aws

ユーザーガイド

AWS Toolkit for JetBrains

AWS Toolkit for JetBrains: ユーザーガイド

Table of Contents

AWS Toolkit for JetBrains とは

AWS Toolkit for JetBrains は、 JetBrains からの統合開発環境 (IDE) 用のオープンソースプラグインです。本ツールキットは、AWS リソースを JetBrains IDE から利用可能にすることにより、Amazon Web Services (AWS) を使用してサーバーレスアプリケーションの開発、デバッグ、デプロイをサポートします。

トピック

AWS Toolkit for JetBrains に含まれるもの

AWS Toolkit for JetBrains には以下のツールキットが含まれます:

- CLion 用 AWS Toolkit (C と C++ 開発用)
- GoLand 用 AWS Toolkit (Go 開発用)
- IntelliJ (Java 開発用) 用 AWS Toolkit
- WebStorm (Node.js 開発用) 用 AWS Toolkit
- Rider (.NET 開発用) 用 AWS Toolkit
- PhpStorm 用 AWS Toolkit (PHP 開発用)
- PyCharm (Python 開発用) 用 AWS Toolkit
- RubyMine 用 AWS Toolkit (Ruby 開発用)
- DataGrip 用 AWS Toolkit (データベース管理用)

> **Note**
>
> サポートされている JetBrains IDE のAWS ツールキット間には、機能性に有意な違いがあることを本ガイドに記載しています。

AWS Toolkit for JetBrains は、AWS Lambda 関数、AWS CloudFormation スタック、Amazon Elastic Container Service (Amazon ECS) クラスターと連動して使用することもできます。AWS Toolkit for

JetBrains には、AWS のアプリケーションの記述を簡単にする AWS 認証情報の管理や AWS リージョンの管理のような機能が含まれます。

AWS Toolkit for JetBrains の操作

AWS Toolkit for JetBrains は、以下のことを実行するために使用できます。

- AWS Serverless Application Model (AWS SAM) アプリケーションを作成、デプロイ、更新、削除します。AWS Toolkit for JetBrains を介して AWS SAM を操作する方法の詳細については、本ユーザーガイドの「AWS サーバーレス」トピックを参照してください。
- AWS Lambda 関数をリモート/ローカルに作成、更新、実行、デバッグします。AWS Toolkit for JetBrains を介した AWS Lambda サービスの操作に関する詳細については、本ユーザーガイドの「AWS Lambda」トピックを参照してください。
- AWS CloudFormation スタックのイベントログを表示、削除します。AWS CloudFormation と AWS Toolkit for JetBrains の操作に関する追加情報については、本ユーザーガイドの「AWS CloudFormation」トピックを参照してください。
- Amazon 伸縮性コンテナサービスを使用して AWS クラスターのコードをデバッグします。AWS Toolkit for JetBrains を使用して Amazon ECS を操作する方法に関する詳細については、本ユーザーガイドの「Amazon Elastic Container Service」トピックを参照してください。
- Amazon EventBridge スキーマを使用します。詳細については、本ユーザーガイドの「Amazon EventBridge スケジューラ」トピックを参照してください。

関連情報

関連動画

- お知らせ | IntelliJ IDEA 用 AWS ツールキットのご紹介(16 分、2019 年 4 月、YouTube ウェブサイト)
- AWS Toolkit for JetBrains の開始方法 (PyCharm の AWS Toolkit についてのみ説明、2 分、2018 年 11 月、YouTube ウェブサイト)
- AWS Toolkit for JetBrains を使用したサーバーレスアプリケーションの構築 (PyCharm の AWS Toolkit についてのみ説明、6 分、2018 年 11 月、YouTube ウェブサイト)

関連ウェブページ

- AWS Toolkit for IntelliJ が正式にリリースされました (2019 年 3 月、ブログ記事、AWS ウェブサイト)
- IntelliJ 用 AWS Toolkit - 正式にリリースされました (2019 年 3 月、ブログ記事、AWS ウェブサイト)
- 新規 - PyCharm 用 AWS ツールキット、IntelliJ (プレビュー) (2018 年 11 月、ブログ記事、AWS ウェブサイト)
- AWS Toolkit for PyCharm の紹介 (2018 年 11 月、ブログ記事、AWS ウェブサイト)
- AWS Toolkit for IntelliJ (AWS Toolkit for JetBrains の一部、AWS ウェブサイト)
- AWS Toolkit for PyCharm (AWS Toolkit for JetBrains の一部、AWS ウェブサイト)
- AWS Toolkit (JetBrains ウェブサイト)
- JetBrains ツールを使用して AWS で開発する (JetBrains ウェブサイト)
- JetBrains が提供するすべてのデベロッパーツールと製品(JetBrains ウェブサイト)

質問とヘルプ

AWS 開発者コミュニティに質問したり、サポートを要請するには、次の AWS ディスカッションフォーラムを参照してください。

- C と C++ 開発
- Go 開発
- Java 開発
- JavaScript 開発
- .NET 開発
- PHP 開発
- Python 開発
- Ruby 開発

(これらのフォーラムにアクセスするには、AWS へのサインインが必要になることがあります)。

当社に直接お問い合わせいただくこともできます。

AWS Toolkit でバグを報告または機能をリクエストする

AWS Toolkit for JetBrains でバグを報告または機能をリクエストするには、GitHub ウェブサイトのaws/aws-toolkit-jetbrains リポジトリの [問題] タブに移動します。[New issue (新しい問題)] を選択し、画面の指示に従ってバグレポートまたは機能リクエストの作成を完了します。(このウェブサイトにアクセスするには、GitHub へのサインインが必要になることがあります)。

AWS Toolkit への寄稿

AWS Toolkit に対する寄稿を大きく評価しています。初めて寄稿するには、GitHub ウェブサイトの「aws/aws-toolkit-jetbrains リポジトリの寄稿ガイドライン」をお読みください。(このウェブサイトにアクセスするには、GitHub へのサインインが必要になることがあります)。

AWS Toolkit for JetBrains の開始方法

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

始めるときの参考になるように、以下のトピックでは AWS Toolkit for JetBrains をインストール、セットアップ、および設定する手順について順を追って説明しています。

トピック

AWS Toolkit for JetBrains のインストール

IDE の JetBrains マーケットプレイスから AWS Toolkit for JetBrains をダウンロード、インストール、セットアップできます。または、ウェブブラウザからAWS Toolkit for JetBrains マーケットプレイスリスティングに移動することで、最新の AWS Toolkit for JetBrains インストールファイルをダウンロードできます。

次のセクションでは、お使いの JetBrains IDE からAWS Toolkit for JetBrainsを直接インストールとセットアップする方法について説明します。

AWS ツールキットを JetBrains IDE からインストールする

ご希望の JetBrains IDE から AWS Toolkit for JetBrains を直接ダウンロードしてインストールするには、次の手順を実行します。

1. JetBrains のメインメニューから [ユーザー設定] メニュー (Windows ユーザーの場合は [ファイル] を展開して [設定] を選択) を開きます。
2. [ユーザー設定]/[設定] メニューから [プラグイン] を選択し、[プラグイン] メニューを開きます。

3. [プラグイン] メニューナビゲーションから [マーケットプレイス] を選択し、JetBrains プラグインの [Marketplace] を開きます。
4. 表示される検索フィールドに **`AWS Toolkit`** と入力します。
5. [AWS ツールキット] プラグインエントリから、エントリ タイトルの横にある緑色の [インストール] ボタンを選択します。
6. [サードパーティプラグインのプライバシー通知]を承諾し、インストールを続行します。
7. インストールが完了すると、JetBrains は IDE の再起動を促します。

AWS Toolkit for JetBrains 早期アクセスプログラム (EAP) とカスタムビルドのインストール

AWS Toolkit for JetBrains の早期アクセスプログラム (EAP) ビルドには、新機能や実験的な機能のプレビューが含まれています。

ツールキットを EAP ビルドに設定するには、次の手順を実行してください

1. JetBrains のメインメニューから [ユーザー設定] メニュー (Windows ユーザーの場合は [ファイル] を展開して [設定] を選択) を開きます。
2. [ユーザー設定]/[設定] メニューから [プラグイン] を選択し、[プラグイン] メニューを開きます。
3. [プラグイン] メニューのナビゲーションから、[設定] (リポジトリの管理、プロキシの設定、ディスクからプラグインをインストールする) アイコンを展開し、[プラグインリポジトリの管理] を選択します。
4. [プラグインリポジトリの管理] メニューで [+ (追加)] アイコンを選択して [AWS ツールキットの EAP リポジトリ] フィールドに **`https://plugins.jetbrains.com/plugins/eap/aws.toolkit`** を入力します。
5. [OK] を選択して EAP のインストールを開始します。
6. インストールが完了すると、JetBrains は IDE を再起動するように促します。

AWS Toolkit for JetBrains EAP とカスタムリポジトリリファレンスの削除

AWS Toolkit for JetBrains の特定バージョンを使用するには、EAP またはカスタムリポジトリリファレンスを削除しなければならない場合があります。リポジトリリファレンスを削除するには、次の手順を実行してください。

> **Note**
>
> この手順を完了した後でも、別のバージョンを更新またはインストールする前に、AWS Toolkit for JetBrainsの現行バージョンのアンインストールが必要な場合があります。

EAP リポジトリリファレンスを削除する方法

1. JetBrains のメインメニューから [ユーザー設定] メニュー (Windows ユーザーの場合は [ファイル] を展開して [設定] を選択) を開きます。
2. [ユーザー設定]/[設定] メニューから [プラグイン] を選択し、[プラグイン] メニューを開きます。
3. [プラグイン] メニューのナビゲーションから、[設定] (リポジトリの管理、プロキシの設定、ディスクからプラグインをインストールする) アイコンを展開し、[プラグインリポジトリの管理] を選択します。
4. [プラグインリポジトリの管理] メニューから [削除] アイコンを選択し、削除を確定します。

AWS Toolkit for JetBrains のナビゲーション

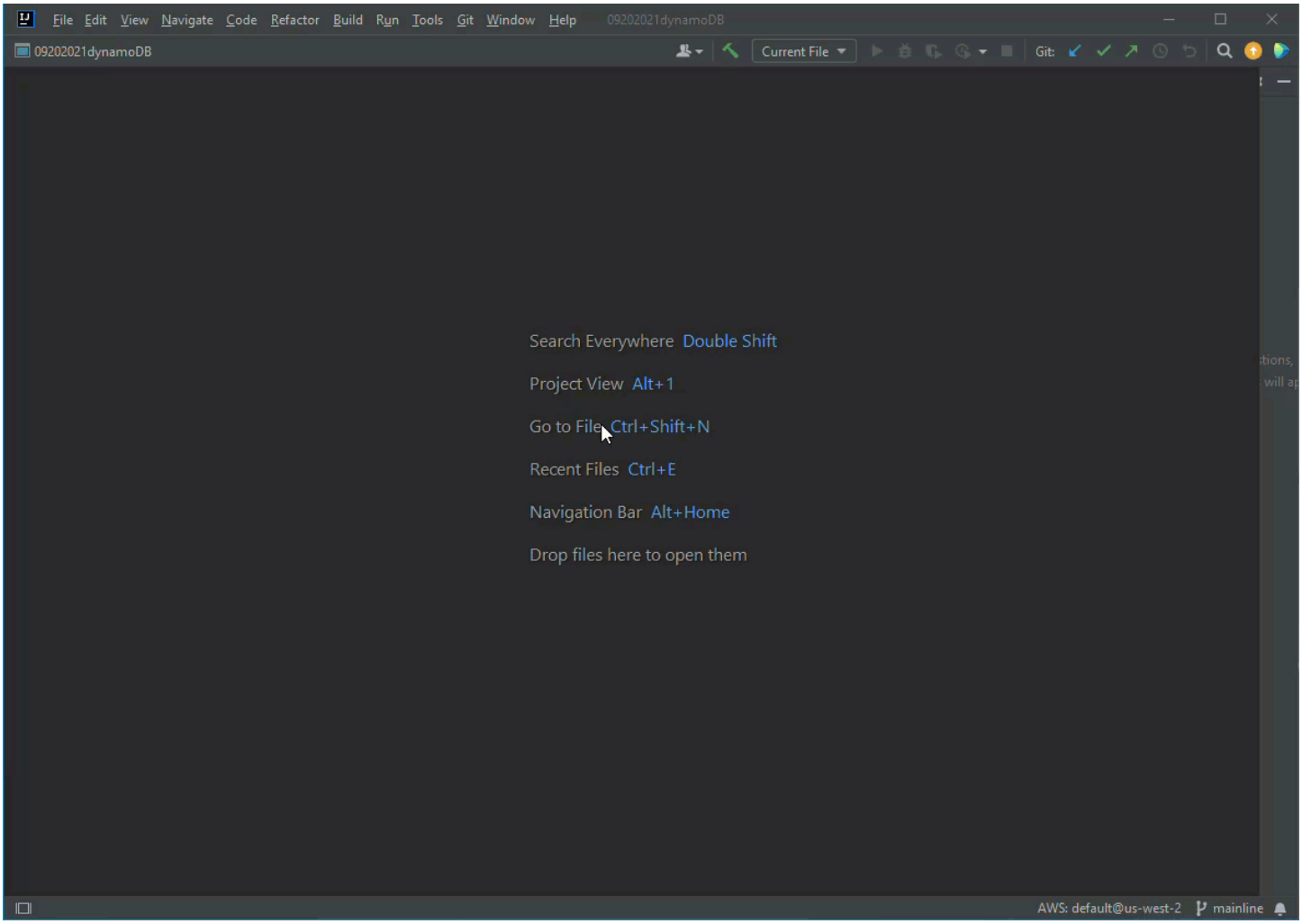
以下のトピックでは、AWS Toolkit for JetBrains の基本的な位置とコンポーネントについて説明します。

トピック

JetBrains からツールキットを表示する

JetBrains IDE 内にツールキットを表示するには、以下の手順を実行します。

1. JetBrains IDE から、JetBrains IDE の左下にある [アクティブツールバー] アイコンを使用して [アクティブツールバー] を展開します。
2. [アクティブツールバー] で、[AWS Toolkit] を選択します。
3. AWS Toolkit for JetBrains が [アクティブツールバー] ウィンドウに開きます。



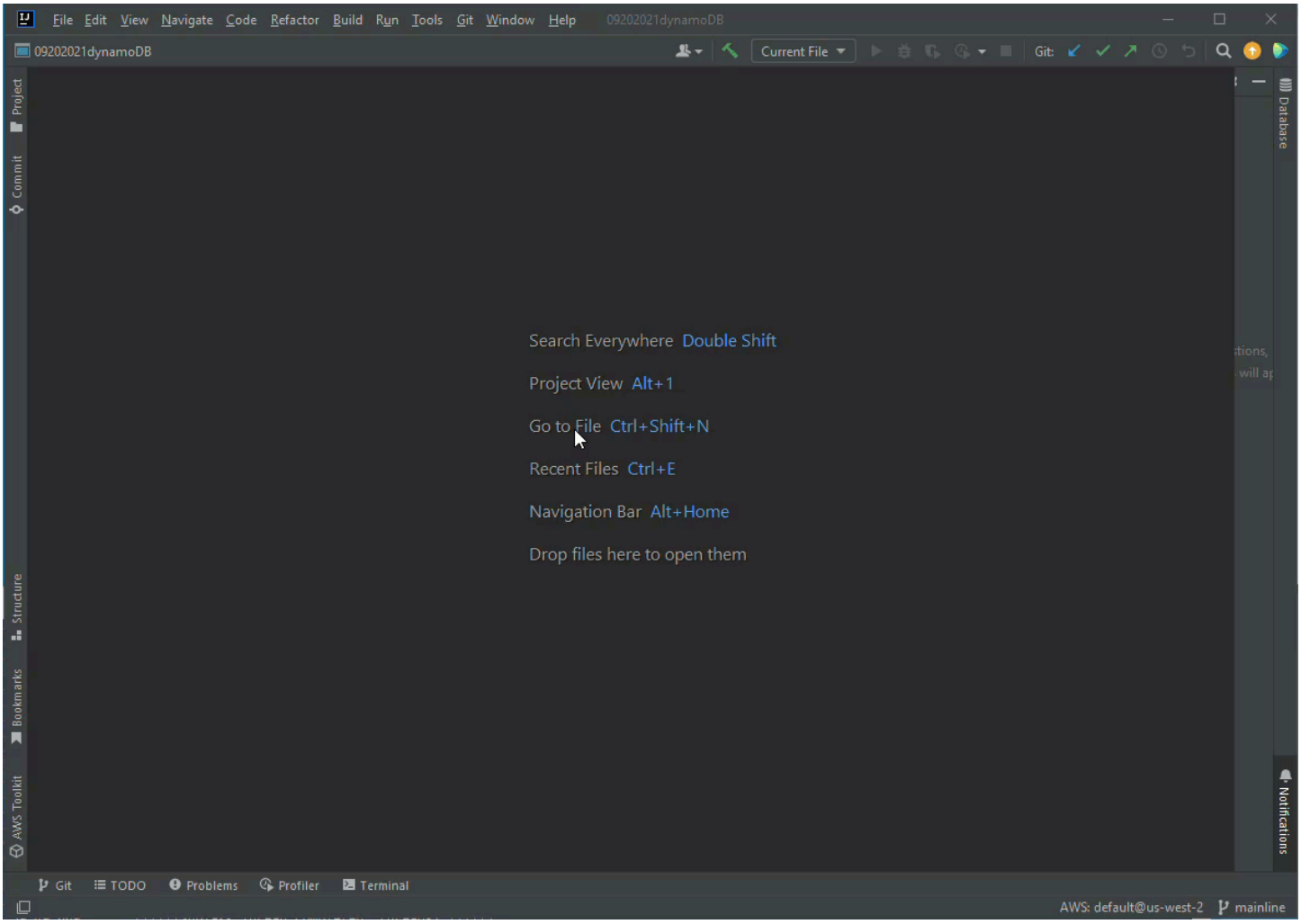
AWS Explorer

AWS サービスとリソースは、AWS Toolkit for JetBrains エクスプローラーから利用できます。

> **Note**
>
> AWS サービスとリソースは、認証を設定して AWS アカウントに接続した後にのみ、AWS エクスプローラーに表示されます。
> 認証と AWS Toolkit for JetBrains に関する補足情報については、本ユーザーガイドの「認証とアクセス」の目次を参照してください。
> AWS Toolkit for JetBrains から AWS アカウントへの接続に関する補足情報については、本ユーザーガイドの「AWS に接続する」トピックを参照してください。

AWS サービスとリソースを AWS Toolkit for JetBrains エクスプローラーに表示する方法:

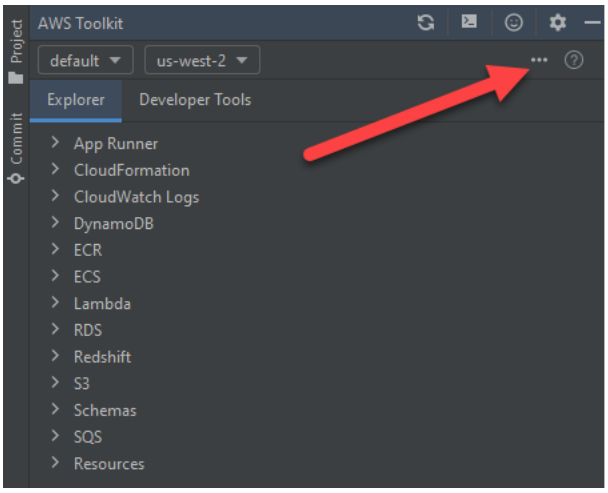
1. AWS Toolkit for JetBrains で [エクスプローラー] タブを選択し、アカウントと地域に関連付けられている AWS サービスを表示します。
2. サービスを選択すると、リソースのリストが拡張されます。
3. リソースのコンテキストメニューを開く (右クリック) と、リソースを変更するための機能の一覧が表示されます。



AWS に接続する

接続設定は AWS Toolkit for JetBrains 接続ペインの [省略記号アイコン] メニューにあります。

> **Note**
>
> AWS Toolkit for JetBrains から AWS アカウントへの接続に関する補足情報については、本ユーザーガイドの「AWS に接続する」トピックを参照してください。



AWS Toolkit for JetBrains を AWS アカウントに接続する

ほとんどの Amazon ウェブサービス (AWS) とリソースは、AWS アカウントをで管理します。AWS Toolkit for JetBrains を使用するのに AWS アカウントは必要ありませんが、接続がないとツールキットの機能が限定的になります。

以前にすでに AWS アカウントと他の AWSサービス (AWS Command Line Interface など) を通した認証をセットアップしている場合には、接続プロセス中に AWS Toolkit for JetBrains が認証情報を自動的に検出します。

AWS を始めて使用しており、まだアカウントを作成していない場合、AWS Toolkit for JetBrains を AWS アカウントに接続する主な手順は 3 つあります。

1. AWS アカウントを登録する: AWS アカウントの登録は AWS サインアップポータル から行います。新規 AWS アカウントをセットアップする際の詳細については、「AWS セットアップユーザーガイド」の「概要」トピックを参照してください。
2. 認証を設定する: AWS Toolkit for JetBrains から AWS アカウントで認証する方法には、主に 3 つの方法があります。これらの方法の詳細については、本ユーザーガイドの「認証とアクセス」トピックを参照してください。

3. AWS Toolkit for JetBrains から AWS アカウントで認証する: AWS アカウントを作成して認証を設定した後、次のセクションにある「AWS Toolkit for JetBrains から AWS に接続する」手順を完了することで、AWS Toolkit for JetBrains を AWS アカウントに接続できます。

AWS Toolkit for JetBrains から AWS に接続する

以下の手順を実行して、AWS Toolkit for JetBrains から、既存の IAM ID センターの認証情報を使用して、AWS アカウントで認証します。

> **Note**
>
> このプロセスにより、既定のウェブブラウザで AWS IAM ID センターポータルが開きます。認証情報の有効期限が切れるたびにこのプロセスを繰り返して、AWS アカウントと AWS Toolkit for JetBrains の間の接続を更新する必要があります。

1. AWS Toolkit for JetBrains から ... (省略記号) アイコンを選択して、[AWS 接続設定] メニューを展開します。
2. [AWS 接続設定] メニューから、[新しい接続を追加...] を選択して [AWS ツールキット: 接続を追加] ダイアログを開きます。
3. [AWS ツールキット: 接続を追加] ダイアログで、認証したい接続オプションを選択し、[接続] を選択して続行します。

   > **Note**
   >
   > AWS アカウントの認証方法の設定に関する AWS Toolkit for JetBrains 特有の情報については、ユーザーガイドの「認証とアクセス」トピックを参照してください。

4. 認証方法を選択したら、画面の指示に従って AWS Toolkit for JetBrains を AWS アカウントに接続すると、セットアッププロセスが完了した時に JetBrains から通知が送信されます。

AWS Toolkit for JetBrains の AWS リージョンの設定

AWS リージョンは、AWS リソースが管理される場所を指定します。デフォルトの AWS リージョンは、AWS Toolkit for JetBrains の AWS アカウントに接続するときに検知され、AWS エクスプローラーに自動的に表示されます。

次のセクションでは、AWS Toolkit for JetBrains エクスプローラーからリージョンを表示して変更する方法について説明します。

現在の AWS リージョンの表示

どの AWS リージョンが現在選択されているかを確認するには、次の手順を実行してください。

1. AWS エクスプローラーから [設定] アイコンを選択し、[オプションメニューの表示] を開きます。
2. [オプションメニューの表示] から [AWS 接続の設定] を展開し、アカウントで利用可能な AWS リージョンのリストを表示します。
3. 現在の AWS リージョンには、リージョン名の横に [チェックマーク]アイコンが表示されています。

AWS リージョンを変更する

現在の AWS リージョンを変更するには、次の手順を実行してください。

1. AWS エクスプローラーから [設定] アイコンを選択し、[オプションメニューの表示] を開きます。
2. [オプションメニューの表示] から [AWS 接続の設定] を展開し、AWS リージョンのリストを表示します。
3. 接続する AWS リージョンをリストから選択します。

> **Note**
>
> 接続するリージョンが表示されない場合、[すべてのリージョン] を選択してすべての AWS リージョンの全体リストを開きます。

AWS Toolkit for JetBrains の HTTP プロキシ設定

AWS Toolkit for JetBrains の HTTP プロキシ設定は、ご使用の JetBrains 統合開発環境 (IDE) を介して処理されます。HTTP プロキシの設定方法の詳細については、次のリストから JetBrains IDE を選択してください。

- CLion - CLion のヘルプウェブサイトの「HTTP プロキシの設定」を参照してください。

- GoLand - GoLand のヘルプウェブサイトの「HTTP プロキシ」を参照してください。
- IntelliJ IDEA - IntelliJ IDEA のヘルプウェブサイトの「HTTP プロキシ」を参照してください。
- WebStorm - WebStorm のヘルプウェブサイトの「HTTP プロキシ」を参照してください。
- JetBrains Rider - JetBrains Rider のヘルプウェブサイトの「HTTP プロキシの設定」を参照してください。
- PhpStorm - PhpStorm のヘルプウェブサイトの「HTTP プロキシ」を参照してください。
- PyCharm - PyCharm のヘルプウェブサイトの「HTTP プロキシ」を参照してください。
- RubyMine - RubyMine のヘルプウェブサイトの「HTTP プロキシ」を参照してください。

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

AWS Toolkit for JetBrains で作業を開始するにあたり、AWS 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 アカウントを通して管理されます。すべての AWS Toolkit for JetBrains サービスと機能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になります。

1. AWS Identity and Access Management (IAM) または AWS アカウントの AWS IAM Identity Center 認証です。ほとんどの AWS サービスとリソースは IAM と IAM ID センターを通じて管理されます。
2. AWS ビルダー ID は一部の他の AWS サービスでは任意となります。

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 IAM Identity Center

AWS IAM Identity Center は AWS アカウント認証の管理に推奨されるベストプラクティスです。

Software Development Kit (SDK) 向けに IAM Identity Center を設定する方法と AWS Toolkit for JetBrains の詳細な説明については、「AWSSDK およびツールリファレンスガイド」の「IAM Identity Center 認証」のセクションを参照してください。

AWS Toolkit for JetBrains から IAM Identity Center でサインインする

以下の手順を実行して、AWS Toolkit for JetBrains から、既存の IAM ID センターの認証情報を使用して、AWS アカウントで認証します。

> **Note**
>
> このプロセスにより、既定のウェブブラウザで AWS IAM ID センターポータルが開きます。認証情報の有効期限が切れるたびにこのプロセスを繰り返して、AWS アカウントと AWS Toolkit for JetBrains の間の接続を更新する必要があります。

1. AWS Toolkit for JetBrains から [...] (省略記号) アイコンを選択して、[AWS 接続設定] を開きます。
2. [AWS 接続設定] メニューから、[新しい接続を追加...] を選択して [AWS ツールキット: 接続を追加] ダイアログを開きます。
3. [AWS Toolkit: 接続を追加] ダイアログで、[AWS IAM Identity Center を使用して接続] ラジアルボタンを選択し、IAM Identity Center のポータル URL を [開始 URL]: テキストフィールドに入力し、その後 [接続] を選択して続行します。
4. プロンプトが表示されたら、優先するウェブブラウザで IAM Identity Center ポータルを開くことを確認し、プロンプトに従って認証プロセスを完了します。認証プロセスが完了すると通知が表示され、ブラウザウィンドウを安全に閉じることができます。

AWS IAM 認証情報

AWS IAM ユーザー認証情報は、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。

次のセクションは、AWS Toolkit for JetBrains を設定して IAM ユーザー認証情報を介して AWS アカウントで認証する方法について説明します。

> **Important**
>
> IAM 認証情報を設定して AWS アカウントで認証する前に、次の点に注意してください。
>
> - すでに別の AWS サービス (AWS CLI など) で IAM 認証情報を設定している場合、AWS Toolkit for JetBrains はこれらの認証情報を自動的に検出して利用可能にします。
> - AWSは IAM Identity Center 認証を推奨します。AWSIAM のベストプラクティスに関する詳細については、AWS の「アイデンティティとアクセス管理」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、専用ソフトウェアの開発や実際のデータを扱うときは、IAM ユーザーを認証に使用しないでください。代わりに、AWS IAM Identity Center

ユーザーガイドの「IAM アイデンティティセンターとは?」に記載している、アイデンティティプロバイダーによるフェデレーションを使用してください。

前提条件

AWS Toolkit for JetBrains を設定して IAM ユーザー認証情報で認証できるようにする前に、次の前提条件を満たす必要があります。別のサービス (AWS Command Line Interface など) で IAM ユーザー認証情報をすでにセットアップしている場合、前提条件の手順をスキップして次のセクションに進んでください。

1. IAM ユーザーの作成。IAM ユーザーを作成する詳細な手順については、「AWS SDK とツールリファレンスガイド」の「ステップ 1: IAM ユーザーの作成」を参照してください。
2. IAM ユーザーアクセスキーの取得。IAM ユーザーアクセスキーを取得する詳細な手順については、「AWS SDK とツールリファレンスガイド」のステップ 2: アクセスキーの取得を参照してください。
3. オプション: 共有認証情報ファイルの更新。共有認証情報ファイルの更新に関する詳細な手順については、「AWS SDK とツールリファレンスガイド」の「ステップ 3: 共有認証情報ファイルの更新」を参照してください。

> **Note**
>
> オプションの前提条件のステップ 3: 共有認証情報ファイルの更新が完了している場合、次のセクションで説明する、AWS Toolkit for JetBrains から共有認証情報ファイルの作成手順が実行される間、AWS Toolkit for JetBrains は認証情報を自動的に検出します。

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

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

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

1. AWS Toolkit for JetBrains から [+ AWSに接続の追加] を選択し、[AWS ツールキット: 接続の追加] ダイアログボックスを開きます。

2. [AWS ツールキット: 接続の追加] ダイアログボックスから [AWS認証情報ファイルの編集] を選択し、[認証情報ファイルの作成] の確認画面を開きます。
3. [認証情報ファイルの作成] の確認で [作成] を選択し、確認内容を閉じて `credential File` を作成します。
4. 作成が完了すると、`credential File` は IDE で自動的に開かれます。

> **Note**
>
> `Credential File` 作成プロセス中に発生する事項:
>
> - エラーが発生した場合、JetBrains は通知を表示してエラー詳細を含む作成ログを開きます。
> - すべての名前付きプロファイルを 1 つのファイルに保存できます。認証情報と設定ファイルの両方を使用している場合、認証情報はデフォルトで IDE で開かれます。
> - 同じ名前を共有するプロファイルの両方のファイルに認証情報がある場合、認証情報ファイルのキーが優先されます。

共有認証情報の設定

AWS アカウントで AWS Toolkit for JetBrains を認証する最後の手順は、認証情報を設定することです。

1. AWS Toolkit for JetBrains から [+ AWSに接続の追加] を選択し、[AWS ツールキット: 接続の追加] ダイアログボックスを開きます。
2. [AWS ツールキット: 接続の追加] ダイアログボックスから [AWS 認証情報ファイルの編集] を選択し、[認証情報ファイル] を開きます。
3. `credentials file` が JetBrains で開かれたら、`[default]` というラベルの付いたセクションを探してください。
4. `[default]` セクションからエントリ `#aws_access_key_id =` を探し、# を削除してAWS アクセスキーを入力します。エントリーは次のように表示されます:

   `aws_access_key_id = AKIAIOSFODNN7EXAMPLE`
5. `[default]` セクションからエントリ `#aws_secret_access_key =` を探し、# を削除して AWS シークレットアクセスキーを入力します。エントリーは次のように表示されます:

   `aws_secret_access_key = wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY`

更新した認証情報ファイルの最終バージョンは、次のような内容になります。

```
[default]
# The access key and secret key pair identify your account and grant access to AWS.
# Treat your secret key like a password. Never share your secret key with anyone.
 Do
# not post it in online forums, or store it in a source control system. If your
 secret
# key is ever disclosed, immediately use IAM to delete the access key and secret
 key
# and create a new key pair. Then, update this file with the replacement key
 details.
aws_access_key_id = AKIAIOSFODNN7EXAMPLE
aws_secret_access_key = wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
```

6. 変更をファイルに保存します。AWS Toolkit for JetBrains は更新された認証情報を自動的に検出し、AWS アカウントに接続します。

   `aws_secret_access_key = wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY`

開発者用 AWS Builder ID

AWS Builder ID は、オプションまたは特定の AWS サービスに必須となる AWS アカウントです。特定のサービスに対して AWS Builder ID を設定する方法の詳細については、本ガイドの「the section called “AWS Builder ID サービス”」セクションを参照してください。

次のセクションでは、AWS Toolkit for JetBrains から AWS Builder ID を作成する方法と認証する方法について説明します。

AWS Builder ID をセットアップする

AWS Toolkit for JetBrains からの AWS Builder ID をセットアップする

1. AWS [エクスプローラー] から、[+ AWS に接続を追加] を選択して [AWS ツールキット: 接続の追加] ダイアログを開きます。
2. [AWS ツールキット: 接続の追加] ダイアログから、[個人メールを使用して登録し、AWS Builder ID でサインイン] を選択して[AWS Builder ID でサインイン] ダイアログを開きます。

3. [AWS Builder ID でサインイン] ダイアログから、[コードを開いてコピー] ボタンを選択して、優先ウェブブラウザに[AWS Authorize リクエスト] サイトを開きます。
4. ウェブブラウザから、提供されている欄に確認コードを貼り付けて、[次へ] を選択して [AWS Builder ID の作成] サイトへと続行します。
5. ウェブブラウザで各手順を完了して続行すると、ブラウザを閉じても安全であることが通知され、プロセスが完了すると JetBrains へと戻ります。
6. JetBrains では、[+ AWS への接続を追加] ドロップダウンが更新され、AWS Builder ID に接続中であることが示されます。

AWS Builder ID サービス

特定のサービスを AWS Builder ID に接続する際に、追加の設定が必要な場合があります。以下は、AWS Toolkit for JetBrains を通してアクセスできる AWS Builder ID との互換性があるサービスです:

- Amazon CodeCatalyst: AWS Builder ID 用に Amazon CodeCatalyst をセットアップする方法に関するその他の情報については、「Amazon CodeCatalyst」ユーザーガイドの「Amazon CodeCatalyst のセットアップ」セクションを参照してください。
- Amazon CodeWhisperer: AWS Builder ID 用に Amazon CodeWhisperer をセットアップする方法に関するその他の情報については、「Amazon CodeWhisperer」ユーザーガイドの「個々の開発者に対する Amazon CodeWhisperer のセットアップ」セクションを参照してください。

[AWS ツールキットエクスプローラー] から AWS サービスを操作する

AWS サービスとリソースは、AWS Toolkit for JetBrains エクスプローラーから利用できます。

AWS Toolkit for JetBrains と AWS エクスプローラーをナビゲートする方法の詳細については、本ユーザーガイドの「ナビゲーション」トピックを参照してください。

AWS Toolkit for JetBrains から特定の AWS サービスを使用する方法の詳細については、次のトピックのリストから選択してください。

トピック

実験的な機能の使用

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

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

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

1. JetBrains の AWS 設定を編集するには、[ファイル]、[設定] を選択します (または Ctrl+Alt+S を押します)。
2. [設定]ペインで [ツール] を展開し、[AWS]、[実験的機能] を選択します。
3. リリース前にアクセスする実験的機能のチェックボックスをオンにします。実験的な機能をオフにするには、該当するチェックボックスをオフにします。
4. 実験的機能を有効にした後は、[AWS エクスプローラー] を開いて [オプション] (歯車アイコン) 、[実験的機能] を開くことで、確認できます。機能名の横にあるチェックマークは、その機能が使用可能であることを示しています。

AWS App Runner で JetBrains 用 AWS ツールキットを操作する

AWS App Runner を使用すると、AWS Cloud で、ソースコードまたはコンテナイメージから、スケーラブルでセキュアなウェブアプリケーションに迅速でシンプルな、費用対効果の高い方法で直接デプロイできます。新しいテクノロジーを学習したり、使用するコンピューティングサービスを決定したり、AWS リソースのプロビジョニングと構成方法を知ったりする必要はありません。

AWS App Runner を使用すると、ソースイメージまたはソースコードに基づいて、サービスを作成して管理できます。ソースイメージを使用する場合は、イメージリポジトリに保存されているパブリックまたはプライベートコンテナイメージを選択できます。App Runner は以下のイメージリポジトリプロバイダーをサポートしています。

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

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

前提条件

本セクションでは、ユーザーはすでに AWS アカウントと、AWS App Runner を備えた最新バージョンの AWS Toolkit for JetBrains を取得していることを前提とします。これらのコア要件に加えて、関連するすべての IAM ユーザーが App Runner サービスを操作するアクセス許可を持っている必要があります。また、コンテナイメージの URI や GitHub リポジトリへの接続など、サービスソースに関する特定の情報を取得する必要があります。この情報は、App Runner サービスを作成するときに必要です。

App Runner の IAM アクセス許可の設定

App Runner に必要なアクセス許可を最も簡単に付与する方法には、既存の AWS マネージドポリシーを関連する IAM エンティティ (具体的にはユーザーまたはグループ) にアタッチします。App Runner には、IAM ユーザーにアタッチできる 2 つのマネージドポリシーが用意されています。

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

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

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

[App Runner サービスの作成] ダイアログボックスを使用してこの IAM ロールを作成できます。

> **Note**
>
> AWSServiceRoleForAppRunner サービスリンクロールを使用すると、AWS App Runner は以下のタスクを実行できます。
>
> - Amazon CloudWatch Logs Logs ロググループにログをプッシュします。

- Amazon CloudWatch Events ルールを作成して、Amazon Elastic Container Registry (Amazon ECR) イメージプッシュを購読します。

サービスにリンクされたロールを手動で作成する必要はありません。AWS App Runnerで AWS Management Console を作成するとき、または AWS Toolkit for JetBrains から呼び出す API オペレーションを使用すると、サービスにリンクされたロールが AWS App Runner によって作成されます。

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

App Runner のサービスソースの取得

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

Source image

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

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

ソースの詳細を [App Runner サービスの作成] ダイアログボックスに入力する際、イメージリポジトリの URI を指定します。

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

Source code

AWS App Runner サービスにソースコードをデプロイする場合は、サポートされているリポジトリプロバイダーが管理する Git リポジトリにソースコードが保存されている必要があ

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

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

GitHub リポジトリから App Runner サービスにソースコードをデプロイする場合、App Runner は GitHub への接続を確立します。リポジトリがプライベートである (つまり GitHub でパブリックにアクセスできない) 場合は、App Runner に接続の詳細情報を指定する必要があります。

> **Important**
>
> GitHub 接続を作成するには、App Runner コンソール (https://console.aws.amazon.com/apprunner) を使用して、GitHub から AWS へリンクする接続を作成する必要があります。ソースコードリポジトリの詳細を指定するには、[App Runner サービスの作成] ダイアログボックスを使用する際に [GitHub 接続] ページに記載されている接続方法を選択します。
> 詳細については、AWS App Runner デベロッパーガイドの「App Runner 接続の管理」を参照してください。

App Runner サービスインスタンスは、コードのビルドと実行を可能にするマネージドランタイムを提供します。AWS App Runner では現在、以下のランタイムをサポートしています。

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

AWS Toolkit for JetBrains の [App Runner サービスの作成] ダイアログボックスを使用することにより、App Runner サービスがどのようにサービスを構築し開始するかを指定します。インターフェイスに情報を直接入力するか、YAML 形式の App Runner 構成ファイルを指定することができます。このファイルの値は、App Runner にサービスを構築して開始し、ランタイムコンテキストを提供する方法を指示します。これには、関連するネットワーク設定と環境変数が含まれます。 設定ファイルの名前は `apprunner.yaml` です。設定ファイルは、アプリケーションのリポジトリのルートディレクトリに自動的に追加されます。

料金

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

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

トピック

- App Runner サービスの作成
- App Runner サービスの管理

App Runner サービスの作成

AWS Toolkit for JetBrains で [App Runner サービスの作成] ダイアログボックスを使用して App Runner サービスを作成できます。インターフェイスを使用してソースリポジトリを選択し、アプリケーションが実行されるサービスインスタンスを設定できます。

App Runner サービスを作成する前に、前提条件をすべて満たしていることを確認してください。これには、関連する IAM 許可の提供と、デプロイするソースリポジトリの特定情報のメモ追加が含まれます。

App Runner サービスを作成するには

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

   [App Runner サービスの作成] ダイアログボックスが表示されます。
3. 固有の[サービス名]を入力してください。
4. ソースタイプ([ECR]、[ECR パブリック]、[ソースコードリポジトリ])を選択して関連する設定を行います

   ECR/ECR public

   プライベートレジストリを使用している場合、次のように[デプロイタイプ]を選択します:

- [手動]: 各デプロイを明示的にサービスに対して開始する場合、手動デプロイを使用します。
- [自動]: サービスに継続的な統合とデプロイ (CI/CD) 動作を実装する場合、自動デプロイを使用します。このオプションを選択する場合、新しいイメージバージョンをイメージリポジトリにプッシュしたり、コードリポジトリに新しいコミットをプッシュしたりするたびに、App Runner はそれを自動的にサービスにデプロイします。ユーザーによる追加アクションは必要ありません。

[コンテナイメージ URI] には、Amazon ECR プライベートレジストリまたは Amazon ECR パブリックギャラリーからコピーしたイメージリポジトリの URI を入力します。

[スタートコマンド] には、コマンドを入力してサービスプロセスを開始します。

[ポート] には、サービスが使用する IP ポートを入力します。

Amazon ECR プライベートレジストリを使用している場合、必要な [ECR アクセスロール]を選択して [作成] を選択します。

- [IAM ロールの作成] ダイアログボックスには、IAM ロールの [名前]、[マネージドポリシー]、[信頼関係] が表示されます。[Create] (作成) を選択します。

Source code repository

[デプロイタイプ] を選択します:

- [手動]: 各デプロイを明示的にサービスに対して開始する場合、手動デプロイを使用します。
- [自動]: サービスに継続的な統合とデプロイ (CI/CD) 動作を実装する場合、自動デプロイを使用します。このオプションを選択する場合、新しいイメージバージョンをイメージリポジトリにプッシュしたり、コードリポジトリに新しいコミットをプッシュしたりするたびに、App Runner はそれを自動的にサービスにデプロイします。ユーザーによる追加アクションは必要ありません。

[接続] には、[GitHub 接続]ページのリストから利用可能な接続方法を選択します。

[リポジトリ URL] には、GitHub でホストされているリモート リポジトリへのリンクを入力します。

[ブランチ] には、デプロイするソース コードの Git ブランチを指定します。

[設定] には、ランタイム設定の定義方法を指定します:

- [こちらですべての設定を実行]: アプリケーションのランタイム環境に次の設定を指定する場合、このオプションを選択します。
  - [Runtime] (ランタイム): [Python 3] または [Nodejs 12] を選択します。
  - [ポート]: サービスが使用する IP ポートを入力します。
  - [Build command] (ビルドコマンド): サービスインスタンスのランタイム環境でアプリケーションをビルドするコマンドを入力します。
  - [Start command] (開始コマンド): サービスインスタンスのランタイム環境でアプリケーションを開始するコマンドを入力します。
- [こちらに構成ファイル設定を入力]: このオプションを選択して `apprunner.yaml` 構成ファイルで定義された設定を使用します。このファイルは、アプリケーションのリポジトリのルートディレクトリにあります。

5. 値を指定して App Runner サービス インスタンスのランタイム構成を定義します
   - [CPU]: App Runner サービス (デフォルト: `1 vCPU`) の各インスタンスに予約された CPU ユニット数。
   - [メモリ]: App Runner サービスの各インスタンスに予約されたメモリ量 (デフォルト: `2 GB`)
   - [環境変数]: サービス インスタンスの動作のカスタマイズに使用する環境変数のオプション。キーと値を定義して環境変数を作成します。
6. [作成] を選択します

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

App Runner サービスの管理

App Runner サービスを作成したら、AWS Explorer ペインを使用して、次のアクティビティを実行します。

App Runner サービスの一時停止と再開

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

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

App Runner サービスを一時停止するには

1. AWS Explorer を開きます (まだ開いていない場合)。
2. [App Runner] を展開して、サービスのリストを表示します。
3. サービスを右クリックし、[Pause] (一時停止) を選択します。
4. 表示されるダイアログボックスで、[一時停止] を選択します。

   サービスが一時停止している間、サービス ステータスが[実行中]から[操作実行中]に変更され、その後に[一時停止中]に変更されます。

App Runner サービスを再開するには

1. AWS Explorer を開きます (まだ開いていない場合)。
2. [App Runner] を展開して、サービスのリストを表示します。
3. サービスを右クリックし、[Resume] (再開) を選択します。
4. 表示されるダイアログボックスで、[再開] を選択します。

サービスが再開している間、サービス ステータスが [一時停止中] から [操作実行中] に変更され、その後に [実行中] に変更されます。

App Runner サービスの展開

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

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

App Runner のログストリームの表示

CloudWatch ログを使用して App Runner などのサービスのログファイルをモニタリング、保存、アクセスします。CloudWatch ログは、ログイベントとログストリームという 2 つの異なるタイプのログファイルを記録します。ログイベントは、アプリケーションまたはリソースが記録したアクティビティの記録であり、CloudWatch ログでモニタリングできます。ログストリームは、同じソースを共有する一連のログイベントです。

App Runner 用の、次の 2 タイプのログストリームにアクセスできます:

- サービスログストリーム: App Runner によって生成されるログ出力が含まれます。このタイプのログストリームでは、ログイベントは App Runner がサービスを管理して処理する方法について記録したものです。
- アプリケーションのログストリーム: 実行中のアプリケーションコードの出力が含まれます。

1. [App Runner] を展開してサービスのリストを表示する
2. サービスを右クリックして次のオプションから 1 つ選択します:

- [サービスログストリームを表示]
- [アプリケーションログストリームを表示]

[ログストリーム] ペインには、ログストリームを構成するログイベントが表示されます。

3. 特定のイベントに関する詳細情報を表示するには、そのイベントを右クリックして [ログストリームをエクスポート]、[エディターで開く] または [ログストリームをエクスポート]、[ファイルに保存] を選択します。

App Runner サービスの削除

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

1. AWS Explorer を開きます (まだ開いていない場合)。
2. [App Runner] を展開して、サービスのリストを表示します。
3. サービスを右クリックし、[Delete Service] (サービスの削除) を選択します。
4. 確認ダイアログボックスで delete me (削除) と入力し、[OK] を選択します。

   削除されるサービスに [進行中] とステータスが表示された後、サービスがリストから削除されます。

JetBrains 用 Amazon CodeCatalyst

Amazon CodeCatalyst とは?

Amazon CodeCatalyst は、ソフトウェア開発チーム向けのクラウドベースのコラボレーションスペースです。AWS Toolkit for JetBrains Gateway を使用することで、JetBrains Gateway から直接 CodeCatalyst リソースを表示し管理することができます。ツールキットを使用して、開発環境の仮想コンピューティング環境の起動、管理、編集を行うこともできます。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。

次のトピックでは、AWS Toolkit for JetBrains Gateway を CodeCatalyst に接続する方法と、JetBrains Gateway を通じて CodeCatalyst を操作する方法について説明します。

トピック

CodeCatalyst と AWS Toolkit for JetBrains の開始方法

JetBrains Gateway から CodeCatalyst の使用を始めるには、以下を完了します。

トピック

JetBrains Gateway をインストールします。

AWS Toolkit を CodeCatalyst アカウントと統合する前に、JetBrains Gateway 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。JetBrains Gateway の最新バージョンをダウンロードするには、以下のリンクの中から必要な JetBrains Gateway ディストリビューションを選択します。

- Linux 用 JetBrains Gateway
- Windows 用 JetBrains Gateway
- macOS 用 JetBrains Gateway
- macOS Apple Silicon 用 JetBrains Gateway

JetBrains Gateway 用に AWS ツールキットをインストールする

JetBrains を CodeCatalyst アカウントに接続するには、ツールキット機能拡張の最新バージョンをインストールする必要があります。JetBrains [プラグインマーケットプレイス]から直接、最新バージョンを確認して、ツールキットのインストールを完了することができます。

JetBrains [プラグインマーケットプレイス]から AWS Toolkit プラグインをインストールするには、次のステップを完了します。

1. JetBrains Gateway のメイン画面から、アプリケーションの左下にある [設定/環境設定] アイコンを選択します。
2. [設定/環境設定] を選択して [設定/環境設定] ビューを開きます。
3. [設定/環境設定] ビューで [プラグイン] を選択して [プラグイン] ビューを開きます。

> **Note**
>
> [プラグイン] ビューは、[マーケットプレイス] ビューまたは[インストール済み] ビューのいずれかから開くことができます。
>
> - AWS Toolkit for JetBrains Gateway を初めてインストールする場合は、[プラグインマーケットプレイス] ビューを選択して続行します。
> - AWS Toolkit for JetBrains Gateway の以前のバージョンをお持ちの場合は、[インストール済み] ビューから更新します。

4. [マーケットプレイス] ビューで、「`AWS Toolkit`」とテキストを入力し、[AWS ツールキット] プラグインエントリと表示されたら、それを選択します。
5. [インストール] を選択してダウンロードし、AWS Toolkit for JetBrains Gateway をインストールします。

> **Note**
>
> JetBrains Gateway に、ダウンロードとインストールの進捗状況が表示されます。ツールキットが正常にインストールされると、JetBrains Gateway [Connections] エクスプローラーに [Amazon CodeCatalyst] プラグインのアイコンが追加されます。

CodeCatalyst アカウントと AWS Builder ID の作成

AWS Toolkit for JetBrains の最新バージョンのインストールに加えて、JetBrains ゲートウェイに接続するための AWS Builder ID と CodeCatalyst アカウントがアクティブである必要があります。アクティブな AWS Builder ID または CodeCatalyst アカウントがない場合は、「CodeCatalyst」ユーザーガイドの「CodeCatalyst の設定」セクションを参照してください。

> **Note**
>
> AWS Builder ID は、AWS 認証情報とは異なります。AWS認証情報は、AWS Toolkit からアクセスできるほとんどの AWS サービス で必要になります。新しい CodeCatalyst アカウン

トを作成し、既存の CodeCatalyst アカウントから作業するには、AWS Builder ID が必要です。これには、AWS Toolkit から入手できるすべての CodeCatalyst 機能の使用が含まれます。

JetBrains Gateway と CodeCatalyst の接続

JetBrains Gateway を CodeCatalyst アカウントに接続するには、以下の手順を実行します。

1. JetBrains Gateway の [接続] エクスプローラーから、[Amazon CodeCatalyst] プラグインを選択して [Amazon CodeCatalyst] プラグインビューを開きます。
2. [CodeCatalyst] プラグインビューから、[AWS Builder ID でサインイン] を選択すると [AWS ログインが必要です] とプロンプトが表示されます。
3. [AWS ログインが必要です] プロンプトで、[ブラウザを開く] を選択して、CodeCatalyst コンソールのサインイン画面を優先ウェブブラウザで開きます。
4. 表示されたフィールドに AWS Builder ID を入力し、指示に従って続行します。
5. プロンプトが表示されたら、[許可] を選択して JetBrains と CodeCatalyst アカウントの接続を確定します。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。
6. JetBrains Gateway では、[CodeCatalyst] プラグインビューが [開発環境] ビューに更新されます。

JetBrains Gateway で Amazon CodeCatalyst を使用する

JetBrains から、開発環境と呼ばれる仮想コンピューティング環境を起動できます。開発環境はカスタマイズ可能なクラウド開発環境であり、スペース内のさまざまなチームメンバー間でコピーや共有をすることができます。開発環境の詳細と CodeCatalyst からアクセスする方法については、「Amazon CodeCatalyst」ユーザーガイドの「開発環境」セクションを参照してください。

以下のセクションでは、JetBrains Gateway から開発環境を作成し、開き、使用する方法について説明します。

トピック

- 開発環境を開く
- 開発環境の作成
- サードパーティのリポジトリから開発環境を作成する

- 開発環境設定を構成する
- 開発環境の一時停止
- 開発環境の再開
- 開発環境の削除
- 開発環境のデフォルトを設定する

開発環境を開く

JetBrains Gateway から既存の開発環境を開くには、次のステップを実行します。

1. [接続] エクスプローラーから [Amazon CodeCatalyst] プラグインを選択します。
2. [リモート開発] ウィザードの本文で、開きたい開発環境の親スペースとプロジェクトに移動します。
3. 開く開発環境を選択します。
4. 開発環境を開くためのプロセスを確認して続行します。

> **Note**
>
> JetBrains の新しいステータスウィンドウに進行状況が表示されます。オープンプロセスが完了すると、新しいウィンドウに開発環境が開きます。

開発環境の作成

新しい開発環境を作成する方法

1. [接続] エクスプローラーから [CodeCatalyst] プラグインを選択します。
2. [リモート開発] ウィザードのヘッダーセクションから、[開発環境の作成] リンクをクリックして、[新規 CodeCatalyst 開発環境] ビューを開きます。
3. [新規 CodeCatalyst 開発環境] ビューで、以下のフィールドを使用して開発環境の設定を構成します。

   - [IDE]: 開発環境で起動する優先 JetBrains IDE を選択します。
   - [CodeCatalyst プロジェクト]: 開発環境用の CodeCatalyst スペースとプロジェクトを選択します。
   - [開発環境のエイリアス]: 開発環境の代替名を入力します。

- [コンピューティング]: 開発環境の仮想ハードウェア構成を選択します。
- [永続ストレージ]: 開発環境用の永続ストレージの容量を選択します。
- [非アクティブタイムアウト]: 開発環境がスタンバイ状態になるまでに経過する、システムのアイドル時間を選択します。

4. 新しい開発環境を作成するには、[開発環境の作成] を選択します。

> **Note**
>
> [開発環境の作成] を選択すると、[新規開発環境] ビューが閉じられて、開発環境を作成するプロセスが開始されます。このプロセスには数分かかることがあり、開発環境が作成されるまでは、他の JetBrains Gateway 機能を使用することはできません。JetBrains の新しいステータスウィンドウに進行状況が表示されます。プロセスが完了すると、新しいウィンドウに開発環境が開きます。

サードパーティのリポジトリから開発環境を作成する

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

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

開発環境設定を構成する

JetBrains Gateway から既存の開発環境の設定を変更するには、次のステップを実行します。

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

1. [接続] エクスプローラーから [Amazon CodeCatalyst] プラグインを選択します。
2. [リモート開発] ウィザードの本文から、設定する開発環境の親スペースとプロジェクトに移動します。

3. 設定する開発環境の横にある [設定] アイコンを選択し、[開発環境の構成] 設定を開きます。
4. [開発環境の設定]: 設定メニューで、以下のオプションを変更して開発環境を設定します。

   - [開発環境のエイリアス]: 開発環境の代替名を指定するオプションのフィールド。
   - [IDE]: 開発環境内で起動する JetBrains IDE を選択します。
   - [コンピューティング]: 開発環境の仮想ハードウェア構成を選択します。
   - [非アクティブタイムアウト]: 開発環境がスタンバイ状態になるまでに経過する、システムのアイドル時間を選択します。

開発環境の一時停止

開発環境内のアクティビティは永続的に保存されます。つまり、作業内容を失うことなく、開発環境を一時停止して再開することができます。

開発環境を一時停止するには、次のステップを実行します。

1. [接続] エクスプローラーから [Amazon CodeCatalyst] プラグインを選択します。
2. [リモート開発] ウィザードの本文で、一時停止したい開発環境の親スペースとプロジェクトに移動します。
3. アクティブな開発環境の横にある [一時停止] アイコンを選択して、[一時停止を確認] ダイアログを開きます。
4. [はい] を選択して [一時停止を確認] ダイアログを閉じ、一時停止プロセスを開始します。

> **Note**
>
> JetBrains の新しいステータスウィンドウに一時停止プロセスの進行状況が表示されます。開発環境が停止すると、[一時停止] アイコンがユーザーインターフェイスから消えます。

開発環境の再開

開発環境内のアクティビティは永続的に保存されます。つまり、一時停止した開発環境は、作業内容を失うことなく、再開することができます。

一時停止した開発環境を再開するには、次のステップを実行します。

1. [接続] エクスプローラーから [Amazon CodeCatalyst] プラグインを選択します。

2. [リモート開発] ウィザードの本文で、再開したい開発環境の親スペースとプロジェクトに移動します。
3. 再開する開発環境を選択します。

> **Note**
>
> JetBrains の新しいステータスウィンドウに再開プロセスの進行状況が表示されます。開発環境が再開されると、[一時停止] アイコンが開発環境の [設定] アイコンの横に追加されます。

開発環境の削除

開発環境を削除するには、次のステップを実行します。

1. [接続] エクスプローラーから [Amazon CodeCatalyst] プラグインを選択します。
2. [リモート開発] ウィザードの本文で、削除したい開発環境の親スペースとプロジェクトに移動します。
3. 開発環境の横にある [X] アイコンボタンを選択して、[削除を確認] ダイアログを開きます。
4. [はい] を選択してダイアログを閉じ、開発環境を削除します。

> **Important**
>
> [はい] を選択した後、開発環境は削除され、元に戻すことはできません。開発環境を削除する前に、必ずコードの変更をコミットして元のソースリポジトリにプッシュしてください。そうしなかった場合、保存されていない変更は永続的に失われます。
> 開発環境を削除すると、[リモート開発] ウィザードのアップデートと開発環境がリソースに含まれなくなります。

開発環境のデフォルトを設定する

開発環境のデフォルト設定は、開発環境の `devfile` で設定できます。`devfile` の仕様はオープンスタンダードであり、YAML ドキュメントで更新できます。

`devfile` を定義および設定する方法の詳細については、「devfile.io」を参照してください。

JetBrains Gateway 開発環境インスタンスから `devfile` を開いて編集するには、次のステップを実行します。

1. アクティブな JetBrains 開発環境の [ナビゲーションバー] で、[Amazon CodeCatalyst 開発環境] ノードを展開して [バックエンドステータスの詳細] メニューを開きます。
2. [開発環境の設定] タブ、次に [Devfile を開く] の順に選択して、JetBrains [エディター] で `devfile` を開きます。
3. [エディター] で、`devfile` に変更を加えて作業を保存します。
4. 変更を保存すると、[Amazon CodeCatalyst 開発環境] ノードに開発環境の再構築が必要であることを示すアラートが表示されます。
5. [Amazon CodeCatalyst 開発環境] ノードを展開し、[開発環境の設定] タブから [開発環境の再構築] ノードを選択します。

AWS Toolkit for JetBrains を使用して AWS CloudFormation を操作する

次のトピックでは、AWS Toolkit for JetBrains を使用して AWS アカウントの AWS CloudFormation スタックを操作する方法について説明します。

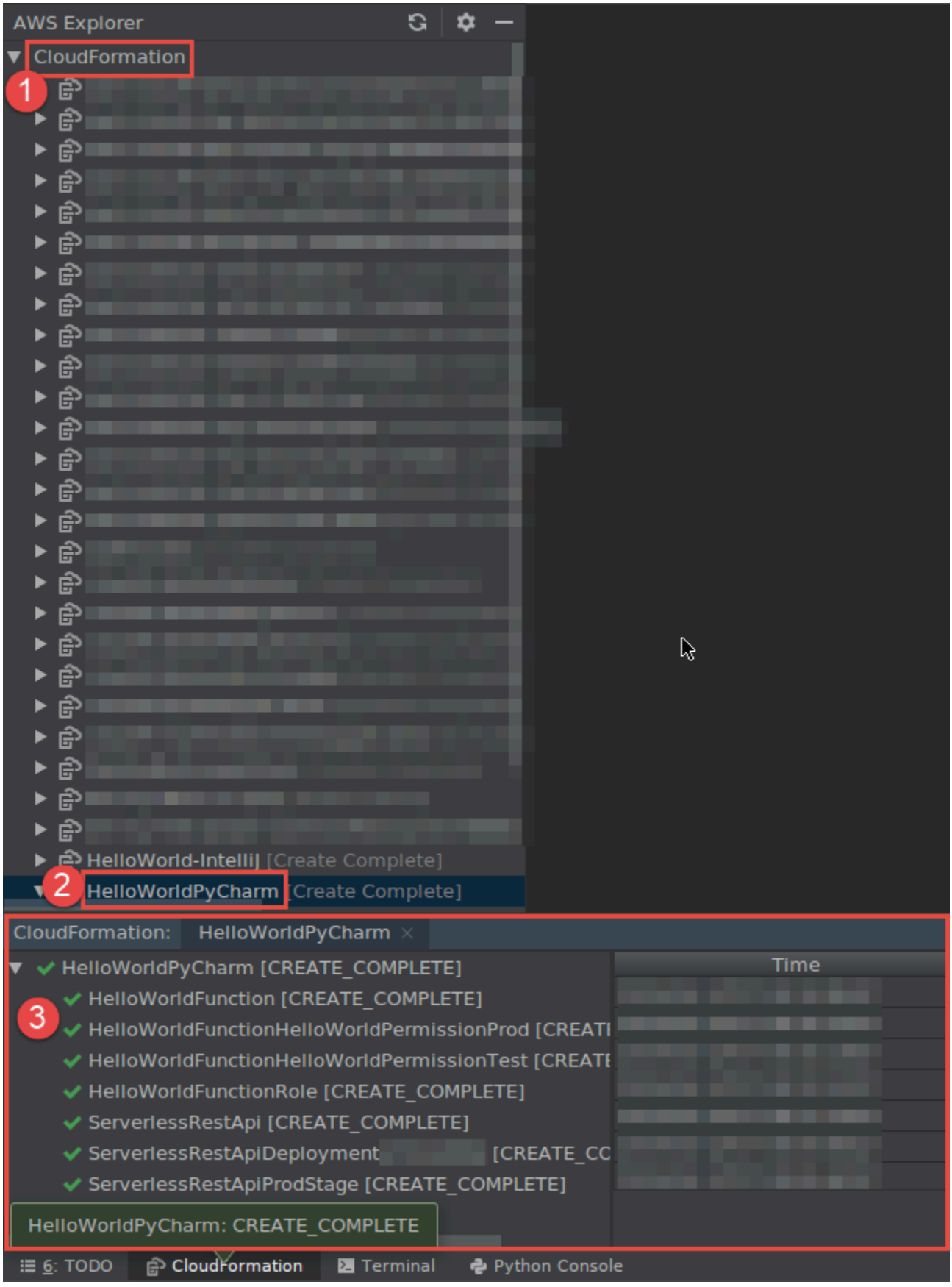
トピック

- AWS Toolkit for JetBrains を使用して AWS CloudFormation スタックのイベントログを表示する
- AWS Toolkit for JetBrains を使用して AWS CloudFormation スタックを削除する

AWS Toolkit for JetBrains を使用して AWS CloudFormation スタックのイベントログを表示する

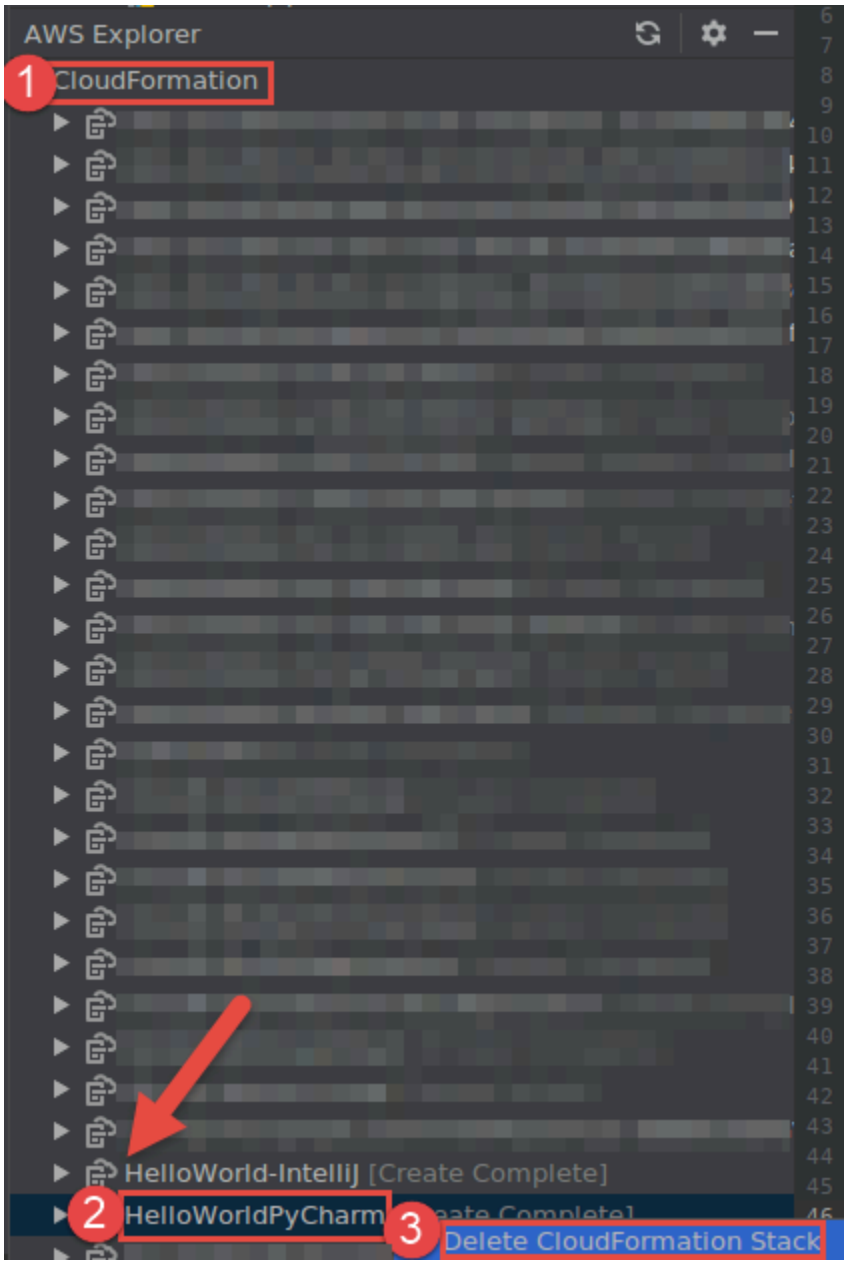
1. AWS Explorer を開きます (まだ開いていない場合)。スタックが現在のリージョンとは異なる AWS リージョンにある場合は、そのスタックが含まれる別の AWS リージョンに切り替えます。
2. [CloudFormation] を展開します。
3. スタックのイベントログを表示するには、スタックの名前を右クリックします。AWS Toolkit for JetBrains は、CloudFormation ツールウィンドウにイベントログを表示します。

[CloudFormation] ツールウィンドウの表示/非表示を切り替えるには、メインメニューで [表示]、[ツールウィンドウ]、[CloudFormation] の順に選択します。


AWS Explorer
CloudFormation
1
HelloWorld-IntelliJ [Create Complete]
2
HelloWorldPyCharm [Create Complete]
CloudFormation: HelloWorldPyCharm
Time
HelloWorldPyCharm [CREATE_COMPLETE]
3
HelloWorldFunction [CREATE_COMPLETE]
HelloWorldFunctionHelloWorldPermissionProd [CREATI
HelloWorldFunctionHelloWorldPermissionTest [CREATE
HelloWorldFunctionRole [CREATE_COMPLETE]
ServerlessRestApi [CREATE_COMPLETE]
ServerlessRestApiDeployment [CREATE_CC
ServerlessRestApiProdStage [CREATE_COMPLETE]
HelloWorldPyCharm: CREATE_COMPLETE
6: TODO
CloudFormation
Terminal
Python Console

AWS Toolkit for JetBrains を使用して AWS CloudFormation スタックを削除する

1. AWS Explorer を開きます (まだ開いていない場合)。スタックが含まれる別の AWS リージョンに切り替える必要がある場合は、ここで行います。
2. [CloudFormation] を展開します。
3. 削除するスタックの名前を右クリックし、[Delete CloudFormation Stack (CloudFormation スタックの削除)] を選択します。



4. スタックの名前を入力してそれが削除されたことを確認し、[OK] を選択します。スタックの削除が成功すると、AWS Toolkit for JetBrains は、AWS エクスプローラーの [CloudFormation] リストからスタック名を削除します。スタックの削除が失敗した場合は、スタックのイベントログを表示してトラブルシューティングできます。

AWS Toolkit for JetBrains ツールキットを使って CloudWatch Logs を使用

Amazon CloudWatch Logs により、使用中のすべてのシステム、アプリケーション、AWS のサービスからのログを、スケーラビリティに優れた 1 つのサービスで一元管理することができます。これにより、ログを簡単に表示したり、特定のエラーコードやパターンを検索したり、特定のフィールドに基づいてフィルタリングしたり、将来の分析のために安全にアーカイブしたりできます。詳細については、「Amazon CloudWatch ユーザーガイド」の「Amazon CloudWatch Logs とは」を参照してください。

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

トピック

- AWS Toolkit for JetBrains を使用して CloudWatch ロググループとログストリームの表示
- AWS Toolkit for JetBrains を使用して、ログストリーミングで CloudWatch ログイベントを操作
- AWS Toolkit for JetBrains を使用して CloudWatch Logs Insights を操作する

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

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

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

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

トピック

- CloudWatch Logs ノードのロググループとログストリームの表示
- [Lambda] ノードでログストリームを表示する
- [Amazon ECS] ノードでログストリームを表示する

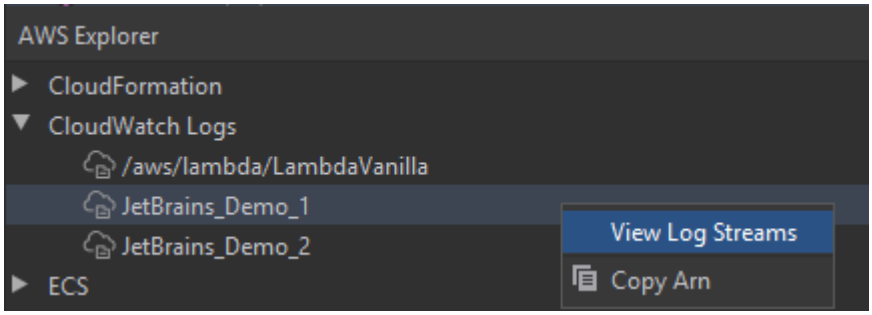
CloudWatch Logs ノードのロググループとログストリームの表示

1. AWS Explorer を開きます (まだ開いていない場合)。
2. CloudWatch Logs ノードをクリックして、ロググループのリストを展開します。

   現在の AWS リージョンのロググループは [CloudWatch Logs] ノードの下に表示されます。
3. ロググループのログストリームを表示するには、次のいずれかを実行します。

   - ロググループの名前をダブルクリックします。
   - ロググループの名前を右クリックし、[ログストリームの表示] を選択します。

   ロググループの内容は [ログストリーム] ウィンドウに表示されます。各ストリームのログイベントを操作する方法については、「CloudWatch ログイベントの操作」を参照してください。



[Lambda] ノードでログストリームを表示する

AWS Lambda 関数の CloudWatch Logs は、AWS エクスプローラーの [Lambda] ノードで確認できます。

> **注意**
>
> すべての AWS サービスのログストリームは、Lambda 関数 も含め、AWS エクスプローラーの [CloudWatch Logs] ノードを使用して表示することもできます。ただし、Lambda 関数に固有のログデータの概要については、[Lambda] ノードを使用することを推奨します。

1. AWS Explorer を開きます (まだ開いていない場合)。
2. [Lambda] ノードをクリックすると、Lambda 関数のリストが展開されます。

   現在の AWS リージョン の Lambda 関数は、[Lambda] ノードの下に表示されます。
3. Lambda 関数を右クリックした後、[ログストリームの表示] を選択します。

   関数のログストリームが [ログストリーム] ウィンドウに表示されます。各ストリームのログイベントを操作する方法については、「CloudWatch ログイベントの操作」を参照してください。

[Amazon ECS] ノードでログストリームを表示する

Amazon Elastic Container Service で実行および管理されているクラスターとコンテナの CloudWatch Logs は、AWS エクスプローラーの [Amazon ECS] ノードを使用して表示できます。

> **注意**
>
> すべての AWS サービスのロググループは、Amazon ECS も含め、AWS エクスプローラーの [CloudWatch Logs] ノードを使用して表示することもできます。ただし、Amazon ECS クラスターとコンテナに固有のログデータの概要については、[Amazon ECS] ノードを使用することを推奨します。

1. AWS Explorer を開きます (まだ開いていない場合)。
2. [Amazon ECS] ノードをクリックすると、Amazon ECS クラスターのリストが展開されます。

   現在の AWS リージョンの Amazon ECS クラスターは、[Amazon ECS] ノードの下に表示されます。
3. クラスターを右クリックした後、[ログストリームの表示] を選択します。

   クラスターのログストリームが [ログストリーム] ウィンドウに表示されます。
4. 特定のコンテナのログストリームを表示するには、クラスターをクリックして登録されているコンテナのリストを展開します。

   クラスターに登録されているコンテナが下に表示されます。
5. コンテナを右クリックして、[コンテナログストリームを表示] を選択します。

コンテナのログストリームが [ログストリーム] ウィンドウに表示されます。クラスターやコンテナのログイベントを操作する方法については、「CloudWatch ログイベントの操作」を参照してください。

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

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

トピック

- ストリーム内のログイベントの表示とフィルタリング
- ログアクションの操作
- CloudWatch ログイベントをファイルまたはエディターにエクスポート

ストリーム内のログイベントの表示とフィルタリング

ログストリームを開くと、[ログイベント] ペインに、そのストリームのログイベントシーケンスが表示されます。

1. 表示するログストリームを探すには、[ログストリーム] ウィンドウを開きます (CloudWatch ロググループとログストリームの表示 を参照)。

   > **注意**
   >
   > パターン一致を使用して、リスト内のストリームを検索できます。[ログストリーム] ウィンドウをクリックして、テキストを入力し始めます。入力したテキストと一致する最初のログストリーム名が強調表示されます。[最終イベント時間] 欄の上部をクリックして、リストを並べ替えることもできます。

2. ログストリームをダブルクリックすると、ログイベントのシーケンスが表示されます。

   [ログイベント] ウィンドウには、ログストリームを構成するログイベントが表示されます。

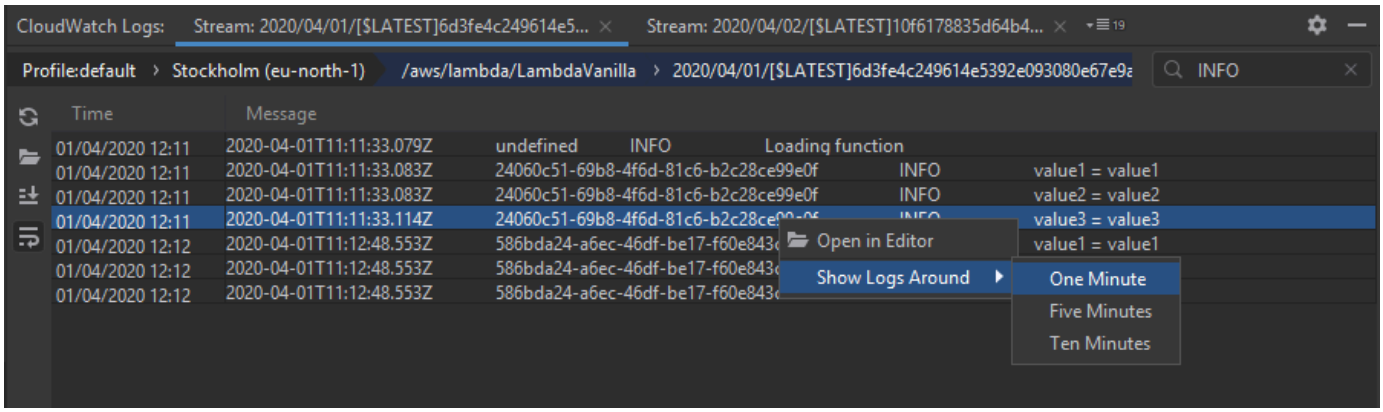
3. ログイベントを内容に基づいてフィルタリングするには、[ログストリームのフィルタリング] にテキストを入力し、[Return] を押します。

フィルターテキストと (大文字と小文字を区別して) 一致するテキストが含まれるログイベントが結果として出力されます。フィルターは、画面に表示されていないイベントも含めたログストリーム全体を検索します。

> **Note**
>
> パターン一致を使用して、ウィンドウ内のログイベントを検索することもできます。[ログイベント] ウィンドウをクリックして、テキストを入力し始めます。入力したテキストと一致する最初のログイベント名が強調表示されます。[ログストリームのフィルタリング] 検索とは異なり、画面上のイベントのみがチェックされます。

4. ログイベントを時間に基づいてフィルタリングするには、ログイベントを右クリックして [周囲にログを表示] を選択します。

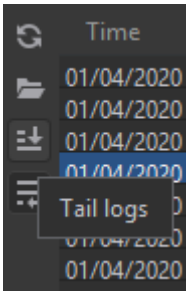
   [1 分]、[5 分]、[10 分] を選択できます。たとえば、[5 分] を選択すると、フィルターされたリストには、選択したエントリの 5 分前と 5 分後に発生したログイベントのみが表示されます。



[ログイベント] ウィンドウの左側にある [ログアクション] からは、ログイベントを操作するためのより多くの方法を選ぶことができます。

ログアクションの操作

[ログイベント] ウィンドウの左側からは、4 つのログアクションを使用して CloudWatch ログイベントの更新、編集、追跡、ラップを行うことができます。



1. 操作するログイベントを検索するには、[ログストリーム] ウィンドウを開きます。
2. 以下のいずれかのログアクションを選択します。

   - [リフレッシュ] — [ログイベント] ウィンドウが開かれた後に発生したログイベントでリストが更新されます。
   - [エディターで開く] — 画面上のログイベントを IDE のデフォルトエディターで開きます。

     > **Note**
     >
     > このアクションは、画面上のログイベントのみを IDE エディターにエクスポートします。ストリームのすべてのイベントをエディターに表示するには、[ログストリームをエクスポート] オプションを選択します。

   - [ログ末尾] — [ログイベント] ウィンドウに新しいログイベントをストリーミングします。これは、Amazon EC2 インスタンスや AWS CodeBuild ビルドなどの、長時間稼働しているサービスの継続的な更新に便利な機能です。
   - [ログのラッピング]— ウィンドウのサイズにより、長いエントリが非表示になる場合は、ログイベントのテキストを複数行に表示します。

CloudWatch ログイベントをファイルまたはエディターにエクスポート

CloudWatch ログストリームをエクスポートすると、そのログイベントを IDE のデフォルトエディターで開いたり、ローカルフォルダにダウンロードすることができます。

1. 表示するログストリームを検索するには、[ログストリーム] ウィンドウを開きます。
2. ログストリームを右クリックして、[ログストリームをエクスポート]、[エディターで開く] または [ログストリームをエクスポート]、[ファイルに保存] を選択します。

   - [エディターで開く] — IDE のデフォルトエディターに、選択したストリームを構成するログイベントのリストが表示されます。

> **Note**
>
> このオプションでは、ログストリームのすべてのイベントが IDE エディターにエクスポートされます。

- [ファイルに保存] — [ログストリームをダウンロード] ダイアログボックスが開きます。ここでダウンロードフォルダを選択し、ログイベントが含まれるファイルの名前を変更することができます。

AWS Toolkit for JetBrains を使用して CloudWatch Logs Insights を操作する

AWS Toolkit for JetBrains を使用して CloudWatch Logs Insights を操作することができます。CloudWatch Logs Insights を使用すると、Amazon CloudWatch Logs のログデータをインタラクティブに検索し分析することが可能になります。詳細については、「Amazon CloudWatch Logs ユーザーガイド」の「CloudWatch Logs Insights でログデータを分析する」を参照してください。

CloudWatch Logs Insights の IAM アクセス許可

CloudWatch Logs Insights のクエリ結果を実行して表示するには、以下のアクセス許可が必要です。

```
{
  "Version": "2012-10-17",
  "Statement" : [
    {
      "Effect" : "Allow",
      "Action" : [
        "logs:StartQuery",
        "logs:GetQueryResults",
        "logs:GetLogRecord",
        "logs:describeLogGroups",
        "logs:describeLogStreams"
      ],
      "Resource" : "*"
    }
  ]
}
```

次のアクセス許可は必須ではありませんが、関連する結果ウィンドウや IDE を閉じたときに、AWS Toolkit for JetBrains が現在実行中のクエリを自動的に停止できるようになります。

```
{
  "Version": "2012-10-17",
  "Statement" : [
    {
      "Effect" : "Allow",
      "Action" : [
        "logs:StopQuery"
      ],
      "Resource" : "*"
    }
  ]
}
```

CloudWatch Logs Insights での操作

CloudWatch Logs Insights クエリエディターの開き方

1. AWS エクスプローラーを開きます。
2. [CloudWatch Logs] ノードをクリックして、ロググループのリストを展開します。
3. 開くロググループを右クリックして、[クエリエディターを開く] を選択します。

CloudWatch Logs Insights クエリの開始方法

1. [クエリロググループ] ウィンドウで、クエリパラメータを必要に応じて変更します。

   時間の範囲は、日付または相対時間で選択できます。

   [クエリロググループ] フィールドでは、CloudWatch Logs Insights のクエリ構文を使うことができます。詳細については、「Amazon CloudWatch Logs ユーザーガイド」の「CloudWatch Logs Insights クエリ構文」を参照してください。
2. [実行] を選択してクエリを開始します。

CloudWatch Logs Insights クエリの保存方法

1. クエリ名を入力します。
2. [クエリの保存] を選択します。

   選択したロググループとクエリが AWS アカウントに保存されます。時間範囲は保存されません。

   保存したクエリは CloudWatch Logs Insights AWS Management Console ページから取得して、再利用することができます。

保存されている CloudWatch Logs Insights クエリの取得方法

1. [クエリロググループ] ウィンドウで、[保存したクエリを取得] を選択します。
2. 目的のクエリを選択して [OK] を選択します。

   選択したロググループとクエリで、既存のダイアログのすべてが置き換えられます。

クエリ結果間の移動方法

- CloudWatch Logs Insights の [クエリ結果] ウィンドウの右上で、[クエリエディターを開く] を選択します。

個々のログレコードを表示する方法

- クエリ結果ウィンドウで、行をダブルクリックすると、そのログレコードの詳細が含まれる新しいタブが開きます。

  右上にある [ログストリームを表示] を選択し、ログストリームに関連付けられたログレコードに移動することもできます。

Amazon CodeWhisperer を使用する

CodeWhisperer とは

Amazon CodeWhisperer は、リアルタイムで推奨コードを提供する、機械学習を利用した汎用コードジェネレーターです。コードを記述すると、CodeWhisperer は既存のコードとコメントに基づい

て、自動的に提案コードを生成します。パーソナライズされた推奨事項のサイズと範囲は、単一行のコードから、完全に形成された関数にいたるまで、多岐にわたります。

CodeWhisperer はコードをスキャンして、セキュリティ上の問題があれば強調して定義することもできます。

CodeWhisperer は以下の JetBrains 製品で利用できます。

- IntelliJ IDEA
- PyCharm
- WebStorm
- Rider

詳細については、「Amazon CodeWhisperer ユーザーガイド」を参照してください。

AWS Toolkit for JetBrains の Amazon DynamoDB

Amazon DynamoDB は、フルマネージド NoSQL データベースサービスであり、シームレスなスケーラビリティを備えた予測可能なパフォーマンスを提供します。DynamoDB サービスに関する詳細については、「Amazon DynamoDB ユーザーガイド」を参照してください。

次の項目では、AWS Toolkit for JetBrains から DynamoDB サービスを使用する方法について説明します。

トピック

- AWS Toolkit for JetBrains から Amazon DynamoDB を使用する
- AWS Toolkit for JetBrains から Amazon DynamoDBを使用する

AWS Toolkit for JetBrains から Amazon DynamoDB を使用する

AWS Toolkit for JetBrains では Amazon リソースネーム (ARN) を表示およびコピーしたり、Amazon DynamoDB リソースを JetBrains IDE から直接削除したりすることができます。

このセクションでは、AWS Toolkit for JetBrains からこれらのサービス機能を使用する方法について説明します。

DynamoDB リソースを表示する

現時点では、DynamoDB リソースをツールキットから直接作成することはできませんが、リソースは表示されます。DynamoDB リソースを表示するには、次の手順を実行します。

1. AWS Toolkit for JetBrains の [エクスプローラー] タブに移動します。
2. [DynamoDB] ノードを展開します。
3. DynamoDB リソースは DynamoDB ノードの下に表示されます。

DynamoDB リソース ARN をコピーする

Amazon リソースネーム (ARN) は、すべての AWS リソースに割り当てられる固有の ID であり、DynamoDB テーブルが含まれます。DynamoDB リソースの ARN ID をコピーするには、次の手順を実行します。

1. AWS Toolkit for JetBrains の [エクスプローラー] タブに移動します。
2. [DynamoDB] ノードを展開します。
3. ARN ID をコピーする DynamoDB リソースのコンテキストメニューを開きます (右クリック)。
4. [ARN をコピー] を選択し、リソース ARN ID を OS クリップボードにコピーします。

DynamoDB リソースを削除する

DynamoDB リソースを削除するには、次の手順を実行します。

1. AWS Toolkit for JetBrains の [エクスプローラー] タブに移動します。
2. [DynamoDB] ノードを展開します。
3. 削除する DynamoDB リソースのコンテキストメニューを開きます (右クリック)。
4. [テーブルを削除...] を選択して [テーブルを削除...] 確認ダイアログを開きます。
5. 確認手順を完了して DynamoDB テーブルを削除します。

AWS Toolkit for JetBrains から Amazon DynamoDBを使用する

Amazon DynamoDB のプライマリリソースはデータベーステーブルです。次のセクションでは、AWS Toolkit for JetBrains から DynamoDB テーブルを使用する方法について説明します。

DynamoDB テーブルを表示する

DynamoDB テーブルを表示するには、次の手順を実行します。

1. AWS Toolkit for JetBrains の [エクスプローラー] タブに移動します。
2. [DynamoDB] ノードを展開します。
3. DynamoDB リソースのリストにあるテーブルをダブルクリックすると、そのテーブル [エディター] ウィンドウに表示されます。

> **Note**
>
> テーブルデータを初めて表示するときには、結果の上限が 50 項目に設定された初回スキャンが取得されます。

結果の上限を設定する

取得されるテーブルエントリのデフォルト制限を変更するには、次の手順を実行します。

1. AWS エクスプローラーでテーブルをダブルクリックして、そのテーブルを JetBrains [エディター] ウィンドウに表示します。
2. テーブルビューから、[エディター] ウィンドウの右上にある [設定アイコン] を選択します。
3. [最大結果数] オプションにカーソルを合わせて、使用可能な最大結果値のリストを表示します。

DynamoDB テーブルをスキャンする

DynamoDB テーブルをスキャンするには、次の手順を実行します。

> **Note**
>
> このスキャンでは、PartiQL クエリが生成され、適切な AWS Identity and Access Management (AWS IAM) ポリシーが設定されている必要があります。PartiQL セキュリティポリシーの要件の詳細については、「Amazon DynamoDB デベロッパーガイド」の「DynamoDB 用 PartiQL における IAM セキュリティポリシー」トピックを参照してください。

1. AWS エクスプローラーでテーブルをダブルクリックして、そのテーブルを JetBrains [エディター] ウィンドウに表示します。
2. テーブルビューから、[スキャン] ヘッダーを展開します。
3. ドロップダウンメニューから、スキャンしたい [テーブル / インデックス] を選択します。
4. [実行] を選択してスキャンへと続行します。テーブルデータが [エディター] ウィンドウに返された時点でスキャンは完了となります。

AWS Toolkit for JetBrains を使用して Amazon Elastic Container Service を操作する

次のトピックでは、AWS Toolkit for JetBrains を使用して AWS アカウントの Amazon ECS リソースを操作する方法について説明します。

トピック

- AWS Toolkit の Amazon Elastic Container Service (Amazon ECS) Exec

AWS Toolkit の Amazon Elastic Container Service (Amazon ECS) Exec

Amazon ECS Exec 機能を使用して、AWS Toolkit から直接単一のコマンドを発行したり、Amazon Elastic Container Service (Amazon ECS) コンテナ でシェルを実行したりすることができます。

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

Amazon Exec の前提条件

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

> **Important**
>
> 特定のサービスで Amazon ECS Exec を有効にする場合には、そのサービスの Amazon ECS Cloud Debugging を無効にする必要があります。

Amazon ECS の要件

Amazon ECS Exec には、タスクが Amazon EC2 でホストされているか AWS Fargate (Fargate) でホストされているかに応じて、異なるバージョン要件があります。

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

AWS アカウントの設定と IAM のアクセス許可

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

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

Amazon ECS Exec の操作

Amazon ECS Exec は、AWS Toolkit for JetBrains の AWS エクスプローラーから直接有効または無効にできます。Amazon ECS Exec を有効にしたら、Amazon ECS メニューからコンテナを選択し、それらに対してコマンドを実行することができます。

Amazon ECS Exec の有効化

1. AWS エクスプローラー で、[Amazon ECS] メニューを展開します。
2. [クラスター] セクションを展開し、変更するクラスターを選択します。

3. 編集するサービスのコンテキストメニュー (右クリック) を開き、[コマンドの実行を有効にする] を選択します。

> **Note**
>
> このサービスで Amazon ECS Cloud Debugging が有効になっている場合は、[コマンドの実行を有効にする] オプションは使用できません。Cloud Debugging を無効にするとオプションが復元されますが、サービスが停止して再起動されます。

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

Amazon ECS Exec の無効化

1. AWS エクスプローラー で、[Amazon ECS] メニューを展開します。
2. [クラスター] セクションを展開し、変更するクラスターを選択します。
3. 編集するサービスのコンテキストメニュー (右クリック) を開き、[コマンドの実行を無効にする] を選択します。

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

コンテナに対するコマンドの実行

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

1. AWS エクスプローラー で、[Amazon ECS] メニューを展開します。
2. [クラスター] セクションを展開し、変更するクラスターを選択します。

3. サービスを展開してコンテナを一覧表示します。
4. 編集するコンテナのコンテキストメニュー (右クリック) を開き、[コンテナでコマンドを実行する] を選択します。
5. [コンテナでコマンドを実行する] ダイアログボックスで、必要となる [タスク ARN] を選択します。
6. 実行するコマンドを入力するか、同じセッション中に実行されたコマンドのリストの中から選択することができます。
7. [Execute] (実行) を選択します。

シェル内からコマンドを実行する

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

1. AWS エクスプローラー で、[Amazon ECS] メニューを展開します。
2. [クラスター] セクションを展開し、変更するクラスターを選択します。
3. サービスを展開して、コンテナを一覧表示します。
4. 編集するコンテナのコンテキストメニュー (右クリック) を開き、[インタラクティブシェルを開く] を選択します。
5. [インタラクティブシェル] ダイアログボックスで、必要な [タスク ARN] を選択します。
6. 対応するドロップダウンからシェルを選択するか、操作したいシェルの名前を入力します。
7. 設定が満足できるものになったら、[実行] を選択します。
8. シェルがターミナルで開いたら、コマンドを入力してコンテナを使用することができます。

AWS Toolkit for JetBrains を使用して Amazon EventBridge を使用する

次のトピックでは、AWS Toolkit for JetBrains を使用して AWS アカウントの Amazon EventBridge スキーマを使用する方法について説明します。

トピック

- Amazon EventBridge スキーマを使用する

Amazon EventBridge スキーマを使用する

AWS Toolkit for JetBrains を使用して、次のように Amazon EventBridge スキーマを使用することができます。

> **Note**
>
> EventBridge スキーマの使用は、現在 AWS Toolkit for IntelliJ および AWS Toolkit for PyCharm でのみサポートされています。

次の情報は、すでにAWS Toolkit for JetBrains を設定していることを前提としています。

目次

使用可能なスキーマを表示する

1. [AWS Explorer] ツールウィンドウが表示されたら、[スキーマ] を展開します。
2. 表示するスキーマを含むレジストリの名前を展開します。例えば、AWS が提供するスキーマの多くは aws.events レジストリにあります。
3. エディタでスキーマを表示するには、スキーマのタイトルを右クリックし、コンテキストメニューで [スキーマの表示] を選択します。

使用可能なスキーマを検索する

[AWS Explorer] ツールウィンドウが表示されたら、次のいずれかの操作を行います。

- 検索するスキーマのタイトルの入力を開始します。[AWS Explorer] で、一致を含むスキーマのタイトルがハイライト表示されます。
- [スキーマ] を右クリックし、コンテキストメニューから [スキーマの検索] を選択します。[EventBridge スキーマの検索] ダイアログボックスで、検索するスキーマのタイトルの入力を開始します。ダイアログボックスには、一致を含むスキーマのタイトルが表示されます。

- [スキーマ] を展開します。検索するスキーマが含まれているレジストリの名前を右クリックし、[Search Schemas in Registry (レジストリのスキーマの検索)] を選択します。[EventBridge スキーマの検索] ダイアログボックスで、検索するスキーマのタイトルの入力を開始します。ダイアログボックスには、一致を含むスキーマのタイトルが表示されます。

一致のリスト内のスキーマを表示するには、次のいずれかの操作を行います。

- エディタにスキーマを表示するには、AWS Explorer でスキーマのタイトルを右クリックし、[スキーマの表示] を選択します。
- [EventBridge スキーマの検索] ダイアログボックスで、スキーマを表示するスキーマのタイトルを選択します。

使用可能なスキーマのコードを生成する

1. [AWS Explorer] ツールウィンドウが表示されたら、[スキーマ] を展開します。
2. コードを生成するスキーマを含むレジストリの名前を展開します。
3. スキーマのタイトルを右クリックし、[Download code bindings (コードバインドのダウンロード)] を選択します。
4. [Download code bindings (コードバインドのダウンロード)] ダイアログボックスで、次の項目を選択します。
   - コードを生成するスキーマのバージョン。
   - コードを生成する、サポートされているプログラミング [言語] および言語バージョン。
   - 生成されたコードをローカルの開発マシンに保存する [ファイルの場所]。
5. [ダウンロード] を選択します。

使用可能なスキーマを使用する AWS Serverless Application Model アプリケーションを作成する

1. ファイル メニューで 新規、プロジェクトを選択します。
2. [New Project] ダイアログボックスで、[AWS] を選択します。
3. [AWS サーバーレスアプリケーション] を選択してから、[Next] (次へ) をクリックします。
4. 次を指定します:
   - プロジェクトのプロジェクト名。

- プロジェクトのローカル開発マシン上の [プロジェクトの場所]。
- プロジェクトに対してサポートされている AWS Lambda ランタイム。
- プロジェクトの AWS Serverless Application Model (AWS SAM) SAM テンプレート。現在、次の選択肢があります。
  - AWS SAM EventBridge Hello World (EC2 インスタンスの状態変更) – デプロイすると、AWS アカウントに AWS Lambda 関数と関連付けられた Amazon API Gateway エンドポイントが作成されます。デフォルトでは、この関数とエンドポイントは Amazon EC2 インスタンスのステータスの変更にのみ対応します。
  - [一から AWS SAM EventBridge アプリを作成する (スキーマレジストリからのイベントトリガー用)] – デプロイすると、AWS アカウントに AWS Lambda 関数および関連付けられた Amazon API Gateway エンドポイントが作成されます。この関数とエンドポイントは、指定したスキーマで使用可能なイベントに応答できます。

    このテンプレートを選択する場合は、以下も指定する必要があります。
    - 使用する名前付きプロファイル、[認証情報]。
    - 使用する AWS リージョン。
    - 使用する EventBridge の [イベントスキーマ]。
- プロジェクトに使用する SDK のバージョン (プロジェクト SDK)。

AWS サーバーレスアプリケーションプロジェクトを作成した後は、次の操作を実行できます。

- アプリケーションのデプロイ
- アプリケーションの設定の変更(更新)
- デプロイされたアプリケーションの削除

また、アプリケーションの一部である Lambda 関数を使用して、次の操作を実行することもできます。

- 関数のローカルバージョンの実行 (呼び出し) またはデバッグ
- 関数のリモートバージョンを実行する (呼び出す)
- 関数の設定を変更する
- 関数を削除する

AWS Toolkit for JetBrains から AWS Lambda を操作する

次の項目では、AWS Toolkit for JetBrains から AWS Lambda 関数を使用する方法について説明します。

トピック

AWS Toolkit for JetBrains における AWS Lambda ランタイムとサポート

AWS Lambda では、複数の言語が、ランタイムで使用できるようにサポートされています。ランタイムは、Lambda と関数の間の呼び出しイベント、コンテキスト情報、レスポンスを中継する言語固有の環境を提供します。Lambda サービスとサポートされているランタイムの詳細については、「AWS Lambda ユーザーガイド」の「Lambda ランタイム」トピックを参照してください。

次の内容では、AWS Toolkit for JetBrains の使用で現在サポートされているランタイム環境について説明しています。

名前	識別子	オペレーティングシステム	アーキテクチャ	
Node.js 18	nodejs18.x	Amazon Linux 2	x86_64、arm64	
Node.js 16	nodejs16.x	Amazon Linux 2	x86_64、arm64	
Node.js 14	nodejs14.x	Amazon Linux 2	x86_64、arm64	
Python 3.11	python3.11	Amazon Linux 2	x86_64、arm64	
Python 3.10	python3.10	Amazon Linux 2	x86_64、arm64	

名前	識別子	オペレーティングシステム	アーキテクチャ
Python 3.9	python3.9	Amazon Linux 2	x86_64、arm64
Python 3.8	python3.8	Amazon Linux 2	x86_64、arm64
Python 3.7	python3.7	Amazon Linux 2	x86_64
Java 17	java17	Amazon Linux 2	x86_64、arm64
Java 11	java11	Amazon Linux 2	x86_64、arm64
Java 8	java8.al2	Amazon Linux 2	x86_64、arm64
Java 8	java8	Amazon Linux 2	x86_64
.NET 6	dotnet6	Amazon Linux 2	x86_64、arm64
Go 1.x	go1.x	Amazon Linux 2	x86_64

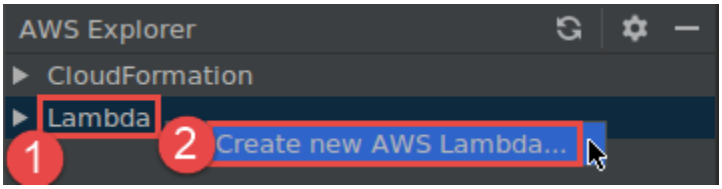
AWS Toolkit for JetBrains を使用して AWS Lambda 関数を作成する

AWS Toolkit for JetBrains を使用して、AWS サーバーレスアプリケーションの一部である AWS Lambda 関数を作成できます。または、スタンドアロンの Lambda 関数を作成することもできます。

AWS サーバーレスアプリケーションの一部である Lambda 関数を作成するには、この項目の残り部分をスキップし、アプリケーションを作成する を参照してください。

スタンドアロン Lambda 関数を作成するには、最初に AWS Toolkit for JetBrains をインストールする必要があります。まだ AWS アカウントへ接続していない場合、初回接続を行ってください。その後、IntelliJ IDEA、PyCharm、WebStorm、JetBrains Rider が実行されている状態で、次のいずれかを実行します

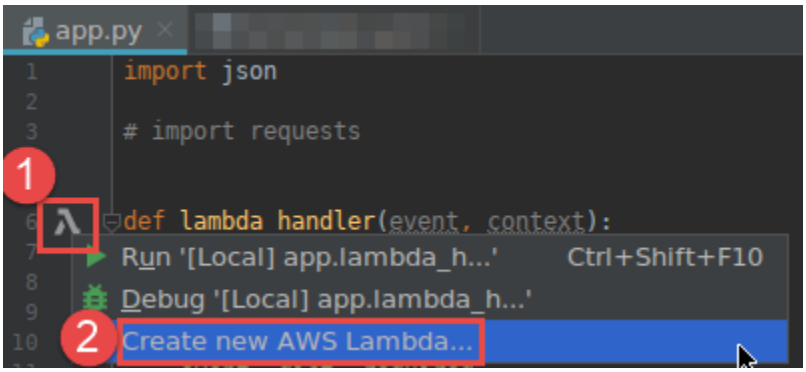
- AWS Explorer を開きます (まだ開いていない場合)。関数を作成するために別の AWS リージョンに切り替える必要がある場合、今すぐ行います。その後、[Lambda] を右クリックして[新規 AWS Lambda の作成] を選択します。



[関数の作成] ダイアログボックスに入力し、[関数の作成] を選択します。AWS Toolkit for JetBrains はデプロイに対応する AWS CloudFormation スタックを作成し、AWS エクスプローラーの [Lambda] リストに関数名を追加します。デプロイが失敗した場合、スタックのイベント ログを確認して原因を特定することができます。

- Java、Python、Node.js、または C# の関数ハンドラを実装するコードファイルを作成します。

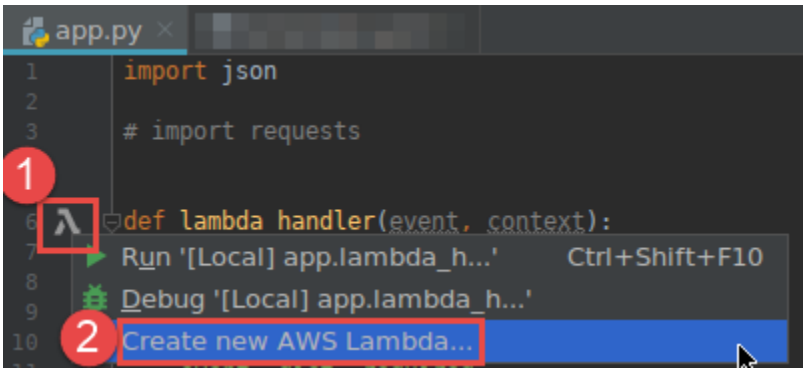
  実行 (呼び出し) するリモート関数を作成するために、別の AWS リージョンに切り替える必要がある場合、今すぐ行います。その後、コードファイルで関数ハンドラーの横にあるガター内の [Lambda] アイコンを選択し、[新規 AWS Lambda の作成] を選択します。[関数の作成] ダイアログボックスに入力し、[関数の作成] を選択します。



> **Note**
>
> [Lambda] アイコンが関数ハンドラーの横にあるガターに表示さない場合、[設定]/[ユーザー設定]: [ツール]、[AWS]、[プロジェクト設定]、[すべての潜在的な AWS Lambda ハンドラーのガターアイコンを表示する] とボックスを選択し、現在のプロジェクトで表示されるようにしてください。また、関数ハンドラーが対応する AWS SAM テンプレートですでに定義されている場合、[新規 AWS Lambda の作成] コマンドは表示されません。

[関数の作成] を選択すると、AWS Toolkit for JetBrains は、接続された AWS アカウントに対応する関数を Lambda サービスに作成します。オペレーションが正常に実行された場合、AWS エクスプローラーを更新した後、[Lambda] リストに新しい関数の名前が表示されます。

- AWS Lambda 関数を含むプロジェクトがすでにあり、最初に別の AWS リージョンに切り替えて関数を作成する必要がある場合、今すぐ行います。その後、Java、Python、Node.js、C# の関数ハンドラーを含むコードファイルで、関数ハンドラーの横にあるガターの [Lambda] アイコンを選択します。[新規 AWS Lambda の作成] を選択し、[関数の作成] ダイアログボックスに入力したら、[関数の作成] を選択します。



> **Note**
>
> [Lambda] アイコンが関数ハンドラーの横にあるガターに表示さない場合、[設定]/[ユーザー設定]: [ツール]、[AWS]、[プロジェクト設定]、[すべての潜在的な AWS Lambda ハンドラーのガターアイコンを表示する] とボックスを選択し、現在のプロジェクトで表示されるようにしてください。また、関数ハンドラーが対応する AWS SAM テンプレートですでに定義されている場合、[新規 AWS Lambda の作成] コマンドは表示されません。

[関数の作成] を選択すると、AWS Toolkit for JetBrains は、接続された AWS アカウントに対応する関数を Lambda サービスに作成します。AWS エスクプローラーを更新した後、操作が正常に実行された場合、[Lambda] リストに新しい関数の名前が表示されます。

関数を作成すると、関数のローカルバージョンを実行 (呼び出し) またはデバッグしたり、リモートバージョンを実行 (呼び出し) したりすることができます。

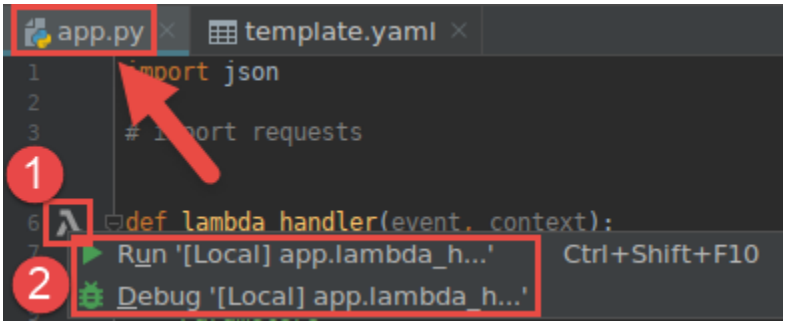
AWS Toolkit for JetBrains を使用して AWS Lambda 関数のローカル バージョンの実行 (呼び出し) またはデバッグを行う

この手順を完了するには、実行 (呼び出す) またはデバッグする AWS Lambda 関数を作成する必要があります(まだ作成していない場合)。

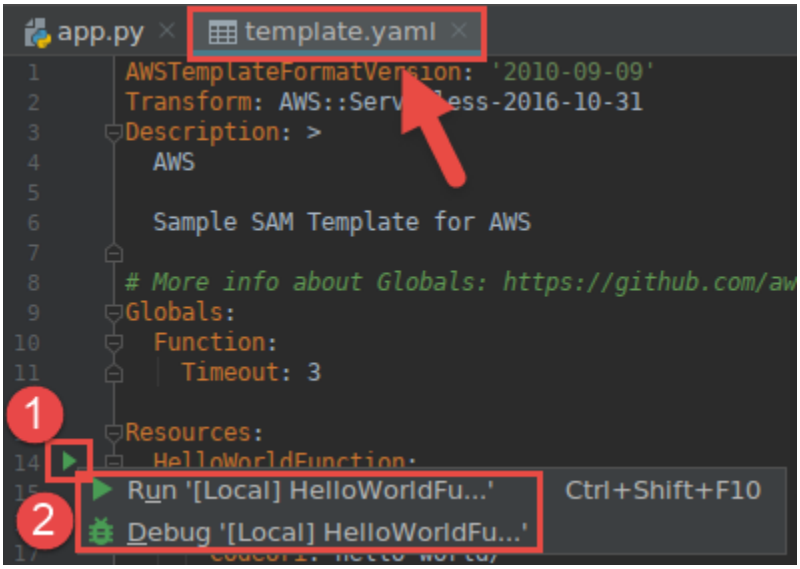
> **Note**
>
> Lambda 関数のローカル バージョンを実行(呼び出す)またはデバッグし、デフォルト以外の任意プロパティまたはオプションのプロパティでその関数をローカルで実行(呼び出す)またはデバッグするには、その関数の対応する AWS SAM テンプレートファイル (たとえば、プロジェクト内の `template.yaml` という名前のファイル内) にこれらのプロパティを最初に設定する必要があります。利用可能なプロパティのリストについては、GitHub の awslabs/serverless-application-model リポジトリで「AWS::Serverless::Function」を参照してください。

1. 以下のいずれかを実行します。

   - Java、Python、Node.js、C# の関数ハンドラーを含むコードファイルで、関数ハンドラーの横にあるガターの Lambda アイコンを選択します。[Run '[Local]' (「[ローカル]」の実行)] または [Debug '[Local]' (「[ローカル]」のデバッグ)] を選択します。

   - [プロジェクト] ツールウィンドウがすでに開かれていて、関数を含むプロジェクトが表示されている状態で、プロジェクトの `template.yaml` ファイルを開きます。関数のリソース定義の横にあるガターで [実行] アイコンを選択し、[Run '[Local]' (「[ローカル]」の実行)] または [Debug '[Local]' (「[ローカル]」のデバッグ)] を選択します。



2. [設定の編集(ローカル関数の設定)] ダイアログボックスが表示されたら入力し、[実行] または [デバッグ] を選択します。結果は、[実行] または [デバッグ] ツールウィンドウに表示されます。

   - [設定の編集] ダイアログボックスが表示されず、既存の設定を変更する場合、最初に設定を変更して最初からこの手順を繰り返します。
   - 設定の詳細が見つからない場合は、[テンプレート]、[AWS Lambda] を展開し、[ローカル] を選択します。[OK] を選択し、最初からこの手順を繰り返します。

AWS Toolkit for JetBrains を使用して AWS Lambda 関数のリモートバージョンを実行する (呼び出す)

AWS Lambda 関数のリモートバージョンは、AWS アカウントの Lambda サービス内にすでに存在するソースコードの関数です。

この手順を完了するには、最初に AWS Toolkit for JetBrains をインストールする必要があります。まだインストールしていない場合、最初に AWS アカウントに接続してください。その後、IntelliJ IDEA、PyCharm、WebStorm、JetBrains Rider が実行されている状態で、次の手順を実行してください。

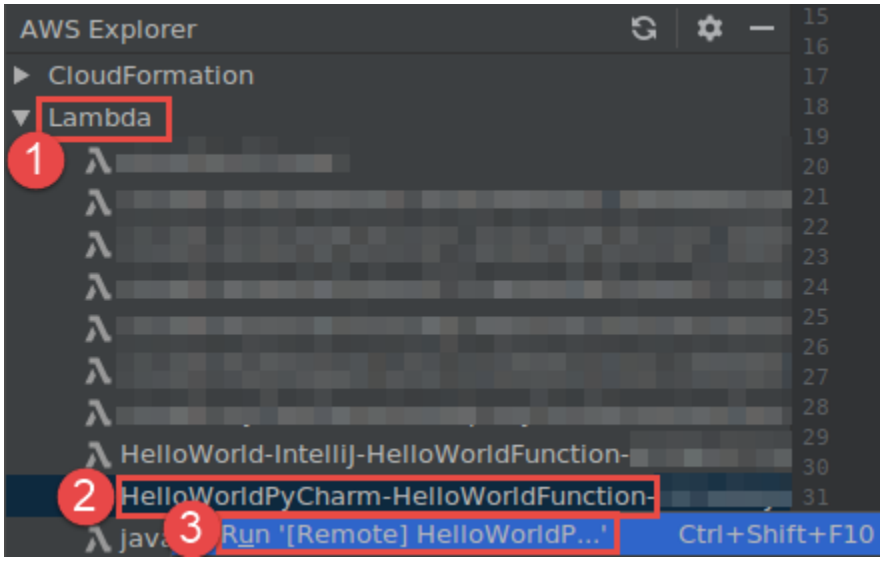
1. AWS Explorer を開きます (まだ開いていない場合)。関数を含む別の AWS リージョンに切り替える必要がある場合、今すぐ行います。
2. [Lambda] を展開し、関数の名前が一覧表示されていることを確認します。次に、この手順のステップ 3 にスキップします。

関数の名前が一覧表示されていない場合、実行する (呼び出す) Lambda 関数を作成してください。

AWS サーバーレスアプリケーションの一部として関数を作成した場合、そのアプリケーションもデプロイする必要があります。

Java、Python、Node.js、C# の関数ハンドラーを実装するコードファイルの作成によって関数を作成した場合、コードファイル内で関数ハンドラーの横にある Lambda アイコンを選択します。その後、[新規 AWS Lambda の作成] を選択します。[関数の作成] ダイアログボックスに入力し、[関数の作成] を選択します。

3. AWS エクスプローラーで [Lambda] を開いた状態で、関数の名前を右クリックして [Remote (リモート) を実行]を選択します。



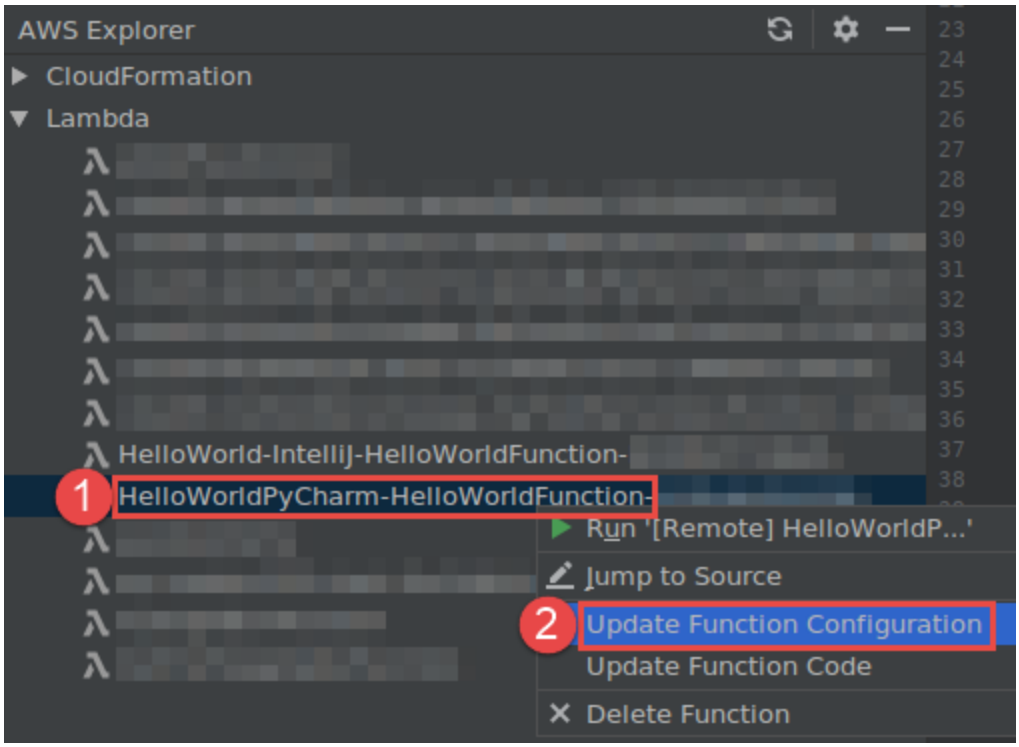
4. [設定の編集 (リモート関数の設定)] ダイアログボックスが表示されたら入力し、[実行] または [デバッグ] を選択します。結果は、[実行] または [デバッグ] ツールウィンドウに表示されます。

   - [設定の編集] ダイアログボックスが表示されず、既存の設定を変更する場合、最初に設定を変更して最初からこの手順を繰り返します。
   - 設定の詳細が見つからない場合は、[テンプレート]、[AWS Lambda] を展開し、[ローカル] を選択します。[OK] を選択し、最初からこの手順を繰り返します。

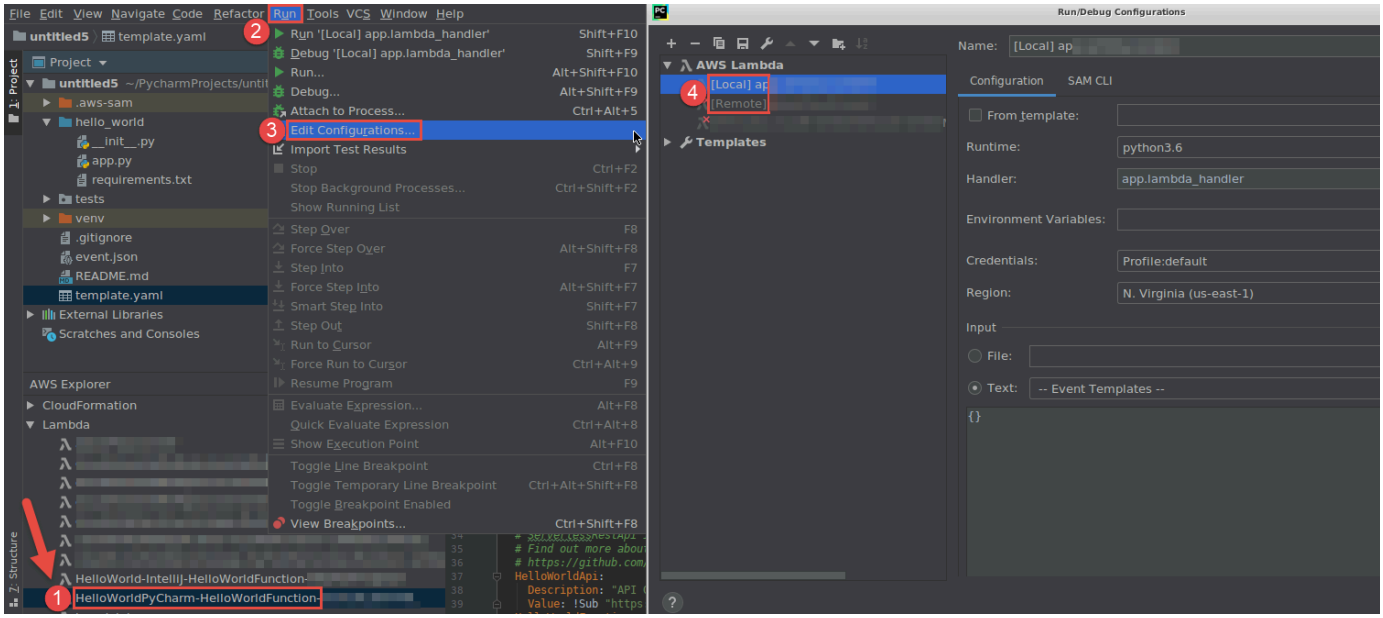
AWS Toolkit for JetBrains を使用して AWS Lambda 関数設定を変更 (更新) する

AWS Toolkit for JetBrains を使用して AWS Lambda 関数の設定を変更 (更新) するには、次のいずれかの操作を行います。

- Java、Python、Node.js、または C# の関数ハンドラを含むコードファイルを開いた状態で、メインメニューで、[実行]、[設定の編集] を選択します。[Run/Debug Configurations (実行/デバッグ設定)] ダイアログボックスに入力し、[OK] を選択します。
- AWS Explorer を開きます (まだ開いていない場合)。関数を含む別の AWS リージョンに切り替える必要がある場合、今すぐ行います。[Lambda] を展開して設定を変更する関数の名前を選択したら、次のいずれかの操作を行います。
  - タイムアウト、メモリ、環境変数、実行ロールなどの設定の変更 - 関数の名前を右クリックし、[関数設定の更新] を選択します。

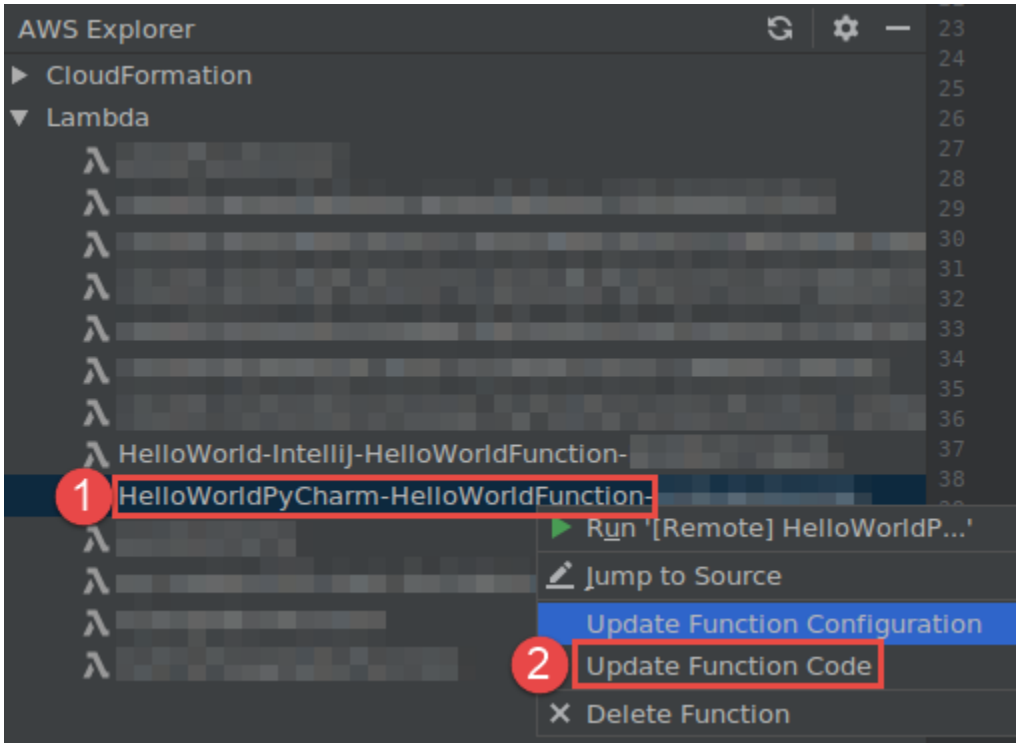


    [設定の更新] ダイアログボックスに入力し、[更新] を選択します。
  - 入力ペイロードなどの設定の変更 - メインメニューで[実行]、[設定の編集] を選択します。[Run/Debug Configurations (実行/デバッグ設定)] ダイアログボックスに入力し、[OK] を選択します。



設定の詳細が見つからない場合は、まず [テンプレート]、[AWS Lambda] を展開し、次に (関数のローカルバージョンの) [ローカル] または (同じ関数のリモートバージョンの) [リモート] を選択します。[OK] を選択し、この手順を最初から繰り返します)。

- 関数ハンドラー名や Amazon Simple Storage Service (Amazon S3) ソース バケットなどの設定の変更 - 関数名を右クリックし、[関数コードの更新] を選択します。



[Update Code (コードの更新)] ダイアログボックスに入力し、[更新] を選択します。

- 上記の箇条書きに記載されていないその他の利用可能なプロパティ設定の変更 - 関数の対応する AWS SAM テンプレートファイル(たとえば、プロジェクト内の`template.yaml`という名前のファイル)でこれらの設定を変更します。

  利用可能なプロパティ設定のリストについては、GitHub のawslabs/serverless-application-model リポジトリのAWS::Serverless::Functionを参照してください。

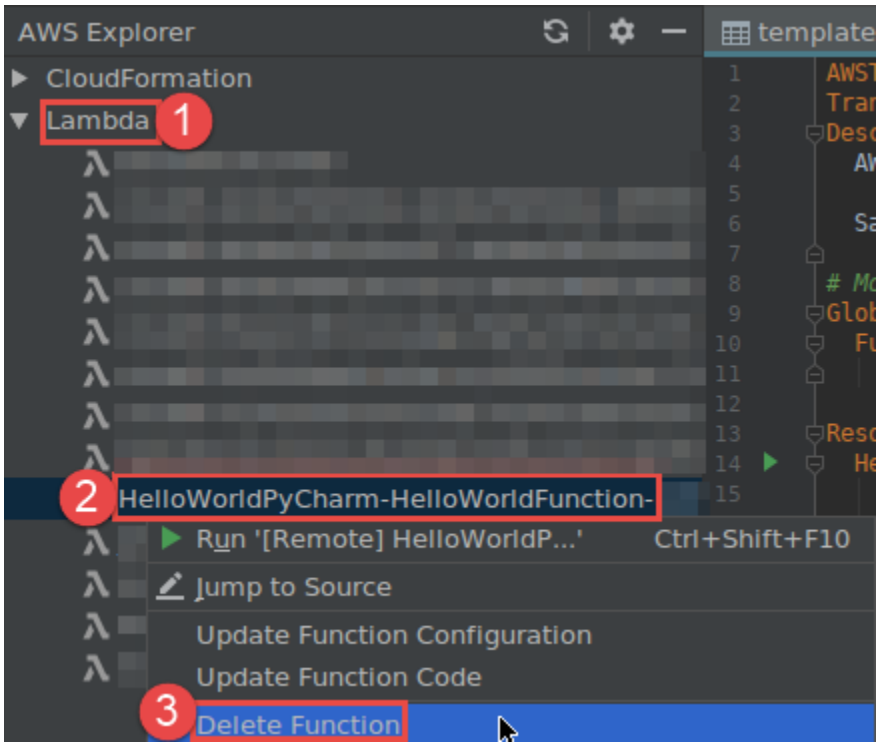
AWS Toolkit for JetBrains を使用して AWS Lambda 関数を削除する

AWS Toolkit を使用し、AWS サーバーレスアプリケーションの一部である AWS Lambda 関数を削除したり、スタンドアロン Lambda 関数を削除したりすることができます。

AWS サーバーレスアプリケーションの一部である Lambda 関数を削除するには、この項目の残り部分をスキップし、アプリケーションの削除 を参照してください。

スタンドアロン Lambda 関数を削除するには、次の手順を実行してください。

1. AWS Explorer を開きます (まだ開いていない場合)。関数を含む別の AWS リージョンに切り替える必要がある場合、今すぐ行います。
2. [Lambda] を展開します。
3. 削除する関数の名前を右クリックし、[Delete Function (関数の削除)] を選択します。



4. 関数名を入力して削除を確認し、[OK] を選択します。関数の削除が成功すると、AWS Toolkit for JetBrains により [Lambda] リストから関数名が削除されます。

AWS Toolkit for JetBrains を使用して Amazon RDS にアクセスする

Amazon Relational Database Service (Amazon RDS)を使用すると、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と管理ができます。AWS Toolkit for JetBrains を使用すると、次の Amazon RDS データベースエンジンに接続して対話を行えます。

- Aurora - MySQL と PostgreSQL と互換性のあるリレーショナル データベースで、クラウド用に構築されたものです。詳細については、「Amazon Aurora ユーザーガイド」を参照してください。
- MySQL - Amazon RDS は、オープンソースリレーショナルデータベースのいくつかの主要バージョンをサポートしています。詳細については、「Amazon RDS ユーザーガイド」の「Amazon RDS の MySQL」を参照してください。
- PostgreSQL - Amazon RDS は、オープンソース オブジェクト リレーショナル データベースのいくつかの主要バージョンをサポートしています。詳細については、「Amazon RDS ユーザーガイド」の「Amazon RDS の PostgreSQL」を参照してください。

次の項目では、RDS データベースにアクセスするための前提条件、ならびに AWS Toolkit for JetBrains を使用してデータベースインスタンスに接続する方法について説明します。

トピック

- Amazon RDS データベースにアクセスするための前提条件
- Amazon RDS データベースに接続する

Amazon RDS データベースにアクセスするための前提条件

AWS Toolkit for JetBrains を使用して Amazon RDS データベースに接続する前に、次のタスクを完了する必要があります

- DB インスタンスを作成して認証方法のセットアップ
- DataGrip のダウンロードとインストール

Amazon RDS DB インスタンスの作成と認証方法の設定

AWS Toolkit for JetBrains は、すでに作成されて AWS で設定された Amazon RDS DB インスタンスに接続できるようにします。DB インスタンスは、クラウドで実行される独立したデータベース環境で、ユーザーが作成したデータベースを複数含むことがあります。サポートされているデータベースエンジンの DB インスタンス作成の詳細については、「Amazon RDS ユーザーガイド」の「Amazon RDS リソースの開始方法」を参照してください。

AWS Toolkit for JetBrains を使用してデータベースに接続すると、ユーザーは IAM 認証情報または Secrets Manager を選択して認証することができます。次の表は、両方のオプションの主な機能と情報リソースを説明したものです。

認証方法	使用方法	詳細情報
IAM 認証情報で接続する	IAM データベース認証を使用すると、認証は AWS Identity and Access Management (IAM) 認証情報を使用して外部的に管理されるため、ユーザー認証情報をデータベースに保存する必要はありません。 デフォルトでは、IAM データベース認証は DB インスタンスで無効になります。AWS Management Console、AWS CLI、または API を使用して、IAM データベース認証を有効にする (またはもう一度無効にする) ことができます。	• 「Amazon RDS ユーザーガイド」の「Amazon RDS のアイデンティティとアクセス管理」。 • AWS ナレッジ センターの記事: 「ユーザーが IAM 認証情報を使用して Amazon RDS MySQL DB インスタンスに認証を許可する方法を教えてください」。
AWS Secrets Manager を使用して接続	データベース管理者は、Secrets Manager でデータベースの認証情報をシークレットとして保存できます。Secrets Manager は認証情報をシークレット内に保	• 「AWS Secrets Manager ユーザーガイド」の「AWS Secrets Managerとは?」。 • 「AWS Secrets Manager ユーザーガイド」の

認証方法	使用方法	詳細情報
	護されたシークレットテキストとして暗号化して保存します。 許可されたアプリケーションがデータベースにアクセスすると、Secrets Manager は保護されたシークレットテキストを解読し、安全なチャネルに返します。クライアントは、認証情報、接続文字列、その他の必要な情報をレスポンスから解析し、その情報を使用してデータベースにアクセスします。	「チュートリアル: AWS データベースのシークレットのローテーション」。 • AWS セキュリティブログ: 「Secrets Manager で Amazon RDS データベースの認証情報を自動的にローテーション」。

DataGrip を使用して Amazon RDS データベースを操作する

Amazon RDS データソースに接続した後、対話を開始できます。JetBrains から DataGrip を使用することで、SQL の作成、クエリの実行、データのインポート/エクスポートなどのデータベース タスクを実行できます。DataGrip が提供する機能は、さまざまな JetBrains IDE のデータベースプラグインでも利用可能です。DataGrip の詳細については、https://www.jetbrains.com/datagrip/ を参照してください。

Amazon RDS データベースに接続する

[AWS エクスプローラー]で Amazon RDS データベースを選択して認証方法を選択した後に、接続設定を構成できます。接続のテストが正常に実行した後、JetBrains DataGrip を使用してデータソースと対話を開始できます。

> **Important**
>
> ユーザーが Amazon RDS データベースにアクセスして対話できるようにするには、前提条件をすべて満たしていることを確認します。

ご希望の認証方法を使用し、データベースインスタンスに接続する手順のタブを選択します。

Connect with IAM credentials

1. AWS Explorer を開きます (まだ開いていない場合)。
2. [Amazon RDS] ノードをクリックし、サポートされているデータベースエンジンのリストを展開します。
3. サポートされているデータベースエンジン (Aurora、MySQL、PostgreSQL) ノードをクリックして、利用可能なデータベースインスタンスのリストを展開します。

   > **Note**
   >
   > Aurora を選択した場合、MySQL クラスターまたは PostgreSQL クラスターを展開するか選択できます。

4. データベースを右クリックして [IAM 認証情報で接続] を選択します。

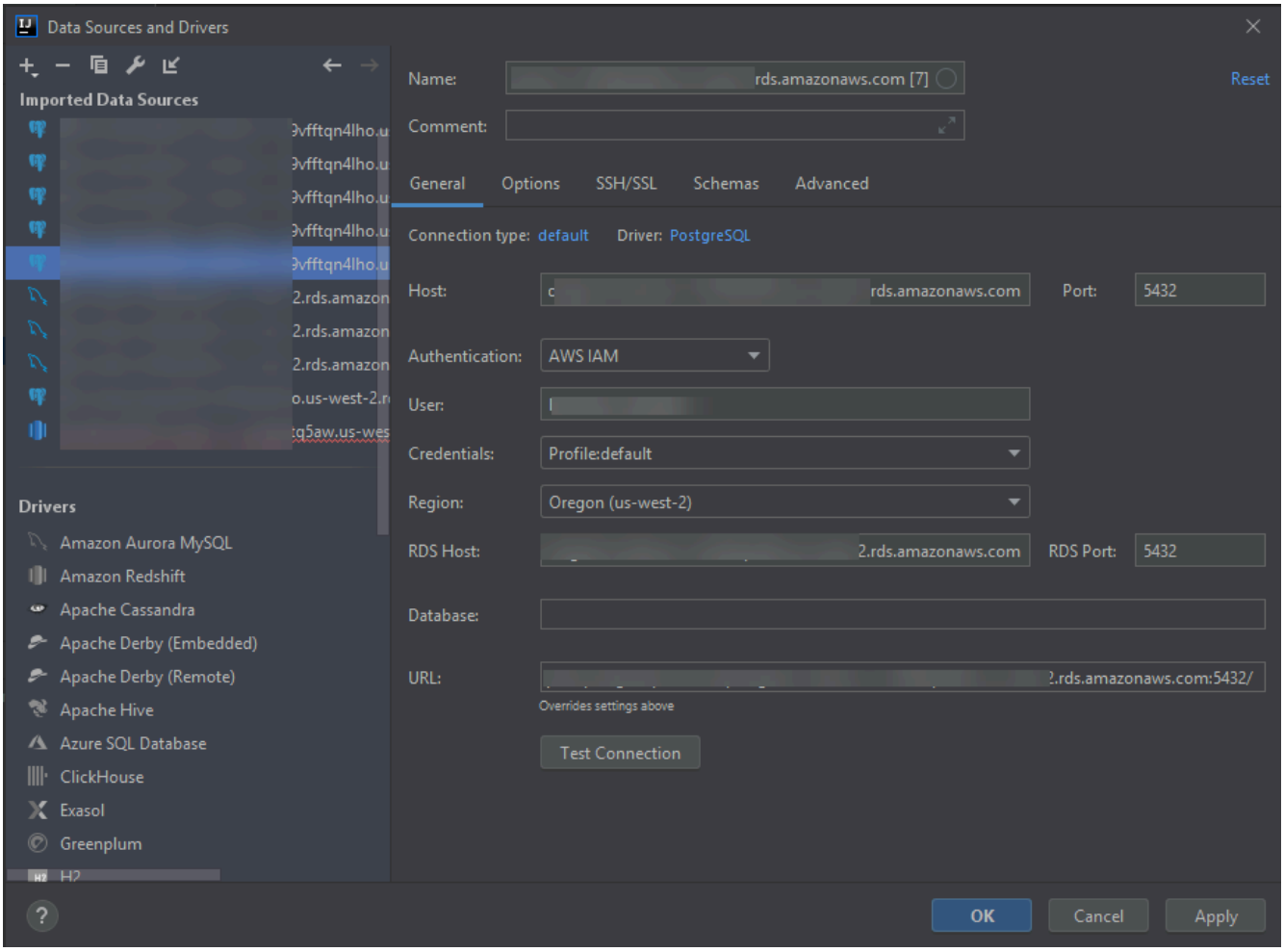
   > **Note**
   >
   > [ARN のコピー] を選択し、データベースの Amazon リソースネーム (ARN)をクリップボードに追加することもできます。

5. [データソースとドライバー] ダイアログボックスで、データベース接続を開けるようにするためには、次の操作を実行してください:

   - [インポート済データソース] ペインで、正しいデータソースが選択されていることを確認します。
   - [不足しているドライバー ファイルのダウンロード] が必要であるというメッセージが表示される場合、[ドライバーに移動] (レンチアイコン) を選択し、必要なファイルをダウンロードしてください。
6. [設定] ペインの [全般] タブで、次のフィールドに正しい値が表示されていることを確認します。

   - [ホスト/ポート] - データベースへの接続に使用されるエンドポイントとポート。AWS クラウドでホストされている Amazon RDS データベースの場合、エンドポイント末尾は常に `rds.amazon.com` です。プロキシ経由で DB インスタンスに接続している場合、これらのフィールドを使用してプロキシの接続詳細を指定します。
   - [認証] - AWS IAM(IAM 認証情報で認証)

- ユーザー - データベースユーザーアカウントの名前。
- [認証情報] - AWS アカウントにアクセスするために使用する認証情報。
- [リージョン] - データベースがホストされている AWS リージョン。
- [RDS ホスト/ポート] - AWS Management Consoleに記載されているデータベースのエンドポイントとポート 別のエンドポイントを使用して DB インスタンスに接続する場合、プロキシの接続詳細を[ホスト/ポート]フィールド(以前に説明済)に指定してください。
- [データベース] - データベースの名前。
- [URL] - JetBrains IDE がデータベースに接続するために使用する URL。



> **Note**
>
> [データソースとドライバー] ダイアログボックスを使用して設定可能な接続設定の詳細については、お使いの JetBrains IDE のドキュメンテーションを参照してください。

7. 接続設定が正しいことを確認するには、[接続のテスト] を選択します。

   緑色のチェックマークはテストが正常に完了したことを示します。

8. [適用] を選択して設定を適用し、[OK] を選択してデータソースの使用を開始します。

   [データベース] ツールウィンドウが開きます。これは利用可能なデータソースをツリーとして表示します。スキーマ、表、キーなどのデータベース要素はノードで示されます。

> **Important**
>
> [データベース] ツールウィンドウを使用するには、最初に JetBrains から DataGrip をダウンロードしてインストールする必要があります。詳細については、https://www.jetbrains.com/datagrip/ を参照してください。

Connect with Secrets Manager

1. AWS Explorer を開きます (まだ開いていない場合)。
2. [Amazon RDS] ノードをクリックし、サポートされているデータベースエンジンのリストを展開します。
3. サポートされているデータベースエンジン (Aurora、MySQL、PostgreSQL) ノードをクリックして、利用可能なデータベースインスタンスのリストを展開します。

> **Note**
>
> Aurora を選択した場合、MySQL クラスターまたは PostgreSQL クラスターを展開するか選択できます。

4. データベースを右クリックして [Secrets Manager で接続] を選択します。

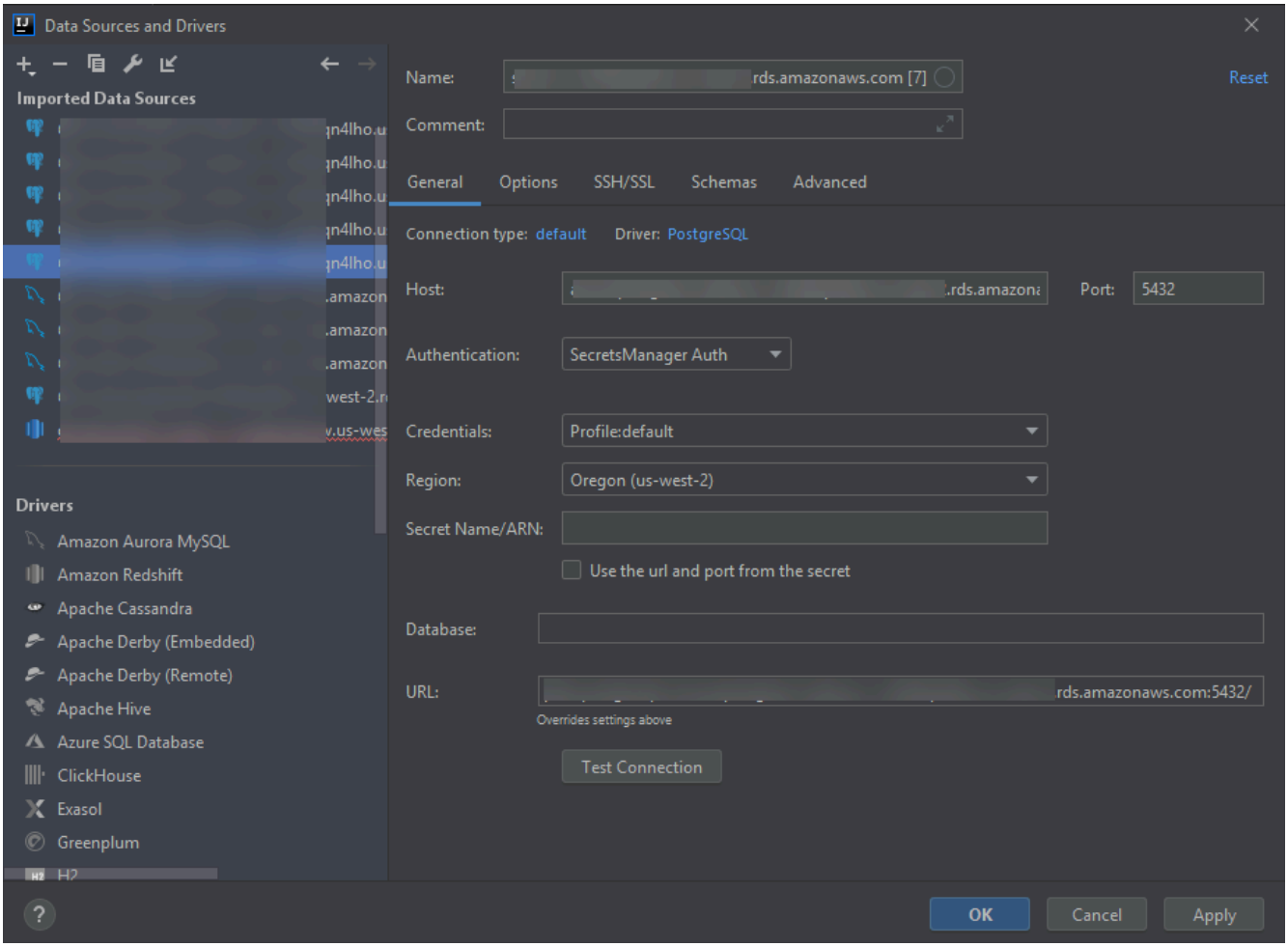
> **Note**
>
> [ARN のコピー] を選択し、データベースの Amazon リソースネーム (ARN)をクリップボードに追加することもできます。

5. [データベースシークレットの選択] ダイアログボックスで、ドロップダウン フィールドを使用してデータベースの認証情報を選択したら、[作成] を選択します。
6. [データソースとドライバー] ダイアログボックスで、データベース接続を開けるようにするためには、次の操作を実行してください:

   - [インポート済データソース] ペインで、正しいデータソースが選択されていることを確認します。
   - [不足しているドライバー ファイルのダウンロード] が必要であるというメッセージが表示される場合、[ドライバーに移動] (レンチアイコン) を選択し、必要なファイルをダウンロードしてください。
7. [設定] ペインの [全般] タブで、次のフィールドに正しい値が表示されていることを確認します。

   - [ホスト/ポート] - データベースへの接続に使用されるエンドポイントとポート。AWS クラウドでホストされている Amazon RDS データベースの場合、エンドポイント末尾は常に `rds.amazon.com` です。プロキシ データベース経由でデータベースに接続している場合、これらのフィールドを使用してプロキシの接続詳細を指定します。
   - [認証] - [SecretsManager Auth] (AWS Secrets Manager を使用した認証)。
   - [認証情報] - AWS アカウントにアクセスするために使用する認証情報。
   - [リージョン] - データベースがホストされている AWS リージョン。
   - [シークレット名/ARN] - 認証情報を含むシークレットの名前と ARN。[ホスト/ポート] フィールドで接続設定を上書きするには、[シークレットの URL とポートの使用] チェックボックスを選択します。
   - [データベース] - AWS エクスプローラーで選択したデータベースインスタンスの名前。
   - [URL] - JetBrains IDE がデータベースに接続するために使用する URL。

> **Note**
>
> 認証に Secrets Manager を使用している場合、データベースにユーザー名とパスワードのフィールドはありません。この情報は、シークレットの暗号化されたシークレットデータ部分に含まれています。



> **Note**
>
> [データソースとドライバー] ダイアログボックスを使用して設定可能な接続設定の詳細については、お使いの JetBrains IDE のドキュメンテーションを参照してください。

8. 接続設定が正しいことを確認するには、[接続のテスト] を選択します。

緑色のチェックマークはテストが正常に完了したことを示します。

9. [適用] を選択して設定を適用し、[OK] を選択してデータソースの使用を開始します。

   [データベース] ツールウィンドウが開きます。これは利用可能なデータソースをツリーとして表示します。スキーマ、表、キーなどのデータベース要素はノードで示されます。

> **Important**
>
> [データベース] ツールウィンドウを使用するには、最初に JetBrains から DataGrip をダウンロードしてインストールする必要があります。詳細については、https://www.jetbrains.com/datagrip/ を参照してください。

AWS Toolkit for JetBrains を使用して Amazon Redshift にアクセスする

Amazon Redshift データウェアハウスは、エンタープライズクラスのリレーショナルデータベースのクエリおよび管理を行うためのシステムです。AWS Toolkit for JetBrains を使用すると、Amazon Redshift クラスターに接続して対話できます。Amazon Redshift クラスターは、クライアントがそのクラスターにホストされているデータベースにクエリを実行できるようにする一連のノードで構成されています。

次の項目では、Amazon Redshift クラスターにアクセスするための前提条件、ならびに AWS Toolkit for JetBrains を使用してクラスターのデータベースに接続する方法について説明します。

トピック

- Amazon Redshift クラスターにアクセスするための前提条件
- Amazon Redshift クラスターに接続する

Amazon Redshift クラスターにアクセスするための前提条件

AWS Toolkit for JetBrains を使用して Amazon Redshift クラスターと対話を開始する前に、次のタスクを完了する必要があります。

- Amazon Redshift クラスターを作成して認証方法のセットアップ
- DataGrip のダウンロードとインストール

Amazon Redshift クラスターの作成と認証方法の設定

AWS Toolkit for JetBrains は、AWS ですでに作成と設定されている Amazon Redshift クラスターに接続できるようにします。各クラスターには、1 つ以上のデータベースが含まれています。Amazon Redshift クラスターの作成と設定方法の情報については、「Amazon Redshift 入門ガイド」の「Amazon Redshift の開始方法」を参照してください。

AWS Toolkit for JetBrains を使用してクラスターに接続すると、ユーザーは IAM 認証情報または AWS Secrets Manager を選択して認証することができます。次の表は、両方のオプションの主な機能と情報リソースを説明したものです。

認証方法	使用方法	詳細情報
IAM 認証情報で接続する	IAM データベース認証を使用すると、認証は AWS Identity and Access Management (IAM) 認証情報を使用して外部的に管理されるため、ユーザー認証情報をデータベースに保存する必要はありません。 デフォルトでは、IAM データベース認証はデータベースインスタンスで無効になります。AWS Management Console、AWS CLI、または API を使用して、IAM データベース認証を有効にする (またはもう一度無効にする) ことができます。	• 「Amazon Redshift 管理ガイド」の「Amazon Redshift でアイデンティティとアクセス管理」。
AWS Secrets Manager を使用して接続;	データベース管理者は、Secrets Manager でデータベースの認証情報をシークレットとして保存できます。Secrets Manager は認証情報をシークレット内に保	• 「AWS Secrets Manager ユーザーガイド」の「AWS Secrets Managerとは?」。 • 「AWS Secrets Manager ユーザーガイド」の

認証方法	使用方法	詳細情報
	護されたシークレットテキストとして暗号化して保存します。 許可されたアプリケーションがデータベースにアクセスすると、Secrets Manager は保護されたシークレットテキストを解読し、安全なチャネルに返します。クライアントは、認証情報、接続文字列、その他の必要な情報をレスポンスから解析し、その情報を使用してデータベースにアクセスします。	「Amazon Redshift のシークレットのローテーション」。 • AWS セキュリティブログ:「Secrets Manager で Amazon DocumentDB と Amazon Redshift 認証情報をローテーションする方法」。

DataGrip を使用して Amazon RDS データベースを操作する

Amazon Redshift クラスターのデータベースに接続した後、対話を開始できます。JetBrains から DataGrip を使用すると、SQL の作成、クエリの実行、データのインポート/エクスポートなどのデータベース タスクを実行できます。DataGrip が提供する機能は、さまざまな JetBrains IDE のデータベースプラグインでも利用可能です。DataGrip の詳細については、https://www.jetbrains.com/datagrip/ を参照してください。

Amazon Redshift クラスターに接続する

AWS エクスプローラーを使用すると、Amazon Redshift クラスターを選択して認証方法を選択したら、接続設定を構成できます。接続のテストが正常に実行した後、JetBrains DataGrip を使用してデータソースと対話を開始できます。

> **Important**
>
> ユーザーが Amazon Redshift クラスターとデータベースにアクセスして対話するには、前提条件をすべて満たしていることを確認してください。

ご希望の認証方法を使用し、クラスターに接続する手順のタブを選択します。

Connect with IAM credentials

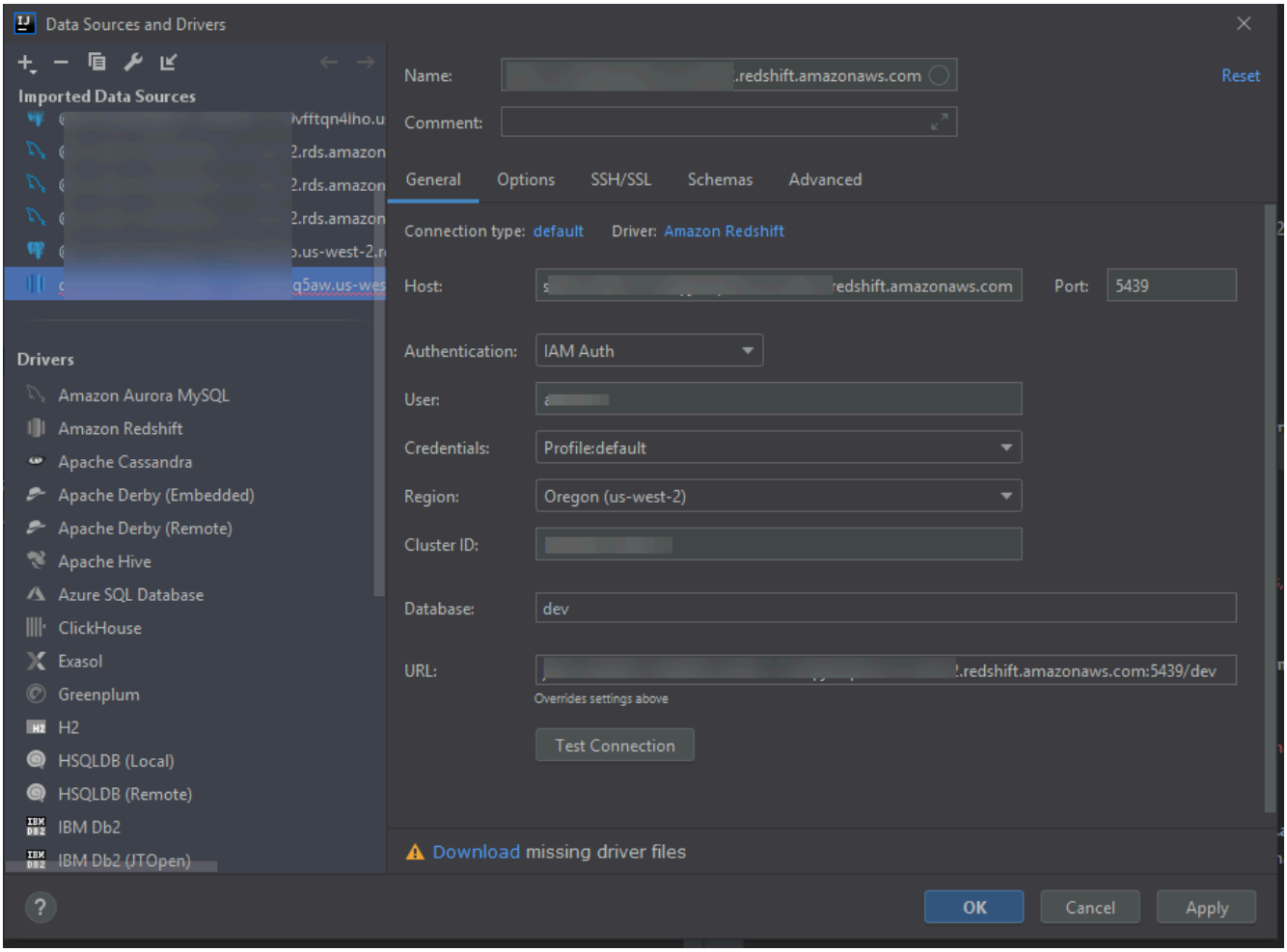
1. AWS Explorer を開きます (まだ開いていない場合)。
2. [Amazon Redshift] ノードをクリックして利用可能なクラスターのリストを展開します。
3. クラスターを右クリックして [IAM 認証情報で接続] を選択します。

   > **Note**
   >
   > [ARN のコピー] を選択し、クラスターの Amazon リソースネーム (ARN) をクリップボードに追加することもできます。

4. [データソースとドライバー] ダイアログボックスで、データベース接続を開けるようにするためには、次の操作を実行してください:

   - [インポートされたデータソース] ペインで、正しいデータソースが選択されていることを確認します。
   - [不足しているドライバー ファイルのダウンロード] が必要であるというメッセージが表示される場合、[ドライバーに移動] (レンチアイコン) を選択し、必要なファイルをダウンロードしてください。
5. [設定] ペインの [一般] タブで、次のフィールドに正しい値が表示されていることを確認します。

   - [ホスト/ポート] - クラスターへの接続に使用されるエンドポイントとポート。AWS クラウドでホストされている Amazon Redshift クラスターの場合、エンドポイント末尾は常に `redshift.amazon.com` です。
   - [認証] - AWS IAM(IAM 認証情報で認証)
   - ユーザー - データベースユーザーアカウントの名前。
   - [認証情報] - AWS アカウントにアクセスするために使用する認証情報。
   - [リージョン] - データベースがホストされている AWS リージョン。
   - [クラスター ID] - AWS エクスプローラーで選択したクラスターの ID。
   - [データベース] - 接続するクラスター内のデータベースの名前。
   - [URL] - JetBrains IDE がクラスターのデータベースに接続するために使用する URL。



> **Note**
>
> [データソースとドライバー] ダイアログボックスを使用して設定可能な接続設定の詳細については、お使いの JetBrains IDE のドキュメンテーションを参照してください。

6. 接続設定が正しいことを確認するには、[接続のテスト] を選択します。

   緑色のチェックマークはテストが正常に完了したことを示します。

7. [適用] を選択して設定を適用し、[OK] を選択してデータソースの使用を開始します。

   [データベース] ツールウィンドウが開きます。これは利用可能なデータソースをツリーとして表示します。スキーマ、表、キーなどのデータベース要素はノードで示されます。

> **Important**
>
> [データベース] ツールウィンドウを使用するには、最初に JetBrains から DataGrip をダウンロードしてインストールする必要があります。詳細については、https://www.jetbrains.com/datagrip/ を参照してください。

Connect with Secrets Manager

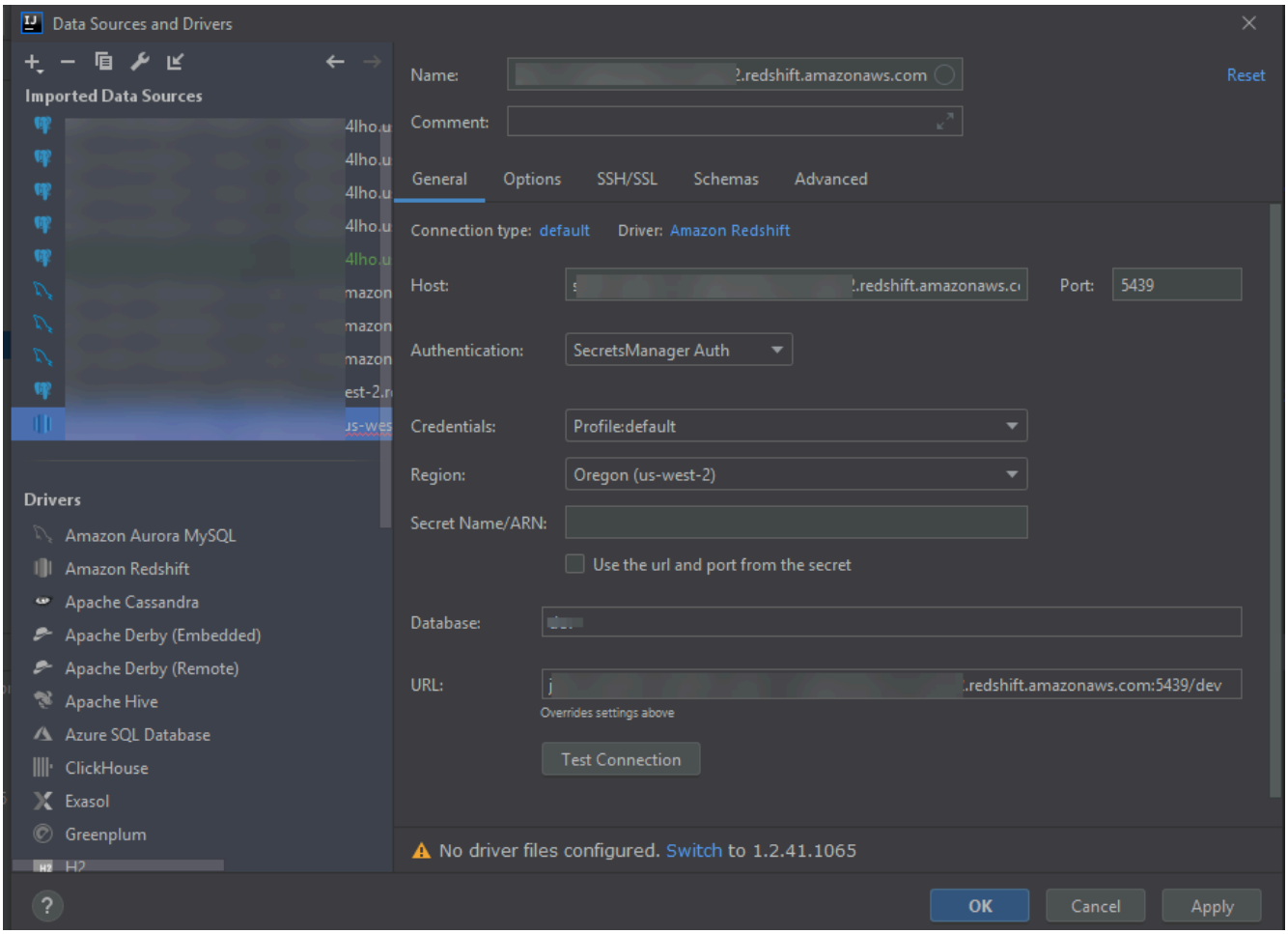
1. AWS Explorer を開きます (まだ開いていない場合)。
2. [Amazon Redshift] ノードをクリックして利用可能なクラスターのリストを展開します。
3. クラスターを右クリックして[Secrets Manager に接続] を選択します。

> **Note**
>
> [ARN のコピー] を選択し、クラスターの Amazon リソースネーム (ARN) をクリップボードに追加することもできます。

4. [データベースシークレットの選択] ダイアログボックスで、ドロップダウン フィールドを使用してデータベースの認証情報を選択したら、[作成] を選択します。
5. [データソースとドライバー] ダイアログボックスで、データベース接続を開けるようにするためには、次の操作を実行してください:
   - [インポートされたデータソース] で、正しいデータソースが選択されていることを確認します。
   - ダイアログボックスに [不足しているドライバーファイルのダウンロード] というメッセージが表示された場合、[ドライバーに移動] (レンチアイコン) を選択し、必要なファイルをダウンロードします。
6. [設定] ペインの [一般] タブで、次のフィールドに正しい値が表示されていることを確認します。
   - [ホスト/ポート] - クラスターへの接続に使用されるエンドポイントとポート。AWS クラウドでホストされている Amazon Redshift クラスターの場合、エンドポイント末尾は常に `redshift.amazon.com` です。
   - [認証] - [SecretsManager Auth] (AWS Secrets Manager を使用した認証)。
   - [認証情報] - AWS アカウントに接続するために使用される認証情報。

- [リージョン] - クラスターがホストされている AWS リージョン。
- [シークレット名/ARN] - 認証情報を含むシークレットの名前と ARN。[ホスト/ポート] フィールドで接続設定を上書きする場合、[シークレットから URL とポートを使用する] チェックボックスを選択します。
- [データベース] - 接続するクラスター内のデータベースの名前。
- [URL] - JetBrains IDE がデータベースに接続するために使用する URL。

> **Note**
>
> 認証に AWS Secrets Manager を使用している場合、クラスターのユーザー名とパスワードを指定するフィールドはありません。この情報は、シークレットの暗号化されたシークレットデータ部分に含まれています。



> **Note**
>
> [データソースとドライバー] ダイアログボックスを使用して設定可能な接続設定の詳細については、お使いの JetBrains IDE のドキュメンテーションを参照してください。

7. 接続設定が正しいことを確認するには、[接続のテスト] を選択します。

   緑色のチェックマークはテストが正常に完了したことを示します。

8. [適用] を選択して設定を適用し、[OK] を選択してデータソースの使用を開始します。

   [データベース] ツールウィンドウが開きます。これは利用可能なデータソースをツリーとして表示します。スキーマ、表、キーなどのデータベース要素はノードで示されます。

> **Important**
>
> [データベース] ツールウィンドウを使用するには、最初に JetBrains から DataGrip をダウンロードしてインストールする必要があります。詳細については、https://www.jetbrains.com/datagrip/ を参照してください。

AWS Toolkit for JetBrains を使用して Amazon S3 を操作する

次の項目では、AWS Toolkit for JetBrains を使用して AWS アカウントの Amazon S3 バケットとオブジェクトを操作する方法について説明します。

トピック

- AWS Toolkit for JetBrains を使用して Amazon S3 バケットを操作する
- AWS Toolkit for JetBrains を使用して Amazon S3 オブジェクトを操作する

AWS Toolkit for JetBrains を使用して Amazon S3 バケットを操作する

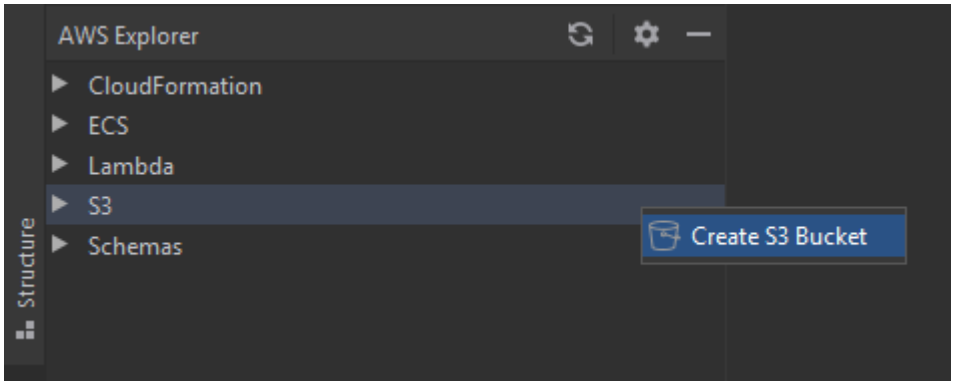
Amazon S3 に保存したオブジェクトはすべて、バケット内にあります。ファイルシステムでディレクトリを使用してファイルをグループ化するのと同じ方法で、バケットを使用して関連するオブジェクトをグループ化できます。

トピック

- Amazon S3 バケットの作成
- Amazon S3 バケットの表示
- Amazon S3 バケットの削除

Amazon S3 バケットの作成

1. AWS Explorer を開きます (まだ開いていない場合)。
2. [Amazon S3] ノードを右クリックして [S3 バケットの作成] を選択します。



3. [Create S3 Bucket (S3 バケットの作成)] ダイアログボックスで、バケットの名前を入力します。

> **注意**
>
> Amazon S3 ではバケットをパブリックにアクセス可能な URL として使用できるので、グローバルに一意なバケット名を選択する必要があります。他のアカウントで同じ名前のバケットがすでに作成されている場合は、別の名前を使用する必要があります。詳細については、Amazon Simple Storage Service ユーザーガイドの「バケットの制約と制限」を参照してください。

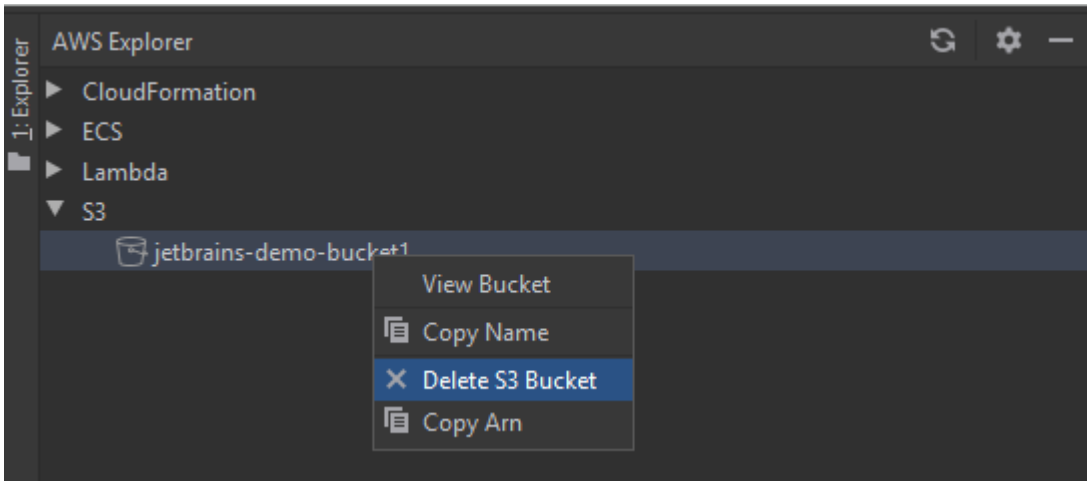
4. [Create] (作成) を選択します。

Amazon S3 バケットの表示

1. AWS Explorer を開きます (まだ開いていない場合)。
2. [Amazon S3] ノードをクリックしてバケットのリストを展開します。
   - 現在の AWS リージョンの S3 バケットは、[Amazon S3] ノードの下に表示されます。

Amazon S3 バケットの削除

1. AWS エクスプローラーを開きます(まだ開いていない場合)。
2. [Amazon S3] ノードをクリックしてバケットのリストを展開します。
3. 削除するバケットを右クリックし、[Delete S3 Bucket] (S3 バケットの削除) を選択します。



4. バケット名を入力して削除を確認し、[OK] を選択します。

   - バケットにオブジェクトが含まれている場合、バケットは削除前に空になります。削除が完了すると、通知が表示されます。

AWS Toolkit for JetBrains を使用して Amazon S3 オブジェクトを操作する

オブジェクトとは、Amazon S3 に保存される基本エンティティです。オブジェクトは、オブジェクトデータとメタデータで構成されます。

トピック

- Amazon S3 バケットでオブジェクトの表示
- IDE でオブジェクトを開く
- オブジェクトのアップロード
- オブジェクトのダウンロード
- オブジェクトの削除

Amazon S3 バケットでオブジェクトの表示

この手順では、S3 バケットビューアが開きます。これを使用して Amazon S3 バケット内のフォルダごとにグループ化されたオブジェクトを表示、アップロード、ダウンロード、削除できます。

1. AWS Explorer を開きます (まだ開いていない場合)。
2. バケットのオブジェクトを表示するには、次のいずれかを行います。

   - バケットの名前をダブルクリックします。
   - バケットの名前を右クリックし、[View Bucket (バケットの表示)] を選択します。

S3 バケットビューア には、バケット名、Amazon リソースネーム (ARN)、および作成日に関する情報が表示されます。バケット内のオブジェクトとフォルダは、下のペインで使用できます。

IDE でオブジェクトを開く

Amazon S3 バケット内のオブジェクトが IDE によって認識されるファイル タイプである場合、読み取り専用コピーをダウンロードして IDE で開くことができます。

1. ダウンロードするオブジェクトを検索するには、[S3 バケットビューア] を開きます(「Amazon S3 バケットでオブジェクトの表示」を参照)。
2. オブジェクトの名前をダブルクリックします。

ファイルは、そのファイルタイプのデフォルトの IDE ウィンドウで開きます。

オブジェクトのアップロード

1. オブジェクトをアップロードするフォルダを検索するには、S3 バケットビューアを開きます (「Amazon S3 バケットでオブジェクトの表示」を参照)。
2. フォルダを右クリックし、[アップロード] を選択します。
3. ダイアログボックスで、アップロードするファイルを選択します。

   > **注意**
   >
   > 一度に複数のファイルをアップロードできます。ディレクトリはアップロードできません。

4. [OK] をクリックします。

オブジェクトのダウンロード

1. オブジェクトをダウンロードするフォルダを検索するには、[S3 バケットビューア] を開きます(「Amazon S3 バケットでオブジェクトの表示」を参照)。
2. オブジェクトを表示するフォルダを選択します。
3. オブジェクトを右クリックし、[ダウンロード] を選択します。
4. ダイアログボックスで、ダウンロード場所を選択します。

> **注意**
>
> 複数のファイルをダウンロードする場合は、フォルダではなくパス名を選択してください。ディレクトリはダウンロードできません。

5. [OK] をクリックします。

> **注意**
>
> ファイルがすでにダウンロード場所に存在する場合は、上書きするか、ダウンロードをスキップしてそのまま残すことができます。

オブジェクトの削除

1. 削除するオブジェクトを見つけるには、[S3 バケットビューア] を開きます(「Amazon S3 バケットでオブジェクトの表示」を参照)。
2. オブジェクトを選択した後、次のいずれかの操作を行って削除します。
   - [Delete] (削除) を押します。
   - 右クリックし、[Delete] (削除) を選択します。

> **注意**
>
> 複数のオブジェクトを一度に選択および削除できます。

3. 削除を確認するには、[削除] を選択します。

AWS Toolkit for JetBrains を使用して AWS サーバーレスアプリケーションを操作する

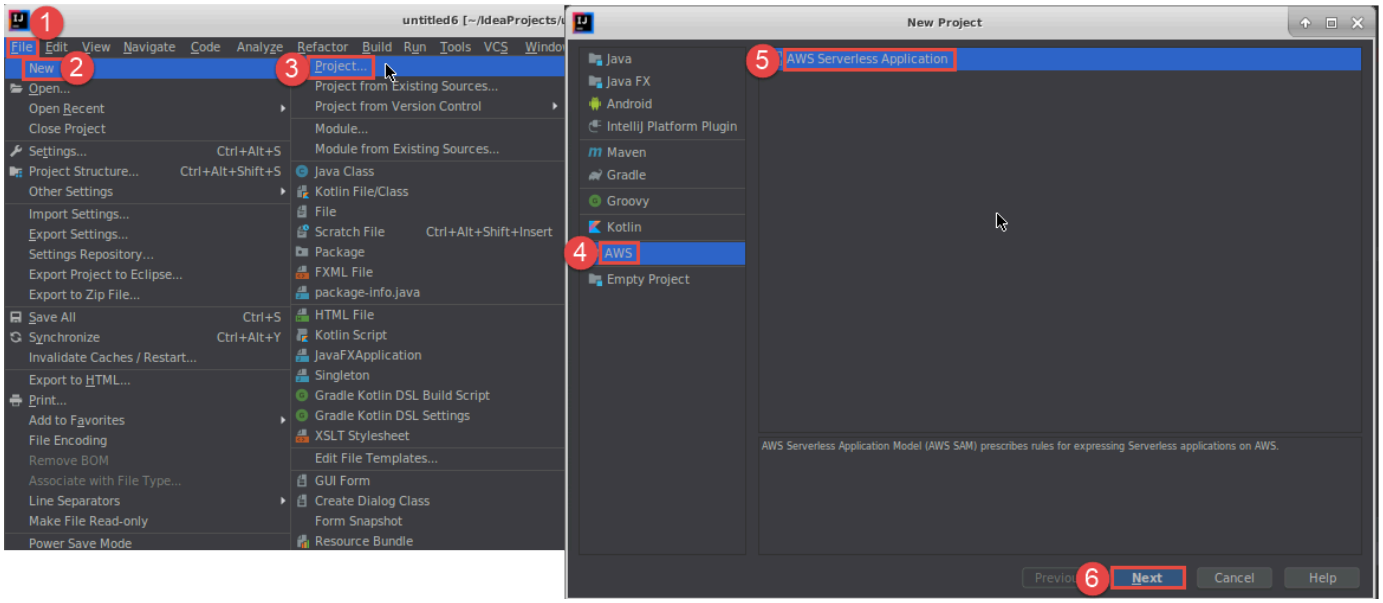
次のトピックでは、AWS Toolkit for JetBrains を使用して AWS アカウントの AWS サーバーレスアプリケーションを操作する方法について説明します。

トピック

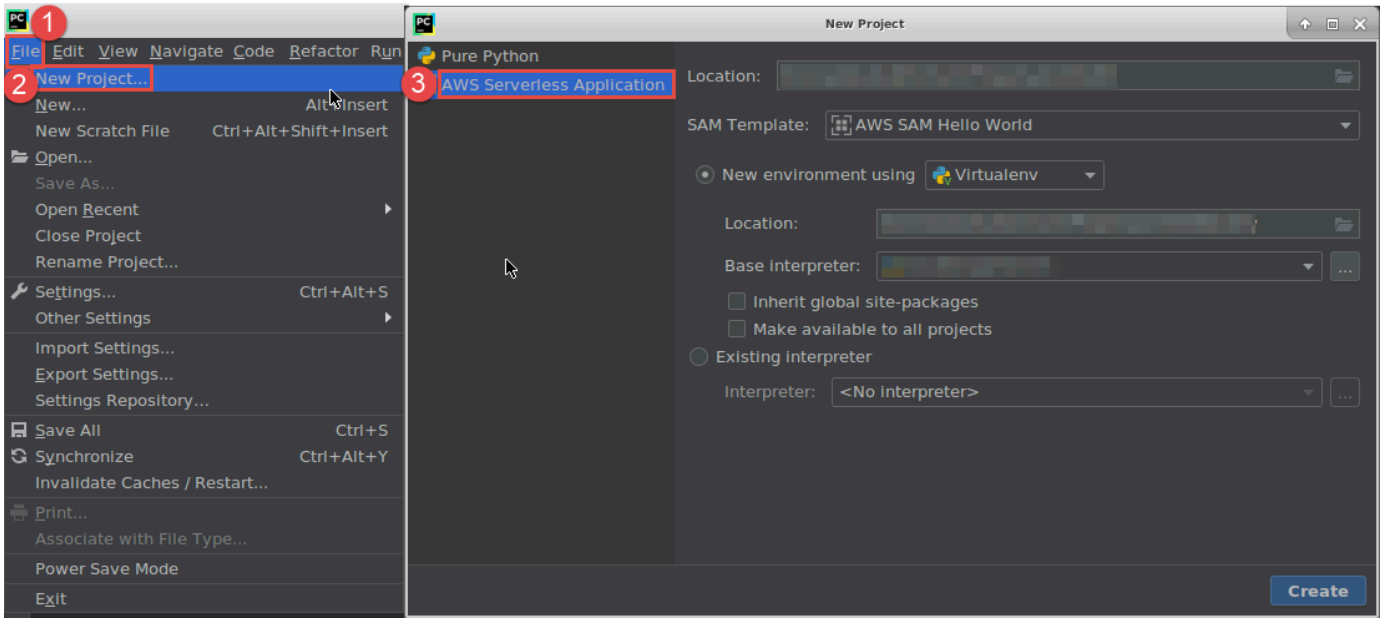
AWS Toolkit for JetBrains を使用して AWS サーバーレスアプリケーションを作成する

この手順を完了するには、最初に AWS Toolkit をインストールする必要があります。まだインストールしていない場合、最初に AWS アカウントに接続してください。その後、IntelliJ IDEA、PyCharm、WebStorm、JetBrains Rider が実行されている状態で、次の手順を実行します。

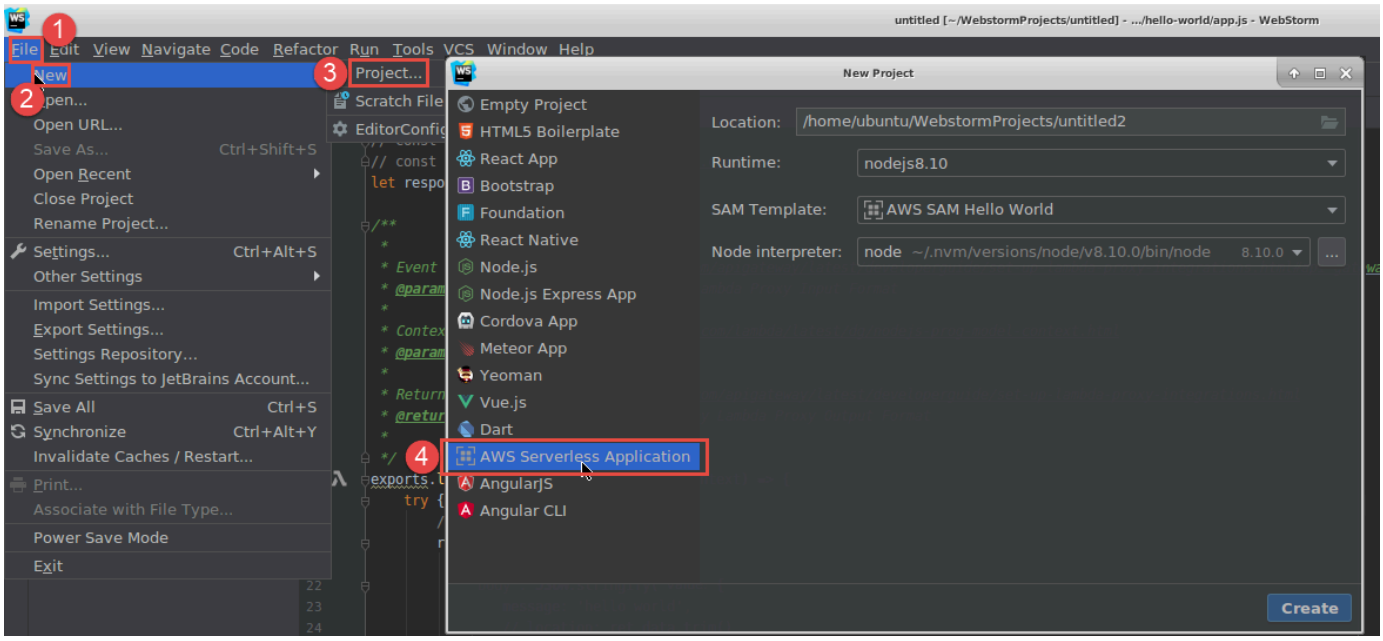
1. IntelliJ IDEA、PyCharm、WebStorm、JetBrains Rider が実行されている状態で、次のいずれかを実行します。
   - IntelliJ IDEA または WebStorm の場合は、[ファイル]、[新規]、[プロジェクト] の順に選択します。
   - PyCharm の場合は、[ファイル]、[新しいプロジェクト] の順に選択します。
   - JetBrains Rider の場合は、[ファイル]、[新規作成] の順に選択して、新しいソリューションを作成します。または、[Explorer] ツールウィンドウで既存のソリューションを右クリックし、[追加]、[New Project (新しいプロジェクト)] の順に選択します。
2. IntelliJ IDEA の場合は、[AWS]、[AWS サーバーレスアプリケーション] を選択し、その後 [次へ] を選択します。



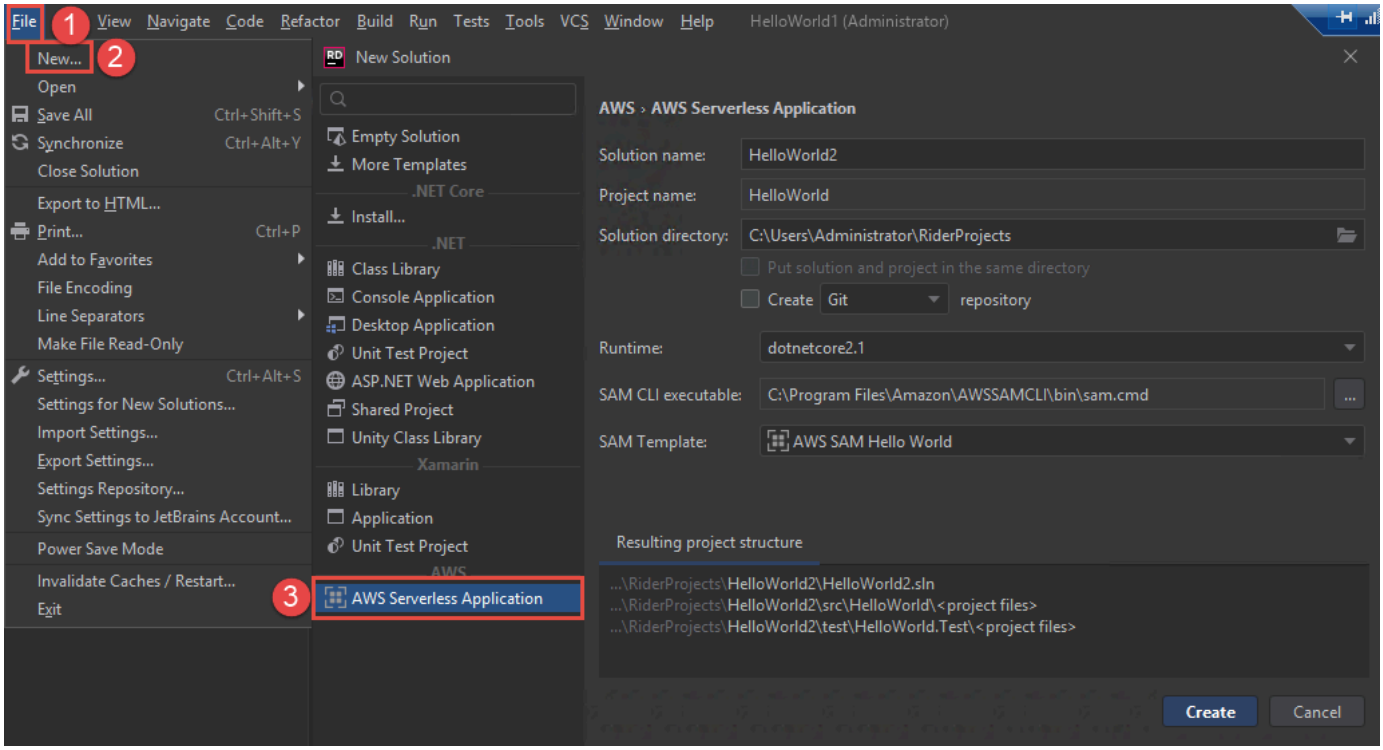
PyCharm の場合は、[AWS サーバーレスアプリケーション] を選択します。



WebStorm の場合は、[AWS サーバーレスアプリケーション] を選択します。



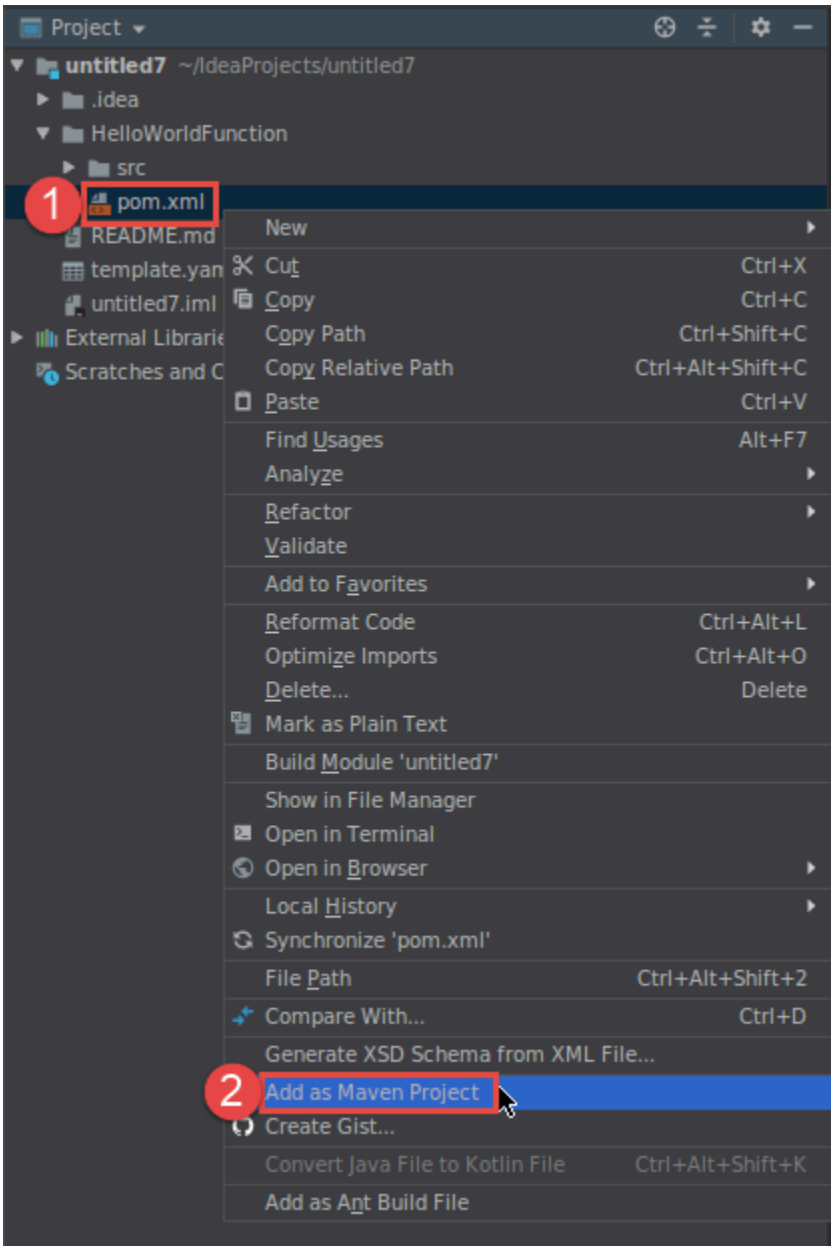
JetBrains Rider の場合は、[AWS サーバーレスアプリケーション] を選択します。



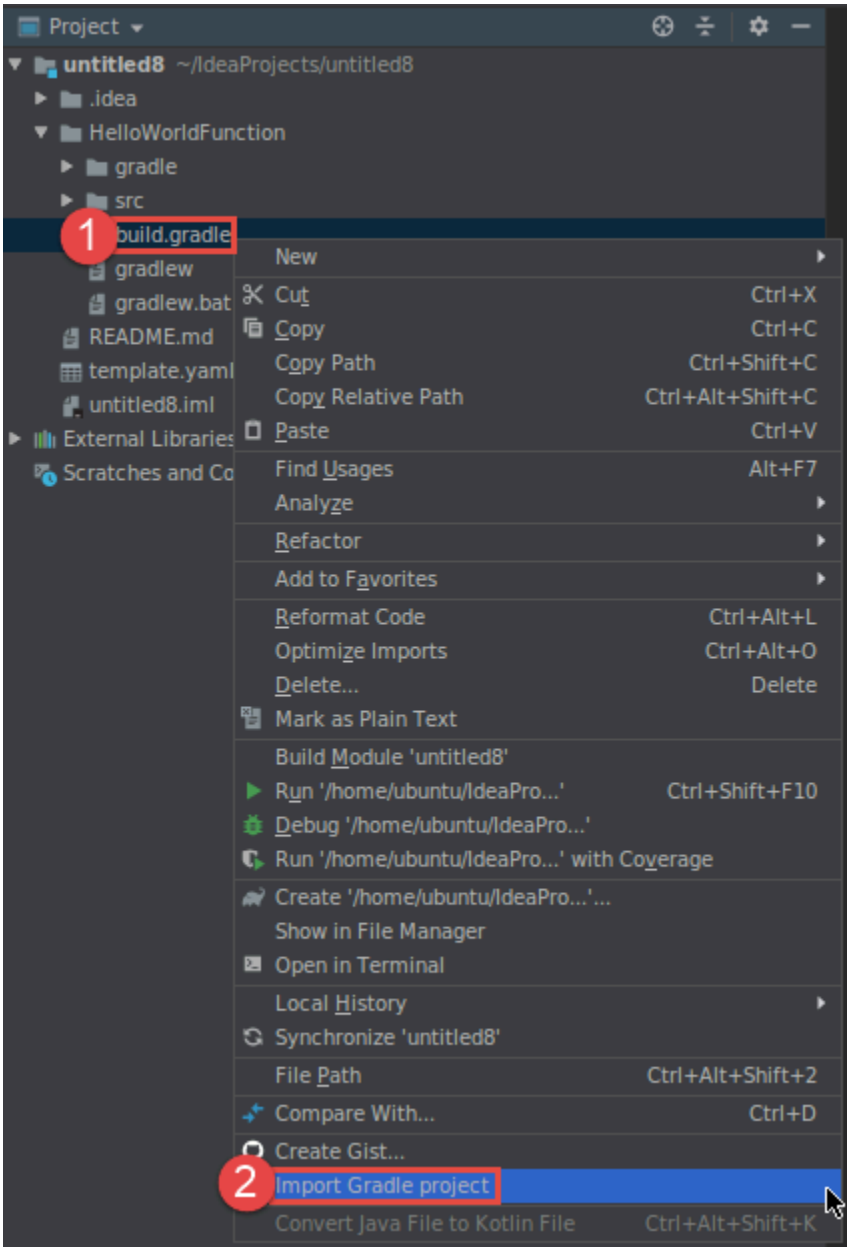
3. [新しいプロジェクト] ダイアログボックス (または JetBrains Rider の [新しいソリューション] ダイアログボックス場合) に入力し、[完了] (IntelliJ IDEA の場合) または [作成] (PyCharm、WebStorm、JetBrains Rider の場合) を選択します。AWS Toolkit for JetBrains によってプロジェクトが作成され、サーバーレスアプリケーションのコードファイルが新しいプロジェクトに追加されます。

4. IntelliJ IDEA を使用している場合は、[プロジェクト] ツールウィンドウがすでに開かれていて、サーバーレスアプリケーションのファイルを含むプロジェクトが表示されている状態で、次のいずれかの操作を行います。

   - Maven ベースのプロジェクトの場合、プロジェクトの `pom.xml` ファイルを右クリックし、[Add as Maven Project (Maven プロジェクトとして追加)] を選択します。



   - Gradle ベースのプロジェクトの場合は、プロジェクトの `build.gradle` ファイルを右クリックし、[Import Gradle project (Gradle プロジェクトのインポート)] を選択します。



[Import Module from Gradle (Gradle からモジュールをインポート)] ダイアログボックスに入力し、[OK] を選択します。

サーバーレスアプリケーションを作成した後、そのアプリケーションに含まれる AWS Lambda 関数のローカルバージョンを実行 (呼び出し) またはデバッグすることができます。

サーバーレスアプリケーションをデプロイすることもできます。デプロイ後、デプロイされたアプリケーションの一部である Lambda 関数のリモートバージョンを実行 (呼び出し) することができます。

AWS Toolkit for JetBrains の AWS SAM アプリケーションを同期する

AWS Serverless Application Model (AWS SAM) `sam sync` は、サーバーレスアプリケーションに加えた変更を自動的に識別し、それらの変更を AWS クラウド に構築してデプロイする最適な方法を選択する AWS SAM CLI コマンドデプロイプロセスです。インフラストラクチャーを変更せずにアプリケーションコードのみを変更した場合、AWS SAM 同期は AWS CloudFormation スタックを再デプロイせずにアプリケーションを更新します。

`sam sync` と AWS SAM CLI コマンドの詳細については、「AWS Serverless Application Model ユーザーガイド」の「AWS SAM CLI コマンドリファレンス」トピックを参照してください。

次のセクションでは、AWS SAM Sync の開始方法について説明します。

前提条件

AWS SAM 同期を使用する前に、次の前提条件を満たす必要があります。

- 動作している AWS SAM アプリケーションがあること。AWS SAM アプリケーションを作成する方法詳細については、本ユーザーガイドの「AWS SAM の使用について」トピックを参照してください。
- バージョン 1.78.0 以降の AWS SAM CLI をインストールしていること。AWS SAM CLI のインストールの詳細については、「AWS Serverless Application Model ユーザーガイド」の「AWS SAM CLI のインストール」トピックを参照してください。
- アプリケーションは開発環境で実行されていること。

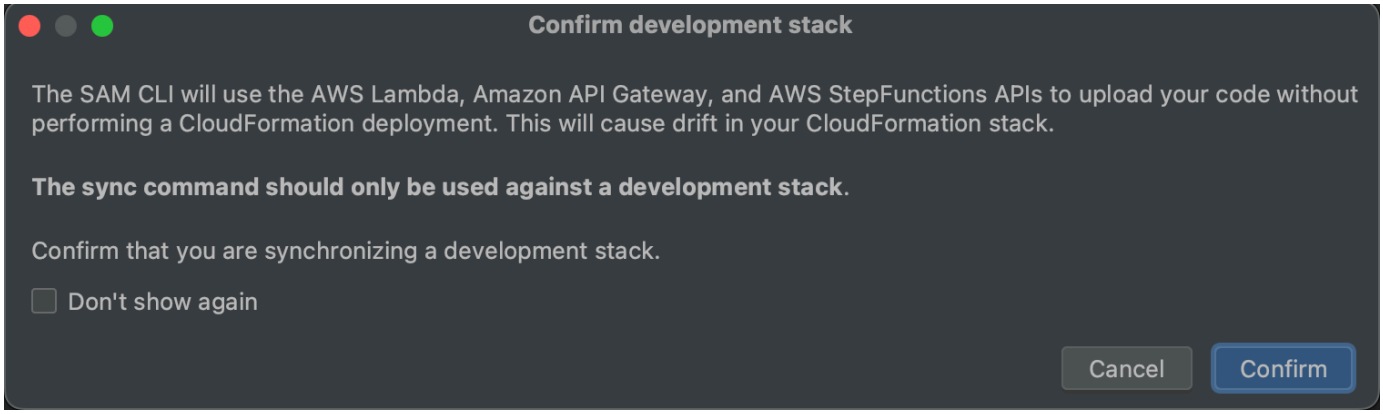
> **Note**
>
> デフォルト以外の任意のプロパティを持った AWS Lambda 関数を含むサーバーレスアプリケーションを同期してデプロイするには、デプロイする前に AWS Lambda 関数に関連付けされた AWS SAM テンプレートファイルに、オプションのプロパティを設定する必要があります。
> AWS Lambda プロパティの詳細については、GitHub の「AWS Serverless Application Model ユーザーガイド」の「AWS::Serverless::Function」セクションを参照してください。

使用開始

AWS SAM 同期を始めるには、次の手順を実行してください。

> **Note**
>
> お使いの AWS リージョンがサーバーレスアプリケーションに関連付けられたロケーションに設定されていることを確認してください。
> AWS Toolkit for JetBrains から AWS リージョンを変更する方法の詳細については、このユーザーガイドの「AWS リージョン間の切り替え」トピックを参照してください。

1. [プロジェクト] ツールウィンドウのサーバーレスアプリケーションプロジェクトで、`template.yaml` ファイルのコンテキストメニューを開きます (右クリック)。
2. `template.yaml` コンテキストメニューから、[サーバーレスアプリケーションの同期 (以前は「Deploy」)]を選択し、[開発スタックの確認] ダイアログを開きます。
3. 開発スタックから操作していることを確認し、[サーバーレスアプリケーションの同期] ダイアログを開きます。



4. [サーバーレスアプリケーションの同期] ダイアログの手順を完了したら、[同期] を選択して AWS SAM 同期プロセスを開始します。[サーバーレスアプリケーションの同期] ダイアログの詳細については、以下の「the section called “ サーバーレスアプリケーションの同期ダイアログ ”」セクションを参照してください。
5. 同期処理中、AWS Toolkit for JetBrains の [ウィンドウの実行] はデプロイ ステータスで更新されます。
6. 同期が正常に実行された後、AWS CloudFormationスタックの名前がAWS エクスプローラーに追加されます。

   同期に失敗した場合、トラブルシューティングの詳細は JetBrains の[ウィンドウの実行] または AWS CloudFormation の[イベントログ] で確認できます。AWS CloudFormation イベントログの表示に関する詳細については、本ユーザーガイドの「スタックのイベント ログの表示」トピックを参照してください。

サーバーレスアプリケーションの同期ダイアログ

[サーバーレスアプリケーションの同期] ダイアログは AWS SAM 同期プロセスを支援します。次のセクションは、ダイアログの各コンポーネントの説明と詳細を示します。

スタックの作成または更新

[必須]: 新しいデプロイスタックを作成するには、表示されるフィールドに名前を入力し、サーバーレスアプリケーションのデプロイを行うために AWS CloudFormation スタックを作成し、設定します。

または、既存の AWS CloudFormation スタックにデプロイするには、お持ちの AWS アカウントに関連付けされたスタックの自動入力リストからスタック名を選択します。

テンプレートパラメーター

[オプション]: プロジェクトの `template.yaml` ファイルから検出されたパラメータのリストが入力されます。パラメータ値を指定するには、[値] の列に表示されるテキスト フィールドに新しいパラメータ値を入力します。

S3 バケット

[必須]: AWS CloudFormation テンプレートを保存するために既存の Amazon Simple Storage Service (Amazon S3) バケットを選択するには、リストから選択してください。

ストレージ用に新しい Amazon S3 バケットを作成して使用するには、[作成] を選択してプロンプトに従います。

ECR リポジトリ

[必須。Image パッケージタイプを使用する場合にのみ表示されます:] サーバーレスアプリケーションのデプロイ用に既存の Amazon Elastic Container Registry (Amazon ECR) リポジトリ URI を選択します。

AWS Lambda パッケージタイプ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」セクションを参照してください。

CloudFormation 機能

[必須:] スタックの作成時に AWS CloudFormation が使用できる機能を選択します。

タグ

[オプション:] 表示されるテキスト フィールドに任意のタグを入力してパラメータにタグ付けします。

コンテナ内で関数を構築する

[オプション。Docker が必須:] このオプションを選択すると、デプロイ前にローカルの Docker コンテナ内にサーバーレスアプリケーション機能が構築されます。このオプションは、機能がネイティブにコンパイルされた依存関係やプログラムを持つパッケージに依存する場合に便利です。

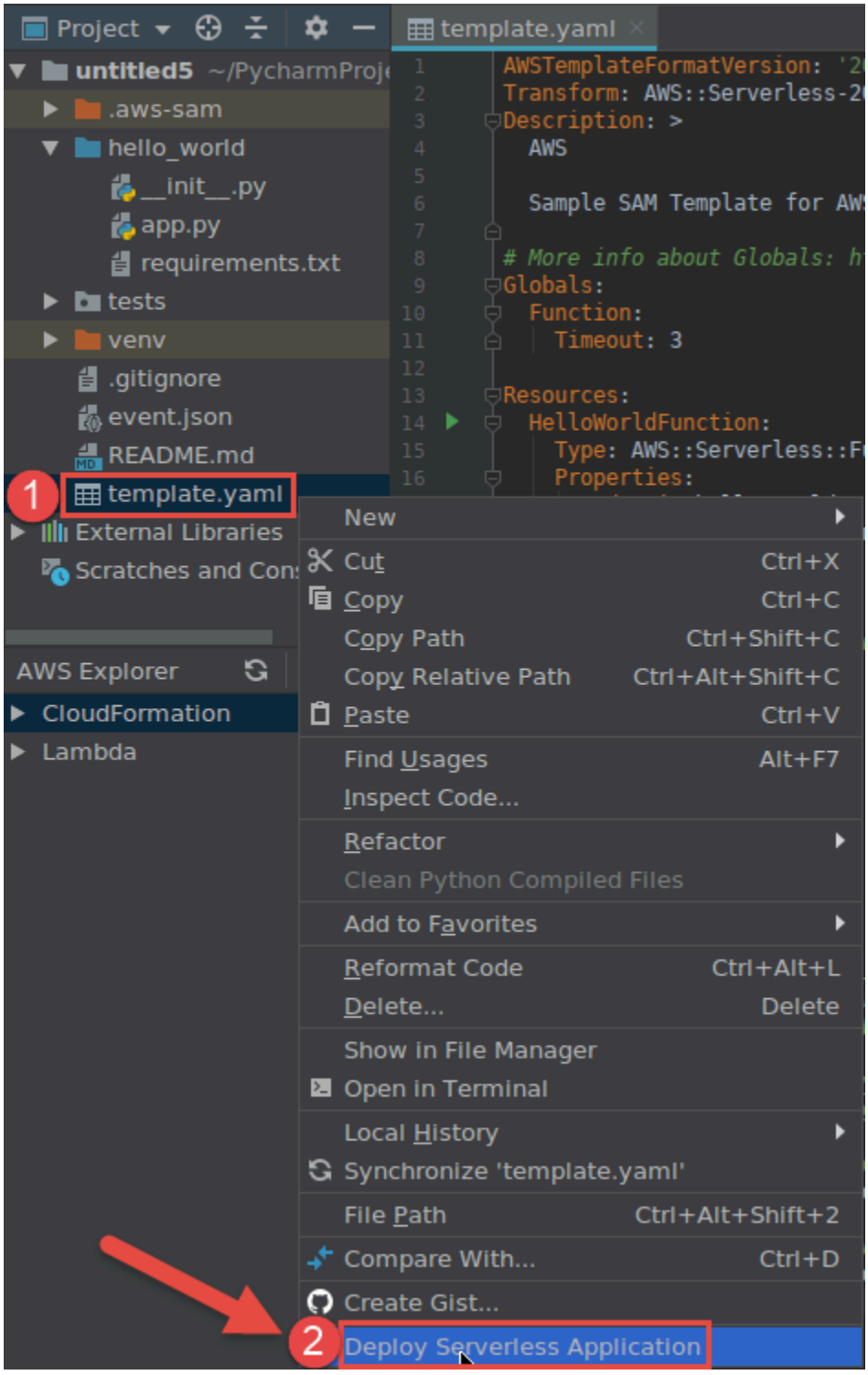
詳細については、「AWS Serverless Application Model デベロッパーガイド」の「アプリケーション構築」トピックを参照してください。

AWS Toolkit for JetBrains を使用して AWS サーバーレスアプリケーション設定を変更 (更新) する

まだデプロイしていない場合、変更する AWS サーバーレスアプリケーションを最初にデプロイする必要があります。

> **Note**
>
> AWS Lambda 関数を含むサーバーレスアプリケーションをデプロイし、デフォルト以外のプロパティまたはオプションのプロパティを使用してその関数をデプロイするには、最初にその関数の対応する AWS SAM テンプレートファイル (プロジェクト内の `template.yaml` という名前のファイルなど) のこれらのプロパティを設定する必要があります。利用可能なプロパティのリストについては、GitHub の awslabs/serverless-application-model リポジトリで「AWS::Serverless::Function」を参照してください。

1. [プロジェクト] ツールウィンドウがすでに開かれていて、サーバーレスアプリケーションのファイルを含むプロジェクトが表示されている状態で、プロジェクトの `template.yaml` ファイルを開きます。ファイルの内容を変更して新しい設定を反映し、ファイルを保存して閉じます。
2. サーバーレスアプリケーションをデプロイする別の AWS リージョンに切り替える必要がある場合、今すぐ行います。
3. プロジェクトの `template.yaml` ファイルを右クリックし、[Deploy Serverless Application (サーバーレスアプリケーションのデプロイ)] を選択します。



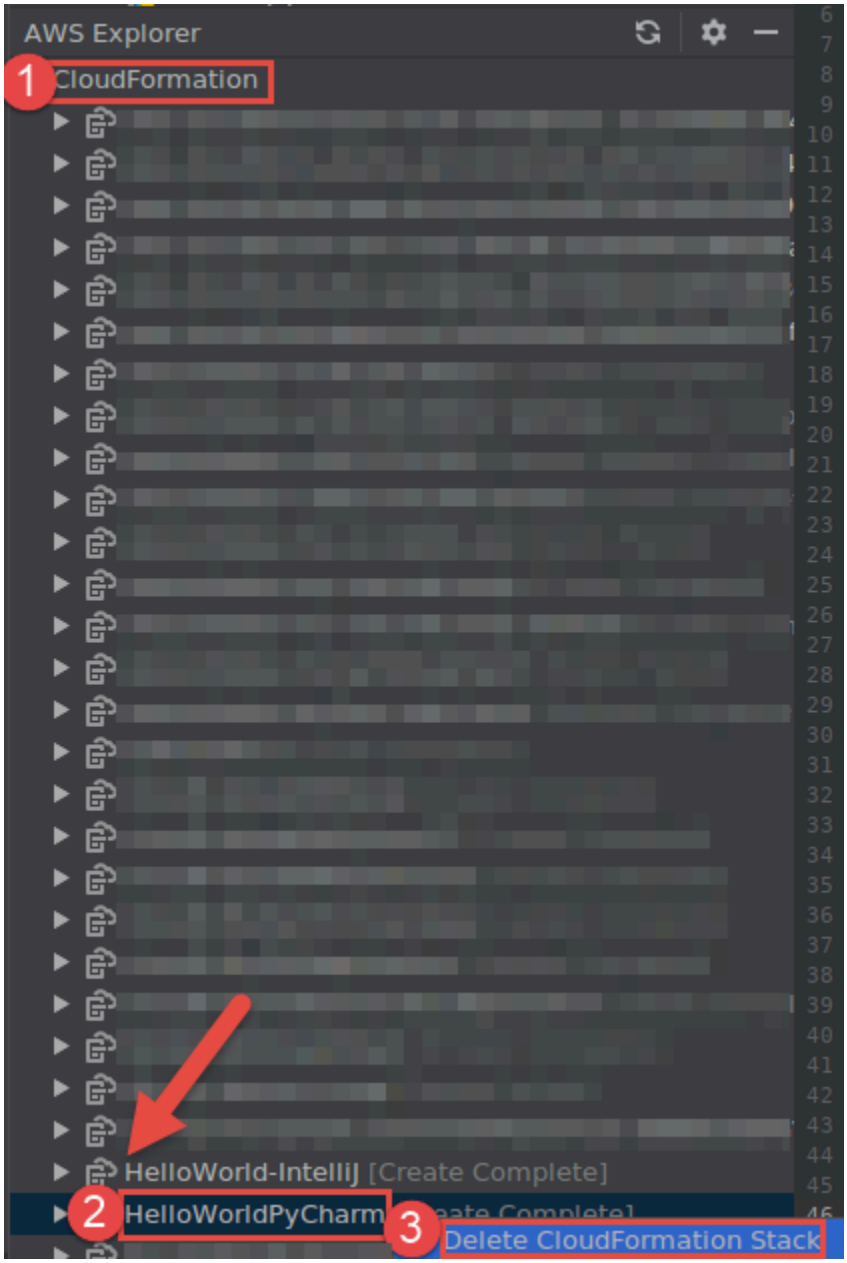
4. [Deploy Serverless Application (サーバーレスアプリケーションのデプロイ)] ダイアログボックスに入力し、[デプロイ] を選択します。AWS Toolkit for JetBrains は、デプロイに対応する AWS CloudFormation スタックを更新します。

デプロイが失敗した場合、スタックのイベント ログを確認して原因を特定することができます。

AWS Toolkit for JetBrains を使用して AWS サーバーレスアプリケーションを削除する

AWS サーバーレスアプリケーションを削除する前に、デプロイする必要があります。

1. AWS Explorer を開きます (まだ開いていない場合)。サーバーレスアプリケーションを含む別の AWS リージョンに切り替える必要がある場合、今すぐ行います。
2. [CloudFormation] を展開します。
3. 削除するサーバーレスアプリケーションを含む AWS CloudFormation スタック名を右クリックし、[Delete CloudFormation Stack (CloudFormation スタックの削除)] を選択します。



4. スタック名を入力して削除を確認し、[OK] を選択します。スタックの削除が成功すると、AWS Toolkit for JetBrains は、AWS エクスプローラーの [CloudFormation] リストからスタック名を削除します。スタックの削除が失敗した場合、スタックのイベント ログを確認して原因を特定することができます。

AWS Toolkit for JetBrains から Amazon Simple Queue Service を操作する

次の項目では、AWS Toolkit for JetBrains から Amazon Simple Queue Service を操作する方法について説明します。

トピック

Amazon Simple Queue Service キューの操作

次の項目では、AWS Toolkit for JetBrains を使用して Amazon Simple Queue Service のキューとメッセージを操作する方法について説明します。

AWS Toolkit for JetBrains で Amazon SQS を使用して送信可能なメッセージには、標準と FIFO (先入れ先出し) の 2 種類があります。

Amazon SQS キュー を作成する方法

1. AWS Toolkit for JetBrains から、AWS エクスプローラーを展開して AWS サービスを表示します。
2. AWS エクスプローラーから [Amazon SQS] サービスのコンテキスト メニュー(右クリック) を開き、[キューの作成] を選択します。
3. キュー名を指定してキュータイプを選択します。

> **Note**
>
> キュータイプの詳細については、「Amazon Simple Queue Service デベロッパーガイド」の「Amazon SQS 標準キュー」と「Amazon SQS FIFO (先入れ先出し) キュー」トピックを参照してください。

4. [Create] (作成) を選択します。

Amazon SQS メッセージを表示する方法

1. AWS Toolkit for JetBrains から、AWS エクスプローラーを展開して AWS サービスを表示します。
2. AWS エクスプローラーから、[Amazon SQS] サービスを展開して既存キューのリストを表示します。
3. 表示するキューを右クリックし、[メッセージの表示] を選択します。
4. [メッセージの表示] を選択し、このキューのメッセージを表示します。

Amazon SQS キューのプロパティを編集する方法

1. AWS Toolkit for JetBrains から、AWS エクスプローラーを展開して AWS サービスを表示します。
2. AWS エクスプローラーから、[Amazon SQS] サービスを展開して既存キューのリストを表示します。
3. 編集するキューを右クリックして、[キュープロパティの編集] を選択します。
4. 開いた [キュープロパティの編集] ダイアログボックスで、キューのプロパティを確認して変更します。Amazon SQS プロパティの詳細については、「Amazon Simple Queue Service のデベロッパーガイド」の「キューパラメータの設定 (コンソール)」を参照してください。

標準メッセージを送信する方法

1. AWS Toolkit for JetBrains から、AWS エクスプローラーを展開して AWS サービスを表示します。
2. AWS エクスプローラーから、[Amazon SQS] サービスを展開して既存キューのリストを表示します。
3. メッセージを送信するキューを右クリックし、[メッセージの送信] を選択します。
4. メッセージを入力して [送信] を選択します。メッセージを送信した後、メッセージ ID を含む確認メッセージが表示されます。

FIFO メッセージを送信する方法

1. AWS Toolkit for JetBrains から、AWS エクスプローラーを展開して AWS サービスを表示します。

2. AWS エクスプローラーから、[Amazon SQS] サービスを展開して既存キューのリストを表示します。
3. メッセージを送信するキューを右クリックし、[メッセージの送信] を選択します。
4. メッセージ、グループ ID、重複排除 ID (オプション) を入力します。

> **Note**
>
> 重複排除 ID を指定しない場合、重複排除 ID が 1 つ生成されます。

5. [Send] (送信) を選択します。メッセージを送信した後、メッセージ ID を含む確認メッセージが表示されます。

Amazon SQS キューを削除する方法

1. キューを削除する前に、空であることを確認してください。詳細については、「Amazon Simple Queue Service のデベロッパーガイド」の「キューが空であることを確認する」を参照してください。
2. AWS Toolkit for JetBrains から、AWS エクスプローラーを展開して AWS サービスを表示します。
3. AWS エクスプローラーから、[Amazon SQS] サービスを展開して既存キューのリストを表示します。
4. [Amazon SQS] を右クリックし、[キューの削除] を選択します。
5. キューを削除することを確認し、削除ダイアログボックスで [OK] を選択します。

AWS Toolkit for JetBrains で AWS Lambda を使用して Amazon SQS を操作する

次の手順では、AWS Toolkit for JetBrains で Amazon SQS キューを Lambda トリガーとして設定する方法の詳細を説明します。

Amazon SQS キューを Lambda のトリガーとして設定する方法

1. AWS Toolkit for JetBrains から、AWS エクスプローラーを展開して AWS サービスを表示します。
2. AWS エクスプローラーから、[Amazon SQS] サービスを展開して既存キューのリストを表示します。

3. 使用するキューを右クリックし、[Lambda トリガーの設定] を選択します。
4. ダイアログボックスのドロップダウンメニューから、トリガーする Lambda 関数を選択します。
5. [Configure] (設定) を選択します。
6. Lambda 関数に Amazon SQS の実行に必要な IAM 許可がない場合、ツールキットが最小限のポリシーを生成したものを、Lambda 関数の IAM ロールに追加できます。

   [Add policy] を選択します。

キューを設定した後、適用された変更に関するステータスメッセージ (該当するエラーメッセージを含む) が表示されます。

AWS Toolkit for JetBrains で Amazon SQS を使用して Amazon SNS を操作する

次の手順では、AWS Toolkit for JetBrains を使用して標準 Amazon SQS キューを Amazon SNS トピックにサブスクライブする方法について詳しく説明します。

> **Note**
>
> FIFO Amazon SQS キューを Amazon SNS トピックにサブスクライブすることはできません。

標準 Amazon SQS キューを Amazon SNS トピックにサブスクライブする方法

1. AWS Toolkit for JetBrains から、AWS エクスプローラーを展開して AWS サービスを表示します。
2. AWS エクスプローラーから、[Amazon SQS] サービスを展開して既存キューのリストを表示します。
3. 操作するキューを右クリックして[SNS トピックのサブスクライブ] を選択します。
4. ダイアログボックスのドロップダウンメニューから、Amazon SNS トピックを選択し、[サブスクライブ] を選択します。

リソースの使用

AWS Explorer にデフォルトで表示されている AWS サービスへのアクセスに加えて、[Resources] (リソース) を選択して、数百ものリソースから選択してインターフェイスに追加することもできます。AWS では、リソースはユーザーが操作できるエンティティです。追加できるリソースには、Amazon AppFlow、Amazon Kinesis Data Streams、AWS IAM ロール、Amazon VPC、および Amazon CloudFront ディストリビューションなどがあります。

選択したら、リソース に進み、リソースタイプを展開して、そのタイプで使用可能なリソースを一覧表示します。例えば、`AWS::Lambda::Function` リソースタイプを選択すると、さまざまな関数、そのプロパティ、および属性を定義するリソースにアクセスできます。

リソースタイプを [Resources] (リソース) に追加すると、次の方法でリソースタイプとそのリソースを操作できます。

- このリソースタイプについて、現在の AWS リージョンで使用可能な既存のリソースを一覧表示します。
- リソースを記述する JSON ファイルの読み取り専用バージョンを表示します。
- リソースのリソース識別子をコピーします。
- リソースタイプの目的およびリソースをモデリングするためのスキーマ (JSON および YAML 形式) について説明する AWS のドキュメントを表示します。
- スキーマに準拠する JSON 形式のテンプレートを編集して保存して、新しいリソースを作成します。*
- 既存のリソースを更新または削除します。*

> **Important**
>
> * 現在の AWS Toolkit for JetBrains では、リソースの作成、編集、削除のオプションは実験的な機能です。実験的な機能は引き続きテストおよび更新されるため、ユーザビリティに問題がある可能性があります。実験的な特徴は、通知なしに AWS Toolkit for JetBrains から削除される可能性があります。
> リソースに実験的機能を使用できるようにするには、JetBrains IDE の [設定] ペインを開き、[ツール] を展開してから、[AWS]、[実験的機能] を選択します。[JSON リソースの変更] を選択して、リソースを作成、更新、削除できるようにします。
> 詳細については、「実験的な機能の使用」を参照してください。

リソースにアクセスするための IAM アクセス許可

AWS サービスに関連付けられたリソースにアクセスするには、特定の AWS Identity and Access Management アクセス許可が必要です。例えば、IAM エンティティ (ユーザーまたはロールなど) は、`AWS::Lambda::Function` リソースにアクセスするために Lambda アクセス許可を必要とします。

IAM エンティティには、サービスリソースへのアクセス許可だけでなく、自身に代わって AWS Toolkit for JetBrains が AWS クラウドコントロール API オペレーションを呼び出すためのアクセス許可も必要になります。Cloud Control API オペレーションでは、IAM ユーザーまたはロールがリモートリソースにアクセスして更新できます。

アクセス許可を付与する場合、ツールキットインターフェイスを使用して API オペレーションを呼び出す IAM エンティティに AWS マネージドポリシー (PowerUserAccess) をアタッチするのが最も簡単な方法です。このマネージドポリシーでは、API オペレーションの呼び出しなど、アプリケーション開発タスクを実行するために一定の範囲のアクセス許可が付与されます。

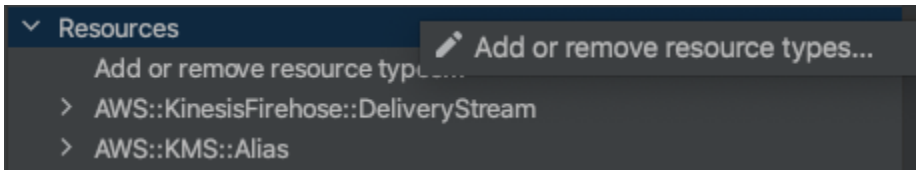
リモートリソースで使用可能な API オペレーションを定義する特定のアクセス許可については、「AWS クラウドコントロール API ユーザーガイド」を参照してください。

既存のリソースの追加と操作

1. [AWS エクスプローラー] で [リソース] を右クリックし、[リソースの追加または削除] を選択します。

   [設定] ペインの [その他の エクスプローラーリソース] には、選択可能なリソースタイプのリストが表示されます。

   > **Note**
   >
   > リソースタイプのリストは、[リソース] の下にある [リソースを追加または削除] ノードをダブルクリックして表示することもできます。



2. [その他の エクスプローラーリソース] で、[AWS エクスプローラー] に追加するリソースタイプを選択し、[戻る] を押すか [OK] を選択して確定します。

   選択したリソースタイプは、[リソース] に表示されます

   > **Note**
   >
   > リソースタイプを [AWS エクスプローラー] にすでに追加済みの場合は、そのタイプのチェックボックスをオフにすると、[OK] をクリックした後 [リソース] の下のリストに表示されなくなります。現在選択されているリソースタイプのみが、[AWS エクスプローラー] に表示されます。

3. リソースタイプに既に存在するリソースを表示するには、そのタイプのエントリを展開します。

   使用可能なリソースのリストが、リソースタイプの下に表示されます。

4. 特定のリソースを操作するには、リソースの名前を右クリックし、次のいずれかを選択します。

   - [リソースの表示]: リソースを記述する JSON 形式のテンプレートの読み取り専用バージョンを表示します。

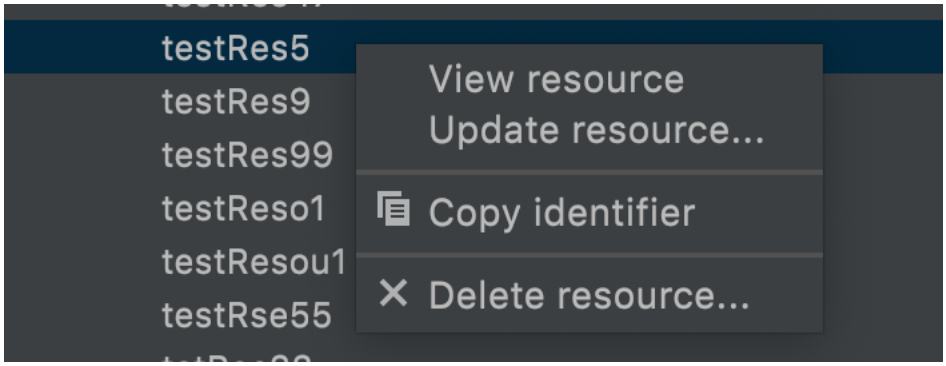
     テンプレートが表示され、必要となる ??? が有効になっている場合は、[編集] を選択することで変更できます。

     > **Note**
     >
     > リソースをダブルクリックして表示することもできます。

   - [識別子をコピー]: 特定のリソースの識別子をクリップボードにコピーします。(例えば、`AWS::DynamoDB::Table` リソースは `TableName` プロパティを使用して識別できます。)
   - [リソースの更新]: JetBrains エディターでリソースの JSON 形式のテンプレートを編集します。詳細については、「リソースの作成と編集」を参照してください。
   - [リソースの削除]: 表示されたダイアログボックスで削除を確認して、リソースを削除します。(リソースの削除は、AWS Toolkit for JetBrains のこのバージョンでは現在 ??? です。)

     > **Warning**
     >
     > リソースを削除すると、そのリソースを使用する AWS CloudFormation スタックは更新に失敗します。この更新の失敗を修正するには、リソースを再作成するか、スタッ

クの AWS CloudFormation テンプレートテンプレートにリソースへのリファレンスを削除する必要があります。詳細情報は、この ナレッジセンターの記事 を参照してください。



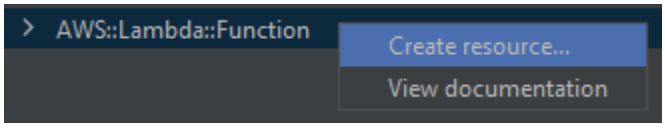
リソースの作成と編集

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

新しいリソースを作成するには、[リソース] リストにリソースタイプを追加し、リソース、そのプロパティ、および属性を定義する JSON 形式のテンプレートを編集します。

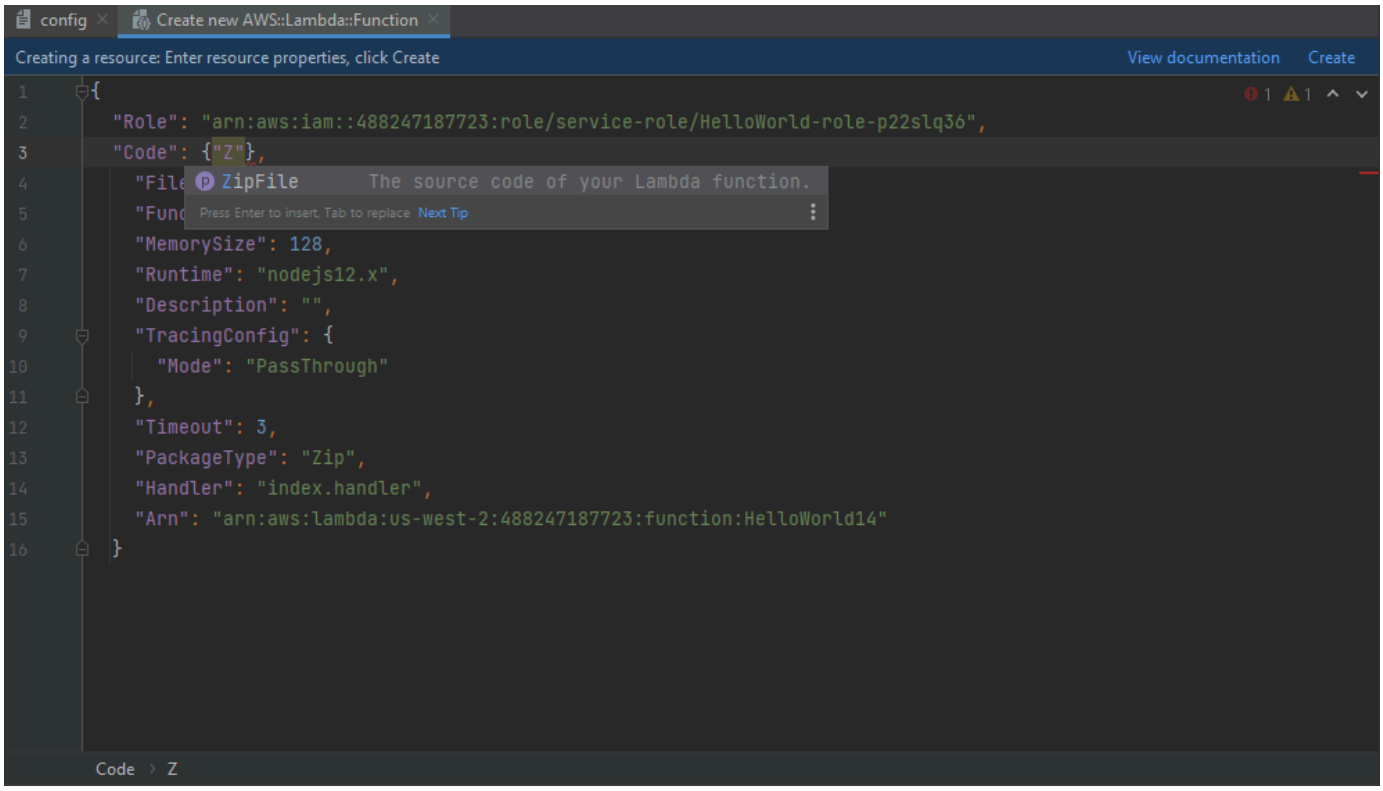
例えば、`AWS::SageMaker::UserProfile` リソースタイプに属しているリソースは、Amazon SageMaker Studio のユーザープロファイルを作成するテンプレートで定義されます。このユーザープロファイルリソースを定義するテンプレートは、`AWS::SageMaker::UserProfile` のリソースタイプスキーマに準拠している必要があります。例えば、プロパティが見つからないか正しくないためにテンプレートがスキーマに準拠していない場合は、リソースを作成または更新することはできません。

1. [リソース] を右クリックし、[リソースの追加または削除] を選択して、作成するリソースのリソースタイプを追加します。
2. リソースタイプが [リソース] の下に追加されたら、そのリソース名を右クリックして [リソースを作成] を選択します。[ドキュメントの表示] を選択することで、リソースをモデル化する方法に関する情報にアクセスすることもできます。



3. エディタで、リソーステンプレートを構成するプロパティの定義を開始します。オートコンプリート機能では、テンプレートのスキーマに適合するプロパティ名が提案されます。テンプレートが JSON 構文に完全に準拠すると、エラー数が緑色のチェックマークに置き換えられます。スキーマの詳細な情報については、[ドキュメントの表示] を選択してください。

> **Note**
>
> テンプレートは基本的な JSON 構文に準拠しているだけでなく、リソースタイプをモデル化するスキーマにも準拠している必要があります。リモートリソースを作成または更新するときに、テンプレートがスキーマモデルに対して検証されます。



4. リソースの宣言が完了したら、[作成] を選択してテンプレートを検証し、リモート AWS Cloud にリソースを保存します。(既存のリソースを変更する場合は [更新] を選択します。)

テンプレートがスキーマに従ってリソースを定義している場合は、リソースが作成されたことを確認するメッセージが表示されます。(リソースが既に存在する場合は、リソースが更新されたことを確認するメッセージが表示されます。)

リソースが作成されると、リソースタイプの見出しの下のリストに追加されます。

5. ファイルにエラーが含まれている場合は、リソースを作成または更新できなかったことを示すメッセージが表示されます。[イベントログ] を開いて、修正する必要があるテンプレート要素を特定します。

AWS Toolkit for JetBrains のユーザーインターフェイスリファレンス

AWS Toolkit for JetBrains ユーザーインターフェイスの操作に関するヘルプが必要な場合、次のトピックを参照してください。

トピック

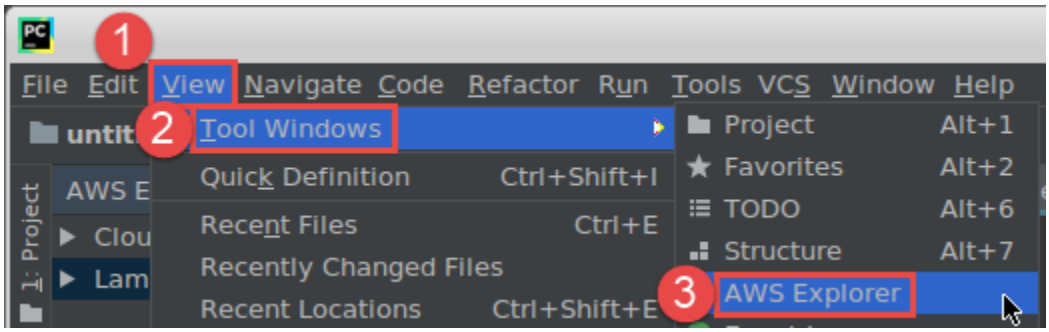
AWS Explorer

AWS エクスプローラーを使用すると、AWS Toolkit for JetBrains の複数の機能に簡単にアクセスできます。これには、ツールキットから AWS アカウントへの接続の管理、AWS リージョンの切り替え、アカウント内の AWS Lambda 機能や AWS CloudFormation スタックの操作などが含まれます。

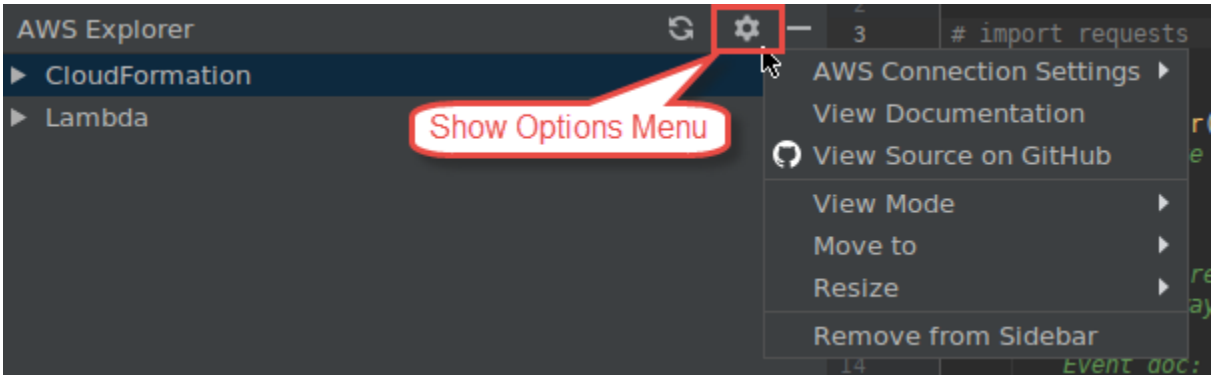
AWS エクスプローラーを開くには、AWS Toolkit for JetBrains がインストールされ、IntelliJ IDEA、PyCharm、WebStorm、または JetBrains Rider が実行されている状態で、次のいずれかを実行します。

- ツールウィンドウバーで、[AWS Explorer] を選択します。

- メインメニューで、[表示]、[ツールウィンドウ,]、[AWS Explorer] の順に選択します。



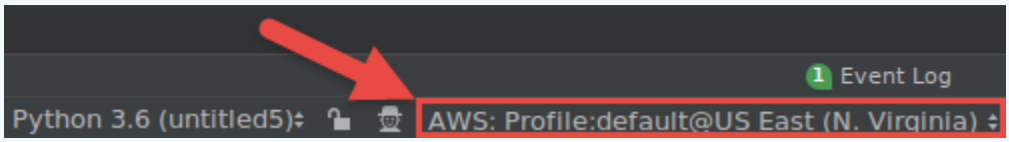
AWS エクスプローラーで、以下のオプションの設定アイコン ([オプションメニューを表示]) を選択します。



AWS 接続の設定

以下のオプションが含まれます。

- AWS リージョンリスト – AWS Toolkit for JetBrains は選択したリージョンを使用します。別のリージョンを使用するようにツールキットを切り替えるには、表示されている別のリージョンを選択します。
- 最近の認証情報リスト – AWS Toolkit for JetBrains から AWS アカウントに最近行われた接続を一覧表示します。ツールキットは選択した接続を使用します。別の最近使用した接続を使用するようにツールキットを切り替えるには、その接続の名前を選択します。
- すべての認証情報 – AWS Toolkit for JetBrains から AWS アカウントに作成できるすべての利用可能な接続が一覧表示されます。ツールキットは選択した接続を使用します。別の接続を使用するようにツールキットを切り替えるには、その接続の名前を選択します。他の接続タスクを実行するには、[AWS 認証情報ファイルの編集] を選択します。

> **Note**
>
> ステータスバーの [AWS 接続設定] エリアに、AWS アカウント接続と AWS Toolkit for JetBrains が現在使用しているリージョンが表示されます。
>
> 
>
>
> このエリアを選択すると、[オプションメニューを表示] と同じ [AWS 接続設定] オプションが表示されます。

ドキュメントの表示

「AWS Toolkit for JetBrains ユーザーガイド」(本ガイド) に移動します。

GitHub でソースを表示

GitHub ウェブサイトの aws/aws-toolkit-jetbrains リポジトリに移動します。

表示モード

AWS Explorer のツールウィンドウを調整して、エディタやその他のツールウィンドウで作業するときにすばやくアクセスしてスペースを節約できるようにします。

IntelliJ IDEA ビューモードについては、IntelliJ IDEA ヘルプウェブサイトに掲載されている「ツールウィンドウのビューモード」を参照してください。

PyCharm ビューモードについては、PyCharm ヘルプウェブサイトに掲載されている「ツールウィンドウのビューモード」を参照してください。

WebStorm ビューモードについては、WebStorm ヘルプウェブサイトに掲載されている「ツールウィンドウのビューモード」を参照してください。

JetBrains Rider ビューモードについては、JetBrains Rider ヘルプウェブサイトに掲載されている「ツールウィンドウのビューモード」を参照してください。

移動先

[AWS エクスプローラー] ツールウィンドウを、IntelliJ IDEA、PyCharm、WebStorm、または JetBrains Rider の別の場所へと移動します。

サイズ変更

AWS Explorer ツールウィンドウのサイズを変更します。

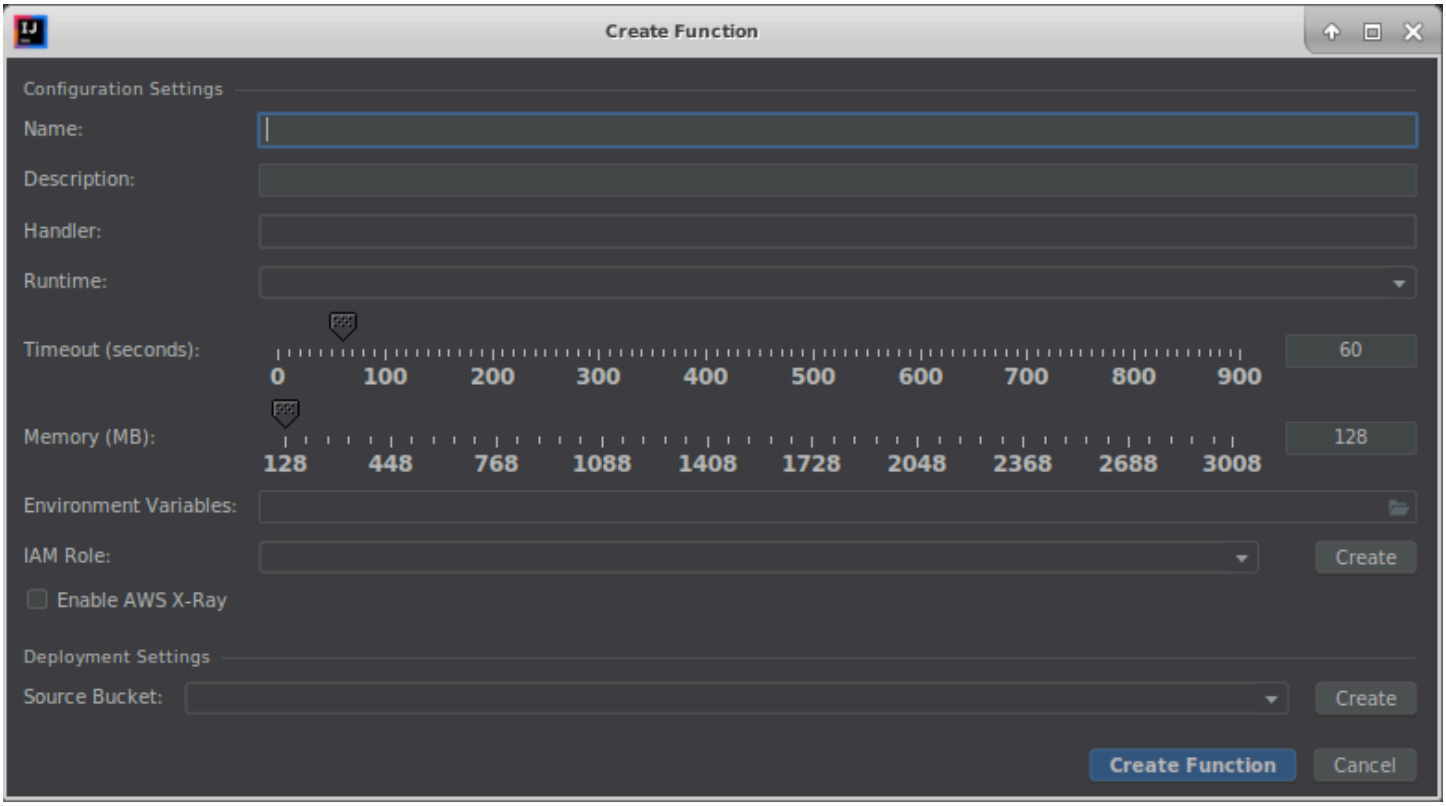
サイドバーから削除

ツールウィンドウバーから [AWS エクスプローラー] ツールウィンドウを削除します。再度表示するには、メインメニューバーで、[表示]、[ツールウィンドウ]、[AWS Explorer] の順に選択します。

AWS エクスプローラーを使用して Lambda 関数を操作し、AWS アカウントの AWS CloudFormation を使用することもできます。

[関数の作成] ダイアログボックス

スタンドアロンの AWS Lambda 関数を作成すると、AWS Toolkit for JetBrains に [関数を作成] ダイアログボックスが表示されます。



[関数の作成] ダイアログボックスには、次の項目があります。

名前

(必須) 関数の名前。A から Z までの大文字、a から z までの小文字、0 から 9 までの数字、ハイフン (-)、アンダースコア (_) のみを使用できます。名前は 64 文字未満にする必要があります。

説明

(オプション) 関数についてのわかりやすい説明。

Handler

(必須) Java、Python、Node.js、C# に対応する Lambda 関数ハンドラーの ID。

ランタイム

(必須) 使用する Lambda ランタイムの ID。

タイムアウト (秒)

(必須) Lambda が関数を停止するまでに許可する実行時間。最大で 900 秒 (15 分) を指定します。

メモリ (MB)

(必須) 関数が実行する際に使用可能なメモリ量。メモリの量を 128 MB ~ 3,008 MB の範囲 (64 MB 単位) で指定します。

環境可変

(オプション) キー値ペアとして指定された Lambda 関数が使用する環境変数。環境変数を追加、変更、または削除するには、フォルダアイコンを選択し、画面の指示に従います。

IAM ロール

(必須) 接続された AWS アカウントの Lambda 実行ロールで Lambda が関数に使用するものを選択します。アカウントに実行ロールを作成し、代わりに Lambda がそのロールを使用するには、[作成] を選択して画面の指示に従ってください。

AWS X-Ray の有効化

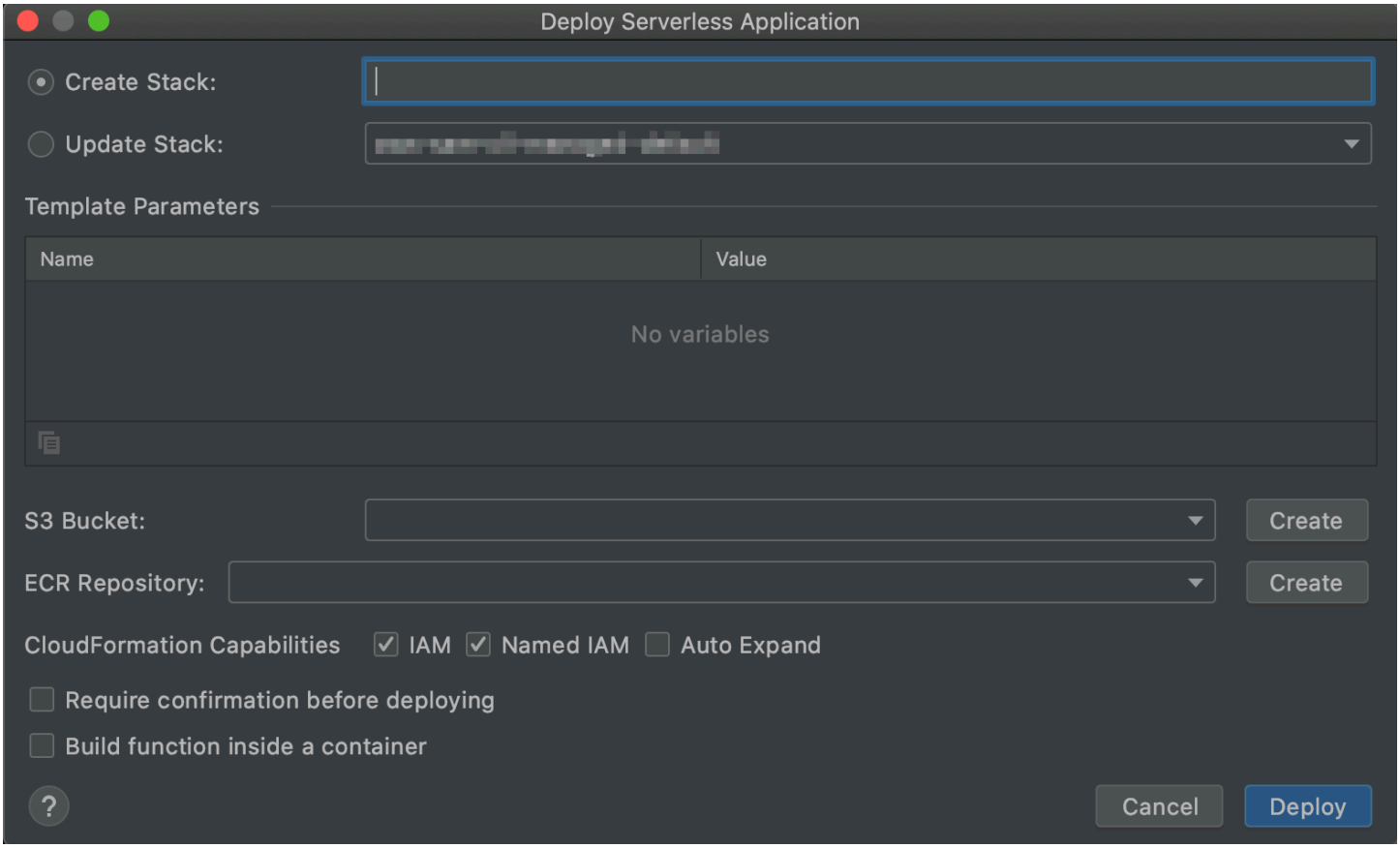
(オプション) 選択した場合、Lambda は AWS X-Ray を有効にして関数のパフォーマンス問題を検出、分析、最適化します。X-Ray は、Lambda や関数を構成するアップストリームまたはダウンストリームサービスからメタデータを収集します。X-Ray はこのメタデータを使用して、関数パフォーマンスに影響を及ぼすパフォーマンスボトルネック、レイテンシースパイク、およびその他問題を示す詳細なサービスグラフを生成します。

ソースバケット

(必須) AWS Serverless Application Model (AWS SAM) コマンドラインインターフェイス (CLI) の接続された AWS アカウント内にある、利用可能な Amazon Simple Storage Service (Amazon S3)バケットから使用するものを選択し、Lambda に関数をデプロイします。アカウントに Amazon S3 バケットを作成し、AWS SAM CLI がそのバケットを代わりに使用するようにするには、[作成] を選択して、画面の指示に従います。

[サーバーレスアプリケーションのデプロイ] ダイアログボックス

AWS Toolkit for JetBrains の [サーバーレスアプリケーションのデプロイ] ダイアログボックスは、AWS サーバーレスアプリケーションをデプロイする際に表示されます。



[サーバーレスアプリケーションのデプロイ] ダイアログボックスには、次の項目があります。

スタックの作成

(必須) 接続された AWS アカウント用に、AWS CloudFormation で作成する AWS Serverless Application Model (AWS SAM) コマンドラインインターフェイス (CLI) のスタック名を指定します。次に、AWS SAM CLI はこのスタックを使用して AWS サーバーレスアプリケーションをデプロイします。

スタックの更新

(必須) AWS サーバーレスアプリケーションをデプロイするために、AWS SAM CLI の接続された AWS アカウントで使用するの既存の AWS CloudFormation スタック名を選択します。

> **Note**
>
> [スタックの作成] または [スタックの更新] のいずれかが必要です (両方ではありません)。

テンプレートパラメーター

(オプション) AWS Toolkit for JetBrains が対応するプロジェクトの `template.yaml` ファイル内で検出する任意のパラメータ。パラメータの値を指定するには、パラメータの横にある [値] 列でボックスを選択し、値を入力して [Enter] を押します。詳細については、「AWS CloudFormation ユーザーガイド」の「パラメータ」を参照してください。

S3 バケット

(必須) AWS SAM CLI の接続された AWS アカウントで、AWS サーバーレスアプリケーションをデプロイするために使用する既存の Amazon Simple Storage Service (Amazon S3) バケットを選択します。アカウントに Amazon S3 バケットを作成し、AWS SAM CLI でそのバケットを使用するには、[作成] を選択し、画面の指示に従います。

ECR リポジトリ

(`Image` パッケージタイプの場合のみ必須) AWS SAM CLI の接続された AWS アカウントで、AWS サーバーレスアプリケーションをデプロイするために使用する既存の Amazon Elastic Container Registry (Amazon ECR) レポジトリ URI を選択します。AWS Lambda パッケージタイプの詳細については、「AWS Lambda デベロッパーガイド」の「Lambda デプロイパッケージ」を参照してください。

デプロイ前に確認が必要

(オプション) 選択すると、AWS CloudFormation で設定されたスタックの現在の変更を実行することで、対応するスタックの作成または更新が終わるまで待機するように AWS CloudFormation に指示します。この変更セットを実行しないと、AWS サーバーレスアプリケーションはデプロイフェーズに移行しません。

コンテナ内で関数を構築する

(オプション) 選択した場合、AWS SAM CLI は、デプロイ前にサーバーレスアプリケーションの関数をローカルに Lambda に類似した Docker コンテナ内に構築します。これは、関数がネイティブにコンパイルされた依存関係やプログラムを持つパッケージに依存する場合に便利です。詳細については、AWS Serverless Application Model デベロッパーガイドの「アプリケーションの構築」を参照してください。

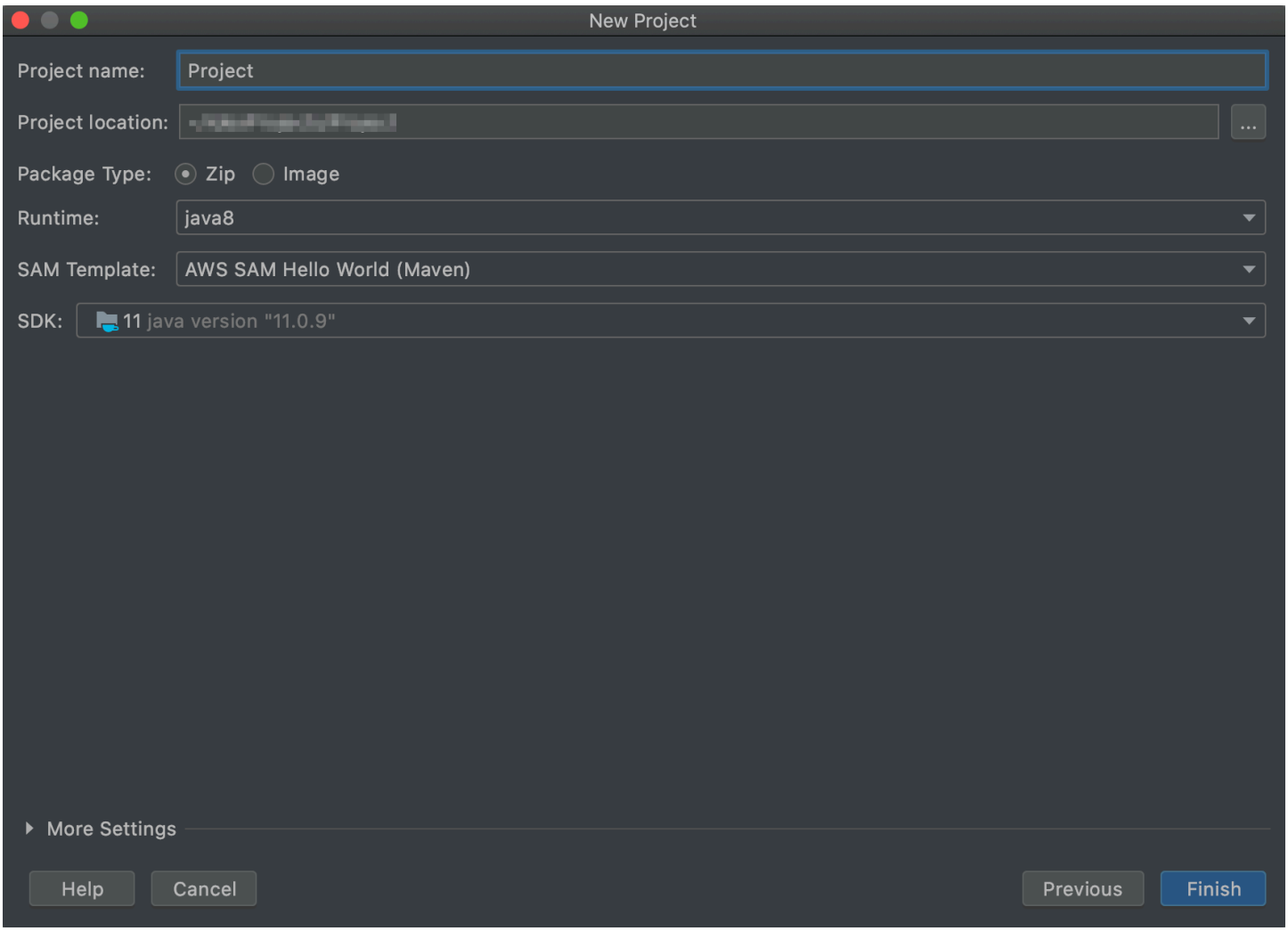
[New Project] (新しいプロジェクト) ダイアログボックス

AWS サーバーレスアプリケーションを作成すると、AWS Toolkit for JetBrains に [新しいプロジェクト] ダイアログボックスが表示されます。

トピック

[新規プロジェクト] ダイアログボックス (IntelliJ IDEA、PyCharm、WebStorm)

> **Note**
>
> 次のスクリーンショットは、IntelliJ IDEA の [新規プロジェクト] ダイアログボックスですが、フィールドの説明は PyCharm と WebStorm にも当てはまります。



[新規プロジェクト] ダイアログボックスには、次の項目があります。

プロジェクト名

(必須) プロジェクトの名前。

プロジェクトの場所

(必須) IntelliJ IDEA がプロジェクトを作成する場所。

パッケージタイプ

(必須) AWS Lambda 関数のデプロイパッケージ。`Image` または `Zip` になります。`Zip` パッケージタイプと `Image` パッケージタイプの違いの詳細については、「AWS Lambda デベロッパーガイド」の「Lambda デプロイパッケージ」を参照してください。

ランタイム

(必須) 使用する Lambda ランタイムの ID。

SAM テンプレート

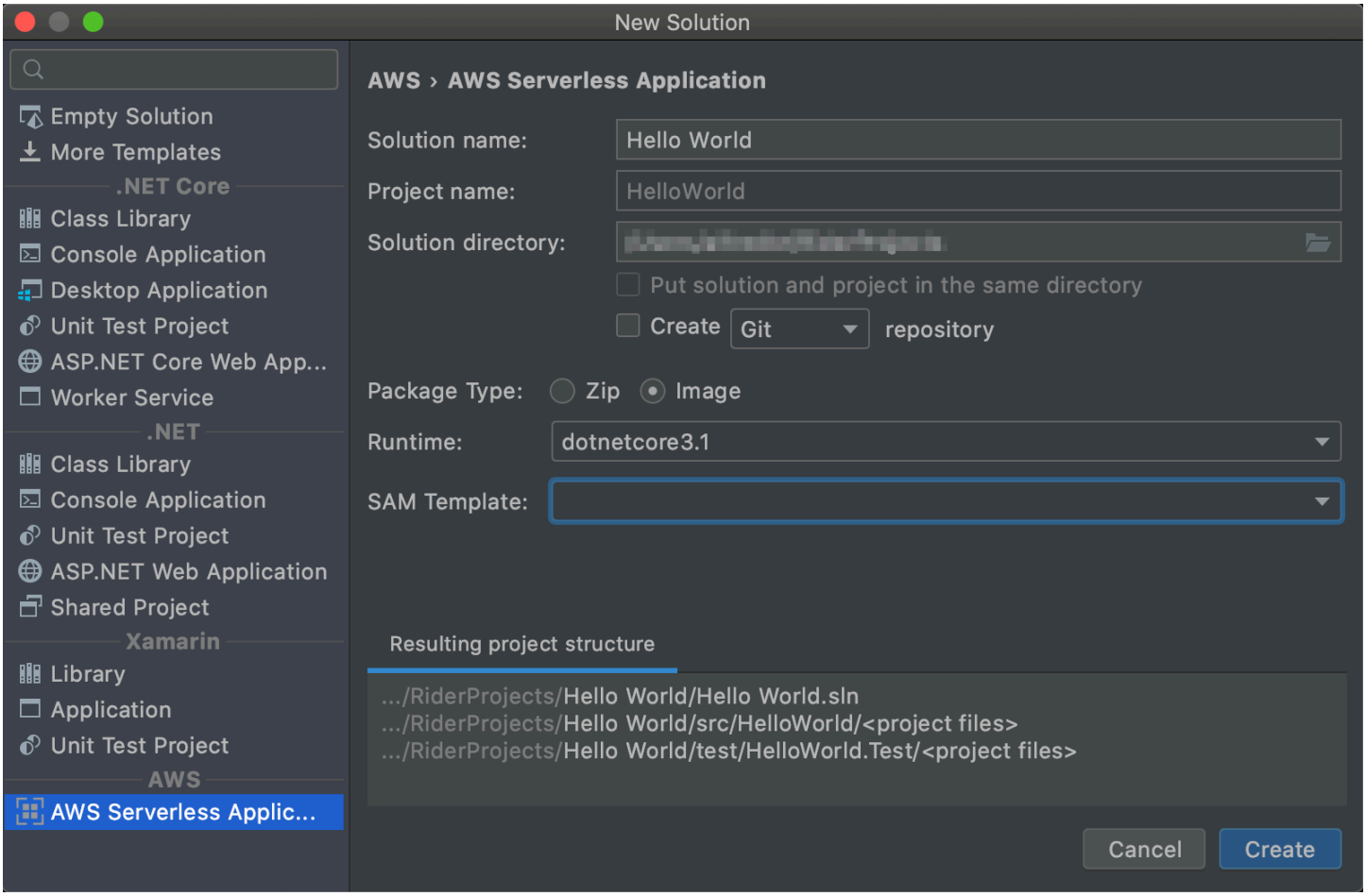
(必須) 使用する AWS Serverless Application Model (AWS SAM) テンプレートの名前。

プロジェクト SDK

(必須) 使用する Java 開発キット (JDK)。詳細については、「IntelliJ IDEA ヘルプ」ウェブサイトの「Java 開発キット (JDK)」を参照してください。

[新規プロジェクト] ダイアログボックス (JetBrains Rider)

> **Note**
>
> 新しいソリューションを作成すると、このダイアログボックスには [新規プロジェクト] ではなく [新規ソリューション] が含まれます。ただし、ダイアログボックスの内容は同じです。



[新規プロジェクト] ダイアログボックスには、次の項目があります。

ソリューション名

(必須) ソリューションの名前。

プロジェクト名

(必須) プロジェクトの名前。

ソリューションディレクトリ

(必須) ソリューションのディレクトリへのパス。

ソリューションとプロジェクトを同じディレクトリに置く

(オプション) 選択すると、ソリューションのファイルがプロジェクトのファイルと同じ場所に置かれます。

リポジトリの作成

(オプション) 選択すると、指定したプロバイダを持つプロジェクトのリモートリポジトリが作成されます。

パッケージタイプ

(必須) Lambda 関数のパッケージタイプは、`Zip` または `Image` のいずれかであること。`Zip` パッケージタイプと `Image` パッケージタイプの違いの詳細については、「AWS Lambda デベロッパーガイド」の「Lambda デプロイパッケージ」を参照してください。

ランタイム

(必須) 使用する Lambda ランタイムの ID。

SAM テンプレート

(必須) 使用する AWS SAM テンプレートの名前。

結果として生じるプロジェクトの構造

(編集不可) 作成されるプロジェクトのディレクトリとファイルのパス。

[設定の実行/デバッグ] ダイアログボックス

AWS Toolkit for JetBrains の [実行/デバッグ設定] ダイアログボックスは、ローカル、リモート、Amazon Elastic Container Service (Amazon ECS)クラスターのいずれかの場合で、実行/デバッグ設定を変更するたびに表示されます。

トピック

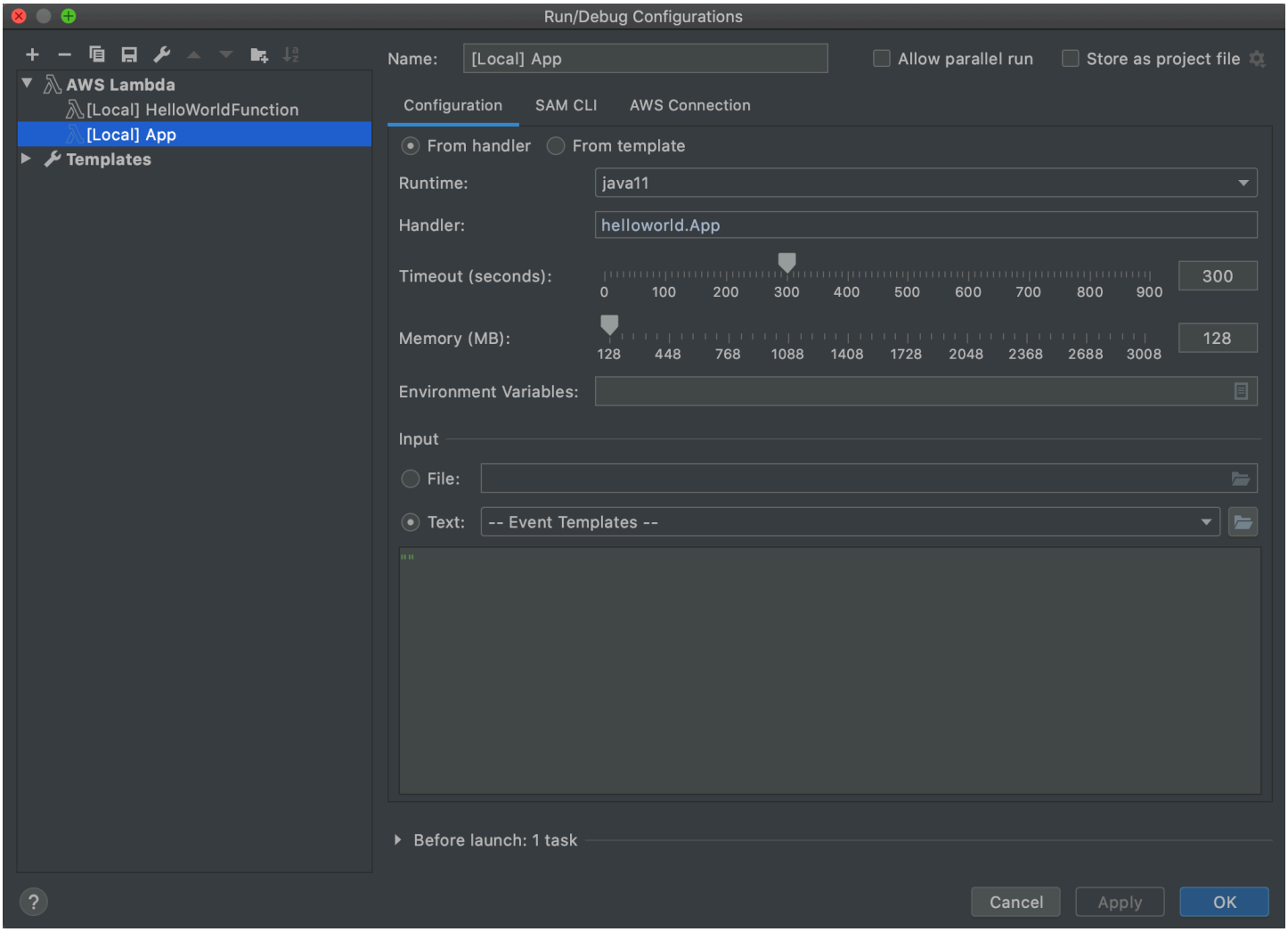
- [実行/デバッグ設定] ダイアログボックス (ローカル関数の設定)
- [実行/デバッグ設定] ダイアログボックス (リモート関数の設定)
- [設定の編集] ダイアログボックス (Amazon ECS クラスター)

[実行/デバッグ設定] ダイアログボックス (ローカル関数の設定)

このダイアログボックスは、AWS Lambda 関数のローカルバージョンの設定を更新するたびに表示されます。

> **Note**
>
> 同じ関数 (関数のソース コードはお使いの AWS アカウントの Lambda にあります) のリモートバージョンの設定を更新するには、代わりに [実行/デバッグ設定] ダイアログボックス (リモート関数の設定) を参照してください。

このダイアログボックスには、[設定]、[SAM CLI]、[AWS 接続]という 3 つのタブがあります。



ローカル関数設定の [実行/デバッグ設定] ダイアログボックスの [設定] タブには、次の項目が含まれています。

名前

(必須) この設定の名前。

並列実行を許可/並列実行を許可する

(オプション) 選択した場合、IntelliJ IDEA、PyCharm、WebStorm、JetBrains Rider が必要に応じて設定のインスタンスをいくつでも起動し、並行で実行できるようにします。[1]

ハンドラーから/テンプレートから

(必須) 選択したオプションにより、追加設定を行う必要があります。

ランタイム

(必須) 使用する Lambda ランタイムの ID。

Handler

([ハンドラーから] オプションで必須) Java、Python、Node.js、C#に対応する関数ハンドラーの識別子。

タイムアウト (秒)

([ハンドラーから] オプションで必須) Lambda が関数を停止するまでに許可する実行時間。最大で 900 秒 (15 分) を指定します。

メモリ (MB)

([ハンドラーから] オプションで必須) 関数が実行時に利用可能なメモリ量。メモリの量を 128 MB ~ 3,008 MB の範囲 (64 MB 単位) で指定します。

環境可変

([ハンドラーから] オプションで任意) Lambda 関数が使用する環境変数、キー値ペアとして指定されたもの。環境変数を追加、変更、または削除するには、フォルダアイコンを選択し、画面の指示に従います。

テンプレート

([ハンドラーから] オプションで必須) この設定で使用するAWS Serverless Application Model (AWS SAM) テンプレート(たとえば、`template.yaml`) の場所とファイル名、ならびにこの設定に関連付けるテンプレートのリソース。

File

(必須) 関数に渡すイベントデータの場所とファイル名(JSON 形式)。イベントデータの例については、「AWS Lambda デベロッパーガイド」の「Lambda 関数の呼び出し」と「AWS Serverless Application Model デベロッパーガイド」の「サンプルイベントペイロード」の生成を参照してください。

[Text] (テキスト)

(必須) 関数に渡すイベントデータ (JSON 形式)。イベントデータの例については、「AWS Lambda デベロッパーガイド」の「Lambda 関数の呼び出し」と「AWS Serverless Application Model デベロッパーガイド」の「サンプルイベントペイロード」の生成を参照してください。

> **Note**
>
> ファイルまたはテキストのいずれかが必要です (両方とも必要ではありません)。

起動前: ウィンドウ

(オプション) この設定を開始する前に実行する必要があるタスクをすべて一覧表示します。[2]

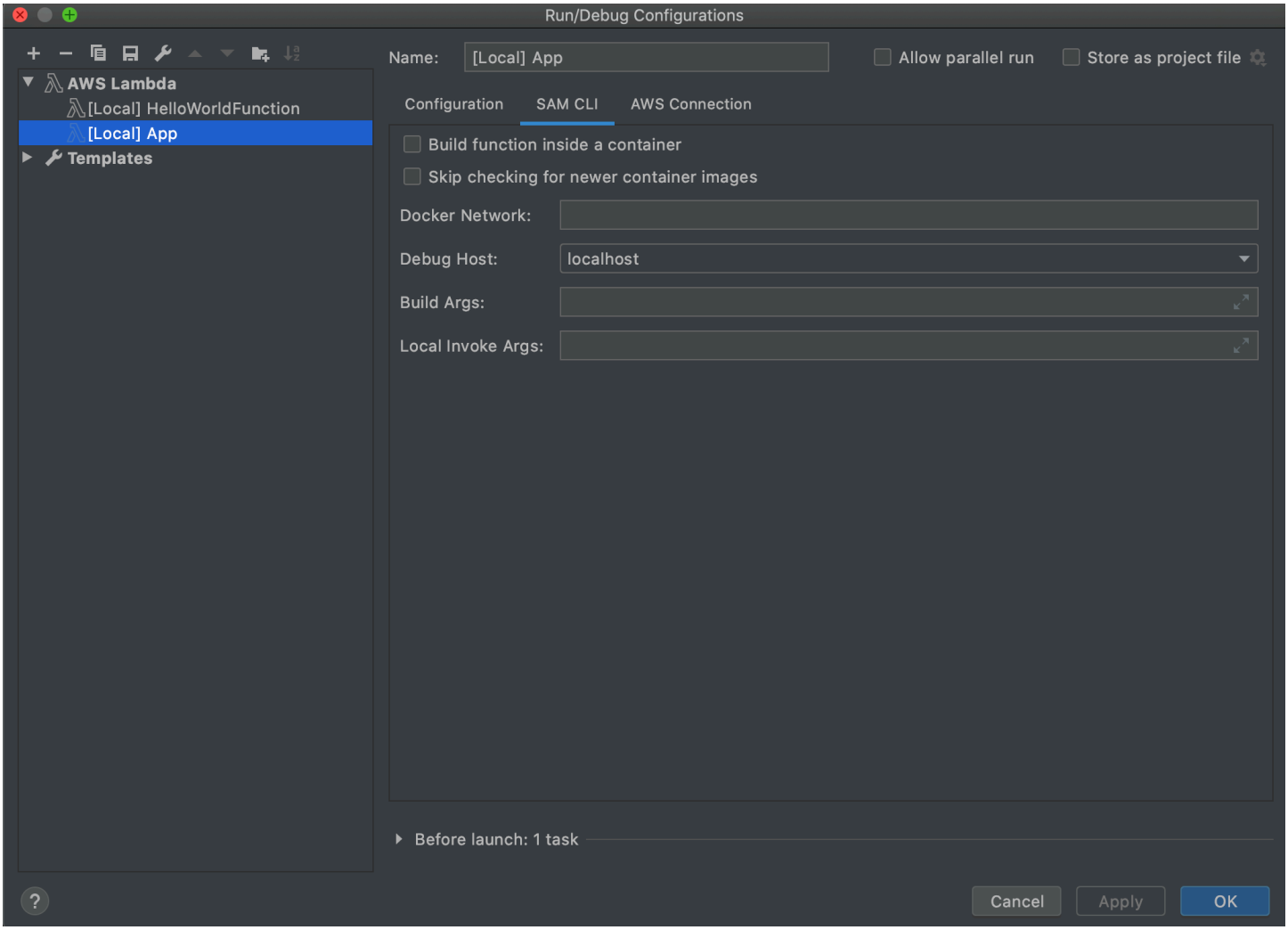
Notes (メモ)

[1] 詳細については、以下を参照してください。

- IntelliJ IDEA については、IntelliJ IDEA ヘルプウェブサイトの共通オプションを参照してください。
- PyCharm については、PyCharm ヘルプウェブサイトの共通オプションを参照してください。
- WebStorm については、WebStorm ヘルプウェブサイトの共通オプションを参照してください。
- JetBrains Rider については、JetBrains Rider ヘルプウェブサイトの共通オプションを参照してください。

[2] 詳細については、以下を参照してください。

- IntelliJ IDEA については、IntelliJ IDEA ヘルプウェブサイトの起動前オプションを参照してください。
- PyCharm については、PyCharm ヘルプウェブサイトの起動前オプションを参照してください。
- WebStorm については、WebStorm ヘルプウェブサイトの起動前オプションを参照してください。
- JetBrains Rider については、JetBrains Rider ヘルプウェブサイトの起動前オプションを参照してください。



ローカル関数設定の [実行/デバッグ設定] ダイアログボックスの [SAM CLI] タブには、次の項目が含まれています。

名前

(必須) この設定の名前。

並列実行を許可/並列実行を許可する

(オプション) 選択した場合、IntelliJ IDEA、PyCharm、WebStorm、JetBrains Rider が必要に応じて設定のインスタンスをいくつでも起動し、並行で実行できるようにします。[1]

コンテナ内で関数を構築する

(オプション) 選択した場合、AWS SAM CLI は、デプロイ前にサーバーレスアプリケーションの関数をローカルに Lambda に類似した Docker コンテナ内に構築します。これは、関数がネイティブにコンパイルされた依存関係やプログラムを持つパッケージに依存する場合に便利です。

詳細については、AWS Serverless Application Model デベロッパーガイドの「アプリケーションの構築」を参照してください。

新しいコンテナイメージのチェックをスキップする

(オプション) 選択した場合、AWS SAMCLI は、[設定] タブで指定されたランタイムの最新 Docker イメージのプルダウンをスキップします。

Docker ネットワーク

(オプション) デフォルトのブリッジ ネットワークを使用して、Lambda Docker コンテナが接続する既存の Docker ネットワークの名前または ID。指定されていない場合、Lambda コンテナはデフォルトのブリッジ Docker ネットワークのみに接続します。

起動前: ウィンドウ

(オプション) この設定を開始する前に実行する必要があるタスクをすべて一覧表示します。[2]

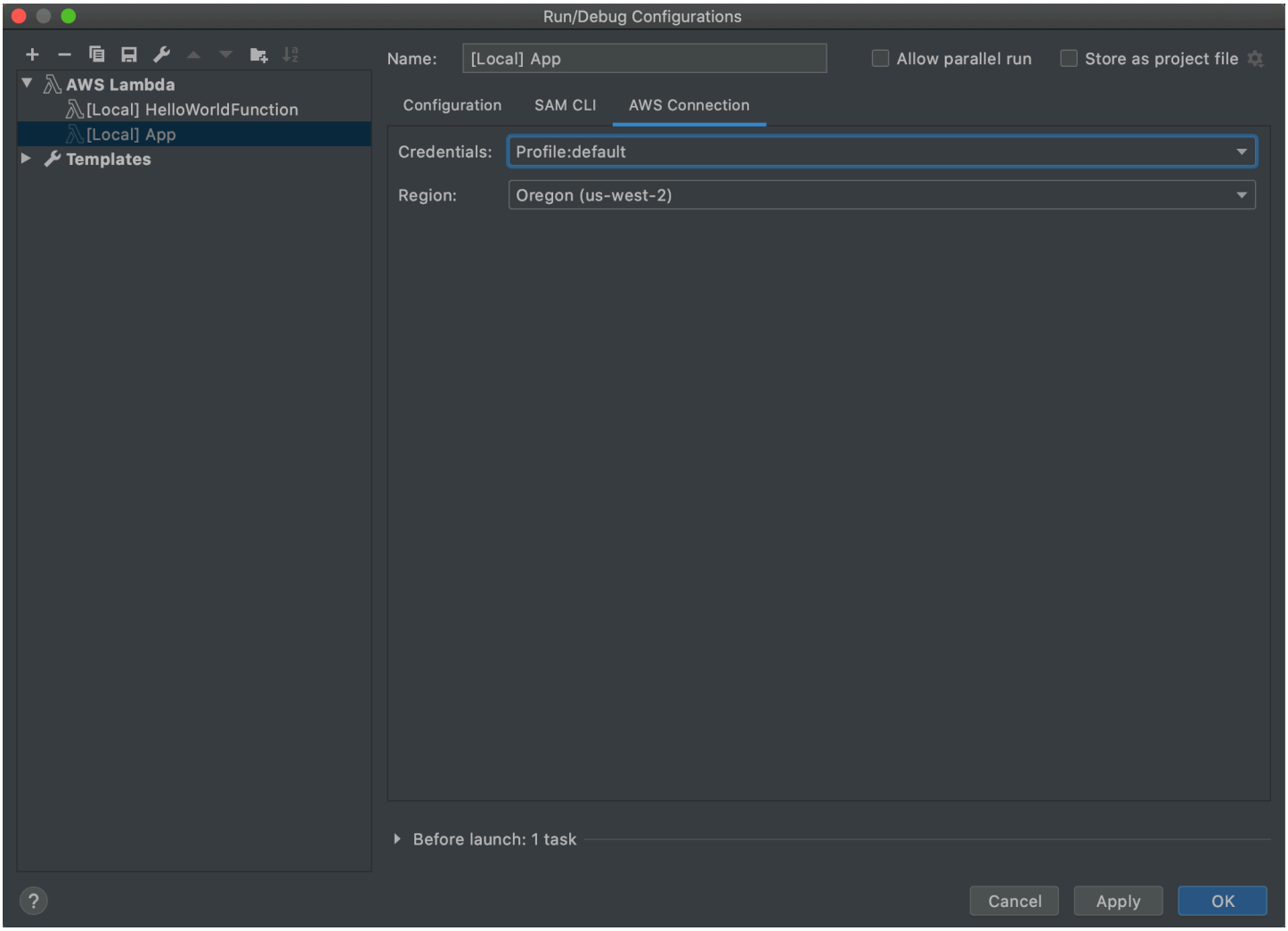
Notes (メモ)

[1] 詳細については、以下を参照してください。

- IntelliJ IDEA については、IntelliJ IDEA ヘルプウェブサイトの共通オプションを参照してください。
- PyCharm については、PyCharm ヘルプウェブサイトの共通オプションを参照してください。
- WebStorm については、WebStorm ヘルプウェブサイトの共通オプションを参照してください。
- JetBrains Rider については、JetBrains Rider ヘルプウェブサイトの共通オプションを参照してください。

[2] 詳細については、以下を参照してください。

- IntelliJ IDEA については、IntelliJ IDEA ヘルプウェブサイトの起動前オプションを参照してください。
- PyCharm については、PyCharm ヘルプウェブサイトの起動前オプションを参照してください。
- WebStorm については、WebStorm ヘルプウェブサイトの起動前オプションを参照してください。
- JetBrains Rider については、JetBrains Rider ヘルプウェブサイトの起動前オプションを参照してください。



ローカル関数設定の [実行/デバッグ設定] ダイアログボックスの [AWS 接続] タブには、次の項目が含まれています。

認証情報

(必須) 使用する既存の AWS アカウント接続の名前。

リージョン

(必須) 接続されているアカウントに使用する AWS リージョンの名前。

Notes (メモ)

[1] 詳細については、以下を参照してください。

- IntelliJ IDEA については、IntelliJ IDEA ヘルプウェブサイトの共通オプションを参照してください。

- PyCharm については、PyCharm ヘルプウェブサイトの共通オプションを参照してください。
- WebStorm については、WebStorm ヘルプウェブサイトの共通オプションを参照してください。
- JetBrains Rider については、JetBrains Rider ヘルプウェブサイトの共通オプションを参照してください。

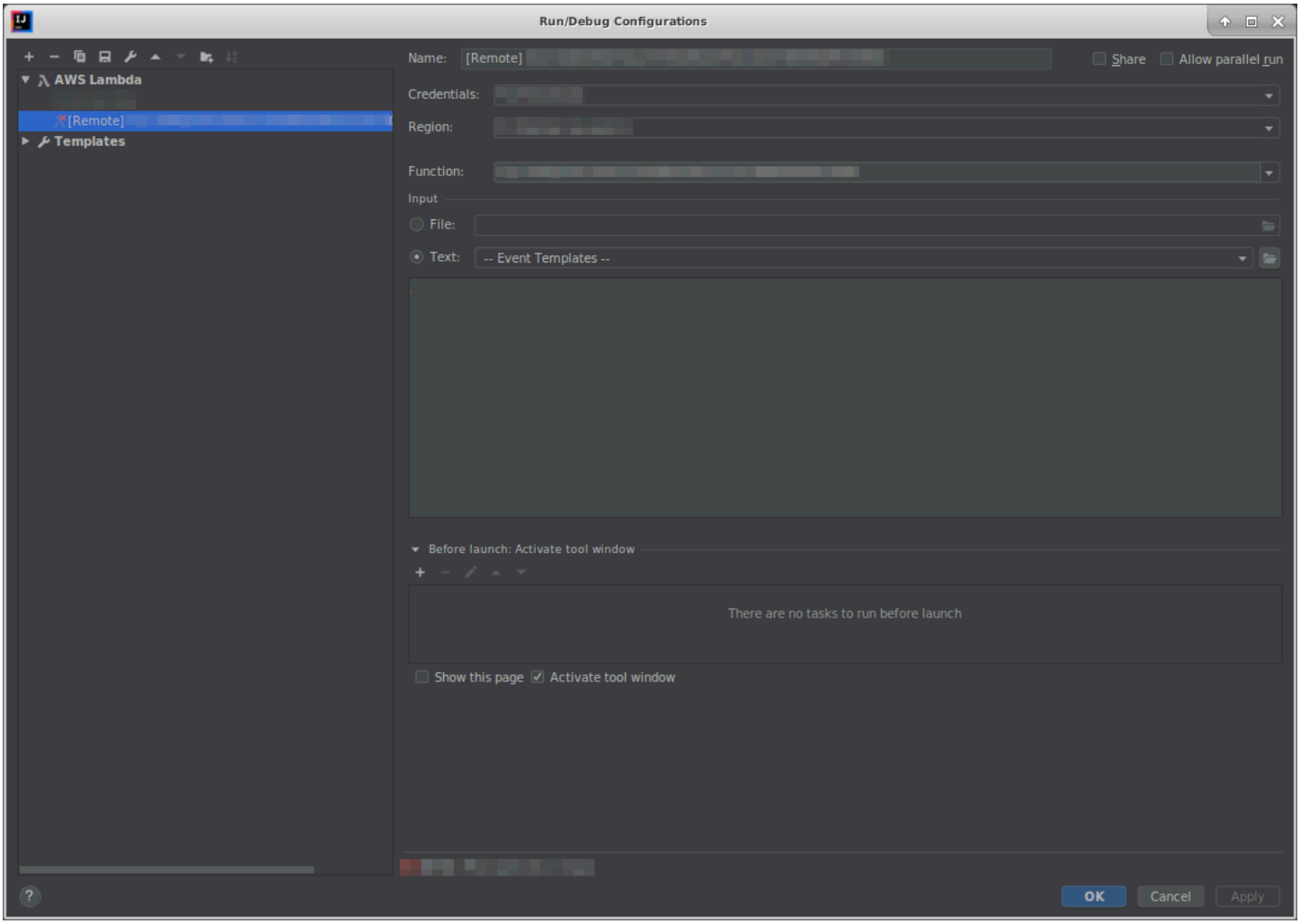
[2] 詳細については、以下を参照してください。

- IntelliJ IDEA については、IntelliJ IDEA ヘルプウェブサイトの起動前オプションを参照してください。
- PyCharm については、PyCharm ヘルプウェブサイトの起動前オプションを参照してください。
- WebStorm については、WebStorm ヘルプウェブサイトの起動前オプションを参照してください。
- JetBrains Rider については、JetBrains Rider ヘルプウェブサイトの起動前オプションを参照してください。

[実行/デバッグ設定] ダイアログボックス (リモート関数の設定)

このダイアログボックスは、AWS Lambda 関数 (関数のソース コードはお持ちの AWS アカウントの Lambda にあります) のリモートバージョンの設定を更新するたびに表示されます。

> **Note**
>
> 同じ関数のローカルバージョンの設定を変更するには、代わりに [実行/デバッグ設定] ダイアログボックス (ローカル関数の設定) を参照してください。
> ダイアログボックスの名前は [実行/デバッグ設定] ですが、AWS Toolkit for JetBrains を使用して Lambda 関数のリモート バージョンをデバッグすることはできません。実行できるのは関数のみです。



リモート関数設定の [実行/デバッグ設定] ダイアログボックスには、次の項目が含まれています。

名前

(必須) この設定の名前。

共有/VCS で共有

(オプション) 選択した場合、この設定を他のチームメンバーが使用できるようになります。[1]

並列実行を許可/並列実行を許可する

(オプション) 選択した場合、IntelliJ IDEA、PyCharm、WebStorm、JetBrains Rider が必要に応じて設定のインスタンスをいくつでも起動し、並行で実行できるようにします。[1]

認証情報

(必須) 使用する既存の AWS アカウント接続の名前。

リージョン

(必須) 接続されているアカウントに使用する AWS リージョンの名前。

関数

(必須) 使用する Lambda 関数の名前。

File

(必須) 関数に渡すイベントデータの場所とファイル名(JSON 形式)。イベントデータの例については、「AWS Lambda デベロッパーガイド」の「Lambda 関数の呼び出し」と「AWS Serverless Application Model デベロッパーガイド」の「サンプルイベントペイロード」の生成を参照してください。

[Text] (テキスト)

(必須) 関数に渡すイベントデータ (JSON 形式)。イベントデータの例については、「AWS Lambda デベロッパーガイド」の「Lambda 関数の呼び出し」と「AWS Serverless Application Model デベロッパーガイド」の「サンプルイベントペイロード」の生成を参照してください。

> **Note**
>
> ファイルまたはテキストのいずれかが必要です (両方とも必要ではありません)。

起動前: ツールウィンドウを有効化

(オプション) この設定を開始する前に実行する必要があるタスクをすべて一覧表示します。[2]

このページを表示

(オプション) 選択すると、この設定を開始する前にこれらの構成設定が表示されます。[2]

ツールウィンドウを有効化

(オプション) 選択した場合、この設定の開始時に[実行] または [デバッグ] ツールウィンドウが開きます。[2]

Notes (メモ)

[1] 詳細については、以下を参照してください。

- IntelliJ IDEA については、IntelliJ IDEA ヘルプウェブサイトの共通オプションを参照してください。

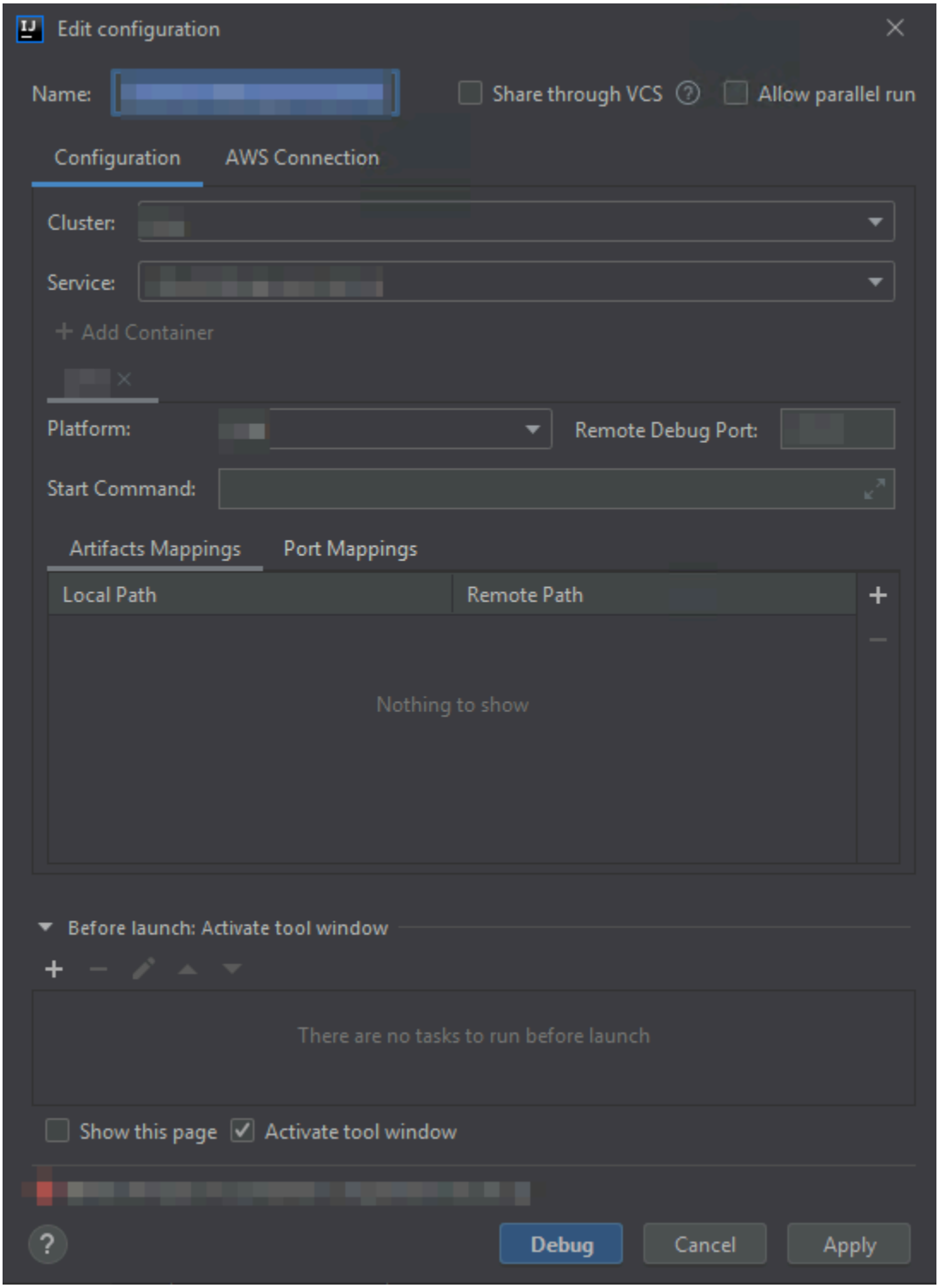
- PyCharm については、PyCharm ヘルプウェブサイトの共通オプションを参照してください。
- WebStorm については、WebStorm ヘルプウェブサイトの共通オプションを参照してください。
- JetBrains Rider については、JetBrains Rider ヘルプウェブサイトの共通オプションを参照してください。

[2] 詳細については、以下を参照してください。

- IntelliJ IDEA については、IntelliJ IDEA ヘルプウェブサイトの起動前オプションを参照してください。
- PyCharm については、PyCharm ヘルプウェブサイトの起動前オプションを参照してください。
- WebStorm については、WebStorm ヘルプウェブサイトの起動前オプションを参照してください。
- JetBrains Rider については、JetBrains Rider ヘルプウェブサイトの起動前オプションを参照してください。

[設定の編集] ダイアログボックス (Amazon ECS クラスター)

[Edit configuration (設定の編集)] ダイアログボックスには、[設定] と [AWS 接続] の 2 つのタブがあります。



[設定の編集] ダイアログボックスの [設定] タブには、次の項目が含まれています。

名前

(必須) この設定の名前。

共有/VCS で共有

(オプション) 選択した場合、この設定を他のチームメンバーが使用できるようになります。[1]

並列実行を許可/並列実行を許可する

(オプション) 選択した場合、IntelliJ IDEA、PyCharm、WebStorm、JetBrains Rider が必要に応じて設定のインスタンスをいくつでも起動し、並行で実行できるようにします。[1]

クラスター

(必須) デバッグする Amazon Elastic Container Service (Amazon ECS) クラスターの名前。

サービス

(必須) デバッグするクラスター内の Amazon ECS サービスの名前。

コンテナを追加

この設定にコンテナを追加します。少なくとも 1 つのタブが表示されている場合は、省略可能です。各タブは個別のコンテナを表します。

選択したコンテナには次の項目が適用されます: [プラットフォーム]、[Remote Debug Port (リモートデバッグポート)]、[Remote Debug Port (起動コマンド)]、[Artifacts Mappings (アーティファクトマッピング)]、および [ポートマッピング]。

プラットフォーム

(必須)使用するデバッグ プラットフォーム。

リモートデバッグポート

(オプション) デバッガーにアタッチするポート 一般的に、サービスがポート 20020～20030 を使用しない限り、これを指定しないでください。使用している場合、コンテナが他の場所で使用されている可能性のあるポートをバインドしないように、使用しているポートをこちらで指定します。

開始コマンド

(必須) デバッガーがアタッチできるように、プログラムを起動するコマンド。Java の場合、`java` で始まっている必要があり、`-Xdebug` などのデバッガー情報が含まれていてはなりません。Python の場合は、`python`、`python2`、または `python3` で始まり、その後に実行するファイルのパスと名前を指定する必要があります。

アーティファクトマッピング

(必須) コンテナ内の[リモートパス]にマッピングするローカル開発マシン上の[ローカルパス]。実行するすべてのコードとアーティファクトをマッピングする必要があります。ローカルパスとリモートパスマッピングを指定するには、[追加] ([+] アイコン) を選択します。

ポートマッピング

(オプション) コンテナ内の[リモートパス]にマッピングするローカル開発マシン上の[ローカルパス]。これにより、ローカル ポートがリモート リソースのポートと直接通信できるようになります。たとえば、コマンド `curl localhost:3422` の場合、ポート 3422 は一部のサービスにマッピングされます。ローカルおよびリモートのポートマッピングを指定するには、[Add] ([+] アイコン) を選択します。

起動前: ツールウィンドウを有効化

(オプション) この設定を開始する前に実行する必要があるタスクをすべて一覧表示します。[2]

このページを表示

(オプション) 選択すると、この設定を開始する前にこれらの構成設定が表示されます。[2]

ツールウィンドウを有効化

(オプション)選択した場合、この設定の開始時に [実行] または [デバッグ] ツールウィンドウが開きます。[2]

Notes (メモ)

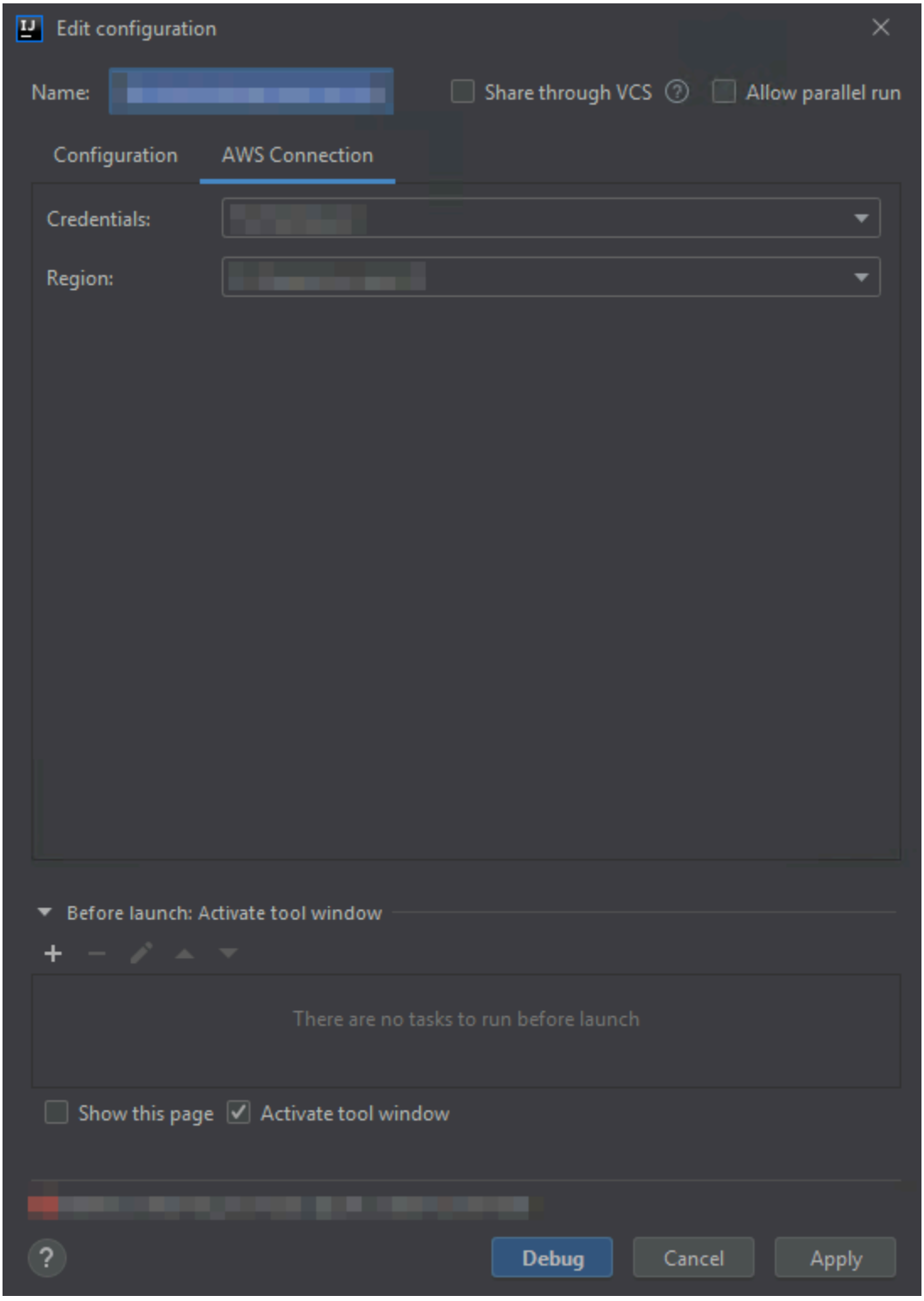
[1] 詳細については、以下を参照してください。

- IntelliJ IDEA については、IntelliJ IDEA ヘルプウェブサイトの共通オプションを参照してください。
- PyCharm については、PyCharm ヘルプウェブサイトの共通オプションを参照してください。
- WebStorm については、WebStorm ヘルプウェブサイトの共通オプションを参照してください。
- JetBrains Rider については、JetBrains Rider ヘルプウェブサイトの共通オプションを参照してください。

[2] 詳細については、以下を参照してください。

- IntelliJ IDEA については、IntelliJ IDEA ヘルプウェブサイトの起動前オプションを参照してください。

- PyCharm については、PyCharm ヘルプウェブサイトの起動前オプションを参照してください。
- WebStorm については、WebStorm ヘルプウェブサイトの起動前オプションを参照してください。
- JetBrains Rider については、JetBrains Rider ヘルプウェブサイトの起動前オプションを参照してください。



[設定の編集] ダイアログボックスの [AWS接続] タブには、次の項目が含まれています。

名前

(必須) この設定の名前。

認証情報

(必須) 使用する既存の AWS アカウント接続の名前。

リージョン

(必須) 接続されているアカウントに使用する AWS リージョンの名前。

共有/VCS で共有

(オプション) 選択した場合、この設定を他のチームメンバーが使用できるようになります。[1]

並列実行を許可/並列実行を許可する

(オプション) 選択した場合、IntelliJ IDEA、PyCharm、WebStorm、JetBrains Rider が必要に応じて設定のインスタンスをいくつでも起動し、並行で実行できるようにします。[1]

起動前: ツールウィンドウを有効化

(オプション) この設定を開始する前に実行する必要があるタスクをすべて一覧表示します。[2]

このページを表示

(オプション) 選択すると、この設定を開始する前にこれらの構成設定が表示されます。[2]

ツールウィンドウを有効化

(オプション)選択した場合、この設定の開始時に [実行] または [デバッグ] ツールウィンドウが開きます。[2]

Notes (メモ)

[1] 詳細については、以下を参照してください。

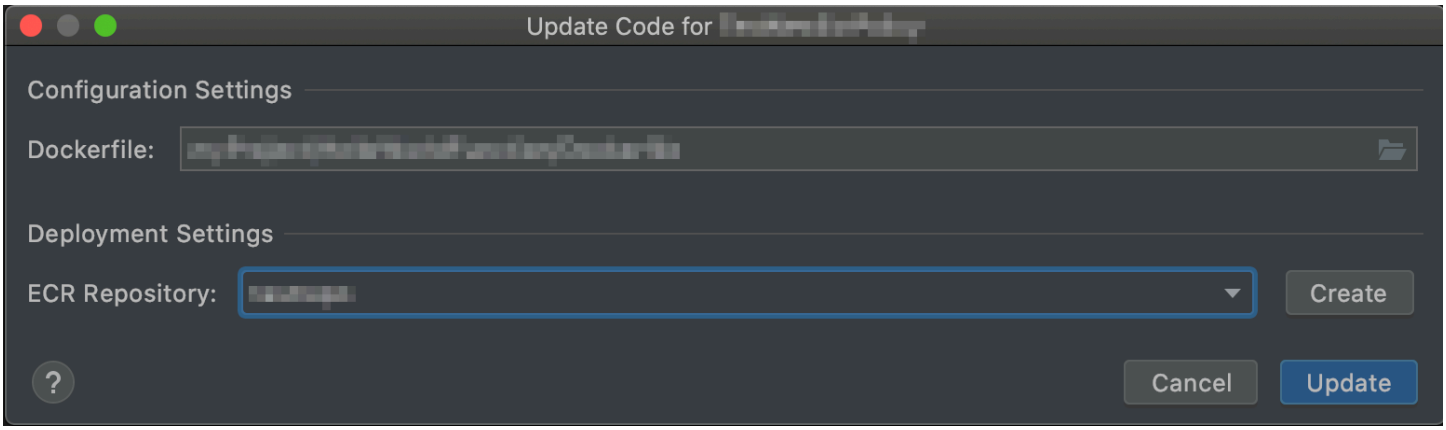
- IntelliJ IDEA については、IntelliJ IDEA ヘルプウェブサイトの共通オプションを参照してください。
- PyCharm については、PyCharm ヘルプウェブサイトの共通オプションを参照してください。
- WebStorm については、WebStorm ヘルプウェブサイトの共通オプションを参照してください。
- JetBrains Rider については、JetBrains Rider ヘルプウェブサイトの共通オプションを参照してください。

[2] 詳細については、以下を参照してください。

- IntelliJ IDEA については、IntelliJ IDEA ヘルプウェブサイトの起動前オプションを参照してください。
- PyCharm については、PyCharm ヘルプウェブサイトの起動前オプションを参照してください。
- WebStorm については、WebStorm ヘルプウェブサイトの起動前オプションを参照してください。
- JetBrains Rider については、JetBrains Rider ヘルプウェブサイトの起動前オプションを参照してください。

[コードの更新] ダイアログボックス

AWS Toolkit for JetBrains の [コードの更新] ダイアログボックスは、AWS Lambda 関数を更新するたびに表示されます。



[コードの更新] ダイアログボックスには、次の項目が含まれます。

Handler

(必須) Java、Python、Node.js、C#に対応する Lambda 関数ハンドラーの ID。

ソースバケット

(`Zip` パッケージタイプのみに必要) 使用する AWS Serverless Application Model (AWS SAM) コマンドラインインターフェイス (CLI) の接続された AWS アカウントの既存 Amazon Simple Storage Service (Amazon S3) バケットを選択し、Lambda に関数をデプロイします。アカウントに Amazon S3 バケットを作成し、AWS SAM CLI でそのバケットを使用するには、[作成] を選択し、画面の指示に従います。Lambda パッケージタイプ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。

ECR リポジトリ

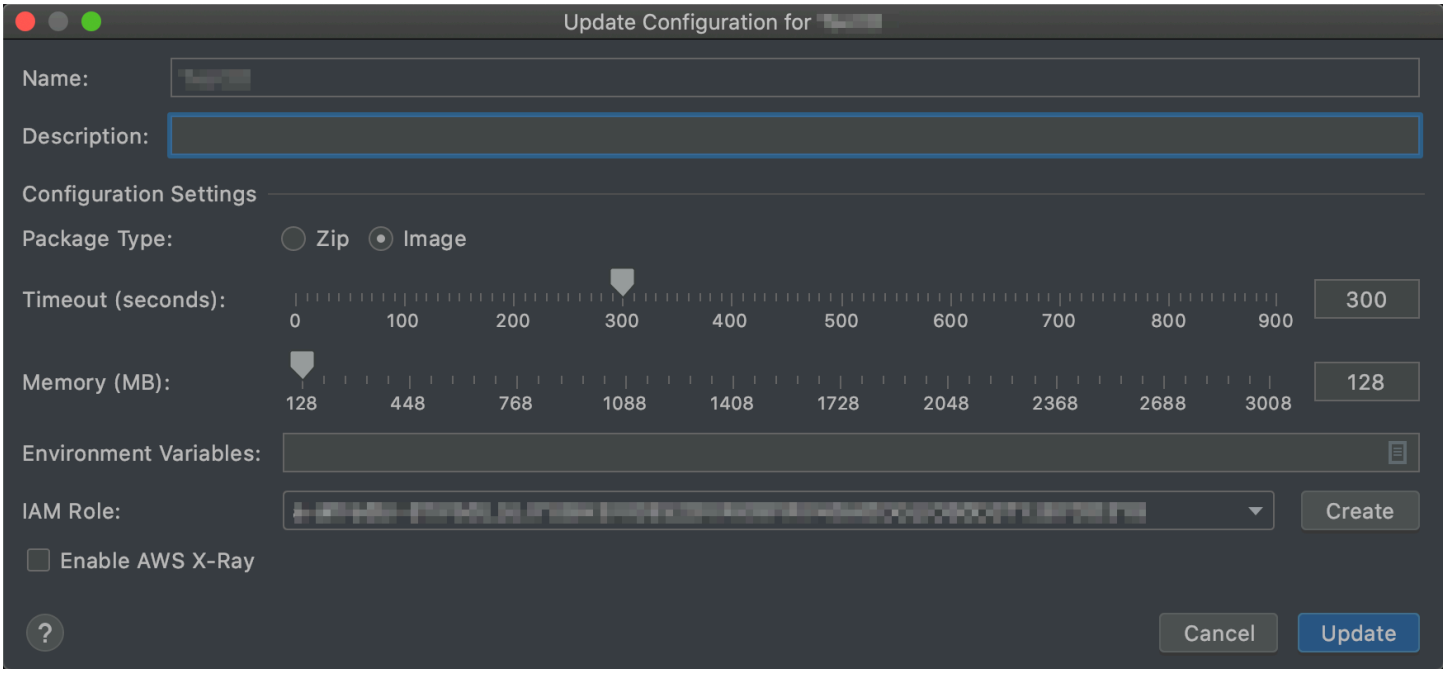
(Image パッケージタイプにのみ必要) 使用する AWS SAM CLI の接続された AWS アカウントの既存 Amazon Elastic Container Registry (Amazon ECR) リポジトリを選択し、Lambda に関数をデプロイします。

[設定の更新] ダイアログボックス

AWS Toolkit for JetBrains の [設定の更新] ダイアログボックスは、AWS Lambda 関数の設定を更新するたびに表示されます。入力する情報は、プロジェクトの Lambda 関数のパッケージタイプが `Zip` か `Image` かによって若干異なります。

`Zip` パッケージタイプの [設定の更新] ダイアログボックス:

`Image` パッケージタイプの [設定の更新] ダイアログボックス:



[設定の更新] ダイアログボックスには、次の項目が含まれます

名前

(必須) 関数の名前。A から Z の大文字、a から z の小文字、0 から 9の数字、ハイフン (-)、アンダースコア (_)のみ使用できます。名前は 64 文字未満にする必要があります。

説明

(オプション) 関数についてのわかりやすい説明。

パッケージタイプ

(必須) Lambda 関数のパッケージタイプは、`Zip` または `Image` のいずれかであること。

Handler

(`Zip` パッケージのみ必須) Java、Python、Node.js、C# に対応する Lambda 関数ハンドラーの ID。

ランタイム

(`Zip` パッケージのみに必須) 使用するLambda ランタイムの ID。

タイムアウト (秒)

(必須) Lambda が関数を停止するまでに許可する実行時間。最大で 900 秒 (15 分) を指定します。

メモリ (MB)

(必須) 関数が実行する際に使用可能なメモリ量。メモリの量を 128 MB ~ 3,008 MB の範囲 (64 MB 単位) で指定します。

環境可変

(オプション) キー値ペアとして指定された Lambda 関数が使用する環境変数。環境変数を追加、変更、または削除するには、フォルダアイコンを選択し、画面の指示に従います。

IAM ロール

(必須) 接続された AWS アカウントの Lambda 実行ロールで Lambda が関数に使用するものを選択します。アカウントに実行ロールを作成し、代わりに Lambda がそのロールを使用するには、[作成] を選択して画面の指示に従ってください。

AWS X-Ray の有効化

(オプション) 選択した場合、Lambda は AWS X-Ray を有効にして関数のパフォーマンス問題を検出、分析、最適化します。X-Ray は、Lambda や関数を構成するアップストリームまたはダウンストリームサービスからメタデータを収集します。X-Ray はこのメタデータを使用して、関数パフォーマンスに影響を及ぼすパフォーマンスボトルネック、レイテンシースパイク、およびその他問題を示す詳細なサービスグラフを生成します。

この AWS 製品またはサービスのセキュリティ

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 とユーザーの間の共有責任です。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 は、AWS クラウド内でサービスを実行するインフラストラクチャを保護する責任を担い、安全に使用できるサービスを提供します。当社のセキュリティ責任は AWS における最優先事項であり、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人によって定期的にテストおよび検証されています。

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 のサービスや、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、サポートしている特定の Amazon Web Services (AWS) のサービスを通じて、責任共有モデルに従います。AWS サービスのセキュリティ情報については、「AWS のサービスのセキュリティに関するドキュメントページ」と、「AWS コンプライアンスプログラムごとのコンプライアンスの取り組みのAWS 対象となるサービス」を参照してください。

トピック

- AWS Toolkit for JetBrains のデータ保護
- 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

AWS Toolkit for JetBrains のデータ保護

AWS 責任共有モデルは、AWS Toolkit for JetBrains のデータ保護に適用されます。このモデルで説明されているように、AWS は、AWS クラウド のすべてを実行するグローバルインフラストラクチャを保護する責任を負います。顧客は、このインフラストラクチャでホストされているコンテンツに対する管理を維持する責任があります。このコンテンツには、使用される AWS サービス のセキュリティ構成と管理タスクが含まれます。データプライバシーの詳細については、「データプライ

バシーのよくある質問」を参照してください。欧州でのデータ保護の詳細については、AWS セキュリティブログに投稿された「AWS 責任共有モデルおよび GDPR」のブログ記事を参照してください。

データを保護するため、AWS アカウント の認証情報を保護し、AWS IAM Identity Center または AWS Identity and Access Management (IAM) を使用して個々のユーザーをセットアップすることをお勧めします。この方法により、それぞれのジョブを遂行するために必要なアクセス許可のみを各ユーザーに付与できます。また、次の方法でデータを保護することをお勧めします。

- 各アカウントで多要素認証 (MFA) を使用します。
- SSL/TLS を使用して AWS リソースと通信します。TLS 1.2 および TLS 1.3 をお勧めします。
- AWS CloudTrail で API とユーザーアクティビティログをセットアップします。
- AWS サービス 内でデフォルトである、すべてのセキュリティ管理に加え、AWS の暗号化ソリューションを使用します。
- Amazon Macie などの高度なマネージドセキュリティサービスを使用します。これらは、Amazon S3 に保存されている機密データの検出と保護を支援します。
- コマンドラインインターフェイスまたは API を使用して AWS にアクセスするときに FIPS 140-2 検証済みの暗号化モジュールが必要な場合は、FIPS エンドポイントを使用します。利用可能な FIPS エンドポイントの詳細については、連邦情報処理規格 (FIPS) 140-2 を参照してください。

お客様の E メールアドレスなどの機密情報やセンシティブ情報は、タグや [Name] フィールドなどの自由形式のフィールドに配置しないことを強くお勧めします。これには、コンソール、API、AWS CLI、AWS SDK を使用して AWS Toolkit for JetBrains またはその他の AWS サービス を使用する場合も含まれます。名前に使用する自由記述のテキストフィールドやタグに入力したデータは、課金や診断ログに使用される場合があります。外部サーバーへ URL を供給する場合は、そのサーバーへのリクエストを検証するために、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するために役立つ AWS サービスです。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 (アクセス許可を持たせる) かを制御します。IAM は、無料で使用できる AWS サービスです。

トピック

- 対象者

- アイデンティティを使用した認証
- ポリシーを使用したアクセスの管理
- AWS サービスと IAM の連携の仕組み
- AWS アイデンティティとアクセスに関するトラブルシューティング

対象者

AWS Identity and Access Management (IAM) の用途は、AWS で行う作業によって異なります。

サービスユーザー – ジョブを実行するために AWS サービスを使用する場合は、管理者が必要なアクセス許可と認証情報を用意します。作業を実行するためにさらに多くの AWS の機能を使用するとき、追加の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、管理者から適切な許可をリクエストするのに役に立ちます。AWS の機能にアクセスできない場合は、「AWS アイデンティティとアクセスに関するトラブルシューティング」を参照するか、使用している AWS サービス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、AWS へのフルアクセスがあります。サービスのユーザーがどの 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許可を変更する必要があります。このページの情報を確認して、IAM の基本概念を理解してください。AWS で IAM を利用する方法の詳細については、使用している AWS 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作成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 ID ベースのポリシーの例を表示するには、使用している AWS サービス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、アイデンティティ認証情報を使用して AWS にサインインする方法です。ユーザーは、AWS アカウントのルートユーザー として、または IAM ロールを引き受けることによって、認証済み (AWS にサインイン済み) である必要があります。

ID ソースから提供された認証情報を使用して、フェデレーティッドアイデンティティとして AWS にサインインできます。 AWS IAM Identity Centerフェデレーティッドアイデンティティの例としては、(IAM Identity Center) ユーザー、会社のシングルサインオン認証、Google または Facebook の認証情報などがあります。フェデレーティッドアイデンティティとしてサインインする場合、IAM ロールを使用して、前もって管理者により ID フェデレーションが設定されています。フェデレーションを使用して AWS にアクセスする場合、間接的にロールを引き受けることになります。

ユーザーのタイプに応じて、AWS Management Console または AWS アクセスポータルにサインインできます。AWS への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。

プログラムを使用して AWS にアクセスする場合、AWS は Software Development Kit (SDK) とコマンドラインインターフェイス (CLI) を提供し、認証情報を使用してリクエストに暗号で署名します。AWS ツールを使用しない場合は、リクエストに自分で署名する必要があります。リクエストに署名する推奨方法の使用については、「IAM ユーザーガイド」の「AWS API リクエストの署名」を参照してください。

使用する認証方法を問わず、セキュリティ情報の提供を追加でリクエストされる場合もあります。例えば、AWS は、アカウントのセキュリティを強化するために多要素認証 (MFA) を使用することをお勧めします。詳細については、「AWS IAM Identity Center ユーザーガイド」の「Multi-factor authentication」(多要素認証) および「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。

AWS アカウント ルートユーザー

AWS アカウント を作成する場合は、このアカウントのすべての AWS サービス とリソースに対して完全なアクセス権を持つ 1 つのサインインアイデンティティから始めます。この ID は AWS アカウント ルートユーザーと呼ばれ、アカウントの作成に使用した E メールアドレスとパスワードでサインインすることによってアクセスできます。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。ルートユーザーの認証情報を保護し、それらを使用してルートユーザーのみが実行できるタスクを実行します。ルートユーザーとしてサインインする必要があるタスクの完全なリストについては、「AWS Account Management 全般のリファレンスガイド」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。

フェデレーティッド ID

ベストプラクティスとして、管理者アクセスを必要とするユーザーを含む人間のユーザーに対し、ID プロバイダーとのフェデレーションを使用して、一時的な認証情報の使用により、AWS サービス にアクセスすることを要求します。

フェデレーティッド ID は、エンタープライズユーザーディレクトリ、ウェブ ID プロバイダー、AWS Directory Service、Identity Center ディレクトリのユーザーか、または ID ソースから提供された認証情報を使用して AWS サービス にアクセスするユーザーです。フェデレーティッド ID が AWS アカウント にアクセスすると、ロールが継承され、ロールは一時的な認証情報を提供します。

アクセスを一元管理する場合は、AWS IAM Identity Center を使用することをお勧めします。IAM Identity Center でユーザーとグループを作成するか、すべての AWS アカウント とアプリケーションで使用するために、独自の ID ソースで一連のユーザーとグループに接続して同期することもできます。IAM Identity Center の詳細については、「AWS IAM Identity Center ユーザーガイド」の「What is IAM Identity Center?」(IAM Identity Center 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、1 人のユーザーまたは 1 つのアプリケーションに対して特定の許可を持つ AWS アカウント 内のアイデンティティです。可能であれば、パスワードやアクセスキーなどの長期的な認証情報を保有する IAM ユーザーを作成する代わりに、一時的な認証情報を使用することをお勧めします。ただし、IAM ユーザーとの長期的な認証情報が必要な特定のユースケースがある場合は、アクセスキーをローテーション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「長期的な認証情報を必要とするユースケースのためにアクセスキーを定期的にローテーションする」を参照してください。

IAM グループは、IAM ユーザーの集団を指定するアイデンティティです。グループとしてサインインすることはできません。グループを使用して、複数のユーザーに対して一度に許可を指定できます。多数のユーザーグループがある場合、グループを使用することで許可の管理が容易になります。例えば、IAMAdmins という名前のグループを設定して、そのグループに IAM リソースを管理する許可を与えることができます。

ユーザーは、ロールとは異なります。ユーザーは 1 人の人または 1 つのアプリケーションに一意に関連付けられますが、ロールはそれを必要とする任意の人が引き受けるようになっています。ユーザーには永続的な長期の認証情報がありますが、ロールでは一時的な認証情報が提供されます。詳細については、「IAM ユーザーガイド」の「IAM ユーザー (ロールではなく) の作成が適している場合」を参照してください。

IAM ロール

IAM ロールは、特定の許可を持つ、AWS アカウント 内のアイデンティティです。これは IAM ユーザーに似ていますが、特定のユーザーには関連付けられていません。ロールを切り替えることによって、AWS Management Console で IAM ロールを一時的に引き受けることができます。ロールを引き受けるには、AWS CLI または AWS API オペレーションを呼び出すか、カスタム URL を使用します。ロールを使用する方法の詳細については、「IAM ユーザーガイド」の「IAM ロールの使用」を参照してください。

IAM ロールと一時的な認証情報は、次の状況で役立ちます。

- フェデレーティッドユーザーアクセス – フェデレーティッドアイデンティティに許可を割り当てるには、ロールを作成してそのロールの許可を定義します。フェデレーティッドアイデンティティが認証されると、そのアイデンティティはロールに関連付けられ、ロールで定義されている許可が付与されます。フェデレーションの詳細については、「IAM ユーザーガイド」の「Creating a role for a third-party Identity Provider」(サードパーティーアイデンティティプロバイダー向けロールの作成) を参照してください。IAM アイデンティティセンターを使用する場合、許可セットを設定します。アイデンティティが認証後にアクセスできるものを制御するため、IAM Identity Center は、アクセス許可セットを IAM のロールに関連付けます。アクセス許可セットの詳細については、「AWS IAM Identity Center ユーザーガイド」の「アクセス許可セット」を参照してください。
- 一時的な IAM ユーザー許可 - IAM ユーザーまたはロールは、特定のタスクに対して複数の異なる許可を一時的に IAM ロールで引き受けることができます。
- クロスアカウントアクセス - IAM ロールを使用して、自分のアカウントのリソースにアクセスすることを、別のアカウントの人物 (信頼済みプリンシパル) に許可できます。クロスアカウントアクセスを許可する主な方法は、ロールを使用することです。ただし、一部の AWS サービス では、(ロールをプロキシとして使用する代わりに) リソースにポリシーを直接アタッチできます。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違いについては、「IAM ユーザーガイド」の「IAM ロールとリソースベースのポリシーとの相違点」を参照してください。
- クロスサービスアクセス - 一部の AWS サービス では、他の AWS サービス の機能を使用します。例えば、サービスで呼び出しを行うと、通常そのサービスによって Amazon EC2 でアプリケーションが実行されたり、Amazon S3 にオブジェクトが保存されたりします。サービスでは、呼び出し元プリンシパルの許可、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使用してこれを行う場合があります。
  - プリンシパル許可 - IAM ユーザーまたはロールを使用して AWS でアクションを実行する場合、そのユーザーはプリンシパルと見なされます。ポリシーによって、プリンシパルに許可が付与されます。一部のサービスを使用する際に、アクションを実行することで、別のサービスの別のアクションがトリガーされることがあります。この場合、両方のアクションを実行するための許可が必要です。アクションがポリシーで追加の依存アクションを必要とするかどうかを確認するには、「サービス認可リファレンス」の「AWS サービス のアクション、リソース、および条件キー」を参照してください。
  - サービスロール - サービスがユーザーに代わってアクションを実行するために引き受ける IAM ロールです。IAM 管理者は、IAM 内からサービスロールを作成、変更、削除できます。詳細については、「IAM ユーザーガイド」の「AWS サービス にアクセス許可を委任するロールの作成」を参照してください。
  - サービスにリンクされたロール - サービスにリンクされたロールは、AWS サービス にリンクされたサービスロールの一種です。サービスは、ユーザーに代わってアクションを実行するロー

ルを引き受けることができます。サービスにリンクされたロールは、AWS アカウント に表示され、サービスによって所有されます。IAM 管理者は、サービスにリンクされたロールの許可を表示できますが、編集することはできません。

- Amazon EC2 で実行されているアプリケーション - EC2 インスタンスで実行され、AWS CLI または AWS API 要求を行っているアプリケーションの一時的な認証情報を管理するには、IAM ロールを使用できます。これは、EC2 インスタンス内でのアクセスキーの保存に推奨されます。AWS ロールを EC2 インスタンスに割り当て、そのすべてのアプリケーションで使用できるようにするには、インスタンスにアタッチされたインスタンスプロファイルを作成します。インスタンスプロファイルにはロールが含まれ、EC2 インスタンスで実行されるプログラムは一時的な認証情報を取得することができます。詳細については、「IAM ユーザーガイド」の「Amazon EC2 インスタンスで実行されるアプリケーションに IAM ロールを使用して許可を付与する」を参照してください。

IAM ロールと IAM ユーザーのどちらを使用するかについては、「IAM ユーザーガイド」の「(IAM ユーザーではなく) IAM ロールをいつ作成したら良いのか?」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

AWS でアクセスをコントロールするには、ポリシーを作成して AWS アイデンティティまたはリソースにアタッチします。ポリシーは AWS のオブジェクトであり、アイデンティティやリソースに関連付けて、これらのアクセス許可を定義します。AWS は、プリンシパル (ユーザー、ルートユーザー、またはロールセッション) がリクエストを行うと、これらのポリシーを評価します。ポリシーでの許可により、リクエストが許可されるか拒否されるかが決まります。大半のポリシーは JSON ドキュメントとして AWS に保存されます。JSON ポリシードキュメントの構造と内容の詳細については、「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 AWS JSON ポリシーを使用して、誰が何にアクセスできるかを指定できます。つまり、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下でアクションを実行できるかということです。

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、リソースで必要なアクションを実行するためのアクセス許可をユーザーに付与する IAM ポリシーを作成できます。その後、管理者はロールに IAM ポリシーを追加し、ユーザーはロールを引き受けることができます。

IAM ポリシーは、オペレーションの実行方法を問わず、アクションの許可を定義します。例えば、`iam:GetRole` アクションを許可するポリシーがあるとします。このポリシーがあるユーザーは、AWS Management Console、AWS CLI、または AWS API からロールの情報を取得できます。

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ポリシーは、IAM ユーザー、ユーザーのグループ、ロールなど、アイデンティティにアタッチできる JSON 許可ポリシードキュメントです。これらのポリシーは、ユーザーとロールが実行できるアクション、リソース、および条件を制御します。アイデンティティベースのポリシーを作成する方法については、「IAM ユーザーガイド」の「IAM ポリシーの作成」を参照してください。

アイデンティティベースのポリシーは、さらにインラインポリシーまたはマネージドポリシーに分類できます。インラインポリシーは、単一のユーザー、グループ、またはロールに直接埋め込まれます。マネージドポリシーは、AWS アカウント 内の複数のユーザー、グループ、およびロールにアタッチできるスタンドアロンポリシーです。マネージドポリシーには、AWS マネージドポリシーとカスタマーマネージドポリシーがあります。マネージドポリシーまたは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「マネージドポリシーとインラインポリシーの比較」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、リソースにアタッチする JSON ポリシードキュメントです。リソースベースのポリシーには例として、IAM ロールの信頼ポリシーや Amazon S3 バケットポリシーがあげられます。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。ポリシーが添付されているリソースの場合、指定されたプリンシパルがそのリソースに対して実行できるアクションと条件は、ポリシーによって定義されます。リソースベースのポリシーで、プリンシパルを指定する必要があります。プリンシパルには、アカウント、ユーザー、ロール、フェデレーティッドユーザー、または AWS サービス を含めることができます。

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は使用できません。

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許可を持つかをコントロールします。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。

Simple Storage Service (Amazon S3)、AWS WAF、および Amazon VPC は、ACL をサポートするサービスの例です。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 では、その他の一般的ではないポリシータイプもサポートしています。これらのポリシータイプでは、より一般的なポリシータイプで付与された最大の許可を設定できます。

- アクセス許可の境界 - アクセス許可の境界は、アイデンティティベースのポリシーによって IAM エンティティ (IAM ユーザーまたはロール) に付与できる許可の上限を設定する高度な機能です。エンティティにアクセス許可の境界を設定できます。結果として得られるアクセス許可は、エンティティのアイデンティティベースポリシーとそのアクセス許可の境界の共通部分になります。`Principal` フィールドでユーザーまたはロールを指定するリソースベースのポリシーでは、アクセス許可の境界は制限されません。これらのポリシーのいずれかを明示的に拒否した場合、許可は無効になります。アクセス許可の境界の詳細については、「IAM ユーザーガイド」の「IAM エンティティのアクセス許可の境界」を参照してください。
- サービスコントロールポリシー (SCP) - SCP は、AWS Organizations で組織や組織単位 (OU) の最大許可を指定する JSON ポリシーです。AWS Organizations は、顧客のビジネスが所有する複数の AWS アカウント をグループ化し、一元的に管理するサービスです。組織内のすべての機能を有効にすると、サービスコントロールポリシー (SCP) を一部またはすべてのアカウントに適用できます。SCP はメンバーアカウントのエンティティに対するアクセス許可を制限します (各 AWS アカウントのルートユーザー など)。Organizations と SCP の詳細については、AWS Organizations ユーザーガイドの「SCP の仕組み」を参照してください。
- セッションポリシー - セッションポリシーは、ロールまたはフェデレーティッドユーザーの一時的なセッションをプログラムで作成する際にパラメータとして渡す高度なポリシーです。結果としてセッションの許可される範囲は、ユーザーまたはロールのアイデンティティベースポリシーとセッションポリシーの共通部分になります。また、リソースベースのポリシーから許可が派生する場合もあります。これらのポリシーのいずれかを明示的に拒否した場合、許可は無効になります。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許可を理解するのがさらに難しくなります。複数のポリシータイプが関連するとき、リクエストを許可するかどうかを AWS が決定する方法の詳細については、「IAM ユーザーガイド」の「ポリシーの評価ロジック」を参照してください。

AWS サービスと IAM の連携の仕組み

AWS サービスが IAM のほとんどの機能と連携する仕組みの概要については、「IAM ユーザーガイド」の「IAM と連携する AWS のサービス」を参照してください。

特定の AWS サービスで IAM を使用する方法については、該当するサービスのユーザーガイドでセキュリティに関するセクションを参照してください。

AWS アイデンティティとアクセスに関するトラブルシューティング

次の情報は、AWS と IAM の使用に伴って発生する可能性がある一般的な問題の診断や修復に役立ちます。

トピック

- AWS でアクションを実行する権限がない
- iam:PassRole を実行する権限がありません
- 自分の AWS アカウント 以外のユーザーに AWS リソースへのアクセスを許可したい

AWS でアクションを実行する権限がない

ある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、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 管理者に問い合わせてください。サインイン資格情報を提供した担当者が管理者です。

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 サービスでは、新しいサービスロールやサービスにリンクされたロールを作成せずに、既存のロールを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許可が必要です。

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アクセス許可が必要です。Mary には、ロールをサービスに渡す許可がありません。

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

この場合、メアリー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 管理者に問い合わせてください。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 AWS アカウント 以外のユーザーに AWS リソース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ユーザーが、リソースへのアクセス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することができます。リソースベースのポリシーまたはアクセス制御リスト (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、以下をご参照ください。

- AWS がこれらの機能をサポートしているかどうかを確認するには、AWS サービスと IAM の連携の仕組み をご参照ください。
- 所有している AWS アカウント 全体のリソースへのアクセス権を提供する方法については、IAM ユーザーガイドの「所有している別の AWS アカウント アカウントへのアクセス権を IAM ユーザーに提供」を参照してください。
- サードパーティーの AWS アカウント にリソースへのアクセス権を提供する方法については、「IAM ユーザーガイド」の「第三者が所有する AWS アカウント へのアクセス権を付与する」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でのロールとリソースベースのポリシーの使用の違いの詳細については、「IAM ユーザーガイド」の「IAM ロールとリソースベースのポリシーとの相違点」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 サービス が特定のコンプライアンスプログラムの対象であるかどうかを確認するには、「コンプライアンスプログラムによる対象範囲内の AWS サービス のサービス」をご覧いただき、関心のあるコンプライアンスプログラムを選択してください。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。

AWS Artifact 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。詳細については、「AWS Artifact におけるダウンロードレポート」を参照してください。

AWS サービス を使用する際のユーザーのコンプライアンス責任は、ユーザーのデータの機密性や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。AWS では、コンプライアンスに役立つ次のリソースを提供しています。

- セキュリティとコンプライアンスのクイックスタートガイド — これらのデプロイガイドでは、アーキテクチャ上の考慮事項について説明し、セキュリティとコンプライアンスに重点を置いたベースライン環境を AWS にデプロイするためのステップを示します。
- 「Amazon Web Services での HIPAA のセキュリティとコンプライアンスのためのアーキテクチャ」 – このホワイトペーパーは、企業が AWS を使用して HIPAA 対象アプリケーションを作成する方法を説明しています。

> **Note**
>
> すべての AWS サービス が HIPAA 適格であるわけではありません。詳細については、「HIPAA 対応サービスのリファレンス」を参照してください。

- AWS コンプライアンスのリソース– このワークブックおよびガイドのコレクションは、顧客の業界と拠点に適用されるものである場合があります。
- AWS Config デベロッパーガイドのルールでのリソースの評価 - AWS Config サービスでは、自社のプラクティス、業界ガイドライン、および規制に対するリソースの設定の準拠状態を評価します。

- AWS Security Hub – この AWS サービス は、AWS 内のセキュリティ状態の包括的なビューを提供します。Security Hub では、セキュリティコントロールを使用して AWS リソースを評価し、セキュリティ業界標準とベストプラクティスに対するコンプライアンスをチェックします。サポートされているサービスとコントロールのリストについては、「Security Hub のコントロールリファレンス」を参照してください。
- AWS Audit Manager - この AWS サービス は AWS の使用状況を継続的に監査し、リスクの管理方法やコンプライアンスを業界スタンダードへの準拠を簡素化するために役立ちます。

この AWS 製品またはサービスは、サポートしている特定の Amazon Web Services (AWS) のサービスを通じて、責任共有モデルに従います。AWS サービスのセキュリティ情報については、「AWS のサービスのセキュリティに関するドキュメントページ」と、「AWS コンプライアンスプログラムごとのコンプライアンスの取り組みのAWS 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 のグローバルインフラストラクチャは AWS リージョン とアベイラビリティーゾーンを中心として構築されます。

AWS リージョン は、物理的に独立· 隔離されたアベイラビリティーゾーンがあり、低レイテンシー、高スループット、そして高冗長性のネットワークで接続されています。

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イ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、耐障害性、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、サポートしている特定の Amazon Web Services (AWS) のサービスを通じて、責任共有モデルに従います。AWS サービスのセキュリティ情報については、「AWS のサービスのセキュリティに関するドキュメントページ」と、「AWS コンプライアンスプログラムごとのコンプライアンスの取り組みのAWS 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しているため、AWS グローバルネットワークセキュリティによって保護されています。AWSセキュリティサービスと AWS がインフラストラクチャ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、「セキュリティの柱 - AWS Well-Architected Framework」の「インフラストラクチャ保護」を参照してください。

AWS の公開された API コールを使用し、ネットワークを介してこの AWS 製品またはサービスにアクセスします。クライアントは以下をサポートする必要があります。

- Transport Layer Security (TLS) TLS 1.2 および TLS 1.3 をお勧めします。
- DHE (Ephemeral Diffie-Hellman) や ECDHE (Elliptic Curve Ephemeral Diffie-Hellman) などの Perfect Forward Secrecy (PFS) を使用した暗号スイートです。これらのモードは、Java 7 以降など、最近のほとんどのシステムでサポートされています。

また、リクエストは、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、サポートしている特定の Amazon Web Services (AWS) のサービスを通じて、責任共有モデルに従います。AWS サービスのセキュリティ情報については、「AWS のサービスのセキュリティに関するドキュメントページ」と、「AWS コンプライアンスプログラムごとのコンプライアンスの取り組みのAWS 対象となるサービス」を参照してください。

AWS Toolkit for JetBrains ユーザーガイドのドキュメント履歴

次の表に、AWS Toolkit for JetBrains ユーザーガイドの主要なドキュメントの更新を示します。

AWS Toolkit for JetBrains の変更点の詳細なリストについては、GitHub ウェブサイトの aws/aws-Toolkit-Jetbrains リポジトリ内にある .changes ディレクトリを参照してください。

変更	説明	日付
Amazon Elastic Container Service Exec のユーザーガイドを作成しました	これは Amazon ECS Exec の概要です。	2021 年 12 月 16 日
実験的機能のサポート	AWS サービスの実験的機能をオンにするサポートが追加されました。	2021 年 10 月 14 日
AWS リソースのサポート	リソースを作成、編集、および削除するインターフェイスオプションと、リソースタイプにアクセスするためのサポートが追加されました。	2021 年 10 月 14 日
AWS App Runner が利用可能になりました	AWS Toolkit for JetBrains を使用して App Runner を操作し、ソースコードまたはコンテナイメージから、AWS クラウド内のスケーラブルでセキュアなウェブアプリケーションを直接デプロイします。	2021 年 5 月 26 日
サーバーレスアプリケーションで Lambda コンテナイメージを利用可能になりました	AWS Toolkit を使用して、サーバーレスアプリケーションで AWS Lambda コンテナイメージを利用できるようになりました。	2020 年 12 月 1 日

CloudWatch Logs Insights の使用が可能になりました	AWS Toolkit for JetBrains を使用して Amazon CloudWatch Logs を利用できるようになりました。	2020 年 11 月 24 日
Amazon SQS が使用可能になりました	AWS Toolkit for JetBrains を使用して、Amazon Simple Queue Service (Amazon SQS) を利用できるようになりました。	2020 年 11 月 24 日
Amazon RDS と Amazon Redshift が利用可能になりました	AWS Toolkit を使用して、Amazon Relational Database Service (Amazon RDS) と Amazon Redshift を利用できるようになりました。	2020 年 9 月 23 日
IAM Identity Center がサポート可能になりました	AWS IAM Identity Center が AWS Toolkit でサポートされるようになりました。	2020 年 9 月 23 日
AWS Toolkit をさらに 4 つの JetBrains IDE で利用できるようになりました	AWS Toolkit が追加された 4 つの JetBrains IDE のプラグインとして利用できるようになりました。 • CLion 用 AWS Toolkit (C と C++ 開発用) • GoLand 用 AWS Toolkit (Go 開発用) • PhpStorm 用 AWS Toolkit (PHP 開発用) • RubyMine 用 AWS Toolkit (Ruby 開発用)	2020 年 5 月 28 日

CloudWatch Logs の使用が可能になりました	Amazon CloudWatch Logs で作業するための AWS Toolkit の使用が利用可能になりました。	2020 年 4 月 15 日
Amazon S3 バケットとオブジェクトが利用可能になりました	Amazon Simple Storage Service (Amazon S3) のバケットとオブジェクトで作業をするための AWS Toolkit の使用が利用可能になりました。	2020 年 3 月 27 日
EventBridge スキーマの使用が利用可能になりました	AWS Toolkit を使用して Amazon EventBridge スキーマで作業できるようになりました。	2019 年 12 月 2 日
Amazon ECS クラスター内のデバッグコードがベータ版で利用可能になりました	Amazon Elastic Container Service (Amazon ECS) クラスター内のコードをデバッグするための AWS Toolkit の使用が、ベータ版で利用可能になりました。	2019 年 11 月 25 日
Rider 用 AWS Toolkit が利用可能になりました	Rider 用 AWS Toolkit が利用可能になりました。	2019 年 11 月 25 日
WebStorm 用の AWS Toolkit が利用可能になりました	WebStorm 用の AWS Toolkit が利用可能になりました。	2019 年 10 月 23 日
AWS Toolkit for IntelliJ の一般提供を開始	AWS Toolkit for IntelliJ が一般公開されました。それに応じて、対応するドキュメントが更新されました。	2019 年 3 月 27 日

初回リリース	これは、「AWS Toolkit for JetBrains ユーザーガイド」の初回リリースです。AWS Toolkit for PyCharm が一般公開されました。AWS Toolkit for IntelliJ はまだ開発者プレビュー段階です。	2018 年 11 月 27 日