認証情報プロファイルが作成されると、Team Explorer と AWS CodeCommit の間の接続を使用できるようになります。

Toolkit for Visual Studio で複数の AWS 認証情報プロファイルが見つかった場合には、Team Explorer で使用するアカウントを選択するように求められます。



認証情報プロファイルが 1 つのみの場合、Toolkit for Visual Studio では、プロファイル選択のダイアログボックスは表示されずに直ちに接続されます。

認証情報プロファイルを使って Team Explorer と AWS CodeCommit 間で接続が確立されると、ダイアログボックスは閉じられて、接続パネルが表示されます。

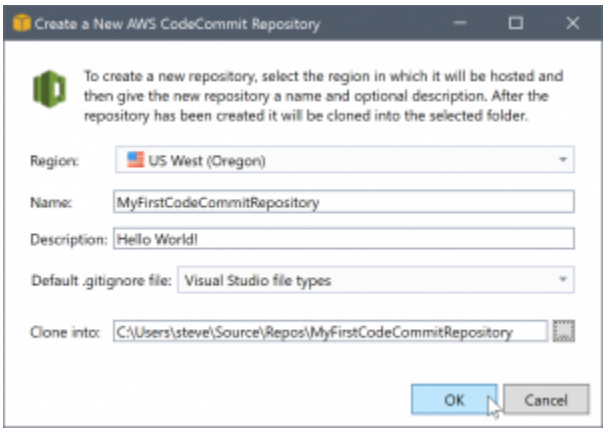


ローカルにクローンが作成されたリポジトリがないため、パネルには実行可能なオペレーションとして [Clone] (クローン)、[Create] (作成)、および [Sign out] (サインアウト) のみが表示されます。他のプロバイダーと同様に、Team Explorer の AWS CodeCommit はいつでも 1 つの AWS 認証情報プロファイルにのみバインドできます。アカウントを切り替えるには、[Sign out (サインアウト)] を使って接続を削除し、別のアカウントを使って新しい接続を開始します。

接続を確立できたら、[Create] リンクをクリックして、リポジトリを作成することができます。

リポジトリの作成

[Create] (作成) リンクをクリックすると、[Create a New AWS CodeCommit Repository] (新しいリポジトリの作成) ダイアログボックスが開きます。



AWS CodeCommit リポジトリはリージョンごとに整理されており、リポジトリをホストするリージョンを [Region] で選択できます。リストには、AWS CodeCommit をサポートしているすべてのリージョンが表示されます。新しいリポジトリの名前 (必須) と説明 (オプション) を入力します。

ダイアログボックスのデフォルトの動作では、新しいリポジトリのフォルダの場所にリポジトリの名前のサフィックスを追加します (名前を入力すると、フォルダの場所も更新されます)。別のフォルダ名を使用するには、リポジトリの名前の入力の終了後に [Clone into] のフォルダのパスを編集します。

リポジトリに .gitignore の初期ファイルを自動的に作成することを選ぶこともできます。AWS Toolkit for Visual Studio は、Visual Studio のファイルの種類の組み込みのデフォルトを提供します。ファイルを使用しないか、または既存のカスタムのファイルの使用を選択して複数のリポジトリにわたって再利用することもできます。それには、リストで [Use custom (カスタムを使用)] を選択し、使用するカスタムファイルに移動します。

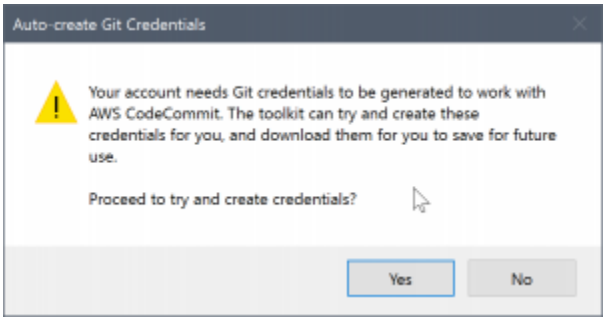
リポジトリ名と場所が用意できたら、[OK] をクリックして、リポジトリの作成を開始します。Toolkit for Visual Studio によりリポジトリの作成がサービスにリクエストされ、新しいリポジトリのクローンがローカルで作成されます。.gitignore ファイルを使用している場合には、初期のコミットが追加されます。この時点で、Git リモートを使用できるようになります。Toolkit for Visual Studio では、前に説明した Git 認証情報へのアクセスが必要となります。

Git 認証情報を設定する

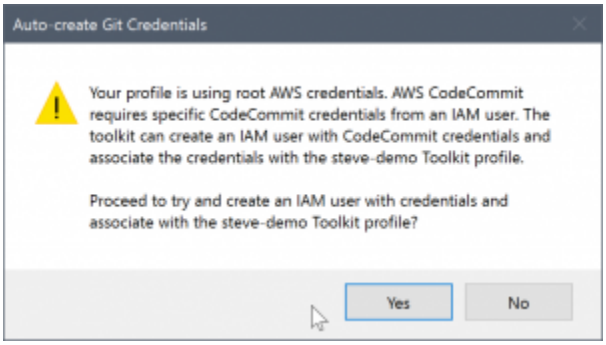
ここまでは、AWS のアクセスキーとシークレットキーを使って、リポジトリの作成をサービスにリクエストしました。ここで Git 自体を使用して実際のクローンオペレーションを実行する必要がありますが、Git は AWS のアクセスキーとシークレットキーを認識しません。代わりに、Git にユーザー名とパスワードの認証情報を指定する必要があります。Git はリモートでこれを HTTPS 接続に使用します。

「Git 認証情報を設定する」に記載されているように、使用する Git 認証情報は IAM ユーザーに関連付けられている必要があります。ルート認証情報に生成することはできません。AWS 認証情報プロファイルは常に、ルートキーでなく、IAM ユーザーのアクセスキーとシークレットキーを含むように設定する必要があります。Toolkit for Visual Studio は AWS CodeCommit 用の Git 認証情報の設定を行い、それを以前に Team Explorer で接続に使用した AWS 認証情報プロファイルに関連付けることができます。

[新しい AWS CodeCommit リポジトリの作成] ダイアログボックスで [OK] を選択し、リポジトリが正常に作成された場合、Toolkit for Visual Studio は Team Explorer で接続された AWS 認証情報プロファイルをチェックして、AWS CodeCommit 用の Git 認証情報が存在し、ローカルでプロファイルに関連付けられているかどうかを確認します。確認された場合、Toolkit for Visual Studio は Team Explorer に新しいリポジトリでクローンオペレーションを開始するように指示します。Git 認証情報がローカルで使用できない場合、Toolkit for Visual Studio は Team Explorer の接続で使用されたアカウントの認証情報の種類を確認します。認証情報が推奨されたように IAM ユーザーのものである場合、次のメッセージが表示されます。

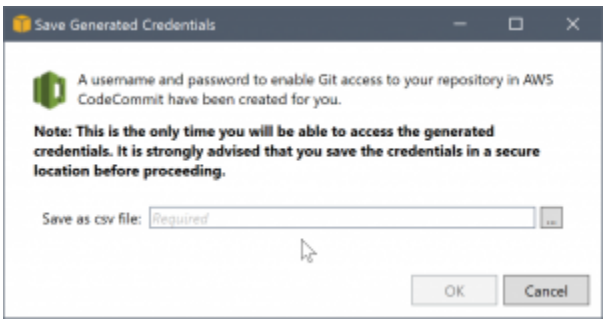


認証情報がルート認証情報の場合には、次のメッセージが表示されます。



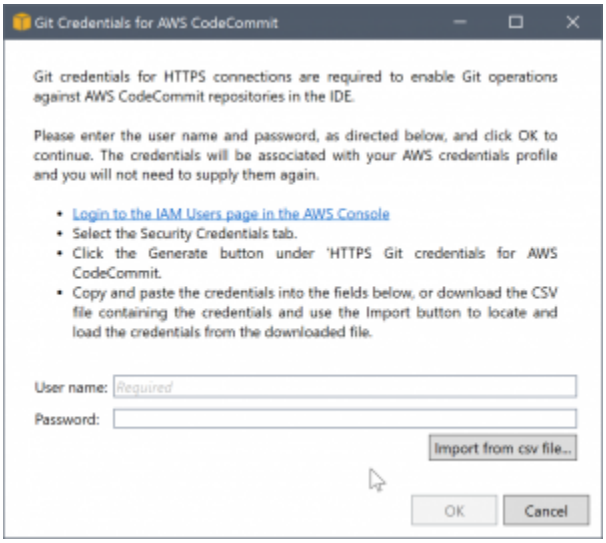
どちらの場合も、Toolkit for Visual Studio は必要な Git 認証情報を作成しようとします。最初のシナリオでは、作成する必要があるのは、IAM ユーザー用の Git 認証情報です。ルートアカウントが使用されている場合、Toolkit for Visual Studio はまず IAM ユーザーの作成を試みてから新しいユーザーの Git 認証情報の作成に進みます。Toolkit for Visual Studio が新しいユーザーを作成する必要がある場合、AWS CodeCommit パワーユーザーマネージドポリシーをその新しいユーザーアカウントに適用します。このポリシーでは、アクセスを AWS CodeCommit のみに許可します。リポジトリの削除以外のすべてのオペレーションを AWS CodeCommit で実行できるようにします。

認証情報を作成する際には、それを 1 回のみ表示できます。そのため、Toolkit for Visual Studio は続行する前に新たに作成した認証情報を .csv ファイルとして保存するように求めるメッセージを表示します。



必ず安全な場所に保存しておくことを、強くお勧めします。

Toolkit for Visual Studio が自動的に認証情報を作成できない場合があります。例えば、AWS CodeCommit で Git 認証情報セットの最大数 (2) を既に作成している場合、または Toolkit for Visual Studio がプログラムによって認証情報を作成するために十分な権限がない場合 (IAM ユーザーとしてサインインしている場合) などがあります。このような場合は、AWS マネジメントコンソール にログインして認証情報を管理するか、または管理者から認証情報を取得します。Toolkit for Visual Studio で表示される [Git Credentials for AWS CodeCommit] (Git 認証情報) ダイアログボックスでそれを入力できます。

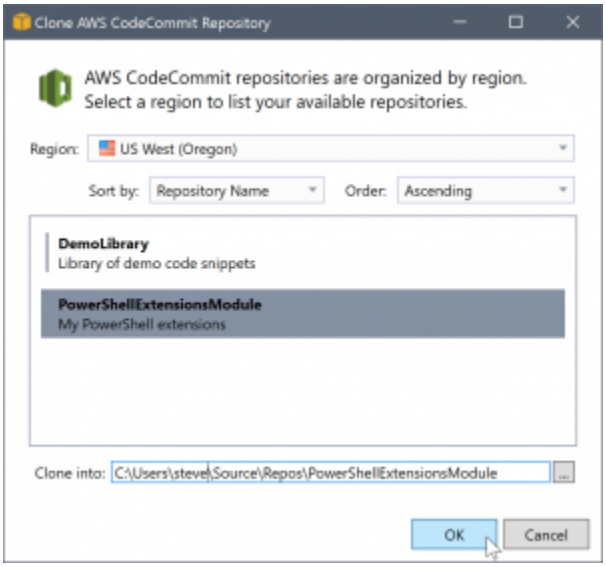


Git 認証情報が利用可能になったので、新しいリポジトリのクローンオペレーションが進行します (Team Explorer 内の操作の進行状態の表示を確認してください)。デフォルトの `.gitignore` ファイルの適用を選択した場合、「Initial Commit」のコメントでリポジトリにコミットされます。

以上で Team Explorer 内で認証情報を設定し、リポジトリを作成できました。必要な認証情報が設定されたので、今後新しいリポジトリを作成する場合には、[Create a New AWS CodeCommit Repository] (新しいリポジトリの作成) ダイアログボックス自体のみが表示されます。

リポジトリのクローンを作成する

既存のリポジトリのクローンを作成するには、Team Explorer の AWS CodeCommit の接続パネルに戻ります。[Clone] (クローン) リンクをクリックして、[Clone AWS CodeCommit Repository] (リポジトリのクローン) ダイアログボックスを開き、クローンを作成するリポジトリと、配置するディスク上の場所を選択します。



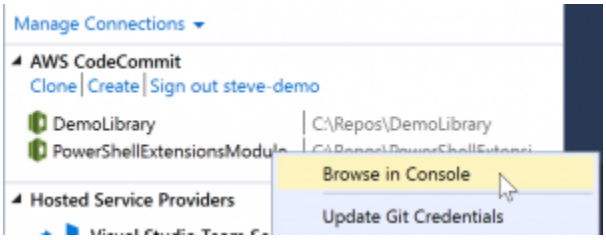
リージョンを選択すると、Toolkit for Visual Studio はサービスのクエリを行って、リージョンで使用できるリポジトリを検出し、それをダイアログボックスの中央のリストに表示します。各リポジトリの名前と説明 (オプション) も表示されます。リポジトリ名または最終更新日で、昇順または降順に、リストの順序をソートすることができます。

リポジトリを選択すると、クローンを作成する場所を選択できます。Team Explorer の他のプラグインで使用されたものと同じ場所がデフォルトとなりますが、他の場所を参照するか、または入力することもできます。デフォルトでは、リポジトリ名が、選択されたパスにサフィックスとして追加されます。ただし、特定のパスを指定する場合には、フォルダを選択した後にテキストボックスを編集します。[OK] をクリックしたときにボックスに含まれるテキストを使ったフォルダに、リポジトリのクローンが作成されます。

リポジトリとフォルダの場所を選択したら、[OK] をクリックして、クローンオペレーションを続行します。リポジトリの作成と同様に、クローンオペレーションの進行状況も Team Explorer に表示されます。

リポジトリを操作する

リポジトリのクローンまたはリポジトリを作成する場合、接続のためのローカルリポジトリは、Team Explorer のオペレーションリンクの下の接続パネルに一覧表示されます。これらのエントリを使うと、リポジトリにアクセスしてコンテンツを参照するのに便利です。リポジトリを右クリックして、[Browse in Console] (コンソールで参照) を選択します。



[Update Git Credentials] (Git 認証情報の更新) を使用して、認証情報プロファイルに関連付けられて保存された Git 認証情報を更新することもできます。これは、認証情報をローテーションする場合に役立ちます。このコマンドを使用すると、[Git Credentials for AWS CodeCommit] (Git 認証情報) ダイアログボックスが開き、新しい認証情報を入力またはインポートできます。

リポジトリ上での Git オペレーションは、期待どおりに機能します。ローカルコミットを行うことができ、共有する準備ができたら、Team Explorer で [Sync] オプションを使用します。Git 認証情報は既にローカルに保存され、接続された AWS 認証情報プロファイルに関連付けられているため、AWS CodeCommit リモートに対するオペレーションのために再入力を求められることはありません。

Visual Studio での CodeArtifact の使用

AWS CodeArtifact は、フルマネージドのアーティファクトリポジトリサービスであり、組織がアプリケーション開発に使用するソフトウェアパッケージを安全に保存および共有できるようにします。CodeArtifact は、NuGet、.NET Core CLI、Visual Studio などの一般的なビルドツールやパッケージマネージャーで使用できます。また、Nuget.org などのパブリックリポジトリからパッケージをプルするように CodeArtifact を設定することもできます。

CodeArtifact では、パッケージはリポジトリに格納され、リポジトリはドメイン内に保存されます。AWS Toolkit for Visual Studio CodeArtifact リポジトリを使用して Visual Studio の構成を簡素化し、CodeArtifact と NuGet.org の両方から Visual Studio のパッケージを簡単に利用できます。

CodeArtifact リポジトリをパッケージソースとして追加します。

CodeArtifact からパッケージを使用するには、Visual Studio でリポジトリをパッケージソースとして NuGet パッケージマネージャーに追加する必要があります。

リポジトリをパッケージソースとして追加するには

1. AWS Explorer で、AWS CodeArtifact ノード内のリポジトリに移動します。
2. 追加したいリポジトリのコンテキスト (右クリック) メニューを開き、[Copy NuGet Source Endpoint] (NuGet ソースエンドポイントのコピー) を選択します。

3. [Tools > Options] (ツール > オプション) メニューで [NuGet Package Manager] ノードの下にある [Package Sources] (パッケージソース) 移動します。
4. [Package Sources] (パッケージソース) で、プラス記号 (+) を選択し、名前を編集し、以前にコピーした NuGet ソースエンドポイントの URL を [Source] (ソース) フィールドに貼り付けます。
5. 新しく追加したパッケージソースの横にあるチェックボックスをオンにします。

> **Note**
>
> Nuget.org への外部接続を CodeArtifact に追加し、Visual Studio で nuget.org パッケージソースを無効にすることをお勧めします。外部接続の使用時には、Nuget.org からプルされるすべての依存関係は CodeArtifact に保存されます。何らかの理由で Nuget.org が停止しても、必要なパッケージは利用可能なままです。外部接続の詳細については、「AWS CodeArtifact ユーザーガイド」の「外部接続を追加する」を参照してください。

6. [OK] をクリックしてメニューを閉じます。

Visual Studio で CodeArtifact を使用する方法の詳細については、「AWS CodeArtifact ユーザーガイド」の「Visual Studio で CodeArtifact を使用する」を参照してください。

AWS Explorer で Amazon RDS を使用

Amazon Relational Database Service (Amazon RDS) は、クラウドでの SQL リレーショナルデータベースシステムのプロビジョニングおよび管理を可能にするサービスです。Amazon RDS は、3 種類のデータベースシステムをサポートします。

- MySQL Community Edition
- Oracle Database エンタープライズエディション
- Microsoft SQL Server (Express Edition、Standard Edition、または Web Edition)

詳細については、Amazon RDS ユーザーガイドを参照してください。

ここで説明されている機能の多くは、Amazon RDS の AWS マネジメントコンソールでも利用できます。

トピック

- Amazon RDS データベースインスタンスの起動

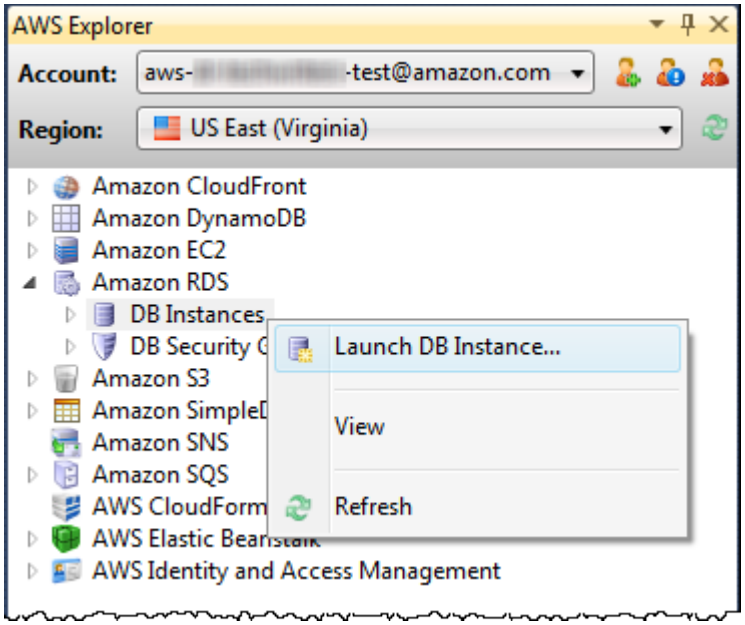
- RDS インスタンスでの Microsoft SQL Server データベースの作成
- Amazon RDS セキュリティグループ

Amazon RDS データベースインスタンスの起動

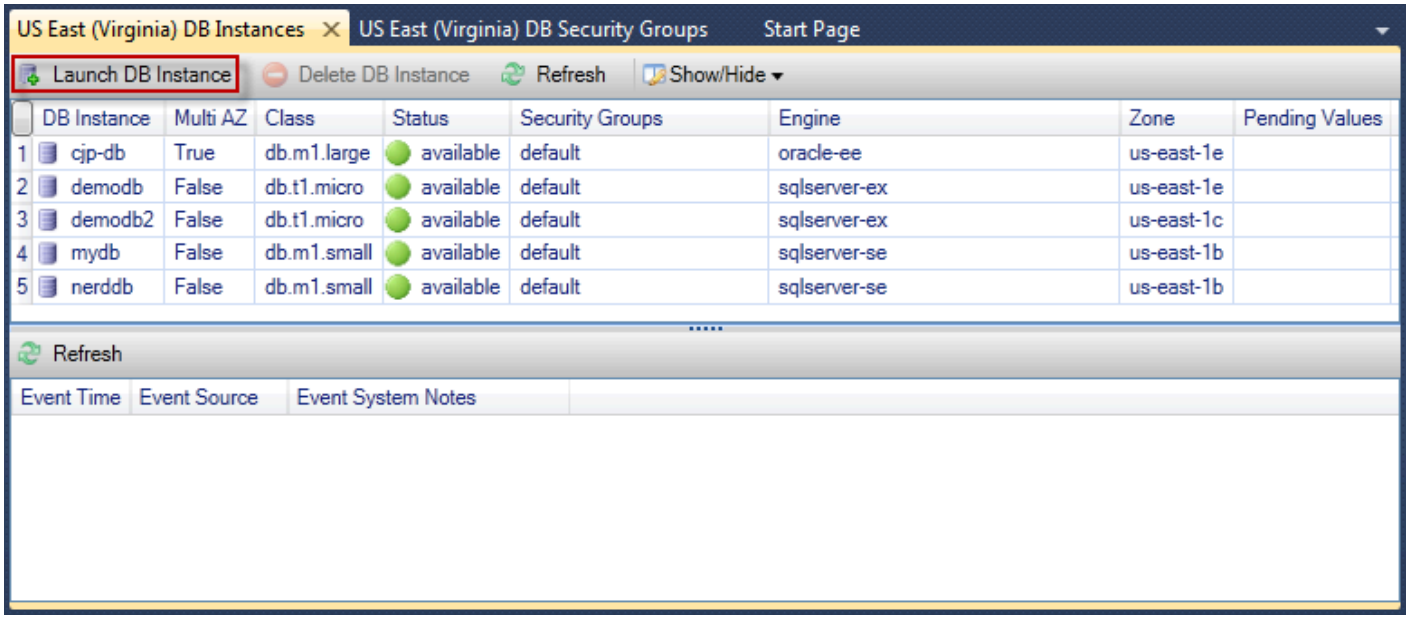
AWS Explorer では、Amazon RDS がサポートする任意のデータベースエンジンのインスタンスを起動できます。次のウォークスルーでは、Microsoft SQL Server Standard Edition のインスタンスを起動するためのユーザーエクスペリエンスについて記していますが、ユーザーエクスペリエンスはサポートするすべてのエンジンで類似しています。

Amazon RDS インスタンスを起動するには

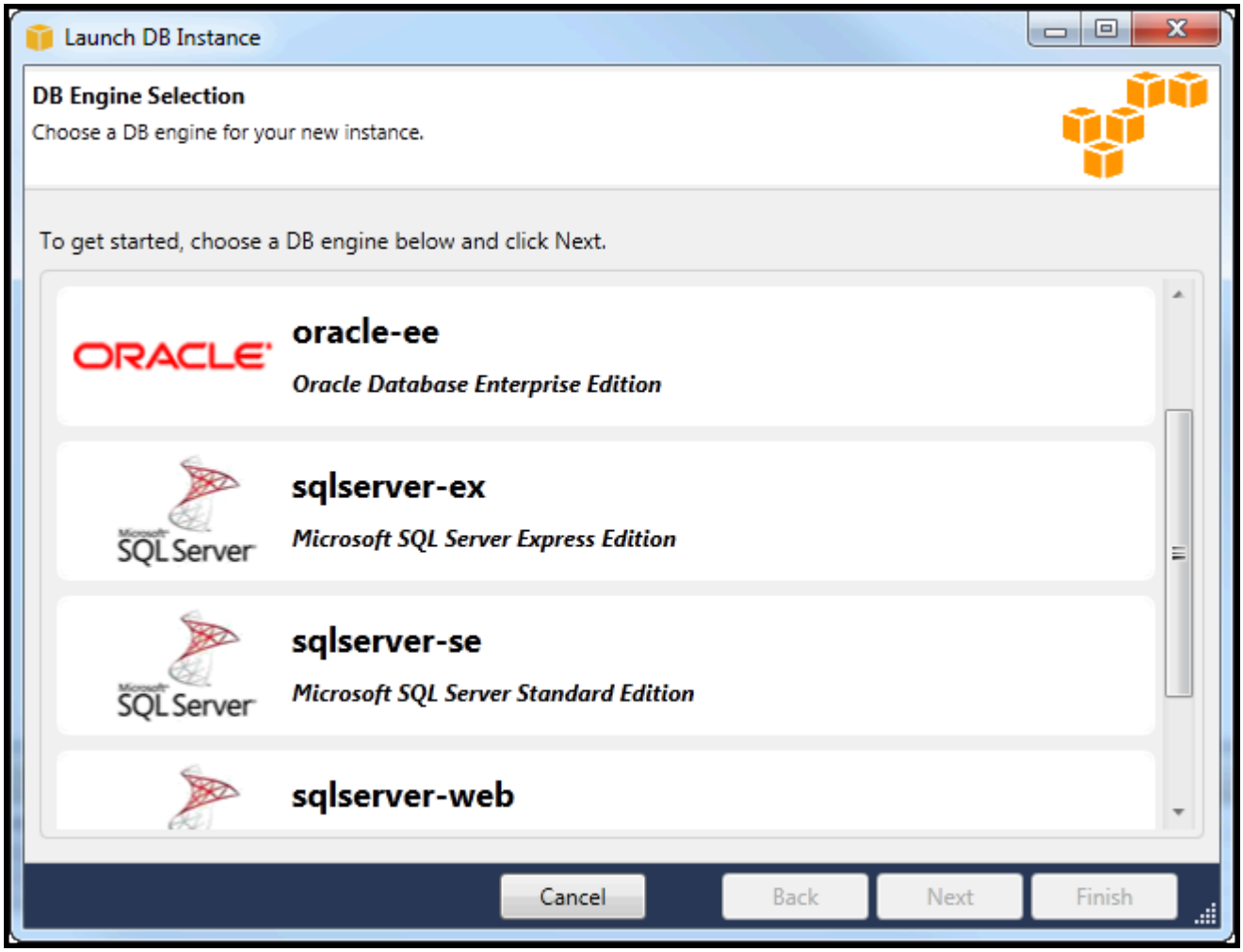
1. AWS Explorer で Amazon RDS ノードのコンテキスト (右クリック) メニューを開いて、[Launch DB Instance] (DB インスタンスの起動) を選択します。



   または、[DB Instances (DB インスタンス)] タブで、[Launch DB Instance (DB インスタンスの起動)] を選択します。



2. [DB Engine Selection (DB エンジンの選択)] ダイアログボックスで、起動するデータベースエンジンのタイプを選択します。このウォークスルーでは、Microsoft SQL Server Standard Edition (sqlserver-se) を選択し、[Next (次へ)] を選択します。



3. [DB Engine Instance Options (DB エンジンインスタンスオプション)] ダイアログボックスで、設定オプションを選択します。

   [DB Engine Instance Options and Class (DB エンジンインスタンスオプションとクラス)] セクションで、次の設定を指定できます。

   ライセンスモデル

エンジンのタイプ	ライセンス
Microsoft SQL Server	license-included
MySql	general-public-license
Oracle	自分のライセンス使用

データベースエンジンによってライセンスモデルは異なります。エンジンのタイプ、ライセンス、Microsoft SQL Server、license-included、MySql、general-public-license、Oracle、bring-your-own-license

DB インスタンスのバージョン

使用するデータベースエンジンのバージョンを選択します。サポートするバージョンが 1 つのみの場合は、そのバージョンが選択されています。

DB インスタンスのクラス:

データベースエンジンのインスタンスクラスを選択します。インスタンスクラスの料金はさまざまです。詳細については、Amazon RDS 料金表を参照してください。

マルチ AZ 配置の実行

このオプションを選択し、データの冗長性と可用性を強化するマルチ AZ 配置を作成します。Amazon RDS は、計画されたシステム停止または予期しない停止の際の自動フェイルオーバーのために、異なるアベイラビリティーゾーンにあるデータベースのスタンバイコピーのプロビジョニングおよび維持を行います。マルチ AZ 配置の料金の詳細については、Amazon RDS の詳細ページの料金表セクションを参照してください。このオプションは、Microsoft SQL Server ではサポートされていません。

マイナーバージョンの自動アップグレード

このオプションを選択すると、RDS インスタンスのマイナーバージョンアップを AWS が自動で実行します。

[RDS Database Instance (RDS データーベースインスタンス)] セクションで、以下の設定を指定します。

ストレージ割り当て

エンジン	最小 (GB)	最大 (GB)
MySQL	5	1024
Oracle Enterprise Edition	10	1024

エンジン	最小 (GB)	最大 (GB)
Microsoft SQL Server Express Edition	30	1024
Microsoft SQL Server Standard Edition	250	1024
Microsoft SQL Server Web Edition	30	1024

割り当てられたストレージの最小値および最大値はデータベースエンジンのタイプによって異なります。Engine Minimum (GB)、Maximum (GB)、MySQL、5、1024、Oracle Enterprise Edition、10、1024、Microsoft SQL Server Express Edition、30、1024、Microsoft SQL Server Standard Edition、250、1024、Microsoft SQL Server Web Edition、30、1024

DB Instance Identifier

データベースインスタンスの名前を指定します。この値は大文字と小文字が区別されません。AWS Explorer では小文字で表示されます。

マスターユーザー名

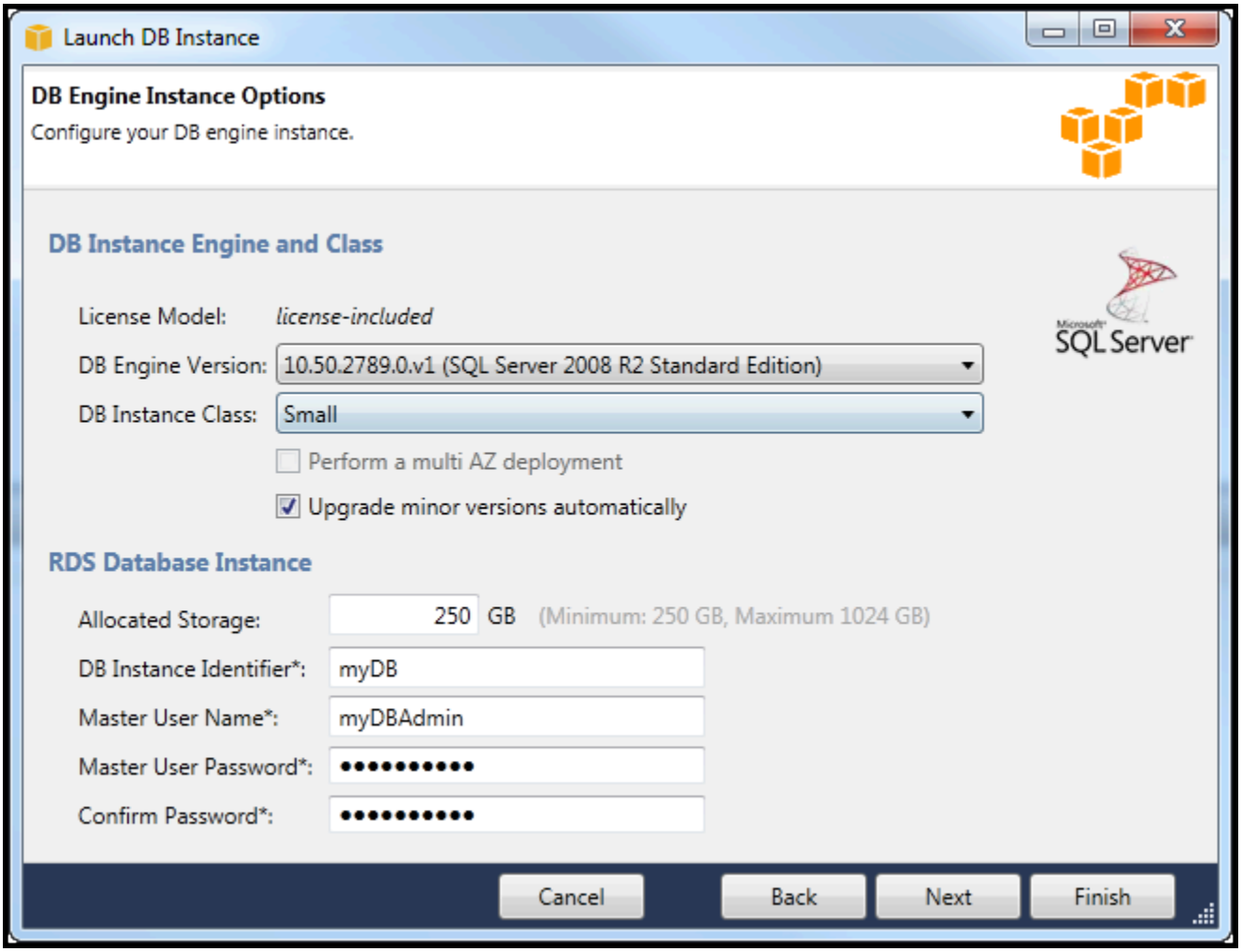
データベースインスタンスの管理者名を入力します。

マスターユーザのパスワード

データベースインスタンスの管理者パスワードを入力します。

Confirm Password] (パスワードの確認

パスワードをもう一度入力して誤りがないことを確認します。



1. [Additional Options (追加オプション)] ダイアログボックスで、以下の設定を指定します。

   Database Port

   これは、ネットワークで通信を行う際にインスタンスが使用する TCP ポートです。コンピュータがファイアウォールを介してインターネットにアクセスしている場合は、ファイアウォールが許可するポートを設定します。

   アベイラビリティーゾーン

   リージョン内の特定のアベイラビリティーゾーンでインスタンスを起動する場合は、このオプションを使用します。指定したデータベースインスタンスがすべてのアベイラビリティーゾーンで使用できない場合があります。

   RDS Security Group

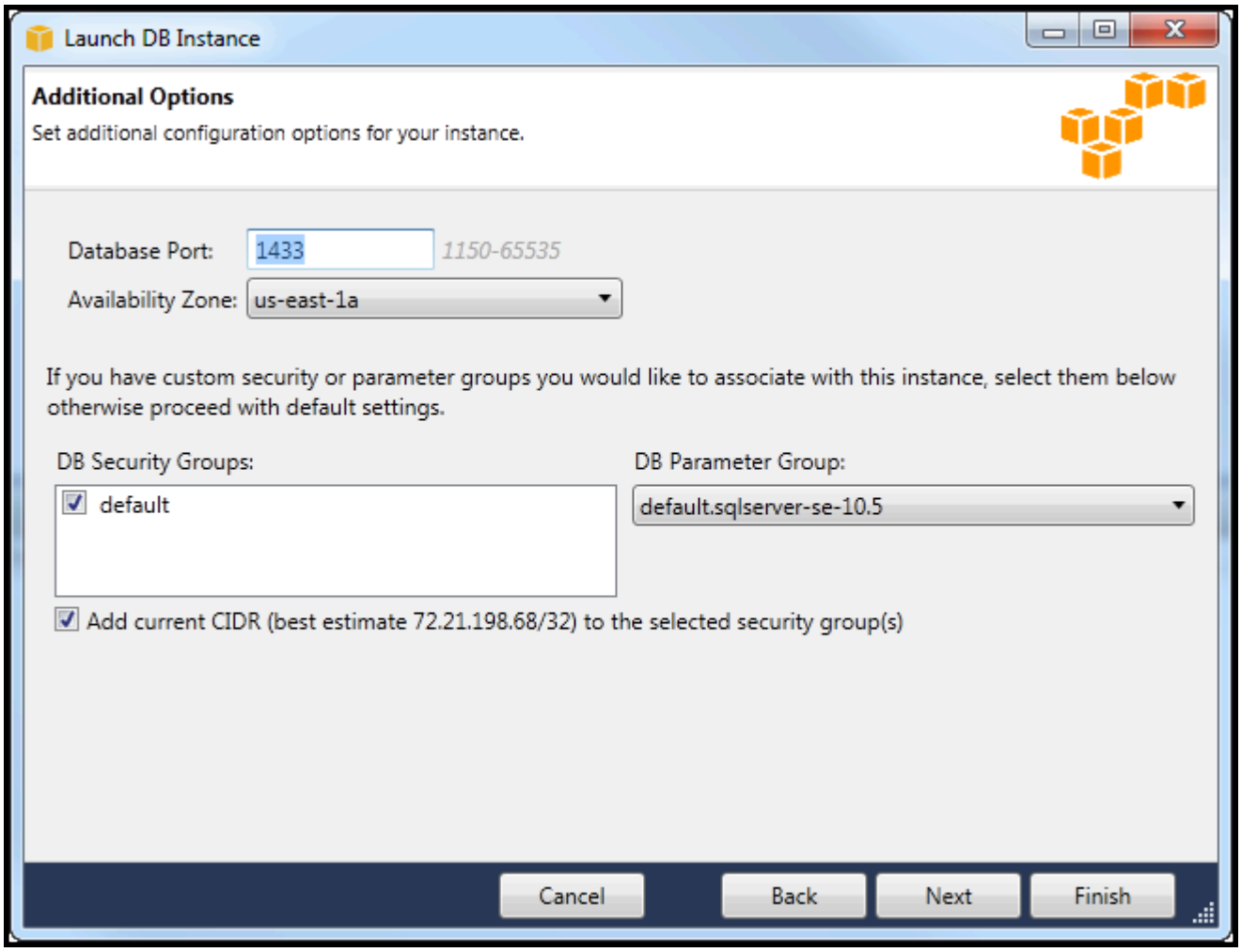
   インスタンスに関連付ける RDS セキュリティグループを 1 つ以上選択します。インスタンスでアクセスできる IP アドレス、Amazon EC2 インスタンス、および AWS アカウント

は、RDS セキュリティグループで指定されます。RDS セキュリティグループの詳細については、Amazon RDS セキュリティグループを参照してください。The Toolkit for Visual Studioはユーザーの現在の IP アドレスを特定しようと試み、このアドレスをインスタンスに関連付けられたセキュリティグループに追加するオプションを提供します。ただし、コンピュータがファイアウォール経由でインターネットにアクセスしている場合、Toolkit により生成されたコンピュータの IP アドレスが正しくない可能性があります。使用する IP アドレスを決定するには、システム管理者にお問い合わせください。

DB パラメータグループ

(オプション) このドロップダウンリストで、インスタンスに関連付ける DB パラメータグループを選択します。DB パラメータグループを使用することで、インスタンスのデフォルト設定を変更できます。詳細については、Amazon Relational Database Service ユーザーガイドおよびこちらの記事を参照してください。

このダイアログボックスで設定を指定している場合は、[Next (次へ)] を選択します。

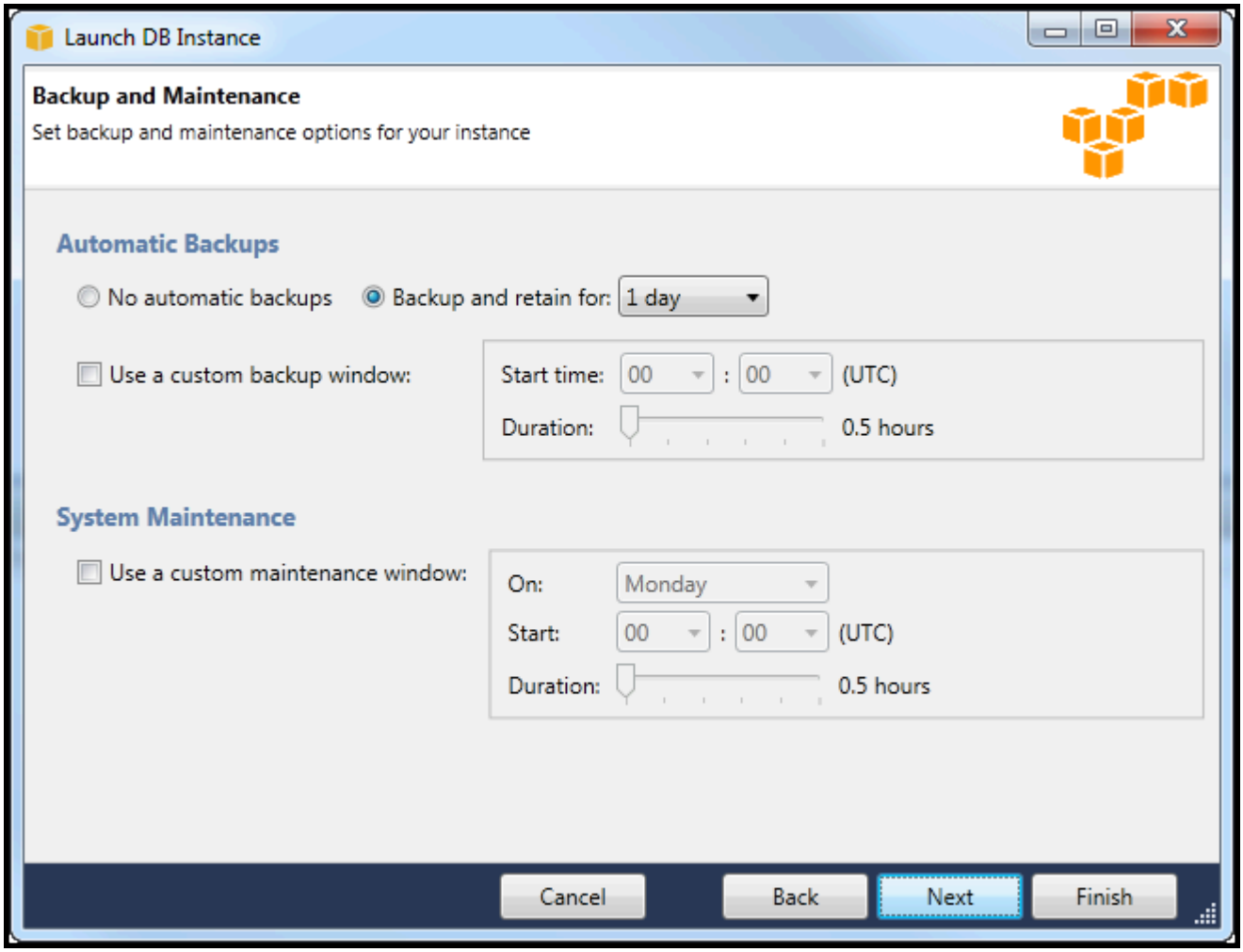


2. [Backup and Maintenance] (バックアップとメンテナンス) ダイアログボックスでは、 Amazon RDSがインスタンスをバックアップするかどうか、そしてもしバックアップするのであれば、その保持期間を設定できます。バックアップを行う時間帯も指定できます。

   このダイアログボックスでは、 Amazon RDSでインスタンスのシステムメンテナンスを行うかも指定できます。メンテナンスには、定期的なパッチとマイナーバージョンアップグレードが含まれます。

   システムメンテナンスに指定した時間帯とバックアップに指定した時間帯は重複できません。

   [次へ] を選択します。



3. ウィザードの最後のダイアログボックスでは、インスタンスの設定を確認することができます。設定を変更する必要がある場合は、[Back (戻る)] ボタンをクリックします。すべての設定が正しい場合は、[Launch (起動)] を選択します。

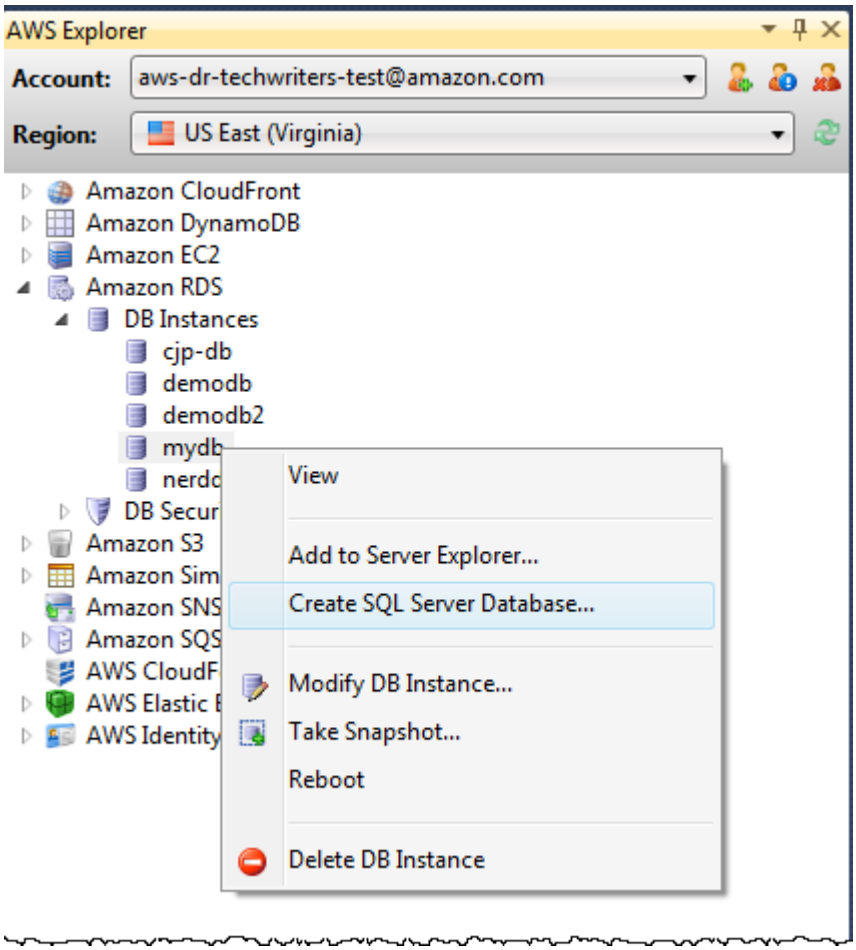
RDS インスタンスでの Microsoft SQL Server データベースの作成

Microsoft SQL Server は、Amazon RDS インスタンスの起動後に、RDS インスタンスに SQL Server データベースを作成する必要があるように設計されています。

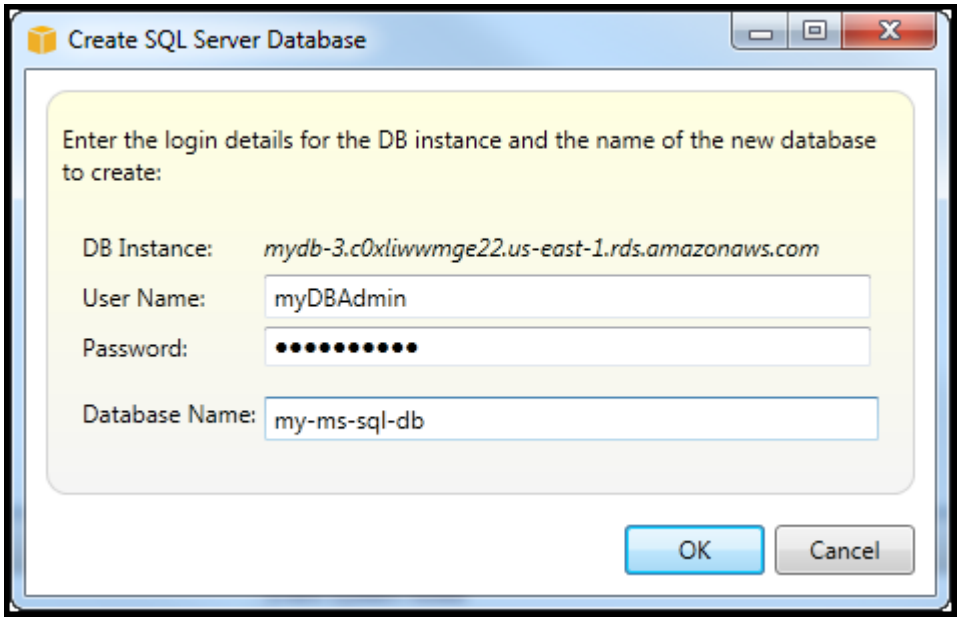
Amazon RDS インスタンスを作成する方法については、「Amazon RDS データベースインスタンスの起動」を参照してください。

Microsoft SQL Server データベースを作成するには

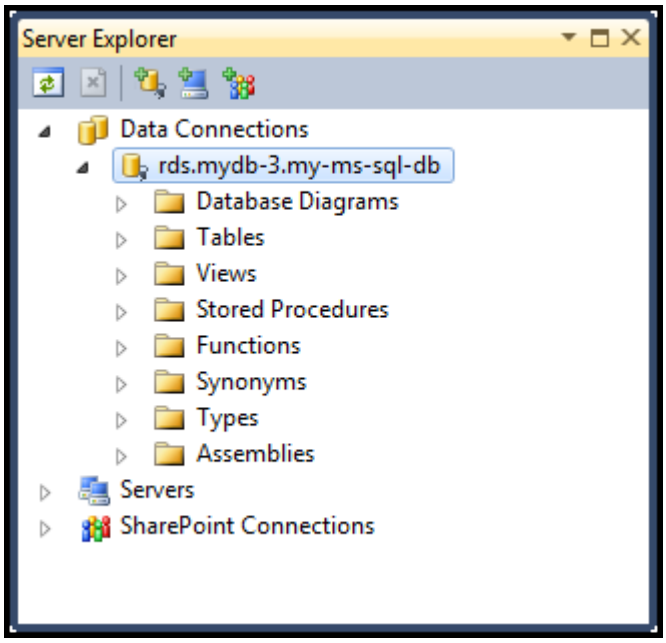
1. AWS Explorer で、Microsoft SQL Server の RDS インスタンスに対応するノードのコンテキスト (右クリック) メニューを開き、[Create SQL Server Database] (SQL サーバーデーターベースの作成) を選択します。



2. [Create SQL Server Database (SQL サーバーデーターベースの作成)] ダイアログボックスで、RDS インスタンスの作成時に指定したパスワードを入力し、Microsoft SQL Server データベースの名前を入力して、[OK] を選択します。



3. The Toolkit for Visual Studio は、Microsoft SQL Server データベースを作成して、Visual Studio の Server Explorer に追加します。



Amazon RDS セキュリティグループ

Amazon RDS セキュリティグループを使用すると、Amazon RDS インスタンスへのネットワークアクセスを管理できます。セキュリティグループを使用して、CIDR 表記を使って IP アドレスのセットを指定すると、Amazon RDS インスタンスは、これらのアドレスから送信されるネットワークトラフィックのみを認識します。

Amazon RDS セキュリティグループは、Amazon EC2 セキュリティグループと類似の動作をしますが、両者は異なります。RDS セキュリティグループに EC2 セキュリティグループを追加することができます。すると、EC2 セキュリティグループのメンバーである EC2 インスタンスは、RDS セキュリティグループのメンバーである RDS インスタンスにアクセスできます。

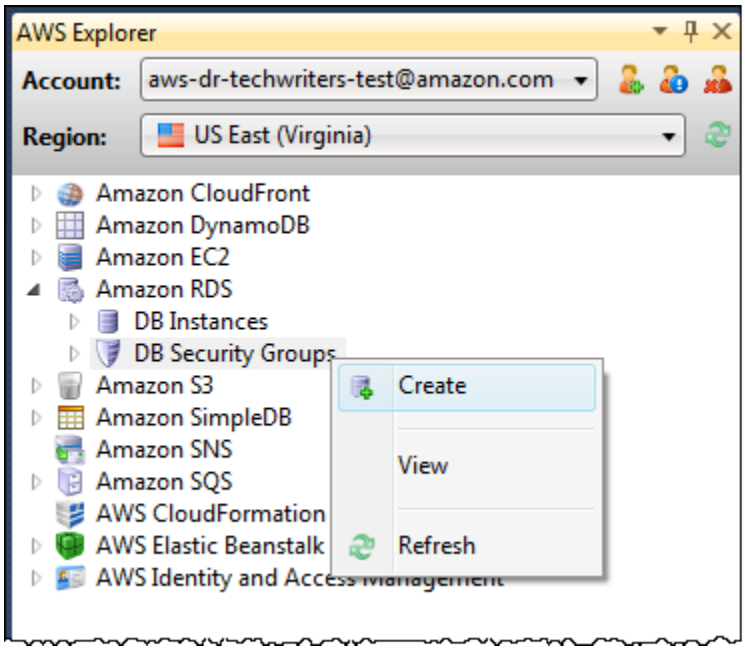
Amazon RDS セキュリティグループの詳細については、「RDS セキュリティグループ」を参照してください。Amazon EC2 セキュリティグループの詳細については、EC2 ユーザーガイドを参照してください。

Amazon RDS セキュリティグループを作成する

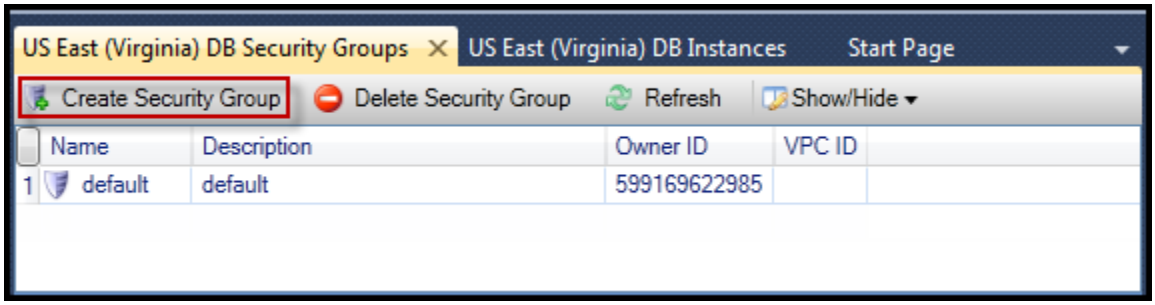
The Toolkit for Visual Studio を使用して、RDS セキュリティグループを作成できます。AWS Toolkit を使用して RDS インスタンスを起動する場合は、ウィザードを使ってインスタンスで使用する RDS セキュリティグループを指定することができます。ウィザードを開始する前に、次の手順を使用してそのセキュリティグループを作成することができます。

Amazon RDS セキュリティグループを作成するには

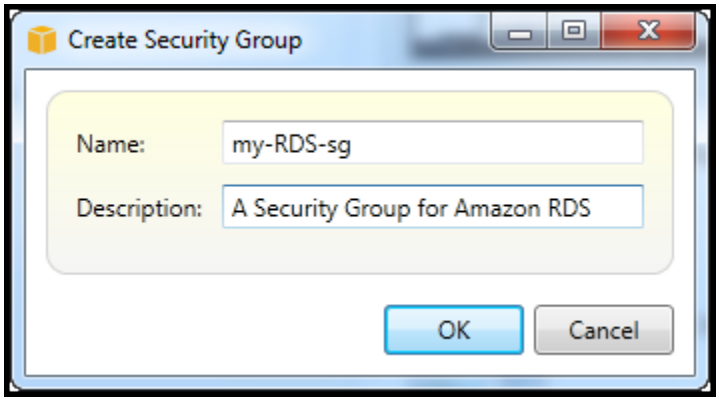
1. AWS Explorer で Amazon RDS ノードを展開し、[DB Security Groups] (DB セキュリティグループ) サブノードのコンテキスト (右クリック) メニューを開いて、[Create] (作成) を選択します。



   または、[Security Groups (セキュリティグループ)] タブで、[Create Security Group (セキュリティグループの作成)] をクリックします。このタブが表示されていない場合は、[DB Security Groups (DB セキュリティグループ)] サブノードのコンテキスト (右クリック) メニューを開き、[View (表示)] を選択します。



2. [Create Security Group (セキュリティグループの作成)] ダイアログボックスで、セキュリティグループ名と説明を入力し、[OK] を選択します。



Amazon RDS セキュリティグループのアクセス許可の設定

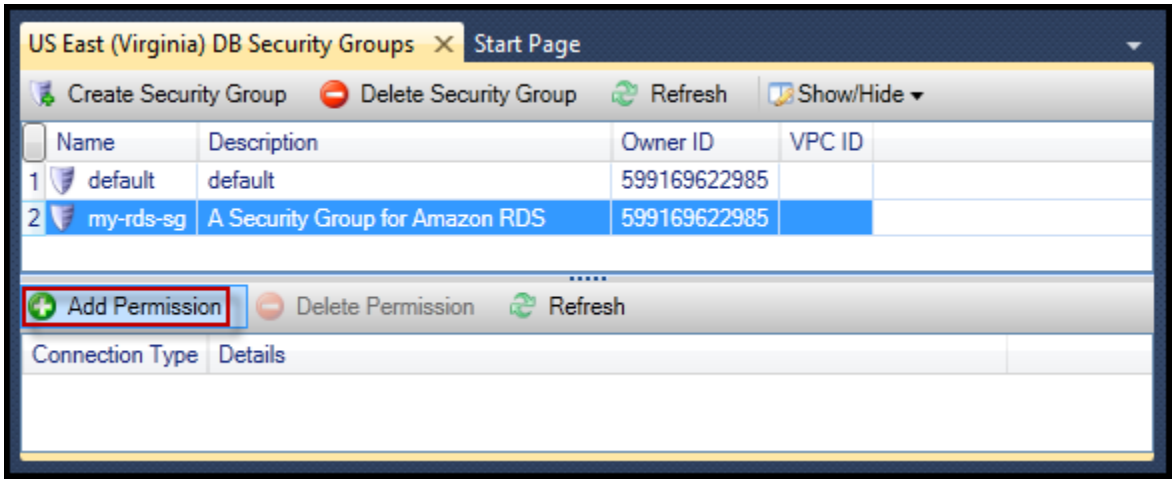
デフォルトでは、新しい Amazon RDS セキュリティグループはネットワークアクセスを提供しません。セキュリティグループを使用する Amazon RDS インスタンスへのアクセスを有効にするには、次の手順を使用してアクセス許可を設定します。

Amazon RDS セキュリティグループへのアクセスを設定するには

1. [Security Groups (セキュリティグループ)] タブで、リストビューからセキュリティグループを選択します。セキュリティグループがリストに表示されない場合は、[Refresh (更新)] を選択します。それでもセキュリティグループがリストに表示されない場合は、適切な AWS リージョンのリストが表示されていることを確認します。AWS Toolkit の [Security Group] (セキュリティグループ) タブはリージョン固有です。

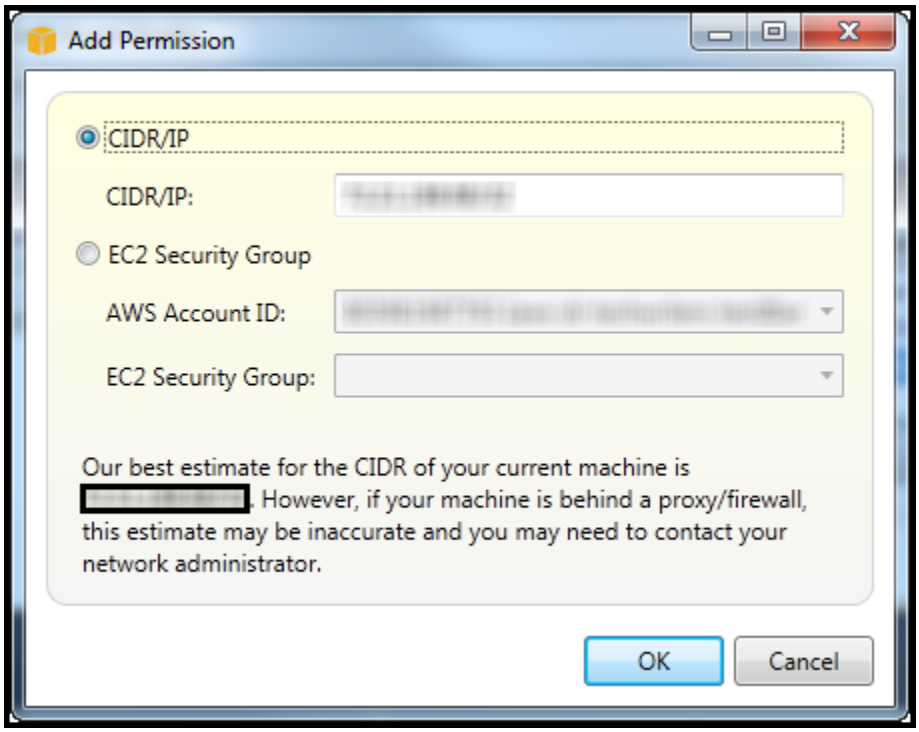
   [Security Group] (セキュリティグループ)) タブが表示されない場合は、AWS Explorer で、[DB Security Groups] (DB セキュリティグループ)] サブノードのコンテキスト (右クリック) メニューを開き、[View] (表示) を選択します。

2. [Add Permission] (アクセス許可の追加) を選択します。



[Security Groups] (セキュリティグループ) タブの [Add Permissions] (アクセス許可の追加) ボタン

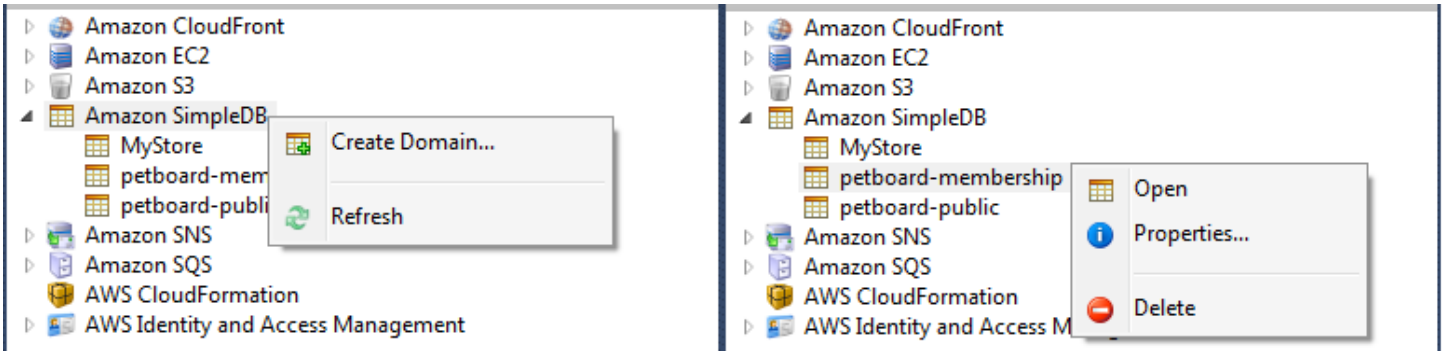
3. [Add Permission (アクセス許可の追加)] ダイアログボックスで、CIDR 表記を使用して、RDS インスタンスにアクセスできる IP アドレスを指定するか、または RDS インスタンスにアクセスできる EC2 セキュリティグループを指定することができます。[EC2 Security Group] (EC2 セキュリティグループ) を選択すると、AWS アカウント に関連付けられたすべての EC2 インスタンスへのアクセスを指定することができます。または、ドロップダウンリストから EC2 セキュリティグループを選択することができます。



AWS Toolkit は、IP アドレスを判別して、適切な CIDR 表記を使ってダイアログボックスを自動入力しようとします。コンピュータがファイアウォール経由でインターネットにアクセスしている場合は、ツールキットにより検出された CIDR が正しくない可能性があります。

AWSエクスプローラから Amazon SimpleDB を使用する

AWS Explorer では、アクティブな AWS アカウントに関連付けられているすべての Amazon SimpleDB ドメインが表示されます。AWS Explorer で、Amazon SimpleDB ドメインを作成または削除することができます。

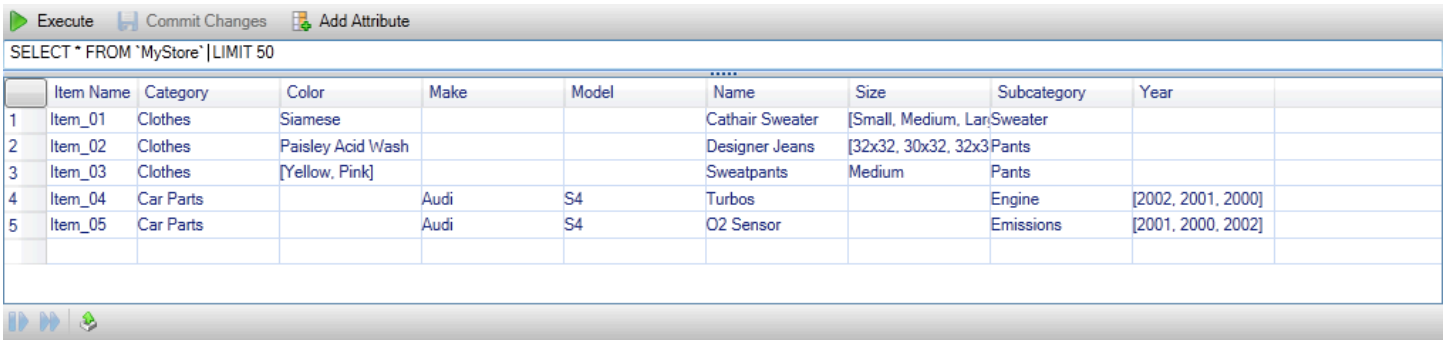


Create, delete, or open Amazon SimpleDB domains associated with your account

クエリの実行と結果の編集

AWS Explorer では、Amazon SimpleDB ドメインのグリッドビューを表示して、ドメインの項目、属性、値を表示することもできます。クエリを実行して、ドメインの項目のサブセットのみを表示することができます。セルをダブルクリックすると、その項目に対応する属性の値を編集できます。ドメインに新しい属性を追加することもできます。

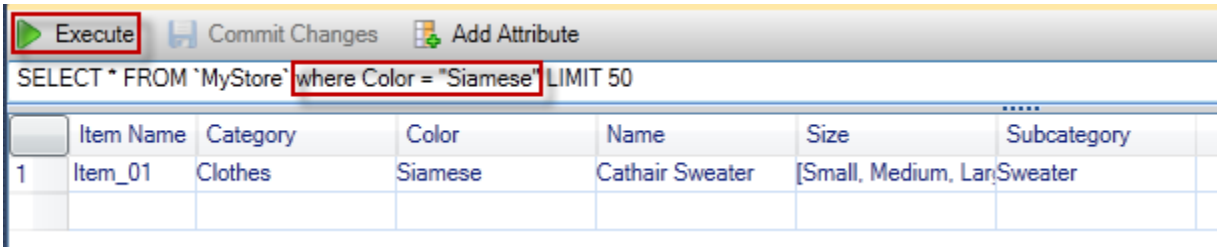
ここに表示されるドメインは、AWS SDK for .NET に付属の Amazon SimpleDB サンプルです。



	Item Name	Category	Color	Make	Model	Name	Size	Subcategory	Year
1	Item_01	Clothes	Siamese			Cathair Sweater	[Small, Medium, Lar	Sweater	
2	Item_02	Clothes	Paisley Acid Wash			Designer Jeans	[32x32, 30x32, 32x3	Pants	
3	Item_03	Clothes	[Yellow, Pink]			Sweatpants	Medium	Pants	
4	Item_04	Car Parts		Audi	S4	Turbos		Engine	[2002, 2001, 2000]
5	Item_05	Car Parts		Audi	S4	O2 Sensor		Emissions	[2001, 2000, 2002]

Amazon SimpleDB grid view

クエリを実行するには、グリッドビューの上部にあるテキストボックスでクエリを編集し、[Execute (実行)] を選択します。このビューは、クエリに一致する項目のみを表示するようにフィルタリングされます。

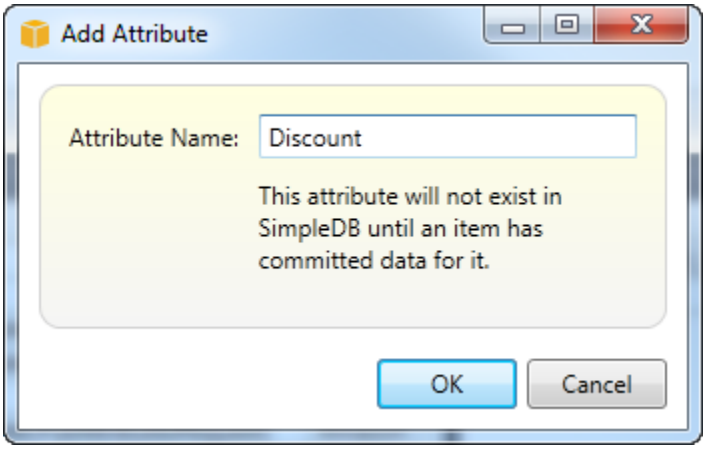


Execute query from AWS Explorer

属性に関連付けられている値を編集するには、対応するセルをダブルクリックし、値を編集して、[Commit Changes (変更をコミット)] を選択します。

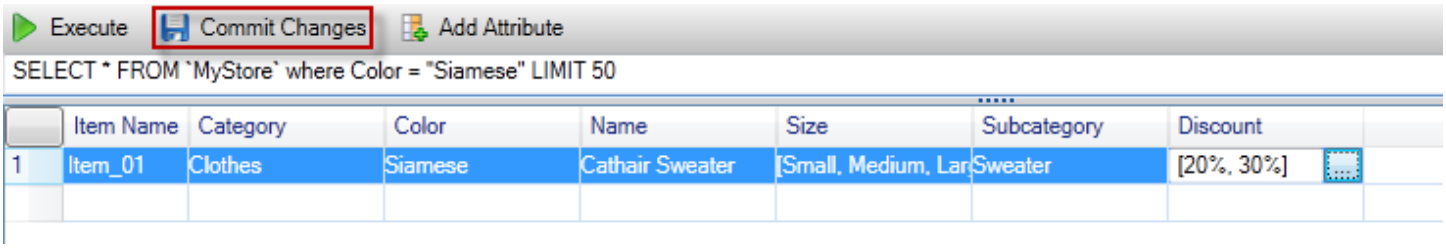
属性の追加

属性を追加するには、ビューの上部で、[Add Attribute (属性の追加)] を選択します。



属性の追加 dialog box

ドメインの属性部分を作成するには、そのための値を少なくとも 1 つの項目に追加して、[Commit Changes (変更をコミット)] ボタンを選択する必要があります。



Commit changes for a new attribute

クエリ結果をページ分割する

ビューの下部には 3 つのボタンがあります。

Paginate and export buttons

最初の 2 つのボタンを使うと、クエリ結果がページ分割されます。最初のボタンでは、結果の次のページが表示されます。2 番目のボタンでは、結果の 10 ページ先が表示されます。この場合では、1 ページは 100 行または LIMIT 値によって指定された行数 (クエリに含まれている場合) となります。

CSV へエクスポート

最後のボタンを使うと、現在の結果が CSV ファイルにエクスポートされます。

AWS Explorer からの Amazon SQS の使用

Amazon Simple Queue Service (Amazon SQS) は柔軟なキューサービスであり、ソフトウェアアプリケーションで実行されている異なるプロセス間でメッセージの受け渡しを行うことができます。Amazon SQS キューは AWS インフラストラクチャ内に配置されていますが、メッセージを受け渡すプロセスはローカル、Amazon EC2 インスタンス、またはそれらを組み合わせて配置できます。Amazon SQS は複数のコンピュータ間での分散作業を調整するために最適です。

Toolkit for Visual Studio を使用すると、アクティブなアカウントに関連付けられている Amazon SQS キューの表示、キューの作成と削除、キューを使ったメッセージの送信を行うことができます。(アクティブなアカウントとは、AWS Explorer で選択されたアカウントを意味します。)

Amazon SQS の詳細については、AWS ドキュメントの「SQS のご紹介」を参照してください。

キューの作成

AWS Explorer から Amazon SQS キューを作成することができます。キューの ARN と URL は、アクティブなアカウントのアカウント番号と、作成時に指定したキューの名前に基づきます。

キューを作成するには

1. AWS Explorer で Amazon SQS ノードのコンテキスト (右クリック) メニューを開いて、[Create Queue] (キューの作成) を選択します。
2. [Create Queue (キューの作成)] ダイアログボックスで、キュー名、デフォルトの可視性タイムアウト、およびデフォルト配信遅延を指定します。デフォルトの可視性タイムアウトとデフォル

トの配信遅延は秒単位で指定します。デフォルトの可視性タイムアウトは、あるプロセスがメッセージを取得した後に、潜在的な受信プロセスに対してそのメッセージが不可視になる時間です。デフォルトの配信遅延は、メッセージが送信されてから、潜在的な受信プロセスに対して可視となるまでの時間です。

3. [OK] を選択してください。新しいキューは、[Amazon SQS] ノードの下のサブノードとして表示されます。

キューの削除

AWS Explorer から既存のキューを削除することができます。キューを削除すると、キューに関連付けられているメッセージは利用できなくなります。

キューを削除するには

1. AWS Explorer で、削除するキューのコンテキスト (右クリック) メニューを開き、[Delete] (削除) を選択します。

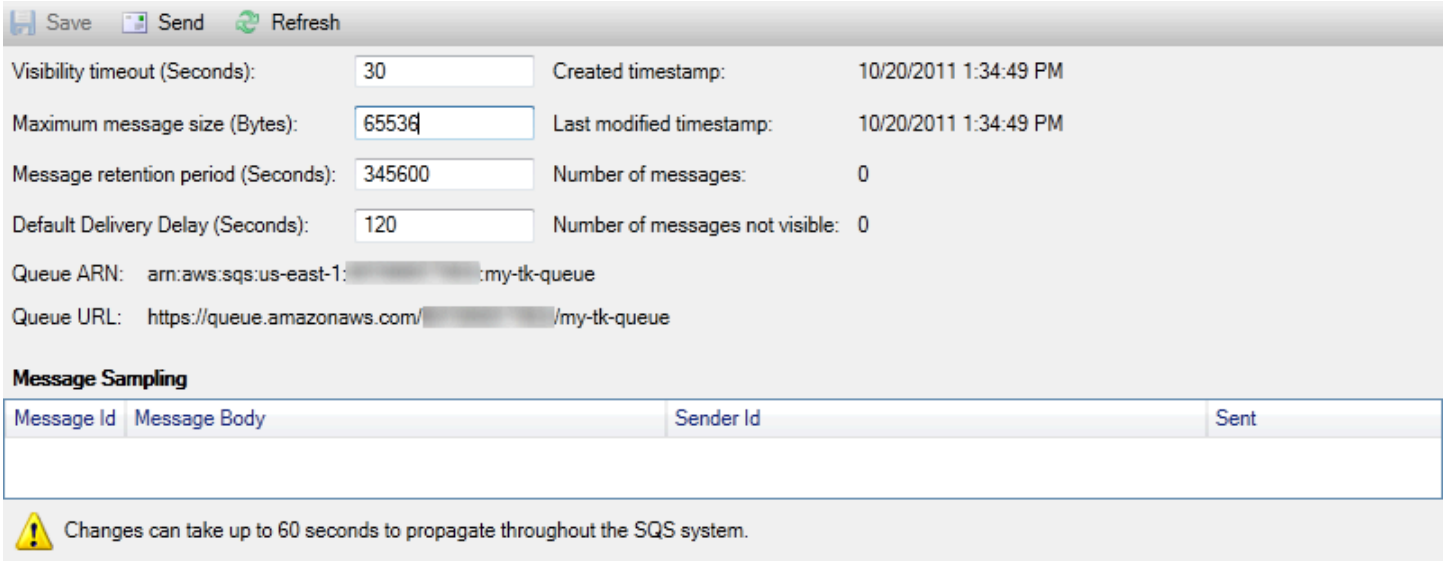
キューのプロパティの管理

AWS Explorer に表示されているすべてのキューのプロパティを表示および編集できます このプロパティビューから、キューにメッセージを送信することもできます。

キューのプロパティを管理するには

- AWS Explorer で、プロパティを管理するキューのコンテキスト (右クリック) メニューを開き、[View Queue] (キューの表示) を選択します。

  キューのプロパティビューから、可視性タイムアウト、最大メッセージサイズ、メッセージの保持期間、デフォルトの配信遅延を編集できます。デフォルトの配信遅延は、メッセージの送信時に上書きできます。次のスクリーンショットでは、キューの ARN と URL に含まれるアカウント番号は非表示にしています。



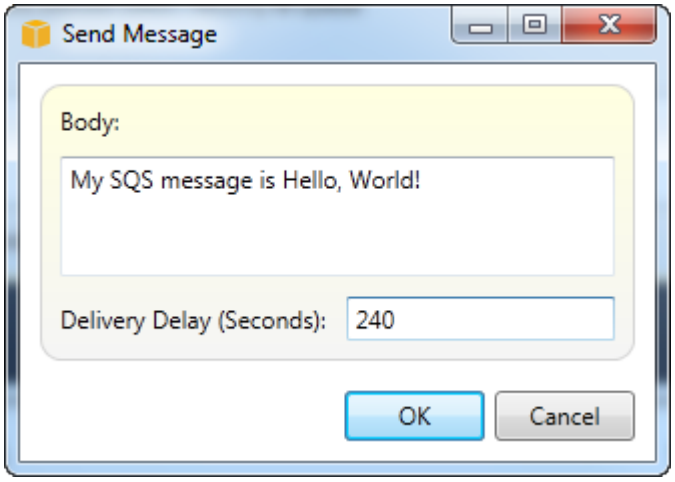
SQS queue properties view

キューへのメッセージ送信

キューのプロパティビューから、キューにメッセージを送信できます。

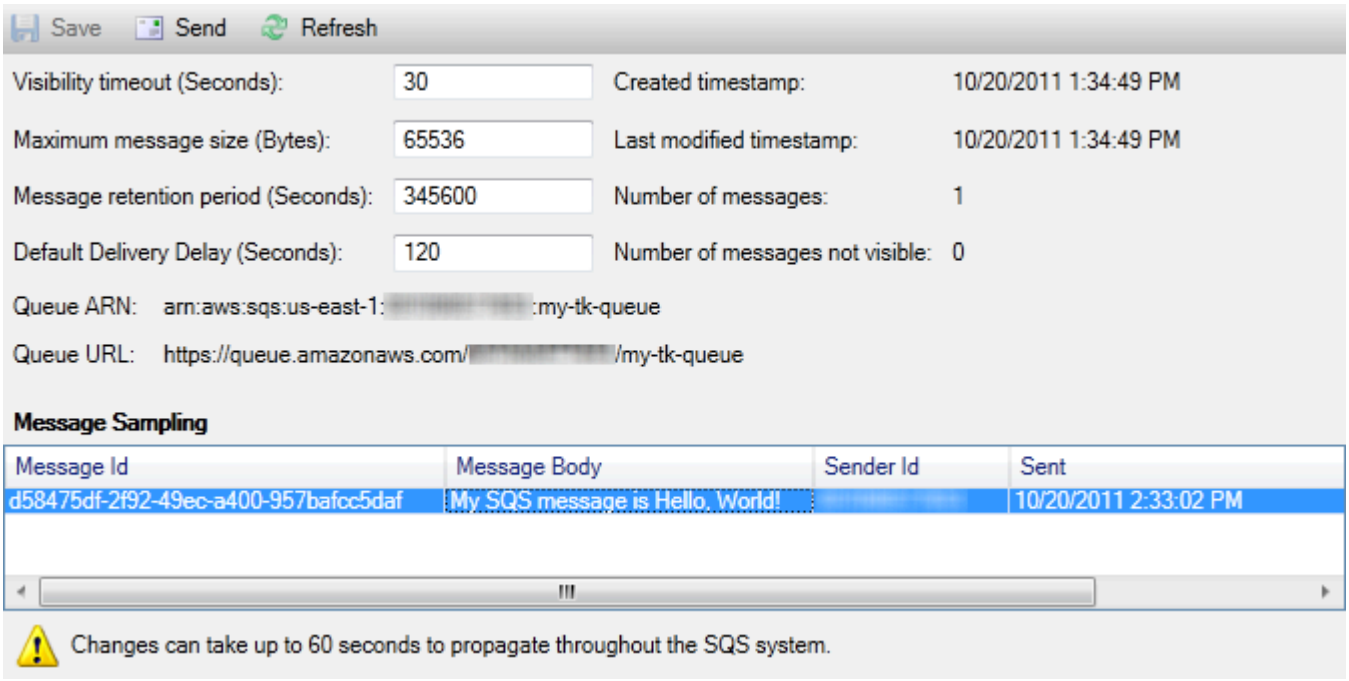
単一のメッセージを送信するには

1. キューのプロパティビューの上部で、[Send (送信)] ボタンを選択します。
2. メッセージを入力します。(オプション) 配信遅延を入力すると、キューのデフォルトの配信遅延を上書きします。次の例では、遅延の値を 240 秒で上書きします。[OK] を選択してください。



メッセージの送信 dialog box

3. 約 240 秒 (4 分) 待ちます。メッセージが、キューのプロパティビューの [Message Sampling (メッセージの送信)] セクションに表示されます。



SQS properties view with sent message

キューのプロパティビューのタイムスタンプは、[Send (送信)] ボタンを選択した時間です。これには遅延は含まれません。したがって、メッセージがキューに表示されてレシーバーに利用可能になる時間は、このタイムスタンプよりも後になる場合があります。タイムスタンプは、コンピュータの現地時間で表示されます。

Identity and Access Management

AWS Identity and Access Management (IAM) を使用すると、 AWS アカウント および リソースへのアクセスをより安全に管理できます。IAM を使用すると、プライマリ (ルート) に複数のユーザーを作成できます AWS アカウント。これらのユーザーは自分の認証情報 (パスワード、アクセスキー ID、およびシークレットキー) を持つことができますが、すべての IAM ユーザーは 1 つのアカウント番号を共有します。

ユーザーに IAMポリシーをアタッチして、IAM ユーザーのリソースアクセスのレベルを管理することができます。例えば、Amazon S3サービスおよびアカウントの関連リソースへのユーザーアクセスを付与するポリシーを IAMユーザーにアタッチすることができますが、他のサービスもしくはリソースへのアクセスはできません。

より効率的なアクセス管理を行うためには、ユーザーの集合である、IAM グループを作成できます。グループにポリシーをアタッチすると、そのグループのメンバーであるすべてのユーザーに反映されます。

IAMは、ユーザーおよびグループレベルでのアクセス許可の管理に加えて、IAMロールの概念もサポートしています。ユーザーおよびグループと同様に、IAM ロールにポリシーをアタッチすることができます。IAM ロールを Amazon EC2インスタンスに関連付けることができます。EC2 インスタンスで実行されるアプリケーションは、IAM ロールによって提供されるアクセス許可 AWS を使用して にアクセスできます。ツールキットでの IAM ロールの使用の詳細については、「IAM ロールの作成」を参照してください。IAM の詳細については、IAM ユーザーガイドを参照してください。

IAM ユーザーを作成して設定する

IAM ユーザーを使用すると、他のユーザーに へのアクセスを許可できます AWS アカウント。IAM ユーザーにポリシーをアタッチすることができるため、IAM ユーザーがアクセスできるリソースおよびそれらのリソースに実行できる操作を正確に制限することができます。

ベストプラクティスとして、 にアクセスするすべてのユーザーは、アカウントの所有者であっても、IAM ユーザーとしてアクセス AWS アカウント する必要があります。これにより、IAM ユーザーのうちの 1 人の認証情報が漏洩した場合でも、それらの認証情報のみを非アクティブ化することができます。アカウントのルート認証情報を非アクティブ化または変更する必要はありません。

Toolkit for Visual Studio からIAM ユーザーにアクセス許可を割り当てるには、ユーザーに IAM ポリシーをアタッチするか、ユーザーをグループに割り当てます。グループに割り当てた IAM ユーザーは、グループにアタッチされているポリシーからアクセス許可を引き継ぎます。詳細については、「IAM グループを作成する」と「IAM グループに IAM ユーザーを追加する」を参照してください。

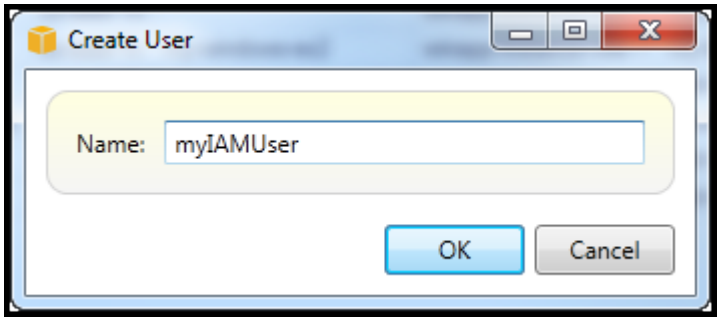
Toolkit for Visual Studio から、IAM ユーザーの AWS 認証情報 (アクセスキー ID とシークレットキー) を生成することもできます。詳細については、「IAM ユーザーの認証情報を生成する」を参照してください。

Toolkit for Visual Studio は、 AWS Explorer を介してサービスにアクセスするための IAM ユーザー認証情報の指定をサポートしています。通常、IAM ユーザーにはすべての Amazon Web Services へのフルアクセスがないため、 AWS Explorer の一部の機能を利用できない場合があります。Explorer を使用してアクティブなアカウントが IAM ユーザーである間にリソースを変更し、アクティブなアカウントをルートアカウントに切り替えると、 AWS Explor AWS er でビューを更新するまで変更が表示されない場合があります。表示を更新するには、[Refresh (更新)] ボタンを選択します。

から IAM ユーザーを設定する方法については AWS マネジメントコンソール、IAM ユーザーガイドの「ユーザーとグループの使用」を参照してください。

IAM ユーザーを作成するには

1. AWS Explorer でノードを展開しAWS Identity and Access Management、ユーザーのコンテキスト (右クリック) メニューを開き、ユーザーの作成を選択します。
2. [Create User] (ユーザーの作成) ダイアログボックスで IAM ユーザーの名前を入力して、[OK] を選択します。これは IAM のフレンドリ名です。IAM ユーザー名の制約については、IAM ユーザーガイドを参照してください。



Create an IAM user

新しいユーザーは AWS Identity and Access Management ノードの下の [Users] (ユーザー) の下のサブノードとして表示されます。

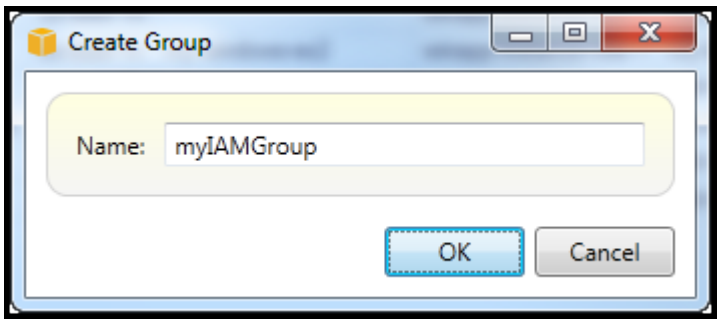
ポリシーを作成してユーザーにアタッチする方法については、「IAM ポリシーを作成する」を参照してください。

IAM グループを作成する

グループは、IAMポリシーをユーザーの集合に適用する方法を提供します。IAM ユーザーおよびグループを管理する方法の詳細については、IAM ユーザーガイドの「ユーザーおよびグループの使用」を参照してください。

IAM グループを作成するには

1. AWS Explorer の Identity and Access Management で、Groups のコンテキスト (右クリック) メニューを開き、Create Group を選択します。
2. [Create Group] (グループの作成) ダイアログボックスで IAM グループの名前を入力して、[OK] を選択します。



Create IAM group

新しい IAM グループは、[Identity and Access Management] の [Groups]] (グループ) サブノードの下に表示されます。

ポリシーを作成して IAM グループにアタッチする方法については、「IAM ポリシーを作成する」を参照してください。

IAM グループに IAM ユーザーを追加する

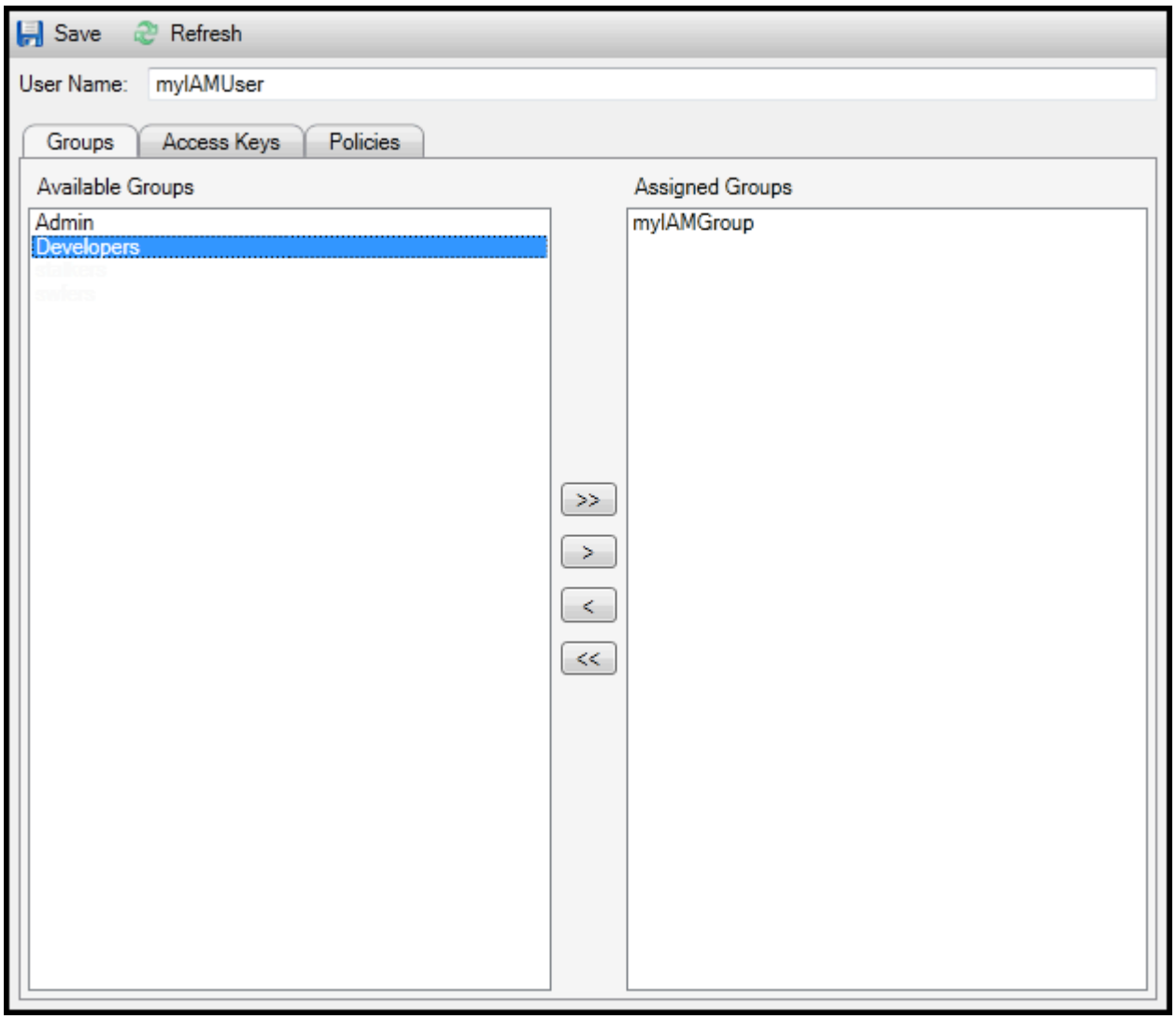
IAMグループのメンバーである IAMユーザーは、グループにアタッチされているポリシーからアクセス許可を引き継ぎます。IAMグループの目的は、IAMユーザーの集合全体のアクセス許可の管理を容易にすることです。

IAM グループにアタッチされたポリシーが、その IAM グループのメンバーである IAM ユーザーにアタッチされたポリシーと連携する方法の詳細については、IAM ユーザーガイドの「IAM ポリシーを管理する」を参照してください。

AWS Explorer では、Groups サブノードではなく Users サブノードから IAM グループに IAM ユーザーを追加します。

IAM グループに IAM ユーザーを追加するには

1. AWS Explorer の Identity and Access Management で、ユーザーのコンテキスト (右クリック) メニューを開き、編集を選択します。



Assign an IAM user to a IAM group

2. [Groups] (グループ) タブの左ペインに、使用可能な IAM グループが表示されます。右側のペインには、指定された IAM ユーザーが既にメンバーであるグループが表示されます。

   IAM ユーザーをグループに追加するには、左側のペインで、IAM グループを選択し、[>] ボタンを選択します。

   IAM ユーザーをグループから削除するには、右側のペインで、IAM グループを選択し、[<] ボタンを選択します。

   IAM ユーザーをすべての IAM グループに追加するには、[>>] ボタンを選択します。同様に、IAM ユーザーをすべてのグループから削除するには、[<<] ボタンを選択します。

   複数のグループを選択するには、それらを順に選択します。コントロールキーを押しながら選択する必要はありません。グループを選択からクリアするには、再度選択します。

3. IAM ユーザーの IAM グループへの割り当てが完了したら、[Save] (保存) を選択します。

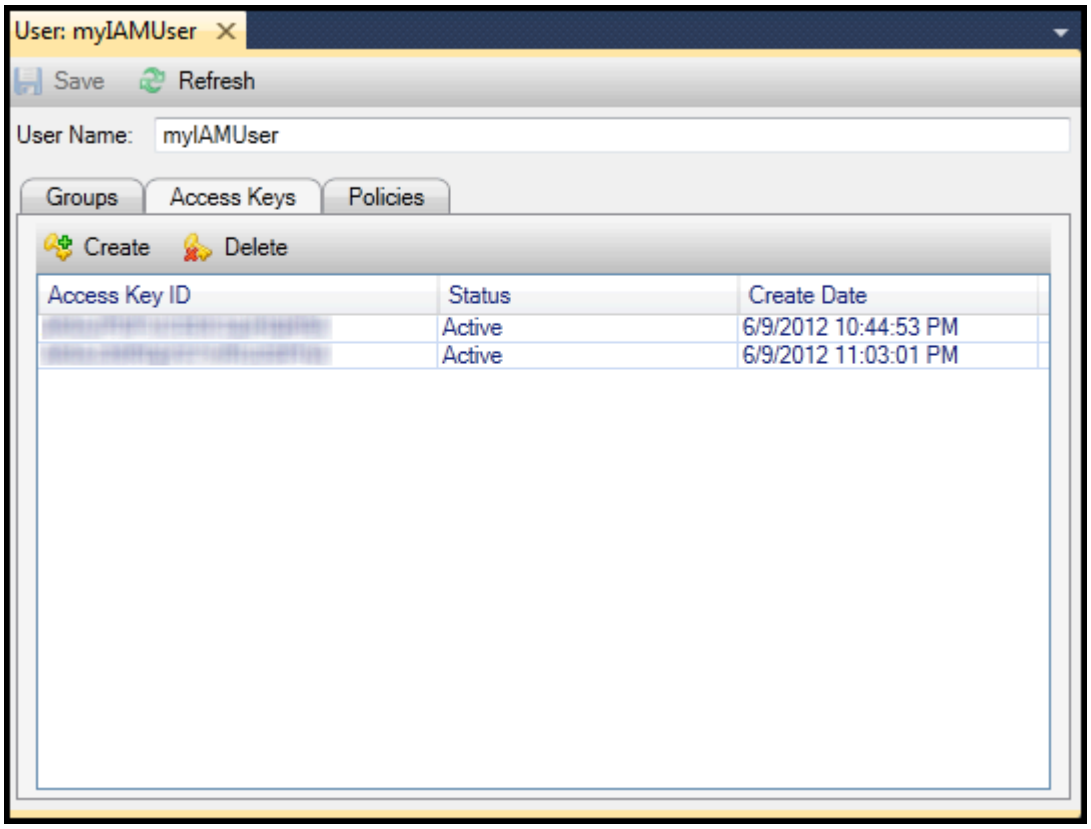
IAMユーザーの認証情報を生成する

Toolkit for Visual Studioを使用すると、 AWSへの API 呼び出しを実行するために使用する、アクセスキー ID およびシークレットキーを生成できます。これらのキーは、ツールキットを使って Amazon Web Services にアクセスするために指定することもできます。Toolkit 用に認証情報を指定する方法については、「認証情報」を参照してください。認証情報を安全に処理する方法の詳細については、AWS 「アクセスキーを管理するためのベストプラクティス」を参照してください。

ツールキットは、IAM ユーザーのパスワードを生成するために使用することはできません。

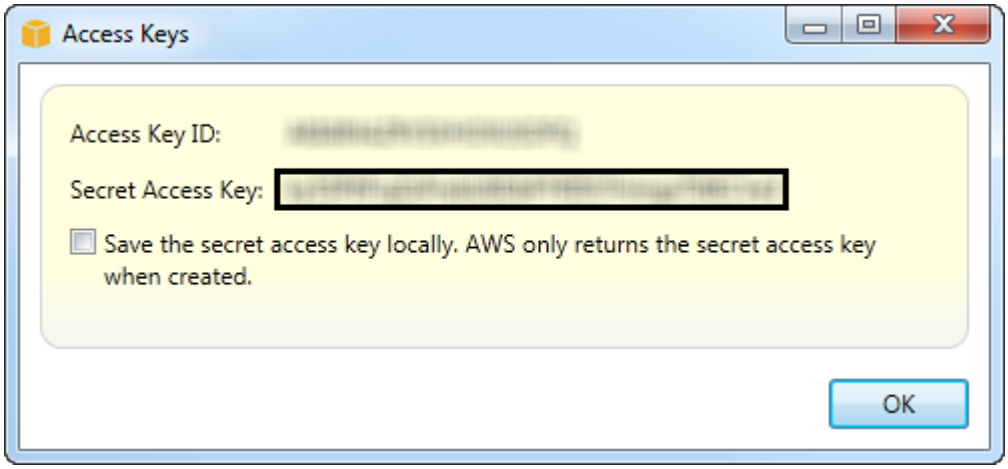
IAM ユーザーの認証情報を生成するには

1. AWS Explorer で、IAM ユーザーのコンテキスト (右クリック) メニューを開き、編集を選択します。



2. 認証情報を生成するには、[Access Keys (アクセスキー)] タブで、[Create (作成)] を選択します。

IAM ユーザーごとに 2 セットのみの認証情報を生成できます。2 セットの認証情報が既にあり、追加のセットを作成する必要がある場合は、既存のセットのいずれかを削除する必要があります。

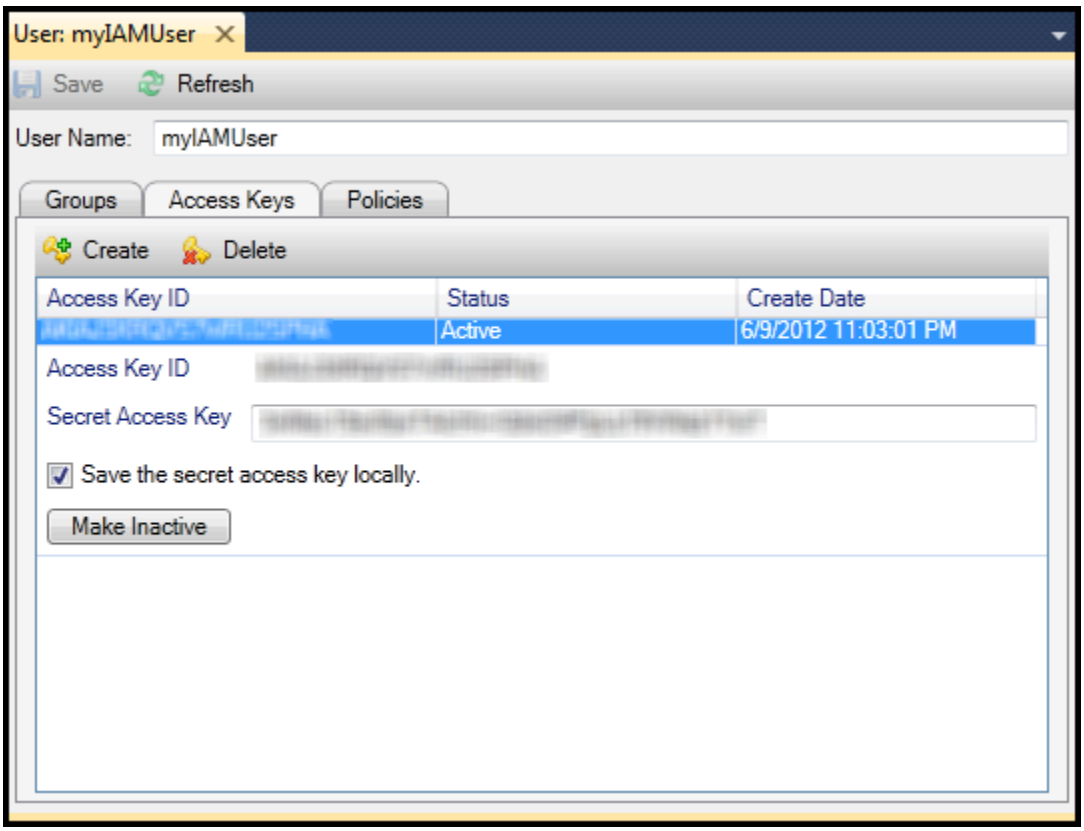


reate credentials for IAM user

Toolkit でシークレットアクセスキーの暗号化されたコピーをローカルドライブに保存する場合は、シークレットアクセスキーをローカルに保存を選択します。 は作成時に AWS のみシークレットアクセスキーを返します。ダイアログボックスからシークレットアクセスキーをコピーして、それを安全な場所に保存することもできます。

3. [OK] を選択してください。

認証情報を生成した後、それを [Access Keys (アクセスキー)] タブから表示できます。ツールキットでシークレットキーをローカルに保存するオプションを選択した場合は、ここに表示されます。



Create credentials for IAM user

自分でシークレットキーを保存した場合で、ツールキットでもそれを保存する場合は、[Secret Access Key (シークレットアクセスキー)] ボックスにシークレットアクセスキーを入力し、[Save the secret access key locally (シークレットアクセスキーをローカルに保存)] を選択します。

認証情報を無効化するには、[Make Inactive (無効化)] を選択します。(認証情報の漏洩が疑われる場合は、これを実行します。 認証情報が安全であることが確認できたら、再アクティブ化することができます)。

IAM ロールを作成します。

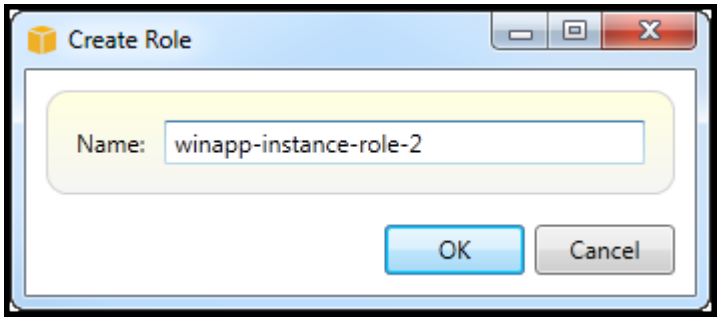
Toolkit for Visual Studio では、IAM ロールの作成および設定をサポートしています。ユーザーおよびグループと同様に、IAM ロールにポリシーをアタッチすることができます。IAM ロールを Amazon EC2 インスタンスに関連付けることができます。EC2 インスタンスとの関連付けは、インスタンスプロファイルを介して処理されます。これはロールの論理コンテナです。EC2 インスタンスで実行されるアプリケーションは、IAM ロールに関連付けられたポリシーによって指定されるアクセスのレベルを自動的に付与されます。これは、アプリケーションが他の AWS 認証情報を指定していない場合でも当てはまります。

例えば、ロールを作成して、Amazon S3 のみへのアクセスに制限するポリシーをそのロールにアタッチできます。このロールを EC2 インスタンスに関連付けた後、そのインスタンスでアプリケーションを実行すると、アプリケーションは Amazon S3 にアクセスできますが、他のサービスまたはリソースにはアクセスできません。このアプローチの利点は、EC2 インスタンスで AWS 認証情報を安全に転送して保存する必要がないことです。

IAM ロールの詳細については、IAM ユーザーガイドの「IAM ロールを使用する」を参照してください。Amazon EC2 インスタンスに関連付けられた IAM ロール AWS を使用して にアクセスするプログラムの例については、Java、.NET、PHP、Ruby (IAM を使用した認証情報の設定、IAM ロールの作成、IAM ポリシーの使用) の AWS 開発者ガイドを参照してください。

IAM ロールを作成するには

1. AWS Explorer の Identity and Access Management で、ロールのコンテキスト (右クリック) メニューを開き、ロールの作成を選択します。
2. [Create Role] (ロールの作成) ダイアログボックスで IAM ロールの名前を入力して、[OK] を選択します。



Create IAM role

新しい IAM ロールは [Identity and Access Management] の [Roles] (ロール) の下に表示されます。

ポリシーを作成してロールにアタッチする方法については、「IAM ポリシーを作成する」を参照してください。

IAM ポリシーの作成

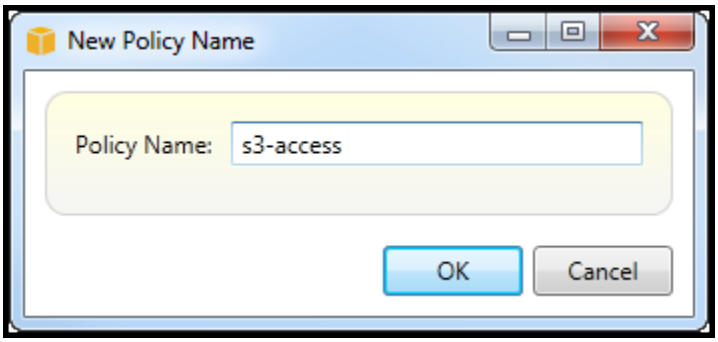
ポリシーは IAM にとって非常に重要です。ポリシーは IAM エンティティ (ユーザー、グループ、ロールなど) に関連付けることができます。ポリシーによって、ユーザー、グループ、またはロールに対して有効にする、アクセスのレベルを指定できます。

IAM ポリシーを作成するには

AWS Explorer で、AWS Identity and Access Managementノードを展開し、ポリシーをアタッチするエンティティのタイプ (グループ、ロール、またはユーザー) のノードを展開します。例えば、IAM ロールのコンテキストメニューを開き、[Edit] (編集) を選択します。

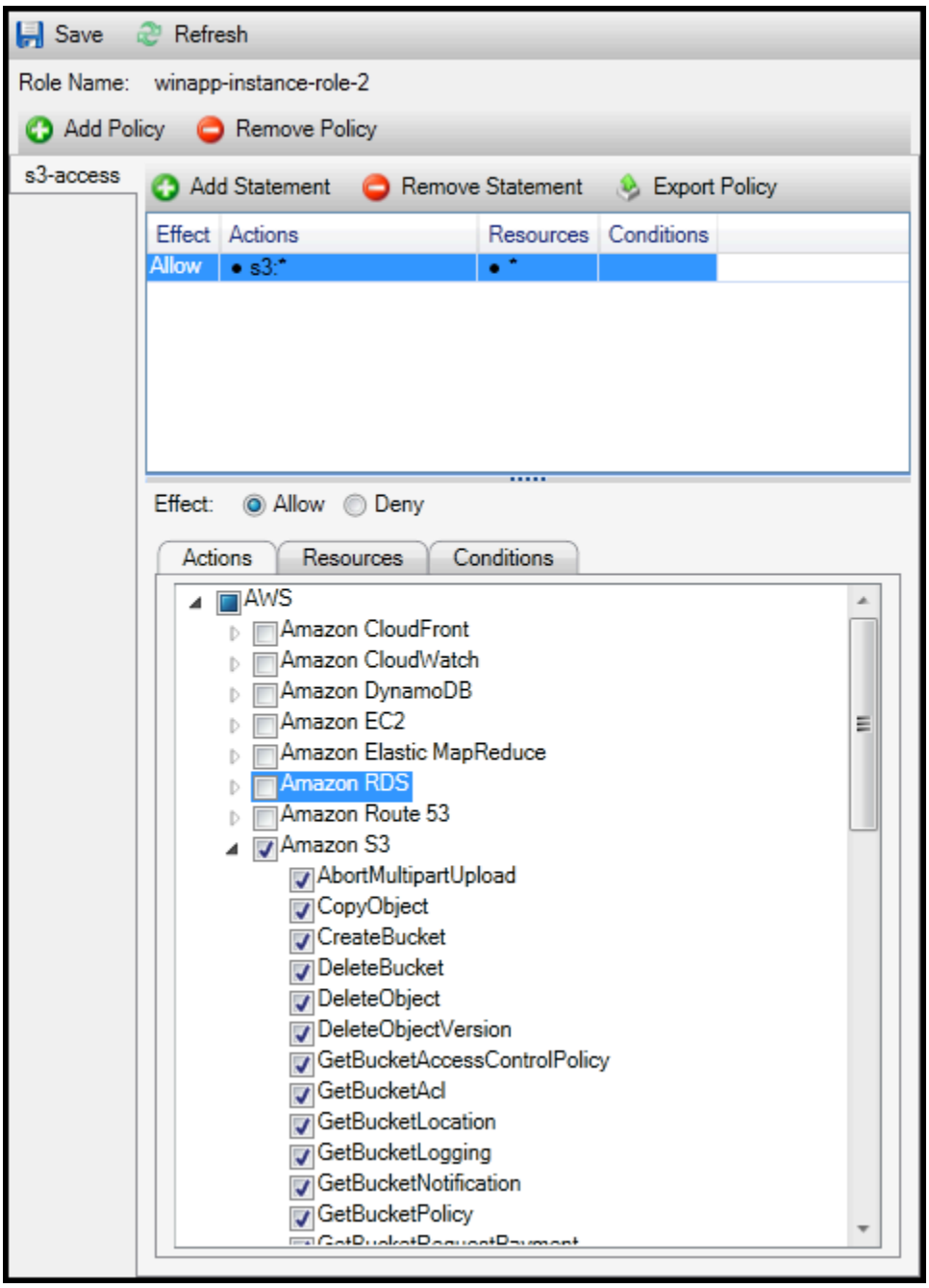
ロールに関連付けられたタブが AWS Explorer に表示されます。[Add Policy (ポリシーの追加)] リンクを選択します。

[New Policy Name (新しいポリシー名)] ダイアログボックスで、ポリシーの名前 (たとえば s3-access) を入力します。



New Policy Name dialog box

ポリシーエディタで、ポリシーステートメントを追加して、ロールに提供するアクセスレベルを指定します (この例では、ポリシーに関連付けられている winapp-instance-role-2 です)。この例では、ポリシーは Amazon S3 へのフルアクセスを提供しますが、他のリソースへのアクセス権は提供しません。



Specify IAM policy

より正確なアクセス制御には、ポリシーエディタでサブノードを展開して、Amazon Web Services に関連付けられたアクションを許可または拒否します。

ポリシーを編集したら、[Save (保存)] リンクを選択します。

AWS Lambda

を使用して .NET Core ベースの C# Lambda 関数を開発およびデプロイします AWS Toolkit for Visual Studio。 AWS Lambda は、サーバーのプロビジョニングや管理を行わずにコードを実行できるようにするコンピューティングサービスです。Toolkit for Visual Studio には、Visual Studio 用の AWS Lambda .NET Core プロジェクトテンプレートが含まれています。

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。

.NET Core の詳細については、Microsoft の「.NET Core」ガイドを参照してください。Windows、macOS、Linux の各プラットフォームでの、.NET Core の前提条件とインストール方法については、「.NET Core のダウンロード」を参照してください。

以下のトピックでは、 Toolkit for Visual Studio AWS Lambda を使用して を操作する方法について説明します。

トピック

- 基本 AWS Lambda プロジェクト
- Docker イメージの作成の基本 AWS Lambda プロジェクト
- チュートリアル: を使用してサーバーレスアプリケーションを構築およびテストする AWS Lambda
- チュートリアル: Amazon Rekognition Lambda アプリケーションの作成
- チュートリアル: AWS Lambda で Amazon ログ記録フレームワークを使用してアプリケーションログを作成する

基本 AWS Lambda プロジェクト

AWS Toolkit for Visual Studioでは、Microsoft .NET Core プロジェクト テンプレートを使用して Lambda 関数を作成できます。

Visual Studio .NET Core Lambda プロジェクトを作成する

Lambda-Visual Studio テンプレートとブループリントを使用して、プロジェクトの初期化を高速化できます。Lambda ブループリントには、柔軟なプロジェクト基盤の作成を簡素化する、事前に作成された関数が含まれています。

> **Note**
>
> Lambda サービスでは、パッケージタイプごとにデータ制限があります。データ制限の詳細については、AWS Lambda ユーザーガイドの「Lambda クォータ」のトピックを参照してください。

Visual Studio で Lambda プロジェクトを作成するには

1. Visual Studio の [ファイル] メニューを展開し、[新規] を展開し、[プロジェクト] を選択します。
2. [新しいプロジェクト] ダイアログボックスで、[言語]、[プラットフォーム]、および [プロジェクトタイプ] ドロップダウンボックスを [すべて] に設定し、[検索] フィールドに「aws lambda」と入力します。[AWS Lambda プロジェクト (.NET Core - C#)] テンプレートを選択します。
3. [名前] フィールドに「**`AWSLambdaSample`**」と入力し、ファイルの [場所]を指定して、[作成] を選択して続行します。
4. [ブループリントの選択] ページで、[空の関数] ブループリントを選択し、[完了] を選択して Visual Studio プロジェクトを作成します。

プロジェクトファイルを確認する

見直すべきプロジェクトファイルとして、`aws-lambda-tools-defaults.json` と `Function.cs` の 2 つがあります。

次の例は、プロジェクトの一部として自動的に作成される `aws-lambda-tools-defaults.json` ファイルを示しています。このファイルのフィールドを使用してビルドオプションを設定できます。

> **Note**
>
> Visual Studio のプロジェクトテンプレートにはさまざまなフィールドが含まれています。以下の点に注意してください。
>
> - 関数ハンドラー: Lambda 関数の実行時に実行されるメソッドを指定します。
> - [関数ハンドラー] フィールドに値を指定すると、公開ウィザードにその値が自動的に入力されます。
> - ただし、関数、クラス、またはアセンブリの名前を変更した場合、`aws-lambda-tools-defaults.json` ファイル内の対応するフィールドも更新する必要があります。

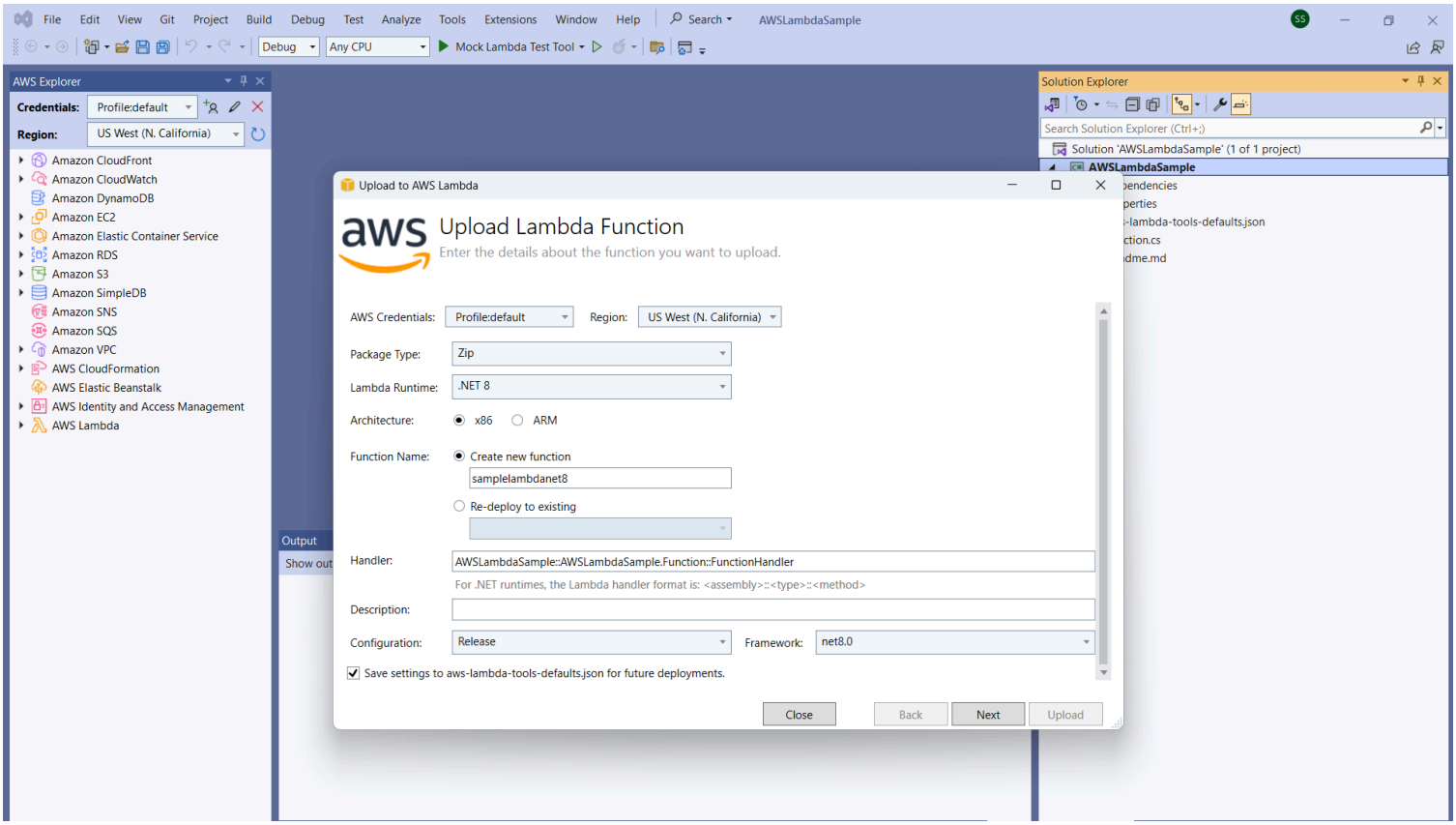
```
{
  "Information": [
    "This file provides default values for the deployment wizard inside Visual Studio
 and the AWS Lambda commands added to the .NET Core CLI.",
    "To learn more about the Lambda commands with the .NET Core CLI execute the
 following command at the command line in the project root directory.",
    "dotnet lambda help",
    "All the command line options for the Lambda command can be specified in this
 file."
  ],
  "profile": "default",
  "region": "us-west-2",
  "configuration": "Release",
  "function-architecture": "x86_64",
  "function-runtime": "dotnet8",
  "function-memory-size": 512,
  "function-timeout": 30,
  "function-handler": "AWSLambdaSample::AWSLambdaSample.Function::FunctionHandler"
}
```

Function.cs ファイルを調べます。Function.cs は、c# 関数を Lambda 関数として公開できるように定義します。このFunctionHandlerは、Lambda 関数の実行時に実行される Lambda 関数です。このプロジェクトでは、FunctionHandler という 1 つの関数が定義され、入力テキストに ToUpper() を呼び出します。

これでプロジェクトを Lambda に発行できるようになりました。

Lambda への発行

以下の手順と画像は、 AWS Toolkit for Visual Studioを使用して関数を Lambda にアップロードする方法を示しています。



Lambda への関数の公開

1. ビューを展開し、 AWS Explorer を選択して AWS Explorer に移動します。
2. Solution Explorer で、公開するプロジェクトのコンテキストメニューを開き (右クリック)、Lambda に発行を選択して AWS Lambda 関数のアップロードウィンドウを開きます。
3. [Lambda 関数のアップロード] ウィンドウで、以下のフィールドに入力します。

    a. パッケージタイプ: [**Zip**] を選択します。ZIP ファイルはビルドプロセスの結果として作成され、Lambda にアップロードされます。または、[パッケージタイプ] で [**Image**] を選択することもできます。「チュートリアル: 基本 AWS Lambda プロジェクトによる Docker イメージの作成」では、[パッケージタイプ] で [**Image**] を使用した公開方法について説明しています。

    b. Lambda ランタイム: ドロップダウンメニューから Lambda ランタイムを選択します。

    c. アーキテクチャ: 任意のアーキテクチャの矢印を選択します。

    d. 関数名: [新しい関数を作成する] の矢印を選択し、Lambda インスタンスの表示名を入力します。この名前は AWS Explorer と AWS マネジメントコンソール ディスプレイの両方で参照されます。

e. ハンドラー: このフィールドを使用して関数ハンドラーを指定します。例: **`AWSLambdaSample::AWSLambdaSample.Function::FunctionHandler`**。

f. (オプション) [説明]: AWS マネジメントコンソールでインスタンスに表示される説明文を入力します。

g. 設定: ドロップダウンメニューから任意の設定を選択します。

h. フレームワーク: ドロップダウンメニューから任意のフレームワークを選択します。

i. 設定の保存: このボックスを選択すると、現在の設定が `aws-lambda-tools-defaults.json` に保存され、今後のデプロイメントのデフォルトとして使用されます。

j. [次へ] を選択して、[関数の高度な詳細] ウィンドウに進みます。

4. [関数の高度な詳細] ウィンドウで、以下のフィールドに入力します。

a. ロール名: アカウントに関連付けられているロールを選択します。ロールは、 関数のコードによって行われた AWS サービス呼び出しの一時的な認証情報を提供します。ロールがない場合は、スクロールしてドロップダウンセレクタの AWS マネージドポリシーに基づいて新しいロールを見つけ、AWSLambdaBasicExecutionRole を選択します。このロールには最小限のアクセス許可があります。

> **Note**
>
> アカウントには IAM ListPolicies アクションを実行できるアクセス許可が必要です。この権限がない場合には、[Role Name (ロール名)] リストは空となり、続行できません。

b. (オプション) Lambda 関数が Amazon VPC 上のリソースにアクセスする場合、サブネットおよびセキュリティグループを選択します。

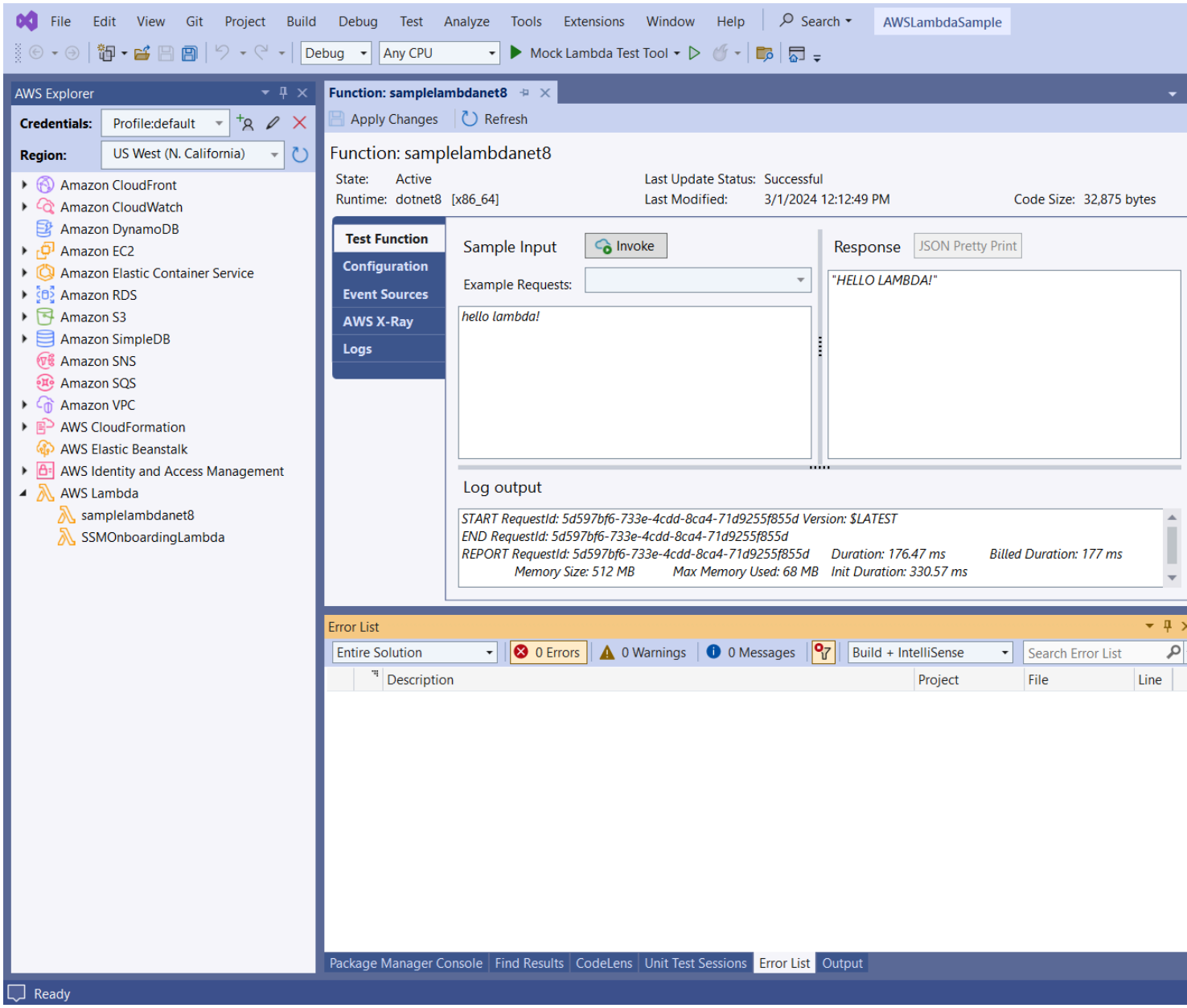
c. (オプション) Lambda 関数に必要となる環境変数を設定します。キーは、無料のデフォルトのサービスキーによって自動的に暗号化されます。または、料金が発生する AWS KMS キーを指定することもできます。KMS は、データの暗号化に使用される暗号化キーの作成と管理を行うために使用できる、マネージド型サービスです。 AWS KMS キーがある場合は、リストから選択できます。

5. [アップロード] を選択すると、[関数のアップロード中] ウィンドウが開き、アップロードプロセスが開始されます。

> **Note**
>
> 関数のアップロード中に、関数のアップロードページが表示されます AWS。アップロード後にレポートを表示するためにウィザードを開いたままにするには、アップロードが完了する前に、フォームの最後にある [Automatically close wizard on successful completion] (正常完了時にウィザードを自動で閉じる) のチェックボックスをオフにします。
>
> 関数がアップロードされると、Lambda 関数はライブ状態になります。[Function:] (関数:) ビューページが開き、新しい Lambda 関数の設定が表示されます。

6. [関数のテスト] タブの自由文入力フィールドに 「`hello lambda!`」と入力し、[呼び出し] を選択して Lambda 関数を手動で呼び出します。入力したテキストは、大文字に変換されて [レスポンス] タブに表示されます。

> **Note**
>
> [AWS Explorer] で、AWS Lambda ノードの下にあるデプロイ済みインスタンスをダブルクリックすると、いつでも [関数:] ビューを再び開くことができます。



7. (オプション) Lambda 関数が正常に発行されたことを確認するには、 にログイン AWS マネジメントコンソール し、Lambda を選択します。コンソールには、先ほど作成した関数を含め、発行されたすべての Lambda 関数が表示されます。

クリーンアップ

この例で開発を続行しない場合は、デプロイした関数を削除して、アカウント内の未使用のリソースに対して請求されないようにします。

> **Note**
>
> Lambda は、ユーザーに代わって Lambda 関数を自動でモニタリングし、Amazon CloudWatch からメトリクスを報告します。関数をモニタリングおよびトラブルシューティングするには、 AWS Lambda デベロッパーガイドのAmazon CloudWatch を使用した AWS Lambda 関数のトラブルシューティングとモニタリング」トピックを参照してください。

関数を削除するには

1. AWS Explorer から AWS Lambda ノードを展開します。
2. デプロイしたインスタンスを右クリックし、[削除]を選択します。

Docker イメージの作成の基本 AWS Lambda プロジェクト

Toolkit for Visual Studio を使用して、 AWS Lambda 関数を Docker イメージとしてデプロイできます。Docker を使用すると、ランタイムをより細かく制御できます。例えば、.NET 8.0 などのカスタムランタイムを選択できます。Docker イメージは、他のコンテナイメージと同じ方法でデプロイします。このチュートリアルは、「チュートリアル: 基本的な Lambda プロジェクト」に忠実に従っていますが、2 つには以下の違いがあります。

- このプロジェクトには Dockerfile が含まれています。
- 別の公開設定が選択されています。

Lambda コンテナイメージに関する詳細については、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 の使用の詳細については AWS Toolkit for Visual Studio、このユーザーガイドの トピックの AWS Lambda 「テンプレートの使用 AWS Toolkit for Visual Studio」を参照してください。

Visual Studio .NET Core Lambda プロジェクトを作成する

Lambda Visual Studio テンプレートとブループリントを使用して、プロジェクトの初期化を高速化できます。Lambda ブループリントには、柔軟なプロジェクト基盤の作成を簡素化する、事前に作成された関数が含まれています。

Visual Studio .NET Core Lambdaプロジェクトを作成するには

1. Visual Studio の [ファイル] メニューを展開し、[新規] を展開し、[プロジェクト] を選択します。
2. [新しいプロジェクト] ダイアログボックスで、[言語]、[プラットフォーム]、および [プロジェクトタイプ] ドロップダウンボックスを [すべて] に設定し、[検索] フィールドに「**aws lambda**」と入力します。[AWS Lambda プロジェクト (.NET Core - C#)] テンプレートを選択します。
3. [プロジェクト名] フィールドに「**AWSLambdaDocker**」と入力し、ファイルの [場所] を指定して [作成] を選択します。
4. [ブループリントの選択] ページで、[.NET 8 (コンテナイメージ)] ブループリントを選択し、[完了] を選択して Visual Studio プロジェクトを作成します。これでプロジェクトの構造とコードを確認できるようになりました。

プロジェクトファイルを確認する

以下のセクションでは、.NET 8 (コンテナイメージ) ブループリントによって作成された 3 つのプロジェクトファイルについて説明します。

1. `Dockerfile`
2. `aws-lambda-tools-defaults.json`
3. `Function.cs`

1. Dockerfile

`Dockerfile` は 3 つの主要なアクションを実行します。

- `FROM`: このイメージで使用するベースイメージを確立します。このベースイメージは、.NET ランタイム、Lambda ランタイム、および Lambda .NET プロセスのエントリポイントを提供するシェルスクリプトを提供します。
- `WORKDIR`: イメージの内部作業ディレクトリを として確立します`/var/task`。
- `COPY`: ビルドプロセスから生成されたファイルを、ローカルの場所からイメージのワークディレクトリにコピーします。

以下は、オプションで指定できる `Dockerfile` アクションです。

- ENTRYPOINT: ベースイメージには、イメージの起動時に実行されるスタートアッププロセスである ENTRYPOINT が既に含まれています。独自のエントリを指定する場合は、その基本エントリポイントを上書きします。
- CMD: 実行する AWS カスタムコードを指示します。カスタムメソッドには完全修飾名が必要です。この行は Dockerfile に直接含める必要があるか、発行プロセス中に指定することができます。

```
# Example of alternative way to specify the Lambda target method rather than during
 the publish process.
CMD [ "AWSLambdaDocker::AWSLambdaDocker.Function::FunctionHandler"]
```

以下は、.NET 8 (コンテナイメージ) ブループリントによって作成された Dockerfile の例です。

```
FROM public.ecr.aws/lambda/dotnet:8

WORKDIR /var/task

# This COPY command copies the .NET Lambda project's build artifacts from the host
 machine into the image.
# The source of the COPY should match where the .NET Lambda project publishes its build
 artifacts. If the Lambda function is being built
# with the AWS .NET Lambda Tooling, the `--docker-host-build-output-dir` switch
 controls where the .NET Lambda project
# will be built. The .NET Lambda project templates default to having `--docker-host-
build-output-dir`
# set in the aws-lambda-tools-defaults.json file to "bin/Release/lambda-publish".
#
# Alternatively Docker multi-stage build could be used to build the .NET Lambda project
 inside the image.
# For more information on this approach checkout the project's README.md file.
COPY "bin/Release/lambda-publish"  .
```

2. aws-lambda-tools-defaults.json

aws-lambda-tools-defaults.json ファイルは、Toolkit for Visual Studio のデプロイウィザードと .NET Core CLI のデフォルト値を指定するために使用されます。aws-lambda-tools-defaults.json ファイルで設定できるフィールドについて以下に説明します。

- profile: AWS プロファイルを設定します。
- region: リソースが保存されている AWS リージョンを設定します。

- configuration: 関数の公開に使用される設定を設定します。
- package-type: デプロイパッケージタイプをコンテナイメージまたは .zip ファイルアーカイブに設定します。
- function-memory-size: 関数のメモリ割り当てを MB 単位で設定します。
- function-timeout: タイムアウトとは、Lambda 関数がタイムアウトするまでの最大実行時間です。これは 1 秒単位で調整でき、最大値は 15 分です。
- docker-host-build-output-dir: Dockerfile 内の指示に対応するビルドプロセスの出力ディレクトリを設定します。
- image-command: メソッド、つまりLambda 関数で実行するコードの完全修飾名です。構文は次のとおりです：{Assembly}::{Namespace}.{ClassName}::{MethodName}詳細については、「ハンドラー署名」を参照してください。ここで image-command を設定すると、後ほど Visual Studio の発行ウィザードにこの値が事前入力されます。

以下は、.NET 8 (コンテナイメージ) ブループリントによって作成された aws-lambda-tools-defaults.json の例です。

```
{
  "Information": [
    "This file provides default values for the deployment wizard inside Visual Studio
 and the AWS Lambda commands added to the .NET Core CLI.",
    "To learn more about the Lambda commands with the .NET Core CLI execute the
 following command at the command line in the project root directory.",
    "dotnet lambda help",
    "All the command line options for the Lambda command can be specified in this
 file."
  ],
  "profile": "default",
  "region": "us-west-2",
  "configuration": "Release",
  "package-type": "image",
  "function-memory-size": 512,
  "function-timeout": 30,
  "image-command": "AWSLambdaDocker::AWSLambdaDocker.Function::FunctionHandler",
  "docker-host-build-output-dir": "./bin/Release/lambda-publish"
}
```

3. Function.cs

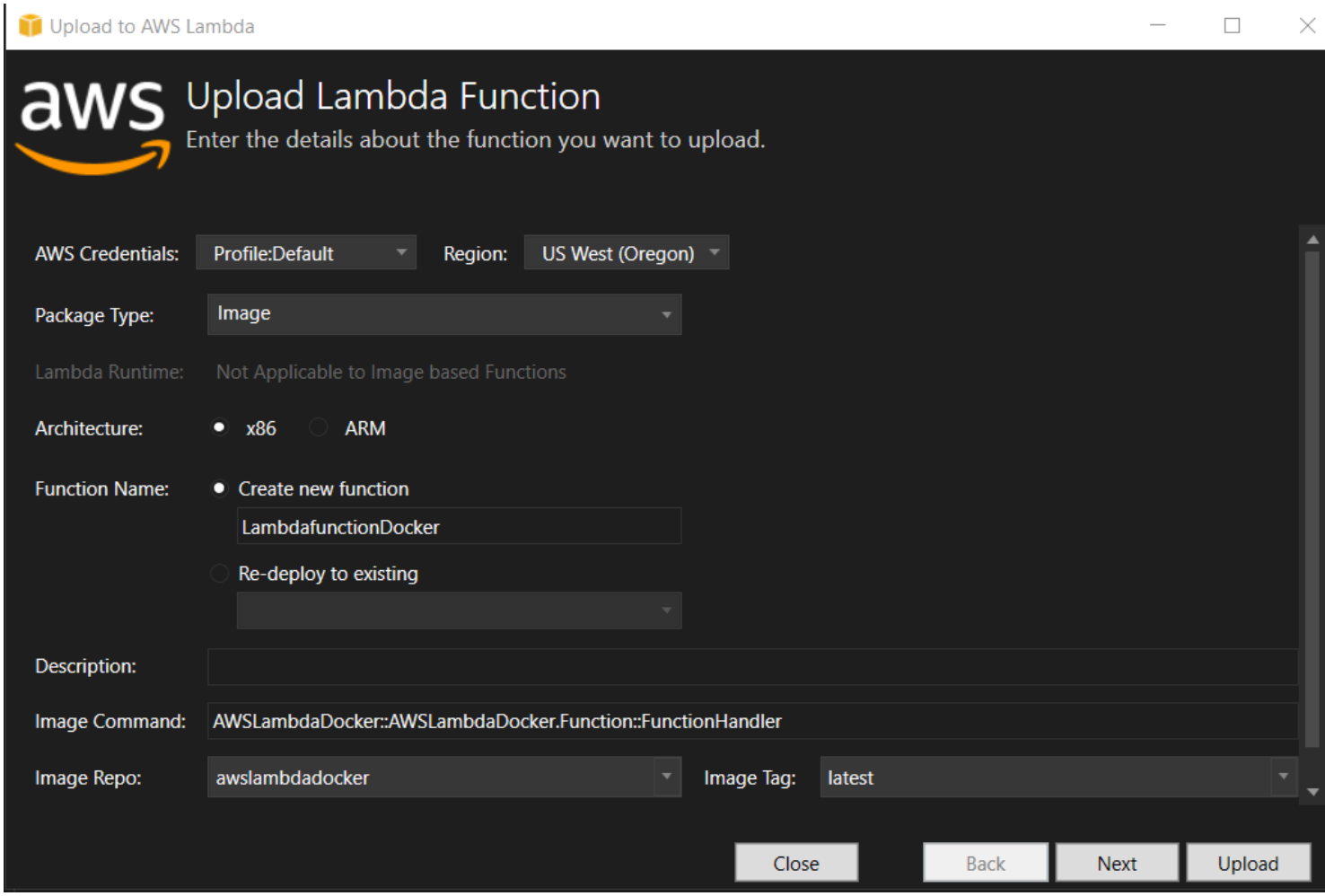
`Function.cs` ファイルは、Lambda 関数として公開する C# 関数を定義します。-`FunctionHandler`は、Lambda 関数の実行時に実行される Lambda 関数です。このプロジェクトでは、`FunctionHandler` は入力テキストに対して `ToUpper()` を呼び出します。

Lambda に発行する

ビルドプロセスによって生成された Docker イメージは、Amazon Elastic Container Registry (Amazon ECR) にアップロードされます。Amazon ECR は、デベロッパーが Docker コンテナイメージを簡単に保存、管理、デプロイできる完全マネージド型の Docker コンテナレジストリです。Amazon ECR はイメージをホストし、Lambda は呼び出されたときにプログラムされた Lambda 機能を提供するために参照します。

関数を Lambdaに発行するには

1. [Solution Explorer] で、プロジェクトのコンテキストメニュー (右クリック) を開き、[AWS Lambda Lambda に公開] を選択して [Lambda 関数のアップロード] ウィンドウを開きます。
2. [Lambda 関数のアップロード] ページで、次を実行します。



a. [Package Type] (パッケージタイプ) については、プロジェクト内で発行ウィザードが `Dockerfile` を検出したので、**`Image`** が パッケージタイプとして自動的に選択されます。

b. [Function Name] (関数名) には、Lambda インスタンスの表示名を入力します。この名前は、Visual Studio の AWS Explorer および AWS マネジメントコンソールの両方に表示される参照名です。

c. [Description] (説明) には、 AWS マネジメントコンソール内のインスタンスとともに表示するテキストを入力します。

d. [Image Command] (イメージコマンド) では、Lambda 関数で実行するメソッドへの完全修飾パスを入力します：**`AWSLambdaDocker::AWSLambdaDocker.Function::FunctionHandler`**

> **Note**
>
> ここに入力されたメソッド名は、Dockerfile 内のすべての CMD 命令を上書きします。[Image Command] (イメージコマンド) を入力するのが、オプションであるのは、`Dockerfile`が`CMD`Lambda 関数の起動方法を指示するを含む場合においてだけです。

e. [Image Repo] (イメージ Repo) では、新規もしくは既存の Amazon Elastic コンテナレジストリの名前を入力します。ビルドプロセスが作成する Docker イメージは、このレジストリにアップロードされます。公開されている Lambda 定義は、その Amazon ECR イメージを参照します。

f. [Image Tag] (イメージタグ)で、リポジトリ内のイメージに関連付ける Docker タグを入力します。

g. [次へ] を選択します。

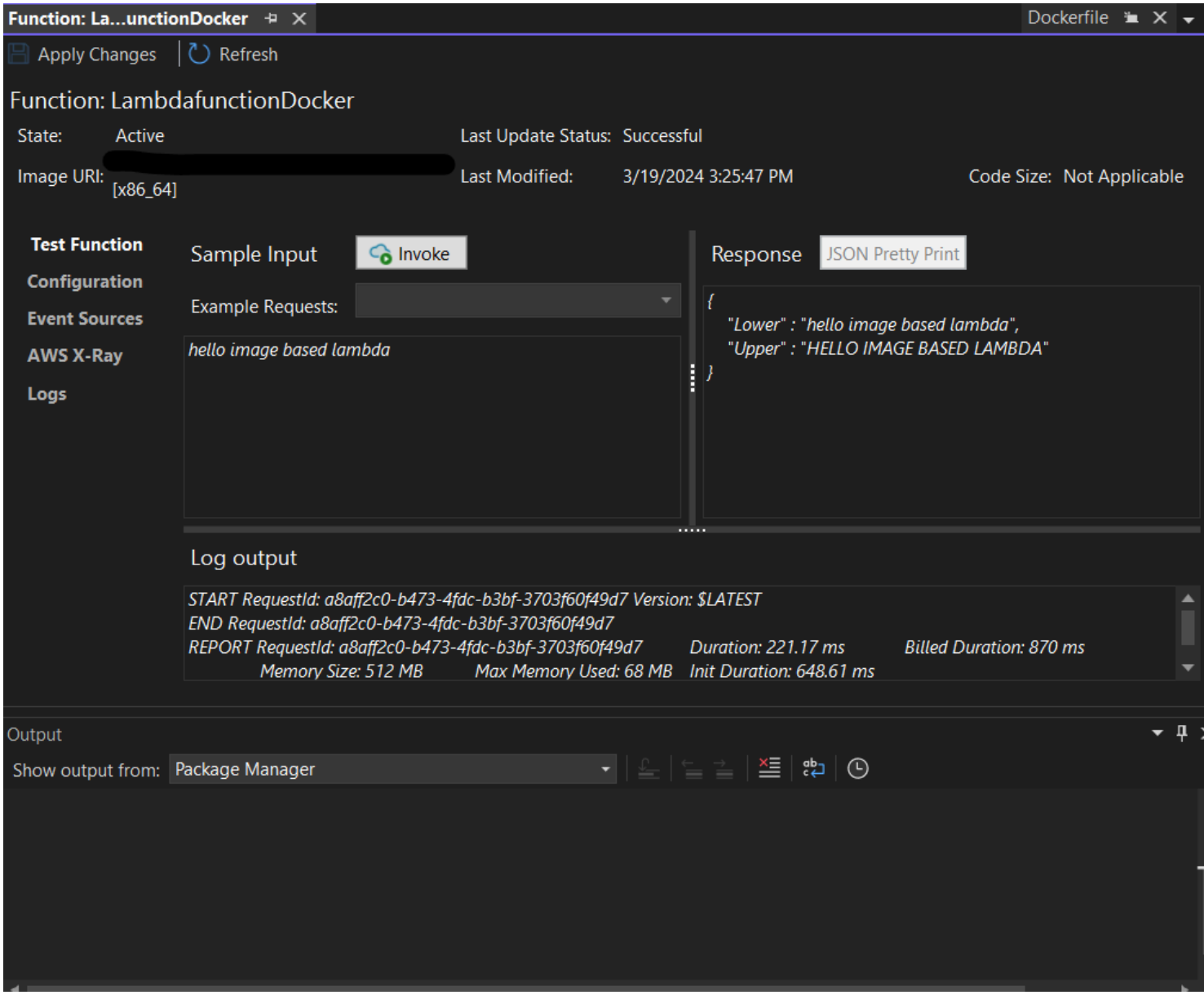
3. [Advanced Function Details] (アドバンスト関数の詳細) ページの [Role Name] (ロール名) で、アカウントに関連付けられているロールを選択します。このロールは、関数内のコードによって行われる Amazon Web Services コールに一時的な認証情報を提供するために使用されます。ロールがない場合は、 AWS 管理ポリシーに基づいて新しいロールを選択し、AWSLambdaBasicExecutionRole を選択します。

> **Note**
>
> アカウントには IAM ListPolicies アクションを実行できるアクセス許可が必要であり、そうしないと [Role Name] (ロール名) リストは空となります。

4. [アップロード] を選択し、アップロードと公開のプロセスを開始します。

> **Note**
>
> -関数がアップロード中にアップロード機能ページが表示されます。発行プロセスは、設定パラメータに基づいてイメージをビルドし、必要に応じて Amazon ECR リポジトリを作成し、必要ならばイメージをリポジトリにアップロードし、そのイメージを使用してそのリポジトリを参照する Lambda を作成します。
> 関数がアップロードされると、関数 ページが開き、新しい Lambda 関数の設定が表示されます。

5. Lambda 関数を手動で呼び出すには、[Test Function] (関数のテスト) タブで、リクエストの自由文入力フィールドに `hello image based lambda` を入力してから [Invoke] (呼び出し) を選択します。大文字に変換されたテキストは、[Response] (レスポンス) に表示されます。



6. リポジトリを参照するには、AWS Explorer で [Amazon Elastic Container Service] の下にある [Repositories] (リポジトリ) を選択します。

   [AWS Explorer] で、AWS Lambda ノードの下にあるデプロイ済みインスタンスをダブルクリックすると、いつでも [関数:] ビューを再び開くことができます。

> **Note**
>
> AWS Explorer ウィンドウが開いていない場合は、ビュー -> AWS Explorer を使用してドッキングできます。

7. [Configuration] (設定) タブに、イメージ固有の追加設定オプションがあります。このタブには、Dockerfile 内で指定された可能性がある `ENTRYPOINT`、`CMD`、および `WORKDIR` に上書きする方法が用意されています。説明アップロード/公開中に入力した説明 (ある場合) です。

クリーンアップ

この例で開発を続行しない場合は、デプロイされた関数および ECR イメージを削除して、アカウント内の未使用のリソースに対して請求されないようにすることを忘れないでください。

- 関数を削除するには、AWS Explorer で AWS Lambda ノードの下にあるデプロイ済みインスタンスを右クリックします。
- リポジトリは、AWS Explorer で [Amazon Elastic Container Service] - [Repositories] (リポジトリ) の下で削除できます。

次のステップ

Lambda イメージの作成およびテストについては、「Lambda でのコンテナイメージの使用」を参照してください。

コンテナイメージのデプロイ、アクセス許可、および構成設定の上書きの詳細については、「関数の設定」を参照してください。

チュートリアル: を使用してサーバーレスアプリケーションを構築およびテストする AWS Lambda

AWS Toolkit for Visual Studio テンプレートを使用してサーバーレス Lambda アプリケーションを構築できます。Lambda プロジェクトテンプレートには、AWS サーバーレスアプリケーション (SAM) の AWS Toolkit for Visual Studio 実装であるサーバーレスアプリケーション用のテンプレートが含まれています。 AWSAWSこのプロジェクトタイプを使用すると、 を使用してデプロイをオーケストレーションし、 AWS Lambda 関数のコレクションを開発し、アプリケーション全体で必要な AWS リソース AWS CloudFormation でデプロイできます。

の設定に関する前提条件と情報については AWS Toolkit for Visual Studio、 AWS Toolkit for Visual Studio の AWS 「Lambda テンプレートの使用」を参照してください。

トピック

- 新しい AWS サーバーレスアプリケーションプロジェクトを作成する
- サーバーレスアプリケーションファイルの確認
- サーバーレスアプリケーションのデプロイ
- サーバーレスアプリケーションのテスト

新しい AWS サーバーレスアプリケーションプロジェクトを作成する

AWS サーバーレスアプリケーションプロジェクトは、サーバーレス CloudFormation テンプレートを使用して Lambda 関数を作成します。 CloudFormation テンプレートを使用すると、データベースなどの追加のリソースを定義し、IAM ロールを追加し、一度に複数の関数をデプロイできます。これは、単一の AWS Lambda 関数の開発とデプロイに焦点を当てた Lambda プロジェクトとは異なります。

次の手順では、新しい AWS サーバーレスアプリケーションプロジェクトを作成する方法について説明します。

1. Visual Studio の [ファイル] メニューを展開し、[新規] を展開し、[プロジェクト] を選択します。
2. [新しいプロジェクト] ダイアログボックスで、[言語]、[プラットフォーム]、および [プロジェクトタイプ] のドロップダウンボックスが [すべて...] に設定されていることを確認して、[検索] フィールドに **`aws lambda`** を入力します。
3. AWS Serverless Application with Tests (.NET Core - C#) テンプレートを選択します。

   > **Note**
   >
   > AWS Serverless Application with Tests (.NET Core - C#) テンプレートが結果の上部に入力されない可能性があります。

4. [次へ] をクリックして、[新しいプロジェクトを設定] ダイアログを開きます。
5. [新しいプロジェクトを設定] ダイアログで、[名前] に **`ServerlessPowertools`** と入力し、残りのフィールドを必要に応じて入力します。[作成] ボタンを選択して、[ブループリントを選択] ダイアログに進みます。

6. [ブループリントを選択] ページで、[Powertools for AWS Lambda] ブループリントを選択し、[完了] を選択して Visual Studio プロジェクトを作成します。

サーバーレスアプリケーションファイルの確認

以下のセクションでは、プロジェクト用に作成された 3 つのサーバーレスアプリケーションファイルについて詳しく説明します。

1. serverless.template
2. Functions.cs
3. aws-lambda-tools-defaults.json

1. serverless.template

`serverless.template` ファイルは、サーバーレス関数やその他の AWS リソースを宣言するための AWS CloudFormation テンプレートです。このプロジェクトに含まれるファイルには、Amazon API Gateway を通じて HTTP *Get* オペレーションとして公開される単一の Lambda 関数の宣言が含まれています。このテンプレートを編集して、既存の関数をカスタマイズしたり、アプリケーションに必要な関数やその他のリソースを追加したりできます。

次は、`serverless.template` ファイルの例です。

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Transform": "AWS::Serverless-2016-10-31",
  "Description": "An AWS Serverless Application.",
  "Resources": {
    "Get": {
      "Type": "AWS::Serverless::Function",
      "Properties": {
         "Architectures": [
            "x86_64"
            ],
         "Handler": "ServerlessPowertools::ServerlessPowertools.Functions::Get",
         "Runtime": "dotnet8",
         "CodeUri": "",
         "MemorySize": 512,
         "Timeout": 30,
         "Role": null,
         "Policies": [
```

```
            "AWSLambdaBasicExecutionRole"
            ],
         "Environment": {
            "Variables": {
               "POWERTOOLS_SERVICE_NAME": "ServerlessGreeting",
               "POWERTOOLS_LOG_LEVEL": "Info",
               "POWERTOOLS_LOGGER_CASE": "PascalCase",
               "POWERTOOLS_TRACER_CAPTURE_RESPONSE": true,
               "POWERTOOLS_TRACER_CAPTURE_ERROR": true,
               "POWERTOOLS_METRICS_NAMESPACE": "ServerlessGreeting"
               }
            },
         "Events": {
            "RootGet": {
               "Type": "Api",
               "Properties": {
                  "Path": "/",
                  "Method": "GET"
                  }
               }
            }
         }
   },
  "Outputs": {
    "ApiURL": {
      "Description": "API endpoint URL for Prod environment",
      "Value": {
        "Fn::Sub": "https://${ServerlessRestApi}.execute-api.
${AWS::Region}.amazonaws.com/Prod/"
      }
    }
  }
}
```

`...AWS:: Serverless::Function...` 宣言フィールドの多くは、Lambda プロジェクトのデプロイメントのフィールドと類似していることに注意してください。Powertools のログ記録、メトリクス、トレースは、次の環境変数を使用して設定されます。

- POWERTOOLS_SERVICE_NAME=ServerlessGreeting
- POWERTOOLS_LOG_LEVEL=Info
- POWERTOOLS_LOGGER_CASE=PascalCase

- POWERTOOLS_TRACER_CAPTURE_RESPONSE=true
- POWERTOOLS_TRACER_CAPTURE_ERROR=true
- POWERTOOLS_METRICS_NAMESPACE=ServerlessGreeting

環境変数の定義と詳細については、「Powertools for AWS Lambda references」ウェブサイトを参照してください。

2. Functions.cs

Functions.cs は、テンプレートファイルで宣言された単一の関数にマッピングされた C# メソッドを含むクラスファイルです。Lambda 関数は API Gateway からの HTTP Get メソッドに応答します。以下は、Functions.cs ファイルの例です。

```
public class Functions
{
    [Logging(LogEvent = true, CorrelationIdPath = CorrelationIdPaths.ApiGatewayRest)]
    [Metrics(CaptureColdStart = true)]
    [Tracing(CaptureMode = TracingCaptureMode.ResponseAndError)]
    public APIGatewayProxyResponse Get(APIGatewayProxyRequest request, ILambdaContext
 context)
    {
        Logger.LogInformation("Get Request");

        var greeting = GetGreeting();

        var response = new APIGatewayProxyResponse
        {
            StatusCode = (int)HttpStatusCode.OK,
            Body = greeting,
            Headers = new Dictionary (string, string) { { "Content-Type", "text/
plain" } }
        };

        return response;
    }

    [Tracing(SegmentName = "GetGreeting Method")]
    private static string GetGreeting()
    {
        Metrics.AddMetric("GetGreeting_Invocations", 1, MetricUnit.Count);
```

```
        return "Hello Powertools for AWS Lambda (.NET)";
    }
}
```

3. aws-lambda-tools-defaults.json

aws-lambda-tools-defaults.json は、Visual Studio 内の AWS デプロイウィザードのデフォルト値と、.NET Core CLI に追加された AWS Lambda コマンドを提供します。このプロジェクトに含まれる aws-lambda-tools-defaults.json ファイルの例を次に示します。

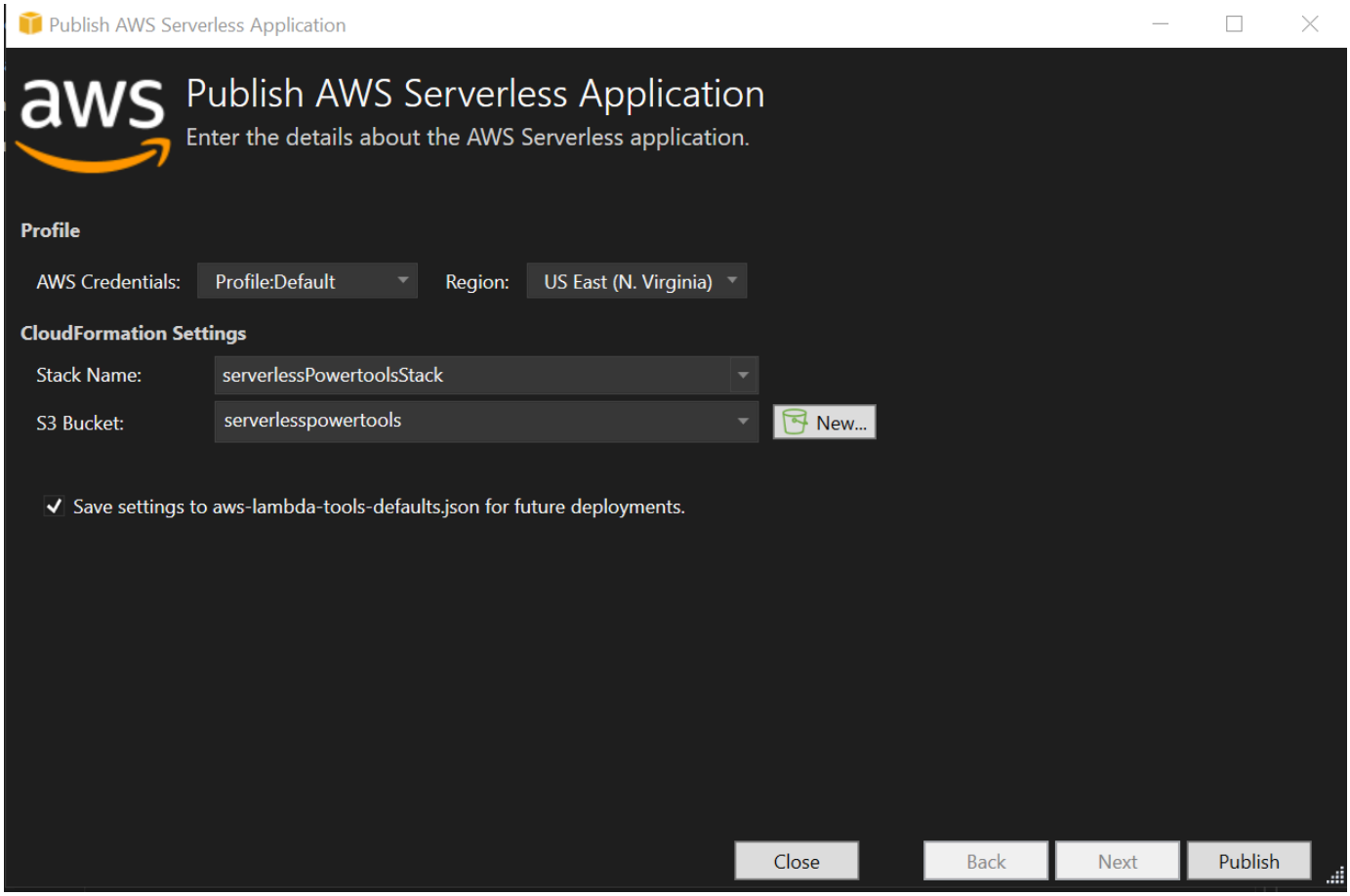
```
{
  "profile": "Default",
  "region": "us-east-1",
  "configuration": "Release",
  "s3-prefix": "ServerlessPowertools/",
  "template": "serverless.template",
  "template-parameters": ""
}
```

サーバーレスアプリケーションのデプロイ

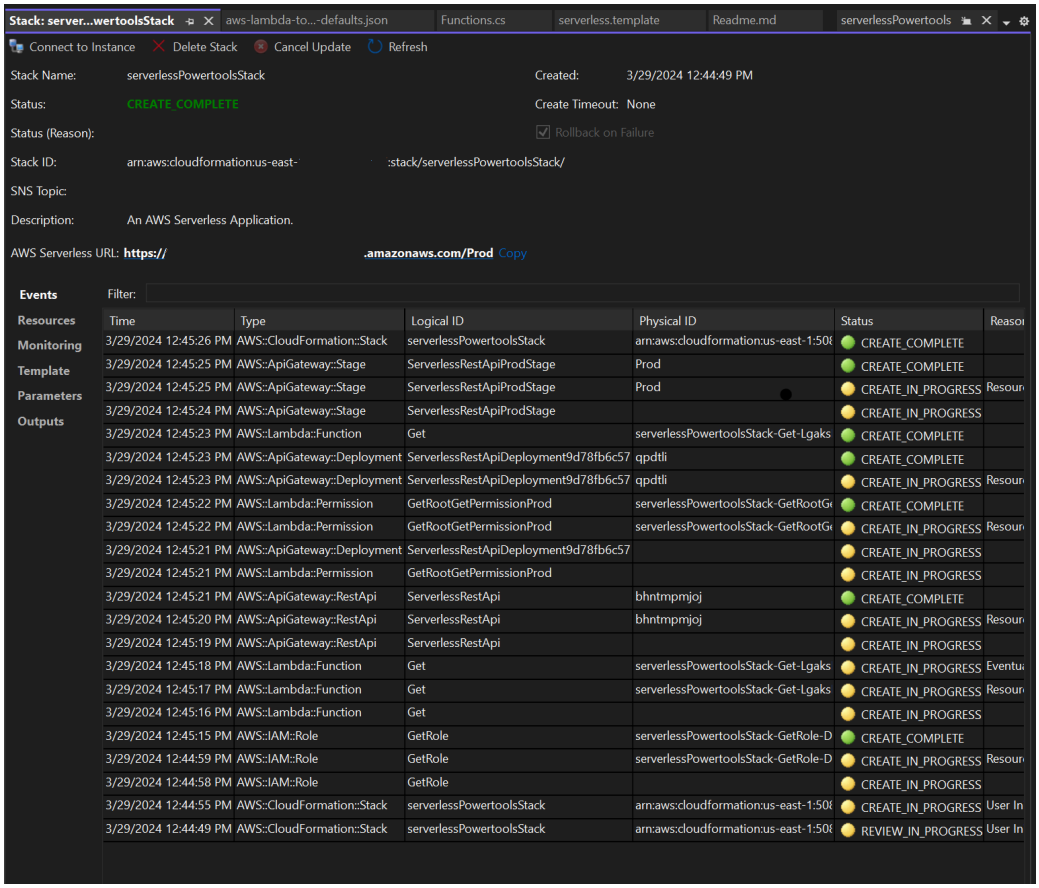
サーバーアプリケーションから以下のステップを実行します。

1. Solution Explorer から、プロジェクトのコンテキストメニューを開き (右クリック)、 AWS Lambda に発行を選択して AWS サーバーレスアプリケーションの発行ダイアログを開きます。
2. AWS サーバーレスアプリケーションの発行ダイアログで、 CloudFormation スタック名フィールドにスタックコンテナの名前を入力します。
3. [S3 バケット] フィールドで、アプリケーションバンドルがアップロードする Amazon S3 バケットを選択するか、[新規...] ボタンを選択して、新しい Amazon S3 バケットの名前を入力します。次に、[公開] を選択してアプリケーションを公開し、デプロイします。

> **Note**
>
> CloudFormation スタックと Amazon S3 バケットは同じ AWS リージョンに存在する必要があります。プロジェクトの残りの設定は、serverless.template ファイルで定義されます。



4. 公開プロセス中に [スタック] ビューウィンドウが開き、デプロイが完了すると、[ステータス] フィールドに CREATE_COMPLETE と表示されます。



サーバーレスアプリケーションのテスト

スタックの作成が完了したら、AWS サーバーレス URL を使用してアプリケーションを表示できます。関数やパラメータを追加せずにこのチュートリアルを完了した場合、 AWS サーバーレス URL にアクセスすると、ウェブブラウザに というフレーズが表示されますHello Powertools for AWS Lambda (.NET)。

チュートリアル: Amazon Rekognition Lambda アプリケーションの作成

このチュートリアルでは、Amazon Rekognition を使用して、検出ラベルのタグを Amazon S3 オブジェクトに付ける Lambdaアプリケーションを作成する方法を示します。

の設定に関する前提条件と情報については AWS Toolkit for Visual Studio、 AWS Toolkit for Visual Studio の AWS 「Lambda テンプレートの使用」を参照してください。

Visual Studio の NET Core Lambda Image Rekognition プロジェクトを作成します

次の手順では、 AWS Toolkit for Visual Studioから Amazon Rekognition Lambda アプリケーションを作成する方法について説明します。

> **Note**
>
> 作成すると、アプリケーションには 2 つのプロジェクトを含むソリューションが作成されます。1 つは Lambda にデプロイする Lambda 関数コードを含むソースプロジェクト、もう 1 つは関数をローカルでテストするための xUnit を使用したテストプロジェクトです。
> Visual Studio がプロジェクトの NuGet の参照をすべて見つけられない場合があります。これは、これらのブループリントで必要な依存関係を、NuGet から取得する必要があるからです。新しいプロジェクトが作成されたとき、Visual Studio はローカル参照だけを取り込み、NuGet からリモート参照を取り込みません。NuGet エラーを修正するには: 参照を右クリックし、[パッケージの復元] を選択します。

1. Visual Studio の [ファイル] メニューを展開し、[新規] を展開し、[プロジェクト] を選択します。
2. [新しいプロジェクト] ダイアログボックスで、[言語]、[プラットフォーム]、および [プロジェクトタイプ] のドロップダウンボックスが [すべて...] に設定されていることを確認して、[検索] フィールドに **`aws lambda`** を入力します。
3. AWS Lambda with Tests (.NET Core - C#) テンプレートを選択します。
4. [次へ] へをクリックして、[新しいプロジェクトを設定] ダイアログを開きます。
5. [新しいプロジェクトを設定] ダイアログで、[名前] に 「ImageRekognition」と入力し、残りのフィールドを必要に応じて入力します。[作成] ボタンを選択して、[ブループリントを選択] ダイアログに進みます。
6. [ブループリントの選択] ダイアログで、[Detect Image Labels] ブループリントを選択し、[完了] を選択して Visual Studio プロジェクトを作成します。

   > **Note**
   >
   > このブループリントには Amazon S3 イベントをリッスンするためのコードがあり、Amazon Rekognition を使用してラベルを検出し、それらを Amazon S3 オブジェクトにタグとして追加できます。

プロジェクトファイルを確認する

以下のセクションでは、これらのプロジェクトファイルについて説明します。

1. Function.cs
2. aws-lambda-tools-defaults.json

1. Function.cs

Function.cs ファイル内の最初のコードセグメントは、ファイルの先頭にある assembly 属性です。デフォルトでは、Lambda はタイプ System.IO.Stream の戻り値型と入力パラメータのみを受け入れます。入力パラメータや戻り値型で型付きのクラスを使用するには、シリアライザーを登録する必要があります。この assembly 属性は、Newtonsoft.Json を使用してストリームを型付きクラスに変換する Lambda JSON シリアライザーを登録します。シリアライザーをアセンブリまたはメソッドレベルで設定できます。

以下は assembly 属性の例です。

```
// Assembly attribute to enable the Lambda function's JSON input to be converted into
 a .NET class.
[assembly:
 LambdaSerializer(typeof(Amazon.Lambda.Serialization.SystemTextJson.DefaultLambdaJsonSerializer
```

このクラスには、2 つのコンストラクタがあります。1 つ目は、デフォルトコンストラクタで、Lambda がユーザーの関数を呼び出すときに使用されます。このコンストラクタは、Amazon S3 および Amazon Rekognition のサービスクライアントを作成します。また、コンストラクタは、デプロイ時に関数に割り当てる IAM ロールからこれらのクライアントの AWS 認証情報を取得します。クライアントの AWS リージョンは、Lambda 関数が実行されているリージョンに設定されます。このブループリントでは、Amazon Rekognition サービスがラベルに関して最低限の信頼度を持っている場合にのみ、Amazon S3 オブジェクトにタグを追加します。このコンストラクタは環境変数 MinConfidence をチェックして、許容できる信頼レベルを決定します。この環境変数は、Lambda 関数をデプロイするときに、設定できます。

以下は、Function.cs の最初のクラスコンストラクタの例です。

```
public Function()
{
    this.S3Client = new AmazonS3Client();
    this.RekognitionClient = new AmazonRekognitionClient();
```

```
    var environmentMinConfidence =
 System.Environment.GetEnvironmentVariable(MIN_CONFIDENCE_ENVIRONMENT_VARIABLE_NAME);
    if(!string.IsNullOrWhiteSpace(environmentMinConfidence))
    {
        float value;
        if(float.TryParse(environmentMinConfidence, out value))
        {
            this.MinConfidence = value;
            Console.WriteLine($"Setting minimum confidence to {this.MinConfidence}");
        }
        else
        {
            Console.WriteLine($"Failed to parse value {environmentMinConfidence} for
 minimum confidence. Reverting back to default of {this.MinConfidence}");
        }
    }
    else
    {
        Console.WriteLine($"Using default minimum confidence of {this.MinConfidence}");
    }
}
```

次の例は、2番目のコンストラクタをテストにどのように活用するかを示しています。テストプロジェクトでは、独自の S3 クライアントと Rekognition クライアントを設定し、それらを渡します。

```
public Function(IAmazonS3 s3Client, IAmazonRekognition rekognitionClient, float
 minConfidence)
{
    this.S3Client = s3Client;
    this.RekognitionClient = rekognitionClient;
    this.MinConfidence = minConfidence;
}
```

以下は、FunctionHandler ファイル内の Function.cs メソッドの例です。

```
public async Task FunctionHandler(S3Event input, ILambdaContext context)
{
    foreach(var record in input.Records)
    {
        if(!SupportedImageTypes.Contains(Path.GetExtension(record.S3.Object.Key)))
        {
            Console.WriteLine($"Object {record.S3.Bucket.Name}:{record.S3.Object.Key}
 is not a supported image type");
```

```
            continue;
        }

        Console.WriteLine($"Looking for labels in image {record.S3.Bucket.Name}:
{record.S3.Object.Key}");
        var detectResponses = await this.RekognitionClient.DetectLabelsAsync(new
 DetectLabelsRequest
        {
            MinConfidence = MinConfidence,
            Image = new Image
            {
                S3Object = new Amazon.Rekognition.Model.S3Object
                {
                    Bucket = record.S3.Bucket.Name,
                    Name = record.S3.Object.Key
                }
            }
        });

        var tags = new List();
        foreach(var label in detectResponses.Labels)
        {
            if(tags.Count < 10)
            {
                Console.WriteLine($"\tFound Label {label.Name} with confidence
 {label.Confidence}");
                tags.Add(new Tag { Key = label.Name, Value =
 label.Confidence.ToString() });
            }
            else
            {
                Console.WriteLine($"\tSkipped label {label.Name} with confidence
 {label.Confidence} because maximum number of tags reached");
            }
        }

        await this.S3Client.PutObjectTaggingAsync(new PutObjectTaggingRequest
        {
            BucketName = record.S3.Bucket.Name,
            Key = record.S3.Object.Key,
            Tagging = new Tagging
            {
                TagSet = tags
            }
```

```
        });
    }
    return;
}
```

FunctionHandler は、インスタンスの作成後に Lambda が呼び出すメソッドです。入力パラメータは S3Event 型で Stream ではないことに注意してください。これができるのは、登録された Lambda JSON シリアライザーのためです。S3Event には Amazon S 3 でトリガーされたイベントに関するすべての情報が含まれています。関数は、イベントの一部であるすべての S3 オブジェクトをループし、Rekognition に対しラベルの検出を指示します。ラベルが検出された後、それらは S3 オブジェクトにタグとして追加されます。

> **Note**
>
> コードには Console.WriteLine() の呼び出しが含まれています。Lambda で関数を実行する際に、すべての Console.WriteLine() の呼び出しを Amazon CloudWatch Logs にリダイレクトします。

2. aws-lambda-tools-defaults.json

aws-lambda-tools-defaults.json ファイルには、ブループリントによって設定されたデフォルト値が含まれており、デプロイウィザードの一部のフィールドに事前入力されます。また、.NET Core CLI との統合のためのコマンドラインオプションの設定にも役立ちます。

.NET Core CLI 統合にアクセスするには、関数のプロジェクトディレクトリに移動し、「**dotnet lambda help**」と入力します。

> **Note**
>
> 関数ハンドラーは、呼び出した関数に応答して Lambda が呼び出すメソッドを示します。このフィールドの形式は <assembly-name>::<full-type-name>::<method-name> です。型名には名前空間を含める必要があります。

関数をデプロイする

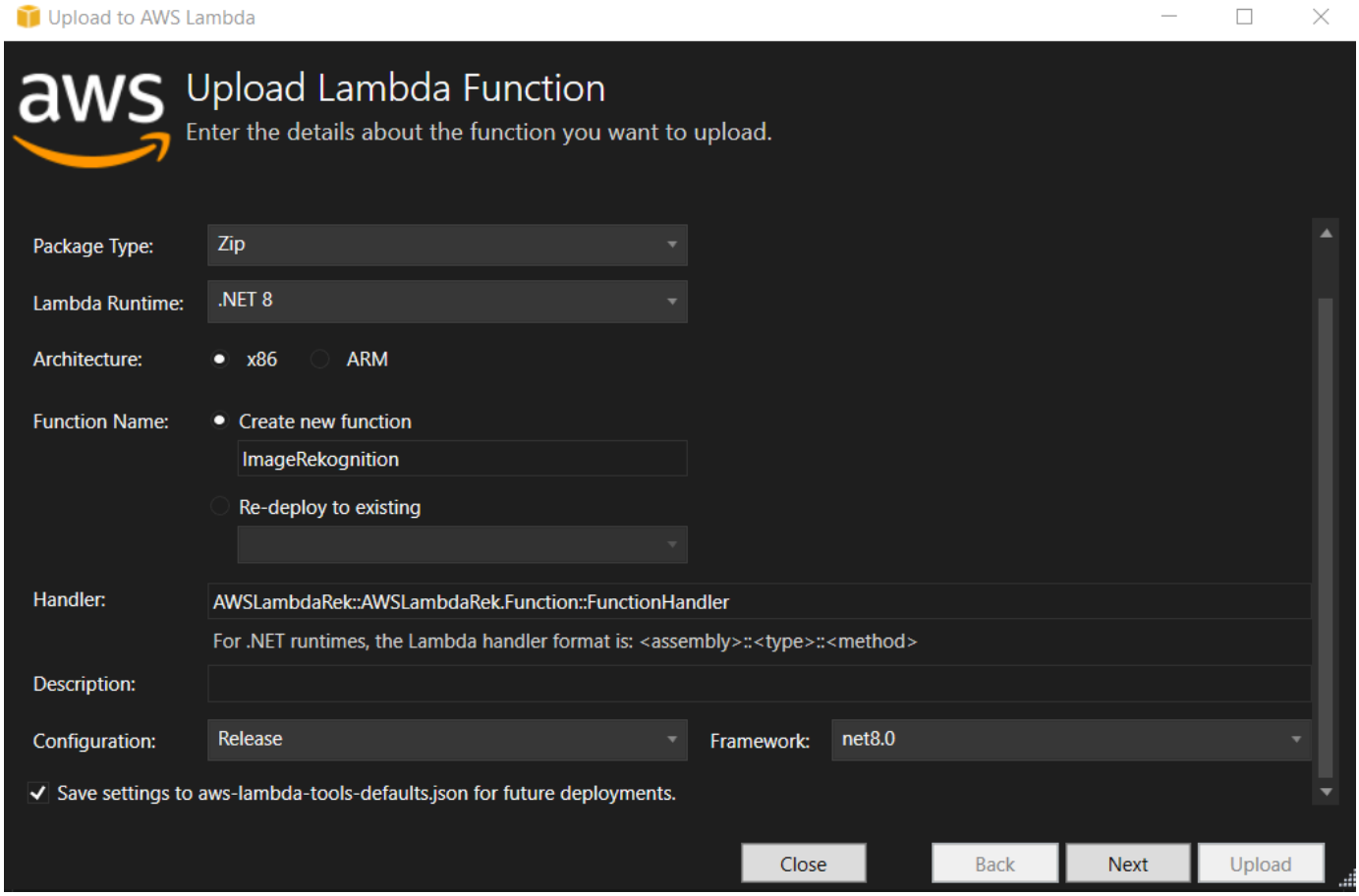
次の手順では、Lambda 関数をデプロイする方法について説明します。

1. Solution Explorer から、Lambda プロジェクトを右クリックし、 AWS Lambda に発行を選択して、Upload to AWS Lambda ウィンドウを開きます。

> **Note**
>
> プリセット値は `aws-lambda-tools-defaults.json` ファイルから取得されます。

2. [AWS Lambdaにアップロード] ウィンドウで [関数名] フィールドに名前を入力し、[次へ] ボタンを選択して [関数の高度な詳細] ウィンドウに進みます。

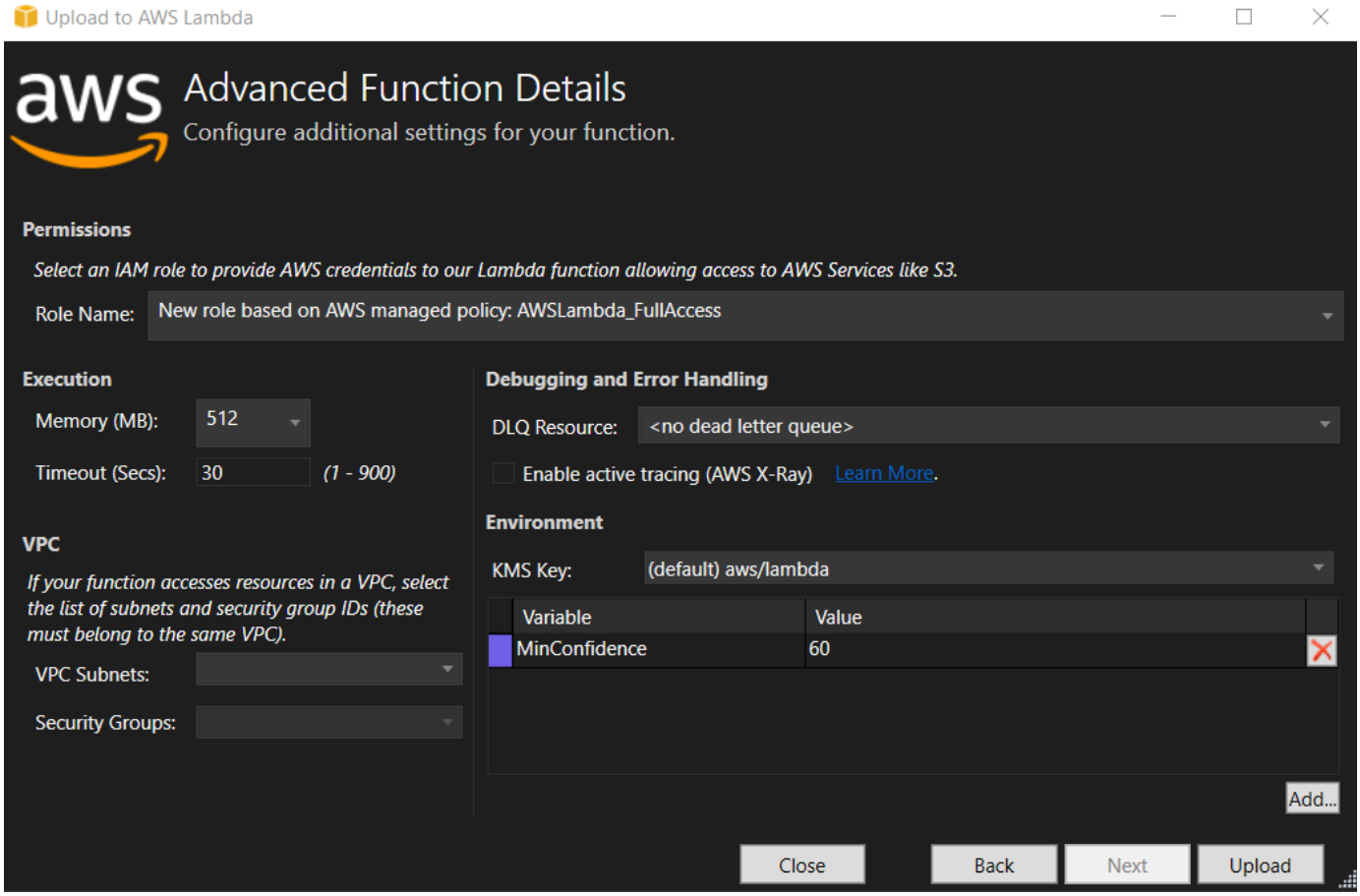
> **Note**
>
> この例では、[関数名] として「**`ImageRekognition`**」を使用します。



3. [関数の高度な詳細] ページで、S3 と Amazon Rekognition にアクセスするコードに対してアクセス許可を付与する IAM ロールを選択します。

> **Note**
>
> この例に従っている場合は、AWSLambda_FullAccess ロールを選択します。

4. 環境変数 MinConfidence を 60 に設定し、[アップロード] を選択してデプロイプロセスを起動します。[関数] ビューが [AWS Explorer] に表示されると、発行プロセスが完了します。



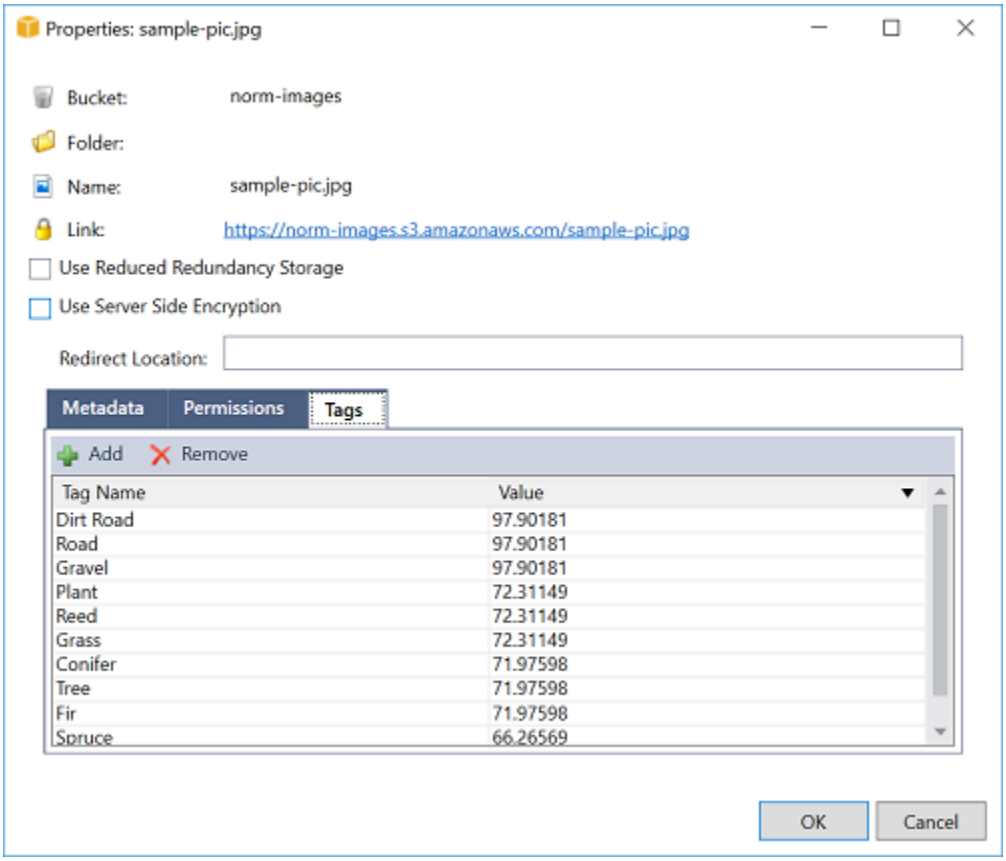
5. デプロイが成功したら、[イベントソース] タブに移動して、新しい関数にイベントを送信するように Amazon S3 を設定します。
6. [イベントソース] タブで、[追加] ボタンを選択し、Lambda 関数に接続する Amazon S3 バケットを選択します。

> **Note**
>
> バケットは Lambda 関数と同じ AWS リージョンにある必要があります。

関数をテストする

これで、関数がデプロイされ、イベントのソースとして S3 バケットが関数に対して設定されたので、S3 バケットブラウザを AWS Explorer から、選択したバケットについて開きます。次に、いくつかのイメージをアップロードします。

アップロードが完了すると、使用する関数のビューからログを調べることで、関数が実行されたことを確認できます。または、バケットブラウザ内のイメージを右クリックし、[Properties (プロパティ)] を選択します。[Tags (タグ)] タブで、使用するオブジェクトに適用されたタグを表示できます。



チュートリアル: AWS Lambda で Amazon ログ記録フレームワークを使用してアプリケーションログを作成する

「Amazon CloudWatch Logs」を使用して、アプリケーションのログのモニタリング、保存、アクセスを行うことができます。CloudWatch Logs にログデータを取得するには、 AWS SDK を使用するか、CloudWatch Logs エージェントをインストールして特定のログフォルダをモニタリングします。CloudWatch Logs は、いくつかの一般的な .NET ログ記録フレームワークと統合されており、ワークフローを簡素化します。

CloudWatch Logs と .NET ログ記録フレームワークの使用を開始するには、適切な NuGet パッケージと CloudWatch Logs 出力ソースをアプリケーションに追加し、通常どおりにログ記録ライブラリを使用します。これにより、アプリケーションは .NET フレームワークを使用してメッセージをログに記録し、CloudWatch Logs に送信して、アプリケーションのログメッセージを CloudWatch Logs コンソールに表示できるようになります。CloudWatch Logs コンソールから、アプリケーションのログメッセージに基づくメトリクスおよびアラームをセットアップすることもできます。

サポートされている .NET ログ記録フレームワークは次のとおりです。

- NLog: 表示するには、「nuget.org NLog package」を参照してください。
- Log4net: 表示するには、「nuget.org Log4net package」を参照してください。
- ASP.NET Core ログ記録フレームワーク: 表示するには、「nuget.org ASP.NET Core logging Framework package」を参照してください。

以下は、AWS.Logger.NLogNuGet パッケージと AWS ターゲットを に追加することで、CloudWatch Logs とコンソールの両方をログメッセージの出力として有効にする NLog.configファイルの例ですNLog.config。

```
<?xml version="1.0" encoding="utf-8" ?>
<nlog xmlns="http://www.nlog-project.org/schemas/NLog.xsd"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      throwExceptions="true">
  <targets>
    <target name="aws" type="AWSTarget" logGroup="NLog.ConfigExample" region="us-
east-1"/>
    <target name="logfile" xsi:type="Console" layout="${callsite} ${message}" />
  </targets>
  <rules>
    <logger name="*" minlevel="Info" writeTo="logfile,aws" />
  </rules>
</nlog>
```

ログ記録プラグインはすべて 上に構築 AWS SDK for .NET され、SDK と同様のプロセスで AWS 認証情報を認証します。次の例は、ログプラグインの認証情報が CloudWatch Logs にアクセスするために必要なアクセス許可を示しています。

> **Note**
>
> AWS .NET ログ記録プラグインはオープンソースプロジェクトです。追加情報、サンプル、および手順については、AWS Logging .NET GitHub リポジトリのサンプルと手順に関するトピックを参照してください。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:PutLogEvents",
        "logs:DescribeLogGroups"
      ],
      "Resource": [
        "arn:aws:logs:*:*:*"
      ]
    }
  ]
}
```

へのデプロイAWS

Toolkit for Visual Studio は、AWS Elastic Beanstalk コンテナまたは CloudFormation スタックへのアプリケーションのデプロイをサポートします。

> **Note**
>
> Visual Studio Express Edition を使用している場合:
>
> - Docker CLI を使用して、アプリケーションを Amazon ECS コンテナにデプロイできます。
> - AWS マネジメントコンソールを使用して、アプリケーションを Elastic Beanstalk コンテナにデプロイできます。
>
> Elastic Beanstalk デプロイでは、まずウェブデプロイパッケージを作成する必要があります。詳細については、「Visual Studio でウェブデプロイパッケージを作成する方法」を参照してください。Amazon ECS デプロイには、Docker イメージが必要です。詳細については、「Visual Studio Tools for Docker」を参照してください。

トピック

- Visual Studio で使用する Publish to AWS の操作
- .NET Core CLI を使用した AWS Lambda プロジェクトのデプロイ
- AWS Toolkit for Visual Studio AWS Elastic Beanstalk と Amazon Q を使用した Visual Studio でのへのデプロイ
- Amazon EC2 Container Service へのデプロイ

Visual Studio で使用する Publish to AWS の操作

Publish to AWS は、.NET アプリケーションを AWS デプロイターゲットに発行するのに役立つインタラクティブなデプロイエクスペリエンスで、.NET Core 3.1 以降をターゲットとするアプリケーションをサポートしています。Publish to AWS を使用することで、以下のデプロイ機能を IDE から直接利用できるようになるため、Visual Studio 内のワークフローを維持できます。

- クリック 1 つでアプリケーションをデプロイできます。

- アプリケーションに基づいたデプロイのレコメンデーション。
- デプロイ先の環境 (デプロイターゲット) に関連する、必要な Dockerfile を自動作成します。
- デプロイターゲットの要求に応じて、アプリケーションの構築とパッケージングに関する設定を最適化します。

> **Note**
>
> .NET Framework アプリケーションの発行の詳細については、「Elastic Beanstalk での .NET アプリケーションの作成とデプロイ」ガイドを参照してください。
> .NET CLI から Publish to AWS にアクセスすることもできます。詳細は、「AWS での .NET アプリケーションのデプロイ」ガイドを参照してください。

トピック

前提条件

.NET アプリケーションを AWS のサービスに正常に発行するには、ローカルデバイスに以下をインストールします。

- .NET Core 3.1 以降 (.NET5 と.NET6 を含む): これらの製品の詳細およびダウンロードに関する情報については、Microsoft ダウンロードサイトを参照してください。
- Node.js 14.x 以降のバージョン: Node.js は AWS Cloud Development Kit (AWS CDK) を実行するために必要です。Node.js のダウンロード、または詳細の入手については、Node.js ダウンロードサイトにアクセスしてください。

> **Note**
>
> Publish to AWS は、アプリケーションとそのすべてのデプロイインフラストラクチャを単一のプロジェクトとしてデプロイするのに AWS CDK を利用します。AWS CDK の詳細については、「Cloud Development Kit ガイド」を参照してください。

- (オプション) Docker は、Amazon ECS などのコンテナベースのサービスにデプロイするときに使用されます。詳細および Docker のダウンロードについては、Docker ダウンロードサイトを参照してください。

サポートされるアプリケーションタイプ

新しいターゲットまたは既存のターゲットに発行する前に、まず Visual Studio で以下のいずれかのプロジェクトタイプを作成するか開きます。

- ASP.NET Core Applications
- .NET コンソールアプリケーション
- Blazor WebAssembly アプリケーション

アプリケーションを AWS ターゲットに発行する

新しいターゲットに発行する場合、Publish to AWS がレコメンデーションを作成したり一般的な設定を使用したりしてプロセスを案内します。以前に設定したターゲットに発行する必要がある場合、設定は保存されて調整できるほか、ワンクリックですぐにデプロイできます。

> **Note**
>
> Toolkits と .NET CLI Server の統合:
> 公開すると、.NET サーバープロセスが localhost で起動され、公開プロセスが実行されます。

新しいターゲットに発行

以下では、新しいターゲットに発行する場合における Publish to AWS のデプロイ設定方法について説明します。

1. [AWS Explorer] から [認証情報] ドロップダウンメニューを展開し、デプロイに必要なリージョンと AWS のサービスに対応する AWS プロファイルを選択します。
2. [リージョン] ドロップダウンメニューを展開し、デプロイに必要な AWS サービスを含む AWS リージョンを選択します。

3. Visual Studio の [ソリューション エクスプローラー] ペインから、プロジェクトの名前のコンテキスト (右クリック) メニューを開いて、[Publish to AWS] を選択します。Publish to AWS が開きます。
4. Publish to AWS から[新しいターゲットに発行] を選択し、新しいデプロイを設定します。

> **Note**
>
> デフォルトのデプロイ認証情報を変更するには、Publish to AWS の [認証情報] セクションの横にある [編集] リンクを選択またはクリックします。
> ターゲット設定プロセスをバイパスするには、[既存のターゲットに発行] を選択し、以前のデプロイターゲットのリストから希望する設定を選択します。

5. [ターゲットを発行] ペインから、アプリケーションのデプロイを管理する AWS サービスを選択します。
6. 設定に問題がなければ、[発行] を選択してデプロイプロセスを開始します。

> **Note**
>
> デプロイを開始すると、以下のステータスの更新情報が Publish to AWS に表示されます。
>
> - デプロイプロセス中には、デプロイの進行状況が Publish to AWS に表示されます。
> - デプロイプロセス後には、デプロイが成功したか失敗したかが Publish to AWS に表示されます。
> - デプロイが成功すると、作成されたリソースに関する追加情報が [リソース] パネルに表示されます。この情報は、アプリケーションのタイプとデプロイ設定によって異なります。

既存のターゲットへの発行

以下では、.NET アプリケーションを既存の AWS ターゲットに再発行する方法について説明します。

1. [AWS Explorer] から [認証情報] ドロップダウンメニューを展開し、デプロイに必要なリージョンと AWS のサービスに対応する AWS プロファイルを選択します。

2. [リージョン] ドロップダウンメニューを展開し、デプロイに必要な AWS サービスを含む AWS リージョンを選択します。
3. Visual Studio の [ソリューション エクスプローラー] ペインでプロジェクトの名前を右クリックし、[Publish to AWS] を選択して [Publish to AWS] を開きます。
4. Publish to AWS から [既存のターゲットに発行] を選択し、既存のターゲットのリストからデプロイ環境を選択します。

> **Note**
>
> 最近 AWS クラウドにアプリケーションを発行した場合、それらのアプリケーションが Publish to AWS に表示されます。

5. アプリケーションをデプロイする発行ターゲットを選択し、[発行] をクリックしてデプロイプロセスを開始します。

.NET Core CLI を使用した AWS Lambda プロジェクトのデプロイ

AWS Toolkit for Visual Studio には、Visual Studio 用の includes AWS Lambda .NET Core プロジェクトテンプレートが含まれています。.NET コアコマンドラインインターフェイス (CLI) を使用して、Visual Studio に組み込まれた Lambda 関数をデプロイできます。

トピック

前提条件

.NET Core CLI を使用して Lambda 関数をデプロイする前に、次の前提条件を満たす必要があります。

- Visual Studio 2015 Update 3 がインストールされていることを確認してください。
- .NET Core for Widnows をインストールします。

- .NET Core CLI をセットアップして Lambda を操作します。詳細については、AWS Lambda デベロッパーガイドの「 NET Core CLI」を参照してください。
- Toolkit for Visual Studio をインストールします。詳細については、「のインストール AWS Toolkit for Visual Studio」を参照してください。

関連トピック

.NET Core CLI を使用して Lambda 関数をデプロイする場合、次の関連トピックが役立ちます。

- Lambda 関数の詳細については、「 AWS Lambda デベロッパーガイド」のAWS 「Lambda とは」を参照してください。
- Visual Studio で Lambda 関数を作成する方法については、「AWS Lambda」を参照してください。
- Microsoft .NET Core の詳細については、Microsoft オンラインドキュメントの「.NET Core」を参照してください。

.NET Core CLI を介して使用可能な Lambda コマンドを一覧表示する

.NET Core CLI を介して使用できる Lambda コマンドを一覧表示するには、次の手順を実行します。

1. コマンドプロンプトウインドウを開いて、Visual Studio の .NET Core Lambda プロジェクトが含まれるフォルダに移動します。
2. `dotnet lambda --help` と入力します。

```
C:\Lambda\AWSLambda1\AWSLambda1>dotnet lambda --help AWS Lambda Tools for .NET Core
 functions
    Project Home: https://github.com/aws/aws-lambda-dotnet
    .
    Commands to deploy and manage Lambda functions:
    .
            deploy-function         Deploy the project to Lambda
            invoke-function         Invoke the function in Lambda with an optional
 input
            list-functions          List all of your Lambda functions
            delete-function         Delete a Lambda function
            get-function-config     Get the current runtime configuration for a Lambda
 function
```

```
            update-function-config  Update the runtime configuration for a Lambda
function
   .
   Commands to deploy and manage AWS serverless applications using AWS CloudFormation:
   .
            deploy-serverless       Deploy an AWS serverless application
            list-serverless         List all of your AWS serverless applications
            delete-serverless       Delete an AWS serverless application
   .
   Other Commands:
   .
            package                 Package a Lambda project into a .zip file ready for
deployment
   .
   To get help on individual commands, run the following:

            dotnet lambda help <command>
```

.NET Core CLI から .NET Core Lambda プロジェクトを発行する

次の手順では、Visual Studio で AWS Lambda .NET Core 関数を作成していることを前提としています。

1. コマンドプロンプトウィンドウを開いて、Visual Studio の .NET Core Lambda プロジェクトが含まれるフォルダに移動します。
2. `dotnet lambda deploy-function` と入力します。
3. プロンプトされたら、デプロイする関数の名前を入力します。新しい名前または既存の関数の名前を入力できます。
4. プロンプトが表示されたら、 AWS リージョン (Lambda 関数がデプロイされるリージョン) を入力します。
5. プロンプトがされたら、関数が実行されるときに Lambda が継承する IAM ロールを選択または作成します。

正常に完了すると、[New Lambda function created (新しい Lambda 関数が作成されました)] のメッセージが表示されます。

```
C:\Lambda\AWSLambda1\AWSLambda1>dotnet lambda deploy-function
Executing publish command
```

```
... invoking 'dotnet publish', working folder 'C:\Lambda\AWSLambda1\AWSLambda1\bin
\Release\netcoreapp1.0\publish'
... publish: Publishing AWSLambda1 for .NETCoreApp,Version=v1.0
... publish: Project AWSLambda1 (.NETCoreApp,Version=v1.0) will be compiled because
 expected outputs are missing
... publish: Compiling AWSLambda1 for .NETCoreApp,Version=v1.0
... publish: Compilation succeeded.
... publish:     0 Warning(s)
... publish:     0 Error(s)
... publish: Time elapsed 00:00:01.2479713
... publish:
... publish: publish: Published to C:\Lambda\AWSLambda1\AWSLambda1\bin\Release
\netcoreapp1.0\publish
... publish: Published 1/1 projects successfully
Zipping publish folder C:\Lambda\AWSLambda1\AWSLambda1\bin\Release
\netcoreapp1.0\publish to C:\Lambda\AWSLambda1\AWSLamb
da1\bin\Release\netcoreapp1.0\AWSLambda1.zip
Enter Function Name: (AWS Lambda function name)
DotNetCoreLambdaTest
Enter AWS Region: (The region to connect to AWS services)
us-west-2
Creating new Lambda function
Select IAM Role that Lambda will assume when executing function:
    1) lambda_exec_LambdaCoreFunction
    2) *** Create new IAM Role ***
1
New Lambda function created
```

既存の関数をデプロイする場合、デプロイ機能は AWS リージョンのみを求めます。

```
C:\Lambda\AWSLambda1\AWSLambda1>dotnet lambda deploy-function
Executing publish command
Deleted previous publish folder
... invoking 'dotnet publish', working folder 'C:\Lambda\AWSLambda1\AWSLambda1\bin
\Release\netcoreapp1.0\publish'
... publish: Publishing AWSLambda1 for .NETCoreApp,Version=v1.0
... publish: Project AWSLambda1 (.NETCoreApp,Version=v1.0) was previously compiled.
 Skipping compilation.
... publish: publish: Published to C:\Lambda\AWSLambda1\AWSLambda1\bin\Release
\netcoreapp1.0\publish
... publish: Published 1/1 projects successfully
Zipping publish folder C:\Lambda\AWSLambda1\AWSLambda1\bin\Release
\netcoreapp1.0\publish to C:\Lambda\AWSLambda1\AWSLamb
```

```
da1\bin\Release\netcoreapp1.0\AWSLambda1.zip
Enter Function Name: (AWS Lambda function name)
DotNetCoreLambdaTest
Enter AWS Region: (The region to connect to AWS services)
us-west-2
Updating code for existing function
```

デプロイされた Lambda 関数は、すぐに使用できる状態になっています。詳細については、「AWS Lambda の使用例」を参照してください。

Lambda は、ユーザーに代わって Lambda 関数を自動でモニタリングし、Amazon CloudWatch からメトリクスを報告します。Lambda 関数をモニタリングおよびトラブルシューティングするには、Amazon CloudWatch を使用した AWS Lambda 関数のトラブルシューティングとモニタリング」を参照してください。

AWS Toolkit for Visual Studio AWS Elastic Beanstalk と Amazon Q を使用した Visual Studio での へのデプロイ

AWS Elastic Beanstalk は、アプリケーションの AWS リソースのプロビジョニングプロセスを簡素化するサービスです。Elastic Beanstalk は、アプリケーションのデプロイに必要なすべての AWS インフラストラクチャを提供します。このインフラストラクチャには次のものが含まれます。

- 実行可能ファイルとアプリケーションのコンテンツをホストする Amazon EC2 インスタンス。
- Amazon EC2 インスタンスの数を適切に維持することでアプリケーションをサポートする Auto Scaling グループ。
- 受信トラフィックを帯域幅が最も広い Amazon EC2 インスタンスにルーティングする Elastic Load Balancing ロードバランサー。

このユーザーガイドトピックでは、 AWS Toolkit with Amazon Q で Elastic Beanstalk ウィザードを使用する方法について説明します。Elastic Beanstalk に固有の詳細については、「 AWS Elastic Beanstalkデベロッパーガイド」を参照してください。 AWS Toolkit with Amazon Q の Elastic Beanstalk ウィザードについては、次のトピックセクションで説明します。

トピック

- .Traditional ASP NET アプリケーションを Elastic Beanstalk にデプロイします。
- Elastic Beanstalk への ASP.NET Core アプリケーションのデプロイ (レガシー)

- アプリケーションの AWS セキュリティ認証情報を指定する方法
- Elastic Beanstalk 環境にアプリケーションを再発行する方法 (レガシー)
- Elastic Beanstalk アプリケーションのカスタムデプロイ
- カスタム ASP.NET Core Elastic Beanstalk デプロイ
- .NET および Elastic Beanstalk 用の複数アプリケーションのサポート

.Traditional ASP NET アプリケーションを Elastic Beanstalk にデプロイします。

このセクションでは、Elastic Beanstalk を介してアプリケーションをデプロできるように、Toolkit for Visual Studio の一部として提供される [Publish to Elastic Beanstalk] ウィザードの使用方法を記述します。演習を行う場合は、ウェブアプリケーションのスタータープロジェクトのインスタンスを使うことができます。これは Visual Studio に組み込まれています。または独自のプロジェクトを使用することもできます。

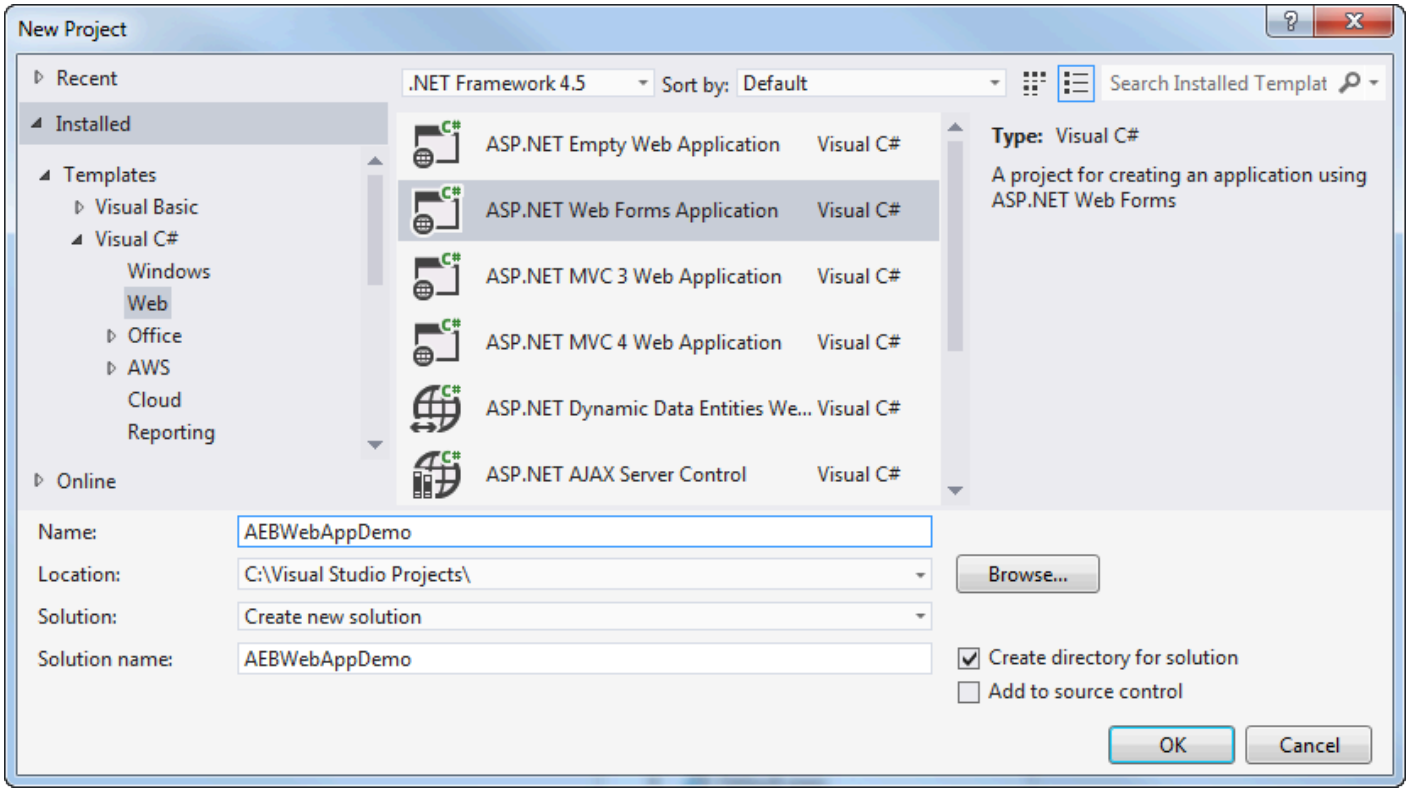
> **Note**
>
> ウィザードでは ASP.NET Core アプリケーションのデプロイもサポートしています。ASP.NET Core については、「AWS .NET Deployment tool」ガイドと、目次にある更新された「AWSへのデプロイ」を参照してください。

> **Note**
>
> [Publish to Elastic Beanstalk] ウィザードを使用するには、Web Deploy をダウンロードしてインストールする必要があります。ウィザードでは、Web Deploy を用いて Internet Information Services (IIS) ウェブサーバーにウェブアプリケーションやウェブサイトをデプロイします。

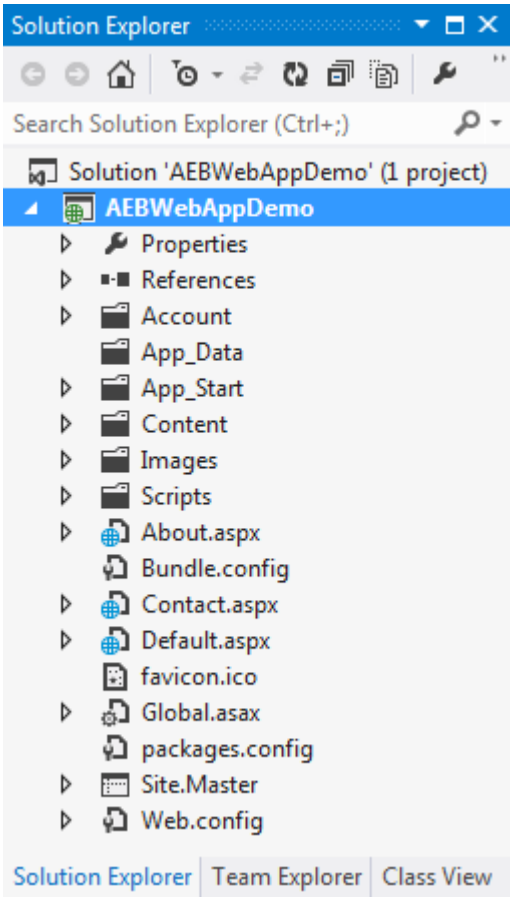
サンプルウェブアプリケーションのスタータープロジェクトを作成するには

1. Visual Studio の [File (ファイル)] メニューで [New (新規)] を選択し、[Project (プロジェクト)] を選択します。
2. [New Project (新規プロジェクト)] ダイアログボックスのナビゲーションペインで、[Installed (インストール済み)]、[Templates (テンプレート)]、[Visual C#] の順に展開し、[Web] を選択します。

3. ウェブプロジェクトテンプレートのリストの中から、Web と Application が説明に含まれるテンプレートのいずれかを選択します。この例では、[ASP.NET Web Forms Application (ASP.NET Web Forms アプリケーション)] を選択します。

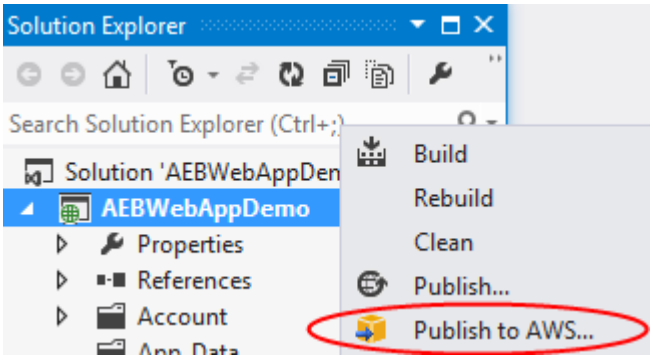


4. [Name (名前)] ボックスに、「AEBWebAppDemo」と入力します。
5. [Location] ボックスで、開発マシンのソリューションフォルダへのパスを入力するか、[Browse] を選択し、ソリューションフォルダを参照して選択し、[Select Folder] を選択します。
6. [Create directory for solution (ソリューションのディレクトリの作成)] が選択されていることを確認します。[Solution] (ソリューション) ドロップダウンリストで、[Create new solution] (新しいソリューションの作成) が選択されていることを確認し、[OK] を選択します。Visual Studio を用いて、ASP.NET Web Forms Application プロジェクトテンプレートをベースにしたソリューションやプロジェクトを作成できます。新しいソリューションやプロジェクトが表示される Solution Explorer が、Visual Studio の画面に表示されます。

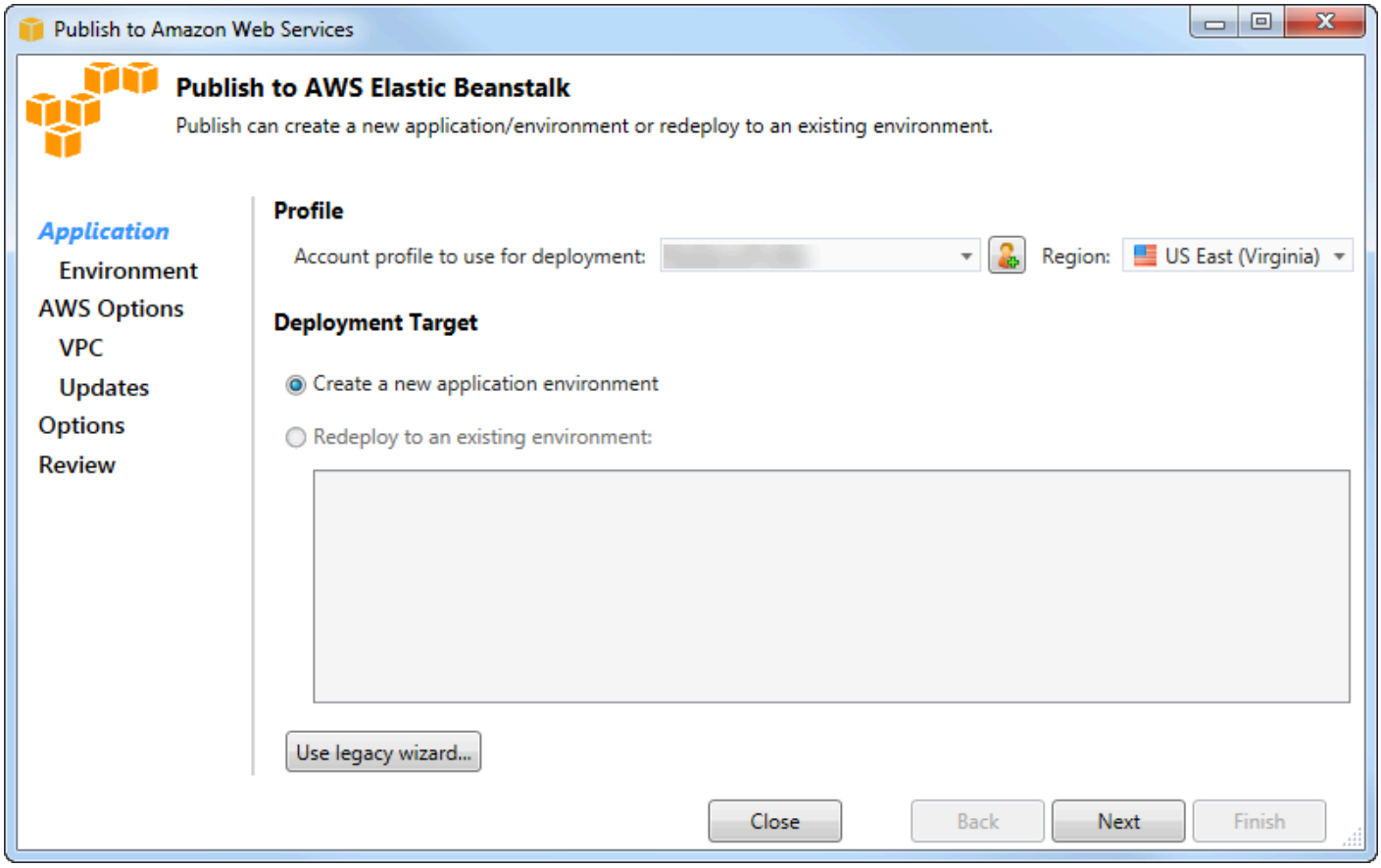


[Publish to Elastic Beanstalk] ウィザードを使用してアプリケーションをデプロイするには

1. Solution Explorer で、前のセクションで作成したプロジェクトの AEBWebAppDemo プロジェクトフォルダのコンテキスト (右クリック) メニューを開くか、独自のアプリケーションのプロジェクトフォルダのコンテキストメニューを開き、Elastic Beanstalk AWS に発行を選択します。



   [Publish to Elastic Beanstalk] ウィザードが表示されます。



2. プロファイルで、デプロイに使用するアカウントプロファイルのドロップダウンリストから、デプロイに使用する AWS アカウントプロファイルを選択します。

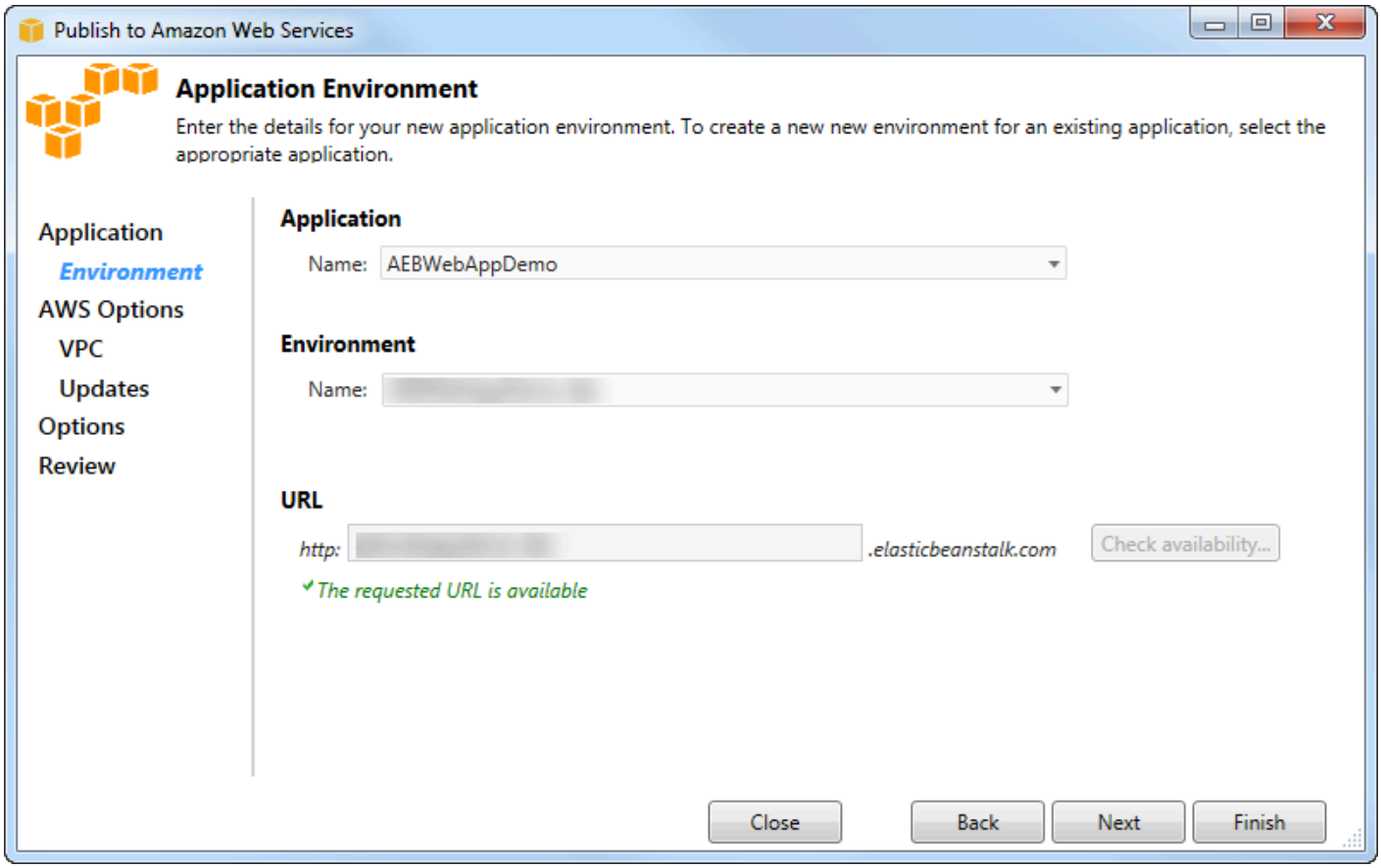
   オプションで、使用する AWS アカウントがあり、その AWS アカウントプロファイルをまだ作成していない場合は、プラス記号 (+) が付いたボタンを選択して、 AWS アカウントプロファイルを追加できます。

3. [Region] (リージョン) ドロップダウンリストから、Elastic Beanstalkをアプリケーションにデプロイしたいリージョンを選択します。

4. [Deployment Target] (デプロイ先) では、[Create a new application environment] (新しいアプリケーション環境を作成) を選択してアプリケーションの初期デプロイを行うことも、[Redeploy to an existing environment] (既存の環境に再デプロイ) を選択してデプロイ済みアプリケーションの再デプロイを行うかこともできます。(前回のデプロイをウィザードまたは [Standalone Deployment Tool] (スタンドアロンデプロイツール) のどちらで実行していてもかまいません。) [Redeploy to an existing environment] を選択した場合は、現在稼働中の前回のデプロイに関する情報をウィザードが取得するのに時間を要することがあります。

> **Note**
>
> [Redeploy to an existing environment (既存の環境に再デプロイする)] を選択した場合は、リストの中から環境を選択し、[Next (次へ)] を選択すると、[Application Options (アプリケーションオプション)] ページを直接表示できます。この方法でページを表示した場合は、本セクションで口述する [Application Options (アプリケーションオプション)] ページの使用方法について説明している箇所までスキップしてください。

5. [次へ] をクリックします。

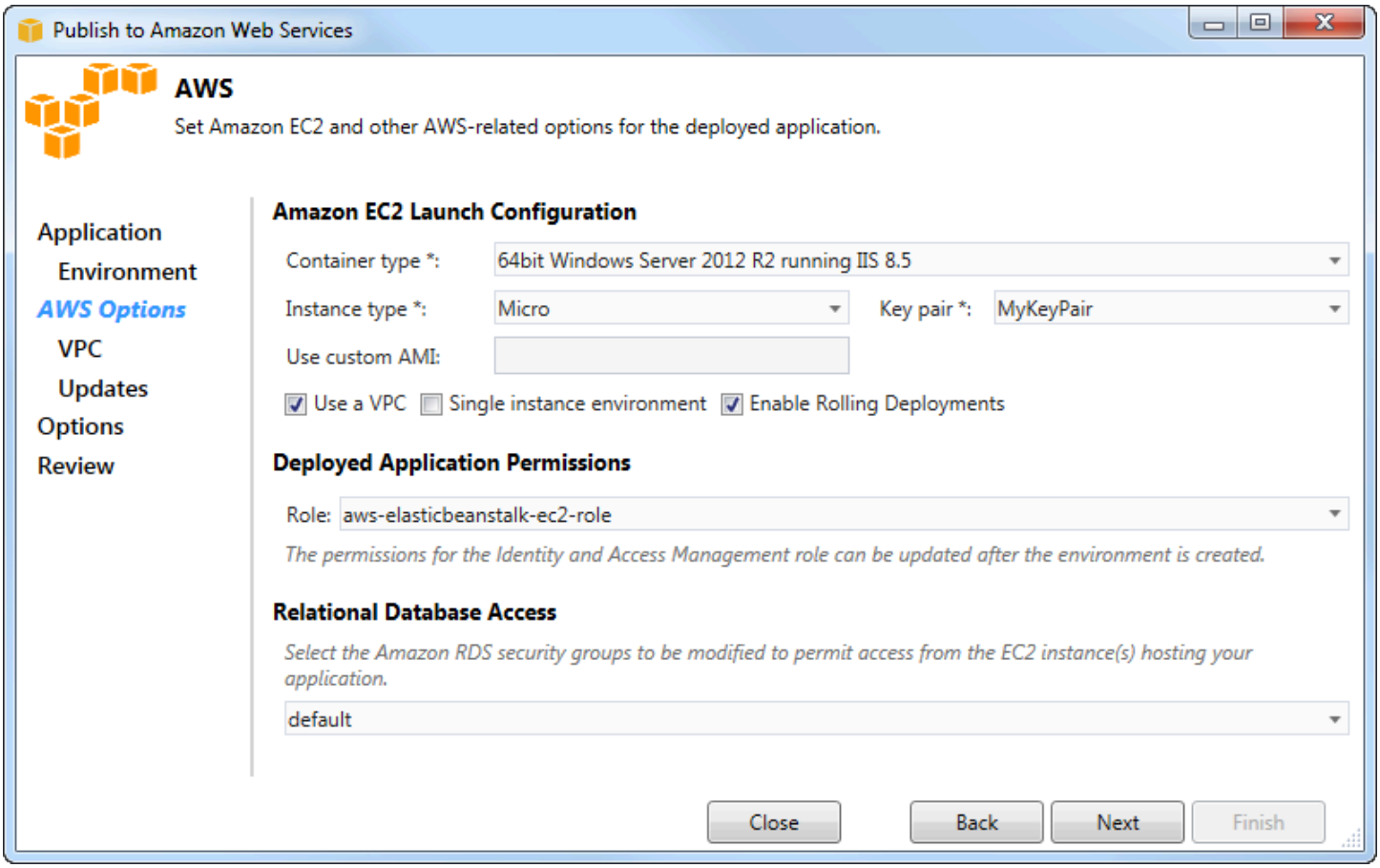


6. [Application Environment] (アプリケーション環境) ページの [Application] (アプリケーション) エリアにある [Name] (名前) ドロップダウンリストに、推奨されるデフォルトのアプリケーション名が表示されます。ドロップダウンリストの別の名前を選択することで、デフォルト名を変更できます。

7. [Environment] (環境) エリアの [Name] (名前) ドロップダウンリストで、Elastic Beanstalk 環境の名前を入力します。このコンテキストでは、環境という用語は、アプリケーションのインフラストラクチャ Elastic Beanstalk プロビジョニングを意味します。推奨されるデフォルト名がドロップダウンリストに既に表示されている場合があります。推奨されるデフォルト名が表示されてい

ない場合は自身で入力するか、他に選択候補がある場合は、ドロップダウンリストからいずれか 1 つを選択します。環境の名前は 23 文字より長くすることはできません。

8. [URL] エリアでは、ウェブアプリケーションの URL として使用される、デフォルトの `.elasticbeanstalk.com` のサブドメイン候補が表示されます。新しいサブドメイン名を入力することで、デフォルトのサブドメインを変更できます。
9. [Check availability] を選択して、ウェブアプリケーションの URL が使用されていないことを確認します。
10. ウェブアプリケーションの URL が使用可能な場合は、[Next] を選択します。



1. [AWS オプション] ページの [Amazon EC2 起動設定] にある [コンテナタイプ] ドロップダウンリストから、アプリケーションで使用する Amazon マシンイメージ (AMI) を選択します。
2. [Instance type] (インスタンスタイプ) ドロップダウンリストで、使用する Amazon EC2 インスタンスタイプを指定します。この例では、[Micro (マイクロ)] の使用をお勧めします。これにより、インスタンスの実行に関連するコストが最小化されます。Amazon EC2 の料金の詳細については、「EC2 料金」のページを参照してください。

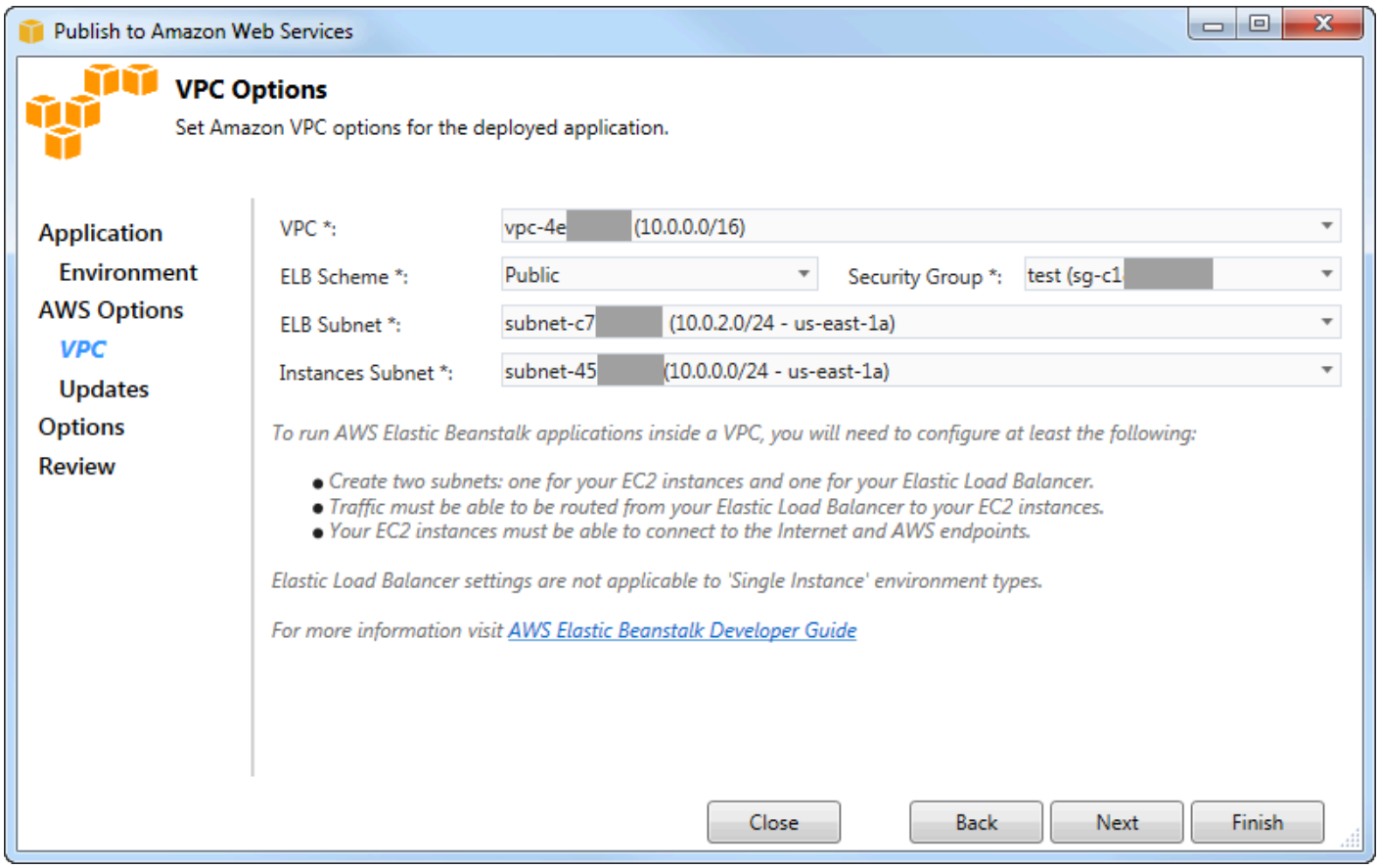
3. [Key pair] (キーペア) ドロップダウンリストで、アプリケーションで使用するインスタンスにサインインする際の Amazon EC2 インスタンスキーペアを選択します。
4. オプションとして、[Use custom AMI] (カスタム AMI を使用) ボックスで、[Container type] (カスタムタイプ) ドロップダウンリストで指定した AMI を上書きするカスタム AMI を指定できます。カスタム AMI の作成方法の詳細については、AWS Elastic Beanstalk デベロッパーガイドの「カスタム AMI の使用」と「Amazon EC2 インスタンスからの AMI の作成」を参照してください。
5. VPC 内でインスタンスを起動する場合は、[Use a VPC] ボックスを選択します。
6. オプションとして、Amazon EC2 インスタンスを 1 つだけ起動してアプリケーションをそのインスタンスにデプロイしたい場合、[Single instance environment] (単一インスタンス環境) ボックスを選択します。

   このボックスを選択する場合、 Elastic BeanstalkによりAuto Scaling グループが作成されますが、設定はされません。Auto Scaling グループを後で設定したい場合、 AWS マネジメントコンソールを使用できます。
7. オプションとして、アプリケーションをデプロイするインスタンスを条件で制御する場合は、[Enable Rolling Deployments] (ローリングデプロイを有効にする) ボックスを選択します。このボックスは [Single instance environment (単一のインスタンス環境)] ボックスを選択していない場合のみ選択できます。
8. アプリケーションが Amazon S3 や DynamoDB などの AWS サービスを使用している場合、認証情報を提供する最善の方法は IAM ロールを使用することです。[Deployed Application Permissions] (デプロイ済みアプリケーションアクセス許可) エリアでは、ウィザードが環境を起動する際に使用するロールとして既存の IAM ロールを選択するか、または新しくロールを作成することができます。を使用するアプリケーション AWS SDK for .NET は、 AWS サービスにリクエストを行うときに、この IAM ロールによって提供される認証情報を自動的に使用します。
9. アプリケーションから Amazon RDS データベースにアクセスしている場合は、[Relational Database Access] エリアのドロップダウンリストで、ウィザードが更新する Amazon RDS セキュリティグループの横のボックスを選択することで、Amazon EC2 インスタンスからそのデータベースにアクセスできます。
10. [次へ] を選択します。

    - [Use a VPC] を選択した場合は、[VPC Options] ページが表示されます。
    - [Enable Rolling Deployments] (ローリングデプロイを有効にする) を選択し、かつ [Use a VPC] (VPC を使用) を選択しなかった場合は、[Rolling Deployments] ページが表示されます。本セクションで後述している [Rolling Deployments (ローリングデプロイ)] ページの使用方法について説明している箇所までスキップしてください。

- [Use a VPC] (VPC を使用) および [Enable Rolling Deployments] (ローリングデプロイを有効にする) を選択しなかった場合は、[Application Options] (アプリケーションオプション) ページが表示されます。本セクションで後述されている [アプリケーションオプション] ページの使用方法について説明している箇所までスキップしてください。

11[Use a VPC] を選択した場合は、[VPC Options] ページの情報を指定して VPC でアプリケーションを起動します。



VPC は既に作成されている必要があります。Toolkit for Visual Studio で VPC を作成した場合、Toolkit for Visual Studio によってこのページが自動的に設定されます。AWS マネジメントコンソールで VPC を作成した場合、VPC に関する情報をこのページに入力します。

VPC へのデプロイに際して特に考慮すべき事項

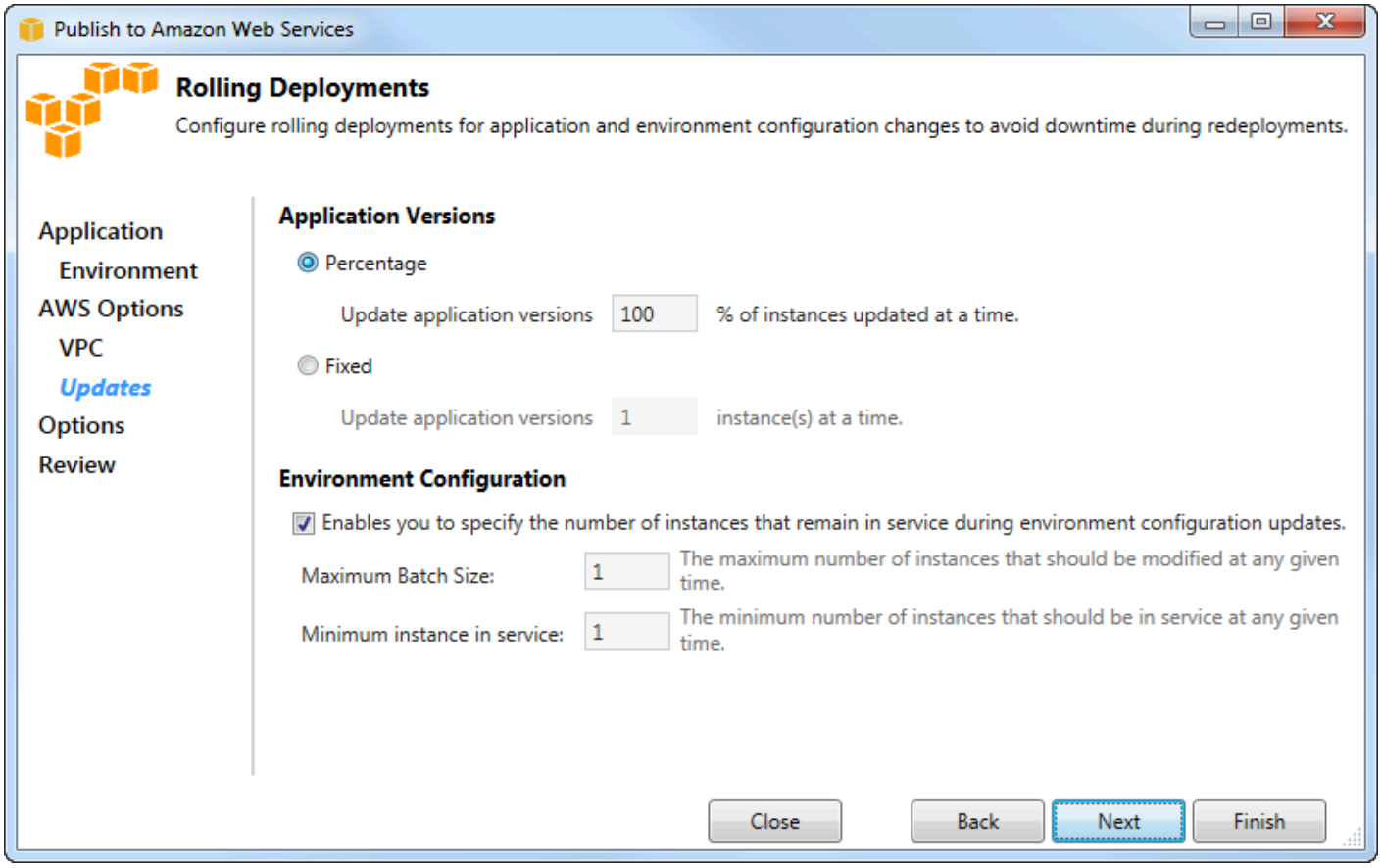
- VPC には、少なくとも 1 つのパブリックサブネットと 1 つのプライベートサブネットが必要です。
- [ELB Subnet] (ELB サブネット) ドロップダウンリストでパブリックサブネットを設定します。Toolkit for Visual Studio は、アプリケーションの Elastic Load Balancing ロードバランサーを

パブリックサブネットに向けてデプロイします。パブリックサブネットは、インターネットゲートウェイを指すエントリが含まれているルーティングテーブルに関連付けられています。インターネットゲートウェイは `igw-` で始まる ID を持つため (例 :`igw-83cddaex`)、インターネットゲートウェイを認識することができます。Toolkit for Visual Studio を使用して作成したパブリックサブネットには、パブリックとして識別するタグ値が付与されています。

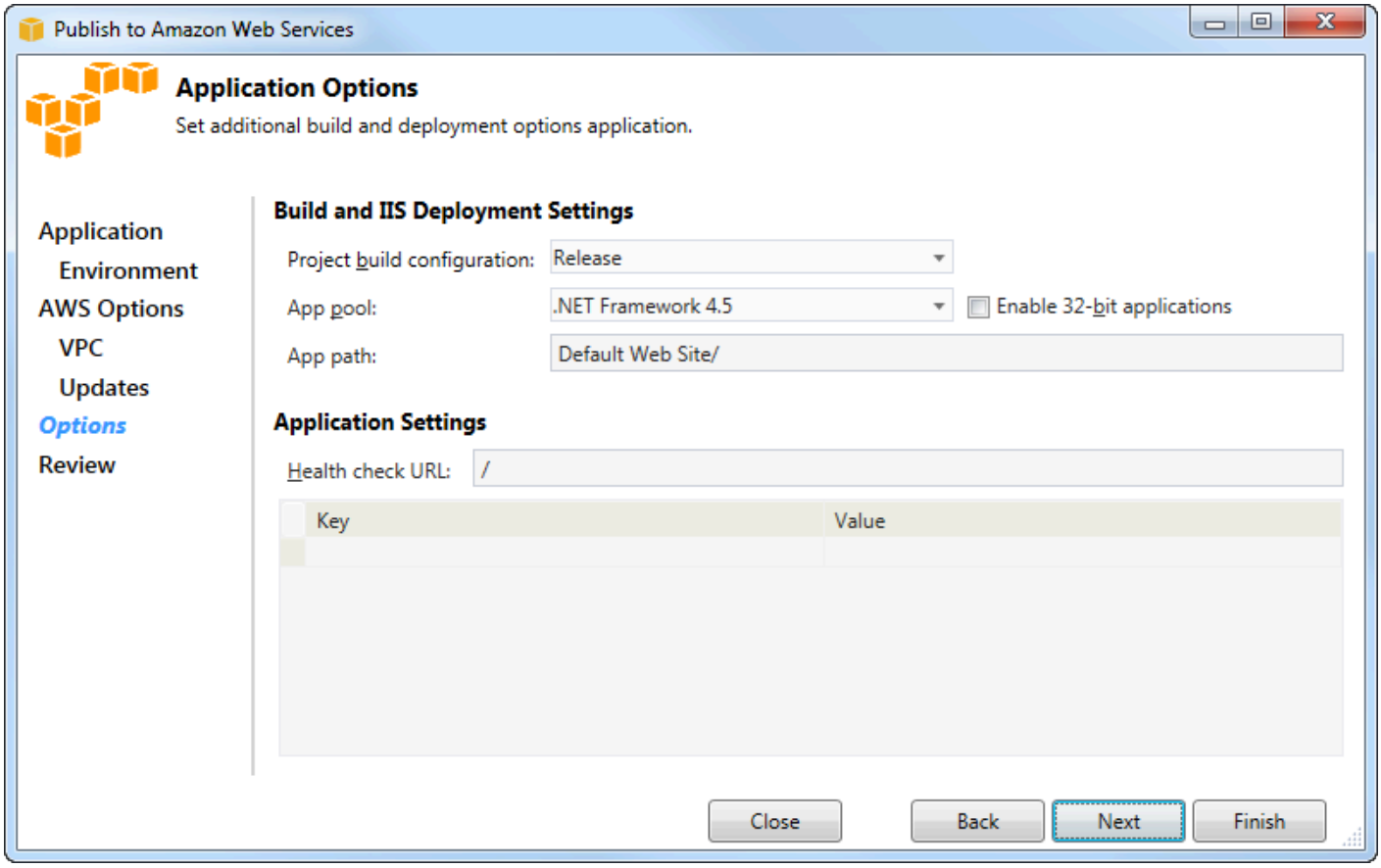
- [Instances Subnet] (インスタンスサブネット) ドロップダウンリストでプライベートサブネットを設定します。Toolkit for Visual Studio はアプリケーションのAmazon EC2 Eインスタンスをプライベートサブネットに向けてデプロイします。
- アプリケーションの Amazon EC2 インスタンスは、ネットワークアドレス変換 (NAT) を行うパブリックサブネットの Amazon EC2 インスタンスを介してプライベートサブネットからインターネットに通信します。通信を可能にするには、プライベートサブネットから NAT インスタンスにトラフィックが流れるようにするための VPC セキュリティグループが必要です。VPC セキュリティグループを [Security Group] ドロップダウンリストから指定します。

Elastic Beanstalkアプリケーションを VPC にデプロイする方法の詳細については、AWS Elastic Beanstalk デベロッパーガイドを参照してください。

1. VPC Options ページですべての情報を入力してから、[Next] を選択します。
   - [Enable Rolling Deployments] を選択した場合は、[Rolling Deployments] ページが表示されます。
   - [Enable Rolling Deployments] (ローリングデプロイを有効にする) を選択しなかった場合は、[Application Options] (アプリケーションオプション) ページが表示されます。本セクションで後述されている [アプリケーションオプション] ページの使用方法について説明している箇所までスキップしてください。
2. [Enable Rolling Deployments] (ローリングデプロイを有効にする) を選択した場合は、[Rolling Deployments] (ローリングデプロイ) ページの情報を設定して、新しいバージョンのアプリケーションをロードバランス環境のインスタンスにデプロイする際の方法を構成します。例えば、環境内に 4 つのインスタンスがあってインスタンスタイプを変更する場合、一度に 2 つのインスタンスを変更するように環境を設定できます。これにより、アプリケーションを停止することなく変更できます。



3. [Application Versions] (アプリケーションのバージョン) エリアで、一度にデプロイするインスタンスを全体の割合にするか実際の数にするかを制御するオプションを選択します。目的のパーセンテージまたは最大数を指定します。
4. オプションとして、デプロイ中にサービスを実行し続けるインスタンス数を指定する場合は、[Environment Configuration] (環境の設定) エリアのボックスを選択します。このボックスを選択した場合は、一度に変更するインスタンスの最大数、サービスを実行し続ける最小のインスタンス数、またはその両方を設定します。
5. [次へ] を選択します。
6. [Application Options] ページで、構築、Internet Information Services (IIS)、およびアプリケーション設定に関する情報を指定します。



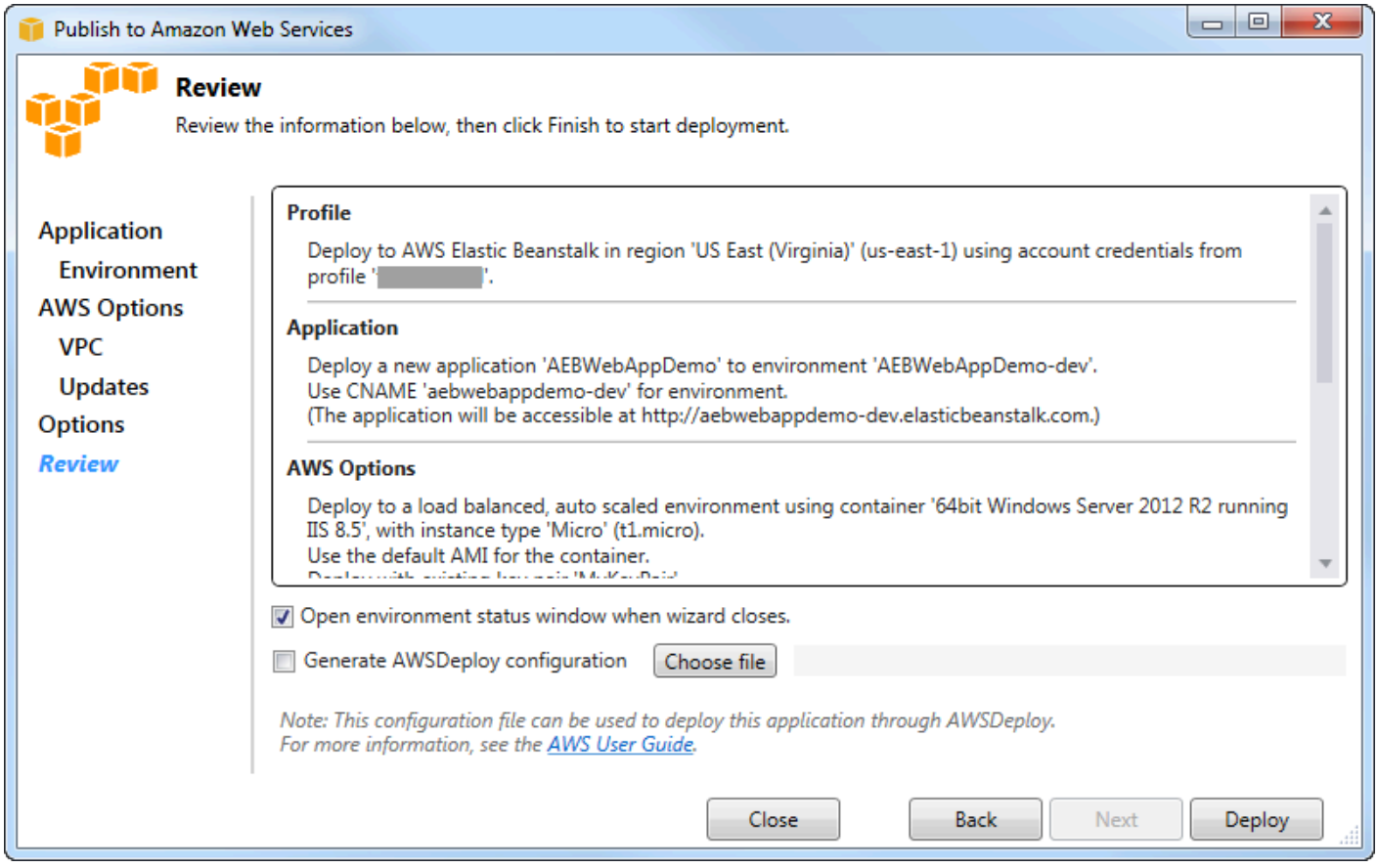
7. [Build and IIS Deployment Settings] (ビルドと IIS デプロイの設定) エリアの [Project build configuration] (プロジェクトビルド設定) ドロップダウンリストで、ターゲットビルド設定を選択します。ウィザードが発見できた場合は [Release (リリース)] が、発見できなかった場合はアクティブ設定がこのボックスに表示されます。
8. [App pool] (アプリケーションプール) ドロップダウンリストで、アプリケーションが必要とする .NET Framework のバージョンを選択します。適切な .NET Framework のバージョンが表示されています。
9. アプリケーションが 32 ビットの場合は、[Enable 32-bit applications] ボックスを選択します。
10. [App path] (アプリケーションパス) ボックスで、アプリケーションのデプロイに使用する IIS パスを指定します。デフォルトでは、[Default Web Site/ (デフォルトウェブサイト/)] が指定され、通常は `c:\inetpub\wwwroot` に変換されます。[Default Web Site (デフォルトウェブサイト)] 以外のパスを指定した場合は、ウィザードは指定したパスを指すリダイレクトを [Default Web Site/ (デフォルトウェブサイト/)] に配置します。
11. [Application Settings] (アプリケーションの設定) エリアの [Health check URL] (ヘルスチェック URL) ボックスに、ウェブアプリケーションが応答可能かを判別する際に確認する Elastic Beanstalk の URL を入力します。この URL はルートサーバー URL への相対 URL です。デフォ

ルトではルートサーバーの URL が指定されます。例えば、完全な URL が `example.com/site-is-up.html` である場合は `/site-is-up.html` と入力します。

12[Key] と [Value] には、アプリケーションの `Web.config` ファイルに追加したい任意のキーと値のペアを指定できます。

> **Note**
>
> 推奨されませんが、キーと値の エリアを使用して、アプリケーションを実行する AWS 認証情報を指定できます。推奨されるのは、[AWS Options] (オプション) ページの [Identity and Access Management Role] ドロップダウンリストで IAM ロールを指定する方法です。ただし、IAM ロールの代わりに AWS 認証情報を使用してアプリケーションを実行する必要がある場合は、キー行で AWSAccessKey を選択します。[値] 行にアクセスキーを入力します。AWSSecretKey についても同様のステップを繰り返します。

13[次へ] をクリックします。



14[Review] ページで、選択したオプションを確認し、[Open environment status window when wizard closes] ボックスを選択します。

15.すべてが正しい場合は、[Deploy (デプロイ)] を選択します。

> **Note**
>
> アプリケーションをデプロイすると、アプリケーションが使用する AWS リソースの料金が発生し、それはアクティブなアカウントに請求されます。

Visual Studio ステータスバーおよび [Output] (出力) ウィンドウに、デプロイに関する情報が表示されます。これには数分間かかる場合があります。デプロイが完了すると、確認メッセージが [Output (出力)] ウィンドウに表示されます。

16.デプロイを削除するには、 AWS Explorer で Elastic Beanstalk ノードを展開し、デプロイのサブノードのコンテキスト (右クリック) メニューを開き、削除を選択します。削除には数分かかる場合があります。

Elastic Beanstalk への ASP.NET Core アプリケーションのデプロイ (レガシー)

> **Important**
>
> このドキュメントでは、従来のサービスと機能について言及しています。更新されたガイドとコンテンツについては、「AWS .NET Deployment tool」ガイドと、目次にある更新された「AWSへのデプロイ」を参照してください。

AWS Elastic Beanstalk は、アプリケーションの AWS リソースをプロビジョニングするプロセスを簡素化するサービスです。 は、アプリケーションのデプロイに必要なすべての AWS インフラストラクチャ AWS Elastic Beanstalk を提供します。

Toolkit for Visual Studio では、Elastic Beanstalk AWS を使用して ASP.NET Core アプリケーションを にデプロイできます。ASP.NET Core は、ASP.NET をモジュラー型アーキテクチャで再設計したもので、依存関係のオーバーヘッドを最小限に抑え、アプリケーションのクラウド実行を合理化します。

AWS Elastic Beanstalk を使用すると、さまざまな言語でアプリケーションを簡単にデプロイできます AWS。EElastic Beanstalk は従来の ASP.NET アプリケーションおよび ASP.NET Core アプリケーションの両方をサポートしています。このトピックでは、ASP.NET Core アプリケーションのデプロイについて説明します。

デプロイウィザードを使用する

ASP.NET Core アプリケーションを Elastic Beanstalk にデプロイする最も簡単な方法は、Toolkit for VToolkit for VisualToolkit for Visual Studio を使用します。

以前にツールキットを使用して、従来の ASP.NET アプリケーションをデプロイしたことがある場合、ASP.NET Core の場合もよく似ていることがわかると思います。以下の手順では、デプロイをウォークスルーで体験します。

ツールキットを使用したことがない場合は、ツールキットをインストールした後に最初に行う必要があるのは、ツールキットに AWS 認証情報を登録することです。その方法の詳細については、「How to Specify the AWS Security Credentials for Your Application for Visual Studio ドキュメント」を参照してください。

ASP.NET Core ウェブアプリケーションをデプロイするには、 Solution Explorer でプロジェクトを右クリックし、パブリッシュ先 AWS... を選択します。

Publish to AWS Elastic Beanstalk deployment ウィザードの最初のページで、新しい Elastic Beanstalk アプリケーションの作成を選択します。Elastic Beanstalk アプリケーションは、Elastic Beanstalk コンポーネントの論理コレクションで、環境、バージョン、環境設定などがあります。デプロイウィザードでアプリケーションが生成されます。このアプリケーションには、さらにアプリケーションバージョンと環境のコレクションが含まれます。環境には、アプリケーションバージョンを実行する実際の AWS リソースが含まれています。アプリケーションをデプロイするたびに、新しいアプリケーションバージョンが作成され、ウィザードでそのバージョンが環境に指定されます。これらの概念の詳細については、「Elastic Beanstalk コンポーネント」を参照してください。

次に、アプリケーションとその最初の環境の名前を設定します。各環境には一意の CNAME が関連付けられていて、デプロイが完了したときに、アプリケーションにアクセスするためにこの名前を使用できます。

次のページの AWS Options では、使用する AWS リソースのタイプを設定できます。この例では、[Key pair (キーペア)] セクションを除いて、デフォルト値のままにします。キーペアにより、Windows 管理者パスワードを取得できます。そのため、該当マシンにログオンできます。キーペアをまだ作成していない場合は、[Create new key pair (新しいキーペアの作成)] を選択する必要があります。

アクセス許可

アクセス許可ページは、アプリケーションを実行している EC2 インスタンスに AWS 認証情報を割り当てるために使用されます。これは、アプリケーションが を使用して他の AWS SDK for .NET

AWS サービスにアクセスする場合に重要です。アプリケーションから他のサービスを使用していない場合、このページの設定をデフォルトのまま利用できます。

アプリケーションオプション

[Application Options (アプリケーションオプション)] ページの詳細は、従来の ASP.NET アプリケーションのデプロイ時に指定するものとは異なります。ここでは、アプリケーションをパッケージするために使用するビルド設定とフレームワークを指定し、アプリケーション用の IIS リソースパスも指定します。

[Application Options] ページを完了した後、[Next (次へ)] をクリックして設定を確認し、[Deploy (デプロイ)] をクリックして、デプロイプロセスを開始します。

環境ステータスをチェックする

アプリケーションがパッケージ化されて にアップロードされたら AWS、Visual Studio の AWS Explorer から環境ステータスビューを開いて、Elastic Beanstalk 環境のステータスを確認できます。

環境がオンラインになると、ステータスバーにイベントが表示されます。すべてが完了すると、環境のステータスが正常な状態に移行します。URL をクリックして、サイトを表示することができます。ここから、環境またはリモートデスクトップから、Elastic Beanstalk 環境の一部である Amazon EC2 インスタンスにログを取り込むこともできます。

アプリケーションの最初のデプロイは、新しい AWS リソースを作成するため、その後の再デプロイよりも少し時間がかかります。開発中にアプリケーションを繰り返し更新するときは、ウィザードを最初からもう一度実行するか、または、プロジェクトを右クリックし、[Republish (再発行)] オプションを選択することで、簡単に再デプロイできます。

再公開処理では、デプロイウィザードで以前に一通り実行した設定を使用してアプリケーションをパッケージし、アプリケーションバンドルを既存のElastic Beanstalk環境にアップロードします。

アプリケーションの AWS セキュリティ認証情報を指定する方法

Publish to Elastic Beanstalk ウィザードで指定した AWS アカウントは、ウィザードが Elastic Beanstalk へのデプロイに使用する AWS アカウントです。

推奨されませんが、デプロイ後にアプリケーションが AWS サービスにアクセスするために使用する AWS アカウント認証情報を指定する必要がある場合もあります。推奨される方法は、IAM ロールを指定することです。[Publish to Elastic Beanstalk] ウィザードでは、同じことを [AWS オプション] ページの [Identity and Access Management ロール] ドロップダウンリストから実行できます。レガ

シーの [Publish to Amazon Web Services] ウィザードでは、[AWS オプション]」ページの [IAM ロール] ドロップダウンリストがこれに相当します。

IAM ロールの代わりに AWS アカウント認証情報を使用する必要がある場合は、次のいずれかの方法でアプリケーションのアカウント認証情報を指定できます AWS 。

- プロジェクトの `Web.config`ファイルの `appSettings`要素で AWS 、アカウントの認証情報に対応するプロファイルを参照します。(プロファイルを作成するには、AWS 「認証情報の設定」を参照してください。) 次の例では、プロファイル名が `myProfile` である認証情報を指定しています。

```
<appSettings>
  <!-- AWS CREDENTIALS -->
  <add key="AWSProfileName" value="myProfile"/>
</appSettings>
```

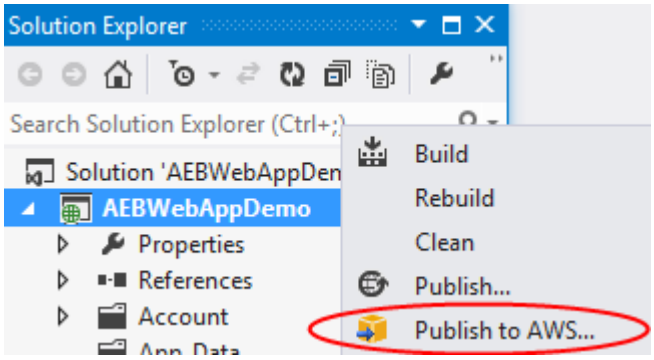
- [Publish to Elastic Beanstalk] ウィザードを使用する場合、[アプリケーションオプション] ページの [キー] と [値] のエリアにある [キー] 列で [AWS AccessKey] を選択します。[値] 行にアクセスキーを入力します。AWS SecretKey についても同じステップを繰り返します。
- レガシーの [Publish to Amazon Web Services] ウィザードでは、[アプリケーションオプション] ページの [アプリケーション認証情報] エリアで、[使用する認証情報] を選択し、[アクセスキー] ボックスと [シークレットキー] ボックスにアクセスキーとシークレットアクセスキーを入力します。

Elastic Beanstalk 環境にアプリケーションを再発行する方法 (レガシー)

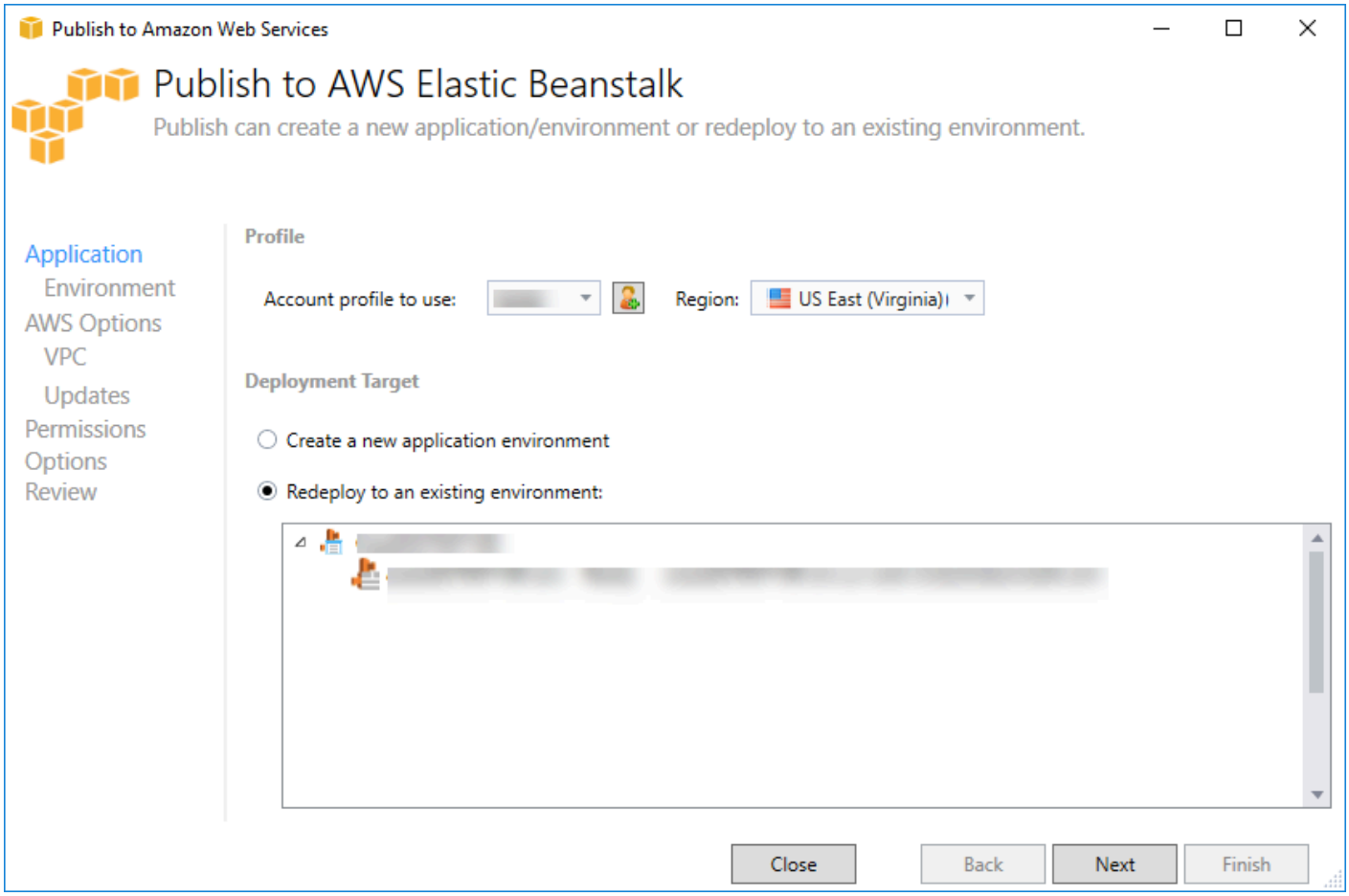
> **Important**
>
> このドキュメントでは、従来のサービスと機能について言及しています。更新されたガイドとコンテンツについては、「AWS .NET デプロイツール」ガイドを参照してください。

個別の変更を行って既に起動しているElastic Beanstalk環境に新しいバージョンを再発行することで、アプリケーションで反復処理することができます。

1. ソリューションエクスプローラーで、前のセクションで発行したプロジェクトの [AEBWebAppDemo] プロジェクトフォルダのコンテキスト (右クリック) メニューを開き、[Publish to AWS Elastic Beanstalk] を選択します。

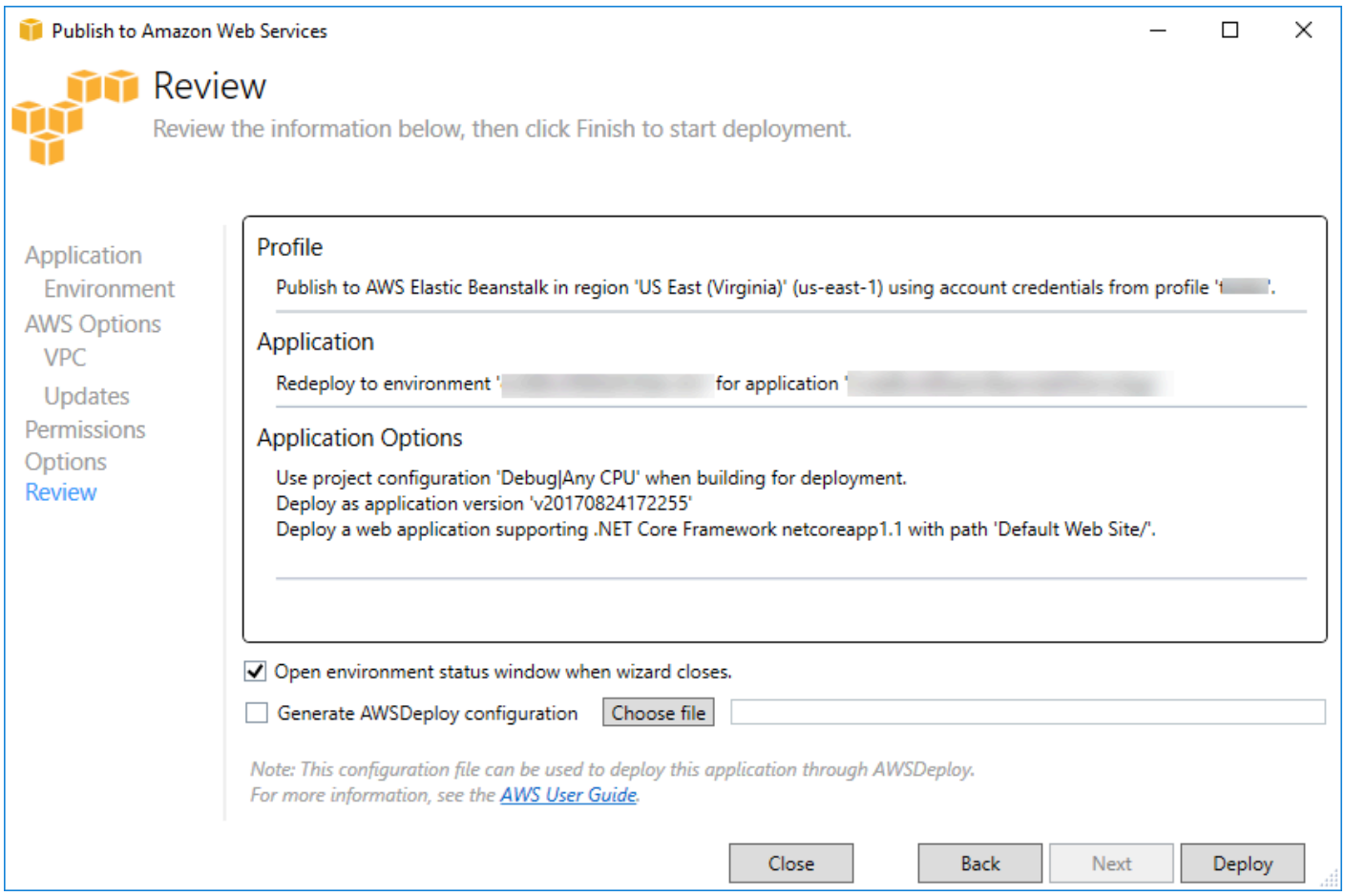


[Publish to Elastic Beanstalk] ウィザードが表示されます。



2. [Redeploy to an existing environment (既存の環境への再デプロイ)] を選択してから、先ほど公開した環境を選択します。[次へ] をクリックします。

   [Review (レビュー)] ウィザードが表示されます。



3. [Deploy (デプロイ)] をクリックします。アプリケーションは同じ環境に再デプロイされます。

アプリケーションが起動中または終了中の場合は再発行することはできません。

Elastic Beanstalk アプリケーションのカスタムデプロイ

このトピックでは、Elastic Beanstalkの Microsoft Windows コンテナ用のデプロイマニフェストがアプリケーションのカスタムデプロイをサポートする方法について説明します。

アプリケーションのカスタムデプロイは、AWS リソースを作成および管理するため、Elastic Beanstalkの力をてこ入れしたいが、アプリケーションがデプロイする方法を完全に制御したいと考えている上級ユーザーのための強力な機能です。アプリケーションのカスタムデプロイでは、Elastic Beanstalk が実行する 3 つの異なるアクションの Windows PowerShell スクリプトを作成します。デプロイが開始されるときには、インストールアクションが使用され、ツールキットまたはウェブコンソールから `RestartAppServer` API が呼び出されるときには、再起動が使用され、新しいデプロイが実行されるときには、以前のデプロイでアンインストールが使用されます。

例えば、デプロイを行う ASP.NET アプリケーションがあり、ドキュメントチームがデプロイに含むための静的ウェブサイトを作成したとします。次のようなデプロイマニフェストを作成して、これを行うことができます。

```
{
  "manifestVersion": 1,
  "deployments": {

    "msDeploy": [
      {
        "name": "app",
        "parameters": {
          "appBundle": "CoolApp.zip",
          "iisPath": "/"
        }
      }
    ],
    "custom": [
      {
        "name": "PowerShellDocs",
        "scripts": {
          "install": {
            "file": "install.ps1"
          },
          "restart": {
            "file": "restart.ps1"
          },
          "uninstall": {
            "file": "uninstall.ps1"
          }
        }
      }
    ]
  }
}
```

各アクションにリストされているスクリプトは、デプロイマニフェストファイルと関連するアプリケーションバンドルに置かれる必要があります。この例では、アプリケーションバンドルには documentation.zip ファイルも含まれ、このファイルにドキュメントチームによって作成された静的ウェブサイトが含まれています。

`install.ps1` スクリプトは、zip ファイルを抽出して、IIS パスをセットアップします。

```
Add-Type -assembly "system.io.compression.filesystem"
[io.compression.zipfile]::ExtractToDirectory('./documentation.zip', 'c:\inetpub\wwwroot
\documentation')

powershell.exe -Command {New-WebApplication -Name documentation -PhysicalPath  c:
\inetpub\wwwroot\documentation -Force}
```

アプリケーションは IIS で実行されているので、再起動アクションは IIS のリセットを呼び出します。

```
iisreset /timeout:1
```

アンインストールスクリプトでは、インストールステージで使用された、すべての設定とファイルをクリーンアップすることが重要です。それによって、新しいバージョンのインストールフェーズで、以前のデプロイとの競合を回避できます。この例では、静的ウェブサイトの IIS アプリケーションとウェブサイトのファイルを削除する必要があります。

```
powershell.exe -Command {Remove-WebApplication -Name documentation}
Remove-Item -Recurse -Force 'c:\inetpub\wwwroot\documentation'
```

アプリケーションバンドルに含まれているこれらのスクリプトファイルと documentation.zip ファイルを使って、デプロイによって、ASP.NET アプリケーションが作成され、ドキュメントのサイトがデプロイされます。

この例では、単純な静的ウェブサイトをデプロイする簡単な例を選択しましたが、アプリケーションのカスタムデプロイを使うと、あらゆる種類のアプリケーションをデプロイして、Elastic Beanstalk が AWS リソースを管理するようにできます。

カスタム ASP.NET Core Elastic Beanstalk デプロイ

このトピックでは、デプロイのしくみと、Elastic BeanstalkおよびToolkit for Visual Studioを使ってASP.NET Core アプリケーションを作成するときにカスタムデプロイで行えることについて説明します。

Toolkit for Visual Studio でデプロイウィザードを完了すると、ツールキットはアプリケーションをバンドルして、それを Elastic Beanstalk に送信します。アプリケーションバンドルを作成する最初のステップは、新しい dotnet CLI を使い、publish コマンドを使用して、アプリケーションの発行の準備をすることです。フレームワークと設定は、ウィザードの設定から publish コマンドに継承されま

す。 に Releaseconfiguration を選び、 に netcoreapp1.0framework を選んだ場合は、ツールキットは次のコマンドを実行します。

```
dotnet publish --configuration Release --framework netcoreapp1.0
```

publish コマンドが完了すると、ツールキットは新しいデプロイマニフェストを発行フォルダに書き込みます。デプロイマニフェストは aws-windows-deployment-manifest.json という JSON ファイルであり、これは Elastic Beanstalk Windows コンテナ (バージョン 1.2 以降) はこれを読み込み、アプリケーションのデプロイ方法を決定します。例えば、IIS のルートにデプロイする ASP.NET Core アプリケーションでは、ツールキットは次のようなマニフェストファイルを生成します。

```
{
  "manifestVersion": 1,
  "deployments": {

    "aspNetCoreWeb": [
      {
        "name": "app",
        "parameters": {
          "appBundle": ".",
          "iisPath": "/",
          "iisWebSite": "Default Web Site"
        }
      }
    ]
  }
}
```

appBundle プロパティは、アプリケーションのバイナリの場所をマニフェストファイルを基準として指定します。このプロパティは、ディレクトリまたは ZIP アーカイブを指すことができます。iisPath プロパティと iisWebSite プロパティは、IIS でアプリケーションをホストする場所を示します。

マニフェストをカスタマイズする

ツールキットは、マニフェストファイルが発行フォルダに存在しない場合のみ、マニフェストファイルを書き込みます。ファイルが存在する場合は、ツールキットは最初のアプリケーションの、マニフェストの appBundle セクションの下に一覧表示されている、iisPath、iisWebSite および aspNetCoreWeb プロパティを更新します。これにより、aws-windows-deployment-manifest.json をプロジェクトに追加して、マニフェストをカスタマイズできます。これを Visual Studio で

ASP.NET Core ウェブアプリケーションに行うには、新しい JSON ファイルをプロジェクトのルートに追加して、その名前を aws-windows-deployment-manifest.json とします。

マニフェストは aws-windows-deployment-manifest.json という名前である必要があり、プロジェクトのルートに存在する必要があります。Elastic Beanstalk コンテナはルートでマニフェストを検索し、検出された場合にはデプロイツールを呼び出します。ファイルが存在しない場合は、Elastic Beanstalk コンテナは以前のデプロイツールにフォールバックします。これは、アーカイブが msdeploy アーカイブであることを想定します。

dotnet CLI の `publish` コマンドにマニフェストが含まれるようにするため、`project.json` ファイルを更新して、マニフェストファイルが `include` の include セクション `publishOptions` に含まれるようにします。

```
{
   "publishOptions": {
     "include": [
       "wwwroot",
       "Views",
       "Areas/**/Views",
       "appsettings.json",
       "web.config",
       "aws-windows-deployment-manifest.json"
     ]
   }
 }
```

マニフェストの宣言が完了し、マニフェストはアプリケーションバンドルに含まれているため、アプリケーションのデプロイの方法をさらに設定することができます。デプロイウィザードがサポートする以上にデプロイをカスタマイズできます。AWS では aws-windows-deploymentmanifest.json ファイルに JSON スキーマを定義しており、Toolkit for Visual Studio をインストールすると、セットアップがそのスキーマの URL を登録します。

`windows-deployment-manifest.json` を開くと、スキーマのドロップダウンボックスで選択した、スキーマの URL があります。URL に移動すると、マニフェストで設定できる機能の完全な説明を取得できます。Visual Studio は、選択されたスキーマを使って、マニフェストの編集時に IntelliSense を提供します。

実行できる 1 つのカスタマイズは、アプリケーションが実行される IIS アプリケーションプールの設定です。以下の例では、IIS アプリケーションプール ("customPool") の定義方法を示しています。こ

のアプリケーションプールはプロセスを 60 分ごとにリサイクルし、"appPool": "customPool" を使ってアプリケーションに割り当てます。

```
{
  "manifestVersion": 1,
  "iisConfig": {
    "appPools": [
      {
        "name": "customPool",
        "recycling": {
          "regularTimeInterval": 60
        }
      }
    ]
  },
  "deployments": {
    "aspNetCoreWeb": [
      {
        "name": "app",
        "parameters": {
          "appPool": "customPool"
        }
      }
    ]
  }
}
```

さらにマニフェストは、インストール、再起動、アンインストールのアクションの前後に実行する、Windows PowerShell スクリプトを宣言できます。例えば、次のマニフェストは、Windows PowerShell スクリプト PostInstallSetup.ps1 を実行して、ASP.NET Core アプリケーションが IIS にデプロイされた後にさらにセットアップを行います。このようなスクリプトを追加する場合、スクリプトが project.json ファイルの publishOptions の下の include セクションに追加されるようにします。これは aws-windows-deployment-manifest.json ファイルの場合と同様です。このようにしない場合は、スクリプトは dotnet CLI の publish コマンドの一部として含まれません。

```
{
  "manifestVersion": 1,
  "deployments": {
    "aspNetCoreWeb": [
      {
        "name": "app",
        "scripts": {
```

```
          "postInstall": {
            "file": "SetupScripts/PostInstallSetup.ps1"
          }
        }
      }
    ]
  }
}
```

.ebextensions について

Elastic Beanstalk .ebextensions 設定ファイルは、他の Elastic Beanstalk コンテナのすべてでサポートされています。.ebextensions を ASP.NET Core アプリケーションに含めるには、`.ebextensions` ディレクトリを、`project.json` ファイル内の `publishOptions` 下の`include` セクションに追加します。.ebextensions の詳細については、Elastic Beanstalk デベロッパーガイドを参照してください。

.NET および Elastic Beanstalk 用の複数アプリケーションのサポート

デプロイマニフェストを使用すると、複数のアプリケーションを同じElastic Beanstalk 環境にデプロイする能力があります。

デプロイマニフェストは、ASP.NET Core ウェブアプリケーションおよび従来の ASP.NET アプリケーションの msdeploy アーカイブをサポートします。例えば、フロントエンドに ASP.NET Core を使用したすばらしい新たなアプリケーションを作成し、拡張 API 用のウェブ API プロジェクトがあるとします。また、従来の ASP.NET を使用して作成した管理者アプリがあるとします。

ツールキットのデプロイウィザードは 1 つのプロジェクトのデプロイに重点を置いています。複数のアプリケーションのデプロイを利用するには、アプリケーションバンドルを手動で作成する必要があります。まず、マニフェストを作成します。この例では、ソリューションのルートにマニフェストを作成します。

マニフェストのデプロイセクションには 2 つの子があります。デプロイを行う一連の ASP.NET Core ウェブアプリケーションと、デプロイを行う一連の msdeploy アーカイブです。各アプリケーションで、IIS パスとアプリケーションのビットのマニフェストへの相対位置を設定します。

```
{
  "manifestVersion": 1,
  "deployments": {

    "aspNetCoreWeb": [
```

```
      {
        "name": "frontend",
        "parameters": {
          "appBundle": "./frontend",
          "iisPath": "/frontend"
        }
      },
      {
        "name": "ext-api",
        "parameters": {
          "appBundle": "./ext-api",
          "iisPath": "/ext-api"
        }
      }
    ],
    "msDeploy": [
      {
        "name": "admin",
        "parameters": {
          "appBundle": "AmazingAdmin.zip",
          "iisPath": "/admin"
        }
      }
    ]
  }
}
```

作成したマニフェストを使い、Windows PowerShell を使用して、アプリケーションバンドルを作成し、それを実行する既存のElastic Beanstalk環境を更新します。スクリプトは、Visual Studio ソリューションが含まれるフォルダから実行されることを前提として作成されます。

スクリプトで最初に必要なことは、アプリケーションバンドルを作成する、ワークスペースフォルダをセットアップすることです。

```
$publishFolder = "c:\temp\publish"

$publishWorkspace = [System.IO.Path]::Combine($publishFolder, "workspace")
$appBundle = [System.IO.Path]::Combine($publishFolder, "app-bundle.zip")

If (Test-Path $publishWorkspace){
  Remove-Item $publishWorkspace -Confirm:$false -Force
}
If (Test-Path $appBundle){
```

```
  Remove-Item $appBundle -Confirm:$false -Force
}
```

フォルダを作成したら、次にフロントエンドの準備を行います。デプロイウィザードと同様に、dotnet CLI を使用してアプリケーションを発行します。

```
Write-Host 'Publish the ASP.NET Core frontend'
$publishFrontendFolder = [System.IO.Path]::Combine($publishWorkspace, "frontend")
dotnet publish .\src\AmazingFrontend\project.json -o $publishFrontendFolder -c Release
 -f netcoreapp1.0
```

サブフォルダ "frontend" は出力フォルダに使用されていることに注意し、マニフェストで設定するフォルダを一致させます。次に、同様の作業をウェブ API プロジェクトに対して行う必要があります。

```
Write-Host 'Publish the ASP.NET Core extensibility API'
$publishExtAPIFolder = [System.IO.Path]::Combine($publishWorkspace, "ext-api")
dotnet publish .\src\AmazingExtensibleAPI\project.json -o $publishExtAPIFolder -c
 Release -f netcoreapp1.0
```

管理者サイトは、従来の ASP.NET アプリケーションであるため、dotnet CLI を使用することはできません。管理者アプリケーションには、msbuild を使用して、ビルドターゲットパッケージを渡して msdeploy アーカイブを作成します。デフォルトでは、パッケージターゲットは msdeploy アーカイブを obj\Release\Package フォルダの下に作成するため、アーカイブを発行ワークスペースにコピーする必要があります。

```
Write-Host 'Create msdeploy archive for admin site'
msbuild .\src\AmazingAdmin\AmazingAdmin.csproj /t:package /p:Configuration=Release
Copy-Item .\src\AmazingAdmin\obj\Release\Package\AmazingAdmin.zip $publishWorkspace
```

これらすべてのアプリケーションに関する指示を Elastic Beanstalk 環境に与えるには、マニフェストをソリューションから発行ワークスペースにコピーしてフォルダを圧縮します。

```
Write-Host 'Copy deployment manifest'
Copy-Item .\aws-windows-deployment-manifest.json $publishWorkspace

Write-Host 'Zipping up publish workspace to create app bundle'
Add-Type -assembly "system.io.compression.filesystem"
[io.compression.zipfile]::CreateFromDirectory( $publishWorkspace, $appBundle)
```

アプリケーションバンドルの作成が完了したので、ウェブコンソールに移動して、アーカイブを Elastic Beanstalk環境にアップロードします。または、引き続き AWS PowerShell コマンドレットを使用して、アプリケーションバンドルを使って、Elastic Beanstalk 環境を更新します。`Set-AWSCredentials` および `Set-DefaultAWSRegion` コマンドレットを使用して、Elastic Beanstalk 環境を含むプロファイルとリージョンに現在のプロファイルとリージョンが設定されていることを確認します。

```
Write-Host 'Write application bundle to S3'
# Determine S3 bucket to store application bundle
$s3Bucket = New-EBStorageLocation
Write-S3Object -BucketName $s3Bucket -File $appBundle


$applicationName = "ASPNETCoreOnAWS"
$environmentName = "ASPNETCoreOnAWS-dev"
$versionLabel = [System.DateTime]::Now.Ticks.ToString()

Write-Host 'Update Beanstalk environment for new application bundle'
New-EBApplicationVersion -ApplicationName $applicationName -VersionLabel $versionLabel
 -SourceBundle_S3Bucket $s3Bucket -SourceBundle_S3Key app-bundle.zip
Update-EBEnvironment -ApplicationName $applicationName -EnvironmentName
 $environmentName -VersionLabel $versionLabel
```

ここで、ツールキットまたはウェブコンソールのどちらかの Elastic Beanstalk 環境のステータスページを使用して、更新のステータスを確認します。完了すると、デプロイマニフェストで設定した IIS パスを使って、デプロイされた各アプリケーションに移動できるようになります。

Amazon EC2 Container Service へのデプロイ

> **Important**
>
> 新機能である Publish to AWS は、.NET アプリケーションを AWS に発行する方法を簡素化するように設計されています。[AWS にコンテナを発行] を選択した後で、この発行方式に切り替えるかどうか質問するメッセージが表示されることがあります。詳細については、「Visual Studio で使用する Publish to AWS の操作」を参照してください。

Amazon Elastic Container は非常にスケーラブルかつ高性能なコンテナ管理サービスで、Docker コンテナに対応しており、Amazon EC2 インスタンスのマネージドクラスターでアプリケーションを簡単に実行できるようになります。

アプリケーションをAmazon Elastic Containerにデプロイするには、アプリケーションコンポーネントを開発して Docker コンテナで実行する必要があります。Docker コンテナは標準化されたソフトウェア開発用のユニットであり、コード、ランタイム、システムツール、システムライブラリなど、ソフトウェアアプリケーションの実行に必要なものがすべて含まれています。

Toolkit for Visual Studio には、Amazon ECS によるアプリケーションの発行を簡素化するウィザードが用意されています。このウィザードについては、次のセクションで説明します。

Amazon ECS の詳細については、「Elastic Container Service のドキュメント」を参照してください。このドキュメントには、Docker の基本とクラスターの作成の概要が示されています。

トピック

- ASP.NET Core AWS 2 アプリケーションの認証情報を指定する
- Amazon ECS への ASP.NET Core 2.0 アプリケーションのデプロイ (Fargate) (レガシー)
- Amazon ECS(EC2) への ASP.NET Core 2.0 アプリケーションのデプロイ する

ASP.NET Core AWS 2 アプリケーションの認証情報を指定する

アプリケーションを Docker コンテナにデプロイするときに必要になる認証情報には、デプロイ認証情報とインスタンス認証情報の 2 タイプがあります。

デプロイ認証情報は、Amazon ECS で環境を作成するために、コンテナの発行 AWS ウィザードによって使用されます。これには、タスク、サービス、IAM ロール、Docker コンテナリポジトリのほか、選択した場合はロードバランサーも含まれます。

インスタンス認証情報は、インスタンス (アプリケーションを含む) がさまざまな AWS サービスにアクセスするために使用します。例えば、ASP.NET Core 2.0 アプリケーションが Amazon S3 オブジェクトに対して読み書きを行う場合は、適切なアクセス許可が必要になります。そのために、環境に基づいて異なる方法で異なる認証情報を提供できます。例えば、ASP.NET Core 2 アプリケーションは開発環境と本番稼働用環境を対象とする場合があります。ローカルの Docker インスタンスおよび認証情報を開発環境に使用し、定義したロールを本番稼働用環境に使用できます。

デプロイ認証情報の指定

コンテナの発行 AWSウィザードで指定した AWS アカウントは、ウィザードが Amazon ECS へのデプロイに使用する AWS アカウントです。アカウントプロファイルには、Amazon Elastic Compute Cloud、Amazon Elastic Container Service、および へのアクセス許可が必要です AWS Identity and Access Management。

ドロップダウンリストにオプションが見つからない場合は、それらのアクセス許可が不足している可能性があります。例えば、アプリケーション用のクラスターを作成したのに [AWSにコンテナを発行] ウィザードの [クラスター] ページに表示されない場合などです。この場合、不足しているアクセス許可を追加してウィザードを再試行してください。

開発用のインスタンス認証情報の指定

本番稼働用でない環境では、appsettings.<environment>.json ファイルに認証情報を設定できます。例えば、Visual Studio 2017 の appsettings.Development.json ファイルで認証情報を設定するには、以下の手順に従います。

1. プロジェクトに AWSSDK.Extensions.NETCore.Setup NuGet パッケージを追加します。
2. appsettings.Development.json AWS に設定を追加します。以下に示しているのは、`Profile` と `Region` の設定です。

```
{
    "AWS": {
        "Profile": "local-test-profile",
        "Region": "us-west-2"
    }
}
```

本番稼働用のインスタンス認証情報の指定

本番稼働用インスタンスでは、アプリケーション (およびサービス) のアクセスコントロールに IAM ロールを使用することをお勧めします。例えば、Amazon ECS による IAM ロールをサービスプリンシパルとし、Amazon Simple Storage Service および Amazon DynamoDB に対するアクセス許可を持つ IMA ロールを AWS マネジメントコンソールから設定するには、以下の手順に従います。

1. にサインイン AWS マネジメントコンソール し、https://console.aws.amazon.com/iam/ で IAM コンソールを開きます。
2. IAM コンソールのナビゲーションペインで、[ロール]、[ロールを作成] を選択します。

3. [AWS Service] ロールタイプを選択してから、[EC2 Container Service] (EC2 コンテナサービス) を選択します。
4. [EC2 Container Service Task (EC2 コンテナサービスタスク)] ユースケースを選択します。ユースケースは、サービスに必要な信頼ポリシーを含める定義になります。その後、[Next: Permissions (次へ: アクセス許可)] を選択します。
5. AmazonS3FullAccess および AmazonDynamoDBFullAccess アクセス許可ポリシーを選択します。各ポリシーの横にあるチェックボックスをオンにし、[Next: Review (次へ: レビュー)] を選択します。
6. [Role name (ロール名)] に、このロールの目的を識別するのに役立つロール名またはロール名サフィックスを入力します。ロール名は AWS アカウント内で一意でなければなりません。大文字と小文字は区別されません。例えば、`PRODROLE` と `prodrole` というロール名を両方作成することはできません。多くのエンティティによりロールが参照されるため、作成後にロール名を変更することはできません。
7. (オプション) [Role description] に、新しいロールの説明を入力します。
8. ロール情報を確認し、ロールの作成 を選択します。

このロールは、[AWSにコンテナを発行] ウィザードの [ECS タスク定義] ページで [タスクロール] として使用できます。

詳細については、「サービスベースのロールの使用」を参照してください。

Amazon ECS への ASP.NET Core 2.0 アプリケーションのデプロイ (Fargate) (レガシー)

> **Important**
>
> このドキュメントでは、従来のサービスと機能について言及しています。更新されたガイドとコンテンツについては、「AWS .NET Deployment tool」ガイドと、目次にある更新された「AWSへのデプロイ」を参照してください。

このセクションでは、 Toolkit for Visual Studioの一部として提供される [AWSにコンテナを発行] ウィザードにより、Fargate 起動タイプを使用してAmazon ECS経由でLinux 向けコンテナ化 ASP.NET Core 2.0 アプリケーションをデプロイする方法について説明します。ウェブアプリケーションは継続的に実行されるため、サービスとしてデプロイされます。

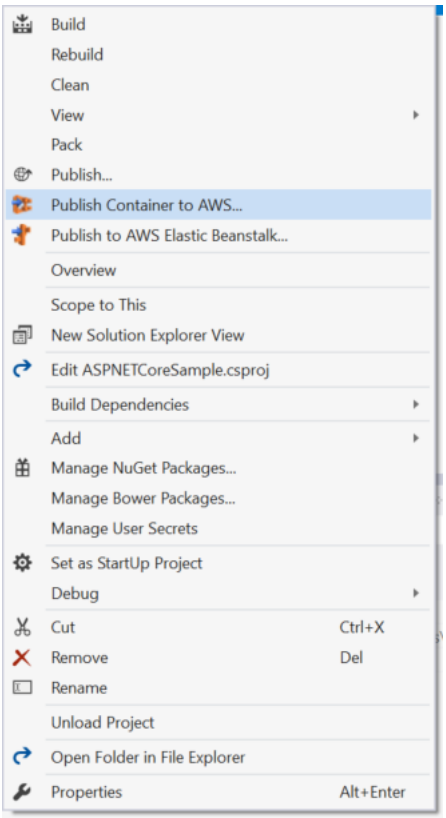
コンテナを発行する前に

[AWSにコンテナを発行] ウィザードを使用して ASP.NET Core 2.0 アプリケーションをデプロイする前に

- AWS 認証情報を指定し、Amazon ECS をセットアップします。
- Docker をインストールします。Docker for Windows などさまざまなインストールオプションがあります。
- Visual Studio で、Linux をターゲットとする ASP.NET Core 2.0 コンテナ化アプリケーションのプロジェクトを作成します (または開きます) 。

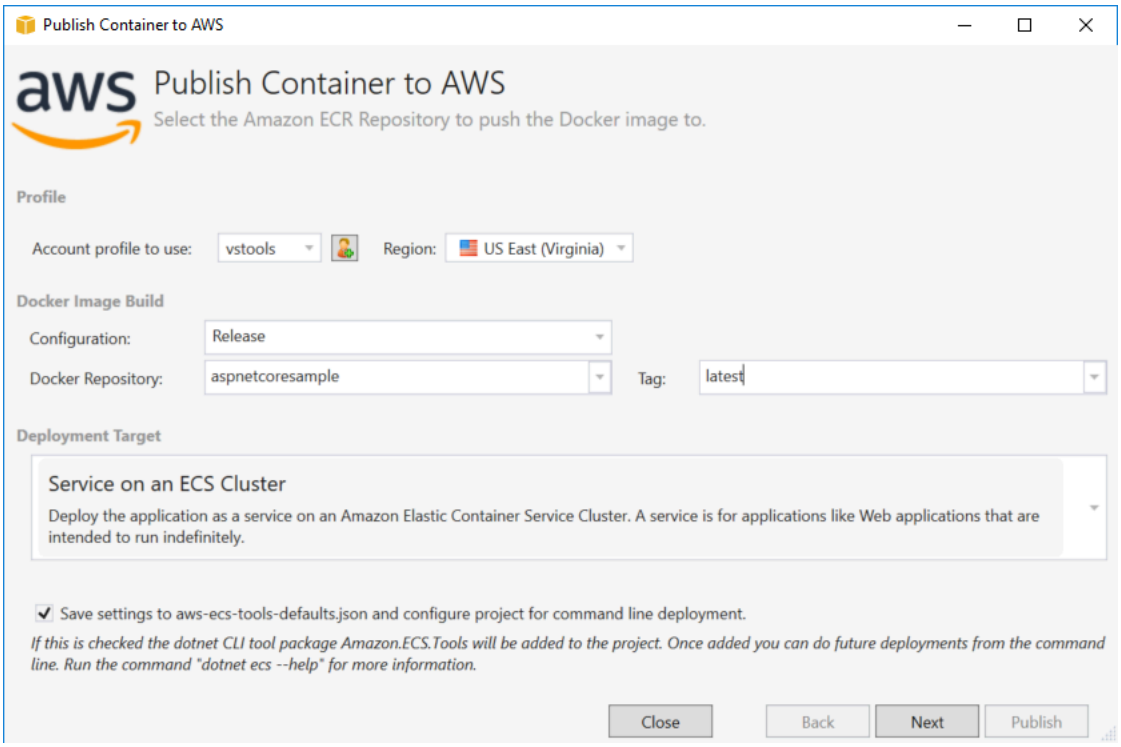
Publish Container to AWS ウィザードへのアクセス

Linux 用 ASP.NET Core 2.0 コンテナ化アプリケーションをデプロイするには、Solution Explorer でプロジェクトを右クリックし、[AWSにコンテナを発行] を選択します。



Visual Studio で [Build] (ビルド) メニューの [Publish Container to AWS] (AWSにコンテナを発行) を選択することもできます。

コンテナを AWS ウィザードに発行する



Account profile to use (使用するアカウントのプロファイル) - 使用するアカウントプロファイルを選択します。

Region (リージョン) - デプロイリージョンを選択します。プロファイルとリージョンは、デプロイ環境リソースのセットアップとデフォルト Docker レジストリの選択に使用されます。

Configuration (設定) - Docker イメージビルド設定を選択します。

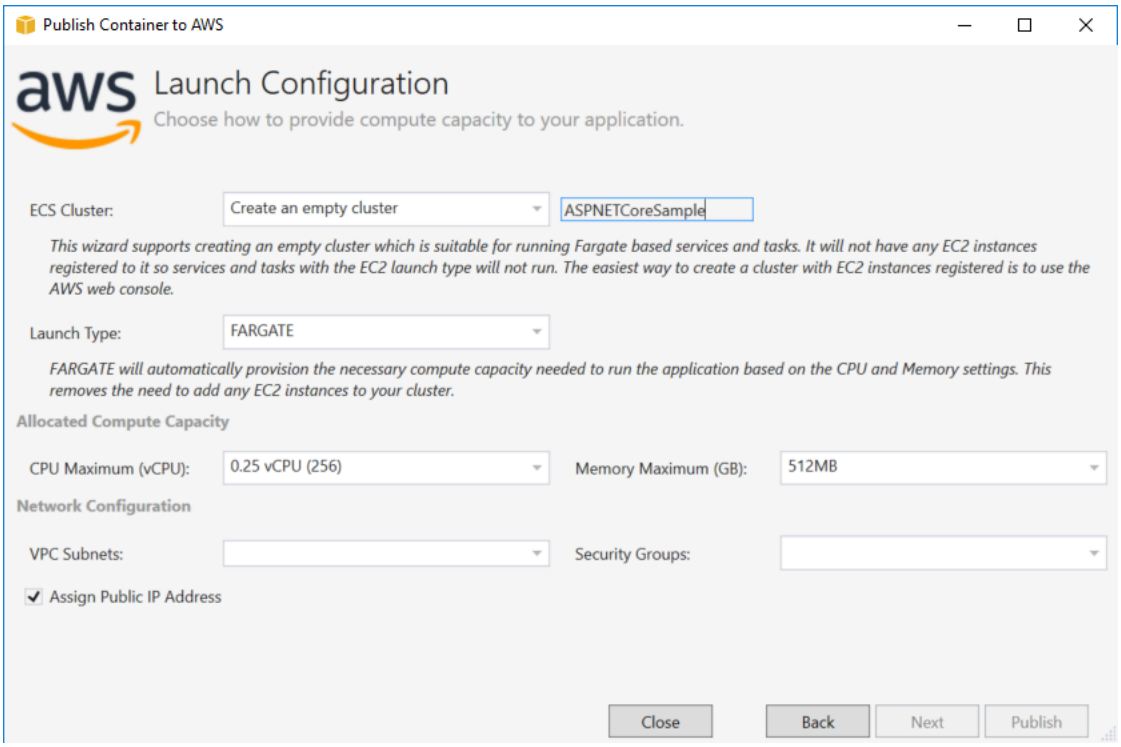
Docker リポジトリ - 既存の Docker リポジトリを選択するか、新しいリポジトリの名前を入力します (新しいリポジトリが作成されます)。これは、ビルドコンテナがプッシュされるリポジトリです。

Tag (タグ) - 既存のタグを選択するか、新しいタグの名前を入力します。タグを使用して、Docker コンテナに固有の設定要素 (バージョン、オプションなど) のような重要な詳細を追跡できます。

Deployment Target (デプロイのターゲット) - [Service on an ECS Cluster (ECS クラスターのサービス)] を選択します。このデプロイオプションは、実行時間の長いアプリケーション (ASP.NET ウェブアプリケーションなど) に使用します。

[**aws-docker-tools-defaults.json** に設定を保存して、コマンドラインデプロイのプロジェクトを設定する] - コマンドラインから柔軟にデプロイしたいて場合には、このチェックボックスをオンにします。`dotnet ecs deploy` を使用してプロジェクトディレクトリからコンテナをデプロイし、`dotnet ecs publish` を使用してコンテナを発行します。

[Launch Configuration (起動設定)] ページ



ECS Cluster (ECS クラスター) - Docker イメージを実行するクラスターを選択します。空のクラスターを作成する場合は、新しいクラスターの名前を入力します。

Launch Type (起動タイプ) - [FARGATE] を選択します。

CPU Maximum (vCPU) - アプリケーションに必要なコンピューティング性能の最大数を選択します。CPU とメモリの値の許容範囲を確認するには、「タスクサイズ」を参照してください。

Memory Maximum (GB) (最大メモリ (GB)) - アプリケーションで使用できるメモリの最大量を選択します。

VPC Subnets (VPC サブネット) - 1 つの VPC の 1 つ以上のサブネットを選択します。複数のサブネットを選択すると、タスクはそれらのサブネット間で分散されます。これにより、可用性が向上します。詳細については、「デフォルトの VPC とサブネット」を参照してください。

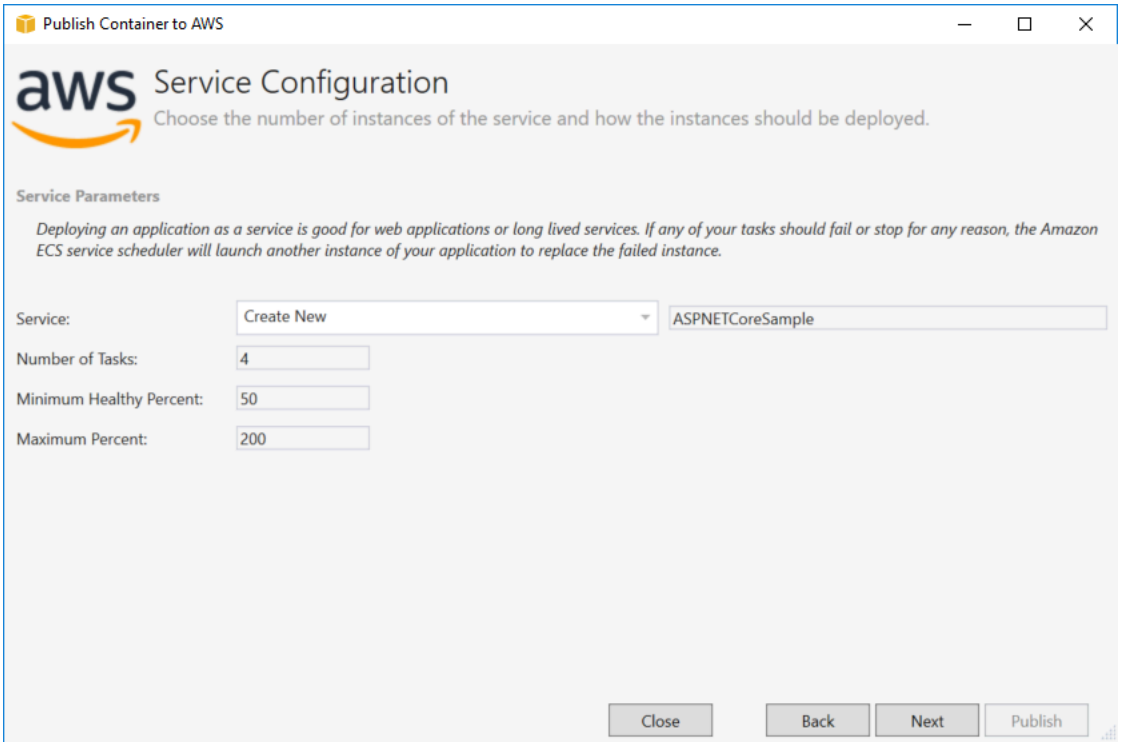
Security Groups (セキュリティグループ) - セキュリティグループを選択します。

セキュリティグループは、関連付けられた Amazon EC2 インスタンスのファイアウォールとして動作し、インバウンドトラフィックとアウトバウンドトラフィックの両方をインスタンスレベルでコントロールします。

デフォルトセキュリティグループは、同じセキュリティグループに割り当てられたインスタンスからのインバウンドトラフィックとすべてのアウトバウンド IPv4 トラフィックを許可するように設定されています。サービスがコンテナリポジトリに到達できるように、アウトバウンドトラフィックを許可している必要があります。

Assign Public IP Address (パブリック IP アドレスの割り当て) - タスクをインターネットからアクセス可能にするには、このチェックボックスをオンにします。

[Service Configuration (サービス設定)] ページ



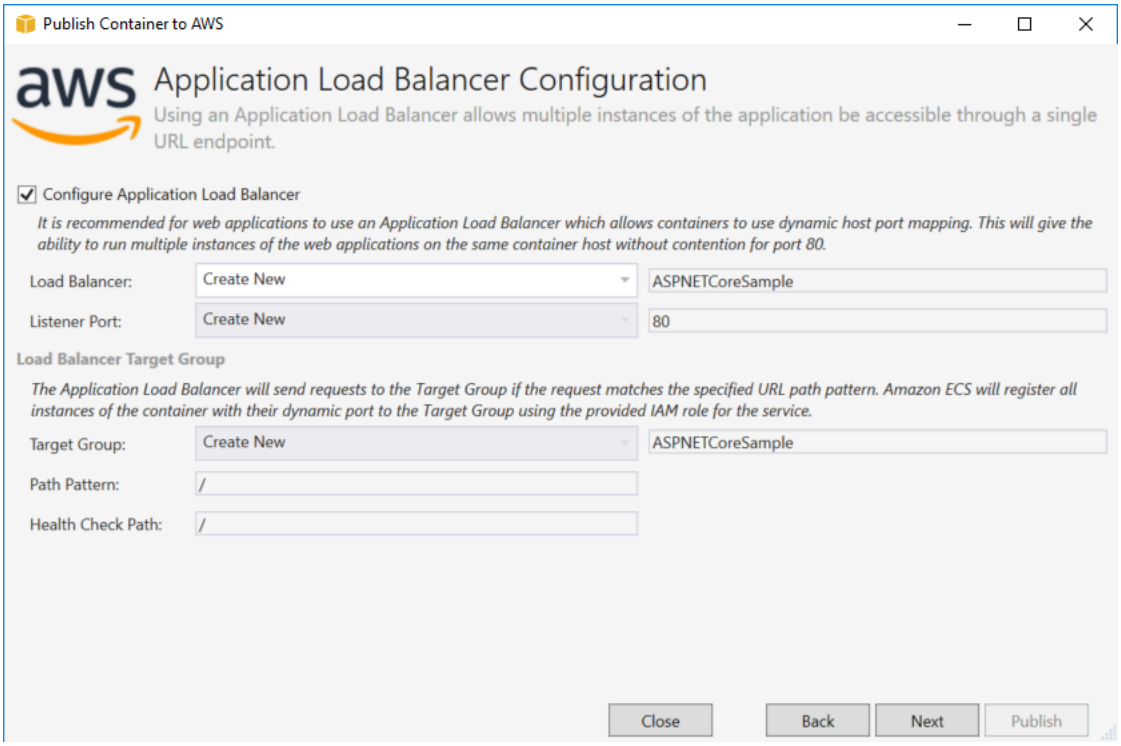
Service (サービス) - 既存のサービス内にコンテナをデプロイするには、ドロップダウンリストでいずれかのサービスを選択します。または、新しいサービスを作成するには、[Create New (新規作成)] を選択します。サービス名は同じクラスター内で一意になるようにしてください。ただし、リージョン内のクラスター間や複数のリージョンにまたがるクラスター間では、同様の名前のサービスがあっても構いません。

Number of Tasks (タスクの数) - クラスターにデプロイして実行状態に保つタスクの数。各タスクはコンテナの 1 つのインスタンスです。

Minimum Healthy Percent (最小ヘルス率) - デプロイ中に `RUNNING` 状態に保つ必要のあるタスクの最小割合 (最も近い整数に切り上げ)。

Maximum Percent (最大率) - デプロイ中に RUNNING または PENDING 状態に保つことのできるタスクの最大割合 (最も近い整数に切り下げ)。

[Application Load Balancer (アプリケーションロードバランサー)] ページ



Configure Application Load Balancer (アプリケーションロードバランサーの設定) - Application Load Balancer を設定するには、このチェックボックスをオンにします。

Load Balancer (ロードバランサー) - 既存のロードバランサーを選択するか、[Create New (新規作成)] を選択して新しいロードバランサーの名前を入力します。

Listener Port (リスナーポート) - 既存のリスナーポートを選択するか、[Create New (新規作成)] を選択してポート番号を入力します。デフォルトのポート 80 はほとんどのウェブアプリケーションに適しています。

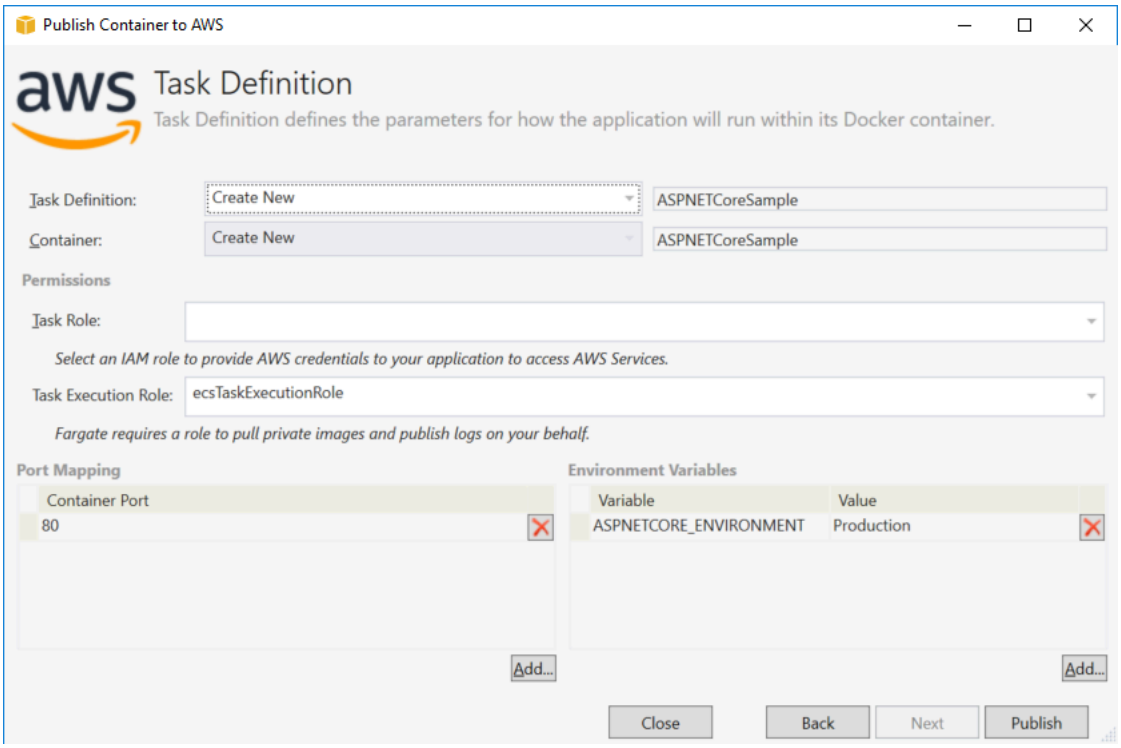
[Target Group] (ターゲットグループ) - Amazon ECS がサービスに対するタスクを登録するターゲットグループを選択します。

Path Pattern (パスパターン) - ロードバランサーがパスベースのルーティングを使用するようになります。デフォルトの / を受け入れるか、別のパターンを指定します。パスパターンでは大文字と小文字が区別され、最大 128 文字までの長さで、特定の文字セットを含めることができます。

Health Check Path (ヘルスチェックパス) - ヘルスチェックのターゲットの ping 先のパス。デフォルトでは、/ です。必要に応じて別のパスを入力します。入力したパスが無効な場合、ヘルスチェックは失敗し、異常とみなされます。

複数のサービスをデプロイし、各サービスが異なるパスまたは場所にデプロイされる場合は、カスタムチェックパスが必要になります。

[Task Definition (タスク定義)] ページ



Task Definition (タスク定義) - 既存のタスク定義を選択するか、[Create New (新規作成)] を選択して新しいタスク定義の名前を入力します。

Container (コンテナ) - 既存のコンテナを選択するか、[Create New (新規作成)] を選択して新しいコンテナの名前を入力します。

タスクロール - アプリケーションが AWS サービスにアクセスするために必要な認証情報を持つ IAM ロールを選択します。これは、アプリケーションに認証情報が渡される方法になります。「アプリケーション用に AWS セキュリティの認証情報を指定する方法」を参照してください。

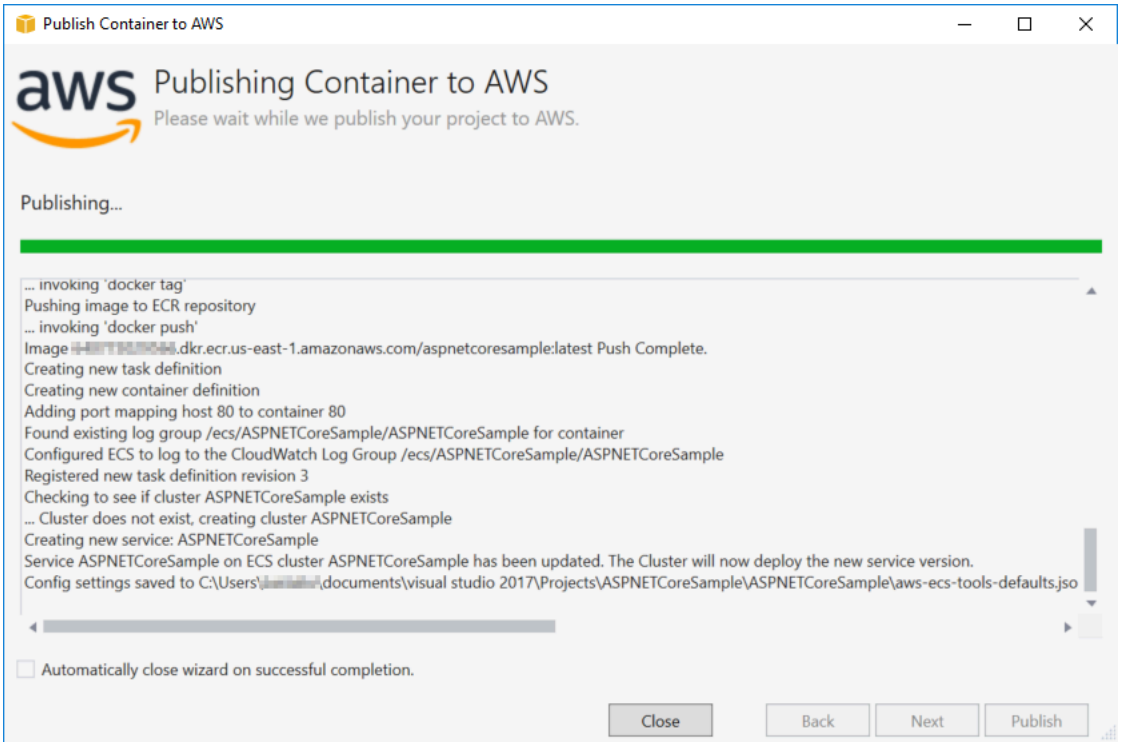
タスク実行ロール - プライベートイメージをプルしてログを発行するアクセス許可を持つロールを選択します。 AWS Fargate はユーザーに代わってそれを使用します。

Port Mapping (ポートマッピング) - 自動的に割り当てられたホストポートにバインドされるコンテナポートの番号を選択します。

Environment Variables (環境変数) - コンテナの環境変数を追加、変更、または削除します。環境変数はデプロイに合わせて変更できます。

設定に問題がなければ、[発行] をクリックしてデプロイプロセスを開始します。

へのコンテナの発行 AWS



デプロイ中にイベントが表示されます。ウィザードは正常終了時に自動的に閉じられます。この動作を無効にするには、ページの下部にあるチェックボックスをオフにします。

新しいインスタンスの URL は AWS Explorer にあります。[Amazon ECS and Clusters (Amazon ECS およびクラスター)] を展開し、クラスターをクリックします。

Amazon ECS(EC2) への ASP.NET Core 2.0 アプリケーションのデプロイする

このセクションでは、 Toolkit for Visual Studio の一部として提供される [AWSにコンテナを発行] ウィザードにより、EC2 起動タイプを使用して Linux 用のコンテナ化 ASP.NET Core 2.0 アプリケーションを Amazon ECS 経由でデプロイする方法について説明します。ウェブアプリケーションは継続的に実行されるため、サービスとしてデプロイされます。

コンテナを発行する前に

[AWSにコンテナを発行] を使用して ASP.NET Core 2.0 アプリケーションをデプロイするには

- AWS 認証情報を指定し、Amazon ECS をセットアップします。
- Docker をインストールします。Docker for Windows などさまざまなインストールオプションがあります。
- ウェブアプリケーションのニーズに基づいて、Amazon ECS クラスターを作成します。このプロセスはほんの数ステップで完了します。
- Visual Studio で、Linux をターゲットとする ASP.NET Core 2.0 コンテナ化アプリケーションのプロジェクトを作成します (または開きます) 。

Publish Container to AWS ウィザードへのアクセス

Linux 用 ASP.NET Core 2.0 コンテナ化アプリケーションをデプロイするには、Solution Explorer でプロジェクトを右クリックし、[AWSにコンテナを発行] を選択します。

Visual Studio で [Build] (ビルド) メニューの [Publish Container to AWS] (AWSにコンテナを発行) を選択することもできます。

コンテナを AWS ウィザードに発行する

Account profile to use (使用するアカウントのプロファイル) - 使用するアカウントプロファイルを選択します。

Region (リージョン) - デプロイリージョンを選択します。プロファイルとリージョンは、デプロイ環境リソースのセットアップとデフォルト Docker レジストリの選択に使用されます。

Configuration (設定) - Docker イメージビルド設定を選択します。

Docker リポジトリ - 既存の Docker リポジトリを選択するか、新しいリポジトリの名前を入力します (新しいリポジトリが作成されます)。これは、ビルドコンテナイメージがプッシュされるリポジトリです。

Tag (タグ) - 既存のタグを選択するか、新しいタグの名前を入力します。タグを使用して、Docker コンテナに固有の設定要素 (バージョン、オプションなど) のような重要な詳細を追跡できます。

Deployment (デプロイ) - [Service on an ECS Cluster (ECS Cluster のサービス)] を選択します。このデプロイオプションは、実行時間の長いアプリケーション (ASP.NET Core 2.0 ウェブアプリケーションなど) に使用します。

[**aws-docker-tools-defaults.json** に設定を保存して、コマンドラインデプロイのプロジェクトを設定する] - コマンドラインから柔軟にデプロイしたいて場合には、このチェックボックスをオンにします。`dotnet ecs deploy` を使用してプロジェクトディレクトリからコンテナをデプロイし、`dotnet ecs publish` を使用してコンテナを発行します。

[Launch Configuration (起動設定)] ページ

ECS Cluster (ECS クラスター) - Docker イメージを実行するクラスターを選択します。 AWS マネジメントコンソールを使用して ECS クラスターを作成できます。

Launch Type (起動タイプ) - [EC2] を選択します。Fargate 起動タイプを使用するには、「Amazon ECS への ASP.NET Core 2.0 アプリケーションのデプロイ (Fargate)」を参照してください。

[Service Configuration (サービス設定)] ページ

Service (サービス) - 既存のサービス内にコンテナをデプロイするには、ドロップダウンリストでいずれかのサービスを選択します。または、新しいサービスを作成するには、[Create New (新規作成)] を選択します。サービス名は同じクラスター内で一意になるようにしてください。ただし、リージョン内のクラスター間や複数のリージョンにまたがるクラスター間では、同様の名前のサービスがあっても構いません。

Number of Tasks (タスクの数) - クラスターにデプロイして実行状態に保つタスクの数。各タスクはコンテナの 1 つのインスタンスです。

Minimum Healthy Percent (最小ヘルス率) - デプロイ中に `RUNNING` 状態に保つ必要のあるタスクの最小割合 (最も近い整数に切り上げ)。

Maximum Percent (最大率) - デプロイ中に `RUNNING` または `PENDING` 状態に保つことのできるタスクの最大割合 (最も近い整数に切り下げ)。

Placement Templates (配置テンプレート) - タスク配置テンプレートを選択します。

クラスター内にタスクを起動するとき、Amazon ECS では、タスク定義で指定されている CPU やメモリなどの要件に基づいてタスクを配置する場所を決定する必要があります。同様に、タスク数を減らすときも、Amazon ECS でどのタスクを終了させるか決定する必要があります。

配置テンプレートは、クラスター内にタスクを起動する方法をコントロールします。

- [AZ Balanced Spread] - アベイラビリティーゾーン間およびアベイラビリティーゾーン内のコンテナインスタンス間でタスクを分散します。

- [AZ Balanced BinPack] - 利用可能な最小メモリでアベイラビリティーゾーン間およびコンテナインスタンス間でタスクを分散します。
- [BinPack] - CPU またはメモリの最小利用可能量に基づいてタスクを配置します。
- [One Task Per Instance] - 各コンテナインスタンスのサービスから最大 1 タスクを配置します。

詳細については、「Amazon ECS のタスク配置」を参照してください。

[Application Load Balancer (アプリケーションロードバランサー)] ページ

Configure Application Load Balancer (アプリケーションロードバランサーの設定) - Application Load Balancer を設定するには、このチェックボックスをオンにします。

Select IAM role for service (サービス用の IAM ロールの選択) - 既存のロールを選択するか、[Create New (新規作成)] を選択して新しいロールを作成します。

Load Balancer (ロードバランサー) - 既存のロードバランサーを選択するか、[Create New (新規作成)] を選択して新しいロードバランサーの名前を入力します。

Listener Port (リスナーポート) - 既存のリスナーポートを選択するか、[Create New (新規作成)] を選択してポート番号を入力します。デフォルトのポート 80 はほとんどのウェブアプリケーションに適しています。

Target Group (ターゲットグループ) - デフォルトでは、ロードバランサーはターゲットグループ用に指定したポートとプロトコルを使用して登録済みターゲットにリクエストを送ります。ターゲットグループに各ターゲットを登録するときに、このポートを上書きできます。

Path Pattern (パスパターン) - ロードバランサーがパスベースのルーティングを使用するようになります。デフォルトの / を受け入れるか、別のパターンを指定します。パスパターンでは大文字と小文字が区別され、最大 128 文字までの長さで、特定の文字セットを含めることができます。

Health Check Path (ヘルスチェックパス) - ヘルスチェックのターゲットの ping 先のパス。デフォルトでは / であり、ウェブアプリケーションに適しています。必要に応じて別のパスを入力します。入力したパスが無効な場合、ヘルスチェックは失敗し、異常とみなされます。

複数のサービスをデプロイし、各サービスが異なるパスまたは場所にデプロイされる場合は、カスタムチェックパスが必要になることがあります。

[ECS Task Definition (ECS タスク定義)] ページ

Task Definition (タスク定義) - 既存のタスク定義を選択するか、[Create New (新規作成)] を選択して新しいタスク定義の名前を入力します。

Container (コンテナ) - 既存のコンテナを選択するか、[Create New (新規作成)] を選択して新しいコンテナの名前を入力します。

Memory (MiB) (メモリ(MiB)) - ソフト制限とハード制限のいずれか、または両方の値を入力します。

コンテナ用に予約するメモリのソフト制限 (MiB 単位)。Docker により、コンテナメモリはソフト制限を超えないように保たれます。コンテナは、メモリパラメータで指定したハード制限 (該当する場合) に達するか、コンテナインスタンス上の使用可能なメモリの上限に達するか、いずれか早く達する方まで、メモリを使用できます。

コンテナに適用されるメモリのハード制限 (MiB 単位)。コンテナは、ここで指定したメモリを超えようとすると、強制終了されます。

タスクロール - ユーザーに代わって、関連付けられたポリシーで指定された AWS APIs を呼び出すアクセス許可をコンテナに付与する IAM ロールのタスクロールを選択します。これは、アプリケーションに認証情報が渡される方法になります。アプリケーションの AWS セキュリティ認証情報を指定する方法を参照してください。

Port Mapping (ポートマッピング) - コンテナのポートマッピングを追加、変更、または削除します。ロードバランサーがオンの場合、ホストポートはデフォルトで 0 になり、ポート割り当ては動的になります。

Environment Variables (環境変数) - コンテナの環境変数を追加、変更、または削除します。

設定に問題がなければ、[発行] をクリックしてデプロイプロセスを開始します。

へのコンテナの発行 AWS

デプロイ中にイベントが表示されます。ウィザードは正常終了時に自動的に閉じられます。この動作を無効にするには、ページの下部にあるチェックボックスをオフにします。

新しいインスタンスの URL は AWS Explorer にあります。[Amazon ECS and Clusters (Amazon ECS およびクラスター)] を展開し、クラスターをクリックします。

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

以下のセクションでは、 AWS Toolkit for Visual Studio 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。

> **Note**
>
> インストールおよびセットアップ固有のトラブルシューティング情報については、このユーザーガイドの「インストールに関する問題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

AWS Toolkit for Visual Studio に関する問題のトラブルシューティングに推奨されるベストプラクティスを以下に示します。

- Visual Studio を修復してシステムを再起動する
- 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 Logs を収集します。Toolkit ログの検索 AWS 方法の詳細については、この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 GitHub リポジトリのAWS Toolkit for Visual Studio 「問題」セクションで未解決の問題を報告します。

Visual Studio を修復してシステムを再起動する

1. Visual Studio で実行中のすべてのインスタンスを閉じます。
2. Windows のスタートメニューから、Visual Studio インストーラー を起動します。

3. 影響を受けている Visual Studio のインストールで修復を実行します。これにより、Visual Studio はインストールされた拡張機能のインデックスを再構築できます。
4. Visual Studio を再起動する前に、Windows を再起動してください。

Toolkit AWS ログを見つける方法

1. Visual Studio のメインメニューから [Extensions] を展開します。
2. Toolkit AWS を選択して AWS Toolkit メニューを展開し、View Toolkit Logs を選択します。
3. Toolkit Logs AWS フォルダがオペレーティングシステムで開いたら、ファイルを日付別にソートし、現在の問題に関連する情報を含むログファイルを見つけます。

Amazon Q セキュリティスキャンの表示とフィルタリング

Visual Studio で Amazon Q セキュリティスキャンを表示するには、Visual Studio のメインメニューで [表示] の見出しを展開し、[エラー一覧] を選択して、Visual Studio の [エラー一覧] を開きます。

デフォルトでは、Visual Studio の [エラー一覧] には、コードベースのすべての警告とエラーが表示されます。Visual Studio の [エラー一覧] で Amazon Q セキュリティスキャンの検出結果をフィルタリングするには、次の手順を実行してフィルターを作成します。

> **Note**
>
> Amazon Q セキュリティスキャンの検出結果は、セキュリティスキャンが実行され、問題が検出された後にのみ表示されます。
> Amazon Q セキュリティスキャンの検出結果は、Visual Studio では警告として表示されます。[エラー一覧] で Amazon Q セキュリティスキャンの検出結果を表示するには、[エラー一覧] の見出しの [警告] オプションを選択する必要があります。

1. Visual Studio のメインメニューで [表示] の見出しを展開し、次に [エラー一覧] を選択して [エラー一覧] ペインを開きます。
2. [エラー一覧] ペインで、ヘッダー行を右クリックしてコンテキストメニューを開きます。
3. コンテキストメニューから [列の表示] を展開し、展開したメニューで [ツール] を選択します。
4. [ツール] 列が [エラー一覧] に追加されます。
5. [ツール] 列ヘッダーで [フィルター] アイコンをクリックし、[Amazon Q] を選択して Amazon Q セキュリティスキャンの検出結果をフィルタリングします。

AWS ツールキットが正しくインストールされていない

問題:

Visual Studio を起動してから 1 分以内に、出力ペインと情報バーにそれぞれ AWS Toolkit for Visual Studio 次のメッセージが表示されます。

```
Some Toolkit components could not be initialized. Some functionality may
not work during this IDE session.

The AWS Toolkit is not properly installed.
```

解決策:

拡張機能の更新またはインストールによって、Visual Studio の内部キャッシュファイルの一部が同期しなくなった可能性があります。次の手順で、次回 Visual Studio を起動するときにこれらのファイルを再構築する方法について説明します。

> **Note**
>
> この解決策は、Visual Studio のカスタマイズに影響を与える可能性があります。この手順を完了すると、 AWS Toolkit 拡張機能はインストール済みとしてリストされ、エラーメッセージは報告されなくなります。次のステップを完了した後もこの問題が解決しない場合は、AWS Toolkit for Visual Studio GitHub リポジトリの「問題 #452」を参照してください。

1. Visual Studio 2022 の最新バージョンをインストールします。

   > **Note**
   >
   > 最小必須バージョンは 17.11.5 です。

2. Visual Studio で実行中のすべてのインスタンスを閉じます。
3. Windows で、管理者として [Developer Command Prompt] を開きます。
4. [Developer Command Prompt] から、`devenv /updateconfiguration /resetExtensions` コマンドを実行し、コマンドが完了するのを待ちます。
5. コマンドが完了したら、Visual Studio を再起動します。
6. Visual Studio では、 AWS 拡張機能がインストール済みとしてリストされ、この問題の上部にリストされているエラーメッセージは報告されなくなりました。

ファイアウォールとプロキシの設定

ファイアウォールとプロキシの設定のトラブルシューティング

セキュリティスキャンソフトウェアは、ダウンロードフォルダからファイルを削除したり、ダウンロード自体をブロックしたりすることで、 AWS Toolkit 言語サーバーからのファイルのダウンロードを妨げる可能性があります。

ファイアウォールとプロキシの設定を確認するには、Visual Studio のインスタンスと同じシステムにインストールされているインターネットブラウザから https://aws-toolkit-language-servers.amazonaws.com/codewhisperer/0/manifest.json に移動します。エラーが発生したり、ページが読み込まれない場合は、ファイアウォールまたはプロキシフィルターによって `aws-toolkit-language-servers.amazonaws.com` に到達できない可能性があります。

カスタム証明書

は、Node.js ランタイムで実行される言語サーバー AWS Toolkit for Visual Studio を使用します。ネットワークでカスタム証明書が使用されているかどうかを確認する方法の詳細については 「AWS Command Line Interface ユーザーガイド」バージョン 1 の「AWS CLIの設定と認証情報ファイルの設定」のトピックを参照してください。

プロキシ設定を構成して証明書を定義するには、`HTTPS_PROXY` 環境変数を設定し、`NODE_OPTIONS` キーと `NODE_EXTRA_CA_CERTS` キー用の Windows 環境変数を作成する必要があります。

`HTTPS_PROXY` 環境変数を構成するには、次の手順を実行します。

1. Visual Studio のメインメニューで、[ツール]、[オプション] の順に選択します。
2. [オプション] メニューで、[AWS Toolkit] を展開して、[プロキシ] を選択します。
3. [プロキシ] メニューから、[ホスト] と [ポート] を定義します。

> **Note**
>
> `HTTPS_PROXY` から を設定する方法については AWS CLI、「 AWS Command Line Interfaceユーザーガイド」の「 の HTTP プロキシ AWS CLIの使用」を参照してください。

次のキーの Windows 環境変数を作成します。

- `NODE_OPTIONS = --use-openssl-ca`
- `NODE_EXTRA_CA_CERTS = Path/To/Corporate/Certs`

> **Note**
>
> 企業のルート証明書の抽出方法の詳細については、learn.microsoft.com の「Export a certificate with its private key」の記事をご覧ください。Windows 環境変数のキーの詳細については、nodejs.org の Node.js v23.3.0 のドキュメントをご覧ください。

許可リストへの登録と追加ステップ

Toolkit AWS 言語サーバーとの干渉に加えて、ファイアウォール設定により、Amazon Q が Amazon S3 にアップロードしてサービス API を呼び出すことを防ぐことができます。これらのエラーの可能性を最小限に抑えるため、以下のエンドポイントに対して、ポート 443 (HTTPS) でのアウトバウンドインターネットアクセスを許可することをお勧めします。

- `https://codewhisperer.us-east-1.amazonaws.com/`
- `https://amazonq-code-transformation-us-east-1-c6160f047e0.s3.amazonaws.com/`
- `https://aws-toolkit-language-servers.amazonaws.com/`
- `https://q.us-east-1.amazonaws.com`
- `https://client-telemetry.us-east-1.amazonaws.com`
- `https://cognito-identity.us-east-1.amazonaws.com`
- `https://oidc.us-east-1.amazonaws.com`

エンドポイントの詳細なリストについては、このユーザーガイドの「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」のトピックを参照してください。Amazon Q の企業プロキシの設定の詳細については、「Amazon Q Developer ユーザーガイド」の「Amazon Q での企業プロキシの設定」のトピックを参照してください。ファイアウォールとプロキシの問題が引き続き発生する場合は、 AWS Toolkit Logs を収集し、 AWS Toolkit for Visual Studio GitHub リポジトリAWS Toolkit for Visual Studio の問題セクションを通じて AWS Toolkit for Visual Studio チームに連絡してください。AWS Toolkit Logs の収集の詳細については、このユーザーガイドトピックの「トラブルシューティングのベストプラクティス」セクションの情報を参照してください。

のセキュリティ AWS Toolkit for Visual Studio

クラウドセキュリティは Amazon Web Services (AWS) の最優先事項です。 AWS のお客様は、セキュリティを非常に重視する組織の要件を満たせるように構築されたデータセンターとネットワークアーキテクチャーから利点を得ます。セキュリティは、 AWS とお客様の間の責任共有です。責任共有モデルでは、これをクラウドのセキュリティおよびクラウド内のセキュリティとして説明しています。

クラウドのセキュリティ – AWS クラウドで提供されるすべてのサービスを実行するインフラストラクチャ AWS を保護し、安全に使用できるサービスを提供します。における当社のセキュリティ責任は最優先事項であり AWS、当社のセキュリティの有効性は、AWS コンプライアンスプログラムの一環としてサードパーティーの監査者によって定期的にテストおよび検証されます。

クラウド内のセキュリティ – お客様の責任は、使用している AWS サービス、データの機密性、組織の要件、適用される法律や規制などのその他の要因によって決まります。

この AWS 製品またはサービスは、サポートする特定の Amazon Web Services (AWS) サービスを通じて責任共有モデルに従います。 AWS サービスセキュリティ情報については、AWS 「サービスセキュリティドキュメント」ページとAWS 、コンプライアンスプログラムによる AWS コンプライアンスの取り組みの対象となるサービスを参照してください。

トピック

- でのデータ保護 AWS Toolkit for Visual Studio
- Identity and Access Management
- この AWS 製品またはサービスのコンプライアンス検証
- この AWS 製品またはサービスの耐障害性
- この AWS 製品またはサービスのインフラストラクチャセキュリティ
- での設定と脆弱性の分析 AWS Toolkit for Visual Studio

でのデータ保護 AWS Toolkit for Visual Studio

AWS 責任共有モデル、Amazon Q を使用した AWS Toolkit for Visual Studio 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、 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 AWS 責任があります AWS クラウド。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。

データ保護の目的で、認証情報を保護し AWS アカウント 、 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設定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 は必須ですが、TLS 1.3 を推奨します。
- で API とユーザーアクティビティのログ記録を設定します AWS CloudTrail。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 暗号化ソリューションと、 内のすべてのデフォルトのセキュリティコントロールを使用します AWS のサービス。
- 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 AWS を介して にアクセスするときに FIPS 140-3 検証済み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を、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含めないことを強くお勧めします。これは、コンソール、API、または SDK を使用して Amazon Q または他の AWS のサービス で AWS Toolkit AWS CLIを使用する場合も同様です。 AWS SDKs 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。

Identity and Access Management

AWS Identity and Access Management (IAM) は、管理者が AWS リソースへのアクセスを安全に制御 AWS のサービス するのに役立つ です。IAM 管理者は、誰を認証 (サインイン) し、誰に AWS リ

ソースの使用を許可する (アクセス許可を付与する) かを制御します。IAM は、追加料金なしで AWS のサービス 使用できる です。

トピック

オーディエンス

AWS Identity and Access Management (IAM) の使用方法は、 で行う作業によって異なります AWS。

サービスユーザー – AWS のサービス を使用してジョブを実行する場合、管理者から必要な認証情報とアクセス許可が提供されます。さらに多くの AWS 機能を使用して作業を行う場合は、追加のアクセス許可が必要になる場合があります。アクセスの管理方法を理解すると、管理者に適切なアクセス許可をリクエストするのに役に立ちます。の機能にアクセスできない場合は AWS、 AWS のサービス AWS ID とアクセスのトラブルシューティング「」または使用している のユーザーガイドを参照してください。

サービス管理者 – 社内の AWS リソースを担当している場合は、通常、 へのフルアクセスがあります AWS。サービスユーザーがどの AWS 機能やリソースにアクセスするかを決めるのは管理者の仕事です。その後、IAM 管理者にリクエストを送信して、サービスユーザーの権限を変更する必要があります。このページの情報を点検して、IAM の基本概念を理解してください。会社で IAM を使用する方法の詳細については AWS、使用している AWS のサービス のユーザーガイドを参照してください。

IAM 管理者 - 管理者は、 AWSへのアクセスを管理するポリシーの書き込み方法の詳細について確認する場合があります。IAM で使用できる AWS アイデンティティベースのポリシーの例を表示するには、 AWS のサービス 使用している のユーザーガイドを参照してください。

アイデンティティを使用した認証

認証とは、ID 認証情報 AWS を使用して にサインインする方法です。、IAM ユーザー AWS アカウントのルートユーザー、または IAM ロールを引き受けることで認証される必要があります。

AWS IAM アイデンティティセンター (IAM Identity Center)、シングルサインオン認証、Google/Facebook 認証情報などの ID ソースからの認証情報を使用して、フェデレーティッド ID としてサインインできます。サインインの詳細については、「AWS サインイン ユーザーガイド」の「AWS アカウントにサインインする方法」を参照してください。

プログラムによるアクセスの場合、 は SDK と CLI AWS を提供してリクエストを暗号化して署名します。詳細については、「IAM ユーザーガイド」の「API リクエストに対するAWS 署名バージョン 4」を参照してください。

AWS アカウント ルートユーザー

を作成するときは AWS アカウント、すべての AWS のサービス および リソースへの完全なアクセス権を持つ AWS アカウント ルートユーザーと呼ばれる 1 つのサインインアイデンティティから始めます。日常的なタスクには、ルートユーザーを使用しないことを強くお勧めします。ルートユーザー認証情報を必要とするタスクについては、「IAM ユーザーガイド」の「ルートユーザー認証情報が必要なタスク」を参照してください。

フェデレーテッドアイデンティティ

ベストプラクティスとして、人間のユーザーが一時的な認証情報 AWS のサービス を使用して にアクセスするには、ID プロバイダーとのフェデレーションを使用する必要があります。

フェデレーティッド ID は、エンタープライズディレクトリ、ウェブ ID プロバイダー、または ID Directory Service ソースの認証情報 AWS のサービス を使用して にアクセスするユーザーです。フェデレーテッドアイデンティティは、一時的な認証情報を提供するロールを引き受けます。

アクセスを一元管理する場合は、 AWS IAM アイデンティティセンターをお勧めします。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とは」を参照してください。

IAM ユーザーとグループ

IAM ユーザーは、特定の個人やアプリケーションに対する特定のアクセス許可を持つアイデンティティです。長期認証情報を持つ IAM ユーザーの代わりに一時的な認証情報を使用することをお勧めします。詳細については、IAM ユーザーガイドの「ID プロバイダーとのフェデレーションを使用して にアクセスする必要がある AWS」を参照してください。

IAM グループは、IAM ユーザーの集合を指定し、大量のユーザーに対するアクセス許可の管理を容易にします。詳細については、「IAM ユーザーガイド」の「IAM ユーザーに関するユースケース」を参照してください。

IAM ロール

IAM ロールは、特定のアクセス許可を持つアイデンティであり、一時的な認証情報を提供します。ユーザーから IAM ロール (コンソール) に切り替えるか、 または API オペレーションを呼び出すことで、ロールを引き受けることができます。 AWS CLI AWS 詳細については、「IAM ユーザーガイド」の「ロールを引き受けるための各種方法」を参照してください。

IAM ロールは、フェデレーションユーザーアクセス、一時的な IAM ユーザーのアクセス許可、クロスアカウントアクセス、クロスサービスアクセス、および Amazon EC2 で実行するアプリケーションに役立ちます。詳細については、IAM ユーザーガイド の IAM でのクロスアカウントリソースアクセス を参照してください。

ポリシーを使用したアクセスの管理

でアクセスを制御する AWS には、ポリシーを作成し、ID AWS またはリソースにアタッチします。ポリシーは、アイデンティティまたはリソースに関連付けられたときにアクセス許可を定義します。は、プリンシパルがリクエストを行うときにこれらのポリシー AWS を評価します。ほとんどのポリシーは JSON ドキュメント AWS として に保存されます。JSON ポリシードキュメントの詳細については、「IAM ユーザーガイド」の「JSON ポリシー概要」を参照してください。

管理者は、ポリシーを使用して、どのプリンシパルがどのリソースに対して、どのような条件でアクションを実行できるかを定義することで、誰が何にアクセスできるかを指定します。

デフォルトでは、ユーザーやロールにアクセス許可はありません。IAM 管理者は IAM ポリシーを作成してロールに追加し、このロールをユーザーが引き受けられるようにします。IAM ポリシーは、オペレーションの実行方法を問わず、アクセス許可を定義します。

アイデンティティベースのポリシー

アイデンティティベースのポリシーは、アイデンティティ (ユーザー、グループ、またはロール) にアタッチできる JSON アクセス許可ポリシードキュメントです。これらのポリシーは、アイデンティティがどのリソースに対してどのような条件下でどのようなアクションを実行できるかを制御します。アイデンティティベースポリシーの作成方法については、IAM ユーザーガイド の カスタマー管理ポリシーでカスタム IAM アクセス許可を定義する を参照してください。

アイデンティティベースのポリシーは、インラインポリシー (単一の ID に直接埋め込む) または管理ポリシー (複数の ID にアタッチされたスタンドアロンポリシー) にすることができます。管理ポリシーとインラインポリシーのいずれかを選択する方法については、「IAM ユーザーガイド」の「管理ポリシーとインラインポリシーのいずれかを選択する」を参照してください。

リソースベースのポリシー

リソースベースのポリシーは、リソースに添付する JSON ポリシードキュメントです。例としては、IAM ロール信頼ポリシーや Amazon S3 バケットポリシーなどがあります。リソースベースのポリシーをサポートするサービスでは、サービス管理者はポリシーを使用して特定のリソースへのアクセスを制御できます。リソースベースのポリシーでは、プリンシパルを指定する必要があります。

リソースベースのポリシーは、そのサービス内にあるインラインポリシーです。リソースベースのポリシーでは、IAM の AWS マネージドポリシーを使用できません。

アクセスコントロールリスト (ACL)

アクセスコントロールリスト (ACL) は、どのプリンシパル (アカウントメンバー、ユーザー、またはロール) がリソースにアクセスするためのアクセス許可を持つかを制御します。ACL はリソースベースのポリシーに似ていますが、JSON ポリシードキュメント形式は使用しません。

Amazon S3、および Amazon VPC は AWS WAF、ACLs。ACL の詳細については、Amazon Simple Storage Service デベロッパーガイド の アクセスコントロールリスト (ACL) の概要 を参照してください。

その他のポリシータイプ

AWS は、より一般的なポリシータイプによって付与されるアクセス許可の最大数を設定できる追加のポリシータイプをサポートしています。

- アクセス許可の境界 – アイデンティティベースのポリシーで IAM エンティティに付与することのできるアクセス許可の数の上限を設定します。詳細については、「IAM ユーザーガイド」の「IAM エンティティのアクセス許可境界」を参照してください。
- サービスコントロールポリシー (SCP) - AWS Organizations内の組織または組織単位の最大のアクセス許可を指定します。詳細については、「AWS Organizations ユーザーガイド」の「サービスコントロールポリシー」を参照してください。
- リソースコントロールポリシー (RCP) – は、アカウント内のリソースで利用できる最大数のアクセス許可を定義します。詳細については、「AWS Organizations ユーザーガイド」の「リソースコントロールポリシー (RCP)」を参照してください。
- セッションポリシー – ロールまたはフェデレーションユーザーの一時セッションを作成する際にパラメータとして渡される高度なポリシーです。詳細については、「IAM ユーザーガイド」の「セッションポリシー」を参照してください。

複数のポリシータイプ

1 つのリクエストに複数のタイプのポリシーが適用されると、結果として作成されるアクセス許可を理解するのがさらに難しくなります。が複数のポリシータイプが関与する場合にリクエストを許可するかどうか AWS を決定する方法については、「IAM ユーザーガイド」の「ポリシー評価ロジック」を参照してください。

IAM AWS のサービス の操作方法

ほとんどの IAM 機能と AWS のサービス の連携方法の概要については、「IAM ユーザーガイド」のAWS 「IAM と連携する のサービス」を参照してください。

IAM AWS のサービス で特定の を使用する方法については、関連するサービスのユーザーガイドのセキュリティセクションを参照してください。

AWS ID とアクセスのトラブルシューティング

次の情報は、 および IAM の使用時に発生する可能性がある一般的な問題の診断 AWS と修復に役立ちます。

トピック

- でアクションを実行する権限がありません AWS
- iam:PassRole を実行する権限がありません
- 自分の 以外のユーザーに自分の AWS リソース AWS アカウント へのアクセスを許可したい

でアクションを実行する権限がありません AWS

アクションを実行する権限がないというエラーが表示された場合は、そのアクションを実行できるようにポリシーを更新する必要があります。

次のエラー例は、mateojackson IAM ユーザーがコンソールを使用して、ある *my-example-widget* リソースに関する詳細情報を表示しようとしたことを想定して、その際に必要な awes:*GetWidget* アクセス許可を持っていない場合に発生するものです。

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 awes:GetWidget on resource: my-example-widget
```

この場合、awes:*GetWidget* アクションを使用して *my-example-widget* リソースへのアクセスを許可するように、mateojackson ユーザーのポリシーを更新する必要があります。

サポートが必要な場合は、 AWS 管理者にお問い合わせください。サインイン認証情報を提供した担当者が管理者です。

iam:PassRole を実行する権限がありません

`iam:PassRole` アクションを実行する権限がないというエラーが表示された場合は、ポリシーを更新して AWSにロールを渡すことができるようにする必要があります。

一部の AWS のサービス では、新しいサービスロールまたはサービスにリンクされたロールを作成する代わりに、既存のロールをそのサービスに渡すことができます。そのためには、サービスにロールを渡す権限が必要です。

以下の例のエラーは、`marymajor` という IAM ユーザーがコンソールを使用して AWSでアクションを実行しようとする場合に発生します。ただし、このアクションをサービスが実行するには、サービスロールから付与された権限が必要です。Mary には、ロールをサービスに渡すアクセス許可がありません。

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

この場合、Mary のポリシーを更新してメアリーに `iam:PassRole` アクションの実行を許可する必要があります。

サポートが必要な場合は、 AWS 管理者にお問い合わせください。サインイン資格情報を提供した担当者が管理者です。

自分の 以外のユーザーに自分の AWS リソース AWS アカウント へのアクセスを許可したい

他のアカウントのユーザーや組織外の人が、リソースにアクセスするために使用できるロールを作成できます。ロールの引き受けを委託するユーザーを指定できます。リソースベースのポリシーまたはアクセスコントロールリスト (ACL) をサポートするサービスの場合、それらのポリシーを使用して、リソースへのアクセスを付与できます。

詳細については、以下を参照してください:

- がこれらの機能 AWS をサポートしているかどうかを確認するには、「」を参照してくださいIAM AWS のサービス の操作方法。

- 所有 AWS アカウント している のリソースへのアクセスを提供する方法については、「IAM ユーザーガイド」の「所有 AWS アカウント している別の の IAM ユーザーへのアクセスを提供する」を参照してください。
- リソースへのアクセスをサードパーティーに提供する方法については AWS アカウント、IAM ユーザーガイドの「サードパーティー AWS アカウント が所有する へのアクセスを提供する」を参照してください。
- ID フェデレーションを介してアクセスを提供する方法については、IAM ユーザーガイド の 外部で認証されたユーザー (ID フェデレーション) へのアクセスの許可 を参照してください。
- クロスアカウントアクセスにおけるロールとリソースベースのポリシーの使用方法の違いについては、IAM ユーザーガイド の IAM でのクロスアカウントのリソースへのアクセス を参照してください。

この AWS 製品またはサービスのコンプライアンス検証

AWS のサービス が特定のコンプライアンスプログラムの範囲内にあるかどうかを確認するには、「コンプライアンスAWS のサービス プログラムによるスコープ」の「コンプライアンス」を参照して、関心のあるコンプライアンスプログラムを選択します。一般的な情報については、AWS 「コンプライアンスプログラム」を参照してください。

を使用して、サードパーティーの監査レポートをダウンロードできます AWS Artifact。詳細については、「Downloading Reports in AWS Artifact」を参照してください。

を使用する際のお客様のコンプライアンス責任 AWS のサービス は、お客様のデータの機密性、貴社のコンプライアンス目的、適用される法律および規制によって決まります。を使用する際のコンプライアンス責任の詳細については AWS のサービス、AWS 「 セキュリティドキュメント」を参照してください。

この AWS 製品またはサービスは、サポートする特定の Amazon Web Services (AWS) サービスを通じて責任共有モデルに従います。 AWS サービスセキュリティ情報については、AWS 「サービスセキュリティドキュメント」ページとAWS 、コンプライアンスプログラムによる AWS コンプライアンスの取り組みの対象となるサービスを参照してください。

この AWS 製品またはサービスの耐障害性

AWS グローバルインフラストラクチャは、 AWS リージョン およびアベイラビリティーゾーンを中心に構築されています。

AWS リージョン は、低レイテンシー、高スループット、および高度に冗長なネットワークで接続された複数の物理的に分離されたアベイラビリティーゾーンを提供します。

アベイラビリティーゾーンでは、ゾーン間で中断することなく自動的にフェールオーバーするアプリケーションとデータベースを設計および運用することができます。アベイラビリティーゾーンは、従来の単一または複数のデータセンターインフラストラクチャよりも可用性、フォールトトレランス、および拡張性が優れています。

AWS リージョンとアベイラビリティーゾーンの詳細については、AWS 「 グローバルインフラストラクチャ」を参照してください。

この AWS 製品またはサービスは、サポートする特定の Amazon Web Services (AWS) サービスを通じて責任共有モデルに従います。 AWS サービスセキュリティ情報については、AWS 「サービスセキュリティドキュメント」ページとAWS 、コンプライアンスプログラムによる AWS コンプライアンスの取り組みの対象となるサービスを参照してください。

この AWS 製品またはサービスのインフラストラクチャセキュリティ

この AWS 製品またはサービスはマネージドサービスを使用するため、 グローバルネットワークセキュリティによって AWS 保護されています。 AWS セキュリティサービスと がインフラストラクチャ AWS を保護する方法については、AWS 「 クラウドセキュリティ」を参照してください。インフラストラクチャセキュリティのベストプラクティスを使用して AWS 環境を設計するには、「Security Pillar AWS Well-Architected Framework」の「Infrastructure Protection」を参照してください。

AWS 公開された API コールを使用して、ネットワーク経由でこの AWS 製品またはサービスにアクセスします。クライアントは次をサポートする必要があります。

- Transport Layer Security (TLS)。TLS 1.2 が必須で、TLS 1.3 をお勧めします。
- DHE (楕円ディフィー・ヘルマン鍵共有) や ECDHE (楕円曲線ディフィー・ヘルマン鍵共有) などの完全前方秘匿性 (PFS) による暗号スイート。これらのモードは Java 7 以降など、ほとんどの最新システムでサポートされています。

また、リクエストにはアクセスキー ID と、IAM プリンシパルに関連付けられているシークレットアクセスキーを使用して署名する必要があります。または、AWS Security Token Service (AWS STS) を使用して、一時的なセキュリティ認証情報を生成し、リクエストに署名することもできます。

この AWS 製品またはサービスは、サポートする特定の Amazon Web Services (AWS) サービスを通じて責任共有モデルに従います。 AWS サービスセキュリティ情報については、AWS 「サービスセキュリティドキュメント」ページとAWS 、コンプライアンスプログラムによる AWS コンプライアンスの取り組みの対象となるサービスを参照してください。

での設定と脆弱性の分析 AWS Toolkit for Visual Studio

新しい機能や修正が開発されると、Toolkit for Visual Studio が Visual Studio Marketplace にリリースされます。これらの更新にはセキュリティ更新が含まれる場合があるため、 AWS Toolkit with Amazon Q を最新の状態に保つことが重要です。

拡張機能の自動更新が有効になっていることを確認するには

1. [ツール] を選択し、Visual Studio 2017 の場合には [拡張機能と更新プログラム、Visual Studio 2019 の場合には [拡張機能]、[拡張機能の管理] の順に選択します。
2. [Change your Extensions and Updates settings] (拡張機能と更新設定の変更) (Visual Studio 2017) または [Change your settings for Extensions] (拡張機能と設定の変更) (Visual Studio 2019) を選択します。
3. 環境の設定を調整します。

拡張機能の自動更新を無効にする場合は、 AWS Toolkit with Amazon Q の更新が環境に適した間隔で実行されていることを必ず確認してください。

AWS Toolkit for Visual Studio ユーザーガイドのドキュメント履歴

ドキュメント履歴

次の表は、AWS Toolkit for Visual Studio ユーザーガイドの最近の変更点をまとめたものです。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
開始方法のコンテンツの更新	UI に加えられた変更を反映するために、「開始方法」および「AWS への接続」コンテンツが更新されました。	2025 年 4 月 24 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	拡張機能用の Amazon Q を使用して AWS Toolkit for Visual Studio 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 3 月 20 日
ファイアウォールとプロキシの設定のトラブルシューティング	AWS Toolkit for Visual Studio および Amazon Q のファイアウォールとプロキシの設定に関する新しいトラブルシューティングトピックを追加しました。	2024 年 12 月 15 日
インストールのトラブルシューティングの更新	Microsoft からの更新を考慮してインストールの問題コンテンツを更新します。	2024 年 11 月 20 日
開始方法のコンテンツの更新	UI に加えられた変更を反映するために、「開始方法」およ	2024 年 10 月 24 日

	び「AWS への接続」コンテンツが更新されました。	
AWS への接続の更新	AWS への接続コンテンツの更新。	2024 年 9 月 26 日
Amazon EC2 AMI コンテンツの更新	Amazon EC2 AMI のプロセスと手順の変更を文書化するために、コンテンツが更新されました。	2024 年 9 月 13 日
AWS ツールキットコンポーネントを初期化できませんでした	初期化されていない AWS Toolkit for Visual Studio コンポーネントの問題に対処するためのトラブルシューティングトピックを追加しました。	2024 年 9 月 13 日
Amazon Q セキュリティスキャンの表示とフィルタリング	Amazon Q セキュリティスキャンの表示とフィルタリングに役立つトラブルシューティングトピックを追加しました。	2024 年 7 月 31 日
Amazon Q for AWS Toolkit for Visual Studio	Amazon Q が AWS Toolkit for Visual Studio で利用可能になりました。	2024 年 6 月 30 日
コンテンツの更新とメンテナンス	UI および AWS スタイルガイドラインの変更に関するコンテンツの更新。	2024 年 3 月 6 日
コンテンツの更新とメンテナンス	UI および AWS スタイルガイドラインの変更に関するコンテンツの更新。	2024 年 3 月 6 日
コンテンツの更新とメンテナンス	UI および AWS スタイルガイドラインの変更に関するコンテンツの更新。	2024 年 3 月 6 日

コンテンツの更新とメンテナンス	UI および AWS スタイルガイドラインの変更に関するコンテンツの更新。	2024 年 3 月 6 日
コンテンツの更新とメンテナンス	UI および AWS スタイルガイドラインの変更に関するコンテンツの更新。	2024 年 3 月 6 日
セットアップと認証を更新	セキュリティとツールキットのオンボーディングエクスペリエンスを向上させるため、「セットアップと認証」のトピックが更新されました。変更点については、「開始方法」および「認証とアクセス」トピックの目次を参照してください。	2023 年 6 月 22 日
認証とアクセス	「AWS 認証情報の提供」は「認証とアクセス」になりました。AWS スタイルとセキュリティの要件を満たすように目次とサブトピックをリファクタリングしました。	2023 年 5 月 4 日
セットアップ」セクションとトピックを更新	AWS Toolkit for Visual Studio のオンボーディングエクスペリエンスを向上させるため、このユーザーガイドの「AWS Toolkit for Visual Studio のセットアップ」のセクションとトピックが更新されました。	2023 年 1 月 30 日

セットアップ」セクションとトピックを更新	AWS Toolkit for Visual Studio のオンボーディングエクスペリエンスを向上させるため、このユーザーガイドの「AWS Toolkit for Visual Studio のセットアップ」のセクションとトピックが更新されました。	2023 年 1 月 30 日
AWS Toolkit for Visual Studio 2022 の情報を追加しました。	Visual Studio 2022 のサポートが、AWS Toolkit for Visual Studio に追加されました。	2022 年 12 月 20 日
Publish to AWS ガイドの更新	一般提供の開始に向けてサービスに加えられた変更を反映するために、ドキュメントを更新しました。	2022 年 7 月 6 日
タイトルの更新と再配置	内容をより正確に反映するため、タイトルを若干変更しました。現在、ガイドは「AWS への公開」ガイドで参照できます。	2022 年 7 月 6 日

AWS へのデプロイ: タイトルとコンテンツの更新	このガイドのセクションの正式なタイトルは「AWS Toolkit を使用したデプロイ」でしたが、目次が更新され、「AWS へのデプロイ」となりました。「Elastic Beanstalk へのデプロイ (レガシー)」および「AWS CloudFormation へのデプロイ (レガシー)」のガイドは廃止となり、アクセスできなくなりました。Elastic Beanstalk と Cloudformation へのデプロイに関する更新されたコンテンツは、このガイドの更新された目次から参照できます。	2022 年 7 月 6 日
ASP.NET Core 2.0 アプリケーションのデプロイ (Fargate)」はレガシーガイドになりました	このドキュメントでは、従来のサービスと機能について言及しています。更新されたガイドとコンテンツについては、「AWS .NET Deployment tool」ガイドと更新された目次の「AWS へのデプロイ」を参照してください。	2022 年 7 月 6 日
ASP.NET アプリケーションのデプロイ」はレガシーガイドになりました	このドキュメントでは、従来のサービスと機能について言及しています。更新されたガイドとコンテンツについては、「AWS .NET Deployment tool」ガイドと、目次にある更新された「AWS へのデプロイ」を参照してください。	2022 年 7 月 6 日

ASP.NET アプリケーションのデプロイ」はレガシーガイドになりました	このドキュメントでは、従来のサービスと機能について言及しています。更新されたガイドとコンテンツについては、「AWS .NET Deployment tool」ガイドと、目次にある更新された「AWS へのデプロイ」を参照してください。	2022 年 7 月 6 日
新しいガイドトピック: Visual Studio 上での CloudWatch Logs の操作	「Visual Studio での Amazon CloudWatch Logs の統合」ガイドの概要のトピックを新たに作成しました。	2022 年 1 月 29 日
新しいガイドトピック: CloudWatch Logs の Visual Studio への統合をセットアップする	「Visual Studio での Amazon CloudWatch Logs の統合」ガイドに新しいセットアップセクションを作成しました。	2022 年 1 月 29 日
CloudWatch Logs の Visual Studio への統合	Amazon CloudWatch Logs の Visual Studio への統合に関する新しいガイドを作成しました。これには「Visual Studio での CloudWatch Logs のセットアップ」と「Visual Studio 上での CloudWatch Logs の操作」というガイドトピックが含まれます。	2022 年 1 月 29 日
Publish to AWS	Publish to AWS はプレビュー版で利用できなくなりました。UI の変更ならびに発行の提案に関する改善を反映するために更新しました。	2022 年 6 月 1 日

新しい Publish to AWS がプレビュー可能	アプリケーションに適した AWS サービスを示唆するガイダンスの提供によるデプロイエクスペリエンスの強化。	2021 年 10 月 21 日
AWS 認識情報のための SSO および MFA のサポート	AWS 認証情報における AWS シングルサインオン (IAM アイデンティティセンター) および多要素認証の新しいサポートに関するドキュメントが更新されました。	2021 年 4 月 21 日
基本 AWS Lambda プロジェクトによる Docker イメージの作成	Lambda コンテナイメージのサポートが追加されました。	2020 年 12 月 1 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
AWS 認証情報の提供	共有 AWS 認証情報ファイルでの認証情報プロファイルの作成に関する情報が更新されました。	2019 年 6 月 20 日
AWS Toolkit for Visual Studio での AWS Lambda の使用	Visual Studio 2019 のサポートが、AWS Toolkit for Visual Studio に追加されました。	2019 年 3 月 28 日
チュートリアル: Amazon Rekognition Lambda アプリケーションの作成	Visual Studio 2019 のサポートが、AWS Toolkit for Visual Studio に追加されました。	2019 年 3 月 28 日
チュートリアル: AWS Lambda でのサーバーレスアプリケーションの構築およびテスト	Visual Studio 2019 のサポートが、AWS Toolkit for Visual Studio に追加されました。	2019 年 3 月 28 日

のセットアップAWS Toolkit for Visual Studio	Visual Studio 2019 のサポートが、AWS Toolkit for Visual Studio に追加されました。	2019 年 3 月 28 日
ASP.NET Core 2.0 アプリケーションのデプロイ (Fargate)	Visual Studio 2019 のサポートが、AWS Toolkit for Visual Studio に追加されました。	2019 年 3 月 28 日
ASP.NET Core 2.0 アプリケーションのデプロイ (EC2)	Visual Studio 2019 のサポートが、AWS Toolkit for Visual Studio に追加されました。	2019 年 3 月 28 日
Visual Studio での AWS CloudFormation テンプレートプロジェクトの作成	Visual Studio 2019 のサポートが、AWS Toolkit for Visual Studio に追加されました。	2019 年 3 月 28 日
Container Service の詳細ビュー	AWS Explorer によって提供される Amazon Elastic Container Service クラスターおよびコンテナリポジトリの詳細なビューに関する情報を追加しました。	2018 年 2 月 16 日
Amazon EC2 Container Service へのデプロイ	Amazon EC2 Container Service へのデプロイに関する情報を追加しました。	2018 年 2 月 16 日
Fargate の使用による Container Service のデプロイ	Fargate 起動タイプを使用して Amazon ECS を通じて Linux をターゲットとするコンテナ化された ASP.NET Core 2.0 アプリケーションをデプロイする方法についての情報を追加しました。	2018 年 2 月 16 日

EC2 の使用による Container Service のデプロイ	EC2 起動タイプを使用して Amazon ECS を通じて Linux をターゲットとするコンテナ化された ASP.NET Core 2.0 アプリケーションをデプロイする方法についての情報を追加しました。	2018 年 2 月 16 日
Amazon EC2 Container Service にデプロイするための認証情報	Amazon EC2 Container Service にデプロイするときに認証情報を指定する方法についての情報を追加しました。	2018 年 2 月 16 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。