ユーザーガイド

AWS CodeBuild

API バージョン 2016-10-06

AWS CodeBuild: ユーザーガイド

Table of Contents

AWS CodeBuild とは

AWS CodeBuild は、cloud. CodeBuild compiles のフルマネージド型のビルドサービスです。ソースコードをコンパイルし、ユニットテストを実行して、すぐにデプロイできるアーティファクトを生成します。 CodeBuild は、独自のビルドサーバーをプロビジョニング、管理、スケーリングする必要性を排除します。Apache Maven、Gradle などの一般的なプログラミング言語とビルドツール用のパッケージ済みのビルド環境を提供します。でビルド環境をカスタマイズ CodeBuild して、独自のビルドツールを使用することもできます。 はピーク時のビルドリクエストに合わせて自動的に CodeBuild スケーリングします。

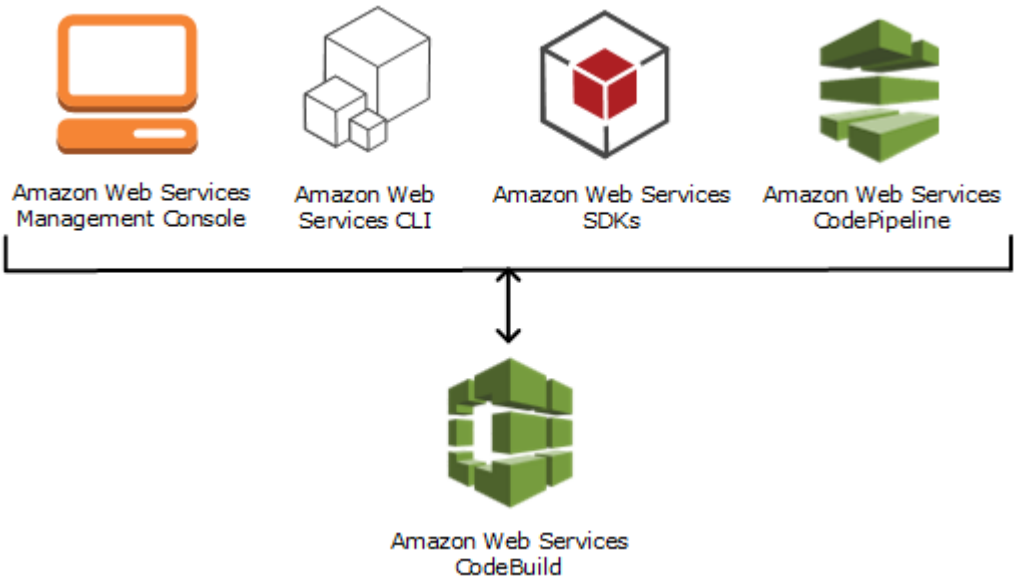
CodeBuild には次の利点があります。

- 完全マネージド型 – CodeBuild 独自のビルドサーバーのセットアップ、パッチ適用、更新、管理の必要性を排除します。
- オンデマンド — ビルドのニーズに合わせてオンデマンドで CodeBuild スケーリングします。料金は、使用したビルド分数に対してのみ発生します。
- すぐに使える - 最も一般的なプログラミング言語用に事前設定されたビルド環境 CodeBuild を提供します。最初のビルドを開始するには、ビルドスクリプトを指すだけです。

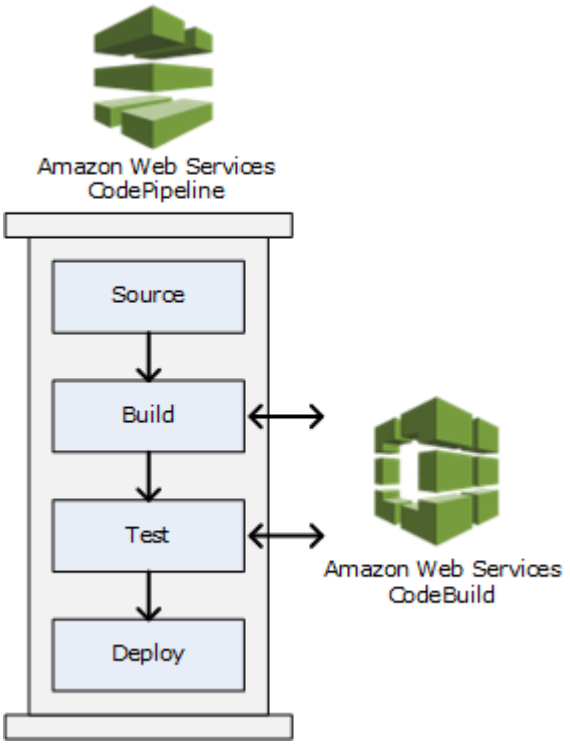
詳細については、「AWS CodeBuild」を参照してください。

の実行方法 CodeBuild

CodeBuild を実行するには、AWS CodeBuild コンソールまたは AWS CodePipeline コンソールを使用できます AWS Command Line Interface (AWS CLI) または AWS SDK CodeBuild を使用して、 の実行を自動化することもできます。 SDKs



次の図に示すように、 のパイプラインのビルドまたはテストステージにビルドまたはテストアクション CodeBuild として を追加できますAWS CodePipeline。 AWS CodePipeline は、コードをリリースするために必要なステップをモデル化、視覚化、および自動化するために使用できる継続的な配信サービスです。これには、コードの構築が含まれます。パイプラインは、リリースプロセスを通したコードの変更を説明したワークフロー構造です。



CodePipeline を使用してパイプラインを作成し、 CodeBuild ビルドまたはテストアクションを追加するには、「」を参照してください CodePipeline で を使用する CodeBuild。の詳細については CodePipeline、「 AWS CodePipelineユーザーガイド」を参照してください。

CodeBuild コンソールでは、リポジトリ、ビルドプロジェクト、デプロイアプリケーション、パイプラインなどのリソースをすばやく検索することもできます。[Go to resource] を選択するか、/ キーを押して、リソースの名前を入力します。一致するものはすべてリストに表示されます。検索では大文字と小文字が区別されません。リソースを表示する権限がある場合のみ表示されます。詳細については、「コンソールでのリソースの表示」を参照してください。

の料金 CodeBuild

詳細については、CodeBuild 「 の料金」を参照してください。

の使用を開始するにはどうすればよいですか CodeBuild？

次の手順を実行することをお勧めします。

1. の詳細については CodeBuild 、「」の情報を参照してください概念。
2. 「」の手順に従って、シナリオ例 CodeBuild で を試しますコンソールを使用した開始方法。
3. 「」の手順に従って、独自のシナリオで を使用します CodeBuild ビルドを計画する。

AWS CodeBuild の概念

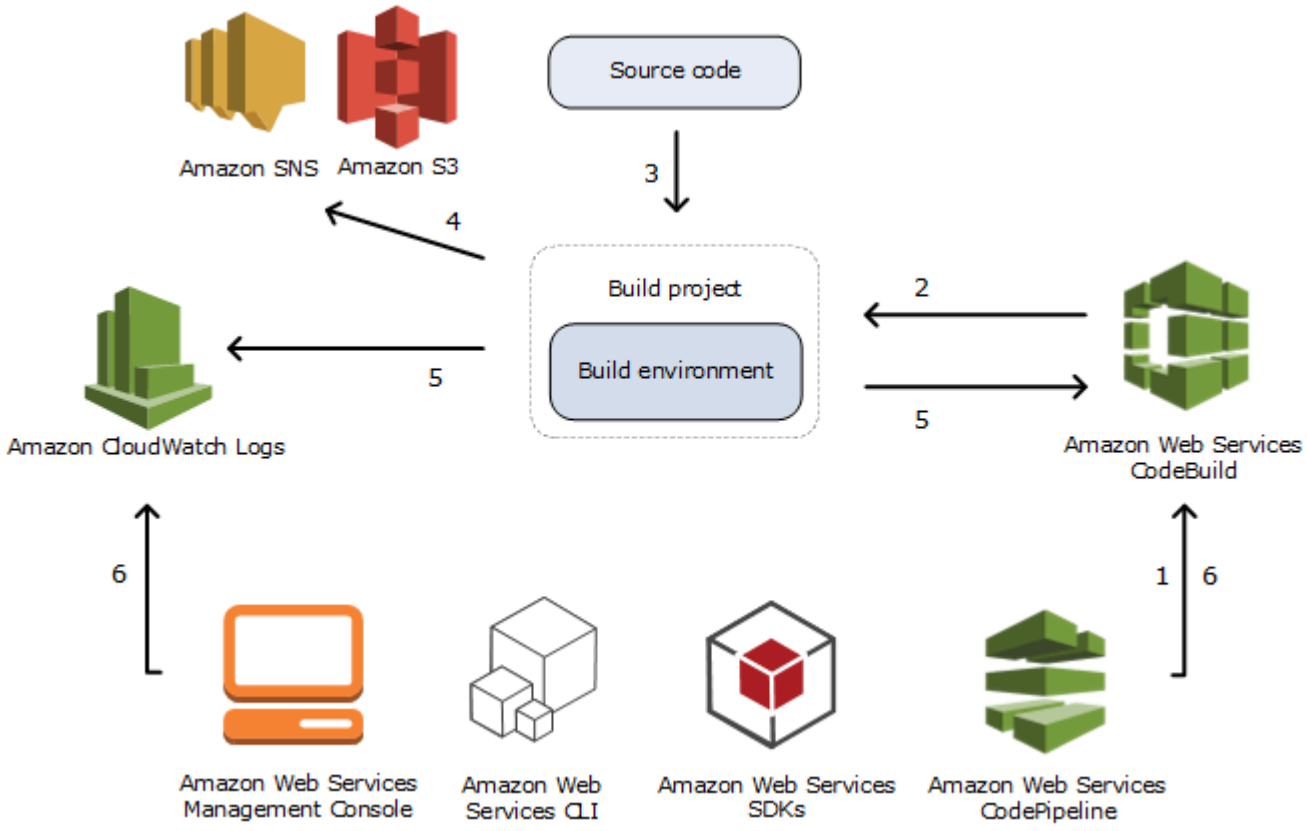
以下の概念は、CodeBuild の仕組みを理解するうえで重要です。

トピック

- CodeBuild の仕組み
- 次のステップ

CodeBuild の仕組み

次の図は、CodeBuild でビルドを実行するとどうなるかを示しています。



1. 入力として、CodeBuild にビルドプロジェクトを指定する必要があります。ビルドプロジェクトには、ビルドの実行方法に関する情報が含まれています。これには、ソースコードの取得先、使用するビルド環境、実行するビルドコマンド、ビルド出力の格納先が含まれます。ビルド環境は、CodeBuild がビルドを実行するために使用するオペレーティングシステム、プログラミング言語ランタイム、およびツールの組み合わせを表します。詳細については、以下を参照してください。

   - ビルドプロジェクトの作成
   - ビルド環境に関するリファレンス

2. CodeBuild は、ビルドプロジェクトを使用して、ビルド環境を作成します。
3. CodeBuild は、ビルド環境にソースコードをダウンロードし、ビルドプロジェクトで定義されている、または、ソースコードに直接含まれているビルド仕様 (buildspec) を使用します。ビルド環境は、CodeBuild がビルドを実行するために使用するオペレーティングシステム、プログラミング言語ランタイム、およびツールの組み合わせを表します。詳細については、「ビルド仕様 (buildspec) に関するリファレンス」を参照してください。
4. ビルド出力がある場合、ビルド環境はその出力を S3 バケットにアップロードします。ビルド環境では、buildspec で指定したタスク (たとえば、ビルド通知を Amazon SNS トピックに送信す

るなど) を実行することもできます。例については、「ビルド通知サンプル」を参照してください。

5. ビルドが実行されている間に、ビルド環境は CodeBuild および Amazon CloudWatch Logs に情報を送信します。
6. ビルドが実行されている間は、AWS CodeBuild コンソール、AWS CLI、または AWS SDK を使用して、CodeBuild の要約されたビルド情報および Amazon CloudWatch Logs の詳細なビルド情報を取得できます。AWS CodePipeline を使用してビルドを実行する場合は、CodePipeline から制限されたビルド情報を取得できます。

次のステップ

AWS CodeBuild の詳細を確認した後で、次の手順をお勧めします。

1. 「コンソールを使用した開始方法」の手順に従って、サンプルのシナリオで CodeBuild を試してみてください。
2. 自分のシナリオで CodeBuild を使用するには、「ビルドを計画する」の手順に従います。

CodeBuild の開始方法

次のチュートリアルでは、AWS CodeBuild を使用し、サンプルソースコード入力ファイルのコレクションから、ソースコードのデプロイ可能バージョンを生成します。

どちらのチュートリアルでも入力と結果は同じですが、一方では AWS CodeBuild コンソールを使用し、他方では AWS CLI を使用します。

> **Important**
>
> このチュートリアルを完了するために AWS ルートアカウントは使用しないでください。

コンソールを使用した AWS CodeBuild の開始方法

このチュートリアルでは、AWS CodeBuild を使用して、サンプルのソースコード入力ファイル (ビルド入力アーティファクトまたはビルド入力) のコレクションから、ソースコードのデプロイ可能バージョン (ビルド出力アーティファクトまたはビルド出力) を生成します。具体的には、一般的なビルドツールである Apache Maven CodeBuild を使用して、一連の Java クラスファイルを Java Archive (JAR) ファイルにビルドするように に指示します。このチュートリアルを完了するために、Apache Maven または Java に精通している必要はありません。

コンソール CodeBuild 、、AWS CodePipeline、AWS CLIまたは AWS SDKs CodeBuild を使用して を操作できます。このチュートリアルでは、 CodeBuild コンソールの使用方法を示します。CodePipeline の使用の詳細については、「 CodePipeline で を使用する CodeBuild」を参照してください。

> **Important**
>
> このチュートリアルのステップでは、AWS アカウントに課金される可能性のあるリソース (S3 バケットなど) を作成する必要があります。これには、Amazon S3、、 CodeBuild および CloudWatch Logs に関連するAWSリソースとアクションに対する AWS KMSおよびの料金が含まれます。詳細については、「 のAWS CodeBuild料金」、「Amazon S3 の料金」、AWS Key Management Service「 の料金」、および「Amazon の CloudWatch 料金」を参照してください。

ステップ

ステップ 1: ソースコードを作成する

(一部: コンソールを使用した AWS CodeBuild の開始方法)

このステップでは、出力バケットに CodeBuild ビルドするソースコードを作成します。このソースコードは 2 つの Java クラスファイルと Apache Maven プロジェクトオブジェクトモデル (POM) ファイルで構成されています。

1. ローカルコンピュータまたはインスタンスの空のディレクトリに、このディレクトリ構造を作成します。

```
(root directory name)
    `-- src
         |-- main
         |     `-- java
         `-- test
               `-- java
```

2. 任意のテキストエディタを使用して、このファイルを作成し、`MessageUtil.java` という名前を付けて、`src/main/java` ディレクトリに保存します。

```
public class MessageUtil {
  private String message;
```

```
  public MessageUtil(String message) {
    this.message = message;
  }

  public String printMessage() {
    System.out.println(message);
    return message;
  }

  public String salutationMessage() {
    message = "Hi!" + message;
    System.out.println(message);
    return message;
  }
}
```

このクラスファイルは、渡された文字列を出力として作成します。MessageUtil コンストラクタは、文字列を設定します。printMessage メソッドは出力を作成します。salutationMessage メソッドが Hi! を出力した後に文字列が続きます。

3. このファイルを作成し、TestMessageUtil.java という名前を付けて、/src/test/java ディレクトリに保存します。

```
import org.junit.Test;
import org.junit.Ignore;
import static org.junit.Assert.assertEquals;

public class TestMessageUtil {

  String message = "Robert";
  MessageUtil messageUtil = new MessageUtil(message);

  @Test
  public void testPrintMessage() {
    System.out.println("Inside testPrintMessage()");
    assertEquals(message,messageUtil.printMessage());
  }

  @Test
  public void testSalutationMessage() {
    System.out.println("Inside testSalutationMessage()");
    message = "Hi!" + "Robert";
```

```
    assertEquals(message,messageUtil.salutationMessage());
  }
}
```

このクラスファイルは message クラスの MessageUtil 変数を Robert に設定します。その後、文字列 message および Robert が出力に表示されているかどうかを調べることによって、Hi!Robert 変数が正常に設定されたかどうかを調べます。

4. このファイルを作成し、pom.xml という名前を付けて、ルート (最上位) ディレクトリに保存します。

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/
maven-v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>org.example</groupId>
  <artifactId>messageUtil</artifactId>
  <version>1.0</version>
  <packaging>jar</packaging>
  <name>Message Utility Java Sample App</name>
  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>4.11</version>
      <scope>test</scope>
    </dependency>
  </dependencies>
  <build>
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-compiler-plugin</artifactId>
        <version>3.8.0</version>
      </plugin>
    </plugins>
  </build>
</project>
```

Apache Maven では、このファイルの指示に従って、MessageUtil.java および TestMessageUtil.java ファイルを messageUtil-1.0.jar という名前のファイルに変換し、指定されたテストを実行します。

この時点で、ディレクトリ構造は次のようになります。

```
(root directory name)
    |-- pom.xml
    `-- src
         |-- main
         |     `-- java
         |           `-- MessageUtil.java
         `-- test
               `-- java
                     `-- TestMessageUtil.java
```

次のステップ

ステップ 2: buildspec ファイルを作成する

ステップ 2: buildspec ファイルを作成する

(前のステップ: ステップ 1: ソースコードを作成する)

このステップでは、ビルド仕様ファイルを作成します。buildspec は、 がビルドの実行 CodeBuild に使用する YAML 形式のビルドコマンドと関連設定のコレクションです。ビルド仕様がないと、ビルド入力をビルド出力に正常に変換したり、ビルド環境でビルド出力アーティファクトを見つけて出力バケットにアップロードしたり CodeBuild することはできません。

このファイルを作成し、buildspec.yml という名前を付けて、ルート (最上位) ディレクトリに保存します。

```
version: 0.2

phases:
  install:
    runtime-versions:
      java: corretto11
  pre_build:
    commands:
```

```
      - echo Nothing to do in the pre_build phase...
  build:
    commands:
      - echo Build started on `date`
      - mvn install
  post_build:
    commands:
      - echo Build completed on `date`
artifacts:
  files:
    - target/messageUtil-1.0.jar
```

Important

ビルド仕様宣言は有効な YAML である必要があるため、ビルド仕様宣言のスペースは重要です。ビルド仕様宣言のスペース数が YAML と一致しない場合、ビルドは即座に失敗する場合があります。YAML validator を使用して、ビルド仕様宣言が有効な YAML かどうかをテストできます。

Note

ソースコードにビルド仕様ファイルを含める代わりに、ビルドプロジェクトを作成するときに個別にビルドコマンドを宣言することができます。これは、毎回ソースコードのリポジトリを更新せずに、異なるビルドコマンドでソースコードをビルドする場合に役立ちます。詳細については、「buildspec の構文」を参照してください。

このビルド仕様宣言の詳細は次の通りです。

- `version` は、使用されているビルド仕様スタンダードのバージョンを表します。このビルド仕様宣言では、最新バージョン `0.2` が使用されます。
- `phases` は、 CodeBuild にコマンドの実行を指示するビルドフェーズを表します。これらのビルドフェーズは `install`、`pre_build`、`build`、`post_build` として、ここにリストされています。これらのビルドフェーズ名のスペルを変更することはできず、追加のビルドフェーズ名を作成することもできません。

  この例では、 `build`フェーズ中に、 は `mvn install` コマンド CodeBuild を実行します。このコマンドは、コンパイルされた Java クラスファイルをビルド出力アーティファクトにコンパイ

ル、テスト、パッケージ化するように Apache Maven に指示します。完全にするために、この例では、いくつかの echo コマンドが各ビルドフェーズに配置されています。このチュートリアルの後半で詳細なビルド情報を見る際に、これらの echo コマンドの出力は、 CodeBuild がコマンドを実行する方法とその順序を理解するのに役立ちます。(この例にはすべてのビルドフェーズが含まれていますが、コマンドを実行しないビルドフェーズは含める必要がありません)。ビルドフェーズごとに、 は指定された各コマンドを、最初から最後まで、リストされた順序で 1 つずつ CodeBuild 実行します。

- `artifacts` は、 が CodeBuild出力バケットにアップロードするビルド出力アーティファクトのセットを表します。 は、ビルド出力に含めるファイル`files`を表します。 CodeBuild は、ビルド環境の`target`相対ディレクトリにある単一の`messageUtil-1.0.jar`ファイルをアップロードします。ファイル名 `messageUtil-1.0.jar` およびディレクトリ名 `target` は、この例でのみ Apache Maven がビルド出力アーティファクトを作成して格納する方法に基づいています。独自のビルドでは、これらのファイル名とディレクトリは異なります。

詳細については、「ビルド仕様 (buildspec) に関するリファレンス」を参照してください。

この時点で、ディレクトリ構造は次のようになります。

```
(root directory name)
    |-- pom.xml
    |-- buildspec.yml
    `-- src
         |-- main
         |     `-- java
         |           `-- MessageUtil.java
         `-- test
               `-- java
                     `-- TestMessageUtil.java
```

次のステップ

ステップ 3: 2 つの S3 バケットを作成する

ステップ 3: 2 つの S3 バケットを作成する

(前のステップ: ステップ 2: buildspec ファイルを作成する)

このチュートリアル用として 1 つのバケットを使用することもできますが、2 つのバケットを使用する方が、ビルド入力の送信元およびビルド出力の送信先の確認が簡単になります。

- これらのバケットのいずれか (入力バケット) にビルド入力が保存されます。このチュートリアルでは、この入力バケットの名前は codebuild-*region-ID*-*account-ID*-input-bucket です (*region-ID* はバケットの AWS リージョン、*account-ID* は AWS アカウント ID です)。
- もう 1 つのバケット (出力バケット) にはビルド出力が保存されます。このチュートリアルでは、この出力バケットの名前は codebuild-*region-ID*-*account-ID*-output-bucket です。

これらのバケットに別の名前を選択した場合は、このチュートリアル全体で、その名前を使用してください。

これらの 2 つのバケットは、ビルドと同じ AWS リージョン内にある必要があります。例えば、米国東部 (オハイオ) リージョンでビルドを実行する CodeBuild ように に指示する場合、これらのバケットも米国東部 (オハイオ) リージョンにある必要があります。

詳細については、Amazon Simple Storage Service コンソールユーザーガイドの「バケットの作成」を参照してください。

> **Note**
>
> は、 CodeCommit、 GitHub、および Bitbucket リポジトリに保存されているビルド入力 CodeBuild もサポートしていますが、このチュートリアルでは、それらの使用方法については説明しません。詳細については、「ビルドを計画する」を参照してください。

次のステップ

ステップ 4: ソースコードと buildspec ファイルをアップロードする

ステップ 4: ソースコードと buildspec ファイルをアップロードする

(前のステップ: ステップ 3: 2 つの S3 バケットを作成する)

このステップでは、入力バケットにソースコードとビルド仕様ファイルを追加します。

オペレーティングシステムの zip ユーティリティを使用して、MessageUtil.zip、MessageUtil.java、TestMessageUtil.java、および pom.xml を含む buildspec.yml という名前のファイルを作成します。

MessageUtil.zip ファイルのディレクトリ構造は、次のようになっている必要があります。

```
MessageUtil.zip
    |-- pom.xml
    |-- buildspec.yml
    `-- src
         |-- main
         |     `-- java
         |           `-- MessageUtil.java
         `-- test
               `-- java
                     `-- TestMessageUtil.java
```

> **Important**
>
> *(root directory name)* ディレクトリを含めないでください。*(root directory name)* ディレクトリ内のディレクトリとファイルのみを含めます。

MessageUtil.zip ファイルを codebuild-*region-ID*-*account-ID*-input-bucket という名前の入力バケットにアップロードします。

> **Important**
>
> CodeCommit、 GitHub、および Bitbucket リポジトリの場合、規則により、 という名前のビルド仕様ファイルを各リポジトリのbuildspec.ymlルート (最上位) に保存するか、ビルド仕様宣言をビルドプロジェクト定義の一部として含める必要があります。リポジトリのソースコードとビルド仕様ファイルを含む ZIP ファイルを作成しないでください。
> S3 バケットに保存されたビルド入力に限り、ソースコードおよび規約に基づく buildspec.yml というビルド仕様ファイルを圧縮した ZIP ファイルをルート (最上位) に作成するか、ビルド仕様宣言をビルドプロジェクト定義の一部として含める必要があります。
> ビルド仕様ファイルに別の名前を使用するか、ルート以外の場所でビルド仕様を参照する場合は、ビルドプロジェクト定義の一部としてビルド仕様の上書きを指定できます。詳細については、「buildspec ファイル名とストレージの場所」を参照してください。

次のステップ

ステップ 5: ビルドプロジェクトを作成する

ステップ 5: ビルドプロジェクトを作成する

(前のステップ: ステップ 4: ソースコードと buildspec ファイルをアップロードする)

このステップでは、AWS CodeBuild がビルドの実行に使用するビルドプロジェクトを作成します。ビルドプロジェクトには、ビルドの実行方法に関する情報が含まれています。これには、ソースコードの取得先、使用するビルド環境、実行するビルドコマンド、ビルド出力の格納先が含まれます。ビルド環境は、 がビルドの実行 CodeBuild に使用するオペレーティングシステム、プログラミング言語ランタイム、およびツールの組み合わせを表します。ビルド環境は Docker イメージとして表されます。詳細については、Docker Docs ウェブサイトの Docker overview を参照してください。

このビルド環境では、Java Development Kit (JDK) と Apache Maven のバージョンを含む Docker イメージを使用する CodeBuild ように に指示します。

ビルドプロジェクトを作成するには

1. AWS Management Console にサインインして、AWS CodeBuild コンソール (https://console.aws.amazon.com/codesuite/codebuild/home) を開きます。
2. AWS リージョンセレクタを使用して、 CodeBuild がサポートされている AWSリージョンを選択します。詳細については、「Amazon Web Services 全般のリファレンス」の「AWS CodeBuild エンドポイントとクォータ」を参照してください。
3. CodeBuild 情報ページが表示されている場合は、ビルドプロジェクトの作成を選択します。それ以外の場合は、ナビゲーションペインでビルドを展開し、[ビルドプロジェクト] を選択し、次に [Create build project (ビルドプロジェクトの作成)] を選択します。
4. [Create build project (ビルドプロジェクトの作成)] ページで、[Project configuration (プロジェクト設定)] の [プロジェクト名] にこのビルドプロジェクトの名前を入力します (この例では、`codebuild-demo-project`)。ビルドプロジェクトの名前は、各 AWS アカウントで一意である必要があります。別の名前を選択した場合は、このチュートリアル全体でその名前を使用してください。

> **Note**
>
> [Create build project (ビルドプロジェクトの作成)] ページに「このオペレーションを実行する権限がありません」というようなエラーメッセージが表示される場合があります。これは、ビルドプロジェクトを作成するアクセス許可を持たないユーザーとして AWS Management Console にサインインしたことが原因と考えられます。これを修正するには、AWS Management Console からサインアウトし、次の IAM エンティティのいずれかに属する認証情報でサインインし直します。

- AWS アカウントの管理者ユーザー。詳細については、ユーザーガイドの「最初の AWS アカウントルートユーザーおよびグループの作成」を参照してください。
- ユーザー (またはユーザーが属する IAM グループ) に `AWSCodeBuildAdminAccess`、`AmazonS3ReadOnlyAccess`、および `IAMFullAccess` 管理ポリシーがアタッチされている、AWS アカウントのユーザー。これらのアクセス許可を持つユーザーやグループが AWS アカウントに存在せず、これらのアクセス許可をユーザーやグループに追加できない場合は、AWS アカウント管理者に連絡してサポートを依頼してください。詳細については、「AWS のマネージド (事前定義) ポリシー AWS CodeBuild」を参照してください。

どちらのオプションにも、このチュートリアルを完了できるように、ビルドプロジェクトの作成を許可する管理者権限が含まれています。常に必要最小限のアクセス許可を使用してタスクを達成してください。詳細については、「AWS CodeBuild アクセス許可リファレンス」を参照してください。

5. [Source] (ソース) で、[Source provider] (ソースプロバイダー) として [Amazon S3] を選択します。
6. [Bucket] (バケット) として、[codebuild-***region-ID*-*account-ID***-input-bucket] を選択します。
7. [S3 オブジェクトキー] に「**`MessageUtil.zip`**」と入力します。
8. [環境] の [環境イメージ] で、[Managed image (マネージド型イメージ)] を選択したままにしておきます。
9. [オペレーティングシステム] で、[Amazon Linux 2] を選択します。
10. [ランタイム] で、[Standard (標準)] を選択します。
11. [イメージ] で、[aws/codebuild/amazonlinux2-x86_64-standard:4.0] を選択します。
12. [サービスロール] で、[New service role (新しいサービスロール)] は選択したままにして、[Role name (ロール名)] は変更しません。
13. [Buildspec] で、[Use a buildspec file (buildspec ファイルを使用)] を選択したままにしておきます。
14. [Artifacts] (アーティファクト) で、[Type] (タイプ) として [Amazon S3] を選択します。
15. [Bucket name] (バケット名) として、[codebuild-***region-ID*-*account-ID***-output-bucket] を選択します。
16. [名前] と [パス] を空白のままにします。
17. [Create build project (ビルドプロジェクトの作成)] を選択します。

次のステップ

ステップ 6: ビルドを実行する

ステップ 6: ビルドを実行する

(前のステップ: ステップ 5: ビルドプロジェクトを作成する)

このステップでは、ビルドプロジェクトの設定でビルドを実行するように AWS CodeBuild に指示します。

ビルドを実行するには

1. AWS CodeBuild コンソール (https://console.aws.amazon.com/codesuite/codebuild/home) を開きます。
2. ナビゲーションペインで、[Build projects] を選択します。
3. ビルドプロジェクトのリストで、 を選択しcodebuild-demo-project、ビルドの開始 を選択します。ビルドがすぐに開始されます。

次のステップ

ステップ 7: ビルド情報の要約を表示する

ステップ 7: ビルド情報の要約を表示する

(前のステップ: ステップ 6: ビルドを実行する)

このステップでは、ビルドのステータスに関する要約情報を表示します。

要約されたビルド情報を表示するには

1. codebuild-demo-project : ***<build-ID>*** ページが表示されない場合は、ナビゲーションバーでビルド履歴 を選択します。次に、ビルドプロジェクトのリストで、プロジェクト で、 のビルド実行リンクを選択しますcodebuild-demo-project。一致するリンクは 1 つだけです。(このチュートリアルを完了したことがある場合は、[完了済み] 列で最新の値のリンクを選択します。)
2. [Build status] (ビルドステータス) ページの [Phase details] (フェーズ詳細) に以下のビルドフェーズが表示され、[Status] (ステータス) 列に [Succeeded] (成功) と示されます。

   - SUBMITTED

- QUEUED
- PROVISIONING
- DOWNLOAD_SOURCE
- INSTALL
- PRE_BUILD
- BUILD
- POST_BUILD
- UPLOAD_ARTIFACTS
- FINALIZING
- COMPLETED

[ビルドステータス] で、[Succeeded (成功)] が表示されます。

代わりに [進行中] と表示される場合は、更新ボタンを選択します。

3. 各ビルドフェーズの横に表示される [所要時間] 値は、ビルドフェーズの所要時間を示します。[終了時間] 値は、ビルドフェーズの完了日時を示します。

次のステップ

ステップ 8: 詳細なビルド情報を表示する

ステップ 8: 詳細なビルド情報を表示する

(前のステップ: ステップ 7: ビルド情報の要約を表示する)

このステップでは、 CloudWatch ログでビルドに関する詳細情報を表示します。

> **Note**
>
> 機密情報を保護するために、ログでは以下が非表示になっています CodeBuild 。
>
> - AWS アクセスキー ID。詳細については、AWS Identity and Access Management ユーザーガイドの IAM ユーザーのアクセスキーの管理を参照してください。
> - パラメータストアを使用して指定された文字列。詳細については、「Amazon EC2 Systems Manager ユーザーガイド」の「Systems Manager パラメータストア」および

「Systems Manager パラメータストアコンソールのチュートリアル」を参照してください。

- AWS Secrets Manager を使用して指定された文字列。詳細については、「キー管理」を参照してください。

詳細なビルド情報を表示するには

1. ビルドの詳細ページが前のステップから引き続き表示された状態で、ビルドログの最後の 10,000 行が [Build logs] に表示されます。Logs でビルドログ全体を表示するには、 CloudWatch ログ全体を表示リンクを選択します。
2. CloudWatch Logs ログストリームでは、ログイベントを参照できます。デフォルトでは、ログイベントの最後のセットだけが表示されます。以前のログイベントを表示するには、リストの先頭にスクロールします。
3. このチュートリアルでは、ほとんどのログイベントに、 CodeBuild でビルド依存ファイルをビルド環境にダウンロードおよびインストールする操作に関する詳細情報が含まれますが、これらの情報は不要な場合があります。[Filter events] ボックスを使用すると、表示する情報量を減らすことができます。例えば、[Filter events] (イベントのフィルター) で「`"[INFO]"`」と入力した場合、「`[INFO]`」を含むイベントのみが表示されます。詳細については、「Amazon ユーザーガイド」の「フィルターとパターン構文」を参照してください。 CloudWatch

次のステップ

ステップ 9: ビルド出力アーティファクトを取得する

ステップ 9: ビルド出力アーティファクトを取得する

(前のステップ: ステップ 8: 詳細なビルド情報を表示する)

このステップでは、 を CodeBuild ビルドして出力バケットにアップロードした `messageUtil-1.0.jar` ファイルを取得します。

このステップを完了するには、 CodeBuild コンソールまたは Amazon S3 コンソールを使用します。

ビルド出力アーティファクトを取得するには (AWS CodeBuild コンソール)

1. CodeBuild コンソールが開いたままで、前のステップのビルドの詳細ページが表示されたら、ビルドの詳細タブを選択し、アーティファクトセクションまでスクロールします。

> **Note**
>
> ビルドの詳細ページが表示されていない場合は、ナビゲーションバーで [Build history] (ビルド履歴)、[Build run] (ビルドの実行) リンクの順に選択します。

2. Amazon S3 フォルダへのリンクは、[Artifacts upload location] (アーティファクトのアップロード場所) の下にあります。Amazon S3 内のフォルダが開き、「`messageUtil-1.0.jar`」という名前のビルド出力アーティファクトファイルを見つけます。

ビルド出力アーティファクトを取得するには (Amazon S3 コンソール)

1. https://console.aws.amazon.com/s3/でAmazon S3 コンソールを開きます。
2. `codebuild-`*`region-ID`*`-`*`account-ID`*`-output-bucket` を開きます。
3. `codebuild-demo-project` フォルダを開きます。
4. `target` という名前のフォルダを開き、`messageUtil-1.0.jar` という名前のビルド出力アーティファクトファイルを見つけます。

次のステップ

ステップ 10: S3 入力バケットを削除する

ステップ 10: S3 入力バケットを削除する

(前のステップ: ステップ 9: ビルド出力アーティファクトを取得する)

AWS アカウントで継続的に料金が発生しないように、このチュートリアルで使用した入力・出力バケットを削除することもできます。手順については、Amazon Simple Storage Service ユーザーガイドでバケットを削除、または空にする方法を参照してください。

IAM ユーザー を使用している場合、このバケットを削除するユーザーまたは管理者 IAM ユーザーには、さらに高いアクセス権限が必要です。マーカー間で次のステートメント (*`###BEGIN ADDING STATEMENT HERE###`* と *`###END ADDING STATEMENTS HERE###`*) を既存のアクセスポリシーに追加します。

このステートメントでは、簡潔にするために省略記号 (...) が使用されています。既存のアクセスポリシーのステートメントは削除しないでください。これらの省略記号はポリシーに入力しないでください。

```
{
  "Version": "2012-10-17",
  "Id": "...",
  "Statement": [
    ### BEGIN ADDING STATEMENT HERE ###
    {
      "Effect": "Allow",
      "Action": [
        "s3:DeleteBucket",
        "s3:DeleteObject"
      ],
      "Resource": "*"
    }
    ### END ADDING STATEMENT HERE ###
  ]
}
```

次のステップ

まとめ

まとめ

このチュートリアルでは、AWS CodeBuild を使用して、一連の Java クラスファイルから JAR ファイルを作成しました。次に、ビルドの結果を表示しました。

独自のシナリオ CodeBuild で を使用できるようになりました。「ビルドを計画する」の手順に従ってください。もう少し準備が必要な場合は、用意されているサンプルでビルドを試すことができます。詳細については、「サンプル」を参照してください。

AWS CodeBuild を使用した AWS CLI の開始方法

このチュートリアルでは、AWS CodeBuild を使用して、サンプルのソースコード入力ファイル (ビルド入力アーティファクトまたはビルド入力と呼ばれる) のコレクションから、ソースコードのデプロイ可能バージョン (ビルド出力アーティファクトまたはビルド出力と呼ばれる) を生成します。具体的には、一般的なビルドツールである Apache Maven CodeBuild を使用して、一連の Java クラスファイルを Java Archive (JAR) ファイルにビルドするように に指示します。このチュートリアルを完了するために、Apache Maven または Java に精通している必要はありません。

コンソール CodeBuild 、、AWS CodePipeline、AWS CLIまたは AWS SDKs CodeBuild を使用して を操作できます。このチュートリアルでは、 CodeBuild で を使用する方法を示しますAWS CLI。の使用の詳細については CodePipeline、「」を参照してください CodePipeline で を使用する CodeBuild。

> **Important**
>
> このチュートリアルのステップでは、AWS アカウントに課金される可能性のあるリソース（S3 バケットなど）を作成する必要があります。これには、Amazon S3、、 CodeBuild および CloudWatch Logs に関連するAWSリソースやアクションに対する AWS KMSおよび の料金が含まれます。詳細については、「 のCodeBuild料金」、「Amazon S3 の料金」、AWS Key Management Service「 の料金」、および「Amazon の CloudWatch 料金」を参照してください。

ステップ

ステップ 1: ソースコードを作成する

（一部: AWS CodeBuild を使用した AWS CLI の開始方法）

このステップでは、出力バケットに CodeBuild ビルドするソースコードを作成します。このソースコードは 2 つの Java クラスファイルと Apache Maven プロジェクトオブジェクトモデル (POM) ファイルで構成されています。

1. ローカルコンピュータまたはインスタンスの空のディレクトリに、このディレクトリ構造を作成します。

```
(root directory name)
    `-- src
         |-- main
         |     `-- java
         `-- test
               `-- java
```

2. 任意のテキストエディタを使用して、このファイルを作成し、MessageUtil.java という名前を付けて、src/main/java ディレクトリに保存します。

```
public class MessageUtil {
  private String message;

  public MessageUtil(String message) {
    this.message = message;
  }

  public String printMessage() {
    System.out.println(message);
    return message;
  }

  public String salutationMessage() {
    message = "Hi!" + message;
    System.out.println(message);
    return message;
  }
}
```

このクラスファイルは、渡された文字列を出力として作成します。MessageUtil コンストラクタは、文字列を設定します。printMessage メソッドは出力を作成します。salutationMessage メソッドが Hi! を出力した後に文字列が続きます。

3. このファイルを作成し、TestMessageUtil.java という名前を付けて、/src/test/java ディレクトリに保存します。

```
import org.junit.Test;
import org.junit.Ignore;
import static org.junit.Assert.assertEquals;

public class TestMessageUtil {

  String message = "Robert";
  MessageUtil messageUtil = new MessageUtil(message);

  @Test
  public void testPrintMessage() {
    System.out.println("Inside testPrintMessage()");
    assertEquals(message,messageUtil.printMessage());
  }

  @Test
  public void testSalutationMessage() {
    System.out.println("Inside testSalutationMessage()");
    message = "Hi!" + "Robert";
    assertEquals(message,messageUtil.salutationMessage());
  }
}
```

このクラスファイルは message クラスの MessageUtil 変数を Robert に設定します。その後、文字列 message および Robert が出力に表示されているかどうかを調べることによって、Hi!Robert 変数が正常に設定されたかどうかを調べます。

4. このファイルを作成し、pom.xml という名前を付けて、ルート (最上位) ディレクトリに保存します。

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/
maven-v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>org.example</groupId>
  <artifactId>messageUtil</artifactId>
  <version>1.0</version>
  <packaging>jar</packaging>
  <name>Message Utility Java Sample App</name>
  <dependencies>
    <dependency>
```

```
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>4.11</version>
      <scope>test</scope>
    </dependency>
  </dependencies>
  <build>
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-compiler-plugin</artifactId>
        <version>3.8.0</version>
      </plugin>
    </plugins>
  </build>
</project>
```

Apache Maven では、このファイルの指示に従って、MessageUtil.java および TestMessageUtil.java ファイルを messageUtil-1.0.jar という名前のファイルに変換し、指定されたテストを実行します。

この時点で、ディレクトリ構造は次のようになります。

```
(root directory name)
    |-- pom.xml
    `-- src
         |-- main
         |     `-- java
         |           `-- MessageUtil.java
         `-- test
               `-- java
                     `-- TestMessageUtil.java
```

次のステップ

ステップ 2: buildspec ファイルを作成する

ステップ 2: buildspec ファイルを作成する

(前のステップ: ステップ 1: ソースコードを作成する)

このステップでは、ビルド仕様ファイルを作成します。buildspec は、 がビルドの実行 CodeBuild に使用する YAML 形式のビルドコマンドと関連設定のコレクションです。ビルド仕様がないと、ビルド入力をビルド出力に正常に変換したり、ビルド環境でビルド出力アーティファクトを見つけて出力バケットにアップロードしたり CodeBuild することはできません。

このファイルを作成し、`buildspec.yml` という名前を付けて、ルート (最上位) ディレクトリに保存します。

```
version: 0.2

phases:
  install:
    runtime-versions:
      java: corretto11
  pre_build:
    commands:
      - echo Nothing to do in the pre_build phase...
  build:
    commands:
      - echo Build started on `date`
      - mvn install
  post_build:
    commands:
      - echo Build completed on `date`
artifacts:
  files:
    - target/messageUtil-1.0.jar
```

Important

ビルド仕様宣言は有効な YAML である必要があるため、ビルド仕様宣言のスペースは重要です。ビルド仕様宣言のスペース数が YAML と一致しない場合、ビルドは即座に失敗する場合があります。YAML validator を使用して、ビルド仕様宣言が有効な YAML かどうかをテストできます。

Note

ソースコードにビルド仕様ファイルを含める代わりに、ビルドプロジェクトを作成するときに個別にビルドコマンドを宣言することができます。これは、毎回ソースコードのリポジト

リを更新せずに、異なるビルドコマンドでソースコードをビルドする場合に役立ちます。詳細については、「buildspec の構文」を参照してください。

このビルド仕様宣言の詳細は次の通りです。

- version は、使用されているビルド仕様スタンダードのバージョンを表します。このビルド仕様宣言では、最新バージョン 0.2 が使用されます。
- phases は、 CodeBuild にコマンドの実行を指示するビルドフェーズを表します。これらのビルドフェーズは install、pre_build、build、post_build として、ここにリストされています。これらのビルドフェーズ名のスペルを変更することはできず、追加のビルドフェーズ名を作成することもできません。

  この例では、 buildフェーズ中に が mvn install コマンド CodeBuild を実行します。このコマンドは、コンパイルされた Java クラスファイルをビルド出力アーティファクトにコンパイル、テスト、パッケージ化するように Apache Maven に指示します。完全にするために、この例では、いくつかの echo コマンドが各ビルドフェーズに配置されています。このチュートリアルの後半で詳細なビルド情報を見る際に、これらの echo コマンドの出力は、 CodeBuild がコマンドを実行する方法とその順序を理解するのに役立ちます。(この例にはすべてのビルドフェーズが含まれていますが、コマンドを実行しないビルドフェーズは含める必要がありません)。ビルドフェーズごとに、 は指定された各コマンドを、最初から最後まで、リストされた順序で 1 つずつ CodeBuild 実行します。
- artifacts は、 が CodeBuild出力バケットにアップロードするビルド出力アーティファクトのセットを表します。 は、ビルド出力に含めるファイルfilesを表します。 CodeBuild は、ビルド環境のtarget相対ディレクトリにある単一のmessageUtil-1.0.jarファイルをアップロードします。ファイル名 messageUtil-1.0.jar およびディレクトリ名 target は、この例でのみ Apache Maven がビルド出力アーティファクトを作成して格納する方法に基づいています。独自のビルドでは、これらのファイル名とディレクトリは異なります。

詳細については、「ビルド仕様 (buildspec) に関するリファレンス」を参照してください。

この時点で、ディレクトリ構造は次のようになります。

```
(root directory name)
    |-- pom.xml
    |-- buildspec.yml
    `-- src
         |-- main
```

```
|     `-- java
|           `-- MessageUtil.java
`-- test
      `-- java
            `-- TestMessageUtil.java
```

次のステップ

ステップ 3: 2 つの S3 バケットを作成する

ステップ 3: 2 つの S3 バケットを作成する

(前のステップ: ステップ 2: buildspec ファイルを作成する)

このチュートリアル用として 1 つのバケットを使用することもできますが、2 つのバケットを使用する方が、ビルド入力の送信元およびビルド出力の送信先の確認が簡単になります。

- これらのバケットのいずれか (入力バケット) にビルド入力が保存されます。このチュートリアルでは、この入力バケットの名前は codebuild-*region-ID*-*account-ID*-input-bucket です (*region-ID* はバケットの AWS リージョン、*account-ID* は AWS アカウント ID です)。
- もう 1 つのバケット (出力バケット) にはビルド出力が保存されます。このチュートリアルでは、この出力バケットの名前は codebuild-*region-ID*-*account-ID*-output-bucket です。

これらのバケットに別の名前を選択した場合は、このチュートリアル全体で、その名前を使用してください。

これらの 2 つのバケットは、ビルドと同じ AWS リージョン内にある必要があります。例えば、米国東部 (オハイオ) リージョンでビルドを実行する CodeBuild ように に指示する場合、これらのバケットも米国東部 (オハイオ) リージョンにある必要があります。

詳細については、Amazon Simple Storage Service コンソールユーザーガイドの「バケットの作成」を参照してください。

> **Note**
>
> は、 CodeCommit、 GitHub、および Bitbucket リポジトリに保存されているビルド入力 CodeBuild もサポートしていますが、このチュートリアルでは、それらの使用方法については説明しません。詳細については、「ビルドを計画する」を参照してください。

次のステップ

ステップ 4: ソースコードと buildspec ファイルをアップロードする

ステップ 4: ソースコードと buildspec ファイルをアップロードする

(前のステップ: ステップ 3: 2 つの S3 バケットを作成する)

このステップでは、入力バケットにソースコードとビルド仕様ファイルを追加します。

オペレーティングシステムの zip ユーティリティを使用して、MessageUtil.zip、MessageUtil.java、TestMessageUtil.java、および pom.xml を含む buildspec.yml という名前のファイルを作成します。

MessageUtil.zip ファイルのディレクトリ構造は、次のようになっている必要があります。

```
MessageUtil.zip
    |-- pom.xml
    |-- buildspec.yml
    `-- src
         |-- main
         |     `-- java
         |           `-- MessageUtil.java
         `-- test
               `-- java
                     `-- TestMessageUtil.java
```

> **Important**
>
> *(root directory name)* ディレクトリを含めないでください。*(root directory name)* ディレクトリ内のディレクトリとファイルのみを含めます。

MessageUtil.zip ファイルを codebuild-*region-ID*-*account-ID*-input-bucket という名前の入力バケットにアップロードします。

> **Important**
>
> CodeCommit、 GitHub、および Bitbucket リポジトリの場合、規則により、 という名前のビルド仕様ファイルを各リポジトリのbuildspec.ymlルート (最上位) に保存するか、ビルド

仕様宣言をビルドプロジェクト定義の一部として含める必要があります。リポジトリのソースコードとビルド仕様ファイルを含む ZIP ファイルを作成しないでください。
S3 バケットに保存されたビルド入力に限り、ソースコードおよび規約に基づく `buildspec.yml` というビルド仕様ファイルを圧縮した ZIP ファイルをルート（最上位）に作成するか、ビルド仕様宣言をビルドプロジェクト定義の一部として含める必要があります。
ビルド仕様ファイルに別の名前を使用するか、ルート以外の場所でビルド仕様を参照する場合は、ビルドプロジェクト定義の一部としてビルド仕様の上書きを指定できます。詳細については、「buildspec ファイル名とストレージの場所」を参照してください。

次のステップ

ステップ 5: ビルドプロジェクトを作成する

ステップ 5: ビルドプロジェクトを作成する

(前のステップ: ステップ 4: ソースコードと buildspec ファイルをアップロードする)

このステップでは、AWS CodeBuild がビルドの実行に使用するビルドプロジェクトを作成します。ビルドプロジェクトには、ビルドの実行方法に関する情報が含まれています。これには、ソースコードの取得先、使用するビルド環境、実行するビルドコマンド、ビルド出力の格納先が含まれます。ビルド環境は、オペレーティングシステム、プログラミング言語ランタイム、および が ビルドの実行 CodeBuild に使用するツールの組み合わせを表します。ビルド環境は Docker イメージとして表されます。詳細については、Docker Docs ウェブサイトの Docker overview を参照してください。

このビルド環境では、Java Development Kit (JDK) と Apache Maven のバージョンを含む Docker イメージを使用する CodeBuild ように に指示します。

ビルドプロジェクトを作成するには

1. AWS CLI を使用して create-project コマンドを実行します。

```
aws codebuild create-project --generate-cli-skeleton
```

JSON 形式のデータが出力に表示されます。`create-project.json` がインストールされているローカルコンピュータまたはインスタンス上の場所にある AWS CLI というファイルにデータをコピーします。別のファイル名を使用する場合は、このチュートリアル全体でそのファイル名を使用してください。

コピーされたデータをこの形式に従って変更して、結果を保存します。

```
{
  "name": "codebuild-demo-project",
  "source": {
    "type": "S3",
    "location": "codebuild-region-ID-account-ID-input-bucket/MessageUtil.zip"
  },
  "artifacts": {
    "type": "S3",
    "location": "codebuild-region-ID-account-ID-output-bucket"
  },
  "environment": {
    "type": "LINUX_CONTAINER",
    "image": "aws/codebuild/standard:5.0",
    "computeType": "BUILD_GENERAL1_SMALL"
  },
  "serviceRole": "serviceIAMRole"
}
```

serviceIAMRole を CodeBuild サービスロールの Amazon リソースネーム (ARN) に置き換えます (例:)arn:aws:iam::*account-ID*:role/*role-name*。サービスロールを作成する場合は、「CodeBuild サービスロールの作成」を参照してください。

このデータの各要素は以下のとおりです。

- name は、このビルドプロジェクト (この例では codebuild-demo-project) に必要な識別子を表します。ビルドプロジェクト名は、アカウント内のすべてのビルドプロジェクト間で一意である必要があります。
- source の場合、type はソースコードのリポジトリのタイプを表す必須の値です (この例では、Amazon S3 バケットの場合は S3)。
- source の場合、location は、ソースコードへのパスを表します (この例では、入力バケット名の後に ZIP ファイル名が続きます)。
- artifacts の場合、type は、ビルド出力アーティファクトのリポジトリのタイプを表す必須の値です (この例では、Amazon S3 バケットの場合は S3)。
- artifacts の場合、location は、以前に作成または識別した出力バケットの名前を表します (この例では、codebuild-*region-ID*-*account-ID*-output-bucket)。

- environment を使用する場合、type はビルド環境のタイプを表す必須の値です (この例では LINUX_CONTAINER)。
- の場合environment、 imageは Docker イメージリポジトリタイプ (この例では Docker イメージリポジトリ内の Docker イメージaws/codebuild/standard:5.0の場合) で指定された、このビルドプロジェクトが使用する CodeBuild Docker イメージ名とタグの組み合わせを表す必須の値です。 aws/codebuild/standardは Docker イメージの名前です。 は Docker イメージのタグ5.0です。

  シナリオで使用できる Docker イメージをさらに見つけるには、「ビルド環境に関するリファレンス」を参照してください。
- の場合environment、 computeTypeは が CodeBuild 使用するコンピューティングリソースを表す必須の値です (この例では BUILD_GENERAL1_SMALL) 。

> **Note**
>
> description、buildspec、auth (type と resource を含む)、path、namespaceType、name (artifacts)、packaging、environmentVariables (name と value を含む)、timeoutInMinutes、encryptionKey、tags (key と value を含む) などの元の JSON 形式のデータで使用可能なその他の値はオプションです。これらは、このチュートリアルで使用されていないため、ここには示されていません。詳細については、「ビルドプロジェクトの作成 (AWS CLI)」を参照してください。

2. 保存したばかりのファイルがあるディレクトリに移動してから、create-project コマンドをもう一度実行します。

```
aws codebuild create-project --cli-input-json file://create-project.json
```

成功した場合、次のようなデータが出力に表示されます。

```
{
  "project": {
    "name": "codebuild-demo-project",
    "serviceRole": "serviceIAMRole",
    "tags": [],
    "artifacts": {
      "packaging": "NONE",
```

```
      "type": "S3",
      "location": "codebuild-region-ID-account-ID-output-bucket",
      "name": "message-util.zip"
    },
    "lastModified": 1472661575.244,
    "timeoutInMinutes": 60,
    "created": 1472661575.244,
    "environment": {
      "computeType": "BUILD_GENERAL1_SMALL",
      "image": "aws/codebuild/standard:5.0",
      "type": "LINUX_CONTAINER",
      "environmentVariables": []
    },
    "source": {
      "type": "S3",
      "location": "codebuild-region-ID-account-ID-input-bucket/MessageUtil.zip"
    },
    "encryptionKey": "arn:aws:kms:region-ID:account-ID:alias/aws/s3",
    "arn": "arn:aws:codebuild:region-ID:account-ID:project/codebuild-demo-project"
  }
}
```

- `project` は、このビルドプロジェクトに関する情報を表します。
  - `tags` は、宣言されたタグを表します。
  - `packaging` は、ビルド出力アーティファクトが出力バケットに保存される方法を表します。`NONE` は、出力バケット内にフォルダが作成されることを意味します。ビルド出力アーティファクトはそのフォルダ内に格納されます。
  - `lastModified` は、ビルドプロジェクトに関する情報が最後に変更された時刻 (Unix の時間形式) を表します。
  - `timeoutInMinutes` は、ビルドが完了していない場合に がビルド CodeBuild を停止するまでの分数を表します。(デフォルトは 60 分です。)
  - `created` は、ビルドプロジェクトが作成された時刻 (Unix の時間形式) を表します。
  - `environmentVariables` は、宣言され、ビルド中に CodeBuild が使用できる環境変数を表します。
  - `encryptionKey` は、ビルド出力アーティファクトの暗号化に が CodeBuild 使用したカスタマーマネージドキーの ARN を表します。
  - `arn` は、ビルドプロジェクトの ARN を表します。

> **Note**
>
> create-project コマンドを実行すると、次のようなエラーメッセージが出力される場合があります: User: ***user-ARN*** is not authorized to perform: codebuild:CreateProject。これは、ビルドプロジェクトの作成 CodeBuild に使用する十分なアクセス許可を持たない ユーザーの認証情報AWS CLIで を設定したことが原因と考えられます。これを修正するには、次の IAM エンティティのいずれかに属する認証情報を使用して AWS CLI を設定します。
>
> - AWS アカウントの管理者ユーザー。詳細については、ユーザーガイドの「最初の AWS アカウントルートユーザーおよびグループの作成」を参照してください。
> - ユーザー (またはユーザーが属する IAM グループ) に `AWSCodeBuildAdminAccess`、`AmazonS3ReadOnlyAccess`、および `IAMFullAccess` 管理ポリシーがアタッチされている、AWS アカウントのユーザー。これらのアクセス許可を持つユーザーやグループが AWS アカウントに存在せず、これらのアクセス許可をユーザーやグループに追加できない場合は、AWS アカウント管理者に連絡してサポートを依頼してください。詳細については、「AWS の マネージド (事前定義) ポリシー AWS CodeBuild」を参照してください。

次のステップ

ステップ 6: ビルドを実行する

ステップ 6: ビルドを実行する

(前のステップ: ステップ 5: ビルドプロジェクトを作成する)

このステップでは、ビルドプロジェクトの設定でビルドを実行するように AWS CodeBuild に指示します。

ビルドを実行するには

1. AWS CLI を使用して start-build コマンドを実行します。

   ```
   aws codebuild start-build --project-name project-name
   ```

   project-name を、前の手順のビルドプロジェクト名 (例: `codebuild-demo-project`) に置き換えます。
2. 成功すると、次のようなデータが出力に表示されます。

```
{
  "build": {
    "buildComplete": false,
    "initiator": "user-name",
    "artifacts": {
      "location": "arn:aws:s3:::codebuild-region-ID-account-ID-output-bucket/
message-util.zip"
    },
    "projectName": "codebuild-demo-project",
    "timeoutInMinutes": 60,
    "buildStatus": "IN_PROGRESS",
    "environment": {
      "computeType": "BUILD_GENERAL1_SMALL",
      "image": "aws/codebuild/standard:5.0",
      "type": "LINUX_CONTAINER",
      "environmentVariables": []
    },
    "source": {
      "type": "S3",
      "location": "codebuild-region-ID-account-ID-input-bucket/MessageUtil.zip"
    },
    "currentPhase": "SUBMITTED",
    "startTime": 1472848787.882,
    "id": "codebuild-demo-project:0cfbb6ec-3db9-4e8c-992b-1ab28EXAMPLE",
    "arn": "arn:aws:codebuild:region-ID:account-ID:build/codebuild-demo-
project:0cfbb6ec-3db9-4e8c-992b-1ab28EXAMPLE"
  }
}
```

- build は、このビルドに関する情報を表します。
  - buildComplete は、ビルドの完了 () を表します。trueそうでない場合は、false です。
  - initiator は、ビルドを開始したエンティティを表します。
  - artifacts は、場所を含む、ビルド出力に関する情報を表します。
  - projectName は、ビルドプロジェクトの名前を表します。
  - buildStatus は、start-build コマンドが実行されたときの現在のビルドのステータスを表します。
  - currentPhase は、start-build コマンドが実行されたときの現在のビルドフェーズを表します。
  - startTime は、ビルドプロセスが開始された時刻 (Unix の時間形式) を表します。

- id は、ビルドの ID を表します。
- arn は、ビルドの ARN を表します。

[id] の値を書き留めておきます。これは次の手順で必要です。

次のステップ

ステップ 7: ビルド情報の要約を表示する

ステップ 7: ビルド情報の要約を表示する

(前のステップ: ステップ 6: ビルドを実行する)

このステップでは、ビルドのステータスに関する要約情報を表示します。

要約されたビルド情報を表示するには

AWS CLI を使用して batch-get-builds コマンドを実行します。

```
aws codebuild batch-get-builds --ids id
```

id を、前のステップの出力に表示された id 値に置き換えます。

成功した場合、次のようなデータが出力に表示されます。

```
{
  "buildsNotFound": [],
  "builds": [
    {
      "buildComplete": true,
      "phases": [
        {
          "phaseStatus": "SUCCEEDED",
          "endTime": 1472848788.525,
          "phaseType": "SUBMITTED",
          "durationInSeconds": 0,
          "startTime": 1472848787.882
        },
        ... The full list of build phases has been omitted for brevity ...
        {
          "phaseType": "COMPLETED",
```

```
          "startTime": 1472848878.079
        }
      ],
      "logs": {
        "groupName": "/aws/codebuild/codebuild-demo-project",
        "deepLink": "https://console.aws.amazon.com/cloudwatch/home?region=region-
ID#logEvent:group=/aws/codebuild/codebuild-demo-project;stream=38ca1c4a-e9ca-4dbc-bef1-
d52bfEXAMPLE",
        "streamName": "38ca1c4a-e9ca-4dbc-bef1-d52bfEXAMPLE"
      },
      "artifacts": {
        "md5sum": "MD5-hash",
        "location": "arn:aws:s3:::codebuild-region-ID-account-ID-output-bucket/message-
util.zip",
        "sha256sum": "SHA-256-hash"
      },
      "projectName": "codebuild-demo-project",
      "timeoutInMinutes": 60,
      "initiator": "user-name",
      "buildStatus": "SUCCEEDED",
      "environment": {
        "computeType": "BUILD_GENERAL1_SMALL",
        "image": "aws/codebuild/standard:5.0",
        "type": "LINUX_CONTAINER",
        "environmentVariables": []
      },
      "source": {
        "type": "S3",
        "location": "codebuild-region-ID-account-ID-input-bucket/MessageUtil.zip"
      },
      "currentPhase": "COMPLETED",
      "startTime": 1472848787.882,
      "endTime": 1472848878.079,
      "id": "codebuild-demo-project:38ca1c4a-e9ca-4dbc-bef1-d52bfEXAMPLE",
      "arn": "arn:aws:codebuild:region-ID:account-ID:build/codebuild-demo-
project:38ca1c4a-e9ca-4dbc-bef1-d52bfEXAMPLE"
    }
  ]
}
```


- buildsNotFound は、情報が利用できないビルドのビルド ID を表します。この例では、空である必要があります。

- builds は、利用可能な各ビルドに関する情報を表します。この例では、出力に 1 つのビルドのみに関する情報が表示されます。
  - phases は、ビルドプロセス中に CodeBuild 実行されるビルドフェーズのセットを表します。各ビルドフェーズに関する情報が startTime、endTime、durationInSeconds (フェーズの開始時と終了時、Unix 時間形式で表示、持続時間 (秒))、phaseType (SUBMITTED、PROVISIONING、DOWNLOAD_SOURCE、INSTALL、PRE_BUILD、BUILD、POST_BUIL など)、phaseStatus (SUCCEEDED、FAILED、FAULT、TIMED_OUT、IN_PROGRESS、STOPPED など) として個別にリストされます。batch-get-builds コマンドを初めて実行するときは、多くの (またはまったく) フェーズが存在しない可能性があります。同じビルド ID を持つ batch-get-builds コマンドを続けて実行すると、より多くのビルドフェーズが出力に表示されます。
  - logs は、ビルドのログに関する Amazon CloudWatch Logs の情報を表します。
  - md5sum および sha256sum は、ビルドの出力アーティファクトの MD5 と SHA-256 ハッシュを表します。これらは、関連するビルドプロジェクトの packaging 値が ZIP に設定されている場合にのみ出力に表示されます。(このチュートリアルでは設定していません。) これらのハッシュとチェックサムツールを使用して、ファイルの完全性と信頼性を確認することができます。

    > **Note**
    >
    > Amazon S3 コンソールを使用して、これらのハッシュを表示することもできます。ビルド出力アーティファクトの横にあるボックスを選択し、[アクション]、[プロパティ] の順に選択します。プロパティペインで、メタデータ を展開し、x-amz-meta-codebuild-content-md5 と x-amz-meta-codebuild-content-sha256 の値を表示します。(Amazon S3 コンソールでは、ビルド出力アーティファクトの [ETag] 値を MD5 または SHA-256 ハッシュのいずれかに解釈することはできません。)
    > AWS SDK を使用して、これらのハッシュを取得する場合、値の名前は codebuild-content-md5 および codebuild-content-sha256 です。

  - endTime は、ビルドプロセスが終了した時刻 (Unix の時間形式) を表します。

> **Note**
>
> Amazon S3 メタデータには、Amazon S3 x-amz-meta-codebuild-buildarn にアーティファクトを発行する CodeBuild ビルドbuildArnの を含む という名前の CodeBuild

ヘッダーがあります。通知のソーストラッキングを許可し、アーティファクトの生成元であるビルドを参照するために buildArn を追加します。

次のステップ

ステップ 8: 詳細なビルド情報を表示する

ステップ 8: 詳細なビルド情報を表示する

(前のステップ: ステップ 7: ビルド情報の要約を表示する)

このステップでは、 CloudWatch ログでビルドに関する詳細情報を表示します。

> **Note**
>
> 機密情報を保護するために、ログでは以下が非表示になっています CodeBuild 。
>
> - AWS アクセスキー ID。詳細については、AWS Identity and Access Management ユーザーガイドの IAM ユーザーのアクセスキーの管理を参照してください。
> - パラメータストアを使用して指定された文字列。詳細については、「Amazon EC2 Systems Manager ユーザーガイド」の「Systems Manager パラメータストア」および「Systems Manager パラメータストアコンソールのチュートリアル」を参照してください。
> - AWS Secrets Manager を使用して指定された文字列。詳細については、「キー管理」を参照してください。

詳細なビルド情報を表示するには

1. ウェブブラウザを使用して、前の手順の出力に表示された deepLink の場所に移動します (例: `https://console.aws.amazon.com/cloudwatch/home?region=region-ID#logEvent:group=/aws/codebuild/codebuild-demo-project;stream=38ca1c4a-e9ca-4dbc-bef1-d52bfEXAMPLE`)。
2. CloudWatch Logs ログストリームでは、ログイベントを参照できます。デフォルトでは、ログイベントの最後のセットだけが表示されます。以前のログイベントを表示するには、リストの先頭にスクロールします。

3. このチュートリアルでは、ほとんどのログイベントに、 CodeBuild でビルド依存ファイルをビルド環境にダウンロードおよびインストールする操作に関する詳細情報が含まれますが、これらの情報は不要な場合があります。[Filter events] ボックスを使用すると、表示する情報量を減らすことができます。例えば、[Filter events] (イベントのフィルター) で「"[INFO]"」と入力した場合、「[INFO]」を含むイベントのみが表示されます。詳細については、「Amazon ユーザーガイド」の「フィルターとパターンの構文」を参照してください。 CloudWatch

CloudWatch Logs ログストリームのこれらの部分は、このチュートリアルに関連しています。

```
...
[Container] 2016/04/15 17:49:42 Entering phase PRE_BUILD
[Container] 2016/04/15 17:49:42 Running command echo Entering pre_build phase...
[Container] 2016/04/15 17:49:42 Entering pre_build phase...
[Container] 2016/04/15 17:49:42 Phase complete: PRE_BUILD Success: true
[Container] 2016/04/15 17:49:42 Entering phase BUILD
[Container] 2016/04/15 17:49:42 Running command echo Entering build phase...
[Container] 2016/04/15 17:49:42 Entering build phase...
[Container] 2016/04/15 17:49:42 Running command mvn install
[Container] 2016/04/15 17:49:44 [INFO] Scanning for projects...
[Container] 2016/04/15 17:49:44 [INFO]
[Container] 2016/04/15 17:49:44 [INFO]
 ------------------------------------------------------------------------
[Container] 2016/04/15 17:49:44 [INFO] Building Message Utility Java Sample App 1.0
[Container] 2016/04/15 17:49:44 [INFO]
 ------------------------------------------------------------------------
...
[Container] 2016/04/15 17:49:55
 -------------------------------------------------------
[Container] 2016/04/15 17:49:55  T E S T S
[Container] 2016/04/15 17:49:55
 -------------------------------------------------------
[Container] 2016/04/15 17:49:55 Running TestMessageUtil
[Container] 2016/04/15 17:49:55 Inside testSalutationMessage()
[Container] 2016/04/15 17:49:55 Hi!Robert
[Container] 2016/04/15 17:49:55 Inside testPrintMessage()
[Container] 2016/04/15 17:49:55 Robert
[Container] 2016/04/15 17:49:55 Tests run: 2, Failures: 0, Errors: 0, Skipped: 0, Time
 elapsed: 0.018 sec
[Container] 2016/04/15 17:49:55
[Container] 2016/04/15 17:49:55 Results :
[Container] 2016/04/15 17:49:55
[Container] 2016/04/15 17:49:55 Tests run: 2, Failures: 0, Errors: 0, Skipped: 0
```

```
...
[Container] 2016/04/15 17:49:56 [INFO]
 ------------------------------------------------------------------------
[Container] 2016/04/15 17:49:56 [INFO] BUILD SUCCESS
[Container] 2016/04/15 17:49:56 [INFO]
 ------------------------------------------------------------------------
[Container] 2016/04/15 17:49:56 [INFO] Total time: 11.845 s
[Container] 2016/04/15 17:49:56 [INFO] Finished at: 2016-04-15T17:49:56+00:00
[Container] 2016/04/15 17:49:56 [INFO] Final Memory: 18M/216M
[Container] 2016/04/15 17:49:56 [INFO]
 ------------------------------------------------------------------------
[Container] 2016/04/15 17:49:56 Phase complete: BUILD Success: true
[Container] 2016/04/15 17:49:56 Entering phase POST_BUILD
[Container] 2016/04/15 17:49:56 Running command echo Entering post_build phase...
[Container] 2016/04/15 17:49:56 Entering post_build phase...
[Container] 2016/04/15 17:49:56 Phase complete: POST_BUILD Success: true
[Container] 2016/04/15 17:49:57 Preparing to copy artifacts
[Container] 2016/04/15 17:49:57 Assembling file list
[Container] 2016/04/15 17:49:57 Expanding target/messageUtil-1.0.jar
[Container] 2016/04/15 17:49:57 Found target/messageUtil-1.0.jar
[Container] 2016/04/15 17:49:57 Creating zip artifact
```

この例では、 はビルド前、ビルド、ビルド後のビルドフェーズを CodeBuild 正常に完了しました。ユニットテストを実行し、`messageUtil-1.0.jar` ファイルは正常に構築されました。

次のステップ

ステップ 9: ビルド出力アーティファクトを取得する

ステップ 9: ビルド出力アーティファクトを取得する

(前のステップ: ステップ 8: 詳細なビルド情報を表示する)

このステップでは、 を CodeBuild ビルドして出力バケットにアップロードした `messageUtil-1.0.jar` ファイルを取得します。

このステップを完了するには、 CodeBuild コンソールまたは Amazon S3 コンソールを使用します。

ビルド出力アーティファクトを取得するには (AWS CodeBuild コンソール)

1. CodeBuild コンソールを開いたままにして、前のステップのビルドの詳細ページを表示したら、ビルドの詳細タブを選択し、アーティファクトセクションまでスクロールします。

> **Note**
>
> ビルドの詳細ページが表示されていない場合は、ナビゲーションバーで [Build history] (ビルド履歴)、[Build run] (ビルドの実行) リンクの順に選択します。

2. Amazon S3 フォルダへのリンクは、[Artifacts upload location] (アーティファクトのアップロード場所) の下にあります。Amazon S3 内のフォルダが開き、「`messageUtil-1.0.jar`」という名前のビルド出力アーティファクトファイルを見つけます。

ビルド出力アーティファクトを取得するには (Amazon S3 コンソール)

1. https://console.aws.amazon.com/s3/でAmazon S3 コンソールを開きます。
2. `codebuild-`*`region-ID`*`-`*`account-ID`*`-output-bucket` を開きます。
3. `codebuild-demo-project` フォルダを開きます。
4. `target` という名前のフォルダを開き、`messageUtil-1.0.jar` という名前のビルド出力アーティファクトファイルを見つけます。

次のステップ

ステップ 10: S3 入力バケットを削除する

ステップ 10: S3 入力バケットを削除する

(前のステップ: ステップ 9: ビルド出力アーティファクトを取得する)

AWS アカウントで継続的に料金が発生しないように、このチュートリアルで使用した入力・出力バケットを削除することもできます。手順については、Amazon Simple Storage Service ユーザーガイドでバケットを削除、または空にする方法を参照してください。

IAM ユーザー を使用している場合、このバケットを削除するユーザーまたは管理者 IAM ユーザーには、さらに高いアクセス権限が必要です。マーカー間で次のステートメント (*`###BEGIN ADDING STATEMENT HERE###`* と *`###END ADDING STATEMENTS HERE###`*) を既存のアクセスポリシーに追加します。

このステートメントでは、簡潔にするために省略記号 (...) が使用されています。既存のアクセスポリシーのステートメントは削除しないでください。これらの省略記号はポリシーに入力しないでください。

```
{
  "Version": "2012-10-17",
  "Id": "...",
  "Statement": [
    ### BEGIN ADDING STATEMENT HERE ###
    {
      "Effect": "Allow",
      "Action": [
        "s3:DeleteBucket",
        "s3:DeleteObject"
      ],
      "Resource": "*"
    }
    ### END ADDING STATEMENT HERE ###
  ]
}
```

次のステップ

まとめ

まとめ

このチュートリアルでは、AWS CodeBuild を使用して、一連の Java クラスファイルから JAR ファイルを作成しました。次に、ビルドの結果を表示しました。

独自のシナリオ CodeBuild で を使用できるようになりました。「ビルドを計画する」の手順に従ってください。もう少し準備が必要な場合は、用意されているサンプルでビルドを試すことができます。詳細については、「サンプル」を参照してください。

CodeBuild サンプル

これらのサンプルグループは、 を試すために使用できます AWS CodeBuild。

トピック

- のユースケースベースのサンプル CodeBuild
- の Microsoft Windows サンプル CodeBuild

のユースケースベースのサンプル CodeBuild

これらのユースケースベースのサンプルを使用して、 を試すことができます AWS CodeBuild。

クロスサービスのサンプル

で実験するクロスサービスサンプルのリスト AWS CodeBuild。

ビルドバッジサンプル

ビルドバッジ CodeBuild を使用して を設定する方法を示します。

AWS CLI サンプルを使用したテストレポートの作成

を使用して、テストレポートの結果 AWS CLI を作成、実行、および表示します。

の Docker サンプル CodeBuild

カスタム Docker イメージの使用、Amazon ECR のリポジトリへの Docker イメージの公開、プライベートレジストリでの Docker イメージの使用方法について説明します。

S3 バケットでのビルド出力のホスティング

暗号化されていないビルドアーティファクトを使用して S3 バケットに静的なウェブサイトを作成する方法を示します。

複数の入力ソースと出力アーティファクトのサンプル

ビルドプロジェクトで複数の入力ソースと複数の出力アーティファクトを使用する方法を示します。

buildspec ファイルサンプルのランタイムバージョン

buildspec ファイルでランタイムとバージョンを指定する方法を示します。

ソースバージョンのサンプル

CodeBuild ビルドプロジェクトで特定のバージョンのソースを使用する方法を示します。

のサードパーティーのソースリポジトリのサンプル CodeBuild

を使用して、ウェブフックで BitBucket、 GitHub Enterprise Server、 GitHub プルリクエストを作成する方法を示します CodeBuild。

セマンティックバージョニングを使用してビルドアーティファクトに名前を付けるサンプル

セマンティックバージョニングを使用して、ビルド時にアーティファクト名を作成する方法を示します。

のクロスサービスサンプル CodeBuild

これらのクロスサービスサンプルを使用して、 を試すことができます AWS CodeBuild。

Amazon ECR のサンプル

Amazon ECR リポジトリの Docker イメージを使用して、Apache Maven を使用して単一の JAR ファイルを生成します。

Amazon EFS のサンプル

CodeBuild プロジェクトが Amazon EFS ファイルシステムにマウントおよび構築されるように buildspec ファイルを設定する方法を示します。

AWS CodePipeline サンプル

AWS CodePipeline を使用して、バッチビルドと複数の入力ソースおよび複数の出力アーティファクトを含むビルドを作成する方法を示します。

AWS Config サンプル

のセットアップ方法を示します AWS Config。追跡する CodeBuild リソースを一覧表示し、 で CodeBuild プロジェクトを検索する方法について説明します AWS Config。

ビルド通知サンプル

Apache Maven を使用して単一の JAR ファイルを生成します。Amazon SNS トピックのサブスクライバーにビルド通知を送信します。

の Amazon ECR サンプル CodeBuild

このサンプルでは Amazon Elastic Container Registry (Amazon ECR) イメージレポジトリの Docker イメージを使用して、サンプルの Go プロジェクトをビルドします。

> **Important**
>
> このサンプルを実行すると、 AWS アカウントに課金される場合があります。これには、Amazon S3、、 CloudWatch Logs AWS KMS、Amazon ECR に関連する AWS リソースやアクション AWS CodeBuild に対する および の料金が含まれます。詳細については、「 のCodeBuild 料金」、「Amazon S3 の料金」、AWS Key Management Service 「 の料金」、「Amazon の CloudWatch 料金」、「Amazon Elastic Container Registry の料金」を参照してください。

サンプルの実行

このサンプルを実行するには

1. Amazon ECR で Docker イメージを作成してイメージリポジトリにプッシュするには、Docker イメージを Amazon ECR イメージリポジトリサンプルに公開する の「サンプルの実行」セクションにある手順を完了します。
2. Go プロジェクトの作成:
   a. このトピックの セクションGo プロジェクトの構造と Go プロジェクトのファイルセクションの説明に従ってファイルを作成し、S3 入力バケット、 AWS CodeCommit GitHub、または Bitbucket リポジトリにアップロードします。

      > **Important**
      >
      > *(root directory name)* をアップロードしないでください。アップロードするのは、*(root directory name)* 内のファイルのみです。
      > S3 入力バケットを使用している場合は、ファイルを必ず ZIP ファイルに圧縮してから入力バケットにアップロードしてください。*(root directory name)* を ZIP ファイルに追加しないでください。追加するのは、*(root directory name)* 内のファイルのみです。

   b. ビルドプロジェクトを作成し、ビルドを実行し、関連するビルド情報を表示します。

を使用してビルドプロジェクト AWS CLI を作成する場合、 `create-project` コマンドへの JSON 形式の入力は次のようになります。(プレースホルダは独自の値に置き換えてください。)

```
{
  "name": "sample-go-project",
  "source": {
    "type": "S3",
    "location": "codebuild-region-ID-account-ID-input-bucket/GoSample.zip"
  },
  "artifacts": {
    "type": "S3",
    "location": "codebuild-region-ID-account-ID-output-bucket",
    "packaging": "ZIP",
    "name": "GoOutputArtifact.zip"
  },
  "environment": {
    "type": "LINUX_CONTAINER",
    "image": "aws/codebuild/standard:5.0",
    "computeType": "BUILD_GENERAL1_SMALL"
  },
  "serviceRole": "arn:aws:iam::account-ID:role/role-name",
  "encryptionKey": "arn:aws:kms:region-ID:account-ID:key/key-ID"
}
```

c. ビルド出力アーティファクトを取得するには、S3 出力バケットを開きます。

d. *GoOutputArtifact*.zip ファイルをローカルコンピュータまたはインスタンスへダウンロードし、ファイルの内容を抽出します。展開したコンテンツから、hello ファイルを取得します。

3. 次のいずれかに当てはまる場合は、 が Docker イメージをビルド環境に AWS CodeBuild プルできるように、Amazon ECR のイメージリポジトリにアクセス許可を追加する必要があります。

- プロジェクトでは、 CodeBuild 認証情報を使用して Amazon ECR イメージをプルします。これは、CODEBUILD の `imagePullCredentialsType` 属性で `ProjectEnvironment` の値で示されます。
- プロジェクトでクロスアカウントの Amazon ECR イメージを使用している場合。この場合は、プロジェクトでサービスロールを使用して Amazon ECR イメージをプルする必要があります。この動作を有効にするには、`imagePullCredentialsType` の `ProjectEnvironment` 属性を `SERVICE_ROLE` に設定します。

1. Amazon ECR コンソール (https://console.aws.amazon.com/ecr/) を開きます。
2. リポジトリ名のリストで、作成または選択したリポジトリの名前を選択します。
3. ナビゲーションペインで、[アクセス許可]、[編集]、[ステートメントを追加] の順に選択します。
4. [ステートメント名] で、識別子 (**CodeBuildAccess** など) を入力します。
5. [効果] で、[許可] を選択したままにしておきます。これにより、別の AWS アカウントへのアクセスを許可します。
6. [プリンシパル] で、次のいずれかを実行します。
   - プロジェクトが CodeBuild 認証情報を使用して Amazon ECR イメージをプルする場合は、サービスプリンシパル で と入力します**codebuild.amazonaws.com**。
   - プロジェクトでクロスアカウントの Amazon ECR イメージを使用する場合は、[AWS アカウント ID] に、アクセス権を付与する AWS アカウントの ID を入力します。
7. [すべての IAM エンティティ] リストをスキップします。
8. アクション で、プル専用アクション ecr:GetDownloadUrlForLayer、ecr:BatchGetImage、ecr:BatchCheckLayerAvailability を選択します。
9. [条件] で、以下を追加します。

```
{
   "StringEquals":{
      "aws:SourceAccount":"<AWS-account-ID>",
      "aws:SourceArn":"arn:aws:codebuild:<region>:<AWS-account-
ID>:project/<project-name>"
   }
}
```

10. [Save] を選択します。

このポリシーは [アクセス許可] に表示されます。プリンシパルは、この手順のステップ 3 で [プリンシパル] に入力した値です。

- プロジェクトが CodeBuild 認証情報を使用して Amazon ECR イメージをプルする場合、はサービスプリンシパル の下"codebuild.amazonaws.com"に表示されます。
- プロジェクトでクロスアカウントの Amazon ECR イメージを使用している場合、アクセスを許可する AWS アカウントの ID はAWS アカウント IDs の下に表示されます。

次のサンプルポリシーでは、 CodeBuild 認証情報とクロスアカウントの Amazon ECR イメージの両方を使用します。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"CodeBuildAccessPrincipal",
         "Effect":"Allow",
         "Principal":{
            "Service":"codebuild.amazonaws.com"
         },
         "Action":[
            "ecr:GetDownloadUrlForLayer",
            "ecr:BatchGetImage",
            "ecr:BatchCheckLayerAvailability"
         ],
         "Condition":{
            "StringEquals":{
               "aws:SourceArn":"arn:aws:codebuild:<region>:<aws-account-
id>:project/<project-name>",
               "aws:SourceAccount":"<aws-account-id>"
            }
         }
      },
      {
         "Sid":"CodeBuildAccessCrossAccount",
         "Effect":"Allow",
         "Principal":{
            "AWS":"arn:aws:iam::<AWS-account-ID>:root"
         },
         "Action":[
            "ecr:GetDownloadUrlForLayer",
            "ecr:BatchGetImage",
            "ecr:BatchCheckLayerAvailability"
         ]
      }
   ]
}
```

- プロジェクトで CodeBuild 認証情報を使用し、 CodeBuild プロジェクトに Amazon ECR リポジトリへのオープンアクセスを許可する場合は、Conditionキーを省略して次のサンプルポリシーを追加できます。

```
{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Sid":"CodeBuildAccessPrincipal",
      "Effect":"Allow",
      "Principal":{
        "Service":"codebuild.amazonaws.com"
      },
      "Action":[
        "ecr:GetDownloadUrlForLayer",
        "ecr:BatchGetImage",
        "ecr:BatchCheckLayerAvailability"
      ]
    },
    {
      "Sid":"CodeBuildAccessCrossAccount",
      "Effect":"Allow",
      "Principal":{
        "AWS":"arn:aws:iam::<AWS-account-ID>:root"
      },
      "Action":[
        "ecr:GetDownloadUrlForLayer",
        "ecr:BatchGetImage",
        "ecr:BatchCheckLayerAvailability"
      ]
    }
  ]
}
```

4. ビルドプロジェクトを作成し、ビルドを実行し、ビルド情報を表示します。

   を使用してビルドプロジェクト AWS CLI を作成する場合、 create-project コマンドへの JSON 形式の入力は次のようになります。(プレースホルダは独自の値に置き換えてください。)

```
{
  "name": "amazon-ecr-sample-project",
  "source": {
```

```
    "type": "S3",
    "location": "codebuild-region-ID-account-ID-input-bucket/GoSample.zip"
  },
  "artifacts": {
    "type": "S3",
    "location": "codebuild-region-ID-account-ID-output-bucket",
    "packaging": "ZIP",
    "name": "GoOutputArtifact.zip"
  },
  "environment": {
    "type": "LINUX_CONTAINER",
    "image": "account-ID.dkr.ecr.region-ID.amazonaws.com/your-Amazon-ECR-repo-
name:tag",
    "computeType": "BUILD_GENERAL1_SMALL"
  },
  "serviceRole": "arn:aws:iam::account-ID:role/role-name",
  "encryptionKey": "arn:aws:kms:region-ID:account-ID:key/key-ID"
}
```

5. ビルド出力アーティファクトを取得するには、S3 出力バケットを開きます。
6. *GoOutputArtifact*.zip ファイルをローカルコンピュータまたはインスタンスへダウンロードし、*GoOutputArtifact*.zip ファイルの内容を抽出します。展開したコンテンツから、hello ファイルを取得します。

Go プロジェクトの構造

このサンプルのディレクトリ構造は次のとおりとします。

```
(root directory name)
### buildspec.yml
### hello.go
```

Go プロジェクトのファイル

このサンプルで使用するファイルは以下のとおりです。

buildspec.yml (内)*(root directory name)*

```
version: 0.2

phases:
  install:
```

```
   runtime-versions:
     golang: 1.13
  build:
    commands:
      - echo Build started on `date`
      - echo Compiling the Go code
      - go build hello.go
  post_build:
    commands:
      - echo Build completed on `date`
artifacts:
  files:
    - hello
```

hello.go (内)*(root directory name)*

```
package main
import "fmt"

func main() {
  fmt.Println("hello world")
  fmt.Println("1+1 =", 1+1)
  fmt.Println("7.0/3.0 =", 7.0/3.0)
  fmt.Println(true && false)
  fmt.Println(true || false)
  fmt.Println(!true)
}
```

関連リソース

- の開始方法については AWS CodeBuild、「」を参照してくださいコンソールを使用した AWS CodeBuild の開始方法。
- での問題のトラブルシューティングについては CodeBuild、「」を参照してくださいトラブルシューティング AWS CodeBuild。
- のクォータの詳細については CodeBuild、「」を参照してくださいAWS CodeBuild のクォータ。

の Amazon Elastic File System サンプル AWS CodeBuild

Amazon EC2 インスタンス用のスケーラブルな共有ファイルサービスである Amazon Amazon Elastic File Systemで AWS CodeBuild ビルドを作成することもできます。Amazon EFS のストレー

ジ容量は伸縮自在なため、ファイルの追加および削除に合わせて拡大または縮小されます。また、ファイルシステムを作成、設定するために使用できるシンプルなウェブサービスインターフェイスを提供します。さらに、ファイルストレージインフラストラクチャも自動的に管理されるため、ファイルシステム設定のデプロイ、パッチ適用、保守について心配する必要がありません。詳細については、Amazon Elastic File System ユーザーガイドの「Amazon Elastic File System とは」を参照してください。

このサンプルでは、Java アプリケーションをマウントして Amazon EFS ファイルシステムに構築するように CodeBuild プロジェクトを設定する方法を示します。開始する前に、S3 入力バケットにアップロードされる Java アプリケーション AWS CodeCommit、、 GitHub GitHub エンタープライズサーバー、または Bitbucket リポジトリを作成する準備が必要です。

ファイルシステムの転送中のデータは暗号化されます。別のイメージを使用して転送中のデータを暗号化するには、「転送中のデータの暗号化」を参照してください。

概要ステップ

このサンプルでは、 で Amazon EFS を使用するために必要な 3 つの大まかなステップについて説明します AWS CodeBuild。

1. AWS アカウントに Virtual Private Cloud (VPC) を作成します。
2. この VPC を使用するファイルシステムを作成します。
3. VPC を使用する CodeBuild プロジェクトを作成して構築します。 CodeBuild プロジェクトでは、以下を使用してファイルシステムを識別します。

   - 一意のファイルシステム識別子。ビルドプロジェクトでファイルシステムを指定するときに識別子を選択します。
   - ファイルシステム ID。ID は、Amazon EFS コンソールでファイルシステムを開くと表示されます。
   - マウントポイント。ファイルシステムをマウントする Docker コンテナ内のディレクトリです。
   - マウントオプション。ファイルシステムのマウント方法に関する詳細が含まれます。

> **Note**
>
> Amazon EFS で作成されたファイルシステムは Linux プラットフォームでのみサポートされます。

を使用して VPC を作成する AWS CloudFormation

AWS CloudFormation テンプレートを使用して VPC を作成します。

1. 「」の手順に従ってAWS CloudFormation VPC テンプレート、 AWS CloudFormation を使用して VPC を作成します。

   > **Note**
   >
   > この AWS CloudFormation テンプレートによって作成された VPC には、2 つのプライベートサブネットと 2 つのパブリックサブネットがあります。プライベートサブネットは、 AWS CodeBuild を使用して Amazon EFS で作成したファイルシステムをマウントする場合にのみ使用する必要があります。いずれかのパブリックサブネットを使用する場合、ビルドに失敗します。

2. にサインイン AWS Management Console し、https://console.aws.amazon.com/vpc/ で Amazon VPC コンソールを開きます。
3. で作成した VPC を選択します AWS CloudFormation。
4. [説明] タブに表示される VPC の名前と ID を書き留めます。どちらも、このサンプルの後半で AWS CodeBuild プロジェクトを作成するときに必要です。

VPC を使用した Amazon Elastic File System ファイルシステムを作成する

先ほど作成した VPC を使用して、このサンプルのシンプルな Amazon EFS ファイルシステムを作成します。

1. にサインイン AWS Management Console し、https://console.aws.amazon.com/efs/ で Amazon EFS コンソールを開きます。
2. [Create file system] を選択します。
3. [VPC] で、このサンプルの前のステップで書き留めた VPC 名を選択します。
4. サブネットに関連付けられているアベイラビリティーゾーンの選択したままにしておきます。
5. [Next Step] (次のステップ) をクリックします。
6. [Add tags] (タグの追加) のデフォルトの [Name] (名前) キーにある [Value] (値) に、Amazon EFS ファイルシステムの名前を入力します。

7. デフォルトのパフォーマンスモードおよびスループットモードとして [General Purpose（汎用）] および [Bursting（バースト）] を選択したまま [Next Step（次のステップ）] を選択します。
8. [Configure client access (クライアントアクセスの設定)] で、[Next Step (次のステップ)] を選択します。
9. [Create File System (ファイルシステムの作成)] を選択します。
10. (オプション) 転送時のデータ暗号化を適用するポリシーを、Amazon EFS ファイルシステムに追加することをお勧めします。Amazon EFS コンソールで、[ファイルシステムポリシー]、[編集]、[すべてのクライアントに転送中の暗号化を適用する] ボックス、[保存] の順に選択します。

Amazon EFS で使用する CodeBuild プロジェクトを作成する

このサンプルの前半で作成した VPC を使用する AWS CodeBuild プロジェクトを作成します。ビルドを実行すると、先ほど作成した Amazon EFS ファイルシステムがマウントされます。次に、Java アプリケーションによって作成された .jar ファイルがファイルシステムのマウントポイントディレクトリに保存されます。

1. https://console.aws.amazon.com/codesuite/codebuild/home で AWS CodeBuild コンソールを開きます。
2. ナビゲーションペインで [ビルドプロジェクト] を選択し、次に [ビルドプロジェクトの作成] を選択します。
3. [Project name (プロジェクト名)] にプロジェクトの名前を入力します。
4. [ソースプロバイダー] で、構築する Java アプリケーションが含まれているリポジトリを選択します。
5. がアプリケーションを検索 CodeBuild するために使用するリポジトリ URL などの情報を入力します。オプションはソースプロバイダーごとに異なります。詳細については、「Choose source provider」を参照してください。
6. [環境イメージ] で、[Managed image (マネージド型イメージ)] を選択します。
7. [オペレーティングシステム] で、[Amazon Linux 2] を選択します。
8. [ランタイム] で、[Standard (標準)] を選択します。
9. [イメージ] で、[aws/codebuild/amazonlinux2-x86_64-standard:4.0] を選択します。
10. [環境タイプ] で、[Linux] を選択します。
11. [Service role (サービスロール)] で、[New service role (新しいサービスロール)] を選択します。「ロール名」に、 が CodeBuild 作成するロールの名前を入力します。

12. Additional configuration] (追加設定) を展開します。
13. [Enable this flag if you want to build Docker images or want your builds to get elevated privileges (Docker イメージを構築する場合、またはビルドで昇格された権限を取得する場合は、このフラグを有効にする)] を選択します。

> **Note**
>
> デフォルトでは、Docker デーモンは VPC 以外のビルドで有効になっています。VPC ビルドに Docker コンテナを使用する場合は、Docker Docs ウェブサイトの「ランタイム特権と Linux 機能」を参照して、特権モードを有効にします。また、Windows は特権モードをサポートしていません。

14. [VPC (VPC)] で、VPC ID を選択します。
15. [サブネット] で、VPC に関連付けられているプライベートサブネットのうち 1 つ以上を選択します。Amazon EFS ファイルシステムをマウントするビルドでプライベートサブネットを使用する必要があります。パブリックサブネットを使用している場合、ビルドに失敗します。
16. [Security groups (セキュリティグループ)] で、デフォルトのセキュリティグループを選択します。
17. [ファイルシステム] で、以下の情報を入力します。

- [識別子] に、一意のファイルシステム識別子を入力します。識別子の長さは 129 文字未満である必要があります。英数字とアンダースコアのみを使用できます。 CodeBuild はこの識別子を使用して、Elastic ファイルシステムを識別する環境変数を作成します。環境変数の形式は大文字の `CODEBUILD_<file_system_identifier>` です。たとえば、`my_efs` と入力すると、環境変数は `CODEBUILD_MY_EFS` になります。
- [ID] で、ファイルシステム ID を選択します。
- (オプション) ファイル system. CodeBuild mounts にディレクトリを入力します。ディレクトリパスを空白のままにすると、 CodeBuild はファイルシステム全体をマウントします。パスはファイルシステムのルートからの相対です。
- [マウントポイント] に、ファイルシステムをマウントするディレクトリの絶対パスを入力します。このディレクトリが存在しない場合は、ビルド中に によって CodeBuild 作成されます。
- (オプション) マウントオプションを入力します。マウントオプションを空白のままにすると、はデフォルトのマウントオプション CodeBuild を使用します。

```
nfsvers=4.1
rsize=1048576
```

```
wsize=1048576
hard
timeo=600
retrans=2
```

詳細については、Amazon Elastic File System ユーザーガイドの「NFS の推奨されるマウントオプション」を参照してください。

18. [ビルド仕様] で、[ビルドコマンドの挿入]、[Switch to editor (エディタに切り替え)] の順に選択します。
19. エディタに次のビルド仕様コマンドを入力します。*<file_system_identifier>* をステップ 17 で入力した識別子に置き換えます。大文字を使用します (CODEBUILD_MY_EFS など)。

```
version: 0.2
phases:
  install:
    runtime-versions:
      java: corretto11
  build:
    commands:
      - mvn compile -Dgpg.skip=true -Dmaven.repo.local=
$CODEBUILD_<file_system_identifier>
```

20. 他のすべての設定にはデフォルト値を使用し、[Create build project (ビルドプロジェクトの作成)] を選択します。ビルドが完了すると、プロジェクトのコンソールページが表示されます。
21. [Start build] を選択します。

CodeBuild および Amazon EFS サンプルの概要

AWS CodeBuild プロジェクトが構築された後：

- Java アプリケーションによって作成された .jar ファイルがあります。このファイルは Amazon EFS ファイルシステムのマウントポイントディレクトリにビルドされています。
- ファイルシステムを識別する環境変数は、プロジェクトの作成時に入力したファイルシステム識別子を使用して作成されます。

詳細については、Amazon Elastic File System ユーザーガイドの「ファイルシステムのマウント」を参照してください。

トラブルシューティング

以下は、 で Amazon EFS をセットアップするときに発生する可能性のあるエラーです CodeBuild。

トピック

- CLIENT_ERROR: mounting '127.0.0.1:/' failed. permission denied (クライアントエラー:'127.0.0.1: /' のマウントに失敗しました。パーミッションが拒否されました)
- CLIENT_ERROR: mounting '127.0.0.1:/' failed. connection reset by peer (クライアントエラー:'127.0.0.1: /' のマウントに失敗しました。ピアによって接続がリセットされました)
- VPC_CLIENT_ERROR: 予期しない EC2 エラー： UnauthorizedOperation

CLIENT_ERROR: mounting '127.0.0.1:/' failed. permission denied (クライアントエラー:'127.0.0.1: /' のマウントに失敗しました。パーミッションが拒否されました)

IAM 認証は、 を使用した Amazon EFS のマウントではサポートされていません CodeBuild。カスタム Amazon EFS ファイルシステムポリシーを使用している場合は、すべての IAM プリンシパルへの読み取りおよび書き込みアクセスを許可する必要があります。例:

```
"Principal": {
  "AWS": "*"
}
```

CLIENT_ERROR: mounting '127.0.0.1:/' failed. connection reset by peer (クライアントエラー:'127.0.0.1: /' のマウントに失敗しました。ピアによって接続がリセットされました)

この問題の原因は 2 つ考えられます。

- CodeBuild VPC サブネットが Amazon EFS マウントターゲットとは異なるアベイラビリティーゾーンにある。Amazon EFS マウントターゲットと同じアベイラビリティーゾーンに VPC サブネットを追加することで、この問題を解決できます。
- セキュリティグループには、Amazon EFS と通信する許可がありません。これを解決するには、VPC (VPC のプライマリ CIDR ブロックを追加する) またはセキュリティグループ自体からのすべてのトラフィックを許可するインバウンドルールを追加します。

VPC_CLIENT_ERROR: 予期しない EC2 エラー： UnauthorizedOperation

このエラーは、プロジェクトの VPC 設定内のすべてのサブネットがパブリックサブネットである場合に CodeBuild発生します。ネットワーク接続を確保するには、VPC 内に少なくとも 1 つのプライベートサブネットが必要です。

CodePipeline のサンプル CodeBuild

トピック

- AWS CodePipeline CodeBuild と および バッチビルドとの統合
- AWS CodePipeline CodeBuild と および複数の入力ソースおよび出力アーティファクトとの統合サンプル

AWS CodePipeline CodeBuild と および バッチビルドとの統合

AWS CodeBuild でバッチビルドがサポートされるようになりました。このサンプルでは、 AWS CodePipeline を使用してバッチビルドを使用するビルドプロジェクトを作成する方法を示します。

パイプラインの構造を定義する JSON 形式のファイルを使用し、それを とともに使用 AWS CLI してパイプラインを作成できます。詳細については、『AWS CodePipeline ユーザーガイド』の「AWS CodePipeline パイプライン構造のリファレンス」を参照してください。

個々のアーティファクトを使用した Batch 構築

個別のアーティファクトを含むバッチビルドを作成するパイプライン構造の例として、次の JSON ファイルを使用してください。でバッチビルドを有効にするには CodePipeline、 `configuration` オブジェクトの `BatchEnabled`パラメータを に設定します`true`。

```
{
  "pipeline": {
    "roleArn": "arn:aws:iam::account-id:role/my-AWS-CodePipeline-service-role-name",
    "stages": [
      {
        "name": "Source",
        "actions": [
          {
            "inputArtifacts": [],
            "name": "Source1",
            "actionTypeId": {
              "category": "Source",
              "owner": "AWS",
```

```
            "version": "1",
            "provider": "S3"
          },
          "outputArtifacts": [
            {
              "name": "source1"
            }
          ],
          "configuration": {
            "S3Bucket": "<my-input-bucket-name>",
            "S3ObjectKey": "my-source-code-file-name.zip"
          },
          "runOrder": 1
        },
        {
          "inputArtifacts": [],
          "name": "Source2",
          "actionTypeId": {
            "category": "Source",
            "owner": "AWS",
            "version": "1",
            "provider": "S3"
          },
          "outputArtifacts": [
            {
              "name": "source2"
            }
          ],
          "configuration": {
            "S3Bucket": "<my-other-input-bucket-name>",
            "S3ObjectKey": "my-other-source-code-file-name.zip"
          },
          "runOrder": 1
        }
      ]
    },
    {
      "name": "Build",
      "actions": [
        {
          "inputArtifacts": [
            {
              "name": "source1"
            },
```

```
            {
              "name": "source2"
            }
          ],
          "name": "Build",
          "actionTypeId": {
            "category": "Build",
            "owner": "AWS",
            "version": "1",
            "provider": "CodeBuild"
          },
          "outputArtifacts": [
            {
              "name": "build1"
            },
            {
              "name": "build1_artifact1"
            },
            {
              "name": "build1_artifact2"
            },
            {
              "name": "build2_artifact1"
            },
            {
              "name": "build2_artifact2"
            }
          ],
          "configuration": {
            "ProjectName": "my-build-project-name",
            "PrimarySource": "source1",
            "BatchEnabled": "true"
          },
          "runOrder": 1
        }
      ]
    }
  ],
  "artifactStore": {
    "type": "S3",
    "location": "<AWS-CodePipeline-internal-bucket-name>"
  },
  "name": "my-pipeline-name",
  "version": 1
```

```
  }
}
```

このパイプライン設定で動作する CodeBuild buildspec ファイルの例を次に示します。

```
version: 0.2
batch:
  build-list:
    - identifier: build1
      env:
        compute-type: BUILD_GENERAL1_SMALL
    - identifier: build2
      env:
        compute-type: BUILD_GENERAL1_MEDIUM

phases:
  build:
    commands:
      - echo 'file' > output_file

artifacts:
  files:
    - output_file
  secondary-artifacts:
    artifact1:
      files:
        - output_file
    artifact2:
      files:
        - output_file
```

パイプラインの JSON ファイルで指定されている出力成果物の名前は、buildspec ファイルで定義されているビルドおよびアーティファクトの識別子と一致していなければなりません。構文は、プライマリアーティファクトの場合は *buildIdentifier* で、セカンダリアーティファクトの場合は *buildIdentifier_artifactIdentifier* です。

例えば、出力アーティファクト名 の場合build1、 のプライマリアーティファクトbuild1を の場所 CodeBuild にアップロードしますbuild1。出力名 の場合build1_artifact1、 artifact1のセカンダリアーティファクトbuild1を の場所 CodeBuild にアップロードbuild1_artifact1します。出力場所が 1 つだけ指定されている場合、名前は *buildIdentifier* のみにします。

JSON ファイルを作成したら、パイプラインを作成することができます。を使用して create-pipeline コマンド AWS CLI を実行し、 ファイルを `--cli-input-json`パラメータに渡します。詳細については、『AWS CodePipeline ユーザーガイド』の「パイプラインの作成 (CLI)」を参照してください。

複合アーチファクトを使用したBatch ビルド

結合アーティファクトを含むバッチビルドを作成するパイプライン構造の例として、次の JSON ファイルを使用してください。でバッチビルドを有効にするには CodePipeline、 `configuration` オブジェクトの `BatchEnabled`パラメータを に設定します`true`。ビルド成果物を同じ場所に結合するには、「`CombineArtifacts`」オブジェクトの「`configuration`」パラメータのパラメータを「`true`」に設置します。

```
{
 "pipeline": {
  "roleArn": "arn:aws:iam::account-id:role/my-AWS-CodePipeline-service-role-name",
  "stages": [
    {
      "name": "Source",
      "actions": [
        {
          "inputArtifacts": [],
          "name": "Source1",
          "actionTypeId": {
            "category": "Source",
            "owner": "AWS",
            "version": "1",
            "provider": "S3"
          },
          "outputArtifacts": [
            {
              "name": "source1"
            }
          ],
          "configuration": {
            "S3Bucket": "<my-input-bucket-name>",
            "S3ObjectKey": "my-source-code-file-name.zip"
          },
          "runOrder": 1
        },
        {
          "inputArtifacts": [],
```

```
          "name": "Source2",
          "actionTypeId": {
            "category": "Source",
            "owner": "AWS",
            "version": "1",
            "provider": "S3"
          },
          "outputArtifacts": [
            {
              "name": "source2"
            }
          ],
          "configuration": {
            "S3Bucket": "<my-other-input-bucket-name>",
            "S3ObjectKey": "my-other-source-code-file-name.zip"
          },
          "runOrder": 1
        }
      ]
    },
    {
      "name": "Build",
      "actions": [
        {
          "inputArtifacts": [
            {
              "name": "source1"
            },
            {
              "name": "source2"
            }
          ],
          "name": "Build",
          "actionTypeId": {
            "category": "Build",
            "owner": "AWS",
            "version": "1",
            "provider": "CodeBuild"
          },
          "outputArtifacts": [
            {
              "name": "output1 "
            }
          ],
```

```
          "configuration": {
            "ProjectName": "my-build-project-name",
            "PrimarySource": "source1",
             "BatchEnabled": "true",
             "CombineArtifacts": "true"
          },
          "runOrder": 1
        }
      ]
    }
  ],
  "artifactStore": {
    "type": "S3",
    "location": "<AWS-CodePipeline-internal-bucket-name>"
  },
  "name": "my-pipeline-name",
  "version": 1
 }
}
```

このパイプライン設定で動作する CodeBuild buildspec ファイルの例を次に示します。

```
version: 0.2
batch:
  build-list:
    - identifier: build1
      env:
        compute-type: BUILD_GENERAL1_SMALL
    - identifier: build2
      env:
        compute-type: BUILD_GENERAL1_MEDIUM

phases:
  build:
    commands:
      - echo 'file' > output_file

artifacts:
  files:
    - output_file
```

バッチビルドで結合アーティファクトが有効になっている場合、許可される出力は 1 つだけです。CodeBuild は、すべてのビルドのプライマリアーティファクトを 1 つの ZIP ファイルに結合します。

JSON ファイルを作成したら、パイプラインを作成することができます。を使用して create-pipeline コマンド AWS CLI を実行し、 ファイルを `--cli-input-json`パラメータに渡します。詳細については、『AWS CodePipeline ユーザーガイド』の「パイプラインの作成 (CLI)」を参照してください。

AWS CodePipeline CodeBuild と および複数の入力ソースおよび出力アーティファクトとの統合サンプル

AWS CodeBuild プロジェクトは複数の入力ソースを使用できます。また、複数の出力アーティファクトを作成することもできます。このサンプルでは、 AWS CodePipeline を使用して、複数の入力ソースを使用して複数の出力アーティファクトを作成するビルドプロジェクトを作成する方法を示します。詳細については、「複数の入力ソースと出力アーティファクトのサンプル」を参照してください。

パイプラインの構造を定義する JSON 形式のファイルを使用し、それを とともに使用 AWS CLI してパイプラインを作成できます。複数の入力ソースと複数の出力アーティファクトを含むビルドを作成するパイプライン構造の例として、次の JSON ファイルを使用してください。このサンプルの後半では、このファイルが複数の入力と出力をどのように指定しているかが分かります。詳細については、「 ユーザーガイド」のCodePipeline 「パイプライン構造リファレンスAWS CodePipeline 」を参照してください。

```
{
 "pipeline": {
  "roleArn": "arn:aws:iam::account-id:role/my-AWS-CodePipeline-service-role-name",
  "stages": [
    {
      "name": "Source",
      "actions": [
        {
          "inputArtifacts": [],
          "name": "Source1",
          "actionTypeId": {
            "category": "Source",
            "owner": "AWS",
            "version": "1",
            "provider": "S3"
          },
```

```
        "outputArtifacts": [
          {
            "name": "source1"
          }
        ],
        "configuration": {
          "S3Bucket": "my-input-bucket-name",
          "S3ObjectKey": "my-source-code-file-name.zip"
        },
        "runOrder": 1
      },
      {
        "inputArtifacts": [],
        "name": "Source2",
        "actionTypeId": {
          "category": "Source",
          "owner": "AWS",
          "version": "1",
          "provider": "S3"
        },
        "outputArtifacts": [
          {
            "name": "source2"
          }
        ],
        "configuration": {
          "S3Bucket": "my-other-input-bucket-name",
          "S3ObjectKey": "my-other-source-code-file-name.zip"
        },
        "runOrder": 1
      }
    ]
  },
  {
    "name": "Build",
    "actions": [
      {
        "inputArtifacts": [
          {
            "name": "source1"
          },
          {
            "name": "source2"
          }
```

```
          ],
          "name": "Build",
          "actionTypeId": {
            "category": "Build",
            "owner": "AWS",
            "version": "1",
            "provider": "AWS CodeBuild"
          },
          "outputArtifacts": [
            {
              "name": "artifact1"
            },
            {
              "name": "artifact2"
            }
          ],
          "configuration": {
            "ProjectName": "my-build-project-name",
            "PrimarySource": "source1"
          },
          "runOrder": 1
        }
      ]
    }
  ],
  "artifactStore": {
    "type": "S3",
    "location": "AWS-CodePipeline-internal-bucket-name"
  },
  "name": "my-pipeline-name",
  "version": 1
 }
}
```

この JSON ファイルの制約事項:

- 入力ソースの 1 つを PrimarySourceに指定する必要があります。このソースは、が buildspec ファイル CodeBuild を検索して実行するディレクトリです。キーワードPrimarySourceは、JSON ファイルの CodeBuild ステージの configurationセクションでプライマリソースを指定するために使用されます。

- 各入力ソースは、それぞれのディレクトリにインストールされます。このディレクトリは、組み込み環境変数 $CODEBUILD_SRC_DIR (プライマリソースの場合) と $CODEBUILD_SRC_DIR_yourInputArtifactName (他のすべてのソースの場合) に保存されます。このサンプルのパイプラインでは、2 つの入力ソースディレクトリは $CODEBUILD_SRC_DIR と $CODEBUILD_SRC_DIR_source2 です。詳細については、「ビルド環境の環境変数」を参照してください。
- パイプラインの JSON ファイルで指定されている出力成果物の名前は、buildspec ファイルで定義されているセカンダリアーティファクトの名前と一致していなければなりません。このパイプラインは、次の buildspec ファイルを使用します。詳細については、「buildspec の構文」を参照してください。

```
version: 0.2

phases:
  build:
    commands:
      - touch source1_file
      - cd $CODEBUILD_SRC_DIR_source2
      - touch source2_file

artifacts:
  files:
    - '**/*'
  secondary-artifacts:
    artifact1:
      base-directory: $CODEBUILD_SRC_DIR
      files:
        - source1_file
    artifact2:
      base-directory: $CODEBUILD_SRC_DIR_source2
      files:
        - source2_file
```

JSON ファイルを作成したら、パイプラインを作成することができます。を使用して create-pipeline コマンド AWS CLI を実行し、 ファイルを `--cli-input-json`パラメータに渡します。詳細については、『AWS CodePipeline ユーザーガイド』の「パイプラインの作成 (CLI)」を参照してください。

CodeBuild サンプル AWS Config で を使用する

AWS Config は、 AWS リソースのインベントリと、これらのリソースに対する設定変更の履歴を提供します。 は、 AWS リソース AWS CodeBuild として をサポートする AWS Config ようになりました。つまり、サービスは CodeBuild プロジェクトを追跡できます。の詳細については AWS Config、「 AWS Config デベロッパーガイド」の「 とは AWS Config」を参照してください。

CodeBuild リソースに関する以下の情報は、 AWS Config コンソールのリソースインベントリページで確認できます。

- CodeBuild 設定変更のタイムライン。
- 各 CodeBuild プロジェクトの設定の詳細。
- 他の AWS リソースとの関係。
- CodeBuild プロジェクトに対する変更のリスト。

このトピックの手順では、プロジェクトをセットアップ AWS Config して検索および表示する CodeBuild方法について説明します。

トピック

- 前提条件
- セットアップ AWS Config
- AWS CodeBuild プロジェクトの検索
- コンソールで AWS CodeBuildAWS Config の設定の詳細の表示

前提条件

AWS CodeBuild プロジェクトを作成します。手順については、「ビルドプロジェクトの作成」を参照してください。

セットアップ AWS Config

- AWS Config のセットアップ (コンソール)
- AWS Config （ AWS CLI) のセットアップ

> **Note**
>
> セットアップが完了すると、 AWS Config コンソールに AWS CodeBuild プロジェクトが表示されるまでに最大 10 分かかる場合があります。

AWS CodeBuild プロジェクトの検索

1. AWS マネジメントコンソールにサインインし、https://console.aws.amazon.com/config で AWS Config コンソールを開きます。
2. リソースインベントリページで、リソースタイプ の下のAWS CodeBuild プロジェクトを選択します。 下にスクロールし、CodeBuild プロジェクトチェックボックスを選択します。
3. [検索] を選択します。
4. CodeBuild プロジェクトのリストを追加したら、Config タイムライン列で CodeBuild プロジェクト名のリンクを選択します。

コンソールで AWS CodeBuildAWS Config の設定の詳細の表示

リソースインベントリページでリソースを検索するときに、 AWS Config タイムラインを選択して CodeBuild プロジェクトの詳細を表示できます。リソースの詳細ページは、リソースの設定、関係、および変更回数の情報を提供します。

ページの上部にあるブロックは、まとめてタイムラインと呼ばれます。タイムラインは、記録を取った日付と時刻を示します。

詳細については、「 デベロッパーガイド」の「 AWS Config コンソールで設定の詳細を表示する」を参照してください。 AWS Config

のビルド通知のサンプル CodeBuild

Amazon CloudWatch Events には、 のサポートが組み込まれています AWS CodeBuild。CloudWatch Events は、 AWS リソースの変更を説明するシステムイベントのストリームです。CloudWatch イベントでは、対象となるイベントを実行される自動アクションに関連付ける宣言ルールを記述します。このサンプルでは、Amazon CloudWatch Events と Amazon Simple Notification Service (Amazon SNS) を使用して、ビルドが成功、失敗、あるビルドフェーズから別のビルドフェーズへの移行、またはこれらのイベントの任意の組み合わせを行うたびに、サブスクライバーにビルド通知を送信します。

> **Important**
>
> このサンプルを実行すると、 AWS アカウントに料金が発生する可能性があります。これには、Amazon CodeBuild および Amazon SNS に関連する AWS リソースやアクションに対する CloudWatch および の料金が含まれます。詳細については、「 のCodeBuild 料金」、「Amazon CloudWatch の料金」、および「Amazon SNS の料金」を参照してください。

サンプルの実行

このサンプルを実行するには

1. このサンプルで使用するトピックをすでに設定して Amazon SNS で購読している場合は、ステップ 4 に進みます。それ以外の場合、 AWS ルートアカウントまたは管理者ユーザーの代わりに IAM ユーザーを使用して Amazon SNS を操作する場合は、ユーザー (またはユーザーが関連付けられている IAM グループ) に次のステートメントを (*### BEGIN ADDING STATEMENT HERE ###* と *### END ADDING STATEMENT HERE ###* の間に) 追加します。 AWS ルートアカウントの使用は推奨されません。このステートメントにより、Amazon SNS のトピックへの通知の表示、作成、サブスクライブ、および送信テストができます。省略記号 (`...`) は、簡潔にするために使用され、ステートメントを追加する場所の特定に役立ちます。ステートメントを削除しないでください、また、これらの省略記号を既存のポリシーに入力しないでください。

```
{
  "Statement": [
    ### BEGIN ADDING STATEMENT HERE ###
    {
      "Action": [
        "sns:CreateTopic",
        "sns:GetTopicAttributes",
        "sns:List*",
        "sns:Publish",
        "sns:SetTopicAttributes",
        "sns:Subscribe"
      ],
      "Resource": "*",
      "Effect": "Allow"
    },
    ### END ADDING STATEMENT HERE ###
    ...
```

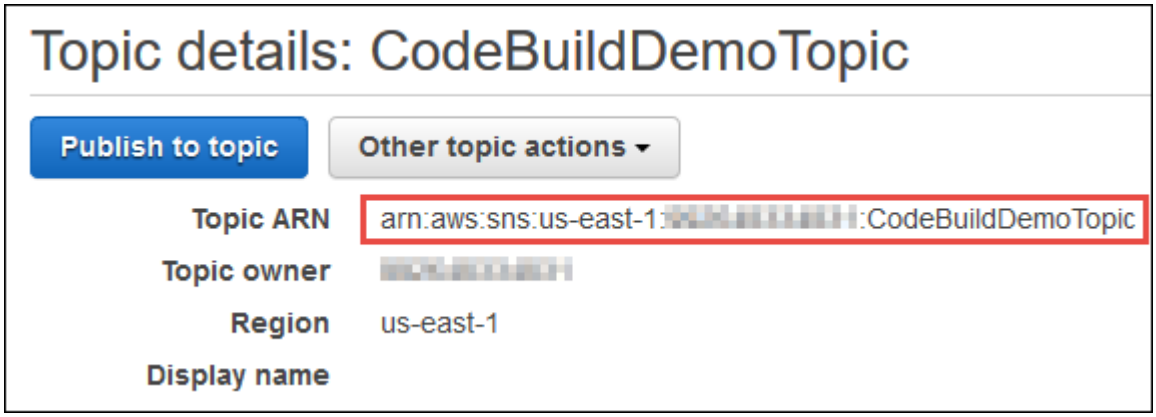
```
  ],
  "Version": "2012-10-17"
}
```

> **Note**
>
> このポリシーを変更する IAM エンティティは、ポリシーを変更するために IAM のアクセス許可を持っている必要があります。
> 詳細については、「カスタマー管理ポリシーの編集」または、「IAM ユーザーガイド」の「インラインポリシーの使用 (コンソール)」の「グループ、ユーザー、ロールのインラインポリシーを編集または削除するには」セクションを参照してください。

2. Amazon SNS でトピックを作成または識別します。 AWS CodeBuild イベントを使用して CloudWatch 、Amazon SNS を介してこのトピックにビルド通知を送信します。

   トピックを作成するには:

   1. Amazon SNS コンソール (https://console.aws.amazon.com/sns) を開きます。
   2. [トピックの作成] を選択します。
   3. [新しいトピックの作成] で、[トピック名] にトピックの名前 (**CodeBuildDemoTopic** など) を入力します。(別の名前を選択する場合は、このサンプル全体でそれを置き換えてください。)
   4. [トピックの作成] を選択します。
   5. トピックの詳細： CodeBuildDemoTopic ページで、トピック ARN 値をコピーします。この値は次のステップで必要になります。

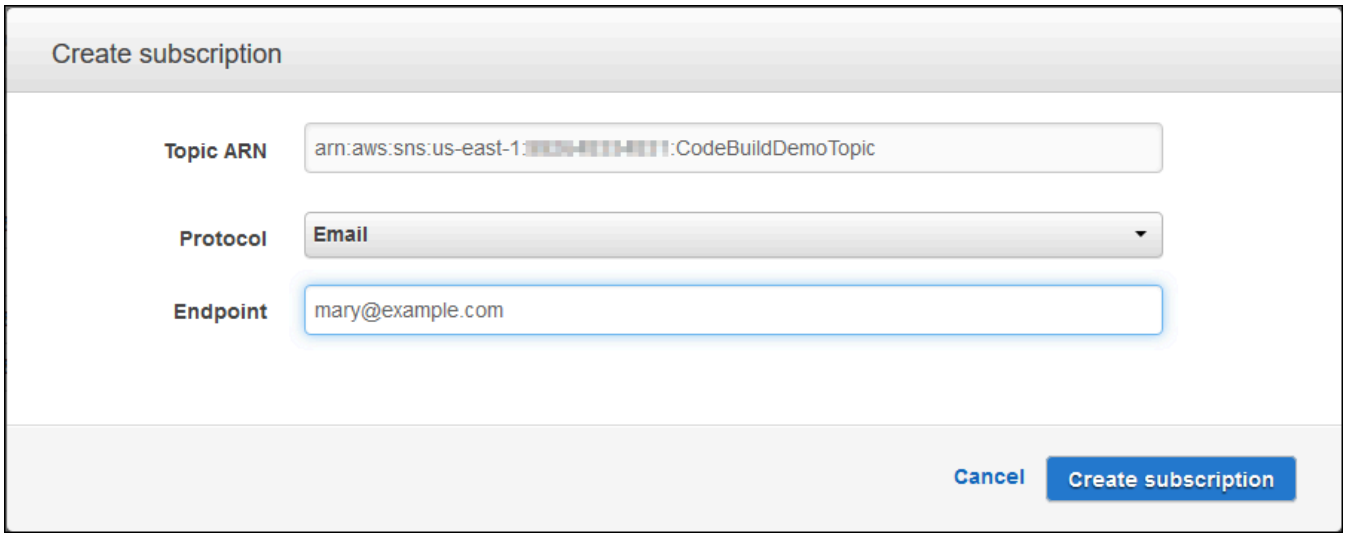


詳細については、Amazon SNS デベロッパーガイドの「トピックの作成」を参照してください。

3. 1 つかそれ以上の受信者にトピックをサブスクライブさせ、E メール通知を受け取ります。

   受信者にトピックをサブスクライブさせるには:

   1. 前のステップで Amazon SNS コンソールを開いた状態のまま、ナビゲーションペインで、[Subscriptions] (サブスクリプション) を選択してから、[Create subscription] (サブスクリプションの作成) を選択します。
   2. [サブスクリプションの作成] の [トピック ARN] に、前のステップからコピーしたトピック ARN を貼り付けます。
   3. [Protocol] で [Email] を選択します。
   4. [エンドポイント] に、受信者の完全な E メールアドレスを入力します。



   5. [Create Subscription] を選択します。
   6. Amazon SNS は受信者にサブスクリプション確認の E メールを送信します。E メール通知の受信を開始するには、受信者は受信登録確認メールで [Confirm subscription] リンクを選択する必要があります。受信者がリンクをクリックした後、正常にサブスクライブされたら、Amazon SNS により受信者のウェブブラウザに確認メッセージが表示されます。

   詳細については、Amazon SNS 開発者ガイドの「トピックのサブスクライブ」を参照してください。

4. AWS ルートアカウントまたは管理者ユーザーの代わりに ユーザーを使用して CloudWatch イベントを操作する場合は、ユーザー (またはユーザーが関連付けられている IAM グループ) に次のステートメントを (*### BEGIN ADDING STATEMENT HERE ###* と *### END ADDING STATEMENT HERE ###* の間に) 追加します。 AWS ルートアカウントの使用は推奨されません。このステートメントは、ユーザーが CloudWatch イベントを操作できるようにするために使用されます。省略記号 (. . .) は、簡潔にするために使用され、ステートメントを追加する場所の特定に役立ちます。ステートメントを削除しないでください、また、これらの省略記号を既存のポリシーに入力しないでください。

```
{
  "Statement": [
    ### BEGIN ADDING STATEMENT HERE ###
    {
      "Action": [
        "events:*",
        "iam:PassRole"
      ],
      "Resource": "*",
      "Effect": "Allow"
    },
    ### END ADDING STATEMENT HERE ###
    ...
  ],
  "Version": "2012-10-17"
}
```

> **Note**
>
> このポリシーを変更する IAM エンティティは、ポリシーを変更するために IAM のアクセス許可を持っている必要があります。
> 詳細については、「カスタマー管理ポリシーの編集」または、「IAM ユーザーガイド」の「インラインポリシーの使用 (コンソール)」の「グループ、ユーザー、ロールのインラインポリシーを編集または削除するには」セクションを参照してください。

5. CloudWatch イベントでルールを作成します。これを行うには、https://console.aws.amazon.com/cloudwatch で CloudWatch コンソールを開きます。
6. ナビゲーションペインの [Events] で、[Rules] を選択してから、[Create rule] を選択します。
7. [ステップ 1: ルールの作成] ページで、[イベントパターン] と [サービス別のイベントに一致するイベントパターンの構築] が選択済みであることを確認します。

8. [Service Name] (サービス名)には CodeBuild を選択します。[イベントタイプ] で、[すべてのイベント] が選択済みであることを確認します。
9. [イベントパターンのプレビュー] には、次のコードが表示されます。

```
{
  "source": [
    "aws.codebuild"
  ]
}
```

10. [編集] を選択し、[イベントパターンのプレビュー] のコードを、次の 2 つのルールパターンのいずれかに置き換えます。

    この最初のルールパターンは、 AWS CodeBuildで指定されたビルドプロジェクトのビルドが開始または完了すると、イベントをトリガーします。

```
{
  "source": [
    "aws.codebuild"
  ],
  "detail-type": [
    "CodeBuild Build State Change"
  ],
  "detail": {
    "build-status": [
      "IN_PROGRESS",
      "SUCCEEDED",
      "FAILED",
      "STOPPED"
    ],
    "project-name": [
      "my-demo-project-1",
      "my-demo-project-2"
    ]
  }
}
```

    前述のルールで、必要に応じて次のコードを変更します。

    - ビルドが開始または完了したときにイベントをトリガーするには、build-status 配列に表示されているすべての値をそのままにするか、build-status 配列を完全に削除します。

- ビルドが完了したときにのみイベントをトリガーするには、IN_PROGRESS 配列から build-status を削除します。
- ビルドの開始時にのみイベントをトリガーするには、IN_PROGRESS 配列から build-status を除くすべての値を削除します。
- すべてのビルドプロジェクトのイベントをトリガーするには、project-name 配列を完全に削除します。
- 個々のビルドプロジェクトのイベントのみをトリガーするには、project-name 配列に各ビルドプロジェクトの名前を指定します。

この 2 番目のルールパターンでは、 AWS CodeBuildで指定されたビルドプロジェクトのビルドフェーズが別のビルドフェーズに移動するたびに、イベントをトリガーします。

```
{
  "source": [
    "aws.codebuild"
  ],
  "detail-type": [
    "CodeBuild Build Phase Change"
  ],
  "detail": {
    "completed-phase": [
      "SUBMITTED",
      "PROVISIONING",
      "DOWNLOAD_SOURCE",
      "INSTALL",
      "PRE_BUILD",
      "BUILD",
      "POST_BUILD",
      "UPLOAD_ARTIFACTS",
      "FINALIZING"
    ],
    "completed-phase-status": [
      "TIMED_OUT",
      "STOPPED",
      "FAILED",
      "SUCCEEDED",
      "FAULT",
      "CLIENT_ERROR"
    ],
    "project-name": [
```

```
      "my-demo-project-1",
      "my-demo-project-2"
    ]
  }
}
```

前述のルールで、必要に応じて次のコードを変更します。

- ビルドフェーズの変更 (各ビルドで送信される通知は最大 9 個) ごとにイベントをトリガーするには、completed-phase 配列に表示されているすべての値をそのままにするか、completed-phase 配列を完全に削除します。
- 個々のビルドフェーズの変更に対してのみイベントをトリガーするには、イベントをトリガーしない completed-phase 配列の各ビルドフェーズの名前を削除します。
- 各ビルドフェーズステータスが変更するたびにイベントをトリガーするには、completed-phase-status 配列に示すように、すべて値をそのままにするか、completed-phase-status 配列を完全に削除します。
- 個々のビルドフェーズステータスの変更に対してのみイベントをトリガーするには、イベントをトリガーしない completed-phase-status 配列の各ビルドフェーズステータスの名前を削除します。
- すべてのビルドプロジェクトのイベントをトリガーするには、project-name 配列を削除します。
- 個々のビルドプロジェクトのイベントをトリガーするには、project-name 配列に各ビルドプロジェクトの名前を指定します。

イベントパターンの詳細については、「Amazon EventBridge ユーザーガイド」の「イベントパターン」を参照してください。

イベントパターンによるフィルタリングの詳細については、「Amazon EventBridge ユーザーガイド」の「イベントパターンによるコンテンツベースのフィルタリング」を参照してください。

> **Note**
>
> ビルド状態の変更とビルドフェーズの変更の両方に応じてイベントをトリガーする場合は、ビルド状態の変更用とビルドフェーズの変更用に 2 つの別個のルールを作成する必要があります。両方のルールを 1 つのルールに結合すると、結合したルールは予期しない結果を引き起こすか、まったく動作しなくなる可能性があります。

コードの置換を完了したら、[Save] を選択します。

11. [Targets] で、[Add target] を選択します。
12. ターゲットのリストで、[SNS トピック] を選択します。
13. [Topic] で、以前に指定した、または作成したトピックを選択します。
14. [入力の設定] を展開して、[インプットトランスフォーマー] を閉じます。
15. [Input Path] ボックスに、次のいずれかの入力パスを入力します。

    `detail-type` の値が `CodeBuild Build State Change` であるルールの場合は、次のように入力します。

    ```
    {"build-id":"$.detail.build-id","project-name":"$.detail.project-name","build-status":"$.detail.build-status"}
    ```

    `detail-type` の値が `CodeBuild Build Phase Change` であるルールの場合は、次のように入力します。

    ```
    {"build-id":"$.detail.build-id","project-name":"$.detail.project-name","completed-phase":"$.detail.completed-phase","completed-phase-status":"$.detail.completed-phase-status"}
    ```

    他のタイプの情報を取得するには、「ビルド通知の入力形式に関するリファレンス」を参照してください。
16. [入力テンプレート] ボックスに、次のいずれかの入力テンプレートを入力します。

    `detail-type` の値が `CodeBuild Build State Change` であるルールの場合は、次のように入力します。

    ```
    "Build '<build-id>' for build project '<project-name>' has reached the build status of '<build-status>'."
    ```

    `detail-type` の値が `CodeBuild Build Phase Change` であるルールの場合は、次のように入力します。

    ```
    "Build '<build-id>' for build project '<project-name>' has completed the build phase of '<completed-phase>' with a status of '<completed-phase-status>'."
    ```

17. [設定の詳細] を選択します。
18. [ステップ 2: ルールの詳細を設定する] ページで、名前と説明 (オプション) を入力します。[状態] は、[有効] のままとします。
19. [Create rule] を選択します。
20. ビルドプロジェクトの作成、ビルドの実行、ビルド情報の表示を行います。
21. CodeBuild がビルド通知を正常に送信していることを確認します。たとえば、ビルド通知 E メールが受信トレイにあるかどうかを確認します。

ルールの動作を変更するには、 CloudWatch コンソールで変更するルールを選択し、アクション を選択し、編集 を選択します。ルールを編集し、[設定の詳細]、[ルールの更新] の順に選択します。

ルールを使用してビルド通知を送信するのを停止するには、 CloudWatch コンソールで、使用を停止するルールを選択し、アクション を選択し、次に 無効化 を選択します。

ルールを完全に削除するには、 CloudWatch コンソールで削除するルールを選択し、アクション を選択してから、削除 を選択します。

関連リソース

- の開始方法については AWS CodeBuild、「」を参照してくださいコンソールを使用した AWS CodeBuild の開始方法。
- での問題のトラブルシューティングについては CodeBuild、「」を参照してくださいトラブルシューティング AWS CodeBuild。
- のクォータの詳細については CodeBuild、「」を参照してくださいAWS CodeBuild のクォータ。

ビルド通知の入力形式に関するリファレンス

CloudWatch は JSON 形式で通知を送信します。

ビルド状態変更通知は次の形式を使用します。

```
{
  "version": "0",
  "id": "c030038d-8c4d-6141-9545-00ff7b7153EX",
  "detail-type": "CodeBuild Build State Change",
  "source": "aws.codebuild",
  "account": "123456789012",
  "time": "2017-09-01T16:14:28Z",
  "region": "us-west-2",
```

```
  "resources":[
    "arn:aws:codebuild:us-west-2:123456789012:build/my-sample-project:8745a7a9-
c340-456a-9166-edf953571bEX"
  ],
  "detail":{
    "build-status": "SUCCEEDED",
    "project-name": "my-sample-project",
    "build-id": "arn:aws:codebuild:us-west-2:123456789012:build/my-sample-
project:8745a7a9-c340-456a-9166-edf953571bEX",
    "additional-information": {
      "artifact": {
        "md5sum": "da9c44c8a9a3cd4b443126e823168fEX",
        "sha256sum":
 "6ccc2ae1df9d155ba83c597051611c42d60e09c6329dcb14a312cecc0a8e39EX",
        "location": "arn:aws:s3:::codebuild-123456789012-output-bucket/my-output-
artifact.zip"
      },
      "environment": {
        "image": "aws/codebuild/standard:5.0",
        "privileged-mode": false,
        "compute-type": "BUILD_GENERAL1_SMALL",
        "type": "LINUX_CONTAINER",
        "environment-variables": []
      },
      "timeout-in-minutes": 60,
      "build-complete": true,
      "initiator": "MyCodeBuildDemoUser",
      "build-start-time": "Sep 1, 2017 4:12:29 PM",
      "source": {
        "location": "codebuild-123456789012-input-bucket/my-input-artifact.zip",
        "type": "S3"
      },
      "logs": {
        "group-name": "/aws/codebuild/my-sample-project",
        "stream-name": "8745a7a9-c340-456a-9166-edf953571bEX",
        "deep-link": "https://console.aws.amazon.com/cloudwatch/home?region=us-
west-2#logEvent:group=/aws/codebuild/my-sample-project;stream=8745a7a9-c340-456a-9166-
edf953571bEX"
      },
      "phases": [
        {
          "phase-context": [],
          "start-time": "Sep 1, 2017 4:12:29 PM",
          "end-time": "Sep 1, 2017 4:12:29 PM",
```

```
          "duration-in-seconds": 0,
          "phase-type": "SUBMITTED",
          "phase-status": "SUCCEEDED"
        },
        {
          "phase-context": [],
          "start-time": "Sep 1, 2017 4:12:29 PM",
          "end-time": "Sep 1, 2017 4:13:05 PM",
          "duration-in-seconds": 36,
          "phase-type": "PROVISIONING",
          "phase-status": "SUCCEEDED"
        },
        {
          "phase-context": [],
          "start-time": "Sep 1, 2017 4:13:05 PM",
          "end-time": "Sep 1, 2017 4:13:10 PM",
          "duration-in-seconds": 4,
          "phase-type": "DOWNLOAD_SOURCE",
          "phase-status": "SUCCEEDED"
        },
        {
          "phase-context": [],
          "start-time": "Sep 1, 2017 4:13:10 PM",
          "end-time": "Sep 1, 2017 4:13:10 PM",
          "duration-in-seconds": 0,
          "phase-type": "INSTALL",
          "phase-status": "SUCCEEDED"
        },
        {
          "phase-context": [],
          "start-time": "Sep 1, 2017 4:13:10 PM",
          "end-time": "Sep 1, 2017 4:13:10 PM",
          "duration-in-seconds": 0,
          "phase-type": "PRE_BUILD",
          "phase-status": "SUCCEEDED"
        },
        {
          "phase-context": [],
          "start-time": "Sep 1, 2017 4:13:10 PM",
          "end-time": "Sep 1, 2017 4:14:21 PM",
          "duration-in-seconds": 70,
          "phase-type": "BUILD",
          "phase-status": "SUCCEEDED"
        },
```

```
        {
          "phase-context": [],
          "start-time": "Sep 1, 2017 4:14:21 PM",
          "end-time": "Sep 1, 2017 4:14:21 PM",
          "duration-in-seconds": 0,
          "phase-type": "POST_BUILD",
          "phase-status": "SUCCEEDED"
        },
        {
          "phase-context": [],
          "start-time": "Sep 1, 2017 4:14:21 PM",
          "end-time": "Sep 1, 2017 4:14:21 PM",
          "duration-in-seconds": 0,
          "phase-type": "UPLOAD_ARTIFACTS",
          "phase-status": "SUCCEEDED"
        },
         {
          "phase-context": [],
          "start-time": "Sep 1, 2017 4:14:21 PM",
          "end-time": "Sep 1, 2017 4:14:26 PM",
          "duration-in-seconds": 4,
          "phase-type": "FINALIZING",
          "phase-status": "SUCCEEDED"
        },
        {
          "start-time": "Sep 1, 2017 4:14:26 PM",
          "phase-type": "COMPLETED"
        }
      ]
    },
    "current-phase": "COMPLETED",
    "current-phase-context": "[]",
    "version": "1"
  }
}
```

ビルドフェーズ変更通知は次の形式を使用します。

```
{
  "version": "0",
  "id": "43ddc2bd-af76-9ca5-2dc7-b695e15adeEX",
  "detail-type": "CodeBuild Build Phase Change",
  "source": "aws.codebuild",
```

```
  "account": "123456789012",
  "time": "2017-09-01T16:14:21Z",
  "region": "us-west-2",
  "resources":[
    "arn:aws:codebuild:us-west-2:123456789012:build/my-sample-project:8745a7a9-
c340-456a-9166-edf953571bEX"
  ],
  "detail":{
    "completed-phase": "COMPLETED",
    "project-name": "my-sample-project",
    "build-id": "arn:aws:codebuild:us-west-2:123456789012:build/my-sample-
project:8745a7a9-c340-456a-9166-edf953571bEX",
    "completed-phase-context": "[]",
    "additional-information": {
      "artifact": {
        "md5sum": "da9c44c8a9a3cd4b443126e823168fEX",
        "sha256sum":
 "6ccc2ae1df9d155ba83c597051611c42d60e09c6329dcb14a312cecc0a8e39EX",
        "location": "arn:aws:s3:::codebuild-123456789012-output-bucket/my-output-
artifact.zip"
      },
      "environment": {
        "image": "aws/codebuild/standard:5.0",
        "privileged-mode": false,
        "compute-type": "BUILD_GENERAL1_SMALL",
        "type": "LINUX_CONTAINER",
        "environment-variables": []
      },
      "timeout-in-minutes": 60,
      "build-complete": true,
      "initiator": "MyCodeBuildDemoUser",
      "build-start-time": "Sep 1, 2017 4:12:29 PM",
      "source": {
        "location": "codebuild-123456789012-input-bucket/my-input-artifact.zip",
        "type": "S3"
      },
      "logs": {
        "group-name": "/aws/codebuild/my-sample-project",
        "stream-name": "8745a7a9-c340-456a-9166-edf953571bEX",
        "deep-link": "https://console.aws.amazon.com/cloudwatch/home?region=us-
west-2#logEvent:group=/aws/codebuild/my-sample-project;stream=8745a7a9-c340-456a-9166-
edf953571bEX"
      },
      "phases": [
```

```
        {
          "phase-context": [],
          "start-time": "Sep 1, 2017 4:12:29 PM",
          "end-time": "Sep 1, 2017 4:12:29 PM",
          "duration-in-seconds": 0,
          "phase-type": "SUBMITTED",
          "phase-status": "SUCCEEDED"
        },
        {
          "phase-context": [],
          "start-time": "Sep 1, 2017 4:12:29 PM",
          "end-time": "Sep 1, 2017 4:13:05 PM",
          "duration-in-seconds": 36,
          "phase-type": "PROVISIONING",
          "phase-status": "SUCCEEDED"
        },
        {
          "phase-context": [],
          "start-time": "Sep 1, 2017 4:13:05 PM",
          "end-time": "Sep 1, 2017 4:13:10 PM",
          "duration-in-seconds": 4,
          "phase-type": "DOWNLOAD_SOURCE",
          "phase-status": "SUCCEEDED"
        },
        {
          "phase-context": [],
          "start-time": "Sep 1, 2017 4:13:10 PM",
          "end-time": "Sep 1, 2017 4:13:10 PM",
          "duration-in-seconds": 0,
          "phase-type": "INSTALL",
          "phase-status": "SUCCEEDED"
        },
        {
          "phase-context": [],
          "start-time": "Sep 1, 2017 4:13:10 PM",
          "end-time": "Sep 1, 2017 4:13:10 PM",
          "duration-in-seconds": 0,
          "phase-type": "PRE_BUILD",
          "phase-status": "SUCCEEDED"
        },
        {
          "phase-context": [],
          "start-time": "Sep 1, 2017 4:13:10 PM",
          "end-time": "Sep 1, 2017 4:14:21 PM",
```

```
          "duration-in-seconds": 70,
          "phase-type": "BUILD",
          "phase-status": "SUCCEEDED"
        },
        {
          "phase-context": [],
          "start-time": "Sep 1, 2017 4:14:21 PM",
          "end-time": "Sep 1, 2017 4:14:21 PM",
          "duration-in-seconds": 0,
          "phase-type": "POST_BUILD",
          "phase-status": "SUCCEEDED"
        },
        {
          "phase-context": [],
          "start-time": "Sep 1, 2017 4:14:21 PM",
          "end-time": "Sep 1, 2017 4:14:21 PM",
          "duration-in-seconds": 0,
          "phase-type": "UPLOAD_ARTIFACTS",
          "phase-status": "SUCCEEDED"
        },
        {
          "phase-context": [],
          "start-time": "Sep 1, 2017 4:14:21 PM",
          "end-time": "Sep 1, 2017 4:14:26 PM",
          "duration-in-seconds": 4,
          "phase-type": "FINALIZING",
          "phase-status": "SUCCEEDED"
        },
        {
          "start-time": "Sep 1, 2017 4:14:26 PM",
          "phase-type": "COMPLETED"
        }
      ]
    },
    "completed-phase-status": "SUCCEEDED",
    "completed-phase-duration-seconds": 4,
    "version": "1",
    "completed-phase-start": "Sep 1, 2017 4:14:21 PM",
    "completed-phase-end": "Sep 1, 2017 4:14:26 PM"
  }
}
```

でビルドバッジのサンプル CodeBuild

AWS CodeBuild では、ビルドバッジの使用がサポートされるようになりました。ビルドバッジは、埋め込み可能な動的に生成されたイメージ (バッジ) を提供し、プロジェクトの最新のビルドのステータスを表示します。このイメージには、 CodeBuild プロジェクト用に生成された公開 URL からアクセスできます。これにより、誰でも CodeBuild プロジェクトのステータスを表示できます。ビルドバッジにはセキュリティ情報が含まれないため、認証は不要です。

ビルドバッジが有効化されたビルドプロジェクトの作成 (コンソール)

1. https://console.aws.amazon.com/codesuite/codebuild/home で AWS CodeBuild コンソールを開きます。
2. CodeBuild 情報ページが表示されたら、ビルドプロジェクトの作成を選択します。それ以外の場合は、ナビゲーションペインでビルドを展開し、[ビルドプロジェクト] を選択し、次に [Create build project (ビルドプロジェクトの作成)] を選択します。
3. [プロジェクト名] に、このビルドプロジェクトの名前を入力します。ビルドプロジェクト名は、各 AWS アカウントで一意である必要があります。また、他のユーザーがこのプロジェクトの使用目的を理解できるように、ビルドプロジェクトの説明を任意で指定することもできます。
4. [ソース] の [ソースプロバイダ] で、ソースコードプロバイダタイプを選択し、次のいずれかの操作を行います。

   > **Note**
   >
   > CodeBuild は、Amazon S3 ソースプロバイダーによるビルドバッジをサポートしていません。はアーティファクト転送に Amazon S3 AWS CodePipeline を使用するため、 で作成されたパイプラインの一部であるビルドプロジェクトではビルドバッジはサポートされていません CodePipeline。

   - を選択した場合CodeCommitは、リポジトリ でリポジトリの名前を選択します。[Enable build badge (ビルドバッジを有効にする)] を選択すると、プロジェクトのビルドステータスが表示可能および埋め込み可能になります。
   - を選択した場合はGitHub、指示に従って と接続 (または再接続) します GitHub。「アプリケーション GitHubの承認」ページの「組織アクセス」で、アクセス可能 AWS CodeBuild にする各リポジトリの横にある「アクセスのリクエスト」を選択します。 [Authorize application (アプリケーションの承認)] を選択した後で AWS CodeBuild コンソールに戻り、[リポジトリ] でソースコードが含まれているリポジトリの名前を選択します。[Enable build badge (ビルド

バッジを有効にする)] を選択すると、プロジェクトのビルドステータスが表示可能および埋め込み可能になります。

- [Bitbucket] を選択した場合は、手順に従って Bitbucket に接続 (または再接続) します。Bitbucket の [Confirm access to your account] ページで、[Organization access] の [Grant access] を選択します。アクセス権の付与 を選択したら、 AWS CodeBuild コンソールに戻り、リポジトリ で、ソースコードを含むリポジトリの名前を選択します。[Enable build badge (ビルドバッジを有効にする)] を選択すると、プロジェクトのビルドステータスが表示可能および埋め込み可能になります。

> **Important**
>
> プロジェクトソースを更新すると、プロジェクトのビルドバッジの正確性に影響する場合があります。

5. [環境] で以下の操作を行います。

   [Environment image (環境イメージ)] で、次のいずれかの操作を行います。

   - によって管理される Docker イメージを使用するには AWS CodeBuild、マネージドイメージを選択し、オペレーティングシステム 、ランタイム (s)、イメージ 、イメージバージョン から選択します。利用可能な場合は、[環境タイプ] から選択します。
   - 別の Docker イメージを使用するには、[カスタムイメージ] を選択します。[Environment type (環境タイプ)] で、 [ARM]、[Linux]、[Linux GPU] または [Windows] を選択します。[Other registry (その他のレジストリ)] を選択した場合は、[External registry URL (外部のレジストリ URL)] に *docker repository*/*docker image name* の形式に従って Docker Hub の Docker イメージの名前とタグを入力します。Amazon ECR を選択した場合は、Amazon ECR リポジトリと Amazon ECR イメージを使用して、 AWS アカウントの Docker イメージを選択します。
   - プライベート Docker イメージを使用するには、[カスタムイメージ] を選択します。[Environment type (環境タイプ)] で、 [ARM]、[Linux]、[Linux GPU] または [Windows] を選択します。[Image registry (イメージレジストリ)] に [Other registry (その他のレジストリ)] を選択して、その後プライベート Docker イメージの認証情報の ARN を入力します。認証情報は、Secrets Manager で作成する必要があります。詳細については、AWS Secrets Manager ユーザーガイドの「AWS Secrets Managerとは」を参照してください。

6. [Service role (サービスロール)] で、次のいずれかの操作を行います。

- CodeBuild サービスロールがない場合は、新しいサービスロール を選択します。[Role name] に、新しいロールの名前を入力します。
- CodeBuild サービスロールがある場合は、既存のサービスロール を選択します。[Role ARN] で、サービスロールを選択します。

> **Note**
>
> コンソールを使用してビルドプロジェクトを作成または更新する場合、同時に CodeBuild サービスロールを作成できます。デフォルトでは、ロールはそのビルドプロジェクトでのみ使用できます。コンソールでは、このサービスロールを別のビルドプロジェクトと関連付けると、この別のビルドプロジェクトで使用できるようにロールが更新されます。サービスロールは最大 10 個のビルドプロジェクトで使用できます。

7. [Buildspec] で、次のいずれかを行います。

   - [Use a buildspec file] (ビルド仕様ファイルの使用) を選択して、ソースコードのルートディレクトリの buildspec.yml を使用します。
   - [ビルドコマンドの挿入] を選択して、コンソールを使用してビルドコマンドを挿入します。

   詳細については、「ビルド仕様 (buildspec) に関するリファレンス」を参照してください。
8. [アーティファクト] の [タイプ] で、次のいずれかの操作を行います。

   - ビルド出力アーティファクトを作成しない場合は、[No artifacts (アーティファクトなし)] を選択します。
   - ビルド出力を S3 バケットに保存する場合は、[Amazon S3] を選択して次のいずれかの操作を行います。
     - ビルド出力 ZIP ファイルまたはフォルダにプロジェクト名を使用する場合は、[Name (名前)] を空白のままにします。それ以外の場合は、名前を入力します。デフォルトでは、アーティファクト名はプロジェクト名です。別の名前を使用する場合は、アーティファクト名ボックスに名前を入力します。ZIP ファイルを出力する場合は、zip 拡張子を含めます。
     - [Bucket name (バケット名)] で、出力バケットの名前を選択します。
     - この手順の前の方で [ビルドコマンドの挿入] を選択した場合は、[出力ファイル] に、ビルド出力 ZIP ファイルまたはフォルダに格納するビルドのファイルの場所を入力します。複数の場所の場合は、各場所をコンマで区切ります (例: `appspec.yml, target/my-app.jar`)。詳細については、「`files`」で buildspec の構文 の説明を参照してください。

9. [Additional configuration (追加設定)] オプションを展開し、必要に応じてオプションを選択します。
10. [Create build project (ビルドプロジェクトの作成)] を選択します。[確認] ページで、[ビルドの開始] を選択してビルドを実行します。

ビルドバッジが有効化されたビルドプロジェクトの作成 (CLI)

ビルドプロジェクトの作成の詳細については、「ビルドプロジェクトの作成 (AWS CLI)」を参照してください。ビルドバッジを AWS CodeBuild プロジェクトに含めるには、*badgeEnabled* を `true` の値で指定する必要があります。

AWS CodeBuild ビルドバッジにアクセスする

ビルドバッジにアクセスするには AWS CLI 、 AWS CodeBuild コンソールまたは を使用できます。

- CodeBuild コンソールのビルドプロジェクトのリストの Name 列で、ビルドプロジェクトに対応するリンクを選択します。[ビルドプロジェクト: ***project-name***] ページで、[設定] の [Copy badge URL (バッジ URL のコピー)] を選択します。詳細については、「ビルドプロジェクトの詳細を表示する (コンソール)」を参照してください。
- で AWS CLI、 `batch-get-projects` コマンドを実行します。ビルドバッジの URL は出力のプロジェクト環境の詳細セクションを含まれています。詳細については、「ビルドプロジェクトの詳細を表示する (AWS CLI)」を参照してください。

ビルドバッジのリクエスト URL は共通のデフォルトブランチのものですが、ビルドの実行に使用したソースリポジトリの任意のブランチを指定できます。次に例を示します。

```
https://codebuild.us-east-1.amazon.com/badges?uuid=...&branch=<branch>
```

また、バッジの URL の「`branch`」パラメーターで「`tag`」パラメーターを置き換えることにより、ソースリポジトリからタグを指定することもできます。例:

```
https://codebuild.us-east-1.amazon.com/badges?uuid=...&tag=<tag>
```

CodeBuild ビルドバッジを公開する

マークダウンのイメージのビルドバッジ URL を使用して、マークダウンファイルに最新ビルドのステータスを表示できます。これは、ソースリポジトリの readme.md ファイル (GitHub 例えば、) にある最新のビルドのステータスを表示するのに役立ちます CodeCommit。例:

```
![](<build badge URL>)
```

CodeBuild バッジのステータス

- PASSING 該当するブランチで最新ビルドが成功しました。
- FAILING 該当するブランチで最新ビルドがタイムアウト、失敗、途中終了、または停止しました。
- IN_PROGRESS 該当するブランチで最新ビルドが進行中です。
- UNKNOWN 該当するブランチでプロジェクトがビルドをまだ実行していないか、まったく実行したことがありません。また、ビルドバッジ機能が無効になっている可能性もあります。

サンプル CodeBuild を使用して AWS CLI でテストレポートを作成する

buildspec ファイルで指定したテストは、ビルド中に実行されます。このサンプルでは、 を使用してテストを のビルド AWS CLI に組み込む方法を示します CodeBuild。JUnit を使用して単体テストを作成または、別のツールを使用して構成テストを作成することもできます。その後、テスト結果を評価して、問題を修正したり、アプリケーションを最適化したりできます。

CodeBuild API または AWS CodeBuild コンソールを使用してテスト結果にアクセスできます。このサンプルでは、テスト結果が S3 バケットにエクスポートされるようにレポートを設定する方法を示します。

トピック

- 前提条件
- Create a report group
- レポートグループによるプロジェクトの設定
- レポートの実行と結果の表示

前提条件

- テストケースの作成 このサンプルは、サンプルテストレポートに含めるテストケースがあるという前提で書かれています。buildspec ファイルでテストファイルの場所を指定します。

  以下のテストレポートファイル形式がサポートされています。

  - Cucumber JSON (.json)

- JUnit XML (.xml)
- NUnit XML (.xml)
- NUnit3 XML (.xml)
- TestNG XML (.xml)
- Visual Studio TRX (.trx)
- Visual Studio TRX XML (.xml)

  Surefire JUnit plugin、TestNG、Cucumber などのいずれかの形式でレポートファイルを作成できる任意のテストフレームワークを使用して、テストケースを作成します。
- S3 バケットを作成し、その名前を書き留めます。詳細については、Amazon S3 ユーザーガイドの「S3 バケットを作成する方法」を参照してください。
- IAM ロールを作成し、その ARN を書き留めます。ビルドプロジェクトを作成する際は、ARN が必要です。
- ロールに次の権限がない場合は、追加します。

```
{
    "Effect": "Allow",
    "Resource": [
        "*"
    ],
    "Action": [
        "codebuild:CreateReportGroup",
        "codebuild:CreateReport",
        "codebuild:UpdateReport",
        "codebuild:BatchPutTestCases"
    ]
}
```

詳細については、「テストレポートオペレーションのアクセス許可」を参照してください。

Create a report group

1. `CreateReportGroupInput.json` という名前のファイルを作成します。
2. S3 バケットに、テスト結果をエクスポートするフォルダを作成します。

3. 以下を CreateReportGroupInput.json にコピーします。*<bucket-name>* で、S3 バケットの名前を使用します。*<path-to-folder>* で、S3 バケット内のフォルダへのパスを入力します。

```
{
  "name": "<report-name>",
  "type": "TEST",
  "exportConfig": {
    "exportConfigType": "S3",
    "s3Destination": {
      "bucket": "<bucket-name>",
      "path": "<path-to-folder>",
      "packaging": "NONE"
    }
  }
}
```

4. CreateReportGroupInput.json が含まれているディレクトリで次のコマンドを実行します。

```
aws codebuild create-report-group --cli-input-json file://
CreateReportGroupInput.json
```

出力は次のようになります。reportGroup の ARN を書き留めます。これは、このレポートグループを使用するプロジェクトを作成するときに使用します。

```
{
  "reportGroup": {
    "arn": "arn:aws:codebuild:us-west-2:123456789012:report-group/<report-name>",
    "name": "<report-name>",
    "type": "TEST",
    "exportConfig": {
      "exportConfigType": "S3",
      "s3Destination": {
        "bucket": "<s3-bucket-name>",
        "path": "<folder-path>",
        "packaging": "NONE",
        "encryptionKey": "arn:aws:kms:us-west-2:123456789012:alias/aws/s3"
      }
    },
    "created": 1570837165.885,
    "lastModified": 1570837165.885
```

```
  }
}
```

レポートグループによるプロジェクトの設定

レポートを実行するには、まずレポートグループで設定された CodeBuild ビルドプロジェクトを作成します。レポートグループに指定されたテストケースは、ビルドの実行時に実行されます。

1. `buildspec.yml` という名前の buildspec ファイルを作成します。
2. 次のYAMLを `buildspec.yml` ファイルのテンプレートとして使用します。テストを実行するコマンドを必ず含めてください。`reports` セクションで、テストケースの結果を含むファイルを指定します。これらのファイルには、 でアクセスできるテスト結果が保存されます CodeBuild。作成から 30 日後に有効期限が切れます。これらのファイルは、S3 バケットにエクスポートする生のテストケース結果ファイルとは異なります。

```
version: 0.2
    phases:
    install:
        runtime-versions:
            java: openjdk8
    build:
      commands:
        - echo Running tests
        - <enter commands to run your tests>

    reports:
      <report-name-or-arn>: #test file information
      files:
        - '<test-result-files>'
      base-directory: '<optional-base-directory>'
      discard-paths: false #do not remove file paths from test result files
```

> **Note**
>
> 既存のレポートグループの ARN の代わりに、作成されていないレポートグループの名前を指定することもできます。ARN の代わりに名前を指定すると、 CodeBuild はビルドの実行時にレポートグループを作成します。この名前には、プロジェクト名と buildspec ファイルで指定した名前が `project-name-report-group-name` の形

式で含まれます。詳細については、「テストレポートの作成」および「Report group naming」を参照してください。

3. project.json という名前のファイルを作成します。このファイルには、create-project コマンドの入力が含まれます。
4. 次の JSON を project.json にコピーします。source で、ソースファイルを含むリポジトリのタイプと場所を入力します。serviceRole で、使用しているロールの ARN を指定します。

```
{
  "name": "test-report-project",
  "description": "sample-test-report-project",
  "source": {
    "type": "CODECOMMIT|CODEPIPELINE|GITHUB|S3|BITBUCKET|GITHUB_ENTERPRISE|
NO_SOURCE",
    "location": "<your-source-url>"
  },
  "artifacts": {
    "type": "NO_ARTIFACTS"
  },
  "cache": {
    "type": "NO_CACHE"
  },
  "environment": {
    "type": "LINUX_CONTAINER",
    "image": "aws/codebuild/standard:5.0",
    "computeType": "small"
  },
  "serviceRole": "arn:aws:iam::<your-aws-account-id>:role/service-role/<your-role-
name>"
}
```

5. project.json が含まれているディレクトリで次のコマンドを実行します。これにより、test-project という名前のプロジェクトが作成されます。

```
aws codebuild create-project --cli-input-json file://project.json
```

レポートの実行と結果の表示

このセクションでは、前に作成したプロジェクトのビルドを実行します。ビルドプロセス中に、 はテストケースの結果を含むレポート CodeBuild を作成します。レポートは、指定したレポートグループに含まれます。

1. ビルドを開始するには、次のコマンドを実行します。「test-report-project」は、上記で作成されたビルドプロジェクトの名前です。出力に表示されるビルド ID を書き留めます。

```
aws codebuild start-build --project-name test-report-project
```

2. 次のコマンドを実行して、レポートの ARN を含むビルドに関する情報を取得します。*<build-id>* で、ビルド ID を指定します。出力の「reportArns」プロパティのレポート ARN を書き留めます。

```
aws codebuild batch-get-builds --ids <build-id>
```

3. 次のコマンドを実行して、レポートの詳細を取得します。*<report-arn>* で、レポート ARN を指定します。

```
aws codebuild batch-get-reports --report-arns <report-arn>
```

出力は次のようになります。このサンプル出力は、成功、失敗、スキップされたテスト、エラーの結果、または不明なステータスを返したテストの数を示しています。

```
{
  "reports": [
    {
      "status": "FAILED",
      "reportGroupArn": "<report-group-arn>",
      "name": "<report-group-name>",
      "created": 1573324770.154,
      "exportConfig": {
        "exportConfigType": "S3",
        "s3Destination": {
          "bucket": "<your-S3-bucket>",
          "path": "<path-to-your-report-results>",
          "packaging": "NONE",
          "encryptionKey": "<encryption-key>"
        }
      },
```

```
      "expired": 1575916770.0,
      "truncated": false,
      "executionId": "arn:aws:codebuild:us-west-2:123456789012:build/<name-of-
build-project>:2c254862-ddf6-4831-a53f-6839a73829c1",
      "type": "TEST",
      "arn": "<report-arn>",
      "testSummary": {
        "durationInNanoSeconds": 6657770,
        "total": 11,
        "statusCounts": {
          "FAILED": 3,
          "SKIPPED": 7,
          "ERROR": 0,
          "SUCCEEDED": 1,
          "UNKNOWN": 0
        }
      }
    }
  ],
  "reportsNotFound": []
}
```

4. レポートのテストケースに関する情報を一覧表示するには、次のコマンドを実行します。*<report-arn>* で、レポートの ARN を指定します。オプションの `--filter` パラメータでは、(SUCCEEDED、FAILED、SKIPPED、 ERROR、または UNKNOWN) の 1 つのステータス結果指定できます。

```
aws codebuild describe-test-cases \
    --report-arn <report-arn> \
    --filter status=SUCCEEDED|FAILED|SKIPPED|ERROR|UNKNOWN
```

出力は次のようになります。

```
{
  "testCases": [
    {
      "status": "FAILED",
      "name": "Test case 1",
      "expired": 1575916770.0,
      "reportArn": "<report-arn>",
      "prefix": "Cucumber tests for agent",
      "message": "A test message",
```

```
      "durationInNanoSeconds": 1540540,
      "testRawDataPath": "<path-to-output-report-files>"
    },
    {
      "status": "SUCCEEDED",
      "name": "Test case 2",
      "expired": 1575916770.0,
      "reportArn": "<report-arn>",
      "prefix": "Cucumber tests for agent",
      "message": "A test message",
      "durationInNanoSeconds": 1540540,
      "testRawDataPath": "<path-to-output-report-files>"
    }
  ]
}
```

の Docker サンプル CodeBuild

トピック

- のカスタムイメージサンプルの Docker CodeBuild
- の Amazon Elastic Container Registry イメージリポジトリサンプルに Docker イメージを公開する CodeBuild
- の AWS Secrets Manager サンプルを使用したプライベートレジストリ CodeBuild

のカスタムイメージサンプルの Docker CodeBuild

このサンプルでは、 とカスタム Docker ビルドイメージ (`docker:dind`Docker Hub の) を使用して Docker イメージをビルド AWS CodeBuild して実行します。

Docker サポート CodeBuild で が提供するビルドイメージを使用して Docker イメージを構築する方法については、「」を参照してくださいDocker イメージを Amazon ECR イメージリポジトリサンプルに公開する。

> **Important**
>
> このサンプルを実行すると、 AWS アカウントに料金が発生する可能性があります。これには、Amazon S3 に関連する AWS リソースとアクション AWS KMS、および CloudWatch Logs CodeBuild に対して発生する可能性がある料金が含まれます。 Amazon S3 詳細につ

いては、「 のCodeBuild 料金」、「Amazon S3 の料金」、AWS Key Management Service 「 の料金」、および「Amazon CloudWatch の料金」を参照してください。

トピック

- サンプルの実行
- ディレクトリ構造
- ファイル
- 関連リソース

サンプルの実行

このサンプルを実行するには

1. このトピックの「ディレクトリ構造」セクションと「ファイル」セクションの説明に従ってファイルを作成し、S3 入力バケットにアップロードするか AWS CodeCommit、 GitHub、、または Bitbucket リポジトリにアップロードします。

   > **Important**
   >
   > *(root directory name)* をアップロードしないでください。アップロードするのは、*(root directory name)* 内のファイルのみです。
   > S3 入力バケットを使用している場合は、ファイルを必ず ZIP ファイルに圧縮してから入力バケットにアップロードしてください。*(root directory name)* を ZIP ファイルに追加しないでください。追加するのは、*(root directory name)* 内のファイルのみです。

2. ビルドプロジェクトを作成し、ビルドを実行し、関連するビルド情報を表示します。

   を使用してビルドプロジェクト AWS CLI を作成する場合、 `create-project` コマンドへの JSON 形式の入力は次のようになります。(プレースホルダは独自の値に置き換えてください。)

```
{
  "name": "sample-docker-custom-image-project",
  "source": {
    "type": "S3",
    "location": "codebuild-region-ID-account-ID-input-
bucket/DockerCustomImageSample.zip"
```

```
  },
  "artifacts": {
    "type": "NO_ARTIFACTS"
  },
  "environment": {
    "type": "LINUX_CONTAINER",
    "image": "docker:dind",
    "computeType": "BUILD_GENERAL1_SMALL",
    "privilegedMode": false
  },
  "serviceRole": "arn:aws:iam::account-ID:role/role-name",
  "encryptionKey": "arn:aws:kms:region-ID:account-ID:key/key-ID"
}
```

> **Note**
>
> デフォルトでは、Docker デーモンは VPC 以外のビルドで有効になっています。VPC ビルドに Docker コンテナを使用する場合は、Docker Docs ウェブサイトの「ランタイム特権と Linux 機能」を参照して、特権モードを有効にします。また、Windows は特権モードをサポートしていません。

3. ビルドの結果を表示するには、ビルドの口グで文字列 `Hello, World!` を探します。詳細については、「ビルドの詳細の表示」を参照してください。

ディレクトリ構造

このサンプルのディレクトリ構造は次のとおりとします。

```
(root directory name)
### buildspec.yml
### Dockerfile
```

ファイル

このサンプルで使用されているオペレーティングシステムの基本イメージは Ubuntu です。このサンプルで使用するファイルは以下のとおりです。

`buildspec.yml` (内)*(root directory name)*

```
version: 0.2
```

```
phases:
  pre_build:
    commands:
      - docker build -t helloworld .
  build:
    commands:
      - docker images
      - docker run helloworld echo "Hello, World!"
```

`Dockerfile` (内)*(root directory name)*

```
FROM maven:3.3.9-jdk-8

RUN echo "Hello World"
```

関連リソース

- の開始方法については AWS CodeBuild、「」を参照してくださいコンソールを使用した AWS CodeBuild の開始方法。
- での問題のトラブルシューティングについては CodeBuild、「」を参照してくださいトラブルシューティング AWS CodeBuild。
- のクォータの詳細については CodeBuild、「」を参照してくださいAWS CodeBuild のクォータ。

の Amazon Elastic Container Registry イメージリポジトリサンプルに Docker イメージを公開する CodeBuild

このサンプルでは、Docker イメージをビルド出力として生成し、Docker イメージを Amazon Elastic Container Registry (Amazon ECR) イメージリポジトリにプッシュします。このサンプルを適応させて、Docker イメージを Docker Hub にプッシュすることができます。詳細については、「イメージを Docker Hub にプッシュするためのサンプルの調整」を参照してください。

カスタム Docker ビルドイメージ (Docker Hub の `docker:dind`) を使用して Docker イメージをビルドする方法については、「カスタム Docker イメージのサンプル」を参照してください。

このサンプルは、`golang:1.12` を参照してテストされています。

このサンプルでは、新しいマルチステージの Docker ビルド機能を使用しています。この機能により、Docker イメージがビルド出力として生成されます。次に、Docker イメージがAmazon ECR イメージリポジトリにプッシュされます。マルチステージの Docker イメージビルドは、最終的な

Docker イメージのサイズを縮小するのに役立ちます。詳細については、「Docker でのマルチステージビルドの使用」を参照してください。

> **Important**
>
> このサンプルを実行すると、 AWS アカウントに課金される場合があります。これには、Amazon S3、、 CloudWatch Logs AWS KMS、Amazon ECR に関連する AWS リソースやアクション AWS CodeBuild に対する および の料金が含まれます。詳細については、「 のCodeBuild 料金」、「Amazon S3 の料金」、AWS Key Management Service 「 の料金」、「Amazon の CloudWatch 料金」、「Amazon Elastic Container Registry の料金」を参照してください。

トピック

- サンプルの実行
- ディレクトリ構造
- ファイル
- イメージを Docker Hub にプッシュするためのサンプルの調整
- 関連リソース

サンプルの実行

このサンプルを実行するには

1. 使用する Amazon ECR にイメージリポジトリがすでにある場合は、ステップ 3 に進みます。それ以外の場合、 AWS ルートアカウントまたは管理者ユーザーの代わりに ユーザーを使用して Amazon ECR を操作する場合は、このステートメントを (*### BEGIN ADDING STATEMENT HERE ###* と *### END ADDING STATEMENT HERE ###* の間に) ユーザー (またはユーザーが関連付けられている IAM グループ) に追加します。 AWS ルートアカウントの使用は推奨されません。このステートメントでは、Docker イメージを保存するための Amazon ECR リポジトリを作成できます。省略記号 (. . .) は、簡潔にするために使用され、ステートメントを追加する場所の特定に役立ちます。ステートメントを削除しないでください、また、これらの省略記号をポリシーに入力しないでください。詳細については、ユーザーガイドの「AWS Management Consoleでのインラインポリシーの使用」を参照してください。

```
{
  "Statement": [
```

```
    ### BEGIN ADDING STATEMENT HERE ###
    {
      "Action": [
        "ecr:CreateRepository"
      ],
      "Resource": "*",
      "Effect": "Allow"
    },
    ### END ADDING STATEMENT HERE ###
    ...
  ],
  "Version": "2012-10-17"
}
```

> **Note**
>
> このポリシーを変更する IAM エンティティは、ポリシーを変更するために IAM のアクセス許可を持っている必要があります。

2. Amazon ECR にイメージリポジトリを作成します。リポジトリは、ビルド環境を作成してビルドを実行するリージョンと同じ AWS リージョンに作成してください。詳細については、Amazon ECR ユーザーガイドの「リポジトリの作成」を参照してください。このリポジトリの名前は、この手順で後ほど IMAGE_REPO_NAME 環境変数を使用して指定するリポジトリ名と一致させる必要があります。Amazon ECR リポジトリポリシーが、 CodeBuild サービス IAM ロールにイメージプッシュアクセスを許可していることを確認します。

3. このステートメントを (*### BEGIN ADDING STATEMENT HERE###* と *### END ADDING STATEMENT HERE###* の間に) AWS CodeBuild サービスロールにアタッチしたポリシーに追加します。このステートメントにより、 CodeBuild は Docker イメージを Amazon ECR リポジトリにアップロードできます。省略記号 (`...`) は、簡潔にするために使用され、ステートメントを追加する場所の特定に役立ちます。ステートメントを削除しないでください、また、これらの省略記号をポリシーに入力しないでください。

```
{
  "Statement": [
    ### BEGIN ADDING STATEMENT HERE ###
    {
      "Action": [
        "ecr:BatchCheckLayerAvailability",
        "ecr:CompleteLayerUpload",
        "ecr:GetAuthorizationToken",
```

```
        "ecr:InitiateLayerUpload",
        "ecr:PutImage",
        "ecr:UploadLayerPart"
      ],
      "Resource": "*",
      "Effect": "Allow"
    },
    ### END ADDING STATEMENT HERE ###
    ...
  ],
  "Version": "2012-10-17"
}
```

> **Note**
>
> このポリシーを変更する IAM エンティティは、ポリシーを変更するために IAM のアクセス許可を持っている必要があります。

4. このトピックの「ディレクトリ構造」セクションと「ファイル」セクションの説明に従ってファイルを作成し、S3 入力バケットにアップロードするか AWS CodeCommit、 GitHub、、または Bitbucket リポジトリにアップロードします。詳細については、「AWS CodePipeline ユーザーガイド」の「イメージ定義ファイルのリファレンス」を参照してください。

> **Important**
>
> *(root directory name)* をアップロードしないでください。アップロードするのは、*(root directory name)* 内のファイルのみです。
> S3 入力バケットを使用している場合は、ファイルを必ず ZIP ファイルに圧縮してから入力バケットにアップロードしてください。*(root directory name)* を ZIP ファイルに追加しないでください。追加するのは、*(root directory name)* 内のファイルのみです。

5. ビルドプロジェクトを作成し、ビルドを実行し、ビルド情報を表示します。

   コンソールを使用してプロジェクトを作成する場合:

   a. [Operating system] で、[Ubuntu] を選択します。

   b. [ランタイム] で、[Standard (標準)] を選択します。

   c. [イメージ] で、[aws/codebuild/standard:5.0] を選択します。

d. 次の環境変数を設定します。

- AWS_DEFAULT_REGION (値は *region-ID*)
- AWS_ACCOUNT_ID (値は *account-ID*)
- IMAGE_TAG (最新の値)
- IMAGE_REPO_NAME (値は *Amazon-ECR-repo-name*)

を使用してビルドプロジェクト AWS CLI を作成する場合、 `create-project` コマンドへの JSON 形式の入力は次のようになります。(プレースホルダは独自の値に置き換えてください。)

```
{
  "name": "sample-docker-project",
  "source": {
    "type": "S3",
    "location": "codebuild-region-ID-account-ID-input-bucket/DockerSample.zip"
  },
  "artifacts": {
    "type": "NO_ARTIFACTS"
  },
  "environment": {
    "type": "LINUX_CONTAINER",
    "image": "aws/codebuild/standard:5.0",
    "computeType": "BUILD_GENERAL1_SMALL",
    "environmentVariables": [
      {
        "name": "AWS_DEFAULT_REGION",
        "value": "region-ID"
      },
      {
        "name": "AWS_ACCOUNT_ID",
        "value": "account-ID"
      },
      {
        "name": "IMAGE_REPO_NAME",
        "value": "Amazon-ECR-repo-name"
      },
      {
        "name": "IMAGE_TAG",
        "value": "latest"
      }
    ],
```

```
  },
  "serviceRole": "arn:aws:iam::account-ID:role/role-name",
  "encryptionKey": "arn:aws:kms:region-ID:account-ID:key/key-ID"
}
```

6. が Docker イメージをリポジトリに CodeBuild 正常にプッシュしたことを確認します。

   1. Amazon ECR コンソール (https://console.aws.amazon.com/ecr/) を開きます。
   2. リポジトリ名を選択します。イメージは、[Image tag (イメージタグ)] 列に表示されています。

ディレクトリ構造

このサンプルのディレクトリ構造は次のとおりとします。

```
(root directory name)
### buildspec.yml
### Dockerfile
```

ファイル

このサンプルで使用するファイルは以下のとおりです。

buildspec.yml (内)*(root directory name)*

```
version: 0.2

phases:
  pre_build:
    commands:
      - echo Logging in to Amazon ECR...
      - aws ecr get-login-password --region $AWS_DEFAULT_REGION | docker login --
username AWS --password-stdin $AWS_ACCOUNT_ID.dkr.ecr.$AWS_DEFAULT_REGION.amazonaws.com
  build:
    commands:
      - echo Build started on `date`
      - echo Building the Docker image...
      - docker build -t $IMAGE_REPO_NAME:$IMAGE_TAG .
      - docker tag $IMAGE_REPO_NAME:$IMAGE_TAG $AWS_ACCOUNT_ID.dkr.ecr.
$AWS_DEFAULT_REGION.amazonaws.com/$IMAGE_REPO_NAME:$IMAGE_TAG
  post_build:
    commands:
```

```
      - echo Build completed on `date`
      - echo Pushing the Docker image...
      - docker push $AWS_ACCOUNT_ID.dkr.ecr.$AWS_DEFAULT_REGION.amazonaws.com/
$IMAGE_REPO_NAME:$IMAGE_TAG
```

Dockerfile (内)*(root directory name)*

```
FROM golang:1.12-alpine AS build
#Install git
RUN apk add --no-cache git
#Get the hello world package from a GitHub repository
RUN go get github.com/golang/example/hello
WORKDIR /go/src/github.com/golang/example/hello
# Build the project and send the output to /bin/HelloWorld
RUN go build -o /bin/HelloWorld

FROM golang:1.12-alpine
#Copy the build's output binary from the previous build container
COPY --from=build /bin/HelloWorld /bin/HelloWorld
ENTRYPOINT ["/bin/HelloWorld"]
```

> **Note**
>
> CodeBuild は、カスタム Docker イメージENTRYPOINTの を上書きします。

イメージを Docker Hub にプッシュするためのサンプルの調整

Docker イメージのプッシュ先を Amazon ECR ではなく Docker Hub にするには、このサンプルのコードを編集します。

> **Note**
>
> 使用している Docker のバージョンが 17.06 より前のものである場合は、--no-include-email オプションを削除します。

1. buildspec.yml ファイルで、以下の Amazon ECR 固有のコード行を置き換えます。

```
...
```

```
  pre_build:
    commands:
      - echo Logging in to Amazon ECR...
      - aws ecr get-login-password --region $AWS_DEFAULT_REGION |
 docker login --username AWS --password-stdin $AWS_ACCOUNT_ID.dkr.ecr.
$AWS_DEFAULT_REGION.amazonaws.com
  build:
    commands:
      - echo Build started on `date`
      - echo Building the Docker image...
      - docker build -t $IMAGE_REPO_NAME:$IMAGE_TAG .
      - docker tag $IMAGE_REPO_NAME:$IMAGE_TAG $AWS_ACCOUNT_ID.dkr.ecr.
$AWS_DEFAULT_REGION.amazonaws.com/$IMAGE_REPO_NAME:$IMAGE_TAG
  post_build:
    commands:
      - echo Build completed on `date`
      - echo Pushing the Docker image...
      - docker push $AWS_ACCOUNT_ID.dkr.ecr.$AWS_DEFAULT_REGION.amazonaws.com/
$IMAGE_REPO_NAME:$IMAGE_TAG
...
```

代わりに、以下の Docker Hub 固有のコード行を使用します。

```
...
  pre_build:
    commands:
      - echo Logging in to Docker Hub...
      # Type the command to log in to your Docker Hub account here.
  build:
    commands:
      - echo Build started on `date`
      - echo Building the Docker image...
      - docker build -t $IMAGE_REPO_NAME:$IMAGE_TAG .
      - docker tag $IMAGE_REPO_NAME:$IMAGE_TAG $IMAGE_REPO_NAME:$IMAGE_TAG
  post_build:
    commands:
      - echo Build completed on `date`
      - echo Pushing the Docker image...
      - docker push $IMAGE_REPO_NAME:$IMAGE_TAG
...
```

2. 編集したコードを S3 入力バケットにアップロードするか AWS CodeCommit、、 GitHub、または Bitbucket リポジトリにアップロードします。

> **Important**
>
> *(root directory name)* をアップロードしないでください。アップロードするのは、*(root directory name)* 内のファイルのみです。
> S3 入力バケットを使用している場合は、ファイルを必ず ZIP ファイルに圧縮してから入力バケットにアップロードしてください。*(root directory name)* を ZIP ファイルに追加しないでください。追加するのは、*(root directory name)* 内のファイルのみです。

3. `create-project` コマンドに対する JSON 形式の入力で、以下のコード行が置き換えの対象です。

```
...
    "environmentVariables": [
      {
        "name": "AWS_DEFAULT_REGION",
        "value": "region-ID"
      },
      {
        "name": "AWS_ACCOUNT_ID",
        "value": "account-ID"
      },
      {
        "name": "IMAGE_REPO_NAME",
        "value": "Amazon-ECR-repo-name"
      },
      {
        "name": "IMAGE_TAG",
        "value": "latest"
      }
    ]
...
```

以下のコード行に置き換えます。

```
...
    "environmentVariables": [
      {
        "name": "IMAGE_REPO_NAME",
        "value": "your-Docker-Hub-repo-name"
```

```
      },
      {
        "name": "IMAGE_TAG",
        "value": "latest"
      }
    ]
...
```

4. ビルド環境を作成し、ビルドを実行し、関連するビルド情報を表示します。
5. が Docker イメージをリポジトリに AWS CodeBuild 正常にプッシュしたことを確認します。Docker Hub にサインインし、リポジトリに進み、[Tags] タブを選択します。`latest` タグには、ごく最近の [Last Updated] (最終更新) の値が含まれています。

関連リソース

- の開始方法については AWS CodeBuild、「」を参照してくださいコンソールを使用した AWS CodeBuild の開始方法。
- での問題のトラブルシューティングについては CodeBuild、「」を参照してくださいトラブルシューティング AWS CodeBuild。
- のクォータの詳細については CodeBuild、「」を参照してくださいAWS CodeBuild のクォータ。

の AWS Secrets Manager サンプルを使用したプライベートレジストリ CodeBuild

このサンプルでは、プライベートレジストリに保存されている Docker イメージを AWS CodeBuild ランタイム環境として使用する方法を示します。プライベートレジストリの認証情報は AWS Secrets Managerに保存されています。プライベートレジストリは で動作します CodeBuild。このサンプルでは Docker Hub を使用します。

> **Note**
>
> シークレットはアクションに表示され、ファイルに書き込まれる際にマスクされません。

プライベートレジストリのサンプルの要件

でプライベートレジストリを使用するには AWS CodeBuild、以下が必要です。

- Docker Hub 認証情報を保存する Secrets Manager シークレット。この認証情報を使用してプライベートリポジトリにアクセスします。

> **Note**
>
> 作成したシークレットに対して料金が発生します。

- プライベートリポジトリまたはアカウント。
- Secrets Manager シークレットへのアクセスを許可する CodeBuild サービスロール IAM ポリシー。

これらのリソースを作成し、プライベートレジストリに保存されている Docker イメージを使用して CodeBuild ビルドプロジェクトを作成するには、次の手順に従います。

プライベートレジストリを使用して CodeBuild プロジェクトを作成する

1. 無料のプライベートリポジトリを作成する方法については、Docker Hub のリポジトリに関するページを参照してください。ターミナルで以下のコマンドを実行して、イメージのプル、ID の取得、新しいリポジトリへのプッシュを行うこともできます。

   ```
   docker pull amazonlinux
   docker images amazonlinux --format {{.ID}}
   docker tag image-id your-username/repository-name:tag
   docker login
   docker push your-username/repository-name
   ```

2. 「 ユーザーガイド」の「 AWS Secrets Manager シークレットを作成するAWS Secrets Manager 」の手順に従います。

   a. ステップ 3 の [シークレットのタイプを選択] で、[他の種類のシークレット] を選択します。

   b. [キー/値のペア] で、Docker Hub ユーザー名として 1 つのキーと値のペアを作成し、Docker Hub パスワードとして 1 つのキーと値のペアを作成します

   c. 「 AWS Secrets Manager シークレットを作成する」のステップを続行します。

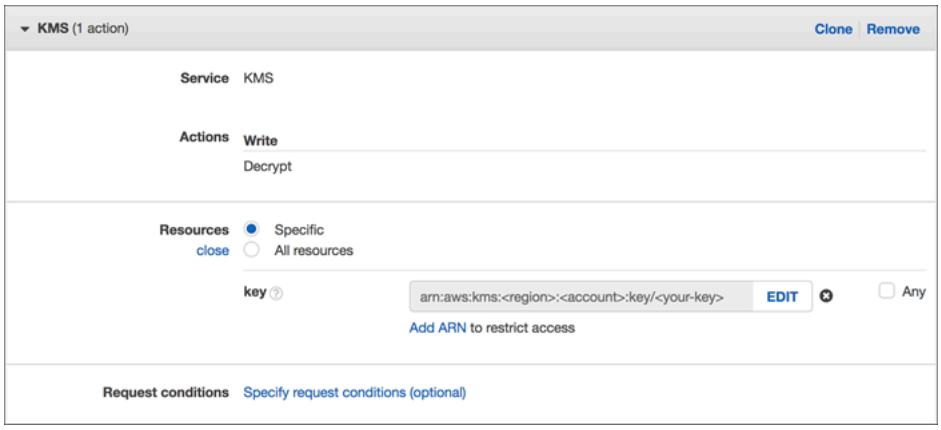
   d. ステップ 5 の [自動ローテーションの設定] ページで、自動ローテーションをオフにします。キーは Docker Hub の認証情報に対応しているためです。

   e. 「 AWS Secrets Manager シークレットを作成する」の手順に従います。

   詳細については、「What is AWS Secrets Manager?」を参照してください。

3. コンソールで AWS CodeBuild プロジェクトを作成すると、 は必要なアクセス許可をア CodeBuild タッチします。以外の AWS KMS キーを使用する場合はDefaultEncryptionKey、サービスロールに追加する必要があります。詳細については、『[g852]IAM ユーザーガイド[/g852]』の「[g851]ロールの修正 (コンソール)[/g851]」を参照してください。

   Secrets Manager で使用するサービスロールには、少なくとも secretsmanager:GetSecretValue アクセス許可が必要です。



4. コンソールでプライベートレジストリに保存されている環境を使用してプロジェクトを作成するには、プロジェクトの作成時に以下の操作を行います。詳細については、ビルドプロジェクトの作成 (コンソール) を参照してください。

   > **Note**
   >
   > プライベートレジストリが VPC 内にある場合は、パブリックインターネットアクセスが必要です。VPC CodeBuild 内のプライベート IP アドレスからイメージをプルすることはできません。

   a. [環境イメージ] で、[カスタムイメージ] を選択します。

   b. [Environment type (環境タイプ)] で、[Linux] または [Windows] を選択します。

   c. [イメージレジストリ] で、[その他のレジストリ] を選択します。

   d. [外部レジストリの URL] で、画像の場所を入力し、[レジストリの認証情報 - オプション] で Secrets Manager の認証情報の ARN または名前を入力します。

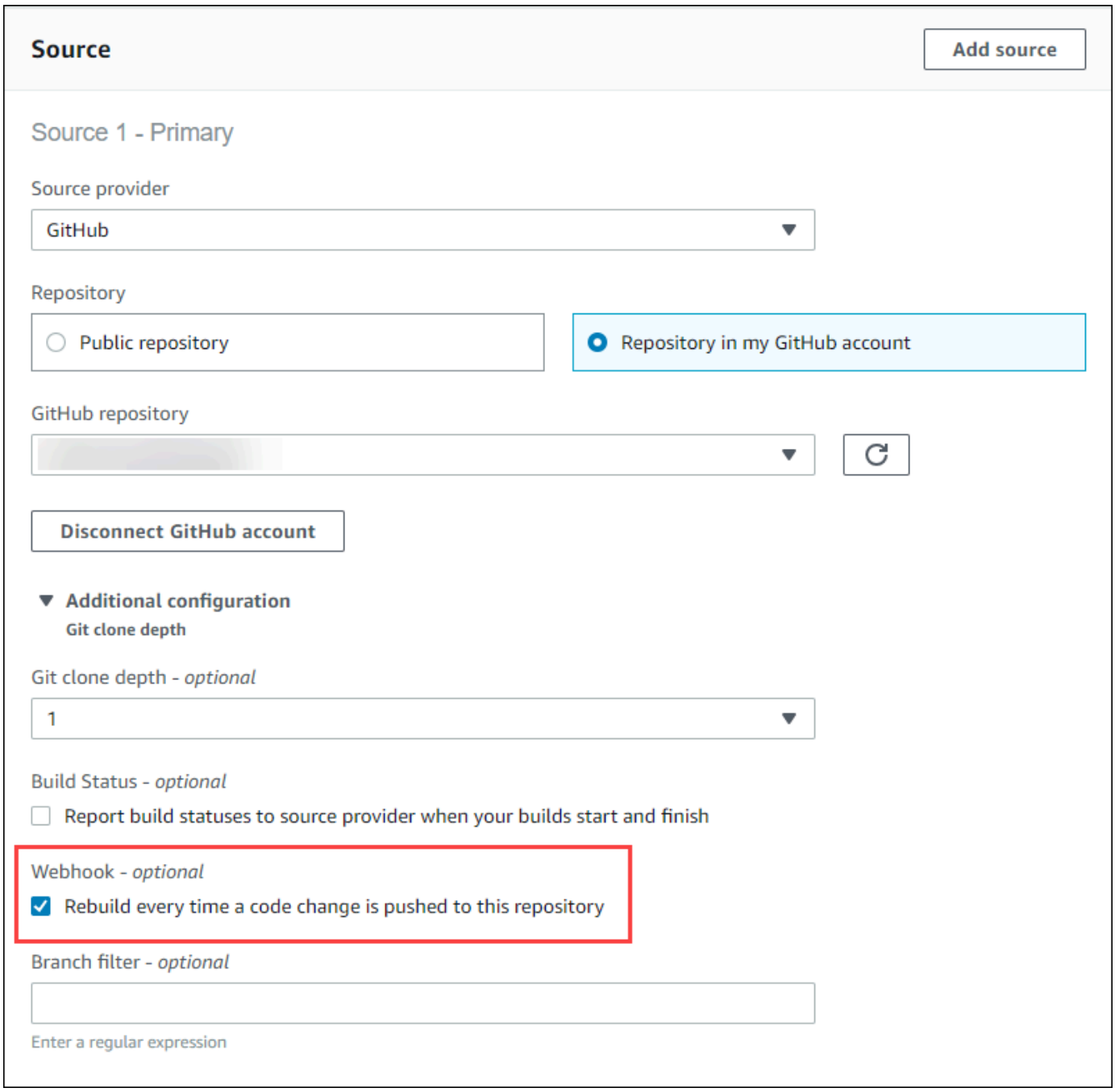
> **Note**
>
> 認証情報が現在のリージョンに存在しない場合は、ARN を使用する必要があります。認証情報が別のリージョンに存在する場合、認証情報の名前は使用できません。

ビルド出力を S3 バケットでホストする静的ウェブサイトの作成

ビルドのアーティファクトの暗号化を無効にすることができます。これにより、ウェブサイトをホストするように設定した場所にアーティファクトを発行できます。(暗号化されたアーティファクトを発行することはできません。) このサンプルは、ウェブフックを使用してビルドをトリガーし、ウェブサイトとして設定した S3 バケットにそのアーティファクトを発行する方法を示しています。

1. 「静的ウェブサイトのセットアップ」の手順に従って、S3 バケットをウェブサイトとして動作するように設定します。
2. https://console.aws.amazon.com/codesuite/codebuild/home で AWS CodeBuild コンソールを開きます。
3. CodeBuild 情報ページが表示されたら、ビルドプロジェクトの作成を選択します。それ以外の場合は、ナビゲーションペインでビルドを展開し、[ビルドプロジェクト] を選択し、次に [Create build project (ビルドプロジェクトの作成)] を選択します。
4. [プロジェクト名] に、このビルドプロジェクトの名前を入力します。ビルドプロジェクト名は、各 AWS アカウントで一意である必要があります。また、他のユーザーがこのプロジェクトの使用目的を理解できるように、ビルドプロジェクトの説明を任意で指定することもできます。
5. ソース で、ソースプロバイダー で、 を選択しますGitHub。「」の指示に従って と接続 (または再接続) し GitHub、「承認」を選択します。

   [ウェブフック] で、[Rebuild every time a code change is pushed to this repository (コードの変更がこのレポジトリにプッシュされるたびに再構築する)] を選択します。このチェックボックスは、[Use a repository in my account (自分のアカウントでリポジトリを使用する)] を選択した場合のみオンにできます。



6. [環境] で以下の操作を行います。

   [Environment image (環境イメージ)] で、次のいずれかの操作を行います。

   - によって管理される Docker イメージを使用するには AWS CodeBuild、マネージドイメージを選択し、オペレーティングシステム 、ランタイム (s)、イメージ 、イメージバージョン から選択します。利用可能な場合は、[環境タイプ] から選択します。

- 別の Docker イメージを使用するには、[カスタムイメージ] を選択します。[Environment type (環境タイプ)] で、 [ARM]、[Linux]、[Linux GPU] または [Windows] を選択します。[Other registry (その他のレジストリ)] を選択した場合は、[External registry URL (外部のレジストリ URL)] に *docker repository*/*docker image name* の形式に従って Docker Hub の Docker イメージの名前とタグを入力します。Amazon ECR を選択した場合は、Amazon ECR リポジトリと Amazon ECR イメージを使用して、 AWS アカウントの Docker イメージを選択します。
- プライベート Docker イメージを使用するには、[カスタムイメージ] を選択します。[Environment type (環境タイプ)] で、 [ARM]、[Linux]、[Linux GPU] または [Windows] を選択します。[Image registry (イメージレジストリ)] に [Other registry (その他のレジストリ)] を選択して、その後プライベート Docker イメージの認証情報の ARN を入力します。認証情報は、Secrets Manager で作成する必要があります。詳細については、AWS Secrets Manager ユーザーガイドの「AWS Secrets Managerとは」を参照してください。

7. [Service role (サービスロール)] で、次のいずれかの操作を行います。

   - CodeBuild サービスロールがない場合は、新しいサービスロール を選択します。[Role name] に、新しいロールの名前を入力します。
   - CodeBuild サービスロールがある場合は、既存のサービスロール を選択します。[Role ARN] で、サービスロールを選択します。

   > **Note**
   >
   > コンソールを使用してビルドプロジェクトを作成または更新する場合、同時に CodeBuild サービスロールを作成できます。デフォルトでは、ロールはそのビルドプロジェクトでのみ使用できます。コンソールでは、このサービスロールを別のビルドプロジェクトと関連付けると、この別のビルドプロジェクトで使用できるようにロールが更新されます。サービスロールは最大 10 個のビルドプロジェクトで使用できます。

8. [Buildspec] で、次のいずれかを行います。

   - [Use a buildspec file] (ビルド仕様ファイルの使用) を選択して、ソースコードのルートディレクトリの buildspec.yml を使用します。
   - [ビルドコマンドの挿入] を選択して、コンソールを使用してビルドコマンドを挿入します。

   詳細については、「ビルド仕様 (buildspec) に関するリファレンス」を参照してください。

9. [アーティファクト] で、[タイプ] として Amazon S3 を選択し、S3 バケットにビルド出力を保存します。
10. [バケット名] で、ステップ 1 でウェブサイトとして動作するよう設定した S3 バケットの名前を選択します。
11. [環境] で [ビルドコマンドの挿入] を選択した場合は、[出力ファイル] に、出力バケットに入れるビルドのファイルの場所を入力します。複数の場所がある場合は、カンマを使用して場所ごとに区切ります (例: **`appspec.yml, target/my-app.jar`**)。詳細については、「Artifacts reference-key in the buildspec file」を参照してください。
12. [Disable artifacts encryption (アーティファクトの暗号化を無効化)] を選択します。
13. [Additional configuration (追加設定)] オプションを展開し、必要に応じてオプションを選択します。
14. [Create build project (ビルドプロジェクトの作成)] を選択します。ビルドプロジェクトページの [ビルド履歴] で、[ビルドの開始] を選択してビルドを実行します。
15. (オプション) 「Amazon S3 デベロッパーガイド」の「例: Amazon でウェブサイトを高速化する CloudFront」の手順に従います。 Amazon S3

複数の入力ソースと出力アーティファクトのサンプル

複数の入力ソースと複数の出力アーティファクトのセットを含む AWS CodeBuild ビルドプロジェクトを作成できます。このサンプルは、ビルドプロジェクトをセットアップする方法を示しています。

- さまざまなタイプの複数のソースとリポジトリを使用します。
- ビルドアーティファクトを複数の S3 バケットに 1 つのビルドで発行します。

このサンプルでは、ビルドプロジェクトを作成し、それを使用してビルドを実行します。このサンプルでは、ビルドプロジェクトの buildspec ファイルを使用して、複数のソースを組み込み、複数のアーティファクトセットを作成する方法を示します。

1. ソースを 1 つ以上の S3 バケット、 CodeCommit GitHub、 GitHub Enterprise Server、または Bitbucket リポジトリにアップロードします。
2. プライマリソースを選択します。これは、 が buildspec ファイル CodeBuild を検索して実行するソースです。
3. ビルドプロジェクトを作成します。詳細については、「 でのビルドプロジェクトの作成AWS CodeBuild」を参照してください。

4. ビルドプロジェクトを作成し、ビルドを実行して、ビルドに関する情報を取得します。
5. を使用してビルドプロジェクト AWS CLI を作成する場合、 `create-project` コマンドへの JSON 形式の入力は次のようになります。

```
{
  "name": "sample-project",
  "source": {
    "type": "S3",
    "location": "<bucket/sample.zip>"
  },
  "secondarySources": [
    {
      "type": "CODECOMMIT",
      "location": "https://git-codecommit.us-west-2.amazonaws.com/v1/repos/repo",
      "sourceIdentifier": "source1"
    },
    {
      "type": "GITHUB",
      "location": "https://github.com/awslabs/aws-codebuild-jenkins-plugin",
      "sourceIdentifier": "source2"
    }
  ],
  "secondaryArtifacts": [ss
    {
      "type": "S3",
      "location": "<output-bucket>",
      "artifactIdentifier": "artifact1"
    },
    {
      "type": "S3",
      "location": "<other-output-bucket>",
      "artifactIdentifier": "artifact2"
    }
  ],
  "environment": {
    "type": "LINUX_CONTAINER",
    "image": "aws/codebuild/standard:5.0",
    "computeType": "BUILD_GENERAL1_SMALL"
  },
  "serviceRole": "arn:aws:iam::account-ID:role/role-name",
  "encryptionKey": "arn:aws:kms:region-ID:account-ID:key/key-ID"
}
```

プライマリソースは、source 属性で定義されます。他のすべてのソースはセカンダリソースと呼ばれ、secondarySources の下に表示されます。すべてのセカンダリソースは、独自のディレクトリにインストールされます。このディレクトリは、組み込みの環境変数 CODEBUILD_SRC_DIR_*sourceIdentifer* に保存されます。詳細については、「ビルド環境の環境変数」を参照してください。

secondaryArtifacts 属性には、アーティファクトの定義のリストが含まれます。これらのアーティファクトは、secondary-artifacts ブロック内にネストされている buildspec ファイルの artifacts ブロックを使用します。

buildspec ファイル内のセカンダリアーティファクトは、アーティファクトと同じ構造を持ち、アーティファクト識別子で区切られます。

> **Note**
>
> CodeBuild API では、セカンダリアーティファクトartifactIdentifierの は、CreateProjectおよび の必須属性ですUpdateProject。セカンダリアーティファクトを参照するために使用される必要があります。

前述の JSON 形式の入力を使用すると、プロジェクトの buildspec ファイルは次のようになります。

```
version: 0.2

phases:
  install:
    runtime-versions:
      java: openjdk11
  build:
    commands:
      - cd $CODEBUILD_SRC_DIR_source1
      - touch file1
      - cd $CODEBUILD_SRC_DIR_source2
      - touch file2

artifacts:
  files:
    - '**.*'
  secondary-artifacts:
    artifact1:
      base-directory: $CODEBUILD_SRC_DIR_source1
```

```
      files:
        - file1
    artifact2:
      base-directory: $CODEBUILD_SRC_DIR_source2
      files:
        - file2
```

sourceVersion の StartBuild 属性で API を使用して、プライマリソースのバージョンを上書きすることができます。1 つまたは複数のセカンダリソースバージョンを上書きするには、secondarySourceVersionOverride 属性を使用します。

の start-build コマンドへの JSON 形式の入力 AWS CLI は次のようになります。

```
{
   "projectName": "sample-project",
   "secondarySourcesVersionOverride": [
      {
        "sourceIdentifier": "source1",
        "sourceVersion": "codecommit-branch"
      },
      {
        "sourceIdentifier": "source2",
        "sourceVersion": "github-branch"
      },
   ]
}
```

ソースサンプルがないプロジェクト

NO_SOURCE ソースを設定するときに、ソースタイプを選択して CodeBuild プロジェクトを設定できます。ソースタイプが **NO_SOURCE** である場合、プロジェクトにはソースがないため、buildspec ファイルを指定することはできません。代わりに、CLI の buildspec コマンドに対する JSON 形式の入力の create-project 属性で、YAML 形式の buildspec 文字列を指定する必要があります。次のように指定します。

```
{
  "name": "project-name",
  "source": {
    "type": "NO_SOURCE",
    "buildspec": "version: 0.2\n\nphases:\n  build:\n    commands:\n      - command"
   },
  "environment": {
```

```
    "type": "LINUX_CONTAINER",
    "image": "aws/codebuild/standard:5.0",
    "computeType": "BUILD_GENERAL1_SMALL",
  },
  "serviceRole": "arn:aws:iam::account-ID:role/role-name",
  "encryptionKey": "arn:aws:kms:region-ID:account-ID:key/key-ID"
}
```

詳細については、「ビルドプロジェクトの作成 (AWS CLI)」を参照してください。

複数のソース入力を使用して複数の出力アーティファクトを作成するパイプラインを作成する方法については、 CodeBuild 「」を参照してくださいAWS CodePipeline CodeBuild と および複数の入力ソースおよび出力アーティファクトとの統合サンプル 。

の buildspec ファイルサンプルのランタイムバージョン CodeBuild

Amazon Linux 2 (AL2) 標準イメージバージョン 1.0 以降、または Ubuntu 標準イメージバージョン 2.0 以降を使用している場合は、buildspec ファイルの `runtime-versions` セクションで 1 つ以上のランタイムを指定できます。このサンプルでは、プロジェクトランタイムを変更する方法、複数のランタイムを指定する方法、および別のランタイムに依存するランタイムを指定する方法を示します。サポートされているランタイムについては、「が提供する Docker イメージ CodeBuild」を参照してください。

> **Note**
>
> ビルドコンテナで Docker を使用している場合、ビルドは特権モードで実行する必要があります。詳細については、「AWS CodeBuild でのビルドの実行」および「 でのビルドプロジェクトの作成AWS CodeBuild」を参照してください。

ランタイムバージョンの更新

プロジェクトで使用されるランタイムを新しいバージョンに変更するには、buildpec ファイルの `runtime-versions` セクションを更新します。以下の例では、Java バージョン 8 および 11 を指定する方法を示します。

- Java バージョン 8 を指定する `runtime-versions` セクション:

```
phases:
  install:
```

```
    runtime-versions:
      java: corretto8
```

- Java バージョン 11 を指定する `runtime-versions` セクション:

```
phases:
  install:
    runtime-versions:
      java: corretto11
```

次の例では、Ubuntu 標準イメージ 5.0 または Amazon Linux 2 標準イメージ 3.0 を使用して、Python の異なるバージョンを指定する方法を示しています。

- Python バージョン 3.7 を指定する `runtime-versions` セクション:

```
phases:
  install:
    runtime-versions:
      python: 3.7
```

- Python バージョン 3.8 を指定する `runtime-versions` セクション:

```
phases:
  install:
    runtime-versions:
      python: 3.8
```

このサンプルでは、Java バージョン 8 ランタイムで始まり、その後で Java バージョン 10 ランタイムに更新されるプロジェクトを示します。

1. Maven をダウンロードし、インストールします。詳細については、Apache Maven ウェブサイトの「Apache Maven のダウンロード」および「Apache Maven のインストール」を参照してください。
2. ローカルコンピュータまたはインスタンスの空のディレクトリに切り替えて、この Maven コマンドを実行します。

```
mvn archetype:generate "-DgroupId=com.mycompany.app" "-DartifactId=ROOT" "-
DarchetypeArtifactId=maven-archetype-webapp" "-DinteractiveMode=false"
```

成功すると、このディレクトリ構造とファイルが作成されます。

```
.
### ROOT
    ### pom.xml
    ### src
        ### main
            ### resources
            ### webapp
                ### WEB-INF
                #   ### web.xml
                ### index.jsp
```

3. 次の内容で、buildspec.yml というファイルを作成します。ファイルを *(root directory name)*/my-web-app ディレクトリ内に保存します。

```
version: 0.2

phases:
  install:
    runtime-versions:
      java: corretto8
  build:
    commands:
      - java -version
      - mvn package
artifacts:
  files:
    - '**/*'
  base-directory: 'target/my-web-app'
```

buildspec ファイル:

- この runtime-versions セクションでは、プロジェクトでバージョン 8 のJava ランタイムを使用することを指定します。
- この - java -version コマンドは、ビルド時にプロジェクトで使用されている Java のバージョンを表示します。

ファイル構造は次のようになります。

```
(root directory name)
### my-web-app
    ### src
    #   ### main
    #   ### resources
    #   ### webapp
    #       ### WEB-INF
    #           ### web.xml
    #               ### index.jsp
    ### buildspec.yml
    ### pom.xml
```

4. my-web-app ディレクトリの内容を S3 入力バケットにアップロードするか CodeCommit、、GitHub、または Bitbucket リポジトリにアップロードします。

> **Important**
>
> *(root directory name)* または *(root directory name)*/my-web-app をアップロードしないでください。アップロードするのは、*(root directory name)*/my-web-app のディレクトリとファイルだけです。
> S3 入力バケットを使用している場合は、ディレクトリ構造とファイルを必ず ZIP ファイルに圧縮してから入力バケットにアップロードしてください。*(root directory name)* または *(root directory name)*/my-web-app を ZIP ファイルに追加しないでください。追加するのは、*(root directory name)*/my-web-app のディレクトリとファイルだけです。

5. https://console.aws.amazon.com/codesuite/codebuild/home で AWS CodeBuild コンソールを開きます。
6. ビルドプロジェクトを作成します。詳細については、「ビルドプロジェクトの作成 (コンソール)」および「ビルドの実行 (コンソール)」を参照してください。これらの設定を除いて、すべての設定をデフォルト値のままにします。

   - [環境] の場合:
     - [環境イメージ] で、[Managed image (マネージド型イメージ)] を選択します。
     - [オペレーティングシステム] で、[Amazon Linux 2] を選択します。
     - [ランタイム] で、[Standard (標準)] を選択します。
     - [イメージ] で、[aws/codebuild/amazonlinux2-x86_64-standard:4.0] を選択します。

7. [Start build] を選択します。
8. [ビルド設定] でデフォルト値をそのまま使用して、[ビルドの開始] を選択します。
9. ビルドが完了したら、[ビルドログ] タブでビルド出力を表示します。次のような出力が表示されます。

```
[Container] Date Time Phase is DOWNLOAD_SOURCE
[Container] Date Time CODEBUILD_SRC_DIR=/codebuild/output/src460614277/src
[Container] Date Time YAML location is /codebuild/output/src460614277/src/
buildspec.yml
[Container] Date Time Processing environment variables
[Container] Date Time Selecting 'java' runtime version 'corretto8' based on manual
 selections...
[Container] Date Time Running command echo "Installing Java version 8 ..."
Installing Java version 8 ...

[Container] Date Time Running command export JAVA_HOME="$JAVA_8_HOME"

[Container] Date Time Running command export JRE_HOME="$JRE_8_HOME"

[Container] Date Time Running command export JDK_HOME="$JDK_8_HOME"

[Container] Date Time Running command for tool_path in "$JAVA_8_HOME"/bin/*
 "$JRE_8_HOME"/bin/*;
```

10. `runtime-versions` セクションを Java バージョン 11 で更新します。

```
install:
    runtime-versions:
      java: corretto11
```

11. 変更を保存したら、ビルドを再実行し、ビルド出力を表示します。現在インストールされている Java のバージョンが 11 であることが表示されます。次のような出力が表示されます:

```
[Container] Date Time Phase is DOWNLOAD_SOURCE
[Container] Date Time CODEBUILD_SRC_DIR=/codebuild/output/src460614277/src
[Container] Date Time YAML location is /codebuild/output/src460614277/src/
buildspec.yml
[Container] Date Time Processing environment variables
[Container] Date Time Selecting 'java' runtime version 'corretto11' based on manual
 selections...
Installing Java version 11 ...
```

```
[Container] Date Time Running command export JAVA_HOME="$JAVA_11_HOME"

[Container] Date Time Running command export JRE_HOME="$JRE_11_HOME"

[Container] Date Time Running command export JDK_HOME="$JDK_11_HOME"

[Container] Date Time Running command for tool_path in "$JAVA_11_HOME"/bin/*
 "$JRE_11_HOME"/bin/*;
```

2 つのランタイムの指定

同じ CodeBuild ビルドプロジェクトで複数のランタイムを指定できます。このサンプルプロジェクトでは、2 つのソースファイルを使用します。1 つは Go ランタイムを使用し、もう 1 つは Node.js ランタイムを使用します。

1. my-source という名前のディレクトリを作成します。
2. my-source ディレクトリ内に golang-app という名前のディレクトリを作成します。
3. 次の内容で、hello.go というファイルを作成します。ファイルを golang-app ディレクトリ内に保存します。

```
package main
import "fmt"

func main() {
  fmt.Println("hello world from golang")
  fmt.Println("1+1 =", 1+1)
  fmt.Println("7.0/3.0 =", 7.0/3.0)
  fmt.Println(true && false)
  fmt.Println(true || false)
  fmt.Println(!true)
  fmt.Println("good bye from golang")
}
```

4. my-source ディレクトリ内に nodejs-app という名前のディレクトリを作成します。これは golang-app ディレクトリと同じレベルにある必要があります。
5. 次の内容で、index.js というファイルを作成します。ファイルを nodejs-app ディレクトリ内に保存します。

```
console.log("hello world from nodejs");
console.log("1+1 =" + (1+1));
```

```
console.log("7.0/3.0 =" + 7.0/3.0);
console.log(true && false);
console.log(true || false);
console.log(!true);
console.log("good bye from nodejs");
```

6. 次の内容で、package.json というファイルを作成します。ファイルを nodejs-app ディレクトリ内に保存します。

```
{
  "name": "mycompany-app",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"run some tests here\""
  },
  "author": "",
  "license": "ISC"
}
```

7. 次の内容で、buildspec.yml というファイルを作成します。my-source および nodejs-app ディレクトリと同じレベルで、ファイルを golang-app ディレクトリに保存します。runtime-versions セクションでは、Node.js バージョン 12 および Go バージョン 1.13 ランタイムを指定します。

```
version: 0.2

phases:
  install:
    runtime-versions:
      golang: 1.13
      nodejs: 12
  build:
    commands:
      - echo Building the Go code...
      - cd $CODEBUILD_SRC_DIR/golang-app
      - go build hello.go
      - echo Building the Node code...
      - cd $CODEBUILD_SRC_DIR/nodejs-app
      - npm run test
artifacts:
  secondary-artifacts:
```

```
    golang_artifacts:
      base-directory: golang-app
      files:
        - hello
    nodejs_artifacts:
      base-directory: nodejs-app
      files:
        - index.js
        - package.json
```

8. ファイル構造は次のようになります。

```
my-source
### golang-app
#   ### hello.go
### nodejs.app
#   ### index.js
#   ### package.json
### buildspec.yml
```

9. `my-source` ディレクトリの内容を S3 入力バケットにアップロードするか CodeCommit、、GitHub、または Bitbucket リポジトリにアップロードします。

> **Important**
>
> S3 入力バケットを使用している場合は、ディレクトリ構造とファイルを必ず ZIP ファイルに圧縮してから入力バケットにアップロードしてください。`my-source` を ZIP ファイルに追加しないでください。追加するのは、`my-source` のディレクトリとファイルのみです。

10. https://console.aws.amazon.com/codesuite/codebuild/home で AWS CodeBuild コンソールを開きます。
11. ビルドプロジェクトを作成します。詳細については、「ビルドプロジェクトの作成 (コンソール)」および「ビルドの実行 (コンソール)」を参照してください。これらの設定を除いて、すべての設定をデフォルト値のままにします。

    - [環境] の場合:
      - [環境イメージ] で、[Managed image (マネージド型イメージ)] を選択します。
      - [オペレーティングシステム] で、[Amazon Linux 2] を選択します。
      - [ランタイム] で、[Standard (標準)] を選択します。

- [イメージ] で、[aws/codebuild/amazonlinux2-x86_64-standard:4.0] を選択します。

12. [Create build project (ビルドプロジェクトの作成)] を選択します。
13. [Start build] を選択します。
14. [ビルド設定] でデフォルト値をそのまま使用して、[ビルドの開始] を選択します。
15. ビルドが完了したら、[ビルドログ] タブでビルド出力を表示します。次のような出力が表示されます。Go ランタイムおよび Node.js ランタイムからの出力が表示されます。また、Go アプリケーションおよび Node.js アプリケーションからの出力も表示されます。

```
[Container] Date Time Processing environment variables
[Container] Date Time Selecting 'golang' runtime version '1.13' based on manual
 selections...
[Container] Date Time Selecting 'nodejs' runtime version '12' based on manual
 selections...
[Container] Date Time Running command echo "Installing Go version 1.13 ..."
Installing Go version 1.13 ...

[Container] Date Time Running command echo "Installing Node.js version 12 ..."
Installing Node.js version 12 ...

[Container] Date Time Running command n $NODE_12_VERSION
   installed : v12.20.1 (with npm 6.14.10)

[Container] Date Time Moving to directory /codebuild/output/src819694850/src
[Container] Date Time Registering with agent
[Container] Date Time Phases found in YAML: 2
[Container] Date Time  INSTALL: 0 commands
[Container] Date Time  BUILD: 1 commands
[Container] Date Time Phase complete: DOWNLOAD_SOURCE State: SUCCEEDED
[Container] Date Time Phase context status code:  Message:
[Container] Date Time Entering phase INSTALL
[Container] Date Time Phase complete: INSTALL State: SUCCEEDED
[Container] Date Time Phase context status code:  Message:
[Container] Date Time Entering phase PRE_BUILD
[Container] Date Time Phase complete: PRE_BUILD State: SUCCEEDED
[Container] Date Time Phase context status code:  Message:
[Container] Date Time Entering phase BUILD
[Container] Date Time Running command echo Building the Go code...
Building the Go code...

[Container] Date Time Running command cd $CODEBUILD_SRC_DIR/golang-app
```

```
[Container] Date Time Running command go build hello.go

[Container] Date Time Running command echo Building the Node code...
Building the Node code...

[Container] Date Time Running command cd $CODEBUILD_SRC_DIR/nodejs-app

[Container] Date Time Running command npm run test

> mycompany-app@1.0.0 test /codebuild/output/src924084119/src/nodejs-app
> echo "run some tests here"

run some tests here
```

のソースバージョンサンプル AWS CodeBuild

このサンプルでは、コミット ID (コミット SHA とも呼ばれます) 以外の形式を使用してソースのバージョンを指定する方法を示します。ソースのバージョンは、以下の方法で指定できます。

- Amazon S3 ソースプロバイダーの場合は、ビルド入力 ZIP ファイルを表すオブジェクトのバージョン ID を使用します。
- CodeCommit、Bitbucket GitHub、および GitHub Enterprise Server の場合は、次のいずれかを使用します。
  - プルリクエストの参照としてのプルリクエスト (例: `refs/pull/1/head`)。
  - ブランチ名としてのブランチ。
  - コミット ID。
  - タグ。
  - 参照とコミット ID。参照は、次のいずれかになります。
    - タグ (例 : `refs/tags/mytagv1.0^{full-commit-SHA}`)。
    - ブランチ (例 : `refs/heads/mydevbranch^{full-commit-SHA}`)。
    - プルリクエスト (例 : `refs/pull/1/head^{full-commit-SHA}`)。
- GitLab および GitLab セルフマネージド の場合は、次のいずれかを使用します。
  - ブランチ名としてのブランチ。
  - コミット ID。
  - タグ。

> **Note**
>
> プルリクエストソースのバージョンは、リポジトリが GitHub または GitHub Enterprise Server の場合にのみ指定できます。

参照とコミット ID を使用してバージョンを指定すると、バージョンのみを指定した場合よりもビルドの DOWNLOAD_SOURCE フェーズが高速になります。これは、リファレンスを追加するときに、コミットを見つけるためにリポジトリ全体をダウンロードする CodeBuild 必要がないためです。

- ソースバージョンはコミット ID のみで指定できます (例: 1234567890123456789012346789012345678 9)。これを行う場合は、リポジトリ全体をダウンロードしてバージョンを見つける CodeBuild 必要があります。
- ソースバージョンを指定するには、参照とコミット ID を次の形式で使用できます: *refs/heads/branchname*^{*full-commit-SHA*} (例 : refs/heads/main^{1234567890123456789012346789012345678 9})。これを行うと、 は指定されたブランチのみ CodeBuild をダウンロードしてバージョンを検索します。

> **Note**
>
> ビルドの DOWNLOAD_SOURCEフェーズを高速化するために、Git クローンの深度を低い数値に設定することもできます。 CodeBuild ダウンロードするリポジトリのバージョンは少なくなります。

コミット ID で GitHub リポジトリバージョンを指定するには

1. https://console.aws.amazon.com/codesuite/codebuild/home で AWS CodeBuild コンソールを開きます。
2. ビルドプロジェクトを作成します。詳細については、「ビルドプロジェクトの作成 (コンソール)」および「ビルドの実行 (コンソール)」を参照してください。以下の設定を除いて、すべての設定をデフォルト値のままにします。

   - [Source (ソース)] で、次のようにします。
     - ソースプロバイダー で、 を選択しますGitHub。に接続していない場合は GitHub、手順に従って接続します。
     - [レポジトリ] で、[パブリックレポジトリ] を選択します。

- [リポジトリの URL] に、「**https://github.com/aws/aws-sdk-ruby.git**」と入力します。
- [環境] で以下の操作を行います。
  - [環境イメージ] で、[Managed image (マネージド型イメージ)] を選択します。
  - [オペレーティングシステム] で、[Amazon Linux 2] を選択します。
  - [ランタイム] で、[Standard (標準)] を選択します。
  - [イメージ] で、[aws/codebuild/amazonlinux2-x86_64-standard:4.0] を選択します。

3. [ビルド仕様] で、[ビルドコマンドの挿入] を選択して [Switch to editor (エディタに切り替え)] を選択します。
4. [ビルドコマンド] で、プレースホルダーテキストを次のように置き換えます。

```
version: 0.2

phases:
  install:
    runtime-versions:
      ruby: 2.6
  build:
    commands:
       - echo $CODEBUILD_RESOLVED_SOURCE_VERSION
```

Ubuntu 標準イメージ 2.0 を使用する場合、`runtime-versions` セクションは必須です。ここでは Ruby バージョン 2.6 ランタイムを指定していますが、任意のランタイムを使用できます。`echo` コマンドは、`CODEBUILD_RESOLVED_SOURCE_VERSION` 環境変数に保存されているソースコードのバージョンを表示します。

5. [ビルド設定] でデフォルト値をそのまま使用して、[ビルドの開始] を選択します。
6. [ソースバージョン] に「**046e8b67481d53bdc86c3f6affdd5d1afae6d369**」と入力します。これは、`https://github.com/aws/aws-sdk-ruby.git` リポジトリのコミットの SHA です。
7. [Start build] を選択します。
8. ビルドが完了すると、以下が表示されます。
   - [ビルドログ] タブに、使用されたプロジェクトソースのバージョン。以下はその例です。

```
[Container] Date Time Running command echo $CODEBUILD_RESOLVED_SOURCE_VERSION
046e8b67481d53bdc86c3f6affdd5d1afae6d369
```

```
[Container] Date Time Phase complete: BUILD State: SUCCEEDED
```

- [環境変数] タブに、ビルドを作成するために使用されたコミット ID と一致する [Resolved source version (解決されたソースバージョン)]。
- [フェーズ詳細] タブに、 DOWNLOAD_SOURCE フェーズの所要時間。

以下のステップでは、ソースの同じバージョンを使用してビルドを作成する方法を示します。今回は、ソースのバージョンを指定するのに参照とコミット ID を使用します。

コミット ID とリファレンスを使用して GitHub リポジトリバージョンを指定するには

1. 左のナビゲーションペインで、[ビルドプロジェクト] を選択し、先ほど作成したプロジェクトを選択します。
2. [Start build] を選択します。
3. [ソースバージョン] に「**refs/heads/main^{046e8b67481d53bdc86c3f6affdd5d1afae6d369}**」と入力します。これは、次の形式のブランチへのコミット ID および参照と同じです: *refs*/*heads*/*branchname*^{*full-commit-SHA*}。
4. [Start build] を選択します。
5. ビルドが完了すると、以下が表示されます。

   - [ビルドログ] タブに、使用されたプロジェクトソースのバージョン。以下はその例です。

     ```
     [Container] Date Time Running command echo $CODEBUILD_RESOLVED_SOURCE_VERSION
     046e8b67481d53bdc86c3f6affdd5d1afae6d369

     [Container] Date Time Phase complete: BUILD State: SUCCEEDED
     ```

   - [環境変数] タブに、ビルドを作成するために使用されたコミット ID と一致する [Resolved source version (解決されたソースバージョン)]。
   - [フェーズ詳細] タブに、DOWNLOAD_SOURCE フェーズの所要時間。これは、コミット ID のみを使用してソースのバージョンを指定する場合よりも短い時間であることが必要です。

のサードパーティーのソースリポジトリのサンプル CodeBuild

トピック

- の Bitbucket プルリクエストとウェブフックフィルターのサンプル CodeBuild
- GitHub の Enterprise Server サンプル CodeBuild
- GitHub のプルリクエストとウェブフックフィルターのサンプル CodeBuild

の Bitbucket プルリクエストとウェブフックフィルターのサンプル CodeBuild

AWS CodeBuild は、ソースリポジトリが Bitbucket の場合にウェブフックをサポートします。つまり、ソースコードが Bitbucket リポジトリに保存されている CodeBuild ビルドプロジェクトの場合、ウェブフックを使用して、コード変更がリポジトリにプッシュされるたびにソースコードを再構築できます。詳細については、「Bitbucket ウェブフックイベント」を参照してください。

このサンプルでは、Bitbucket リポジトリを使用してプルリクエストを作成する方法について説明します。また、Bitbucket ウェブフックを使用して をトリガー CodeBuild し、プロジェクトのビルドを作成する方法についても説明します。

> **Note**
>
> Webhook を使用する場合、ユーザーが予期しないビルドをトリガーする可能性があります。このリスクを軽減するには、「ウェブフック使用のベストプラクティス。」を参照してください。

トピック

- 前提条件
- Bitbucket をソースリポジトリとするビルドプロジェクトを作成し、ウェブフックを有効にする
- Bitbucket ウェブフックを使用してビルドをトリガーする

前提条件

このサンプルを実行するには、 AWS CodeBuild プロジェクトを Bitbucket アカウントに接続する必要があります。

> **Note**
>
> CodeBuild は Bitbucket でアクセス許可を更新しました。以前にプロジェクトを Bitbucket に接続し、Bitbucket 接続エラーを受け取った場合は、再接続してウェブフックを管理するアクセス CodeBuild 許可を付与する必要があります。

Bitbucket をソースリポジトリとするビルドプロジェクトを作成し、ウェブフックを有効にする

次のステップでは、Bitbucket をソースリポジトリとして AWS CodeBuild プロジェクトを作成し、ウェブフックを有効にする方法について説明します。

1. https://console.aws.amazon.com/codesuite/codebuild/home で AWS CodeBuild コンソールを開きます。
2. CodeBuild 情報ページが表示されたら、ビルドプロジェクトの作成を選択します。それ以外の場合は、ナビゲーションペインでビルドを展開し、[ビルドプロジェクト] を選択し、次に [Create build project (ビルドプロジェクトの作成)] を選択します。
3. [Create build project (ビルドプロジェクトの作成)] を選択します。
4. [Project configuration (プロジェクトの設定)] で、次のようにします。

   [Project name] (プロジェクト名)

   このビルドプロジェクトの名前を入力します。ビルドプロジェクト名は、各 AWS アカウントで一意である必要があります。また、他のユーザーがこのプロジェクトの使用目的を理解できるように、ビルドプロジェクトの説明を任意で指定することもできます。
5. [Source (ソース)] で、次のようにします。

   ソースプロバイダー

   [Bitbucket] を選択します。手順に従って Bitbucket に接続 (または再接続) し、[Authorize] (承認) を選択します。

   リポジトリ

   [Bitbucket アカウントのリポジトリ] を選択します。

   まだ Bitbucket アカウントに接続していない場合は、Bitbucket のユーザーネームとパスワードを入力し、[Bitbucket 認証情報の保存] を選択します。

Bitbucket リポジトリ

Bitbucket リポジトリの URL を入力します。

6. [プライマリソース Webhook イベント] で、以下を選択します。

> **Note**
>
> [プライマリソース Webhook イベント] セクションは、前のステップで [Bitbucket アカウントのリポジトリ] を選択した場合のみに表示されます。

1. プロジェクトの作成時に [コードの変更がこのレポジトリにプッシュされるたびに再構築する] を選択します。
2. [イベントタイプ] から、1 つ以上のイベントを選択します。
3. イベントでビルドをトリガーされた時間をフィルタリングするには、[これらの条件でビルドを開始する] で、1 つ以上のオプションフィルタを追加します。
4. イベントがトリガーされていない時間をフィルタリングするには、[これらの条件でビルドを開始しない] で、1 つ以上のオプションフィルタを追加します。
5. 別のフィルタグループを追加する必要がある場合、[フィルタグループの追加] を選択します。

Bitbucket ウェブフックイベントタイプとフィルターの詳細については、「Bitbucket ウェブフックイベント」を参照してください。

7. [環境] で以下の操作を行います。

環境イメージ

以下のうちのひとつを選択します。

によって管理される Docker イメージを使用するには AWS CodeBuild :

[Managed image (マネージドイメージ)] を選択し、次に [オペレーティングシステム]、[ランタイム]、[イメージ]、および [ランタイムバージョン] で適切な選択を行います。利用可能な場合は、[環境タイプ] から選択します。

別の Docker イメージを使用するには:

[カスタムイメージ] を選択します。[Environment type (環境タイプ)] で、 [ARM]、[Linux]、[Linux GPU] または [Windows] を選択します。[Other registry (その他のレジストリ)] を選択した場合は、[External registry URL (外部のレジストリ URL)] に *docker repository*/*docker image name* の形式に従って Docker Hub の Docker イメージの名前とタグを入力します。[Amazon ECR] を選択した場合は、[Amazon ECR repository] (Amazon ECR レポジトリ) および [Amazon ECR image] (Amazon ECR イメージ) を使用して AWS アカウントの Docker イメージを選択します。

プライベートDockerイメージを使用するには :

[カスタムイメージ] を選択します。[Environment type (環境タイプ)] で、 [ARM]、[Linux]、[Linux GPU] または [Windows] を選択します。[Image registry (イメージレジストリ)] に [Other registry (その他のレジストリ)] を選択して、その後プライベート Docker イメージの認証情報の ARN を入力します。認証情報は、Secrets Manager で作成する必要があります。詳細については、「 AWS Secrets Manager ユーザーガイド」の「 AWS Secrets Managerとは」を参照してください。

サービスロール

以下のうちのひとつを選択します。

- CodeBuild サービスロールがない場合は、新しいサービスロール を選択します。[Role name] に、新しいロールの名前を入力します。
- CodeBuild サービスロールがある場合は、既存のサービスロール を選択します。[Role ARN] で、サービスロールを選択します。

> **Note**
>
> コンソールを使用してビルドプロジェクトを作成または更新する場合、同時に CodeBuild サービスロールを作成できます。デフォルトでは、ロールはそのビルドプロジェクトでのみ使用できます。コンソールでは、このサービスロールを別のビルドプロジェクトと関連付けると、この別のビルドプロジェクトで使用できるようにロールが更新されます。サービスロールは最大 10 個のビルドプロジェクトで使用できます。

8. [Buildspec] で、次のいずれかを行います。

- [Use a buildspec file] (ビルド仕様ファイルの使用) を選択して、ソースコードのルートディレクトリの buildspec.yml を使用します。
- [ビルドコマンドの挿入] を選択して、コンソールを使用してビルドコマンドを挿入します。

詳細については、「ビルド仕様 (buildspec) に関するリファレンス」を参照してください。

9. [アーティファクト] で、次のようにします。

   タイプ

   次のいずれかを選択します。

   - ビルド出力アーティファクトを作成しない場合は、[No artifacts (アーティファクトなし)] を選択します。
   - ビルド出力を S3 バケットに保存する場合は、[Amazon S3] を選択して次のいずれかの操作を行います。
     - ビルド出力 ZIP ファイルまたはフォルダにプロジェクト名を使用する場合は、[Name (名前)] を空白のままにします。それ以外の場合は、名前を入力します。デフォルトでは、アーティファクト名はプロジェクト名です。別の名前を使用する場合は、アーティファクト名ボックスに名前を入力します。ZIP ファイルを出力する場合は、zip 拡張子を含めます。
     - [Bucket name (バケット名)] で、出力バケットの名前を選択します。
     - この手順の前の方で [ビルドコマンドの挿入] を選択した場合は、[出力ファイル] に、ビルド出力 ZIP ファイルまたはフォルダに格納するビルドのファイルの場所を入力します。複数の場所の場合は、各場所をコンマで区切ります (例: `appspec.yml, target/my-app.jar`)。詳細については、「`files`」で buildspec の構文 の説明を参照してください。

   追加設定

   [Additional configuration (追加設定)] オプションを展開し、必要に応じてオプションを設定します。

10. [Create build project (ビルドプロジェクトの作成)] を選択します。[確認] ページで、[ビルドの開始] を選択してビルドを実行します。

Bitbucket ウェブフックを使用してビルドをトリガーする

Bitbucket ウェブフックを使用するプロジェクトの場合、Bitbucket リポジトリがソースコードの変更を検出したときにビルド AWS CodeBuild を作成します。

1. https://console.aws.amazon.com/codesuite/codebuild/home で AWS CodeBuild コンソールを開きます。
2. ナビゲーションペインで [ビルドプロジェクト] を選択後、ウェブフックを使用して Bitbucket リポジトリと関連付けられているプロジェクトを選択します。Bitbucket Webhook プロジェクトの作成の詳細については、「the section called “Bitbucket をソースリポジトリとするビルドプロジェクトを作成し、ウェブフックを有効にする”」を参照してください。
3. プロジェクトの Bitbucket リポジトリのコードに一部変更を加えます。
4. Bitbucket リポジトリにプルリクエストを作成します。詳細については、「プルリクエストを行う」を参照してください。
5. Bitbucket ウェブフックページで、[View request (リクエストの表示)] を選択して最新イベントのリストを表示します。
6. 詳細を表示を選択して、 によって返されたレスポンスの詳細を表示します CodeBuild。次のように表示されます。

```
"response":"Webhook received and build started: https://us-
east-1.console.aws.amazon.com/codebuild/home..."
"statusCode":200
```

7. Bitbucket プルリクエストページに移動して、ビルドのステータスを表示します。

GitHub の Enterprise Server サンプル CodeBuild

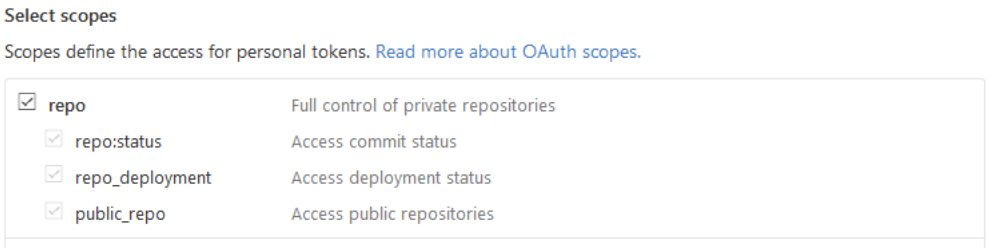
AWS CodeBuild は、ソースリポジトリとして GitHub Enterprise Server をサポートします。このサンプルでは、 GitHub Enterprise Server リポジトリに証明書がインストールされている場合に CodeBuild プロジェクトを設定する方法を示します。また、ウェブフックを有効にして、コード変更が GitHub Enterprise Server リポジトリにプッシュされるたびに がソースコードを CodeBuild 再構築する方法も示します。

前提条件

1. CodeBuild プロジェクトの個人用アクセストークンを生成します。 GitHub Enterprise ユーザーを作成し、このユーザーの個人用アクセストークンを生成することをお勧めします。 CodeBuild プロジェクトの作成時に使用できるように、クリップボードにコピーします。詳細については、

ヘルプウェブサイトの「コマンドライン用の個人用アクセストークンの作成」を参照してください。 GitHub

個人用アクセストークンを作成するときには、定義にリポジトリスコープを含めてください。



2. GitHub Enterprise Server. CodeBuild uses から証明書をダウンロードして、リポジトリへの信頼できる SSL 接続を確立します。

   Linux/macOS クライアント:

   のターミナルウィンドウから、以下のコマンドを実行します。

   ```
   echo -n | openssl s_client -connect HOST:PORTNUMBER \
       | sed -ne '/-BEGIN CERTIFICATE-/,/-END CERTIFICATE-/p' > /folder/filename.pem
   ```

   コマンドのプレースホルダを次の値で置き換えます。

   HOST GitHub Enterprise Server リポジトリの IP アドレス。

   PORTNUMBER。接続するときに使用するポート番号 (たとえば、443)。

   folder 証明書をダウンロードしたフォルダ。

   filename 証明書ファイルのファイル名。

   > **Important**
   > 証明書を .pem ファイルとして保存します。

   Windows クライアント:

   ブラウザを使用して、 GitHub Enterprise Server から証明書をダウンロードします。サイトの証明書の詳細を表示するには、南京錠アイコンを選択します。証明書をエクスポートする方法についての詳細は、ブラウザのドキュメントを参照してください。

> **Important**
>
> 証明書を .pem ファイルとして保存します。

3. 証明書ファイルを S3 バケットにアップロードします。S3 バケットを作成する方法については、「S3 バケットを作成する方法」を参照してください。S3 バケットにオブジェクトをアップロードする方法については、「バケットにファイルとフォルダをアップロードする方法」を参照してください。

> **Note**
>
> このバケットはビルドと同じ AWS リージョンにある必要があります。例えば、米国東部 (オハイオ) リージョンでビルドを実行する CodeBuild ように に指示する場合、バケットは米国東部 (オハイオ) リージョンにある必要があります。

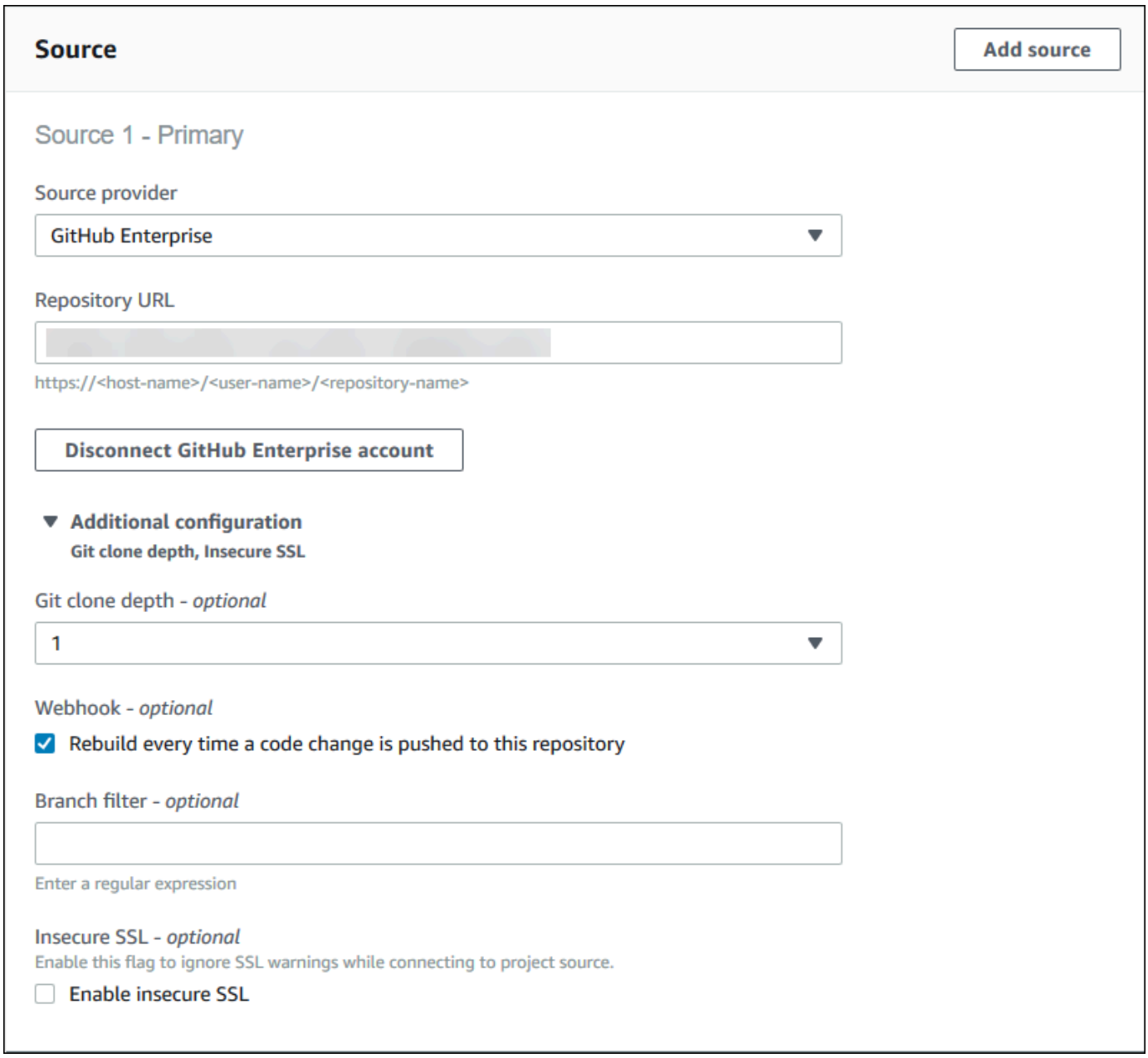
Enterprise Server を GitHubソースリポジトリとしてビルドプロジェクトを作成し、ウェブフックを有効にする (コンソール)

1. https://console.aws.amazon.com/codesuite/codebuild/home で AWS CodeBuild コンソールを開きます。
2. CodeBuild 情報ページが表示される場合は、ビルドプロジェクトの作成を選択します。それ以外の場合は、ナビゲーションペインでビルドを展開し、[ビルドプロジェクト] を選択し、次に [Create build project (ビルドプロジェクトの作成)] を選択します。
3. [プロジェクト名] に、このビルドプロジェクトの名前を入力します。ビルドプロジェクト名は、各 AWS アカウントで一意である必要があります。また、他のユーザーがこのプロジェクトの使用目的を理解できるように、ビルドプロジェクトの説明を任意で指定することもできます。
4. ソース のソースプロバイダー で、GitHub エンタープライズ を選択します。
   - [個人用アクセストークン] には、クリップボードにコピーしたトークンを貼り付け、[トークンの保存] を選択します。リポジトリ URL に、 GitHub Enterprise Server リポジトリの URL を入力します。

> **Note**
>
> 個人用アクセストークンは、一回のみ入力して保存することが必要となります。以降のプロジェクトはすべてこのトークン AWS CodeBuild を使用します。

- [Repository URL] に、リポジトリへのパス (例: リポジトリの名前) を入力します。
- [Additional configuration (追加設定)] を展開します。
- [Rebuild every time a code change is pushed to this repository (コード変更がこのリポジトリにプッシュされるたび再構築)] を選択して、コード変更がこのリポジトリにプッシュされるたびに再構築します。
- GitHub Enterprise Server プロジェクトリポジトリへの接続中に SSL 警告を無視するには、安全でない SSL を有効にするを選択します。

> **Note**
>
> [Enable insecure SSL (セキュアでない SSL を有効にする)] はテストのみに使用することが推奨されます。本番環境では使用しないでください。



5. [環境] で以下の操作を行います。

   [Environment image (環境イメージ)] で、次のいずれかの操作を行います。

   - によって管理される Docker イメージを使用するには AWS CodeBuild、マネージドイメージを選択し、オペレーティングシステム 、ランタイム (s)、イメージ 、イメージバージョン から選択します。利用可能な場合は、[環境タイプ] から選択します。
   - 別の Docker イメージを使用するには、[カスタムイメージ] を選択します。[Environment type (環境タイプ)] で、 [ARM]、[Linux]、[Linux GPU] または [Windows] を選択します。[Other

registry (その他のレジストリ)] を選択した場合は、[External registry URL (外部のレジストリ URL)] に *docker repository*/*docker image name* の形式に従って Docker Hub の Docker イメージの名前とタグを入力します。Amazon ECR を選択した場合は、Amazon ECR リポジトリと Amazon ECR イメージを使用して、 AWS アカウントの Docker イメージを選択します。

- プライベート Docker イメージを使用するには、[カスタムイメージ] を選択します。[Environment type (環境タイプ)] で、 [ARM]、[Linux]、[Linux GPU] または [Windows] を選択します。[Image registry (イメージレジストリ)] に [Other registry (その他のレジストリ)] を選択して、その後プライベート Docker イメージの認証情報の ARN を入力します。認証情報は、Secrets Manager で作成する必要があります。詳細については、AWS Secrets Manager ユーザーガイドの「AWS Secrets Managerとは」を参照してください。

6. [Service role (サービスロール)] で、次のいずれかの操作を行います。

   - CodeBuild サービスロールがない場合は、新しいサービスロール を選択します。[Role name] に、新しいロールの名前を入力します。
   - CodeBuild サービスロールがある場合は、既存のサービスロール を選択します。[Role ARN] で、サービスロールを選択します。

   > **Note**
   >
   > コンソールを使用してビルドプロジェクトを作成または更新する場合、同時に CodeBuild サービスロールを作成できます。デフォルトでは、ロールはそのビルドプロジェクトでのみ使用できます。コンソールでは、このサービスロールを別のビルドプロジェクトと関連付けると、この別のビルドプロジェクトで使用できるようにロールが更新されます。サービスロールは最大 10 個のビルドプロジェクトで使用できます。

7. [Additional configuration (追加設定)] を展開します。

   VPC CodeBuild を使用する場合：

   - VPC の場合、 が CodeBuild 使用する VPC ID を選択します。
   - VPC サブネット で、 が CodeBuild 使用するリソースを含むサブネットを選択します。
   - VPC セキュリティグループ で、 CodeBuild が VPCs内のリソースへのアクセスを許可するために使用するセキュリティグループを選択します。

詳細については、「Amazon Virtual Private Cloud AWS CodeBuild で を使用する」を参照してください。

8. [Buildspec] で、次のいずれかを行います。

   - [Use a buildspec file] (ビルド仕様ファイルの使用) を選択して、ソースコードのルートディレクトリの buildspec.yml を使用します。
   - [ビルドコマンドの挿入] を選択して、コンソールを使用してビルドコマンドを挿入します。

   詳細については、「ビルド仕様 (buildspec) に関するリファレンス」を参照してください。

9. [アーティファクト] の [タイプ] で、次のいずれかの操作を行います。

   - ビルド出力アーティファクトを作成しない場合は、[No artifacts (アーティファクトなし)] を選択します。
   - ビルド出力を S3 バケットに保存する場合は、[Amazon S3] を選択して次のいずれかの操作を行います。
     - ビルド出力 ZIP ファイルまたはフォルダにプロジェクト名を使用する場合は、[Name (名前)] を空白のままにします。それ以外の場合は、名前を入力します。デフォルトでは、アーティファクト名はプロジェクト名です。別の名前を使用する場合は、アーティファクト名ボックスに名前を入力します。ZIP ファイルを出力する場合は、zip 拡張子を含めます。
     - [Bucket name (バケット名)] で、出力バケットの名前を選択します。
     - この手順の前の方で [ビルドコマンドの挿入] を選択した場合は、[出力ファイル] に、ビルド出力 ZIP ファイルまたはフォルダに格納するビルドのファイルの場所を入力します。複数の場所の場合は、各場所をコンマで区切ります (例: `appspec.yml, target/my-app.jar`)。詳細については、「`files`」で buildspec の構文 の説明を参照してください。

10. [キャッシュタイプ] で、以下のいずれかを選択します。

    - キャッシュを使用しない場合は、[No cache] を選択します。
    - Amazon S3 キャッシュを使用するには、[Amazon S3] を選択して次の操作を行います。
      - [バケット] では、キャッシュが保存される S3 バケットの名前を選択します。
      - (オプション) [Cache path prefix (キャッシュパスのプレフィックス)] に、Amazon S3 パスのプレフィックスを入力します。[キャッシュパスのプレフィックス] 値はディレクトリ名に似ています。これにより、バケット内の同じディレクトリにキャッシュを保存できます。

> **Important**
> パスのプレフィックスの末尾にスラッシュ (/) を付加しないでください。

- ローカルキャッシュを使用する場合は、[ローカル] を選択し、ローカルキャッシュモードを 1 つ以上選択します。

> **Note**
> Docker レイヤーキャッシュモードは Linux でのみ利用可能です。このモードを選択する場合、プロジェクトは権限モードで実行する必要があります。

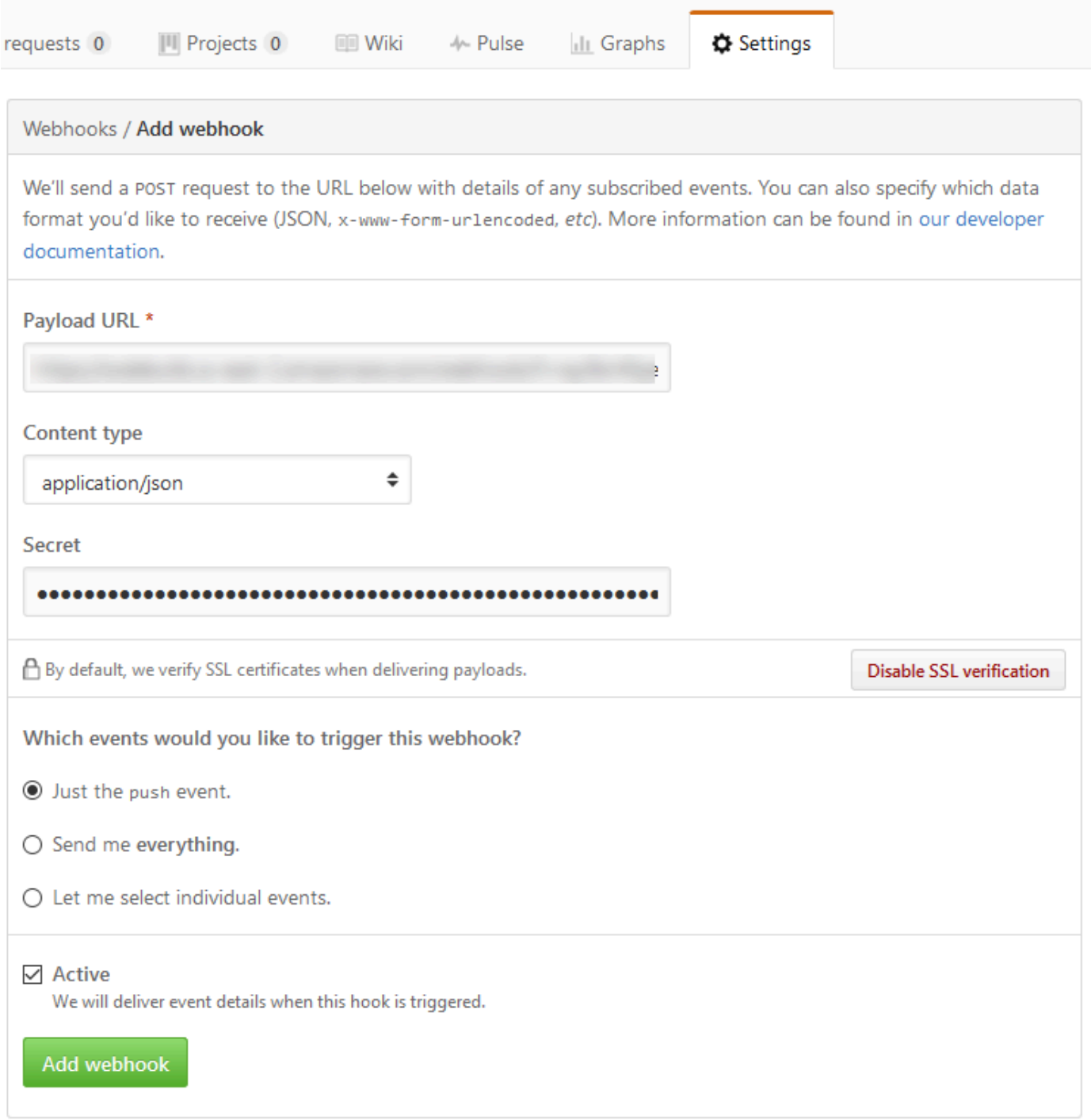
キャッシュを使用すると、再利用可能なビルド環境がキャッシュに保存され、ビルド全体で使用されるため、かなりのビルド時間が節約されます。ビルド仕様ファイルのキャッシュの指定に関する詳細については、「buildspec の構文」を参照してください。キャッシングの詳細については、「AWS CodeBuild でのキャッシュのビルド」を参照してください。

11. [Create build project (ビルドプロジェクトの作成)] を選択します。ビルドプロジェクトページで、[Start build (ビルドの開始)] を選択します。
12. [ソース] でウェブフックを有効にすると、[ペイロードの URL] の値および [シークレット] を示した [ウェブフックの作成] ダイアログボックスが表示されます。

> **Important**
> [ウェブフックの作成] ダイアログボックスが表示されるのは 1 回だけです。ペイロード URL とシークレットキーをコピーします。Enterprise GitHub Server でウェブフックを追加するときに必要になります。
> ペイロード URL とシークレットキーを再度生成する必要がある場合は、まず GitHub Enterprise Server リポジトリからウェブフックを削除する必要があります。 CodeBuild プロジェクトで、Webhook チェックボックスをオフにし、保存を選択します。その後、CodeBuildWebhook チェックボックスをオンにしてプロジェクトを作成または更新できます。[ウェブフックの作成] ダイアログボックスが再度表示されます。

13. GitHub Enterprise Server で、 CodeBuild プロジェクトが保存されているリポジトリを選択します。
14. [設定]、[Hooks & services]、[Add webhook] の順に選択します。

15. ペイロード URL とシークレットキーを入力し、その他のフィールドにはデフォルト値を選択して、[Add webhook] を選択します。



16. CodeBuild プロジェクトに戻ります。[ウェブフックの作成] ダイアログボックスを閉じ、[ビルドの開始] を選択します。

GitHub のプルリクエストとウェブフックフィルターのサンプル CodeBuild

AWS CodeBuild は、ソースリポジトリが の場合、ウェブフックをサポートします GitHub。つまり、ソースコードが GitHub リポジトリに保存されている CodeBuild ビルドプロジェクトでは、コード変更がリポジトリにプッシュされるたびにウェブフックを使用してソースコードを再構築できます。サンプルについては CodeBuild 、「サンプルAWS CodeBuild」を参照してください。

> **Note**
>
> Webhook を使用する場合、ユーザーが予期しないビルドをトリガーする可能性があります。このリスクを軽減するには、「ウェブフック使用のベストプラクティス。」を参照してください。

ソースリポジトリとして を使用して GitHubビルドプロジェクトを作成し、ウェブフックを有効にする (コンソール)

1. https://console.aws.amazon.com/codesuite/codebuild/home で AWS CodeBuild コンソールを開きます。
2. CodeBuild 情報ページが表示されたら、ビルドプロジェクトの作成を選択します。それ以外の場合は、ナビゲーションペインでビルドを展開し、[ビルドプロジェクト] を選択し、次に [Create build project (ビルドプロジェクトの作成)] を選択します。
3. [Create build project (ビルドプロジェクトの作成)] を選択します。
4. [Project configuration (プロジェクトの設定)] で、次のようにします。

   [Project name] (プロジェクト名)

   このビルドプロジェクトの名前を入力します。ビルドプロジェクト名は、各 AWS アカウントで一意である必要があります。また、他のユーザーがこのプロジェクトの使用目的を理解できるように、ビルドプロジェクトの説明を任意で指定することもできます。
5. [Source (ソース)] で、次のようにします。

   ソースプロバイダー

   を選択しますGitHub。「」の指示に従って と接続 (または再接続) し GitHub 、「承認」を選択します。

リポジトリ

アカウント GitHub のリポジトリ を選択します。

GitHub リポジトリ

GitHub リポジトリの URL を入力します。

6. [プライマリソース Webhook イベント] で、以下を選択します。

> **Note**
> プライマリソースのウェブフックイベントセクションは、前のステップで GitHub アカウントでリポジトリを選択した場合にのみ表示されます。

1. プロジェクトの作成時に [コードの変更がこのレポジトリにプッシュされるたびに再構築する] を選択します。
2. [イベントタイプ] から、1 つ以上のイベントを選択します。
3. イベントでビルドをトリガーされた時間をフィルタリングするには、[これらの条件でビルドを開始する] で、1 つ以上のオプションフィルタを追加します。
4. イベントがトリガーされていない時間をフィルタリングするには、[これらの条件でビルドを開始しない] で、1 つ以上のオプションフィルタを追加します。
5. 別のフィルタグループを追加する必要がある場合、[フィルタグループの追加] を選択します。

GitHub Webhook イベントタイプとフィルターの詳細については、「」を参照してくださいGitHub ウェブフックイベント。

7. [環境] で以下の操作を行います。

環境イメージ

以下のうちのひとつを選択します。

によって管理される Docker イメージを使用するには AWS CodeBuild :

[Managed image (マネージドイメージ)] を選択し、次に [オペレーティングシステム]、[ランタイム]、[イメージ]、および [ランタイムバージョン] で適切な選択を行います。利用可能な場合は、[環境タイプ] から選択します。

別の Docker イメージを使用するには:

[カスタムイメージ] を選択します。[Environment type (環境タイプ)] で、 [ARM]、[Linux]、[Linux GPU] または [Windows] を選択します。[Other registry (その他のレジストリ)] を選択した場合は、[External registry URL (外部のレジストリ URL)] に *docker repository*/*docker image name* の形式に従って Docker Hub の Docker イメージの名前とタグを入力します。[Amazon ECR] を選択した場合は、[Amazon ECR repository] (Amazon ECR レポジトリ) および [Amazon ECR image] (Amazon ECR イメージ) を使用して AWS アカウントの Docker イメージを選択します。

プライベートDockerイメージを使用するには :

[カスタムイメージ] を選択します。[Environment type (環境タイプ)] で、 [ARM]、[Linux]、[Linux GPU] または [Windows] を選択します。[Image registry (イメージレジストリ)] に [Other registry (その他のレジストリ)] を選択して、その後プライベート Docker イメージの認証情報の ARN を入力します。認証情報は、Secrets Manager で作成する必要があります。詳細については、「 AWS Secrets Manager ユーザーガイド」の「 AWS Secrets Managerとは」を参照してください。

サービスロール

以下のうちのひとつを選択します。

- CodeBuild サービスロールがない場合は、新しいサービスロール を選択します。[Role name] に、新しいロールの名前を入力します。
- CodeBuild サービスロールがある場合は、既存のサービスロール を選択します。[Role ARN] で、サービスロールを選択します。

> **Note**
>
> コンソールを使用してビルドプロジェクトを作成または更新する場合、同時に CodeBuild サービスロールを作成できます。デフォルトでは、ロールはそのビルドプロジェクトでのみ使用できます。コンソールでは、このサービスロールを別のビルドプロジェクトと関連付けると、この別のビルドプロジェクトで使用できるようにロールが更新されます。サービスロールは最大 10 個のビルドプロジェクトで使用できます。

8. [Buildspec] で、次のいずれかを行います。

- [Use a buildspec file] (ビルド仕様ファイルの使用) を選択して、ソースコードのルートディレクトリの buildspec.yml を使用します。
- [ビルドコマンドの挿入] を選択して、コンソールを使用してビルドコマンドを挿入します。

詳細については、「ビルド仕様 (buildspec) に関するリファレンス」を参照してください。

9. [アーティファクト] で、次のようにします。

   タイプ

   次のいずれかを選択します。

   - ビルド出力アーティファクトを作成しない場合は、[No artifacts (アーティファクトなし)] を選択します。
   - ビルド出力を S3 バケットに保存する場合は、[Amazon S3] を選択して次のいずれかの操作を行います。
     - ビルド出力 ZIP ファイルまたはフォルダにプロジェクト名を使用する場合は、[Name (名前)] を空白のままにします。それ以外の場合は、名前を入力します。デフォルトでは、アーティファクト名はプロジェクト名です。別の名前を使用する場合は、アーティファクト名ボックスに名前を入力します。ZIP ファイルを出力する場合は、zip 拡張子を含めます。
     - [Bucket name (バケット名)] で、出力バケットの名前を選択します。
     - この手順の前の方で [ビルドコマンドの挿入] を選択した場合は、[出力ファイル] に、ビルド出力 ZIP ファイルまたはフォルダに格納するビルドのファイルの場所を入力します。複数の場所の場合は、各場所をコンマで区切ります (例: `appspec.yml, target/my-app.jar`)。詳細については、「`files`」で buildspec の構文 の説明を参照してください。

   追加設定

   [Additional configuration (追加設定)] オプションを展開し、必要に応じてオプションを設定します。

10. [Create build project (ビルドプロジェクトの作成)] を選択します。[確認] ページで、[ビルドの開始] を選択してビルドを実行します。

検証チェック

1. https://console.aws.amazon.com/codesuite/codebuild/home で AWS CodeBuild コンソールを開きます。
2. ナビゲーションペインで、[Build projects] を選択します。
3. 次のいずれかを行ってください。
   - 確認する Webhook を持つビルドプロジェクトのリンクを選択し、[ビルドの詳細] を選択します。
   - 確認する Webhook を持つビルドプロジェクトの横にあるラジオボタンを選択して、[View details] (詳細を表示) を選択後、[ビルドの詳細] タブを選択します。
4. [プライマリソース Webhook イベント] で、[Webhook] の URL リンクを選択します。
5. GitHub リポジトリの「設定」ページの「ウェブフック」で、プルリクエストとプッシュが選択されていることを確認します。
6. GitHub プロファイル設定の「個人設定」、「アプリケーション」、「認可された OAuth アプリ」で、選択した AWS リージョンへのアクセスがアプリケーションに許可されていることがわかります。

セマンティックバージョニングを使用してビルドアーティファクトに名前を付けるサンプル

このサンプルには、ビルド時に作成するアーティファクト名の指定方法を示す buildspec ファイルのサンプルが含まれています。buildspec ファイルで指定される名前には、シェルコマンドと環境変数を組み込んで、一意の名前にすることができます。buildspec で指定した名前は、プロジェクトの作成時にコンソールに入力した名前よりも優先されます。

複数回ビルドする場合、buildspec ファイルで指定されたアーティファクト名を使用すると、出力アーティファクトファイル名が一意であることが保証されます。たとえば、ビルド時にアーティファクト名に日付とタイムスタンプを挿入できます。

コンソールで入力したアーティファクト名を buildspec ファイルの名前で上書きする場合は、次のようにします。

1. ビルドプロジェクトを設定して、アーティファクト名を buildspec ファイル内の名前で上書きします。

- コンソールを使用してビルドプロジェクトを作成する場合は、[Enable semantic versioning (セマンティックバージョニングを有効にする)] を選択します。詳細については、「ビルドプロジェクトの作成 (コンソール)」を参照してください。
- を使用する場合は AWS CLI、 に渡された JSON 形式のファイルで を `overrideArtifactName` true に設定します`create-project`。詳細については、「ビルドプロジェクトの作成 (AWS CLI)」を参照してください。
- AWS CodeBuild API を使用する場合は、プロジェクトの作成または更新時、またはビルドの開始時に`ProjectArtifacts`、 オブジェクトに `overrideArtifactName`フラグを設定します。

2. buildspec ファイルに名前を指定します 次のサンプルの buildspec ファイルを参考として使用してください。

この Linux の例は、ビルドが作成された日付を含むアーティファクト名を指定する方法を示しています。

```
version: 0.2
phases:
  build:
    commands:
      - rspec HelloWorld_spec.rb
artifacts:
  files:
    - '**/*'
  name: myname-$(date +%Y-%m-%d)
```

この Linux の例は、 CodeBuild 環境変数を使用するアーティファクト名を指定する方法を示しています。詳細については、「ビルド環境の環境変数」を参照してください。

```
version: 0.2
phases:
  build:
    commands:
      - rspec HelloWorld_spec.rb
artifacts:
  files:
    - '**/*'
  name: myname-$AWS_REGION
```

この Windows の例は、ビルドが作成された日時を含むアーティファクト名を指定する方法を示しています。

```
version: 0.2
env:
  variables:
    TEST_ENV_VARIABLE: myArtifactName
phases:
  build:
    commands:
      - cd samples/helloworld
      - dotnet restore
      - dotnet run
artifacts:
  files:
    - '**/*'
  name: $Env:TEST_ENV_VARIABLE-$(Get-Date -UFormat "%Y%m%d-%H%M%S")
```

この Windows の例は、buildspec ファイルで宣言された変数と CodeBuild 環境変数を使用するアーティファクト名を指定する方法を示しています。詳細については、「ビルド環境の環境変数」を参照してください。

```
version: 0.2
env:
  variables:
    TEST_ENV_VARIABLE: myArtifactName
phases:
  build:
    commands:
      - cd samples/helloworld
      - dotnet restore
      - dotnet run
artifacts:
  files:
    - '**/*'
  name: $Env:TEST_ENV_VARIABLE-$Env:AWS_REGION
```

詳細については、「のビルド仕様リファレンス CodeBuild」を参照してください。

の Microsoft Windows サンプル CodeBuild

これらのサンプルでは、Microsoft Windows Server 2019、.NET Framework、.NET Core SDK を実行する AWS CodeBuild ビルド環境を使用して、F# と Visual Basic で記述されたコードからランタイムファイルを構築します。

> **Important**
>
> これらのサンプルを実行すると、 AWS アカウントに課金される場合があります。これには、Amazon S3 に関連する AWS リソースとアクション AWS KMS、および CloudWatch Logs CodeBuild に対して発生する可能性がある料金が含まれます。 Amazon S3 詳細については、「 のCodeBuild料金」、「Amazon S3 の料金」、AWS Key Management Service「 の料金」、および「Amazon の CloudWatch 料金」を参照してください。

サンプルの実行

これらのサンプルを実行するには

1. このトピックの「ディレクトリ構造」セクションと「ファイル」セクションの説明に従ってファイルを作成し、S3 入力バケットにアップロードするか、 CodeCommit または GitHub リポジトリにアップロードします。

   > **Important**
   >
   > *(root directory name)* をアップロードしないでください。アップロードするのは、*(root directory name)* 内のファイルのみです。
   > S3 入力バケットを使用している場合は、ファイルを必ず ZIP ファイルに圧縮してから入力バケットにアップロードしてください。*(root directory name)* を ZIP ファイルに追加しないでください。追加するのは、*(root directory name)* 内のファイルのみです。

2. ビルドプロジェクトを作成します。ビルドプロジェクトは、`mcr.microsoft.com/dotnet/framework/sdk:4.8` イメージを使用して、.NET Framework プロジェクトをビルドします。

   を使用してビルドプロジェクト AWS CLI を作成する場合、 `create-project` コマンドへの JSON 形式の入力は次のようになります。(プレースホルダは独自の値に置き換えてください。)

```
{
  "name": "sample-windows-build-project",
  "source": {
    "type": "S3",
    "location": "codebuild-region-ID-account-ID-input-bucket/windows-build-input-artifact.zip"
  },
  "artifacts": {
    "type": "S3",
    "location": "codebuild-region-ID-account-ID-output-bucket",
    "packaging": "ZIP",
    "name": "windows-build-output-artifact.zip"
  },
  "environment": {
    "type": "WINDOWS_SERVER_2019_CONTAINER",
    "image": "mcr.microsoft.com/dotnet/framework/sdk:4.8",
    "computeType": "BUILD_GENERAL1_MEDIUM"
  },
  "serviceRole": "arn:aws:iam::account-ID:role/role-name",
  "encryptionKey": "arn:aws:kms:region-ID:account-ID:key/key-ID"
}
```

3. ビルドを実行し、「ビルドの実行」の手順を実行します。
4. ビルド出力のアーティファクトを取得するには、S3 出力バケットで、*windows-build-output-artifact*.zip ファイルをローカルコンピュータまたはインスタンスにダウンロードします。コンテンツを抽出し、ランタイムおよび他のファイルにアクセスします。

   - .NET Framework を使用する F# サンプルのランタイムファイルは FSharpHelloWorld.exe、 FSharpHelloWorld\bin\Debug ディレクトリにあります。
   - .NET Framework を使用する Visual Basic サンプルのランタイムファイルは VBHelloWorld.exe、 VBHelloWorld\bin\Debug ディレクトリにあります。

ディレクトリ構造

これらのサンプルで想定しているディレクトリ構造は以下のとおりです。

F# と .NET Framework

```
(root directory name)
```

```
### buildspec.yml
### FSharpHelloWorld.sln
### FSharpHelloWorld
    ### App.config
    ### AssemblyInfo.fs
    ### FSharpHelloWorld.fsproj
    ### Program.fs
```

Visual Basic と .NET Framework

```
(root directory name)
### buildspec.yml
### VBHelloWorld.sln
### VBHelloWorld
    ### App.config
    ### HelloWorld.vb
    ### VBHelloWorld.vbproj
    ### My Project
        ### Application.Designer.vb
        ### Application.myapp
        ### AssemblyInfo.vb
        ### Resources.Designer.vb
        ### Resources.resx
        ### Settings.Designer.vb
        ### Settings.settings
```

ファイル

これらのサンプルでは、以下のファイルを使用します。

F# と .NET Framework

buildspec.yml (*(root directory name)* 内)

```
version: 0.2

env:
  variables:
    SOLUTION: .\FSharpHelloWorld.sln
    PACKAGE_DIRECTORY: .\packages
    DOTNET_FRAMEWORK: 4.8

phases:
```

```
  build:
    commands:
      - '& nuget restore $env:SOLUTION -PackagesDirectory $env:PACKAGE_DIRECTORY'
      - '& msbuild -p:FrameworkPathOverride="C:\Program Files (x86)\Reference
 Assemblies\Microsoft\Framework\.NETFramework\v$env:DOTNET_FRAMEWORK" $env:SOLUTION'
artifacts:
  files:
    - .\FSharpHelloWorld\bin\Debug\*
```

FSharpHelloWorld.sln (*(root directory name)* 内)

```
Microsoft Visual Studio Solution File, Format Version 12.00
# Visual Studio 14
VisualStudioVersion = 14.0.25420.1
MinimumVisualStudioVersion = 10.0.40219.1
Project("{F2A71F9B-5D33-465A-A702-920D77279786}") = "FSharpHelloWorld",
 "FSharpHelloWorld\FSharpHelloWorld.fsproj", "{D60939B6-526D-43F4-9A89-577B2980DF62}"
EndProject
Global
  GlobalSection(SolutionConfigurationPlatforms) = preSolution
    Debug|Any CPU = Debug|Any CPU
    Release|Any CPU = Release|Any CPU
  EndGlobalSection
  GlobalSection(ProjectConfigurationPlatforms) = postSolution
    {D60939B6-526D-43F4-9A89-577B2980DF62}.Debug|Any CPU.ActiveCfg = Debug|Any CPU
    {D60939B6-526D-43F4-9A89-577B2980DF62}.Debug|Any CPU.Build.0 = Debug|Any CPU
    {D60939B6-526D-43F4-9A89-577B2980DF62}.Release|Any CPU.ActiveCfg = Release|Any CPU
    {D60939B6-526D-43F4-9A89-577B2980DF62}.Release|Any CPU.Build.0 = Release|Any CPU
  EndGlobalSection
  GlobalSection(SolutionProperties) = preSolution
    HideSolutionNode = FALSE
  EndGlobalSection
EndGlobal
```

App.config (*(root directory name)*\FSharpHelloWorld 内)

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <startup>
    <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.8" />
  </startup>
</configuration>
```

AssemblyInfo.fs (*(root directory name)*\FSharpHelloWorld 内)

```
namespace FSharpHelloWorld.AssemblyInfo

open System.Reflection
open System.Runtime.CompilerServices
open System.Runtime.InteropServices

// General Information about an assembly is controlled through the following
// set of attributes. Change these attribute values to modify the information
// associated with an assembly.
[<assembly: AssemblyTitle("FSharpHelloWorld")>]
[<assembly: AssemblyDescription("")>]
[<assembly: AssemblyConfiguration("")>]
[<assembly: AssemblyCompany("")>]
[<assembly: AssemblyProduct("FSharpHelloWorld")>]
[<assembly: AssemblyCopyright("Copyright ©  2017")>]
[<assembly: AssemblyTrademark("")>]
[<assembly: AssemblyCulture("")>]

// Setting ComVisible to false makes the types in this assembly not visible
// to COM components.  If you need to access a type in this assembly from
// COM, set the ComVisible attribute to true on that type.
[<assembly: ComVisible(false)>]

// The following GUID is for the ID of the typelib if this project is exposed to COM
[<assembly: Guid("d60939b6-526d-43f4-9a89-577b2980df62")>]

// Version information for an assembly consists of the following four values:
//
// Major Version
// Minor Version
// Build Number
// Revision
//
// You can specify all the values or you can default the Build and Revision Numbers
// by using the '*' as shown below:
// [<assembly: AssemblyVersion("1.0.*")>]
[<assembly: AssemblyVersion("1.0.0.0")>]
[<assembly: AssemblyFileVersion("1.0.0.0")>]

do
  ()
```

FSharpHelloWorld.fsproj (*(root directory name)*\FSharpHelloWorld 内)

```
<?xml version="1.0" encoding="utf-8"?>
<Project ToolsVersion="14.0" DefaultTargets="Build" xmlns="http://
schemas.microsoft.com/developer/msbuild/2003">
  <Import Project="$(MSBuildExtensionsPath)\
$(MSBuildToolsVersion)\Microsoft.Common.props"
 Condition="Exists('$(MSBuildExtensionsPath)\
$(MSBuildToolsVersion)\Microsoft.Common.props')" />
  <PropertyGroup>
    <Configuration Condition=" '$(Configuration)' == '' ">Debug</Configuration>
    <Platform Condition=" '$(Platform)' == '' ">AnyCPU</Platform>
    <SchemaVersion>2.0</SchemaVersion>
    <ProjectGuid>d60939b6-526d-43f4-9a89-577b2980df62</ProjectGuid>
    <OutputType>Exe</OutputType>
    <RootNamespace>FSharpHelloWorld</RootNamespace>
    <AssemblyName>FSharpHelloWorld</AssemblyName>
    <TargetFrameworkVersion>v4.8</TargetFrameworkVersion>
    <AutoGenerateBindingRedirects>true</AutoGenerateBindingRedirects>
    <TargetFSharpCoreVersion>4.4.0.0</TargetFSharpCoreVersion>
    <Name>FSharpHelloWorld</Name>
  </PropertyGroup>
  <PropertyGroup Condition=" '$(Configuration)|$(Platform)' == 'Debug|AnyCPU' ">
    <DebugSymbols>true</DebugSymbols>
    <DebugType>full</DebugType>
    <Optimize>false</Optimize>
    <Tailcalls>false</Tailcalls>
    <OutputPath>bin\Debug\</OutputPath>
    <DefineConstants>DEBUG;TRACE</DefineConstants>
    <WarningLevel>3</WarningLevel>
    <PlatformTarget>AnyCPU</PlatformTarget>
    <DocumentationFile>bin\Debug\FSharpHelloWorld.XML</DocumentationFile>
    <Prefer32Bit>true</Prefer32Bit>
  </PropertyGroup>
  <PropertyGroup Condition=" '$(Configuration)|$(Platform)' == 'Release|AnyCPU' ">
    <DebugType>pdbonly</DebugType>
    <Optimize>true</Optimize>
    <Tailcalls>true</Tailcalls>
    <OutputPath>bin\Release\</OutputPath>
    <DefineConstants>TRACE</DefineConstants>
    <WarningLevel>3</WarningLevel>
    <PlatformTarget>AnyCPU</PlatformTarget>
    <DocumentationFile>bin\Release\FSharpHelloWorld.XML</DocumentationFile>
    <Prefer32Bit>true</Prefer32Bit>
```

```
  </PropertyGroup>
  <ItemGroup>
    <Reference Include="mscorlib" />
    <Reference Include="FSharp.Core, Version=$(TargetFSharpCoreVersion),
 Culture=neutral, PublicKeyToken=b03f5f7f11d50a3a">
      <Private>True</Private>
    </Reference>
    <Reference Include="System" />
    <Reference Include="System.Core" />
    <Reference Include="System.Numerics" />
  </ItemGroup>
  <ItemGroup>
    <Compile Include="AssemblyInfo.fs" />
    <Compile Include="Program.fs" />
    <None Include="App.config" />
  </ItemGroup>
  <PropertyGroup>
    <MinimumVisualStudioVersion Condition="'$(MinimumVisualStudioVersion)' == ''">11</
MinimumVisualStudioVersion>
  </PropertyGroup>
  <Choose>
    <When Condition="'$(VisualStudioVersion)' == '11.0'">
      <PropertyGroup Condition="Exists('$(MSBuildExtensionsPath32)\..\Microsoft SDKs\F#
\3.0\Framework\v4.0\Microsoft.FSharp.Targets')">
        <FSharpTargetsPath>$(MSBuildExtensionsPath32)\..\Microsoft SDKs\F#
\3.0\Framework\v4.0\Microsoft.FSharp.Targets</FSharpTargetsPath>
      </PropertyGroup>
    </When>
    <Otherwise>
      <PropertyGroup Condition="Exists('$(MSBuildExtensionsPath32)\Microsoft
\VisualStudio\v$(VisualStudioVersion)\FSharp\Microsoft.FSharp.Targets')">
        <FSharpTargetsPath>$(MSBuildExtensionsPath32)\Microsoft\VisualStudio\v
$(VisualStudioVersion)\FSharp\Microsoft.FSharp.Targets</FSharpTargetsPath>
      </PropertyGroup>
    </Otherwise>
  </Choose>
  <Import Project="$(FSharpTargetsPath)" />
  <!-- To modify your build process, add your task inside one of the targets below and
 uncomment it.
       Other similar extension points exist, see Microsoft.Common.targets.
  <Target Name="BeforeBuild">
  </Target>
  <Target Name="AfterBuild">
  </Target>
```

```
  -->
</Project>
```

Program.fs (内)*(root directory name)*\FSharpHelloWorld

```
// Learn more about F# at http://fsharp.org
// See the 'F# Tutorial' project for more help.

[<EntryPoint>]
let main argv =
  printfn "Hello World"
  0 // return an integer exit code
```

Visual Basic と .NET Framework

buildspec.yml (*(root directory name)* 内)

```
version: 0.2

env:
  variables:
    SOLUTION: .\VBHelloWorld.sln
    PACKAGE_DIRECTORY: .\packages
    DOTNET_FRAMEWORK: 4.8

phases:
  build:
    commands:
      - '& "C:\ProgramData\chocolatey\bin\NuGet.exe" restore $env:SOLUTION -
PackagesDirectory $env:PACKAGE_DIRECTORY'
      - '& "C:\Program Files (x86)\MSBuild\14.0\Bin\MSBuild.exe" -
p:FrameworkPathOverride="C:\Program Files (x86)\Reference Assemblies\Microsoft
\Framework\.NETFramework\v$env:DOTNET_FRAMEWORK" $env:SOLUTION'
artifacts:
  files:
    - .\VBHelloWorld\bin\Debug\*
```

VBHelloWorld.sln (*(root directory name)* 内)

```
Microsoft Visual Studio Solution File, Format Version 12.00
# Visual Studio 14
VisualStudioVersion = 14.0.25420.1
```

```
MinimumVisualStudioVersion = 10.0.40219.1
Project("{F184B08F-C81C-45F6-A57F-5ABD9991F28F}") = "VBHelloWorld", "VBHelloWorld
\VBHelloWorld.vbproj", "{4DCEC446-7156-4FE6-8CCC-219E34DD409D}"
EndProject
Global
  GlobalSection(SolutionConfigurationPlatforms) = preSolution
    Debug|Any CPU = Debug|Any CPU
    Release|Any CPU = Release|Any CPU
  EndGlobalSection
  GlobalSection(ProjectConfigurationPlatforms) = postSolution
    {4DCEC446-7156-4FE6-8CCC-219E34DD409D}.Debug|Any CPU.ActiveCfg = Debug|Any CPU
    {4DCEC446-7156-4FE6-8CCC-219E34DD409D}.Debug|Any CPU.Build.0 = Debug|Any CPU
    {4DCEC446-7156-4FE6-8CCC-219E34DD409D}.Release|Any CPU.ActiveCfg = Release|Any CPU
    {4DCEC446-7156-4FE6-8CCC-219E34DD409D}.Release|Any CPU.Build.0 = Release|Any CPU
  EndGlobalSection
  GlobalSection(SolutionProperties) = preSolution
    HideSolutionNode = FALSE
  EndGlobalSection
EndGlobal
```

App.config (*(root directory name)*\VBHelloWorld 内)

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <startup>
    <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.8" />
  </startup>
</configuration>
```

HelloWorld.vb (*(root directory name)*\VBHelloWorld 内)

```
Module HelloWorld

  Sub Main()
    MsgBox("Hello World")
  End Sub

End Module
```

VBHelloWorld.vbproj (*(root directory name)*\VBHelloWorld 内)

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<Project ToolsVersion="14.0" DefaultTargets="Build" xmlns="http://
schemas.microsoft.com/developer/msbuild/2003">
  <Import Project="$(MSBuildExtensionsPath)\
$(MSBuildToolsVersion)\Microsoft.Common.props"
 Condition="Exists('$(MSBuildExtensionsPath)\
$(MSBuildToolsVersion)\Microsoft.Common.props')" />
  <PropertyGroup>
    <Configuration Condition=" '$(Configuration)' == '' ">Debug</Configuration>
    <Platform Condition=" '$(Platform)' == '' ">AnyCPU</Platform>
    <ProjectGuid>{4DCEC446-7156-4FE6-8CCC-219E34DD409D}</ProjectGuid>
    <OutputType>Exe</OutputType>
    <StartupObject>VBHelloWorld.HelloWorld</StartupObject>
    <RootNamespace>VBHelloWorld</RootNamespace>
    <AssemblyName>VBHelloWorld</AssemblyName>
    <FileAlignment>512</FileAlignment>
    <MyType>Console</MyType>
    <TargetFrameworkVersion>v4.8</TargetFrameworkVersion>
    <AutoGenerateBindingRedirects>true</AutoGenerateBindingRedirects>
  </PropertyGroup>
  <PropertyGroup Condition=" '$(Configuration)|$(Platform)' == 'Debug|AnyCPU' ">
    <PlatformTarget>AnyCPU</PlatformTarget>
    <DebugSymbols>true</DebugSymbols>
    <DebugType>full</DebugType>
    <DefineDebug>true</DefineDebug>
    <DefineTrace>true</DefineTrace>
    <OutputPath>bin\Debug\</OutputPath>
    <DocumentationFile>VBHelloWorld.xml</DocumentationFile>
    <NoWarn>42016,41999,42017,42018,42019,42032,42036,42020,42021,42022</NoWarn>
  </PropertyGroup>
  <PropertyGroup Condition=" '$(Configuration)|$(Platform)' == 'Release|AnyCPU' ">
    <PlatformTarget>AnyCPU</PlatformTarget>
    <DebugType>pdbonly</DebugType>
    <DefineDebug>false</DefineDebug>
    <DefineTrace>true</DefineTrace>
    <Optimize>true</Optimize>
    <OutputPath>bin\Release\</OutputPath>
    <DocumentationFile>VBHelloWorld.xml</DocumentationFile>
    <NoWarn>42016,41999,42017,42018,42019,42032,42036,42020,42021,42022</NoWarn>
  </PropertyGroup>
  <PropertyGroup>
    <OptionExplicit>On</OptionExplicit>
  </PropertyGroup>
  <PropertyGroup>
    <OptionCompare>Binary</OptionCompare>
```

```
  </PropertyGroup>
  <PropertyGroup>
    <OptionStrict>Off</OptionStrict>
  </PropertyGroup>
  <PropertyGroup>
    <OptionInfer>On</OptionInfer>
  </PropertyGroup>
  <ItemGroup>
    <Reference Include="System" />
    <Reference Include="System.Data" />
    <Reference Include="System.Deployment" />
    <Reference Include="System.Xml" />
    <Reference Include="System.Core" />
    <Reference Include="System.Xml.Linq" />
    <Reference Include="System.Data.DataSetExtensions" />
    <Reference Include="System.Net.Http" />
  </ItemGroup>
  <ItemGroup>
    <Import Include="Microsoft.VisualBasic" />
    <Import Include="System" />
    <Import Include="System.Collections" />
    <Import Include="System.Collections.Generic" />
    <Import Include="System.Data" />
    <Import Include="System.Diagnostics" />
    <Import Include="System.Linq" />
    <Import Include="System.Xml.Linq" />
    <Import Include="System.Threading.Tasks" />
  </ItemGroup>
  <ItemGroup>
    <Compile Include="HelloWorld.vb" />
    <Compile Include="My Project\AssemblyInfo.vb" />
    <Compile Include="My Project\Application.Designer.vb">
      <AutoGen>True</AutoGen>
      <DependentUpon>Application.myapp</DependentUpon>
    </Compile>
    <Compile Include="My Project\Resources.Designer.vb">
      <AutoGen>True</AutoGen>
      <DesignTime>True</DesignTime>
      <DependentUpon>Resources.resx</DependentUpon>
    </Compile>
    <Compile Include="My Project\Settings.Designer.vb">
      <AutoGen>True</AutoGen>
      <DependentUpon>Settings.settings</DependentUpon>
      <DesignTimeSharedInput>True</DesignTimeSharedInput>
```

```
    </Compile>
  </ItemGroup>
  <ItemGroup>
    <EmbeddedResource Include="My Project\Resources.resx">
      <Generator>VbMyResourcesResXFileCodeGenerator</Generator>
      <LastGenOutput>Resources.Designer.vb</LastGenOutput>
      <CustomToolNamespace>My.Resources</CustomToolNamespace>
      <SubType>Designer</SubType>
    </EmbeddedResource>
  </ItemGroup>
  <ItemGroup>
    <None Include="My Project\Application.myapp">
      <Generator>MyApplicationCodeGenerator</Generator>
      <LastGenOutput>Application.Designer.vb</LastGenOutput>
    </None>
    <None Include="My Project\Settings.settings">
      <Generator>SettingsSingleFileGenerator</Generator>
      <CustomToolNamespace>My</CustomToolNamespace>
      <LastGenOutput>Settings.Designer.vb</LastGenOutput>
    </None>
    <None Include="App.config" />
  </ItemGroup>
  <Import Project="$(MSBuildToolsPath)\Microsoft.VisualBasic.targets" />
  <!-- To modify your build process, add your task inside one of the targets below and
 uncomment it.
       Other similar extension points exist, see Microsoft.Common.targets.
  <Target Name="BeforeBuild">
  </Target>
  <Target Name="AfterBuild">
  </Target>
  -->
</Project>
```

Application.Designer.vb (*(root directory name)*\VBHelloWorld\My Project 内)

```
'------------------------------------------------------------------------------
' <auto-generated>
'   This code was generated by a tool.
'   Runtime Version:4.0.30319.42000
'
'   Changes to this file may cause incorrect behavior and will be lost if
'   the code is regenerated.
' </auto-generated>
```

```
'------------------------------------------------------------------------------

Option Strict On
Option Explicit On
```

Application.myapp (*(root directory name)*\VBHelloWorld\My Project 内)

```
<?xml version="1.0" encoding="utf-8"?>
<MyApplicationData xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <MySubMain>false</MySubMain>
  <SingleInstance>false</SingleInstance>
  <ShutdownMode>0</ShutdownMode>
  <EnableVisualStyles>true</EnableVisualStyles>
  <AuthenticationMode>0</AuthenticationMode>
  <ApplicationType>2</ApplicationType>
  <SaveMySettingsOnExit>true</SaveMySettingsOnExit>
</MyApplicationData>
```

AssemblyInfo.vb (*(root directory name)*\VBHelloWorld\My Project 内)

```
Imports System
Imports System.Reflection
Imports System.Runtime.InteropServices

' General Information about an assembly is controlled through the following
' set of attributes. Change these attribute values to modify the information
' associated with an assembly.

' Review the values of the assembly attributes

<Assembly: AssemblyTitle("VBHelloWorld")>
<Assembly: AssemblyDescription("")>
<Assembly: AssemblyCompany("")>
<Assembly: AssemblyProduct("VBHelloWorld")>
<Assembly: AssemblyCopyright("Copyright ©  2017")>
<Assembly: AssemblyTrademark("")>

<Assembly: ComVisible(False)>

'The following GUID is for the ID of the typelib if this project is exposed to COM
<Assembly: Guid("137c362b-36ef-4c3e-84ab-f95082487a5a")>
```

```
' Version information for an assembly consists of the following four values:
'
' Major Version
' Minor Version
' Build Number
' Revision
'
' You can specify all the values or you can default the Build and Revision Numbers
' by using the '*' as shown below:
' <Assembly: AssemblyVersion("1.0.*")>

<Assembly: AssemblyVersion("1.0.0.0")>
<Assembly: AssemblyFileVersion("1.0.0.0")>
```

Resources.Designer.vb (*(root directory name)*\VBHelloWorld\My Project 内)

```
'------------------------------------------------------------------------------
' <auto-generated>
'   This code was generated by a tool.
'   Runtime Version:4.0.30319.42000
'
'   Changes to this file may cause incorrect behavior and will be lost if
'   the code is regenerated.
' </auto-generated>
'------------------------------------------------------------------------------

Option Strict On
Option Explicit On

Namespace My.Resources

  'This class was auto-generated by the StronglyTypedResourceBuilder
  'class via a tool like ResGen or Visual Studio.
  'To add or remove a member, edit your .ResX file then rerun ResGen
  'with the /str option, or rebuild your VS project.
  '''<summary>
  '''  A strongly-typed resource class, for looking up localized strings, etc.
  '''</summary>

 <Global.System.CodeDom.Compiler.GeneratedCodeAttribute("System.Resources.Tools.StronglyTypedRe
 "4.0.0.0"), _
  Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), _
  Global.System.Runtime.CompilerServices.CompilerGeneratedAttribute(), _
```

```
  Global.Microsoft.VisualBasic.HideModuleNameAttribute()> _
  Friend Module Resources

    Private resourceMan As Global.System.Resources.ResourceManager

    Private resourceCulture As Global.System.Globalization.CultureInfo

    '''<summary>
    '''  Returns the cached ResourceManager instance used by this class.
    '''</summary>

 <Global.System.ComponentModel.EditorBrowsableAttribute(Global.System.ComponentModel.EditorBrow
 _
    Friend ReadOnly Property ResourceManager() As
 Global.System.Resources.ResourceManager
      Get
        If Object.ReferenceEquals(resourceMan, Nothing) Then
          Dim temp As Global.System.Resources.ResourceManager = New
 Global.System.Resources.ResourceManager("VBHelloWorld.Resources",
 GetType(Resources).Assembly)
          resourceMan = temp
        End If
        Return resourceMan
      End Get
    End Property

    '''<summary>
    '''  Overrides the current thread's CurrentUICulture property for all
    '''  resource lookups using this strongly typed resource class.
    '''</summary>

 <Global.System.ComponentModel.EditorBrowsableAttribute(Global.System.ComponentModel.EditorBrow
 _
    Friend Property Culture() As Global.System.Globalization.CultureInfo
      Get
        Return resourceCulture
      End Get
      Set(ByVal value As Global.System.Globalization.CultureInfo)
        resourceCulture = value
      End Set
    End Property
  End Module
End Namespace
```

Resources.resx (*(root directory name)*\VBHelloWorld\My Project 内)

```
<?xml version="1.0" encoding="utf-8"?>
<root>
  <!--
    Microsoft ResX Schema

    Version 2.0

    The primary goals of this format is to allow a simple XML format
    that is mostly human readable. The generation and parsing of the
    various data types are done through the TypeConverter classes
    associated with the data types.

    Example:

    ... ado.net/XML headers & schema ...
    <resheader name="resmimetype">text/microsoft-resx</resheader>
    <resheader name="version">2.0</resheader>
    <resheader name="reader">System.Resources.ResXResourceReader,
System.Windows.Forms, ...</resheader>
    <resheader name="writer">System.Resources.ResXResourceWriter,
System.Windows.Forms, ...</resheader>
    <data name="Name1"><value>this is my long string</value><comment>this is a
comment</comment></data>
    <data name="Color1" type="System.Drawing.Color, System.Drawing">Blue</data>
    <data name="Bitmap1" mimetype="application/x-microsoft.net.object.binary.base64">
      <value>[base64 mime encoded serialized .NET Framework object]</value>
    </data>
    <data name="Icon1" type="System.Drawing.Icon, System.Drawing"
mimetype="application/x-microsoft.net.object.bytearray.base64">
      <value>[base64 mime encoded string representing a byte array form of the .NET
Framework object]</value>
      <comment>This is a comment</comment>
    </data>

    There are any number of "resheader" rows that contain simple
    name/value pairs.

    Each data row contains a name, and value. The row also contains a
    type or mimetype. Type corresponds to a .NET class that support
    text/value conversion through the TypeConverter architecture.
    Classes that don't support this are serialized and stored with the
    mimetype set.
```

```
    The mimetype is used for serialized objects, and tells the
    ResXResourceReader how to depersist the object. This is currently not
    extensible. For a given mimetype the value must be set accordingly:

    Note - application/x-microsoft.net.object.binary.base64 is the format
    that the ResXResourceWriter will generate, however the reader can
    read any of the formats listed below.

    mimetype: application/x-microsoft.net.object.binary.base64
    value   : The object must be serialized with
            : System.Serialization.Formatters.Binary.BinaryFormatter
            : and then encoded with base64 encoding.

    mimetype: application/x-microsoft.net.object.soap.base64
    value   : The object must be serialized with
            : System.Runtime.Serialization.Formatters.Soap.SoapFormatter
            : and then encoded with base64 encoding.

    mimetype: application/x-microsoft.net.object.bytearray.base64
    value   : The object must be serialized into a byte array
            : using a System.ComponentModel.TypeConverter
            : and then encoded with base64 encoding.
 -->
 <xsd:schema id="root" xmlns="" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:msdata="urn:schemas-microsoft-com:xml-msdata">
   <xsd:element name="root" msdata:IsDataSet="true">
     <xsd:complexType>
       <xsd:choice maxOccurs="unbounded">
         <xsd:element name="metadata">
           <xsd:complexType>
             <xsd:sequence>
               <xsd:element name="value" type="xsd:string" minOccurs="0" />
             </xsd:sequence>
             <xsd:attribute name="name" type="xsd:string" />
             <xsd:attribute name="type" type="xsd:string" />
             <xsd:attribute name="mimetype" type="xsd:string" />
           </xsd:complexType>
         </xsd:element>
         <xsd:element name="assembly">
           <xsd:complexType>
             <xsd:attribute name="alias" type="xsd:string" />
             <xsd:attribute name="name" type="xsd:string" />
           </xsd:complexType>
```

```
          </xsd:element>
          <xsd:element name="data">
            <xsd:complexType>
              <xsd:sequence>
                <xsd:element name="value" type="xsd:string" minOccurs="0"
 msdata:Ordinal="1" />
                <xsd:element name="comment" type="xsd:string" minOccurs="0"
 msdata:Ordinal="2" />
              </xsd:sequence>
              <xsd:attribute name="name" type="xsd:string" msdata:Ordinal="1" />
              <xsd:attribute name="type" type="xsd:string" msdata:Ordinal="3" />
              <xsd:attribute name="mimetype" type="xsd:string" msdata:Ordinal="4" />
            </xsd:complexType>
          </xsd:element>
          <xsd:element name="resheader">
            <xsd:complexType>
              <xsd:sequence>
                <xsd:element name="value" type="xsd:string" minOccurs="0"
 msdata:Ordinal="1" />
              </xsd:sequence>
              <xsd:attribute name="name" type="xsd:string" use="required" />
            </xsd:complexType>
          </xsd:element>
        </xsd:choice>
      </xsd:complexType>
    </xsd:element>
  </xsd:schema>
  <resheader name="resmimetype">
    <value>text/microsoft-resx</value>
  </resheader>
  <resheader name="version">
    <value>2.0</value>
  </resheader>
  <resheader name="reader">
    <value>System.Resources.ResXResourceReader, System.Windows.Forms, Version=2.0.0.0,
 Culture=neutral, PublicKeyToken=b77a5c561934e089</value>
  </resheader>
  <resheader name="writer">
    <value>System.Resources.ResXResourceWriter, System.Windows.Forms, Version=2.0.0.0,
 Culture=neutral, PublicKeyToken=b77a5c561934e089</value>
  </resheader>
</root>
```

Settings.Designer.vb (*(root directory name)*\VBHelloWorld\My Project 内)

```
'------------------------------------------------------------------------------
' <auto-generated>
'     This code was generated by a tool.
'     Runtime Version:4.0.30319.42000
'
'     Changes to this file may cause incorrect behavior and will be lost if
'     the code is regenerated.
' </auto-generated>
'------------------------------------------------------------------------------

Option Strict On
Option Explicit On

Namespace My

  <Global.System.Runtime.CompilerServices.CompilerGeneratedAttribute(), _

 Global.System.CodeDom.Compiler.GeneratedCodeAttribute("Microsoft.VisualStudio.Editors.Settings
 "11.0.0.0"), _

 Global.System.ComponentModel.EditorBrowsableAttribute(Global.System.ComponentModel.EditorBrows
 _
  Partial Friend NotInheritable Class MySettings
    Inherits Global.System.Configuration.ApplicationSettingsBase

    Private Shared defaultInstance As MySettings =
 CType(Global.System.Configuration.ApplicationSettingsBase.Synchronized(New
 MySettings), MySettings)

    #Region "My.Settings Auto-Save Functionality"
      #If _MyType = "WindowsForms" Then
        Private Shared addedHandler As Boolean

        Private Shared addedHandlerLockObject As New Object

        <Global.System.Diagnostics.DebuggerNonUserCodeAttribute(),
 Global.System.ComponentModel.EditorBrowsableAttribute(Global.System.ComponentModel.EditorBrows
 _
        Private Shared Sub AutoSaveSettings(ByVal sender As Global.System.Object, ByVal
 e As Global.System.EventArgs)
          If My.Application.SaveMySettingsOnExit Then
            My.Settings.Save()
```

```
          End If
        End Sub
      #End If
    #End Region

    Public Shared ReadOnly Property [Default]() As MySettings
      Get

        #If _MyType = "WindowsForms" Then
          If Not addedHandler Then
            SyncLock addedHandlerLockObject
              If Not addedHandler Then
                AddHandler My.Application.Shutdown, AddressOf AutoSaveSettings
                addedHandler = True
              End If
            End SyncLock
          End If
        #End If
        Return defaultInstance
      End Get
    End Property
  End Class
End Namespace

Namespace My

  <Global.Microsoft.VisualBasic.HideModuleNameAttribute(), _
  Global.System.Diagnostics.DebuggerNonUserCodeAttribute(), _
  Global.System.Runtime.CompilerServices.CompilerGeneratedAttribute()> _
  Friend Module MySettingsProperty

    <Global.System.ComponentModel.Design.HelpKeywordAttribute("My.Settings")> _
    Friend ReadOnly Property Settings() As Global.VBHelloWorld.My.MySettings
      Get
        Return Global.VBHelloWorld.My.MySettings.Default
      End Get
    End Property
  End Module
End Namespace
```

Settings.settings (*(root directory name)*\VBHelloWorld\My Project 内)

```
<?xml version='1.0' encoding='utf-8'?>
```

```
<SettingsFile xmlns="http://schemas.microsoft.com/VisualStudio/2004/01/settings"
 CurrentProfile="(Default)" UseMySettingsClassName="true">
  <Profiles>
    <Profile Name="(Default)" />
  </Profiles>
  <Settings />
</SettingsFile>
```

AWS CodeBuild でビルドを計画する

AWS CodeBuild を使用する前に、次の質問に答える必要があります。

1. ソースコードはどこに保存されていますか？ CodeBuild は現在、次のソースコードリポジトリプロバイダーからのビルドをサポートしています。ソースコードには、ビルド仕様 (buildspec) ファイルが含まれている必要があります。buildspec は、 がビルドの実行 CodeBuild に使用する YAML 形式のビルドコマンドと関連設定のコレクションです。buildspec は、ビルドプロジェクト定義で宣言できます。

リポジトリプロバイダ	必須	ドキュメント
CodeCommit	リポジトリ名。 (オプション) ソースコードに関連付けられているコミット ID。	AWS CodeCommit ユーザーガイドで以下のトピックを参照してください。 CodeCommit リポジトリを作成する でコミットを作成する CodeCommit
Amazon S3	バケット名を入力します。 ソースコードを含むビルド入力 ZIP ファイルに対応するオブジェクト名。 (オプション) ビルド入力 ZIP ファイルに関連付けられているバージョン ID。	「Amazon S3 入門ガイド」の以下のトピックを参照してください。 バケットの作成 バケットにオブジェクトを追加する

リポジトリプロバイダ	必須	ドキュメント
GitHub	リポジトリ名。 (オプション) ソースコードに関連付けられているコミット ID。	GitHub ヘルプウェブサイトでこのトピックを参照してください。 リポジトリを作成する
Bitbucket	リポジトリ名。 (オプション) ソースコードに関連付けられているコミット ID。	Bitbucket Cloud のドキュメントウェブサイトでこのトピックを参照してください。 リポジトリの作成

2. どのビルドコマンドを、どのような順番で実行する必要がありますか? デフォルトでは、 は指定したプロバイダーからビルド入力 CodeBuild をダウンロードし、指定したバケットにビルド出力をアップロードします。ビルド仕様を使用して、ダウンロードされたビルド入力を想定されるビルド出力に変換する方法を指示します。詳細については、「ビルド仕様 (buildspec) に関するリファレンス」を参照してください。
3. ビルドを実行するためにどのランタイムとツールが必要ですか? たとえば、Java、Ruby、Python、Node.js を構築していますか? ビルドでは、Maven、Ant または、Java、Ruby、Python のコンパイラが必要ですか? ビルドでは、Git、AWS CLI、またはその他のツールが必要ですか?

   CodeBuild は、Docker イメージを使用するビルド環境でビルドを実行します。これらの Docker イメージを CodeBuild でサポートされているリポジトリタイプに保存する必要があります。これには、 CodeBuild Docker イメージリポジトリ、Docker Hub、Amazon Elastic Container Registry (Amazon ECR) が含まれます。 CodeBuild Docker イメージリポジトリの詳細については、「」を参照してくださいが提供する Docker イメージ CodeBuild。
4. によって自動的に提供されないAWSリソースが必要です CodeBuildか？ そのようなリソースには、どのセキュリティポリシーが必要ですか? 例えば、 がこれらのリソースと CodeBuild 連携できるように CodeBuild サービスロールを変更する必要がある場合があります。

5. VPC CodeBuild を操作しますか？ その場合は、VPC ID、サブネット ID、および VPC 設定のセキュリティグループ ID が必要です。詳細については、「Amazon Virtual Private Cloud AWS CodeBuild で を使用する」を参照してください。

これらの質問に答えると、ビルドを正常に実行するために必要な設定とリソースがあるはずです。ビルドを実行するには、次の操作を実行できます。

- AWS CodeBuild コンソール、AWS CLI、または AWS SDK を使用します。詳細については、「ビルドの実行」を参照してください。
- でパイプラインを作成または識別しAWS CodePipeline、コードの自動テスト、ビルドの実行、またはその両方 CodeBuild を行うよう に指示するビルドまたはテストアクションを追加します。詳細については、「 CodePipeline で を使用する CodeBuild」を参照してください。

のビルド仕様リファレンス CodeBuild

このトピックでは、ビルド仕様 (buildspec) ファイルに関する重要なリファレンス情報を提供します。buildspec は、 がビルドの実行 CodeBuild に使用する YAML 形式のビルドコマンドと関連設定のコレクションです。buildspec をソースコードの一部として含めることも、ビルドプロジェクトの作成時に buildspec を定義することもできます。ビルド仕様の仕組みについては、「CodeBuild の仕組み」を参照してください。

トピック

- buildspec ファイル名とストレージの場所
- buildspec の構文
- buildspec の例
- buildspec のバージョン
- バッチビルドのビルド仕様 (buildspec) のリファレンス

buildspec ファイル名とストレージの場所

buildspec をソースコードの一部として含める場合、デフォルトの buildspec ファイルの名前は `buildspec.yml` で、ソースディレクトリのルートに配置する必要があります。

デフォルトの buildspec ファイルの名前と場所を変更することができます。たとえば、以下のことが可能です。

- 同じリポジトリ内の異なるビルドに、buildspec_debug.yml や buildspec_release.yml などの異なる buildspec ファイルを使用する。
- config/buildspec.yml など、ソースディレクトリのルート以外の場所や、S3 バケットに buildspec ファイルを保存する。S3 バケットは、ビルドプロジェクトと同じ AWS リージョンに存在する必要があります。ARN を使用して buildspec ファイルを指定します (例: arn:aws:s3:::*<my-codebuild-sample2>*/buildspec.yml) 。

buildspec ファイルの名前に関係なく、ビルドプロジェクトには 1 つの buildspec しか指定できません。

デフォルトの buildspec ファイルの名前、場所、またはその両方をオーバーライドするには、次のいずれかを実行します。

- または update-project コマンドを実行し AWS CLI create-project、 のbuildspec値を、組み込みの環境変数 の値を基準にした代替 buildspec ファイルのパスに設定しますCODEBUILD_SRC_DIR。 AWS SDKs の create projectオペレーションと同等の操作を行うこともできます。詳細については、「ビルドプロジェクトの作成」または「ビルドプロジェクトの設定の変更」を参照してください。
- コマンドを実行し AWS CLI start-build、 のbuildspecOverride値を、組み込みの環境変数 の値を基準にした代替 buildspec ファイルのパスに設定しますCODEBUILD_SRC_DIR。 AWS SDKs の start buildオペレーションと同等の操作を行うこともできます。詳細については、「ビルドの実行」を参照してください。
- AWS CloudFormation テンプレートSourceで、タイプ のリソースの BuildSpecプロパティAWS::CodeBuild::Projectを、組み込みの環境変数 の値に対する代替 buildspec ファイルのパスに設定しますCODEBUILD_SRC_DIR。詳細については、「 ユーザーガイド BuildSpec」のAWS CodeBuild 「プロジェクトソースの AWS CloudFormation プロパティ」を参照してください。

buildspec の構文

buildspec ファイルは YAML 形式で表現する必要があります。

YAML でサポートされていない文字または文字列がコマンドに含まれている場合は、そのコマンドを引用符 ("") で囲む必要があります。次のコマンドが引用符で囲まれているのは、YAML ではコロン (:) に続けてスペースを使用できないためです。コマンド内の引用符はエスケープ (\") されます。

```
"export PACKAGE_NAME=$(cat package.json | grep name | head -1 | awk -F: '{ print $2 }'
 | sed 's/[\",]//g')"
```

buildspec の構文は次のとおりです。

```
version: 0.2

run-as: Linux-user-name

env:
  shell: shell-tag
  variables:
    key: "value"
    key: "value"
  parameter-store:
    key: "value"
    key: "value"
  exported-variables:
    - variable
    - variable
  secrets-manager:
    key: secret-id:json-key:version-stage:version-id
  git-credential-helper: no | yes

proxy:
  upload-artifacts: no | yes
  logs: no | yes

batch:
  fast-fail: false | true
  # build-list:
  # build-matrix:
  # build-graph:

phases:
  install:
    run-as: Linux-user-name
    on-failure: ABORT | CONTINUE
    runtime-versions:
      runtime: version
      runtime: version
    commands:
      - command
```

```
      - command
    finally:
      - command
      - command
    # steps:
  pre_build:
    run-as: Linux-user-name
    on-failure: ABORT | CONTINUE
    commands:
      - command
      - command
    finally:
      - command
      - command
    # steps:
  build:
    run-as: Linux-user-name
    on-failure: ABORT | CONTINUE
    commands:
      - command
      - command
    finally:
      - command
      - command
    # steps:
  post_build:
    run-as: Linux-user-name
    on-failure: ABORT | CONTINUE
    commands:
      - command
      - command
    finally:
      - command
      - command
    # steps:
reports:
  report-group-name-or-arn:
    files:
      - location
      - location
    base-directory: location
    discard-paths: no | yes
    file-format: report-format
artifacts:
```

```
  files:
    - location
    - location
  name: artifact-name
  discard-paths: no | yes
  base-directory: location
  exclude-paths: excluded paths
  enable-symlinks: no | yes
  s3-prefix: prefix
  secondary-artifacts:
    artifactIdentifier:
      files:
        - location
        - location
      name: secondary-artifact-name
      discard-paths: no | yes
      base-directory: location
    artifactIdentifier:
      files:
        - location
        - location
      discard-paths: no | yes
      base-directory: location
cache:
  paths:
    - path
    - path
```

buildspec には、次のものが含まれています。

version

必要なマッピング。buildspec のバージョンを表します。`0.2` を使用することをお勧めします。

> **Note**
>
> バージョン 0.1 も引き続きサポートされていますが、可能な場合はバージョン 0.2 を使用することをお勧めします。詳細については、「buildspec のバージョン」を参照してください。

run-as

オプションのシーケンス。Linux ユーザーのみが使用できます。この buildspec ファイルでコマンドを実行する Linux ユーザーを指定します。`run-as` は、指定したユーザーに読み取りおよび実行の許可を付与します。buildspec ファイルの上で `run-as` を指定すると、すべてのコマンドにグローバルに適用されます。すべての buildspec ファイルコマンドのユーザーを指定しない場合、`phases` ブロックのいずれかで `run-as` を使用することによりフェーズでいずれかのコマンドを指定できます。`run-as` を指定しない場合、すべてのコマンドがルートユーザーとして実行されます。

env

オプションのシーケンス。1 つ以上のカスタム環境変数の情報を表します。

> **Note**
>
> 機密情報を保護するために、 CodeBuild ログでは以下が非表示になっています。
>
> - AWS アクセスキー IDs 詳細については、AWS Identity and Access Management ユーザーガイドの IAM ユーザーのアクセスキーの管理を参照してください。
> - パラメータストアを使用して指定された文字列。詳細については、「Amazon EC2 Systems Manager ユーザーガイド」の「Systems Manager パラメータストア」および「Systems Manager パラメータストアコンソールのチュートリアル」を参照してください。
> - を使用して指定された文字列 AWS Secrets Manager。詳細については、「キー管理」を参照してください。

env/shell

オプションのシーケンス。Linux または Windows オペレーティングシステムでサポートされるシェルを指定します。

Linux オペレーティングシステムで、サポートされているシェルタグは次のとおりです。

- `bash`
- `/bin/sh`

Windows オペレーティングシステムで、サポートされているシェルタグは次のとおりです。

- `powershell.exe`

- cmd.exe

env/variables

env を指定し、プレーンテキストでカスタム環境変数を定義する場合は必須です。##と#のスカラーのマッピングを含み、各マッピングはプレーンテキストで 1 つのカスタム環境変数を表します。##は、カスタム環境変数の名前で、#はその変数の値です。

> **Important**
>
> 機密の値は環境変数に保存しないことを強くお勧めします。環境変数は、 CodeBuild コンソールや などのツールを使用してプレーンテキストで表示できます AWS CLI。機密情報については、このセクションの後半で説明するように、parameter-store マッピングまたは secrets-manager マッピングを代わりに使用することをお勧めします。
> 既存の環境変数は、設定した環境変数によって置き換えられます。たとえば、Docker イメージに my_value の値を持つ MY_VAR という名前の環境変数が既に含まれていて、other_value の値を持つ MY_VAR という名前の環境変数を設定した場合、my_value が other_value に置き換えられます。同様に、Docker イメージに /usr/local/sbin:/usr/local/bin の値を持つ PATH という名前の環境変数が既に含まれていて、$PATH:/usr/share/ant/bin の値を持つ PATH という名前の環境変数を設定した場合、/usr/local/sbin:/usr/local/bin はリテラル値 $PATH:/usr/share/ant/bin に置き換えられます。
> CODEBUILD_ で始まる名前の環境変数は設定しないでください。このプレフィックスは内部使用のために予約されています。
> 同じ名前の環境変数が複数の場所で定義されている場合は、その値は次のように決定されます。
>
> - ビルド開始オペレーション呼び出しの値が最も優先順位が高くなります。ビルドの作成時に環境変数を追加または上書きできます。詳細については、「AWS CodeBuild でのビルドの実行」を参照してください。
> - ビルドプロジェクト定義の値が次に優先されます。プロジェクトを作成または編集するときに、プロジェクトレベルで環境変数を追加できます。詳細については、「 でのビルドプロジェクトの作成AWS CodeBuild」および「AWS CodeBuild でのビルドプロジェクトの設定の変更」を参照してください。
> - ビルド仕様宣言の値の優先順位が最も低くなります。

env/parameter-store

env が指定されていて、Amazon EC2 Systems Manager パラメータストアに保存されているカスタム環境変数を取得する場合は必須です。##と#のマッピングを含み、各マッピングは単一のカスタム環境変数を表し、Amazon EC2 Systems Manager パラメータストアに保存されます。##は、後でビルドコマンドで使用するこのカスタム環境変数を参照する名前で、#は Amazon EC2 Systems Manager パラメータストアに保存されているカスタム環境変数の名前です。重要な値を保存するには、Amazon EC2 Systems Manager ユーザーガイドの「Systems Manager パラメータストア」および「チュートリアル: String パラメータの作成とテスト (コンソール)」を参照してください。

> **Important**
>
> CodeBuild が Amazon EC2 Systems Manager パラメータストアに保存されているカスタム環境変数を取得できるようにするには、 `ssm:GetParameters`アクションを CodeBuild サービスロールに追加する必要があります。詳細については、「CodeBuild サービスロールの作成」を参照してください。
>
> Amazon EC2 Systems Manager パラメータストアから取得する環境変数は、既存の環境変数を置き換えます。たとえば、Docker イメージに `MY_VAR` という名前で値が `my_value` の環境変数がすでに含まれていて、`MY_VAR` という名前で値が `other_value` の環境変数を取得した場合、`my_value` は `other_value` に置き換えられます。同様に、Docker イメージに `/usr/local/sbin:/usr/local/bin` という名前で値が `PATH` の環境変数がすでに含まれていて、`$PATH:/usr/share/ant/bin` という名前で値が `PATH` の環境変数を取得した場合、`/usr/local/sbin:/usr/local/bin` はリテラル値 `$PATH:/usr/share/ant/bin` に置き換えられます。
>
> `CODEBUILD_` で始まる名前の環境変数は保存しないでください。このプレフィックスは内部使用のために予約されています。
>
> 同じ名前の環境変数が複数の場所で定義されている場合は、その値は次のように決定されます。
>
> - ビルド開始オペレーション呼び出しの値が最も優先順位が高くなります。ビルドの作成時に環境変数を追加または上書きできます。詳細については、「AWS CodeBuild でのビルドの実行」を参照してください。
> - ビルドプロジェクト定義の値が次に優先されます。プロジェクトを作成または編集するときに、プロジェクトレベルで環境変数を追加できます。詳細については、「 でのビルドプロジェクトの作成AWS CodeBuild」および「AWS CodeBuild でのビルドプロジェクトの設定の変更」を参照してください。

- ビルド仕様宣言の値の優先順位が最も低くなります。

env/secrets-manager

に保存されているカスタム環境変数を取得する場合に必要です AWS Secrets Manager。次のパターンを使用して、Secrets Manager `reference-key` を指定します。

<key>: *<secret-id>*:*<json-key>*:*<version-stage>*:*<version-id>*

<key>

(必須) ローカル環境変数の名前。この名前を使用して、ビルド中に変数にアクセスします。

<secret-id>

(必須) シークレットの一意の識別子として機能する名前または Amazon リソースネーム (ARN) です。 AWS アカウントのシークレットにアクセスするには、シークレット名を指定します。別の AWS アカウントのシークレットにアクセスするには、シークレット ARN を指定します。

<json-key>

(オプション) 値を取得する Secrets Manager のキーと値のペアのキー名を指定します。を指定しない場合`json-key`、 はシークレットテキスト全体 CodeBuild を取得します。

<version-stage>

(オプション) バージョンに添付されているステージングラベルによって取得するシークレットのバージョンを指定します。ステージングラベルは、ローテーション処理中にさまざまなバージョンを追跡するために使用されます。`version-stage` を使用する場合は、`version-id` を指定しないでください。バージョンステージもバージョン ID も指定しない場合、デフォルトでは `AWSCURRENT` のバージョンステージ値でバージョンが取得されます。

<version-id>

(オプション) 使用したいシークレットのバージョンの固有 ID を指定します。`version-id` を指定した場合は、`version-stage` を指定しないでください。バージョンステージもバージョン ID も指定しない場合、デフォルトでは `AWSCURRENT` のバージョンステージ値でバージョンが取得されます。

次の例で、`TestSecret` は Secrets Manager に保存されているキーと値のペアの名前です。`TestSecret` のキーは `MY_SECRET_VAR` です。ビルド中に変数にアクセスするには、名前「`LOCAL_SECRET_VAR`」を使用します。

```
env:
  secrets-manager:
    LOCAL_SECRET_VAR: "TestSecret:MY_SECRET_VAR"
```

詳細については、AWS Secrets Managerユーザーガイドの「AWS Secrets Manager とは?」を参照してください。

env/exported-variables

オプションのマッピング。エクスポートする環境変数をリストするために使用します。エクスポートする各変数の名前を、exported-variables の別の行で指定します。エクスポートする変数は、ビルド中にコンテナで使用できる必要があります。エクスポートする変数は、環境変数にすることができます。

エクスポートされた環境変数は、 と組み合わせて使用され AWS CodePipeline 、現在のビルドステージからパイプラインの後続のステージに環境変数をエクスポートします。詳細については、AWS CodePipeline ユーザーガイドの変数の操作を参照してください。

ビルド中、変数の値は、install フェーズから開始して使用できます。これは、install フェーズの開始と post_build フェーズの終了の間に更新することができます。post_build フェーズが終了すると、エクスポートされた変数の値は変更できません。

> **Note**
>
> 以下はエクスポートできません:
>
> - ビルドプロジェクトで指定された Amazon EC2 Systems Manager Parameter Store シークレット
> - ビルドプロジェクトで指定された Secrets Manager のシークレット
> - AWS_ で始まる環境変数。

env/git-credential-helper

オプションのマッピング。が Git 認証情報ヘルパー CodeBuild を使用して Git 認証情報を提供するかどうかを示すために使用されます。 が使用されyesている場合は 。それ以外の場合は、no または指定なしです。詳細については、Git ウェブサイトの「gitcredentials」を参照してください。

> **Note**
>
> `git-credential-helper` は、パブリック Git リポジトリの Webhook によってトリガーされるビルドではサポートされません。

proxy

オプションのシーケンス。明示的なプロキシサーバーでビルドを実行する場合、設定を表すために使用されます。詳細については、「明示的なプロキシサーバーでの CodeBuild の実行」を参照してください。

proxy/upload-artifacts

オプションのマッピング。明示的なプロキシサーバーのビルドでアーティファクトをアップロードする場合は、`yes` に設定します。デフォルト: no。

proxy/logs

オプションのマッピング。明示的なプロキシサーバーでのビルドで CloudWatch ログを作成する`yes`場合は、 に設定します。デフォルトは no です。

phases

必要なシーケンス。ビルドの各フェーズで CodeBuild 実行されるコマンドを表します。

> **Note**
>
> buildspec バージョン 0.1 では、 はビルド環境のデフォルトシェルの個別のインスタンスで各コマンド CodeBuild を実行します。つまり、各コマンドは他のすべてのコマンドとは独立して実行されます。したがって、デフォルトでは、以前のコマンド (ディレクトリの変更や環境変数の設定など) の状態に依存する単一のコマンドを実行することはできません。この制限を回避するには、バージョン 0.2 を使用することをお勧めします。これにより、問題が解決されます。buildspec バージョン 0.1 を使用する必要がある場合は、「ビルド環境のシェルとコマンド」のアプローチをお勧めします。

phases/*/run-as

オプションのシーケンス。ビルドフェーズで使用し、そのコマンドを実行する Linux ユーザーを指定します。buildspec ファイルの上ですべてのコマンドに対して `run-as` もグローバルに指定されている場合、フェーズレベルのユーザーが優先されます。例えば、`run-as` がグローバルに User-1 を指定し、`run-as` ステートメントが `install` フェーズでのみ User-2 を指定した場合、buildspec ファイル内のすべてのコマンドは User-1 として実行されますが、`install` フェーズのコマンドは除きます (これらのコマンドは User-2 として実行されます)。

phases/*/on-failure

オプションのシーケンス。フェーズ中に障害が発生した場合に実行するアクションを指定します。これには、次のいずれかの値を指定できます。

- `ABORT` - ビルドを中止します。
- `CONTINUE` - 次のフェーズに進みます。

このプロパティを指定しない場合は、ビルドフェーズの移行 に示すように、失敗処理が遷移フェーズに続きます。

phases/*/finally

オプションのブロック。`finally` ブロックに指定したコマンドは、`commands` ブロックのコマンドの実行後に実行されます。`finally` ブロックに指定したコマンドは、`commands` ブロックのコマンドが失敗した場合でも実行されます。例えば、 `commands`ブロックに 3 つのコマンドが含まれていて、最初のコマンドが失敗した場合、 は残りの 2 CodeBuild つのコマンドをスキップし、 `finally`ブロック内のコマンドを実行します。`commands` ブロックと `finally` ブロックのすべてのコマンドが正常に実行されると、フェーズは成功します。フェーズのいずれかのコマンドが失敗すると、フェーズは失敗します。

許可されるビルドフェーズ名は次のとおりです。

phases/install

オプションのシーケンス。インストール中に CodeBuild 実行されるコマンドがある場合は、それを表します。`install` フェーズは、ビルド環境でのパッケージのインストールにのみ使用することをお勧めします。たとえば、このフェーズを使用して、Mocha や RSpec などのコードテストフレームワークをインストールすることができます。

phases/install/runtime-versions

オプションのシーケンス。ランタイムバージョンは、Ubuntu 標準イメージ 5.0 以降および Amazon Linux 2 標準イメージ 4.0 以降でサポートされています。指定した場合、少なくとも 1 つのランタイムをこのセクションに含める必要があります。特定のバージョン、メジャーバージョン、 の後に続く を使用してランタイムを指定.xし、 が CodeBuild最新のマイナーバージョンでそのメジャーバージョンを使用するか、最新のメジャーバージョンとマイナーバージョン (ruby: 3.2、nodejs: 18.x、 などjava: latest) latestを使用します。数値または環境変数を使用してランタイムを指定できます。例えば、Amazon Linux 2 標準イメージ 4.0 を使用している場合、次の例は、Java のバージョン 17、Python バージョン 3 の最新マイナーバージョン、および Ruby の環境変数内のバージョンをインストールすることを指定します。詳細については、「が提供する Docker イメージ CodeBuild」を参照してください。

```
phases:
  install:
    runtime-versions:
      java: corretto8
      python: 3.x
      ruby: "$MY_RUBY_VAR"
```

buildspec ファイルの runtime-versions セクションで 1 つ以上のランタイムを指定できます。ランタイムが別のランタイムに依存している場合は、依存しているランタイムを buildspec ファイルで指定することもできます。buildspec ファイルでランタイムを指定しない場合、 は使用するイメージで使用できるデフォルトのランタイム CodeBuild を選択します。1 つ以上のランタイムを指定した場合、 はそれらのランタイムのみ CodeBuild を使用します。依存ランタイムが指定されていない場合、 は依存ランタイムを選択 CodeBuild しようとします。

2 つの指定されたランタイムが競合する場合、ビルドは失敗します。たとえば、android: 29 と java: openjdk11 が矛盾するので、両方が指定されている場合は、ビルドは失敗します。

使用できるランタイムの詳細については、「使用可能なランタイム」を参照してください。

> **Note**
>
> runtime-versions セクションを指定して、Ubuntu 標準イメージ 2.0 以降や Amazon Linux 2 (AL2) 標準イメージ 1.0 以降以外のイメージを使用した場合は、ビル

ドで「Skipping install of runtimes. Runtime version selection is not supported by this build image」の警告が表示されます。

phases/install/commands

オプションのシーケンス。一連のスカラーが含まれ、各スカラーはインストール中に CodeBuild 実行される単一のコマンドを表し、リストされた順序で、最初から最後まで CodeBuild 各コマンドを実行します。

phases/pre_build

オプションのシーケンス。ビルドの前に CodeBuild 実行されるコマンドがある場合は、それを表します。たとえば、このフェーズを使用して Amazon ECR にサインインするか、npm の依存関係をインストールすることができます。

phases/pre_build/commands

`pre_build` が指定されている場合は必須のシーケンスです。一連のスカラーが含まれ、各スカラーはビルドの前に CodeBuild 実行される 1 つのコマンドを表します。 CodeBuild は、リストされた順序で、最初から最後まで各コマンドを 1 つずつ実行します。

phases/build

オプションのシーケンス。ビルド中に CodeBuild 実行されるコマンドがある場合は、それを表します。たとえば、このフェーズを使用して、Mocha、RSpec、または sbt を実行できます。

phases/build/commands

`build` が指定されている場合は必須です。一連のスカラーが含まれ、各スカラーは build. CodeBuild runs 中に CodeBuild 実行される 1 つのコマンドを、リストされた順序で最初から最後まで表します。

phases/post_build

オプションのシーケンス。ビルド後に CodeBuild 実行されるコマンドがある場合は、それを表します。たとえば、Maven を使用してビルドアーティファクトを JAR または WAR ファイルにパッケージ化するか、Docker イメージを Amazon ECR にプッシュすることができます。次に、Amazon SNS を介してビルド通知を送信できます。

phases/post_build/commands

post_build が指定されている場合は必須です。一連のスカラーが含まれます。各スカラーは、build. CodeBuild runs の後に実行される CodeBuild 1 つのコマンドを、リストされた順序で最初から最後まで表します。

レポート

report-group-name-or-arn

オプションのシーケンス。レポートの送信先のレポートグループを指定します。プロジェクトには、最大 5 つのレポートグループを含めることができます。既存のレポートグループの ARN、または新しいレポートグループの名前を指定します。名前を指定すると、 はプロジェクト名と形式で指定した名前を使用してレポートグループ CodeBuild を作成します<project-name>-<report-group-name>。レポートグループ名は、 などの buildspec の環境変数を使用して設定することもできます$REPORT_GROUP_NAME。詳細については、「Report group naming」を参照してください。

reports/<report-group>/files

必要なシーケンス。レポートによって生成されたテスト結果の生データを含む場所を表します。一連のスカラーが含まれ、各スカラーは、 が元のビルド場所、または設定されている場合は を基準にしたテストファイルを見つける CodeBuild ことができる個別の場所を表しますbase-directory。場所には次のものが含まれます。

- 1 つのファイル (例: my-test-report-file.json)。
- サブディレクトリ内の単一のファイル (*my-subdirectory*/my-test-report-file.json や *my-parent-subdirectory*/*my-subdirectory*/my-test-report-file.json など)。
- '**/*' はすべてのファイルを再帰的に表します。
- *my-subdirectory*/* は、*my-subdirectory* という名前のサブディレクトリ内のすべてのファイルを表します。
- *my-subdirectory*/**/* は、*my-subdirectory* という名前のサブディレクトリから再帰的にすべてのファイルを表します。

reports/<report-group>/file-format

オプションのマッピング。レポートファイル形式を表します。指定しない場合は、JUNITXML を使用します。この値は大文字と小文字が区別されません。想定される値は次のとおりです。

テストレポート

`CUCUMBERJSON`

Cucumber JSON

`JUNITXML`

JUnit XML

`NUNITXML`

NUnit XML

`NUNIT3XML`

NUnit 3 XML

`TESTNGXML`

TestNG XML

`VISUALSTUDIOTRX`

Visual Studio TRX

コードカバレッジレポート

`CLOVERXML`

クローバー XML

`COBERTURAXML`

Cobertura XML

`JACOCOXML`

JaCoCo XML

`SIMPLECOV`

SimpleCov JSON

> **Note**
>
> CodeBuild は、simplecov-json ではなく、simplecov によって生成された JSON コードカバレッジレポートを受け入れます。

reports/<report-group>/base-directory

オプションのマッピング。が生のテストファイルの場所を特定 CodeBuild するために使用する元のビルド場所を基準にした 1 つ以上の最上位ディレクトリを表します。

reports/<report-group>/discard-paths

オプション。レポートファイルのディレクトリを出力でフラット化するかどうかを指定します。これが指定されていない場合、または no を含む場合、レポートファイルはディレクトリ構造のまま出力されます。このディレクトリに yes が含まれている場合、すべてのテストファイルが同じ出力ディレクトリに配置されます。たとえば、テスト結果へのパスが com/myapp/mytests/TestResult.xml である場合、yes を指定すると、このファイルが /TestResult.xml に配置されます。

artifacts

オプションのシーケンス。 CodeBuild がビルド出力を見つけることができる場所と、 が S3 出力バケットにアップロードする CodeBuild 準備を行う方法に関する情報を表します。たとえば、Docker イメージを作成して Amazon ECR にプッシュしている場合、または、ソースコードでユニットテストを実行していてもビルドしていない場合、このシーケンスは必要ありません。

> **Note**
>
> Amazon S3 メタデータには、Amazon S3 x-amz-meta-codebuild-buildarn にアーティファクトを発行する CodeBuild ビルドbuildArnの を含む という名前の CodeBuild ヘッダーがあります。通知のソーストラッキングを許可し、アーティファクトの生成元であるビルドを参照するには、buildArn を追加します。

artifacts/files

必要なシーケンス。ビルド環境でのビルド出力アーティファクトを含む場所を表します。一連のスカラーが含まれ、各スカラーは、 が CodeBuild元のビルドの場所、または設定されている場合はベースディレクトリを基準にビルド出力アーティファクトを見つけることができる個別の場所を表します。場所には次のものが含まれます。

- 1 つのファイル (例: my-file.jar)。
- サブディレクトリ内の単一のファイル (*my-subdirectory*/my-file.jar や *my-parent-subdirectory*/*my-subdirectory*/my-file.jar など)。

- `'**/*'` はすべてのファイルを再帰的に表します。
- *my-subdirectory*`/*` は、*my-subdirectory* という名前のサブディレクトリ内のすべてのファイルを表します。
- *my-subdirectory*`/**/*` は、*my-subdirectory* という名前のサブディレクトリから再帰的にすべてのファイルを表します。

ビルド出力アーティファクトの場所を指定すると、ビルド環境で元のビルドの場所を見つける CodeBuild ことができます。ビルドアーティファクトの出力先の場所に、元のビルドの場所へのパスを追加する、または `./` などで場所を指定する必要はありません。この場所へのパスを知りたい場合は、ビルド中に `echo $CODEBUILD_SRC_DIR` などのコマンドを実行できます。各ビルド環境の場所は多少異なる場合があります。

artifacts/name

オプション名。ビルドアーティファクトの名前を指定します。この名前は、次のいずれかに該当する場合に使用されます。

- CodeBuild API を使用してビルドを作成し、プロジェクトの更新、プロジェクトの作成、またはビルドの開始時に `ProjectArtifacts` オブジェクトに `overrideArtifactName`フラグが設定されます。
- CodeBuild コンソールを使用してビルドを作成し、buildspec ファイルで名前を指定し、プロジェクトを作成または更新するときにセマンティックバージョニングを有効にするを選択します。詳細については、「ビルドプロジェクトの作成 (コンソール)」を参照してください。

ビルド時に計算される buildspec ファイルの名前を指定できます。buildspec ファイルで指定された名前は、Shell コマンド言語を使用します。たとえば、アーティファクト名に日付と時刻を追加して常に一意にできます。アーティファクト名を一意にすると、アーティファクトが上書きされるのを防ぐことができます。詳細については、「Shell コマンド言語」を参照してください。

- これは、アーティファクトが作成された日付が付加されたアーティファクト名の例です。

```
version: 0.2
phases:
  build:
    commands:
      - rspec HelloWorld_spec.rb
artifacts:
  files:
    - '**/*'
  name: myname-$(date +%Y-%m-%d)
```

- これは、 CodeBuild 環境変数を使用するアーティファクト名の例です。詳細については、「ビルド環境の環境変数」を参照してください。

```
version: 0.2
phases:
  build:
    commands:
      - rspec HelloWorld_spec.rb
artifacts:
  files:
    - '**/*'
  name: myname-$AWS_REGION
```

- これは、アーティファクトの作成日が追加された CodeBuild 環境変数を使用するアーティファクト名の例です。

```
version: 0.2
phases:
  build:
    commands:
      - rspec HelloWorld_spec.rb
artifacts:
  files:
    - '**/*'
  name: $AWS_REGION-$(date +%Y-%m-%d)
```

名前にパス情報を追加して、名前のアーチファクトが名前のパスに基づいてディレクトリに配置されるようにできます。この例では、ビルドアーティファクトは、builds/<build number>/my-artifacts の下に配置されます。

```
version: 0.2
phases:
  build:
    commands:
      - rspec HelloWorld_spec.rb
artifacts:
  files:
    - '**/*'
  name: builds/$CODEBUILD_BUILD_NUMBER/my-artifacts
```

artifacts/discard-paths

オプション。ビルドアーティファクトのディレクトリが出力でフラット化されるかどうかを指定します。これが指定されていない場合、または no を含む場合は、ディレクトリ構造はそのままで、ビルドアーティファクトが出力されます。このディレクトリに yes が含まれる場合、すべてのビルドアーティファクトが同じ出力ディレクトリに配置されます。たとえば、ビルド出力アーティファクト内のファイルへのパスが com/mycompany/app/HelloWorld.java である場合、yes を指定すると、このファイルが /HelloWorld.java に配置されます。

artifacts/base-directory

オプションのマッピング。がビルド出力アーティファクトに含めるファイルとサブディレクトリを決定 CodeBuild するために使用する、元のビルド場所を基準とした 1 つ以上の最上位ディレクトリを表します。有効な値を次に示します。

- 単一の最上位ディレクトリ (例:my-directory)。
- 'my-directory*' my-directory で始まる名前を持つすべての最上位ディレクトリを表します。

一致する最上位ディレクトリはビルド出力アーティファクトに含まれず、ファイルとサブディレクトリにのみ含まれます。

files および discard-paths を使用して、どのファイルとサブディレクトリを含めるかをさらに制限できます。たとえば、以下のようなディレクトリ構造があります。

```
.
### my-build-1
#   ### my-file-1.txt
### my-build-2
    ### my-file-2.txt
    ### my-subdirectory
        ### my-file-3.txt
```

また、次のような artifacts シーケンスがあります。

```
artifacts:
  files:
    - '*/my-file-3.txt'
  base-directory: my-build-2
```

次のサブディレクトリとファイルが、ビルド出力アーティファクトに含まれます。

```
.
### my-subdirectory
    ### my-file-3.txt
```

さらに、次のような artifacts シーケンスがあります。

```
artifacts:
  files:
    - '**/*'
  base-directory: 'my-build*'
  discard-paths: yes
```

次のファイルが、ビルド出力アーティファクトに含まれます。

```
.
### my-file-1.txt
### my-file-2.txt
### my-file-3.txt
```

artifacts/exclude-paths

オプションのマッピング。ビルドアーティファクトから除外 CodeBuild する を基準にbase-directoryした 1 つ以上のパスを表します。アスタリスク (*) 記号は、フォルダの境界を超えない、0 文字以上の名前の要素と一致します。二重アスタリスク (**) は、すべてのディレクトリをまたいで、0 文字以上の名前の要素と一致します。

除外パスの例は以下のとおりです。

- すべてのディレクトリからファイルを除外する場合: "**/*file-name*/**/*"
- すべてのドットフォルダを除外する場合: "**/.*/**/*"
- すべてのドットファイルを除外する場合: "**/.*"

artifacts/enable-symlinks

オプション。出力タイプが ZIP の場合、内部シンボリックリンクを ZIP ファイルに保持するかどうかを指定します。これに yes が含まれる場合、ソース内のすべての内部シンボリックリンクがアーティファクト ZIP ファイルに保持されます。

artifacts/s3-prefix

オプション。アーティファクトを Amazon S3 バケットに出力し、名前空間タイプが BUILD_ID の場合に使用するプレフィックスを指定します。使用した場合、バケット内の出力パスは「<s3-prefix>/<build-id>/<name>.zip」となります。

artifacts/secondary-artifacts

オプションのシーケンス。アーティファクト識別子とアーティファクト定義との間のマッピングとしての 1 つ以上のアーティファクト定義を表します。このブロック内の各アーティファクト識別子は、プロジェクトの secondaryArtifacts 属性で定義されたアーティファクトと一致する必要があります。各個別の定義は、上記の artifacts ブロックと同じ構文を持っています。

> **Note**
>
> 「artifacts/files」シーケンスは、セカンダリアーティファクトしか定義されていない場合でも、常に必要です。

たとえば、プロジェクトに次の構造があるとします。

```
{
  "name": "sample-project",
  "secondaryArtifacts": [
    {
      "type": "S3",
      "location": "<output-bucket1>",
      "artifactIdentifier": "artifact1",
      "name": "secondary-artifact-name-1"
    },
    {
      "type": "S3",
      "location": "<output-bucket2>",
      "artifactIdentifier": "artifact2",
      "name": "secondary-artifact-name-2"
    }
  ]
}
```

次に、buildspec は次のようになります。

```
version: 0.2
```

```
phases:
build:
  commands:
    - echo Building...
artifacts:
  files:
    - '**/*'
  secondary-artifacts:
    artifact1:
      files:
        - directory/file1
      name: secondary-artifact-name-1
    artifact2:
      files:
        - directory/file2
      name: secondary-artifact-name-2
```

cache

オプションのシーケンス。がキャッシュを S3 キャッシュバケットにアップロードするためのファイルを準備 CodeBuild できる場所に関する情報を表します。プロジェクトのキャッシュタイプが No Cache の場合、このシーケンスは不要です。

cache/paths

必要なシーケンス。キャッシュの場所を表します。一連のスカラーが含まれ、各スカラーは、 が元のビルドの場所、または設定されている場合はベースディレクトリに関連するビルド出力アーティファクトを見つける CodeBuild ことができる個別の場所を表します。場所には次のものが含まれます。

- 1 つのファイル (例: `my-file.jar`)。
- サブディレクトリ内の単一のファイル (*`my-subdirectory`*`/my-file.jar` や *`my-parent-subdirectory`*`/`*`my-subdirectory`*`/my-file.jar` など)。
- `'**/*'` はすべてのファイルを再帰的に表します。
- *`my-subdirectory`*`/*` は、*`my-subdirectory`* という名前のサブディレクトリ内のすべてのファイルを表します。
- *`my-subdirectory`*`/**/*` は、*`my-subdirectory`* という名前のサブディレクトリから再帰的にすべてのファイルを表します。

> **Important**
>
> buildspec 宣言は有効な YAML である必要があるため、buildspec 宣言のスペースは重要です。buildspec の宣言にあるスペースの数が無効な場合、すぐにビルドが失敗する可能性があります。YAML validator を使用して、buildspec の宣言が有効な YAML かどうかをテストできます。
>
> AWS CLIビルドプロジェクトを作成または更新するときに 、、または AWS SDKs を使用して buildspec を宣言する場合、buildspec は必要な空白と改行のエスケープ文字とともに YAML 形式で表される単一の文字列である必要があります。次のセクションに例があります。
>
> buildspec.yml ファイルの代わりに CodeBuild または AWS CodePipeline コンソールを使用する場合は、 `build`フェーズのコマンドのみを挿入できます。上記の構文を使用する代わりに、ビルドフェーズで実行するすべてのコマンドを 1 行に記載します。複数のコマンドについては、`&&` で各コマンドを区切ります (例: `mvn test && mvn package`)。
>
> buildspec.yml ファイルの代わりに CodeBuild または CodePipeline コンソールを使用して、ビルド環境のビルド出力アーティファクトの場所を指定できます。上記の構文を使用する代わりに、すべての場所を 1 行に記載します。複数の場所の場合は、各場所をコンマで区切ります (例: `buildspec.yml, target/my-app.jar`)。

buildspec の例

buildspec.yml ファイルの例を次に示します。

```
version: 0.2

env:
  variables:
    JAVA_HOME: "/usr/lib/jvm/java-8-openjdk-amd64"
  parameter-store:
    LOGIN_PASSWORD: /CodeBuild/dockerLoginPassword

phases:
  install:
    commands:
      - echo Entered the install phase...
      - apt-get update -y
      - apt-get install -y maven
    finally:
```

```
      - echo This always runs even if the update or install command fails
  pre_build:
    commands:
      - echo Entered the pre_build phase...
      - docker login -u User -p $LOGIN_PASSWORD
    finally:
      - echo This always runs even if the login command fails
  build:
    commands:
      - echo Entered the build phase...
      - echo Build started on `date`
      - mvn install
    finally:
      - echo This always runs even if the install command fails
  post_build:
    commands:
      - echo Entered the post_build phase...
      - echo Build completed on `date`

reports:
  arn:aws:codebuild:your-region:your-aws-account-id:report-group/report-group-name-1:
    files:
      - "**/*"
    base-directory: 'target/tests/reports'
    discard-paths: no
  reportGroupCucumberJson:
    files:
      - 'cucumber/target/cucumber-tests.xml'
    discard-paths: yes
    file-format: CUCUMBERJSON # default is JUNITXML
artifacts:
  files:
    - target/messageUtil-1.0.jar
  discard-paths: yes
  secondary-artifacts:
    artifact1:
      files:
        - target/artifact-1.0.jar
      discard-paths: yes
    artifact2:
      files:
        - target/artifact-2.0.jar
      discard-paths: yes
cache:
```

```
  paths:
    - '/root/.m2/**/*'
```

、、 AWS CLIまたは AWS SDKs で使用するための単一の文字列で表される前述の buildspec の例を次に示します。

```
"version: 0.2\n\nenv:\n  variables:\n    JAVA_HOME: \"/usr/lib/jvm/java-8-openjdk-
amd64\\"\n  parameter-store:\n    LOGIN_PASSWORD: /CodeBuild/dockerLoginPassword\n
 phases:\n\n  install:\n    commands:\n      - echo Entered the install phase...\n
 - apt-get update -y\n      - apt-get install -y maven\n    finally:\n      - echo This
 always runs even if the update or install command fails \n  pre_build:\n    commands:
\n      - echo Entered the pre_build phase...\n      - docker login -u User -p
 $LOGIN_PASSWORD\n    finally:\n      - echo This always runs even if the login command
 fails \n  build:\n    commands:\n      - echo Entered the build phase...\n      - echo
 Build started on `date`\n      - mvn install\n    finally:\n      - echo This always
 runs even if the install command fails\n  post_build:\n    commands:\n      - echo
 Entered the post_build phase...\n      - echo Build completed on `date`\n\n reports:
\n  reportGroupJunitXml:\n    files:\n      - \"**/*\"\n    base-directory: 'target/
tests/reports'\n    discard-paths: false\n  reportGroupCucumberJson:\n    files:\n
 - 'cucumber/target/cucumber-tests.xml'\n    file-format: CUCUMBERJSON\n\nartifacts:\n
 files:\n    - target/messageUtil-1.0.jar\n  discard-paths: yes\n  secondary-artifacts:
\n    artifact1:\n      files:\n       - target/messageUtil-1.0.jar\n      discard-
paths: yes\n    artifact2:\n      files:\n       - target/messageUtil-1.0.jar\n
 discard-paths: yes\n cache:\n  paths:\n    - '/root/.m2/**/*'"
```

CodeBuild または CodePipeline コンソールで使用する buildフェーズのコマンドの例を次に示します。

```
echo Build started on `date` && mvn install
```

これらの例では:

- JAVA_HOME のキーと /usr/lib/jvm/java-8-openjdk-amd64 の値を持つプレーンテキストのカスタム環境変数が設定されます。
- Amazon EC2 Systems Manager パラメータストアに保存した dockerLoginPassword というカスタム環境変数は、後で LOGIN_PASSWORD キーを使用してビルドコマンドで参照します。
- これらのビルドフェーズ名は変更できません。この例で実行されるコマンドは、apt-get update -y と apt-get install -y maven (Apache Maven をインストールする)、mvn install (ソースコードをコンパイル、テストして、ビルド出力アーティファクトにパッケージ化し、ビルド出力アーティファクトを内部リポジトリにインストールする)、docker

login (Amazon EC2 Systems Manager パラメータストアで設定したカスタム環境変数 dockerLoginPassword の値に対応するパスワードを使用して Docker へサインインする)、および いくつかの echo コマンドです。echo コマンドは、 がコマンド CodeBuild を実行する方法と、コマンドを実行する順序を示すためにここに含まれています。

- files はビルド出力場所にアップロードするファイルを表します。この例では、 は単一のファイル CodeBuild をアップロードしますmessageUtil-1.0.jar。messageUtil-1.0.jar ファイルは、ビルド環境の target という名の相対ディレクトリ内にあります。discard-paths: yes が指定されたため、messageUtil-1.0.jar は直接アップロードされます (中間の target ディレクトリにはアップロードされません)。ファイル名 messageUtil-1.0.jar および相対ディレクトリ名 target は、Apache Maven がこの例のビルド出力アーティファクトを作成して保存する方法にのみ基づいています。独自のシナリオでは、これらのファイル名とディレクトリは異なります。
- reportsは、ビルド中にレポートを生成する 2 つのレポートグループを表します。
  - arn:aws:codebuild:your-region:your-aws-account-id:report-group/report-group-name-1 は、レポートグループの ARN を指定します。テストフレームワークによって生成されたテスト結果は、target/tests/reports ディレクトリにあります。ファイル形式は JunitXml であり、パスはテスト結果を含むファイルから削除されません。
  - reportGroupCucumberJson は、新しいレポートグループを指定します。プロジェクトの名前が my-project の場合、ビルドの実行時に my-project-reportGroupCucumberJson という名前のレポートグループが作成されます。テストフレームワークによって生成されたテスト結果は、cucumber/target/cucumber-tests.xml にあります。テストファイルの形式は、CucumberJson で、テスト結果を含むファイルからパスが削除されます。

buildspec のバージョン

次の表に、buildspec のバージョンとバージョン間の変更を示します。

バージョン	変更
0.2	• environment_variables が env に名称変更されました。 • plaintext が variables に名称変更されました。 • artifacts の type プロパティは廃止されました。

バージョン	変更
	• バージョン 0.1 では、 はビルド環境のデフォルトシェルの個別のインスタンスで各ビルドコマンド AWS CodeBuild を実行します。バージョン 0.2 では、 はビルド環境のデフォルトシェルの同じインスタンスですべてのビルドコマンド CodeBuild を実行します。
0.1	これは、ビルド仕様形式の最初の定義です。

バッチビルドのビルド仕様 (buildspec) のリファレンス

このトピックでは、バッチビルドプロパティの ビルド仕様 (buildspec) リファレンスを示します。

batch

オプションのマッピング。プロジェクトのバッチビルド設定。

batch/fast-fail

オプション。1 つ以上のビルドタスクが失敗した場合のバッチビルドの動作を指定します。

`false`

デフォルト値。実行中のすべてのビルドが完了します。

`true`

ビルドタスクの 1 つが失敗すると、実行中のすべてのビルドが停止します。

デフォルトでは、ビルド仕様 (buildspec) ファイルで指定された、すべてのバッチビルドタスクは、envおよびphases を実行します。デフォルトのビルド設定をオーバーライドするには、「env」値または別の buildspec ファイルを「batch/*<batch-type>*/buildspec」パラメータで指定します。

「batch」プロパティの内容は、指定されたバッチビルドのタイプによって異なります。可能なバッチビルドタイプは次のとおりです。

- batch/build-graph

- batch/build-list
- batch/build-matrix

batch/build-graph

ビルドグラフを定義。ビルドグラフは、バッチ内の他のタスクに依存する一連のタスクを定義します。詳細については、「ビルドグラフ」を参照してください。

この要素には、ビルドタスクの配列が含まれます。各ビルドタスクには、以下のプロパティが含まれます。

ID

必須。タスクの識別子。

buildspec

オプション。このタスクに使用する buildspec ファイルのパスとファイル名。このパラメータを指定しないと、現在の buildspec ファイルが使用されます。

debug-session

オプション。セッションデバッグがこのバッチビルドで有効かどうかを示す、ブール値。セッションデバッグの詳細については、「セッションマネージャーで実行中のビルドを表示する」を参照してください。

`false`

セッションデバッグは無効です。

`true`

セッションデバッグは有効です。

依存

オプション。このタスクが依存するタスク識別子の配列。このタスクは、これらのタスクが完了するまで実行されません。

env

オプション。タスクのビルド環境がオーバーライドされます。次のプロパティが含まれています。

compute-type

タスクに使用するコンピューティングタイプの識別子。可能な値については、the section called “ビルド環境のコンピューティングモードおよびタイプ” の「computeType」を参照してください。

イメージ

タスクに使用するイメージの識別子。可能な値については、the section called “が提供する Docker イメージ CodeBuild” の「イメージ識別子」を参照してください。

privileged-mode

Docker コンテナ内の Docker デーモンを実行するかどうかを示すブール値。ビルドプロジェクトを使用して Docker イメージを構築する場合にのみ、`true` に設定します。そうしないと、Docker デーモンとやり取りしようとするビルドは失敗します。デフォルトの設定は、`false` です。

type

タスクに使用する環境タイプの識別子です。可能な値については、the section called “ビルド環境のコンピューティングモードおよびタイプ” の「環境タイプ」を参照してください。

variables

ビルド環境に存在する環境変数。詳細については、「env/variables」を参照してください。

ignore-failure

オプション。このビルドタスクの失敗を無視できるかどうかを示すブール値。

`false`

デフォルト値。このビルドタスクが失敗すると、バッチビルドが失敗します。

`true`

このビルドタスクが失敗した場合でも、バッチビルドが成功します。

ビルドグラフの buildspec エントリの例を次に示します。

```
batch:
  fast-fail: false
  build-graph:
    - identifier: build1
      env:
        variables:
```

```
          BUILD_ID: build1
      ignore-failure: false
    - identifier: build2
      buildspec: build2.yml
      env:
        variables:
          BUILD_ID: build2
      depend-on:
        - build1
    - identifier: build3
      env:
        variables:
          BUILD_ID: build3
      depend-on:
        - build2
```

batch/build-list

ビルドリストを定義。ビルドリストは、並行して実行されるタスクの数を定義するために使用されます。詳細については、「ビルドリスト」を参照してください。

この要素には、ビルドタスクの配列が含まれます。各ビルドタスクには、以下のプロパティが含まれます。

ID

必須。タスクの識別子。

buildspec

オプション。このタスクに使用する buildspec ファイルのパスとファイル名。このパラメータを指定しないと、現在の buildspec ファイルが使用されます。

debug-session

オプション。セッションデバッグがこのバッチビルドで有効かどうかを示す、ブール値。セッションデバッグの詳細については、「セッションマネージャーで実行中のビルドを表示する」を参照してください。

`false`

セッションデバッグは無効です。

`true`

セッションデバッグは有効です。

env

オプション。タスクのビルド環境がオーバーライドされます。次のプロパティが含まれています。

compute-type

タスクに使用するコンピューティングタイプの識別子。可能な値については、the section called “ビルド環境のコンピューティングモードおよびタイプ” の「computeType」を参照してください。

イメージ

タスクに使用するイメージの識別子。可能な値については、the section called “が提供する Docker イメージ CodeBuild” の「イメージ識別子」を参照してください。

privileged-mode

Docker コンテナ内の Docker デーモンを実行するかどうかを示すブール値。ビルドプロジェクトを使用して Docker イメージを構築する場合にのみ、`true` に設定します。そうしないと、Docker デーモンとやり取りしようとするビルドは失敗します。デフォルトの設定は、`false` です。

type

タスクに使用する環境タイプの識別子です。可能な値については、the section called “ビルド環境のコンピューティングモードおよびタイプ” の「環境タイプ」を参照してください。

variables

ビルド環境に存在する環境変数。詳細については、「env/variables」を参照してください。

ignore-failure

オプション。このビルドタスクの失敗を無視できるかどうかを示すブール値。

`false`

デフォルト値。このビルドタスクが失敗すると、バッチビルドが失敗します。

`true`

このビルドタスクが失敗しても、バッチビルドは成功します。

buildspec エントリの例を次に示します。

```
batch:
```

```
  fast-fail: false
  build-list:
    - identifier: build1
      env:
        variables:
          BUILD_ID: build1
      ignore-failure: false
    - identifier: build2
      buildspec: build2.yml
      env:
        variables:
          BUILD_ID: build2
      ignore-failure: true
```

batch/build-matrix

ビルドマトリックスを定義する。ビルドマトリックスは、並行して実行される異なる構成のタスクを定義します。CodeBuild は、設定可能な組み合わせごとに個別のビルドを作成します。詳細については、「ビルドマトリックス」を参照してください。

static

静的プロパティは、すべてのビルドタスクに適用されます。

ignore-failure

オプション。このビルドタスクの失敗を無視できるかどうかを示すブール値。

`false`

デフォルト値。このビルドタスクが失敗すると、バッチビルドが失敗します。

`true`

このビルドタスクが失敗しても、バッチビルドは成功します。

env

オプション。ビルド環境はすべてのタスクに対して上書きされます。

privileged-mode

Docker コンテナ内の Docker デーモンを実行するかどうかを示すブール値。ビルドプロジェクトを使用して Docker イメージを構築する場合にのみ、`true` に設定します。そうしないと、Docker デーモンとやり取りしようとするビルドは失敗します。デフォルトの設定は、`false` です。

type

タスクに使用する環境タイプの識別子です。可能な値については、the section called “ビルド環境のコンピューティングモードおよびタイプ” の「環境タイプ」を参照してください。

dynamic

動的プロパティはビルドマトリックスを定義します。

buildspec

オプション。これらのタスクに使用する buildspec ファイルのパスとファイル名を含む配列。このパラメータを指定しないと、現在の buildspec ファイルが使用されます。

env

オプション。これらのタスクでは、ビルド環境が上書きされます。

compute-type

これらのタスクに使用するコンピューティングタイプの識別子を含む配列。可能な値については、the section called “ビルド環境のコンピューティングモードおよびタイプ” の「computeType」を参照してください。

イメージ

これらのタスクに使用するイメージの識別子を含む配列。可能な値については、the section called “が提供する Docker イメージ CodeBuild” の「イメージ識別子」を参照してください。

変数

これらのタスクのビルド環境に存在する環境変数を含む配列。詳細については、「env/variables」を参照してください。

ビルドマトリックス「buildspec」のエントリの例を次に示します。

```
batch:
  build-matrix:
    static:
      ignore-failure: false
    dynamic:
      buildspec:
        - matrix1.yml
        - matrix2.yml
```

```
    env:
      variables:
        MY_VAR:
          - VALUE1
          - VALUE2
          - VALUE3
```

詳細については、「ビルドマトリックス」を参照してください。

AWS CodeBuild のビルド環境に関するリファレンス

ビルドを実行するために AWS CodeBuild を呼び出すときは、ビルド環境に関する情報を入力する必要があります。ビルド環境は、CodeBuild がビルドを実行するために使用するオペレーティングシステム、プログラミング言語ランタイム、およびツールの組み合わせを表します。ビルド環境の仕組みについては、「CodeBuild の仕組み」を参照してください。

ビルド環境には Docker イメージが含まれています。詳細については、Docker Docs ウェブサイトの Docker 用語集を参照してください。

ビルド環境について CodeBuild に情報を提供する場合は、サポートされているリポジトリタイプの Docker イメージの識別子を指定します。これには、CodeBuild Docker イメージリポジトリ、Docker Hub に公開されているイメージ、および AWS アカウントにアクセス権限がある Amazon Elastic Container Registry (Amazon ECR) リポジトリなどがあります。

- CodeBuild Docker イメージリポジトリに格納されている Docker イメージは、サービスで使用するために最適化されているため、使用することをお勧めします。詳細については、「が提供する Docker イメージ CodeBuild」を参照してください。
- Docker Hub に保存されて公開されている Docker イメージの識別子を取得するには、Docker Docs ウェブサイトの Searching for Repositories を参照してください。
- AWS アカウントの Amazon ECR リポジトリに保存されている Docker イメージの操作方法については、Amazon ECR のサンプル を参照してください。

Docker イメージ識別子に加えて、ビルド環境で使用する一連のコンピューティングリソースも指定します。詳細については、「ビルド環境のコンピューティングモードおよびタイプ」を参照してください。

トピック

- が提供する Docker イメージ CodeBuild

- ビルド環境のコンピューティングモードおよびタイプ
- ビルド環境のシェルとコマンド
- ビルド環境の環境変数
- ビルド環境のバックグラウンドタスク

が提供する Docker イメージ CodeBuild

サポートされているイメージは、 で利用可能なイメージの最新のメジャーバージョン CodeBuild であり、マイナーバージョンとパッチバージョンの更新で更新されます。 CodeBuild は、サポートされているイメージを使用してビルドをマシンの Amazon マシンイメージ (AMI) にキャッシュすることで、ビルドのプロビジョニング期間を最適化します。キャッシュしてビルドのプロビジョニング時間を最小限に抑える場合は、 など、より詳細なバージョンではなく、 CodeBuild コンソールの「イメージバージョン」セクションで、このランタイムバージョンの最新のイメージを常に使用してくださいaws/codebuild/amazonlinux2-x86_64-standard:4.0-1.0.0。

CodeBuild は Docker イメージのリストを頻繁に更新して最新のイメージを追加し、古いイメージを非推奨にします。最新のリストを取得するには、次のいずれかを実行します。

- CodeBuild コンソールの「ビルドプロジェクトの作成」ウィザードまたは「ビルドプロジェクトの編集」ページで、「環境イメージ」で「マネージドイメージ」を選択します。[オペレーティングシステム]、[ランタイム]、[ランタイムバージョン] の各ドロップダウンリストで適切な選択を行います。詳細については、「ビルドプロジェクトの作成 (コンソール)」または「ビルドプロジェクトの設定の変更 (コンソール)」を参照してください。
- で AWS CLI、 `list-curated-environment-images` コマンドを実行します。

```
aws codebuild list-curated-environment-images
```

- AWS SDKs、ターゲットプログラミング言語の `ListCuratedEnvironmentImages`オペレーションを呼び出します。詳細については、「AWS SDK とツールのリファレンス」を参照してください。

Windows Server Core 2019 プラットフォームの基本イメージは、以下のリージョンでのみ利用可能です。

- 米国東部 (バージニア北部)
- 米国東部 (オハイオ)

- 米国西部 (オレゴン)
- 欧州 (アイルランド)

EC2 コンピューティングイメージ

AWS CodeBuild は、 の EC2 コンピューティングで使用できる次の Docker イメージをサポートします CodeBuild。

プラットフォーム	イメージ識別子	定義
Amazon Linux 2	aws/codebuild/amazonlinux2-x86_64-standard:4.0	al2/standard/4.0
Amazon Linux 2023	aws/codebuild/amazonlinux2-x86_64-standard:5.0	al2/standard/5.0
Amazon Linux 2	aws/codebuild/amazonlinux2-x86_64-standard:corretto8	al2/standard/corretto8
Amazon Linux 2	aws/codebuild/amazonlinux2-x86_64-standard:corretto11	al2/standard/corretto11
Amazon Linux 2	aws/codebuild/amazonlinux2-aarch64-standard:2.0	al2/aarch64/standard/2.0
Amazon Linux 2023	aws/codebuild/amazonlinux2-aarch64-standard:3.0	al2/aarch64/standard/3.0
Ubuntu 20.04	aws/codebuild/standard:5.0	ubuntu/standard/5.0

プラットフォーム	イメージ識別子	定義
Ubuntu 22.04	aws/codebuild/standard:6.0	ubuntu/standard/6.0
Ubuntu 22.04	aws/codebuild/standard:7.0	ubuntu/standard/7.0
Windows Server Core 2019	aws/codebuild/windows-base:2019-1.0	該当なし
Windows Server Core 2019	aws/codebuild/windows-base:2019-2.0	該当なし
Windows Server Core 2019	aws/codebuild/windows-base:2019-3.0	該当なし
Windows Server Core 2022	aws/codebuild/windows-base:2022-1.0	該当なし

Lambda コンピューティングイメージ

AWS CodeBuild は、 で AWS Lambda コンピューティングに使用できる次の Docker イメージをサポートしています CodeBuild。

`aarch64` アーキテクチャ

プラットフォーム	イメージ識別子	定義
Amazon Linux 2	aws/codebuild/amazonlinux-aarch64-lambda-standard:dotnet6	al-lambda/aarch64/dotnet6
Amazon Linux 2023	aws/codebuild/amazonlinux-aarch64-lambda-standard:dotnet8	al-lambda/aarch64/dotnet8

プラットフォーム	イメージ識別子	定義
Amazon Linux 2	aws/codebuild/amazonlinux-aarch64-lambda-standard:go1.21	al-lambda/aarch64/go1.21
Amazon Linux 2	aws/codebuild/amazonlinux-aarch64-lambda-standard:corretto11	al-lambda/aarch64/corretto11
Amazon Linux 2	aws/codebuild/amazonlinux-aarch64-lambda-standard:corretto17	al-lambda/aarch64/corretto17
Amazon Linux 2023	aws/codebuild/amazonlinux-aarch64-lambda-standard:corretto21	al-lambda/aarch64/corretto21
Amazon Linux 2	aws/codebuild/amazonlinux-aarch64-lambda-standard:nodejs18	al-lambda/aarch64/nodejs18
Amazon Linux 2023	aws/codebuild/amazonlinux-aarch64-lambda-standard:nodejs20	al-lambda/aarch64/nodejs20
Amazon Linux 2	aws/codebuild/amazonlinux-aarch64-lambda-standard:python3.11	al-lambda/aarch64/python3.11

プラットフォーム	イメージ識別子	定義
Amazon Linux 2023	aws/codebuild/amazonlinux-aarch64-lambda-standard:python3.12	al-lambda/aarch64/python3.12
Amazon Linux 2	aws/codebuild/amazonlinux-aarch64-lambda-standard:ruby3.2	al-lambda/aarch64/ruby3.2

x86_64 アーキテクチャ

プラットフォーム	イメージ識別子	定義
Amazon Linux 2	aws/codebuild/amazonlinux-x86_64-lambda-standard:dotnet6	al-lambda/x86_64/dotnet6
Amazon Linux 2023	aws/codebuild/amazonlinux-x86_64-lambda-standard:dotnet8	al-lambda/x86_64/dotnet8
Amazon Linux 2	aws/codebuild/amazonlinux-x86_64-lambda-standard:go1.21	al-lambda/x86_64/go1.21
Amazon Linux 2	aws/codebuild/amazonlinux-x86_64-lambda-standard:corretto11	al-lambda/x86_64/corretto11
Amazon Linux 2	aws/codebuild/amazonlinux-x86_64-lam	al-lambda/x86_64/corretto17

プラットフォーム	イメージ識別子	定義
	bda-standard:corretto17	
Amazon Linux 2023	aws/codebuild/amazonlinux-x86_64-lambda-standard:corretto21	al-lambda/x86_64/corretto21
Amazon Linux 2	aws/codebuild/amazonlinux-x86_64-lambda-standard:nodejs18	al-lambda/x86_64/nodejs18
Amazon Linux 2023	aws/codebuild/amazonlinux-x86_64-lambda-standard:nodejs20	al-lambda/x86_64/nodejs20
Amazon Linux 2	aws/codebuild/amazonlinux-x86_64-lambda-standard:python3.11	al-lambda/x86_64/python3.11
Amazon Linux 2023	aws/codebuild/amazonlinux-x86_64-lambda-standard:python3.12	al-lambda/x86_64/python3.12
Amazon Linux 2	aws/codebuild/amazonlinux-x86_64-lambda-standard:ruby3.2	al-lambda/x86_64/ruby3.2

廃止イメージ

非推奨イメージは、 によってキャッシュまたは更新されなくなったイメージです CodeBuild。廃止イメージは、マイナーバージョンやパッチバージョンの更新の対象外となって、更新されなくなるため、使用すると安全ではない場合があります。 CodeBuild プロジェクトが古いイメージバージョンを使用するように設定されている場合、プロビジョニングプロセスがこの Docker イメージをダウンロードしてコンテナ化されたランタイム環境を作成するため、プロビジョニング時間と全体的なビルド時間が長くなる可能性があります。

CodeBuild は次の Docker イメージを廃止しました。これらのイメージは引き続き使用できますが、ビルドホストにキャッシュされないため、プロビジョニング時間が長くなります。

プラットフォーム	イメージ識別子	定義	廃止日
Amazon Linux 2	`aws/codebuild/amazonlinux2-x86_64-standard:3.0`	al2/standard/3.0	2023 年 5 月 9 日
Ubuntu 18.04	`aws/codebuild/standard:4.0`	ubuntu/standard/4.0	2023 年 3 月 31 日
Amazon Linux 2	`aws/codebuild/amazonlinux2-aarch64-standard:1.0`	al2/aarch64/standard/1.0	2023 年 3 月 31 日
Ubuntu 18.04	`aws/codebuild/standard:3.0`	ubuntu/standard/3.0	2022 年 6 月 30 日
Amazon Linux 2	`aws/codebuild/amazonlinux2-x86_64-standard:2.0`	al2/standard/2.0	2022 年 6 月 30 日

トピック

- 使用可能なランタイム

- ランタイムバージョン

使用可能なランタイム

buildspec ファイルの `runtime-versions` セクションで 1 つ以上のランタイムを指定できます。ランタイムが別のランタイムに依存している場合は、依存しているランタイムを buildspec ファイルで指定することもできます。buildspec ファイルでランタイムを指定しない場合、 は使用するイメージで使用できるデフォルトのランタイム CodeBuild を選択します。1 つ以上のランタイムを指定した場合、 はそれらのランタイムのみ CodeBuild を使用します。依存ランタイムが指定されていない場合、 は依存ランタイムを選択 CodeBuild しようとします。詳細については、「Specify runtime versions in the buildspec file」を参照してください。

トピック

- Linux イメージのランタイム
- Windows イメージのランタイム

Linux イメージのランタイム

次の表に、使用可能なランタイムと、それらをサポートする標準 Linux イメージを示します。

Ubuntu および Amazon Linux プラットフォームランタイム

ランタイム名	バージョン	イメージ
dotnet	3.1	Amazon Linux 2 AArch64 standard: 2.0 Ubuntu standard:5.0
	5.0	Ubuntu standard:5.0
	6.0	Amazon Linux 2 x86_64 Lambda standa dotnet6 Amazon Linux 2 AArch64 Lambda stan dotnet6 Amazon Linux 2 x86_64 standard: 4.0

ランタイム名	バージョン	イメージ
		Amazon Linux 2023 x86_64 standard: 5.0 Amazon Linux 2023 AArch64 standard: 3.0 Ubuntu standard:6.0 Ubuntu standard:7.0
	8.0	Amazon Linux 2023 x86_64 standard: 5.0 Amazon Linux 2023 AArch64 standard: 3.0 Ubuntu standard:7.0
golang	1.12	Amazon Linux 2 AArch64 standard: 2.0
	1.13	Amazon Linux 2 AArch64 standard: 2.0
	1.14	Amazon Linux 2 AArch64 standard: 2.0
	1.15	Ubuntu standard:5.0
	1.16	Ubuntu standard:5.0
	1.18	Amazon Linux 2 x86_64 standard: 4.0 Ubuntu standard:6.0

ランタイム名	バージョン	イメージ
	1.20	Amazon Linux 2023 x86_64 standard: 5.0 Amazon Linux 2023 AArch64 standard: 3.0 Ubuntu standard:7.0
	1.21	Amazon Linux 2 x86_64 Lambda stand go1.21 Amazon Linux 2 AArch64 Lambda stan go1.21 Amazon Linux 2023 x86_64 standard: 5.0 Amazon Linux 2023 AArch64 standard: 3.0 Ubuntu standard:7.0
	1.22	Amazon Linux 2023 x86_64 standard: 5.0 Ubuntu standard:7.0
java	corretto8	Amazon Linux 2 x86_64 standard: corretto8 Amazon Linux 2023 x86_64 standard: 5.0 Amazon Linux 2 AArch64 standard: 2.0 Ubuntu standard:5.0 Ubuntu standard:7.0

ランタイム名	バージョン	イメージ
	corretto11	Amazon Linux 2 x86_64 standard: corretto11 Amazon Linux 2 x86_64 Lambda standa corretto11 Amazon Linux 2023 x86_64 standard: 5.0 Amazon Linux 2 AArch64 Lambda stan corretto11 Amazon Linux 2 AArch64 standard: 2.0 Ubuntu standard:5.0 Ubuntu standard:7.0
	corretto17	Amazon Linux 2 x86_64 Lambda standa corretto17 Amazon Linux 2 AArch64 Lambda stan corretto17 Amazon Linux 2 x86_64 standard: 4.0 Amazon Linux 2023 x86_64 standard: 5.0 Amazon Linux 2023 AArch64 standard: 3.0 Ubuntu standard:6.0 Ubuntu standard:7.0

ランタイム名	バージョン	イメージ
	corretto21	Amazon Linux 2 x86_64 Lambda standa corretto21 Amazon Linux 2 AArch64 Lambda stan corretto21 Amazon Linux 2023 x86_64 standard: 5.0 Amazon Linux 2023 AArch64 standard: 3.0 Ubuntu standard:7.0
NodeJS	10	Amazon Linux 2 AArch64 standard: 2.0
	12	Amazon Linux 2 AArch64 standard: 2.0 Ubuntu standard:5.0
	14	Ubuntu standard:5.0
	16	Amazon Linux 2 x86_64 standard: 4.0 Ubuntu standard:6.0

ランタイム名	バージョン	イメージ
	18	Amazon Linux 2 x86_64 Lambda standa nodejs18 Amazon Linux 2 AArch64 Lambda stan nodejs18 Amazon Linux 2023 x86_64 standard: 5.0 Amazon Linux 2023 AArch64 standard: 3.0 Ubuntu standard:7.0
	20	Amazon Linux 2 x86_64 Lambda standa nodejs20 Amazon Linux 2 AArch64 Lambda stan nodejs20 Amazon Linux 2023 x86_64 standard: 5.0 Amazon Linux 2023 AArch64 standard: 3.0 Ubuntu standard:7.0
php	73	Amazon Linux 2 AArch64 standard: 2.0 Ubuntu standard:5.0
	7.4	Amazon Linux 2 AArch64 standard: 2.0 Ubuntu standard:5.0
	8.0	Ubuntu standard:5.0

ランタイム名	バージョン	イメージ
	8.1	Amazon Linux 2 x86_64 standard: 4.0 Amazon Linux 2023 AArch64 standard: 3.0 Ubuntu standard:6.0
	8.2	Amazon Linux 2023 x86_64 standard: 5.0 Amazon Linux 2023 AArch64 standard: 3.0 Ubuntu standard:7.0
	8.3	Amazon Linux 2023 x86_64 standard: 5.0 Amazon Linux 2023 AArch64 standard: 3.0 Ubuntu standard:7.0
python	3.7	Amazon Linux 2 AArch64 standard: 2.0 Ubuntu standard:5.0
	3.8	Amazon Linux 2 AArch64 standard: 2.0 Ubuntu standard:5.0

ランタイム名	バージョン	イメージ
	3.9	Amazon Linux 2 x86_64 standard: 4.0 Amazon Linux 2023 x86_64 standard: 5.0 Amazon Linux 2 AArch64 standard: 2.0 Ubuntu standard:5.0 Ubuntu standard:7.0
	3.10	Amazon Linux 2023 x86_64 standard: 5.0 Ubuntu standard:6.0 Ubuntu standard:7.0
	3.11	Amazon Linux 2 x86_64 Lambda standa python3.11 Amazon Linux 2 AArch64 Lambda stan python3.11 Amazon Linux 2023 x86_64 standard: 5.0 Amazon Linux 2023 AArch64 standard: 3.0 Ubuntu standard:7.0

ランタイム名	バージョン	イメージ
	3.12	Amazon Linux 2 x86_64 Lambda standa python3.12 Amazon Linux 2 AArch64 Lambda stan python3.12 Amazon Linux 2023 x86_64 standard: 5.0 Amazon Linux 2023 AArch64 standard: 3.0 Ubuntu standard:7.0
ruby	2.6	Amazon Linux 2 AArch64 standard: 2.0 Ubuntu standard:5.0
	2.7	Amazon Linux 2 AArch64 standard: 2.0 Ubuntu standard:5.0
	3.1	Amazon Linux 2 x86_64 standard: 4.0 Amazon Linux 2023 x86_64 standard: 5.0 Ubuntu standard:6.0 Ubuntu standard:7.0

ランタイム名	バージョン	イメージ
	3.2	Amazon Linux 2 x86_64 Lambda stand ruby3.2 Amazon Linux 2 AArch64 Lambda stan ruby3.2 Amazon Linux 2023 x86_64 standard: 5.0 Amazon Linux 2023 AArch64 standard: 3.0 Ubuntu standard:7.0
	3.3	Amazon Linux 2023 x86_64 standard: 5.0 Ubuntu standard:7.0

Windows イメージのランタイム

Windows Server Core 2019 のベースイメージには、以下のランタイムが含まれています。

Windows プラットフォームのランタイム

ランタイム名	Windows Server Core 2019 標準: 1.0 バージョン	Windows Server Core 2019 標準: 2.0 バージョン	Windows Server Core 2019 標準: 3.0 バージョン
dotnet	3.1 5.0	3.1 6.0 7.0	6.0 7.0 8.0
dotnet SDK	3.1 5.0	3.1 6.0	8.0

ランタイム名	Windows Server Core 2019 標準: 1.0 バージョン	Windows Server Core 2019 標準: 2.0 バージョン	Windows Server Core 2019 標準: 3.0 バージョン
		7.0	
golang	1.14	1.18	1.21
gradle	6.7	7.6	8.5
java	Corretto11	Corretto11 Corretto17	Corretto21
Maven (メイヴン)	3.6	3.8	3.9
nodejs	14.15	16.19	20.11
php	7.4	8.1	8.3
powershell	7.1	7.2	7.4
python	3.8	3.10	3.12
ruby	2.7	3.1	3.3

ランタイムバージョン

buildspec ファイルの `runtime-versions` セクションでランタイムを指定するときは、特定のバージョン、特定のメジャーバージョンと最新のマイナーバージョン、または最新バージョンを指定できます。次の表に、使用可能なランタイムとその指定方法を示します。すべてのランタイムバージョンが、すべてのイメージで使用できるわけではありません。ランタイムバージョンの選択は、カスタムイメージでもサポートされていません。詳細については、「使用可能なランタイム」を参照してください。プリインストールされたランタイムバージョンの代わりにカスタムランタイムバージョンをインストールして使用する場合は、「」を参照してくださいカスタムランタイムバージョン。

Ubuntu および Amazon Linux 2 プラットフォームランタイムバージョン

ランタイム名	バージョン	特定のバージョン	特定のメジャーバージョンと最新のマイナーバージョン	最新バージョン
android	28	`android: 28`	`android: 28.x`	`android: latest`
	29	`android: 29`	`android: 29.x`	
dotnet	3.1	`dotnet: 3.1`	`dotnet: 3.x`	`dotnet: latest`
	5.0	`dotnet: 5.0`	`dotnet: 5.x`	
	6.0	`dotnet: 6.0`	`dotnet: 6.x`	
	8.0	`dotnet: 8.0`	`dotnet: 8.x`	
golang	1.12	`golang: 1.12`	`golang: 1.x`	`golang: latest`
	1.13	`golang: 1.13`		
	1.14	`golang: 1.14`		
	1.15	`golang: 1.15`		
	1.16	`golang: 1.16`		
	1.18	`golang: 1.18`		
	1.20	`golang: 1.20`		
	1.21	`golang: 1.21`		
	1.22	`golang: 1.22`		
java	corretto8	`java: corretto`	`java: corretto.x`	`java: latest`
	corretto11	`java: corretto11`	`java: corretto1.x`	

ランタイム名	バージョン	特定のバージョン	特定のメジャーバージョンと最新のマイナーバージョン	最新バージョン
	corretto17	`java: corretto7`	`java: corretto7.x`	
	corretto21	`java: corretto1`	`java: corretto1.x`	
NodeJS	10	`nodejs: 10`	`nodejs: 10.x`	`nodejs: latest`
	12	`nodejs: 12`	`nodejs: 12.x`	
	14	`nodejs: 14`	`nodejs: 14.x`	
	16	`nodejs: 16`	`nodejs: 16.x`	
	18	`nodejs: 18`	`nodejs: 18.x`	
	20	`nodejs: 20`	`nodejs: 20.x`	
php	73	`php: 7.3`	`php: 7.x`	`php: latest`
	7.4	`php: 7.4`		
	8.0	`php: 8.0`	`php: 8.x`	
	8.1	`php: 8.1`		
	8.2	`php: 8.2`		
	8.3	`php: 8.3`		
python	37	`python: 3.7`	`python: 3.x`	`python: latest`
	3.8	`python: 3.8`		
	3.9	`python: 3.9`		

ランタイム名	バージョン	特定のバージョン	特定のメジャーバージョンと最新のマイナーバージョン	最新バージョン
ruby	3.10	python: 3.10	ruby: 2.x	ruby: latest
	3.11	python: 3.11		
	3.12	python: 3.12		
	2.6	ruby: 2.6		
	2.7	ruby: 2.7		
	3.1	ruby: 3.1	ruby: 3.x	
	3.2	ruby: 3.2		
	3.3	ruby: 3.3		

ビルドフェーズでは、installビルド仕様を使用して他のコンポーネント (、 AWS CLI Apache Maven、Apache Ant、Mocha、RSpec など) をインストールできます。詳細については、「buildspec の例」を参照してください。

カスタムランタイムバージョン

CodeBuildマネージドイメージでプリインストールされたランタイムバージョンを使用する代わりに、選択したカスタムバージョンをインストールして使用できます。次の表に、使用可能なカスタムランタイムとその指定方法を示します。

> **Note**
>
> カスタムランタイムバージョンの選択は、Ubuntu および Amazon Linux イメージでのみサポートされています。

カスタムランタイムバージョン

ランタイム名	Syntax	例
dotnet	*<major>*.*<minor>*.*<patch>*	5.0.408
golang	*<major>*.*<minor>*	1.19
	<major>.*<minor>*.*<patch>*	1.19.1
java	corretto*<major>*	corretto15
nodejs	*<major>*	14
	<major>.*<minor>*	14.21
	<major>.*<minor>*.*<patch>*	14.21.3
php	*<major>*.*<minor>*.*<patch>*	8.0.30
python	*<major>*	3
	<major>.*<minor>*	3.7
	<major>.*<minor>*.*<patch>*	3.7.16
ruby	*<major>*.*<minor>*.*<patch>*	3.0.6

カスタムランタイム buildspec の例

カスタムランタイムバージョンを指定する buildspec の例を次に示します。

```
version: 0.2
phases:
  install:
    runtime-versions:
      java: corretto15
      php: 8.0.30
      ruby: 3.0.6
      golang: 1.19
      python: 3.7
      nodejs: 14
```

```
      dotnet: 5.0.408
```

ビルド環境のコンピューティングモードおよびタイプ

では CodeBuild、 がビルドの実行 CodeBuild に使用するコンピューティングおよびランタイム環境イメージを指定できます。コンピューティングとは、 が管理および維持するコンピューティングエンジン (CPU、メモリ、オペレーティングシステム) を指します CodeBuild。ランタイム環境イメージは、選択したコンピュートプラットフォーム上で実行されるコンテナイメージで、ビルドで必要になる可能性があるその他のツール (AWS CLIなど) が含まれています。

トピック

コンピューティングモードについて

CodeBuild には、次のコンピューティングモードがあります。

- EC2
- AWS Lambda

EC2 は、ビルド中の柔軟性が最適化され、起動速度が最適化されます。 AWS Lambda は起動のレイテンシーが低いため、ビルドの高速化 AWS Lambda をサポートします。 AWS Lambda また、自動的にスケールするため、ビルドはキューでの実行を待つことはありません。詳細については、「での AWS Lambda コンピューティングの使用 AWS CodeBuild」を参照してください。

環境タイプについて

AWS CodeBuild は、EC2 コンピューティングモード用の次の使用可能なメモリ、vCPUsおよびディスクスペースを備えたビルド環境を提供します。

コンピューティングタイプ	環境 computeType 値	環境タイプ値	メモリ	vCPU	ディスク容量
ARM Small	BUILD_GENERAL1_SMALL	ARM_CONTAINER	4 GB	2	50 GB
ARM large	BUILD_GENERAL1_LARGE	ARM_CONTAINER	16 GB	8	50 GB
Linux Small [1]	BUILD_GENERAL1_SMALL	LINUX_CONTAINER	3 GB	2	64 GB
Linux Medium [1]	BUILD_GENERAL1_MEDIUM	LINUX_CONTAINER	7 GB	4	128 GB
Linux Large [1]	BUILD_GENERAL1_LARGE	LINUX_CONTAINER	15 GB	8	128 GB
Linux XLarge	BUILD_GENERAL1_XLARGE	LINUX_CONTAINER	70 GB	36	256 GB
Linux 2xlarge	BUILD_GENERAL1_2XLARGE	LINUX_CONTAINER	145 GB	72	824 GB (SSD)
Linux GPU Sma	BUILD_GENERAL1_SMALL	LINUX_GPU_CONTAINER	16 GB	4	220 GB

コンピューティングタイプ	環境 computeType 値	環境タイプ値	メモリ	vCPU	ディスク容量
Linux GPU large	BUILD_GENERAL1_LARGE	LINUX_GPU_CONTAINER	255 GB	32	50 GB
Windows mediu	BUILD_GENERAL1_MEDIUM	WINDOWS_SERVER_2019_CONTAINER	7 GB	4	128 GB
Windows large	BUILD_GENERAL1_LARGE	WINDOWS_SERVER_2019_CONTAINER	15 GB	8	128 GB

[1] 各イメージの最新バージョンがキャッシュされます。より具体的なバージョンを指定すると、 はキャッシュされたバージョンの代わりにそのバージョンを CodeBuild プロビジョニングします。これにより、ビルド時間が長くなることがあります。たとえば、キャッシュのメリットを得るには、`aws/codebuild/amazonlinux2-x86_64-standard:5.0` のような詳細バージョンではなく `aws/codebuild/amazonlinux2-x86_64-standard:5.0-1.0.0` を指定します。

AWS CodeBuild は、 AWS Lambda コンピューティングモード用に次の使用可能なメモリとディスクスペースを備えたビルド環境を提供します。

コンピューティングタイプ	環境 computeType 値	環境タイプ値	「メモリ」	ディスク容量
ARM Lambda 1GB	BUILD_LAMBDA_1GB	ARM_LAMBDA_CONTAINER	1 GB	10 GB
ARM Lambda 2GB	BUILD_LAMBDA_2GB	ARM_LAMBDA_CONTAINER	2 GB	10 GB

コンピューティングタイプ	環境 computeType 値	環境タイプ値	「メモリ」	ディスク容量
ARM Lambda 4GB	BUILD_LAMBDA_4GB	ARM_LAMBDA_CONTAINER	4 GB	10 GB
ARM Lambda 8GB	BUILD_LAMBDA_8GB	ARM_LAMBDA_CONTAINER	8 GB	10 GB
ARM Lambda 10GI	BUILD_LAMBDA_10GB	ARM_LAMBDA_CONTAINER	10 GB	10 GB
Linux Lambda 1GE	BUILD_LAMBDA_1GB	LINUX_LAMBDA_CONTAINER	1 GB	10 GB
Linux Lambda 2GE	BUILD_LAMBDA_2GB	LINUX_LAMBDA_CONTAINER	2 GB	10 GB
Linux Lambda 4GE	BUILD_LAMBDA_4GB	LINUX_LAMBDA_CONTAINER	4 GB	10 GB
Linux Lambda 8GE	BUILD_LAMBDA_8GB	LINUX_LAMBDA_CONTAINER	8 GB	10 GB
Linux Lambda 10G	BUILD_LAMBDA_10GB	LINUX_LAMBDA_CONTAINER	10 GB	10 GB

他の環境タイプを使用する場合は、キャッシュされたイメージを使用してビルド時間を短縮することをお勧めします。

各ビルド環境にリストされているディスク容量は、CODEBUILD_SRC_DIR 環境変数で指定されたディレクトリでのみ使用できます。

コンピューティングタイプを選択するには:

- CodeBuild コンソールの「ビルドプロジェクトの作成」ウィザードまたは「ビルドプロジェクトの編集」ページの「環境」で「追加設定」を展開し、「コンピューティングタイプ」からいずれかのオプションを選択します。詳細については、ビルドプロジェクトの作成 (コンソール)またはビルドプロジェクトの設定の変更 (コンソール)を参照してください。
- には AWS CLI、 environment オブジェクトcomputeTypeの値を指定して、 create-projectまたは update-project コマンドを実行します。詳細については、ビルドプロジェクトの作成 (AWS CLI)またはビルドプロジェクトの設定の変更 (AWS CLI)を参照してください。
- AWS SDKs、ターゲットプログラミング言語の CreateProjectまたは UpdateProjectオペレーションと同等のコマンドを呼び出し、 environment オブジェクトと同等のcomputeType値を指定します。詳細については、「AWS SDK とツールのリファレンス」を参照してください。

一部の環境タイプとリージョン可用性には制限があります。

- コンピューティングタイプ Linux GPU Small (LINUX_GPU_CONTAINER) は、次のリージョンのみで利用可能です。
  - 米国東部 (バージニア北部)
  - 米国西部 (オレゴン)
  - アジアパシフィック (東京)
  - カナダ (中部)
  - 欧州 (フランクフルト)
  - 欧州 (アイルランド)
  - 欧州 (ロンドン)
- コンピューティングタイプ Linux GPU Large (LINUX_GPU_CONTAINER) は、次のリージョンのみで利用可能です。
  - 米国東部 (オハイオ)
  - 米国東部 (バージニア北部)
  - 米国西部 (オレゴン)
  - アジアパシフィック (ソウル)
  - アジアパシフィック (シンガポール)

- アジアパシフィック (シドニー)
- アジアパシフィック (東京)
- カナダ (中部)
- 中国 (北京)
- 中国 (寧夏)
- 欧州 (フランクフルト)
- 欧州 (アイルランド)
- 欧州 (ロンドン)

- 環境タイプ「ARM_CONTAINER」は、次のリージョンのみで利用可能です。
  - 米国東部 (オハイオ)
  - 米国東部 (バージニア北部)
  - 米国西部 (北カリフォルニア)
  - 米国西部 (オレゴン)
  - アジアパシフィック (香港)
  - アジアパシフィック (ジャカルタ)
  - アジアパシフィック (ハイデラバード)
  - アジアパシフィック (ムンバイ)
  - アジアパシフィック (大阪)
  - アジアパシフィック (ソウル)
  - アジアパシフィック (シンガポール)
  - アジアパシフィック (シドニー)
  - アジアパシフィック (東京)
  - カナダ (中部)
  - 中国 (北京)
  - 中国 (寧夏)
  - 欧州 (フランクフルト)
  - 欧州 (アイルランド)
  - 欧州 (ロンドン)
  - 欧州 (ミラノ)
  - 欧州 (パリ)

- 欧州 (スペイン)
- 欧州 (ストックホルム)
- イスラエル (テルアビブ)
- 中東 (バーレーン)
- 中東 (アラブ首長国連邦)
- 南米 (サンパウロ)

- コンピューティングタイプ「BUILD_GENERAL1_2XLARGE」は、次のリージョンのみで利用可能です。
  - 米国東部 (オハイオ)
  - 米国東部 (バージニア北部)
  - 米国西部 (北カリフォルニア)
  - 米国西部 (オレゴン)
  - アジアパシフィック (ハイデラバード)
  - アジアパシフィック (香港)
  - アジアパシフィック (ジャカルタ)
  - アジアパシフィック (メルボルン)
  - アジアパシフィック (ムンバイ)
  - アジアパシフィック (ソウル)
  - アジアパシフィック (シンガポール)
  - アジアパシフィック (シドニー)
  - アジアパシフィック (東京)
  - カナダ (中部)
  - 中国 (北京)
  - 中国 (寧夏)
  - 欧州 (フランクフルト)
  - 欧州 (アイルランド)
  - 欧州 (ロンドン)
  - 欧州 (パリ)
  - 欧州 (スペイン)
  - 欧州 (ストックホルム)

  - 欧州 (チューリッヒ)
  - イスラエル (テルアビブ)
  - 中東 (バーレーン)
  - 中東 (アラブ首長国連邦)
  - 南米 (サンパウロ)
- コンピューティングモード AWS Lambda (`ARM_LAMBDA_CONTAINER` および `LINUX_LAMBDA_CONTAINER`) は、次のリージョンでのみ使用できます。
  - 米国東部 (バージニア北部)
  - 米国東部 (オハイオ)
  - 米国西部 (オレゴン)
  - アジアパシフィック (ムンバイ)
  - アジアパシフィック (シンガポール)
  - アジアパシフィック (シドニー)
  - アジアパシフィック (東京)
  - 欧州 (フランクフルト)
  - 欧州 (アイルランド)
  - 南米 (サンパウロ)

コンピューティングタイプ `BUILD_GENERAL1_2XLARGE` では、最大 100 GB までの圧縮されていない Docker イメージがサポートされています。

> **Note**
>
> カスタムビルド環境イメージの場合、 は、コンピューティングタイプに関係なく、Linux および Windows で最大 50 GB の非圧縮 Docker イメージ CodeBuild をサポートします。ビルドイメージのサイズを確認するには、Docker を使用して `docker images` *`REPOSITORY`*`:`*`TAG`* コマンドを実行します。

Amazon EFS を使用してビルドコンテナのより多くの領域にアクセスできます。詳細については、「の Amazon Elastic File System サンプル AWS CodeBuild」を参照してください。コンテナのディスク領域をビルド中に操作する場合は、ビルドを特権モードで実行している必要があります。

> **Note**
>
> デフォルトでは、Docker デーモンは VPC 以外のビルドで有効になっています。VPC ビルドに Docker コンテナを使用する場合は、Docker Docs ウェブサイトの「ランタイム特権と Linux 機能」を参照して、特権モードを有効にします。また、Windows は特権モードをサポートしていません。

ビルド環境のシェルとコマンド

ビルドのライフサイクル中にビルド環境で実行するための AWS CodeBuild の一連のコマンドを提供します (たとえば、ビルドの依存関係のインストール、ソースコードのテストおよびコンパイルなど)。これらのコマンドを指定する方法はいくつかあります。

- ビルド仕様ファイルを作成し、それをソースコードに組み込みます。このファイルでは、ビルドライフサイクルの各段階で実行するコマンドを指定します。詳細については、「のビルド仕様リファレンス CodeBuild」を参照してください。
- CodeBuild コンソールを使用してビルドプロジェクトを作成します。[ビルドコマンドの挿入] の [ビルドコマンド] に、[build] フェーズで実行するコマンドを入力します。詳細については、「ビルドプロジェクトの作成 (コンソール)」を参照してください。
- CodeBuild コンソールを使用してビルドプロジェクトの設定を変更します。[ビルドコマンドの挿入] の [ビルドコマンド] に、[build] フェーズで実行するコマンドを入力します。詳細については、「ビルドプロジェクトの設定の変更 (コンソール)」を参照してください。
- AWS CLI または AWS SDK を使用して、ビルドプロジェクトを作成するか、ビルドプロジェクトの設定を変更します。コマンドを使用して buildspec ファイルを含むソースコードを参照するか、buildspec ファイルと同等の内容を含む単一の文字列を指定します。詳細については、「ビルドプロジェクトの作成」または「ビルドプロジェクトの設定の変更」を参照してください。
- AWS CLI または AWS SDK を使用してビルドを開始し、buildspec ファイルを指定するか、buildspec ファイルと同等の内容を含む単一の文字列を指定します。詳細については、`buildspecOverride` にある ビルドの実行 値の説明を参照してください。

任意の Shell コマンド言語 (sh) のコマンドを指定できます。ビルド仕様バージョン 0.1 では、CodeBuild は各ビルド環境の各インスタンスで各シェルコマンドを実行します。つまり、各コマンドは他のすべてのコマンドとは独立して実行されます。したがって、デフォルトでは、以前のコマンド (ディレクトリの変更や環境変数の設定など) の状態に依存する単一のコマンドを実行すること

はできません。この制限を回避するには、バージョン 0.2 を使用することをお勧めします。これにより、問題が解決されます。バージョン 0.1 を使用する場合は、以下のアプローチをお勧めします。

- デフォルトシェルの単一のインスタンスで実行するコマンドを含むシェルスクリプトをソースコードに含めます。たとえば、`my-script.sh` という名前のファイルを、`cd MyDir; mkdir -p mySubDir; cd mySubDir; pwd;` などのコマンドを含むソースコードに含めます。次に、buildspec ファイルで `./my-script.sh` コマンドを指定します。
- buildspec ファイル (または フェーズに限ってはコンソールの [Build commandbuild] 設定) で、デフォルトシェルの単一のインスタンスで実行するすべてのコマンドが含まれている単一のコマンドを指定します (例: `cd MyDir && mkdir -p mySubDir && cd mySubDir && pwd`)。

CodeBuild でエラーが発生した場合は、デフォルトシェルの独自のインスタンスで単一のコマンドを実行するのに比べて、トラブルシューティングが難しくなる場合があります。

Windows Server Core イメージで実行されるコマンドには、Powershell シェルが使用されます。

ビルド環境の環境変数

AWS CodeBuild には、ビルドコマンドで使用できるいくつかの環境変数が用意されています。

AWS_DEFAULT_REGION

ビルドが実行されている AWS リージョン (例: `us-east-1`)。この環境変数は、AWS CLI で主に使用されます。

AWS_REGION

ビルドが実行されている AWS リージョン (例: `us-east-1`)。この環境変数は、AWS SDK で主に使用されます。

CODEBUILD_BATCH_BUILD_IDENTIFIER

バッチビルドでのビルドの識別子。これは、バッチの buildspec で指定されています。詳細については、「the section called “バッチのビルド仕様 (buildspec) に関するリファレンス”」を参照してください。

CODEBUILD_BUILD_ARN

ビルドの Amazon リソースネーム (ARN) (例:`arn:aws:codebuild:region-ID:account-ID:build/codebuild-demo-project:b1e6661e-e4f2-4156-9ab9-82a19EXAMPLE`)。

CODEBUILD_BUILD_ID

ビルドの CodeBuild ID (例: `codebuild-demo-project:b1e6661e-e4f2-4156-9ab9-82a19EXAMPLE`)。

CODEBUILD_BUILD_IMAGE

CodeBuild のビルドイメージ識別子 (例: `aws/codebuild/standard:2.0`)。

CODEBUILD_BUILD_NUMBER

プロジェクトの現在のビルド番号。

CODEBUILD_BUILD_SUCCEEDING

現在のビルドが成功かどうか。ビルドが失敗の場合は `0` に設定され、成功の場合は `1` に設定されます。

CODEBUILD_INITIATOR

ビルドを開始したエンティティ。CodePipeline でビルドが開始された場合は、パイプラインの名前を表します (例: `codepipeline/my-demo-pipeline`)。ユーザーがビルドを開始した場合は、ユーザーの名前を表します (例: `MyUserName`)。CodeBuild の Jenkins プラグインがビルドを開始した場合、これは文字列「`CodeBuild-Jenkins-Plugin`」です。

CODEBUILD_KMS_KEY_ID

CodeBuild がビルド出力アーティファクトを暗号化するために使用している AWS KMS キーの識別子 (例: `arn:aws:kms:`*`region-ID`*`:`*`account-ID`*`:key/`*`key-ID`* または `alias/`*`key-alias`*)。

CODEBUILD_LOG_PATH

ビルド用の CloudWatch Logs のログストリーム名。

CODEBUILD_PUBLIC_BUILD_URL

パブリックビルドのウェブサイトにある、このビルドのビルド結果の URL。この変数は、ビルドプロジェクトでパブリックビルドが有効になっている場合にのみ設定されます。詳細については、「AWS CodeBuild でのパブリックビルドプロジェクト」を参照してください。

CODEBUILD_RESOLVED_SOURCE_VERSION

ビルドのソースコードのバージョンの識別子。内容は、以下のようなソースコードリポジトリによって異なります。

CodeCommit、GitHub、GitHub Enterprise Server、Bitbucket

この変数には、コミット ID が含まれます。

CodePipeline

この変数には、CodePipeline によって提供されるソースのリビジョンが含まれます。

ソースがバージョニングが有効になっていない Amazon S3 バケットである場合など、CodePipeline がソースリビジョンを解決できない場合、この環境変数は設定されません。

Amazon S3

この変数は設定されていません。

該当する場合、CODEBUILD_RESOLVED_SOURCE_VERSION 変数は、フェーズ DOWNLOAD_SOURCE の後でのみ利用可能です。

CODEBUILD_SOURCE_REPO_URL

入力アーティファクトまたはソースコードリポジトリの URL。Amazon S3 では、これは s3:// の後にバケット名と入力アーティファクトへのパスが続きます。CodeCommit および GitHub の場合、これはリポジトリのクローン URL です。CodePipeline から生成されたビルドの場合、この環境変数は空の場合があります。

セカンダリソースの場合、セカンダリソースリポジトリの URL の環境変数は「CODEBUILD_SOURCE_REPO_URL_*<sourceIdentifier>*」です。「*<sourceIdentifier>*」は、作成するソース識別子です。

CODEBUILD_SOURCE_VERSION

値の形式は、ソースコードリポジトリによって異なります。

- Amazon S3 では、入力アーティファクトに関連付けられたバージョン ID です。
- CodeCommit では、ビルドするソースコードのバージョンに関連付けられたコミット ID またはブランチ名です。
- GitHub、GitHub Enterprise Server、Bitbucket の場合、ビルドするソースコードのバージョンに関連付けられたコミット ID、ブランチ名、またはタグ名です。

> **Note**
>
> Webhook プルリクエストイベントによりトリガーされた GitHub または GitHub Enterprise Server ビルドの場合、pr/*pull-request-number* です。

セカンダリソースの場合、セカンダリソースバージョンの環境変数は「CODEBUILD_SOURCE_VERSION_*<sourceIdentifier>*」です。「*<sourceIdentifier>*」は、作成するソース識別子です。詳細については、「複数の入力ソースと出力アーティファクトのサンプル」を参照してください。

CODEBUILD_SRC_DIR

CodeBuild がビルドに使用するディレクトリパス (例: /tmp/src123456789/src)。

セカンダリソースの場合、ディレクトリパスの環境変数は「CODEBUILD_SRC_DIR_*<sourceIdentifier>*」です。「*<sourceIdentifier>*」は作成するソース識別子です。詳細については、「複数の入力ソースと出力アーティファクトのサンプル」を参照してください。

CODEBUILD_START_TIME

Unix タイムスタンプとして指定されたビルドの開始時間 (ミリ秒単位)。

CODEBUILD_WEBHOOK_ACTOR_ACCOUNT_ID

Webhook イベントをトリガーしたユーザーのアカウント ID。

CODEBUILD_WEBHOOK_BASE_REF

現在のビルドをトリガーする Webhook イベントの基本参照名。プルリクエストでは、ブランチ参照を表します。

CODEBUILD_WEBHOOK_EVENT

現在のビルドをトリガーした Webhook イベント。

CODEBUILD_WEBHOOK_MERGE_COMMIT

ビルドに使用されるマージコミットの識別子。この変数は、Bitbucket プルリクエストがスカッシュ戦略とマージされ、プルリクエストブランチが閉じられたときに設定されます。この場合、元のプルリクエストコミットは存在しなくなるため、この環境変数には圧縮されたマージコミットの識別子が含まれます。

CODEBUILD_WEBHOOK_PREV_COMMIT

現在のビルドをトリガーする Webhook プッシュイベントの前の最新のコミットの ID。

CODEBUILD_WEBHOOK_HEAD_REF

現在のビルドをトリガーする Webhook イベントのヘッド参照名。ブランチ参照またはタグ参照を表します。

CODEBUILD_WEBHOOK_TRIGGER

ビルドをトリガーした Webhook イベントを表示します。この変数は、Webhook によってトリガーされるビルドにのみ使用できます。値は、GitHub、GitHub Enterprise Server、または Bitbucket から CodeBuild に送信されたペイロードから解析されます。値の形式は、ビルドをトリガーしたイベントのタイプによって異なります。

- プルリクエストによってトリガーされたビルドの場合、`pr/`*`pull-request-number`* です。
- 新しいブランチを作成するか、ブランチにコミットをプッシュすることでトリガーされたビルドの場合、`branch/`*`branch-name`* です。
- タグをリポジトリにプッシュすることでトリガーされたビルドの場合、`tag/`*`tag-name`* です。

HOME

この環境変数は常に「`/root`」に設定されます。

独自の環境変数を持つビルド環境を提供することもできます。詳細については、以下のトピックを参照してください。

- CodePipeline で を使用する CodeBuild
- ビルドプロジェクトの作成
- ビルドプロジェクトの設定の変更
- ビルドの実行
- ビルド仕様 (buildspec) に関するリファレンス

ビルド環境で使用できる環境変数を一覧表示するには、構築時に `printenv` コマンド (Linux ベースのビルド環境) または `"Get-ChildItem Env:"` (Windows ベースのビルド環境) を実行できます。前述のものを除いて、「`CODEBUILD_`」で始まる環境変数は、CodeBuild の内部使用のためのものです。それらはビルドコマンドで使用できません。

> **Important**
>
> 機密情報 (特に AWS アクセスキー ID) を保存する場合は、環境変数を使用しないことを強くお勧めします。環境変数は、CodeBuild コンソールや AWS CLI などのツールを使用してプレーンテキストで表示できます。
> 機密値は Amazon EC2 Systems Manager パラメータストアに保存後、ビルド仕様から取得することをお勧めします。重要な値を保存するには、Amazon EC2 Systems Manager ユーザーガイドの「Systems Manager パラメータストア」および「チュートリアル: String パラメータの作成とテスト (コンソール)」を参照してください。これらを取得するには、「`parameter-store`」の buildspec の構文 マッピングを参照してください。

ビルド環境のバックグラウンドタスク

ビルド環境でバックグラウンドタスクを実行できます。これを行うには、ビルドプロセスでシェルが終了される場合でも、buildspec で nohup コマンドを使用してバックグラウンドのタスクとしてコマンドを実行します。実行中のバックグラウンドタスクを強制終了するには、disown コマンドを使用します。

例:

- バックグラウンドプロセスを開始し、その後、完了するまで待機します。

```
|
nohup sleep 30 & echo $! > pidfile
…
wait $(cat pidfile)
```

- バックグラウンドプロセスを開始し、その後、完了するまで待機しません。

```
|
nohup sleep 30 & disown $!
```

- バックグラウンドプロセスを開始し、その後、強制終了します。

```
|
nohup sleep 30 & echo $! > pidfile
…
kill $(cat pidfile)
```

AWS CodeBuild エージェントを使用してビルドをローカルで実行

AWS CodeBuild エージェントを使用して、ローカルマシンで CodeBuild ビルドを実行できます。x86_64 および ARM プラットフォームで使用できるエージェントがあります。

通知にサブスクライブして、 エージェントの新しいバージョンがリリースされたときに通知を受信できます。

前提条件

開始する前に、以下を実行する必要があります。

- ローカルマシンで Git をインストールします。
- ローカルマシンで、Docker をインストールしてセットアップします。

ビルドイメージの設定方法

ビルドイメージを設定する必要があるのは、エージェントを初めて実行するとき、またはイメージが変更されたときだけです。

ビルドイメージの設定方法

1. キュレートされた Amazon Linux 2 イメージを使用する場合は、以下のコマンドを使用して、https://gallery.ecr.aws/codebuild/amazonlinux2-x86_64-standard にある CodeBuild パブリック Amazon ECR リポジトリからイメージをプルできます。

   ```
   $ docker pull public.ecr.aws/codebuild/amazonlinux2-x86_64-standard:4.0
   ```

   その代わりに別の Linux イメージを使用する場合は、以下のステップを実行してください。

   a. CodeBuild イメージレポジトリをクローンします。

      ```
      $ git clone https://github.com/aws/aws-codebuild-docker-images.git
      ```

   b. イメージディレクトリを変更します。この例では、aws/codebuild/standard:5.0 イメージを使用します。

      ```
      $ cd aws-codebuild-docker-images/ubuntu/standard/5.0
      ```

c. イメージを構築します。これには数分間かかります。

```
$ docker build -t aws/codebuild/standard:5.0 .
```

2. CodeBuild エージェントをダウンロードします。

エージェントの x86_64 バージョンをダウンロードするには、次のコマンドを実行します。

```
$ docker pull public.ecr.aws/codebuild/local-builds:latest
```

次のコマンドを使用して、ARM バージョンのエージェントをダウンロードしてインストールします。

```
$ docker pull public.ecr.aws/codebuild/local-builds:aarch64
```

3. CodeBuild エージェントは、https://gallery.ecr.aws/codebuild/local-builds から入手できます。

エージェントの x86_64 バージョンのセキュアハッシュアルゴリズム (SHA) 署名は次のとおりです。

```
sha256:fac17c6d6c3cb500f6e7975887de1e41d29a9e70a86d6f49f76a2beacfcf967e
```

エージェントの ARM バージョンの SHA 署名は次のとおりです。

```
sha256:57a5dfda63be50edce13dea16dcd5e73e8d8559029658ba08b793c9a7adc68c7
```

SHA を使用してエージェントのバージョンを識別できます。エージェントの SHA 署名を表示するには、次のコマンドを実行して、`RepoDigests` の下で SHA を探します。

```
$ docker inspect public.ecr.aws/codebuild/local-builds:latest
```

CodeBuild エージェントを実行する

CodeBuild エージェントを実行するには

1. ビルドプロジェクトソースを含むディレクトリに移動します。
2. codebuild.sh スクリプトをダウンロードします。

```
$ curl -O  https://raw.githubusercontent.com/aws/aws-codebuild-docker-images/
master/local_builds/codebuild_build.sh
$ chmod +x codebuild_build.sh
```

3. codebuild_build.sh スクリプトを実行し、コンテナイメージおよび出力ディレクトリを指定します。

   x86_64 ビルドを実行するには、次のコマンドを実行します。

   ```
   $ ./codebuild_build.sh -i <container-image> -a <output directory>
   ```

   ARM ビルドを開始するには、次のコマンドを実行します。

   ```
   $ ./codebuild_build.sh -i <container-image> -a <output directory> -l
    public.ecr.aws/codebuild/local-builds:aarch64
   ```

   <container-image> は、コンテナイメージの名前 (aws/codebuild/standard:5.0 または public.ecr.aws/codebuild/amazonlinux2-x86_64-standard:4.0 など) に置き換えてください。

   スクリプトはビルドイメージを起動し、現在のディレクトリにあるプロジェクトを使用してビルドを実行します。ビルドプロジェクトの場所を指定するには、-s *<build project directory>* オプションをスクリプトコマンドに追加します。

CodeBuild エージェントの新しいバージョンに関する通知の受信

Amazon SNS 通知にサブスクライブして、AWS CodeBuild エージェントの新しいバージョンがリリースされたときに通知を受信できます。

CodeBuild エージェントの通知にサブスクライブするには

1. https://console.aws.amazon.com/sns/v3/home で Amazon SNS コンソールを開きます。
2. ナビゲーションバーで、AWS リージョンを米国東部 (バージニア北部) に変更します (まだ選択していない場合)。サブスクライブする Amazon SNS 通知は、この AWS リージョンで作成されるため、このリージョンを選択する必要があります。
3. ナビゲーションペインで [Subscriptions] を選択します。
4. [Create subscription] を選択します。

5. [Create subscription] (サブスクリプションの作成) で、次の操作を行います。

   a. [Topic ARN] (トピック ARN) で、以下の Amazon リソースネーム (ARN) を使用します。

   ```
   arn:aws:sns:us-east-1:850632864840:AWS-CodeBuild-Local-Agent-Updates
   ```

   b. [プロトコル] で、[E メール] または [SMS] を選択します。

   c. [エンドポイント] で、通知を受信する場所 (E メールまたは SMS) を選択します。E メール、住所、または電話番号 (市外局番を含む) を入力します。

   d. [Create subscription] (サブスクリプションの作成) を選択します。

   e. [Email] (E メール) を選択した場合は、サブスクリプションの確認を求める E メールが届きます。E メールの指示に従ってサブスクリプションを完了します。

   通知が不要になった場合は、次の手順で受信登録を解除します。

CodeBuild エージェントの通知のサブスクリプションを解除するには

1. Amazon SNS コンソール (https://console.aws.amazon.com/sns/v3/home) を開きます。
2. ナビゲーションペインで [Subscriptions] (サブスクリプション) を選択します。
3. サブスクリプションを選択し、[Actions] (アクション) から [Delete subscriptions] (サブスクリプションの削除) を選択します。確認を求められたら [Delete] (削除) を選択します。

Amazon Virtual Private Cloud AWS CodeBuild で を使用する

通常、 AWS CodeBuild VPC 内のリソースにアクセスすることはできません。アクセスを有効にするには、 CodeBuild プロジェクト設定で追加の VPC 固有の設定情報を指定する必要があります。これには、VPC ID、VPC サブネット ID、および VPC セキュリティグループ ID が含まれます。これにより、VPC 対応のビルドは VPC 内のリソースにアクセスできます。Amazon VPC で VPC を設定する方法の詳細については、Amazon VPC ユーザーガイドを参照してください。

トピック

ユースケース

AWS CodeBuild ビルドからの VPC 接続により、次のことが可能になります。

- プライベートサブネット上に分離された Amazon RDS データベース内のデータに対して、ビルドから統合テストを実行する。
- テストから直接 Amazon ElastiCache クラスター内のデータをクエリします。
- Amazon EC2、Amazon ECS、または内部 Elastic Load Balancing を使用するサービスでホストされる内部ウェブサービスを操作する。
- Python 用 PyPI、Java 用 Maven、Node.js 用 npm など、セルフホスト型の内部アーティファクトリポジトリから依存関係を取得する。
- Amazon VPC エンドポイント経由でのみアクセスできるように設定された S3 バケット内のオブジェクトにアクセスする。

- 固定 IP アドレスを必要とする外部ウェブサービスを、サブネットに関連付けられた NAT ゲートウェイまたは NAT インスタンスの Elastic IP アドレスを使用してクエリする。

お客様のビルドは、VPC でホストされている任意のリソースにアクセスできます。

プロジェクトで Amazon VPC アクセスを許可する CodeBuild

以下の設定を VPC 設定に含めます。

- VPC ID で、 が CodeBuild 使用する VPC ID を選択します。
- サブネット では、 が使用するリソースを含む、またはリソースへのルートを持つ NAT 変換でプライベートサブネットを選択します CodeBuild。
- セキュリティグループ で、 CodeBuild が VPCs内のリソースへのアクセスを許可するために使用するセキュリティグループを選択します。

コンソールを使用してビルドプロジェクトを作成する方法については、「ビルドプロジェクトの作成 (コンソール)」を参照してください。 CodeBuild プロジェクトを作成または変更する場合、VPC で VPC ID、サブネット、セキュリティグループを選択します。

を使用してビルドプロジェクト AWS CLI を作成するには、「」を参照してくださいビルドプロジェクトの作成 (AWS CLI)。 AWS CLI で を使用している場合 CodeBuild、IAM ユーザーに代わってサービスとやり取り CodeBuild するために で使用されるサービスロールには、 ポリシーがアタッチされている必要があります。詳細については、「VPC ネットワークインターフェイスの作成に必要な AWS サービス CodeBuild へのアクセスを許可する」を参照してください。

vpcConfig オブジェクトには、*vpcId* 、*securityGroupIds*、サブネット が含まれている必要があります。

- *vpcId*: 必須。が CodeBuild 使用する VPC ID。リージョン内の Amazon VPC ID を一覧表示するには、次のコマンドを実行します。

```
aws ec2 describe-vpcs
```

- *subnets*: 必須。で使用されるリソースを含むサブネット IDs CodeBuild。この ID を取得するには、次のコマンドを実行します。

```
aws ec2 describe-subnets --filters "Name=vpc-id,Values=<vpc-id>" --region us-east-1
```

> **Note**
>
> us-east-1 は、実際のリージョンに置き換えます。

- *securityGroupIds*: 必須。VPCs 内のリソースへのアクセスを許可 CodeBuild するために で使用されるセキュリティグループ IDs。この ID を取得するには、次のコマンドを実行します。

```
aws ec2 describe-security-groups --filters "Name=vpc-id,Values=<vpc-id>" --region us-east-1
```

> **Note**
>
> us-east-1 は、実際のリージョンに置き換えます。

VPC のベストプラクティス

と連携するように VPC を設定する場合は、このチェックリストを使用します CodeBuild。

- パブリックおよびプライベートサブネットと NAT ゲートウェイを使用して VPC を設定します。NAT ゲートウェイはパブリックサブネットにある必要があります。詳細については、Amazon VPC ユーザーガイド の「パブリックサブネットとプライベートサブネットを持つ VPC (NAT)」を参照してください。

> **Important**
>
> がパブリックエンドポイント CodeBuild に到達できるように、VPC CodeBuild で使用する NAT ゲートウェイまたは NAT インスタンスが必要です (ビルドの実行時に CLI コマンドを実行する場合など)。 CodeBuild は、作成したネットワークインターフェイスへの Elastic IP アドレスの割り当てをサポートしていません。また、Amazon EC2 インスタンスの起動以外で作成されたネットワークインターフェイスでは、パブリック IP アドレスの自動割り当ては Amazon EC2 でサポートされていないため、NAT ゲートウェイまたは NAT インスタンスの代わりにインターネットゲートウェイを使用することはできません。

- VPC に複数のアベイラビリティーゾーンを含めます。

- セキュリティグループに、builds. CodeBuild does へのインバウンド (進入) トラフィックが許可されていないことを確認してください。アウトバウンドトラフィックに特定の要件はありませんが、GitHub や Amazon S3 など、ビルドに必要なインターネットリソースへのアクセスを許可する必要があります。

  詳細については、「Amazon VPC ユーザーガイド」の「セキュリティグループルール」を参照してください。
- ビルド用に別個のサブネットを設定します。
- VPC にアクセスするように CodeBuild プロジェクトを設定するときは、プライベートサブネットのみを選択します。

Amazon VPC で VPC を設定する方法の詳細については、Amazon VPC ユーザーガイドを参照してください。

AWS CloudFormation を使用して VPC 機能を使用するように CodeBuild VPC を設定する方法の詳細については、「」を参照してくださいAWS CloudFormation VPC テンプレート。

VPC 設定のトラブルシューティング

エラーメッセージに表示される情報を、問題の特定、診断、対処に役立てます。

一般的な CodeBuild VPC エラーのトラブルシューティングに役立つガイドラインを以下に示します。`Build does not have internet connectivity. Please check subnet network configuration`

1. インターネットゲートウェイが VPC にアタッチされていることを確認します。
2. パブリックサブネットのルートテーブルがインターネットゲートウェイを参照していることを確認します。
3. ネットワーク ACL がトラフィックのフローを許可していることを確認します。
4. セキュリティグループがトラフィックのフローを許可していることを確認します。
5. NAT ゲートウェイのトラブルシューティングを行います。
6. プライベートサブネットのルートテーブルが NAT ゲートウェイを参照していることを確認します。
7. IAM ユーザーに代わって サービスとやり取り CodeBuild するために が使用するサービスロールに、このポリシー のアクセス許可があることを確認します。詳細については、「CodeBuild サービスロールの作成」を参照してください。

CodeBuild にアクセス許可がない場合、 というエラーが表示されることがありますUnexpected EC2 error: UnauthorizedOperation。このエラーは、 CodeBuild に VPC の操作に必要な Amazon EC2 アクセス許可がない場合に発生する可能性があります。

VPC の制限事項

- からの VPC 接続 CodeBuild は Windows ではサポートされていません。
- からの VPC 接続 CodeBuild は、共有 VPCsではサポートされていません。

VPC エンドポイントの使用

インターフェイス VPC エンドポイントを使用するように AWS CodeBuild を設定することで、ビルドのセキュリティを強化できます。インターフェイスエンドポイントは、プライベート IP アドレスを通じて Amazon EC2 および CodeBuild にプライベートにアクセスできるテクノロジーである PrivateLink を使用しています。PrivateLink は、マネージドインスタンス、CodeBuild および Amazon EC2 間のすべてのネットワークトラフィックを Amazon ネットワークに限定します。(マネージドインスタンスはインターネットにアクセスできません)。また、インターネットゲートウェイ、NAT デバイスあるいは仮想プライベートゲートウェイの必要はありません。PrivateLink の設定は要件ではありませんが、推奨されます。PrivateLink と VPC エンドポイントの詳細については、「AWS PrivateLink とは」を参照してください。

VPC エンドポイントを作成する前に

AWS CodeBuild の VPC エンドポイントを設定する前に、以下の制約と制限に注意してください。

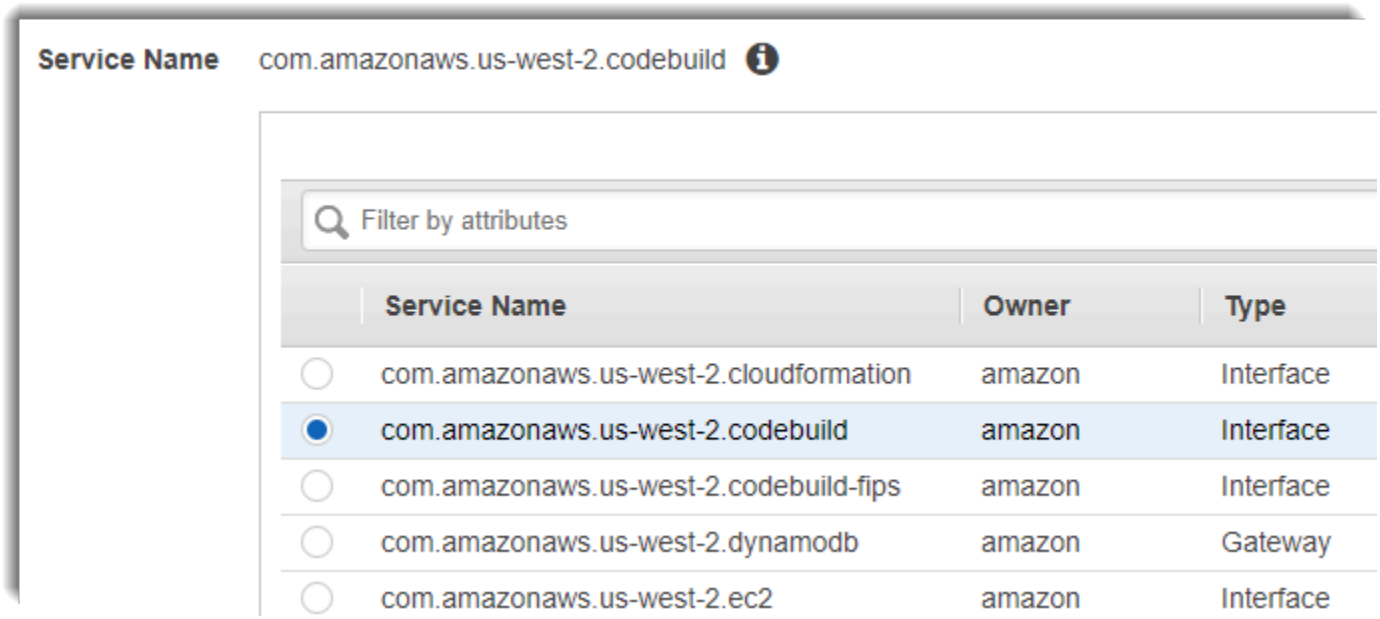
> **Note**
>
> Amazon VPC PrivateLink 接続をサポートしていない AWS サービスで CodeBuild を使用する場合、NAT ゲートウェイを使用します。

- VPC エンドポイントは、Amazon Route 53 を介してのみ Amazon 提供の DNS をサポートします。独自の DNS を使用する場合には、条件付き DNS 転送を使用できます。詳細については、「Amazon VPC ユーザーガイド」の「DHCP オプションセット」を参照してください。
- 現在、VPC エンドポイントはクロスリージョンリクエストをサポートしていません。ビルド入力と出力を保存する S3 バケットと同じ AWS リージョンで、エンドポイントを作成することを確認

します。バケットの場所は、Amazon S3 コンソールまたは get-bucket-locationコマンドを使用して確認できます。リージョン固有の Amazon S3 エンドポイントを使用してバケットにアクセスします (例: *<bucket-name>*.s3-us-west-2.amazonaws.com)。Amazon S3 のリージョン固有のエンドポイントの詳細については、「Amazon Web Services 全般のリファレンス」の「Amazon Simple Storage Service」を参照してください。AWS CLI を使用して Amazon S3 にリクエストを実行する場合は、デフォルトリージョンをバケットと同じリージョンに設定するか、またはリクエストで --region パラメータを使用します。

CodeBuild 用の VPC エンドポイントの作成

「インターフェイスエンドポイントの作成」の手順に従って、エンドポイント com.amazonaws.*region*.codebuild を作成します。これは、AWS CodeBuild の VPC エンドポイントです。



region は、米国東部 (オハイオ) リージョンの us-east-2 のように、CodeBuild でサポートされている AWS リージョンのリージョン識別子を表します。サポートされている AWS リージョンのリストについては、AWS 全般のリファレンスの「CodeBuild」を参照してください。エンドポイントには、AWS にサインインしたときに指定したリージョンが事前に設定されています。リージョンを変更すると、それに応じて VPC エンドポイントが更新されます。

CodeBuild 用の VPC エンドポイントポリシーを作成する

AWS CodeBuild には Amazon VPC エンドポイントのポリシーを作成することができます。以下の内容を指定できます。

- アクションを実行できるプリンシパル。
- 実行可能なアクション。
- 自身に対してアクションを実行できたリソース。

次のポリシー例では、すべてのプリンシパルが `project-name` プロジェクトのビルドの開始と表示のみ行えることを示します。

```
{
    "Statement": [
        {
            "Action": [
                "codebuild:ListBuildsForProject",
                "codebuild:StartBuild",
                "codebuild:BatchGetBuilds"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:codebuild:region-ID:account-ID:project/project-name",
            "Principal": "*"
        }
    ]
}
```

詳細については、Amazon VPC ユーザーガイドの「VPC エンドポイントによるサービスのアクセスコントロール」を参照してください。

AWS CloudFormation VPC テンプレート

AWS CloudFormation では、テンプレートファイルを使用してリソース群を単体 (スタック) としてまとめて作成および削除することで、AWS インフラストラクチャのデプロイを想定どおりに繰り返し作成およびプロビジョンできます。詳細については、AWS CloudFormation ユーザーガイドを参照してください。

AWS CloudFormation を使用するように VPC を設定するための AWS CodeBuild YAML テンプレートは次のとおりです。このファイルは「samples.zip」からも入手可能です。

```
Description:  This template deploys a VPC, with a pair of public and private subnets
 spread
  across two Availability Zones. It deploys an internet gateway, with a default
  route on the public subnets. It deploys a pair of NAT gateways (one in each AZ),
  and default routes for them in the private subnets.
```

```
Parameters:
  EnvironmentName:
    Description: An environment name that is prefixed to resource names
    Type: String

  VpcCIDR:
    Description: Please enter the IP range (CIDR notation) for this VPC
    Type: String
    Default: 10.192.0.0/16

  PublicSubnet1CIDR:
    Description: Please enter the IP range (CIDR notation) for the public subnet in the
 first Availability Zone
    Type: String
    Default: 10.192.10.0/24

  PublicSubnet2CIDR:
    Description: Please enter the IP range (CIDR notation) for the public subnet in the
 second Availability Zone
    Type: String
    Default: 10.192.11.0/24

  PrivateSubnet1CIDR:
    Description: Please enter the IP range (CIDR notation) for the private subnet in
 the first Availability Zone
    Type: String
    Default: 10.192.20.0/24

  PrivateSubnet2CIDR:
    Description: Please enter the IP range (CIDR notation) for the private subnet in
 the second Availability Zone
    Type: String
    Default: 10.192.21.0/24

Resources:
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: !Ref VpcCIDR
      EnableDnsSupport: true
      EnableDnsHostnames: true
      Tags:
        - Key: Name
```

```
        Value: !Ref EnvironmentName

  InternetGateway:
    Type: AWS::EC2::InternetGateway
    Properties:
      Tags:
        - Key: Name
          Value: !Ref EnvironmentName

  InternetGatewayAttachment:
    Type: AWS::EC2::VPCGatewayAttachment
    Properties:
      InternetGatewayId: !Ref InternetGateway
      VpcId: !Ref VPC

  PublicSubnet1:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId: !Ref VPC
      AvailabilityZone: !Select [ 0, !GetAZs '' ]
      CidrBlock: !Ref PublicSubnet1CIDR
      MapPublicIpOnLaunch: true
      Tags:
        - Key: Name
          Value: !Sub ${EnvironmentName} Public Subnet (AZ1)

  PublicSubnet2:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId: !Ref VPC
      AvailabilityZone: !Select [ 1, !GetAZs  '' ]
      CidrBlock: !Ref PublicSubnet2CIDR
      MapPublicIpOnLaunch: true
      Tags:
        - Key: Name
          Value: !Sub ${EnvironmentName} Public Subnet (AZ2)

  PrivateSubnet1:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId: !Ref VPC
      AvailabilityZone: !Select [ 0, !GetAZs  '' ]
      CidrBlock: !Ref PrivateSubnet1CIDR
      MapPublicIpOnLaunch: false
```

```
      Tags:
        - Key: Name
          Value: !Sub ${EnvironmentName} Private Subnet (AZ1)

  PrivateSubnet2:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId: !Ref VPC
      AvailabilityZone: !Select [ 1, !GetAZs  '' ]
      CidrBlock: !Ref PrivateSubnet2CIDR
      MapPublicIpOnLaunch: false
      Tags:
        - Key: Name
          Value: !Sub ${EnvironmentName} Private Subnet (AZ2)

  NatGateway1EIP:
    Type: AWS::EC2::EIP
    DependsOn: InternetGatewayAttachment
    Properties:
      Domain: vpc

  NatGateway2EIP:
    Type: AWS::EC2::EIP
    DependsOn: InternetGatewayAttachment
    Properties:
      Domain: vpc

  NatGateway1:
    Type: AWS::EC2::NatGateway
    Properties:
      AllocationId: !GetAtt NatGateway1EIP.AllocationId
      SubnetId: !Ref PublicSubnet1

  NatGateway2:
    Type: AWS::EC2::NatGateway
    Properties:
      AllocationId: !GetAtt NatGateway2EIP.AllocationId
      SubnetId: !Ref PublicSubnet2

  PublicRouteTable:
    Type: AWS::EC2::RouteTable
    Properties:
      VpcId: !Ref VPC
      Tags:
```

```
        - Key: Name
          Value: !Sub ${EnvironmentName} Public Routes

  DefaultPublicRoute:
    Type: AWS::EC2::Route
    DependsOn: InternetGatewayAttachment
    Properties:
      RouteTableId: !Ref PublicRouteTable
      DestinationCidrBlock: 0.0.0.0/0
      GatewayId: !Ref InternetGateway

  PublicSubnet1RouteTableAssociation:
    Type: AWS::EC2::SubnetRouteTableAssociation
    Properties:
      RouteTableId: !Ref PublicRouteTable
      SubnetId: !Ref PublicSubnet1

  PublicSubnet2RouteTableAssociation:
    Type: AWS::EC2::SubnetRouteTableAssociation
    Properties:
      RouteTableId: !Ref PublicRouteTable
      SubnetId: !Ref PublicSubnet2


  PrivateRouteTable1:
    Type: AWS::EC2::RouteTable
    Properties:
      VpcId: !Ref VPC
      Tags:
        - Key: Name
          Value: !Sub ${EnvironmentName} Private Routes (AZ1)

  DefaultPrivateRoute1:
    Type: AWS::EC2::Route
    Properties:
      RouteTableId: !Ref PrivateRouteTable1
      DestinationCidrBlock: 0.0.0.0/0
      NatGatewayId: !Ref NatGateway1

  PrivateSubnet1RouteTableAssociation:
    Type: AWS::EC2::SubnetRouteTableAssociation
    Properties:
      RouteTableId: !Ref PrivateRouteTable1
      SubnetId: !Ref PrivateSubnet1
```

```
  PrivateRouteTable2:
    Type: AWS::EC2::RouteTable
    Properties:
      VpcId: !Ref VPC
      Tags:
        - Key: Name
          Value: !Sub ${EnvironmentName} Private Routes (AZ2)

  DefaultPrivateRoute2:
    Type: AWS::EC2::Route
    Properties:
      RouteTableId: !Ref PrivateRouteTable2
      DestinationCidrBlock: 0.0.0.0/0
      NatGatewayId: !Ref NatGateway2

  PrivateSubnet2RouteTableAssociation:
    Type: AWS::EC2::SubnetRouteTableAssociation
    Properties:
      RouteTableId: !Ref PrivateRouteTable2
      SubnetId: !Ref PrivateSubnet2

  NoIngressSecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupName: "no-ingress-sg"
      GroupDescription: "Security group with no ingress rule"
      VpcId: !Ref VPC

Outputs:
  VPC:
    Description: A reference to the created VPC
    Value: !Ref VPC

  PublicSubnets:
    Description: A list of the public subnets
    Value: !Join [ ",", [ !Ref PublicSubnet1, !Ref PublicSubnet2 ]]

  PrivateSubnets:
    Description: A list of the private subnets
    Value: !Join [ ",", [ !Ref PrivateSubnet1, !Ref PrivateSubnet2 ]]

  PublicSubnet1:
    Description: A reference to the public subnet in the 1st Availability Zone
```

```
    Value: !Ref PublicSubnet1

  PublicSubnet2:
    Description: A reference to the public subnet in the 2nd Availability Zone
    Value: !Ref PublicSubnet2

  PrivateSubnet1:
    Description: A reference to the private subnet in the 1st Availability Zone
    Value: !Ref PrivateSubnet1

  PrivateSubnet2:
    Description: A reference to the private subnet in the 2nd Availability Zone
    Value: !Ref PrivateSubnet2

  NoIngressSecurityGroup:
    Description: Security group with no ingress rule
    Value: !Ref NoIngressSecurityGroup
```

プロキシサーバーでの AWS CodeBuild の使用

プロキシサーバーで AWS CodeBuild を使用して、インターネットとの間の HTTP および HTTPS トラフィックを規制できます。プロキシサーバーで CodeBuild を実行するには、パブリックサブネットにプロキシサーバーをインストールし、VPC のプライベートサブネットに CodeBuild をインストールします。

プロキシサーバーで CodeBuild を実行するためのプライマリユースケースが 2 つあります。

- これにより、VPC で NAT ゲートウェイや NAT インスタンスを使用する必要がなくなります。
- プロキシサーバーのインスタンスがアクセスを許可する URL と、プロキシサーバーがアクセスを拒否する URL を指定できます。

CodeBuild は、2 種類のプロキシサーバーで使用できます。どちらの場合も、プロキシサーバーはパブリックサブネットで動作し、CodeBuild はプライベートサブネットで動作します。

- 明示的なプロキシ: 明示的なプロキシサーバーを使用する場合は、NO_PROXY、HTTP_PROXY、およびHTTPS_PROXY の環境変数を CodeBuild のプロジェクトレベルで設定する必要があります。詳細については、「AWS CodeBuild でのビルドプロジェクトの設定の変更」および「 でのビルドプロジェクトの作成AWS CodeBuild」を参照してください。
- Transparent Proxy: 透過的なプロキシサーバーを使用する場合は、特別な設定は必要ありません。

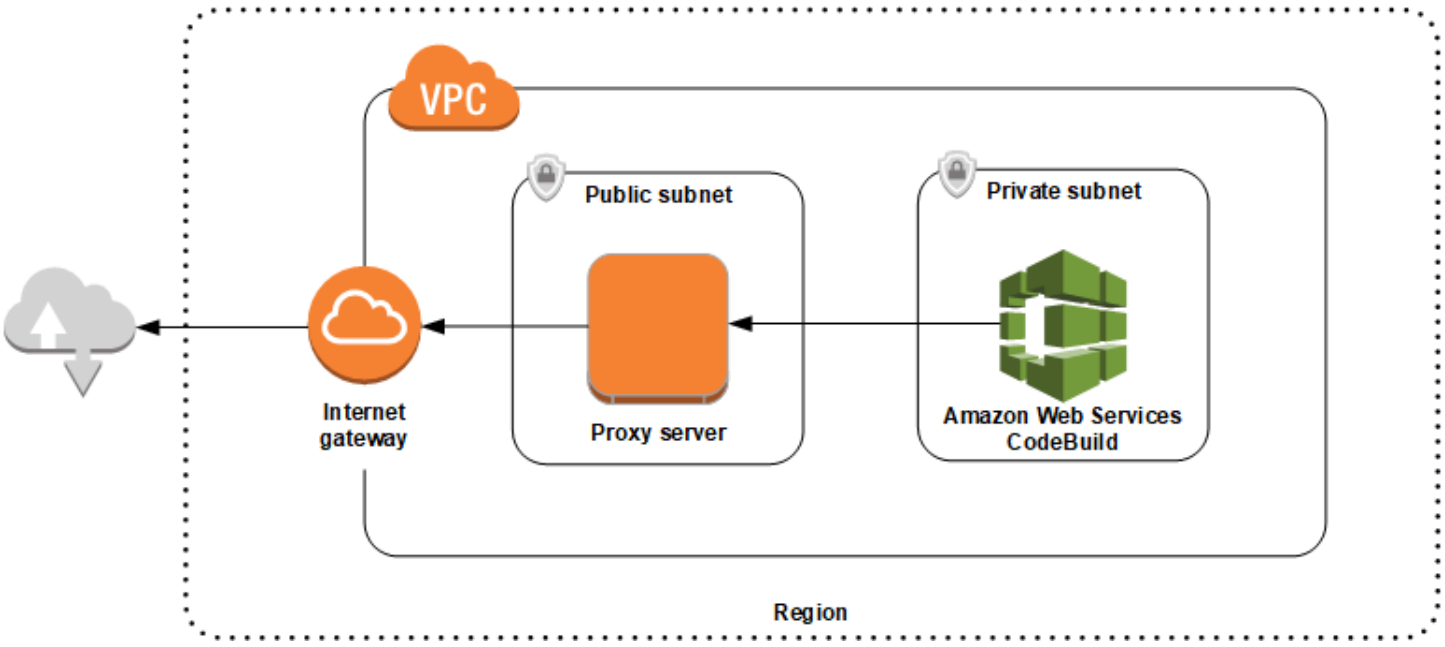
トピック

プロキシサーバーで CodeBuild を実行するために必要なコンポーネント

このコンポーネントは、透過的または明示的なプロキシサーバーで AWS CodeBuild を実行するために必要です。

- VPC。
- プロキシサーバー用に VPC 内の 1 つのパブリックサブネット。
- CodeBuild 用に VPC 内の 1 つのプライベートサブネット。
- VPC とインターネットの間の通信を可能にするインターネットゲートウェイ。

次の図は、これらのコンポーネントがどのように連携するかを示しています。



VPC、サブネット、ネットワークゲートウェイのセットアップ

以下のステップは、透過的または明示的なプロキシサーバーで AWS CodeBuild を実行するために必要です。

1. VPC を作成します。VPC の作成の詳細については、Amazon VPC ユーザーガイドの「VPC を作成する」をご参照ください。
2. VPC 内に 2 つのサブネットを作成します。1 つは、プロキシサーバーを実行する `Public Subnet` という名前のパブリックサブネットです。もう 1 つは、CodeBuild を実行する `Private Subnet` という名前のプライベートサブネットです。

   詳細については、「VPC でのサブネットの作成」を参照してください。
3. インターネットゲートウェイを作成して VPC にアタッチします。詳細については、「インターネットゲートウェイの作成とアタッチ」を参照してください。
4. VPC (0.0.0.0/0) からインターネットゲートウェイに送信トラフィックをルーティングするルールをデフォルトルートテーブルに追加します。詳細については、「ルートテーブルでのルートの追加および削除」を参照してください。
5. VPC (0.0.0.0/0) からの着信 SSH トラフィック (TCP 22) を許可するルールを VPC のデフォルトセキュリティグループに追加します。
6. Amazon EC2 ユーザーガイドの「インスタンス起動ウィザードを使用したインスタンスの起動」の指示に従って Amazon Linux インスタンスを起動します。ウィザードを実行する場合は次のオプションを選択してください。
   - [Choose an Instance Type (インスタンスタイプの選択)] で、Amazon Linux の Amazon マシンイメージ (AMI) を選択します。
   - [サブネット] で、このトピックで先に作成したパブリックサブネットを選択します。推奨された名前を使用した場合は、[Public Subnet] です。
   - [Auto-assign Public IP] で、[Enable] を選択します。
   - [セキュリティグループの設定] ページの [セキュリティグループの割り当て] で、[Select an existing security group (既存のセキュリティグループの選択)] を選択します。次に、デフォルトのセキュリティグループを選択します。
   - [起動] を選択したら、既存のキーペアを選択するか、新しいキーペアを作成します。

   それ以外のオプションについては、デフォルト設定を選択します。
7. EC2 インスタンスの実行後は、送信元/送信先チェックを無効にします。詳細については、Amazon VPC ユーザーガイドの「送信元/送信先チェックを無効にする」を参照してください。
8. VPC にルートテーブルを作成します。インターネット用のトラフィックをプロキシサーバーにルーティングするためのルールをルートテーブルに追加します。このルートテーブルをプライベートサブネットに関連付けます。これは、CodeBuild が実行されているプライベートサブネット内のインスタンスからのアウトバウンドリクエストを、常にプロキシサーバーを介してルーティングするために必要です。

プロキシサーバーのインストールと設定

選択できるプロキシサーバーは多数あります。ここでは、オープンソースのプロキシサーバー Squid を使用して、AWS CodeBuild がプロキシサーバーでどのように動作するかを説明します。同じ概念を他のプロキシサーバーにも適用できます。

Squid をインストールするには、次のコマンドを実行して yum repo を使用します。

```
sudo yum update -y
sudo yum install -y squid
```

Squid をインストールしたら、このトピックで後述する手順に従って、その `squid.conf` ファイルを編集します。

HTTPS トラフィック用の Squid の設定

HTTPS では、HTTP トラフィックは Transport Layer Security (TLS) 接続でカプセル化されます。Squid では、SslPeekAndSplice と呼ばれる機能を使用して、リクエストされたインターネットホストを含む TLS 初期化から Server Name Indication (SNI) を取得します。これは必須のため、Squid で HTTPS トラフィックを復元する必要はありません。SslPeekAndSplice を有効にするには、Squid に証明書が必要です。OpenSSL を使用してこの証明書を作成する:

```
sudo mkdir /etc/squid/ssl
cd /etc/squid/ssl
sudo openssl genrsa -out squid.key 2048
sudo openssl req -new -key squid.key -out squid.csr -subj "/C=XX/ST=XX/L=squid/O=squid/
CN=squid"
sudo openssl x509 -req -days 3650 -in squid.csr -signkey squid.key -out squid.crt
sudo cat squid.key squid.crt | sudo tee squid.pem
```

> **Note**
>
> HTTP では、Squid の設定は必要ありません。すべての HTTP/1.1 リクエストメッセージから、ホストヘッダーフィールドを取得することができます。これにより、リクエストされているインターネットホストが指定されます。

明示的なプロキシサーバーでの CodeBuild の実行

トピック

- 明示的なプロキシサーバーとしての Squid の設定
- CodeBuild プロジェクトを作成する
- 明示的なプロキシサーバーのサンプル squid.conf ファイル

明示的なプロキシサーバーで AWS CodeBuild を実行するには、外部サイトとのトラフィックを許可または拒否するようにプロキシサーバーを設定し、さらに HTTP_PROXY 環境変数と HTTPS_PROXY 環境変数を設定する必要があります。

明示的なプロキシサーバーとしての Squid の設定

Squid プロキシサーバーが明示的になるように設定するには、/etc/squid/squid.conf ファイルに次の変更を加える必要があります。

- 以下のデフォルトのアクセスコントロールリスト (ACL) ルールを削除します。

```
acl localnet src 10.0.0.0/8
acl localnet src 172.16.0.0/12
acl localnet src 192.168.0.0/16
acl localnet src fc00::/7
acl localnet src fe80::/10
```

削除したデフォルトの ACL ルールの代わりに次のコードを追加します。最初の行では VPC からのリクエストを許可します。次の 2 つの行では、AWS CodeBuild によって使用されている可能性のある送信先 URL へのアクセス権をプロキシサーバーに付与します。AWS リージョンで S3 バケットまたは CodeCommit リポジトリを指定するように、最終行の正規表現を変更します。次に例を示します。

- 送信元が Amazon S3 の場合は、acl download_src dstdom_regex .*s3\.us-west-1\.amazonaws\.com コマンドを使用して、us-west-1 リージョンの S3 バケットへのアクセスを許可します。
- 送信元が AWS CodeCommit の場合は、git-codecommit.<*your-region*>.amazonaws.com を使用して AWS リージョンを許可リストに追加します。

```
acl localnet src 10.1.0.0/16 #Only allow requests from within the VPC
acl allowed_sites dstdomain .github.com #Allows to download source from GitHub
acl allowed_sites dstdomain .bitbucket.com #Allows to download source from Bitbucket
acl download_src dstdom_regex .*\.amazonaws\.com #Allows to download source from
 Amazon S3 or CodeCommit
```

- `http_access allow localnet` を次のように置き換えます。

```
http_access allow localnet allowed_sites
http_access allow localnet download_src
```

- ビルドでログとアーティファクトをアップロードする場合は、次のいずれかを実行します。

1. `http_access deny all` ステートメントの前に、次のステートメントを挿入します。これにより、CodeBuild が CloudWatch と Amazon S3 にアクセスできるようになります。CodeBuild が CloudWatch Logs を作成できるようにするには、CloudWatch へのアクセスが必要です。Amazon S3 へのアクセスは、アーティファクトのアップロードと Amazon S3 のキャッシングを行う上で必要です。

   -
   ```
   https_port 3130 cert=/etc/squid/ssl/squid.pem ssl-bump intercept
   acl SSL_port port 443
   http_access allow SSL_port
   acl allowed_https_sites ssl::server_name .amazonaws.com
   acl step1 at_step SslBump1
   acl step2 at_step SslBump2
   acl step3 at_step SslBump3
   ssl_bump peek step1 all
   ssl_bump peek step2 allowed_https_sites
   ssl_bump splice step3 allowed_https_sites
   ssl_bump terminate step2 all
   ```

   - `squid.conf` 保存後、次のコマンドを実行します。

   ```
   sudo iptables -t nat -A PREROUTING -p tcp --dport 443 -j REDIRECT --to-port 3130
   sudo service squid restart
   ```

2. `proxy` を buildspec ファイルに追加します。詳細については、「buildspec の構文」を参照してください。

```
version: 0.2
proxy:
  upload-artifacts: yes
  logs: yes
phases:
  build:
    commands:
      - command
```

> **Note**
>
> RequestError タイムアウトエラーが表示される場合は、「RequestError プロキシサーバー CodeBuild での実行時のタイムアウトエラー」を参照してください。

詳細については、このトピックで後述する「明示的なプロキシサーバーのサンプル squid.conf ファイル」を参照してください。

CodeBuild プロジェクトを作成する

明示的なプロキシサーバーで AWS CodeBuild を実行するには、その HTTP_PROXY 環境変数と HTTPS_PROXY 環境変数に、プロキシサーバー用に作成した EC2 インスタンスのプライベート IP アドレスとポート 3128 をプロジェクトレベルで設定します。プライベート IP アドレスは、http://*your-ec2-private-ip-address*:3128 のようになります。詳細については、「でのビルドプロジェクトの作成AWS CodeBuild」および「AWS CodeBuild でのビルドプロジェクトの設定の変更」を参照してください。

Squid プロキシのアクセスログを表示するには、次のコマンドを使用します。

```
sudo tail -f /var/log/squid/access.log
```

明示的なプロキシサーバーのサンプル **squid.conf** ファイル

明示的なプロキシサーバー用に設定した squid.conf ファイルの例を次に示します。

```
 acl localnet src 10.0.0.0/16 #Only allow requests from within the VPC
 # add all URLS to be whitelisted for download source and commands to be run in build
environment
 acl allowed_sites dstdomain .github.com    #Allows to download source from github
 acl allowed_sites dstdomain .bitbucket.com #Allows to download source from bitbucket
 acl allowed_sites dstdomain ppa.launchpad.net #Allows to run apt-get in build
environment
 acl download_src dstdom_regex .*\.amazonaws\.com #Allows to download source from S3
or CodeCommit
 acl SSL_ports port 443
 acl Safe_ports port 80  # http
 acl Safe_ports port 21  # ftp
 acl Safe_ports port 443  # https
 acl Safe_ports port 70  # gopher
 acl Safe_ports port 210  # wais
```

```
  acl Safe_ports port 1025-65535 # unregistered ports
  acl Safe_ports port 280  # http-mgmt
  acl Safe_ports port 488  # gss-http
  acl Safe_ports port 591  # filemaker
  acl Safe_ports port 777  # multiling http
  acl CONNECT method CONNECT
  #
  # Recommended minimum Access Permission configuration:
  #
  # Deny requests to certain unsafe ports
  http_access deny !Safe_ports
  # Deny CONNECT to other than secure SSL ports
  http_access deny CONNECT !SSL_ports
  # Only allow cachemgr access from localhost
  http_access allow localhost manager
  http_access deny manager
  # We strongly recommend the following be uncommented to protect innocent
  # web applications running on the proxy server who think the only
  # one who can access services on "localhost" is a local user
  #http_access deny to_localhost
  #
  # INSERT YOUR OWN RULE(S) HERE TO ALLOW ACCESS FROM YOUR CLIENTS
  #
  # Example rule allowing access from your local networks.
  # Adapt localnet in the ACL section to list your (internal) IP networks
  # from where browsing should be allowed
  http_access allow localnet allowed_sites
  http_access allow localnet download_src
  http_access allow localhost
  # Add this for CodeBuild to access CWL end point, caching and upload artifacts S3
 bucket end point
  https_port 3130 cert=/etc/squid/ssl/squid.pem ssl-bump intercept
  acl SSL_port port 443
  http_access allow SSL_port
  acl allowed_https_sites ssl::server_name .amazonaws.com
  acl step1 at_step SslBump1
  acl step2 at_step SslBump2
  acl step3 at_step SslBump3
  ssl_bump peek step1 all
  ssl_bump peek step2 allowed_https_sites
  ssl_bump splice step3 allowed_https_sites
  ssl_bump terminate step2 all
  # And finally deny all other access to this proxy
  http_access deny all
```

```
  # Squid normally listens to port 3128
  http_port 3128
  # Uncomment and adjust the following to add a disk cache directory.
  #cache_dir ufs /var/spool/squid 100 16 256
  # Leave coredumps in the first cache dir
  coredump_dir /var/spool/squid
  #
  # Add any of your own refresh_pattern entries above these.
  #
  refresh_pattern ^ftp:  1440 20% 10080
  refresh_pattern ^gopher: 1440 0% 1440
  refresh_pattern -i (/cgi-bin/|\?) 0 0% 0
  refresh_pattern .  0 20% 4320
```

透過的なプロキシサーバーでの CodeBuild の実行

透過的なプロキシサーバーで AWS CodeBuild を実行するには、やり取りするウェブサイトおよびドメインへのアクセス権を持つプロキシサーバーを設定する必要があります。

透過的なプロキシサーバーとしての Squid の設定

プロキシサーバーが透過的になるように設定するには、アクセスするドメインやウェブサイトへのアクセス権を付与する必要があります。透過的なプロキシサーバーで AWS CodeBuild を実行するには、amazonaws.com へのアクセス権を付与する必要があります。また、CodeBuild で使用する他のウェブサイトへのアクセス権も付与します。これらのアクセス権は、CodeBuild プロジェクトの作成方法によって異なります。ウェブサイトの例は、GitHub、Bitbucket、Yum、Maven などのリポジトリ用です。特定のドメインやウェブサイトへのアクセスを Squid に許可するには、次のようなコマンドを使用して squid.conf ファイルを更新します。このサンプルコマンドは amazonaws.com、github.com、および bitbucket.com へのアクセスを許可します。このサンプルは、他のウェブサイトへのアクセス権を付与するように編集できます。

```
cat | sudo tee /etc/squid/squid.conf #EOF
visible_hostname squid
#Handling HTTP requests
http_port 3129 intercept
acl allowed_http_sites dstdomain .amazonaws.com
#acl allowed_http_sites dstdomain domain_name [uncomment this line to add another
 domain]
http_access allow allowed_http_sites
#Handling HTTPS requests
https_port 3130 cert=/etc/squid/ssl/squid.pem ssl-bump intercept
```

```
acl SSL_port port 443
http_access allow SSL_port
acl allowed_https_sites ssl::server_name .amazonaws.com
acl allowed_https_sites ssl::server_name .github.com
acl allowed_https_sites ssl::server_name .bitbucket.com
#acl allowed_https_sites ssl::server_name [uncomment this line to add another website]
acl step1 at_step SslBump1
acl step2 at_step SslBump2
acl step3 at_step SslBump3
ssl_bump peek step1 all
ssl_bump peek step2 allowed_https_sites
ssl_bump splice step3 allowed_https_sites
ssl_bump terminate step2 all
http_access deny all
EOF
```

プライベートサブネット内のインスタンスからの着信リクエストで、Squid ポートにリダイレクトする必要があります。Squid は HTTP トラフィック (80 の代理) をポート 3129、HTTPS トラフィック (443 の代理) をポート 3130 でリッスンします。トラフィックをルーティングするには、iptables コマンドを使用します。

```
sudo iptables -t nat -A PREROUTING -p tcp --dport 80 -j REDIRECT --to-port 3129
sudo iptables -t nat -A PREROUTING -p tcp --dport 443 -j REDIRECT --to-port 3130
sudo service iptables save
sudo service squid start
```

CodeBuild プロジェクトを作成する

プロキシサーバーを設定したら、プライベートサブネットの AWS CodeBuild で使用できます。追加設定は不要です。HTTP および HTTPS リクエストはすべて、パブリックプロキシサーバーを経由します。Squid プロキシのアクセスログを表示するには、次のコマンドを使用します。

```
sudo tail -f /var/log/squid/access.log
```

プロキシサーバーでのパッケージマネージャーなどのツールの実行

パッケージマネージャーなどのツールをプロキシサーバーで実行する方法

1. `squid.conf` ファイルにステートメントを追加し、プロキシサーバーの許可リストにツールを追加します。

2. プロキシサーバーのプライベートエンドポイントを指す行を buildspec ファイルに追加します。

次の例では、apt-get、curl、および maven でこの作業を行う方法を示しています。別のツールを使用する場合は、同じ原則が適用されます。これを squid.conf ファイルの許可リストに追加し、プロキシサーバーのエンドポイントを CodeBuild に認識させるためのコマンドを buildspec ファイルに追加します。

プロキシサーバーで **apt-get** を実行するには

1. 次のステートメントを squid.conf ファイルに追加し、プロキシサーバーの許可リストに apt-get を追加します。最初の 3 行では、ビルド環境で apt-get を実行できるようになります。

```
acl allowed_sites dstdomain ppa.launchpad.net # Required for apt-get to run in the
 build environment
acl apt_get dstdom_regex .*\.launchpad.net # Required for CodeBuild to run apt-get
 in the build environment
acl apt_get dstdom_regex .*\.ubuntu.com    # Required for CodeBuild to run apt-get
 in the build environment
http_access allow localnet allowed_sites
http_access allow localnet apt_get
```

2. apt-get コマンドで /etc/apt/apt.conf.d/00proxy のプロキシ設定を検索できるように、次のステートメントを buildspec ファイルを追加します。

```
echo 'Acquire::http::Proxy "http://<private-ip-of-proxy-server>:3128";
 Acquire::https::Proxy "http://<private-ip-of-proxy-server>:3128";
 Acquire::ftp::Proxy "http://<private-ip-of-proxy-server>:3128";' > /etc/apt/
apt.conf.d/00proxy
```

プロキシサーバーで **curl** を実行するには

1. 次の内容を squid.conf ファイルに追加し、ビルド環境の許可リストに curl を追加します。

```
acl allowed_sites dstdomain ppa.launchpad.net # Required to run apt-get in the
 build environment
acl allowed_sites dstdomain google.com # Required for access to a webiste. This
 example uses www.google.com.
http_access allow localnet allowed_sites
```

```
http_access allow localnet apt_get
```

2. curl でプライベートプロキシサーバーを使用して squid.conf に追加したウェブサイトにアクセスできるように、次のステートメントを buildspec ファイルに追加します。この例では、ウェブサイトは google.com です。

```
curl -x <private-ip-of-proxy-server>:3128 https://www.google.com
```

プロキシサーバーで **maven** を実行するには

1. 次の内容を squid.conf ファイルに追加し、ビルド環境の許可リストに maven を追加します。

```
acl allowed_sites dstdomain ppa.launchpad.net # Required to run apt-get in the
 build environment
acl maven dstdom_regex .*\.maven.org # Allows access to the maven repository in the
 build environment
http_access allow localnet allowed_sites
http_access allow localnet maven
```

2. buildspec ファイルに次のステートメントを追加します。

```
maven clean install -DproxySet=true -DproxyHost=<private-ip-of-proxy-server> -
DproxyPort=3128
```

AWS CodeBuild でのビルドプロジェクトとビルドの使用

開始するには、「ビルドプロジェクトの作成」の手順に従い、次に「ビルドの実行」の手順に従ってください。ビルドプロジェクトおよびビルドの詳細については、以下のトピックを参照してください。

トピック

- ビルドプロジェクトを操作する
- AWS CodeBuild でのビルドの使用

ビルドプロジェクトを操作する

ビルドプロジェクトには、ビルドの実行方法に関する情報が含まれています。これには、ソースコードの取得先、使用するビルド環境、実行するビルドコマンド、ビルド出力の格納先が含まれます。

ビルドプロジェクトを操作して以下のタスクを実行できます。

トピック

- でのビルドプロジェクトの作成AWS CodeBuild
- 通知ルールの作成
- AWS CodeBuild でのビルドプロジェクト名の一覧表示
- AWS CodeBuild でビルドプロジェクトの詳細を表示する
- AWS CodeBuild でのキャッシュのビルド
- でトリガーを構築する AWS CodeBuild
- GitLab 接続
- でのウェブフックの使用 AWS CodeBuild
- AWS CodeBuild でのビルドプロジェクトの設定の変更
- AWS CodeBuild でのビルドプロジェクトの削除
- 共有プロジェクトの使用
- AWS CodeBuild でのプロジェクトのタグ付け
- でのバッチビルド AWS CodeBuild
- GitHub のアクションランナー AWS CodeBuild
- AWS CodeBuild でのパブリックビルドプロジェクト

でのビルドプロジェクトの作成AWS CodeBuild

ビルドプロジェクトを作成するには、AWS CodeBuild コンソール、AWS CLI、または AWS SDK を使用できます。

Prerequisites

ビルドプロジェクトを作成する前に、ビルドを計画する の質問に回答します。

トピック

- ビルドプロジェクトの作成 (コンソール)
- ビルドプロジェクトの作成 (AWS CLI)
- ビルドプロジェクトの作成 (AWS SDK)
- ビルドプロジェクトの作成 (AWS CloudFormation)

ビルドプロジェクトの作成 (コンソール)

https://console.aws.amazon.com/codesuite/codebuild/home で AWS CodeBuild コンソールを開きます。

CodeBuild 情報ページが表示されたら、ビルドプロジェクトの作成を選択します。それ以外の場合は、ナビゲーションペインでビルドを展開し、[ビルドプロジェクト] を選択し、次に [Create build project (ビルドプロジェクトの作成)] を選択します。

[Create build project (ビルドプロジェクトの作成)] を選択します。

次のセクションに入力します。完了したら、ページの下部にある [Create build project] (ビルドプロジェクトを作成する) を選択します。

セクション:

- プロジェクトの設定
- ソース
- 環境
- Buildspec
- Batch 構成
- アーティファクト
- ログ

プロジェクトの設定

[Project name] (プロジェクト名)

このビルドプロジェクトの名前を入力します。ビルドプロジェクト名は、各 AWS アカウントで一意である必要があります。

説明

また、他のユーザーがこのプロジェクトの使用目的を理解できるように、ビルドプロジェクトの説明を任意で指定することもできます。

ビルドバッジ

(オプション)[Enable build badge] (ビルドバッジを有効にする) を選択すると、プロジェクトのビルドステータスが表示可能および埋め込み可能になります。詳細については、「ビルドバッジサンプル」を参照してください。

> **Note**
>
> ソースプロバイダーが Amazon S3 の場合、ビルドバッジは適用されません。

同時ビルド制限を有効にする

(オプション) このプロジェクトで同時ビルド数を制限するには、次の手順を実行します。

1. [Restrict number of concurrent builds this project can start] (このジョブで許可される同時実行の最大数を設定) を選択します。
2. [Concurrent build limit] (同時ビルド制限) で、このジョブで許可される同時実行の最大数を設定します。この制限は、アカウントに設定された同時ビルド制限より大きくすることはできません。アカウント制限を超える数値を入力しようとすると、エラーメッセージが表示されます。

新しいビルドは、現在のビルド数がこの制限以下の場合にのみ開始されます。現在のビルドカウントがこの制限を満たす場合、新しいビルドはスロットルされ、実行されません。

追加情報

(オプション) タグには、サポート AWS サービスで使用するタグの名前と値を入力します。[Add row] を使用して、タグを追加します。最大 50 個のタグを追加できます。

ソース

ソースプロバイダー

ソースコードプロバイダーのタイプを選択します。次のリストを使用して、ソースプロバイダーに関する適切な選択を行います。

> **Note**
>
> CodeBuild は Bitbucket サーバーをサポートしていません。

Amazon S3

バケット

ソースコードが格納されている入力バケットの名前を選択します。

S3 オブジェクトキーまたは S3 フォルダ

ZIP ファイルの名前、またはソースコードを含むフォルダへのパスを入力します。S3 バケットの中身をすべてダウンロードするには、スラッシュ記号 (/) を入力します。

ソースバージョン

入力ファイルのビルドを表すオブジェクトのバージョン IDを入力。詳細については、「」を参照してくださいのソースバージョンサンプル AWS CodeBuild

CodeCommit

リポジトリ

使用するリポジトリを選択します。

参照タイプ

[Branch] (ブランチ) または [Git tag] (Git タグ) を選択するか、[Commit ID] (コミット ID) を入力して、ソースコードのバージョンを指定します。詳細については、「のソースバージョンサンプル AWS CodeBuild」を参照してください。

> **Note**
>
> 811dd1ba1aba14473856cee38308caed7190c0d または 5392f7 のように、コミット ID と似ていない Git ブランチ名を選択することをお勧めします。これにより、Git checkout が実際のコミットと衝突するのを防ぐことができます。

Git クローンの深度

選択して、指定されるコミット数で切り捨てられる履歴の浅いクローンを作成します。完全クローンを希望する場合には、[Full (完全)] を選択します。

Git サブモジュール

リポジトリに Git サブモジュールを含める場合は、[Git サブモジュールを使用する] を選択します。

Bitbucket

リポジトリ

[Connect using OAuth] (OAuth を使用して接続する) または [Connect with a Bitbucket app password] (Bitbucket アプリパスワードで接続する) を選択し、手順に従って Bitbucket に接続 (または再接続) します。

パブリックのリポジトリかアカウント内のリポジトリかを選択します。

ソースバージョン

ブランチ、コミット ID、タグあるいはリファレンスとコミット ID を入力します。詳細については、「のソースバージョンサンプル AWS CodeBuild」を参照してください。

> **Note**
>
> 811dd1ba1aba14473856cee38308caed7190c0d または 5392f7 のように、コミット ID と似ていない Git ブランチ名を選択することをお勧めします。これにより、Git checkout が実際のコミットと衝突するのを防ぐことができます。

Git クローンの深度

[Git のクローンの深さ] を選択して、指定されるコミット数で切り捨てられる履歴の浅いクローンを作成します。完全クローンを希望する場合には、[Full (完全)] を選択します。

Git サブモジュール

リポジトリに Git サブモジュールを含める場合は、[Git サブモジュールを使用する] を選択します。

ビルドステータス

ビルドの開始と終了のステータスをソースプロバイダーにレポートする場合は、[Report build statuses to source provider when your builds start and finish] (ビルドの開始と終了時にソースプロバイダーにビルドステータスをレポートする) を選択します。

ソースプロバイダにビルド状態を報告できるようにするには、ソースプロバイダに関連付けられたユーザーがリポジトリへの書き込みアクセス権を持っている必要があります。ユーザーが書き込みアクセス権を持っていない場合、ビルドのステータスは更新できません。詳細については、「ソースプロバイダーのアクセス」を参照してください。

[Status context] (ステータスコンテキスト) に、Bitbucket コミットステータスの `name` パラメータに使用する値を記入します。詳細については、Bitbucket API ドキュメントの「ビルド」を参照してください。

[Target URL] (ターゲットURL) に、Bitbucket コミットステータスの `url` パラメータに使用する値を記入します。詳細については、Bitbucket API ドキュメントの「ビルド」を参照してください。

webhook によってトリガーされたビルドのステータスは常にソースプロバイダーにレポートされます。コンソールから開始されたビルドのステータスまたはソースプロバイダーに報告された API 呼び出しを取得するには、この設定を選択する必要があります。

プロジェクトのビルドが webhook によってトリガーされた場合、この設定への変更を有効にするには、新しいコミットをリポジトリにプッシュする必要があります。

プライマリソースウェブフックイベント で、コード変更がこのリポジトリにプッシュされるたびにソースコードを構築する場合は、コード変更がこのリポジトリにプッシュされるたびに再構築を選択します。 CodeBuild Webhook およびフィルターグループの詳細については、「Bitbucket ウェブフックイベント」を参照してください。

GitHub

リポジトリ

OAuth を使用して接続 または GitHub 個人アクセストークンを使用して接続 を選択し、手順に従って に接続 (または再接続) GitHub し、 へのアクセスを許可します AWS CodeBuild。

パブリックのリポジトリかアカウント内のリポジトリかを選択します。

ソースバージョン

ブランチ、コミット ID、タグあるいはリファレンスとコミット ID を入力します。詳細については、「のソースバージョンサンプル AWS CodeBuild」を参照してください。

> **Note**
>
> 811dd1ba1aba14473856cee38308caed7190c0d または 5392f7 のように、コミット ID と似ていない Git ブランチ名を選択することをお勧めします。これにより、Git checkout が実際のコミットと衝突するのを防ぐことができます。

Git クローンの深度

[Git のクローンの深さ] を選択して、指定されるコミット数で切り捨てられる履歴の浅いクローンを作成します。完全クローンを希望する場合には、[Full (完全)] を選択します。

Git サブモジュール

リポジトリに Git サブモジュールを含める場合は、[Git サブモジュールを使用する] を選択します。

ビルドステータス

ビルドの開始と終了のステータスをソースプロバイダーにレポートする場合は、[Report build statuses to source provider when your builds start and finish] (ビルドの開始と終了時にソースプロバイダーにビルドステータスをレポートする) を選択します。

ソースプロバイダにビルド状態を報告できるようにするには、ソースプロバイダに関連付けられたユーザーがリポジトリへの書き込みアクセス権を持っている必要があります。ユーザーが書き込みアクセス権を持っていない場合、ビルドのステータスは更新できません。詳細については、「ソースプロバイダーのアクセス」を参照してください。

ステータスコンテキスト には、 GitHub コミットステータスの `context`パラメータに使用する値を入力します。詳細については、「 デベロッパーガイド」の「コミットステータスの作成」を参照してください。 GitHub

ターゲット URL に、 GitHub コミットステータスの `target_url`パラメータに使用する値を入力します。詳細については、「 デベロッパーガイド」の「コミットステータスの作成」を参照してください。 GitHub

webhook によってトリガーされたビルドのステータスは常にソースプロバイダーにレポートされます。コンソールから開始されたビルドのステータスまたはソースプロバイダーに報告された API 呼び出しを取得するには、この設定を選択する必要があります。

プロジェクトのビルドが webhook によってトリガーされた場合、この設定への変更を有効にするには、新しいコミットをリポジトリにプッシュする必要があります。

プライマリソースウェブフックイベント で、コード変更がこのリポジトリにプッシュされるたびにソースコードを構築する場合は、コード変更がこのリポジトリにプッシュされるたびに再構築を選択します。 CodeBuild Webhook およびフィルターグループの詳細については、「GitHub ウェブフックイベント」を参照してください。

GitHub Enterprise Server

GitHub エンタープライズ個人用アクセストークン

個人用アクセストークンのクリップボードにコピーする方法に関しては「GitHub Enterprise Server のサンプル」参照してください。テキストフィールドにトークンを貼り付け、[トークンの保存] を選択します。

> **Note**
>
> 個人用アクセストークンは、一回のみ入力して保存することが必要となります。 CodeBuild は、今後のすべてのプロジェクトでこのトークンを使用します。

ソースバージョン

プルリクエスト、ブランチ、コミット ID、コミット ID、参照、およびコミット ID を入力します。詳細については、「のソースバージョンサンプル AWS CodeBuild」を参照してください。

> **Note**
>
> 811dd1ba1aba14473856cee38308caed7190c0d または 5392f7 のように、コミット ID と似ていない Git ブランチ名を選択することをお勧めします。これにより、Git checkout が実際のコミットと衝突するのを防ぐことができます。

Git クローンの深度

[Git のクローンの深さ] を選択して、指定されるコミット数で切り捨てられる履歴の浅いクローンを作成します。完全クローンを希望する場合には、[Full (完全)] を選択します。

Git サブモジュール

リポジトリに Git サブモジュールを含める場合は、[Git サブモジュールを使用する] を選択します。

ビルドステータス

ビルドの開始と終了のステータスをソースプロバイダーにレポートする場合は、[Report build statuses to source provider when your builds start and finish] (ビルドの開始と終了時にソースプロバイダーにビルドステータスをレポートする) を選択します。

ソースプロバイダにビルド状態を報告できるようにするには、ソースプロバイダに関連付けられたユーザーがリポジトリへの書き込みアクセス権を持っている必要があります。ユーザーが書き込みアクセス権を持っていない場合、ビルドのステータスは更新できません。詳細については、「ソースプロバイダーのアクセス」を参照してください。

ステータスコンテキスト には、 GitHub コミットステータスの `context`パラメータに使用する値を入力します。詳細については、「 デベロッパーガイド」の「コミットステータスの作成」を参照してください。 GitHub

ターゲット URL には、 GitHub コミットステータスの `target_url`パラメータに使用する値を入力します。詳細については、「 デベロッパーガイド」の「コミットステータスの作成」を参照してください。 GitHub

webhook によってトリガーされたビルドのステータスは常にソースプロバイダーにレポートされます。コンソールから開始されたビルドのステータスまたはソースプロバイダーに報告された API 呼び出しを取得するには、この設定を選択する必要があります。

プロジェクトのビルドが webhook によってトリガーされた場合、この設定への変更を有効にするには、新しいコミットをリポジトリにプッシュする必要があります。

安全でない SSL

Enterprise プロジェクトリポジトリへの接続中に SSL 警告を無視するには、安全でない SSL GitHub を有効にするを選択します。

プライマリソースのウェブフックイベント で、コード変更がこのリポジトリにプッシュされるたびにソースコードを構築する場合は、コード変更がこのリポジトリにプッシュされるたびに再構築を選択します。 CodeBuild Webhook およびフィルターグループの詳細については、「GitHub ウェブフックイベント」を参照してください。

GitLab

Connection

を使用して GitLab アカウントを接続し AWS CodeConnections、その接続を使用してサードパーティーのリポジトリをビルドプロジェクトのソースとして関連付けます。

デフォルト接続 またはカスタム接続 を選択します。

デフォルトの接続では、すべてのプロジェクトにデフォルトの GitLab 接続が適用されます。カスタム接続は、アカウントのデフォルト設定を上書きするカスタム GitLab 接続を適用します。

デフォルト接続

アカウントに関連付けられているデフォルトの接続の名前。

プロバイダーへの接続をまだ作成していない場合は、への接続 GitLabを作成する (コンソール)「」の手順を参照してください。

カスタム接続

使用するカスタム接続の名前を選択します。

プロバイダーへの接続をまだ作成していない場合は、への接続 GitLabを作成する (コンソール)「」の手順を参照してください。

リポジトリ

使用するリポジトリを選択します。

ソースバージョン

プルリクエスト ID、ブランチ、コミット ID、タグ、またはリファレンスとコミット ID を入力します。詳細については、「のソースバージョンサンプル AWS CodeBuild」を参照してください。

> **Note**
>
> 811dd1ba1aba14473856cee38308caed7190c0d または 5392f7 のように、コミット ID と似ていない Git ブランチ名を選択することをお勧めします。これにより、Git checkout が実際のコミットと衝突するのを防ぐことができます。

Git クローンの深度

[Git のクローンの深さ] を選択して、指定されるコミット数で切り捨てられる履歴の浅いクローンを作成します。完全クローンを希望する場合には、[Full (完全)] を選択します。

ビルドステータス

ビルドの開始と終了のステータスをソースプロバイダーにレポートする場合は、[Report build statuses to source provider when your builds start and finish] (ビルドの開始と終了時にソースプロバイダーにビルドステータスをレポートする) を選択します。

ソースプロバイダにビルド状態を報告できるようにするには、ソースプロバイダに関連付けられたユーザーがリポジトリへの書き込みアクセス権を持っている必要があります。ユーザーが書き込みアクセス権を持っていない場合、ビルドのステータスは更新できません。詳細については、「ソースプロバイダーのアクセス」を参照してください。

GitLab Self Managed

Connection

を使用して GitLab アカウントを接続し AWS CodeConnections、その接続を使用してサードパーティーのリポジトリをビルドプロジェクトのソースとして関連付けます。

デフォルト接続 またはカスタム接続 を選択します。

デフォルトの接続では、すべてのプロジェクトにデフォルトの GitLab セルフマネージド接続が適用されます。カスタム接続は、アカウントのデフォルト設定を上書きするカスタム GitLab セルフマネージド接続を適用します。

デフォルト接続

アカウントに関連付けられているデフォルトの接続の名前。

プロバイダーへの接続をまだ作成していない場合は、「 デベロッパーツールコンソールユーザーガイド」の GitLab 「自己管理型 への接続を作成する」を参照してください。

カスタム接続

使用するカスタム接続の名前を選択します。

プロバイダーへの接続をまだ作成していない場合は、「 デベロッパーツールコンソールユーザーガイド」の GitLab 「自己管理型 への接続を作成する」を参照してください。

リポジトリ

使用するリポジトリを選択します。

ソースバージョン

プルリクエスト ID、ブランチ、コミット ID、タグ、またはリファレンスとコミット ID を入力します。詳細については、「のソースバージョンサンプル AWS CodeBuild」を参照してください。

> **Note**
>
> 811dd1ba1aba14473856cee38308caed7190c0d または 5392f7 のように、コミット ID と似ていない Git ブランチ名を選択することをお勧めします。これにより、Git checkout が実際のコミットと衝突するのを防ぐことができます。

Git クローンの深度

[Git のクローンの深さ] を選択して、指定されるコミット数で切り捨てられる履歴の浅いクローンを作成します。完全クローンを希望する場合には、[Full (完全)] を選択します。

ビルドステータス

ビルドの開始と終了のステータスをソースプロバイダーにレポートする場合は、[Report build statuses to source provider when your builds start and finish] (ビルドの開始と終了時にソースプロバイダーにビルドステータスをレポートする) を選択します。

ソースプロバイダにビルド状態を報告できるようにするには、ソースプロバイダに関連付けられたユーザーがリポジトリへの書き込みアクセス権を持っている必要があります。ユーザーが

書き込みアクセス権を持っていない場合、ビルドのステータスは更新できません。詳細については、「ソースプロバイダーのアクセス」を参照してください。

環境

プロビジョニングモデル

次のいずれかを行います。

- によって管理されるオンデマンドフリートを使用するには AWS CodeBuild、オンデマンド を選択します。オンデマンドフリートでは、 CodeBuild はビルドのコンピューティングを提供します。マシンはビルドが終了すると破棄されます。オンデマンドフリートはフルマネージド型で、需要の急増にも対応できる自動スケーリング機能を備えています。
- によって管理されるリザーブドキャパシティフリートを使用するには AWS CodeBuild、リザーブドキャパシティ を選択し、フリート名 を選択します。リザーブドキャパシティフリートでは、ビルド環境に合わせて専有インスタンスのセットを設定します。これらのマシンはアイドル状態のままで、ビルドやテストをすぐに処理できる状態になり、ビルド時間を短縮します。リザーブドキャパシティフリートでは、マシンは常に稼働しており、プロビジョニングされている間はコストが発生し続けます。

詳細については、「でのリザーブドキャパシティの操作 AWS CodeBuild」を参照してください。

環境イメージ

次のいずれかを行います。

- によって管理される Docker イメージを使用するには AWS CodeBuild、マネージドイメージ を選択し、オペレーティングシステム 、ランタイム (s)、イメージ 、イメージバージョン から選択します。利用可能な場合は、[環境タイプ] から選択します。
- 別の Docker イメージを使用するには、[カスタムイメージ] を選択します。[Environment type (環境タイプ)] で、 [ARM]、[Linux]、[Linux GPU] または [Windows] を選択します。[Other registry (その他のレジストリ)] を選択した場合は、[External registry URL (外部のレジストリ URL)] に *docker repository*/*docker image name* の形式に従って Docker Hub の Docker イメージの名前とタグを入力します。Amazon ECR を選択した場合は、Amazon ECR リポジトリと Amazon ECR イメージを使用して、 AWS アカウントの Docker イメージを選択します。
- プライベート Docker イメージを使用するには、[カスタムイメージ] を選択します。[Environment type (環境タイプ)] で、 [ARM]、[Linux]、[Linux GPU] または [Windows] を選択します。[Image registry (イメージレジストリ)] に [Other registry (その他のレジストリ)] を

選択して、その後プライベート Docker イメージの認証情報の ARN を入力します。認証情報は、Secrets Manager で作成する必要があります。詳細については、AWS Secrets Manager ユーザーガイドの「AWS Secrets Managerとは」を参照してください。

> **Note**
>
> CodeBuild は、カスタム Docker イメージENTRYPOINTの を上書きします。

コンピューティング

次のいずれかを行います。

- EC2 コンピューティングを使用するには、EC2 を選択します。EC2 コンピューティングは、アクションの実行中に最適化された柔軟性を提供します。
- Lambda コンピューティングを使用するには、Lambda を選択します。Lambda コンピューティングは、ビルドの起動速度を最適化します。Lambda は、起動のレイテンシーが低いため、ビルドの高速化をサポートします。Lambda も自動的にスケーリングするため、ビルドはキュー内で実行を待機しません。詳細については、「での AWS Lambda コンピューティングの使用 AWS CodeBuild」を参照してください。

サービスロール

次のいずれかを行います。

- CodeBuild サービスロールがない場合は、新しいサービスロール を選択します。[Role name] に、新しいロールの名前を入力します。
- CodeBuild サービスロールがある場合は、既存のサービスロール を選択します。[Role ARN] で、サービスロールを選択します。

> **Note**
>
> コンソールを使用してビルドプロジェクトを作成すると、同時に CodeBuild サービスロールを作成できます。デフォルトでは、ロールはそのビルドプロジェクトでのみ使用できます。コンソールでは、このサービスロールを別のビルドプロジェクトと関連付けると、この別のビルドプロジェクトで使用できるようにロールが更新されます。サービスロールは最大 10 個のビルドプロジェクトで使用できます。

追加設定

タイムアウト

5 分から 8 時間の間の値を指定し、その後、完了しないとビルドを CodeBuild 停止します。[hours] と [minutes] を空白のままにすると、デフォルト値の 60 分が使用されます。

特権付与

(オプション) Docker イメージを構築する場合、またはこのビルドプロジェクトを使用して Docker イメージを構築する場合にのみ、ビルドで昇格された権限を取得したい場合は、このフラグを有効にするを選択します。それ以外の場合、関連付けられているビルドで Docker デーモンと通信しようとすると、すべて失敗します。ビルドが Docker デーモンと連係動作できるように、Docker デーモンも起動する必要があります。これを行う 1 つの方法は、次のビルドコマンドを実行してビルドスペックの `install` フェーズで Docker デーモンを初期化することです。Docker サポート CodeBuild で が提供するビルド環境イメージを選択した場合は、これらのコマンドを実行しないでください。

> **Note**
>
> デフォルトでは、Docker デーモンは VPC 以外のビルドで有効になっています。VPC ビルドに Docker コンテナを使用する場合は、Docker Docs ウェブサイトの「ランタイム特権と Linux 機能」を参照して、特権モードを有効にします。また、Windows は特権モードをサポートしていません。

```
- nohup /usr/local/bin/dockerd --host=unix:///var/run/docker.sock --
host=tcp://127.0.0.1:2375 --storage-driver=overlay2 &
- timeout 15 sh -c "until docker info; do echo .; sleep 1; done"
```

VPC

VPC CodeBuild を使用する場合 :

- VPC の場合、 が CodeBuild 使用する VPC ID を選択します。
- VPC サブネット で、 が CodeBuild 使用するリソースを含むサブネットを選択します。
- VPC セキュリティグループ で、 CodeBuild が VPCs内のリソースへのアクセスを許可するために使用するセキュリティグループを選択します。

詳細については、「Amazon Virtual Private Cloud AWS CodeBuild で を使用する」を参照してください。

コンピューティング

使用可能なオプションの 1 つを選択します。

環境変数

[環境変数] で、名前と値を入力してから、ビルドによって使用される各環境変数の種類を選択します。

> **Note**
>
> CodeBuild は、 AWS リージョンの環境変数を自動的に設定します。以下の環境変数を buildspec.yml に追加していない場合は、それらの変数を設定する必要があります。
>
> - AWS_ACCOUNT_ID
> - IMAGE_REPO_NAME
> - IMAGE_TAG

コンソールと AWS CLI ユーザーは環境変数を表示できます。環境変数の表示に懸念がない場合は、[Name] および [Value] フィールドを設定し、[Type] を [Plaintext] に設定します。

AWS アクセスキー ID、 AWS シークレットアクセスキー、パスワードなどの機密値を持つ環境変数をパラメータとして Amazon EC2 Systems Manager パラメータストアまたは に保存することをお勧めします AWS Secrets Manager。

Amazon EC2 Systems Manager パラメータストアを使用する場合は、[Type (タイプ)] で、[Parameter (パラメータ)] を選択します。名前に、参照 CodeBuild する の識別子を入力します。[Value] (値) に、Amazon EC2 Systems Manager パラメータストアに保存されているパラメータの名前を入力します。たとえば、`/CodeBuild/dockerLoginPassword` という名前のパラメータを使用して、[タイプ] で [Parameter (パラメータ)] を選択します。[Name (名前)] に `LOGIN_PASSWORD` と入力します。[Value (値)] に「`/CodeBuild/dockerLoginPassword`」と入力します。

> **Important**
>
> Amazon EC2 Systems Manager パラメータストアを使用する場合、パラメータは `/CodeBuild/` で始まるパラメータ名 (例: `/CodeBuild/dockerLoginPassword`) で保存することをお勧めします。 CodeBuild コンソールを使用して、Amazon EC2 Systems Manager でパラメータを作成できます。[パラメータの作成] を選択し、ダイ

アログボックスの手順に従います。(このダイアログボックスの KMS キー では、アカウントの AWS KMS キーの ARN を指定できます。 Amazon EC2 Systems Manager は、このキーを使用して、ストレージ中にパラメータの値を暗号化し、取得中に復号します)。 CodeBuild コンソールを使用してパラメータを作成する場合、コンソールは保存されているパラメータ名を /CodeBuild/ で開始します。詳細については、Amazon EC2 Systems Manager ユーザーガイドの「Systems Manager パラメータストア」および「Systems Manager パラメータストアコンソールのチュートリアル」を参照してください。
ビルドプロジェクトが Amazon EC2 Systems Manager パラメータストアに保存されているパラメータを参照する場合、ビルドプロジェクトのサービスロールで ssm:GetParameters アクションを許可する必要があります。以前に新しいサービスロールを選択した場合、 はビルドプロジェクトのデフォルトのサービスロールにこのアクション CodeBuild を含めます。ただし [既存のサービスロール] を選択した場合は、このアクションをサービスロールに個別に含める必要があります。
ビルドプロジェクトが、/CodeBuild/ で始まらないパラメータ名を持つ、Amazon EC2 Systems Manager パラメータストアに保存されているパラメータを参照し、[新しいサービスロール] を選択した場合、/CodeBuild/ で始まらないパラメータ名にアクセスできるようにサービスロールを更新する必要があります。これは、サービスロールで、/CodeBuild/ で始まるパラメータ名にのみアクセスが許可されるためです。
[新しいサービスロールを作成] を選択した場合、サービスロールには、Amazon EC2 Systems Manager パラメータストアの /CodeBuild/ 名前空間ですべてのパラメータを復号するアクセス権限が含まれます。
既存の環境変数は、設定した環境変数により置き換えられます。たとえば、Docker イメージに my_value の値を持つ MY_VAR という名前の環境変数が既に含まれていて、other_value の値を持つ MY_VAR という名前の環境変数を設定した場合、my_value が other_value に置き換えられます。同様に、Docker イメージに /usr/local/sbin:/usr/local/bin の値を持つ PATH という名前の環境変数が既に含まれていて、$PATH:/usr/share/ant/bin の値を持つ PATH という名前の環境変数を設定した場合、/usr/local/sbin:/usr/local/bin はリテラル値 $PATH:/usr/share/ant/bin に置き換えられます。
CODEBUILD_ で始まる名前の環境変数は設定しないでください。このプレフィックスは内部使用のために予約されています。
同じ名前の環境変数が複数の場所で定義されている場合は、その値は次のように決定されます。

- ビルド開始オペレーション呼び出しの値が最も優先順位が高くなります。

- ビルドプロジェクト定義の値が次に優先されます。
- ビルド仕様宣言の値の優先順位が最も低くなります。

Secrets Manager を使用する場合は、[Type] (タイプ) で、[Secrets Manager] を選択します。名前に、参照 CodeBuild する の識別子を入力します。[Value (値)] に、パターン `reference-key` を使用して *`secret-id`*`:`*`json-key`*`:`*`version-stage`*`:`*`version-id`* を入力します。詳細については、「Secrets Manager reference-key in the buildspec file」を参照してください。

> **Important**
>
> Secrets Manager を使用する場合は、「`/CodeBuild/`」で始まる名前でシークレットを保存することをお勧めします(たとえば、`/CodeBuild/dockerLoginPassword`)。詳細については、 AWS Secrets Managerユーザーガイドの「AWS Secrets Manager とは」を参照してください。
>
> ビルドプロジェクトが Secrets Manager パラメータストアに保存されているパラメータを参照する場合、ビルドプロジェクトのサービスロールで `secretsmanager:GetSecretValue` アクションを許可する必要があります。以前に新しいサービスロールを選択した場合、 はビルドプロジェクトのデフォルトのサービスロールにこのアクション CodeBuild を含めます。ただし [既存のサービスロール] を選択した場合は、このアクションをサービスロールに個別に含める必要があります。
>
> ビルドプロジェクトが、`/CodeBuild/` で始まらないパラメータ名を持つ、Secrets Manager に保存されているパラメータを参照し、[新しいサービスロール] を選択した場合、`/CodeBuild/` で始まらないシークレット名にアクセスできるようにサービスロールを更新する必要があります。これは、サービスロールで、`/CodeBuild/` で始まるシークレット名にのみアクセスが許可されるためです。
>
> [新しいサービスロール] を選択した場合、作成されるサービスロールには、Secrets Manager の `/CodeBuild/` 名前空間ですべてのシークレットを復号するアクセス許可が含まれます。

Buildspec

ビルド仕様

次のいずれかを行ってください。

- ソースコードにビルド仕様ファイルが含まれている場合は、[Use a buildspec file (buildspec ファイルを使用)] を選択します。デフォルトでは、 CodeBuild はソースコードのルートディレクトリbuildspec.ymlで という名前のファイルを検索します。buildspec ファイルに別の名前または場所が使用されている場合は、Buildspec 名 にソースルートからのパスを入力します (例えば、buildspec-two.yml または configuration/buildspec.yml)。buildspec ファイルが S3 バケットにある場合は、ビルドプロジェクトと同じ AWS リージョンに存在する必要があります。ARN を使用して buildspec ファイルを指定します (例: arn:aws:s3:::*<my-codebuild-sample2>*/buildspec.yml)。
- ソースコードにビルド仕様ファイルが含まれていない場合、または、ソースコードのルートディレクトリで build ファイルの buildspec.yml フェーズに指定されているものと異なるビルドコマンドを実行する場合は、[ビルドコマンドの挿入] を選択します。[ビルドコマンド] に、build フェーズで実行するコマンドを入力します。複数のコマンドについては、&& で各コマンドを区切ります (例: mvn test && mvn package)。他のフェーズでコマンドを実行する場合、または build フェーズのコマンドの長いリストがある場合は、ソースコマンドのルートディレクトリに buildspec.yml ファイルを追加し、ファイルにコマンドを追加してから、[Use the buildspec.yml in the source code root directory] (ソースコードのルートディレクトリの 「buildspec.yml」を使用) を選択します。

詳細については、「ビルド仕様 (buildspec) に関するリファレンス」を参照してください。

Batch 構成

ビルドのグループを 1 つの操作として実行できます。詳細については、「でのバッチビルド AWS CodeBuild」を参照してください。

バッチ構成の定義

このプロジェクトでバッチビルドを許可する場合に選択します。

Batch サービスロール

バッチビルドのサービスロールを提供します。

次のいずれかを選択します。

- バッチサービスロールがない場合は、[New service role] (新しいサービスロール) を選択します。[Service role] (サービスロール) に、新しいロールの名前を入力します。
- バッチサービスロールがある場合は、[Existing service role] (既存のサービスロール) を選択します。[Service role] (サービスロール) で、サービスロールを選択します。

バッチビルドでは、バッチ設定に新しいセキュリティロールが導入されます。この新しいロールは、バッチの一部としてビルドを実行するために、 がユーザーに代わって StartBuild、StopBuild、および RetryBuildアクションを呼び出すことができる CodeBuild 必要があるため必要です。次の2つの理由により、お客様はビルドで使用するものと同じロールではなく、新しいロールを使用する必要があります。

- ビルドの役割を与える StartBuild、StopBuild、および RetryBuild アクセス権限を使用すると、単一のビルドが buildspec を介してより多くのビルドを開始することができます。
- CodeBuild バッチビルドには、バッチ内のビルドに使用できるビルドとコンピューティングタイプの数を制限する制限があります。ビルドロールにこれらの権限がある場合、ビルド自体がこれらの制限を回避する可能性があります。

バッチに使用できる計算タイプ

バッチに使用できる計算タイプを選択します。該当するものをすべて選択します。

バッチで許可される最大ビルド

バッチで許可されるビルドの最大数を入力します。バッチがこの制限を超えると、バッチは失敗します。

バッチのタイムアウト

バッチビルドが完了する最大時間を入力します。

アーティファクトの結合

[Combine all artifacts from batch into a single location] (バッチのすべてのアーチファクト) を 1 つの場所に結合するを選択して、バッチのすべてのアーチファクトを単一の場所に結合します。

バッチレポートモード

バッチビルドに対して望ましいビルドステータスレポートモードを選択します。

> **Note**
>
> このフィールドは、プロジェクトソースが Bitbucket GitHub、、または GitHub Enterprise の場合にのみ使用でき、ビルドの開始と終了がソース で選択されたときにビルドステータスをソースプロバイダーに報告します。

集約されたビルド

これを選択して、バッチ内にあるすべてのビルドのステータスを単一のステータスレポートにまとめます。

個々のビルド

これを選択して、バッチ内にあるすべてのビルドのビルドステータスが個別に報告されるようにします。

アーティファクト

タイプ

次のいずれかを行ってください。

- ビルド出力アーティファクトを作成しない場合は、[No artifacts] を選択します。ビルドテストのみを実行している場合や、Docker イメージを Amazon ECR リポジトリにプッシュする場合には、これを行うことができます。
- ビルド出力を S3 バケットに保存する場合は、[Amazon S3] を選択して次のいずれかの操作を行います。
  - ビルド出力 ZIP ファイルまたはフォルダにプロジェクト名を使用する場合は、[Name (名前)] を空白のままにします。それ以外の場合は、名前を入力します。(ZIP ファイルを出力して ZIP ファイルにファイル拡張子を付ける場合は、必ず ZIP ファイル名の後に含めます)。
  - buildspec ファイルで指定した名前で、コンソールで指定した名前を上書きする場合は、[Enable semantic versioning (セマンティックバージョニングを有効にする)] を選択します。buildspec ファイル内の名前は、ビルド時に計算され、Shell コマンド言語を使用します。たとえば、アーティファクト名に日付と時刻を追加して常に一意にできます。アーティファクト名を一意にすると、アーティファクトが上書きされるのを防ぐことができます。詳細については、「buildspec の構文」を参照してください。
  - [Bucket name (バケット名)] で、出力バケットの名前を選択します。

- この手順の前の方で [ビルドコマンドの挿入] を選択した場合は、[出力ファイル] に、ビルド出力 ZIP ファイルまたはフォルダに格納するビルドのファイルの場所を入力します。複数の場所の場合は、各場所をコンマで区切ります (例: `appspec.yml, target/my-app.jar`)。詳細については、「`files`」で buildspec の構文 の説明を参照してください。
- ビルドアーティファクトを暗号化しない場合は、[アーティファクト暗号化の削除] を選択します。

アーティファクトのセカンダリセットごとに:

1. [Artifact 識別子] には、英数字とアンダースコアのみを使用して 128 文字未満の値を入力します。
2. [アーティファクトの追加] を選択します。
3. セカンダリアーティファクトを設定するには、前のステップに従います。
4. [アーティファクトの保存] を選択します。

追加設定

暗号化キー

次のいずれかを実行します。

- アカウントの Amazon S3 の AWS マネージドキー を使用してビルド出力アーティファクトを暗号化するには、[暗号化キー] を空白のままにします。これがデフォルトです。
- カスタマー管理のキーを使用してビルド出力アーティファクトを暗号化するには、[暗号化キー] に KMS キーの ARN を入力します。`arn:aws:kms:`*`region-ID`*`:`*`account-ID`*`:key/`*`key-ID`* の形式を使用します。

キャッシュタイプ

[キャッシュタイプ] で、以下のいずれかを選択します。

- キャッシュを使用しない場合は、[No cache] を選択します。
- Amazon S3 キャッシュを使用するには、[Amazon S3] を選択して次の操作を行います。
  - [バケット] では、キャッシュが保存される S3 バケットの名前を選択します。
  - (オプション) [Cache path prefix (キャッシュパスのプレフィックス)] に、Amazon S3 パスのプレフィックスを入力します。[キャッシュパスのプレフィックス] 値はディレクトリ名に似ています。これにより、バケット内の同じディレクトリにキャッシュを保存できます。

> **Important**
>
> パスのプレフィックスの末尾にスラッシュ (/) を付加しないでください。

- ローカルキャッシュを使用する場合は、[ローカル] を選択し、ローカルキャッシュモードを 1 つ以上選択します。

> **Note**
>
> Docker レイヤーキャッシュモードは Linux でのみ利用可能です。このモードを選択する場合、プロジェクトは権限モードで実行する必要があります。

キャッシュを使用すると、再利用可能なビルド環境がキャッシュに保存され、ビルド全体で使用されるため、かなりのビルド時間が節約されます。ビルド仕様ファイルのキャッシュの指定に関する詳細については、「buildspec の構文」を参照してください。キャッシングの詳細については、「AWS CodeBuild でのキャッシュのビルド」を参照してください。

ログ

作成するログを選択します。Amazon CloudWatch Logs、Amazon S3 ログ、またはその両方を作成できます。

CloudWatch

Amazon CloudWatch Logs ログが必要な場合：

CloudWatch ログ

CloudWatch ログを選択します。

グループ名

Amazon CloudWatch Logs ロググループの名前を入力します。

ストリーム名

Amazon CloudWatch Logs ログストリーム名を入力します。

S3

Amazon S3 ログが必要な場合は、以下のようになります。

S3 ログ

[S3 logs (S3 ログ)] を選択します。

バケット

ログを保存する S3 バケットの名前を選択します。

パスプレフィックス

ログのプレフィックスを入力します。

S3 ログの暗号化を無効にする

S3 ログを暗号化しない場合は、選択します。

ビルドプロジェクトの作成 (AWS CLI)

AWS CLI で を使用する方法の詳細については CodeBuild、「」を参照してくださいコマンドラインリファレンス。

を使用して CodeBuild ビルドプロジェクトを作成するには AWS CLI、JSON 形式のプロジェクト構造を作成し、構造を入力し、 `create-project` コマンドを呼び出してプロジェクトを作成します。

JSON ファイルの作成

`--generate-cli-skeleton` オプションを使用して、`create-project` コマンドでスケルトン JSON ファイルを作成します。

```
aws codebuild create-project --generate-cli-skeleton > <json-file>
```

これにより、*<json-file>* で指定されるパスとファイル名で JSON ファイルが作成されます。

JSON ファイルを入力します。

JSON データを次のように変更して、結果を保存します。

```
{
  "name": "<project-name>",
  "description": "<description>",
  "source": {
    "type": "CODECOMMIT" | "CODEPIPELINE" | "GITHUB" | "GITHUB_ENTERPRISE" | "GITLAB" |
 "GITLAB_SELF_MANAGED" | "BITBUCKET" | "S3" | "NO_SOURCE",
    "location": "<source-location>",
    "gitCloneDepth": "<git-clone-depth>",
```

```
    "buildspec": "<buildspec>",
    "InsecureSsl": "<insecure-ssl>",
    "reportBuildStatus": "<report-build-status>",
    "buildStatusConfig": {
      "context": "<context>",
      "targetUrl": "<target-url>"
    },
    "gitSubmodulesConfig": {
      "fetchSubmodules": "<fetch-submodules>"
    },
    "auth": {
      "type": "<auth-type>",
      "resource": "<auth-resource>"
    },
    "sourceIdentifier": "<source-identifier>"
  },
  "secondarySources": [
    {
        "type": "CODECOMMIT" | "CODEPIPELINE" | "GITHUB" | "GITHUB_ENTERPRISE" |
"GITLAB" | "GITLAB_SELF_MANAGED" | "BITBUCKET" | "S3" | "NO_SOURCE",
        "location": "<source-location>",
        "gitCloneDepth": "<git-clone-depth>",
        "buildspec": "<buildspec>",
        "InsecureSsl": "<insecure-ssl>",
        "reportBuildStatus": "<report-build-status>",
        "auth": {
          "type": "<auth-type>",
          "resource": "<auth-resource>"
        },
        "sourceIdentifier": "<source-identifier>"
    }
  ],
  "secondarySourceVersions": [
    {
      "sourceIdentifier": "<secondary-source-identifier>",
      "sourceVersion": "<secondary-source-version>"
    }
  ],
  "sourceVersion": "<source-version>",
  "artifacts": {
    "type": "CODEPIPELINE" | "S3" | "NO_ARTIFACTS",
    "location": "<artifacts-location>",
    "path": "<artifacts-path>",
    "namespaceType": "<artifacts-namespacetype>",
```

```
    "name": "<artifacts-name>",
    "overrideArtifactName": "<override-artifact-name>",
    "packaging": "<artifacts-packaging>"
  },
  "secondaryArtifacts": [
    {
      "type": "CODEPIPELINE" | "S3" | "NO_ARTIFACTS",
      "location": "<secondary-artifact-location>",
      "path": "<secondary-artifact-path>",
      "namespaceType": "<secondary-artifact-namespaceType>",
      "name": "<secondary-artifact-name>",
      "packaging": "<secondary-artifact-packaging>",
      "artifactIdentifier": "<secondary-artifact-identifier>"
    }
  ],
  "cache": {
    "type": "<cache-type>",
    "location": "<cache-location>",
    "mode": [
      "<cache-mode>"
    ]
  },
  "environment": {
    "type": "LINUX_CONTAINER" | "LINUX_GPU_CONTAINER" | "ARM_CONTAINER" |
"WINDOWS_SERVER_2019_CONTAINER" | "WINDOWS_SERVER_2022_CONTAINER",
    "image": "<image>",
    "computeType": "BUILD_GENERAL1_SMALL" | "BUILD_GENERAL1_MEDIUM" |
"BUILD_GENERAL1_LARGE" | "BUILD_GENERAL1_2XLARGE",
    "certificate": "<certificate>",
    "environmentVariables": [
      {
        "name": "<environmentVariable-name>",
        "value": "<environmentVariable-value>",
        "type": "<environmentVariable-type>"
      }
    ],
    "registryCredential": [
      {
        "credential": "<credential-arn-or-name>",
        "credentialProvider": "<credential-provider>"
      }
    ],
    "imagePullCredentialsType": "CODEBUILD" | "SERVICE_ROLE",
    "privilegedMode": "<privileged-mode>"
```

```
  },
  "serviceRole": "<service-role>",
  "timeoutInMinutes": <timeout>,
  "queuedTimeoutInMinutes": <queued-timeout>,
  "encryptionKey": "<encryption-key>",
  "tags": [
    {
      "key": "<tag-key>",
      "value": "<tag-value>"
    }
  ],
  "vpcConfig": {
    "securityGroupIds": [
         "<security-group-id>"
    ],
    "subnets": [
         "<subnet-id>"
    ],
    "vpcId": "<vpc-id>"
  },
  "badgeEnabled": "<badge-enabled>",
  "logsConfig": {
    "cloudWatchLogs": {
      "status": "<cloudwatch-logs-status>",
      "groupName": "<group-name>",
      "streamName": "<stream-name>"
    },
    "s3Logs": {
      "status": "<s3-logs-status>",
      "location": "<s3-logs-location>",
      "encryptionDisabled": "<s3-logs-encryption-disabled>"
    }
  },
  "fileSystemLocations": [
    {
      "type": "EFS",
      "location": "<EFS-DNS-name-1>:/<directory-path>",
      "mountPoint": "<mount-point>",
      "identifier": "<efs-identifier>",
      "mountOptions": "<efs-mount-options>"
    }
  ],
  "buildBatchConfig": {
    "serviceRole": "<batch-service-role>",
```

```
    "combineArtifacts": <combine-artifacts>,
    "restrictions": {
      "maximumBuildsAllowed": <max-builds>,
      "computeTypesAllowed": [
        "<compute-type>"
      ]
    },
    "timeoutInMins": <batch-timeout>,
    "batchReportMode": "REPORT_AGGREGATED_BATCH" | "REPORT_INDIVIDUAL_BUILDS"
  },
  "concurrentBuildLimit": <concurrent-build-limit>
}
```

以下に置き換えます。

name

必須。このビルドプロジェクトの名前。この名前は、 AWS アカウント内のすべてのビルドプロジェクトで一意である必要があります。

[Description] (説明)

オプション。このビルドの説明。

source

必須。このビルドプロジェクトのソースコード設定に関する情報を含む ProjectSource オブジェクト。source オブジェクトを追加したら、 を使用して最大 12 個のソースを追加できます。これらの設定には以下が含まれます。

source/type

必須。ビルドするソースコードを含むリポジトリのタイプ。有効な値を次に示します。

- CODECOMMIT
- CODEPIPELINE
- GITHUB
- GITHUB_ENTERPRISE
- GITLAB
- GITLAB_SELF_MANAGED

- BITBUCKET
- S3
- NO_SOURCE

NO_SOURCE を使用すると、プロジェクトにはソースがないため、buildspec をファイルとして使用できません。代わりに、buildspec 属性を使用して buildspec に YAML 形式の文字列を指定する必要があります。詳細については、「ソースサンプルがないプロジェクト」を参照してください。

source/location

<source-type> を CODEPIPELINE に設定しない場合は必須です。指定されたリポジトリタイプのソースコードの場所。

- の場合 CodeCommit、ソースコードと buildspec ファイルを含むリポジトリへの HTTPS クローン URL (例: https://git-codecommit.*<region-id>*.amazonaws.com/v1/repos/*<repo-name>*) 。
- Amazon S3 では、ビルド入力バケット名の後に、ソースコードと buildspec を含む ZIP ファイルのパスと名前が続きます。次に例を示します。
  - 入力バケットのルートにある ZIP ファイルの場合: *<bucket-name>*/*<object-name>*.zip。
  - 入力バケットのサブフォルダーにある ZIP ファイルの場合: *<bucket-name>*/*<subfoler-path>*/*<object-name>*.zip。
- の場合 GitHub、ソースコードと buildspec ファイルを含むリポジトリへの HTTPS クローン URL。URL には github.com が含まれている必要があります。 AWS アカウントを GitHub アカウントに接続する必要があります。これを行うには、 CodeBuild コンソールを使用してビルドプロジェクトを作成します。
  - [Authorize application] を選択します。(GitHub アカウントに接続したら、ビルドプロジェクトの作成を完了する必要はありません。 CodeBuild コンソールを閉じることができます) 。
- GitHub Enterprise Server の場合、ソースコードと buildspec ファイルを含むリポジトリへの HTTP または HTTPS クローン URL。また、 AWS アカウントを GitHub Enterprise Server アカウントに接続する必要があります。これを行うには、 CodeBuild コンソールを使用してビルドプロジェクトを作成します。
  1. GitHub Enterprise Server で個人用アクセストークンを作成します。
  2. このトークンをクリップボードにコピーして、 CodeBuild プロジェクトの作成時に使用できます。詳細については、ヘルプウェブサイトの「コマンドライン用の個人用アクセストークンの作成」を参照してください。 GitHub

3. コンソールを使用して CodeBuild プロジェクトを作成する場合は、ソース でソースプロバイダー にGitHubエンタープライズ を選択します。
4. [個人用アクセストークン] には、クリップボードにコピーしたトークンを貼り付けます。[トークンの保存] を選択します。これで、 CodeBuild アカウントが GitHub Enterprise Server アカウントに接続されました。

- GitLab および GitLab セルフマネージド型の場合、ソースコードと buildspec ファイルを含むリポジトリへの HTTPS クローン URL。を使用する場合 GitLab、URL には gitlab.com が含まれている必要があります。 GitLab セルフマネージド を使用する場合、URL に gitlab.com を含める必要はありません。 AWS アカウントを または GitLab 自己管理型 GitLabアカウントに接続する必要があります。これを行うには、 CodeBuildコンソールを使用してビルドプロジェクトを作成します。
  - デベロッパーツールナビゲーションペインで、設定 、接続 を選択し、接続 を作成します。このページで、 GitLab または GitLab セルフマネージド型の接続を作成し、Connect to GitLabを選択します。
- Bitbucket の場合は、ソースコードと buildspec ファイルが格納されているリポジトリへの HTTPS クローン URL。URL には bitbucket.org が含まれている必要があります。また、 AWS アカウントを Bitbucket アカウントに接続する必要があります。これを行うには、 CodeBuild コンソールを使用してビルドプロジェクトを作成します。
  1. コンソールを使用して Bitbucket に接続 (または再接続) する場合は、Bitbucket の [Confirm access to your account] ページで、[Grant access] を選択します (Bitbucket アカウントに接続したら、ビルドプロジェクトの作成を完了する必要はありません。 CodeBuild コンソールを閉じることができます)。
- では AWS CodePipeline、 `location`の値を指定しないでください`source`。 CodePipeline でパイプラインを作成するときは CodePipeline、パイプラインのソースステージでソースコードの場所を指定するためです。

ソース/gitCloneDepth

オプション。ダウンロードする履歴の深さ。最小値は 0 です。この値が 0、あるいは 25 より大きいか指定されていない場合、完全な履歴が各ビルドプロジェクトと共にダウンロードされます。ソースタイプが Amazon S3 である場合、この値はサポートされません。

source/buildspec

オプション。使用するビルド仕様定義またはファイル。この値が指定されていない場合や、空の文字列に設定されている場合、ソースコードのルートディレクトリに `buildspec.yml` ファイルが含まれている必要があります。この値が設定されている場合は、インラインのビルド仕様定義か、プライマリソースのルートディレクトリからの相対的な代替 buildspec ファ

イルへのパス、S3 バケットへのパスになります。バケットは、ビルドプロジェクトと同じ AWS リージョンに存在する必要があります。ARN を使用して buildspec ファイルを指定します（例: `arn:aws:s3:::<my-codebuild-sample2>/buildspec.yml`）。詳細については、「buildspec ファイル名とストレージの場所」を参照してください。

source/auth

使用しません。このオブジェクトは CodeBuild コンソールでのみ使用されます。

ソース/reportBuildStatus

ビルドの開始と完了のステータスをソースプロバイダーに送信するかどうかを指定します。これを GitHub、 GitHub Enterprise Server、または Bitbucket 以外のソースプロバイダーで設定すると、 `invalidInputException`がスローされます。

ソースプロバイダにビルド状態を報告できるようにするには、ソースプロバイダに関連付けられたユーザーがリポジトリへの書き込みアクセス権を持っている必要があります。ユーザーが書き込みアクセス権を持っていない場合、ビルドのステータスは更新できません。詳細については、「ソースプロバイダーのアクセス」を参照してください。

ソース/buildStatusConfig

CodeBuild ビルドプロジェクトがソースプロバイダーにビルドステータスを報告する方法を定義する情報が含まれています。このオプションは、ソースタイプが `GITHUB`、`GITHUB_ENTERPRISE`、または `BITBUCKET` の場合にのみ使用されます。

source/buildStatusConfig/context

Bitbucket リソースでは、このパラメータは、Bitbucket コミットステータスの `name` パラメータに使用されます。 GitHub ソースの場合、このパラメータは GitHub コミットステータスの `context`パラメータに使用されます。

例えば、 に環境変数を使用して CodeBuildビルド番号とウェブフックトリガー`context`を含めることができます。

```
AWS CodeBuild sample-project Build #$CODEBUILD_BUILD_NUMBER -
 $CODEBUILD_WEBHOOK_TRIGGER
```

これにより、webhook プルリクエストイベントによってトリガーされた build #24 では、コンテキストは次のようになります。

```
AWS CodeBuild sample-project Build #24 - pr/8
```

source/buildStatusConfig/targetUrl

Bitbucket リソースでは、このパラメータは、Bitbucket コミットステータスの `url` パラメータに使用されます。 GitHub ソースの場合、このパラメータは GitHub コミットステータスの `target_url`パラメータに使用されます。

たとえば、「`targetUrl`」と「`https://aws.amazon.com/codebuild/<path to build>`」とコミットステータスをこのURLにリンクします。

に CodeBuild 環境変数を含めて`targetUrl`、URL に情報を追加することもできます。例えば、ビルド領域を URL に追加するには、`targetUrl` を以下に設定します:

```
"targetUrl": "https://aws.amazon.com/codebuild/<path to build>?region=
$AWS_REGION"
```

ビルド領域が `us-east-2` の場合、これは次のように展開されます。

```
https://aws.amazon.com/codebuild/<path to build>?region=us-east-2
```

ソース/gitSubmodulesConfig

オプション。Git サブモジュール設定に関する情報。 CodeCommit、 GitHub Enterprise Server GitHub、Bitbucket でのみ使用されます。

source/gitSubmodulesConfig/fetchSubmodules

リポジトリに Git サブモジュールを含める場合は、`fetchSubmodules` を `true` に設定します。含まれている Git サブモジュールは HTTPS として設定する必要があります。

ソース/InsecureSsl

オプション。 GitHub Enterprise Server でのみ使用されます。Enterprise Server プロジェクトリポジトリへの接続中に TLS 警告を無視`true`するには、この値を GitHub に設定します。デフォルト値は `false` です。`InsecureSsl` は、テスト目的でのみ使用してください。本番環境では使用しないでください。

source/sourceIdentifier

プロジェクトソースのユーザー定義識別子。プライマリソースの場合、省略可能です。セカンダリソースでは必須です。

secondarySources

オプション。ビルドプロジェクトのセカンダリソースに関する情報を含む ProjectSource オブジェクトの配列。最大 12 個のセカンダリソースを追加できます。secondarySources オブジェクトは、 オブジェクトで使用されるのと同じプロパティを使用します。セカンダリソースオブジェクトでは、sourceIdentifier は必須です。

secondarySourceVersions

オプション。ProjectSourceVersion オブジェクトの配列。secondarySourceVersions をビルドレベルで指定すると、これよりも優先されます。

sourceVersion

オプション。このプロジェクト用に構築するビルド入力のバージョン。指定しない場合、最新のバージョンが使用されます。指定した場合、次のいずれかであることが必要です。

- の場合 CodeCommit、使用するコミット ID、ブランチ、または Git タグ。
- の場合 GitHub、ビルドするソースコードのバージョンに対応するコミット ID、プルリクエスト ID、ブランチ名、またはタグ名。プルリクエスト ID を指定する場合、pr/pull-request-ID (例: pr/25) 形式を使用する必要があります。ブランチ名を指定すると、ブランチの HEAD コミット ID が使用されます。指定しない場合は、デフォルトブランチの HEAD コミット ID が使用されます。
- の場合 GitLab、コミット ID、プルリクエスト ID、ブランチ名、タグ名、またはリファレンスとコミット ID。詳細については、「のソースバージョンサンプル AWS CodeBuild」を参照してください。
- Bitbucket の場合、ビルドするソースコードのバージョンに対応するコミット ID、ブランチ名、またはタグ名。ブランチ名を指定すると、ブランチの HEAD コミット ID が使用されます。指定しない場合は、デフォルトブランチの HEAD コミット ID が使用されます。
- Amazon S3 の場合、使用するビルド入力 ZIP ファイルを表すオブジェクトのバージョン ID。

sourceVersion をビルドレベルで指定した場合、そのバージョンはこの (プロジェクトレベルの) sourceVersion より優先されます。詳細については、「のソースバージョンサンプル AWS CodeBuild」を参照してください。

artifacts

必須。このビルドプロジェクトの出力アーティファクト設定に関する情報を含む ProjectArtifacts オブジェクト。artifacts オブジェクトを追加したら、 を使用して最大 12 個のアーティファクトを追加できます。これらの設定には以下が含まれます。

artifacts/type

必須。ビルド出力アーティファクトのタイプ。有効な値は次のとおりです。

- CODEPIPELINE
- NO_ARTIFACTS
- S3

artifacts/location

S3 アーティファクトタイプでのみ使用されます。他のアーティファクトタイプには使用されません。

前提条件で作成または識別した出力バケットの名前。

artifacts/path

S3 アーティファクトタイプでのみ使用されます。他のアーティファクトタイプには使用されません。

ZIP ファイルまたはフォルダを配置する出力バケットのパス。の値を指定しない場合path、は namespaceType (指定されている場合) と CodeBuild を使用して、ビルド出力 ZIP ファイルまたはフォルダのパスと名前nameを決定します。たとえば、MyPath を path に、MyArtifact.zip に name 指定すると、パスと名前は「MyPath/MyArtifact.zip」になります。

artifacts/namespaceType

S3 アーティファクトタイプでのみ使用されます。他のアーティファクトタイプには使用されません。

ビルド出力 ZIP ファイルまたはフォルダの名前空間。有効な値は、BUILD_ID および NONE です。BUILD_ID を使用してビルド出力 ZIP ファイルまたはフォルダのパスにビルド ID を挿入します。それ以外の場合は、NONE を使用します。の値を指定しない場合namespaceType、は path (指定されている場合) と CodeBuild を使用して、ビルド出力 ZIP ファイルまたは

フォルダのパスと名前nameを決定します。たとえば、MyPath を path に、BUILD_ID を namespaceType、MyArtifact.zip に name 指定すると、パスと名前は「MyPath/*build-ID*/MyArtifact.zip」になります。

artifacts/name

S3 アーティファクトタイプでのみ使用されます。他のアーティファクトタイプには使用されません。

location 内のビルド出力 ZIP ファイルまたはフォルダの名前。たとえば、MyPath を path に、MyArtifact.zip に name 指定すると、パスと名前は「MyPath/MyArtifact.zip」になります。

artifacts/overrideArtifactName

S3 アーティファクトタイプ でのみ使用されます。他のアーティファクトタイプには使用されません。

オプション。true に設定すると、buildspec ファイルの artifacts ブロックで指定された名前が、name を上書きします。詳細については、「のビルド仕様リファレンス CodeBuild」を参照してください。

artifacts/packaging

S3 アーティファクトタイプでのみ使用されます。他のアーティファクトタイプには使用されません。

オプション。アーティファクトをパッケージ化する方法を指定します。許可された値は次のとおりです:

なし

ビルドアーティファクトを含むフォルダを作成します。これは、デフォルト値です。

ZIP

ビルドアーティファクトを含む ZIP ファイルを作成します。

secondaryArtifacts

オプション。ビルドプロジェクトのセカンダリアーティファクト設定に関する情報を含む ProjectArtifacts オブジェクトの配列。最大 12 個のセカンダリアーティファクトを追加できま

す。secondaryArtifacts は、 オブジェクトで使用されているのと同じ設定の多くを使用します。

cache

必須。このビルドプロジェクトのキャッシュ設定に関する情報を含む ProjectCache オブジェクト。詳細については、「キャッシュのビルド」を参照してください。

環境

必須。このプロジェクトのビルド環境設定に関する情報を含む ProjectEnvironment オブジェクト。設定は次のとおりです。

environment/type

必須。構築環境のタイプ。詳細については、 CodeBuild API リファレンスのhttps://docs.aws.amazon.com/codebuild/latest/APIReference/API_ProjectEnvironment.html#CodeBuild-Type-ProjectEnvironment-type「type」を参照してください。

environment/image

必須。このビルド環境で使用される Docker イメージ識別子。通常、この識別子は *image-name*:*tag* として表されます。例えば、 が Docker イメージの管理 CodeBuild に使用する Docker リポジトリでは、これは になりますaws/codebuild/standard:5.0。Docker Hub では、maven:3.3.9-jdk-8 です。Amazon ECR では、*account-id*.dkr.ecr.*region-id*.amazonaws.com/*your-Amazon-ECR-repo-name*:*tag* です。詳細については、「が提供する Docker イメージ CodeBuild」を参照してください。

environment/computeType

必須。このビルド環境で使用されるコンピュートリソースを指定します。詳細については、CodeBuild API リファレンスの「computeType」を参照してください。

environment/certificate

オプション。Amazon S3 バケットの ARN、パスのプレフィックス、および PEM エンコードされた証明書を含むオブジェクトキー。オブジェクトキーとして、PEM エンコードされた証明書が含まれている .pem ファイルまたは .zip ファイルのいずれかを使用できます。たとえば、Amazon S3 バケット名が *<my-bucket>*、パスのプレフィックスが *<cert>*、オブジェクトキー名が *<certificate.pem>* である場合、certificate に使用できる形式は *<my-bucket/cert/certificate.pem>* または arn:aws:s3:::*<my-bucket/cert/certificate.pem>* です。

environment/environmentVariables

オプション。このビルド環境に指定する環境変数を含む EnvironmentVariable オブジェクトの配列。各環境変数は、オブジェクトとして表されます。name、value、および type の name、value、 および type。

コンソールと AWS CLI ユーザーは、すべての環境変数を表示できます。環境変数の表示に懸念がない場合は、「name」を「value」および 「type」を「PLAINTEXT」に設定します。

Amazon EC2 Systems Manager パラメータストアまたは には、 AWS アクセスキー ID、 AWS シークレットアクセスキー、パスワードなどの機密値を持つ環境変数をパラメータとして保存することをお勧めします AWS Secrets Manager。の場合name、そのストアドパラメータに、参照 CodeBuild する の識別子を設定します。

Amazon EC2 Systems Manager パラメータストアを使用する場合、value には、パラメータストアに保存されているとおりにパラメータの名前を設定します。type を PARAMETER_STORE に設定します。/CodeBuild/dockerLoginPassword という名前のパラメータを使用するには、たとえば、「name」を「LOGIN_PASSWORD」に設定。value を /CodeBuild/dockerLoginPassword に設定します。type を PARAMETER_STORE に設定します。

> **Important**
>
> Amazon EC2 Systems Manager パラメータストアを使用する場合、パラメータは /CodeBuild/ で始まるパラメータ名 (例: /CodeBuild/dockerLoginPassword) で保存することをお勧めします。 CodeBuild コンソールを使用して、Amazon EC2 Systems Manager でパラメータを作成できます。[パラメータの作成] を選択し、ダイアログボックスの手順に従います。 (このダイアログボックスの KMS キー では、アカウントの AWS KMS キーの ARN を指定できます。 Amazon EC2 Systems Manager は、このキーを使用して、ストレージ中にパラメータの値を暗号化し、取得中に復号します) 。CodeBuild コンソールを使用してパラメータを作成する場合、コンソールは保存されているパラメータ名を /CodeBuild/ で開始します。詳細については、Amazon EC2 Systems Manager ユーザーガイドの「Systems Manager パラメータストア」および「Systems Manager パラメータストアコンソールのチュートリアル」を参照してください。
> ビルドプロジェクトが Amazon EC2 Systems Manager パラメータストアに保存されているパラメータを参照する場合、ビルドプロジェクトのサービスロールで ssm:GetParameters アクションを許可する必要があります。以前に新しいサービスロールを選択した場合、 はビルドプロジェクトのデフォルトのサービスロールにこのア

クション CodeBuild を含めます。ただし [既存のサービスロール] を選択した場合は、このアクションをサービスロールに個別に含める必要があります。
ビルドプロジェクトが、/CodeBuild/ で始まらないパラメータ名を持つ、Amazon EC2 Systems Manager パラメータストアに保存されているパラメータを参照し、[新しいサービスロール] を選択した場合、/CodeBuild/ で始まらないパラメータ名にアクセスできるようにサービスロールを更新する必要があります。これは、サービスロールで、/CodeBuild/ で始まるパラメータ名にのみアクセスが許可されるためです。
[新しいサービスロールを作成] を選択した場合、サービスロールには、Amazon EC2 Systems Manager パラメータストアの /CodeBuild/ 名前空間ですべてのパラメータを復号するアクセス権限が含まれます。
既存の環境変数は、設定した環境変数により置き換えられます。たとえば、Docker イメージに my_value の値を持つ MY_VAR という名前の環境変数が既に含まれていて、other_value の値を持つ MY_VAR という名前の環境変数を設定した場合、my_value が other_value に置き換えられます。同様に、Docker イメージに /usr/local/sbin:/usr/local/bin の値を持つ PATH という名前の環境変数が既に含まれていて、$PATH:/usr/share/ant/bin の値を持つ PATH という名前の環境変数を設定した場合、/usr/local/sbin:/usr/local/bin はリテラル値 $PATH:/usr/share/ant/bin に置き換えられます。
CODEBUILD_ で始まる名前の環境変数は設定しないでください。このプレフィックスは内部使用のために予約されています。
同じ名前の環境変数が複数の場所で定義されている場合は、その値は次のように決定されます。

- ビルド開始オペレーション呼び出しの値が最も優先順位が高くなります。
- ビルドプロジェクト定義の値が次に優先されます。
- ビルド仕様宣言の値の優先順位が最も低くなります。

Secrets Manager を使用する場合、value には、Secrets Manager に保存されているパラメータの名前を設定します。type を SECRETS_MANAGER に設定します。/CodeBuild/dockerLoginPassword という名前のシークレットを使用するには、たとえば、「name」を「LOGIN_PASSWORD」に設定。value を /CodeBuild/dockerLoginPassword に設定します。type を SECRETS_MANAGER に設定します。

> **Important**
>
> Secrets Manager を使用する場合は、「/CodeBuild/」で始まる名前でシークレットを保存することをお勧めします(たとえば、/CodeBuild/dockerLoginPassword)。詳細については、 AWS Secrets Managerユーザーガイドの「AWS Secrets Manager とは」を参照してください。
> ビルドプロジェクトが Secrets Manager パラメータストアに保存されているパラメータを参照する場合、ビルドプロジェクトのサービスロールで secretsmanager:GetSecretValue アクションを許可する必要があります。以前に新しいサービスロールを選択した場合、 はビルドプロジェクトのデフォルトのサービスロールにこのアクション CodeBuild を含めます。ただし [既存のサービスロール] を選択した場合は、このアクションをサービスロールに個別に含める必要があります。
> ビルドプロジェクトが、/CodeBuild/ で始まらないパラメータ名を持つ、Secrets Manager に保存されているパラメータを参照し、[新しいサービスロール] を選択した場合、/CodeBuild/ で始まらないシークレット名にアクセスできるようにサービスロールを更新する必要があります。これは、サービスロールで、/CodeBuild/ で始まるシークレット名にのみアクセスが許可されるためです。
> [新しいサービスロール] を選択した場合、作成されるサービスロールには、Secrets Manager の /CodeBuild/ 名前空間ですべてのシークレットを復号するアクセス許可が含まれます。

environment/registryCredential

オプション。プライベート Docker レジストリへのアクセスを提供する認証情報を指定する RegistryCredential オブジェクト。

environment/registryCredential/credential

AWS Managed Servicesを使用して作成された認証情報の ARN または名前を指定します。認証情報の名前を使用できるのは、認証情報が現在のリージョン内に存在する場合のみです。

environment/registryCredential/credentialProvider

唯一の有効な値は SECRETS_MANAGER です。

これを設定した場合:

- imagePullCredentials を SERVICE_ROLE に設定する必要があります。
- 選別されたイメージや Amazon ECR イメージは使用できません。

環境/imagePullCredentialsタイプ

オプション。ビルドでイメージをプルするために CodeBuild が使用する認証情報のタイプ。2 つの有効な値があります。

CODEBUILD

`CODEBUILD` は、 が独自の認証情報 CodeBuild を使用することを指定します。 CodeBuild サービスプリンシパルを信頼するには、Amazon ECR リポジトリポリシーを編集する必要があります。

SERVICE_ROLE

がビルドプロジェクトのサービスロール CodeBuild を使用するように指定します。

クロスアカウントまたはプライベートレジストリイメージを使用する場合は、`SERVICE_ROLE` の認証情報を使用する必要があります。 CodeBuild キュレートされたイメージを使用する場合は、`CODEBUILD`認証情報を使用する必要があります。

environment/privilegedMode

このビルドプロジェクトを使用して Docker イメージをビルドする計画の場合のみ、`true` に設定します。それ以外の場合、関連付けられているビルドで Docker デーモンと通信しようとすると、すべて失敗します。ビルドが Docker デーモンと連係動作できるように、Docker デーモンも起動する必要があります。これを行う 1 つの方法は、次のビルドコマンドを実行して buildspec ファイルの `install` フェーズで Docker デーモンを初期化することです。Docker サポート CodeBuild で が提供するビルド環境イメージを指定した場合は、これらのコマンドを実行しないでください。

> **Note**
>
> デフォルトでは、Docker デーモンは VPC 以外のビルドで有効になっています。VPC ビルドに Docker コンテナを使用する場合は、Docker Docs ウェブサイトの「ランタイム特権と Linux 機能」を参照して、特権モードを有効にします。また、Windows は特権モードをサポートしていません。

```
- nohup /usr/local/bin/dockerd --host=unix:///var/run/docker.sock --
host=tcp://127.0.0.1:2375 --storage-driver=overlay2 &
- timeout 15 sh -c "until docker info; do echo .; sleep 1; done"
```

serviceRole

必須。サービスロールの ARN は、 CodeBuild を使用してユーザーに代わってサービスとやり取りします (例: `arn:aws:iam::account-id:role/role-name`)。

timeoutInMinutes

オプション。5～480 (8 時間) の分数。完了していない場合、 CodeBuildはビルドを停止します。指定しない場合は、デフォルトの 60 が使用されます。タイムアウトが原因でビルドが CodeBuild 停止したかどうか、およびいつ停止したかを確認するには、 `batch-get-builds` コマンドを実行します。ビルドが停止しているかどうかを確認するには、出力で `FAILED` の `buildStatus` 値を調べます。ビルドがタイムアウトした時間を確認するには、 出力で `TIMED_OUT` の`phaseStatus` 値に関連付けられている `endTime` 値を調べます。

queuedTimeoutIn分

オプション。5 ～ 480 (8 時間) の分数。この時間が経過すると CodeBuild、まだキューに入っている場合はビルドが停止します。指定しない場合は、デフォルトの 60 が使用されます。

encryptionKey

オプション。ビルド出力を暗号化 CodeBuild するために AWS KMS key で使用される のエイリアスまたは ARN。エイリアスを指定する場合に、`arn:aws:kms:region-ID:account-ID:key/key-ID` 形式を使用し、エイリアスが存在する場合には、`alias/key-alias` 形式を使用します。指定しない場合、Amazon AWS S3 の マネージド KMS キーが使用されます。 Amazon S3

タグ

オプション。このビルドプロジェクトに関連付けるタグを提供する Tag オブジェクトの配列。最大 50 個のタグを指定できます。これらのタグは、 CodeBuild ビルドプロジェクトタグをサポートする任意の AWS サービスで使用できます。各タグは、「`key`」と「`value`」オブジェクトとして表現されます。

vpcConfig

オプション。プロジェクトの VPC 設定に関する情報を含む VpcConfig オブジェクト。詳細については、「Amazon Virtual Private Cloud AWS CodeBuild で を使用する」を参照してください。

これらのプロパティには、次のものがあります。

vpcId

必須。が CodeBuild 使用する VPC ID。リージョン内の VPC ID を一覧表示するには、次のコマンドを実行します。

```
aws ec2 describe-vpcs --region <region-ID>
```

サブネット

必須。で使用されるリソースを含むサブネット IDs の配列 CodeBuild。これらの ID を取得するには、次のコマンドを実行します。

```
aws ec2 describe-subnets --filters "Name=vpc-id,Values=<vpc-id>" --region <region-ID>
```

securityGroupIds

必須。VPC 内のリソースへのアクセスを許可 CodeBuild するために で使用されるセキュリティグループ IDs の配列。これらの ID を取得するには、次のコマンドを実行します。

```
aws ec2 describe-security-groups --filters "Name=vpc-id,Values=<vpc-id>" --<region-ID>
```

badgeEnabled

オプション。ビルドバッジを CodeBuild プロジェクトに含めるかどうかを指定します。true に設定してビルドバッジを有効にするか、そうでない場合は false に設定します。詳細については、「でビルドバッジのサンプル CodeBuild」を参照してください。

logsConfig

このビルドのログの場所に関する情報を含む LogsConfig オブジェクト。

logsConfigcloudWatchLogs/

CloudWatch Logs へのログのプッシュに関する情報を含む CloudWatchLogsConfig オブジェクト。

logsConfig/s3Logs

Amazon S3LogsConfig へのログのプッシュに関する情報を含む S3 オブジェクト。 Amazon S3

fileSystemLocations

オプション。Amazon EFS 設定に関する情報を含む ProjectFileSystemsLocation オブジェクトの配列。

buildBatchConfig

オプション。buildBatchConfig オブジェクトは、プロジェクトのバッチビルド設定情報を含むProjectBuildBatchConfig構造です。

buildBatchConfig/serviceRole

バッチビルドプロジェクトのサービスロール ARN を指定します。

buildBatchConfig/combineArtifacts

バッチビルドのビルドアーティファクトを 1 つのアーティファクトの場所に結合するかどうかを指定するブール値。

buildBatchConfig/restrictions/maximumBuildsAllowed

許可されるビルドの最大数。

buildBatchConfig/restrictions/computeTypesAllowed

バッチビルドで許可されるコンピューティングタイプを指定する文字列の配列。これらの値については、「ビルド環境のコンピューティングタイプ」を参照してください。

buildBatchConfig/timeoutInMinutes

バッチビルドを完了するまでの最大時間 (分単位) 。

buildBatchConfig/batchReportMode

バッチビルドのソースプロバイダーにビルドステータスレポートを送信する方法を指定します。有効な値を次に示します。

REPORT_AGGREGATED_BATCH

(デフォルト) すべてのビルドステータスを 1 つのステータスレポートに集約します。

REPORT_INDIVIDUAL_BUILDS

個々のビルドごとに個別のステータスレポートを送信します。

concurrentBuildLimit

このジョブで許可される同時実行の最大数を設定します。

新しいビルドは、現在のビルド数がこの制限以下の場合にのみ開始されます。現在のビルドカウントがこの制限を満たす場合、新しいビルドはスロットルされ、実行されません。

プロジェクトの作成

プロジェクトを作成するには、**create-project** コマンドを再度実行し、JSON ファイルを渡します。

```
aws codebuild create-project --cli-input-json file://<json-file>
```

成功した場合、JSON 表現の Project オブジェクトが、コンソール出力に表示されます。このデータの例については、CreateProject 「レスポンス構文」を参照してください。

ビルドプロジェクトの名前を除いて、後でビルドプロジェクトの設定を変更することができます。詳細については、「ビルドプロジェクトの設定の変更 (AWS CLI)」を参照してください。

ビルドの実行を開始するには、「ビルドの実行 (AWS CLI)」を参照してください。

ソースコードが GitHub リポジトリに保存されており、コード変更がリポジトリにプッシュされるたびにソースコードを CodeBuild 再構築する場合は、「」を参照してくださいビルドの実行の自動開始 (AWS CLI)。

ビルドプロジェクトの作成 (AWS SDK)

AWS CodeBuild を AWS SDK と組み合わせて使用する方法については、「AWS SDK とツールのリファレンス」を参照してください。

ビルドプロジェクトの作成 (AWS CloudFormation)

AWS CloudFormation での AWS CodeBuild の使用については、AWS CloudFormation ユーザーガイドの「CodeBuild の AWS CloudFormation テンプレート」を参照してください。

通知ルールの作成

通知ルールを使用すると、ビルドの成功や失敗などの重要な変更が発生したときにユーザーに通知できます。通知ルールは、イベントと、通知の送信に使用される Amazon SNS トピックの両方を指定します。詳細については、「通知とは」を参照してください。

コンソールまたは AWS CLI を使用して AWS CodeBuild の通知ルールを作成できます。

通知ルールを作成するには (コンソール)

1. AWS Management Console にサインインして、CodeBuild コンソール (https://console.aws.amazon.com/codebuild/) を開きます。
2. [Build (ビルド)]、[Build projects (ビルドプロジェクト)] の順に選択し、通知を追加するビルドプロジェクトを選択します。
3. ビルドプロジェクトページで、[Notify (通知)]、[Create notification rule (通知ルールの作成)] の順に選択します。ビルドプロジェクトの [Settings (設定)] ページに移動し、[Create notification rule (通知ルールの作成)] を選択することもできます。
4. [通知名] に、ルールの名前を入力します。
5. Amazon EventBridge に提供された情報のみを通知に含める場合は、[Detail type (詳細タイプ)] で [Basic (基本)] を選択します。Amazon EventBridge に提供される情報に加えて、CodeBuild または通知マネージャから提供される場合がある情報も含める場合は、[完全] を選択します。

   詳細については、「通知の内容とセキュリティについて」を参照してください。
6. [Events that trigger notifications (通知をトリガーするイベント)] で、通知を送信するイベントを選択します。詳細については、「ビルドプロジェクトでの通知ルールのイベント」を参照してください。
7. [Targets (ターゲット)] で、次のいずれかの操作を行います。

   - 通知で使用するリソースを設定済みである場合は、[Choose target type] で、[AWS Chatbot (Slack)] または [SNS topic] を選択します。[Choose target] で、クライアントの名前 (AWS Chatbot で設定した Slack クライアントの場合) または Amazon SNS トピックの Amazon リソースネーム (ARN) (通知に必要なポリシーが設定済みである Amazon SNS トピックの場合) を選択します。
   - 通知で使用するリソースを設定していない場合は、[Create target]、[SNS topic] の順に選択します。codestar-notifications- の後にトピックの名前を指定し、[Create] を選択します。

> **Note**
>
> - 通知ルールの作成の一環として Amazon SNS トピックを作成すると、トピックへのイベント発行を通知機能に許可するポリシーが適用されます。通知ルール用に作成したトピックを使用すると、このリソースに関する通知を受信するユーザーのみをサブスクライブできます。

- 通知ルールの作成の一環として AWS Chatbot クライアントを作成することはできません。AWS Chatbot (Slack) を選択すると、AWS Chatbot でクライアントを設定するように指示するボタンが表示されます。このオプションを選択すると、AWS Chatbot コンソールが開きます。詳細については、「通知と AWS Chatbot 間の統合を設定する」を参照してください。
- 既存の Amazon SNS トピックをターゲットとして使用する場合は、このトピック用の他のすべてのポリシーに加えて、AWS CodeStar Notifications に必要なポリシーを追加する必要があります。詳細については、「通知用の Amazon SNS トピックを設定する」および「通知の内容とセキュリティについて」を参照してください。

8. ルールの作成を終了するには、[Submit (送信)] を選択します。
9. 通知を受け取るには、そのルールの Amazon SNS トピックにユーザーをサブスクライブする必要があります。詳細については、「ターゲットである Amazon SNS トピックへのユーザーのサブスクライブ」を参照してください。また、通知と AWS Chatbot の統合をセットアップして、Amazon Chime チャットルームに通知を送信することもできます。詳細については、「通知と AWS Chatbot 間の統合を設定する」を参照してください。

通知ルールを作成するには (AWS CLI)

1. ターミナルまたはコマンドプロンプトで、create-notification rule コマンドを実行して、JSON スケルトンを生成します。

```
aws codestarnotifications create-notification-rule --generate-cli-skeleton
 > rule.json
```

ファイルには任意の名前を付けることができます。この例では、ファイルの名前を *rule.json* とします。

2. プレーンテキストエディタで JSON ファイルを開き、これを編集してルールに必要なリソース、イベントタイプ、ターゲットを含めます。次の例は、AWS アカウントで ID が *123456789012* の *MyBuildProject* というビルドプロジェクトのための **MyNotificationRule** という通知ルールを示しています。ビルドが成功したとき、通知は、完全な詳細タイプで Amazon SNS トピック *codestar-notifications-MyNotificationTopic* に送信されます：

```
{
    "Name": "MyNotificationRule",
```

```
    "EventTypeIds": [
        "codebuild-project-build-state-succeeded"
    ],
    "Resource": "arn:aws:codebuild:us-east-2:123456789012:MyBuildProject",
    "Targets": [
        {
            "TargetType": "SNS",
            "TargetAddress": "arn:aws:sns:us-east-2:123456789012:codestar-
notifications-MyNotificationTopic"
        }
    ],
    "Status": "ENABLED",
    "DetailType": "FULL"
}
```

ファイルを保存します。

3. 先ほど編集したファイルを使用して、ターミナルまたはコマンドラインで、create-notification-rule コマンドを再度実行し、通知ルールを作成します。

```
aws codestarnotifications create-notification-rule --cli-input-json
 file://rule.json
```

4. 成功すると、コマンドは次のような通知ルールの ARN を返します。

```
{
    "Arn": "arn:aws:codestar-notifications:us-east-1:123456789012:notificationrule/
dc82df7a-EXAMPLE"
}
```

AWS CodeBuild でのビルドプロジェクト名の一覧表示

CodeBuild でビルドプロジェクトのリストを表示するには、AWS CodeBuild コンソール、AWS CLI、または AWS SDK を使用します。

トピック

- ビルドプロジェクト名の一覧表示 (コンソール)
- ビルドプロジェクト名の一覧表示 (AWS CLI)
- ビルドプロジェクト名の一覧表示 (AWS SDK)

ビルドプロジェクト名の一覧表示 (コンソール)

コンソールで AWS リージョンにあるビルドプロジェクトのリストを表示できます。この情報には、名前、ソースプロバイダー、リポジトリ、最新のビルドステータス、説明 (ある場合) が含まれます。

1. AWS CodeBuild コンソール (https://console.aws.amazon.com/codesuite/codebuild/home) を開きます。
2. ナビゲーションペインで、[Build projects] を選択します。

> **Note**
>
> デフォルトでは、最新の 10 個のビルドプロジェクトのみが表示されます。さらに多くのビルドプロジェクトを表示するには、歯車アイコンを選択して [Projects per page (ページ毎プロジェクト数)] で別の値を選択するか、前後の矢印を使用します。

ビルドプロジェクト名の一覧表示 (AWS CLI)

list-projects コマンドを実行します。

```
aws codebuild list-projects --sort-by sort-by --sort-order sort-order --next-
token next-token
```

上記のコマンドで、次のプレースホルダを置き換えます。

- *sort-by*: ビルドプロジェクト名を一覧表示するために使用する条件を示すためのオプションの文字列。有効な値を次に示します。
  - CREATED_TIME: 各ビルドプロジェクトがいつ作成されたかに基づいて、ビルドプロジェクト名を一覧表示します。
  - LAST_MODIFIED_TIME: 各ビルドプロジェクトに関する情報が最後に変更されたときに基づいてビルドプロジェクト名を一覧表示します。
  - NAME 各ビルドプロジェクト名に基づいて、ビルドプロジェクト名を一覧表示します。
- *sort-order*: ビルドプロジェクトのリストを表示するためのオプションの文字列。*sort-by* に基づく。有効な値は、ASCENDING および DESCENDING です。
- *next-token*: オプションの文字列。以前の実行中に、リストに 100 を超える項目がある場合、最初の 100 項目だけが、next token と呼ばれる一意の文字列と共に返されます。リスト内の項目の

次のバッチを取得するには、次のコマンドを再度実行し、次のトークンを呼び出しに追加します。リスト内のすべての項目を取得するには、次のトークンが返されなくなるまで、このコマンドを、以後のすべての次のトークンで実行し続けます。

たとえば、次のコマンドを実行するとします。

```
aws codebuild list-projects --sort-by NAME --sort-order ASCENDING
```

次のような結果が出力に表示されることがあります。

```
{
  "nextToken": "Ci33ACF6...The full token has been omitted for brevity...U+AkMx8=",
  "projects": [
    "codebuild-demo-project",
    "codebuild-demo-project2",
    ... The full list of build project names has been omitted for brevity ...
    "codebuild-demo-project99"
  ]
}
```

このコマンドをもう一度実行します。

```
aws codebuild list-projects  --sort-by NAME --sort-order ASCENDING --next-token
 Ci33ACF6...The full token has been omitted for brevity...U+AkMx8=
```

次のような結果が出力に表示されることがあります。

```
{
  "projects": [
    "codebuild-demo-project100",
    "codebuild-demo-project101",
    ... The full list of build project names has been omitted for brevity ...
    "codebuild-demo-project122"
  ]
}
```

ビルドプロジェクト名の一覧表示 (AWS SDK)

AWS CodeBuild を AWS SDK と組み合わせて使用する方法については、「AWS SDK とツールのリファレンス」を参照してください。

AWS CodeBuild でビルドプロジェクトの詳細を表示する

CodeBuild でビルドプロジェクトの詳細を表示するには、AWS CodeBuild コンソール、AWS CLI、または AWS SDK を使用します。

トピック

ビルドプロジェクトの詳細を表示する (コンソール)

1. AWS CodeBuild コンソール (https://console.aws.amazon.com/codesuite/codebuild/home) を開きます。
2. ナビゲーションペインで、[Build projects] を選択します。

> **Note**
>
> デフォルトでは、最新の 10 個のビルドプロジェクトのみが表示されます。さらに多くのビルドプロジェクトを表示するには、歯車アイコンを選択して [Projects per page (ページ毎プロジェクト数)] で別の値を選択するか、前後の矢印を使用します。

3. ビルドプロジェクトのリストの [名前] 列で、ビルドプロジェクトのリンクを選択します。
4. [ビルドプロジェクト: ***project-name***] ページで、[ビルドの詳細] を選択します。

ビルドプロジェクトの詳細を表示する (AWS CLI)

batch-get-projects コマンドを実行します。

```
aws codebuild batch-get-projects --names names
```

上記のコマンドで、次のプレースホルダを置き換えます。

- *names*: 詳細を表示する 1 つ以上のビルドプロジェクト名を示すのに必要な文字列。複数のビルドプロジェクトを指定するには、各ビルドプロジェクトの名前をスペースで区切ります。最大 100

のビルドプロジェクト名を指定できます。ビルドプロジェクトのリストを表示するには、「ビルドプロジェクト名の一覧表示 (AWS CLI)」を参照してください。

たとえば、次のコマンドを実行するとします。

```
aws codebuild batch-get-projects --names codebuild-demo-project codebuild-demo-project2
 my-other-demo-project
```

次のような結果が出力に表示されます。省略記号 (...) は簡潔にするために省略されたデータを表すのに使用されます。

```
{
  "projectsNotFound": [
    "my-other-demo-project"
  ],
  "projects": [
    {
      ...
      "name": codebuild-demo-project,
      ...
    },
    {
      ...
      "name": codebuild-demo-project2",
      ...
    }
  ]
}
```

上記の出力では、指定されたビルドプロジェクト名はすべて projectsNotFound 配列にリストされていますが、情報は見つかりませんでした。projects 配列は、情報が見つかった各ビルドプロジェクトの詳細を示しています。ビルドプロジェクトの詳細は、簡潔にするために前の出力から省略されています。詳細については、「ビルドプロジェクトの作成 (AWS CLI)」の出力を参照してください。

「batch-get-projects」コマンドは、特定のプロパティ値のフィルタリングをサポートしていませんが、プロジェクトのプロパティを列挙するスクリプトを記述できます。たとえば、次の Linux シェルスクリプトは、現在のアカウントの現在のリージョンのプロジェクトを列挙し、各プロジェクトで使用されるイメージを出力します。

```
#!/usr/bin/sh

# This script enumerates all of the projects for the current account
# in the current region and prints out the image that each project is using.

imageName=""

function getImageName(){
  local environmentValues=(${1//$'\t'/ })
  imageName=${environmentValues[1]}
}

function processProjectInfo() {
  local projectInfo=$1

  while IFS=$'\t' read -r section value; do
    if [[ "$section" == *"ENVIRONMENT"* ]]; then
      getImageName "$value"
    fi
  done <<< "$projectInfo"
}

# Get the list of projects.
projectList=$(aws codebuild list-projects --output=text)

for projectName in $projectList
do
  if [[ "$projectName" != *"PROJECTS"* ]]; then
    echo "============================================="

    # Get the detailed information for the project.
    projectInfo=$(aws codebuild batch-get-projects --output=text --names
 "$projectName")

    processProjectInfo "$projectInfo"

    printf 'Project "%s" has image "%s"\n' "$projectName" "$imageName"
  fi
done
```

AWS CLI を AWS CodeBuild と組み合わせて使用する方法については、「コマンドラインリファレンス」を参照してください。

ビルドプロジェクトの詳細を表示する (AWS SDK)

AWS CodeBuild を AWS SDK と組み合わせて使用する方法については、「AWS SDK とツールのリファレンス」を参照してください。

AWS CodeBuild でのキャッシュのビルド

キャッシュを使用すると、プロジェクトを構築する時間を短縮できます。キャッシュでは、ビルド環境の再利用可能な部分が保存され、複数のビルドでそれらを使用することができます。ビルドプロジェクトでは、Amazon S3 とローカルの 2 種類のキャッシュのうち、いずれかを使用できます。ローカルキャッシュを使用する場合は、3 つのキャッシュモード (ソースキャッシュ、Docker レイヤーキャッシュ、カスタムキャッシュ) のうち 1 つ以上を選択する必要があります。

> **Note**
>
> Docker レイヤーキャッシュモードは Linux 環境でのみ利用可能です。このモードを選択した場合は、特権モードのビルドを実行する必要があります。特権モードが付与された CodeBuild プロジェクトは、すべてのデバイスへのコンテナアクセスを許可します。詳細については、Docker Docs ウェブサイトの「ランタイム特権と Linux 機能」を参照してください。

トピック

- Amazon S3 のキャッシュ
- ローカルキャッシュ

Amazon S3 のキャッシュ

Amazon S3 キャッシュでは、複数のビルドホスト間で利用できるキャッシュを Amazon S3 バケットに保存します。これは、ダウンロードするよりも構築にコストがかかる小規模から中間ビルドアーティファクトに適したオプションです。また、ネットワーク経由で転送するには長い時間がかかる場合があるため、大規模なビルドアーティファクトには適していません。ビルドパフォーマンスに影響を及ぼす可能性があります。また、Docker レイヤーを使用する場合、これは最適なオプションではありません。

ローカルキャッシュ

ローカルキャッシュは、そのビルドホストのみが利用できるキャッシュをそのビルドホストにローカルに保存します。キャッシュはビルドホストですぐに利用できるため、この方法は大規模から中間ビルドアーティファクトに適しています。ビルドの頻度が低い場合、これは最適なオプションではありません。つまり、ビルドパフォーマンスはネットワーク転送時間の影響を受けません。

ローカルキャッシングを選択した場合は、次のキャッシュモードを 1 つ以上選択する必要があります。

- ソースキャッシュモードは、プライマリソースとセカンダリソースの Git メタデータをキャッシュします。キャッシュ作成後のビルドでは、コミット間の変更のみプルされます。このモードは、クリーンな作業ディレクトリと、大きな Git リポジトリであるソースを持つプロジェクトに適しています。このオプションを選択し、プロジェクトで Git リポジトリ (AWS CodeCommit、、 GitHub Enterprise Server GitHub、または Bitbucket) を使用しない場合、 オプションは無視されます。
- Docker レイヤーキャッシュモードは、既存の Docker レイヤーをキャッシュします。このモードは、大きな Docker イメージを構築または取得するプロジェクトに適しています。そのため、大きな Docker イメージをネットワークからプルすることによって生じるパフォーマンス上の問題を回避できます。

  > **Note**
  >
  > - Docker レイヤーキャッシュは Linux 環境でのみ使用できます。
  > - プロジェクトに必要な Docker アクセス許可が付与されるように、`privileged` フラグを設定する必要があります 。
  >
  >   デフォルトでは、Docker デーモンは VPC 以外のビルドで有効になっています。VPC ビルドに Docker コンテナを使用する場合は、Docker Docs ウェブサイトの「ランタイム特権と Linux 機能」を参照して、特権モードを有効にします。また、Windows は特権モードをサポートしていません。
  > - Docker レイヤーキャッシュを使用する前に、セキュリティへの影響を考慮してください。

- カスタムキャッシュモードは buildspec ファイルで指定したディレクトリをキャッシュします。このシナリオは、ビルドシナリオが他の 2 つのローカルキャッシュモードのいずれにも適していない場合に適しています。カスタムキャッシュを使用する場合:

- キャッシュに指定できるのはディレクトリのみです。個々のファイルを指定することはできません。
- キャッシュされたディレクトリを参照するには、シンボリックリンクを使用します。
- キャッシュされたディレクトリは、プロジェクトソースをダウンロードする前にビルドにリンクされます。キャッシュされたアイテムにより、同じ名前のソースアイテムが上書きされます。ディレクトリは buildspec ファイルのキャッシュパスを使って指定されます。詳細については、「buildspec の構文」を参照してください。
- ソースとキャッシュで同じディレクトリ名は使用しないでください。ローカルにキャッシュされたディレクトリにより、ソースリポジトリ内の同じ名前のディレクトリの内容が上書きまたは削除される場合があります。

> **Note**
>
> ローカルキャッシュは、環境タイプ `LINUX_GPU_CONTAINER` とコンピューティングタイプ `BUILD_GENERAL1_2XLARGE` ではサポートされていません。詳細については、「ビルド環境のコンピューティングモードおよびタイプ」を参照してください。

> **Note**
>
> VPC と連携 CodeBuild するように を設定する場合、ローカルキャッシュはサポートされません。で VPCs「」を参照してくださいAmazon Virtual Private Cloud AWS CodeBuild で を使用する。 CodeBuild

トピック

- ローカルキャッシュの指定 (CLI)
- ローカルキャッシュの指定 (コンソール)
- ローカルキャッシュの指定 (AWS CloudFormation)

ローカルキャッシュを指定するには、AWS CLI、コンソール、SDK、または AWS CloudFormation を使用できます。

ローカルキャッシュの指定 (CLI)

3 つの各ローカルキャッシュタイプを指定するには、AWS CLI で --cache パラメータを使用します。

- ソースキャッシュを指定するには:

```
--cache type=LOCAL,mode=[LOCAL_SOURCE_CACHE]
```

- Docker レイヤーキャッシュを指定するには:

```
--cache type=LOCAL,mode=[LOCAL_DOCKER_LAYER_CACHE]
```

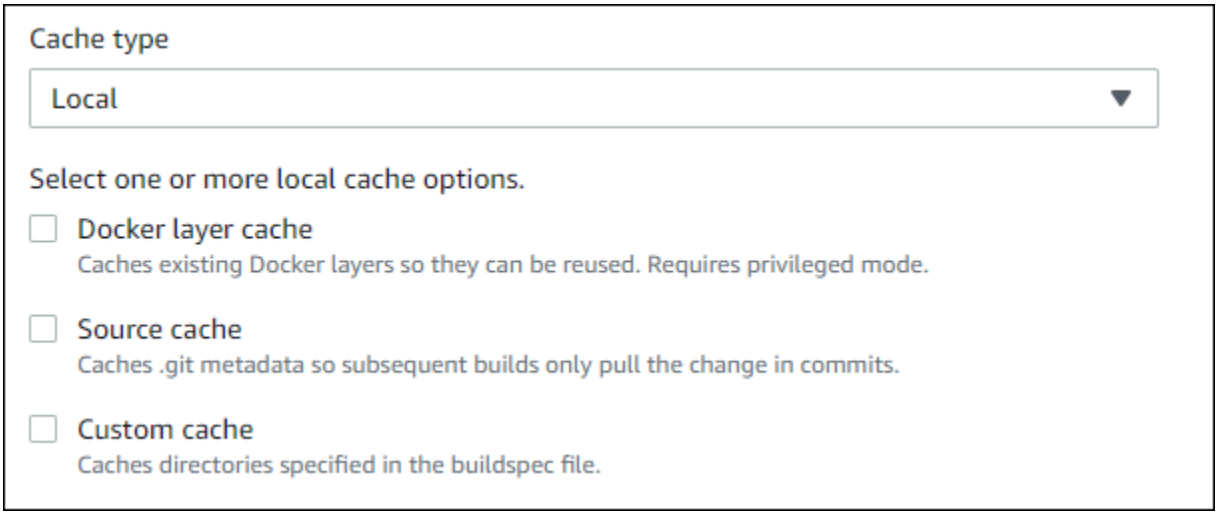
- カスタムキャッシュを指定するには:

```
--cache type=LOCAL,mode=[LOCAL_CUSTOM_CACHE]
```

詳細については、「ビルドプロジェクトの作成 (AWS CLI)」を参照してください。

ローカルキャッシュの指定 (コンソール)

キャッシュは、コンソールの [アーティファクト] セクションで指定します。[Cache type] (キャッシュタイプ) で、[Amazon S3] または [Local] (ローカル) を選択します。[ローカル] を選択した場合は、3 つのローカルキャッシュオプションのうち、1 つ以上を選択します。



詳細については、「ビルドプロジェクトの作成 (コンソール)」を参照してください。

ローカルキャッシュの指定 (AWS CloudFormation)

AWS CloudFormation を使用してローカルキャッシュを指定する場合は、Cache プロパティの Type で、LOCAL を指定します。以下の YAML 形式の AWS CloudFormation コード例では、3 つのローカルキャッシュタイプをすべて指定しています。任意のタイプの組み合わせを指定できます。Docker レイヤーキャッシュを使用する場合は、Environment で、PrivilegedMode を true、Type を LINUX_CONTAINER に設定する必要があります。

```
CodeBuildProject:
    Type: AWS::CodeBuild::Project
    Properties:
      Name: MyProject
      ServiceRole: <service-role>
      Artifacts:
        Type: S3
        Location: <bucket-name>
        Name: myArtifact
        EncryptionDisabled: true
        OverrideArtifactName: true
      Environment:
        Type: LINUX_CONTAINER
        ComputeType: BUILD_GENERAL1_SMALL
        Image: aws/codebuild/standard:5.0
        Certificate: <bucket/cert.zip>
        # PrivilegedMode must be true if you specify LOCAL_DOCKER_LAYER_CACHE
        PrivilegedMode: true
      Source:
        Type: GITHUB
        Location: <github-location>
        InsecureSsl: true
        GitCloneDepth: 1
        ReportBuildStatus: false
      TimeoutInMinutes: 10
      Cache:
        Type: LOCAL
        Modes: # You can specify one or more cache mode,
          - LOCAL_CUSTOM_CACHE
          - LOCAL_DOCKER_LAYER_CACHE
          - LOCAL_SOURCE_CACHE
```

> **Note**
>
> デフォルトでは、Docker デーモンは VPC 以外のビルドで有効になっています。VPC ビルドに Docker コンテナを使用する場合は、Docker Docs ウェブサイトの「ランタイム特権と Linux 機能」を参照して、特権モードを有効にします。また、Windows は特権モードをサポートしていません。

詳細については、「ビルドプロジェクトの作成 (AWS CloudFormation)」を参照してください。

でトリガーを構築する AWS CodeBuild

トピック

- AWS CodeBuild トリガーの作成
- AWS CodeBuild トリガーの編集

AWS CodeBuild トリガーの作成

AWS CodeBuild トリガーの作成 (コンソール)

プロジェクトでトリガーを作成し、1 時間、1 日、または 1 週間に 1 回ビルドをスケジュールできます。Amazon CloudWatch cron 式でカスタムルールを使用してトリガーを作成することもできます。たとえば、cron 式を使用して、毎週特定の時間にビルドをスケジュールできます。

> **Note**
>
> ビルドトリガー、Amazon EventBridge イベント、または AWS Step Functionsタスクからバッチビルドを開始することはできません。

トリガーを作成するには

1. AWS CodeBuild コンソール (https://console.aws.amazon.com/codesuite/codebuild/home) を開きます。
2. ナビゲーションペインで、[Build projects] を選択します。
3. トリガーを追加するビルドプロジェクトのリンクを選択し、[トリガーのビルド] タブを選択します。

> **Note**
>
> デフォルトでは、最新の 100 個のビルドプロジェクトが表示されます。さらに多くのビルドプロジェクトを表示するには、歯車アイコンを選択して [Projects per page (ページ毎プロジェクト数)] で別の値を選択するか、前後の矢印を使用します。

4. [Create trigger (トリガーの作成)] を選択します。
5. [トリガー名] に名前を入力します。
6. [Frequency] (頻度) ドロップダウンリストから、トリガーの頻度を選択します。CRON 式を使用して頻度を作成する場合は、[Custom] (カスタム) を選択します。
7. トリガーの頻度のパラメータを指定します。選択肢の最初の数文字をテキストボックスに入力すると、ドロップダウンメニュー項目がフィルタリングされます。

> **Note**
>
> 開始時間と分はゼロベースです。開始分は 0 から 59 までの数値です。開始時は 0 から 23 までの数値です。たとえば、毎日午後 12:15 に開始する日次トリガーは、開始時が 12、開始分が 15 になります。毎日深夜に開始される日次トリガーは、開始時がゼロで、開始分がゼロです。毎日午後 11:59 に開始する毎日のトリガーは、開始時が 23、開始分が 59 です。

Frequency	必須パラメータ	詳細
時間単価	開始時間 (分)	[開始時間 (分)] ドロップダウンメニューを使用します。
1 日 1 回	開始時間 (分) 開始時間 (時)	[開始時間 (分)] ドロップダウンメニューを使用します。 [開始時間 (時)] ドロップダウンメニューを使用します。
毎週	開始時間 (分) 開始時間 (時)	[開始時間 (分)] ドロップダウンメニューを使用します。

Frequency	必須パラメータ	詳細
	開始日	[開始時間 (時)] ドロップダウンメニューを使用します。 [開始時間 (日)] ドロップダウンメニューを使用します。
Custom	Cron 式	[Cron 式] に Cron 式を入力します。Cron 式には 、空白で区切られた 6 つの必須フィールドがあります。これらのフィールドでは、分、時、日付、月、曜日、および年の開始値を指定します。範囲や追加の値などを指定するには、ワイルドカードを使用します。例えば、cron 式は毎週平日の午前 9 時にビルドを**`0 9 ? * MON-FRI *`**スケジュールします。詳細については、「Amazon CloudWatch Events ユーザーガイド」の「cron 式」を参照してください。

8. [Enable this trigger (このトリガーの有効化)] を選択します。
9. (オプション) [アドバンスト] セクションを展開します。[ソースバージョン] に、ソースのバージョンを入力します。

   - Amazon S3 の場合、ビルドする入力アーティファクトのバージョンに対応するバージョン ID を入力します。[ソースバージョン] が空白のままの場合は、最新バージョンが使用されます。
   - AWS CodeCommit の場合は、コミット ID を入力します。[ソースバージョン] が空白のままの場合は、デフォルトブランチの HEAD コミット ID が使用されます。
   - GitHub または GitHub Enterprise の場合は、ビルドするソースコードのバージョンに対応するコミット ID、プルリクエスト ID、ブランチ名、またはタグ名を入力します。プルリクエスト ID を指定する場合、`pr/`*`pull-request-ID`* (例: `pr/25`) 形式を使用する必要があります。ブ

ランチ名を指定すると、ブランチの HEAD コミット ID が使用されます。[Source version] が空白の場合は、デフォルトのブランチの HEAD コミット ID が使用されます。

- Bitbucket の場合、ビルドするソースコードのバージョンに対応するコミット ID、ブランチ名、またはタグ名を入力します。ブランチ名を指定すると、ブランチの HEAD コミット ID が使用されます。[Source version] が空白の場合は、デフォルトのブランチの HEAD コミット ID が使用されます。

10. (オプション) 5 分 ~ 480 分 (8 時間) の間のタイムアウトを指定します。この値で、AWS CodeBuild が停止するまでビルドを試みる時間を指定します。[時間] と [分] が空白のままの場合、プロジェクトで指定されたデフォルトのタイムアウト値が使用されます。
11. [Create trigger (トリガーの作成)] を選択します。

プログラムでAWS CodeBuildトリガーを作成する

CodeBuild は、ビルドトリガーに Amazon EventBridge ルールを使用します。 EventBridge API を使用して、 CodeBuild プロジェクトのビルドトリガーをプログラムで作成できます。詳細については、「Amazon EventBridge API リファレンス」を参照してください。

AWS CodeBuild トリガーの編集

AWS CodeBuild トリガーの編集 (コンソール)

プロジェクトでトリガーを編集し、1 時間、1 日、または 1 週間に 1 回ビルドをスケジュールできます。Amazon CloudWatch cron 式でカスタムルールを使用するようにトリガーを編集することもできます。たとえば、cron 式を使用して、毎週特定の時間にビルドをスケジュールできます。トリガーの作成方法については、「AWS CodeBuild トリガーの作成」を参照してください。

トリガーを編集するには

1. AWS CodeBuild コンソール (https://console.aws.amazon.com/codesuite/codebuild/home) を開きます。
2. ナビゲーションペインで、[Build projects] を選択します。
3. 変更するビルドプロジェクトのリンクを選択し、[ビルドのトリガー] タブを選択します。

> **Note**
>
> デフォルトでは、最新の 100 個のビルドプロジェクトが表示されます。さらに多くのビルドプロジェクトを表示するには、歯車アイコンを選択して [Projects per page (ページ毎プロジェクト数)] で別の値を選択するか、前後の矢印を使用します。

4. 変更するトリガーの横にあるラジオボタンを選択して、[Edit (編集)] を選択します。
5. [Frequency] (頻度) ドロップダウンリストから、トリガーの頻度を選択します。CRON 式を使用して頻度を作成する場合は、[Custom] (カスタム) を選択します。
6. トリガーの頻度のパラメータを指定します。選択肢の最初の数文字をテキストボックスに入力すると、ドロップダウンメニュー項目がフィルタリングされます。

> **Note**
>
> 開始時間と分はゼロベースです。開始分は 0 から 59 までの数値です。開始時は 0 から 23 までの数値です。たとえば、毎日午後 12:15 に開始する日次トリガーは、開始時が 12、開始分が 15 になります。毎日深夜に開始される日次トリガーは、開始時がゼロで、開始分がゼロです。毎日午後 11:59 に開始する毎日のトリガーは、開始時が 23、開始分が 59 です。

Frequency	必須パラメータ	詳細
時間単価	開始時間 (分)	[開始時間 (分)] ドロップダウンメニューを使用します。
1 日 1 回	開始時間 (分) 開始時間 (時)	[開始時間 (分)] ドロップダウンメニューを使用します。 [開始時間 (時)] ドロップダウンメニューを使用します。
毎週	開始時間 (分) 開始時間 (時) 開始日	[開始時間 (分)] ドロップダウンメニューを使用します。 [開始時間 (時)] ドロップダウンメニューを使用します。

Frequency	必須パラメータ	詳細
		[開始時間 (日)] ドロップダウンメニューを使用します。
Custom	Cron 式	[Cron 式] に Cron 式を入力します。Cron 式には 、空白で区切られた 6 つの必須フィールドがあります。これらのフィールドでは、分、時、日付、月、曜日、および年の開始値を指定します。範囲や追加の値などを指定するには、ワイルドカードを使用します。例えば、cron 式は毎週平日の午前 9 時にビルドを**`0 9 ? * MON-FRI *`**スケジュールします。詳細については、「Amazon CloudWatch Events ユーザーガイド」の「cron 式」を参照してください。

7. [Enable this trigger (このトリガーの有効化)] を選択します。

> **Note**
>
> https://console.aws.amazon.com/cloudwatch/ の Amazon CloudWatch コンソールを使用して、 で使用できないソースバージョン、タイムアウト、その他のオプションを編集できます AWS CodeBuild。

AWS CodeBuild トリガーをプログラムで編集する

CodeBuild は、ビルドトリガーに Amazon EventBridge ルールを使用します。 EventBridge API を使用して、 CodeBuild プロジェクトのビルドトリガーをプログラムで編集できます。詳細については、「Amazon EventBridge API リファレンス」を参照してください。

GitLab 接続

接続を使用すると、 を使用してサードパーティープロバイダーを AWS リソースに関連付ける設定を承認および確立できます AWS CodeConnections。サードパーティーのリポジトリをビルドプロジェクトのソースとして関連付けるには、 接続を使用します。

で GitLab または GitLab セルフマネージドソースプロバイダーを追加するには CodeBuild、次のいずれかを選択できます。

- CodeBuild コンソールのビルドプロジェクトの作成ウィザードまたはソースの編集ページを使用して、 GitLabまたはGitLab セルフマネージドプロバイダーオプションを選択します。ソースプロバイダーを追加するにはへの接続 GitLabを作成する (コンソール)、「」を参照してください。このコンソールは、接続リソースの作成に役立ちます。
- CLI を使用して接続リソースを作成します。CLI を使用して接続リソースを作成するには、 (GitLab CLI) への接続を作成する「」を参照してください。

> **Note**
>
> [設定] からデベロッパーツール コンソールを使用して、接続を作成することもできます。[接続を作成する] を参照してください。

> **Note**
>
> でこの接続のインストールを許可することで GitLab、アカウントにアクセスしてデータを処理するアクセス許可を当社のサービスに付与し、アプリケーションをアンインストールすることでアクセス許可をいつでも取り消すことができます。

開始する前に:

- でアカウントを作成しておく必要があります GitLab。

> **Note**
>
> Connections は、接続の作成と承認に使用されたアカウントで所有するリポジトリへのアクセスだけを提供します。

> **Note**
>
> で所有者ロールを持つリポジトリへの接続を作成し GitLab、その接続を などのリソースを持つリポジトリで使用できます CodeBuild。グループ内のリポジトリでは、グループの所有者である必要はありません。

- ビルドプロジェクトのソースを指定するには、 にリポジトリを作成しておく必要があります GitLab。

トピック

- への接続 GitLabを作成する (コンソール)
- (GitLab CLI) への接続を作成する

への接続 GitLabを作成する (コンソール)

CodeBuild コンソールを使用して、 でプロジェクト (リポジトリ) の接続を追加するには、次の手順に従います GitLab。

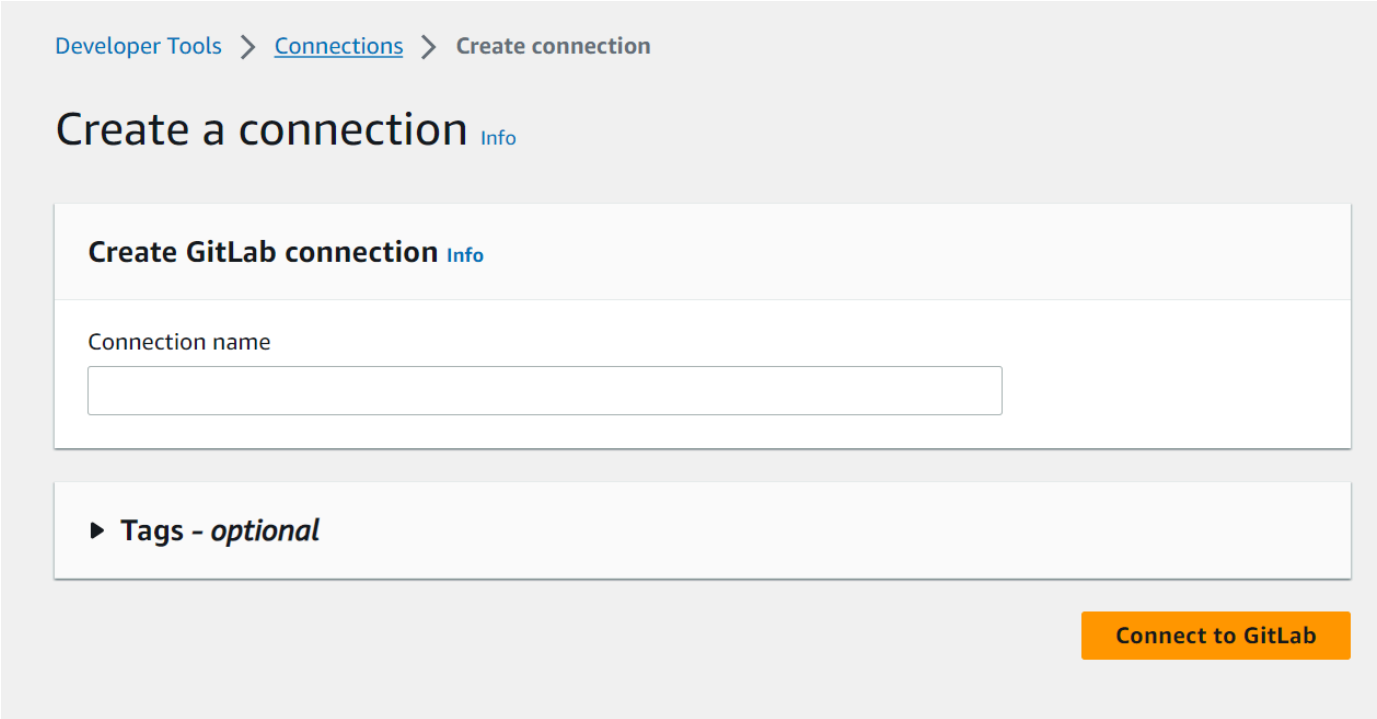
ビルドプロジェクトを作成または編集するには

1. CodeBuild コンソールにサインインします。
2. 次のいずれかを選択します 。
   - ビルドプロジェクトの作成を選択します。ビルドプロジェクトの作成 (コンソール)「」の手順に従って最初の画面を完了し、「ソースプロバイダー」の「ソース」セクションで「」を選択しますGitLab。
   - 既存のビルドプロジェクトを編集するには、 を選択します。編集 を選択し、ソース を選択します。ソースの編集ページのソースプロバイダー で、 を選択しますGitLab。
3. 以下のうちのひとつを選択します。
   - 「接続」で、「デフォルトの接続」を選択します。デフォルトの接続では、すべてのプロジェクトにデフォルトの GitLab接続が適用されます。
   - 接続 で、カスタム接続 を選択します。カスタム接続は、アカウントのデフォルト設定を上書きするカスタム GitLab接続を適用します。
4. 次のいずれかを行います。

- デフォルト接続またはカスタム接続 で、プロバイダーへの接続をまだ作成していない場合は、新しい GitLab接続の作成 を選択します。ステップ 5 に進み、接続を作成します。
- [接続] でプロバイダへの接続を既に作成している場合は、その接続を選択します。ステップ 10 に進みます。

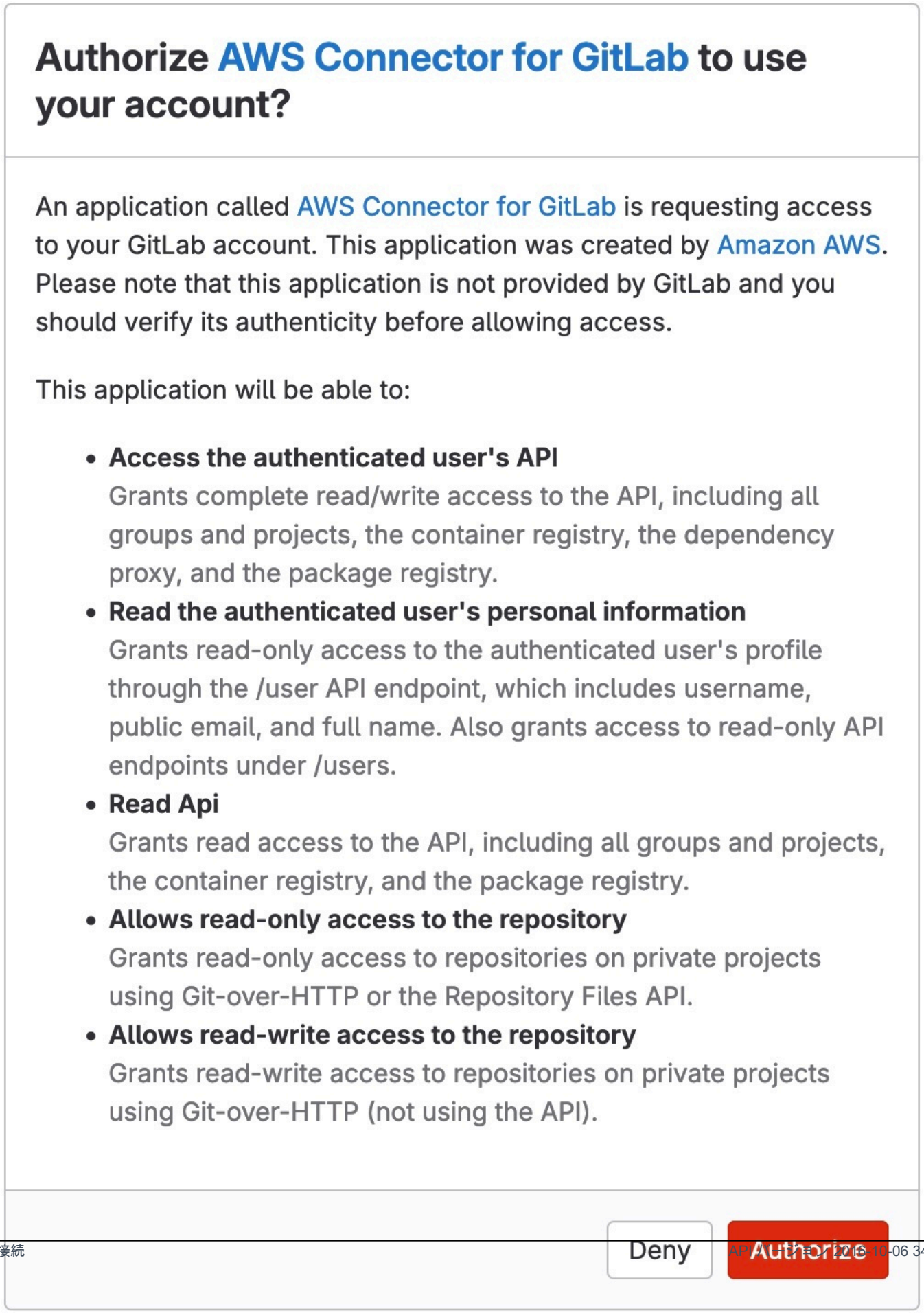
> **Note**
>
> GitLab 接続が作成される前にポップアップウィンドウを閉じる場合は、ページを更新する必要があります。

5. GitLab リポジトリへの接続を作成するには、「プロバイダーの選択」で、 を選択しますGitLab。[接続名] に、作成する接続の名前を入力します。 に接続 GitLab を選択します。



6. のサインインページ GitLab が表示されたら、認証情報を使用してログインし、サインインを選択します。
7. 初めて接続を承認する場合は、接続が GitLab アカウントにアクセスするための承認を要求するメッセージを含む認証ページが表示されます。

   [承認] を選択します。


Authorize AWS Connector for GitLab to use your account?
An application called AWS Connector for GitLab is requesting access to your GitLab account. This application was created by Amazon AWS. Please note that this application is not provided by GitLab and you should verify its authenticity before allowing access.
This application will be able to:
Access the authenticated user's API
Grants complete read/write access to the API, including all groups and projects, the container registry, the dependency proxy, and the package registry.
Read the authenticated user's personal information
Grants read-only access to the authenticated user's profile through the /user API endpoint, which includes username, public email, and full name. Also grants access to read-only API endpoints under /users.
Read Api
Grants read access to the API, including all groups and projects, the container registry, and the package registry.
Allows read-only access to the repository
Grants read-only access to repositories on private projects using Git-over-HTTP or the Repository Files API.
Allows read-write access to the repository
Grants read-write access to repositories on private projects using Git-over-HTTP (not using the API).
Deny
Authorize

8. ブラウザは接続コンソールページに戻ります。GitLab 接続設定 で、新しい接続は接続名 に表示されます。
9. [接続]を選択します。

   GitLab 接続が正常に作成されると、成功バナーが上部に表示されます。
10. 「ビルドプロジェクトの作成」ページの「デフォルト接続」または「カスタム接続」ドロップダウンリストで、接続 ARN がリストされていることを確認します。そうでない場合は、更新ボタンを選択して表示します。
11. リポジトリ で GitLab、 名前空間でプロジェクトパスを指定して、 でプロジェクトの名前を選択します。例えば、グループレベルのリポジトリの場合は、リポジトリ名を `group-name/repository-name` の形式で入力します。パスと名前空間の詳細については、https://docs.gitlab.com/ee/api/projects.html#get-single-project の `path_with_namespace`フィールドを参照してください。の名前空間の詳細については GitLab、https://docs.gitlab.com/ee/user/namespace/ を参照してください。

    > **Note**
    >
    > のグループの場合 GitLab、 名前空間でプロジェクトパスを手動で指定する必要があります。例えば、グループ `mygroup` 内のリポジトリの名前が `myrepo` の場合は、「`mygroup/myrepo`」と入力します。名前空間 を含むプロジェクトパスは、 の URL にあります GitLab。

12. ソースバージョン - オプション で、プルリクエスト ID、ブランチ、コミット ID、タグ、またはリファレンスとコミット ID を入力します。詳細については、「のソースバージョンサンプル AWS CodeBuild」を参照してください。

    > **Note**
    >
    > `811dd1ba1aba14473856cee38308caed7190c0d` または `5392f7` のように、コミット ID と似ていない Git ブランチ名を選択することをお勧めします。これにより、Git checkout が実際のコミットと衝突するのを防ぐことができます。

13. Git クローンの深度 - オプション では、指定されたコミット数に切り捨てられた履歴を持つシャロークローンを作成できます。完全クローンを希望する場合には、[Full (完全)] を選択します。

14. ビルドのステータス - オプション で、ビルドの開始と完了のステータスをソースプロバイダーに報告したい場合は、ビルドの開始と終了時にビルドステータスをソースプロバイダーに報告を選択します。

    ソースプロバイダにビルド状態を報告できるようにするには、ソースプロバイダに関連付けられたユーザーがリポジトリへの書き込みアクセス権を持っている必要があります。ユーザーが書き込みアクセス権を持っていない場合、ビルドのステータスは更新できません。詳細については、「ソースプロバイダーのアクセス」を参照してください。

(GitLab CLI) への接続を作成する

AWS Command Line Interface (AWS CLI) を使用して接続を作成できます。

これを行うには、create-connection コマンドを使用します。

> **Important**
>
> AWS CLI または を介して作成された接続 AWS CloudFormation は、デフォルトで PENDINGステータスになります。CLI または との接続を作成したら AWS CloudFormation、コンソールを使用して接続を編集し、ステータスを にしますAVAILABLE。

接続を作成する

- 「デベロッパーツールコンソールユーザーガイド」の GitLab 「 (CLI) への接続を作成する」の指示に従ってください。

でのウェブフックの使用 AWS CodeBuild

AWS CodeBuild は、、 GitHub Enterprise Server GitHub、 GitLab Self Managed GitLab、Bitbucket とのウェブフック統合をサポートしています。

トピック

- でウェブフックを使用するためのベストプラクティス AWS CodeBuild
- Bitbucket ウェブフックイベント
- GitHub ウェブフックイベント
- GitLab ウェブフックイベント

でウェブフックを使用するためのベストプラクティス AWS CodeBuild

パブリックリポジトリを使用してウェブフックをセットアップするプロジェクトでは、以下のオプションを使用することをお勧めします。

セットアップACTOR_ACCOUNT_IDフィルター

プロジェクトのウェブフックACTOR_ACCOUNT_IDフィルターグループにフィルターを追加して、ビルドをトリガーできるユーザーを指定します。に配信されるすべてのウェブフックイベント CodeBuild には、アクターの識別子を指定する送信者情報が含まれています。 CodeBuild は、フィルターで指定された正規表現パターンに基づいてウェブフックをフィルタリングします。このフィルタを使用して、ビルドのトリガーを許可する特定のユーザーを指定できます。詳細については、「GitHub ウェブフックイベント」および「Bitbucket ウェブフックイベント」を参照してください。

セットアップFILE_PATHフィルター

プロジェクトのウェブフックFILE_PATHフィルターグループにフィルターを追加して、変更時にビルドをトリガーできるファイルを含めたり除外したりします。例えば、 などの正規表現パターンと excludeMatchedPatternプロパティを使用して^buildspec.yml$、 buildspec.yml ファイルへの変更に対するビルドリクエストを拒否できます。詳細については、「GitHub ウェブフックイベント」および「Bitbucket ウェブフックイベント」を参照してください。

ビルドの IAM ロールのアクセス権限を絞り込む

Webhook によってトリガーされたビルドは、プロジェクトで指定された IAM サービスロールを使用します。サービスロールのアクセス許可は、ビルドの実行に必要な最小限のアクセス許可セットに設定することをお勧めします。たとえば、テストおよびデプロイのシナリオでは、テスト用にプロジェクトを 1 つ作成し、デプロイ用に別のプロジェクトを作成します。テストプロジェクトは、リポジトリからの webhook ビルドを受け付けますが、リソースへの書き込み権限は提供しません。デプロイメントプロジェクトはリソースへの書き込み権限を提供し、webhook フィルターは信頼済みのユーザーにのみビルドをトリガーできるように設定されています。

インラインまたは Amazon S3 に保管した buildspec を使用する

プロジェクト自体内で buildspec をインラインで定義する場合、または buildspec ファイルを Amazon S3 バケットに格納する場合、buildspec ファイルはプロジェクト所有者のみに表示されます。これにより、プル要求が buildspec ファイルにコードを変更したり、不要なビルドをトリガーしたりするのを防ぎます。詳細については、 CodeBuild API リファレンスの「ProjectSource.buildspec」を参照してください。

Bitbucket ウェブフックイベント

Webhook フィルタグループを使用して、ビルドをトリガーする Bitbucket ウェブフックイベントを指定できます。たとえば、特定のブランチへの変更に対してのみビルドをトリガーするように指定できます。

ビルドをトリガーするウェブフックイベントを指定するには、ウェブフックフィルタグループを 1 つ以上作成できます。任意のフィルターグループが true と評価されると、ビルドがトリガーされます。これは、グループ内のすべてのフィルターが true と評価されたときに発生します。フィルタグループを作成する際、以下を指定します。

イベント

Bitbucket では、次のイベントのうち、1 つ以上を選択できます:

- PUSH
- PULL_REQUEST_CREATED
- PULL_REQUEST_UPDATED
- PULL_REQUEST_MERGED
- PULL_REQUEST_CLOSED

ウェブフックのイベントタイプは、X-Event-Key フィールドのヘッダーに含まれています。次の表に、X-Event-Key ヘッダー値がイベントタイプにマッピングされる方法を示します。

> **Note**
>
> PULL_REQUEST_MERGED イベントタイプを使用するウェブフックフィルタグループを作成する場合は、Bitbucket ウェブフック設定で merged イベントを有効にする必要があります。また、 declinedイベントタイプを使用するウェブフックフィルタグループを作成する場合は、Bitbucket ウェブフック設定で PULL_REQUEST_CLOSEDイベントを有効にする必要があります。

X-Event-Key ヘッダー値	イベントタイプ
repo:push	PUSH
pullrequest:created	PULL_REQUEST_CREATED

X-Event-Key ヘッダー値	イベントタイプ
pullrequest:updated	PULL_REQUEST_UPDATED
pullrequest:fulfilled	PULL_REQUEST_MERGED
pullrequest:rejected	PULL_REQUEST_CLOSED

PULL_REQUEST_MERGED の場合、プルリクエストがスカッシュ戦略とマージされ、プルリクエストブランチが閉じられると、元のプルリクエストコミットは存在しなくなります。この場合、CODEBUILD_WEBHOOK_MERGE_COMMIT 環境変数には、圧縮されたマージコミットの識別子が含まれます。

1 つ以上のオプションフィルタ

フィルタを指定するには、正規表現を使用します。ビルドをトリガーするイベントでは、関連付けられているグループ内のすべてのフィルターが true と評価される必要があります。

ACTOR_ACCOUNT_ID (コンソール内の ACTOR_ID)

Bitbucket アカウント ID が正規表現パターンと一致すると、ビルドがウェブフックイベントでトリガーされます。この値は、ウェブフックフィルタペイロードの actor オブジェクトの account_id プロパティに表示されます。

HEAD_REF

ヘッドリファレンスが正規表現パターンと一致すると (refs/heads/branch-name と refs/tags/tag-name など)、ウェブフックイベントによってビルドがトリガーされます。HEAD_REF フィルタは、ブランチまたはタグについて Git 参照名を評価します。ブランチ名またはタグ名は、ウェブフックペイロードの push オブジェクトにある、new オブジェクトの name フィールドに表示されます。プルリクエストイベントの場合、ブランチ名はウェブフックペイロードの source オブジェクトにある、branch オブジェクトの name フィールドに表示されます。

BASE_REF

基本参照が正規表現パターンと一致すると、ビルドがウェブフックイベントでトリガーされます。BASE_REF フィルタは、プルリクエストイベントでのみ使用できます (例: refs/heads/branch-name)。BASE_REF フィルタは、ブランチの Git 参照名を評価します。ブランチ名は、ウェブフックペイロードの destination オブジェクトにある、branch オブジェクトの name フィールドに表示されます。

`FILE_PATH`

変更されたファイルのパスが正規表現パターンに一致すると、ビルドが Webhook イベントでトリガーされます。

`COMMIT_MESSAGE`

HEAD コミットメッセージが正規表現パターンに一致する場合に、Webhook はビルドをトリガーします。

`WORKFLOW_NAME`

ワークフロー名が正規表現パターンと一致すると、ウェブフックによってビルドがトリガーされます。

> **Note**
>
> ウェブフックペイロードは、Bitbucket リポジトリのウェブフック設定で見つかります。

トピック

- Bitbucket ウェブフックイベントのフィルタリング (コンソール)
- Bitbucket ウェブフックイベントのフィルタリング (SDK)
- Bitbucket ウェブフックイベントのフィルタリング (AWS CloudFormation)

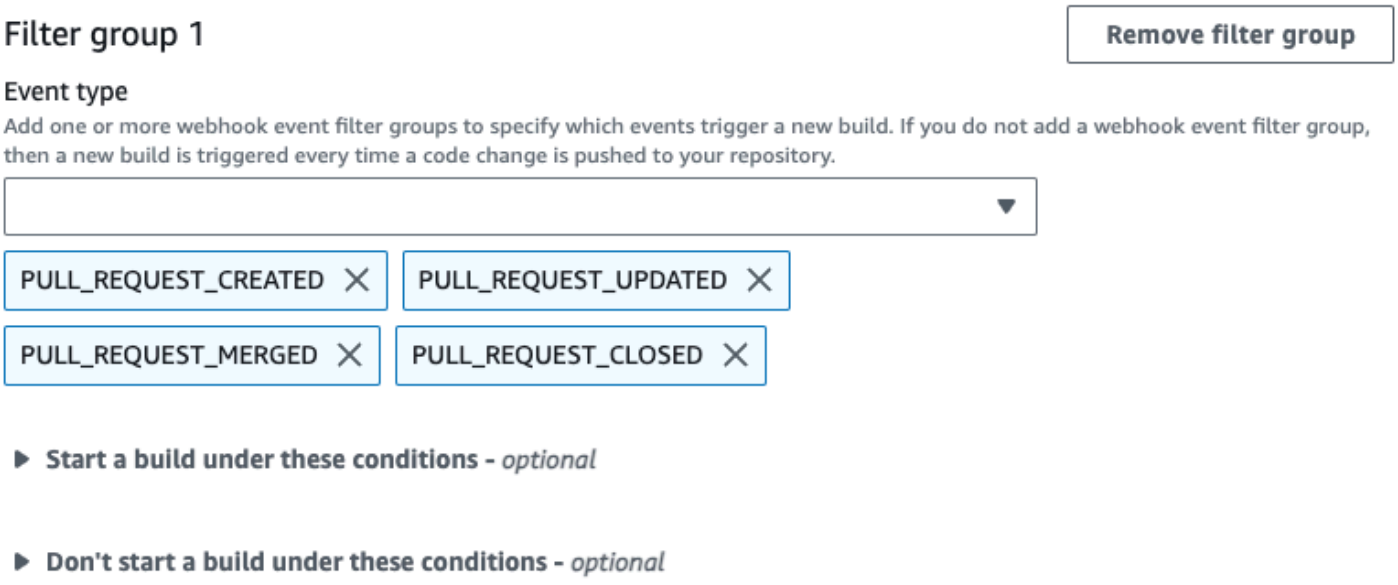
Bitbucket ウェブフックイベントのフィルタリング (コンソール)

を使用してウェブフックイベント AWS Management Console をフィルタリングするには：

1. プロジェクトの作成時に [コードの変更がこのレポジトリにプッシュされるたびに再構築する] を選択します。
2. [イベントタイプ] から、1 つ以上のイベントを選択します。
3. イベントでビルドをトリガーされた時間をフィルタリングするには、[これらの条件でビルドを開始する] で、1 つ以上のオプションフィルタを追加します。
4. イベントがトリガーされていない時間をフィルタリングするには、[これらの条件でビルドを開始しない] で、1 つ以上のオプションフィルタを追加します。
5. 別のフィルタグループを追加するには、[フィルタグループの追加] を選択します。

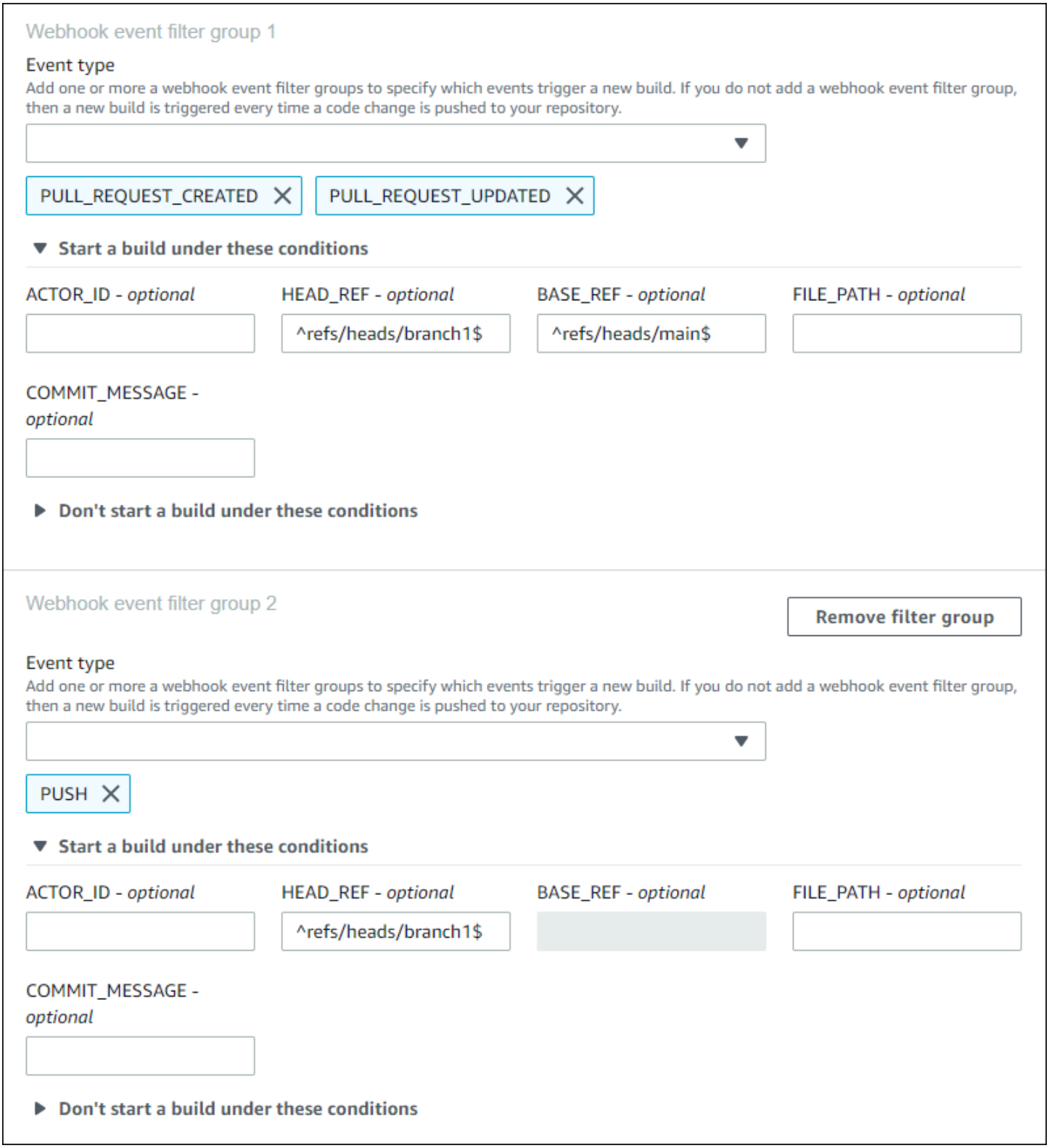
詳細については、 API リファレンスWebhookFilterのビルドプロジェクトの作成 (コンソール)「」および「」を参照してください。 AWS CodeBuild

この例では、ウェブフックフィルタグループは、プルリクエストに対してのみビルドをトリガーします。

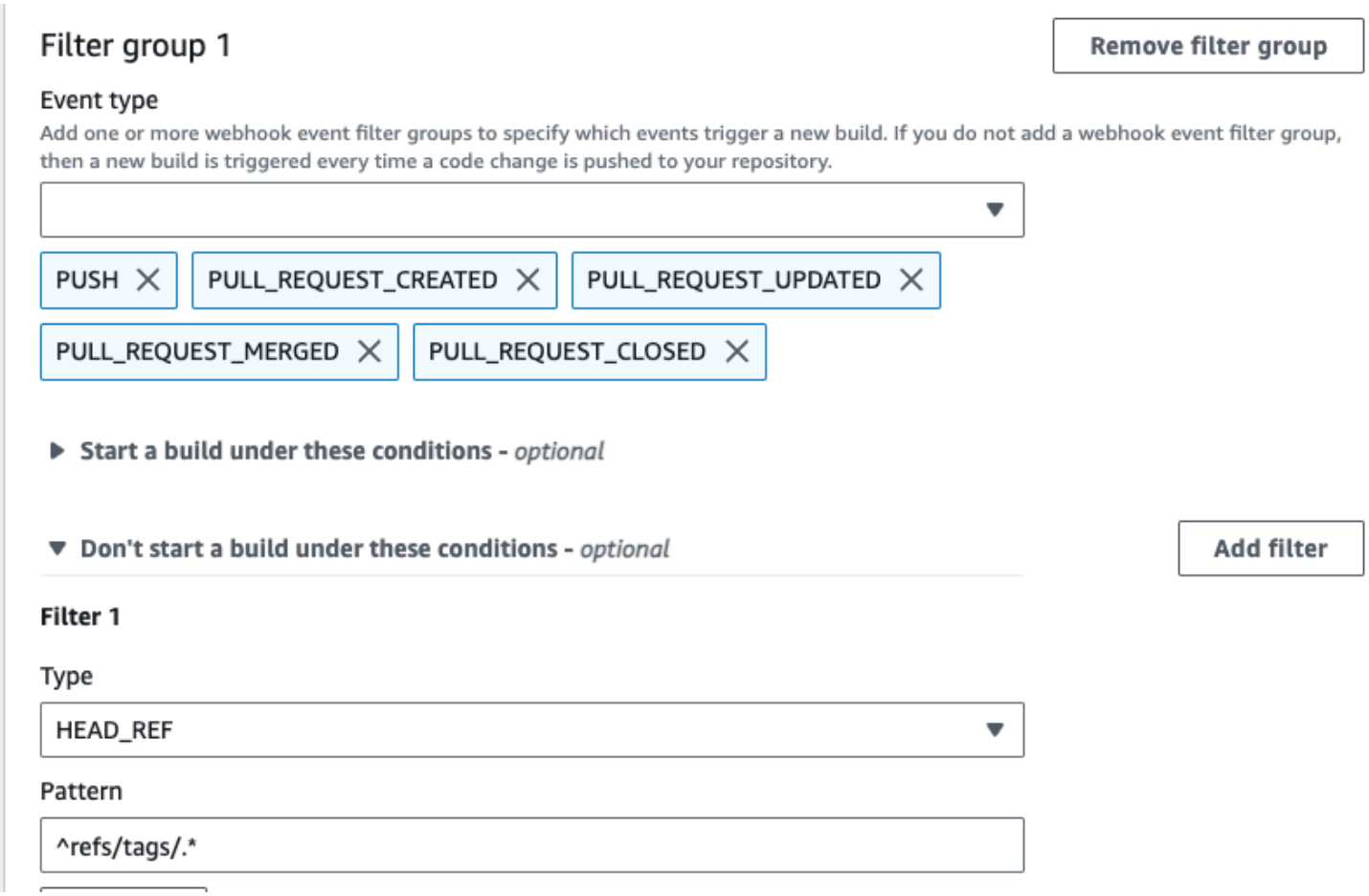


2 つのフィルタグループの例を使用した場合、ビルドは一方または両方が true と評価されるとトリガーされます。

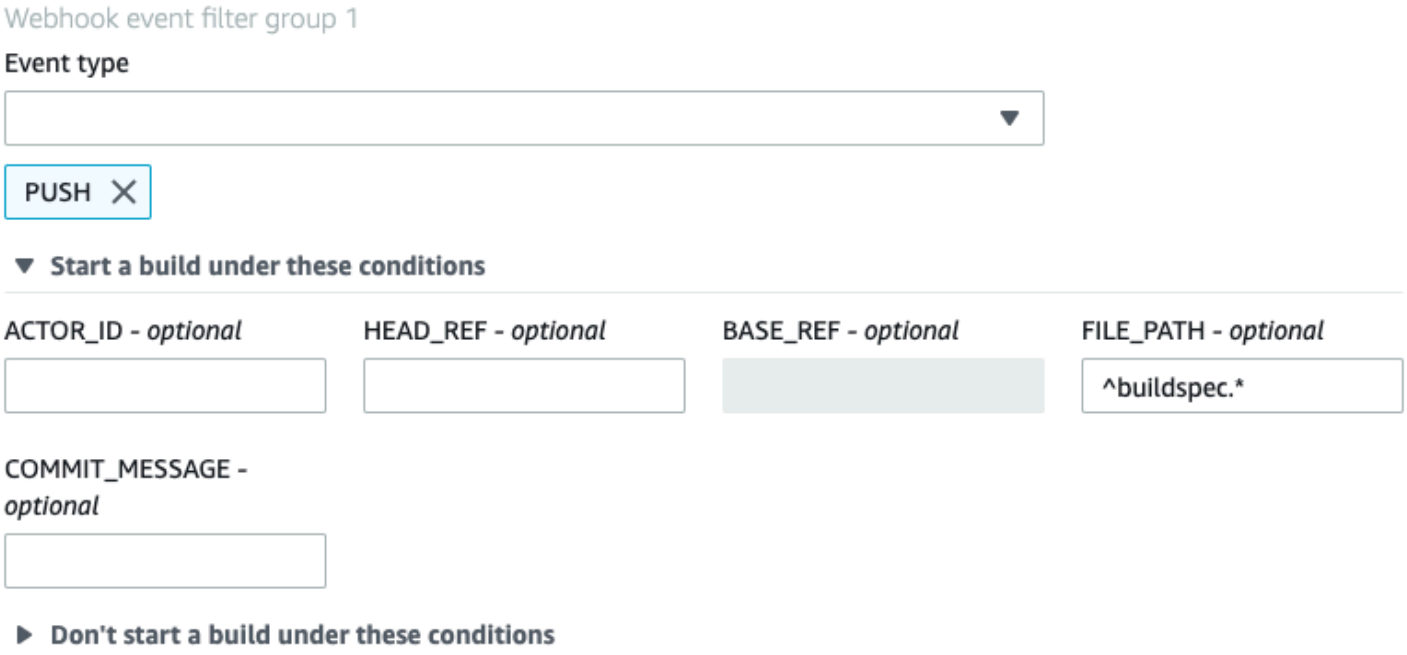
- 最初のフィルタグループでは、正規表現 `^refs/heads/main$` に一致する Git 参照と `^refs/heads/branch1!` に一致するヘッド参照を含むブランチで作成または更新されたプルリクエストを指定します。
- 2 番目のフィルタグループでは、正規表現 `^refs/heads/branch1$` に一致する Git 参照を含むブランチでプッシュリクエストを指定します。



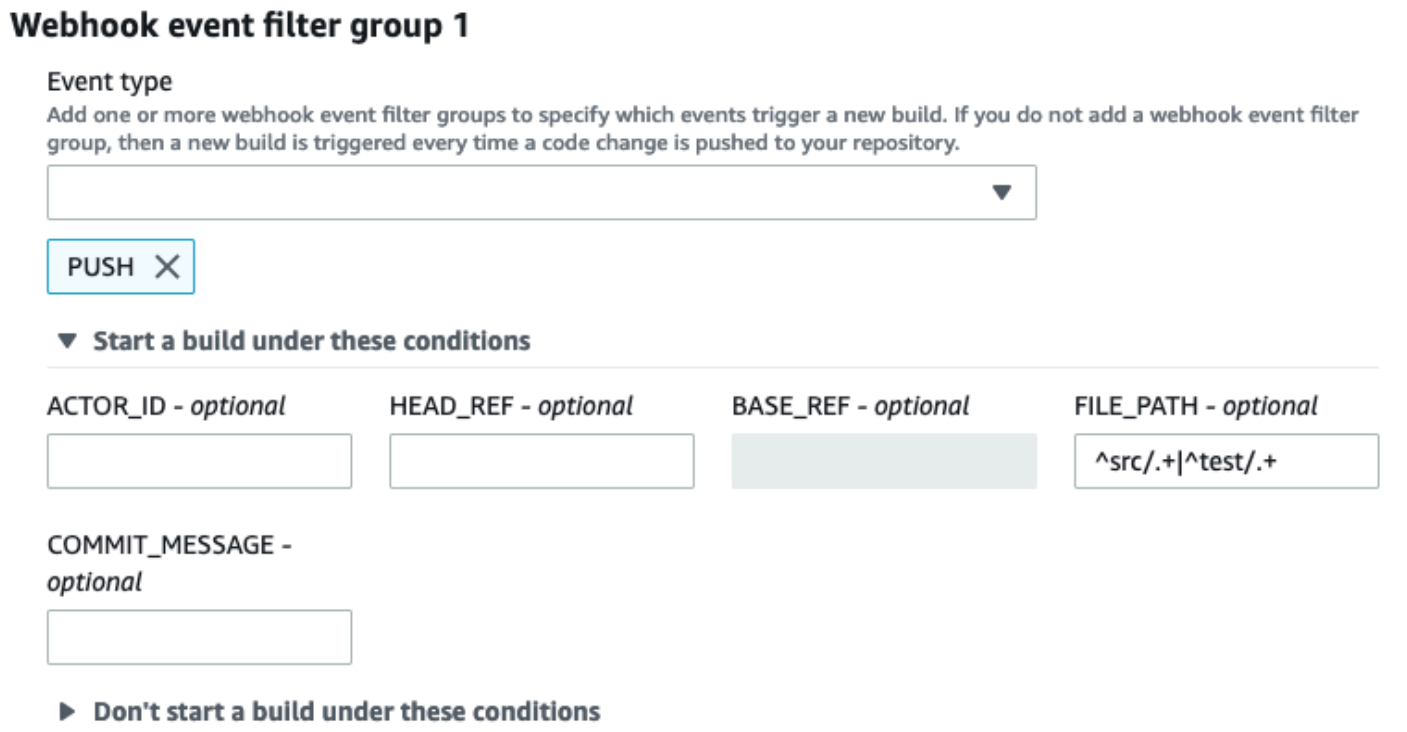
この例では、ウェブフックフィルタグループは、タグイベントを除くすべてのリクエストに対してビルドをトリガーします。



この例では、ウェブフックフィルタグループは、正規表現 `^buildspec.*` に一致する名前のファイルが変更された場合にのみビルドをトリガーします。

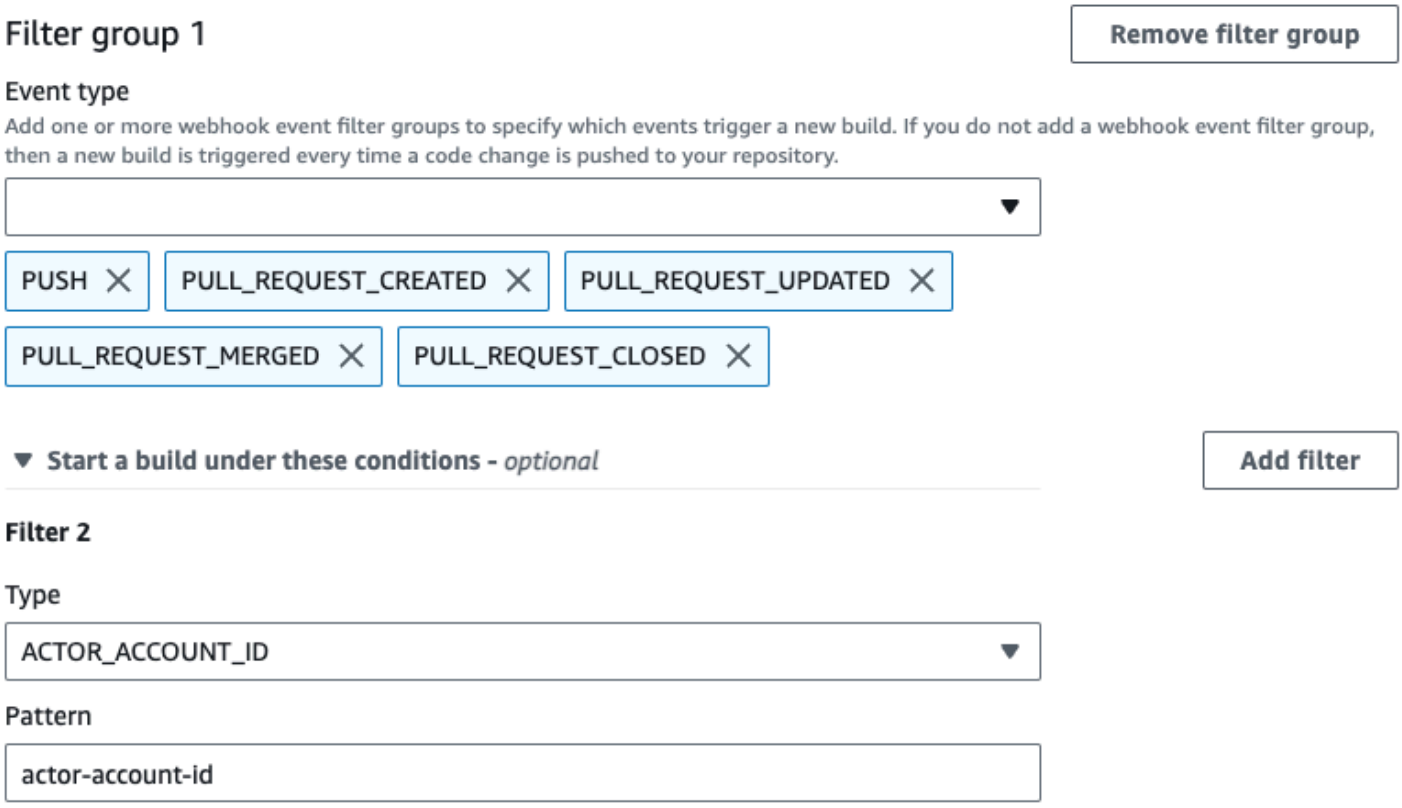


この例で、Webhook フィルターグループは、ファイルが `src` または `test` フォルダーで変更された場合にのみ、ビルドをトリガーします。

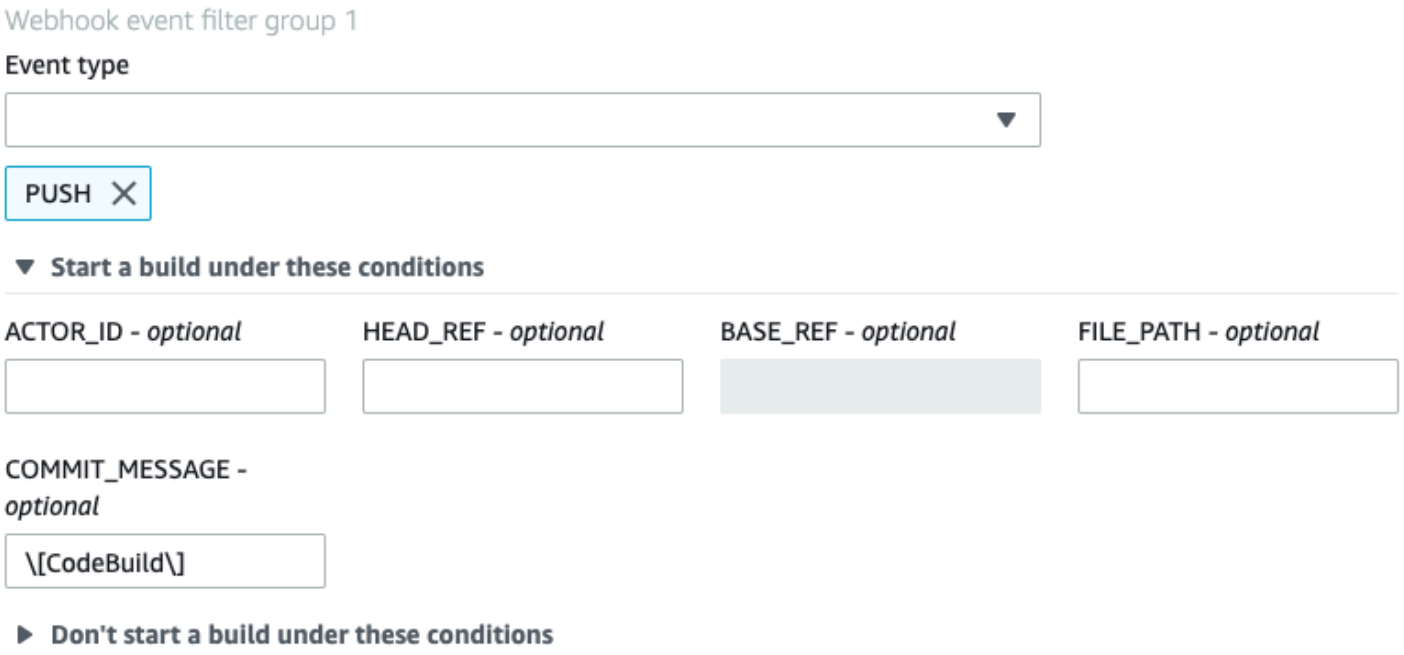


この例では、正規表現 `actor-account-id` と一致するアカウント ID を持たない Bitbucket ユーザーが変更を行った場合にのみ、ウェブフックフィルタグループがビルドをトリガーします。

> **Note**
>
> Bitbucket アカウント ID の検索方法については、「https://api.bitbucket.org/2.0/users/*user-name*」を参照してください。ここで、*user-name* は、Bitbucket のユーザー名を表します。



この例では、HEAD コミットメッセージが正規表現 `\[CodeBuild\]` に一致する場合に、Webhook フィルタグループがプッシュイベントのビルドをトリガーします。



Bitbucket ウェブフックイベントのフィルタリング (SDK)

AWS CodeBuild SDK を使用してウェブフックイベントをフィルタリングするには、`CreateWebhook`または `UpdateWebhook` API メソッドのリクエスト構文で `filterGroups`フィールドを使用します。詳細については、 CodeBuild API リファレンスのWebhookFilter「」を参照してください。

プルリクエストに対してのみビルドをトリガーするウェブフックフィルタを作成するには、以下をリクエスト構文に挿入します。

```
"filterGroups": [
  [
    {
      "type": "EVENT",
      "pattern": "PULL_REQUEST_CREATED, PULL_REQUEST_UPDATED, PULL_REQUEST_MERGED,
 PULL_REQUEST_CLOSED"
    }
  ]
]
```

指定されたブランチに対してのみビルドをトリガーするウェブフックフィルタを作成するには、`pattern` パラメータを使用して、ブランチ名をフィルタリングするよう正規表現を指定します。2 つのフィルタグループの例を使用した場合、ビルドは一方または両方が true と評価されるとトリガーされます。

- 最初のフィルタグループでは、正規表現 ^refs/heads/main$ に一致する Git 参照と ^refs/heads/myBranch$ に一致するヘッド参照を含むブランチで作成または更新されたプルリクエストを指定します。
- 2 番目のフィルタグループでは、正規表現 ^refs/heads/myBranch$ に一致する Git 参照を含むブランチでプッシュリクエストを指定します。

```
"filterGroups": [
  [
    {
      "type": "EVENT",
      "pattern": "PULL_REQUEST_CREATED, PULL_REQUEST_UPDATED, PULL_REQUEST_CLOSED"
    },
    {
      "type": "HEAD_REF",
      "pattern": "^refs/heads/myBranch$"
    },
    {
      "type": "BASE_REF",
      "pattern": "^refs/heads/main$"
    }
  ],
  [
    {
      "type": "EVENT",
      "pattern": "PUSH"
    },
    {
      "type": "HEAD_REF",
      "pattern": "^refs/heads/myBranch$"
    }
  ]
]
```

excludeMatchedPattern パラメータを使用すると、ビルドをトリガーしないイベントを指定することができます。この例では、ビルドは、タグイベントを除くすべてのリクエストに対してトリガーされます。

```
"filterGroups": [
  [
    {
      "type": "EVENT",
```

```
      "pattern": "PUSH, PULL_REQUEST_CREATED, PULL_REQUEST_UPDATED,
 PULL_REQUEST_MERGED, PULL_REQUEST_CLOSED"
    },
    {
      "type": "HEAD_REF",
      "pattern": "^refs/tags/.*",
      "excludeMatchedPattern": true
    }
  ]
]
```

アカウント ID `actor-account-id` を持つ Bitbucket ユーザーによって変更が行われた場合にのみビルドをトリガーするフィルタを作成できます。

> **Note**
>
> Bitbucket アカウント ID の検索方法については、「https://api.bitbucket.org/2.0/users/*user-name*」を参照してください。ここで、*user-name* は、Bitbucket のユーザー名を表します。

```
"filterGroups": [
  [
    {
      "type": "EVENT",
      "pattern": "PUSH, PULL_REQUEST_CREATED, PULL_REQUEST_UPDATED,
 PULL_REQUEST_MERGED, PULL_REQUEST_CLOSED"
    },
    {
      "type": "ACTOR_ACCOUNT_ID",
      "pattern": "actor-account-id"
    }
  ]
]
```

引数 `pattern` の正規表現に一致する名前のファイルが変更される場合にのみビルドをトリガーするフィルタを作成することができます。この例のフィルタグループでは、正規表現 `^buildspec.*` に一致する名前のファイルが変更された場合にのみビルドをトリガーするよう指定します。

```
"filterGroups": [
  [
    {
```

```
      "type": "EVENT",
      "pattern": "PUSH"
    },
    {
      "type": "FILE_PATH",
      "pattern": "^buildspec.*"
    }
  ]
]
```

この例で、フィルターグループは、ファイルが src または test フォルダーで変更された場合にのみ、ビルドをトリガーするように指定しています。

```
"filterGroups": [
    [
        {
            "type": "EVENT",
            "pattern": "PUSH"
        },
        {
            "type": "FILE_PATH",
            "pattern": "^src/.+|^test/.+"
        }
    ]
]
```

HEAD コミットメッセージがパターン引数の正規表現に一致する場合にのみビルドをトリガーするフィルタを作成できます。この例のフィルタグループでは、プッシュイベントの HEAD コミットメッセージが正規表現 \[CodeBuild\] に一致する場合にのみビルドをトリガーするよう指定します。

```
  "filterGroups": [
    [
      {
        "type": "EVENT",
        "pattern": "PUSH"
      },
      {
        "type": "COMMIT_MESSAGE",
        "pattern": "\[CodeBuild\]"
      }
    ]
```

```
  ]
```

Bitbucket ウェブフックイベントのフィルタリング (AWS CloudFormation)

AWS CloudFormation テンプレートを使用してウェブフックイベントをフィルタリングするには、AWS CodeBuild プロジェクトの `FilterGroups`プロパティを使用します。以下の YAML 形式の AWS CloudFormation テンプレート部分によって、2 つのフィルタグループが作成されます。また、一方または両方が true と評価されると、ビルドがトリガーされます。

- 最初のフィルタグループでは、アカウント ID `^refs/heads/main$` を持たない Bitbucket ユーザーが、正規表現 `12345` と一致する Git 参照名を持つブランチに対してプルリクエストを作成または更新することを指定します。
- 2 番目のフィルタグループでは、正規表現 `^refs/heads/.*` と一致する Git 参照名を持つブランチに対するプッシュリクエストを作成することを指定します。
- 3 番目のフィルタグループでは、正規表現 `\[CodeBuild\]` に一致する HEAD コミットメッセージを使用してプッシュリクエストを指定します。

```
CodeBuildProject:
  Type: AWS::CodeBuild::Project
  Properties:
    Name: MyProject
    ServiceRole: service-role
    Artifacts:
      Type: NO_ARTIFACTS
    Environment:
      Type: LINUX_CONTAINER
      ComputeType: BUILD_GENERAL1_SMALL
      Image: aws/codebuild/standard:5.0
    Source:
      Type: BITBUCKET
      Location: source-location
    Triggers:
      Webhook: true
      FilterGroups:
        - - Type: EVENT
            Pattern: PULL_REQUEST_CREATED,PULL_REQUEST_UPDATED
          - Type: BASE_REF
            Pattern: ^refs/heads/main$
            ExcludeMatchedPattern: false
          - Type: ACTOR_ACCOUNT_ID
```

```
            Pattern: 12345
            ExcludeMatchedPattern: true
        - - Type: EVENT
            Pattern: PUSH
          - Type: HEAD_REF
            Pattern: ^refs/heads/.*
          - Type: FILE_PATH
            Pattern: READ_ME
            ExcludeMatchedPattern: true
        - - Type: EVENT
            Pattern: PUSH
          - Type: COMMIT_MESSAGE
            Pattern: \[CodeBuild\]
          - Type: FILE_PATH
            Pattern: ^src/.+|^test/.+
```

GitHub ウェブフックイベント

Webhook フィルターグループを使用して、ビルドをトリガーする GitHub Webhook イベントを指定できます。たとえば、特定のブランチへの変更に対してのみビルドをトリガーするように指定できます。

ビルドをトリガーするウェブフックイベントを指定するには、ウェブフックフィルタグループを 1 つ以上作成できます。任意のフィルターグループが true と評価されると、ビルドがトリガーされます。これは、グループ内のすべてのフィルターが true と評価されたときに発生します。フィルタグループを作成する際、以下を指定します。

イベント

では
GitHub、、、PUSH、、PULL_REQUEST_CREATED、、PULL_REQUEST_UPDATED、PULL_REQUEST_RE
PRERELEASEDの 1 つ以上のイベントを選択できますWORKFLOW_JOB_QUEUED。ウェブフック
のイベントタイプは、ウェブフックペイロードの X-GitHub-Event ヘッダーに含まれています。X-GitHub-Event ヘッダーで、pull_request または push が表示される場合があります。プルリクエストイベントの場合、このタイプはウェブフックイベントペイロードの action フィールドに含まれています。以下の表に示すのは、X-GitHub-Event ヘッダー値とウェブフックのプルリクエストペイロードの action フィールドが、利用可能なイベントタイプにマッピングされる方法を示しています。

X-GitHub-Event ヘッダー値	ウェブフックイベントペイロードの **action** 値	イベントタイプ
pull_request	opened	PULL_REQUEST_CREATED
pull_request	reopened	PULL_REQUEST_REOPENED
pull_request	synchronize	PULL_REQUEST_UPDATED
pull_request	closed、および merged フィールドは true	PULL_REQUEST_MERGED
pull_request	closed、および merged フィールドは false	PULL_REQUEST_CLOSED
push	該当なし	PUSH
release	リリース済み	RELEASED
release	プレリリース	PRERELEASED
workflow_job	queued	WORKFLOW_JOB_QUEUED

> **Note**
>
> PULL_REQUEST_REOPENED イベントタイプは、 および GitHub Enterprise Server でのみ使用できます GitHub。RELEASED、PRERELEASED、および WORKFLOW_JOB_QUEUEDイベントタイプは、 GitHub でのみ使用できます。WORKFLOW_JOB_QUEUED の詳細については、「チュートリアル: CodeBuild セルフホスト GitHub アクションランナーを設定する」をご参照ください。

1 つ以上のオプションフィルタ

フィルタを指定するには、正規表現を使用します。ビルドをトリガーするイベントでは、関連付けられているグループ内のすべてのフィルターが true と評価される必要があります。

ACTOR_ACCOUNT_ID (コンソール内の ACTOR_ID)

GitHub または GitHub Enterprise Server アカウント ID が正規表現パターンと一致すると、Webhook イベントによってビルドがトリガーされます。この値は、ウェブフックペイロードの sender オブジェクトの id プロパティで見つかります。

HEAD_REF

ヘッドリファレンスが正規表現パターンと一致すると、ウェブフックイベントによりビルドがトリガーされます (例: refs/heads/branch-name または refs/tags/tag-name)。プッシュイベントの場合、参照名はウェブフックペイロードの ref プロパティで見つかります。プルリクエストイベントの場合、ブランチ名はウェブフックペイロードの head オブジェクトの ref プロパティで見つかります。

BASE_REF

基本参照が正規表現パターンと一致するとウェブフックイベントによってビルドがトリガーされます。(例 refs/heads/branch-name) BASE_REF フィルタは、プルリクエストイベントでのみ使用できます。ブランチ名は、ウェブフックペイロードで base オブジェクトの ref プロパティで見つかります。

FILE_PATH

変更されたファイルのパスが正規表現パターンと一致すると、ビルドがウェブフックイベントでトリガーされます。FILE_PATH フィルターは、 GitHub プッシュおよびプルリクエストイベントと GitHub Enterprise Server プッシュイベントで使用できます。Enterprise Server プルリクエストイベントで GitHubは使用できません。

COMMIT_MESSAGE

HEAD コミットメッセージが正規表現パターンに一致する場合に、Webhook はビルドをトリガーします。COMMIT_MESSAGE フィルターは、 GitHub プッシュリクエストイベントとプルリクエストイベント、および GitHub Enterprise Server プッシュイベントで使用できます。Enterprise Server プルリクエストイベントで GitHubは使用できません。

TAG_NAME

リリースのタグ名が正規表現パターンと一致すると、ウェブフックによってビルドがトリガーされます。TAG_NAME フィルターは、 GitHub リリース済みおよびプレリリース済みのリクエストイベントで使用できます。

`RELEASE_NAME`

リリース名が正規表現パターンと一致すると、ウェブフックによってビルドがトリガーされます。`RELEASE_NAME` フィルターは、 GitHub リリース済みおよびプレリリース済みのリクエストイベントで使用できます。

`WORKFLOW_NAME`

ワークフロー名が正規表現パターンと一致すると、ウェブフックによってビルドがトリガーされます。`WORKFLOW_NAME` フィルターは、 GitHub アクションワークフロージョブキューに入れられたリクエストイベントで使用できます。

> **Note**
>
> ウェブフックペイロードは、 GitHub リポジトリのウェブフック設定で確認できます。

トピック

- GitHub ウェブフックイベントのフィルタリング (コンソール)
- GitHub ウェブフックイベントのフィルタリング (SDK)
- GitHub ウェブフックイベントのフィルタリング (AWS CloudFormation)

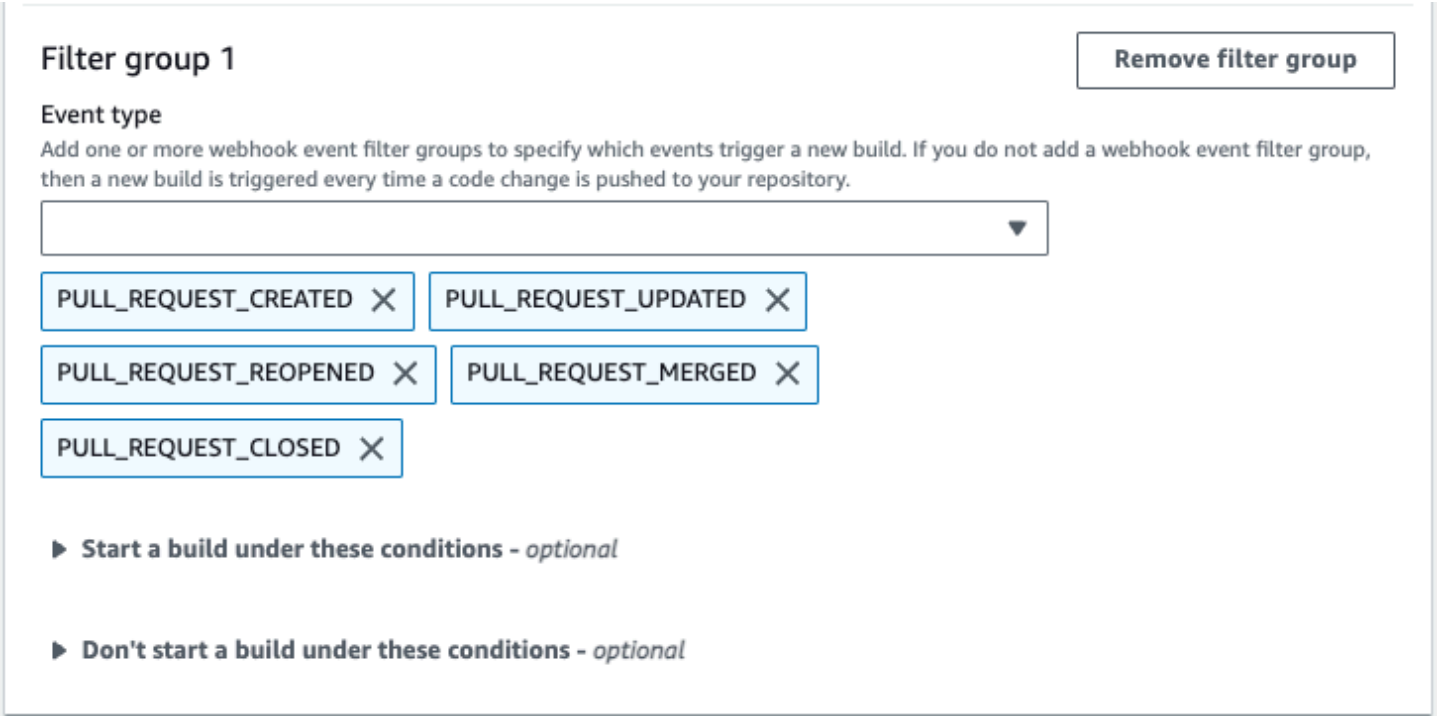
GitHub ウェブフックイベントのフィルタリング (コンソール)

[プライマリソース Webhook イベント] で、以下を選択します。このセクションは、ソースリポジトリのアカウントで GitHubリポジトリを選択した場合にのみ使用できます。

1. プロジェクトの作成時に [コードの変更がこのレポジトリにプッシュされるたびに再構築する] を選択します。
2. [イベントタイプ] から、1 つ以上のイベントを選択します。
3. イベントでビルドをトリガーされた時間をフィルタリングするには、[これらの条件でビルドを開始する] で、1 つ以上のオプションフィルタを追加します。
4. イベントがトリガーされていない時間をフィルタリングするには、[これらの条件でビルドを開始しない] で、1 つ以上のオプションフィルタを追加します。
5. 別のフィルタグループを追加する必要がある場合、[フィルタグループの追加] を選択します。

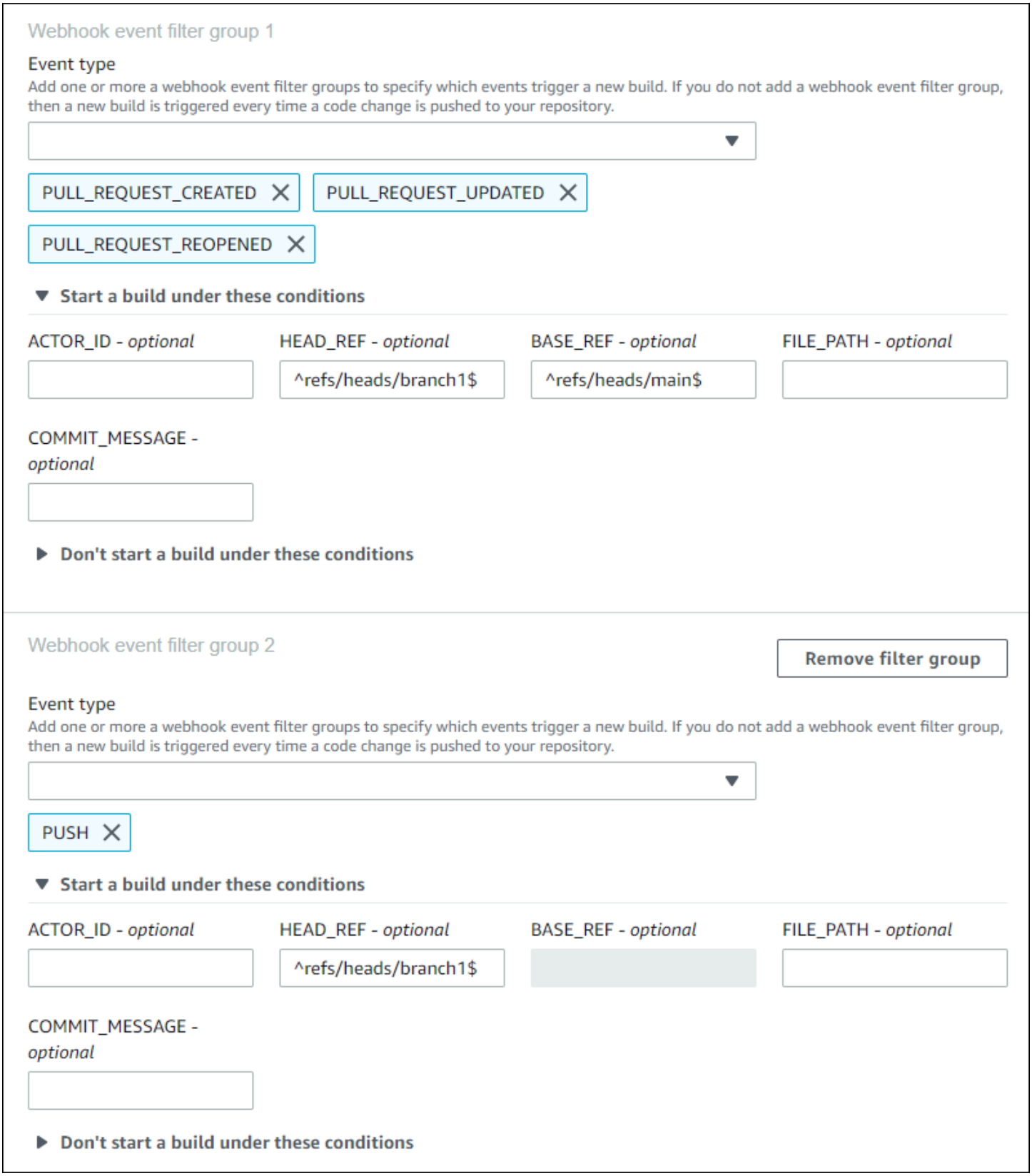
詳細については、 API リファレンスWebhookFilterのビルドプロジェクトの作成 (コンソール)「」および「」を参照してください。 AWS CodeBuild

この例では、ウェブフックフィルタグループは、プルリクエストに対してのみビルドをトリガーします。

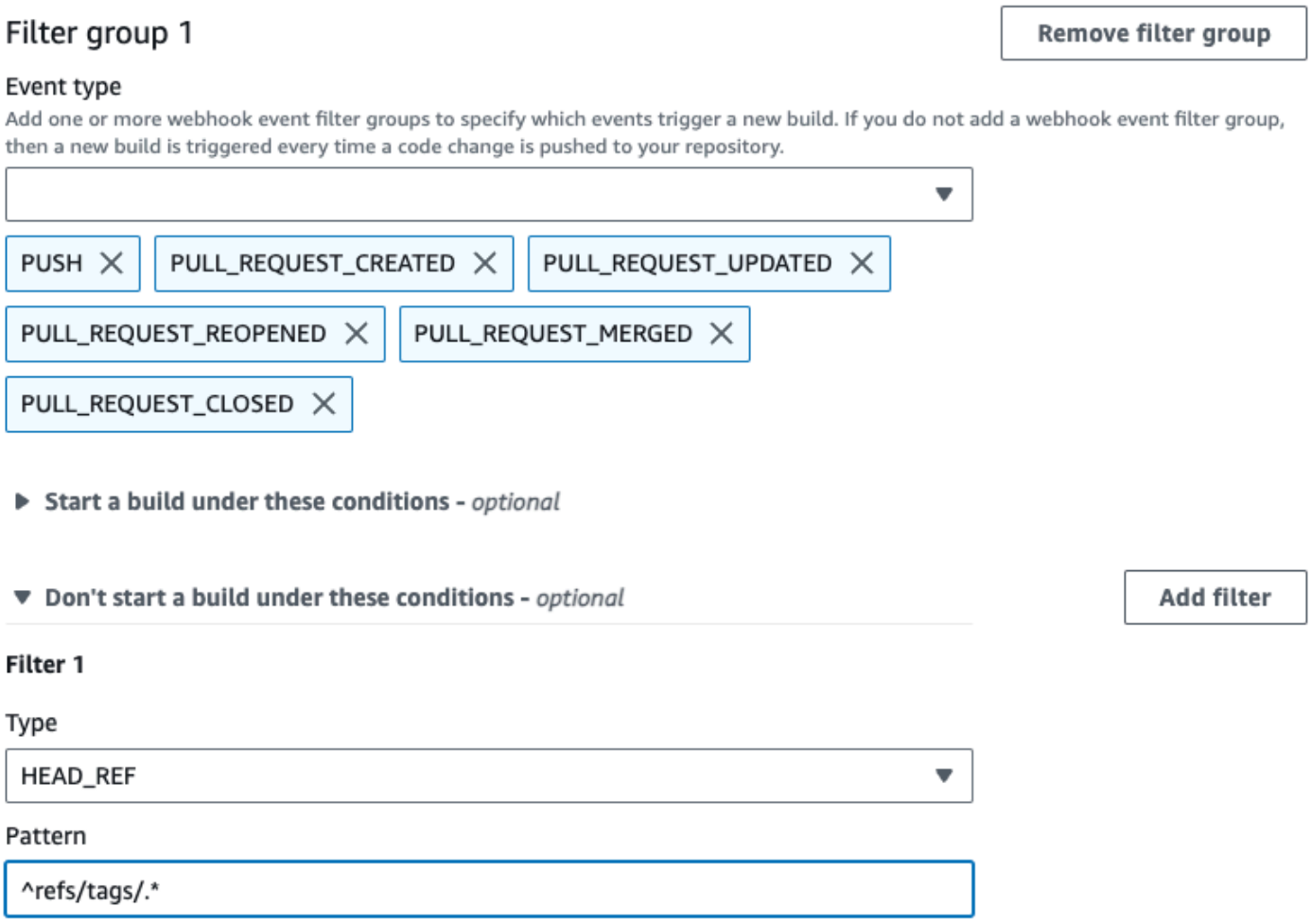


2 つのウェブフックフィルタグループの例を使用した場合、ビルドは一方または両方が true と評価されるとトリガーされます。

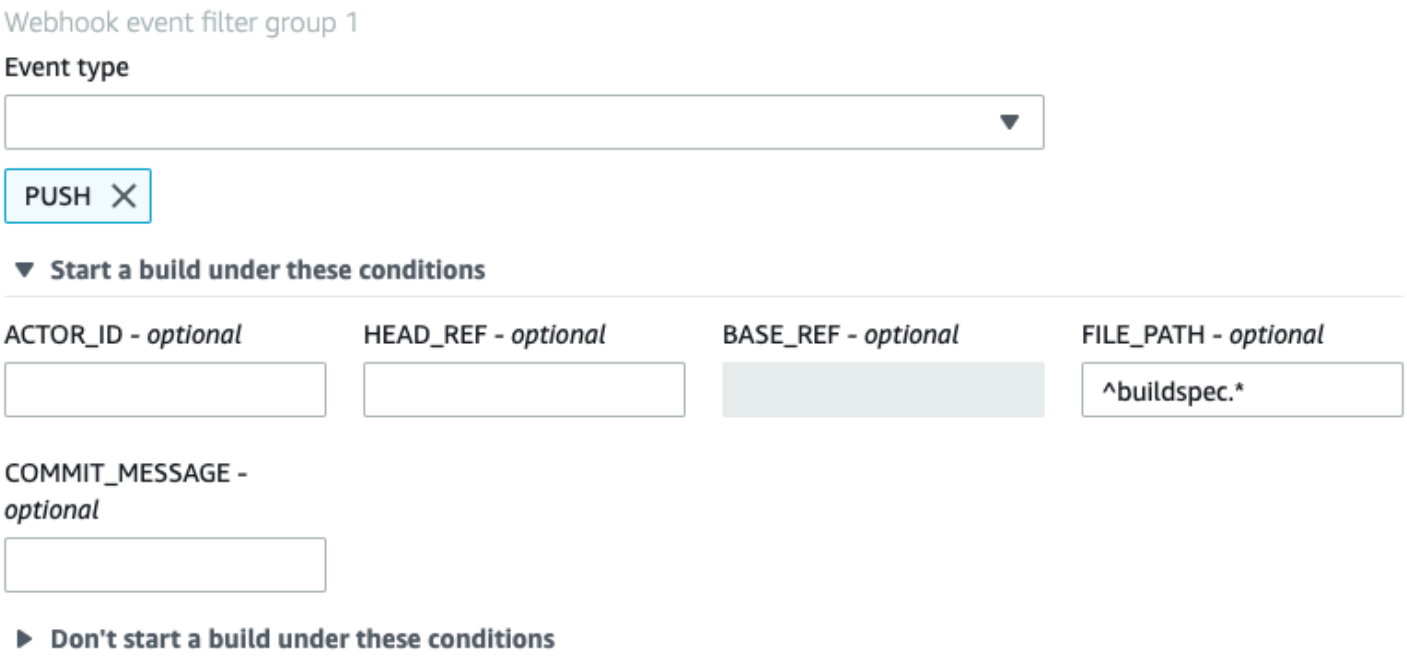
- 最初のフィルタグループでは、正規表現 `^refs/heads/main$` と一致する Git 参照名および `^refs/heads/branch1$` と一致するヘッド参照を持つブランチに対してプルリクエストを作成、更新、または再開することを指定します。
- 2 番目のフィルタグループでは、正規表現 `^refs/heads/branch1$` に一致する Git 参照を含むブランチでプッシュリクエストを指定します。


Webhook event filter group 1
Event type
Add one or more a webhook event filter groups to specify which events trigger a new build. If you do not add a webhook event filter group, then a new build is triggered every time a code change is pushed to your repository.
PULL_REQUEST_CREATED
PULL_REQUEST_UPDATED
PULL_REQUEST_REOPENED
Start a build under these conditions
ACTOR_ID - optional
HEAD_REF - optional
^refs/heads/branch1$
BASE_REF - optional
^refs/heads/main$
FILE_PATH - optional
COMMIT_MESSAGE - optional
Don't start a build under these conditions
Webhook event filter group 2
Remove filter group
Event type
Add one or more a webhook event filter groups to specify which events trigger a new build. If you do not add a webhook event filter group, then a new build is triggered every time a code change is pushed to your repository.
PUSH
Start a build under these conditions
ACTOR_ID - optional
HEAD_REF - optional
^refs/heads/branch1$
BASE_REF - optional
FILE_PATH - optional
COMMIT_MESSAGE - optional
Don't start a build under these conditions

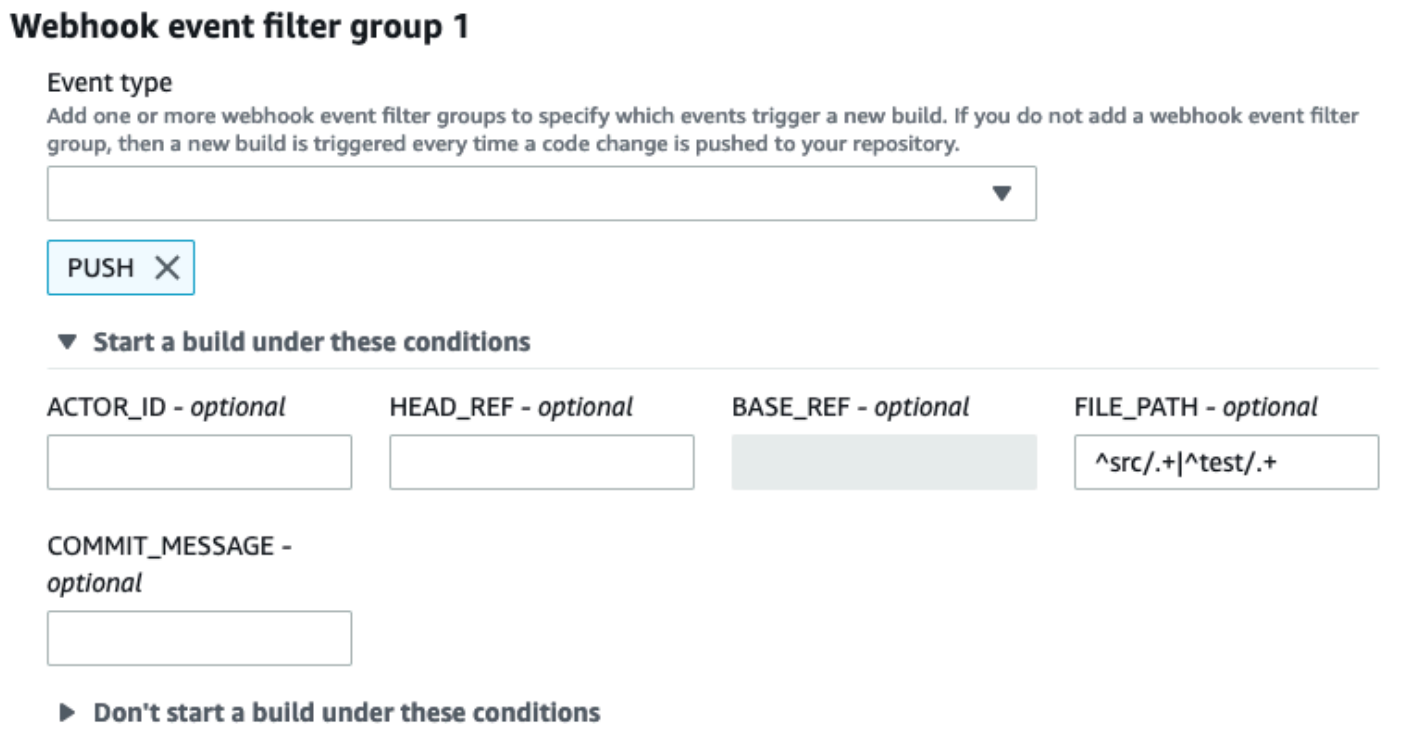
この例では、ウェブフックフィルタグループは、タグイベントを除くすべてのリクエストに対してビルドをトリガーします。



この例では、ウェブフックフィルタグループは、正規表現 ^buildspec.* に一致する名前のファイルが変更された場合にのみビルドをトリガーします。

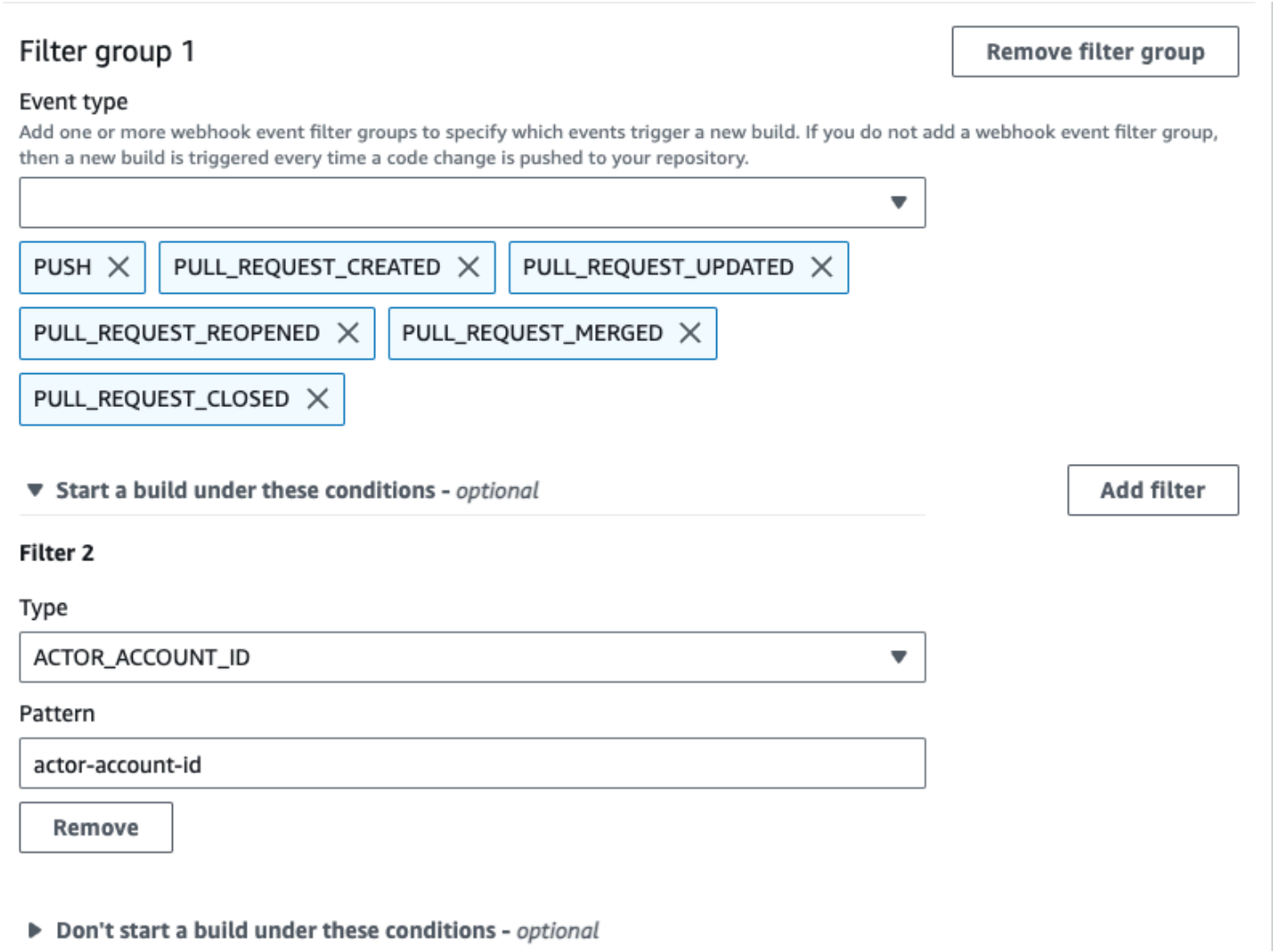


この例で、Webhook フィルターグループは、ファイルが `src` または `test` フォルダーで変更された場合にのみ、ビルドをトリガーします。

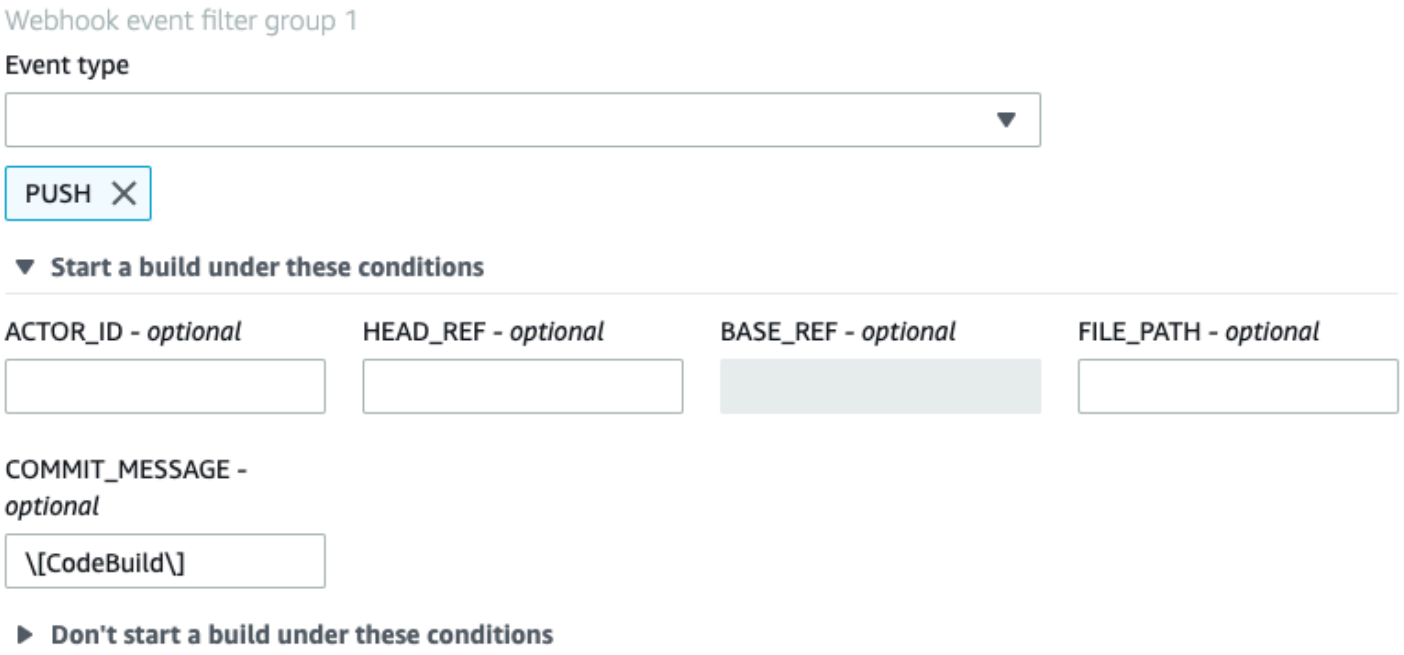


この例では、指定した GitHub または GitHub Enterprise Server ユーザーが正規表現 と一致するアカウント ID を使用して変更を行った場合にのみ、ウェブフックフィルタグループがビルドをトリガーしますactor-account-id。

> **Note**
>
> GitHub アカウント ID を検索する方法については、https://api.github.com/users/*user-name* を参照してください。*user-name* は GitHub ユーザー名です。

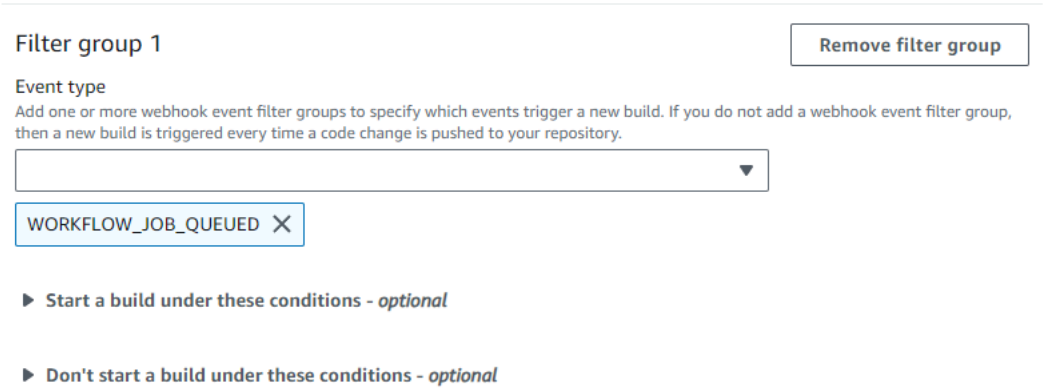


この例では、HEAD コミットメッセージが正規表現 \[CodeBuild\] に一致する場合に、Webhook フィルタグループがプッシュイベントのビルドをトリガーします。

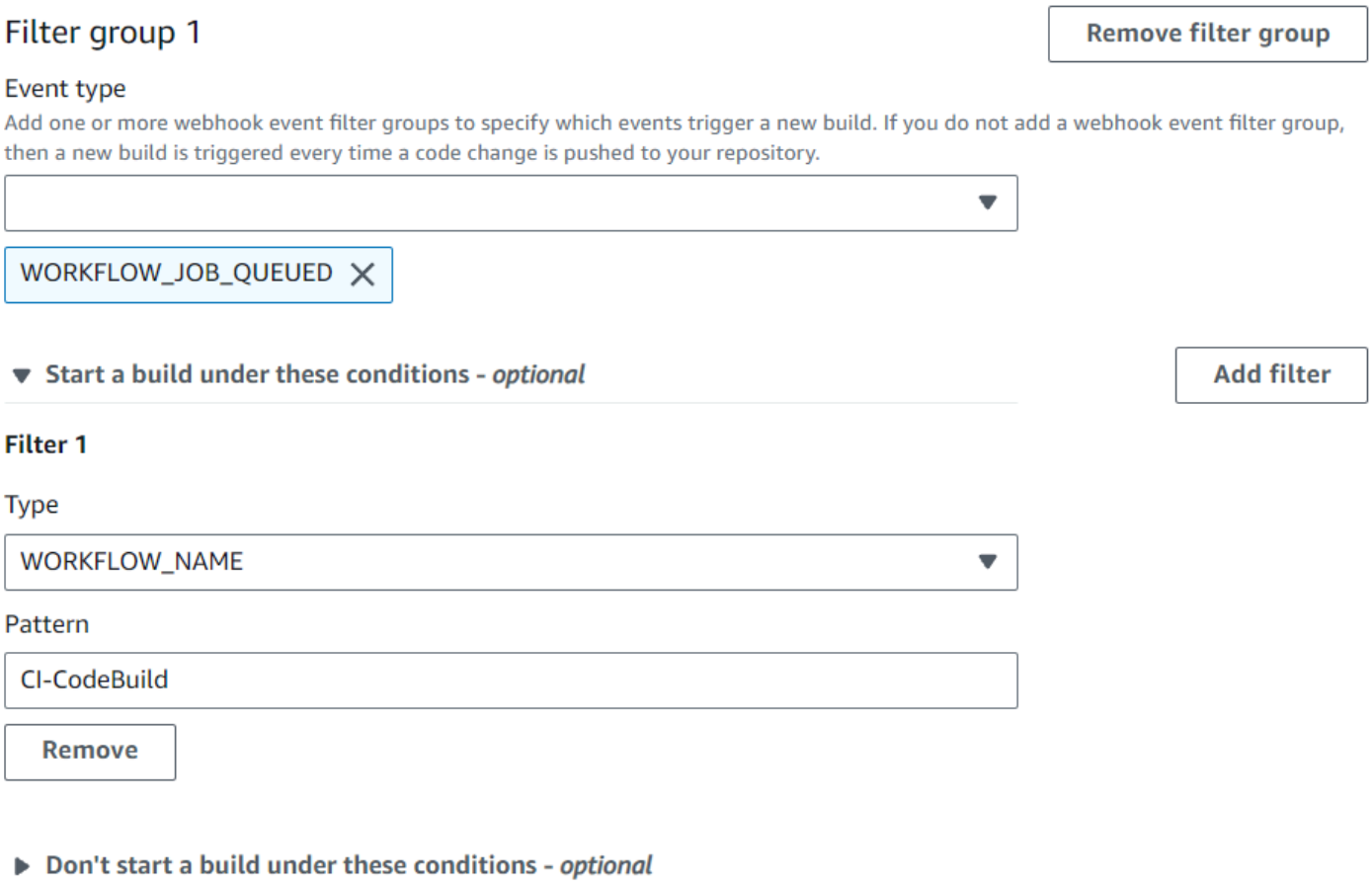


この例では、Webhook フィルターグループは GitHub 、アクションワークフロージョブイベントのみのビルドをトリガーします。

> **Note**
>
> CodeBuild は、ウェブフックに WORKFLOW_JOB_QUEUED イベントフィルターを含むフィルターグループがある場合にのみ、 GitHub アクションワークフロージョブを処理します。



この例では、ウェブフックフィルタグループは、正規表現 に一致するワークフロー名のビルドをトリガーしますCI-CodeBuild。



GitHub ウェブフックイベントのフィルタリング (SDK)

AWS CodeBuild SDK を使用してウェブフックイベントをフィルタリングするには、`CreateWebhook`または `UpdateWebhook` API メソッドのリクエスト構文で `filterGroups`フィールドを使用します。詳細については、 CodeBuild API リファレンスのWebhookFilter「」を参照してください。

プルリクエストに対してのみビルドをトリガーするウェブフックフィルタを作成するには、以下をリクエスト構文に挿入します。

```
"filterGroups": [
   [
        {
            "type": "EVENT",
            "pattern": "PULL_REQUEST_CREATED, PULL_REQUEST_UPDATED,
 PULL_REQUEST_REOPENED, PULL_REQUEST_MERGED, PULL_REQUEST_CLOSED"
        }
    ]
```

```
]
```

指定されたブランチに対してのみビルドをトリガーするウェブフックフィルタを作成するには、pattern パラメータを使用して、ブランチ名をフィルタリングするよう正規表現を指定します。2 つのフィルタグループの例を使用した場合、ビルドは一方または両方が true と評価されるとトリガーされます。

- 最初のフィルタグループでは、正規表現 ^refs/heads/main$ と一致する Git 参照名および ^refs/heads/myBranch$ と一致するヘッド参照を持つブランチに対してプルリクエストを作成、更新、または再開することを指定します。
- 2 番目のフィルタグループでは、正規表現 ^refs/heads/myBranch$ に一致する Git 参照を含むブランチでプッシュリクエストを指定します。

```
"filterGroups": [
    [
        {
            "type": "EVENT",
            "pattern": "PULL_REQUEST_CREATED, PULL_REQUEST_UPDATED,
 PULL_REQUEST_REOPENED"
        },
        {
            "type": "HEAD_REF",
            "pattern": "^refs/heads/myBranch$"
        },
        {
            "type": "BASE_REF",
            "pattern": "^refs/heads/main$"
        }
    ],
    [
        {
            "type": "EVENT",
            "pattern": "PUSH"
        },
        {
            "type": "HEAD_REF",
            "pattern": "^refs/heads/myBranch$"
        }
    ]
]
```

excludeMatchedPattern パラメータを使用すると、ビルドをトリガーしないイベントを指定することができます。たとえば、この例で、ビルドは、タグイベントを除くすべてのリクエストに対してトリガーされます。

```
"filterGroups": [
    [
        {
            "type": "EVENT",
            "pattern": "PUSH, PULL_REQUEST_CREATED, PULL_REQUEST_UPDATED,
 PULL_REQUEST_REOPENED, PULL_REQUEST_MERGED, PULL_REQUEST_CLOSED"
        },
        {
            "type": "HEAD_REF",
            "pattern": "^refs/tags/.*",
            "excludeMatchedPattern": true
        }
    ]
]
```

引数 pattern の正規表現に一致する名前のファイルが変更される場合にのみビルドをトリガーするフィルタを作成することができます。この例のフィルタグループでは、正規表現 ^buildspec.* に一致する名前のファイルが変更された場合にのみビルドをトリガーするよう指定します。

```
"filterGroups": [
    [
        {
            "type": "EVENT",
            "pattern": "PUSH"
        },
        {
            "type": "FILE_PATH",
            "pattern": "^buildspec.*"
        }
    ]
]
```

この例で、フィルターグループは、ファイルが src または test フォルダーで変更された場合にのみ、ビルドをトリガーするように指定しています。

```
"filterGroups": [
    [
```

```
        {
            "type": "EVENT",
            "pattern": "PUSH"
        },
        {
            "type": "FILE_PATH",
            "pattern": "^src/.+|^test/.+"
        }
    ]
]
```

指定された GitHub またはアカウント ID を持つ GitHub Enterprise Server ユーザーによって変更が行われた場合にのみビルドをトリガーするフィルターを作成できますactor-account-id。

> **Note**
>
> GitHub アカウント ID を検索する方法については、https://api.github.com/users/*user-name* を参照してください。*user-name* は GitHub ユーザー名です。

```
"filterGroups": [
    [
        {
            "type": "EVENT",
            "pattern": "PUSH, PULL_REQUEST_CREATED, PULL_REQUEST_UPDATED,
 PULL_REQUEST_REOPENED, PULL_REQUEST_MERGED, PULL_REQUEST_CLOSED"
        },
        {
            "type": "ACTOR_ACCOUNT_ID",
            "pattern": "actor-account-id"
        }
    ]
]
```

HEAD コミットメッセージがパターン引数の正規表現に一致する場合にのみビルドをトリガーするフィルタを作成できます。この例のフィルタグループでは、プッシュイベントの HEAD コミットメッセージが正規表現 \[CodeBuild\] に一致する場合にのみビルドをトリガーするよう指定します。

```
"filterGroups": [
    [
```

```
        {
            "type": "EVENT",
            "pattern": "PUSH"
        },
        {
            "type": "COMMIT_MESSAGE",
            "pattern": "\[CodeBuild\]"
        }
    ]
]
```

Actions GitHub ワークフロージョブのビルドのみをトリガーするウェブフックフィルターを作成するには、リクエスト構文に以下を挿入します。

```
"filterGroups": [
   [
        {
            "type": "EVENT",
            "pattern": "WORKFLOW_JOB_QUEUED"
        }
    ]
]
```

GitHub ウェブフックイベントのフィルタリング (AWS CloudFormation)

AWS CloudFormation テンプレートを使用してウェブフックイベントをフィルタリングするには、AWS CodeBuild プロジェクトの `FilterGroups`プロパティを使用します。以下の YAML 形式の AWS CloudFormation テンプレート部分によって、2 つのフィルタグループが作成されます。また、一方または両方が true と評価されると、ビルドがトリガーされます。

- 最初のフィルターグループは、アカウント ID を持たない`^refs/heads/main$` GitHub ユーザーが、正規表現 と一致する Git 参照名を持つブランチでプルリクエストを作成または更新することを指定します`12345`。
- 2 番目のフィルターグループでは、正規表現 `READ_ME` に一致する Git 参照名を持つブランチで正規表現 `^refs/heads/.*` に一致する名前のファイルに対してプッシュリクエストが作成されることを指定します。
- 3 番目のフィルタグループでは、正規表現 `\[CodeBuild\]` に一致する HEAD コミットメッセージを使用してプッシュリクエストを指定します。
- 4 番目のフィルターグループは、正規表現 に一致するワークフロー名を持つ GitHub Actions ワークフロージョブリクエストを指定します`\[CI-CodeBuild\]`。

```
CodeBuildProject:
  Type: AWS::CodeBuild::Project
  Properties:
    Name: MyProject
    ServiceRole: service-role
    Artifacts:
      Type: NO_ARTIFACTS
    Environment:
      Type: LINUX_CONTAINER
      ComputeType: BUILD_GENERAL1_SMALL
      Image: aws/codebuild/standard:5.0
    Source:
      Type: GITHUB
      Location: source-location
    Triggers:
      Webhook: true
      FilterGroups:
        - - Type: EVENT
            Pattern: PULL_REQUEST_CREATED,PULL_REQUEST_UPDATED
          - Type: BASE_REF
            Pattern: ^refs/heads/main$
            ExcludeMatchedPattern: false
          - Type: ACTOR_ACCOUNT_ID
            Pattern: 12345
            ExcludeMatchedPattern: true
        - - Type: EVENT
            Pattern: PUSH
          - Type: HEAD_REF
            Pattern: ^refs/heads/.*
          - Type: FILE_PATH
            Pattern: READ_ME
            ExcludeMatchedPattern: true
        - - Type: EVENT
            Pattern: PUSH
          - Type: COMMIT_MESSAGE
            Pattern: \[CodeBuild\]
          - Type: FILE_PATH
            Pattern: ^src/.+|^test/.+
        - - Type: EVENT
            Pattern: WORKFLOW_JOB_QUEUED
          - Type: WORKFLOW_NAME
            Pattern: \[CI-CodeBuild\]
```

GitLab ウェブフックイベント

ウェブフックフィルタグループを使用して、ビルドをトリガーする GitLab ウェブフックイベントを指定できます。たとえば、特定のブランチへの変更に対してのみビルドをトリガーするように指定できます。

ビルドをトリガーするウェブフックイベントを指定するには、ウェブフックフィルタグループを 1 つ以上作成できます。任意のフィルターグループが true と評価されると、ビルドがトリガーされます。これは、グループ内のすべてのフィルターが true と評価されたときに発生します。フィルタグループを作成する際、以下を指定します。

イベント

では GitLab、次のイベントを 1 つ以上選択できます。

- PUSH
- PULL_REQUEST_CREATED
- PULL_REQUEST_UPDATED
- PULL_REQUEST_MERGED

ウェブフックのイベントタイプは、X-Event-Key フィールドのヘッダーに含まれています。次の表に、X-Event-Key ヘッダー値がイベントタイプにマッピングされる方法を示します。

> **Note**
>
> merged イベントタイプを使用する GitLab ウェブフックフィルタグループを作成する場合は、ウェブフック設定でPULL_REQUEST_MERGEDイベントを有効にする必要があります。

X-Event-Key ヘッダー値	イベントタイプ
repo:push	PUSH
pullrequest:created	PULL_REQUEST_CREATED
pullrequest:updated	PULL_REQUEST_UPDATED
pullrequest:fulfilled	PULL_REQUEST_MERGED

PULL_REQUEST_MERGED の場合、プルリクエストがスカッシュ戦略とマージされ、プルリクエストブランチが閉じられると、元のプルリクエストコミットは存在しなくなります。この場合、CODEBUILD_WEBHOOK_MERGE_COMMIT 環境変数には、圧縮されたマージコミットの識別子が含まれます。

1 つ以上のオプションフィルタ

フィルタを指定するには、正規表現を使用します。ビルドをトリガーするイベントでは、関連付けられているグループ内のすべてのフィルターが true と評価される必要があります。

ACTOR_ACCOUNT_ID (コンソール内の ACTOR_ID)

GitLab アカウント ID が正規表現パターンと一致すると、Webhook イベントによってビルドがトリガーされます。この値は、ウェブフックフィルタペイロードの actor オブジェクトの account_id プロパティに表示されます。

HEAD_REF

ヘッドリファレンスが正規表現パターンと一致すると (refs/heads/branch-name と refs/tags/tag-name など)、ウェブフックイベントによってビルドがトリガーされます。HEAD_REF フィルタは、ブランチまたはタグについて Git 参照名を評価します。ブランチ名またはタグ名は、ウェブフックペイロードの push オブジェクトにある、new オブジェクトの name フィールドに表示されます。プルリクエストイベントの場合、ブランチ名はウェブフックペイロードの source オブジェクトにある、branch オブジェクトの name フィールドに表示されます。

BASE_REF

基本参照が正規表現パターンと一致すると、ビルドがウェブフックイベントでトリガーされます。BASE_REF フィルタは、プルリクエストイベントでのみ使用できます (例: refs/heads/branch-name)。BASE_REF フィルタは、ブランチの Git 参照名を評価します。ブランチ名は、ウェブフックペイロードの destination オブジェクトにある、branch オブジェクトの name フィールドに表示されます。

FILE_PATH

変更されたファイルのパスが正規表現パターンに一致すると、ビルドが Webhook イベントでトリガーされます。

COMMIT_MESSAGE

HEAD コミットメッセージが正規表現パターンに一致する場合に、Webhook はビルドをトリガーします。

> **Note**
>
> ウェブフックペイロードは、 GitLab リポジトリのウェブフック設定で確認できます。

トピック

- GitLab ウェブフックイベントのフィルタリング (コンソール)
- GitLab ウェブフックイベントのフィルタリング (SDK)
- GitLab ウェブフックイベントのフィルタリング (AWS CloudFormation)

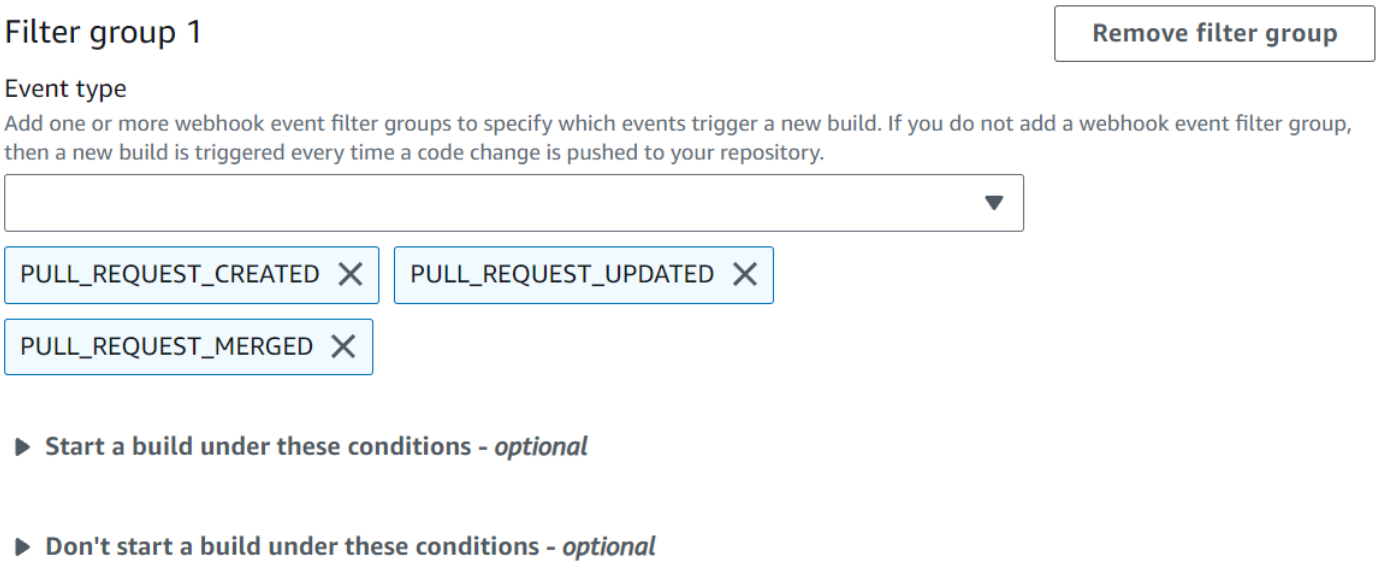
GitLab ウェブフックイベントのフィルタリング (コンソール)

を使用してウェブフックイベント AWS Management Console をフィルタリングするには：

1. プロジェクトの作成時に [コードの変更がこのレポジトリにプッシュされるたびに再構築する] を選択します。
2. [イベントタイプ] から、1 つ以上のイベントを選択します。
3. イベントでビルドをトリガーされた時間をフィルタリングするには、[これらの条件でビルドを開始する] で、1 つ以上のオプションフィルタを追加します。
4. イベントがトリガーされていない時間をフィルタリングするには、[これらの条件でビルドを開始しない] で、1 つ以上のオプションフィルタを追加します。
5. 別のフィルタグループを追加するには、[フィルタグループの追加] を選択します。

詳細については、 API リファレンスWebhookFilterのビルドプロジェクトの作成 (コンソール)「」および「」を参照してください。 AWS CodeBuild

この例では、ウェブフックフィルタグループは、プルリクエストに対してのみビルドをトリガーします。



2 つのフィルタグループの例を使用した場合、ビルドは一方または両方が true と評価されるとトリガーされます。

- 最初のフィルタグループでは、正規表現 `^refs/heads/main$` に一致する Git 参照と `^refs/heads/branch1!` に一致するヘッド参照を含むブランチで作成または更新されたプルリクエストを指定します。
- 2 番目のフィルタグループでは、正規表現 `^refs/heads/branch1$` に一致する Git 参照を含むブランチでプッシュリクエストを指定します。

Webhook event filter groups

A build is triggered if any filter group evaluates to true, which occurs when all the filters in the group evaluate to true.

Filter group 1

Remove filter group

Event type

Add one or more webhook event filter groups to specify which events trigger a new build. If you do not add a webhook event filter group, then a new build is triggered every time a code change is pushed to your repository.

PULL_REQUEST_CREATED ✕ PULL_REQUEST_UPDATED ✕

▼ **Start a build under these conditions - *optional***

Add filter

Filter 1

Type

HEAD_REF

Pattern

^refs/heads/branch1$

Remove

Filter 2

Type

BASE_REF

Pattern

^refs/heads/main

Remove

▶ **Don't start a build under these conditions - *optional***

Filter group 2

Remove filter group

Event type

Add one or more webhook event filter groups to specify which events trigger a new build. If you do not add a webhook event filter group, then a new build is triggered every time a code change is pushed to your repository.

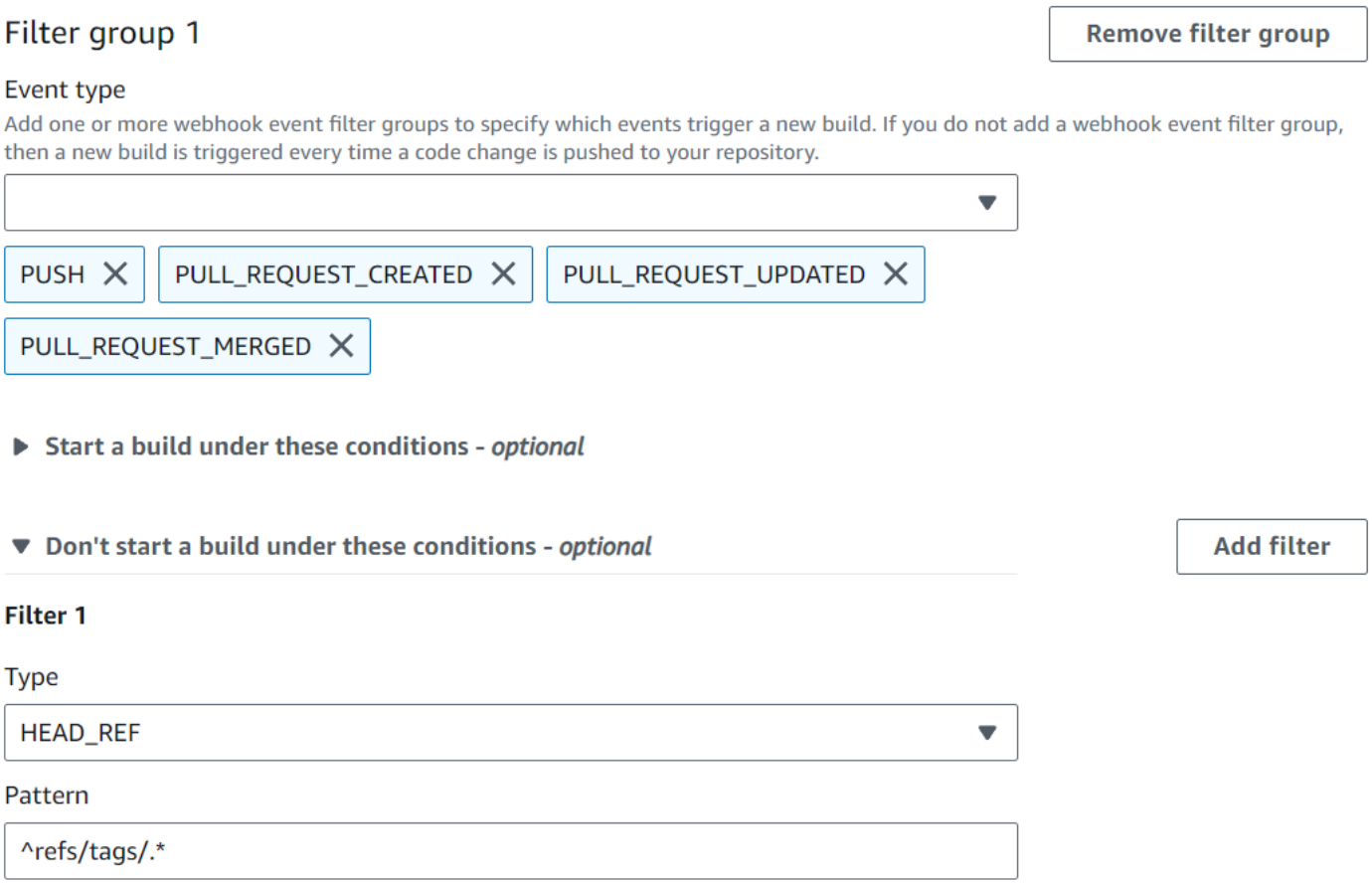
PUSH ✕

▼ **Start a build under these conditions - *optional***

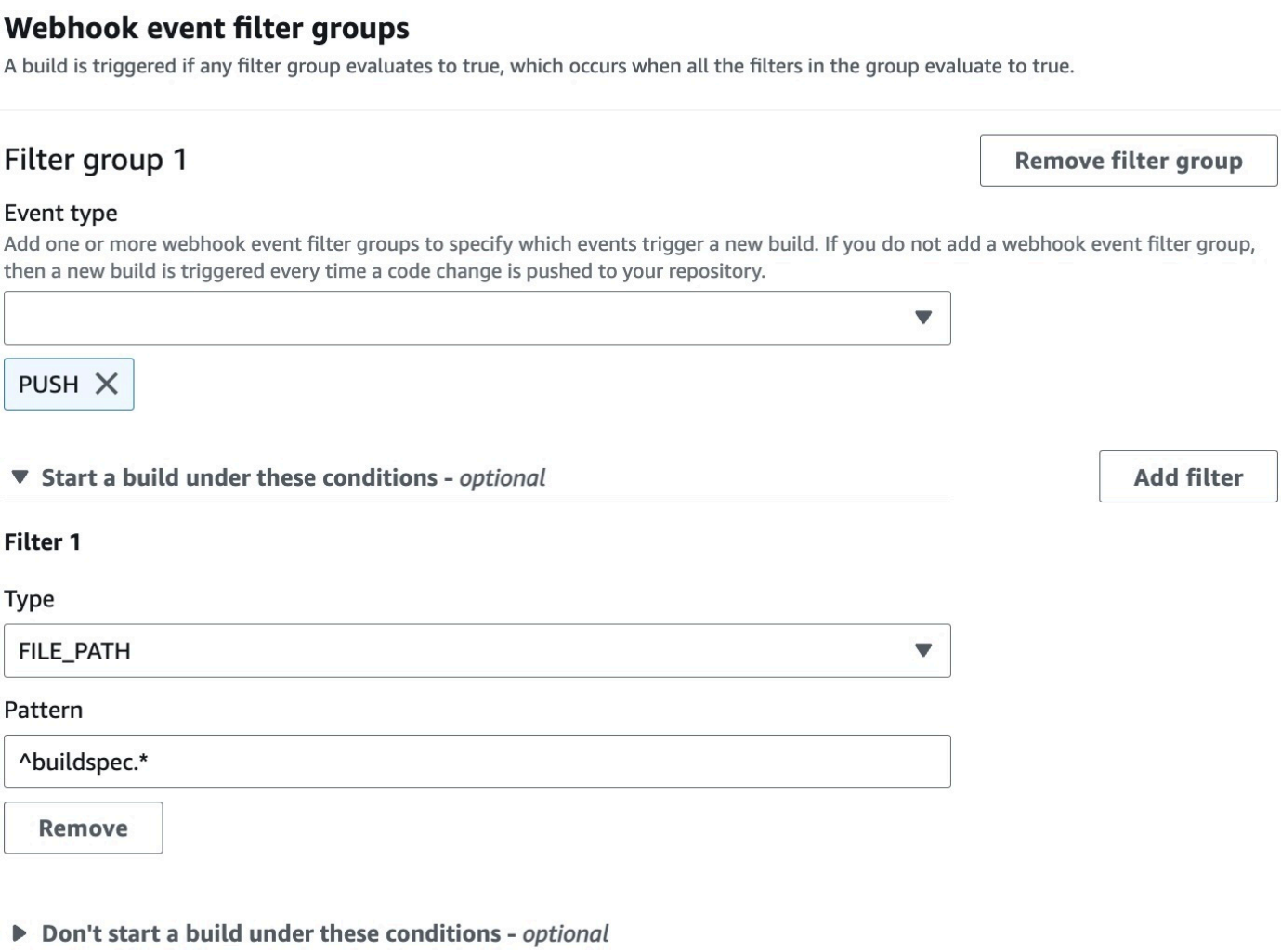
Add filter

Filter 1

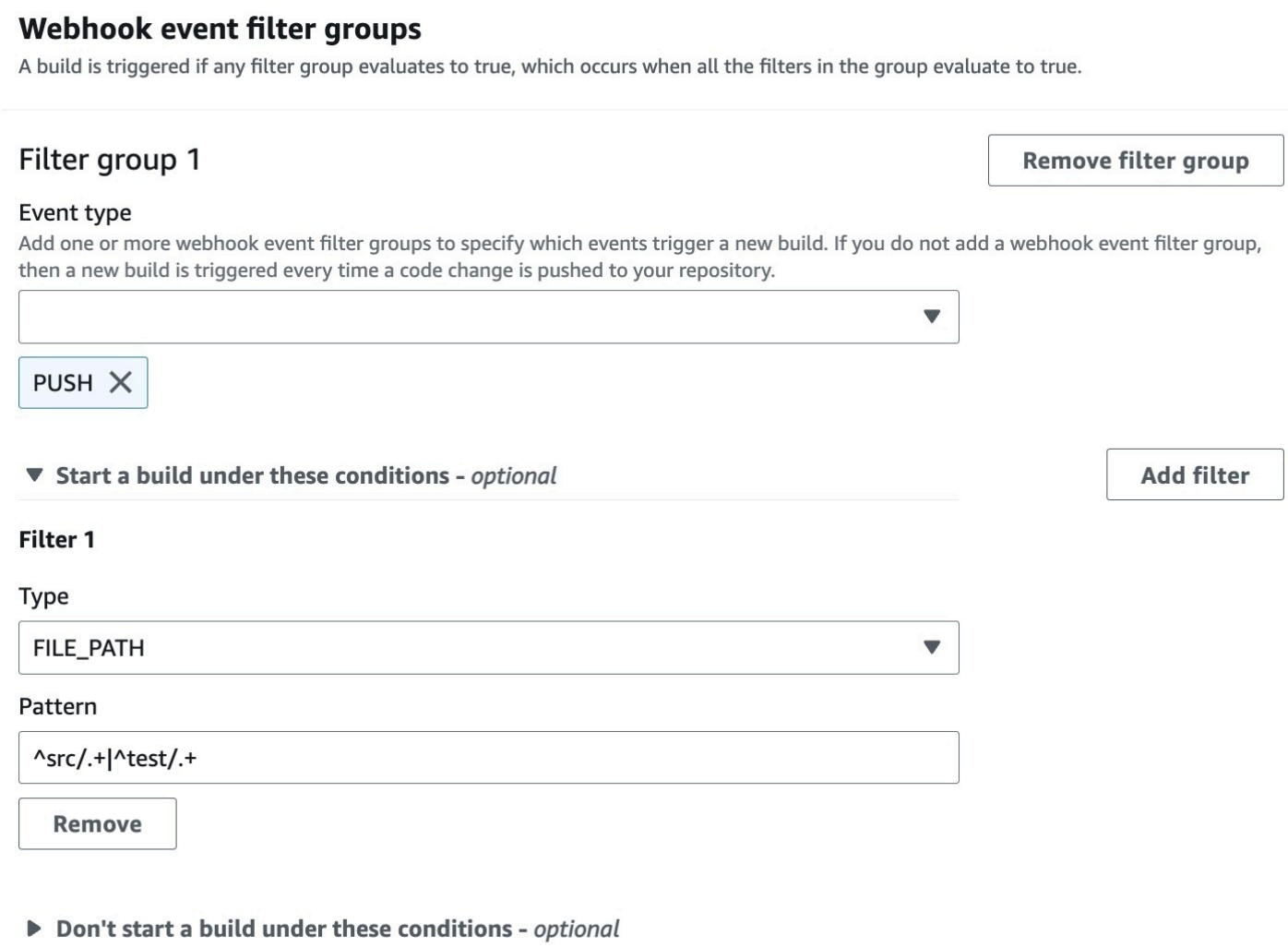
この例では、ウェブフックフィルタグループは、タグイベントを除くすべてのリクエストに対してビルドをトリガーします。



この例では、ウェブフックフィルタグループは、正規表現 `^buildspec.*` に一致する名前のファイルが変更された場合にのみビルドをトリガーします。

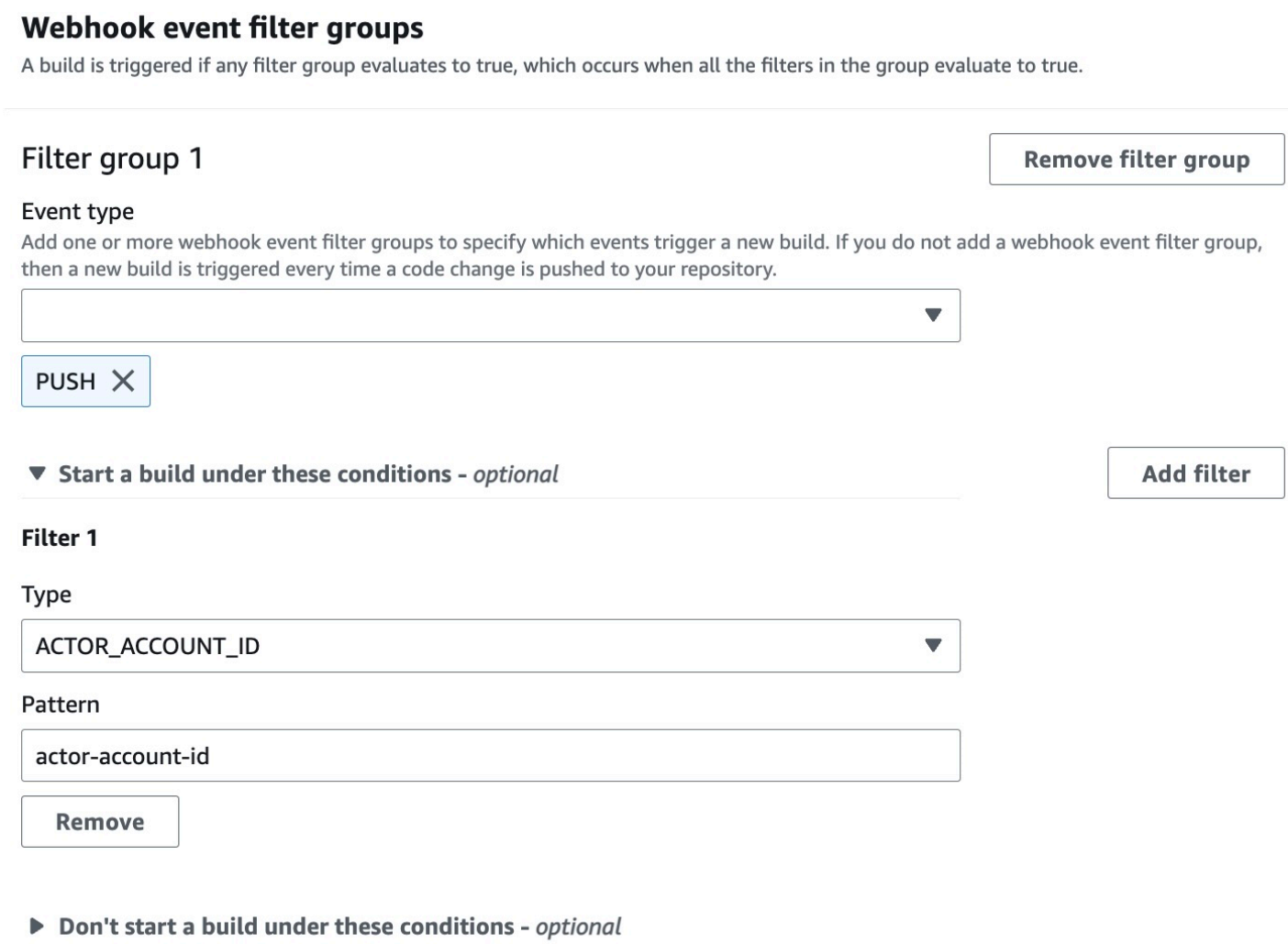


この例で、Webhook フィルターグループは、ファイルが `src` または `test` フォルダーで変更された場合にのみ、ビルドをトリガーします。

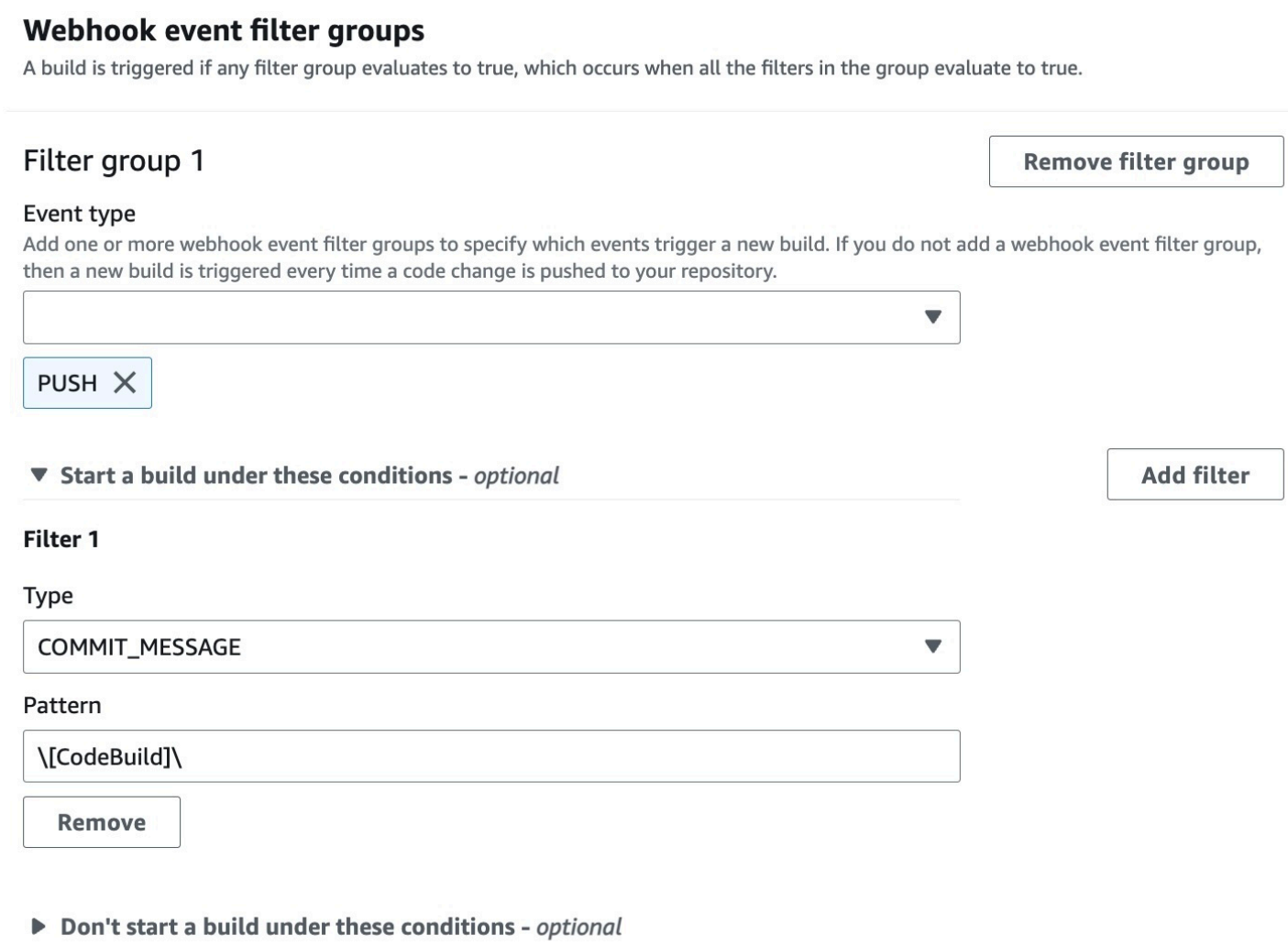


この例では、正規表現 に一致するアカウント ID を持たない GitLab ユーザーによって変更が行われた場合にのみ、ウェブフックフィルターグループがビルドをトリガーしますactor-account-id。

> **Note**
>
> GitLab アカウント ID を検索する方法については、https://api.github.com/users/*user-name* を参照してください。*user-name* は GitLab ユーザー名です。



この例では、HEAD コミットメッセージが正規表現 `\[CodeBuild\]` に一致する場合に、Webhook フィルタグループがプッシュイベントのビルドをトリガーします。



GitLab ウェブフックイベントのフィルタリング (SDK)

AWS CodeBuild SDK を使用してウェブフックイベントをフィルタリングするには、`CreateWebhook`または `UpdateWebhook` API メソッドのリクエスト構文で `filterGroups`フィールドを使用します。詳細については、 CodeBuild API リファレンスのWebhookFilter「」を参照してください。

プルリクエストに対してのみビルドをトリガーするウェブフックフィルタを作成するには、以下をリクエスト構文に挿入します。

```
"filterGroups": [
  [
    {
      "type": "EVENT",
      "pattern": "PULL_REQUEST_CREATED, PULL_REQUEST_UPDATED, PULL_REQUEST_MERGED"
```

```
    }
  ]
]
```

指定されたブランチに対してのみビルドをトリガーするウェブフックフィルタを作成するには、pattern パラメータを使用して、ブランチ名をフィルタリングするよう正規表現を指定します。2 つのフィルタグループの例を使用した場合、ビルドは一方または両方が true と評価されるとトリガーされます。

- 最初のフィルタグループでは、正規表現 ^refs/heads/main$ に一致する Git 参照と ^refs/heads/myBranch$ に一致するヘッド参照を含むブランチで作成または更新されたプルリクエストを指定します。
- 2 番目のフィルタグループでは、正規表現 ^refs/heads/myBranch$ に一致する Git 参照を含むブランチでプッシュリクエストを指定します。

```
"filterGroups": [
  [
    {
      "type": "EVENT",
      "pattern": "PULL_REQUEST_CREATED, PULL_REQUEST_UPDATED"
    },
    {
      "type": "HEAD_REF",
      "pattern": "^refs/heads/myBranch$"
    },
    {
      "type": "BASE_REF",
      "pattern": "^refs/heads/main$"
    }
  ],
  [
    {
      "type": "EVENT",
      "pattern": "PUSH"
    },
    {
      "type": "HEAD_REF",
      "pattern": "^refs/heads/myBranch$"
    }
  ]
```

```
]
```

excludeMatchedPattern パラメータを使用すると、ビルドをトリガーしないイベントを指定することができます。この例では、ビルドは、タグイベントを除くすべてのリクエストに対してトリガーされます。

```
"filterGroups": [
  [
    {
      "type": "EVENT",
      "pattern": "PUSH, PULL_REQUEST_CREATED, PULL_REQUEST_UPDATED,
 PULL_REQUEST_MERGED"
    },
    {
      "type": "HEAD_REF",
      "pattern": "^refs/tags/.*",
      "excludeMatchedPattern": true
    }
  ]
]
```

アカウント ID を持つ GitLab ユーザーによって変更が行われた場合にのみビルドをトリガーするフィルターを作成できますactor-account-id。

> **Note**
>
> GitLab アカウント ID を検索する方法については、https://api.github.com/users/*user-name* を参照してください。*user-name* は GitLab ユーザー名です。

```
"filterGroups": [
  [
    {
      "type": "EVENT",
      "pattern": "PUSH, PULL_REQUEST_CREATED, PULL_REQUEST_UPDATED,
 PULL_REQUEST_MERGED"
    },
    {
      "type": "ACTOR_ACCOUNT_ID",
      "pattern": "actor-account-id"
    }
```

```
  ]
]
```

引数 pattern の正規表現に一致する名前のファイルが変更される場合にのみビルドをトリガーするフィルタを作成することができます。この例のフィルタグループでは、正規表現 ^buildspec.* に一致する名前のファイルが変更された場合にのみビルドをトリガーするよう指定します。

```
"filterGroups": [
  [
    {
      "type": "EVENT",
      "pattern": "PUSH"
    },
    {
      "type": "FILE_PATH",
      "pattern": "^buildspec.*"
    }
  ]
]
```

この例で、フィルターグループは、ファイルが src または test フォルダーで変更された場合にのみ、ビルドをトリガーするように指定しています。

```
"filterGroups": [
    [
        {
            "type": "EVENT",
            "pattern": "PUSH"
        },
        {
            "type": "FILE_PATH",
            "pattern": "^src/.+|^test/.+"
        }
    ]
]
```

HEAD コミットメッセージがパターン引数の正規表現に一致する場合にのみビルドをトリガーするフィルタを作成できます。この例のフィルタグループでは、プッシュイベントの HEAD コミットメッセージが正規表現 \[CodeBuild\] に一致する場合にのみビルドをトリガーするよう指定します。

```
  "filterGroups": [
    [
      {
        "type": "EVENT",
        "pattern": "PUSH"
      },
      {
        "type": "COMMIT_MESSAGE",
        "pattern": "\[CodeBuild\]"
      }
    ]
  ]
```

GitLab ウェブフックイベントのフィルタリング (AWS CloudFormation)

AWS CloudFormation テンプレートを使用してウェブフックイベントをフィルタリングするには、AWS CodeBuild プロジェクトの `FilterGroups`プロパティを使用します。以下の YAML 形式の AWS CloudFormation テンプレート部分によって、2 つのフィルタグループが作成されます。また、一方または両方が true と評価されると、ビルドがトリガーされます。

- 最初のフィルターグループは、アカウント ID を持たない`^refs/heads/main$` GitLab ユーザーが、正規表現 と一致する Git 参照名を持つブランチでプルリクエストを作成または更新することを指定します12345。
- 2 番目のフィルタグループでは、正規表現 `^refs/heads/.*` と一致する Git 参照名を持つブランチに対するプッシュリクエストを作成することを指定します。
- 3 番目のフィルタグループでは、正規表現 `\[CodeBuild\]` に一致する HEAD コミットメッセージを使用してプッシュリクエストを指定します。

```
CodeBuildProject:
  Type: AWS::CodeBuild::Project
  Properties:
    Name: MyProject
    ServiceRole: service-role
    Artifacts:
      Type: NO_ARTIFACTS
    Environment:
      Type: LINUX_CONTAINER
      ComputeType: BUILD_GENERAL1_SMALL
      Image: aws/codebuild/standard:5.0
    Source:
```

```
      Type: GITLAB
      Location: source-location
    Triggers:
      Webhook: true
      FilterGroups:
        - - Type: EVENT
            Pattern: PULL_REQUEST_CREATED,PULL_REQUEST_UPDATED
          - Type: BASE_REF
            Pattern: ^refs/heads/main$
            ExcludeMatchedPattern: false
          - Type: ACTOR_ACCOUNT_ID
            Pattern: 12345
            ExcludeMatchedPattern: true
        - - Type: EVENT
            Pattern: PUSH
          - Type: HEAD_REF
            Pattern: ^refs/heads/.*
        - - Type: EVENT
            Pattern: PUSH
          - Type: COMMIT_MESSAGE
            Pattern: \[CodeBuild\]
```

AWS CodeBuild でのビルドプロジェクトの設定の変更

ビルドプロジェクトの設定を変更するには、AWS CodeBuild コンソール、AWS CLI、または AWS SDK を使用します。

ビルドプロジェクトにテストレポートを追加する場合は、テストレポートのアクセス許可の使用 で記載されている権限が IAM ロールに付与されていることを確認してください。

トピック

- ビルドプロジェクトの設定の変更 (コンソール)
- ビルドプロジェクトの設定の変更 (AWS CLI)
- ビルドプロジェクトの設定の変更 (AWS SDK)

ビルドプロジェクトの設定の変更 (コンソール)

ビルドプロジェクトの設定を変更するには、次の手順を実行します。

1. https://console.aws.amazon.com/codesuite/codebuild/home で AWS CodeBuild コンソールを開きます。

2. ナビゲーションペインで、[Build projects] を選択します。
3. 次のいずれかを行ってください。
   - 変更するビルドプロジェクトのリンクを選択し、[ビルドの詳細] を選択します。
   - 変更するビルドプロジェクトの横にあるラジオボタンを選択して、[View details (詳細を表示)] を選択後 [ビルドの詳細] を選択します。

次のセクションを変更できます。

セクション

- プロジェクトの設定
- ソース
- 環境
- Buildspec
- Batch 構成
- アーティファクト
- ログ

プロジェクトの設定

[プロジェクトの設定] セクションで、[編集] を選択します。変更が完了したら、[設定の更新] を選択して新しい設定を保存します。

次のプロパティを変更できます。

説明

また、他のユーザーがこのプロジェクトの使用目的を理解できるように、ビルドプロジェクトの説明を任意で指定することもできます。

ビルドバッジ

[Enable build badge (ビルドバッジを有効にする)] を選択すると、プロジェクトのビルドステータスが表示可能および埋め込み可能になります。詳細については、「ビルドバッジサンプル」を参照してください。

> **Note**
>
> ソースプロバイダーが Amazon S3 の場合、ビルドバッジは適用されません。

同時ビルド制限を有効にする

このプロジェクトで同時ビルド数を制限するには、次の手順を実行します。

1. [Restrict number of concurrent builds this project can start] (このジョブで許可される同時実行の最大数を設定) を選択します。
2. [Concurrent build limit] (同時ビルド制限) で、このジョブで許可される同時実行の最大数を設定します。この制限は、アカウントに設定された同時ビルド制限より大きくすることはできません。アカウント制限を超える数値を入力しようとすると、エラーメッセージが表示されます。

新しいビルドは、現在のビルド数がこの制限以下の場合にのみ開始されます。現在のビルドカウントがこの制限を満たす場合、新しいビルドはスロットルされ、実行されません。

パブリックビルドアクセスを有効にする

AWS アカウントにアクセスできないユーザーを含め、プロジェクトのビルド結果を一般に公開するには、「パブリックビルドアクセスを有効にする」を選択し、ビルド結果を公開することを確認します。パブリックビルドプロジェクトでは、次のプロパティが使用されます。

パブリックビルドのサービスロール

で新しいサービスロール CodeBuild を作成する場合は、新しいサービスロールを選択します。既存のサービスロールを使用する場合は、既存のサービスロールを選択します。

パブリックビルドサービスロールを使用すると CodeBuild 、 は CloudWatch ログを読み取り、プロジェクトのビルドの Amazon S3 アーティファクトをダウンロードすることができます。これは、プロジェクトのビルドログとアーティファクトを一般に公開するために必要です。

サービスロール

新しいサービスロールまたは既存のサービスロールの名前を入力します。

プロジェクトのビルド結果をプライベートにするには、[Enable public build access] (パブリックビルドアクセスを有効にする) のチェックを外します。

詳細については、「AWS CodeBuild でのパブリックビルドプロジェクト」を参照してください。

> **Warning**
>
> プロジェクトのビルド結果を一般に公開する際には、以下に留意してください。
>
> - プロジェクトがプライベートだったときに実行されたビルドも含めて、プロジェクトのビルド結果、ログ、アーティファクトはすべて、一般に公開されます。
> - すべてのビルドログとアーティファクトが一般に公開されます。環境変数、ソースコード、およびその他の機密情報がビルドログとアーティファクトに出力されている可能性があります。ビルドログに出力される情報には注意が必要です。以下にベストプラクティスを示します。
>   - 機密値、特に AWS アクセスキー IDsとシークレットアクセスキーを環境変数に保存しないでください。Amazon EC2 Systems Manager パラメータストアまたは を使用して AWS Secrets Manager 、機密値を保存することをお勧めします。
>   - ウェブフック使用のベストプラクティス。 に従って、ビルドをトリガーできるエンティティを制限し、buildspec をプロジェクト自体に保存しないことで、Webhook を可能な限り安全に保つことができます。
> - 悪意のあるユーザーがパブリックビルドを利用して、悪意のあるアーティファクトを配信する可能性があります。プロジェクト管理者は、すべてのプルリクエストを確認し、プルリクエストが正当な変更であるか検証することをお勧めします。また、チェックサムを使ってすべてのアーティファクトを検証し、正しいアーティファクトがダウンロードされているか確認することを推奨します。

追加情報

タグ には、サポート AWS サービスで使用するタグの名前と値を入力します。[Add row] を使用して、タグを追加します。最大 50 個のタグを追加できます。

ソース

[ソース] セクションで [編集] を選択します。変更が完了したら、[設定の更新] を選択して新しい設定を保存します。

次のプロパティを変更できます。

ソースプロバイダー

ソースコードプロバイダーのタイプを選択します。次のリストを使用して、ソースプロバイダーに関する適切な選択を行います。

> **Note**
>
> CodeBuild は Bitbucket サーバーをサポートしていません。

Amazon S3

バケット

ソースコードが格納されている入力バケットの名前を選択します。

S3 オブジェクトキーまたは S3 フォルダ

ZIP ファイルの名前、またはソースコードを含むフォルダへのパスを入力します。S3 バケットの中身をすべてダウンロードするには、スラッシュ記号 (/) を入力します。

ソースバージョン

入力ファイルのビルドを表すオブジェクトのバージョン IDを入力。詳細については、「」を参照してくださいのソースバージョンサンプル AWS CodeBuild

CodeCommit

リポジトリ

使用するリポジトリを選択します。

参照タイプ

[Branch] (ブランチ) または [Git tag] (Git タグ) を選択するか、[Commit ID] (コミット ID) を入力して、ソースコードのバージョンを指定します。詳細については、「のソースバージョンサンプル AWS CodeBuild」を参照してください。

> **Note**
>
> 811dd1ba1aba14473856cee38308caed7190c0d または 5392f7 のように、コミット ID と似ていない Git ブランチ名を選択することをお勧めします。これにより、Git checkout が実際のコミットと衝突するのを防ぐことができます。

Git クローンの深度

選択して、指定されるコミット数で切り捨てられる履歴の浅いクローンを作成します。完全クローンを希望する場合には、[Full (完全)] を選択します。

Git サブモジュール

リポジトリに Git サブモジュールを含める場合は、[Git サブモジュールを使用する] を選択します。

Bitbucket

リポジトリ

[Connect using OAuth] (OAuth を使用して接続する) または [Connect with a Bitbucket app password] (Bitbucket アプリパスワードで接続する) を選択し、手順に従って Bitbucket に接続 (または再接続) します。

パブリックのリポジトリかアカウント内のリポジトリかを選択します。

ソースバージョン

ブランチ、コミット ID、タグあるいはリファレンスとコミット ID を入力します。詳細については、「のソースバージョンサンプル AWS CodeBuild」を参照してください。

> **Note**
>
> 811dd1ba1aba14473856cee38308caed7190c0d または 5392f7 のように、コミット ID と似ていない Git ブランチ名を選択することをお勧めします。これにより、Git checkout が実際のコミットと衝突するのを防ぐことができます。

Git クローンの深度

[Git のクローンの深さ] を選択して、指定されるコミット数で切り捨てられる履歴の浅いクローンを作成します。完全クローンを希望する場合には、[Full (完全)] を選択します。

Git サブモジュール

リポジトリに Git サブモジュールを含める場合は、[Git サブモジュールを使用する] を選択します。

ビルドステータス

ビルドの開始と終了のステータスをソースプロバイダーにレポートする場合は、[Report build statuses to source provider when your builds start and finish] (ビルドの開始と終了時にソースプロバイダーにビルドステータスをレポートする) を選択します。

ソースプロバイダにビルド状態を報告できるようにするには、ソースプロバイダに関連付けられたユーザーがリポジトリへの書き込みアクセス権を持っている必要があります。ユーザーが書き込みアクセス権を持っていない場合、ビルドのステータスは更新できません。詳細については、「ソースプロバイダーのアクセス」を参照してください。

[Status context] (ステータスコンテキスト) に、Bitbucket コミットステータスの `name` パラメータに使用する値を記入します。詳細については、Bitbucket API ドキュメントの「ビルド」を参照してください。

[Target URL] (ターゲットURL) に、Bitbucket コミットステータスの `url` パラメータに使用する値を記入します。詳細については、Bitbucket API ドキュメントの「ビルド」を参照してください。

webhook によってトリガーされたビルドのステータスは常にソースプロバイダーにレポートされます。コンソールから開始されたビルドのステータスまたはソースプロバイダーに報告された API 呼び出しを取得するには、この設定を選択する必要があります。

プロジェクトのビルドが webhook によってトリガーされた場合、この設定への変更を有効にするには、新しいコミットをリポジトリにプッシュする必要があります。

プライマリソースウェブフックイベント で、コード変更がこのリポジトリにプッシュされるたびにソースコードを構築する場合は、コード変更がこのリポジトリにプッシュされるたびに再構築を選択します。 CodeBuild Webhook およびフィルターグループの詳細については、「Bitbucket ウェブフックイベント」を参照してください。

GitHub

リポジトリ

OAuth を使用して接続 または GitHub 個人アクセストークンを使用して接続 を選択し、手順に従って に接続 (または再接続) GitHub し、 へのアクセスを許可します AWS CodeBuild。

パブリックのリポジトリかアカウント内のリポジトリかを選択します。

ソースバージョン

ブランチ、コミット ID、タグあるいはリファレンスとコミット ID を入力します。詳細については、「のソースバージョンサンプル AWS CodeBuild」を参照してください。

> **Note**
>
> `811dd1ba1aba14473856cee38308caed7190c0d` または `5392f7` のように、コミット ID と似ていない Git ブランチ名を選択することをお勧めします。これにより、Git checkout が実際のコミットと衝突するのを防ぐことができます。

Git クローンの深度

[Git のクローンの深さ] を選択して、指定されるコミット数で切り捨てられる履歴の浅いクローンを作成します。完全クローンを希望する場合には、[Full (完全)] を選択します。

Git サブモジュール

リポジトリに Git サブモジュールを含める場合は、[Git サブモジュールを使用する] を選択します。

ビルドステータス

ビルドの開始と終了のステータスをソースプロバイダーにレポートする場合は、[Report build statuses to source provider when your builds start and finish] (ビルドの開始と終了時にソースプロバイダーにビルドステータスをレポートする) を選択します。

ソースプロバイダにビルド状態を報告できるようにするには、ソースプロバイダに関連付けられたユーザーがリポジトリへの書き込みアクセス権を持っている必要があります。ユーザーが書き込みアクセス権を持っていない場合、ビルドのステータスは更新できません。詳細については、「ソースプロバイダーのアクセス」を参照してください。

ステータスコンテキスト には、 GitHub コミットステータスの `context`パラメータに使用する値を入力します。詳細については、「 デベロッパーガイド」の「コミットステータスの作成」を参照してください。 GitHub

ターゲット URL には、 GitHub コミットステータスの `target_url`パラメータに使用する値を入力します。詳細については、「 デベロッパーガイド」の「コミットステータスの作成」を参照してください。 GitHub

webhook によってトリガーされたビルドのステータスは常にソースプロバイダーにレポートされます。コンソールから開始されたビルドのステータスまたはソースプロバイダーに報告された API 呼び出しを取得するには、この設定を選択する必要があります。

プロジェクトのビルドが webhook によってトリガーされた場合、この設定への変更を有効にするには、新しいコミットをリポジトリにプッシュする必要があります。

プライマリソースウェブフックイベント で、コード変更がこのリポジトリにプッシュされるたびにソースコードを構築する場合は、コード変更がこのリポジトリにプッシュされるたびに再構築を選択します。 CodeBuild Webhook およびフィルターグループの詳細については、「GitHub ウェブフックイベント」を参照してください。

GitHub Enterprise Server

GitHub エンタープライズ個人用アクセストークン

個人用アクセストークンのクリップボードにコピーする方法に関しては「GitHub Enterprise Server のサンプル」参照してください。テキストフィールドにトークンを貼り付け、[トークンの保存] を選択します。

> **Note**
>
> 個人用アクセストークンは、一回のみ入力して保存することが必要となります。 CodeBuild は、今後のすべてのプロジェクトでこのトークンを使用します。

ソースバージョン

プルリクエスト、ブランチ、コミット ID、コミット ID、参照、およびコミット ID を入力します。詳細については、「のソースバージョンサンプル AWS CodeBuild」を参照してください。

> **Note**
>
> 811dd1ba1aba14473856cee38308caed7190c0d または 5392f7 のように、コミット ID と似ていない Git ブランチ名を選択することをお勧めします。これにより、Git checkout が実際のコミットと衝突するのを防ぐことができます。

Git クローンの深度

[Git のクローンの深さ] を選択して、指定されるコミット数で切り捨てられる履歴の浅いクローンを作成します。完全クローンを希望する場合には、[Full (完全)] を選択します。

Git サブモジュール

リポジトリに Git サブモジュールを含める場合は、[Git サブモジュールを使用する] を選択します。

ビルドステータス

ビルドの開始と終了のステータスをソースプロバイダーにレポートする場合は、[Report build statuses to source provider when your builds start and finish] (ビルドの開始と終了時にソースプロバイダーにビルドステータスをレポートする) を選択します。

ソースプロバイダにビルド状態を報告できるようにするには、ソースプロバイダに関連付けられたユーザーがリポジトリへの書き込みアクセス権を持っている必要があります。ユーザーが書き込みアクセス権を持っていない場合、ビルドのステータスは更新できません。詳細については、「ソースプロバイダーのアクセス」を参照してください。

ステータスコンテキスト には、 GitHub コミットステータスの `context`パラメータに使用する値を入力します。詳細については、「 デベロッパーガイド」の「コミットステータスの作成」を参照してください。 GitHub

ターゲット URL には、 GitHub コミットステータスの `target_url`パラメータに使用する値を入力します。詳細については、「 デベロッパーガイド」の「コミットステータスの作成」を参照してください。 GitHub

webhook によってトリガーされたビルドのステータスは常にソースプロバイダーにレポートされます。コンソールから開始されたビルドのステータスまたはソースプロバイダーに報告された API 呼び出しを取得するには、この設定を選択する必要があります。

プロジェクトのビルドが webhook によってトリガーされた場合、この設定への変更を有効にするには、新しいコミットをリポジトリにプッシュする必要があります。

安全でない SSL

Enterprise プロジェクトリポジトリへの接続中に SSL 警告を無視するには、安全でない SSL GitHub を有効にするを選択します。

プライマリソースウェブフックイベント で、コード変更がこのリポジトリにプッシュされるたびにソースコードを構築する場合は、コード変更がこのリポジトリにプッシュされるたびに再構築を選択します。 CodeBuild Webhook およびフィルターグループの詳細については、「GitHub ウェブフックイベント」を参照してください。

GitLab

Connection

を使用して GitLab アカウントを接続し AWS CodeConnections、その接続を使用してサードパーティーのリポジトリをビルドプロジェクトのソースとして関連付けます。

デフォルト接続 またはカスタム接続 を選択します。

デフォルトの接続では、すべてのプロジェクトにデフォルトの GitLab 接続が適用されます。カスタム接続は、アカウントのデフォルト設定を上書きするカスタム GitLab 接続を適用します。

デフォルト接続

アカウントに関連付けられているデフォルトの接続の名前。

プロバイダーへの接続をまだ作成していない場合は、への接続 GitLabを作成する (コンソール)「」の手順を参照してください。

カスタム接続

使用するカスタム接続の名前を選択します。

プロバイダーへの接続をまだ作成していない場合は、への接続 GitLabを作成する (コンソール)「」の手順を参照してください。

リポジトリ

使用するリポジトリを選択します。

ソースバージョン

プルリクエスト ID、ブランチ、コミット ID、タグ、またはリファレンスとコミット ID を入力します。詳細については、「のソースバージョンサンプル AWS CodeBuild」を参照してください。

> **Note**
>
> 811dd1ba1aba14473856cee38308caed7190c0d または 5392f7 のように、コミット ID と似ていない Git ブランチ名を選択することをお勧めします。これにより、Git checkout が実際のコミットと衝突するのを防ぐことができます。

Git クローンの深度

[Git のクローンの深さ] を選択して、指定されるコミット数で切り捨てられる履歴の浅いクローンを作成します。完全クローンを希望する場合には、[Full (完全)] を選択します。

ビルドステータス

ビルドの開始と終了のステータスをソースプロバイダーにレポートする場合は、[Report build statuses to source provider when your builds start and finish] (ビルドの開始と終了時にソースプロバイダーにビルドステータスをレポートする) を選択します。

ソースプロバイダにビルド状態を報告できるようにするには、ソースプロバイダに関連付けられたユーザーがリポジトリへの書き込みアクセス権を持っている必要があります。ユーザーが書き込みアクセス権を持っていない場合、ビルドのステータスは更新できません。詳細については、「ソースプロバイダーのアクセス」を参照してください。

GitLab Self Managed

Connection

を使用して GitLab アカウントを接続し AWS CodeConnections、その接続を使用してサードパーティーのリポジトリをビルドプロジェクトのソースとして関連付けます。

デフォルト接続 またはカスタム接続 を選択します。

デフォルトの接続では、すべてのプロジェクトにデフォルトの GitLab セルフマネージド接続が適用されます。カスタム接続は、アカウントのデフォルト設定を上書きするカスタム GitLab セルフマネージド接続を適用します。

デフォルト接続

アカウントに関連付けられているデフォルトの接続の名前。

プロバイダーへの接続をまだ作成していない場合は、「 デベロッパーツールコンソールユーザーガイド」の GitLab 「自己管理型 への接続を作成する」を参照してください。

カスタム接続

使用するカスタム接続の名前を選択します。

プロバイダーへの接続をまだ作成していない場合は、「 デベロッパーツールコンソールユーザーガイド」の GitLab 「自己管理型 への接続を作成する」を参照してください。

リポジトリ

使用するリポジトリを選択します。

ソースバージョン

プルリクエスト ID、ブランチ、コミット ID、タグ、またはリファレンスとコミット ID を入力します。詳細については、「のソースバージョンサンプル AWS CodeBuild」を参照してください。

> **Note**
>
> 811dd1ba1aba14473856cee38308caed7190c0d または 5392f7 のように、コミット ID と似ていない Git ブランチ名を選択することをお勧めします。これにより、Git checkout が実際のコミットと衝突するのを防ぐことができます。

Git クローンの深度

[Git のクローンの深さ] を選択して、指定されるコミット数で切り捨てられる履歴の浅いクローンを作成します。完全クローンを希望する場合には、[Full (完全)] を選択します。

ビルドステータス

ビルドの開始と終了のステータスをソースプロバイダーにレポートする場合は、[Report build statuses to source provider when your builds start and finish] (ビルドの開始と終了時にソースプロバイダーにビルドステータスをレポートする) を選択します。

ソースプロバイダにビルド状態を報告できるようにするには、ソースプロバイダに関連付けられたユーザーがリポジトリへの書き込みアクセス権を持っている必要があります。ユーザーが

書き込みアクセス権を持っていない場合、ビルドのステータスは更新できません。詳細については、「ソースプロバイダーのアクセス」を参照してください。

環境

[環境] セクションで、[編集] を選択します。変更が完了したら、[設定の更新] を選択して新しい設定を保存します。

次のプロパティを変更できます。

プロビジョニングモデル

プロビジョニングモデルを変更するには、プロビジョニングモデルの変更を選択し、次のいずれかを実行します。

- によって管理されるオンデマンドフリートを使用するには AWS CodeBuild、オンデマンド を選択します。オンデマンドフリートでは、 CodeBuild はビルドのコンピューティングを提供します。マシンはビルドが終了すると破棄されます。オンデマンドフリートはフルマネージド型で、需要の急増にも対応できる自動スケーリング機能を備えています。
- によって管理されるリザーブドキャパシティフリートを使用するには AWS CodeBuild、リザーブドキャパシティ を選択し、フリート名 を選択します。リザーブドキャパシティフリートでは、ビルド環境に合わせて専有インスタンスのセットを設定します。これらのマシンはアイドル状態のままで、ビルドやテストをすぐに処理できる状態になり、ビルド時間を短縮します。リザーブドキャパシティフリートでは、マシンは常に稼働しており、プロビジョニングされている間はコストが発生し続けます。

詳細については、「でのリザーブドキャパシティの操作 AWS CodeBuild」を参照してください。

環境イメージ

ビルドイメージを変更するには、[イメージの上書き] を選択し、次のいずれかを実行します。

- によって管理される Docker イメージを使用するには AWS CodeBuild、マネージドイメージ を選択し、オペレーティングシステム 、ランタイム (s)、イメージ 、イメージバージョン から選択します。利用可能な場合は、[環境タイプ] から選択します。
- 別の Docker イメージを使用するには、[カスタムイメージ] を選択します。[Environment type (環境タイプ)] で、 [ARM]、[Linux]、[Linux GPU] または [Windows] を選択します。[Other registry (その他のレジストリ)] を選択した場合は、[External registry URL (外部のレジストリ URL)] に *docker repository*/*docker image name* の形式に従って Docker Hub の Docker イメージの名前とタグを入力します。Amazon ECR を選択した場合は、Amazon ECR

リポジトリと Amazon ECR イメージを使用して、 AWS アカウントの Docker イメージを選択します。

- プライベート Docker イメージを使用するには、[カスタムイメージ] を選択します。[Environment type (環境タイプ)] で、 [ARM]、[Linux]、[Linux GPU] または [Windows] を選択します。[Image registry (イメージレジストリ)] に [Other registry (その他のレジストリ)] を選択して、その後プライベート Docker イメージの認証情報の ARN を入力します。認証情報は、Secrets Manager で作成する必要があります。詳細については、AWS Secrets Manager ユーザーガイドの「AWS Secrets Managerとは」を参照してください。

> **Note**
>
> CodeBuild は、カスタム Docker イメージ`ENTRYPOINT`の を上書きします。

サービスロール

次のいずれかを行います。

- CodeBuild サービスロールがない場合は、新しいサービスロール を選択します。[Role name] に、新しいロールの名前を入力します。
- CodeBuild サービスロールがある場合は、既存のサービスロール を選択します。[Role ARN] で、サービスロールを選択します。

> **Note**
>
> コンソールを使用してビルドプロジェクトを作成すると、同時に CodeBuild サービスロールを作成できます。デフォルトでは、ロールはそのビルドプロジェクトでのみ使用できます。コンソールでは、このサービスロールを別のビルドプロジェクトと関連付けると、この別のビルドプロジェクトで使用できるようにロールが更新されます。サービスロールは最大 10 個のビルドプロジェクトで使用できます。

追加設定

タイムアウト

5 分から 8 時間の間の値を指定し、その後、完了しないとビルドを CodeBuild 停止します。[hours] と [minutes] を空白のままにすると、デフォルト値の 60 分が使用されます。

特権付与

Docker イメージを構築する場合、またはビルドに昇格された権限を取得させる場合は、このフラグを有効にするを選択します。このビルドプロジェクトを使用して Docker イメージを構築する予定の場合のみ。それ以外の場合、関連付けられているビルドで Docker デーモンと通信しようとすると、すべて失敗します。ビルドが Docker デーモンと連係動作できるように、Docker デーモンも起動する必要があります。これを行う 1 つの方法は、次のビルドコマンドを実行してビルドスペックの `install` フェーズで Docker デーモンを初期化することです。Docker サポート CodeBuild で が提供するビルド環境イメージを選択した場合は、これらのコマンドを実行しないでください。

> **Note**
>
> デフォルトでは、Docker デーモンは VPC 以外のビルドで有効になっています。VPC ビルドに Docker コンテナを使用する場合は、Docker Docs ウェブサイトの「ランタイム特権と Linux 機能」を参照して、特権モードを有効にします。また、Windows は特権モードをサポートしていません。

```
- nohup /usr/local/bin/dockerd --host=unix:///var/run/docker.sock --
host=tcp://127.0.0.1:2375 --storage-driver=overlay2 &
- timeout 15 sh -c "until docker info; do echo .; sleep 1; done"
```

VPC

VPC CodeBuild を使用する場合：

- VPC の場合、 が CodeBuild 使用する VPC ID を選択します。
- VPC サブネット で、 が CodeBuild 使用するリソースを含むサブネットを選択します。
- VPC セキュリティグループ で、 CodeBuild が VPCs内のリソースへのアクセスを許可するために使用するセキュリティグループを選択します。

詳細については、「Amazon Virtual Private Cloud AWS CodeBuild で を使用する」を参照してください。

コンピューティング

使用可能なオプションの 1 つを選択します。

環境変数

[環境変数] で、名前と値を入力してから、ビルドによって使用される各環境変数の種類を選択します。

> **Note**
>
> CodeBuild は、 AWS リージョンの環境変数を自動的に設定します。以下の環境変数を buildspec.yml に追加していない場合は、それらの変数を設定する必要があります。
>
> - AWS_ACCOUNT_ID
> - IMAGE_REPO_NAME
> - IMAGE_TAG

コンソールと AWS CLI ユーザーは環境変数を表示できます。環境変数の表示に懸念がない場合は、[Name] および [Value] フィールドを設定し、[Type] を [Plaintext] に設定します。

AWS アクセスキー ID、 AWS シークレットアクセスキー、パスワードなどの機密値を持つ環境変数をパラメータとして Amazon EC2 Systems Manager パラメータストアまたは に保存することをお勧めします AWS Secrets Manager。

Amazon EC2 Systems Manager パラメータストアを使用する場合は、[Type (タイプ)] で、[Parameter (パラメータ)] を選択します。名前に、参照 CodeBuild する の識別子を入力します。[Value] (値) に、Amazon EC2 Systems Manager パラメータストアに保存されているパラメータの名前を入力します。たとえば、`/CodeBuild/dockerLoginPassword` という名前のパラメータを使用して、[タイプ] で [Parameter (パラメータ)] を選択します。[Name (名前)] に `LOGIN_PASSWORD` と入力します。[Value (値)] に「`/CodeBuild/dockerLoginPassword`」と入力します。

> **Important**
>
> Amazon EC2 Systems Manager パラメータストアを使用する場合、パラメータは `/CodeBuild/` で始まるパラメータ名 (例: `/CodeBuild/dockerLoginPassword`) で保存することをお勧めします。 CodeBuild コンソールを使用して、Amazon EC2 Systems Manager でパラメータを作成できます。[パラメータの作成] を選択し、ダイアログボックスの手順に従います。(このダイアログボックスの KMS キー では、アカウントの AWS KMS キーの ARN を指定できます。 Amazon EC2 Systems Manager

は、このキーを使用して、ストレージ中にパラメータの値を暗号化し、取得中に復号します）。 CodeBuild コンソールを使用してパラメータを作成する場合、コンソールは保存されているパラメータ名を /CodeBuild/ で開始します。詳細については、Amazon EC2 Systems Manager ユーザーガイドの「Systems Manager パラメータストア」および「Systems Manager パラメータストアコンソールのチュートリアル」を参照してください。
ビルドプロジェクトが Amazon EC2 Systems Manager パラメータストアに保存されているパラメータを参照する場合、ビルドプロジェクトのサービスロールで ssm:GetParameters アクションを許可する必要があります。以前に新しいサービスロールを選択した場合、 はビルドプロジェクトのデフォルトのサービスロールにこのアクション CodeBuild を含めます。ただし [既存のサービスロール] を選択した場合は、このアクションをサービスロールに個別に含める必要があります。
ビルドプロジェクトが、/CodeBuild/ で始まらないパラメータ名を持つ、Amazon EC2 Systems Manager パラメータストアに保存されているパラメータを参照し、[新しいサービスロール] を選択した場合、/CodeBuild/ で始まらないパラメータ名にアクセスできるようにサービスロールを更新する必要があります。これは、サービスロールで、/CodeBuild/ で始まるパラメータ名にのみアクセスが許可されるためです。
[新しいサービスロールを作成] を選択した場合、サービスロールには、Amazon EC2 Systems Manager パラメータストアの /CodeBuild/ 名前空間ですべてのパラメータを復号するアクセス権限が含まれます。
既存の環境変数は、設定した環境変数により置き換えられます。たとえば、Docker イメージに my_value の値を持つ MY_VAR という名前の環境変数が既に含まれていて、other_value の値を持つ MY_VAR という名前の環境変数を設定した場合、my_value が other_value に置き換えられます。同様に、Docker イメージに /usr/local/sbin:/usr/local/bin の値を持つ PATH という名前の環境変数が既に含まれていて、$PATH:/usr/share/ant/bin の値を持つ PATH という名前の環境変数を設定した場合、/usr/local/sbin:/usr/local/bin はリテラル値 $PATH:/usr/share/ant/bin に置き換えられます。
CODEBUILD_ で始まる名前の環境変数は設定しないでください。このプレフィックスは内部使用のために予約されています。
同じ名前の環境変数が複数の場所で定義されている場合は、その値は次のように決定されます。

- ビルド開始オペレーション呼び出しの値が最も優先順位が高くなります。
- ビルドプロジェクト定義の値が次に優先されます。

- ビルド仕様宣言の値の優先順位が最も低くなります。

Secrets Manager を使用する場合は、[Type] (タイプ) で、[Secrets Manager] を選択します。名前に、参照 CodeBuild する の識別子を入力します。[Value (値)] に、パターン `reference-key` を使用して *`secret-id`*`:`*`json-key`*`:`*`version-stage`*`:`*`version-id`* を入力します。詳細については、「Secrets Manager reference-key in the buildspec file」を参照してください。

> **Important**
>
> Secrets Manager を使用する場合は、「`/CodeBuild/`」で始まる名前でシークレットを保存することをお勧めします(たとえば、`/CodeBuild/dockerLoginPassword`)。詳細については、 AWS Secrets Managerユーザーガイドの「AWS Secrets Manager とは」を参照してください。
>
> ビルドプロジェクトが Secrets Manager パラメータストアに保存されているパラメータを参照する場合、ビルドプロジェクトのサービスロールで `secretsmanager:GetSecretValue` アクションを許可する必要があります。以前に新しいサービスロールを選択した場合、 はビルドプロジェクトのデフォルトのサービスロールにこのアクション CodeBuild を含めます。ただし [既存のサービスロール] を選択した場合は、このアクションをサービスロールに個別に含める必要があります。
>
> ビルドプロジェクトが、`/CodeBuild/` で始まらないパラメータ名を持つ、Secrets Manager に保存されているパラメータを参照し、[新しいサービスロール] を選択した場合、`/CodeBuild/` で始まらないシークレット名にアクセスできるようにサービスロールを更新する必要があります。これは、サービスロールで、`/CodeBuild/` で始まるシークレット名にのみアクセスが許可されるためです。
>
> [新しいサービスロール] を選択した場合、作成されるサービスロールには、Secrets Manager の `/CodeBuild/` 名前空間ですべてのシークレットを復号するアクセス許可が含まれます。

Buildspec

[Buildspec] セクションで、[編集] を選択します。変更が完了したら、[設定の更新] を選択して新しい設定を保存します。

次のプロパティを変更できます。

ビルド仕様

次のいずれかを行ってください。

- ソースコードにビルド仕様ファイルが含まれている場合は、[Use a buildspec file (buildspec ファイルを使用)] を選択します。デフォルトでは、 CodeBuild はソースコードのルートディレクトリbuildspec.ymlで という名前のファイルを検索します。buildspec ファイルに別の名前または場所が使用されている場合は、Buildspec 名 にソースルートからのパスを入力します (例えば、buildspec-two.yml または configuration/buildspec.yml)。buildspec ファイルが S3 バケットにある場合は、ビルドプロジェクトと同じ AWS リージョンに存在する必要があります。ARN を使用して buildspec ファイルを指定します (例: arn:aws:s3:::*<my-codebuild-sample2>*/buildspec.yml)。
- ソースコードにビルド仕様ファイルが含まれていない場合、または、ソースコードのルートディレクトリで build ファイルの buildspec.yml フェーズに指定されているものと異なるビルドコマンドを実行する場合は、[ビルドコマンドの挿入] を選択します。[ビルドコマンド] に、build フェーズで実行するコマンドを入力します。複数のコマンドについては、&& で各コマンドを区切ります (例: mvn test && mvn package)。他のフェーズでコマンドを実行する場合、または build フェーズのコマンドの長いリストがある場合は、ソースコマンドのルートディレクトリに buildspec.yml ファイルを追加し、ファイルにコマンドを追加してから、[Use the buildspec.yml in the source code root directory] (ソースコードのルートディレクトリの 「buildspec.yml」を使用) を選択します。

詳細については、「ビルド仕様 (buildspec) に関するリファレンス」を参照してください。

Batch 構成

[バッチ構成] セクションで、[編集] を選択します。変更が完了したら、[設定の更新] を選択して新しい設定を保存します。詳細については、「でのバッチビルド AWS CodeBuild」を参照してください。

次のプロパティを変更できます。

Batch サービスロール

バッチビルドのサービスロールを提供します。

次のいずれかを選択します。

- バッチサービスロールがない場合は、[New service role] (新しいサービスロール) を選択します。[Service role] (サービスロール) に、新しいロールの名前を入力します。

- バッチサービスロールがある場合は、[Existing service role] (既存のサービスロール) を選択します。[Service role] (サービスロール) で、サービスロールを選択します。

バッチビルドでは、バッチ設定に新しいセキュリティロールが導入されます。この新しいロールは、バッチの一部としてビルドを実行するために、 がユーザーに代わって StartBuild、StopBuild、および RetryBuildアクションを呼び出すことができるように CodeBuild する必要があるため必要です。次の2つの理由により、お客様はビルドで使用するものと同じロールではなく、新しいロールを使用する必要があります。

- ビルドの役割を与える StartBuild、StopBuild、および RetryBuild アクセス権限を使用すると、単一のビルドが buildspec を介してより多くのビルドを開始することができます。
- CodeBuild バッチビルドには、バッチ内のビルドに使用できるビルドとコンピューティングタイプの数を制限する制限があります。ビルドロールにこれらの権限がある場合、ビルド自体がこれらの制限を回避する可能性があります。

バッチに使用できる計算タイプ

バッチに使用できる計算タイプを選択します。該当するものをすべて選択します。

バッチで許可される最大ビルド

バッチで許可されるビルドの最大数を入力します。バッチがこの制限を超えると、バッチは失敗します。

バッチのタイムアウト

バッチビルドが完了する最大時間を入力します。

アーティファクトの結合

[Combine all artifacts from batch into a single location] (バッチのすべてのアーチファクト) を 1 つの場所に結合するを選択して、バッチのすべてのアーチファクトを単一の場所に結合します。

バッチレポートモード

バッチビルドに対して望ましいビルドステータスレポートモードを選択します。

> **Note**
>
> このフィールドは、プロジェクトソースが Bitbucket GitHub、、または GitHub Enterprise の場合にのみ使用でき、ビルドの開始と終了がソース で選択されたときにビルドステータスをソースプロバイダーに報告します。

集約されたビルド

これを選択して、バッチ内にあるすべてのビルドのステータスを単一のステータスレポートにまとめます。

個々のビルド

これを選択して、バッチ内にあるすべてのビルドのビルドステータスが個別に報告されるようにします。

アーティファクト

[アーティファクト] セクションで、[編集] を選択します。変更が完了したら、[設定の更新] を選択して新しい設定を保存します。

次のプロパティを変更できます。

タイプ

次のいずれかを行ってください。

- ビルド出力アーティファクトを作成しない場合は、[No artifacts] を選択します。ビルドテストのみを実行している場合や、Docker イメージを Amazon ECR リポジトリにプッシュする場合には、これを行うことができます。
- ビルド出力を S3 バケットに保存する場合は、[Amazon S3] を選択して次のいずれかの操作を行います。
  - ビルド出力 ZIP ファイルまたはフォルダにプロジェクト名を使用する場合は、[Name (名前)] を空白のままにします。それ以外の場合は、名前を入力します。(ZIP ファイルを出力して ZIP ファイルにファイル拡張子を付ける場合は、必ず ZIP ファイル名の後に含めます)。
  - buildspec ファイルで指定した名前で、コンソールで指定した名前を上書きする場合は、[Enable semantic versioning (セマンティックバージョニングを有効にする)] を選択します。buildspec ファイル内の名前は、ビルド時に計算され、Shell コマンド言語を使用します。たとえば、アーティファクト名に日付と時刻を追加して常に一意にできます。アーティファクト名を一意にすると、アーティファクトが上書きされるのを防ぐことができます。詳細については、「buildspec の構文」を参照してください。
  - [Bucket name (バケット名)] で、出力バケットの名前を選択します。
  - この手順の前の方で [ビルドコマンドの挿入] を選択した場合は、[出力ファイル] に、ビルド出力 ZIP ファイルまたはフォルダに格納するビルドのファイルの場所を入力します。複数の

場所の場合は、各場所をコンマで区切ります (例: `appspec.yml, target/my-app.jar`)。詳細については、「`files`」で buildspec の構文 の説明を参照してください。

- ビルドアーティファクトを暗号化しない場合は、[アーティファクト暗号化の削除] を選択します。

アーティファクトのセカンダリセットごとに:

1. [Artifact 識別子] には、英数字とアンダースコアのみを使用して 128 文字未満の値を入力します。
2. [アーティファクトの追加] を選択します。
3. セカンダリアーティファクトを設定するには、前のステップに従います。
4. [アーティファクトの保存] を選択します。

追加設定

暗号化キー

次のいずれかを行います。

- アカウントで AWS マネージドキー Amazon S3 を使用してビルド出力アーティファクトを暗号化するには、暗号化キーを空白のままにします。これがデフォルトです。
- カスタマー管理のキーを使用してビルド出力アーティファクトを暗号化するには、[暗号化キー] にカスタマー管理のキーの ARN を入力します。`arn:aws:kms:`*`region-ID`*`:`*`account-ID`*`:key/`*`key-ID`* の形式を使用します。

キャッシュタイプ

[キャッシュタイプ] で、以下のいずれかを選択します。

- キャッシュを使用しない場合は、[No cache] を選択します。
- Amazon S3 キャッシュを使用するには、[Amazon S3] を選択して次の操作を行います。
  - [バケット] では、キャッシュが保存される S3 バケットの名前を選択します。
  - (オプション) [Cache path prefix (キャッシュパスのプレフィックス)] に、Amazon S3 パスのプレフィックスを入力します。[キャッシュパスのプレフィックス] 値はディレクトリ名に似ています。これにより、バケット内の同じディレクトリにキャッシュを保存できます。

> **Important**
>
> パスのプレフィックスの末尾にスラッシュ (/) を付加しないでください。

- ローカルキャッシュを使用する場合は、[ローカル] を選択し、ローカルキャッシュモードを 1 つ以上選択します。

  > **Note**
  >
  > Docker レイヤーキャッシュモードは Linux でのみ利用可能です。このモードを選択する場合、プロジェクトは権限モードで実行する必要があります。

  キャッシュを使用すると、再利用可能なビルド環境がキャッシュに保存され、ビルド全体で使用されるため、かなりのビルド時間が節約されます。ビルド仕様ファイルのキャッシュの指定に関する詳細については、「buildspec の構文」を参照してください。キャッシングの詳細については、「AWS CodeBuild でのキャッシュのビルド」を参照してください。

ログ

[ログ] セクションで [編集] を選択します。変更が完了したら、[設定の更新] を選択して新しい設定を保存します。

次のプロパティを変更できます。

作成するログを選択します。Amazon CloudWatch Logs、Amazon S3 ログ、またはその両方を作成できます。

CloudWatch

Amazon CloudWatch Logs ログが必要な場合：

CloudWatch ログ

CloudWatch ログを選択します。

グループ名

Amazon CloudWatch Logs ロググループの名前を入力します。

ストリーム名

Amazon CloudWatch Logs ログストリーム名を入力します。

S3

Amazon S3 ログが必要な場合は、以下のようになります。

S3 ログ

[S3 logs (S3 ログ)] を選択します。

バケット

ログを保存する S3 バケットの名前を選択します。

パスプレフィックス

ログのプレフィックスを入力します。

S3 ログの暗号化を無効にする

S3 ログを暗号化しない場合は、選択します。

ビルドプロジェクトの設定の変更 (AWS CLI)

AWS CLI を AWS CodeBuild と組み合わせて使用する方法については、「コマンドラインリファレンス」を参照してください。

AWS CLI で CodeBuild プロジェクトを更新するには、更新されたプロパティを含む JSON ファイルを作成し、そのファイルを「update-project」コマンドに渡します。更新ファイルに含まれていないプロパティは変更されません。

更新 JSON ファイルでは、name プロパティおよび変更されたプロパティのみが必要です。name プロパティにより、変更するプロジェクトを識別します。変更された構造については、それらの構造に必要なパラメータも含める必要があります。たとえば、プロジェクトの環境を変更するには、「environment/type」および「environment/computeType」プロパティが必要です。環境イメージを更新する例を次に示します。

```
{
  "name": "<project-name>",
  "environment": {
    "type": "LINUX_CONTAINER",
    "computeType": "BUILD_GENERAL1_SMALL",
    "image": "aws/codebuild/amazonlinux2-x86_64-standard:4.0"
  }
```

```
}
```

プロジェクトの現在のプロパティ値を取得する必要がある場合は、batch-get-projects コマンドを使用して、変更するプロジェクトの現在のプロパティを取得し、出力をファイルに書き込みます。

```
aws codebuild batch-get-projects --names "<project-name>" > project-info.json
```

project-info.json ファイルには、プロジェクトの配列が含まれているため、プロジェクトを更新するために直接使用することはできません。ただし、変更したいプロパティを *project-info.json* ファイルからコピーして更新ファイル内に貼り付け、変更するプロパティのベースラインとすることができます。詳細については、「ビルドプロジェクトの詳細を表示する (AWS CLI)」を参照してください。

「ビルドプロジェクトの作成 (AWS CLI)」の説明に従って、更新 JSON ファイルを変更し、結果を保存します。更新 JSON ファイルの変更が完了したら、update-project コマンドを実行し、更新 JSON ファイルを渡します。

```
aws codebuild update-project --cli-input-json file://<update-project-file>
```

成功した場合、更新されたプロジェクトの JSON が出力に表示されます。必要なパラメータが欠落している場合は、欠落しているパラメータを識別するエラーメッセージが出力に表示されます。たとえば、このエラーメッセージは、「environment/type」パラメータが無いエラーです。

```
aws codebuild update-project --cli-input-json file://update-project.json

Parameter validation failed:
Missing required parameter in environment: "type"
```

ビルドプロジェクトの設定の変更 (AWS SDK)

AWS CodeBuild を AWS SDK と組み合わせて使用する方法については、「AWS SDK とツールのリファレンス」を参照してください。

AWS CodeBuild でのビルドプロジェクトの削除

CodeBuild コンソール、AWS CLI、または AWS SDK を使用して、CodeBuild のビルドプロジェクトを削除できます。プロジェクトを削除しても、そのビルドは削除されません。

> **Warning**
>
> ビルドとリソースポリシーを持つプロジェクトは削除できません。リソースポリシーとビルドを持つプロジェクトを削除するには、最初にリソースポリシーを削除し、そのビルドを削除する必要があります。

トピック

ビルドプロジェクトの削除 (コンソール)

1. AWS CodeBuild コンソール (https://console.aws.amazon.com/codesuite/codebuild/home) を開きます。
2. ナビゲーションペインで、[Build projects] を選択します。
3. 次のいずれかを行ってください。
   - 削除するビルドプロジェクトの横にあるラジオボタンを選択して、[削除] を選択します。
   - 削除するビルドプロジェクトのリンクを選択し、[Delete] を選択します。

> **Note**
>
> デフォルトでは、最新の 10 個のビルドプロジェクトのみが表示されます。さらに多くのビルドプロジェクトを表示するには、[Projects per page] で別の値を選択するか、[Viewing projects] で前後の矢印を選択します。

ビルドプロジェクトの削除 (AWS CLI)

1. `delete-project` コマンドを実行します。

```
aws codebuild delete-project --name name
```

次のプレースホルダを置き換えます。

- *name*: 必須の文字列。削除するビルドプロジェクトの名前。使用可能なビルドプロジェクトのリストを取得するには、`list-projects` コマンドを実行します。詳細については、「ビルドプロジェクト名の一覧表示 (AWS CLI)」を参照してください。

2. 成功した場合、データは出力されず、エラーも出力に表示されません。

AWS CLI を AWS CodeBuild と組み合わせて使用する方法については、「コマンドラインリファレンス」を参照してください。

ビルドプロジェクトの削除 (AWS SDK)

AWS CodeBuild を AWS SDK と組み合わせて使用する方法については、「AWS SDK とツールのリファレンス」を参照してください。

共有プロジェクトの使用

プロジェクト共有により、プロジェクト所有者は他の AWS アカウントまたはユーザーと AWS CodeBuild プロジェクトを共有できます。このモデルでは、プロジェクトを所有するアカウント (所有者) は、他のアカウント (コンシューマー) とプロジェクトを共有します。コンシューマーは、プロジェクトを編集または実行できません。

コンテンツ

プロジェクトを共有するための前提条件

プロジェクトを共有するには、AWS アカウントがプロジェクトを所有している必要があります。自身が共有を受けているプロジェクトは共有できません。

自分に共有されている共有プロジェクトにアクセスするための前提条件

共有レポートグループにアクセスするには、コンシューマーの IAM ロールに `BatchGetProjects` アクセス許可が必要です。次のポリシーを IAM ロールに付けることができます。

```
{
    "Effect": "Allow",
    "Resource": [
        "*"
    ],
    "Action": [
        "codebuild:BatchGetProjects"
    ]
}
```

詳細については、「でのアイデンティティベースのポリシーの使用 AWS CodeBuild」を参照してください。

関連サービス

プロジェクト共有は、AWS Resource Access Manager (AWS RAM) と統合されます。これは、任意の AWS アカウントまたは AWS を通じて AWS Organizations リソースを共有することを可能にするサービスです。AWS RAM では、リソースを共有するリソースとコンシューマを指定するリソース共有を作成して、リソースを共有します。コンシューマーは、個別の AWS アカウントや、AWS Organizations 内の組織単位または AWS Organizations 組織全体として指定できます。

詳細については、AWS RAM ユーザーガイドを参照してください。

プロジェクトの共有

コンシューマーは、共有しているプロジェクトとビルドを表示するために AWS CLI と AWS CodeBuild コンソールの両方を使用できます。コンシューマーは、プロジェクトを編集または実行できません。

既存のリソース共有にプロジェクトを追加すること、そして、AWS RAM コンソールでプロジェクトを作成することもできます。

> **Note**
>
> リソース共有に追加されたビルドを含むプロジェクトは削除できません。

組織単位または組織全体でプロジェクトを共有するには、AWS Organizations との共有を有効にする必要があります。詳細については、AWS RAM ユーザーガイドの「AWS Organizations で共有を有効化する」を参照してください。

所有しているプロジェクトを共有するには、AWS CodeBuild コンソール、AWS RAM コンソール、または AWS CLI を使用できます。

所有するプロジェクトを共有するには (CodeBuild コンソール)

1. AWS CodeBuild コンソール (https://console.aws.amazon.com/codesuite/codebuild/home) を開きます。
2. ナビゲーションペインで、[Build projects] を選択します。

> **Note**
>
> デフォルトでは、最新の 10 個のビルドプロジェクトのみが表示されます。さらに多くのビルドプロジェクトを表示するには、歯車アイコンを選択して [Projects per page (ページ毎プロジェクト数)] で別の値を選択するか、前後の矢印を使用します。

3. 共有するプロジェクトを選択し、[Share (共有)] を選択します。詳細については、AWS RAM ユーザーガイドの「リソースの共有の作成」を参照してください。

所有するプロジェクトを共有するには (AWS RAM コンソール)

AWS RAM ユーザーガイド の「リソース共有の作成」を参照してください。

所有するプロジェクトを共有するには (AWS RAM コマンド)

create-resource-share コマンドを使用します。

所有するプロジェクトを共有するには (CodeBuild コマンド)

put-resource-policy コマンドを使用します:

1. `policy.json` という名前のファイルを作成し、その中に次をコピーします。

```
{
  "Version":"2012-10-17",
  "Statement":[{
    "Effect":"Allow",
```

```
    "Principal":{
      "AWS":"<consumer-aws-account-id-or-user>"
    },
    "Action":[
      "codebuild:BatchGetProjects",
      "codebuild:BatchGetBuilds",
      "codebuild:ListBuildsForProject"],
    "Resource":"<arn-of-project-to-share>"
  }]
}
```

2. 共有するプロジェクト ARN と識別子で policy.json を更新します。次の例では、「123456789012」で識別される AWS アカウントの root ユーザーに、読み取り専用アクセスを付与します。

```
{
  "Version":"2012-10-17",
  "Statement":[{
    "Effect":"Allow",
    "Principal":{
      "AWS": [
        "123456789012"
      ]
    },
    "Action":[
      "codebuild:BatchGetProjects",
      "codebuild:BatchGetBuilds",
      "codebuild:ListBuildsForProject"],
    "Resource":"arn:aws:codebuild:us-west-2:123456789012:project/my-project"
  }]
}
```

3. put-resource-policy コマンドを使用します。

```
aws codebuild put-resource-policy --resource-arn <project-arn> --policy file://
policy.json
```

4. AWS RAM リソース共有 ARNを取得する。

```
aws ram list-resources --resource-owner SELF --resource-arns <project-arn>
```

これにより、次のような応答が得られます。

```
{
  "resources": [
    {
      "arn": "<project-arn>",
      "type": "<type>",
      "resourceShareArn": "<resource-share-arn>",
      "creationTime": "<creation-time>",
      "lastUpdatedTime": "<last-update-time>"
    }
  ]
}
```

応答から、*<resource-share-arn>*値は、次のステップで使用します。

5. AWS RAM の「promote-resource-share-created-from-policy」コマンドを実行します。

```
aws ram promote-resource-share-created-from-policy --resource-share-arn <resource-share-arn>
```

共有プロジェクトの共有解除

ビルドを含む共有されていないプロジェクトには、その所有者のみがアクセスできます。プロジェクトの共有を解除すると、以前に共有した AWS アカウントまたはユーザーは、プロジェクトまたはそのビルドにアクセスできなくなります。

自己所有の共有プロジェクトを共有解除するには、それをリソース共有から削除する必要があります。これを実行するには、AWS CodeBuild コンソール、AWS RAM コンソール、または AWS CLI を使用できます。

所有している共有プロジェクトの共有を解除するには (AWS RAM コンソール)

AWS RAM ユーザーガイド の「リソース共有の更新」を参照してください。

所有する共有プロジェクトの共有を解除するには (AWS CLI)

disassociate-resource-share コマンドを使用します。

所有するプロジェクトの共有を解除する (CodeBuild コマンド)

delete-resource-policy コマンドを実行し、共有を解除するプロジェクトの ARN を指定します。

```
aws codebuild delete-resource-policy --resource-arn project-arn
```

共有プロジェクトの識別

所有者とコンシューマーは、AWS CLI を使用して共有プロジェクトを識別できます。

AWS アカウントまたはユーザーと共有するプロジェクトを特定するには (AWS CLI)

list-shared-projects コマンドを使用して、自分に共有されているプロジェクトを返します。

共有プロジェクトへのアクセス許可

所有者のアクセス許可

プロジェクトの所有者は、プロジェクトを編集し、それを使用してビルドを実行できます。

コンシューマーのアクセス許可

プロジェクトコンシューマーは、プロジェクトとそのビルドを表示できますが、プロジェクトを編集したり、プロジェクトを使用してビルドを実行することはできません。

AWS CodeBuild でのプロジェクトのタグ付け

タグは、ユーザーまたは AWS が AWS リソースに割り当てるカスタム属性ラベルです。各 AWS タグは 2 つの部分で構成されます:

- タグキー (例: `CostCenter`、`Environment`、`Project`、`Secret`)。タグキーでは、大文字と小文字が区別されます。
- タグ値として知られるオプションのフィールド (例、 `111122223333`、`Production`、チーム名など)。タグ値を省略すると、空の文字列を使用した場合と同じになります。タグキーと同様に、タグ値は大文字と小文字が区別されます。

これらは共にキーと値のペアと呼ばれます。プロジェクトに付けることができるタグの最大数、およびタグのキーと値の制限については、「タグ」を参照してください。

タグを使用すると、AWS リソースの特定と整理に役立ちます。多くの AWS のサービスではタグ付けがサポートされるため、さまざまなサービスからリソースに同じタグを割り当てて、リソースの関連を示すことができます。たとえば、S3 バケットに割り当てたものと同じタグを CodeBuild プロ

ジェクトに割り当てることができます。タグの使用の詳細については、「タグ付けのベストプラクティス」を参照してください。

CodeBuild では、主なリソースはプロジェクトとレポートグループです。CodeBuild コンソール、AWS CLI、CodeBuild API、または AWS SDK を使用して、プロジェクトのタグを追加、管理、削除できます。タグを使用して、プロジェクトを識別、整理、追跡するだけでなく、IAM ポリシーでタグを使用して、プロジェクトを表示および操作できるユーザーを管理することもできます。タグベースのアクセスポリシーの例については、「タグを使用した AWS CodeBuild リソースへのアクセスのコントロール」を参照してください。

トピック

- プロジェクトにタグを追加する
- プロジェクトのタグを表示する
- プロジェクトのタグを編集する
- プロジェクトからタグを削除する

プロジェクトにタグを追加する

プロジェクトにタグを追加すると、AWS リソースの識別と整理、アクセスの管理に役立ちます。まず、プロジェクトに 1 つ以上のタグ (キーと値のペア) を追加します。プロジェクトに付けることができるタグの数には制限があります。キーフィールドおよび値フィールドに使用できる文字には制限があります。詳細については、「タグ」を参照してください。タグを追加した後、IAM ポリシーを作成して、それらのタグに基づいてプロジェクトへのアクセスを管理できます。CodeBuild コンソールまたは AWS CLI を使用して、プロジェクトにタグを追加できます。

> **Important**
>
> プロジェクトにタグを追加する前に、タグを使用してビルドプロジェクトなどのリソースへのアクセスをコントロールする可能性のある IAM ポリシーを必ず確認してください。タグベースのアクセスポリシーの例については、「タグを使用した AWS CodeBuild リソースへのアクセスのコントロール」を参照してください。

プロジェクトの作成時にタグを追加する方法の詳細については、「プロジェクトにタグを追加する (コンソール)」を参照してください。

トピック

- プロジェクトにタグを追加する (コンソール)
- プロジェクトにタグを追加する (AWS CLI)

プロジェクトにタグを追加する (コンソール)

CodeBuild コンソールを使用して、CodeBuild プロジェクトに 1 つ以上のタグを追加できます。

1. CodeBuild コンソール (https://console.aws.amazon.com/codebuild/) を開きます。
2. [Build projects (ビルドプロジェクト)] で、タグを追加するプロジェクトの名前を選択します。
3. ナビゲーションペインで [Settings] (設定) を選択します。[Build project tags (ビルドプロジェクトのタグ)] を選択します。
4. プロジェクトにいずれのタグも追加されていない場合は、[Add tag (タグの追加)] を選択します。それ以外の場合は、[Edit]、[Add tag] の順に選択します。
5. [Key] に、タグの名前を入力します。[Value] では、任意でタグに値を追加できます。
6. (オプション) 別のタグを追加するには、[Add tag] を再度選択します。
7. タグの追加を完了したら、[Submit] を選択します。

プロジェクトにタグを追加する (AWS CLI)

プロジェクトの作成時にタグを追加するには、「ビルドプロジェクトの作成 (AWS CLI)」を参照してください。`create-project.json` で、タグを追加します。

以下のステップでは、AWS CLI の最新版をすでにインストールしているか、最新版に更新しているものとします。詳細については、「AWS Command Line Interface のインストール」を参照してください。

成功すると、このコマンドは何も返しません。

プロジェクトのタグを表示する

タグは、AWS リソースの識別と整理、アクセスの管理に役立ちます。タグの使用の詳細については、「タグ付けのベストプラクティス」ホワイトペーパーを参照してください。タグベースのアクセスポリシーの例については、「タグを使用した AWS CodeBuild リソースへのアクセスのコントロール」を参照してください。

プロジェクトのタグを表示する (コンソール)

CodeBuild コンソールを使用して、CodeBuild プロジェクトに関連付けられたタグを表示できます。

1. CodeBuild コンソール (https://console.aws.amazon.com/codebuild/) を開きます。
2. [Build projects (ビルドプロジェクトの)] で、タグを表示するプロジェクトの名前を選択します。
3. ナビゲーションペインで [Settings] (設定) を選択します。[Build project tags (ビルドプロジェクトのタグ)] を選択します。

プロジェクトのタグを表示する (AWS CLI)

ビルドプロジェクトのタグを表示するには、以下のコマンドを実行します。--names パラメータにはプロジェクトの名前を使用します。

```
aws codebuild batch-get-projects --names your-project-name
```

成功すると、このコマンドは、ビルドプロジェクトに関する以下のような JSON 形式の情報を返します。

```
{
    "tags": {
        "Status": "Secret",
        "Team": "JanesProject"
    }
}
```

プロジェクトにいずれのタグも追加されていない場合、tags セクションは空になります。

```
"tags": []
```

プロジェクトのタグを編集する

プロジェクトに関連付けられたタグの値を変更できます。キーの名前を変更することもできます。これは、現在のタグを削除して、新しい名前と他のタグと同じ値を持つ、別のタグを追加することになります。キーフィールドと値フィールドに使用できる文字には制限があることにご注意ください。詳細については、「タグ」を参照してください。

> **Important**
>
> プロジェクトのタグを編集すると、そのプロジェクトへのアクセスに影響を与える可能性があります。プロジェクトのタグの名前 (キー) または値を編集する前に、タグのキーや値を使用してビルドプロジェクトなどのリソースへのアクセスをコントロールする可能性のある

IAM ポリシーを必ず確認してください。タグベースのアクセスポリシーの例については、「タグを使用した AWS CodeBuild リソースへのアクセスのコントロール」を参照してください。

プロジェクトのタグを編集する (コンソール)

CodeBuild コンソールを使用して、CodeBuild プロジェクトに関連付けられたタグを表示できます。

1. CodeBuild コンソール (https://console.aws.amazon.com/codebuild/) を開きます。
2. [Build projects (ビルドプロジェクト)] で、タグを編集するプロジェクトの名前を選択します。
3. ナビゲーションペインで [Settings] (設定) を選択します。[Build project tags (ビルドプロジェクトのタグ)] を選択します。
4. Edit (編集) を選択します。
5. 次のいずれかを行ってください。
   - タグを変更するには、[Key] に新しい名前を入力します。タグの名前を変更することは、タグを削除して、新しいキー名を持つタグを追加することになります。
   - タグの値を変更するには、新しい値を入力します。値を空にする場合は、現在の値を削除してフィールドを空のままにします。
6. タグの編集を完了したら、[Submit] を選択します。

プロジェクトのタグを編集する (AWS CLI)

ビルドプロジェクトのタグを追加、変更、または削除するには、「ビルドプロジェクトの設定の変更 (AWS CLI)」を参照してください。プロジェクトの更新に使用する JSON 形式のデータの `tags` セクションを更新します。

プロジェクトからタグを削除する

プロジェクトに関連付けられた 1 つ以上のタグを削除できます。タグを削除しても、そのタグに関連付けられた他の AWS リソースからタグを削除することにはなりません。

> **Important**
>
> プロジェクトからタグを削除すると、そのプロジェクトへのアクセスに影響を与える可能性があります。プロジェクトからタグを削除する前に、タグのキーや値を使用してビルドプロ

ジェクトなどのリソースへのアクセスをコントロールする可能性のある IAM ポリシーを必ず確認してください。タグベースのアクセスポリシーの例については、「タグを使用した AWS CodeBuild リソースへのアクセスのコントロール」を参照してください。

プロジェクトからタグを削除する (コンソール)

CodeBuild コンソールを使用して、タグと CodeBuild プロジェクトとの関連付けを解除できます。

1. CodeBuild コンソール (https://console.aws.amazon.com/codebuild/) を開きます。
2. [Build projects (ビルドプロジェクト)] で、タグを削除するプロジェクトの名前を選択します。
3. ナビゲーションペインで [Settings] (設定) を選択します。[Build project tags (ビルドプロジェクトのタグ)] を選択します。
4. Edit (編集) を選択します。
5. 削除するタグを見つけ、[Remove tag] を選択します。
6. タグの削除を完了したら、[Submit] を選択します。

プロジェクトからタグを削除する (AWS CLI)

ビルドプロジェクトから 1 つ以上のタグを削除するには、「ビルドプロジェクトの設定の変更 (AWS CLI)」を参照してください。JSON 形式のデータの `tags` セクションを、削除するタグが含まれていない最新のタグのリストで更新します。すべてのタグを削除する場合は、`tags` セクションを以下のように更新します。

```
"tags: []"
```

> **Note**
>
> CodeBuild ビルドプロジェクトを削除すると、削除されたビルドプロジェクトからすべてのタグの関連付けが解除されます。ビルドプロジェクトを削除する前にタグを削除する必要はありません。

でのバッチビルド AWS CodeBuild

を使用して AWS CodeBuild 、バッチビルドでプロジェクトの同時ビルドと調整ビルドを実行できます。

トピック

セキュリティロール

バッチビルドでは、バッチ設定に新しいセキュリティロールが導入されます。この新しいロールは、バッチの一部としてビルドを実行するために、 がユーザーに代わって StartBuild、StopBuild、および RetryBuildアクションを呼び出すことができる CodeBuild 必要があるため必要です。次の2つの理由により、お客様はビルドで使用するものと同じロールではなく、新しいロールを使用する必要があります。

- ビルドの役割を与える StartBuild、StopBuild、および RetryBuild アクセス権限を使用すると、単一のビルドが buildspec を介してより多くのビルドを開始することができます。
- CodeBuild バッチビルドには、バッチ内のビルドに使用できるビルドとコンピューティングタイプの数を制限する制限があります。ビルドロールにこれらの権限がある場合、ビルド自体がこれらの制限を回避する可能性があります。

バッチビルドのタイプ

CodeBuild では、次のバッチビルドタイプがサポートされています。

バッチビルドのタイプ

ビルドグラフ

ビルドグラフは、バッチ内の他のタスクに依存する一連のタスクを定義します。

次の例では、依存関係チェーンを作成するビルドグラフを定義します。

```
batch:
```

```
  fast-fail: false
  build-graph:
    - identifier: build1
      env:
        variables:
          BUILD_ID: build1
      ignore-failure: false
    - identifier: build2
      buildspec: build2.yml
      env:
        variables:
          BUILD_ID: build2
      depend-on:
        - build1
    - identifier: build3
      env:
        variables:
          BUILD_ID: build3
      depend-on:
        - build2
```

この例では、以下のことを行います。

- build1 は、依存関係を持たないため、最初に実行されます。
- build2 は build1 への依存関係があるため、build2 は build1 の完了後に実行されます。
- build3 は build2 への依存関係があるため、build3 は build2 の完了後に実行されます。

ビルドグラフの buildspec 構文の詳細については、「batch/build-graph」を参照してください。

ビルドリスト

ビルドリストは、並行して実行されるタスクの数を定義します。

次の例では、ビルドリストを定義します。build1 ビルドと build2 ビルドは並行して実行されます。

```
batch:
  fast-fail: false
  build-list:
    - identifier: build1
      env:
```

```
        variables:
          BUILD_ID: build1
      ignore-failure: false
    - identifier: build2
      buildspec: build2.yml
      env:
        variables:
          BUILD_ID: build2
      ignore-failure: true
```

ビルドリストの buildspec 構文の詳細については、「batch/build-list」を参照してください。

ビルドマトリックス

ビルドマトリックスは、並行して実行される異なる構成のタスクを定義します。 CodeBuild は、可能な設定の組み合わせごとに個別のビルドを作成します。

次の例は、2 つの buildspec ファイルと環境変数の 3 つの値を含むビルド行列を示しています。

```
batch:
  build-matrix:
    static:
      ignore-failure: false
    dynamic:
      buildspec:
        - matrix1.yml
        - matrix2.yml
      env:
        variables:
          MY_VAR:
            - VALUE1
            - VALUE2
            - VALUE3
```

この例では、 は 6 つのビルド CodeBuild を作成します。

- matrix1.yml (を含む)$MY_VAR=VALUE1
- matrix1.yml (を含む)$MY_VAR=VALUE2
- matrix1.yml (を含む)$MY_VAR=VALUE3
- matrix2.yml (を含む)$MY_VAR=VALUE1
- matrix2.yml (を含む)$MY_VAR=VALUE2

- matrix2.yml (を含む)$MY_VAR=VALUE3

各ビルドには次の設定があります。

- ignore-failure が false に設定
- env/type が LINUX_CONTAINER に設定
- env/image が aws/codebuild/amazonlinux2-x86_64-standard:4.0 に設定
- env/privileged-mode が true に設定

これらのビルドは並行して実行されます。

ビルドマトリックスの buildspec 構文の詳細については、「batch/build-matrix」を参照してください。

バッチレポートモード

プロジェクトのソースプロバイダーが Bitbucket GitHub、または GitHub Enterprise で、プロジェクトがソースプロバイダーにビルドステータスを報告するように設定されている場合は、バッチビルドステータスをソースプロバイダーに送信する方法を選択できます。バッチに関する単一の集約ステータスレポートとしてステータスを送信する、またはバッチ内の各ビルドのステータスを個別に報告することを選択できます。

詳細については、次のトピックを参照してください。

- Batch 設定 (作成)
- Batch 設定 (更新)

詳細情報

詳細については、以下のトピックを参照してください。

- バッチビルドのビルド仕様 (buildspec) のリファレンス
- Batch 構成
- バッチビルドの実行 (AWS CLI)
- AWS CodeBuild でバッチビルドを停止する

GitHub のアクションランナー AWS CodeBuild

GitHub アクションは、 GitHub ワークフローで使用するために特別に開発されたアクションです。GitHub アクションの詳細については、GitHub 「アクション」のドキュメントを参照してください。

で GitHub アクションを使用するには、次の 2 つの方法があります CodeBuild。

- Actions ワークフロージョブを処理するコンテナにセルフホスト型 GitHub Actions GitHub ランナーを設定する CodeBuild ようにプロジェクトを設定できます。
- CodeBuildマネージドアクションランナーを使用して、 内で GitHub アクションを実行できます CodeBuild。

でセルフホスト型 GitHub Actions ランナーを設定できます CodeBuild。これには、 CodeBuild プロジェクトを使用してウェブフックを設定し、 CodeBuild マシンでホストされているセルフホストランナーを使用するように GitHub アクションワークフロー YAML を更新する必要があります。これにより、 GitHub アクションワークフロージョブは とのネイティブ統合を取得できます AWS。

CodeBuildマネージドアクションランナーを使用して、 内で GitHub アクションを実行することもできます CodeBuild。これには、 CodeBuild コマンドとは別のフェーズで実行される GitHub Actions 構文を使用して buildspec `steps`に を追加する必要があります。これにより、 GitHub アクションを依存関係のキャッシュやバッチビルドなどの CodeBuild 機能と統合できます。

トピック

- でセルフホスト GitHub アクションランナーを設定する AWS CodeBuild
- の buildspec で GitHub Actions 構文を使用する AWS CodeBuild

でセルフホスト GitHub アクションランナーを設定する AWS CodeBuild

Actions GitHub ワークフロージョブを処理するコンテナにセルフホスト型 GitHub Actions ランナーを設定する CodeBuild ようにプロジェクトを設定できます。これを行うには、 CodeBuild プロジェクトを使用してウェブフックを設定し、 CodeBuild マシンでホストされているセルフホストランナーを使用するように GitHub アクションワークフロー YAML を更新します。詳細については、「セルフホストランナーについて」を参照してください。

GitHub Actions ジョブを実行するように CodeBuild プロジェクトを設定する大まかな手順は次のとおりです。

1. まだ行っていない場合は、個人用アクセストークンを作成するか、OAuth アプリに接続してプロジェクトを に接続します GitHub。
2. CodeBuild コンソールに移動し、ウェブフックを使用して CodeBuild プロジェクトを作成し、ウェブフックフィルタを設定します。
3. で GitHub アクションワークフロー YAML を更新 GitHub して、ビルド環境を設定します。

詳細な手順については、「」を参照してくださいチュートリアル: CodeBuild セルフホスト GitHub アクションランナーを設定する。

この機能により、 GitHub アクションワークフロージョブは とネイティブに統合できます。これにより AWS、IAM、 AWS Secrets Manager 統合 AWS CloudTrail、Amazon VPC などの機能を通じてセキュリティと利便性が得られます。ARM ベースのインスタンスを含む最新のインスタンスタイプにアクセスできます。

トピック

- チュートリアル: CodeBuild セルフホスト GitHub アクションランナーを設定する
- CodeBuild-hosted GitHub Actions ランナーについて

チュートリアル: CodeBuild セルフホスト GitHub アクションランナーを設定する

このチュートリアルでは、 GitHub Actions ジョブを実行するように CodeBuild プロジェクトを設定する方法を示します。

前提条件

このチュートリアルを完了するには、まず以下を行う必要があります。

- OAuth アプリに接続するか、個人用アクセストークンを作成します。OAuth アプリに接続する場合は、 CodeBuild コンソールを使用して接続する必要があります。個人用アクセストークンを作成する場合は、 CodeBuild コンソールまたは ImportSourceCredentials API を使用できます。手順については、「」を参照してくださいGitHub および GitHub Enterprise Server アクセストークン。
- GitHub アカウント CodeBuild に接続します。これを行うには、次のいずれかで実行できます。
  - コンソールでソースプロバイダー GitHub として を追加できます。OAuth アプリまたは個人用アクセストークンを使用して接続できます。手順については、「アクセストークン GitHub で接続する (コンソール) 」を参照してください。

- API を介して GitHub 認証情報をインポートできますImportSourceCredentials 。これは、個人用のアクセストークンでのみ実行できます。OAuth アプリを使用して接続する場合は、代わりにコンソールを使用して接続する必要があります。手順については、「 アクセストークン GitHub で接続する (CLI) 」を参照してください。

> **Note**
>
> これは、 GitHub アカウントの に接続していない場合にのみ行う必要があります。

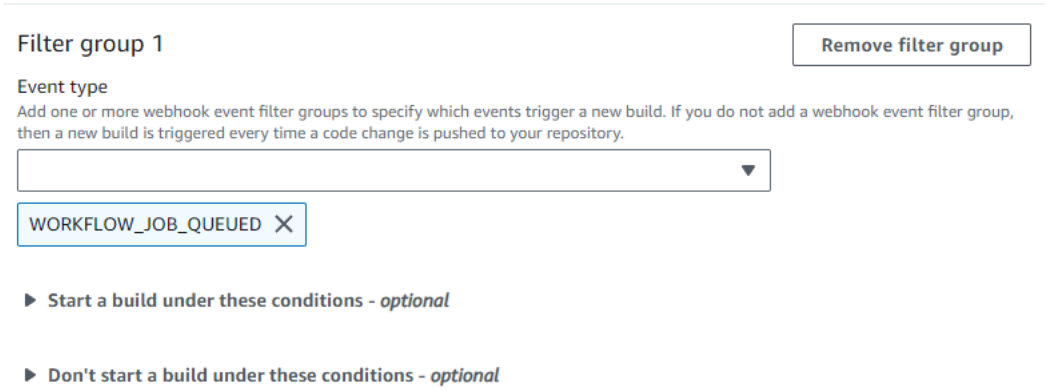
ステップ 1: ウェブフックを使用して CodeBuild プロジェクトを作成する

このステップでは、ウェブフックを使用して CodeBuild プロジェクトを作成し、 GitHub コンソールで確認します。

ウェブフックを使用して CodeBuild プロジェクトを作成するには

1. https://console.aws.amazon.com/codesuite/codebuild/home で AWS CodeBuild コンソールを開きます。
2. ビルドプロジェクトを作成します。詳細については、「ビルドプロジェクトの作成 (コンソール)」および「ビルドの実行 (コンソール)」を参照してください。
   - [Source (ソース)] で、次のようにします。
     - ソースプロバイダー で、 を選択しますGitHub。
     - リポジトリ で、 GitHub アカウント のリポジトリ を選択します。
     - [リポジトリの URL] に、「**`https://github.com/`*`user-name`*`/`*`repository-name`***」と入力します。
   - プライマリソースのウェブフックイベント :
     - Webhook - オプションの で、コード変更がこのリポジトリにプッシュされるたびに再構築を選択します。
     - イベントタイプ で、WORKFLOW_JOB_QUEUED を選択します。これを有効にすると、ビルドは GitHub Actions ワークフロージョブイベントによってのみトリガーされます。

> **Note**
>
> CodeBuild は、ウェブフックに WORKFLOW_JOB_QUEUED イベントフィルターを含むフィルターグループがある場合にのみ、 GitHub アクションワークフロージョブイベントを処理します。



- [環境] で以下の操作を行います。
  - サポートされている環境イメージを選択し、 をコンピューティングします。 GitHub アクションワークフロー YAML のラベルを使用して、イメージとインスタンスの設定を上書きするオプションがあることに注意してください。詳細については、「ステップ 2: GitHub アクションワークフロー YAML を更新する」を参照してください。
- [Buildspec (Buildspec)] で、次のようにします。
  - Buildspec は無視されることに注意してください。代わりに、セルフホストランナーをセットアップするコマンドを使用するように CodeBuild オーバーライドします。このプロジェクトの主な責任は、 でセルフホストランナーをセットアップ CodeBuild して GitHub Actions ワークフロージョブを実行することです。

3. デフォルト値を続行し、ビルドプロジェクトの作成を選択します。
4. で GitHub コンソールを開き`https://github.com/`*`user-name`*`/`*`repository-name`*`/settings/hooks`、ウェブフックが作成され、ワークフロージョブイベントの配信が有効になっていることを確認します。

ステップ 2: GitHub アクションワークフロー YAML を更新する

このステップでは、 の GitHub アクションワークフロー YAML ファイルを更新GitHubしてビルド環境を設定し、 の GitHub アクションセルフホストランナーを使用します CodeBuild。詳細については、「セルフホストランナーでのラベルの使用」を参照してください。

GitHub アクションワークフロー YAML を更新する

GitHub アクションワークフロー YAML の runs-on設定に移動GitHubして更新し、ビルド環境を設定します。これを行うには、次のいずれかで実行できます。

- プロジェクト名と実行 ID を指定できます。その場合、ビルドはコンピューティング、イメージ、イメージバージョン、インスタンスサイズに既存のプロジェクト設定を使用します。プロジェクト名は、 GitHub アクションジョブの AWS関連設定を特定の CodeBuild プロジェクトにリンクするために必要です。プロジェクト名を YAML に含めることで、 CodeBuild は正しいプロジェクト設定でジョブを呼び出すことができます。実行 ID を指定することで、ビルドを特定のワークフロー実行に CodeBuild マッピングし、ワークフロー実行がキャンセルされるとビルドを停止します。詳細については、github「コンテキスト 」を参照してください。

```
runs-on: codebuild-<project-name>-${{ github.run_id }}-${{ github.run_attempt }}
```

> **Note**
>
> *<project-name>* が、前のステップで作成したプロジェクトの名前と一致することを確認します。一致しない場合、ウェブフックを処理し CodeBuild ず、 GitHub アクションワークフローがハングする可能性があります。

GitHub アクションワークフロー YAML の例を次に示します。

```
name: Hello World
on: [push]
jobs:
  Hello-World-Job:
    runs-on: codebuild-myProject-${{ github.run_id }}-${{ github.run_attempt }}
    steps:
      - run: echo "Hello World!"
```

- ラベル内のイメージとコンピューティングタイプを上書きすることもできます。これにより、プロジェクトの環境設定が上書きされます。Amazon EC2 コンピューティングビルドの環境設定を上書きするには、次の構文を使用します。

```
runs-on: codebuild-<project-name>-${{ github.run_id }}-
${{ github.run_attempt }}-<image>-<image-version>-<instance-size>
```

Lambda コンピューティングビルドの環境設定を上書きするには、次の構文を使用します。

```
runs-on: codebuild-<project-name>-${{ github.run_id }}-
${{ github.run_attempt }}-<environment-type>-<runtime-version>-<instance-size>
```

GitHub アクションワークフロー YAML の例を次に示します。

```
name: Hello World
on: [push]
jobs:
  Hello-World-Job:
    runs-on: codebuild-myProject-${{ github.run_id }}-${{ github.run_attempt }}-
arm-3.0-small
    steps:
      - run: echo "Hello World!"
```

> **Note**
>
> GitHubホストされたランナーが提供する依存関係が CodeBuild 環境で使用できない場合は、ワークフロー実行の GitHub アクションを使用して依存関係をインストールできます。例えば、 setup-pythonアクションを使用してビルド環境に Python をインストールできます。

サポートされているコンピューティングイメージ

ラベルでは、最初の 3 つの列の値を使用して Amazon EC2 環境設定を上書きできます。 は、次の Amazon EC2 コンピューティングイメージ CodeBuild を提供します。

イメージ	イメージバージョン	インスタンスサイズ	プラットフォーム	イメージ識別子	定義
linux	4.0	small medium large xlarge	Amazon Linux 2	aws/codebuild/amazonlinux2-x86_64-standard:4.0	al2/standard/4.0
linux	5.0	2xlarge gpu_small gpu_large	Amazon Linux 2023	aws/codebuild/amazonlinux2-x86_64-standard:5.0	al2/standard/5.0
arm	2.0	small large	Amazon Linux 2	aws/codebuild/amazonlinux2-aarch64-standard:2.0	al2/aarch64/standard/2.0
arm	3.0		Amazon Linux 2023	aws/codebuild/amazonlinux2-aarch64-standard:3.0	al2/aarch64/standard/3.0
ubuntu	5.0	small medium large xlarge	Ubuntu 20.04	aws/codebuild/standard:5.0	ubuntu/standard/5.0

イメージ	イメージバージョン	インスタンスサイズ	プラットフォーム	イメージ識別子	定義
ubuntu	6.0	2xlarge gpu_small gpu_large	Ubuntu 22.04	aws/codebuild/standard:6.0	ubuntu/standard/6.0
ubuntu	7.0		Ubuntu 22.04	aws/codebuild/standard:7.0	ubuntu/standard/7.0
windows	1.0	medium large	Windows Server Core 2019	aws/codebuild/windows-base:2019-1.0	該当なし
windows	2.0		Windows Server Core 2019	aws/codebuild/windows-base:2019-2.0	該当なし
windows	3.0		Windows Server Core 2019	aws/codebuild/windows-base:2019-3.0	該当なし

さらに、以下の値を使用して Lambda 環境設定を上書きできます。 CodeBuild Lambda コンピューティングの詳細については、「 の使用」を参照してくださいでの AWS Lambda コンピューティングの使用 AWS CodeBuild。 は、次の Lambda コンピューティングイメージ CodeBuild をサポートします。

環境タイプ	ランタイムバージョン	インスタンスサイズ			
linux-lambda	dotnet6 go1.21	1GB 2GB			

環境タイプ	ランタイムバージョン	インスタンスサイズ			
arm-lambda	corretto11 corretto17 corretto21 nodejs18 nodejs20 python3.11 python3.12 ruby3.2	4GB 8GB 10GB			

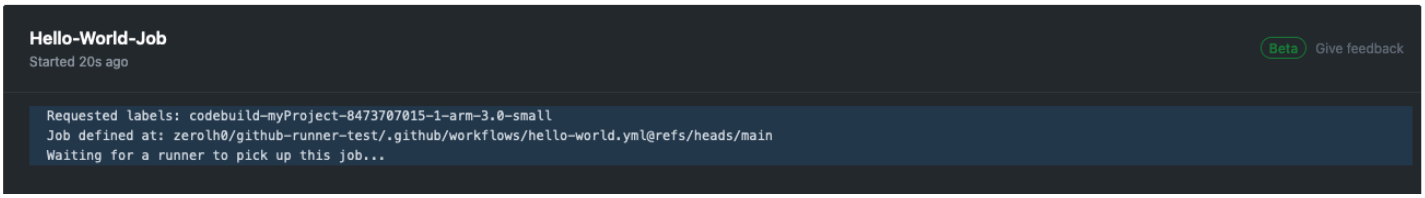
詳細については、「ビルド環境のコンピューティングモードおよびタイプ」および「が提供する Docker イメージ CodeBuild」を参照してください。

ステップ 3: 結果を確認する

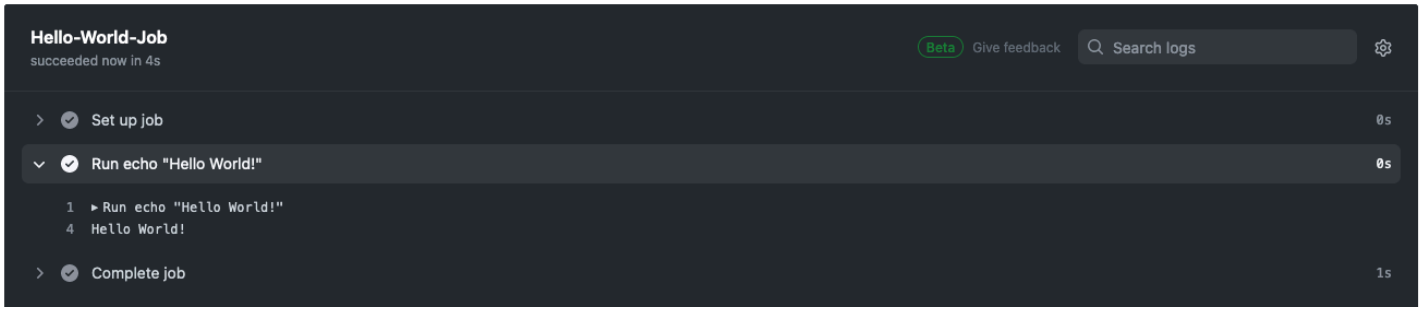
GitHub Actions ワークフローが実行されるたびに、 CodeBuild はウェブフックを介してワークフロージョブイベントを受信できます。ワークフロー内のジョブごとに、 はビルド CodeBuild を開始してエフェメラル GitHub Actions ランナーを実行します。ランナーは 1 つのワークフロージョブを実行します。ジョブが完了すると、ランナーと関連するビルドプロセスはすぐに終了します。

ワークフロージョブログを表示するには、 でリポジトリに移動し GitHub、アクション を選択し、目的のワークフローを選択して、ログを確認する特定のジョブを選択します。

ジョブが のセルフホストランナーによって取得されるのを待っている間に、ログでリクエストされたラベルを確認できます CodeBuild。



ジョブが完了すると、ジョブのログを表示できます。



CodeBuild-hosted GitHub Actions ランナーについて

ラベルにイメージとインスタンスの上書きを含める必要があるのはどのような場合ですか？

Actions GitHub ワークフロージョブごとに異なるビルド環境を指定するために、イメージとインスタンスのオーバーライドをラベルに含めることができます。これは、複数の CodeBuild プロジェクトやウェブフックを作成することなく実行できます。例えば、ワークフロージョブにマトリックス を使用する必要がある場合などに便利です。

```
name: Hello World
on: [push]
jobs:
  Hello-World-Job:
    runs-on: codebuild-myProject-${{ github.run_id }}-${{ github.run_attempt }}-
${{ matrix.os }}
    strategy:
      matrix:
        os: [arm-3.0-small, al2-5.0-large]
    steps:
      - run: echo "Hello World!"
```

> **Note**
>
> Actions GitHub コンテキストを含む複数のラベル`runs-on`がある場合は、引用符が必要になる場合があります。

この機能 AWS CloudFormation に を使用できますか？

はい。プロジェクトウェブフックの GitHub アクションワークフロージョブイベントフィルターを指定するフィルターグループを AWS CloudFormation テンプレートに含めることができます。

```
Triggers:
  Webhook: true
  FilterGroups:
    - - Type: EVENT
        Pattern: WORKFLOW_JOB_QUEUED
```

詳細については、「 GitHub ウェブフックイベントのフィルタリング (AWS CloudFormation) 」を参照してください。

AWS CloudFormation テンプレートでのプロジェクト認証情報の設定に関するヘルプが必要な場合は、「 AWS CloudFormation ユーザーガイドAWS::CodeBuild::SourceCredential」の「」を参照してください。

CodeBuild-hosted GitHub Actions ランナーの使用をサポートするリージョン

CodeBuildホストされた GitHub Actions ランナーは、すべての CodeBuild リージョンでサポートされています。 CodeBuild が利用可能な の場所の詳細については AWS リージョン 、AWS 「リージョン別のサービス」を参照してください。

CodeBuild-hosted GitHub Actions ランナーの使用をサポートするプラットフォーム

CodeBuildホストされた GitHub Actions ランナーは、Amazon EC2 と AWS Lambda コンピューティングの両方でサポートされています。Amazon Linux 2、Amazon Linux 2023、Ubuntu、および Windows Server Core 2019 プラットフォームを使用できます。詳細については、「EC2 コンピューティングイメージ」および「Lambda コンピューティングイメージ」を参照してください。

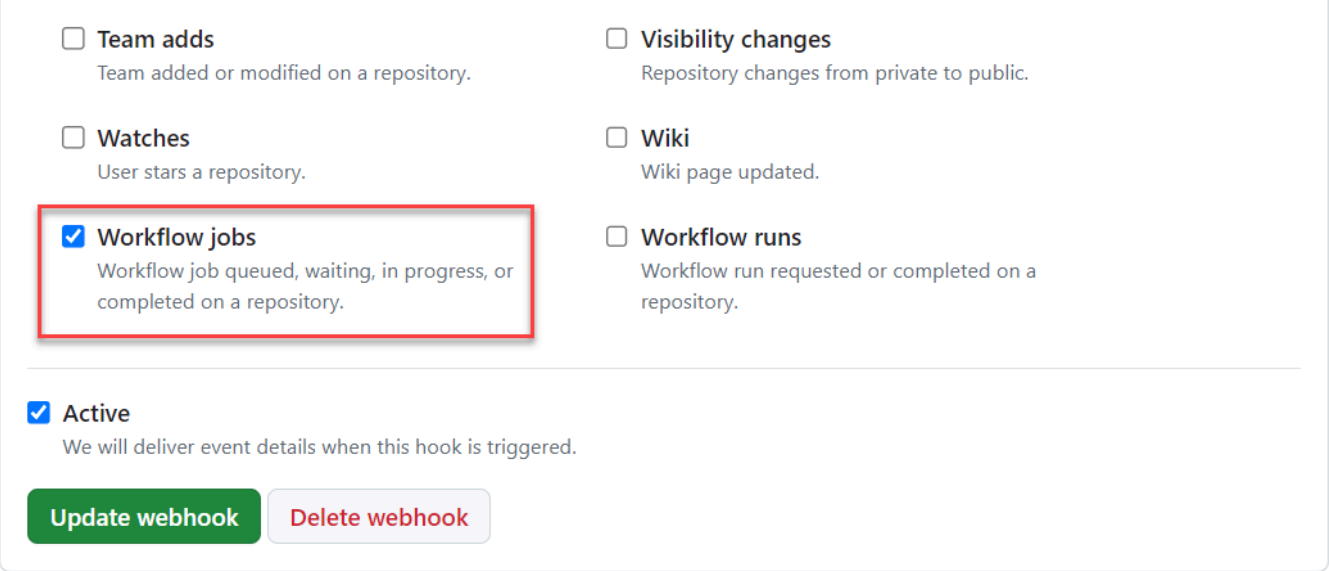
トラブルシューティング: ウェブフックが機能していない場合のトラブルシューティング方法を教えてください。

問題: ウェブフックが機能していないか、ワークフロージョブが にまとめられています GitHub。

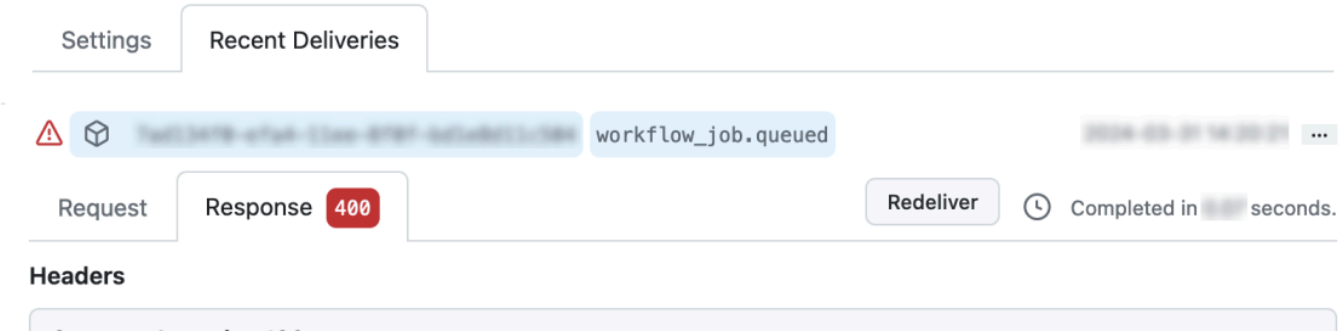
考えられる原因: Webhook ワークフロージョブイベントがビルドのトリガーに失敗している可能性があります。レスポンスログを確認して、レスポンスまたはエラーメッセージを表示します。

推奨される解決策: このエラーをデバッグするには、次の手順を使用します。

1. で GitHub コンソールを開きhttps://github.com/*user-name*/*repository-name*/settings/hooks、リポジトリのウェブフック設定を表示します。このページには、リポジトリ用に作成されたウェブフックが表示されます。
2. 編集を選択し、ウェブフックがワークフロージョブイベントの配信を有効にしていることを確認します。



3. 「最近の配信」タブに移動し、対応するworkflow_job.queuedイベントを見つけて、イベントを展開します。
4. ペイロードのラベルフィールドを確認し、想定どおりに動作していることを確認します。
5. 最後に、レスポンス タブを確認します。これには、 から返されるレスポンスまたはエラーメッセージが含まれているためです CodeBuild。



の buildspec で GitHub Actions 構文を使用する AWS CodeBuild

CodeBuildマネージドアクションランナーを使用して、 内で GitHub アクションを実行できます CodeBuild。これを行うには、buildspec ファイルの任意のフェーズに steps を追加します。

CodeBuild buildspecs は、 CodeBuild コマンドとは別のフェーズで実行されるシーケンシャル GitHub アクションステップのリストをサポートします。これらの GitHub アクションは、依存関係のキャッシュ、バッチビルド、 へのアクセスなど、 CodeBuildの既存の機能と統合されています AWS Secrets Manager。

トピック

- buildspec で GitHub アクションの使用を開始するにはどうすればよいですか？
- buildspec ではどの GitHub アクションを使用できますか？
- buildspec で GitHub アクション GitHub を使用する場合以外のソースプロバイダーを使用できますか？
- buildspec で GitHub アクションを使用するには、ソースプロバイダーとして に接続 GitHub する必要があるのはなぜですか？
- buildspec で GitHub アクションを使用するにはどのくらいのコストがかかりますか？
- buildspec での GitHub アクションの使用がサポートされているリージョン
- buildspec で GitHub アクションを使用するためのベストプラクティス
- で buildspec の GitHub アクションを使用する場合の制限 CodeBuild
- GitHub Action Runner buildspec リファレンス
- GitHub でのアクション構文の例 AWS CodeBuild

buildspec で GitHub アクションの使用を開始するにはどうすればよいですか？

buildspec で GitHub アクションを使用する大まかな手順は次のとおりです。

1. プロジェクトを に接続していない場合は、接続します GitHub。

   これを行うには、次のいずれかで実行できます。

   - コンソールでソースプロバイダー GitHub として を追加できます。詳細については、「アクセストークン GitHub で接続する (コンソール) 」を参照してください。
   - API を介して GitHub 認証情報をインポートできますCodeBuild 。詳細については、「 アクセストークン GitHub で接続する (CLI) 」を参照してください。

   > **Note**
   >
   > これは、別のプロジェクト GitHub で に接続していない場合にのみ行う必要があります。

2. プロジェクトの buildspec で、 を追加できます。steps各 は GitHub アクションを参照します。これは、 CodeBuild コンソールまたはソースリポジトリで編集できます。各ビルドフェーズは、コマンドのリストまたはステップのリストのどちらかをサポートしますが、両方を同じフェーズで使用することはできません。詳細については、「の buildspec で GitHub Actions 構文を使用する AWS CodeBuild」を参照してください。

buildspec ではどの GitHub アクションを使用できますか？

これらの制限と競合しない任意のアクションを GitHub Marketplace で使用できます。

buildspec で GitHub アクション GitHub を使用する場合以外のソースプロバイダーを使用できますか？

はい。ただし、 を使用して GitHub認証し、 GitHub アクションにアクセスするには、 への接続 GitHub が引き続き必要です。詳細については、「GitHub および GitHub Enterprise Server アクセストークン」を参照してください。

buildspec で GitHub アクションを使用するには、ソースプロバイダーとして に接続 GitHub する必要があるのはなぜですか？

buildspec で GitHub Actions を使用するには、ソースをビルドコンピューティングにダウンロードする必要があります。匿名ダウンロードはレート制限されるため、 に接続することで GitHub、一貫したアクセスを確保できます。

buildspec で GitHub アクションを使用するにはどのくらいのコストがかかりますか？

buildspec での GitHub アクションの使用は、追加料金なしでサポートされます。

buildspec での GitHub アクションの使用がサポートされているリージョン

buildspec での GitHub アクションの使用は、すべての CodeBuild リージョンでサポートされています。 CodeBuild が利用可能な の場所の詳細については AWS リージョン 、AWS 「リージョン別のサービス」を参照してください。

buildspec で GitHub アクションを使用するためのベストプラクティス

GitHub アクションはオープンソースであり、コミュニティによって構築され、管理されています。責任共有モデルに従い、 GitHub アクションのソースコードを、お客様の責任を持つ顧客データと見なします。 GitHub アクションには、シークレット、リポジトリトークン、ソースコード、およびアカウントリンクへのアクセス権を付与できます。実行する予定の GitHub アクションの信頼性とセキュリティに自信があることを確認してください。

GitHub アクションのより具体的なガイダンスとセキュリティのベストプラクティス：

- セキュリティ強化
- pwn 要求の阻止
- 信頼できない入力
- ビルディングブロックを信頼する方法

で buildspec の GitHub アクションを使用する場合の制限 CodeBuild

- GitHub `github`コンテキストに内部的に依存する、またはプルリクエストや問題などの GitHub固有のリソースを参照する buildspec 内のアクションは、 ではサポートされていません CodeBuild。例えば、次のアクションは では機能しません CodeBuild。
  - GitHub プルリクエストを更新したり、 で問題を作成したりするアクションなど、リソースの追加、変更、更新 GitHubを試みるアクション GitHub。

  > **Note**
  >
  > https://github.com/actions に記載されているほとんどの公式 GitHub アクションは、`github`コンテキストに依存しています。代わりに、 GitHub Marketplace で利用できるアクションを使用してください。

- GitHub Docker コンテナアクションである buildspec のアクションは機能しますが、ビルドプロジェクトで特権モードが有効になっていて、デフォルトの Docker ユーザー (ルート) によって実行する必要があります。
  - アクションは root ユーザーとして実行する必要があります。詳細については、Dockerfile support for Actions の「USER」トピックを参照してください。 GitHub
- GitHub buildspec のアクションは、Windows で実行するように設定された CodeBuild プロジェクトではサポートされていません。
- GitHub buildspec のアクションジョブ (ステップのグループ) と GitHub アクションジョブのプロパティはサポートされていません。
- GitHub buildspec のアクションは、パブリック Git リポジトリのウェブフックによってトリガーされるように設定された CodeBuild プロジェクトではサポートされていません。詳細については、「」を参照してくださいgit-credential-helper。
- パブリックインターネットアクセスのない VPC ビルドでは、buildspec で GitHub アクションを実行できません。

- 各ビルドフェーズはコマンドのリストまたはステップのリストのどちらかをサポートしますが、両方を同じフェーズで使用することはできません。例えば、次の例では、ビルド前のフェーズでステップを使用して GitHub アクションを一覧表示し、ビルドフェーズでコマンドを使用して CodeBuild コマンドを一覧表示します。

```
version: 0.2
phases:
  pre-build:
    steps:
      - name: Lint Code Base
        uses: github/super-linter@v4
        env:
          VALIDATE_ALL_CODEBASE: 'true'
          DEFAULT_BRANCH: main
  build:
    commands:
      - echo "Building..."
      - npm run build
```

GitHub Action Runner buildspec リファレンス

このトピックには、 GitHub Action ランナープロパティの buildspec リファレンスが含まれています。

steps

オプションのシーケンス。ステップは、 でコマンドとアクションを実行するために使用されます CodeBuild。詳細については、「の buildspec で GitHub Actions 構文を使用する AWS CodeBuild」を参照してください。

> **Note**
>
> 各ビルドフェーズは、commands のリストまたは steps のリストのどちらかをサポートしますが、両方を同じフェーズで使用することはできません。

各ビルドステップには、以下のプロパティが含まれます。

id

オプション。他のコンテキストでステップを参照するために使用できるステップの識別子。

if

オプション。条件が満たされない限り、ステップが実行されないようにするために使用できる条件付きステートメント。このステートメントでは、 からの環境変数の参照や式 https://docs.github.com/en/actions/learn-github-actions/contextsなど CodeBuild、サポートされているコンテキスト を使用できます。

name

オプション。ステップの名前。名前を指定しないと、名前はデフォルトで `run` コマンドで指定したテキストになります。

uses

ステップで実行するアクション。一部のアクションでは、`with` を使用して入力を設定する必要があります。アクションの README を参照して、どの入力が必要かを判断してください。詳細については、「buildspec ではどの GitHub アクションを使用できますか？」を参照してください。

ビルドフェーズで `uses` を指定した場合、`run` を併用することはできません。

> **Note**
>
> 使用しているアクションのバージョンを含めることをお勧めします。これを行うには、Git ref、SHA、または Docker タグを指定します。詳細については、「steps.uses 構文」を参照してください。

run

コマンドラインプログラムを実行するコマンド。1 行のコマンドでも、複数行のコマンドでもかまいません。デフォルトでは、これらのコマンドは非ログインシェルを使用して実行されます。別のシェルを選択するには、`shell` を使用します。

ビルドフェーズで `run` を指定した場合、`uses` を併用することはできません。

shell

オプション。このシーケンスで指定したシェル。サポートされているシェルパラメータについては、「steps.shell」を参照してください。指定しない場合、使用されるシェルは bash です。bash が使用できない場合は、 sh が使用されます。

with

オプション。アクションによって定義された入力パラメータのマップ。各パラメータはキーと値のペアです。

with.args

オプション。Docker コンテナの入力を定義する文字列。

with.entrypoint

オプション。Dockerfile に指定した Docker　エントリポイント。

env

オプション。環境で使用するためにステップに指定した変数。

continue-on-error

オプション。このステップシーケンスの失敗を無視できるかどうかを示すブール値。

`false`

デフォルト値。このステップシーケンスが失敗すると、ビルドは失敗します。

`true`

このステップシーケンスが失敗しても、ビルドは成功する可能性があります。

timeout-minutes

オプション。ステップが、終了するまでに実行できる最大分数。デフォルトでは、タイムアウトがありません。ステップのタイムアウトがビルドのタイムアウトを超えている場合、ビルドのタイムアウトに達すると、ステップは停止します。

Super-Linter GitHub アクションを使用した例を次に示します。

```
version: 0.2
phases:
  build:
    steps:
      - name: Lint Code Base
        uses: github/super-linter@v5
        env:
          VALIDATE_ALL_CODEBASE: true
          USE_FIND_ALGORITHM: true
          FILTER_REGEX_INCLUDE: '/github/workspace/buildspec.yml'
```

GitHub でのアクション構文の例 AWS CodeBuild

これらのサンプルグループは、 の buildspec で GitHub アクションを試すために使用できます CodeBuild。

トピック

- Super-Linter GitHub Action サンプル
- バッチビルドグラフのサンプル
- Amazon CodeGuru Reviewer サンプル
- AWS Secrets Manager サンプル
- 環境変数のサンプル
- エクスポートされた環境変数のサンプル

Super-Linter GitHub Action サンプル

このサンプルでは、 Super-Linter GitHub アクションを CodeBuild プロジェクトに追加する方法を示します。Super-Linter アクションは、コードを検査し、コードにエラー、フォーマットの問題、疑わしいコンストラクトがある領域を検出し、結果を CodeBuild コンソールに出力します。

buildspec ファイルの phase セクションを更新 CodeBuild することで、プロジェクトに Super-Linter GitHub アクションを追加できます。

```
version: 0.2
phases:
  build:
    steps:
      - name: Lint Code Base
        uses: github/super-linter@v5
        env:
          VALIDATE_ALL_CODEBASE: true
```

Super-Linter ログは次のようになります。

```
/github/workspace/hello-world/app.js:3:13: Extra semicolon.
/github/workspace/hello-world/app.js:9:92: Trailing spaces not allowed.
/github/workspace/hello-world/app.js:21:7: Unnecessarily quoted property 'body' found.
/github/workspace/hello-world/app.js:31:1: Expected indentation of 2 spaces but found
 4.
```

```
/github/workspace/hello-world/app.js:32:2: Newline required at end of file but not
 found.
```

バッチビルドグラフのサンプル

次の例では、依存関係チェーンを作成し、steps を使用してコマンドを実行するビルドグラフを定義します。この例では、依存関係を持たない build1 が最初に実行されます。build2 は build1 への依存関係があるため、build1 の完了後に build2 が実行されます。詳細については、「ビルドグラフ」を参照してください。

```
version: 0.2
batch:
  fast-fail: false
  build-graph:
    - identifier: build1
      env:
        variables:
          BUILD_ID: build1
      ignore-failure: false
    - identifier: build2
      env:
        variables:
          BUILD_ID: build2
      depend-on:
        - build1

phases:
  build:
    steps:
      - run: echo $BUILD_ID
```

Amazon CodeGuru Reviewer サンプル

Amazon CodeGuru Reviewer は、Java および Python コードで問題を検出し、修正方法を推奨します。次の例では、 CodeGuru レビュー担当者を使用してリポジトリ分析コードの完全なレビューを提供します。これらのコードレビューは、指定されたブランチ内のすべてのコードをスキャンします。詳細については、「Amazon CodeGuru Reviewer ユーザーガイド」の GitHub 「アクションを使用してコードレビューを作成する」を参照してください。

```
version: 0.2
phases:
```

```
  build:
    steps:
        - name: Amazon CodeGuru Reviewer Scanner
          if: ${{ always() }}
          uses: aws-actions/codeguru-reviewer@v1.1
          with:
            s3_bucket: codeguru-reviewer-user

artifacts:
  files:
    - codeguru-results.sarif.json
```

> **Note**
>
> Amazon S3 バケットは codeguru-reviewer- プレフィックスで始まる必要があります。

ログは、次のようになります。

```
INFO CodeReview created with arn=arn:aws:codeguru-reviewer:region:account-
id:association:id:code-review:RepositoryAnalysis-job for job=job
INFO SARIF persisted to /github/workspace/codeguru-results.sarif.json
INFO Amazon CodeGuru Reviewer job execution completed
```

Amazon CodeGuru Reviewer ジョブが完了すると、sarif レポートが CodeBuild アーティファクトとして生成されます。詳細については、「Amazon CodeGuru Reviewer ユーザーガイド」の「完全なリポジトリ分析」を参照してください。

AWS Secrets Manager サンプル

AWS Secrets Manager は、データベース認証情報、アプリケーション認証情報、OAuth トークン、API キー、およびその他のシークレットをライフサイクルを通じて管理、取得、ローテーションするのに役立ちます。次の例では、Secrets Manager を使用してシークレットを定義し、steps を使用してコマンドを実行します。詳細については、「 AWS Secrets Manager ユーザーガイド」の「 とは AWS Secrets Manager」を参照してください。

```
version: 0.2
env:
  secrets-manager:
    SECRET_VALUE: "arn:aws:secretsmanager:us-east-1:xxxx:secret:/secret-
l3IJg9:my_super_secret_key"
```

```
phases:
  build:
    steps:
      - run: echo $SECRET_VALUE
```

ログは、次のようになります。

```
echo $SECRET_VALUE
env:
  SECRET_VALUE: ***
***
```

環境変数のサンプル

次の例では、env シーケンスの下に環境変数を定義します。*S3_BUCKET* 変数は buildspec で定義され、その値として *<bucket-name>* が割り当てられます。この変数は、ドル記号 ($) を使用して GitHub Action env コンテキストにアクセスすることにより、通常の環境変数と同様に if 条件で参照されます。詳細については、「env シーケンス」を参照してください。

```
version: 0.2
env:
  variables:
    S3_BUCKET: "<bucket-name>"
phases:
  build:
    steps:
      - if: ${{ env.S3_BUCKET == '<bucket-name>' }}
        run: echo "S3 bucket is $S3_BUCKET"
```

ログは、次のようになります。

```
echo "S3 bucket is $S3_BUCKET"
env:
  S3_BUCKET: my-s3-bucket
S3 bucket is my-s3-bucket
```

エクスポートされた環境変数のサンプル

エクスポートされた環境変数は、 と組み合わせて使用され CodePipeline 、現在のビルドステージからパイプラインの後続のステージに環境変数をエクスポートします。次の例では、エクスポートされ

た環境変数を *MY_VARIABLE* という名前の env シーケンスで定義し、*GITHUB_ENV* 環境ファイルに書き込みます。

```
version: 0.2
env:
  exported-variables:
    - MY_VARIABLE
phases:
  build:
    steps:
      - run: echo "MY_VARIABLE=my-value" >> $GITHUB_ENV
```

詳細については、 AWS CodeBuild API リファレンスの ExportedEnvironmentVariable「」を参照してください。

AWS CodeBuild でのパブリックビルドプロジェクト

AWS CodeBuild を使うと、ビルドプロジェクトのビルド結果、ログ、アーティファクトを一般ユーザーに公開できます。これにより、ソースリポジトリのコントリビューターは、AWS アカウントへのアクセスを必要とせずに、ビルドの結果を表示したり、アーティファクトをダウンロードしたりできます。

プロジェクトのビルドを一般に公開すると、プロジェクトがプライベートだったときに実行されたビルドも含めて、プロジェクトのビルド結果、ログ、アーティファクトはすべて、一般に公開されます。同様に、パブリックビルドプロジェクトをプライベートにすると、そのプロジェクトのビルド結果は一般に公開されなくなります。

プロジェクトのビルド結果の公開範囲を変更する方法については、「パブリックビルドアクセスを有効にする」を参照してください。

CodeBuild では、お客様のプロジェクトに固有のパブリックビルド用 URL を提供しています。ビルドプロジェクトのパブリック URL を取得するには、以下の手順を実行します。

1. AWS CodeBuild コンソール (https://console.aws.amazon.com/codesuite/codebuild/home) を開きます。
2. ナビゲーションペインで、[Build projects] を選択します。
3. パブリック URL を取得したいビルドプロジェクトのリンクを選択します。
4. パブリック URL は、[Configuration] (構成) セクションの [Public project URL] (パブリックプロジェクト URL) フィールドに表示されます。リンクを選択して URL を開くか、コピーボタンを使用して URL をコピーすることができます。

> **Warning**
>
> プロジェクトのビルド結果を一般に公開する際には、以下に留意してください。
>
> - プロジェクトがプライベートだったときに実行されたビルドも含めて、プロジェクトのビルド結果、ログ、アーティファクトはすべて、一般に公開されます。
> - すべてのビルドログとアーティファクトが一般に公開されます。環境変数、ソースコード、およびその他の機密情報がビルドログとアーティファクトに出力されている可能性があります。ビルドログに出力される情報には注意が必要です。以下にベストプラクティスを示します。
>   - 機密値、特に AWS アクセスキー ID やシークレットアクセスキーを格納する場合には、環境変数を使用しないようにします。Amazon EC2 Systems Manager パラメータストアまたは AWS Secrets Manager を使用して機密値を格納することをお勧めします。
>   - ウェブフック使用のベストプラクティス。 に従って、ビルドをトリガーできるエンティティを制限し、buildspec をプロジェクト自体に保存しないことで、Webhook を可能な限り安全に保つことができます。
> - 悪意のあるユーザーがパブリックビルドを利用して、悪意のあるアーティファクトを配信する可能性があります。プロジェクト管理者は、すべてのプルリクエストを確認し、プルリクエストが正当な変更であるか検証することをお勧めします。また、チェックサムを使ってすべてのアーティファクトを検証し、正しいアーティファクトがダウンロードされているか確認することを推奨します。

AWS CodeBuild でのビルドの使用

ビルド は、一連の入力アーティファクト (Java クラスファイルのコレクションなど) に基づいて出力アーティファクト (JAR ファイルなど) を作成するために AWS CodeBuild によって実行される一連のアクションを表します。

複数のビルドを実行するときは、以下のルールが適用されます。

- 可能であれば、ビルドが同時に実行されます。同時実行ビルドの最大数は変化する可能性があります。詳細については、「」を参照してくださいAWS CodeBuild のクォータ
- ビルドプロジェクトに同時ビルド制限が設定されている場合、実行中のビルド数がプロジェクトの同時ビルド制限に達すると、ビルドがエラーを返します。詳細については、「同時ビルド制限を有効にする」を参照してください。

- ビルドプロジェクトに同時ビルド制限が設定されていない場合、実行中のビルド数がプラットフォームとコンピューティングタイプの同時ビルド制限に達すると、ビルドがキューに入れられます。キュー内のビルドの最大数は、同時ビルド制限の 5 倍です。詳細については、「」を参照してくださいAWS CodeBuild のクォータ

  タイムアウト値で指定された時間 (分) が経過しても開始されないキュー内のビルドは、キューから削除されます。デフォルトのタイムアウト値は 8 時間です。ビルドを実行するとき、5 分 ~ 8 時間の値でビルドのキュータイムアウトをオーバーライドできます。詳細については、「AWS CodeBuild でのビルドの実行」を参照してください。

  キューに入れられたビルドが開始される順序を予測することはできません。

> **Note**
>
> ビルドの履歴には、1 年間アクセスできます。

ビルドを操作するときに、次のタスクを実行できます。

トピック

AWS CodeBuild でのビルドの実行

AWS CodeBuild コンソール、AWS CLI、または AWS SDK を使用して、CodeBuild でビルドを実行できます。

トピック

- ビルドの実行 (コンソール)
- ビルドの実行 (AWS CLI)
- バッチビルドの実行 (AWS CLI)
- ビルドの実行の自動開始 (AWS CLI)
- ビルドの実行の自動停止 (AWS CLI)
- ビルドの実行 (AWS SDK)

ビルドの実行 (コンソール)

CodeBuild で AWS CodePipeline を使用してビルドを実行するには、この手順をスキップして「CodePipeline で を使用する CodeBuild」の手順に従います。

1. AWS CodeBuild コンソール (https://console.aws.amazon.com/codesuite/codebuild/home) を開きます。
2. ナビゲーションペインで、[Build projects] を選択します。
3. ビルドプロジェクトのリストで、ビルドプロジェクトを選択します。
4. デフォルトのビルドプロジェクト設定でビルドを実行することも、このビルドのみのビルド設定を上書きすることもできます。

   a. デフォルトのビルドプロジェクト設定を使用してビルドを実行するには、Start build を選択します。ビルドがすぐに開始されます。

   b. デフォルトのビルドプロジェクト設定を上書きする場合は、[Start build with overrides] を選択します。左の[Start build] ページで、以下を上書き可能です。

      - ビルドの設定
      - Source (送信元
      - 環境変数の上書き

      より高度なオーバーライドを選択する必要がある場合は、[Advanced build overrides (高度なビルドの上書き)] を選択します。このページでは、以下の操作を上書きできます。

      - ビルドの設定
      - Source (送信元
      - 環境 ()
      - BuildSpec

- アーティファクト
- ログ

オーバーライドを選択したら、[Start build] を選択します。

このビルドの詳細については、「ビルドの詳細の表示 (コンソール)」を参照してください。

ビルドの実行 (AWS CLI)

> **Note**
>
> CodePipeline で AWS CodeBuild を使用してビルドを実行するには、この手順をスキップして「CodeBuild を使用するパイプラインの作成 (AWS CLI)」の手順に従います。
> CodeBuild で AWS CLI を使用する方法の詳細については、「コマンドラインリファレンス」を参照してください。

1. 次のいずれかの方法で `start-build` コマンドを実行します。

   ```
   aws codebuild start-build --project-name <project-name>
   ```

   ビルド入力アーティファクトの最新バージョンとビルドプロジェクトの既存の設定を使用するビルドを実行する場合は、これを使用します。

   ```
   aws codebuild start-build --generate-cli-skeleton
   ```

   以前のバージョンのビルド入力アーティファクトを使用してビルドを実行する場合、またはビルド出力アーティファクト、環境変数、ビルド仕様、またはデフォルトのビルドタイムアウト期間の設定をオーバーライドする場合は、これを使用します。

2. `--project-name` オプションを指定して start-build コマンドを実行する場合は、*<project-name>* をビルドプロジェクトの名前に置き換えて、この手順のステップ 6 に進みます。ビルドプロジェクトのリストを表示するには、「ビルドプロジェクト名の一覧表示」を参照してください。
3. `--idempotency-token` オプションを指定して start-build コマンドを実行すると、大文字と小文字を区別する一意の識別子 (トークン) が `start-build` リクエストに含まれます。このトー

クンは、 リクエスト後 5 分間有効です。同じトークンで start-build リクエストを繰り返し行い、パラメータを変更すると、CodeBuild はパラメータの不一致エラーを返します。

4. start-build オプションを指定して --generate-cli-skeleton コマンドを実行すると、出力に JSON 形式のデータが表示されます。*start-build.json* がインストールされているローカルコンピュータまたはインスタンス上の場所にあるファイル (例: AWS CLI) にデータをコピーします。コピーしたデータを次の形式に変更して、結果を保存します。

```
{
  "projectName": "projectName",
  "sourceVersion": "sourceVersion",
  "artifactsOverride": {
    "type": "type",
    "location": "location",
    "path": "path",
    "namespaceType": "namespaceType",
    "name": "artifactsOverride-name",
    "packaging": "packaging"
  },
  "buildspecOverride": "buildspecOverride",
  "cacheOverride": {
    "location": "cacheOverride-location",
    "type": "cacheOverride-type"
  },
  "certificateOverride": "certificateOverride",
  "computeTypeOverride": "computeTypeOverride",
  "environmentTypeOverride": "environmentTypeOverride",
  "environmentVariablesOverride": {
    "name": "environmentVariablesOverride-name",
    "value": "environmentVariablesValue",
    "type": "environmentVariablesOverride-type"
  },
  "gitCloneDepthOverride": "gitCloneDepthOverride",
  "imageOverride": "imageOverride",
  "idempotencyToken": "idempotencyToken",
  "insecureSslOverride": "insecureSslOverride",
  "privilegedModeOverride": "privilegedModeOverride",
  "queuedTimeoutInMinutesOverride": "queuedTimeoutInMinutesOverride",
  "reportBuildStatusOverride": "reportBuildStatusOverride",
  "timeoutInMinutesOverride": "timeoutInMinutesOverride",
  "sourceAuthOverride": "sourceAuthOverride",
  "sourceLocationOverride": "sourceLocationOverride",
  "serviceRoleOverride": "serviceRoleOverride",
```

```
  "sourceTypeOverride": "sourceTypeOverride"
}
```

次のプレースホルダーを置き換えます。

- *projectName*: 必須の文字列。このビルドに使用するビルドプロジェクトの名前。
- *sourceVersion*: オプションの文字列。作成するソースコードのバージョンで、次のようになります。
  - Amazon S3 の場合、ビルドする入力 ZIP ファイルのバージョンに対応するバージョン ID。*sourceVersion* が指定されなければ、最新のバージョンが使用されます。
  - CodeCommit の場合、ビルドするソースコードのバージョンに対応するコミット ID。*sourceVersion* が指定されなければ、デフォルトブランチの HEAD コミット ID が使用されます。(*sourceVersion* にタグ名は指定できません。しかし、タグのコミット ID は指定できます。)
  - GitHub の場合、ビルドするソースコードのバージョンに対応するコミット ID、プルリクエスト ID、ブランチ名、またはタグ名。プルリクエスト ID を指定する場合、pr/*pull-request-ID* (例: pr/25) 形式を使用する必要があります。ブランチ名を指定すると、ブランチの HEAD コミット ID が使用されます。*sourceVersion* が指定されなければ、デフォルトブランチの HEAD コミット ID が使用されます。
  - Bitbucket の場合、ビルドするソースコードのバージョンに対応するコミット ID、ブランチ名、またはタグ名。ブランチ名を指定すると、ブランチの HEAD コミット ID が使用されます。*sourceVersion* が指定されなければ、デフォルトブランチの HEAD コミット ID が使用されます。
- 次に示すプレースホルダーは、artifactsOverride が対象です。
  - *type*: オプション。このビルドでオーバーライドするビルド出力アーティファクトタイプは、ビルドプロジェクトで定義されたものです。
  - *location*: オプション。このビルドでオーバーライドするビルド出力アーティファクトの場所は、ビルドプロジェクトで定義されたものです。
  - *path*: オプション。このビルドでオーバーライドするビルド出力アーティファクトパスは、ビルドプロジェクトで定義されたものです。
  - *namespaceType*: オプション。このビルドでオーバーライドするビルド出力アーティファクトパスのタイプは、ビルドプロジェクトで定義されたものです。
  - *name*: オプション。このビルドでオーバーライドするビルド出力アーティファクト名は、ビルドプロジェクトで定義されたものです。

  - *packaging*: オプション。このビルドでオーバーライドするビルド出力アーティファクトパッケージタイプは、ビルドプロジェクトで定義されたものです。

- *buildspecOverride*: オプション。ビルドプロジェクトに定義されている buildspec 宣言を上書きする、このビルドの buildspec 宣言。この値が設定されている場合は、インラインのビルド仕様定義か、組み込みの環境変数 CODEBUILD_SRC_DIR の値に相対的な代替 buildspec ファイルへのパスか、S3 バケットへのパスになります。S3 バケットは、ビルドプロジェクトと同じ AWS リージョンに存在する必要があります。ARN を使用して buildspec ファイルを指定します (例: arn:aws:s3:::*<my-codebuild-sample2>*/buildspec.yml) 。この値が指定されていない場合や、空の文字列に設定されている場合、ソースコードのルートディレクトリに buildspec.yml ファイルが含まれている必要があります。詳細については、「buildspec ファイル名とストレージの場所」を参照してください。
- 次に示すプレースホルダーは、cacheOverride が対象です。
  - *cacheOverride-location*: オプション。ビルドプロジェクトで指定された ProjectCache オブジェクトを上書きする、このビルドの ProjectCache オブジェクトの場所。cacheOverride はオプションで、ProjectCache オブジェクトを受け取ります。location は ProjectCache オブジェクトで必要です。
  - *cacheOverride-type*: オプション。ビルドプロジェクトで指定された ProjectCache オブジェクトを上書きする、このビルドの ProjectCache オブジェクトのタイプ。cacheOverride はオプションで、ProjectCache オブジェクトを受け取ります。type は ProjectCache オブジェクトで必要です。
- *certificateOverride*: オプション。ビルドプロジェクトで指定された証明書を上書きする、このビルドの証明書の名前。
- *environmentTypeOverride*: オプション。ビルドプロジェクトで指定されたコンテナタイプを上書きする、このビルドのコンテナタイプ。現在の有効な文字列は LINUX_CONTAINER です。
- 次に示すプレースホルダーは、environmentVariablesOverride が対象です。
  - *environmentVariablesOverride-name*: オプション。このビルドで値を上書きするビルドプロジェクトの環境変数の名前。
  - *environmentVariablesOverride-type*: オプション。このビルドで値を上書きするビルドプロジェクトの環境変数のタイプ。
  - *environmentVariablesValue*: オプション。このビルドで値を上書きするビルドプロジェクトで定義された環境変数の値。

- *gitCloneDepthOverride*: オプション。このビルドで上書きする、ビルドプロジェクトの [Git のクローンの深さ] の値。ソースタイプが Amazon S3 である場合、この値はサポートされません。
- *imageOverride*: オプション。ビルドプロジェクトで指定されたイメージを上書きする、このイメージの名前。
- *idempotencyToken*: オプション。ビルドリクエストがべき等であることを指定する、トークンとして機能する文字列。64 文字以下の任意の文字列を選択できます。このトークンは、ビルド開始リクエスト後 5 分間有効です。同じトークンでビルド開始リクエストを繰り返し行い、パラメータを変更すると、CodeBuild はパラメータの不一致エラーを返します。
- *insecureSslOverride*: ビルドプロジェクトに指定されている安全でない TLS 設定を上書きするかどうかを指定するブール値 (オプション)。安全でない TLS 設定により、プロジェクトのソースコードに接続するときに TLS 警告を無視するかどうかが決まります。この上書きが適用されるのは、ビルドのソースが GitHub Enterprise Server である場合のみです。
- *privilegedModeOverride*: オプションのブール値。true に設定すると、ビルドは、ビルドプロジェクトで権限モードを上書きします。
- *queuedTimeoutInMinutesOverride*: ビルドをキューに入れてからタイムアウトするまでの時間 (分) を指定するオプションの整数。その最小値は 5 分、最大値は 480 分 (8 時間) です。
- *reportBuildStatusOverride*: ビルドの開始と完了のステータスをソースプロバイダに送信するかどうかを指定するオプションのブール値。これを GitHub、GitHub Enterprise Server、Bitbucket 以外のソースプロバイダーに対して設定すると、invalidInputException がスローされます。
- *sourceAuthOverride*: オプションの文字列。ビルドプロジェクトで定義された認証タイプを上書きする、このビルドの認証タイプ。この上書きが適用されるのは、ビルドプロジェクトのソースが Bitbucket または GitHub である場合のみです。
- *sourceLocationOverride*: オプションの文字列。このビルドで、ビルドプロジェクトで定義されたソースの場所を上書きする場所。
- *serviceRoleOverride*: オプションの文字列。ビルドプロジェクトで指定されたサービスロールを上書きする、このビルドのサービスロールの名前。
- *sourceTypeOverride*: オプションの文字列。このビルドで、ビルドプロジェクトで定義されたソース入力を上書きするソース入力タイプ。有効な文字列は、`NO_SOURCE`、`CODECOMMIT`、`CODEPIPELINE`、`GITHUB`、`S3`、`BITBUCKET`、および `GITHUB_ENTERPRISE` です。

- *timeoutInMinutesOverride*: オプション番号。このビルドで上書きするビルドタイムアウトの分数は、ビルドプロジェクトで定義されたものです。

AWS アクセスキー ID、AWS シークレットアクセスキー、パスワードなどの機密値を持つ環境変数は、パラメータとして Amazon EC2 Systems Manager Parameter Store に保存することをお勧めします。CodeBuild では、Amazon EC2 Systems Manager パラメータストアに保存されているパラメータは、そのパラメータの名前が `/CodeBuild/` (例: `/CodeBuild/dockerLoginPassword`) で始まる場合にのみ使用できます。CodeBuild コンソールを使用して、Amazon EC2 Systems Manager にパラメータを作成することができます。[Create a parameter (パラメータの作成)] を選択し、手順に従います。(ダイアログボックスでは、[KMS キー] の場合、オプションでアカウントの AWS KMS キーの ARN を指定できます。Amazon EC2 Systems Manager では、このキーを使用して、保存中にパラメータの値を暗号化し、取得中に復号化します。) CodeBuild コンソールを使用してパラメータを作成した場合、コンソールは保存されている `/CodeBuild/` パラメータを開始します。ただし、Amazon EC2 Systems Manager パラメータストアコンソールを使用してパラメータを作成する場合、パラメータの名前を `/CodeBuild/` で開始する必要があり、[タイプ] を [Secure String (安全な文字列)] に設定する必要があります。詳細については、「Amazon EC2 Systems Manager ユーザーガイド」の「AWS Systems Manager パラメータストア」および「チュートリアル: String パラメータの作成とテスト (コンソール)」を参照してください。

ビルドプロジェクトが Amazon EC2 Systems Manager パラメータストアに保存されているパラメータを参照する場合、ビルドプロジェクトのサービスロールで `ssm:GetParameters` アクションを許可する必要があります。以前に [アカウントに新しいサービスロールを作成する] を選択している場合、CodeBuild は、このアクションをビルドプロジェクトのデフォルトのサービスロールに自動的に含めます。ただし [Choose an existing service role from your account] を選択した場合は、このアクションをサービスロールに個別に含める必要があります。

既存の環境変数は、設定した環境変数により置き換えられます。たとえば、Docker イメージに `my_value` の値を持つ `MY_VAR` という名前の環境変数が既に含まれていて、`other_value` の値を持つ `MY_VAR` という名前の環境変数を設定した場合、`my_value` が `other_value` に置き換えられます。同様に、Docker イメージに `/usr/local/sbin:/usr/local/bin` の値を持つ `PATH` という名前の環境変数が既に含まれていて、`$PATH:/usr/share/ant/bin` の値を持つ `PATH` という名前の環境変数を設定した場合、`/usr/local/sbin:/usr/local/bin` はリテラル値 `$PATH:/usr/share/ant/bin` に置き換えられます。

`CODEBUILD_` で始まる名前の環境変数は設定しないでください。このプレフィックスは内部使用のために予約されています。

同じ名前の環境変数が複数の場所で定義されている場合、環境変数の値は次のように決定されます。

- ビルド開始オペレーション呼び出しの値が最も優先順位が高くなります。
- ビルドプロジェクト定義の値が次に優先されます。
- buildspec ファイル宣言の値の優先順位が最も低くなります。

これらのプレースホルダの有効な値の詳細については、「ビルドプロジェクトの作成 (AWS CLI)」を参照してください。ビルドプロジェクトの最新の設定の一覧については、「ビルドプロジェクトの詳細を表示する」を参照してください。

5. 保存したばかりのファイルがあるディレクトリに移動し、start-build コマンドをもう一度実行します。

```
aws codebuild start-build --cli-input-json file://start-build.json
```

6. 成功した場合は、「ビルドを実行するには」の手順で説明されているのと同様のデータが出力に表示されます。

このビルドの詳細情報を使用するには、出力の id の値を書き留めてから、「ビルドの詳細の表示 (AWS CLI)」を参照してください。

バッチビルドの実行 (AWS CLI)

1. 次のいずれかの方法で start-build-batch コマンドを実行します。

```
aws codebuild start-build-batch --project-name <project-name>
```

ビルド入力アーティファクトの最新バージョンとビルドプロジェクトの既存の設定を使用するビルドを実行する場合は、これを使用します。

```
aws codebuild start-build-batch --generate-cli-skeleton > <json-file>
```

以前のバージョンのビルド入力アーティファクトを使用してビルドを実行する場合、またはビルド出力アーティファクト、環境変数、ビルド仕様、またはデフォルトのビルドタイムアウト期間の設定をオーバーライドする場合は、これを使用します。

2. `--project-name` オプションを指定して start-build-batch コマンドを実行する場合は、*<project-name>* をビルドプロジェクトの名前に置き換えて、この手順のステップ 6 に進みます。ビルドプロジェクトのリストを表示するには、「ビルドプロジェクト名の一覧表示」を参照してください。

3. start-build-batch オプションを指定して `--idempotency-token` コマンドを実行すると、大文字と小文字を区別する一意の識別子 (トークン) が `start-build-batch` リクエストに含まれます。このトークンは、 リクエスト後 5 分間有効です。同じトークンで `start-build-batch` リクエストを繰り返し行い、パラメータを変更すると、CodeBuild はパラメータの不一致エラーを返します。

4. `--generate-cli-skeleton` オプションを指定して start-build-batch コマンドを実行すると、出力に *<json-file>* 形式のデータが表示されます。このファイルは、start-build コマンド実行により生成されるスケルトンに似ていますが、次のオブジェクトが追加されています。共通オブジェクトの詳細については、「ビルドの実行 (AWS CLI)」を参照してください。

   このファイルを変更してビルドオーバーライドを追加し、結果を保存します。

```
"buildBatchConfigOverride": {
  "combineArtifacts": combineArtifacts,
  "restrictions": {
    "computeTypesAllowed": [
      allowedComputeTypes
    ],
    "maximumBuildsAllowed": maximumBuildsAllowed
  },
  "serviceRole": "batchServiceRole",
  "timeoutInMins": batchTimeout
}
```

   buildBatchConfigOverride オブジェクトは、ProjectBuildBatchConfig 構造体であり、このビルドのバッチビルド構成を上書きします。

   combineArtifacts

   バッチビルドのビルドアーティファクトを 1 つのアーティファクトの場所に結合するかどうかを指定するブール値。

   allowedComputeTypes

   バッチビルドで許可されるコンピューティングタイプを指定する文字列の配列。これらの値に対しては、「ビルド環境のコンピューティングタイプ」を参照してください。

maximumBuildsAllowed

許可されるビルドの最大数を指定します。

batchServiceRole

バッチビルドプロジェクトのサービスロール ARN を指定します。

#########

バッチビルドを完了するまでの最大時間 (分単位) を指定します。

5. 保存したばかりのファイルがあるディレクトリに移動し、start-build-batch コマンドをもう一度実行します。

```
aws codebuild start-build-batch --cli-input-json file://start-build.json
```

6. 成功した場合、JSON 表現の BuildBatch オブジェクトが、コンソール出力に表示されます。このデータの例については、「StartBuildBatch レスポンスの構文」を参照してください。

ビルドの実行の自動開始 (AWS CLI)

ソースコードを GitHub または GitHub Enterprise Server リポジトリに保存している場合は、コード変更がリポジトリにプッシュされるたびに AWS CodeBuild でソースコードを再ビルドするのに GitHub Webhook を使用できます。

次のように create-webhook コマンドを実行します。

```
aws codebuild create-webhook --project-name <project-name>
```

<project-name> は、再構築するソースコードを含むビルドプロジェクトの名前です。

GitHub では、次のような情報が出力に表示されます。

```
{
  "webhook": {
    "url": "<url>"
  }
}
```

<url> は GitHub ウェブフックへの URL です。

GitHub Enterprise Server の場合、以下のような情報が出力に表示されます。

```
{
    "webhook": {
        "secret": "YRV4JYAGFsekJiirp5ytx86oZpyhUdySNSDTLNuXoXX1c7aZ6XYDf37-ZFyY02rs4JSE7OmLW3w-gh-ryoVB8OSSSC1aAtBtuPkHw
YuncCCmdogCVCfniQ7ukYX2_xM--nlDma5EngIg_Bi_N465yi33zyTUNPoQlxCpLO-BwghcVa9lAurwR77-uY7i-_XCJFahwMx1f4ubOgBBSmMT2A16apqjq
QJoKSb61XVKyZy1Giuy4nliAXfv9WNn76CaCsndb3fVIE78fpygfo41xYxSQ6vpo6LRTKtPzbyeTHbVXGda1PJvnkBlnKmJDo0RTgI1m2oYr17dWziQlrrvo
CoNgy1SOO_7LKfA-nNXFc_f1SiFy0AqeMB43-dOOcdkzybHncE81QTRwEUCFfmX-AJCwmLXVOkg0G67T92Sjbpz0fRlkh5pwIFl93_bB_jOHDinK6iOiPpf2
dIDAIZgGMagqZeWb-axDeTAbopoU8J6gFI1yKo5aq9ql51zC1PERUsMgJFtJr_a-Z-L_kylr-4hSSxasSJNuJ43_XOBRWqT51xqvH-A69bVO7KbVT_Kc6wxk
SHyYCEMoa_Pfa7ZQgyfY6B00ogMNj31yFbjthORNL1cDo6-3J-McDLoyrRtSEOV9QnxvsG5zu1N5-z20rkJtg_MOfNwocfUutFXb7vrGTduHlR1dzXLRusHu
xOVVuDUWm9vhWMr-hUkeGo_1kDKyk4E2QFvZXpjYw0vFfV-dwxFRR_mifzxW1wyfmt2iFtLkp_YZj_4WeFAckGefr-ilNaYvsZpzXj78Ae1adVoLf48AmDdN
2pWsWJjatU9zt942gLisFFmKakcvJuy5yxXHaxxbhUyC8NHYiESUWPfcfnqrMsr8op3P4AUCHlpiZCYYuiwI_cac-pIUBO0Xaur_lu_fyFghgOJc7cfTnA36
rv5X5DnFDM8P3HNBeLjaF9QZ6AijegPEwTHIkJON3AUDwpkz_hwTXyUoAU8MdZfPTXbBoT6N5Z5THBHsYxR",
        "payloadUrl": "https://codebuild.us-east-2.amazonaws.com/webhooks?t=eyJlbmNyeXB0ZWREYXRhIjoiUmFqMmJERGRQbGhwLzNT
Nld3R0VGRjZZOTNwLzlZVG1NZlpIR1E0RUsxdzhGeWhnVFFqWTR0WEFwT2dJRnNmRHc3S3RNc0xYMENncXFTakg1cE1nSy9zPSIsIml2UGFyYW1ldGVyU3Bl
YyI6IndSQ1Qrc2VPQjBCZzhPeVYiLCJtYXRlcmlhbFNldFNlcmlhbCI6MX0%3D&v=1"
    }
}
```


1. 出力からシークレットキーとペイロード URL をコピーします。これらは、GitHub Enterprise Server に Webhook を追加するために必要となります。
2. GitHub Enterprise Server で、CodeBuild プロジェクトが保存されているリポジトリを選択します。[設定]、[Hooks & services]、[Add webhook] の順に選択します。
3. ペイロード URL とシークレットキーを入力し、その他のフィールドにはデフォルト値を選択して、[Add webhook] を選択します。

ビルドの実行の自動停止 (AWS CLI)

ソースコードを GitHub または GitHub Enterprise Server リポジトリに保存している場合は、コード変更がリポジトリにプッシュされるたびに AWS CodeBuild でソースコードを再ビルドするのに GitHub Webhook を設定できます。詳細については、「ビルドの実行の自動開始 (AWS CLI)」を参照してください。

この動作を有効にしている場合、次の `delete-webhook` コマンドを実行して無効化できます。

```
aws codebuild delete-webhook --project-name <project-name>
```

- *<project-name>* は、再構築するソースコードを含むビルドプロジェクトの名前です。

このコマンドが成功すると、情報やエラーはなにも出力に表示されません。

> **Note**
>
> これは、CodeBuild プロジェクトからのみ webhook を削除します。GitHub または GitHub Enterprise Server でも Webhook を削除する必要があります。

ビルドの実行 (AWS SDK)

CodePipeline で AWS CodeBuild を使用してビルドを実行するには、この手順をスキップして「AWS CodePipeline で AWS CodeBuild を使用してコードをテストし、ビルドを実行する」の手順に従います。

CodeBuild を AWS SDK と組み合わせて使用する方法については、「AWS SDK とツールのリファレンス」を参照してください。

AWS CodeBuild におけるビルドの詳細の表示

CodeBuild で管理されるビルドの詳細を表示するには、AWS CodeBuild コンソール、AWS CLI、または AWS SDK を使用します。

トピック

- ビルドの詳細の表示 (コンソール)
- ビルドの詳細の表示 (AWS CLI)
- ビルドの詳細の表示 (AWS SDK)
- ビルドフェーズの移行

ビルドの詳細の表示 (コンソール)

1. AWS CodeBuild コンソール (https://console.aws.amazon.com/codesuite/codebuild/home) を開きます。
2. 次のいずれかを行ってください。
   - ナビゲーションペインで、[Build history] を選択します。ビルドのリストの [Build run (ビルドの実行)] 列で、ビルドのリンクを選択します。
   - ナビゲーションペインで、[Build projects] を選択します。ビルドプロジェクトのリストの [名前] 列で、ビルドプロジェクト名のリンクを選択します。次に、ビルドのリストの [Build run (ビルドの実行)] 列で、ビルドのリンクを選択します。

> **Note**
>
> デフォルトでは、最新の 10 個のビルドまたはビルドプロジェクトのみ表示されます。さらに多くのビルドまたはビルドプロジェクトを表示するには、歯車アイコンを選択し

てから [ページ毎ビルド数] または [Projects per page (ページ毎プロジェクト数)] で別の値を選択するか、前後の矢印を使用します。

ビルドの詳細の表示 (AWS CLI)

AWS CLI を AWS CodeBuild と組み合わせて使用する方法については、「コマンドラインリファレンス」を参照してください。

batch-get-builds コマンドを実行します。

```
aws codebuild batch-get-builds --ids ids
```

次のプレースホルダを置き換えます。

- *ids*: 必須の文字列。詳細を表示する 1 つ以上のビルド ID。複数のビルド ID を指定するには、各ビルド ID をスペースで区切ります。最大 100 のビルド ID を指定できます。ビルド ID のリストを取得するには、次のトピックを参照してください。
  - ビルド ID の一覧表示 (AWS CLI)
  - ビルドプロジェクトのビルド ID を一覧表示する (AWS CLI)

たとえば、次のコマンドを実行するとします。

```
aws codebuild batch-get-builds --ids codebuild-demo-project:e9c4f4df-3f43-41d2-
ab3a-60fe2EXAMPLE codebuild-demo-project:815e755f-bade-4a7e-80f0-efe51EXAMPLE my-other-
project:813bb6c6-891b-426a-9dd7-6d8a3EXAMPLE
```

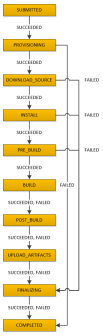
コマンドが正常に実行されると、「要約されたビルド情報を表示するには 」に示されているものと同様のデータが出力に表示されます。

ビルドの詳細の表示 (AWS SDK)

AWS CodeBuild を AWS SDK と組み合わせて使用する方法については、「AWS SDK とツールのリファレンス」を参照してください。

ビルドフェーズの移行

AWS CodeBuild のビルドはフェーズごとに進められます。



> **Important**
>
> UPLOAD_ARTIFACTS フェーズは、BUILD フェーズが失敗した場合でも必ず試行されます。

AWS CodeBuild のビルド ID の一覧表示

CodeBuild によって管理されるビルドのビルド ID のリストを表示するには、AWS CodeBuild コンソール、AWS CLI、または AWS SDK を使用します。

トピック

- ビルド ID の一覧表示 (コンソール)
- ビルド ID の一覧表示 (AWS CLI)
- バッチビルド ID のリストを表示する (AWS CLI)
- ビルド ID の一覧表示 (AWS SDK)

ビルド ID の一覧表示 (コンソール)

1. AWS CodeBuild コンソール (https://console.aws.amazon.com/codesuite/codebuild/home) を開きます。
2. ナビゲーションペインで、[Build history] を選択します。

> **Note**
>
> デフォルトでは、最新の 10 個のビルドのみ表示されます。さらに多くのビルドを表示するには、歯車アイコンを選択し、[Builds per page (ページ毎ビルド数)] で別の値を選択するか、前後の矢印を使用します。

ビルド ID の一覧表示 (AWS CLI)

CodeBuild で AWS CLI を使用する方法については、「コマンドラインリファレンス」を参照してください。

- list-builds コマンドを実行します。

```
aws codebuild list-builds --sort-order sort-order --next-token next-token
```

上記のコマンドで、次のプレースホルダを置き換えます。

- *sort-order*: ビルド ID の一覧表示方法を示すのに使用するオプションの文字列。有効な値は、ASCENDING および DESCENDING です。
- *next-token*: オプションの文字列。以前の実行中に、リストに 100 を超える項目がある場合、最初の 100 項目だけが、next token と呼ばれる一意の文字列と共に返されます。リスト内の項目の次のバッチを取得するには、次のコマンドを再度実行し、次のトークンを呼び出しに追加します。リスト内のすべての項目を取得するには、次のトークンが返されなくなるまで、このコマンドを、以後のすべての次のトークンで実行し続けます。

たとえば、次のコマンドを実行するとします。

```
aws codebuild list-builds --sort-order ASCENDING
```

次のような結果が出力に表示されることがあります。

```
{
  "nextToken": "4AEA6u7J...The full token has been omitted for brevity...MzY2OA==",
  "ids": [
    "codebuild-demo-project:815e755f-bade-4a7e-80f0-efe51EXAMPLE"
    "codebuild-demo-project:84a7f3d1-d40e-4956-b4cf-7a9d4EXAMPLE"
    ... The full list of build IDs has been omitted for brevity ...
    "codebuild-demo-project:931d0b72-bf6f-4040-a472-5c707EXAMPLE"
  ]
}
```

このコマンドをもう一度実行します。

```
aws codebuild list-builds --sort-order ASCENDING --next-token 4AEA6u7J...The full
 token has been omitted for brevity...MzY2OA==
```

次のような結果が出力に表示されることがあります。

```
{
  "ids": [
    "codebuild-demo-project:49015049-21cf-4b50-9708-df115EXAMPLE",
    "codebuild-demo-project:543e7206-68a3-46d6-a4da-759abEXAMPLE",
    ... The full list of build IDs has been omitted for brevity ...
    "codebuild-demo-project:c282f198-4582-4b38-bdc0-26f96EXAMPLE"
  ]
}
```

バッチビルド ID のリストを表示する (AWS CLI)

CodeBuild で AWS CLI を使用する方法については、「コマンドラインリファレンス」を参照してください。

- list-build-batches コマンドを実行します。

```
aws codebuild list-build-batches --sort-order sort-order --next-token next-token
```

上記のコマンドで、次のプレースホルダを置き換えます。

- *sort-order*: バッチビルド ID の一覧表示方法を示すのに使用するオプションの文字列。有効な値は、ASCENDING および DESCENDING です。
- *next-token*: オプションの文字列。以前の実行中に、リストに 100 を超える項目がある場合、最初の 100 項目だけが、next token と呼ばれる一意の文字列と共に返されます。リスト内の項目の次のバッチを取得するには、次のコマンドを再度実行し、次のトークンを呼び出しに追加します。リスト内のすべての項目を取得するには、次のトークンが返されなくなるまで、このコマンドを、以後のすべての次のトークンで実行し続けます。

たとえば、次のコマンドを実行するとします。

```
aws codebuild list-build-batches --sort-order ASCENDING
```

次のような結果が出力に表示されることがあります。

```
{
  "nextToken": "4AEA6u7J...The full token has been omitted for brevity...MzY2OA==",
  "ids": [
    "codebuild-demo-project:815e755f-bade-4a7e-80f0-efe51EXAMPLE"
    "codebuild-demo-project:84a7f3d1-d40e-4956-b4cf-7a9d4EXAMPLE"
    ... The full list of build IDs has been omitted for brevity ...
    "codebuild-demo-project:931d0b72-bf6f-4040-a472-5c707EXAMPLE"
  ]
}
```

このコマンドをもう一度実行します。

```
aws codebuild list-build-batches --sort-order ASCENDING --next-token 4AEA6u7J...The
 full token has been omitted for brevity...MzY2OA==
```

次のような結果が出力に表示されることがあります。

```
{
  "ids": [
    "codebuild-demo-project:49015049-21cf-4b50-9708-df115EXAMPLE",
    "codebuild-demo-project:543e7206-68a3-46d6-a4da-759abEXAMPLE",
    ... The full list of build IDs has been omitted for brevity ...
    "codebuild-demo-project:c282f198-4582-4b38-bdc0-26f96EXAMPLE"
  ]
}
```

ビルド ID の一覧表示 (AWS SDK)

CodeBuild を AWS SDK と組み合わせて使用する方法については、「AWS SDK とツールのリファレンス」を参照してください。

AWS CodeBuild でビルドプロジェクトのビルド ID を一覧表示する

AWS CodeBuild コンソール、AWS CLI、または AWS SDK を使用して、CodeBuild でビルドプロジェクトのビルド ID のリストを表示できます。

トピック

- ビルドプロジェクトのビルド ID を一覧表示する (コンソール)
- ビルドプロジェクトのビルド ID を一覧表示する (AWS CLI)
- ビルドプロジェクトのバッチビルド ID を一覧表示する (AWS CLI)
- ビルドプロジェクトのビルド ID を一覧表示する (AWS SDK)

ビルドプロジェクトのビルド ID を一覧表示する (コンソール)

1. CodeBuild コンソール (https://console.aws.amazon.com/codebuild/) を開きます。
2. ナビゲーションペインで、[Build projects] を選択します。ビルドプロジェクトのリストの [プロジェクト] 列で、ビルドプロジェクトを選択します。

> **Note**
>
> デフォルトでは、最新の 100 個のビルドまたはビルドプロジェクトのみが表示されます。さらに多くのビルドまたはビルドプロジェクトを表示するには、歯車アイコンを選択してから [ページ毎ビルド数] または [Projects per page (ページ毎プロジェクト数)] で別の値を選択するか、前後の矢印を使用します。

ビルドプロジェクトのビルド ID を一覧表示する (AWS CLI)

AWS CLI を AWS CodeBuild と組み合わせて使用する方法については、「コマンドラインリファレンス」を参照してください。

次のように list-builds-for-project コマンドを実行します。

```
aws codebuild list-builds-for-project --project-name project-name --sort-order sort-order --next-token next-token
```

上記のコマンドで、次のプレースホルダを置き換えます。

- *project-name*: ビルド ID を一覧表示するビルドプロジェクトの名前を示すのに必要な文字列。ビルドプロジェクトのリストを表示するには、「ビルドプロジェクト名の一覧表示 (AWS CLI)」を参照してください。
- *sort-order*: ビルド ID の一覧表示方法を示すのに使用するオプションの文字列。有効な値は、ASCENDING および DESCENDING です。

- *next-token*: オプションの文字列。以前の実行中に、リストに 100 を超える項目がある場合、最初の 100 項目だけが、next token と呼ばれる一意の文字列と共に返されます。リスト内の項目の次のバッチを取得するには、次のコマンドを再度実行し、次のトークンを呼び出しに追加します。リスト内のすべての項目を取得するには、次のトークンが返されなくなるまで、次に続くトークンが返されるごとにこのコマンドを実行し続けます。

たとえば、このコマンドを次のように実行するとします。

```
aws codebuild list-builds-for-project --project-name codebuild-demo-project --sort-
order ASCENDING
```

次のような結果が出力に表示されます。

```
{
  "nextToken": "4AEA6u7J...The full token has been omitted for brevity...MzY2OA==",
  "ids": [
    "codebuild-demo-project:9b175d16-66fd-4e71-93a0-50a08EXAMPLE"
    "codebuild-demo-project:a9d1bd09-18a2-456b-8a36-7d65aEXAMPLE"
    ... The full list of build IDs has been omitted for brevity ...
    "codebuild-demo-project:fe70d102-c04f-421a-9cfa-2dc15EXAMPLE"
  ]
}
```

このコマンドをもう一度実行します。

```
aws codebuild list-builds-for-project --project-name codebuild-demo-project --
sort-order ASCENDING --next-token 4AEA6u7J...The full token has been omitted for
 brevity...MzY2OA==
```

次のような結果が出力に表示されます。

```
{
  "ids": [
    "codebuild-demo-project:98253670-7a8a-4546-b908-dc890EXAMPLE"
    "codebuild-demo-project:ad5405b2-1ab3-44df-ae2d-fba84EXAMPLE"
    ... The full list of build IDs has been omitted for brevity ...
    "codebuild-demo-project:f721a282-380f-4b08-850a-e0ac1EXAMPLE"
  ]
}
```

ビルドプロジェクトのバッチビルド ID を一覧表示する (AWS CLI)

AWS CLI を AWS CodeBuild と組み合わせて使用する方法については、「コマンドラインリファレンス」を参照してください。

次のように list-build-batches-for-project コマンドを実行します。

```
aws codebuild list-build-batches-for-project --project-name project-name --sort-
order sort-order --next-token next-token
```

上記のコマンドで、次のプレースホルダを置き換えます。

- *project-name*: ビルド ID を一覧表示するビルドプロジェクトの名前を示すのに必要な文字列。ビルドプロジェクトのリストを表示するには、「ビルドプロジェクト名の一覧表示 (AWS CLI)」を参照してください。
- *sort-order*: ビルド ID の一覧表示方法を示すのに使用するオプションの文字列。有効な値は、ASCENDING および DESCENDING です。
- *next-token*: オプションの文字列。以前の実行中に、リストに 100 を超える項目がある場合、最初の 100 項目だけが、next token と呼ばれる一意の文字列と共に返されます。リスト内の項目の次のバッチを取得するには、次のコマンドを再度実行し、次のトークンを呼び出しに追加します。リスト内のすべての項目を取得するには、次のトークンが返されなくなるまで、次に続くトークンが返されるごとにこのコマンドを実行し続けます。

たとえば、このコマンドを次のように実行するとします。

```
aws codebuild list-build-batches-for-project --project-name codebuild-demo-project --
sort-order ASCENDING
```

次のような結果が出力に表示されます。

```
{
  "nextToken": "4AEA6u7J...The full token has been omitted for brevity...MzY2OA==",
  "ids": [
    "codebuild-demo-project:9b175d16-66fd-4e71-93a0-50a08EXAMPLE"
    "codebuild-demo-project:a9d1bd09-18a2-456b-8a36-7d65aEXAMPLE"
    ... The full list of build IDs has been omitted for brevity ...
    "codebuild-demo-project:fe70d102-c04f-421a-9cfa-2dc15EXAMPLE"
  ]
```

```
}
```

このコマンドをもう一度実行します。

```
aws codebuild list-build-batches-for-project --project-name codebuild-demo-project
 --sort-order ASCENDING --next-token 4AEA6u7J...The full token has been omitted for
 brevity...MzY2OA==
```

次のような結果が出力に表示されます。

```
{
  "ids": [
    "codebuild-demo-project:98253670-7a8a-4546-b908-dc890EXAMPLE"
    "codebuild-demo-project:ad5405b2-1ab3-44df-ae2d-fba84EXAMPLE"
    ... The full list of build IDs has been omitted for brevity ...
    "codebuild-demo-project:f721a282-380f-4b08-850a-e0ac1EXAMPLE"
  ]
}
```

ビルドプロジェクトのビルド ID を一覧表示する (AWS SDK)

AWS CodeBuild を AWS SDK と組み合わせて使用する方法については、「AWS SDK とツールのリファレンス」を参照してください。

AWS CodeBuild でのビルドの停止

AWS CodeBuild でビルドを停止するには、AWS CLI コンソール、AWS、または AWS CodeBuild SDK を使用します。

トピック

- ビルドの停止 (コンソール)
- ビルドの停止 (AWS CLI)
- ビルドの停止 (AWS SDK)

ビルドの停止 (コンソール)

1. AWS CodeBuild コンソール (https://console.aws.amazon.com/codesuite/codebuild/home) を開きます。

2. 次のいずれかを行ってください。

   - [*build-project-name*:*build-ID*] ページが表示された場合は、[ビルドの停止] を選択します。
   - ナビゲーションペインで、[Build history] を選択します。ビルドのリストで、ビルドのボックスを選択後、[ビルドの停止] を選択します。
   - ナビゲーションペインで、[Build projects] を選択します。ビルドプロジェクトのリストの [名前] 列で、ビルドプロジェクト名のリンクを選択します。ビルドのリストで、ビルドのボックスを選択後、[ビルドの停止] を選択します。

> **Note**
>
> デフォルトでは、最新の 100 個のビルドまたはビルドプロジェクトのみが表示されます。さらに多くのビルドまたはビルドプロジェクトを表示するには、歯車アイコンを選択してから [ページ毎ビルド数] または [Projects per page (ページ毎プロジェクト数)] で別の値を選択するか、前後の矢印を使用します。
> AWS CodeBuild でビルドを正常に停止できない場合 (ビルドプロセスが完了済みである場合など) は、[停止] ボタンが使用できないか、表示されないことがあります。

ビルドの停止 (AWS CLI)

- stop-build コマンドを実行します。

```
aws codebuild stop-build --id id
```

上記のコマンドで、次のプレースホルダを置き換えます。

- *id*: 必須の文字列。停止するビルドの ID。ビルド ID のリストを取得するには、次のトピックを参照してください。
  - ビルド ID の一覧表示 (AWS CLI)
  - ビルドプロジェクトのビルド ID を一覧表示する (AWS CLI)

AWS CodeBuild がビルドを正常に停止した場合、出力で `buildStatus` オブジェクトの `build` 値が `STOPPED` になります。

CodeBuild がビルドを正常に停止できない場合 (たとえば、ビルドがすでに完了している場合)、`build` オブジェクトの出力の オブジェクトの `buildStatus` 値が最終的なビルドステータス (例: `SUCCEEDED`) になります。

ビルドの停止 (AWS SDK)

AWS CodeBuild を AWS SDK と組み合わせて使用する方法については、「AWS SDK とツールのリファレンス」を参照してください。

AWS CodeBuild でバッチビルドを停止する

AWS CodeBuild でバッチビルドを停止するには、AWS CodeBuild コンソール、AWS CLI、または AWS SDK を使用します。

トピック

- バッチビルドの停止 (コンソール)
- バッチビルドを停止する (AWS CLI)
- ビルドを停止する (AWS SDK)

バッチビルドの停止 (コンソール)

1. AWS CodeBuild コンソール (https://console.aws.amazon.com/codesuite/codebuild/home) を開きます。
2. 次のいずれかを行ってください。
   - [***build-project-name*:*build-ID***] ページが表示された場合は、[ビルドの停止] を選択します。
   - ナビゲーションペインで、[Build history] を選択します。ビルドのリストで、ビルドのボックスを選択後、[ビルドの停止] を選択します。
   - ナビゲーションペインで、[Build projects] を選択します。ビルドプロジェクトのリストの [名前] 列で、ビルドプロジェクト名のリンクを選択します。ビルドのリストで、ビルドのボックスを選択後、[ビルドの停止] を選択します。

> **Note**
>
> デフォルトでは、最新の 100 個のビルドまたはビルドプロジェクトのみが表示されます。さらに多くのビルドまたはビルドプロジェクトを表示するには、歯車アイコンを選択してから [ページ毎ビルド数] または [Projects per page (ページ毎プロジェクト数)] で別の値を選択するか、前後の矢印を使用します。
> AWS CodeBuild でバッチビルドを正常に停止できない場合 (ビルドプロセスが完了済みである場合など) は、[停止] ボタンを使用できません。

バッチビルドを停止する (AWS CLI)

- `stop-build-batch` コマンドを実行します。

```
aws codebuild stop-build-batch --id <batch-build-id>
```

上記のコマンドで、次のプレースホルダを置き換えます。

- *<batch-build-id>*: 必須の文字列。停止するバッチビルドの ID。バッチビルド ID のリストを取得するには、次のトピックを参照してください。
  - バッチビルド ID のリストを表示する (AWS CLI)
  - ビルドプロジェクトのバッチビルド ID を一覧表示する (AWS CLI)

AWS CodeBuild がバッチビルドを正常に停止した場合、出力で `buildBatch` オブジェクトの `buildBatchStatus` 値が `STOPPED` になります。

CodeBuild がビルドを正常に停止できない場合 (たとえば、ビルドがすでに完了している場合)、`buildBatch` オブジェクトの出力の オブジェクトの `buildBatchStatus` 値が最終的なビルドステータス (例: `SUCCEEDED`) になります。

ビルドを停止する (AWS SDK)

AWS CodeBuild を AWS SDK と組み合わせて使用する方法については、「AWS SDK とツールのリファレンス」を参照してください。

AWS CodeBuild でビルドを再試行する

AWS CodeBuild コンソール、AWS CLI、または AWS を使用して、AWS CodeBuild で単一のビルドまたはバッチビルドを再試行できます。

トピック

ビルドを再試行する (コンソール)

1. AWS CodeBuild コンソール (https://console.aws.amazon.com/codesuite/codebuild/home) を開きます。
2. 次のいずれかを行ってください。
   - [***build-project-name*:*build-ID***] ページが表示された場合は、[Retry build] を選択します。
   - ナビゲーションペインで、[Build history] を選択します。ビルドのリストで、ビルドのボックスを選択後、[Retry build] を選択します。
   - ナビゲーションペインで、[Build projects] を選択します。ビルドプロジェクトのリストの [名前] 列で、ビルドプロジェクト名のリンクを選択します。ビルドのリストで、ビルドのボックスを選択後、[Retry build] を選択します。

> **Note**
>
> デフォルトでは、最新の 100 個のビルドまたはビルドプロジェクトのみが表示されます。さらに多くのビルドまたはビルドプロジェクトを表示するには、歯車アイコンを選択してから [ページ毎ビルド数] または [Projects per page (ページ毎プロジェクト数)] で別の値を選択するか、前後の矢印を使用します。

ビルドを再試行 (AWS CLI)

- retry-build コマンドを実行します。

```
aws codebuild retry-build --id <build-id> --idempotency-token <idempotencyToken>
```

上記のコマンドで、次のプレースホルダを置き換えます。

- *<build-id>*: 必須の文字列。再試行するビルドまたはバッチビルドの ID。ビルド ID のリストを取得するには、次のトピックを参照してください。
  - ビルド ID の一覧表示 (AWS CLI)
  - バッチビルド ID のリストを表示する (AWS CLI)
  - ビルドプロジェクトのビルド ID を一覧表示する (AWS CLI)
  - ビルドプロジェクトのバッチビルド ID を一覧表示する (AWS CLI)
- `--idempotency-token`: オプション。retry-build オプションを指定して コマンドを実行すると、大文字と小文字を区別する一意の識別子 (トークン) が `retry-build` リクエストに含まれます。このトークンは、 リクエスト後 5 分間有効です。同じトークンで `retry-build` リクエストを繰り返し行い、パラメータを変更すると、CodeBuild はパラメータの不一致エラーを返します。

ビルドを再試行 (AWS SDK)

AWS CodeBuild を AWS SDK と組み合わせて使用する方法については、「AWS SDK とツールのリファレンス」を参照してください。

セッションマネージャーで実行中のビルドを表示する

では AWS CodeBuild、実行中のビルドを一時停止し、 AWS Systems Manager Session Manager を使用してビルドコンテナに接続し、コンテナの状態を表示できます。

> **Note**
>
> この機能は、Windows 環境では使用できません。

トピック

- 前提条件
- ビルドの一時停止
- ビルドを開始します

- ビルドコンテナに接続する
- ビルドを再開する

前提条件

ビルドセッションでセッションマネージャーを使用できるようにするには、ビルドのセッション接続を有効にする必要があります。次の 2 つの前提条件があります。

- CodeBuild Linux の標準キュレートイメージでは、SSM エージェントは既にインストールされており、SSM エージェント ContainerMode が有効になっています。

  ビルドにカスタムイメージを使用している場合は、次の操作を行います。

  1. SSM Agent をインストールします。詳細については、 AWS Systems Manager ユーザーガイドの「Linux 用 EC2 インスタンスに SSM Agent を手動でインストールする」を参照してください。SSM Agent は、バージョン 3.0.1295.0 以降である必要があります。
  2. ファイル https://github.com/aws/aws-codebuild-docker-images/blob/master/ubuntu/standard/5.0/amazon-ssm-agent.json をイメージ内の `/etc/amazon/ssm/` ディレクトリにコピーします。これにより、SSM エージェントでコンテナモードがイネーブルになります。

> **Note**
>
> この機能が正常に動作するには、カスタムイメージに最新の SSM エージェントが必要です。

- CodeBuild サービスロールには、次の SSM ポリシーが必要です。

```
{
  "Effect": "Allow",
  "Action": [
    "ssmmessages:CreateControlChannel",
    "ssmmessages:CreateDataChannel",
    "ssmmessages:OpenControlChannel",
    "ssmmessages:OpenDataChannel"
  ],
  "Resource": "*"
}
```

ビルドを開始するときに、 CodeBuild コンソールでこのポリシーをサービスロールに自動的にアタッチできます。または、このポリシーを手動でサービスロールにアタッチすることもできます。

- Systems Manager の設定でセッションアクティビティの監査と口グ記録が有効になっている場合、 CodeBuild サービスロールには追加のアクセス許可も必要です。アクセス許可は、ログが格納されている場所によって異なります。

  CloudWatch ログ

  CloudWatch Logs を使用してログを保存する場合は、 CodeBuild サービスロールに次のアクセス許可を追加します。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "logs:DescribeLogGroups",
      "Resource": "arn:aws:logs:<region-id>:<account-id>:log-group:*:*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogStream",
        "logs:PutLogEvents"
      ],
      "Resource": "arn:aws:logs:<region-id>:<account-id>:log-group:<log-group-
name>:*"
    }
  ]
}
```

  Amazon S3

  Amazon S3 を使用してログを保存する場合は、 CodeBuild サービスロールに次のアクセス許可を追加します。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
```

```
      "Action": [
        "s3:GetEncryptionConfiguration",
        "s3:PutObject"
      ],
      "Resource": [
        "arn:aws:s3:::<bucket-name>",
        "arn:aws:s3:::<bucket-name>/*"
      ]
    }
  ]
}
```

詳細については、AWS Systems Manager ユーザーガイドの「セッションアクティビティのログ記録と監査」を参照してください。

ビルドの一時停止

ビルドを一時停止するには、buildspec ファイルのビルドフェーズのいずれかで codebuild-breakpoint コマンドを実行します。この時点でビルドは一時停止されます。これにより、ビルドコンテナに接続し、コンテナを現在の状態で表示できます。

たとえば、buildspec ファイルのビルドフェーズに、以下を追加します。

```
phases:
  pre_build:
    commands:
      - echo Entered the pre_build phase...
      - echo "Hello World" > /tmp/hello-world
      - codebuild-breakpoint
```

このコードは、/tmp/hello-worldファイルを作成し、この時点でビルドを一時停止します。

ビルドを開始します

ビルドセッションでセッションマネージャーを使用できるようにするには、ビルドのセッション接続を有効にする必要があります。これを行うには、ビルドを開始するときに、以下の手順を実行します。

1. https://console.aws.amazon.com/codesuite/codebuild/home で AWS CodeBuild コンソールを開きます。

2. ナビゲーションペインで、[ビルドプロジェクト] を選択します。ビルドプロジェクトを選択した後、[Start build with overrides] を選択します。
3. [Advanced build overrides (高度なビルドの上書き)] を選択します。
4. [Environment] セクションで、[Enable session connection] オプションを選択します。このオプションが選択されていない場合、codebuild-breakpoint および codebuild-resume コマンドは無視されます。
5. その他の必要な変更を行い、[Start build] を選択します。
6. コンソールでビルドステータスを監視します。セッションが利用可能になると、AWS セッションマネージャーリンクが [Build status] セクションに表示されます。

ビルドコンテナに接続する

ビルドコンテナには、次の 2 つのいずれかに接続できます。

CodeBuild コンソール

ウェブブラウザで、AWS セッションマネージャーリンクをクリックして、ビルドコンテナに接続します。ターミナルセッションが開き、ビルドコンテナを表示して制御できます。

AWS CLI

> **Note**
>
> この手順を実行するには、ローカルマシンにセッションマネージャプラグインがインストールされている必要があります。詳細については、「 ユーザーガイド」の「 AWS CLI 用の Session Manager プラグインをインストールする」を参照してください。 AWS Systems Manager

1. batch-get-builds APIを呼び出し、ビルドIDに置き換えて、セッションターゲット識別子を含むビルドに関する情報を取得します。セッションターゲット識別子のプロパティ名は、aws コマンドの出力タイプによって異なります。これが、コマンドに `--output json` が追加される理由です。

```
aws codebuild batch-get-builds --ids <buildID> --region <region> --output json
```

2. プロパティの値 sessionTarget をコピーします。sessionTarget プロパティ名は、aws コマンドの出力タイプによって異なる場合があります。これが、前のステップでコマンドに --output json が追加される理由です。
3. ビルドコンテナに接続するには、次のコマンドを使用します。

```
aws ssm start-session --target <sessionTarget> --region <region>
```

この例では、/tmp/hello-worldファイルが存在し、Hello World テキストを含む検証です。

ビルドを再開する

ビルドコンテナーを調べ終わったら、codebuild-resume コマンドをコンテナーシェルから実行します。

```
$ codebuild-resume
```

AWS CodeBuild でのビルドの削除

AWS CLI でビルドを削除するには、AWS または AWS CodeBuild SDK を使用できます。

ビルドの削除 (AWS CLI)

batch-delete-builds コマンドを実行します。

```
aws codebuild batch-delete-builds --ids ids
```

上記のコマンドで、次のプレースホルダを置き換えます。

- *ids*: 必須の文字列。削除するビルドの ID。複数のビルドを指定するには、各ビルド ID をスペースで区切ります。ビルド ID のリストを取得するには、次のトピックを参照してください。
  - ビルド ID の一覧表示 (AWS CLI)
  - ビルドプロジェクトのビルド ID を一覧表示する (AWS CLI)

成功すると、buildsDeleted 配列が出力に表示されます。この配列には、正常に削除された各ビルドの Amazon リソースネーム (ARN) が含まれています。正常に削除されなかったビルドに関する情報は、出力の buildsNotDeleted 配列内に表示されます。

たとえば、次のコマンドを実行するとします。

```
aws codebuild batch-delete-builds --ids my-demo-build-project:f8b888d2-5e1e-4032-8645-
b115195648EX my-other-demo-build-project:a18bc6ee-e499-4887-b36a-8c90349c7eEX
```

次のような情報が出力に表示されます。

```
{
  "buildsNotDeleted": [
    {
      "id": "arn:aws:codebuild:us-west-2:123456789012:build/my-demo-build-
project:f8b888d2-5e1e-4032-8645-b115195648EX",
      "statusCode": "BUILD_IN_PROGRESS"
    }
  ],
  "buildsDeleted": [
    "arn:aws:codebuild:us-west-2:123456789012:build/my-other-demo-build-
project:a18bc6ee-e499-4887-b36a-8c90349c7eEX"
  ]
}
```

ビルドの削除 (AWS SDK)

AWS CodeBuild を AWS SDK と組み合わせて使用する方法については、「AWS SDK とツールのリファレンス」を参照してください。

での AWS Lambda コンピューティングの使用 AWS CodeBuild

AWS Lambda コンピューティングは、ビルドの起動速度を最適化します。 は、起動レイテンシーが低いため、ビルドの高速化 AWS Lambda をサポートします。 AWS Lambda また、 は自動的にスケールするため、ビルドはキュー内で実行を待つことはありません。ただし、 AWS Lambda がサポートしていないユースケースがいくつかあり、それらが影響する場合は EC2 コンピューティングを使用します。詳細については、「AWS Lambda コンピューティングの制限」を参照してください。

トピック

AWS Lambda上で実行される、選別されたランタイム環境の Docker イメージには、どのツールとランタイムが含まれますか?

AWS Lambda は、 AWS CLI v2、 AWS SAM CLI、git、go、Java、Node.js、Python、pip、Ruby、.NET の各ツールをサポートしています。

キュレーションされたイメージに必要なツールが含まれていない場合はどうなりますか ?

キュレーションされたイメージに必要なツールが含まれていない場合は、必要なツールを含むカスタム環境の Docker イメージを提供できます。

Lambda コンピューティングにカスタムイメージを使用するには、次の Amazon ECR アクセス許可が必要であることに注意してください。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ecr:GetAuthorizationToken"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "ecr:BatchCheckLayerAvailability",
        "ecr:GetDownloadUrlForLayer",
        "ecr:BatchGetImage"
      ],
      "Resource": "arn:aws:ecr:image-region:image-account-id:repository/image-repo"
    }
  ]
}
```

また、カスタムイメージを使用するには、 curlまたは をインストールwgetする必要があることに注意してください。

どのリージョンが の AWS Lambda コンピューティングをサポートしています CodeBuildか？

では CodeBuild、米国東部 (バージニア北部) 、 AWS リージョン米国東部 (オハイオ) 、米国西部 (オレゴン) 、アジアパシフィック (ムンバイ) 、アジアパシフィック (シンガポール) 、アジアパシフィック (シドニー) 、アジアパシフィック (東京) 、欧州 (フランクフルト) 、欧州 (アイルランド) 、南米 (サンパウロ) で AWS Lambda コンピューティングがサポートされています。 CodeBuild が利用可能な の場所の詳細については AWS リージョン 、AWS 「リージョン別の のサービス」を参照してください。

AWS Lambda コンピューティングの制限

がサポート AWS Lambda していないユースケースがいくつかあり、それらがユーザーに影響を与える場合は EC2 コンピューティングを使用します。

- AWS Lambda は、ルートアクセス許可を必要とするツールをサポートしていません。yum や rpm などのツールには、EC2 コンピューティングタイプや root 権限を必要としないその他のツールを使用してください。
- AWS Lambda は Docker ビルドまたは実行をサポートしていません。Podman など、root 権限を必要としない代替手段を使用できます。
- AWS Lambda は、 外のファイルへの書き込みをサポートしていません/tmp。付属のパッケージマネージャーは、パッケージのダウンロードと参照にデフォルトで /tmp ディレクトリを使用するように設定されています。
- AWS Lambda は 環境タイプをサポートしておらずLINUX_GPU_CONTAINER、Windows Server Core 2019 ではサポートされていません。
- AWS Lambda は、キャッシュ、バッチビルド、カスタムビルドタイムアウト、キュータイムアウト、ビルドバッジ、特権モード、カスタムランタイム環境、または 15 分以上のランタイムをサポートしていません。
- AWS Lambda は、VPC 接続、 CodeBuild ソース IP アドレスの固定範囲、EFS 、セマンティックバージョニング、証明書のインストール、または Session Manager による SSH アクセスをサポートしていません。

AWS Lambda を使用した コンピューティングサンプル AWS CodeBuild

これらのサンプルグループは、 で AWS Lambda コンピューティングを試すために使用できます CodeBuild。

トピック

- Lambda Java AWS SAMで を使用して CodeBuild Lambda 関数をデプロイする
- CodeBuild Lambda Node.js で単一ページの React アプリを作成する
- Lambda Python を使用して CodeBuild Lambda 関数設定を更新する

Lambda Java AWS SAMで を使用して CodeBuild Lambda 関数をデプロイする

AWS Serverless Application Model (AWS SAM) は、サーバーレスアプリケーションを構築するためのオープンソースのフレームワークです。詳細については、「」の「 AWS Serverless Application Modelリポジトリ」を参照してください GitHub。次の Java サンプルでは、Gradle を使

用して AWS Lambda 関数をビルドおよびテストします。その後、 CLI AWS SAM を使用してAWS CloudFormationテンプレートとデプロイバンドルをデプロイします。 CodeBuild Lambda を使用すると、構築、テスト、デプロイの各ステップが自動的に処理されるため、1 つの構築に手動で介入することなくインフラストラクチャをすばやく更新できます。

AWS SAM リポジトリをセットアップする

CLI を使用して AWS SAM AWS SAMHello Worldプロジェクトを作成します。

AWS SAM プロジェクトを作成するには

1. ローカルマシンに AWS SAM CLI をインストールするには、「 AWS Serverless Application Modelデベロッパーガイド」の手順に従います。
2. sam init を実行して、次のプロジェクト設定を選択します。

```
Which template source would you like to use?: 1 - AWS Quick Start Templates
Choose an AWS Quick Start application template: 1 - Hello World Example
Use the most popular runtime and package type? (Python and zip) [y/N]: N
Which runtime would you like to use?: 8 - java21
What package type would you like to use?: 1 - Zip
Which dependency manager would you like to use?: 1 - gradle
Would you like to enable X-Ray tracing on the function(s) in your application? [y/
N]: N
Would you like to enable monitoring using CloudWatch Application Insights? [y/N]: N
Would you like to set Structured Logging in JSON format on your Lambda functions?
 [y/N]:  N
Project name [sam-app]: <insert project name>
```

3. サポートされているソースリポジトリにAWS SAMプロジェクトフォルダをアップロードします。サポートされているソースタイプのリストについては、「」を参照してくださいProjectSource。

CodeBuild Lambda Java プロジェクトを作成する

AWS CodeBuild Lambda Java プロジェクトを作成し、ビルドに必要な IAM アクセス許可を設定します。

CodeBuild Lambda Java プロジェクトを作成するには

1. AWS CodeBuild コンソール (https://console.aws.amazon.com/codesuite/codebuild/home) を開きます。

2. CodeBuild 情報ページが表示される場合は、ビルドプロジェクトの作成を選択します。それ以外の場合は、ナビゲーションペインでビルドを展開し、[ビルドプロジェクト] を選択し、次に [Create build project (ビルドプロジェクトの作成)] を選択します。
3. [プロジェクト名] に、このビルドプロジェクトの名前を入力します。ビルドプロジェクトの名前は、各 AWS アカウントで一意である必要があります。また、他のユーザーがこのプロジェクトの使用目的を理解できるように、ビルドプロジェクトの説明を任意で指定することもできます。
4. ソース で、AWS SAMプロジェクトがあるソースリポジトリを選択します。
5. [環境] で以下の操作を行います。
   - コンピューティング で、Lambda を選択します。
   - Runtime(s) で、Java を選択します。
   - イメージ で、aws/codebuild/amazonlinux-x86_64-lambda-standard :corretto21 を選択します。
   - サービスロール では、新しいサービスロールを選択したままにします。ロール名 を書き留めます。これは、このサンプルの後半でプロジェクトの IAM アクセス許可を更新するときに必要です。
6. Create build project (ビルドプロジェクトの作成)を選択します。
7. IAM コンソール (https://console.aws.amazon.com/iam/) を開きます。
8. ナビゲーションペインで、ロールを選択し、プロジェクトに関連付けられたサービスロールを選択します。でプロジェクトロールを検索するには、ビルドプロジェクト CodeBuild を選択し、編集、環境、サービスロールの順に選択します。
9. [信頼関係] タブを選択し、続いて [信頼ポリシーの編集] を選択します。
10. IAM ロールに次のインラインポリシーを追加します。これは、後でAWS SAMインフラストラクチャをデプロイするために使用されます。詳細については、「 IAM ユーザーガイド」の「IAM ID アクセス許可の追加および削除」を参照してください。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Action": [
                "cloudformation:*",
                "lambda:*",
                "iam:*",
```

```
                "apigateway:*",
                "s3:*"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

プロジェクトの buildspec をセットアップする

Lambda 関数を構築、テスト、デプロイするために、 は buildspec からビルドコマンド CodeBuild を読み取って実行します。

プロジェクトの buildspec を設定するには

1. CodeBuild コンソールでビルドプロジェクトを選択し、編集 と Buildspec を選択します。
2. Buildspec でビルドコマンドを挿入を選択し、エディタ に切り替えます。
3. 事前に入力されたビルドコマンドを削除し、次の buildspec に貼り付けます。

```
version: 0.2
env:
  variables:
    GRADLE_DIR: "HelloWorldFunction"
phases:
  build:
    commands:
      - echo "Running unit tests..."
      - cd $GRADLE_DIR; gradle test; cd ..
      - echo "Running build..."
      - sam build --template-file template.yaml
      - echo "Running deploy..."
      - sam package --output-template-file packaged.yaml --resolve-s3 --template-
file template.yaml
      - yes | sam deploy
```

4. [Update buildspec (buildspec の更新)] を選択します。

AWS SAM Lambda インフラストラクチャをデプロイする

CodeBuild Lambda を使用して Lambda インフラストラクチャを自動的にデプロイする

Lambda インフラストラクチャをデプロイするには

1. [Start build] を選択します。これにより、 AWS Lambdaを使用してAWS SAMアプリケーションが自動的に構築、テスト、デプロイされますAWS CloudFormation。
2. ビルドが完了したら、 AWS Lambdaコンソールに移動し、AWS SAMプロジェクト名で新しい Lambda 関数を検索します。
3. 関数の概要で API Gateway を選択し、API エンドポイント URL をクリックして、Lambda 関数をテストします。メッセージ を含むページが開きます`"message": "hello world"`。

インフラストラクチャのクリーンアップ

このチュートリアルで使用したリソースに対して追加料金が発生しないようにするには、 AWS SAM テンプレートと で作成したリソースを削除します CodeBuild。

インフラストラクチャをクリーンアップするには

1. AWS CloudFormation コンソールに移動し、 を選択します`aws-sam-cli-managed-default`。
2. リソース で、デプロイバケット を空にします`SamCliSourceBucket`。
3. `aws-sam-cli-managed-default` スタックを削除します。
4. AWS SAM プロジェクトに関連付けられている AWS CloudFormation スタックを削除します。このスタックの名前はAWS SAMプロジェクトと同じである必要があります。
5. CloudWatch コンソールに移動し、 CodeBuild プロジェクトに関連付けられている CloudWatch ロググループを削除します。
6. CodeBuild コンソールに移動し、ビルド CodeBuild プロジェクトの削除 を選択してプロジェクトを削除します。

CodeBuild Lambda Node.js で単一ページの React アプリを作成する

React アプリケーションの作成は、単一ページの React アプリケーションを作成する方法です。次の Node.js サンプルでは、Node.js を使用して Create React App からソースアーティファクトを構築し、ビルドアーティファクトを返します。

ソースリポジトリとアーティファクトバケットをセットアップする

YARN を使用してプロジェクトのソースリポジトリを作成し、React アプリを作成します。

ソースリポジトリとアーティファクトバケットを設定するには

1. ローカルマシンで、 `yarn create react-app <app-name>`を実行してシンプルな React アプリを作成します。
2. React アプリケーションプロジェクトフォルダを、サポートされているソースリポジトリにアップロードします。サポートされているソースタイプのリストについては、「」を参照してくださいProjectSource。

CodeBuild Lambda Node.js プロジェクトを作成する

AWS CodeBuild Lambda Node.js プロジェクトを作成します。

CodeBuild Lambda Node.js プロジェクトを作成するには

1. AWS CodeBuild コンソール (https://console.aws.amazon.com/codesuite/codebuild/home) を開きます。
2. CodeBuild 情報ページが表示されたら、ビルドプロジェクトの作成を選択します。それ以外の場合は、ナビゲーションペインでビルドを展開し、[ビルドプロジェクト] を選択し、次に [Create build project (ビルドプロジェクトの作成)] を選択します。
3. [プロジェクト名] に、このビルドプロジェクトの名前を入力します。ビルドプロジェクトの名前は、各 AWS アカウントで一意である必要があります。また、他のユーザーがこのプロジェクトの使用目的を理解できるように、ビルドプロジェクトの説明を任意で指定することもできます。
4. ソース で、AWS SAMプロジェクトがあるソースリポジトリを選択します。
5. [環境] で以下の操作を行います。
   - コンピューティング で、Lambda を選択します。
   - Runtime(s) で、Node.js を選択します。
   - イメージ で、aws/codebuild/amazonlinux-x86_64-lambda-standard :nodejs20 を選択します。
6. [アーティファクト] で、次のようにします。
   - タイプ で、Amazon S3 を選択します。
   - バケット名 で、前に作成したプロジェクトアーティファクトバケットを選択します。
   - アーティファクトパッケージ で、Zip を選択します。

7. Create build project (ビルドプロジェクトの作成)を選択します。

プロジェクトの buildspec をセットアップする

React アプリケーションを構築するために、 は buildspec ファイルからビルドコマンド CodeBuild を読み取り、実行します。

プロジェクトの buildspec を設定するには

1. CodeBuild コンソールでビルドプロジェクトを選択し、編集 と Buildspec を選択します。
2. Buildspec でビルドコマンドを挿入を選択し、エディタ に切り替えます。
3. 事前に入力されたビルドコマンドを削除し、次の buildspec に貼り付けます。

```
version: 0.2
phases:
  build:
    commands:
      - yarn
      - yarn add --dev jest-junit @babel/plugin-proposal-private-property-in-object
      - yarn run build
      - yarn run test -- --coverage --watchAll=false --testResultsProcessor="jest-
junit" --detectOpenHandles
artifacts:
  name: "build-output"
  files:
    - "**/*"
reports:
  test-report:
    files:
      - 'junit.xml'
    file-format: 'JUNITXML'
  coverage-report:
    files:
      - 'coverage/clover.xml'
    file-format: 'CLOVERXML'
```

4. [Update buildspec (buildspec の更新)] を選択します。

React アプリを構築して実行する

CodeBuild Lambda で React アプリを構築し、ビルドアーティファクトをダウンロードして、React アプリをローカルで実行します。

React アプリを構築して実行するには

1. [Start build] を選択します。
2. ビルドが完了したら、Amazon S3 プロジェクトアーティファクトバケットに移動し、React アプリケーションアーティファクトをダウンロードします。
3. React ビルドアーティファクト と をプロジェクトフォルダ`run npm install -g serve && serve -s build`に解凍します。
4. `serve` コマンドは、ローカルポートで静的サイトを提供し、出力をターミナルに出力します。ターミナル出力の `Local:` にある localhost URL にアクセスして、React アプリを表示できます。

React ベースのサーバーのデプロイを処理する方法の詳細については、「Create React App Deployment」を参照してください。

インフラストラクチャのクリーンアップ

このチュートリアルで使用したリソースに対する追加料金が発生しないようにするには、 CodeBuild プロジェクト用に作成されたリソースを削除します。

インフラストラクチャをクリーンアップするには

1. プロジェクトアーティファクト Amazon S3 バケットを削除する
2. CloudWatch コンソールに移動し、 CodeBuild プロジェクトに関連付けられている CloudWatch ロググループを削除します。
3. CodeBuild コンソールに移動し、ビルド CodeBuild プロジェクトの削除 を選択してプロジェクトを削除します。

Lambda Python を使用して CodeBuild Lambda 関数設定を更新する

次の Python サンプルでは、Boto3 と CodeBuild Lambda Python を使用して Lambda 関数の設定を更新します。このサンプルは、他の AWS リソースをプログラムで管理するように拡張できます。詳細については、「Boto3 ドキュメント」を参照してください。

前提条件

アカウントで Lambda 関数を作成または検索します。

このサンプルは、アカウントに Lambda 関数が既に作成されていることを前提としており、CodeBuild を使用して Lambda 関数の環境変数を更新します。による Lambda 関数の設定の詳細については、 Lambda Java AWS SAMで を使用して CodeBuild Lambda 関数をデプロイする サンプルを参照するか CodeBuild、「」を参照してくださいAWS Lambda。

ソースリポジトリをセットアップする

Boto3 Python スクリプトを保存するソースリポジトリを作成します。

ソースリポジトリを設定するには

1. 次の Python スクリプトを という新しいファイルにコピーしますupdate_lambda_environment_variables.py。

```
import boto3
from os import environ


def update_lambda_env_variable(lambda_client):
    lambda_function_name = environ['LAMBDA_FUNC_NAME']
    lambda_env_variable = environ['LAMBDA_ENV_VARIABLE']
    lambda_env_variable_value = environ['LAMBDA_ENV_VARIABLE_VALUE']
    print("Updating lambda function " + lambda_function_name + " environment
 variable "
          + lambda_env_variable + " to " + lambda_env_variable_value)
    lambda_client.update_function_configuration(
        FunctionName=lambda_function_name,
        Environment={
            'Variables': {
                lambda_env_variable: lambda_env_variable_value
            }
        },
    )


if __name__ == "__main__":
    region = environ['AWS_REGION']
    client = boto3.client('lambda', region)
```

```
    update_lambda_env_variable(client)
```

2. サポートされているソースリポジトリに Python ファイルをアップロードします。サポートされているソースタイプのリストについては、「」を参照してくださいProjectSource。

CodeBuild Lambda Python プロジェクトを作成する

CodeBuild Lambda Python プロジェクトを作成します。

CodeBuild Lambda Java プロジェクトを作成するには

1. AWS CodeBuild コンソール (https://console.aws.amazon.com/codesuite/codebuild/home) を開きます。
2. CodeBuild 情報ページが表示される場合は、ビルドプロジェクトの作成を選択します。それ以外の場合は、ナビゲーションペインでビルドを展開し、[ビルドプロジェクト] を選択し、次に [Create build project (ビルドプロジェクトの作成)] を選択します。
3. [プロジェクト名] に、このビルドプロジェクトの名前を入力します。ビルドプロジェクトの名前は、各 AWS アカウントで一意である必要があります。また、他のユーザーがこのプロジェクトの使用目的を理解できるように、ビルドプロジェクトの説明を任意で指定することもできます。
4. ソース で、AWS SAMプロジェクトがあるソースリポジトリを選択します。
5. [環境] で以下の操作を行います。
   - コンピューティング で、Lambda を選択します。
   - Runtime(s) で、Python を選択します。
   - イメージ で、aws/codebuild/amazonlinux-x86_64-lambda-standard :python3.12 を選択します。
   - サービスロール では、新しいサービスロールを選択したままにします。ロール名 を書き留めます。これは、このサンプルの後半でプロジェクトの IAM アクセス許可を更新するときに必要です。
6. Create build project (ビルドプロジェクトの作成)を選択します。
7. IAM コンソール (https://console.aws.amazon.com/iam/) を開きます。
8. ナビゲーションペインで、ロールを選択し、プロジェクトに関連付けられたサービスロールを選択します。でプロジェクトロールを検索するには、ビルドプロジェクト CodeBuild を選択し、編集、環境、サービスロールの順に選択します。
9. [信頼関係] タブを選択し、続いて [信頼ポリシーの編集] を選択します。

10. IAM ロールに次のインラインポリシーを追加します。これは、後でAWS SAMインフラストラクチャをデプロイするために使用されます。詳細については、「 IAM ユーザーガイド」の「IAM ID アクセス許可の追加および削除」を参照してください。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "UpdateLambdaPermissions",
            "Effect": "Allow",
            "Action": [
                "lambda:UpdateFunctionConfiguration"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

プロジェクトの buildspec をセットアップする

Lambda 関数を更新するために、スクリプトは buildspec から環境変数を読み取り、Lambda 関数の名前、環境変数名、環境変数の値を検索します。

プロジェクトの buildspec を設定するには

1. CodeBuild コンソールでビルドプロジェクトを選択し、編集 と Buildspec を選択します。
2. Buildspec でビルドコマンドを挿入を選択し、エディタ に切り替えます。
3. 事前に入力されたビルドコマンドを削除し、次の buildspec に貼り付けます。

```
version: 0.2
env:
  variables:
    LAMBDA_FUNC_NAME: "<lambda-function-name>"
    LAMBDA_ENV_VARIABLE: "FEATURE_ENABLED"
    LAMBDA_ENV_VARIABLE_VALUE: "true"
phases:
  install:
    commands:
```

```
      - pip3 install boto3
build:
  commands:
     - python3 update_lambda_environment_variables.py
```

4. [Update buildspec (buildspec の更新)] を選択します。

Lambda 設定を更新する

CodeBuild Lambda Python を使用して、Lambda 関数の設定を自動的に更新します。

Lambda 関数の設定を更新するには

1. [Start build] を選択します。
2. ビルドが完了したら、Lambda 関数に移動します。
3. 設定を選択し、次に環境変数を選択します。キー FEATURE_ENABLEDと値 を含む新しい環境変数が表示されますtrue。

インフラストラクチャのクリーンアップ

このチュートリアルで使用したリソースに対する追加料金が発生しないようにするには、 CodeBuild プロジェクト用に作成されたリソースを削除します。

インフラストラクチャをクリーンアップするには

1. CloudWatch コンソールに移動し、 CodeBuild プロジェクトに関連付けられている CloudWatch ロググループを削除します。
2. CodeBuild コンソールに移動し、ビルド CodeBuild プロジェクトの削除 を選択してプロジェクトを削除します。
3. このサンプルのために Lambda 関数を作成した場合は、アクション および 削除 関数を選択して Lambda 関数をクリーンアップします。

拡張子

このサンプルを拡張して AWS CodeBuild Lambda Python を使用して他のAWSリソースを管理する場合は、次の手順を実行します。

- Boto3 を使用して新しいリソースを変更するように Python スクリプトを更新します。

- CodeBuild プロジェクトに関連付けられた IAM ロールを更新して、新しいリソースに対するアクセス許可を付与します。
- 新しいリソースに関連付けられた新しい環境変数を buildspec に追加します。

でのリザーブドキャパシティの操作 AWS CodeBuild

CodeBuild は、次のコンピューティングフリートを提供します。

- オンデマンドフリート
- リザーブドキャパシティフリート

オンデマンドフリートでは、 はビルドのコンピューティング CodeBuild を提供します。マシンはビルドが終了すると破棄されます。オンデマンドフリートはフルマネージド型で、需要の急増にも対応できる自動スケーリング機能を備えています。

> **Note**
>
> オンデマンドフリートは Windows Server 2022 をサポートしていません。

CodeBuild は、 によって維持される Amazon EC2 を搭載したインスタンスを含むリザーブドキャパシティフリートも提供します CodeBuild。リザーブドキャパシティフリートでは、ビルド環境に合わせて専有インスタンスのセットを設定します。これらのマシンはアイドル状態のままで、ビルドやテストをすぐに処理できる状態になり、ビルド時間を短縮します。リザーブドキャパシティフリートでは、マシンは常に稼働しており、プロビジョニングされている間はコストが発生し続けます。

> **Important**
>
> インスタンスの実行期間に関係なく、リザーブドキャパシティフリートにはインスタンスごとに初期料金が発生し、その後は追加コストが発生する可能性があります。詳細については、「https://aws.amazon.com/codebuild/pricing/」を参照してください。

トピック

- リザーブドキャパシティフリートの制限

リザーブドキャパシティフリートの使用を開始するには

リザーブドキャパシティフリートを作成するには

1. にサインイン AWS Management Console し、https://console.aws.amazon.com/codesuite/codebuild/home で AWS CodeBuild コンソールを開きます。
2. ナビゲーションペインで、[コンピューティングフリート] を選択し、[コンピューティングフリートを作成] を選択します。
3. [コンピューティングフリート名] テキストフィールドに、フリートの名前を入力します。
4. [オペレーティングシステム] ドロップダウンメニューから、オペレーティングシステムを選択します。
5. [アーキテクチャ] ドロップダウンメニューから、アーキテクチャを選択します。
6. [コンピューティング] ドロップダウンメニューから、マシンのコンピューティングマシンタイプを選択します。
7. [容量] テキストフィールドに、フリート内の最小インスタンス数を入力します。
8. オーバーフロー動作 フィールドで、需要がフリート容量を超えた場合の動作を選択します。これらのパラメータの詳細については、「リザーブドキャパシティフリートのプロパティ」を参照してください。
9. [コンピューティングフリートの作成] を選択します。
10. コンピューティングフリートが作成されたら、新しい CodeBuild プロジェクトを作成するか、既存のプロジェクトを編集します。[環境] から [プロビジョニングモデル] の [リザーブドキャパシティ] を選択し、[フリート名] で指定したフリートを選択します。

ベストプラクティス

リザーブドキャパシティフリートを使用する場合は、以下のベストプラクティスに従うことをお勧めします。

- ソースをキャッシュしてビルドパフォーマンスを向上させるには、ソースキャッシュモードを使用することをお勧めします。
- Docker レイヤーキャッシュを使用し、既存の Docker レイヤーをキャッシュしてビルドパフォーマンスを向上させることをお勧めします。

リザーブドキャパシティフリートを複数の CodeBuild プロジェクトで共有できますか？

はい。フリートのキャパシティを複数のプロジェクトで使用することで、そのキャパシティを最大限に活用できます。

リザーブドキャパシティフリートをサポートしているのはどのリージョンですか?

リザーブドキャパシティフリートは、 AWS リージョン米国東部 (バージニア北部) 、米国東部 (オハイオ) 、米国西部 (オレゴン) 、アジアパシフィック (ムンバイ) 、アジアパシフィック (シンガポール) 、アジアパシフィック (シドニー) 、アジアパシフィック (東京) 、欧州 (フランクフルト) 、欧州 (アイルランド) 、南米 (サンパウロ) でサポートされています。 CodeBuild が利用可能なの詳細については AWS リージョン 、AWS 「リージョン別の のサービス」を参照してください。

リザーブドキャパシティフリートのプロパティ

リザーブドキャパシティフリートには以下のプロパティが含まれます。

オペレーティングシステム

オペレーティングシステム。使用できるオペレーションシステムは次のとおりです。

- Amazon Linux
- [Windows Server 2019]
- Windows Server 2022

アーキテクチャ

プロセッサアーキテクチャ。以下のアーキテクチャが利用可能です。

- x86_64
- Arm64

コンピューティング

各インスタンスのコンピューティングマシンタイプ。次のマシンタイプを使用できます。

コンピューティングタイプ	環境 computeType 値	環境タイプ値	メモリ	vCPU	ディスク容量
ARM Small	BUILD_GENERAL1_SMALL	ARM_CONTAINER	4 GB	2	50 GB
ARM large	BUILD_GENERAL1_LARGE	ARM_CONTAINER	16 GB	8	50 GB
Linux Small [1]	BUILD_GENERAL1_SMALL	LINUX_CONTAINER	3 GB	2	64 GB
Linux Medium [1]	BUILD_GENERAL1_MEDIUM	LINUX_CONTAINER	7 GB	4	128 GB
Linux Large [1]	BUILD_GENERAL1_LARGE	LINUX_CONTAINER	15 GB	8	128 GB
Linux XLarge	BUILD_GENERAL1_XLARGE	LINUX_CONTAINER	70 GB	36	256 GB
Linux 2xlarge	BUILD_GENERAL1_2XLARGE	LINUX_CONTAINER	145 GB	72	824 GB (SSD)
Linux GPU Sm	BUILD_GENERAL1_SMALL	LINUX_GPU_CONTAINER	16 GB	4	220 GB

コンピューティングタイプ	環境 computeType 値	環境タイプ値	メモリ	vCPU	ディスク容量
Linux GPU large	BUILD_GENERAL1_LARGE	LINUX_GPU_CONTAINER	255 GB	32	50 GB
Windows medi	BUILD_GENERAL1_MEDIUM	WINDOWS_SERVER_2019_CONTAINER	7 GB	4	128 GB
Windows medi	BUILD_GENERAL1_MEDIUM	WINDOWS_SERVER_2022_CONTAINER	7 GB	4	128 GB
Windows large	BUILD_GENERAL1_LARGE	WINDOWS_SERVER_2019_CONTAINER	15 GB	8	128 GB
Windows large	BUILD_GENERAL1_LARGE	WINDOWS_SERVER_2022_CONTAINER	15 GB	8	128 GB

容量

フリートに割り当てられるマシンの初期数。これにより、並列で実行できるビルドの数が定義されます。

オーバーフロー動作

ビルド数がフリート容量を超えたときの動作を定義します。

オンデマンド

オーバーフロービルドは CodeBuild オンデマンドで実行されます。

> **Important**
>
> オーバーフロー動作をオンデマンドに設定することを選択した場合、オーバーフロービルドはオンデマンド Amazon EC2 と同様に個別に請求されることに注意してください。詳細については、「https://aws.amazon.com/codebuild/pricing/」を参照してください。

キュー

ビルドの実行は、マシンが使用可能になるまでキューに入れられます。これにより、さらにマシンが割り当てられないため、追加のコストが抑えられます。

を使用したリザーブドキャパシティのサンプル AWS CodeBuild

これらのサンプルは、 でリザーブドキャパシティフリートを試すために使用できます CodeBuild。

トピック

- リザーブドキャパシティのサンプルを使用したキャッシュ

リザーブドキャパシティのサンプルを使用したキャッシュ

キャッシュでは、ビルド環境の再利用可能な部分が保存され、複数のビルドでそれらを使用することができます。このサンプルでは、リザーブドキャパシティを使用してビルドプロジェクト内のキャッシュを有効にする方法を示しました。詳細については、「AWS CodeBuild でのキャッシュのビルド」を参照してください。

プロジェクト設定で 1 つ以上のキャッシュモードを指定することから開始できます。

```
Cache:
        Type: LOCAL
        Modes:
          - LOCAL_CUSTOM_CACHE
          - LOCAL_DOCKER_LAYER_CACHE
          - LOCAL_SOURCE_CACHE
```

> **Note**
>
> Docker レイヤーキャッシュを使用するには、必ず特権モードを有効にしてください。

プロジェクトの buildspec 設定は以下のようになります。

```
version: 0.2
      phases:
        build:
          commands:
            - echo testing local source cache
            - touch /codebuild/cache/workspace/foobar.txt
            - git checkout -b cached_branch
            - echo testing local docker layer cache
            - docker run alpine:3.14 2>&1 | grep 'Pulling from' || exit 1
            - echo testing local custom cache
            - touch foo
            - mkdir bar && ln -s foo bar/foo2
            - mkdir bar/bar && touch bar/bar/foo3 && touch bar/bar/foo4
            - "[ -f foo ] || exit 1"
            - "[ -L bar/foo2 ] || exit 1"
            - "[ -f bar/bar/foo3 ] || exit 1"
            - "[ -f bar/bar/foo4 ] || exit 1"
      cache:
        paths:
           - './foo'
           - './bar/**/*'
           - './bar/bar/foo3'
```

新しいプロジェクトでビルドを実行してキャッシュをシードすることから開始できます。それが完了したら、次のように buildspec を上書きして別のビルドを開始する必要があります。

```
version: 0.2
      phases:
        build:
          commands:
            - echo testing local source cache
            - git branch | if grep 'cached_branch'; then (exit 0); else (exit 1); fi
            - ls /codebuild/cache/workspace | if grep 'foobar.txt'; then (exit 0); else
 (exit 1); fi
            - echo testing local docker layer cache
```

```
            - docker run alpine:3.14 2>&1 | if grep 'Pulling from'; then (exit 1); else
(exit 0); fi
            - echo testing local custom cache
            - "[ -f foo ] || exit 1"
            - "[ -L bar/foo2 ] || exit 1"
            - "[ -f bar/bar/foo3 ] || exit 1"
            - "[ -f bar/bar/foo4 ] || exit 1"
      cache:
        paths:
           - './foo'
           - './bar/**/*'
           - './bar/bar/foo3'
```

リザーブドキャパシティフリートの制限

リザーブドキャパシティフリートではサポートされていないユースケースがいくつかあります。それにより影響が出る場合は、代わりにオンデマンドフリートを使用してください。

- リザーブドキャパシティフリートは、バッチビルド、ビルド使用率メトリクス、またはセマンティックバージョニングをサポートしていません。
- リザーブドキャパシティフリートは VPC 接続をサポートしていません。

クォータと制限の詳細については、「コンピューティングフリート」を参照してください。

でのテストレポートの使用 AWS CodeBuild

ビルド中に実行されるテストの詳細 CodeBuild を含むレポートを で作成できます。単体テスト、設定テスト、機能テストなどのテストを作成できます。

以下のテストレポートファイル形式がサポートされています。

- Cucumber JSON (.json)
- JUnit XML (.xml)
- NUnit XML (.xml)
- NUnit3 XML (.xml)
- TestNG XML (.xml)
- Visual Studio TRX (.trx)
- Visual Studio TRX XML (.xml)

> **Note**
>
> `cucumber-js` のサポートされている最新バージョンは 7.3.2 です。

Surefire JUnit plugin、TestNG、Cucumber などのいずれかの形式でレポートファイルを作成できる任意のテストフレームワークを使用して、テストケースを作成します。

テストレポートを作成するには、ビルドプロジェクトの buildspec ファイルにテストケースに関する情報を含むレポートグループ名を追加します。ビルドプロジェクトを実行すると、テストケースが実行され、テストレポートが作成されます。テストを実行する前にレポートグループを作成する必要はありません。レポートグループ名を指定すると、レポートの実行時に CodeBuild によってレポートグループが作成されます。既に存在するレポートグループを使用する場合は、buildspec ファイルでその ARN を指定します。

テストレポートを使用すると、ビルドの実行中に問題をトラブルシューティングできます。ビルドプロジェクトの複数のビルドから多数のテストレポートがある場合、テストレポートを使用してトレンドやテストと失敗率を表示し、ビルドを最適化できます。

レポートは、作成から 30 日後に有効期限が切れます。期限切れのテストレポートは表示できません。30 日以上テストレポートを保持する場合は、テスト結果の生データファイルをAmazon S3 バ

ケットにエクスポートできます。エクスポートされたテストファイルは期限切れになりません。S3 バケットに関する情報は、レポートグループを作成するときに指定します。

> **Note**
>
> プロジェクトで指定された CodeBuild サービスロールは、S3 バケットにアップロードするアクセス許可に使用されます。

トピック

テストレポートの作成

テストレポートを作成するには、buildspec ファイルに 1 つから 5 つのレポートグループで設定されているビルドプロジェクトを実行します。テストレポートは、実行中に作成されます。レポートグループに対して指定されたテストケースの結果が含まれます。同じ buildspec ファイルを使用する後続のビルドごとに、新しいテストレポートが生成されます。

テストレポートを作成するには

1. ビルドプロジェクトを作成します。詳細については、 でのビルドプロジェクトの作成AWS CodeBuild を参照してください。
2. テストレポート情報を使用してプロジェクトの buildspec ファイルを設定します。

    a. `reports:` セクションを追加し、既存のレポートグループの ARN、またはレポートグループの名前を指定します。

    ARN を指定すると、 はそのレポートグループ CodeBuild を使用します。

名前を指定すると、 はプロジェクト名と指定した名前を *<project-name>-#report-group-name#* 形式で使用してレポートグループ CodeBuild を作成します。名前付きレポートグループが既に存在する場合は、そのレポートグループ CodeBuild を使用します。

b. レポートグループで、テスト結果が含まれるファイルの場所を指定します。複数のレポートグループを使用する場合は、各レポートグループに対してテスト結果ファイルの場所を指定します。ビルドプロジェクトを実行するたびに、新しいテストレポートが作成されます。詳細については、「テストファイルの指定」を参照してください。

c. build または post_build シーケンスの commands セクションで、レポートグループに対して指定したテストケースを実行するコマンドを指定します。詳細については、「テストコマンドの指定」を参照してください。

buildspec reports セクションの例を以下に示します。

```
reports:
  php-reports:
    files:
      - "reports/php/*.xml"
    file-format: "JUNITXML"
  nunit-reports:
    files:
      - "reports/nunit/*.xml"
    file-format: "NUNITXML"
```

3. ビルドプロジェクトのビルドを実行します。詳細については、「AWS CodeBuild でのビルドの実行」を参照してください。
4. ビルドが完了したら、プロジェクトページの [Build history (ビルド 履歴)] から新しいビルド実行を選択します。[Reports (レポート)] を選択して、テストレポートを表示します。詳細については、「ビルドのテストレポートの表示」を参照してください。

テストレポートの使用

レポートグループにはテストレポートが含まれており、共有設定を指定します。buildspec ファイルを使用して、実行するテストケースと、ビルド時に実行するコマンドを指定します。ビルドプロジェクトで設定された各レポートグループに対して、ビルドプロジェクトの実行によってテストレポートが作成されます。レポートグループで設定されたビルドプロジェクトを複数実行すると、そのレポー

トグループに複数のテストレポートが作成され、そのレポートグループに指定された同じテストケースの結果がそれぞれ作成されます。

テストケースは、ビルドプロジェクトの buildspec ファイル内のレポートグループに対して指定されています。1 つのビルドプロジェクトで最大 5 つのレポートグループを指定できます。ビルドを実行すると、すべてのテストケースが実行されます。新しいテストレポートは、レポートグループに指定された各テストケースの結果で作成されます。新しいビルドを実行するたびに、テストケースが実行され、新しいテスト結果を使用して新しいテストレポートが作成されます。

レポートグループは、複数のビルドプロジェクトで使用できます。1つのレポートグループで作成されたすべてのテストレポートは、異なるビルドプロジェクトを使用してテストレポートを作成した場合でも、エクスポートオプションやアクセス権限など、同じ設定を共有します。複数のビルドプロジェクトで 1 つのレポートグループを使用して作成されたテストレポートには、異なるテストケースセット (ビルドプロジェクトごとに 1 セットのテストケース) の実行結果を含めることができます。これは、各プロジェクトの buildspec ファイルで、レポートグループに異なるテストケースファイルを指定できるためです。また、buildspec ファイルを編集して、ビルドプロジェクトのレポートグループのテストケースファイルを変更することもできます。その後のビルド実行では、更新された buildspec のテストケースファイルの結果を含む新しいテストレポートが作成されます。

トピック

- Create a report group
- レポートグループの更新
- テストファイルの指定
- テストコマンドの指定
- Report group naming
- AWS CodeBuild でのレポートグループのタグ付け
- 共有レポートグループの使用

Create a report group

CodeBuild コンソール、 AWS CLI、または buildspec ファイルを使用して、レポートグループを作成できます。IAM ロールには、レポートグループを作成するために必要なアクセス権限が必要です。詳細については、「テストレポートのアクセス許可の使用」を参照してください。

トピック

- レポートグループの作成 (buildspec)

- Create a report group (console)
- レポートグループの作成 (CLI)
- レポートグループの作成 (AWS CloudFormation)

レポートグループの作成 (buildspec)

buildspec を使用して作成されたレポートグループは、生のテスト結果ファイルをエクスポートしません。レポートグループを表示し、エクスポート設定を指定できます。詳細については、「レポートグループの更新」を参照してください。

buildspec ファイルを使用してレポートグループを作成するには

1. AWS アカウントのレポートグループに関連付けられていないレポートグループ名を選択します。
2. buildspec ファイルの `reports` セクションをこの名前で設定します。この例では、レポートグループ名は `new-report-group` で、ユーステストケースは JUnit フレームワークを使用して作成されます。

```
reports:
 new-report-group: #surefire junit reports
   files:
     - '**/*'
   base-directory: 'surefire/target/surefire-reports'
```

レポートグループ名は、buildspec の環境変数を使用して指定することもできます。

```
version: 0.2
env:
  variables:
    REPORT_GROUP_NAME: "new-report-group"
phases:
  build:
    commands:
      - ...
...
reports:
 $REPORT_GROUP_NAME:
   files:
     - '**/*'
```

```
    base-directory: 'surefire/target/surefire-reports'
```

詳細については、「テストファイルの指定」および「Reports syntax in the buildspec file」を参照してください。

3. commands セクションで、テストを実行するコマンドを指定します。詳細については、「テストコマンドの指定」を参照してください。
4. ビルドを実行します。ビルドが完了すると、形式 project-name-report-group-name を使用する名前で新しいレポートグループが作成されます。詳細については、「Report group naming」を参照してください。

Create a report group (console)

テストレポートを作成するには

1. AWS CodeBuild コンソール (https://console.aws.amazon.com/codesuite/codebuild/home) を開きます。
2. ナビゲーションペインで、[Report groups (レポートグループ)] を選択します。
3. [Create report group (レポートグループを作成)] を選択します。
4. [Report group name (レポートグループ名)] に、レポートグループの名前を入力します。
5. (オプション) [タグ] に、サポート対象の AWS のサービスで使用するタグの名前と値を入力します。[Add row] を使用して、タグを追加します。最大 50 個のタグを追加できます。
6. テストレポート結果の raw データを Amazon S3 バケットにアップロードする場合は、次のようにします。

    a. [Export to Amazon S3] を選択します。

    b. [S3 bucket name (S3 バケット名)] に、S3 バケットの名前を入力します。

    c. (オプション)S3 バケット所有者で S3 バケットを所有するアカウントの AWS アカウント識別子を入力します。これにより、レポートデータを、ビルドを実行しているアカウント以外のアカウントが所有する Amazon S3 バケットにエクスポートできます。

    d. [Path prefix (パスプレフィックス)] に、テスト結果をアップロードする S3 バケットのパスを入力します。

    e. 生のテスト結果データファイルを圧縮するには、[Compress test result data in a zip file (テスト結果データを圧縮する)] を選択します。

f. [Additional configuration (追加の設定)] を展開して、暗号化オプションを表示します。次のいずれかを選択します。

- Amazon S3 の AWS マネージドキー を使用するためのデフォルトの AWS 管理のキー。詳細については、AWS Key Management Service ユーザーガイドの「カスタマー マネージド CMKs」を参照してください。これはデフォルトの暗号化オプションです。
- カスタムキーを選択して、ユーザーが作成して設定するカスタマー管理のキーを使用します。AWS KMS 暗号化キーの場合は、暗号化キーの ARN を入力します。形式は `arn:aws:kms:<region-id>: <aws-account-id>:key/<key-id>` です。詳細については、AWS Key Management Service ユーザーガイドの「KMS キーの作成」を参照してください。
- 暗号化を無効にするには、アーティファクト暗号化を無効にします。テスト結果を共有したり、静的ウェブサイトに公開したりする場合は、このオプションを選択します。(動的ウェブサイトでは、テスト結果を復号化するコードを実行できます)。

保管時の暗号化の詳細については、「データ暗号化」を参照してください。

> **Note**
>
> プロジェクトで指定した CodeBuild サービスロールは、S3 バケットにアップロードするアクセス許可に使用されます。

7. [Create report group (レポートグループを作成)] を選択します。

レポートグループの作成 (CLI)

レポートグループの作成

1. `CreateReportGroup.json` という名前のファイルを作成します。
2. 要件に応じて、以下の JSON コードスニペットのいずれかを `CreateReportGroup.json` にコピーします。

- 次の JSON を使用して、テストレポートグループが生のテスト結果ファイルを Amazon S3 バケットにエクスポートするように指定します。

```
{
```

```
  "name": "<report-name>",
  "type": "TEST",
  "exportConfig": {
    "exportConfigType": "S3",
    "s3Destination": {
      "bucket": "<bucket-name>",
      "bucketOwner": "<bucket-owner>",
      "path": "<path>",
      "packaging": "NONE | ZIP",
      "encryptionDisabled": "false",
      "encryptionKey": "<your-key>"
    },
    "tags": [
      {
        "key": "tag-key",
        "value": "tag-value"
      }
    ]
  }
}
```

- 「*<bucket-name>*」を Amazon S3 バケット名に、「*<path>*」をファイルをエクスポートするバケット内のパスに置き換えます。
- エクスポートされたファイルを packaging に圧縮する場合は、ZIP を指定します。それ以外の場合は、NONE を指定します。
- bucketOwner「」はオプションで、Amazon S3 バケットがビルドを実行しているアカウント以外のアカウントによって所有されている場合にのみ必要です。
- エクスポートされたファイルを暗号化するかどうかを指定するために encryptionDisabled を使用します。エクスポートしたファイルを暗号化する場合は、カスタマー管理のキーを入力します。詳細については、「レポートグループの更新」を参照してください。

- 次の JSON を使用して、テストレポートで生のテストファイルをエクスポートしないように指定します。

```
{
  "name": "<report-name>",
  "type": "TEST",
  "exportConfig": {
    "exportConfigType": "NO_EXPORT"
  }
```

```
}
```

> **Note**
>
> プロジェクトで指定した CodeBuild サービスロールは、S3 バケットにアップロードするアクセス許可に使用されます。

3. 次のコマンドを実行します。

```
aws codebuild create-report-group --cli-input-json file://
CreateReportGroupInput.json
```

レポートグループの作成 (AWS CloudFormation)

AWS CloudFormation テンプレートを使用してテストレポートを作成するには

AWS CloudFormation テンプレートファイルを使用して、レポートグループを作成およびプロビジョニングできます。詳細については、AWS CloudFormation ユーザーガイド を参照してください。

次の AWS CloudFormation YAML テンプレートは、生のテスト結果ファイルをエクスポートしないレポートグループを作成します。

```
Resources:
  CodeBuildReportGroup:
    Type: AWS::CodeBuild::ReportGroup
    Properties:
      Name: my-report-group-name
      Type: TEST
      ExportConfig:
        ExportConfigType: NO_EXPORT
```

次の AWS CloudFormation YAML テンプレートは、生のテスト結果ファイルを Amazon S3 バケットにエクスポートするレポートグループを作成します。

```
Resources:
  CodeBuildReportGroup:
    Type: AWS::CodeBuild::ReportGroup
    Properties:
```

```
      Name: my-report-group-name
      Type: TEST
      ExportConfig:
        ExportConfigType: S3
        S3Destination:
          Bucket: my-s3-bucket-name
          Path: path-to-folder-for-exported-files
          Packaging: ZIP
          EncryptionKey: my-KMS-encryption-key
          EncryptionDisabled: false
```

> **Note**
>
> プロジェクトで指定された CodeBuild サービスロールは、S3 バケットにアップロードするアクセス許可に使用されます。

レポートグループの更新

レポートグループを更新するときは、生のテスト結果データを Amazon S3 バケット内のファイルにエクスポートするかどうかに関する情報を指定できます。S3 バケットへのエクスポートを選択した場合は、レポートグループについて以下を指定します。

- 生のテスト結果ファイルが ZIP ファイルに圧縮されているかどうか。
- 生のテスト結果ファイルが暗号化されているかどうか。次のいずれかの方法で暗号化を指定できます。
  - Amazon S3 AWS マネージドキー の 。 Amazon S3
  - ユーザーが作成して設定するカスタマー管理のキー。

詳細については、「データ暗号化」を参照してください。

を使用してレポートグループ AWS CLI を更新する場合は、タグを更新または追加することもできます。詳細については、「AWS CodeBuild でのレポートグループのタグ付け」を参照してください。

> **Note**
>
> プロジェクトで指定された CodeBuild サービスロールは、S3 バケットにアップロードするアクセス許可に使用されます。

トピック

レポートグループの更新 (コンソール)

レポートグループを更新するには

1. https://console.aws.amazon.com/codesuite/codebuild/home で AWS CodeBuild コンソールを開きます。
2. ナビゲーションペインで、[Report groups (レポートグループ)] を選択します。
3. 更新するレポートグループを選択します。
4. [編集] を選択します。
5. [Backup to Amazon S3] (Amazon S3 にバックアップ) を選択または選択解除します。このオプションを選択した場合は、エクスポート設定を指定します。

   a. [S3 bucket name (S3 バケット名)] に、S3 バケットの名前を入力します。

   b. [Path prefix (パスプレフィックス)] に、テスト結果をアップロードする S3 バケットのパスを入力します。

   c. 生のテスト結果データファイルを圧縮するには、[Compress test result data in a zip file (テスト結果データを圧縮する)] を選択します。

   d. [Additional configuration (追加の設定)] を展開して、暗号化オプションを表示します。以下のうちのひとつを選択します。

      - Amazon S3 S3 の を使用するデフォルトの AWS マネージドキー。 AWS マネージドキー詳細については、AWS Key Management Service ユーザーガイドの「カスタマー マネージド CMKs」を参照してください。これはデフォルトの暗号化オプションです。
      - カスタムキーを選択して、ユーザーが作成して設定するカスタマー管理のキーを使用します。AWS KMS 暗号化キーの場合は、暗号化キーの ARN を入力します。形式は `arn:aws:kms:<region-id>: <aws-account-id>:key/<key-id>` です。詳細については、AWS Key Management Service ユーザーガイドの「KMS キーの作成」を参照してください。
      - 暗号化を無効にするには、アーティファクト暗号化を無効にします。テスト結果を共有したり、静的ウェブサイトに公開したりする場合は、このオプションを選択します。(動的ウェブサイトでは、テスト結果を復号化するコードを実行できます)。

レポートグループの更新 (CLI)

レポートグループを更新するには

1. UpdateReportGroupInput.json という名前のファイルを作成します。
2. 以下を UpdateReportGroupInput.json にコピーします。

```
{
    "arn": "",
    "exportConfig": {
        "exportConfigType": "S3",
        "s3Destination": {
            "bucket": "bucket-name",
            "path": "path",
            "packaging": "NONE | ZIP",
            "encryptionDisabled": "false",
            "encryptionKey": "your-key"
         }
     },
     "tags": [
        {
            "key": "tag-key",
            "value": "tag-value"
        }
     ]
}
```

3. レポートグループの ARN を arn 行に入力します ("arn":"arn:aws:codebuild:*region*:*123456789012*:report-group/*report-group-1*") など)。
4. レポートグループに適用する更新内容で UpdateReportGroupInput.json を更新します。

   - レポートグループを更新して生のテスト結果ファイルを S3 バケットにエクスポートする場合は、exportConfig セクションを更新します。bucket-name を S3 バケット名に、path をファイルをエクスポートする S3 バケット内のパスに置き換えます。エクスポートされたファイルを packaging に圧縮する場合は、ZIP を指定します。それ以外の場合は、NONE を指定します。エクスポートされたファイルを暗号化するかどうかを指定するために encryptionDisabled を使用します。エクスポートしたファイルを暗号化する場合は、カスタマー管理のキーを入力します。

- 生のテスト結果ファイルを S3 バケットにエクスポートしないようにレポートグループを更新する場合は、exportConfig セクションを以下の JSON で更新します。

```
{
  "exportConfig": {
      "exportConfigType": "NO_EXPORT"
  }
}
```

- レポートグループのタグを更新する場合は、tags セクションを更新します。タグは変更、追加、または削除できます。すべてのタグを削除する場合は、以下の JSON で更新します。

```
"tags": []
```

5. 次のコマンドを実行します。

```
aws codebuild update-report-group \
--cli-input-json file://UpdateReportGroupInput.json
```

テストファイルの指定

ビルドプロジェクトの buildspec ファイルの reports セクションで、各レポートグループのテスト結果ファイルとその場所を指定します。詳細については、「Reports syntax in the buildspec file」を参照してください。

以下は、ビルドプロジェクトの 2 つのレポートグループを指定するサンプル reports セクションです。1 つは ARN で指定され、もう 1 つは名前で指定されます。files セクションでは、テストケースの結果を含むファイルを指定します。オプション base-directory セクションでは、テストケースファイルがあるディレクトリを指定します。オプションの discard-paths セクションでは、Amazon S3 バケットにアップロードされたテスト結果ファイルへのパスを破棄するかどうかを指定します。

```
reports:
  arn:aws:codebuild:your-region:your-aws-account-id:report-group/report-group-name-1:
 #surefire junit reports
    files:
      - '**/*'
    base-directory: 'surefire/target/surefire-reports'
    discard-paths: false
```

```
  sampleReportGroup: #Cucumber reports from json plugin
    files:
      - 'cucumber-json/target/cucumber-json-report.json'
    file-format: CUCUMBERJSON #Type of the report, defaults to JUNITXML
```

テストコマンドの指定

テストケースを実行するコマンドは、buildspec ファイルの commands セクションで指定します。これらのコマンドは、buildspec ファイルの reports セクションでレポートグループに指定されたテストケースを実行します。次に、テストファイルでテストを実行するコマンドを含むサンプル commands セクションを示します。

```
commands:
    - echo Running tests for surefire junit
    - mvn test -f surefire/pom.xml -fn
    - echo
    - echo Running tests for cucumber with json plugin
    - mvn test -Dcucumber.options="--plugin json:target/cucumber-json-report.json" -f
 cucumber-json/pom.xml -fn
```

詳細については、「buildspec の構文」を参照してください。

Report group naming

AWS CLI または AWS CodeBuild コンソールを使用してレポートグループを作成する場合は、レポートグループの名前を指定します。buildspec を使用して新しいレポートグループを作成する場合、*project-name*-*report-group-name-specified-in-buildspec* 形式を使用して名前が付けられます。そのビルドプロジェクトのビルドを実行することによって作成されたすべてのレポートは、新しい名前を持つ新しいレポートグループに属します。

新しいレポートグループ CodeBuild を作成しない場合は、ビルドプロジェクトの buildspec ファイルでレポートグループの ARN を指定します。レポートグループの ARN は、複数のビルドプロジェクトで指定できます。各ビルドプロジェクトが実行されると、レポートグループには各ビルドプロジェクトによって作成されたテストレポートが含まれます。

たとえば、my-report-group という名前のレポートグループを 1 つ作成し、その名前を my-project-1 と my-project-2 という名前の 2 つの異なるビルドプロジェクトで使用し、両方のプロジェクトのビルドを作成した場合、2 つの新しいレポートグループが作成されます。結果は、次の名前を持つ 3 つのレポートグループになります。

- my-report-group: テストレポートはありません。
- my-project-1-my-report-group: という名前のビルドプロジェクトによって実行されたテストの結果を含むレポートが含まれます。my-project-1
- my-project-2-my-report-group: という名前のビルドプロジェクトによって実行されたテストの結果を含むレポートが含まれます。my-project-2

両方のプロジェクトで my-report-group という名前のレポートグループの ARN を使用し、各プロジェクトのビルドを実行しても、1 つのレポートグループ (my-report-group) は残ります。そのレポートグループには、両方のビルドプロジェクトによって実行されるテストの結果を含むテストレポートが含まれます。

AWS アカウントのレポートグループに属していないレポートグループ名を選択し、buildspec ファイル内のレポートグループにその名前を使用し、ビルドプロジェクトのビルドを実行すると、新しいレポートグループが作成されます。新しいレポートグループの名前の形式は *project-name*-*new-group-name* です。例えば、 という名前のレポートグループが AWS アカウントになくnew-report-group、 という名前のビルドプロジェクトで指定した場合test-project、ビルド実行によって という名前の新しいレポートグループが作成されますtest-project-new-report-group。

AWS CodeBuild でのレポートグループのタグ付け

タグは、ユーザーまたは AWS が AWS リソースに割り当てるカスタム属性ラベルです。各 AWS タグは 2 つの部分で構成されます。

- タグキー (例: CostCenter、Environment、Project、Secret)。タグキーでは、大文字と小文字が区別されます。
- タグ値として知られるオプションのフィールド (例: 111122223333、Production、チーム名など)。タグ値を省略すると、空の文字列を使用した場合と同じになります。タグキーと同様に、タグ値は大文字と小文字が区別されます。

これらは共にキーと値のペアと呼ばれます。レポートグループに付けることができるタグの最大数、およびタグのキーと値の制限については、「タグ」を参照してください。

タグを使用すると、AWS リソースの特定と整理に役立ちます。多くの AWS のサービスではタグ付けがサポートされるため、さまざまなサービスからリソースに同じタグを割り当てて、リソースの関連を示すことができます。たとえば、Amazon S3 バケットに割り当てたものと同じタグを

CodeBuild レポートグループに割り当てることができます。タグの使用の詳細については、「タグ付けのベストプラクティス」ホワイトペーパーを参照してください。

CodeBuild では、主なリソースはレポートグループとプロジェクトです。CodeBuild コンソール、AWS CLI、CodeBuild API、または AWS SDK を使用して、レポートグループのタグの追加、管理、削除ができます。タグを使用して、レポートグループを識別、組織付け、追跡するだけでなく、IAM ポリシーでタグを使用して、レポートグループを表示および操作できるユーザーをコントロールすることもできます。タグベースのアクセスポリシーの例については、「タグを使用した AWS CodeBuild リソースへのアクセスのコントロール」を参照してください。

トピック

レポートグループにタグを追加する

レポートグループにタグを追加すると、AWS リソースの識別と整理、アクセスの管理に役立ちます。まず、レポートグループに 1 つ以上のタグ (キーと値のペア) を追加します。レポートグループに付けることができるタグの数には制限があります。キーフィールドおよび値フィールドに使用できる文字には制限があります。詳細については、「タグ」を参照してください。タグを追加した後、IAM ポリシーを作成して、それらのタグに基づいてレポートグループへのアクセスを管理できます。CodeBuild コンソールまたは AWS CLI を使用して、レポートグループにタグを追加できます。

> **Important**
>
> レポートグループにタグを追加すると、そのレポートグループへのアクセスに影響を与える可能性があります。レポートグループにタグを追加する前に、タグを使用してレポートグループなどのリソースへのアクセスをコントロールする可能性のある IAM ポリシーを必ず確認してください。タグベースのアクセスポリシーの例については、「タグを使用した AWS CodeBuild リソースへのアクセスのコントロール」を参照してください。

レポートグループの作成時にタグを追加する方法の詳細については、「Create a report group (console)」を参照してください。

トピック

- レポートグループにタグを追加する (コンソール)
- レポートグループにタグを追加する (AWS CLI)

レポートグループにタグを追加する (コンソール)

CodeBuild コンソールを使用して、CodeBuild レポートグループに 1 つ以上のタグを追加できます。

1. CodeBuild コンソール (https://console.aws.amazon.com/codebuild/) を開きます。
2. [Report groups (レポートグループ)] で、タグを追加するレポートグループの名前を選択します。
3. ナビゲーションペインで [Settings] (設定) をクリックします。
4. レポートグループにいずれのタグも追加されていない場合は、[Add tag (タグの追加)] を選択します。[Edit (編集)] を選択してから、[Add tag (タグの追加)] を選択することもできます。
5. [Key] に、タグの名前を入力します。[Value] では、任意でタグに値を追加できます。
6. (オプション) 別のタグを追加するには、[Add tag] を再度選択します。
7. タグの追加を完了したら、[Submit] を選択します。

レポートグループにタグを追加する (AWS CLI)

作成時にレポートグループにタグを追加するには、「レポートグループの作成 (CLI)」を参照してください。`CreateReportGroup.json` で、タグを追加します。

既存のレポートグループにタグを追加するには、「レポートグループの更新 (CLI)」を参照し、`UpdateReportGroupInput.json` でタグを追加します。

以下の手順では、AWS CLI の最新版をすでにインストールしているか、最新版に更新しているものとします。詳細については、「AWS Command Line Interface のインストール」を参照してください。

レポートグループのタグを表示する

タグは、AWS リソースの識別と整理、アクセスの管理に役立ちます。タグの使用の詳細については、「タグ付けのベストプラクティス」ホワイトペーパーを参照してください。タグベースのアクセスポリシーの例については、「Deny or allow actions on report groups based on resource tags」を参照してください。

レポートグループのタグを表示する (コンソール)

CodeBuild コンソールを使用して、CodeBuild レポートグループに関連付けられたタグを表示できます。

1. CodeBuild コンソール (https://console.aws.amazon.com/codebuild/) を開きます。
2. [Report groups (レポートグループ)] で、タグを表示するレポートグループの名前を選択します。
3. ナビゲーションペインで [Settings] (設定) をクリックします。

レポートグループのタグを表示する (AWS CLI)

AWS CLI を使用してレポートグループの AWS タグを表示するには、以下の手順に従います。いずれのタグも追加されていない場合、返されるタグは空になります。

1. コンソールまたは AWS CLI を使用して、レポートグループの ARN を見つけます。その ARN をメモしておきます。

   AWS CLI

   次のコマンドを実行します。

   ```
   aws list-report-groups
   ```

   このコマンドは、以下のような JSON 形式の情報を返します。

   ```
   {
       "reportGroups": [
           "arn:aws:codebuild:region:123456789012:report-group/report-group-1",
           "arn:aws:codebuild:region:123456789012:report-group/report-group-2",
           "arn:aws:codebuild:region:123456789012:report-group/report-group-3"
       ]
   }
   ```

   レポートグループの ARN はそのグループの名前で終わります。この名前はレポートグループの ARN を識別するために使用できます。

   Console

   1. CodeBuild コンソール (https://console.aws.amazon.com/codebuild/) を開きます。

2. [Report groups (レポートグループ)] で、表示するタグが付いたレポートグループの名前を選択します。
3. [Configuration (設定)] で、レポートグループの ARN を見つけます。

2. 次のコマンドを実行します。`--report-group-arns` パラメータにはメモしておいた ARN を使用します。

```
aws codebuild batch-get-report-groups --report-group-arns
 arn:aws:codebuild:region:123456789012:report-group/report-group-name
```

成功すると、このコマンドは、以下のような `tags` セクションを含む JSON 形式の情報を返します。

```
{
    ...
    "tags": {
        "Status": "Secret",
        "Project": "TestBuild"
    }
    ...
}
```

レポートグループのタグを編集する

レポートグループに関連付けられたタグの値を変更できます。キーの名前を変更することもできます。これは、現在のタグを削除して、新しい名前と他のタグと同じ値を持つ、別のタグを追加することになります。キーフィールドと値フィールドに使用できる文字には制限があります。詳細については、「タグ」を参照してください。

> **Important**
>
> レポートグループのタグを編集すると、そのレポートグループへのアクセスに影響を与える可能性があります。レポートグループのタグの名前 (キー) または値を編集する前に、タグのキーや値を使用してレポートグループなどのリソースへのアクセスをコントロールする可能性のある IAM ポリシーを必ず確認してください。タグベースのアクセスポリシーの例については、「Deny or allow actions on report groups based on resource tags」を参照してください。

レポートグループのタグを編集する (コンソール)

CodeBuild コンソールを使用して、CodeBuild レポートグループに関連付けられたタグを編集できます。

1. CodeBuild コンソール (https://console.aws.amazon.com/codebuild/) を開きます。
2. [Report groups (レポートグループ)] で、タグを編集するレポートグループの名前を選択します。
3. ナビゲーションペインで [Settings] (設定) をクリックします。
4. [Edit] を選択します。
5. 次のいずれかを行ってください。
   - タグを変更するには、[Key] に新しい名前を入力します。タグの名前を変更することは、タグを削除して、新しいキー名を持つタグを追加することになります。
   - タグの値を変更するには、新しい値を入力します。値を空にする場合は、現在の値を削除してフィールドを空のままにします。
6. タグの編集を完了したら、[Submit] を選択します。

レポートグループのタグを編集する (AWS CLI)

レポートグループのタグを追加、変更、または削除するには、「レポートグループの更新 (CLI)」を参照してください。`UpdateReportGroupInput.json` のタグを更新します。

レポートグループからタグを削除する

レポートグループに関連付けられた 1 つ以上のタグを削除できます。タグを削除しても、そのタグに関連付けられた他の AWS リソースからタグを削除することにはなりません。

> **Important**
>
> レポートグループからタグを削除すると、そのレポートグループへのアクセスに影響を与える可能性があります。レポートグループからタグを削除する前に、タグのキーや値を使用してレポートグループなどのリソースへのアクセスをコントロールする可能性のある IAM ポリシーを必ず確認してください。タグベースのアクセスポリシーの例については、「タグを使用した AWS CodeBuild リソースへのアクセスのコントロール」を参照してください。

レポートグループからタグを削除する (コンソール)

CodeBuild コンソールを使用して、タグと レポートグループとの関連付けを解除できます。

1. CodeBuild コンソール (https://console.aws.amazon.com/codebuild/) を開きます。
2. [Report groups (レポートグループ)] で、タグを削除するレポートグループの名前を選択します。
3. ナビゲーションペインで [Settings] (設定) をクリックします。
4. [Edit] を選択します。
5. 削除するタグを見つけ、[Remove tag] を選択します。
6. タグの削除を完了したら、[Submit] を選択します。

レポートグループからタグを削除する (AWS CLI)

AWS CLI を使用して CodeBuild レポートグループからタグを削除するには、以下のステップに従います。タグを削除してもそのタグがなくなるわけではありません。タグとレポートグループとの関連付けが解除されるだけです。

> **Note**
>
> CodeBuild レポートグループを削除すると、削除されたレポートグループからすべてのタグの関連付けが解除されます。レポートグループを削除する前にタグを削除する必要はありません。

レポートグループから 1 つ以上のタグを削除するには、「レポートグループのタグを編集する (AWS CLI)」を参照してください。JSON 形式のデータの `tags` セクションを、削除するタグが含まれていない最新のタグのリストで更新します。すべてのタグを削除する場合は、`tags` セクションを以下のように更新します。

```
"tags: []"
```

共有レポートグループの使用

レポートグループを共有すると、複数の AWS アカウントまたはユーザーが、レポートグループ、期限切れのレポートおよびレポートのテスト結果を見ることができます。このモデルでは、レポートグ

ループを所有するアカウント (所有者) は、レポートグループを他のアカウント (コンシューマー) と共有します。コンシューマは、レポートグループを編集できません。レポートは、作成から 30 日後に期限切れになります。

コンテンツ

レポートグループを共有するための前提条件

レポートグループを共有するには、AWS アカウントを所有する必要があります。自分と共有されているレポートグループは共有できません。

共有しているレポートグループにアクセスするための前提条件

共有レポートグループにアクセスするには、コンシューマーの IAM ロールに `BatchGetReportGroups` アクセス許可が必要です。次のポリシーを IAM ロールに添付することができます。

```
{
    "Effect": "Allow",
    "Resource": [
        "*"
    ],
    "Action": [
        "codebuild:BatchGetReportGroups"
    ]
}
```

詳細については、「でのアイデンティティベースのポリシーの使用 AWS CodeBuild」を参照してください。

関連サービス

レポートグループ共有は、AWS Resource Access Manager アカウントと、または AWS RAM を介して AWS リソースを共有することを可能にするサービスである AWS (AWS Organizations) と統合されます。AWS RAM では、リソースと共有するコンシューマを指定する リソース共有 を作成して、所有するリソースを共有します。コンシューマーは、個別の AWS アカウントや、AWS Organizations 内の組織単位または AWS Organizations 組織全体として指定できます。

詳細については、AWS RAM ユーザーガイドを参照してください。

レポートグループの共有

レポートグループを共有すると、コンシューマーには、レポートグループとそのレポートに対する読み取り専用アクセス権が付与されます。コンシューマは AWS CLI を使用して、レポートグループ、そのレポート、および各レポートのテストケースの結果を表示できます。コンシューマは次を行うことはできません。

- CodeBuild コンソールでの共有レポートグループまたはそのレポートの表示。
- 共有レポートグループの編集。
- プロジェクト内の共有レポートグループの ARN を使用してレポートを実行。共有レポートグループを指定するプロジェクトのビルドが失敗します。

CodeBuild コンソールを使用して、既存のリソース共有にレポートグループを追加できます。新しいリソース共有にレポートグループを追加する場合は、まずAWS RAM コンソール でレポートグループを作成する必要があります。

レポートグループを組織単位または組織全体と共有するには、AWS Organizations との共有を有効にする必要があります。詳細については、AWS RAM ユーザーガイドの「AWS Organizations で共有を有効化する」を参照してください。

CodeBuild コンソール、AWS RAM コンソールまたは、AWS CLI を使用すると、所有しているレポートグループを共有できます。

所有するレポートグループを共有するには (CodeBuild コンソール)

1. AWS CodeBuild コンソール (https://console.aws.amazon.com/codesuite/codebuild/home) を開きます。
2. ナビゲーションペインで、[Report groups (レポートグループ)] を選択します。

3. 共有するプロジェクトを選択し、[Share (共有)] を選択します。詳細については、AWS RAM ユーザーガイドの「リソースの共有の作成」を参照してください。

所有するレポートグループを共有するには (AWS RAM コンソール)

AWS RAM ユーザーガイド の「リソース共有の作成」を参照してください。

所有するレポートグループを共有するには (AWS RAM コマンド)

create-resource-share コマンドを使用します。

所有するレポートグループを共有するには (CodeBuild コマンド)

put-resource-policy コマンドを使用します:

1. policy.json という名前のファイルを作成し、その中に次をコピーします。

```
{
   "Version":"2012-10-17",
   "Statement":[{
     "Effect":"Allow",
     "Principal":{
       "AWS":"consumer-aws-account-id-or-user"
     },
     "Action":[
       "codebuild:BatchGetReportGroups",
       "codebuild:BatchGetReports",
       "codebuild:ListReportsForReportGroup",
       "codebuild:DescribeTestCases"],
     "Resource":"arn-of-report-group-to-share"
   }]
 }
```

2. レポートグループ ARN とそれを共有する識別子で policy.json を更新します。次の例では、ARN arn:aws:codebuild:us-west-2:123456789012:report-group/my-report-group を持つレポートグループへの読み取り専用アクセスを 123456789012 で識別される AWS アカウントの Alice と root ユーザーに付与します。

```
{
   "Version":"2012-10-17",
   "Statement":[{
     "Effect":"Allow",
```

```
     "Principal":{
       "AWS": [
          "arn:aws:iam::123456789012:user/Alice",
          "123456789012"
        ]
     },
     "Action":[
       "codebuild:BatchGetReportGroups",
       "codebuild:BatchGetReports",
       "codebuild:ListReportsForReportGroup",
       "codebuild:DescribeTestCases"],
     "Resource":"arn:aws:codebuild:us-west-2:123456789012:report-group/my-report-
group"
   }]
 }
```

3. 次のコマンドを実行します。

```
aws codebuild put-resource-policy --resource-arn report-group-arn --policy file://
policy.json
```

共有レポートグループの共有解除

レポートとテストケースの結果が含まれる共有が解除されたレポートグループは、その所有者だけがアクセスできます。レポートグループの共有を解除すると、以前に共有した AWS アカウントまたはユーザーは、レポートグループ、そのレポート、またはレポート内のテストケースの結果にアクセスできなくなります。

所有するレポートグループを共有または共有を解除するには、リソース共有から削除する必要があります。これを実行するには、AWS RAM コンソールまたは AWS CLI を使用します。

所有している共有レポートグループの共有を解除するには (AWS RAM コンソール)

AWS RAM ユーザーガイド の「リソース共有の更新」を参照してください。

所有している共有レポートグループの共有を解除するには (AWS RAM コマンド)

disassociate-resource-share コマンドを使用します。

CodeBuild コマンドを所有しているレポートグループの共有を解除するには)

delete-resource-policy コマンドを実行し、共有を解除したいレポートグループの ARN を指定する:

```
aws codebuild delete-resource-policy --resource-arn report-group-arn
```

共有レポートグループの識別

所有者およびコンシューマーは、AWS CLI を使用して共有レポートグループを識別できます。

共有レポートグループとそのレポートを識別して情報を取得するには、次のコマンドを使用します。

- 自分と共有されているレポートグループの ARN を表示するには、list-shared-report-groups を実行します。

  ```
  aws codebuild list-shared-report-groups
  ```

- レポートグループ内のレポートの ARN を表示するには、レポートグループ ARN を使い list-reports-for-report-group を実行します。

  ```
  aws codebuild list-reports-for-report-group --report-group-arn report-group-arn
  ```

- レポート内のテストケースに関する情報を表示するには、レポート ARN を使い、describe-test-cases を実行します。

  ```
  aws codebuild describe-test-cases --report-arn report-arn
  ```

  出力は次のようになります。

  ```
  {
      "testCases": [
          {
              "status": "FAILED",
              "name": "Test case 1",
              "expired": 1575916770.0,
              "reportArn": "report-arn",
              "prefix": "Cucumber tests for agent",
              "message": "A test message",
              "durationInNanoSeconds": 1540540,
              "testRawDataPath": "path-to-output-report-files"
          },
          {
              "status": "SUCCEEDED",
              "name": "Test case 2",
              "expired": 1575916770.0,
  ```

```
            "reportArn": "report-arn",
            "prefix": "Cucumber tests for agent",
            "message": "A test message",
            "durationInNanoSeconds": 1540540,
            "testRawDataPath": "path-to-output-report-files"
        }
    ]
}
```

共有レポートグループのアクセス許可

所有者のアクセス許可

レポートグループの所有者は、レポートグループを編集し、プロジェクトで指定してレポートを実行できます。

コンシューマーのアクセス許可

レポートグループのコンシューマーは、レポートグループ、そのレポート、およびレポートのテストケース結果を表示できます。コンシューマーは、レポートグループまたはそのレポートを編集したり、レポートを作成したりすることはできません。

Working with reports (レポートの操作)

レポートには、1 つのレポートグループに対して指定されたテストケースの結果が含まれます。テストレポートは、ビルドプロジェクトの実行中に作成されます。buildspec ファイルでテストケースを実行するレポートグループ、テストケースファイル、コマンドを指定します。テストケースを実行するたびに、新しいテストレポートがレポートグループに作成されます。

テストレポートは、作成から 30 日後に有効期限が切れます。期限切れのテストレポートは表示できませんが、S3 バケット内の生のテスト結果ファイルにテスト結果を エクスポートすることはできます。エクスポートされた生のテストファイルは期限切れになりません。詳細については、「レポートグループの更新」を参照してください。

テストレポートのステータスは、次のいずれかになります。

- GENERATING: テストケースの実行はまだ進行中です。
- DELETING: テストレポートは削除されています。テストレポートが削除されると、そのテストケースも削除されます。S3 バケットにエクスポートされた生のテスト結果データファイルは削除されません。

- INCOMPLETE: テストレポートは完了していません。このステータスは、次のいずれかの理由で返されることがあります。
  - レポートのテストケースを指定するレポートグループの設定に問題があります。たとえば、buildspec ファイルのレポートグループのテストケースへのパスが正しくない可能性があります。
  - ビルドを実行した IAM ユーザーには、テストを実行するアクセス権限がありません。詳細については、「テストレポートのアクセス許可の使用」を参照してください。
  - テストに関連していないエラーのため、ビルドは完了しませんでした。
- SUCCEEDED: すべてのテストケースが成功しました。
- FAILED: いくつかのテストケースは成功しませんでした。

各テストケースは、ステータスを返します。テストケースのステータスは、次のいずれかになります。

- SUCCEEDED: テストケースが成功しました。
- FAILED: テストケースが失敗しました。
- ERROR: テストケースで予期しないエラーが発生しました。
- SKIPPED: テストケースは実行されませんでした。
- UNKNOWN: テストケースが、SUCCEEDED、FAILED、ERROR、SKIPPED 以外のステータスを返しました。

テストレポートには、最大 500 件のテストケース結果を設定できます。500 を超えるテストケースが実行された場合、 は ステータスのテストに CodeBuild 優先順位を付けFAILED、テストケースの結果を切り捨てます。

テストレポートのアクセス許可の使用

このトピックでは、テストレポートに関連するアクセス権限に関する重要な情報について説明します。

トピック

- テストレポートのロールの作成
- テストレポートオペレーションのアクセス許可
- テストレポートのアクセス許可の例

テストレポートのロールの作成

テストレポートを実行し、テストレポートを含めるようにプロジェクトを更新するには、IAM ロールに以下のアクセス権限が必要です。これらのアクセス許可は、定義済みの AWS 管理ポリシーに含まれています。既存のビルドプロジェクトにテストレポートを追加する場合は、これらのアクセス権限を自分で追加する必要があります。

- `CreateReportGroup`
- `CreateReport`
- `UpdateReport`
- `BatchPutTestCases`

コードカバレッジレポートを実行するには、IAM ロールに `BatchPutCodeCoverages` アクセス許可が付与されている必要もあります。

> **Note**
>
> `BatchPutTestCases`、`CreateReport`、`UpdateReport`、および `BatchPutCodeCoverages` はパブリック権限ではありません。これらのアクセス許可に対して、対応する AWS CLI コマンドまたは SDK メソッドを呼び出すことはできません。

これらのアクセス許可があることを確認するには、次のポリシーを IAM ロールにアタッチします。

```
{
    "Effect": "Allow",
    "Resource": [
        "*"
    ],
    "Action": [
        "codebuild:CreateReportGroup",
        "codebuild:CreateReport",
        "codebuild:UpdateReport",
        "codebuild:BatchPutTestCases",
        "codebuild:BatchPutCodeCoverages"
    ]
}
```

このポリシーは、使用する必要があるレポートグループだけに制限することをお勧めします。以下の例では、ポリシー内の 2 つの ARN を持つレポートグループのみにアクセス権限を制限します。

```
{
    "Effect": "Allow",
    "Resource": [
        "arn:aws:codebuild:your-region:your-aws-account-id:report-group/report-group-name-1",
        "arn:aws:codebuild:your-region:your-aws-account-id:report-group/report-group-name-2"
    ],
    "Action": [
        "codebuild:CreateReportGroup",
        "codebuild:CreateReport",
        "codebuild:UpdateReport",
        "codebuild:BatchPutTestCases",
        "codebuild:BatchPutCodeCoverages"
    ]
}
```

以下の例では、`my-project` という名前のプロジェクトのビルドを実行することによって作成されたレポートグループのみにアクセス権限を制限しています。

```
{
    "Effect": "Allow",
    "Resource": [
        "arn:aws:codebuild:your-region:your-aws-account-id:report-group/my-project-*"
    ],
    "Action": [
        "codebuild:CreateReportGroup",
        "codebuild:CreateReport",
        "codebuild:UpdateReport",
        "codebuild:BatchPutTestCases",
        "codebuild:BatchPutCodeCoverages"
    ]
}
```

> **Note**
>
> プロジェクトで指定された CodeBuild サービスロールは、S3 バケットにアップロードするアクセス許可に使用されます。

テストレポートオペレーションのアクセス許可

次のテストレポート CodeBuild API オペレーションのアクセス許可を指定できます。

- BatchGetReportGroups
- BatchGetReports
- CreateReportGroup
- DeleteReportGroup
- DeleteReport
- DescribeTestCases
- ListReportGroups
- ListReports
- ListReportsForReportGroup
- UpdateReportGroup

詳細については、「AWS CodeBuild アクセス許可リファレンス」を参照してください。

テストレポートのアクセス許可の例

テストレポートに関連するサンプルポリシーの詳細については、以下を参照してください。

- レポートグループの変更をユーザーに許可する
- レポートグループの作成をユーザーに許可する
- レポートの削除をユーザーに許可する
- レポートグループの削除をユーザーに許可する
- レポートグループに関する情報の取得をユーザーに許可する
- レポートに関する情報の取得をユーザーに許可する
- レポートグループの一覧表示をユーザーに許可する
- レポートの一覧表示をユーザーに許可する
- レポートグループのレポートの一覧表示をユーザーに許可する
- レポートのテストケースの一覧表示をユーザーに許可する

テストレポートの表示

テストケースに関する情報、合格番号と不合格番号、実行にかかった時間など、テストレポートに関する詳細を表示できます。ビルド実行、レポートグループ、または AWS アカウントごとにグループ化されたテストレポートを表示できます。コンソールでテストレポートを選択すると、テストケースの詳細と結果が表示されます。

期限切れでないテストレポートを表示できます。テストレポートは、作成から 30 日後に有効期限が切れます。で期限切れレポートを表示することはできません CodeBuild。

トピック

ビルドのテストレポートの表示

ビルドのテストレポートを表示するには

1. https://console.aws.amazon.com/codesuite/codebuild/home で AWS CodeBuild コンソールを開きます。
2. 表示するビルドを見つけます。テストレポートを作成したビルドを実行したプロジェクトがわかっている場合：

   1. ナビゲーションペインで、[Build projects (ビルドプロジェクト)] を選択し、表示するテストレポートを実行したビルドを含むプロジェクトを選択します。
   2. [Build history (ビルド履歴)] を選択し、表示するレポートを作成したビルドを選択します。

   AWS アカウントのビルド履歴でビルドを見つけることもできます。

   1. ナビゲーションペインで [Build history (ビルド履歴)] を選択し、表示するテストレポートを作成したビルドを選択します。
3. ビルドページで [Reports (レポート)] を選択し、テストレポートを選択して詳細を確認します。

レポートグループのテストレポートの表示

レポートグループ内のテストレポートを表示するには

1. https://console.aws.amazon.com/codesuite/codebuild/home で AWS CodeBuild コンソールを開きます。
2. ナビゲーションペインで、[Report groups (レポートグループ)] を選択します。
3. 表示するテストレポートを含むレポートグループを選択します。
4. テストレポートを選択すると、その詳細が表示されます。

AWS アカウントでのテストレポートの表示

AWS アカウントでテストレポートを表示するには

1. https://console.aws.amazon.com/codesuite/codebuild/home で AWS CodeBuild コンソールを開きます。
2. ナビゲーションペインで [Report history (レポート履歴)] を選択します。
3. テストレポートを選択すると、その詳細が表示されます。

テストフレームワークを使用したテストレポート

このセクションのトピックでは、さまざまなテストフレームワーク AWS CodeBuild のテストレポートを でセットアップする方法を示します。

トピック

- Jasmine によるテストレポートのセットアップ
- Jest によるテストレポートのセットアップ
- pytest によるテストレポートのセットアップ
- RSpec を使用したテストレポートのセットアップ

Jasmine によるテストレポートのセットアップ

次の手順では、JasmineBDD テスト フレームワークを用いた AWS CodeBuild でのテストレポートのセットアップ方法を示しています。

この手順には、次の前提条件が必要です。

- 既存の CodeBuild プロジェクトがある。
- そのプロジェクトは、Jasmine テストフレームワークを使用するようにセットアップされた Node.js プロジェクトである。

「jasmine-reporters」パッケージをdevDependencies セクションのpackage.json ファイルに追加します。このパッケージには、Jasmine で使用できる JavaScript レポータークラスのコレクションがあります。

```
npm install --save-dev jasmine-reporters
```

まだ存在しない場合は、test スクリプトをプロジェクトの package.json ファイルに追加します。test スクリプトは、npm test が実行されたときに Jasmine が確実に呼び出されるようにします。

```
{
  "scripts": {
    "test": "npx jasmine"
  }
}
```

CodeBuild は、以下の Jasmine テストレポーターをサポートしています。

JUnitXmlReporter

JunitXml 形式でレポートを生成するために使用されます。

NUnitXmlReporter

NunitXml 形式でレポートを生成するために使用されます。

Jasmine を使用する Node.js プロジェクトには、デフォルトで Jasmine 設定とテストスクリプトを含む spec サブディレクトリが作成されます。

JunitXML 形式でレポートを生成するように Jasmine を設定するには、テストに次のコードを追加して、JUnitXmlReporter レポーターをインスタンス化します。

```
var reporters = require('jasmine-reporters');
```

```
var junitReporter = new reporters.JUnitXmlReporter({
  savePath: <test report directory>,
  filePrefix: <report filename>,
  consolidateAll: true
});

jasmine.getEnv().addReporter(junitReporter);
```

NunitXML 形式でレポートを生成するように Jasmine を設定するには、テストに次のコードを追加して、NUnitXmlReporter レポーターをインスタンス化します。

```
var reporters = require('jasmine-reporters');

var nunitReporter = new reporters.NUnitXmlReporter({
  savePath: <test report directory>,
  filePrefix: <report filename>,
  consolidateAll: true
});

jasmine.getEnv().addReporter(nunitReporter)
```

テストレポートは、*<test report directory>*/*<report filename>* で指定されたファイルにエクスポートされます。

buildspec.yml ファイルで、次のセクションを追加/更新します。

```
version: 0.2

phases:
  pre_build:
    commands:
      - npm install
  build:
    commands:
      - npm build
      - npm test

reports:
  jasmine_reports:
    files:
      - <report filename>
    file-format: JUNITXML
```

```
    base-directory: <test report directory>
```

NunitXml レポート形式を使用している場合は、file-format 値を次のように変更します。

```
    file-format: NUNITXML
```

Jest によるテストレポートのセットアップ

次の手順では、Jest テスト フレームワークを用いた AWS CodeBuild でのテストレポートのセットアップ方法を示しています。

この手順には、次の前提条件が必要です。

- 既存の CodeBuild プロジェクトがある。
- そのプロジェクトは、Jest テストフレームワークを使用するようにセットアップされた Node.js プロジェクトである。

「jest-junit」パッケージをdevDependencies セクションのpackage.json ファイルに追加します。CodeBuild では、このパッケージを使用して、JunitXml の形式で設定します。

```
npm install --save-dev jest-junit
```

まだ存在しない場合は、test スクリプトをプロジェクトの package.json ファイルに追加します。test スクリプトは、npm test が実行されたときに Jest が確実に呼び出されるようにします。

```
{
  "scripts": {
    "test": "jest"
  }
}
```

Jest の設定ファイルに以下を追加して、JunitXml レポーターを使用するよう Jest を設定します。プロジェクトに Jest 設定ファイルがない場合は、プロジェクトのルートに jest.config.js という名前のファイルを作成し、以下を追加します。テストレポートは、*<test report directory>*/*<report filename>* で指定されたファイルにエクスポートされます。

```
module.exports = {
  reporters: [
```

```
    'default',
    [ 'jest-junit', {
      outputDirectory: <test report directory>,
      outputName: <report filename>,
    } ]
  ]
};
```

buildspec.yml ファイルで、次のセクションを追加/更新します。

```
version: 0.2

phases:
  pre_build:
    commands:
      - npm install
  build:
    commands:
      - npm build
      - npm test

reports:
  jest_reports:
    files:
      - <report filename>
    file-format: JUNITXML
    base-directory: <test report directory>
```

pytest によるテストレポートのセットアップ

次の手順では、pytest テスト フレームワークを用いた AWS CodeBuild でのテストレポートのセットアップ方法を示しています。

この手順には、次の前提条件が必要です。

- 既存の CodeBuild プロジェクトがある。
- そのプロジェクトは、pytest テストフレームワークを使用するようにセットアップされた Python プロジェクトである。

build ファイルの post_build または buildspec.yml フェーズに、次のエントリを追加します。このコードは、自動的に現在のディレクトリ内でテストを検出し、*<test report*

directory>/<*report filename*> で指定されたファイルにテストレポートをエクスポートします。レポートでは、JunitXml 形式が使用されます。

```
      - python -m pytest --junitxml=<test report directory>/<report filename>
```

buildspec.yml ファイルで、次のセクションを追加/更新します。

```
version: 0.2

phases:
  install:
    runtime-versions:
      python: 3.7
    commands:
      - pip3 install pytest
  build:
    commands:
      - python -m pytest --junitxml=<test report directory>/<report filename>

reports:
  pytest_reports:
    files:
      - <report filename>
    base-directory: <test report directory>
    file-format: JUNITXML
```

RSpec を使用したテストレポートのセットアップ

次の手順では、RSpec テスト フレームワークを用いた AWS CodeBuild でのテストレポートのセットアップ方法を示しています。

この手順には、次の前提条件が必要です。

- 既存の CodeBuild プロジェクトがある。
- そのプロジェクトは、RSpec テストフレームワークを使用するようにセットアップされた Ruby プロジェクトである。

buildspec.yml ファイルに以下を追加/更新します。このコードは、*<test source directory>* ディレクトリでテストを実行し、*<test report directory>*/*<report*

filename> で指定されたファイルにテストレポートをエクスポートします。レポートでは、JunitXml 形式が使用されます。

```
version: 0.2

phases:
  install:
    runtime-versions:
      ruby: 2.6
  pre_build:
    commands:
      - gem install rspec
      - gem install rspec_junit_formatter
  build:
    commands:
      - rspec <test source directory>/* --format RspecJunitFormatter --out <test report
 directory>/<report filename>
reports:
    rspec_reports:
        files:
            - <report filename>
        base-directory: <test report directory>
        file-format: JUNITXML
```

コードカバレッジレポート

CodeBuild では、テストのコードカバレッジレポートを生成できます。次のコードカバレッジレポートが用意されています。

ラインカバレッジ

ラインカバレッジは、テストがカバーするステートメントの数を測定します。ステートメントは、コメントや条件を含まない、単一の命令です。

line coverage = (total lines covered)/(total number of lines)

ブランチカバレッジ

ブランチカバレッジは、コントロール構造のすべてのブランチ内のテスト可能なブランチの数を測定します (「if」または「case」ステートメントなど)。

branch coverage = (total branches covered)/(total number of branches)

以下のコードカバレッジレポートファイル形式がサポートされています。

- JaCoCo XML
- SimpleCov JSON1
- クローバー XML
- Cobertura XML
- LCOV INFO

1 simplecov -json ではなく simplecov によって生成された JSON コードカバレッジレポート CodeBuild を受け入れます。

コードカバレッジレポートの作成

コードカバレッジレポートを作成するには、buildspec ファイルに少なくとも 1 つのコードカバレッジレポートグループで設定されたビルドプロジェクトを実行します。 CodeBuild はコードカバレッジ結果を解釈し、実行のコードカバレッジレポートを提供します。同じ buildspec ファイルを使用する後続のビルドごとに、新しいテストレポートが生成されます。

テストレポートを作成するには

1. ビルドプロジェクトを作成します。詳細については、 でのビルドプロジェクトの作成AWS CodeBuild を参照してください。
2. テストレポート情報を使用してプロジェクトの buildspec ファイルを設定します。

   a. `reports:` セクションを追加し、レポートグループの名前を指定します。 は、プロジェクト名と `project-name-` 形式で指定した名前を使用してレポートグループ CodeBuild を作成します`report-group-name-in-buildspec`。使用するレポートグループがすでにある場合は、その ARN を指定します。ARN の代わりに名前 を使用すると、 は新しいレポートグループ CodeBuild を作成します。詳細については、「Reports syntax in the buildspec file」を参照してください。

   b. レポートグループで、コードカバレッジの結果を保存するファイルの場所を指定します。複数のレポートグループを使用する場合は、各レポートグループに対して結果ファイルの場所を指定します。ビルドプロジェクトを実行するたびに、新しいコードカバレッジが作成されます。詳細については、「テストファイルの指定」を参照してください。

      これは test- にある XML 結果ファイルのコードカバレッジレポート JaCoCoを生成する例です`results/jacoco-coverage-report.xml`。

```
reports:
  jacoco-report:
    files:
      - 'test-results/jacoco-coverage-report.xml'
    file-format: 'JACOCOXML'
```

c. 「build」または「post_build」シーケンスの「commands」セクションで、コードカバレッジ分析を実行するコマンドを指定します。詳細については、「テストコマンドの指定」を参照してください。

3. ビルドプロジェクトのビルドを実行します。詳細については、「AWS CodeBuild でのビルドの実行」を参照してください。
4. ビルドが完了したら、プロジェクトページの [Build history (ビルド 履歴)] から新しいビルド実行を選択します。レポートを選択して、コードカバレッジレポートを表示します。詳細については、「ビルドのテストレポートの表示」を参照してください。

自動検出のレポート

自動検出を使用すると、 はビルドフェーズの完了後にすべてのビルドファイル CodeBuild を検索し、サポートされているレポートファイルタイプを検索し、新しいテストおよびコードカバレッジレポートグループとレポートを自動的に作成します。検出されたレポートタイプについて、 は次のパターンで新しいレポートグループ CodeBuild を作成します。

```
<project-name>-<report-file-format>-AutoDiscovered
```

> **Note**
>
> 検出されたレポートファイルの形式タイプが同じ場合、それらは同じレポートグループまたはレポートに配置されます。

レポートの自動検出は、プロジェクト環境変数によって設定されます。

CODEBUILD_CONFIG_AUTO_DISCOVER

この変数は、ビルド中にレポートの自動検出を無効にするかどうかを決定します。デフォルトでは、レポートの自動検出はすべてのビルドで有効になっています。この機能を無効にするには、CODEBUILD_CONFIG_AUTO_DISCOVERを に設定しますfalse。

`CODEBUILD_CONFIG_AUTO_DISCOVER_DIR`

(オプション) この変数は、 が潜在的なレポートファイル CodeBuild を検索する場所を決定します。デフォルトでは、 は**/*デフォルトで CodeBuild を検索することに注意してください。

これらの環境変数は、ビルドフェーズで変更できます。例えば、maingit ブランチのビルドに対してレポート自動検出のみを有効にする場合は、ビルドプロセス中に git mainブランチをチェックし、ビルドがブランチにない場合は false CODEBUILD_CONFIG_AUTO_DISCOVERに設定できます。レポートの自動検出は、コンソールまたはプロジェクト環境変数を使用して無効にできます。

トピック

- コンソールを使用してレポートの自動検出を設定する
- プロジェクト環境変数を使用してレポートの自動検出を設定する

コンソールを使用してレポートの自動検出を設定する

コンソールを使用してレポートの自動検出を設定するには

1. ビルドプロジェクトを作成するか、編集するビルドプロジェクトを選択します。詳細については、「 でのビルドプロジェクトの作成AWS CodeBuild」または「AWS CodeBuild でのビルドプロジェクトの設定の変更」を参照してください。
2. 環境 で、追加設定 を選択します。
3. レポートの自動検出を無効にするには、レポート自動検出 でレポート自動検出 を無効にする を選択します。
4. (オプション) Auto-discover ディレクトリ - オプション で、 CodeBuild のディレクトリパターンを入力して、サポートされているレポート形式のファイルを検索します。は CodeBuild **/*デフォルトで を検索します。

プロジェクト環境変数を使用してレポートの自動検出を設定する

プロジェクト環境変数を使用してレポートの自動検出を設定するには

1. ビルドプロジェクトを作成するか、編集するビルドプロジェクトを選択します。詳細については、「 でのビルドプロジェクトの作成AWS CodeBuild」または「AWS CodeBuild でのビルドプロジェクトの設定の変更」を参照してください。
2. 環境変数 で、次の操作を行います。

a. レポートの自動検出を無効にするには、名前 に **CODEBUILD_CONFIG_AUTO_DISCOVER**を入力し、値 に を入力します**false**。これにより、レポートの自動検出が無効になります。

b. (オプション) 名前 に **CODEBUILD_CONFIG_AUTO_DISCOVER_DIR**を入力し、値 に を入力します。 CodeBuild がサポートされているレポート形式のファイルを検索するディレクトリを入力します。例えば、 は output ディレクトリ内の.xmlファイルoutput/*xmlを検索します。

でのログ記録とモニタリングAWS CodeBuild

モニタリングは、AWS CodeBuild と AWS ソリューションの信頼性、可用性、パフォーマンスを維持する上で重要な部分です。マルチポイント障害が発生した場合は、その障害をより簡単にデバッグできるように、AWS ソリューションのすべての部分からモニタリングデータを収集する必要があります。AWS には、CodeBuild リソースとビルドをモニタリングし、潜在的なインシデントに対応するための以下のツールが用意されています。

トピック

AWS CodeBuild による AWS CloudTrail API 呼び出しのログ記録

AWS CodeBuild は、ユーザー、ロール、または CodeBuild の AWS のサービスによって実行されたアクションを記録するサービスである AWS CloudTrail と統合されています。CloudTrail は、 スタックコンソールからの呼び出しや、 スタック API へのコード呼び出しを含む、 スタックのすべての API コールをイベントとしてキャプチャします。証跡を作成する場合は、CodeBuild のイベントなど、S3 バケットへの CloudTrail イベントの継続的な配信を有効にすることができます。追跡を設定しない場合でも、CloudTrail コンソールの [Event history] (イベント履歴) で最新のイベントを表示できます。CloudTrail で収集された情報を使用して、CodeBuild に対するリクエスト、リクエスト元の IP アドレス、リクエスト者、リクエスト日時などの詳細を確認できます。

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

CloudTrail の AWS CodeBuild 情報

CloudTrail は、アカウントを作成すると AWS アカウントで有効になります。CodeBuild でアクティビティが発生すると、そのアクティビティは [Event history] (イベント履歴) の他の AWS のサービスイベントと共に CloudTrail イベントに記録されます。最近のイベントは、AWSアカウントで表示、検索、ダウンロードできます。詳細については、「AWS CloudTrail ユーザーガイド」の「Viewing events with CloudTrail event history」(CloudTrail イベント履歴でのイベントの表示) を参照してください。

CodeBuild のイベントなど、AWS アカウントのイベントの継続的な記録については、証跡を作成します。証跡により、CloudTrail はログファイルを S3 バケットに配信できます。デフォルトでは、

コンソールで証跡を作成するときに、証跡がすべてのリージョンに適用されます。証跡では、AWS パーティションのすべてのリージョンからのイベントがログに記録され、指定した S3 バケットにログファイルが配信されます。その他の AWS のサービスを設定して、CloudTrail ログで収集されたデータをより詳細に分析し、それに基づく対応を行うことができます。詳細については、以下を参照してください。

- 証跡を作成するための概要
- CloudTrail がサポートされているサービスと統合
- CloudTrail の Amazon SNS 通知の設定
- 複数のリージョンから CloudTrail ログファイルを受け取るおよび複数のアカウントから CloudTrail ログファイルを受け取る

すべての CodeBuild アクションは CloudTrail が記録します。これらのアクションは CodeBuild API リファレンスで説明されています。たとえば、`CreateProject` (AWS CLI で `create-project`)、`StartBuild` (AWS CLI で `start-project`)、`UpdateProject` (AWS CLI で `update-project`) の各アクションを呼び出すと、CloudTrail のログファイルにエントリが生成されます。

各イベントまたはログエントリには、リクエストの生成者に関する情報が含まれます。同一性情報は次の判断に役立ちます。

- リクエストが、ルートと ユーザー認証情報のどちらを使用して送信されたか。
- リクエストが、ロールとフェデレーティッドユーザーのどちらの一時的なセキュリティ認証情報を使用して送信されたか.
- リクエストが、別の AWS のサービスによって送信されたかどうか。

詳細については、AWS CloudTrail ユーザーガイドの CloudTrail userIdentity エレメントを参照してください。

AWS CodeBuild ログファイルエントリの概要

証跡は、指定した S3 バケットにイベントをログファイルとして配信するように設定できます。CloudTrail ログファイルには、1 つ以上のログエントリがあります。イベントはあらゆるソースからの単一のリクエストを表し、リクエストされたアクション、アクションの日時、リクエストのパラメータなどの情報が含まれます。CloudTrail ログファイルは、パブリック API コールの順序付けられたスタックトレースではないため、特定の順序では表示されません。

> **Note**
>
> 機密情報を保護するために、CodeBuild ログでは次の情報が非表示になっています。
>
> - AWS アクセスキー ID。詳細については、AWS Identity and Access Management ユーザーガイドの IAM ユーザーのアクセスキーの管理を参照してください。
> - パラメータストアを使用して指定された文字列。詳細については、「Amazon EC2 Systems Manager ユーザーガイド」の「Systems Manager パラメータストア」および「Systems Manager パラメータストアコンソールのチュートリアル」を参照してください。
> - AWS Secrets Manager を使用して指定された文字列。詳細については、「キー管理」を参照してください。

次の例は、CodeBuild でビルドプロジェクトを作成する方法を示す CloudTrail ログエントリを示しています。

```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "FederatedUser",
    "principalId": "account-ID:user-name",
    "arn": "arn:aws:sts::account-ID:federated-user/user-name",
    "accountId": "account-ID",
    "accessKeyId": "access-key-ID",
    "sessionContext": {
      "attributes": {
        "mfaAuthenticated": "false",
        "creationDate": "2016-09-06T17:59:10Z"
      },
      "sessionIssuer": {
        "type": "IAMUser",
        "principalId": "access-key-ID",
        "arn": "arn:aws:iam::account-ID:user/user-name",
        "accountId": "account-ID",
        "userName": "user-name"
      }
    }
  },
  "eventTime": "2016-09-06T17:59:11Z",
  "eventSource": "codebuild.amazonaws.com",
```

```
  "eventName": "CreateProject",
  "awsRegion": "region-ID",
  "sourceIPAddress": "127.0.0.1",
  "userAgent": "user-agent",
  "requestParameters": {
    "awsActId": "account-ID"
  },
  "responseElements": {
    "project": {
      "environment": {
        "image": "image-ID",
        "computeType": "BUILD_GENERAL1_SMALL",
        "type": "LINUX_CONTAINER",
        "environmentVariables": []
      },
      "name": "codebuild-demo-project",
      "description": "This is my demo project",
      "arn": "arn:aws:codebuild:region-ID:account-ID:project/codebuild-demo-
project:project-ID",
      "encryptionKey": "arn:aws:kms:region-ID:key-ID",
      "timeoutInMinutes": 10,
      "artifacts": {
        "location": "arn:aws:s3:::codebuild-region-ID-account-ID-output-bucket",
        "type": "S3",
        "packaging": "ZIP",
        "outputName": "MyOutputArtifact.zip"
      },
      "serviceRole": "arn:aws:iam::account-ID:role/CodeBuildServiceRole",
      "lastModified": "Sep 6, 2016 10:59:11 AM",
      "source": {
        "type": "GITHUB",
        "location": "https://github.com/my-repo.git"
      },
      "created": "Sep 6, 2016 10:59:11 AM"
    }
  },
  "requestID": "9d32b228-745b-11e6-98bb-23b67EXAMPLE",
  "eventID": "581f7dd1-8d2e-40b0-aeee-0dbf7EXAMPLE",
  "eventType": "AwsApiCall",
  "recipientAccountId": "account-ID"
}
```

のモニタリングAWS CodeBuild

Amazon CloudWatch を使用してビルドをモニタリングし、異常が発生した報告して、必要に応じて対応策を取ることができます。ビルドは、次の 2 つのレベルでモニタリングできます。

プロジェクトレベル

これらのメトリクスは、指定したプロジェクトのすべてのビルドが対象となります。プロジェクトのメトリクスを表示するには、CloudWatch でディメンションとして `ProjectName` を指定します。

AWS アカウント・レベル

これらのメトリクスは、1 つのアカウントのすべてのビルドが対象となります。AWS アカウントレベルでメトリクスを表示するには、CloudWatch でディメンションを入力しないでください。ビルドリソース使用率のメトリクスは、AWS アカウントレベルでは使用できません。

CloudWatch メトリクスには、一定期間におけるビルドの動作が示されます。たとえば、以下のことをモニタリングできます。

- ビルドプロジェクトまたは AWS アカウントで一定期間に試みられたビルドの数。
- ビルドプロジェクトまたは AWS アカウントで一定期間に成功したビルドの数。
- ビルドプロジェクトまたは AWS アカウントで一定期間に失敗したビルドの数。
- ビルドプロジェクトまたは AWS アカウントで一定期間に CodeBuild がビルドの実行に費やした時間。
- ビルドまたはビルドプロジェクト全体のリソース使用率を構築します。CPU、メモリ、ストレージ使用率などのリソース使用率メトリクスを構築します。

詳細については、「CodeBuild メトリクスのモニタリング」を参照してください。

CodeBuild CloudWatch メトリクス

以下のメトリクスは、AWS アカウントまたはビルドプロジェクトごとに追跡できます。

BuildDuration

ビルドの `BUILD` フェーズの所要時間を測定します。

単位: 秒

有効な CloudWatch 統計: Average (推奨)、Maximum、Minimum

ビルド数

トリガーされたビルドの数を測定します。

単位: Count (個)

有効な CloudWatch 統計: Sum

DownloadSourceDuration

ビルドの DOWNLOAD_SOURCE フェーズの所要時間を測定します。

単位: 秒

有効な CloudWatch 統計: Average (推奨)、Maximum、Minimum

Duration

一定期間におけるすべてのビルドの所要時間を測定します。

単位: 秒

有効な CloudWatch 統計: Average (推奨)、Maximum、Minimum

FailedBuilds

クライアントエラーまたはタイムアウトのために失敗したビルドの数を測定します。

単位: Count (個)

有効な CloudWatch 統計: Sum

FinalizingDuration

ビルドの FINALIZING フェーズの所要時間を測定します。

単位: 秒

有効な CloudWatch 統計: Average (推奨)、Maximum、Minimum

InstallDuration

ビルドの INSTALL フェーズの所要時間を測定します。

単位: 秒

有効な CloudWatch 統計: Average (推奨)、Maximum、Minimum

PostBuildDuration

ビルドの POST_BUILD フェーズの所要時間を測定します。

単位: 秒

有効な CloudWatch 統計: Average (推奨)、Maximum、Minimum

PreBuildDuration

ビルドの PRE_BUILD フェーズの所要時間を測定します。

単位: 秒

有効な CloudWatch 統計: Average (推奨)、Maximum、Minimum

ProvisioningDuration

ビルドの PROVISIONING フェーズの所要時間を測定します。

単位: 秒

有効な CloudWatch 統計: Average (推奨)、Maximum、Minimum

QueuedDuration

ビルドの QUEUED フェーズの所要時間を測定します。

単位: 秒

有効な CloudWatch 統計: Average (推奨)、Maximum、Minimum

SubmittedDuration

ビルドの SUBMITTED フェーズの所要時間を測定します。

単位: 秒

有効な CloudWatch 統計: Average (推奨)、Maximum、Minimum

SucceededBuilds

成功したビルドの数を測定します。

単位: Count (個)

有効な CloudWatch 統計: Sum

UploadArtifactsDuration

ビルドの UPLOAD_ARTIFACTS フェーズの所要時間を測定します。

単位: 秒

有効な CloudWatch 統計: Average (推奨)、Maximum、Minimum

CodeBuild CloudWatch リソース使用率メトリックス

> **Note**
>
> CodeBuild リソース使用率メトリクスは、以下のリージョンでのみ利用可能です。
>
> - Asia Pacific (Tokyo) Region
> - Asia Pacific (Seoul) Region
> - Asia Pacific (Mumbai) Region
> - Asia Pacific (Singapore) Region
> - Asia Pacific (Sydney) Region
> - Canada (Central) Region
> - Europe (Frankfurt) Region
> - 欧州 (アイルランド) リージョン
> - 欧州 (ロンドン) リージョン
> - 欧州 (パリ) リージョン
> - South America (São Paulo) Region
> - 米国東部 (バージニア北部) リージョン
> - US East (Ohio) Region
> - 米国西部 (北カリフォルニア) リージョン
> - 米国西部 (オレゴン) リージョン

次のリソース使用率メトリックを記録できます。

CPUUtilized

ビルドコンテナで使用されている、割り当てられた処理の CPU ユニットの数。

単位: CPU 単位

有効な CloudWatch 統計: Average (推奨)、Maximum、Minimum

CPUUtilizedPercent

ビルドコンテナによって使用される割り当てられた処理の割合。

単位: パーセント

有効な CloudWatch 統計: Average (推奨)、Maximum、Minimum

MemoryUtilized

ビルドコンテナで使用されるメモリのメガバイト数。

単位: メガバイト

有効な CloudWatch 統計: Average (推奨)、Maximum、Minimum

MemoryUtilizedPercent

ビルドコンテナで使用されている、割り当てられたメモリの割合。

単位: パーセント

有効な CloudWatch 統計: Average (推奨)、Maximum、Minimum

StorageReadBytes

ビルドコンテナーによって使用されるストレージの読み取り速度。

単位: バイト/秒

有効な CloudWatch 統計: Average (推奨)、Maximum、Minimum

StorageWriteBytes

ビルドコンテナによって使用されるストレージ書き込み速度。

単位: バイト/秒

有効な CloudWatch 統計: Average (推奨)、Maximum、Minimum

CodeBuild CloudWatch のディメンション

CodeBuild には、以下の CloudWatch メトリクスディメンションが用意されています。これらを指定しない場合、現在の AWS アカウントのメトリクスとなります。

BuildId、BuildNumber、ProjectName

メトリクスは、ビルド識別子、ビルド番号、およびプロジェクト名に対して提供されます。

ProjectName

プロジェクト名には、メトリクスが提供されます。

CodeBuild CloudWatch アラーム

CloudWatch コンソールを使用してCodeBuild メトリクスに基づいてアラームを作成できるため、ビルドで問題が発生した場合に対応できます。アラームで最も役に立つ 2 つのメトリクスは次のとおりです。

- `FailedBuild`。事前に設定した秒数内に失敗したビルドが一定数検出されたときにトリガーされるアラームを作成できます。CloudWatch で、秒数と、アラームをトリガーするための失敗したビルド数を指定します。
- `Duration`。ビルドに予想より時間がかかったときにトリガーするアラームを作成できます。ビルドの開始からビルドの完了までの経過所要時間を指定します。この時間を超えるとアラームがトリガーされます。

CodeBuild メトリクスのアラームを作成する方法については、「CloudWatch アラームによるビルドのモニタリング」を参照してください。CloudWatch アラームと、状態の変更の詳細については、Amazon CloudWatch ユーザーガイドの「Amazon CloudWatch アラームの作成」を参照してください。

CodeBuild メトリクスのモニタリング

AWS CodeBuild はお客様の代わりに自動的に関数をモニタリングし、Amazon CloudWatch を通じてメトリクスを報告します。これらのメトリクスには、ビルドの合計数、失敗したビルドの数、成功したビルドの数、ビルドの所要時間が含まれます。

CodeBuild コンソールまたは CloudWatch コンソールを使用することで、CodeBuild のメトリクスをモニタリングできます。次の手順は、メトリクスにアクセスにする方法を示しています。

トピック

- ビルドメトリクスにアクセスする (CodeBuild コンソール)
- ビルドメトリクスにアクセスする (Amazon CloudWatch コンソール)

ビルドメトリクスにアクセスする (CodeBuild コンソール)

> **Note**
>
> CodeBuild コンソールで、表示に使用されるメトリクスまたはグラフをカスタマイズすることはできません。表示をカスタマイズする場合は、Amazon CloudWatch コンソールを使用して設定されたビルドメトリクスを表示します。

アカウントレベルのメトリクス

AWS アカウントレベルのメトリクスにアクセスするには

1. AWS Management Console にサインインして、AWS CodeBuild コンソール (https://console.aws.amazon.com/codesuite/codebuild/home) を開きます。
2. ナビゲーションペインで [Account metrics (アカウントメトリクス)] を選択します。

プロジェクトレベルのメトリクス

プロジェクトレベルのメトリクスにアクセスするには

1. AWS Management Console にサインインして、AWS CodeBuild コンソール (https://console.aws.amazon.com/codesuite/codebuild/home) を開きます。
2. ナビゲーションペインで、[Build projects] を選択します。
3. ビルドプロジェクトのリストの [名前] 列で、メトリクスを表示するプロジェクトを選択します。
4. [Metrics] タブを選択します。

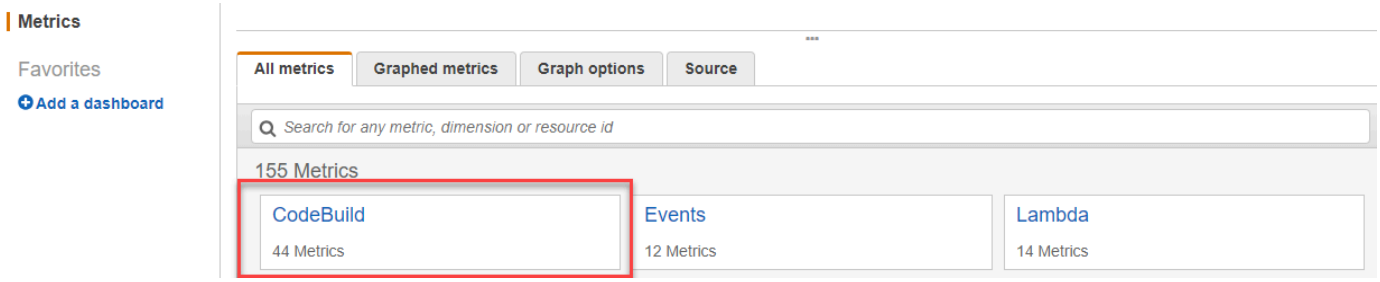
ビルドメトリクスにアクセスする (Amazon CloudWatch コンソール)

CloudWatch コンソールでの表示に使用されるメトリクスまたはグラフをカスタマイズすることはできます。

アカウントレベルのメトリクス

アカウントレベルのメトリクスにアクセスするには

1. AWS Management Console にサインインして、CloudWatch コンソール (https://console.aws.amazon.com/cloudwatch/) を開きます。
2. ナビゲーションペインで [Metrics (メトリクス)] を選択します。
3. [All metrics (すべてのメトリクス)] タブで、[CodeBuild] を選択します。

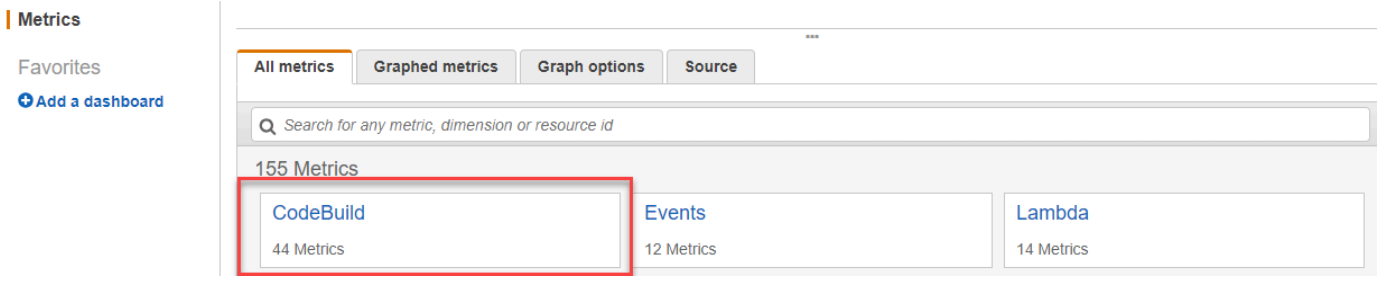


4. [アカウントメトリクス] を選択します。
5. プロジェクトとメトリクスを 1 つ以上選択します。プロジェクトごとに、[SucceededBuilds]、[FailedBuilds]、[Builds]、[Duration] の各メトリクスを選択できます。選択されたすべてのプロジェクトとメトリクスの組み合わせが、ページのグラフに表示されます。

プロジェクトレベルのメトリクス

プロジェクトレベルのメトリクスにアクセスするには

1. AWS Management Console にサインインして、CloudWatch コンソール (https://console.aws.amazon.com/cloudwatch/) を開きます。
2. ナビゲーションペインで [Metrics (メトリクス)] を選択します。
3. [All metrics (すべてのメトリクス)] タブで、[CodeBuild] を選択します。



4. [By Project (プロジェクト別)] を選択します。

5. プロジェクトとメトリクスの組み合わせを 1 つ以上選択します。プロジェクトごとに、[SucceededBuilds]、[FailedBuilds]、[Builds]、[Duration] の各メトリクスを選択できます。選択されたすべてのプロジェクトとメトリクスの組み合わせが、ページのグラフに表示されます。
6. (オプション) メトリクスとグラフをカスタマイズすることができます。たとえば、[Statistic (統計)] 列のドロップダウンリストから、表示する別の統計を選択できます。または、[Period (期間)] 列のドロップダウンメニューから、メトリクスのモニタリングに使用する別の期間を選択できます。

   詳細については、Amazon CloudWatch ユーザーガイドの「グラフメトリクス」および「使用可能なメトリクスの表示」を参照してください。

CodeBuild リソース使用率メトリクスのモニタリング

AWS CodeBuild はお客様の代わりに自動的にビルドリソースをモニタリングし、Amazon CloudWatch を通じてメトリクスを報告します。これには、CPU、メモリ、ストレージ使用率などのメトリクスが含まれます。

> **Note**
>
> CodeBuild リソース使用率メトリクスは、1 分以上実行されるビルドに対してのみ記録されます。

CodeBuild コンソールまたは CloudWatch コンソールを使用して、CodeBuild のリソース使用率メトリクスをモニタリングできます。

> **Note**
>
> CodeBuild リソース使用率メトリクスは、以下のリージョンでのみ利用可能です。
>
> - Asia Pacific (Tokyo) Region
> - Asia Pacific (Seoul) Region
> - Asia Pacific (Mumbai) Region
> - Asia Pacific (Singapore) Region
> - Asia Pacific (Sydney) Region
> - Canada (Central) Region

- Europe (Frankfurt) Region
- 欧州 (アイルランド) リージョン
- 欧州 (ロンドン) リージョン
- 欧州 (パリ) リージョン
- South America (São Paulo) Region
- 米国東部 (バージニア北部) リージョン
- US East (Ohio) Region
- 米国西部 (北カリフォルニア) リージョン
- 米国西部 (オレゴン) リージョン

次の手順は、リソース使用率メトリクスにアクセスする方法を示しています。

トピック

- リソース使用率メトリクスへのアクセス (CodeBuild コンソール)
- リソース使用率メトリクスへのアクセス (Amazon CloudWatch コンソール)

リソース使用率メトリクスへのアクセス (CodeBuild コンソール)

> **Note**
>
> CodeBuild コンソールで、表示に使用されるメトリクスまたはグラフをカスタマイズすることはできません。表示をカスタマイズする場合は、Amazon CloudWatch コンソールを使用して設定されたビルドメトリクスを表示します。

プロジェクトレベルのリソース使用率メトリクス

プロジェクトレベルのリソース使用率メトリクスにアクセスするには

1. AWS Management Console にサインインして、AWS CodeBuild コンソール (https://console.aws.amazon.com/codesuite/codebuild/home) を開きます。
2. ナビゲーションペインで、[Build projects] を選択します。
3. ビルドプロジェクトのリストの [名前] 列で、使用率メトリクスを表示するプロジェクトを選択します。

4. [Metrics] タブを選択します。リソース使用率のメトリックは、リソース使用率メトリクスセクションに表示されます。
5. CloudWatch コンソールにプロジェクトレベルのリソース使用率のメトリクスを表示するには、リソース使用率メトリクスセクションで [CloudWatch で表示 (View in CloudWatch)] を選択します。

ビルドレベルのリソース使用率メトリクス

ビルドレベルのリソース使用率メトリクスにアクセスするには

1. AWS Management Console にサインインして、AWS CodeBuild コンソール (https://console.aws.amazon.com/codesuite/codebuild/home) を開きます。
2. ナビゲーションペインで、[Build history] を選択します。
3. ビルドのリストでは、実行のビルド (Build run) 列で、使用率メトリクスを表示するビルドを選択します。
4. [リソース使用率]タブを選択します。
5. CloudWatch コンソールにビルドレベルのリソース使用率のメトリクスを表示するには、リソース使用率メトリクスセクションで [CloudWatch で表示 (View in CloudWatch)] を選択します。

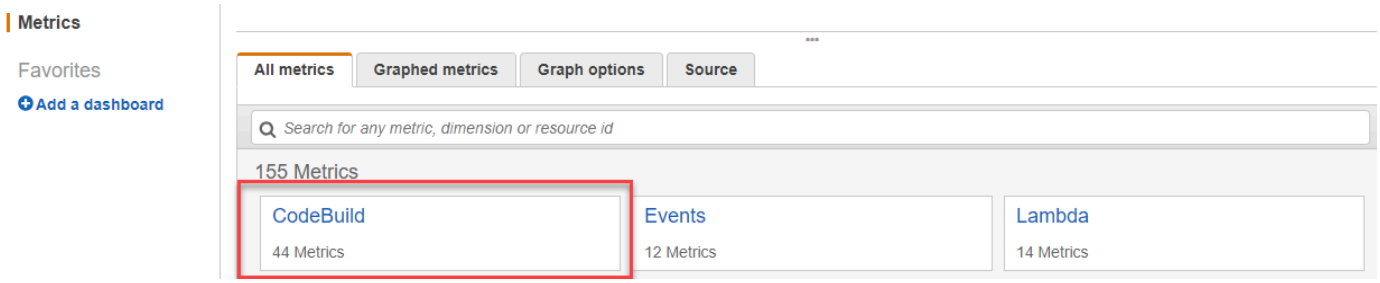
リソース使用率メトリクスへのアクセス (Amazon CloudWatch コンソール)

Amazon CloudWatch コンソールを使用して、CodeBuild リソース使用率メトリクスにアクセスできます。

プロジェクトレベルのリソース使用率メトリクス

プロジェクトレベルのリソース使用率メトリクスにアクセスするには

1. AWS Management Console にサインインして、CloudWatch コンソール (https://console.aws.amazon.com/cloudwatch/) を開きます。
2. ナビゲーションペインで [Metrics (メトリクス)] を選択します。
3. [All metrics (すべてのメトリクス)] タブで、[CodeBuild] を選択します。



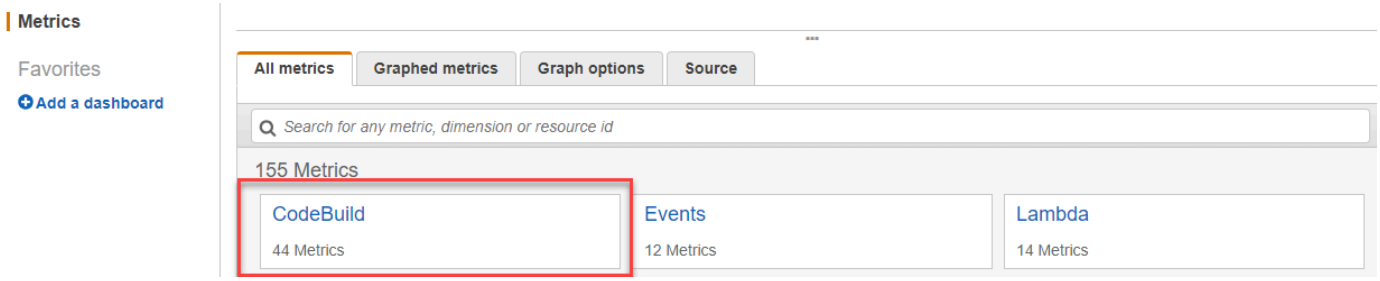
4. [By Project (プロジェクト別)] を選択します。
5. グラフに追加するプロジェクトとメトリクスの組み合わせを 1 つ以上選択します。選択されたすべてのプロジェクトとメトリクスの組み合わせが、ページのグラフに表示されます。
6. （オプション）[Graphed Metrics (グラフ化したメトリクス)] タブからメトリックスとグラフをカスタマイズできます。たとえば、[Statistic (統計)] 列のドロップダウンリストから、表示する別の統計を選択できます。または、[Period (期間)] 列のドロップダウンメニューから、メトリクスのモニタリングに使用する別の期間を選択できます。

   詳細については、Amazon CloudWatch ユーザーガイドの「グラフメトリクス」および「使用可能なメトリクスの表示」を参照してください。

ビルドレベルのリソース使用率メトリクス

ビルドレベルのリソース使用率メトリクスにアクセスするには

1. AWS Management Console にサインインして、CloudWatch コンソール (https://console.aws.amazon.com/cloudwatch/) を開きます。
2. ナビゲーションペインで [Metrics (メトリクス)] を選択します。
3. [All metrics (すべてのメトリクス)] タブで、[CodeBuild] を選択します。



4. BuildId、BuildNumber、ProjectName を選択。
5. グラフに追加するビルドとメトリクスの組み合わせを 1 つ以上選択します。選択されたすべてのビルドとメトリクスの組み合わせが、ページのグラフに表示されます。

6. （オプション）[Graphed Metrics (グラフ化したメトリクス)] タブからメトリックスとグラフをカスタマイズできます。たとえば、[Statistic (統計)] 列のドロップダウンリストから、表示する別の統計を選択できます。または、[Period (期間)] 列のドロップダウンメニューから、メトリクスのモニタリングに使用する別の期間を選択できます。

   詳細については、Amazon CloudWatch ユーザーガイドの「グラフメトリクス」および「使用可能なメトリクスの表示」を参照してください。

CloudWatch アラームによるビルドのモニタリング

ビルドの CloudWatch アラームを作成できます。アラームは、指定した期間にわたって 1 つのメトリクスをモニタリングし、複数期間にわたる指定しきい値との比較結果に基づいて 1 つ以上のアクションを実行します。ネイティブ CloudWatch アラーム機能を使用して、しきい値を超えたときに CloudWatch にサポートされる任意のアクションを指定できます。たとえば、15 分以内にアカウントで 3 つ以上のビルドが失敗したときに Amazon SNS 通知が送信されるように指定できます。

CodeBuild メトリクスの CloudWatch アラームを作成するには

1. AWS Management Console にサインインして、CloudWatch コンソール (https://console.aws.amazon.com/cloudwatch/) を開きます。
2. ナビゲーションペインで、[Alarms] を選択します。
3. [Create Alarm (アラーム作成)] を選択します。
4. [CloudWatch Metrics by Category (カテゴリ別の CloudWatch メトリクス)] で、[CodeBuild Metrics (CodeBuild メトリクス)] を選択します。プロジェクトレベルのメトリクスのみでよいことがわかっている場合は、[By Project (プロジェクト別)] を選択します。アカウントレベルのメトリクスのみでよいことがわかっている場合は、[Account Metrics (アカウントメトリクス)] を選択します。
5. [Create Alarm (アラームの作成)] で、まだ選択されていない場合は [Select Metric (メトリクスの選択)] を選択します。
6. アラームを作成する対象のメトリクスを選択します。オプションは [By Project (プロジェクト別)] または [Account Metrics (アカウントメトリクス)] です。
7. [Next (次へ)] または [Define Alarm (アラームの定義)] を選択し、アラームを作成します。詳細については、「Amazon CloudWatch ユーザーガイド」の「Amazon CloudWatch アラームの作成」を参照してください。アラームがトリガーされたときのAmazon SNS 通知の設定の詳細については、Amazon SNS 開発者ガイドの「Amazon SNS　通知設定」を参照してください。
8. [Create Alarm] を選択します。

のセキュリティ AWS CodeBuild

のクラウドセキュリティが最優先事項 AWS です。 AWS のお客様は、セキュリティを最も重視する組織の要件を満たす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から利点を得られます。

セキュリティとコンプライアンスは、 AWS とユーザー間で共有される責任です。この共有モデルは、ホストオペレーティングシステムや仮想化レイヤーからサービス施設の物理セキュリティまで、コンポーネントを AWS 運用、管理、制御するなど、運用上の負担を軽減するのに役立ちます。お客様は、ゲストオペレーティングシステム (更新やセキュリティパッチなど) とその他の関連アプリケーションソフトウェアの管理責任を負います。また、 AWS 提供されたセキュリティグループファイアウォールの設定も担当します。お客様の責任は、利用するサービス、お客様の IT 環境へのこれらのサービスの統合、適用される法律および規制によって異なります。したがって、貴社で使用するサービスについて注意深く検討してください。詳細については、「責任共有モデル」を参照してください。

CodeBuild リソースを保護する方法については、以下のトピックを参照してください。

トピック

- でのデータ保護 AWS CodeBuild
- での Identity and Access Management AWS CodeBuild
- のコンプライアンス検証 AWS CodeBuild
- の耐障害性 AWS CodeBuild
- のインフラストラクチャセキュリティ AWS CodeBuild
- でソースプロバイダーにアクセスする CodeBuild
- サービス間での不分別な代理処理の防止

でのデータ保護 AWS CodeBuild

AWS 責任共有モデル、 でのデータ保護に適用されます AWS CodeBuild。このモデルで説明したように、 AWS は、すべての 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担います AWS クラウド。お客様は、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。また、使用する AWS のサービス のセキュリティ設定と管理タスクもユーザーの責任となります。データプライバシーの詳細については、「データプライバシーのよくあ

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記事「AWS 責任共有モデルおよび GDPR」を参照してください。

データ保護の目的で、 認証情報を保護し AWS アカウント 、 AWS IAM Identity Center または AWS Identity and Access Management (IAM) を使用して個々のユーザーを設定することをお勧めします。この方法により、それぞれのジョブを遂行するために必要な権限のみが各ユーザーに付与されます。また、次の方法でデータを保護することもお勧めします:

- 各アカウントで多要素認証 (MFA) を使用します。
- SSL/TLS を使用して AWS リソースと通信します。TLS 1.2 は必須であり TLS 1.3 がお勧めです。
- で API とユーザーアクティビティのログ記録を設定します AWS CloudTrail。
- AWS 暗号化ソリューションを、 内のすべてのデフォルトのセキュリティコントロールとともに使用します AWS のサービス。
- 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 AWS を介して 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極秘または機密情報は、タグ、または名前フィールドなどの自由形式のテキストフィールドに配置しないことを強くお勧めします。これは、コンソール CodeBuild 、API、または SDK で AWS CLIまたは他の AWS のサービス を使用する場合も同様です。 AWS SDKs 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。外部サーバーへの URL を提供する場合は、そのサーバーへのリクエストを検証するための認証情報を URL に含めないように強くお勧めします。

機密情報を保護するために、ログでは以下が非表示になっています CodeBuild 。

- CodeBuild プロジェクト環境変数の Parameter Store または buildspec `env/parameter-store`セクションを使用して指定された文字列。詳細については、Amazon EC2 Systems Managerユーザーガイドの「Systems Manager パラメータストア」および「Systems Manager パラメータストアコンソールのチュートリアル」を参照してください。
- CodeBuild プロジェクト環境変数または buildspec `env/secrets-manager`セクション AWS Secrets Manager で を使用して指定された文字列。詳細については、「キー管理」を参照してください。

データ保護の詳細については、AWS セキュリティブログのブログ投稿「AWS の責任共有モデルと GDPR」を参照してください。

トピック

- データ暗号化
- キー管理
- トラフィックのプライバシー

データ暗号化

暗号化は CodeBuild セキュリティの重要な部分です。一部の暗号化 (伝送中のデータの暗号化など) はデフォルトで提供されるため、特に操作は不要です。その他の暗号化 (保管時のデータの暗号化など) については、プロジェクトまたはビルドの作成時に設定できます。

- 保管中のデータの暗号化 - キャッシュ、ログ、エクスポートされた生のテストレポートデータファイル、ビルド結果などのビルドアーティファクトは、デフォルトで を使用して暗号化されます AWS マネージドキー。これらの KMS キーを使用しない場合は、カスタマー管理キーを作成して設定する必要があります。詳細については、AWS Key Management Service ユーザーガイドの「KMS キーの作成」および「AWS Key Management Service の概念」を参照してください。
  - がビルド出力アーティファクトの暗号化 CodeBuild に使用する AWS KMS キーの識別子を `CODEBUILD_KMS_KEY_ID`環境変数に保存できます。詳細については、「ビルド環境の環境変数」を参照してください。
  - ビルドプロジェクトの作成時にカスタマー管理キーを指定できます。詳細については、「Set the Encryption Key Using the Console」および「CLI を使用して暗号化キーを設定する」を参照してください。

  ビルドフリートの Amazon Elastic Block Store ボリュームは、デフォルトで を使用して暗号化されます AWS マネージドキー。
- 転送中のデータの暗号化 - 顧客間、 CodeBuild およびその CodeBuild ダウンストリームの依存関係間のすべての通信は、署名バージョン 4 の署名プロセスを使用して署名された TLS 接続を使用して保護されます。すべての CodeBuild エンドポイントは、 によって管理される SHA-256 証明書を使用します AWS Private Certificate Authority。詳細については、「署名バージョン 4 の署名プロセス」および「ACM PCA とは」を参照してください。
- ビルドアーティファクトの暗号化 - ビルドプロジェクトに関連付けられた CodeBuild サービスロールは、ビルド出力アーティファクトを暗号化するために KMS キーにアクセスする必要があります。デフォルトでは、 は AWS アカウントで Amazon S3 AWS マネージドキー の CodeBuild を

使用します。この AWS マネージドキーを使用しない場合は、カスタマー管理キーを作成して設定する必要があります。詳細については、「カスタマーマネージドキーの作成」および AWS KMS デベロッパーガイドの「Creating Keys」を参照してください。

キー管理

暗号化によりコンテンツを不正使用から保護できます。暗号化キーを に保存し AWS Secrets Manager、ビルドプロジェクトに関連付けられた CodeBuild サービスロールに、Secrets Manager アカウントから暗号化キーを取得する許可を付与します。詳細については、「CodeBuild でカスタマー管理のキーを作成して設定する」、「 でのビルドプロジェクトの作成AWS CodeBuild」、AWS CodeBuild でのビルドの実行」、「チュートリアル: シークレットの保存と取得」を参照してください。

ビルドコマンドで CODEBUILD_KMS_KEY_ID環境変数を使用して、 AWS KMS キー識別子を取得します。詳細については、「ビルド環境の環境変数」を参照してください。

ランタイム環境用の Docker イメージを保存するプライベートレジストリへの認証情報を保護するには、Secrets Manager を使用できます。詳細については、「 の AWS Secrets Manager サンプルを使用したプライベートレジストリ CodeBuild」を参照してください。

トラフィックのプライバシー

インターフェイス VPC エンドポイントを使用する CodeBuild ように を設定することで、ビルドのセキュリティを強化できます。これを行う場合、インターネットゲートウェイ、NAT デバイス、または仮想プライベートゲートウェイは必要ありません。また、 を設定する必要はありませんが PrivateLink、推奨されます。詳細については、「VPC エンドポイントの使用」を参照してください。 PrivateLink および VPC エンドポイントの詳細については、AWS PrivateLink「」およびAWS「 を介したサービスへのアクセス PrivateLink」を参照してください。

での Identity and Access Management AWS CodeBuild

へのアクセスには認証情報 AWS CodeBuild が必要です。これらの認証情報には、S3 バケットへのビルドアーティファクトの保存と取得、ビルド用の Amazon CloudWatch Logs の表示などの AWS リソースへのアクセス許可が必要です。以下のセクションでは、 AWS Identity and Access Management (IAM) と を使用して CodeBuild リソースへのアクセスを保護する方法について説明します。

AWS CodeBuild リソースへのアクセス許可の管理の概要

すべての AWS リソースは AWS アカウントによって所有され、リソースを作成またはアクセスするためのアクセス許可はアクセス許可ポリシーによって管理されます。アカウント管理者は、IAM アイデンティティ (ユーザー、グループ、ロール) にアクセス許可ポリシーをアタッチできます。

> **Note**
>
> アカウント管理者 (または管理者ユーザー) は、管理者権限を持つユーザーです。詳細については、IAM ユーザーガイドの[IAM のベストプラクティス]を参照してください。

アクセス許可を付与するときは、アクセス許可を取得するユーザー、アクセスできるリソース、およびそれらのリソースに対して実行できるアクションを決定します。

トピック

- AWS CodeBuild リソースとオペレーション
- リソース所有権について
- リソースへのアクセスの管理
- ポリシー要素 (アクション、効果、プリンシパル) の指定

AWS CodeBuild リソースとオペレーション

では AWS CodeBuild、プライマリリソースはビルドプロジェクトです。ポリシーで Amazon リソースネーム (ARN) を使用して、ポリシーを適用するリソースを識別します。ビルドもリソースで、ARN が関連付けられています。詳細については、「」の「Amazon リソースネーム (ARN) と AWS のサービスの名前空間」を参照してくださいAmazon Web Services 全般のリファレンス。

リソースタイプ	ARN 形式
ビルドプロジェクト	`arn:aws:codebuild:` *region-ID* `:` *account-ID* `:project/` *project-name*
Build	`arn:aws:codebuild:` *region-ID* `:` *account-ID* `:build/` *build-ID*

リソースタイプ	ARN 形式
レポートグループ	arn:aws:codebuild: *region-ID* :*account-ID* :report-group/ *report-group-name*
レポート	arn:aws:codebuild: *region-ID* :*account-ID* :report/*report-ID*
すべての CodeBuild リソース	arn:aws:codebuild:*
指定された AWS リージョン内の指定されたアカウントが所有するすべての CodeBuild リソース	arn:aws:codebuild: *region-ID* :*account-ID* :*

> **Note**
>
> ほとんどの AWS サービスでは、ARN 内のコロン (:) またはスラッシュ (/) は同じ文字として扱われARNs。ただし、 はリソースパターンとルールで完全一致 CodeBuild を使用します。イベントパターンの作成時に正しい文字を使用して、リソース内の ARN 構文とそれらの文字が一致する必要があります。

例えば、次のように ARN を使用して、ステートメントで特定のビルドプロジェクト (*myBuildProject*) を指定できます。

```
"Resource": "arn:aws:codebuild:us-east-2:123456789012:project/myBuildProject"
```

すべてのリソースを指定する場合、または API アクションが ARN をサポートしていない場合は、以下の要領で、Resource エレメント内でワイルドカード文字 (*) を使用します。

```
"Resource": "*"
```

一部の CodeBuild API アクションは、複数のリソース (例:) を受け入れますBatchGetProjects。単一のステートメントに複数のリソースを指定するには、以下のようにコンマで ARN を区切ります。

```
"Resource": [
  "arn:aws:codebuild:us-east-2:123456789012:project/myBuildProject",
  "arn:aws:codebuild:us-east-2:123456789012:project/myOtherBuildProject"
]
```

CodeBuild には、 CodeBuild リソースを操作するための一連のオペレーションが用意されています。リストについては、「AWS CodeBuild アクセス許可リファレンス」を参照してください。

リソース所有権について

AWS アカウントは、誰がリソースを作成したかにかかわらず、アカウントで作成されたリソースを所有します。具体的には、リソース所有者は、リソース作成リクエスト AWS を認証するプリンシパルエンティティ (ルートアカウント、 ユーザー、または IAM ロール) のアカウントです。次の例は、この仕組みを示しています。

- AWS アカウントのルートアカウントの認証情報を使用してルールを作成する場合、 AWS アカウントは CodeBuild リソースの所有者です。
- AWS アカウントに ユーザーを作成し、そのユーザーに CodeBuild リソースを作成するアクセス許可を付与すると、そのユーザーは CodeBuild リソースを作成できます。ただし、ユーザーが属する AWS アカウントはリソースを所有しています CodeBuild。
- CodeBuild リソースを作成するためのアクセス許可を持つ AWS アカウントに IAM ロールを作成する場合、ロールを引き受けることのできるいずれのユーザーもリソースを作成できます CodeBuild。ロールが属する AWS アカウントがリソースを所有します CodeBuild。

リソースへのアクセスの管理

許可ポリシーでは、誰がどのリソースにアクセスできるかを記述します。

> **Note**
>
> このセクションでは、 AWS CodeBuildでの IAM の使用について説明します。これは、IAM サービスに関する詳細情報を取得できません。完全な IAM ドキュメンテーションについては、「IAM ユーザーガイド」の「IAM とは」を参照してください。IAM ポリシー構文の詳細と説明については、IAM ユーザーガイドの AWS IAM ポリシーの参照を参照してください。

IAM アイデンティティにアタッチされているポリシーは、アイデンティティベースのポリシー (IAM ポリシー) と呼ばれます。リソースにアタッチされたポリシーは、リソースベースのポリシーと呼ばれます。 は、クロスアカウントリソース共有を目的として、特定の読み取り専用 APIs のアイデンティティベースのポリシーとリソースベースのポリシー CodeBuild をサポートします。

S3 バケットへの安全なアクセス

CodeBuild プロジェクトに関連付けられた S3 バケットがユーザーまたはユーザーが信頼するユーザーによって所有されていることを確認するために、IAM ロールに次のアクセス許可を含めることを強くお勧めします。これらのアクセス許可は、 AWS 管理ポリシーおよびロールには含まれていません。自分で追加する必要があります。

- `s3:GetBucketAcl`
- `s3:GetBucketLocation`

プロジェクトで使用している S3 バケットの所有者が変更された場合は、自分を本来のバケット所有者にして IAM ロールのアクセス許可を更新する必要があります (まだ更新していない場合)。詳細については、「IAM グループまたはユーザーに CodeBuild アクセス許可を追加する」および「CodeBuild サービスロールの作成」を参照してください。

ポリシー要素 (アクション、効果、プリンシパル) の指定

サービスは、 AWS CodeBuild リソースごとに一連の API オペレーションを定義します。これらの API オペレーションのアクセス許可を付与するために、 はポリシーで指定できる一連のアクション CodeBuild を定義します。一部の API オペレーションは、API オペレーションを実行するために複数のアクションに対するアクセス許可を要求できます。詳細については、「AWS CodeBuild リソースとオペレーション」および「AWS CodeBuild アクセス許可リファレンス」を参照してください。

以下は、基本的なポリシーの要素です。

- リソース - Amazon リソースネーム (ARN) を使用して、ポリシーを適用するリソースを識別します。
- アクション – アクションのキーワードを使用して、許可または拒否するリソースオペレーションを識別します。たとえば、`codebuild:CreateProject` 許可は、`CreateProject` オペレーションを実行する許可をユーザーに与えます。
- 効果 - ユーザーがアクションをリクエストする際の効果を指定します。許可または拒否のいずれかになります。リソースへのアクセスを明示的に許可していない場合、アクセスは暗黙的に拒否されます。リソースへのアクセスを明示的に拒否することもできます。これは、別のポリシーがアクセ

スを許可している場合でも、ユーザーがリソースにアクセスできないようにするために行うことができます。

- プリンシパル - アイデンティティベースのポリシー (IAM ポリシー) で、ポリシーがアタッチされているユーザーが黙示的なプリンシパルとなります。リソースベースのポリシーでは、権限を受け取りたいユーザー、アカウント、サービス、またはその他のエンティティを指定します。

IAM ポリシーの構文と記述の詳細については、「IAM ユーザーガイド」の「AWS IAM ポリシーリファレンス」を参照してください。

すべての CodeBuild API アクションとそれらが適用されるリソースを示す表については、「」を参照してくださいAWS CodeBuild アクセス許可リファレンス。

でのアイデンティティベースのポリシーの使用 AWS CodeBuild

このトピックでは、アカウント管理者が IAM ID (ユーザー、グループ、ロール) にアクセス権限ポリシーをアタッチし、それによって AWS CodeBuild リソースでオペレーションを実行するアクセス権限を付与する方法を示すアイデンティティベースのポリシーの例を示します。

> **Important**
>
> 初めに、 CodeBuild リソースへのアクセスを管理するための基本概念と使用可能なオプションについて説明する概要トピックをお読みになることをお勧めします。詳細については、「AWS CodeBuild リソースへのアクセス許可の管理の概要」を参照してください。

トピック

- AWS CodeBuild コンソールの使用に必要な許可
- が Amazon Elastic Container Registry に接続 AWS CodeBuild するために必要なアクセス許可
- AWS CodeBuild コンソールがソースプロバイダーに接続するために必要なアクセス許可
- AWS の マネージド (事前定義) ポリシー AWS CodeBuild
- CodeBuild マネージドポリシーと通知
- CodeBuild AWS 管理ポリシーの更新
- カスタマーマネージドポリシーの例

us-east-2 リージョンにあり、123456789012 のアカウントで、名前が my で始まるビルドプロジェクトについての情報のみをユーザーが取得するのを許可するアクセス許可ポリシーの例を次に示します。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codebuild:BatchGetProjects",
      "Resource": "arn:aws:codebuild:us-east-2:123456789012:project/my*"
    }
  ]
}
```

AWS CodeBuild コンソールの使用に必要な許可

AWS CodeBuild コンソールを使用するユーザーは、 AWS アカウントの他の AWS リソースを記述できる最小限のアクセス許可を持っている必要があります。次のサービスからのアクセス許可を持っている必要があります。

- AWS CodeBuild
- Amazon CloudWatch
- CodeCommit （ ソースコードを AWS CodeCommit リポジトリに保存する場合 ）
- Amazon Elastic Container Registry (Amazon ECR) (Amazon ECR リポジトリの Docker イメージに依存するビルド環境を使用している場合)

> **Note**
>
> 2022 年 7 月 26 日に、デフォルトの IAM ポリシーが更新されました。詳細については、「が Amazon Elastic Container Registry に接続 AWS CodeBuild するために必要なアクセス許可」を参照してください。

- Amazon Elastic Container Service (Amazon ECS) (Amazon ECR リポジトリの Docker イメージに依存するビルド環境を使用している場合)
- AWS Identity and Access Management (IAM)
- AWS Key Management Service (AWS KMS)
- Amazon Simple Storage Service (Amazon S3)

これらの最小限必要なアクセス許可よりも制限された IAM ポリシーを作成している場合、コンソールは意図したとおりには機能しません。

が Amazon Elastic Container Registry に接続 AWS CodeBuild するために必要なアクセス許可

2022 年 7 月 26 日に、 AWS CodeBuild は Amazon ECR アクセス許可のデフォルトの IAM ポリシーを更新しました。次のアクセス許可がデフォルトポリシーから削除されました。

```
"ecr:PutImage",
"ecr:InitiateLayerUpload",
"ecr:UploadLayerPart",
"ecr:CompleteLayerUpload"
```

2022 年 7 月 26 日より前に作成された CodeBuild プロジェクトの場合、次の Amazon ECR ポリシーでポリシーを更新することをお勧めします。

```
"Action": [
    "ecr:BatchCheckLayerAvailability",
    "ecr:GetDownloadUrlForLayer",
    "ecr:BatchGetImage"
]
```

ポリシーの更新の詳細については、「IAM グループまたはユーザーに CodeBuild アクセス許可を追加する」を参照してください。

AWS CodeBuild コンソールがソースプロバイダーに接続するために必要なアクセス許可

AWS CodeBuild コンソールは、次の API アクションを使用してソースプロバイダー (リポジトリなど) GitHub に接続します。

- `codebuild:ListConnectedOAuthAccounts`
- `codebuild:ListRepositories`
- `codebuild:PersistOAuthToken`
- `codebuild:ImportSourceCredentials`

AWS CodeBuild コンソールを使用して、ソースプロバイダー (GitHub リポジトリなど) をビルドプロジェクトに関連付けることができます。これを行うには、まず、 AWS CodeBuild コンソールへの

アクセスに使用するユーザーに関連付けられた IAM アクセスポリシーに前述の API アクションを追加する必要があります。

`ListConnectedOAuthAccounts`、`ListRepositories`、および `PersistOAuthToken` の API アクションは、コードで呼び出すことを想定していません。したがって、これらの API アクションは AWS CLI および AWS SDKsには含まれていません。

AWS の マネージド (事前定義) ポリシー AWS CodeBuild

AWS は、 が作成および管理するスタンドアロン IAM ポリシーを提供することで、多くの一般的ユースケースに対応します AWS。これらの AWS 管理ポリシーは、一般的ユースケースに必要なアクセス許可を付与するため、どのアクセス許可が必要かを調査する必要はありません。 の マネージドポリシーは、問題のポリシーが付与されたユーザーの責任に必要な IAM、 AWS CodeCommit Amazon EC2、Amazon ECR、Amazon SNS、Amazon CloudWatch Events などの他のサービスでオペレーションを実行するアクセス許可 CodeBuild も提供します。例えば、`AWSCodeBuildAdminAccess` ポリシーは、このポリシーを持つユーザーが、プロジェクト関連イベントに関する通知 (名前にプレフィックス が付いているトピック`arn:aws:codebuild:`) のプロジェクト構築と Amazon SNS トピックのイベントルールを作成および管理 CloudWatchしたり、 でプロジェクトとレポートグループを管理したりできるようにする管理レベルのユーザーポリシーです CodeBuild。詳細については、「IAM ユーザーガイド」の「AWS マネージドポリシー」を参照してください。

アカウントのユーザーにアタッチできる以下の AWS マネージドポリシーは、 に固有のものです AWS CodeBuild。

AWSCodeBuildAdminAccess

CodeBuild ビルドプロジェクトを管理するためのアクセス許可 CodeBuild を含む へのフルアクセスを提供します。

AWSCodeBuildDeveloperAccess

へのアクセスを許可します CodeBuild が、ビルドプロジェクトの管理は許可しません。

AWSCodeBuildReadOnlyAccess

への読み取り専用アクセスを提供します CodeBuild。

が CodeBuild 作成するビルド出力アーティファクトにアクセスするには、 という名前の AWS 管理ポリシーもアタッチする必要があります`AmazonS3ReadOnlyAccess`。

CodeBuild サービスロールを作成および管理するには、 という名前の AWS マネージドポリシーもアタッチする必要がありますIAMFullAccess。

独自のカスタム IAM ポリシーを作成して、アクションとリソースの CodeBuildアクセス許可を許可することもできます。こうしたカスタムポリシーは、該当するアクセス許可が必要なユーザーまたはグループにアタッチできます。

トピック

- AWSCodeBuildAdminAccess
- AWSCodeBuildDeveloperAccess
- AWSCodeBuildReadOnlyAccess

AWSCodeBuildAdminAccess

このAWSCodeBuildAdminAccessポリシーは CodeBuild、 CodeBuild ビルドプロジェクトを管理するアクセス許可を含む、 へのフルアクセスを提供します。このポリシーは、管理レベルのユーザーにのみ適用し、 CodeBuild プロジェクトやレポートグループを削除する機能など、 AWS アカウント内のプロジェクト、レポートグループ、関連リソースを完全に制御できるようにします。

AWSCodeBuildAdminAccess ポリシーには、次のポリシーステートメントが含まれています。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AWSServicesAccess",
      "Action": [
        "codebuild:*",
        "codecommit:GetBranch",
        "codecommit:GetCommit",
        "codecommit:GetRepository",
        "codecommit:ListBranches",
        "codecommit:ListRepositories",
        "cloudwatch:GetMetricStatistics",
        "ec2:DescribeVpcs",
        "ec2:DescribeSecurityGroups",
        "ec2:DescribeSubnets",
        "ecr:DescribeRepositories",
        "ecr:ListImages",
        "elasticfilesystem:DescribeFileSystems",
        "events:DeleteRule",
```

```
        "events:DescribeRule",
        "events:DisableRule",
        "events:EnableRule",
        "events:ListTargetsByRule",
        "events:ListRuleNamesByTarget",
        "events:PutRule",
        "events:PutTargets",
        "events:RemoveTargets",
        "logs:GetLogEvents",
        "s3:GetBucketLocation",
        "s3:ListAllMyBuckets"
      ],
      "Effect": "Allow",
      "Resource": "*"
    },
    {
      "Sid": "CWLDeleteLogGroupAccess",
      "Action": [
        "logs:DeleteLogGroup"
      ],
      "Effect": "Allow",
      "Resource": "arn:aws:logs:*:*:log-group:/aws/codebuild/*:log-stream:*"
    },
    {
      "Sid": "SSMParameterWriteAccess",
      "Effect": "Allow",
      "Action": [
        "ssm:PutParameter"
      ],
      "Resource": "arn:aws:ssm:*:*:parameter/CodeBuild/*"
    },
    {
      "Sid": "SSMStartSessionAccess",
      "Effect": "Allow",
      "Action": [
        "ssm:StartSession"
      ],
      "Resource": "arn:aws:ecs:*:*:task/*/*"
    },
    {
      "Sid": "CodeStarConnectionsReadWriteAccess",
      "Effect": "Allow",
      "Action": [
        "codestar-connections:CreateConnection",
```

```
        "codestar-connections:DeleteConnection",
        "codestar-connections:UpdateConnectionInstallation",
        "codestar-connections:TagResource",
        "codestar-connections:UntagResource",
        "codestar-connections:ListConnections",
        "codestar-connections:ListInstallationTargets",
        "codestar-connections:ListTagsForResource",
        "codestar-connections:GetConnection",
        "codestar-connections:GetIndividualAccessToken",
        "codestar-connections:GetInstallationUrl",
        "codestar-connections:PassConnection",
        "codestar-connections:StartOAuthHandshake",
        "codestar-connections:UseConnection"
      ],
      "Resource": [
        "arn:aws:codestar-connections:*:*:connection/*",
        "arn:aws:codeconnections:*:*:connection/*"
      ]
    },
    {
      "Sid": "CodeStarNotificationsReadWriteAccess",
      "Effect": "Allow",
      "Action": [
        "codestar-notifications:CreateNotificationRule",
        "codestar-notifications:DescribeNotificationRule",
        "codestar-notifications:UpdateNotificationRule",
        "codestar-notifications:DeleteNotificationRule",
        "codestar-notifications:Subscribe",
        "codestar-notifications:Unsubscribe"
      ],
      "Resource": "*",
      "Condition": {
        "StringLike": {
          "codestar-notifications:NotificationsForResource": "arn:aws:codebuild:*"
        }
      }
    },
    {
      "Sid": "CodeStarNotificationsListAccess",
      "Effect": "Allow",
      "Action": [
        "codestar-notifications:ListNotificationRules",
        "codestar-notifications:ListEventTypes",
        "codestar-notifications:ListTargets",
```

```
        "codestar-notifications:ListTagsforResource"
      ],
      "Resource": "*"
    },
    {
      "Sid": "CodeStarNotificationsSNSTopicCreateAccess",
      "Effect": "Allow",
      "Action": [
        "sns:CreateTopic",
        "sns:SetTopicAttributes"
      ],
      "Resource": "arn:aws:sns:*:*:codestar-notifications*"
    },
    {
      "Sid": "SNSTopicListAccess",
      "Effect": "Allow",
      "Action": [
        "sns:ListTopics",
        "sns:GetTopicAttributes"
      ],
      "Resource": "*"
    },
    {
      "Sid": "CodeStarNotificationsChatbotAccess",
      "Effect": "Allow",
      "Action": [
        "chatbot:DescribeSlackChannelConfigurations",
        "chatbot:ListMicrosoftTeamsChannelConfigurations"
      ],
      "Resource": "*"
    }
  ]
}
```

AWSCodeBuildDeveloperAccess

このAWSCodeBuildDeveloperAccessポリシーは、 CodeBuild および プロジェクトおよびレポートグループ関連リソースのすべての機能へのアクセスを許可します。このポリシーでは、ユーザーは CodeBuild プロジェクトやレポートグループ、または CloudWatch イベントなどの他の AWS サービスの関連リソースを削除することはできません。ほとんどのユーザーにこのポリシーを適用することをお勧めします。

AWSCodeBuildDeveloperAccess ポリシーには、次のポリシーステートメントが含まれています。

```
{
  "Statement": [
    {
      "Sid": "AWSServicesAccess",
      "Action": [
        "codebuild:StartBuild",
        "codebuild:StopBuild",
        "codebuild:StartBuildBatch",
        "codebuild:StopBuildBatch",
        "codebuild:RetryBuild",
        "codebuild:RetryBuildBatch",
        "codebuild:BatchGet*",
        "codebuild:GetResourcePolicy",
        "codebuild:DescribeTestCases",
        "codebuild:DescribeCodeCoverages",
        "codebuild:List*",
        "codecommit:GetBranch",
        "codecommit:GetCommit",
        "codecommit:GetRepository",
        "codecommit:ListBranches",
        "cloudwatch:GetMetricStatistics",
        "events:DescribeRule",
        "events:ListTargetsByRule",
        "events:ListRuleNamesByTarget",
        "logs:GetLogEvents",
        "s3:GetBucketLocation",
        "s3:ListAllMyBuckets"
      ],
      "Effect": "Allow",
      "Resource": "*"
    },
    {
      "Sid": "SSMParameterWriteAccess",
      "Effect": "Allow",
      "Action": [
        "ssm:PutParameter"
      ],
      "Resource": "arn:aws:ssm:*:*:parameter/CodeBuild/*"
    },
    {
```

```
      "Sid": "SSMStartSessionAccess",
      "Effect": "Allow",
      "Action": [
        "ssm:StartSession"
      ],
      "Resource": "arn:aws:ecs:*:*:task/*/*"
    },
    {
      "Sid": "CodeStarConnectionsUserAccess",
      "Effect": "Allow",
      "Action": [
        "codestar-connections:ListConnections",
        "codestar-connections:GetConnection"
      ],
      "Resource": [
        "arn:aws:codestar-connections:*:*:connection/*",
        "arn:aws:codeconnections:*:*:connection/*"
      ]
    },
    {
      "Sid": "CodeStarNotificationsReadWriteAccess",
      "Effect": "Allow",
      "Action": [
        "codestar-notifications:CreateNotificationRule",
        "codestar-notifications:DescribeNotificationRule",
        "codestar-notifications:UpdateNotificationRule",
        "codestar-notifications:Subscribe",
        "codestar-notifications:Unsubscribe"
      ],
      "Resource": "*",
      "Condition": {
        "StringLike": {
          "codestar-notifications:NotificationsForResource": "arn:aws:codebuild:*"
        }
      }
    },
    {
      "Sid": "CodeStarNotificationsListAccess",
      "Effect": "Allow",
      "Action": [
        "codestar-notifications:ListNotificationRules",
        "codestar-notifications:ListEventTypes",
        "codestar-notifications:ListTargets",
        "codestar-notifications:ListTagsforResource"
```

```
      ],
      "Resource": "*"
    },
    {
      "Sid": "SNSTopicListAccess",
      "Effect": "Allow",
      "Action": [
        "sns:ListTopics",
        "sns:GetTopicAttributes"
      ],
      "Resource": "*"
    },
    {
      "Sid": "CodeStarNotificationsChatbotAccess",
      "Effect": "Allow",
      "Action": [
        "chatbot:DescribeSlackChannelConfigurations",
        "chatbot:ListMicrosoftTeamsChannelConfigurations"
      ],
      "Resource": "*"
    }
  ],
  "Version": "2012-10-17"
}
```

AWSCodeBuildReadOnlyAccess

このAWSCodeBuildReadOnlyAccessポリシーは、 およびその他の AWS サービスの CodeBuild 関連リソースへの読み取り専用アクセスを許可します。ビルドの表示と実行、プロジェクトの表示、レポートグループの表示はできるが、それらの変更はできないユーザーにこのポリシーを適用します。

AWSCodeBuildReadOnlyAccess ポリシーには、次のポリシーステートメントが含まれています。

```
{
  "Statement": [
    {
      "Sid": "AWSServicesAccess",
      "Action": [
        "codebuild:BatchGet*",
        "codebuild:GetResourcePolicy",
        "codebuild:List*",
        "codebuild:DescribeTestCases",
        "codebuild:DescribeCodeCoverages",
```

```
        "codecommit:GetBranch",
        "codecommit:GetCommit",
        "codecommit:GetRepository",
        "cloudwatch:GetMetricStatistics",
        "events:DescribeRule",
        "events:ListTargetsByRule",
        "events:ListRuleNamesByTarget",
        "logs:GetLogEvents"
      ],
      "Effect": "Allow",
      "Resource": "*"
    },
    {
      "Sid": "CodeStarConnectionsUserAccess",
      "Effect": "Allow",
      "Action": [
        "codestar-connections:ListConnections",
        "codestar-connections:GetConnection"
      ],
      "Resource": [
        "arn:aws:codestar-connections:*:*:connection/*",
        "arn:aws:codeconnections:*:*:connection/*"
      ]
    },
    {
      "Sid": "CodeStarNotificationsPowerUserAccess",
      "Effect": "Allow",
      "Action": [
        "codestar-notifications:DescribeNotificationRule"
      ],
      "Resource": "*",
      "Condition": {
        "StringLike": {
          "codestar-notifications:NotificationsForResource": "arn:aws:codebuild:*"
        }
      }
    },
    {
      "Sid": "CodeStarNotificationsListAccess",
      "Effect": "Allow",
      "Action": [
        "codestar-notifications:ListNotificationRules",
        "codestar-notifications:ListEventTypes",
        "codestar-notifications:ListTargets"
```

```
      ],
      "Resource": "*"
    }
  ],
  "Version": "2012-10-17"
}
```

CodeBuild マネージドポリシーと通知

CodeBuild は、ビルドプロジェクトへの重要な変更をユーザーに通知できる通知をサポートしています。の管理ポリシー CodeBuild には、通知機能のポリシーステートメントが含まれます。詳細については、通知とは を参照してください。

フルアクセスマネージドポリシーの通知に関連するアクセス許可

AWSCodeBuildFullAccess マネージドポリシーには、通知へのフルアクセスを許可する次のステートメントが含まれています。これらのマネージドポリシーのいずれかが適用されたユーザーは、通知の Amazon SNS トピックの作成と管理、トピックに対するユーザーのサブスクライブとサブスクライブ解除、通知ルールのターゲットとして選択するトピックの一覧表示、Slack 用に設定された AWS Chatbot クライアントの一覧表示を行うこともできます。

```
    {
        "Sid": "CodeStarNotificationsReadWriteAccess",
        "Effect": "Allow",
        "Action": [
            "codestar-notifications:CreateNotificationRule",
            "codestar-notifications:DescribeNotificationRule",
            "codestar-notifications:UpdateNotificationRule",
            "codestar-notifications:DeleteNotificationRule",
            "codestar-notifications:Subscribe",
            "codestar-notifications:Unsubscribe"
        ],
        "Resource": "*",
        "Condition" : {
            "StringLike" : {"codestar-notifications:NotificationsForResource" :
 "arn:aws:codebuild:*"}
        }
    },
    {
        "Sid": "CodeStarNotificationsListAccess",
        "Effect": "Allow",
        "Action": [
```

```
            "codestar-notifications:ListNotificationRules",
            "codestar-notifications:ListTargets",
            "codestar-notifications:ListTagsforResource",
            "codestar-notifications:ListEventTypes"
        ],
        "Resource": "*"
    },
    {
        "Sid": "CodeStarNotificationsSNSTopicCreateAccess",
        "Effect": "Allow",
        "Action": [
            "sns:CreateTopic",
            "sns:SetTopicAttributes"
        ],
        "Resource": "arn:aws:sns:*:*:codestar-notifications*"
    },
    {
        "Sid": "SNSTopicListAccess",
        "Effect": "Allow",
        "Action": [
            "sns:ListTopics"
        ],
        "Resource": "*"
    },
    {
        "Sid": "CodeStarNotificationsChatbotAccess",
        "Effect": "Allow",
        "Action": [
            "chatbot:DescribeSlackChannelConfigurations",
            "chatbot:ListMicrosoftTeamsChannelConfigurations"
          ],
       "Resource": "*"
    }
```

読み取り専用マネージドポリシーの通知に関連するアクセス許可

AWSCodeBuildReadOnlyAccess 管理ポリシーには、通知への読み取り専用アクセスを許可する以下のステートメントが含まれています。この管理ポリシーが適用されたユーザーは、リソースの通知を表示することはできますが、リソースの作成や管理、リソースへのサブスクライブを行うことはできません。

```
  {
       "Sid": "CodeStarNotificationsPowerUserAccess",
```

```
        "Effect": "Allow",
        "Action": [
            "codestar-notifications:DescribeNotificationRule"
        ],
        "Resource": "*",
        "Condition" : {
            "StringLike" : {"codestar-notifications:NotificationsForResource" :
 "arn:aws:codebuild:*"}
        }
    },
    {
        "Sid": "CodeStarNotificationsListAccess",
        "Effect": "Allow",
        "Action": [
            "codestar-notifications:ListNotificationRules",
            "codestar-notifications:ListEventTypes",
            "codestar-notifications:ListTargets"
        ],
        "Resource": "*"
    }
```

その他の管理ポリシーの通知に関連するアクセス許可

AWSCodeBuildDeveloperAccess 管理ポリシーには、ユーザーが通知を作成、編集、サブスクライブできるようにする次のステートメントが含まれています。ユーザーは通知ルールを削除したり、リソースのタグを管理したりすることはできません。

```
    {
        "Sid": "CodeStarNotificationsReadWriteAccess",
        "Effect": "Allow",
        "Action": [
            "codestar-notifications:CreateNotificationRule",
            "codestar-notifications:DescribeNotificationRule",
            "codestar-notifications:UpdateNotificationRule",
            "codestar-notifications:Subscribe",
            "codestar-notifications:Unsubscribe"
        ],
        "Resource": "*",
        "Condition" : {
            "StringLike" : {"codestar-notifications:NotificationsForResource" :
 "arn:aws:codebuild*"}
        }
    },
```

```
    {
        "Sid": "CodeStarNotificationsListAccess",
        "Effect": "Allow",
        "Action": [
            "codestar-notifications:ListNotificationRules",
            "codestar-notifications:ListTargets",
            "codestar-notifications:ListTagsforResource",
            "codestar-notifications:ListEventTypes"
        ],
        "Resource": "*"
    },
    {
        "Sid": "SNSTopicListAccess",
        "Effect": "Allow",
        "Action": [
            "sns:ListTopics"
        ],
        "Resource": "*"
    },
    {
        "Sid": "CodeStarNotificationsChatbotAccess",
        "Effect": "Allow",
        "Action": [
            "chatbot:DescribeSlackChannelConfigurations",
            "chatbot:ListMicrosoftTeamsChannelConfigurations"
          ],
       "Resource": "*"
    }
```

IAM と通知の詳細については、「AWS CodeStar Notifications の Identity and Access Management」を参照してください。

CodeBuild AWS 管理ポリシーの更新

このサービスがこれらの変更の追跡を開始した CodeBuild 以降の、 の AWS マネージドポリシーの更新に関する詳細を表示します。このページの変更に関する自動通知については、AWS CodeBuild ユーザーガイドのドキュメント履歴 の RSS フィードを購読してください。

変更	説明	日付
AWSCodeBuildAdminAccess 、AWSCodeBu	CodeBuild は、 AWS CodeConnections ブランド変	2024 年 4 月 18 日

変更	説明	日付
ildDeveloperAccess 、 AWSCodeBuildReadOnlyAccess - 既存のポリシーへの更新	更をサポートするために、これらのポリシーにリソースを追加しました。 AWSCodeBuildAdminAccess 、AWSCodeBuildDeveloperAccess 、および AWSCodeBuildReadOnlyAccess ポリシーが変更され、リソース が追加されましたarn:aws:codeconnections:*:*:connection/* 。	
AWSCodeBuildAdminAccess と AWSCodeBuildDeveloperAccess - 既存のポリシーに対する更新	CodeBuild は、 を使用して追加の通知タイプをサポートするアクセス許可をこれらのポリシーに追加しました AWS Chatbot。 AWSCodeBuildAdminAccess および AWSCodeBuildDeveloperAccess ポリシーが変更され、アクセス許可 chatbot:ListMicrosoftTeamsChannelConfigurations が追加されました。	2023 年 5 月 16 日
CodeBuild が変更の追跡を開始	CodeBuild が AWS マネージドポリシーの変更の追跡を開始しました。	2021 年 5 月 16 日

カスタマーマネージドポリシーの例

このセクションでは、 AWS CodeBuild アクションのアクセス許可を付与するユーザーポリシー例を示しています。これらのポリシーは、 CodeBuild API、 AWS SDKs、または を使用しているときに機能します AWS CLI。コンソールを使用する場合は、コンソールに固有の追加のアクセス許可を付与する必要があります。詳細については、「AWS CodeBuild コンソールの使用に必要な許可」を参照してください。

次のサンプル IAM ポリシーを使用して、ユーザーとロールの CodeBuild アクセスを制限できます。

トピック

- ビルドプロジェクトに関する情報の取得をユーザーに許可する
- レポートグループに関する情報の取得をユーザーに許可する
- レポートに関する情報の取得をユーザーに許可する
- ビルドプロジェクトの作成をユーザーに許可する
- レポートグループの作成をユーザーに許可する
- レポートグループの削除をユーザーに許可する
- レポートの削除をユーザーに許可する
- ビルドプロジェクトの削除をユーザーに許可する
- ビルドプロジェクト名の一覧表示をユーザーに許可する
- ビルドプロジェクトに関する情報の変更をユーザーに許可する
- レポートグループの変更をユーザーに許可する
- ビルドに関する情報の取得をユーザーに許可する
- ビルドプロジェクトのビルド ID の一覧表示をユーザーに許可する
- ビルド ID の一覧表示をユーザーに許可する
- レポートグループの一覧表示をユーザーに許可する
- レポートの一覧表示をユーザーに許可する
- レポートグループのレポートの一覧表示をユーザーに許可する
- レポートのテストケースの一覧表示をユーザーに許可する
- ビルドの実行開始をユーザーに許可する
- ビルドの停止試行をユーザーに許可する

- ビルドの削除試行をユーザーに許可する
- によって管理される Docker イメージに関する情報の取得をユーザーに許可する CodeBuild
- VPC ネットワークインターフェイスの作成に必要な AWS サービス CodeBuild へのアクセスを許可する
- 拒否ステートメントを使用して、 がソースプロバイダーから切断 AWS CodeBuild されないようにする

ビルドプロジェクトに関する情報の取得をユーザーに許可する

次のポリシーステートメントの例では、us-east-2 リージョンにあり、123456789012 のアカウントで、名前が my で始まるビルドプロジェクトについての情報をユーザーが取得するのを許可します。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codebuild:BatchGetProjects",
      "Resource": "arn:aws:codebuild:us-east-2:123456789012:project/my*"
    }
  ]
}
```

レポートグループに関する情報の取得をユーザーに許可する

次のポリシーステートメント例では、us-east-2 リージョンにある 123456789012 アカウントのレポートグループについての情報をユーザーが取得するのを許可します。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codebuild:BatchGetReportGroups",
      "Resource": "arn:aws:codebuild:us-east-2:123456789012:report-group/*"
    }
  ]
}
```

レポートに関する情報の取得をユーザーに許可する

次のポリシーステートメント例では、123456789012 アカウントの us-east-2 リージョンのレポートに関する情報をユーザーが取得できるようにします。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codebuild:BatchGetReports",
      "Resource": "arn:aws:codebuild:us-east-2:123456789012:report-group/*"
    }
  ]
}
```

ビルドプロジェクトの作成をユーザーに許可する

次のポリシーステートメントの例では、ユーザーは任意の名前でビルドプロジェクトを作成できますが、アカウント の us-east-2リージョンでのみ123456789012、指定された CodeBuild サービスロールのみを使用できます。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codebuild:CreateProject",
      "Resource": "arn:aws:codebuild:us-east-2:123456789012:project/*"
    },
    {
      "Effect": "Allow",
      "Action": "iam:PassRole",
      "Resource": "arn:aws:iam::123456789012:role/CodeBuildServiceRole"
    }
  ]
}
```

次のポリシーステートメントの例では、任意の名前でビルドプロジェクトを作成することをユーザーに許可します。ただし、アカウント の us-east-2リージョンでのみ123456789012、指定された CodeBuild サービスロールのみを使用します。また、ユーザーが指定したサービスロールを でのみ使用でき AWS CodeBuild 、他の AWS サービスでは使用できません。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codebuild:CreateProject",
      "Resource": "arn:aws:codebuild:us-east-2:123456789012:project/*"
    },
    {
      "Effect": "Allow",
      "Action": "iam:PassRole",
      "Resource": "arn:aws:iam::123456789012:role/CodeBuildServiceRole",
      "Condition": {
          "StringEquals": {"iam:PassedToService": "codebuild.amazonaws.com"}
      }
    }
  ]
}}
```

レポートグループの作成をユーザーに許可する

次のポリシーステートメントの例では、123456789012 アカウントの us-east-2 リージョンにレポートグループを作成することをユーザーに許可します。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codebuild:CreateReportGroup",
      "Resource": "arn:aws:codebuild:us-east-2:123456789012:report-group/*"
    }
  ]
}
```

レポートグループの削除をユーザーに許可する

次のポリシーステートメント例では、123456789012 アカウントの us-east-2 リージョンからレポートグループを削除することをユーザーに許可します。

```
{
```

```
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codebuild:DeleteReportGroup",
      "Resource": "arn:aws:codebuild:us-east-2:123456789012:report-group/*"
    }
  ]
}
```

レポートの削除をユーザーに許可する

次のポリシーステートメント例では、123456789012 アカウントの us-east-2 リージョンからレポートを削除することをユーザーに許可します。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codebuild:DeleteReport",
      "Resource": "arn:aws:codebuild:us-east-2:123456789012:report-group/*"
    }
  ]
}
```

ビルドプロジェクトの削除をユーザーに許可する

次のポリシーステートメントの例では、us-east-2 リージョンにあり、123456789012 のアカウントで、名前が my で始まるビルドプロジェクトをユーザーが削除するのを許可します。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codebuild:DeleteProject",
      "Resource": "arn:aws:codebuild:us-east-2:123456789012:project/my*"
    }
  ]
}
```

ビルドプロジェクト名の一覧表示をユーザーに許可する

以下のポリシーステートメントの例では、同じアカウントのビルドプロジェクト名のリストをユーザーが取得するのを許可します。

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

ビルドプロジェクトに関する情報の変更をユーザーに許可する

次のポリシーステートメントの例では、名前は問いませんが、us-east-2 リージョンだけにある 123456789012 アカウントで、特定の AWS CodeBuild のサービスロールのみを使用したビルドプロジェクトに関する情報をユーザーが変更するのを許可します。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codebuild:UpdateProject",
      "Resource": "arn:aws:codebuild:us-east-2:123456789012:project/*"
    },
    {
      "Effect": "Allow",
      "Action": "iam:PassRole",
      "Resource": "arn:aws:iam::123456789012:role/CodeBuildServiceRole"
    }
  ]
}
```

レポートグループの変更をユーザーに許可する

次のポリシーステートメント例では、123456789012 アカウントの us-east-2 リージョンでレポートグループを変更することをユーザーに許可します。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codebuild:UpdateReportGroup",
      "Resource": "arn:aws:codebuild:us-east-2:123456789012:report-group/*"
    }
  ]
}
```

ビルドに関する情報の取得をユーザーに許可する

次のポリシーステートメントの例では、us-east-2 リージョンにあり、123456789012 のアカウントで、my-build-project および my-other-build-project という名前のビルドプロジェクトについての情報をユーザーが取得するのを許可します。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codebuild:BatchGetBuilds",
      "Resource": [
        "arn:aws:codebuild:us-east-2:123456789012:project/my-build-project",
        "arn:aws:codebuild:us-east-2:123456789012:project/my-other-build-project"
      ]
    }
  ]
}
```

ビルドプロジェクトのビルド ID の一覧表示をユーザーに許可する

次のポリシーステートメントの例では、us-east-2 リージョンにあり、123456789012 のアカウントで、my-build-project および my-other-build-project という名前のビルドプロジェクトのビルド ID リストをユーザーが取得するのを許可します。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
```

```
      "Effect": "Allow",
      "Action": "codebuild:ListBuildsForProject",
      "Resource": [
        "arn:aws:codebuild:us-east-2:123456789012:project/my-build-project",
        "arn:aws:codebuild:us-east-2:123456789012:project/my-other-build-project"
      ]
    }
  ]
}
```

ビルド ID の一覧表示をユーザーに許可する

以下のポリシーステートメントの例では、同じアカウントのすべてのビルド ID のリストをユーザーが取得するのを許可します。

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

レポートグループの一覧表示をユーザーに許可する

次のポリシーステートメント例では、123456789012 アカウント us-east-2 のリージョンでレポートグループリストを取得することをユーザーに許可します。

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

レポートの一覧表示をユーザーに許可する

次のポリシーステートメント例では、123456789012 アカウントの us-east-2 のリージョンでレポートリストを取得することをユーザーに許可します。

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

レポートグループのレポートの一覧表示をユーザーに許可する

次のポリシーステートメント例では、123456789012 アカウントの us-east-2 のリージョンでレポートグループのレポートリストを取得することをユーザーに許可します。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codebuild:ListReportsForReportGroup",
      "Resource": "arn:aws:codebuild:us-east-2:123456789012:report-group/*"
    }
  ]
}
```

レポートのテストケースの一覧表示をユーザーに許可する

次のポリシーステートメント例では、123456789012 アカウントの us-east-2 リージョンでレポートのテストケースのリストを取得することをユーザーに許可します。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
```

```
      "Action": "codebuild:DescribeTestCases",
      "Resource": "arn:aws:codebuild:us-east-2:123456789012:report-group/*"
    }
  ]
}
```

ビルドの実行開始をユーザーに許可する

次のポリシーステートメントの例では、us-east-2 リージョンの 123456789012 というアカウントの、my という名前で始まるビルドプロジェクトのビルドをユーザーが実行する許可を与えます。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codebuild:StartBuild",
      "Resource": "arn:aws:codebuild:us-east-2:123456789012:project/my*"
    }
  ]
}
```

ビルドの停止試行をユーザーに許可する

次のポリシーステートメントの例では、us-east-2 リージョンにあり、123456789012 のアカウントで、名前が my で始まるビルドの停止を試みるのをユーザーに許可します。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codebuild:StopBuild",
      "Resource": "arn:aws:codebuild:us-east-2:123456789012:project/my*"
    }
  ]
}
```

ビルドの削除試行をユーザーに許可する

次のポリシーステートメントの例では、123456789012 アカウントで us-east-2 リージョンに限り、名前が my で始まるビルドプロジェクトでのビルドの削除試行をユーザーに許可します。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codebuild:BatchDeleteBuilds",
      "Resource": "arn:aws:codebuild:us-east-2:123456789012:project/my*"
    }
  ]
}
```

によって管理される Docker イメージに関する情報の取得をユーザーに許可する CodeBuild

次のポリシーステートメントの例では、 によって管理されるすべての Docker イメージに関する情報の取得をユーザーに許可します CodeBuild。

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

VPC ネットワークインターフェイスの作成に必要な AWS サービス CodeBuild へのアクセスを許可する

次のポリシーステートメントの例では、2 つのサブネットを持つ VPC にネットワークインターフェイスを作成する AWS CodeBuild アクセス許可を付与します。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ec2:CreateNetworkInterface",
        "ec2:DescribeDhcpOptions",
        "ec2:DescribeNetworkInterfaces",
```

```
        "ec2:DeleteNetworkInterface",
        "ec2:DescribeSubnets",
        "ec2:DescribeSecurityGroups",
        "ec2:DescribeVpcs"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "ec2:CreateNetworkInterfacePermission"
      ],
      "Resource": "arn:aws:ec2:region:account-id:network-interface/*",
      "Condition": {
        "StringEquals": {
          "ec2:AuthorizedService": "codebuild.amazonaws.com"
        },
        "ArnEquals": {
          "ec2:Subnet": [
            "arn:aws:ec2:region:account-id:subnet/subnet-id-1",
            "arn:aws:ec2:region:account-id:subnet/subnet-id-2"
          ]
        }
      }
    }
  ]
}
```

拒否ステートメントを使用して、 がソースプロバイダーから切断 AWS CodeBuild されないようにする

以下のポリシーステートメントの例では、拒否ステートメントを使用して AWS CodeBuild によるソースプロバイダーの切断を防ぎます。ソースプロバイダーと接続するには、`codebuild:PersistOAuthToken` および `codebuild:ImportSourceCredentials` の逆である `codebuild:DeleteOAuthToken` を使用します。詳細については、「AWS CodeBuild コンソールがソースプロバイダーに接続するために必要なアクセス許可」を参照してください。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": "codebuild:DeleteOAuthToken",
```

```
      "Resource": "*"
    }
  ]
}
```

AWS CodeBuild アクセス許可リファレンス

AWS CodeBuild ポリシーで AWS全体の条件キーを使用して条件を表現できます。リストについては、IAM ユーザーガイドの「利用可能なキー」を参照してください。

アクションは、ポリシーの `Action` フィールドで指定します。アクションを指定するには、API オペレーション名 (例えば、`codebuild:` や `codebuild:CreateProject`) の前に `codebuild:StartBuild` プレフィックスを使用します。単一のステートメントに複数のアクションを指定するには、コンマで区切ります (例えば、`"Action": [ "codebuild:CreateProject", "codebuild:StartBuild" ]`)。

ワイルドカード文字の使用

ポリシーの `Resource` フィールドでリソース値として、ワイルドカード文字 (*) を使用して、または使用せずに ARN を指定します。ワイルドカードを使用して複数のアクションまたはリソースを指定することができます。例えば、 `codebuild:*`はすべての CodeBuild アクションを指定し、 は という単語で始まるすべての CodeBuild アクション`codebuild:Batch*`を指定しますBatch。次の例では、my で始まる名前のすべてのビルドプロジェクトへのアクセスを許可します。

```
arn:aws:codebuild:us-east-2:123456789012:project/my*
```

CodeBuild API オペレーションとアクションに必要なアクセス許可

BatchDeleteBuilds

アクション: `codebuild:BatchDeleteBuilds`

ビルドを削除するのに必要です。

リソース: `arn:aws:codebuild:`*`region-ID`*`:`*`account-ID`*`:project/`*`project-name`*

BatchGetBuilds

アクション: `codebuild:BatchGetBuilds`

ビルドに関する情報を取得するのに必要です。

リソース: arn:aws:codebuild:*region-ID*:*account-ID*:project/*project-name*

BatchGetProjects

アクション: codebuild:BatchGetProjects

ビルドプロジェクトに関する情報を取得するのに必要です。

リソース: arn:aws:codebuild:*region-ID*:*account-ID*:project/*project-name*

BatchGetReportGroups

アクション: codebuild:BatchGetReportGroups

レポートグループに関する情報を取得するために必要です。

リソース: arn:aws:codebuild:*region-ID*:*account-ID*:report-group/*report-group-name*

BatchGetReports

アクション: codebuild:BatchGetReports

レポートに関する情報を取得するのに必要です。

リソース: arn:aws:codebuild:*region-ID*:*account-ID*:report-group/*report-group-name*

BatchPutTestCases1

アクション: codebuild:BatchPutTestCases

テストレポートを作成または更新するために必要です。

リソース: arn:aws:codebuild:*region-ID*:*account-ID*:report-group/*report-group-name*

CreateProject

アクション:codebuild:CreateProject、iam:PassRole

ビルドプロジェクトを作成するのに必要です。

リソース:

- arn:aws:codebuild:*region-ID*:*account-ID*:project/*project-name*
- arn:aws:iam::*account-ID*:role/*role-name*

CreateReport1

アクション: codebuild:CreateReport

テストレポートを作成するために必要です。

リソース: arn:aws:codebuild:*region-ID*:*account-ID*:report-group/*report-group-name*

CreateReportGroup

アクション: codebuild:CreateReportGroup

レポートグループを作成するために必要です。

リソース: arn:aws:codebuild:*region-ID*:*account-ID*:report-group/*report-group-name*

CreateWebhook

アクション: codebuild:CreateWebhook

ウェブフックを作成するために必要です。

リソース: arn:aws:codebuild:*region-ID*:*account-ID*:project/*project-name*

DeleteProject

アクション: codebuild:DeleteProject

CodeBuild プロジェクトを削除するために必要です。

リソース: arn:aws:codebuild:*region-ID*:*account-ID*:project/*project-name*

DeleteReport

アクション: codebuild:DeleteReport

レポートを削除するのに必要です。

リソース: arn:aws:codebuild:*region-ID*:*account-ID*:report-group/*report-group-name*

DeleteReportGroup

アクション: codebuild:DeleteReportGroup

レポートグループを削除するのに必要です。

リソース: arn:aws:codebuild:*region-ID*:*account-ID*:report-group/*report-group-name*

DeleteSourceCredentials

アクション: codebuild:DeleteSourceCredentials

、 GitHub Enterprise Server GitHub、または Bitbucket リポジトリの認証情報を含むSourceCredentialsInfoオブジェクトのセットを削除するために必要です。

リソース: *

DeleteWebhook

アクション: codebuild:DeleteWebhook

ウェブフックを作成するために必要です。

リソース: arn:aws:codebuild:*region-ID*:*account-ID*:project/*project-name*

DescribeTestCases

アクション: codebuild:DescribeTestCases

ページ分割されたテストケースのリストを返すために必要です。

リソース: arn:aws:codebuild:*region-ID*:*account-ID*:report-group/*report-group-name*

ImportSourceCredentials

アクション: codebuild:ImportSourceCredentials

、 GitHub Enterprise Server GitHub、または Bitbucket リポジトリの認証情報に関する情報を含むSourceCredentialsInfoオブジェクトのセットをインポートするために必要です。

リソース: *

InvalidateProjectCache

アクション: codebuild:InvalidateProjectCache

プロジェクトのキャッシュをリセットするために必要です。

リソース: arn:aws:codebuild:*region-ID*:*account-ID*:project/*project-name*

ListBuildBatches

アクション: codebuild:ListBuildBatches

ビルドバッチ ID のリストを取得するために必要です。

リソース: *

ListBuildBatchesForProject

アクション: codebuild:ListBuildBatchesForProject

特定のプロジェクトのビルドバッチ ID のリストを取得するために必要です。

リソース: arn:aws:codebuild:*region-ID*:*account-ID*:project/*project-name*

ListBuilds

アクション: codebuild:ListBuilds

ビルド ID のリストを取得するのに必要です。

リソース: *

ListBuildsForProject

アクション: codebuild:ListBuildsForProject

ビルドプロジェクトのビルド ID のリストを取得するために必要です。

リソース: arn:aws:codebuild:*region-ID*:*account-ID*:project/*project-name*

ListCuratedEnvironmentImages

アクション: codebuild:ListCuratedEnvironmentImages

AWS CodeBuildによって管理されるすべての Docker イメージに関する情報を取得するのに必要です。

リソース: * (必須ですが、アドレスで呼び出せる AWS リソースは参照しません)

ListProjects

アクション: codebuild:ListProjects

ビルドプロジェクト名のリストを取得するのに必要です。

リソース: *

ListReportGroups

アクション: codebuild:ListReportGroups

レポートグループのリストを取得するために必要です。

リソース: *

ListReports

アクション: codebuild:ListReports

レポートリストを取得するために必要です。

リソース: *

ListReportsForReportGroup

アクション: codebuild:ListReportsForReportGroup

レポートグループのレポートのリストを取得するために必要です。

リソース: arn:aws:codebuild:*region-ID*:*account-ID*:report-group/*report-group-name*

RetryBuild

アクション: codebuild:RetryBuild

ビルドを再試行するのに必要です。

リソース: arn:aws:codebuild:*region-ID*:*account-ID*:project/*project-name*

StartBuild

アクション: codebuild:StartBuild

ビルドの実行を開始するために必要です。

リソース: arn:aws:codebuild:*region-ID*:*account-ID*:project/*project-name*

StopBuild

アクション: codebuild:StopBuild

実行中のビルドを停止しようとするのに必要です。

リソース: arn:aws:codebuild:*region-ID*:*account-ID*:project/*project-name*

UpdateProject

アクション: codebuild:UpdateProject、iam:PassRole

ビルドに関する情報を変更するのに必要です。

リソース:

- arn:aws:codebuild:*region-ID*:*account-ID*:project/*project-name*
- arn:aws:iam::*account-ID*:role/*role-name*

UpdateProjectVisibility

アクション: codebuild:UpdateProjectVisibility、iam:PassRole

プロジェクトのビルドの公開可視性を変更するために必要です。

リソース:

- arn:aws:codebuild:*region-ID*:*account-ID*:project/*project-name*
- arn:aws:iam::*account-ID*:role/*role-name*

UpdateReport[1]

アクション: codebuild:UpdateReport

テストレポートを作成または更新するために必要です。

リソース: arn:aws:codebuild:*region-ID*:*account-ID*:report-group/*report-group-name*

UpdateReportGroup

アクション: codebuild:UpdateReportGroup

レポートグループを更新するために必要です。

リソース: arn:aws:codebuild:*region-ID*:*account-ID*:report-group/*report-group-name*

UpdateWebhook

アクション: codebuild:UpdateWebhook

Webhook を更新するために必要です。

リソース: arn:aws:codebuild:*region-ID*:*account-ID*:project/*project-name*

[1] アクセス許可にのみ使用されます。このアクションに API はありません。

タグを使用した AWS CodeBuild リソースへのアクセスのコントロール

IAM ポリシーステートメントの条件は、 CodeBuild プロジェクトベースのアクションへのアクセス許可を指定するために使用できる構文の一部です。プロジェクトに関連付けられたタグに基づいてプ

ロジェクトに対するアクションを許可または拒否するポリシーを作成し、これらのポリシーを、ユーザーの管理用に設定した IAM グループに適用できます。コンソールまたは を使用してプロジェクトにタグを適用する方法については AWS CLI、「」を参照してください でのビルドプロジェクトの作成AWS CodeBuild。 CodeBuild SDK を使用したタグの適用については、 CodeBuild API リファレンスのCreateProject「」および「タグ」を参照してください。タグを使用してリソースへのアクセスを制御する方法については、IAM ユーザーガイドの AWS「リソースタグを使用した AWS リソースへのアクセスの制御」を参照してください。

Example 例 1: リソースタグに基づいて CodeBuild プロジェクトアクションを制限する

次の例では、キー BatchGetProjects とキー値 Environment のタグが付いているプロジェクトに対するすべての Production アクションを拒否します。ユーザーの管理者は、この IAM ポリシーをマネージド型のユーザーポリシーに加えて、承認されないユーザーにアタッチする必要があります。aws:ResourceTag 条件キーを使用して、リソースへのアクセスをリソースタグに基づいてコントロールします。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": [
        "codebuild:BatchGetProjects"
      ],
      "Resource": "*",
      "Condition": {
        "ForAnyValue:StringEquals": {
          "aws:ResourceTag/Environment": "Production"
        }
      }
    }
  ]
}
```

Example 例 2: リクエストタグに基づいて CodeBuild プロジェクトアクションを制限する

次のポリシーでは、リクエスト内のタグのキーが CreateProject で、キー値が Environment である場合、ユーザーに Production アクションへのアクセス許可を拒否します。さらに、このポリシーでは、aws:TagKeys 条件キーを使用して、リクエスト内のタグのキーが UpdateProject である場合に、Environment を許可しないことにより、これらの承認されないユーザーにプロジェク

トの変更を禁止します。管理者は、これらのアクションの実行を承認されないユーザーに、マネージド型のユーザーポリシーに加えて、この IAM ポリシーをアタッチする必要があります。この aws:RequestTag 条件キーを使用して、IAM リクエストで渡すことができるタグをコントロールします

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": [
        "codebuild:CreateProject"
      ],
      "Resource": "*",
      "Condition": {
        "ForAnyValue:StringEquals": {
          "aws:RequestTag/Environment": "Production"
        }
      }
    },
    {
      "Effect": "Deny",
      "Action": [
        "codebuild:UpdateProject"
      ],
      "Resource": "*",
      "Condition": {
        "ForAnyValue:StringEquals": {
          "aws:TagKeys": ["Environment"]
        }
      }
    }
  ]
}
```

Example 例 3: リソースタグに基づいてレポートグループのアクションを拒否または許可する

CodeBuild リソース (プロジェクトおよびレポートグループ) に関連付けられた AWS タグに基づいてリソース (プロジェクトおよびレポートグループ) に対するアクションを許可または拒否するポリシーを作成し、それらのポリシーをユーザーの管理用に設定した IAM グループに適用できます。例えば、 AWS タグキー Statusとキー値が のレポートグループに対するすべての CodeBuild アクションを拒否するポリシーを作成しSecret、一般的なデベロッパー (*######*) 用に作成した IAM

グループにそのポリシーを適用できます。次に、上記のタグ付けされたレポートグループに対して作業する開発者が一般的な *Developers* グループのメンバーではなく、代わりに制限されたポリシーが適用されていない別の IAM グループ (SecretDevelopers) に属していることを確認する必要があります。

次の例では、 キーStatusと のキー値でタグ付けされたレポートグループに対するすべての CodeBuild アクションを拒否しますSecret。

```
{
  "Version": "2012-10-17",
  "Statement" : [
    {
      "Effect" : "Deny",
      "Action" : [
        "codebuild:BatchGetReportGroups,"
        "codebuild:CreateReportGroup",
        "codebuild:DeleteReportGroup",
        "codebuild:ListReportGroups",
        "codebuild:ListReportsForReportGroup",
        "codebuild:UpdateReportGroup"
       ]
      "Resource" : "*",
      "Condition" : {
         "StringEquals" : "aws:ResourceTag/Status": "Secret"
        }
    }
  ]
}
```

Example 例 4: リソースタグ AWSCodeBuildDeveloperAccess に基づいて CodeBuild アクションをに制限する

特定のタグが付けられていないすべてのレポートグループおよびプロジェクトに対する CodeBuild アクションを許可するポリシーを作成できます。たとえば、以下のポリシーでは、指定したタグが付けられたものを除くすべてのレポートグループとプロジェクトに AWSCodeBuildDeveloperAccess と同等のアクセス許可を付与します。

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
```

```
            "Effect": "Allow",
            "Action": [
               "codebuild:StartBuild",
               "codebuild:StopBuild",
               "codebuild:BatchGet*",
               "codebuild:GetResourcePolicy",
               "codebuild:DescribeTestCases",
               "codebuild:List*",
               "codecommit:GetBranch",
               "codecommit:GetCommit",
               "codecommit:GetRepository",
               "codecommit:ListBranches",
               "cloudwatch:GetMetricStatistics",
               "events:DescribeRule",
               "events:ListTargetsByRule",
               "events:ListRuleNamesByTarget",
               "logs:GetLogEvents",
               "s3:GetBucketLocation",
               "s3:ListAllMyBuckets"
            ],
            "Resource": "*",
            "Condition": {
               "StringNotEquals": {
                  "aws:ResourceTag/Status": "Secret",
                  "aws:ResourceTag/Team": "Saanvi"
               }
            }
         }
      ]
}
```

コンソールでのリソースの表示

AWS CodeBuild コンソールには、サインインしている AWS リージョンで AWS アカウントのリポジトリのリストを表示するための アクセスListRepositories許可が必要です。このコンソールには、大文字と小文字を区別しない検索をリソースに対して迅速に実行するための [Go to resource (リソースに移動)] 機能も含まれています。この検索は、サインインしている AWS リージョンのアカウント AWS で実行されます。次のリソースは、以下のサービス全体で表示されます。

- AWS CodeBuild: ビルドプロジェクト
- AWS CodeCommit: リポジトリ
- AWS CodeDeploy: アプリケーション

- AWS CodePipeline: パイプライン

この検索をすべてのサービスのリソースにわたって実行するには、次のアクセス権限が必要です。

- CodeBuild: `ListProjects`
- CodeCommit: `ListRepositories`
- CodeDeploy: `ListApplications`
- CodePipeline: `ListPipelines`

あるサービスに対するアクセス権限がない場合、そのサービスのリソースに関して結果は返されません。表示のアクセス権限がある場合でも、表示に対する明示的な `Deny` が設定されているリソースについては、結果が返されません。

のコンプライアンス検証 AWS CodeBuild

サードパーティーの監査者は、さまざまな コンプライアンスプログラム AWS CodeBuild の一環として のセキュリティと AWS コンプライアンスを評価します。これらのプログラムには、SOC、PCI、FedRAMP、HIPAA などがあります。

特定のコンプライアンスプログラムの対象となる AWS のサービスのリストについては、「コンプライアンスAWS プログラムによる 対象範囲内のサービス」を参照してください。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。

サードパーティーの監査レポートは、 を使用してダウンロードできます AWS Artifact。詳細については、AWS 「 Artifact でのレポートのダウンロード」を参照してください。

を使用する際のお客様のコンプライアンス責任 CodeBuild は、お客様のデータの機密性、企業のコンプライアンス目的、適用可能な法律および規制によって決まります。の使用 CodeBuild が HIPAA、PCI、または FedRAMP などの規格に準拠していることを前提としている場合、 は以下を支援するリソース AWS を提供します。

- セキュリティおよび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 にデプロイするための手順を説明します AWS。
- Architecting for HIPAA Security and Compliance ホワイトペーパー – このホワイトペーパーでは、企業が AWS を使用して HIPAA 準拠のアプリケーションを作成する方法について説明します。

- AWS コンプライアンスリソース – このワークブックとガイドのコレクションは、お客様の業界や場所に適用される場合があります。
- AWS Config – この AWS サービスは、自社プラクティス、業界ガイドライン、規制に対するリソース設定の準拠状態を評価します。
- AWS Security Hub — を使用して、セキュリティのベストプラクティス AWS CodeBuild に関連する の使用状況をモニタリングしますAWS Security Hub。Security Hub は、セキュリティコントロールを使用してリソース設定とセキュリティ標準を評価し、お客様がさまざまなコンプライアンスフレームワークに準拠できるようサポートします。Security Hub を使用して CodeBuild リソースを評価する方法の詳細については、「 ユーザーガイド」の「 AWS CodeBuild コントロール AWS Security Hub 」を参照してください。

の耐障害性 AWS CodeBuild

AWS 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に構築されています。 AWS リージョンは、低レイテンシー、高スループット、および高度の冗長ネットワークで接続されている複数の物理的に独立および隔離されたアベイラビリティーゾーンを提供します。アベイラビリティーゾーンでは、アベイラビリティーゾーン間で中断せずに、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、およびスケーラビリティが優れています。

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。

のインフラストラクチャセキュリティ AWS CodeBuild

マネージドサービスである AWS CodeBuild は グローバル AWS ネットワークセキュリティで保護されています。 AWS セキュリティサービスと がインフラストラクチャ AWS を保護する方法については、AWS 「 クラウドセキュリティ」を参照してください。インフラストラクチャセキュリティのベストプラクティスを使用して AWS 環境を設計するには、「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。

が AWS 公開した API コールを使用して、ネットワーク CodeBuild 経由で 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 は必須で TLS 1.3 がお勧めです。

- DHE (楕円ディフィー・ヘルマン鍵共有) や ECDHE (楕円曲線ディフィー・ヘルマン鍵共有) などの完全前方秘匿性 (PFS) による暗号スイート。これらのモードは、Java 7 以降など、ほとんどの最新システムでサポートされています。

また、リクエストには、アクセスキー ID と、IAM プリンシパルに関連付けられているシークレットアクセスキーを使用して署名する必要があります。または、AWS Security Token Service (AWS STS) を使用して、一時的なセキュリティ認証情報を生成し、リクエストに署名することもできます。

でソースプロバイダーにアクセスする CodeBuild

GitHub または GitHub Enterprise Server では、個人用アクセストークンまたは OAuth アプリを使用してソースプロバイダーにアクセスします。Bitbucket では、アクセストークン、アプリパスワード、または OAuth アプリのいずれかを使用して、ソースプロバイダーにアクセスします。

> **Note**
>
> GitLab およびセルフマネージド型の GitLab ソースプロバイダーには、 CodeBuild から直接アクセスされるのではなく、 を通じて直接アクセスされます AWS CodeConnections。

トピック

- GitHub および GitHub Enterprise Server アクセストークン
- GitHub OAuth アプリ
- Bitbucket アプリのパスワードまたはアクセストークン
- Bitbucket OAuth アプリ

GitHub および GitHub Enterprise Server アクセストークン

アクセストークンの前提条件

開始する前に、 GitHub アクセストークンに適切なアクセス許可スコープを追加する必要があります。

の場合 GitHub、個人用アクセストークンには次のスコープが必要です。

- repo: プライベートリポジトリのフルコントロールを許可します。
- repo:status: パブリックおよびプライベートリポジトリのコミットステータスへの読み取り/書き込みアクセスを許可します。
- admin:repo_hook: リポジトリフックのフルコントロールを許可します。このスコープは、トークンに `repo` スコープがある場合は必要ありません。

詳細については、 GitHub ウェブサイトの「OAuth アプリのスコープを理解する」を参照してください。

きめ細かな個人用アクセストークンを使用している場合、ユースケースによっては、個人用アクセストークンに次のアクセス許可が必要になる場合があります。

- コンテンツ: 読み取り専用 : プライベートリポジトリへのアクセスを許可します。このアクセス許可は、プライベートリポジトリをソースとして使用している場合に必要です。
- コミットステータス: 読み取りおよび書き込み: コミットステータスを作成するアクセス許可を付与します。このアクセス許可は、プロジェクトでウェブフックがセットアップされている場合、またはビルドステータスレポート機能が有効になっている場合に必要です。
- ウェブフック: 読み取りと書き込み: ウェブフックを管理するアクセス許可を付与します。このアクセス許可は、プロジェクトにウェブフックが設定されている場合に必要です。
- プルリクエスト: 読み取り専用 : プルリクエストにアクセスするためのアクセス許可を付与します。このアクセス許可は、ウェブフックにプルリクエストイベントに対する`FILE_PATH`フィルターがある場合に必要です。
- 管理: 読み取りと書き込み: このアクセス許可は、 でセルフホスト GitHub アクションランナー機能を使用している場合に必要です CodeBuild。詳細については、「リポジトリの登録トークンを作成する」および「」を参照してくださいでセルフホスト GitHub アクションランナーを設定する AWS CodeBuild。

> **Note**
>
> 組織リポジトリにアクセスする場合は、アクセストークンのリソース所有者として組織を指定してください。

詳細については、 GitHub ウェブサイトの「きめ細かな個人用アクセストークンに必要なアクセス許可」を参照してください。

アクセストークン GitHub で接続する (コンソール)

コンソールを使用してアクセストークン GitHub を使用してプロジェクトを に接続するには、プロジェクトを作成するときに次の操作を行います。詳細については、「ビルドプロジェクトの作成 (コンソール)」を参照してください。

1. ソースプロバイダー で、 を選択しますGitHub。
2. リポジトリ で、 GitHub 個人用アクセストークン で接続 を選択します。
3. GitHub 個人用アクセストークン に GitHub、個人用アクセストークンを入力します。
4. [トークンの保存] を選択します。

アクセストークン GitHub で接続する (CLI)

を使用して、アクセストークン GitHub を使用してプロジェクトを AWS CLI に接続するには、次の手順に従います。 AWS CLI で を使用する方法については AWS CodeBuild、「」を参照してくださいコマンドラインリファレンス。

1. import-source-credentials コマンドを実行します。

   ```
   aws codebuild import-source-credentials --generate-cli-skeleton
   ```

   JSON 形式のデータが出力に表示されます。 AWS CLI がインストールされているローカルコンピュータまたはインスタンス上の場所にあるファイル (など*import-source-credentials.json*) にデータをコピーします。コピーされたデータを次のように変更して、結果を保存します。

   ```
   {
     "serverType": "server-type",
     "authType": "auth-type",
     "shouldOverwrite": "should-overwrite",
     "token": "token",
     "username": "username"
   }
   ```

   以下に置き換えます。

   - *server-type*: 必須値。この認証情報に使用されるソースプロバイダー。有効な値は GITHUB あるいは GITHUB_ENTERPRISE です。

- *auth-type*: 必須値。 GitHub または GitHub Enterprise Server リポジトリへの接続に使用される認証のタイプ。有効な値は、PERSONAL_ACCESS_TOKEN と BASIC_AUTH です。CodeBuild API を使用して OAUTH 接続を作成することはできません。代わりに CodeBuild コンソールを使用する必要があります。
- *should-overwrite*: オプションの値。リポジトリソースの認証情報が上書きされないようにするには、`false` に設定します。リポジトリソースの認証情報を上書きするには、`true` に設定します。デフォルト値は `true` です。
- *token*: 必須値。 GitHub または GitHub Enterprise Server の場合、これは個人用アクセストークンです。
- *username*: オプションの値。このパラメータは、 GitHub および GitHub Enterprise Server ソースプロバイダーでは無視されます。

2. アカウントをアクセストークンに接続するには、ステップ 1 で保存した `import-source-credentials.json` ファイルが含まれるディレクトリに切り替え、もう一度 import-source-credentials コマンドを実行します。

```
aws codebuild import-source-credentials --cli-input-json file://import-source-credentials.json
```

JSON 形式のデータが、Amazon リソースネーム (ARN) を持つ出力に表示されます。

```
{
  "arn": "arn:aws:codebuild:region:account-id:token/server-type"
}
```

> **Note**
>
> 同じサーバータイプと認証タイプを持つ import-source-credentials コマンドを 2 回目に実行した場合、保存されたアクセストークンが更新されます。

アカウントがアクセストークンに接続されたら、 `create-project` を使用して CodeBuild プロジェクトを作成できます。詳細については、「ビルドプロジェクトの作成 (AWS CLI)」を参照してください。

3. 接続されたアクセストークンを表示するには、list-source-credentials コマンドを実行します。

```
aws codebuild list-source-credentials
```

JSON 形式 sourceCredentialsInfos オブジェクトが出力に表示されます。

```
{
    "sourceCredentialsInfos": [
        {
            "authType": "auth-type",
            "serverType": "server-type",
            "arn": "arn"
        }
    ]
}
```

sourceCredentialsObject には、接続されたソース認証情報のリストが含まれています。

- authType は、認証情報により使用される認証のタイプです。これは、OAUTH、BASIC_AUTH、または PERSONAL_ACCESS_TOKEN です。
- serverType は、ソースプロバイダーのタイプです。これは、GITHUB、GITHUB_ENTERPRISE、または BITBUCKET です。
- arn は、トークンの ARN です。

4. ソースプロバイダーから切断してそのアクセストークンを削除するには、その ARN を使用して delete-source-credentials コマンドを実行します。

```
aws codebuild delete-source-credentials --arn arn-of-your-credentials
```

削除された認証情報の ARN とともに JSON 形式のデータが返されます。

```
{
  "arn": "arn:aws:codebuild:region:account-id:token/server-type"
}
```

GitHub OAuth アプリ

OAuth GitHub を使用して接続する (コンソール)

コンソールを使用して OAuth アプリ GitHub を使用してプロジェクトを に接続するには、プロジェクトを作成するときに次の操作を行います。詳細については、「ビルドプロジェクトの作成 (コンソール)」を参照してください。

1. ソースプロバイダー で、 を選択しますGitHub。
2. リポジトリ で、OAuth を使用して接続 を選択します。
3. への接続 GitHub、ログイン、アカウントの承認を選択します。
4. 確認 を選択して GitHub アカウント CodeBuild に接続します。
5. GitHub リポジトリ に GitHubリポジトリリンクを入力します。

認可された OAuth アプリケーションを確認するには、 のアプリケーションに移動し GitHub、aws-code Suite AWS CodeBuild (*region*)が所有する という名前のアプリケーションがリストされていることを確認します。

Bitbucket アプリのパスワードまたはアクセストークン

前提条件

開始する前に、Bitbucket アプリのパスワードまたはアクセストークンに適切なアクセス許可スコープを追加する必要があります。

Bitbucket の場合、アプリパスワードまたはアクセストークンには次のスコープが必要です。

- repository:read: 承認側ユーザーがアクセスできるすべてのリポジトリへの読み取りアクセスを許可します。
- pullrequest:read: プルリクエストの読み取りアクセスを許可します。プロジェクトに Bitbucket ウェブフックがある場合、アプリパスワードまたはアクセストークンにはこのスコープが必要です。
- webhook: Webhook へのアクセスを許可します。プロジェクトにウェブフックオペレーションがある場合、アプリパスワードまたはアクセストークンにはこのスコープが必要です。

詳細については、Bitbucket ウェブサイトの「Scopes for Bitbucket Cloud REST API」と「OAuth on Bitbucket Cloud」を参照してください。

アプリケーションパスワードで Bitbucket へ接続する (コンソール)

コンソールを使用し、アクセストークンを使用してプロジェクトを Bitbucket に接続するには、プロジェクトを作成するときに以下の操作を実行します。詳細については、ビルドプロジェクトの作成 (コンソール) を参照してください。

1. [ソースプロバイダー] で、[Bitbucket] を選択します。

   > **Note**
   >
   > CodeBuild は Bitbucket サーバーをサポートしていません。

2. [リポジトリ] で、[Connect with a Bitbucket app password (Bitbucket アプリパスワードで接続する)] を選択します。
3. [Bitbucket username (Bitbucket ユーザー名)] に、Bitbucket ユーザー名を入力します。
4. [Bitbucket app password (Bitbucket のアプリパスワード)] に、Bitbucket アプリパスワードを入力します。
5. [Save Bitbucket credentials (Bitbucket 認証情報の保存)] を選択します。

Bitbucket をアクセストークンに接続する (コンソール)

コンソールを使用してアクセストークンを使用してプロジェクトを Bitbucket に接続するには、プロジェクトを作成するときに次の操作を行います。詳細については、ビルドプロジェクトの作成 (コンソール) を参照してください。

1. [ソースプロバイダー] で、[Bitbucket] を選択します。

   > **Note**
   >
   > CodeBuild は Bitbucket サーバーをサポートしていません。

2. リポジトリ で、Bitbucket アクセストークン で接続 を選択します。
3. Bitbucket アクセストークン で、Bitbucket アクセストークンを入力します。
4. [トークンの保存] を選択します。

Bitbucket をアプリパスワードまたはアクセストークンに接続する (CLI)

を使用して、アプリパスワードまたはアクセストークンを使用してプロジェクトを Bitbucket AWS CLI に接続するには、次の手順に従います。 AWS CLI で を使用する方法については AWS CodeBuild、「」を参照してくださいコマンドラインリファレンス。

1. import-source-credentials コマンドを実行します。

```
aws codebuild import-source-credentials --generate-cli-skeleton
```

JSON 形式のデータが出力に表示されます。 AWS CLI がインストールされているローカルコンピュータまたはインスタンス上の場所にあるファイル (など*import-source-credentials.json*) にデータをコピーします。コピーされたデータを次のように変更して、結果を保存します。

```
{
  "serverType": "BITBUCKET",
  "authType": "auth-type",
  "shouldOverwrite": "should-overwrite",
  "token": "token",
  "username": "username"
}
```

以下に置き換えます。

- *auth-type*: 必須値。Bitbucket リポジトリに接続するために使用される認証のタイプ。有効な値は、PERSONAL_ACCESS_TOKEN と BASIC_AUTH です。 CodeBuild API を使用して OAUTH 接続を作成することはできません。代わりに CodeBuild コンソールを使用する必要があります。
- *should-overwrite*: オプションの値。リポジトリソースの認証情報が上書きされないようにするには、`false` に設定します。リポジトリソースの認証情報を上書きするには、`true` に設定します。デフォルト値は `true` です。
- *token*: 必須値。Bitbucket の場合、これはアクセストークンまたはアプリパスワードのいずれかです。
- *username*: オプションの値。`authType` が BASIC_AUTH の場合の Bitbucket ユーザー名。その他のタイプのソースプロバイダーまたは接続では、このパラメータは無視されます。

2. アカウントをアプリパスワードまたはアクセストークンに接続するには、ステップ 1 で保存したimport-source-credentials.jsonファイルを含むディレクトリに切り替えて、 import-source-credentials コマンドを再度実行します。

```
aws codebuild import-source-credentials --cli-input-json file://import-source-credentials.json
```

JSON 形式のデータが、Amazon リソースネーム (ARN) を持つ出力に表示されます。

```
{
  "arn": "arn:aws:codebuild:region:account-id:token/server-type"
}
```

> **Note**
>
> 同じサーバータイプと認証タイプを持つ import-source-credentials コマンドを 2 回目に実行した場合、保存されたアクセストークンが更新されます。

アカウントがアプリパスワードに接続されたら、 create-project を使用して CodeBuild プロジェクトを作成できます。詳細については、「ビルドプロジェクトの作成 (AWS CLI)」を参照してください。

3. 接続されているアプリのパスワードまたはアクセストークンを表示するには、 list-source-credentials コマンドを実行します。

```
aws codebuild list-source-credentials
```

JSON 形式 sourceCredentialsInfos オブジェクトが出力に表示されます。

```
{
    "sourceCredentialsInfos": [
        {
            "authType": "auth-type",
            "serverType": "BITBUCKET",
            "arn": "arn"
        }
    ]
}
```

sourceCredentialsObject には、接続されたソース認証情報のリストが含まれています。

- authType は、認証情報により使用される認証のタイプです。これは、OAUTH、BASIC_AUTH、または PERSONAL_ACCESS_TOKEN です。
- arn は、トークンの ARN です。

4. ソースプロバイダーから切断し、アプリパスワードまたはアクセストークンを削除するには、その ARN を指定して delete-source-credentials コマンドを実行します。

```
aws codebuild delete-source-credentials --arn arn-of-your-credentials
```

削除された認証情報の ARN とともに JSON 形式のデータが返されます。

```
{
  "arn": "arn:aws:codebuild:region:account-id:token/server-type"
}
```

Bitbucket OAuth アプリ

OAuth を使用して Bitbucket を接続する (コンソール)

コンソールを使用して OAuth アプリを使用してプロジェクトを Bitbucket に接続するには、プロジェクトを作成するときに次の操作を行います。詳細については、ビルドプロジェクトの作成 (コンソール) を参照してください。

1. [ソースプロバイダー] で、[Bitbucket] を選択します。
2. リポジトリ で、OAuth を使用して接続 を選択します。
3. 「Bitbucket への接続」、「ログイン」、「 アカウントの承認」を選択します。
4. 確認 を選択して Bitbucket アカウント CodeBuild に接続します。
5. Bitbucket リポジトリ に、Bitbucket リポジトリリンクを入力します。

認可された OAuth アプリを確認するには、Bitbucket のアプリケーション認可に移動し、 という名前のアプリケーションAWS CodeBuild (*region*)がリストされていることを確認します。

サービス間での不分別な代理処理の防止

混乱した代理問題は、アクションを実行するためのアクセス許可を持たないエンティティが、より特権のあるエンティティにアクションの実行を強制できてしまう場合に生じる、セキュリティ上の問題です。では AWS、サービス間でなりすましを行うと、混乱した代理問題が発生する可能性があります。サービス間でのなりすましは、1 つのサービス (呼び出し元サービス) が、別のサービス (呼び出し対象サービス) を呼び出すときに発生する可能性があります。呼び出し元サービスは、本来ならアクセスすることが許可されるべきではない方法でその許可を使用して、別のお客様のリソースに対する処理を実行するように操作される場合があります。これを防ぐために、 AWS には、アカウント内のリソースへのアクセス権が付与されたサービスプリンシパルですべてのサービスのデータを保護するために役立つツールが用意されています。

リソースポリシーで aws:SourceArnおよび aws:SourceAccount グローバル条件コンテキストキーを使用して、 が別のサービス AWS CodeBuild に付与するリソースへのアクセス許可を制限することをお勧めします。クロスサービスアクセスにリソースを 1 つだけ関連付けたい場合は、aws:SourceArn を使用します。そのアカウント内のリソースをクロスサービスの使用に関連付けることを許可する場合は、aws:SourceAccount を使用します。

混乱した代理問題から保護するための最も効果的な方法は、リソースの完全な ARN を指定して、aws:SourceArn グローバル条件コンテキストキーを使用することです。リソースの完全な ARN が不明な場合や、複数のリソースを指定する場合には、グローバルコンテキスト条件キー aws:SourceArn で、ARN の未知部分を示すためにワイルドカード文字 (*) を使用します。例えば、arn:aws:codebuild:*:*123456789012*:* です。

aws:SourceArn の値に Amazon S3 バケット ARN などのアカウント ID が含まれていない場合は、両方のグローバル条件コンテキストキーを使用して、アクセス許可を制限する必要があります。

の値は CodeBuild プロジェクト ARN aws:SourceArnである必要があります。

次の例は、 で aws:SourceArnおよび aws:SourceAccount グローバル条件コンテキストキーを使用して、混乱した代理問題 CodeBuild を回避する方法を示しています。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "codebuild.amazonaws.com"
            },
```

```
            "Action": "sts:AssumeRole",
            "Condition": {
                "StringEquals": {
                    "aws:SourceArn": "arn:aws:codebuild:region-ID:account-
ID:project/project-name"
                }
            }
        }
    ]
}
```

高度なトピック

このセクションでは、経験豊富な AWS CodeBuild ユーザーに役立ついくつかの高度なトピックを示します。

トピック

詳細設定

「コンソールを使用した開始方法」の手順に従って初めて AWS CodeBuild にアクセスする場合、このトピックの情報は必要ないと考えられます。ただし、CodeBuild を引き続き使用する場合、組織内の IAM グループやユーザーに CodeBuild へのアクセスを付与したり、CodeBuild にアクセスするために IAM の既存のサービスロールや AWS KMS keys を変更したり、CodeBuild にアクセスするために組織のワークステーション全体で AWS CLI をセットアップしたりすることがあります。このトピックでは、関連するセットアップ手順の実行方法について説明します。

AWS アカウントは既にあるものとします。ただし、まだアカウントがない場合は、http://aws.amazon.com に移動し、[Sign In to the Console] を選択してオンラインの指示に従ってください。

トピック

IAM グループまたはユーザーに CodeBuild アクセス許可を追加する

AWS CodeBuild にアクセスするには、IAM グループまたはユーザーにアクセス許可を追加する必要があります。このセクションでは、IAM コンソールまたは AWS CLI でこれを行う方法について説明します。

AWS ルートアカウント (非推奨) を使用するか、AWS アカウントの管理者ユーザーを使用して CodeBuild にアクセスする場合、以下の手順に従う必要はありません。

AWS ルートアカウントと管理者ユーザーについては、ユーザーガイドの「AWS アカウントルートユーザー」および「最初の AWS アカウントルートユーザーおよびグループの作成」を参照してください。

IAM グループまたはユーザーに CodeBuild アクセス許可を追加するには (コンソール)

1. IAM コンソール (https://console.aws.amazon.com/iam/) を開きます。

   次のいずれかを使用して、AWS Management Console に既にサインインしている必要があります。

   - AWS ルートアカウント。これは推奨されません。詳細については、ユーザーガイドの「AWS アカウントルートユーザー」を参照してください。
   - AWS アカウントの管理者ユーザー。詳細については、ユーザーガイドの「最初の AWS アカウントルートユーザーおよびグループの作成」を参照してください。
   - 以下の最小限のアクションを実行するアクセス許可を持つ AWS アカウントのユーザー。

     ```
     iam:AttachGroupPolicy
     iam:AttachUserPolicy
     iam:CreatePolicy
     iam:ListAttachedGroupPolicies
     iam:ListAttachedUserPolicies
     iam:ListGroups
     iam:ListPolicies
     iam:ListUsers
     ```

     詳細については、ユーザーガイドの「IAM ポリシーの概要」を参照してください。
2. ナビゲーションペインで、[ポリシー] を選択します。
3. カスタムセットの AWS CodeBuild アクセス許可を IAM グループまたは IAM ユーザーに追加するには、この手順のステップ 4 に進んでください。

IAM グループや IAM ユーザーにデフォルトの CodeBuild アクセス許可セットを追加するには、[Policy Type]、[AWS Managed] の順に選択し、以下の操作を行います。

- CodeBuild へのフルアクセス許可を追加するには、[AWSCodeBuildAdminAccess] という名前のボックスを選択し、[ポリシーアクション]、[アタッチ] の順に選択します。対象の IAM グループやユーザーの横にあるボックスを選択し、[Attach Policy] (ポリシーのアタッチ) を選択します。AmazonS3ReadOnlyAccess ポリシーおよび IAMFullAccess ポリシーに対して、この操作を繰り返します。
- ビルドプロジェクトの管理を除くすべてについて CodeBuild へのアクセス許可に追加するには、[AWSCodeBuildDeveloperAccess] という名前のボックスを選択し、[Policy Actions] (ポリシーアクション)、[Attach] (アタッチ) の順に選択します。対象の IAM グループやユーザーの横にあるボックスを選択し、[Attach Policy] (ポリシーのアタッチ) を選択します。AmazonS3ReadOnlyAccess ポリシーに対して、この操作を繰り返します。
- CodeBuild への読み取り専用アクセス許可を追加するには、[AWSCodeBuildReadOnlyAccess] という名前のボックスを選択します。対象の IAM グループやユーザーの横にあるボックスを選択し、[Attach Policy] (ポリシーのアタッチ) を選択します。AmazonS3ReadOnlyAccess ポリシーに対して、この操作を繰り返します。

これで、IAM グループまたはユーザーに CodeBuild へのデフォルトのアクセス許可セットが追加されました。この手順の残りの手順をスキップします。

4. [Create Policy] (ポリシーの作成) を選択します。
5. [Create Policy] ページで、[Create Your Own Policy] の横にある [Select] を選択します。
6. [ポリシーの確認] ページの [ポリシー名] に、ポリシーの名前 (**`CodeBuildAccessPolicy`** など) を入力します。別の名前を使用する場合は、この手順全体でそれを使用してください。
7. [ポリシードキュメント] に、次のように入力し、[ポリシーの作成] を選択します。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "CodeBuildAccessPolicy",
      "Effect": "Allow",
      "Action": [
        "codebuild:*"
      ],
      "Resource": "*"
```

```
    },
    {
      "Sid": "CodeBuildRolePolicy",
      "Effect": "Allow",
      "Action": [
        "iam:PassRole"
      ],
      "Resource": "arn:aws:iam::account-ID:role/role-name"
    },
    {
      "Sid": "CloudWatchLogsAccessPolicy",
      "Effect": "Allow",
      "Action": [
        "logs:FilterLogEvents",
        "logs:GetLogEvents"
      ],
      "Resource": "*"
    },
    {
      "Sid": "S3AccessPolicy",
      "Effect": "Allow",
      "Action": [
        "s3:CreateBucket",
        "s3:GetObject",
        "s3:List*",
        "s3:PutObject"
      ],
      "Resource": "*"
    },
    {
      "Sid": "S3BucketIdentity",
      "Effect": "Allow",
      "Action": [
        "s3:GetBucketAcl",
        "s3:GetBucketLocation"
      ],
      "Resource": "*"
    }
  ]
}
```

> **Note**
>
> このポリシーにより、すべての CodeBuild アクションへのアクセスが許可され、多数の AWS リソースへのアクセスが許可される可能性があります。アクセス許可を特定の CodeBuild アクションに限定するには、CodeBuild ポリシーステートメントの `codebuild:*` の値を変更します。詳細については、「ID およびアクセス管理」を参照してください。特定の AWS リソースへのアクセスを制限するには、`Resource` オブジェクトの値を変更します。詳細については、「ID およびアクセス管理」を参照してください。
>
> `CodeBuildRolePolicy` ステートメントは、ビルドプロジェクトの作成または変更を許可するために必要です。

8. ナビゲーションペインで、[Groups] または [Users] を選択します。
9. グループまたはユーザーのリストで、CodeBuild アクセス許可を追加する IAM グループまたは IAM ユーザーの名前を選択します。
10. グループの場合は、グループ設定ページの [アクセス許可] タブで [管理ポリシー] を展開し、[ポリシーのアタッチ] を選択します。

    ユーザーの場合は、ユーザー設定ページの [Permissions] タブで、[Add permissions] を選択します。
11. グループの場合は、[Attach Policy] (ポリシーのアタッチ) ページで [CodeBuildAccessPolicy]、[Attach Policy] (ポリシーのアタッチ) の順に選択します。

    ユーザーの場合は、[Add permissions] (アクセス許可の付与) ページで [Attach existing policies directly] (既存のポリシーを直接アタッチ) を選択します。[CodeBuildAccessPolicy] を選択し、[Next: Reivew] (次のステップ: 確認)、[Add permissions] (アクセス権限の追加) の順にクリックします。

IAM グループまたはユーザーに CodeBuild アクセス許可を追加するには (AWS CLI)

1. 前の手順で説明しているように、IAM エンティティのいずれかに対応する AWS アクセスキーと AWS シークレットアクセスキーを使用して AWS CLI が設定されていることを確認します。詳細については、AWS Command Line Interface ユーザーガイドの「AWS Command Line Interface のセットアップ」を参照してください。
2. AWS CodeBuild へのアクセス許可のカスタムセットを IAM グループまたは IAM ユーザーに追加するには、この手順のステップ 3 に進んでください。

IAM グループまたは IAM ユーザーに、CodeBuild アクセス許可のデフォルトセットを追加するには以下を実行します。

IAM グループまたはユーザーのどちらにアクセス許可を追加するかに応じて、以下のいずれかのコマンドを実行します。

```
aws iam attach-group-policy --group-name group-name --policy-arn policy-arn

aws iam attach-user-policy --user-name user-name --policy-arn policy-arn
```

コマンドは 3 回実行する必要があります。group-name または user-name は IAM グループ名またはユーザー名に置き換え、policy-arn は 1 回ごとに以下の各ポリシー Amazon リソースネーム (ARN) に置き換えてください。

- CodeBuild にフルアクセス許可を追加するには、以下のポリシー ARN を使用します。
  - `arn:aws:iam::aws:policy/AWSCodeBuildAdminAccess`
  - `arn:aws:iam::aws:policy/AmazonS3ReadOnlyAccess`
  - `arn:aws:iam::aws:policy/IAMFullAccess`
- ビルドプロジェクトの管理以外のすべてに対して CodeBuild にアクセス許可を追加するには、次のポリシー ARN を使用します。
  - `arn:aws:iam::aws:policy/AWSCodeBuildDeveloperAccess`
  - `arn:aws:iam::aws:policy/AmazonS3ReadOnlyAccess`
- CodeBuild に読み取り専用アクセス許可を追加するには、以下のポリシー ARN を使用します。
  - `arn:aws:iam::aws:policy/AWSCodeBuildReadOnlyAccess`
  - `arn:aws:iam::aws:policy/AmazonS3ReadOnlyAccess`

これで、IAM グループまたはユーザーに CodeBuild へのデフォルトのアクセス許可セットが追加されました。この手順の残りの手順をスキップします。

3. AWS CLI がインストールされているローカルワークステーションまたはインスタンス上の空のディレクトリに、`put-group-policy.json` または `put-user-policy.json` という名前のファイルを作成します。別のファイル名を使用する場合は、この手順全体でそれを使用してください。

```
{
```

```
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "CodeBuildAccessPolicy",
      "Effect": "Allow",
      "Action": [
        "codebuild:*"
      ],
      "Resource": "*"
    },
    {
      "Sid": "CodeBuildRolePolicy",
      "Effect": "Allow",
      "Action": [
        "iam:PassRole"
      ],
      "Resource": "arn:aws:iam::account-ID:role/role-name"
    },
    {
      "Sid": "CloudWatchLogsAccessPolicy",
      "Effect": "Allow",
      "Action": [
        "logs:FilterLogEvents",
        "logs:GetLogEvents"
      ],
      "Resource": "*"
    },
    {
      "Sid": "S3AccessPolicy",
      "Effect": "Allow",
      "Action": [
        "s3:CreateBucket",
        "s3:GetObject",
        "s3:List*",
        "s3:PutObject"
      ],
      "Resource": "*"
    },
    {
      "Sid": "S3BucketIdentity",
      "Effect": "Allow",
      "Action": [
        "s3:GetBucketAcl",
        "s3:GetBucketLocation"
```

```
      ],
      "Resource": "*"
    }
  ]
}
```

> **Note**
>
> このポリシーにより、すべての CodeBuild アクションへのアクセスが許可され、多数の AWS リソースへのアクセスが許可される可能性があります。アクセス許可を特定の CodeBuild アクションに限定するには、CodeBuild ポリシーステートメントの `codebuild:*` の値を変更します。詳細については、「ID およびアクセス管理」を参照してください。特定の AWS リソースへのアクセスを制限するには、関連する `Resource` オブジェクトの値を変更します。詳細については、「ID およびアクセス管理」または特定の AWS サービスのセキュリティドキュメントを参照してください。`CodeBuildRolePolicy` ステートメントは、ビルドプロジェクトの作成または変更を許可するために必要です。

4. ファイルを保存したディレクトリに移動し、以下のいずれかのコマンドを実行します。`CodeBuildGroupAccessPolicy` および `CodeBuildUserAccessPolicy` に異なる値を使用できます。異なる値を使用する場合は、ここでそれらを使用してください。

   IAM グループの場合:

   ```
   aws iam put-group-policy --group-name group-name --policy-name
    CodeBuildGroupAccessPolicy --policy-document file://put-group-policy.json
   ```

   ユーザーの場合:

   ```
   aws iam put-user-policy --user-name user-name --policy-name
    CodeBuildUserAccessPolicy --policy-document file://put-user-policy.json
   ```

   前述のコマンドで、group-name または user-name は、対象の IAM グループまたはユーザーの名前に置き換えます。

CodeBuild サービスロールの作成

AWS CodeBuild サービスロールが必要です。これにより、CodeBuild が、ユーザーに代わって依存 AWS サービスとやり取りできるようになります。CodeBuild または AWS CodePipeline コンソールを使用して、CodeBuild サービスロールを作成できます。詳細については、以下を参照してください。

- ビルドプロジェクトの作成 (コンソール)
- CodeBuild を使用するパイプラインを作成する (CodePipeline コンソール)
- CodeBuild ビルドアクションをパイプラインに追加する (CodePipeline コンソール)
- ビルドプロジェクトの設定の変更 (コンソール)

これらのコンソールを使用する予定がない場合のために、このセクションでは、IAM コンソールまたは AWS CLI を使用して CodeBuild サービスロールを作成する方法について説明します。

> **Important**
>
> CodeBuild は、ユーザーのために実行されるすべての操作でサービスロールを使用します。ユーザーが持つべきではないアクセス権限がロールに含まれる場合、ユーザーのアクセス権限を非意図的にエスカレーションできてしまいます。ロールが最小特権を付与することを確認します。
>
> このページで説明されているサービスロールには、CodeBuild を使用するのに必要な最小権限を付与するポリシーが含まれています。ユースケースに応じて、さらに許可を追加する必要がある場合があります。

CodeBuild サービスロールを作成するには (コンソール)

1. IAM コンソール (https://console.aws.amazon.com/iam/) を開きます。

   次のいずれかを使用して、コンソールに既にサインインしている必要があります。

   - AWS ルートアカウント。これは推奨されません。詳細については、ユーザーガイドの「AWS アカウントルートユーザー」を参照してください。
   - AWS アカウントの管理者ユーザー。詳細については、ユーザーガイドの「最初の AWS アカウントルートユーザーおよびグループの作成」を参照してください。
   - 以下の最小限のアクションを実行するアクセス許可を持つ AWS アカウントのユーザー。

```
iam:AddRoleToInstanceProfile
iam:AttachRolePolicy
iam:CreateInstanceProfile
iam:CreatePolicy
iam:CreateRole
iam:GetRole
iam:ListAttachedRolePolicies
iam:ListPolicies
iam:ListRoles
iam:PassRole
iam:PutRolePolicy
iam:UpdateAssumeRolePolicy
```

詳細については、ユーザーガイドの「IAM ポリシーの概要」を参照してください。

2. ナビゲーションペインで、[ポリシー] を選択します。
3. [Create Policy] (ポリシーの作成) を選択します。
4. [Create Policy] ページで、[JSON] を選択します。
5. [JSON ポリシー] に、次のように入力し、[ポリシーの確認] を選択します。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "CloudWatchLogsPolicy",
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:PutLogEvents"
      ],
      "Resource": "*"
    },
    {
      "Sid": "CodeCommitPolicy",
      "Effect": "Allow",
      "Action": [
        "codecommit:GitPull"
      ],
      "Resource": "*"
    },
```

```
    {
      "Sid": "S3GetObjectPolicy",
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:GetObjectVersion"
      ],
      "Resource": "*"
    },
    {
      "Sid": "S3PutObjectPolicy",
      "Effect": "Allow",
      "Action": [
        "s3:PutObject"
      ],
      "Resource": "*"
    },
    {
      "Sid": "ECRPullPolicy",
      "Effect": "Allow",
      "Action": [
        "ecr:BatchCheckLayerAvailability",
        "ecr:GetDownloadUrlForLayer",
        "ecr:BatchGetImage"
      ],
      "Resource": "*"
    },
    {
      "Sid": "ECRAuthPolicy",
      "Effect": "Allow",
      "Action": [
        "ecr:GetAuthorizationToken"
      ],
      "Resource": "*"
    },
    {
      "Sid": "S3BucketIdentity",
      "Effect": "Allow",
      "Action": [
        "s3:GetBucketAcl",
        "s3:GetBucketLocation"
      ],
      "Resource": "*"
    }
```

```
    ]
}
```

> **Note**
>
> このポリシーに含まれているステートメントでは、多数の AWS リソースへのアクセスが許可される可能性があります。AWS CodeBuild に特定の AWS リソースへのアクセスを制限するには、`Resource` 配列の値を変更します。詳細については、AWS サービスのセキュリティドキュメントを参照してください。

6. [ポリシーの確認] ページで、[ポリシー名] にポリシー名 (**`CodeBuildServiceRolePolicy`** など) を入力し、[ポリシーの作成] を選択します。

> **Note**
>
> 別の名前を使用する場合は、この手順全体でそれを使用してください。

7. ナビゲーションペインで [ロール] を選択します。
8. [ロールの作成] を選択します。
9. [ロールの作成] ページで、[AWS のサービス] が選択された状態で、[CodeBuild]、[次の手順: アクセス許可] の順に選択します。
10. [Attach permissions policies (アクセス権限ポリシーをアタッチする)] ページで、[CodeBuildServiceRolePolicy]、[Next: Review (次へ: 確認)] の順に選択します。
11. [Create role and review (ロールの作成と確認)] ページで、[ロール名] にロールの名前 (**`CodeBuildServiceRole`** など) を入力し、[ロールの作成] を選択します。

CodeBuild サービスロールの作成 (AWS CLI)

1. 前の手順で説明しているように、IAM エンティティのいずれかに対応する AWS アクセスキーと AWS シークレットアクセスキーを使用して AWS CLI が設定されていることを確認します。詳細については、AWS Command Line Interface ユーザーガイドの「AWS Command Line Interface のセットアップ」を参照してください。
2. AWS CLI がインストールされているローカルワークステーションまたはインスタンスの空のディレクトリに、`create-role.json` および `put-role-policy.json` という名前の 2 つのファイルを作成します。別のファイル名を選択した場合は、この手順全体でそれを使用してください。

`create-role.json`:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "codebuild.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

> **Note**
>
> 「混乱した代理」問題に対して自分を守るために aws:SourceAccount および aws:SourceArn 条件キーを使用することをお勧めします。例えば、前述の信頼ポリシーを次の条件ブロックで編集できます。aws:SourceAccount は CodeBuild プロジェクトの所有者で、aws:SourceArn は CodeBuild プロジェクトの ARN です。

サービスロールを AWS アカウントに制限する場合、`create-role.json` は次のようになります。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "codebuild.amazonaws.com"
            },
            "Action": "sts:AssumeRole",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": [
                        "account-ID"
                    ]
```

```
                }
            }
        }
    ]
}
```

サービスロールを特定の CodeBuild プロジェクトに制限する場合、`create-role.json` は次のようになります。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "codebuild.amazonaws.com"
            },
            "Action": "sts:AssumeRole",
            "Condition": {
                "StringEquals": {
                    "aws:SourceArn": "arn:aws:codebuild:region-ID:account-
ID:project/project-name"
                }
            }
        }
    ]
}
```

> **Note**
>
> CodeBuild プロジェクトの名前が不明である、または名前を決定しておらず、特定の ARN パターンに信頼ポリシーの制限が必要な場合は、ARN の該当部分をワイルドカード (*) に置き換えることができます。プロジェクトを作成した後は、信頼ポリシーを更新できます。

`put-role-policy.json`:

```
{
  "Version": "2012-10-17",
  "Statement": [
```

```
    {
      "Sid": "CloudWatchLogsPolicy",
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:PutLogEvents"
      ],
      "Resource": "*"
    },
    {
      "Sid": "CodeCommitPolicy",
      "Effect": "Allow",
      "Action": [
        "codecommit:GitPull"
      ],
      "Resource": "*"
    },
    {
      "Sid": "S3GetObjectPolicy",
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:GetObjectVersion"
      ],
      "Resource": "*"
    },
    {
      "Sid": "S3PutObjectPolicy",
      "Effect": "Allow",
      "Action": [
        "s3:PutObject"
      ],
      "Resource": "*"
    },
    {
      "Sid": "S3BucketIdentity",
      "Effect": "Allow",
      "Action": [
        "s3:GetBucketAcl",
        "s3:GetBucketLocation"
      ],
      "Resource": "*"
    }
```

```
  ]
}
```

> **Note**
>
> このポリシーに含まれているステートメントでは、多数の AWS リソースへのアクセスが許可される可能性があります。AWS CodeBuild に特定の AWS リソースへのアクセスを制限するには、`Resource` 配列の値を変更します。詳細については、AWS サービスのセキュリティドキュメントを参照してください。

3. 上記のファイルを保存したディレクトリに移動し、以下の 2 つのコマンドをこの順番で 1 つずつ実行します。`CodeBuildServiceRole` と `CodeBuildServiceRolePolicy` には異なる値を使用する場合は、ここでそれらを使用してください。

```
aws iam create-role --role-name CodeBuildServiceRole --assume-role-policy-document
 file://create-role.json
```

```
aws iam put-role-policy --role-name CodeBuildServiceRole --policy-name
 CodeBuildServiceRolePolicy --policy-document file://put-role-policy.json
```

CodeBuild でカスタマー管理のキーを作成して設定する

AWS CodeBuild がビルド出力アーティファクトを暗号化するには、KMS キーにアクセスする必要があります。デフォルトでは、CodeBuild は AWS アカウントの Amazon S3 用 AWS マネージドキーを使用します。

AWS マネージドキー を使用しない場合は、カスタマー管理のキーを自分で作成して設定する必要があります。このセクションでは、IAM コンソールを使用してこれを行う方法を説明します。

カスタマー管理のキーの詳細については、AWS KMS デベロッパーガイドの「AWS Key Management Service の概念および キーの作成」を参照してください。

CodeBuild で使用するカスタマー管理のキーを設定するには、AWS KMS 開発者ガイドの「キーポリシーの変更」の手順に従ってください。次に、キーポリシーに以下のステートメント (*###BEGIN ADDING STATEMENTS HERE###* と *###END ADDING STATEMENTS HERE###* の間) を追加します。省略記号 (`...`) は、簡潔にするために使用され、ステートメントを追加する場所の特定に役立ちます。ステートメントを削除しないでください、また、これらの省略記号をキーポリシーに入力しないでください。

```
{
  "Version": "2012-10-17",
  "Id": "...",
  "Statement": [
    ### BEGIN ADDING STATEMENTS HERE ###
    {
      "Sid": "Allow access through Amazon S3 for all principals in the account that are
 authorized to use Amazon S3",
      "Effect": "Allow",
      "Principal": {
        "AWS": "*"
      },
      "Action": [
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:ReEncrypt*",
        "kms:GenerateDataKey*",
        "kms:DescribeKey"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "kms:ViaService": "s3.region-ID.amazonaws.com",
          "kms:CallerAccount": "account-ID"
        }
      }
    },
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::account-ID:role/CodeBuild-service-role"
      },
      "Action": [
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:ReEncrypt*",
        "kms:GenerateDataKey*",
        "kms:DescribeKey"
      ],
      "Resource": "*"
    },
    ### END ADDING STATEMENTS HERE ###
    {
```

```
      "Sid": "Enable IAM User Permissions",
      ...
    },
    {
      "Sid": "Allow access for Key Administrators",
      ...
    },
    {
      "Sid": "Allow use of the key",
      ...
    },
    {
      "Sid": "Allow attachment of persistent resources",
      ...
    }
  ]
}
```

- *region-ID* は、CodeBuild に関連付けられた Amazon S3 バケットが配置されている AWS リージョンの ID を表します (たとえば、us-east-1)。
- ***##### ID***は、AWSカスタマー管理のキーを所有する アカウントのID を表します。
- *CodeBuild-service-role* は、このトピックの前半で作成または識別した CodeBuild サービスロールの名前を表します。

> **Note**
>
> IAM コンソールでカスタマー管理のキーを作成または設定するには、まず次のいずれかを使用して AWS Management Console にサインインする必要があります。
>
> - AWS ルートアカウント。これは推奨されません。詳細については、ユーザーガイドの「アカウントルートユーザー」を参照してください。
> - AWS アカウントの管理者ユーザー。詳細については、ユーザーガイドの「最初の AWS アカウントルートユーザーおよびグループの作成」を参照してください。
> - カスタマー管理のキーを作成または変更するアクセス許可を持つ AWS アカウントの IAM ユーザー。詳細については、AWS KMS 開発者ガイドの、「AWS KMS コンソールを使用するために必要なアクセス許可」を参照してください。

AWS CLI のインストールと設定

AWS CodeBuild にアクセスするには、AWS CLI または代わりに CodeBuild コンソール、CodePipeline コンソール、AWS SDKを使用できます。AWS CLI をインストールして設定するには、AWS Command Line Interface ユーザーガイドの「AWS Command Line Interface のセットアップ」を参照してください。

1. 次のコマンドを実行して、AWS CLI のインストールが CodeBuild をサポートしているかどうかを確認します。

   ```
   aws codebuild list-builds
   ```

   成功すると、次のような情報が出力に表示されます。

   ```
   {
     "ids": []
   }
   ```

   空の角括弧は、まだビルドを実行していないことを示しています。
2. エラーが出力された場合は、現在のバージョンの AWS CLI をアンインストールしてから、最新バージョンをインストールする必要があります。詳細については、AWS CLI ユーザーガイドの「AWS Command Line Interface のアンインストール」および「AWS Command Line Interface のインストール」を参照してください。

AWS CodeBuild のコマンドラインリファレンス

AWS CLI は、AWS CodeBuild を自動化するためのコマンドを提供します。このトピックの情報は、AWS Command Line Interface ユーザーガイドと AWS CodeBuild の AWS CLI リファレンスの補足として使用します。

お探しのものではありませんか。AWS SDK を使用して CodeBuild を呼び出す場合は、AWS SDK とツールのリファレンス を参照してください。

このトピックの情報を使用するには、AWS CLI のインストールと設定 の説明に従って AWS CLI をインストールし、CodeBuild 用に設定しておく必要があります。

AWS CLI を使用して CodeBuild のエンドポイントを指定するには、AWS CodeBuild エンドポイントの指定 (AWS CLI) を参照してください。

次のコマンドでは、CodeBuild のコマンドのリストを取得できます。

```
aws codebuild help
```

次のコマンドでは、CodeBuild コマンドに関する情報を取得できます。*command-name* はコマンド名です。

```
aws codebuild command-name help
```

CodeBuild のコマンドは以下の通りです。

- `batch-delete-builds`: CodeBuild の 1 つ以上のビルドを削除します。詳細については、「ビルドの削除 (AWS CLI)」を参照してください。
- `batch-get-builds`: CodeBuild の複数のビルドに関する情報を取得します。詳細については、「ビルドの詳細の表示 (AWS CLI)」を参照してください。
- `batch-get-projects`: 指定された 1 つ以上のビルドプロジェクトに関する情報を取得します。詳細については、「ビルドプロジェクトの詳細を表示する (AWS CLI)」を参照してください。
- `create-project`: ビルドプロジェクトを作成します。詳細については、「ビルドプロジェクトの作成 (AWS CLI)」を参照してください。
- `delete-project`: ビルドプロジェクトを削除します。詳細については、「ビルドプロジェクトの削除 (AWS CLI)」を参照してください。
- `list-builds`: CodeBuild でのビルドの Amazon リソースネーム (ARN) をリスト表示します。詳細については、「ビルド ID の一覧表示 (AWS CLI)」を参照してください。
- `list-builds-for-project`: 指定されたビルドプロジェクトに関連付けられているビルド ID のリストを取得します。詳細については、「ビルドプロジェクトのビルド ID を一覧表示する (AWS CLI)」を参照してください。
- `list-curated-environment-images`: ビルドに使用できる CodeBuild によって管理される Docker イメージのリストを取得します。詳細については、「が提供する Docker イメージ CodeBuild」を参照してください。
- `list-projects`: ビルドプロジェクト名のリストを取得します。詳細については、「ビルドプロジェクト名の一覧表示 (AWS CLI)」を参照してください。
- `start-build`: ビルドの実行を開始します。詳細については、「ビルドの実行 (AWS CLI)」を参照してください。
- `stop-build`: 停止されたビルドの実行を停止しようとします。詳細については、「ビルドの停止 (AWS CLI)」を参照してください。

- `update-project`: 指定されたビルドプロジェクトに関する情報を変更します。詳細については、「ビルドプロジェクトの設定の変更 (AWS CLI)」を参照してください。

AWS の SDK とツールのリファレンスAWS CodeBuild

いずれかの AWS SDK またはツールを使用して AWS CodeBuild を自動化するには、以下のリソースを参照してください。

AWS CLI を使用して CodeBuild を実行する場合は、コマンドラインリファレンス を参照してください。

AWS でサポートされる AWS CodeBuild SDK とツール

CodeBuild でサポートしている AWS SDK とツールは以下のとおりです。

- AWS SDK for C++。詳細については、http://sdk.amazonaws.com/cpp/api/LATEST/namespace_aws_1_1_code_build.html SDK for C++ API リファレンスの AWSAws::CodeBuild 名前空間のセクションを参照してください。
- AWS SDK for Go。詳細については、AWS SDK for Go API リファレンスの CodeBuild セクションを参照してください。
- AWS SDK for Java。詳細については、AWS SDK for Java API リファレンスの `com.amazonaws.services.codebuild` および `com.amazonaws.services.codebuild.model` セクションを参照してください。
- ブラウザでの AWS SDK for JavaScript および Node.js での AWS SDK for JavaScript。詳細については、AWS SDK for Go API リファレンス の Class: AWSCodeBuild のセクションを参照してください。
- AWS SDK for .NET。詳細については、AWS SDK for .NET API リファレンス の Amazon.codeBuild および Amazon.codeBuild.model名前空間セクションを参照してください。
- AWS SDK for PHP。詳細については、AWS SDK for PHP API リファレンスの名前空間 Aws\CodeBuild セクションを参照してください。
- AWS SDK for Python (Boto3)。詳細については、Boto 3 ドキュメント の「CodeBuild」セクションを参照してください。
- AWS SDK for Ruby。詳細については、AWS SDK for Ruby API リファレンスの「モジュール: Aws::CodeBuild 」セクションを参照してください。
- AWS Tools for PowerShell。詳細については、AWS CodeBuild Tools for PowerShell Cmdlet リファレンスの「AWS」セクションを参照してください。

AWS CodeBuild エンドポイントの指定

AWS Command Line Interface で使用するエンドポイントを指定するには、AWS CLI (AWS)、または AWS CodeBuild SDK を使用します。エンドポイントは、CodeBuild が使用可能なリージョンごとに存在します。リージョンのエンドポイントに加えて、4 つのリージョンに連邦情報処理標準 (FIPS) エンドポイントがあります。FIPS エンドポイントの詳細については、「FIPS 140-2 の概要」を参照してください。

エンドポイントの指定はオプションです。使用するエンドポイントを CodeBuild で明示的に指定しない場合、このサービスでは、AWS アカウントで使用しているリージョンに関連付けられているエンドポイントが使用されます。CodeBuild では、FIPS エンドポイントがデフォルトで使用されることはありません。FIPS エンドポイントを使用するには、次のいずれかのメソッドを使用して、CodeBuild と関連付ける必要があります。

> **Note**
>
> AWS SDK を使用してエンドポイントを指定するには、エイリアスまたはリージョン名を使用します。AWS CLI を使用する場合は、完全なエンドポイント名を使用する必要があります。

CodeBuild で使用可能なエンドポイントについては、「CodeBuild のリージョンとエンドポイント」を参照してください。

トピック

- AWS CodeBuild エンドポイントの指定 (AWS CLI)
- AWS CodeBuild エンドポイントの指定 (AWS SDK)

AWS CodeBuild エンドポイントの指定 (AWS CLI)

AWS CLI を使用して、AWS CodeBuild にアクセスするエンドポイントを指定するには、CodeBuild コマンドに `--endpoint-url` 引数を指定します。たとえば、「米国東部 (バージニア北部) リージョン」で連邦情報処理標準 (FIPS) エンドポイントを使用して、プロジェクトビルド名のリストを取得するには、このコマンドを実行します。

```
aws codebuild list-projects --endpoint-url https://codebuild-fips.us-
east-1.amazonaws.com
```

エンドポイントの先頭に `https://` を追加します。

`--endpoint-url` の AWS CLI 引数は、すべての AWS サービスに利用できます。この引数と AWS CLI の引数については、AWS CLI コマンドリファレンスを参照してください。

AWS CodeBuild エンドポイントの指定 (AWS SDK)

AWS にアクセスするエンドポイントを指定するには、AWS CodeBuild SDK を使用します。この例では、「AWS SDK for Java」が使用されていますが、他の AWS SDK を使用してエンドポイントを指定することもできます。

AWSCodeBuild クライアントを作成する場合は、`withEndpointConfiguration` メソッドを使用します。以下の形式を使用します。

```
AWSCodeBuild awsCodeBuild = AWSCodeBuildClientBuilder.standard().
    withEndpointConfiguration(new AwsClientBuilder.EndpointConfiguration("endpoint",
 "region")).
    withCredentials(new AWSStaticCredentialsProvider(sessionCredentials)).
    build();
```

`AWSCodeBuildClientBuilder` については、「AWSCodeBuildClientBuilder クラス」を参照してください。

`withCredentials` の認証情報のタイプは、`AWSCredentialsProvider` を使用する必要があります。詳細については、「AWS 認証情報の使用」を参照してください。

エンドポイントの先頭に `https://` を追加しないでください。

非 FIPS エンドポイントを指定する場合は、実際のエンドポイントではなくリージョンを使用します。例えば、米国東部 (バージニア北部) リージョンのエンドポイントを指定するには、完全なエンドポイント名 (`codebuild.us-east-1.amazonaws.com`) ではなく、`us-east-1` を使用できます。

FIPS エンドポイントを指定する場合は、エイリアスを使用して、コードを簡素化することができます。FIPS エンドポイントのみ、エイリアスが含まれます。他のエンドポイントは、リージョンまたは完全名を使用して指定する必要があります。

利用できる 4 つの FIPS エンドポイントごとのエイリアスを以下のテーブルに示します。

リージョン名	リージョン	エンドポイント	エイリアス
米国東部 (バージニア北部)	us-east-1	codebuild-fips.us-east-1.amazonaws.com	us-east-1-fips
米国東部 (オハイオ)	us-east-2	codebuild-fips.us-east-2.amazonaws.com	us-east-2-fips
米国西部 (北カリフォルニア)	us-west-1	codebuild-fips.us-west-1.amazonaws.com	us-west-1-fips
米国西部 (オレゴン)	us-west-2	codebuild-fips.us-west-2.amazonaws.com	us-west-2-fips

エイリアスを使用して、米国西部 (オレゴン) リージョンの FIPS エンドポイントを指定するには:

```
AWSCodeBuild awsCodeBuild = AWSCodeBuildClientBuilder.standard().
    withEndpointConfiguration(new AwsClientBuilder.EndpointConfiguration("us-west-2-
fips", "us-west-2")).
    withCredentials(new AWSStaticCredentialsProvider(sessionCredentials)).
    build();
```

米国東部 (バージニア北部) リージョンの非 FIPS エンドポイントを指定するには:

```
AWSCodeBuild awsCodeBuild = AWSCodeBuildClientBuilder.standard().
    withEndpointConfiguration(new AwsClientBuilder.EndpointConfiguration("us-east-1",
 "us-east-1")).
    withCredentials(new AWSStaticCredentialsProvider(sessionCredentials)).
    build();
```

アジアパシフィック (ムンバイ) リージョンの非 FIPS エンドポイントを指定するには:

```
AWSCodeBuild awsCodeBuild = AWSCodeBuildClientBuilder.standard().
    withEndpointConfiguration(new AwsClientBuilder.EndpointConfiguration("ap-south-1",
 "ap-south-1")).
    withCredentials(new AWSStaticCredentialsProvider(sessionCredentials)).
    build();
```

AWS CodePipeline で AWS CodeBuild を使用してコードをテストし、ビルドを実行する

リリースプロセスを自動化するには、AWS CodePipeline を使用してコードをテストし、AWS CodeBuild でビルドを実行します。

次の表に示しているのは、タスクとその実行に使用できるメソッドです。これらのタスクを AWS SDK で達成する方法については、このトピックの対象外です。

タスク	使用可能なアプローチ	このトピックで説明するアプローチ
でビルドを自動化 CodePipeline する で継続的デリバリー (CD) パイプラインを作成する CodeBuild	• CodePipeline コンソール • AWS CLI • AWS SDK	• CodePipeline コンソールの使用 • AWS CLI の使用 • このトピックの情報は、AWS SDK を使用するように調整できます。詳細については、AWS CodePipeline の API リファレンス の `CreatePipeline` またはアマゾン ウェブ サービスのツールの SDK セクションでプログラミング言語の `create-pipeline` アクションドキュメントを参照してください。
でテストおよびビルドの自動化 CodeBuild を の既存のパイプラインに追加する CodePipeline	• CodePipeline コンソール • AWS CLI • AWS SDK	• CodePipeline コンソールを使用してビルド自動化を追加する • CodePipeline コンソールを使用してテスト自動化を追加する • の場合AWS CLI、このトピックの情報を調整して、CodeBuild ビルドアクションまたはテストアクションを含むパイプラインを作成できます。詳細について

タスク	使用可能なアプローチ	このトピックで説明するアプローチ
		は、「 ユーザーガイド」の「パイプラインを編集する (AWS CLI)」およびCodePipeline 「パイプライン構造リファレンスAWS CodePipeline」を参照してください。 • このトピックの情報は、AWS SDK を使用するように調整できます。詳細については、AWS CodePipeline の API リファレンス の `UpdatePipeline` またはアマゾン ウェブ サービスのツールの SDK セクションからプログラミング言語の `update-pipeline` アクションドキュメントを参照してください。

前提条件

1. ビルドを計画する の質問に答えます。
2. AWS ルートアカウントまたは管理者ユーザー CodePipeline の代わりに ユーザーを使用して にアクセスする場合は、 という名前の 管理ポリシー`AWSCodePipelineFullAccess`をユーザー (またはユーザーが属する IAM グループ) にアタッチします。AWS ルートアカウントは使用しないでください。このポリシーは、 CodePipeline でパイプラインを作成するためのアクセス許可をユーザーに付与します。詳細については、ユーザーガイドの「管理ポリシーをアタッチする」を参照してください。

   > **Note**
   >
   > ポリシーをユーザー (またはユーザーが属する IAM グループ) にアタッチする IAM エンティティは、ポリシーをアタッチするために IAM でのアクセス許可を持っている必要があります。詳細については、ユーザーガイドの「IAM ユーザー、グループ、および認証情報を管理するためのアクセス許可の委任」を参照してください。

3. AWS アカウントに CodePipeline 利用可能なサービスロールがない場合は、サービスロールを作成します。 はこのサービスロール CodePipeline を使用して、ユーザーに代わって AWS CodeBuildを含む他の AWSのサービスとやり取りします。例えば、 を使用して CodePipeline サービスロールAWS CLIを作成するには、IAM `create-role` コマンドを実行します。

Linux、macOS、Unix の場合:

```
aws iam create-role --role-name AWS-CodePipeline-CodeBuild-Service-Role
 --assume-role-policy-document '{"Version":"2012-10-17","Statement":
{"Effect":"Allow","Principal":
{"Service":"codepipeline.amazonaws.com"},"Action":"sts:AssumeRole"}}'
```

Windows の場合:

```
aws iam create-role --role-name AWS-CodePipeline-CodeBuild-Service-Role --assume-
role-policy-document "{\"Version\":\"2012-10-17\",\"Statement\":{\"Effect\":
\"Allow\",\"Principal\":{\"Service\":\"codepipeline.amazonaws.com\"},\"Action\":
\"sts:AssumeRole\"}}"
```

> **Note**
>
> この CodePipeline サービスロールを作成する IAM エンティティには、サービスロールを作成するための IAM のアクセス許可が必要です。

4. CodePipeline サービスロールを作成した後、または既存の CodePipeline サービスロールポリシーを特定したら、AWS CodePipeline「 ユーザーガイド」の「デフォルトのサービスロールポリシーを確認する」で説明されているように、デフォルトの CodePipeline サービスロールポリシーをサービスロールに追加する必要があります。まだロールのポリシーの一部でない場合は、追加する必要があります。

> **Note**
>
> この CodePipeline サービスロールポリシーを追加する IAM エンティティには、サービスロールポリシーをサービスロールに追加するアクセス許可が IAM に必要です。

5. ソースコードを作成して、、 CodeCommitAmazon S3 CodePipeline、Bitbucket、 など、CodeBuild および でサポートされているリポジトリタイプにアップロードします GitHub。Amazon S3 ソースコードには buildspec ファイルが含まれている必要がありますが、このトピックの後半でビルドプロジェクトを定義するときにそのファイルを宣言できます。詳細については、「ビルド仕様 (buildspec) に関するリファレンス」を参照してください。

> **Important**
>
> パイプラインを使用してビルド済みのソースコードをデプロイする場合、ビルド出力アーティファクトには、使用するデプロイシステムとの互換性が必要です。
>
> - についてはAWS OpsWorks、「 ユーザーガイド」の「アプリケーションソース」および「 CodePipeline で を使用するAWS OpsWorksAWS OpsWorks」を参照してください。

トピック

CodeBuild を使用するパイプラインを作成する (CodePipeline コンソール)

CodeBuild を使用してソースコードをビルドおよびデプロイするパイプラインを作成するには、次の手順を実行します。

ソースコードのみをテストするパイプラインを作成するには、以下の操作を行います。

- 次の手順を使用してパイプラインを作成し、パイプラインからビルドステージとベータステージを削除します。次に、このトピックの「CodeBuild テストアクションをパイプラインに追加する (CodePipeline コンソール)」の手順を使用して、CodeBuild を使用するテストアクションをパイプラインに追加します。
- このトピックの他の手順のいずれかを使用してパイプラインを作成した後、このトピックの「CodeBuild テストアクションをパイプラインに追加する (CodePipeline コンソール)」の手順を使用して CodeBuild を使用するテストアクションをパイプラインに追加します。

CodePipeline でパイプライン作成ウィザードを使用して、CodeBuild を使用するパイプラインを作成するには

1. 以下を使用して AWS Management Console にサインインします。

- AWS ルートアカウント。これは推奨されません。詳細については、ユーザーガイドの「アカウントルートユーザー」を参照してください。
- AWS アカウントの管理者ユーザー。詳細については、ユーザーガイドの「最初の AWS アカウントルートユーザーおよびグループの作成」を参照してください。
- 次の最小限のアクションを使用するアクセス許可を持つ AWS アカウントのユーザー。

```
codepipeline:*
iam:ListRoles
iam:PassRole
s3:CreateBucket
s3:GetBucketPolicy
s3:GetObject
s3:ListAllMyBuckets
s3:ListBucket
s3:PutBucketPolicy
codecommit:ListBranches
codecommit:ListRepositories
codedeploy:GetApplication
codedeploy:GetDeploymentGroup
codedeploy:ListApplications
codedeploy:ListDeploymentGroups
elasticbeanstalk:DescribeApplications
elasticbeanstalk:DescribeEnvironments
lambda:GetFunctionConfiguration
lambda:ListFunctions
opsworks:DescribeStacks
opsworks:DescribeApps
opsworks:DescribeLayers
```

2. AWS CodePipeline コンソール (https://console.aws.amazon.com/codesuite/codepipeline/home) を開きます。
3. AWS リージョンセレクタで、ビルドプロジェクトの AWS リソースが配置されている AWS リージョンを選択します。この AWS リージョンでは、CodeBuild をサポートしている必要があります。詳細については、「Amazon Web Services 全般のリファレンス」の「AWS CodeBuild」を参照してください。
4. パイプラインを作成します。CodePipeline 情報ページが表示されたら、[Create pipeline] (パイプラインの作成) を選択します。[Pipelines (パイプライン)] ページが表示された場合は、[Create pipeline (パイプラインの作成)] を選択します。

5. [Step 1: Choose pipeline settings (ステップ 1: パイプラインの設定の選択)] ページで、[Pipeline name (パイプライン名)] にパイプラインの名前を入力します (例: **`CodeBuildDemoPipeline`**)。別の名前を選択した場合は、この手順全体でそれを使用してください。
6. [Role name (ロール名)] として、以下のいずれかの操作を行います。

   [New service role (新しいサービスロール)] を選択し、[Role Name (ロール名)] に、新しいサービスロールの名前を入力します。

   [Existing service role] (既存のサービスロール) を選択し、このトピックの前提条件の一部として作成または特定した CodePipeline サービスロールを選択します。
7. [Artifact store (アーティファクトストア)] で、次のいずれかの操作を行います。

   - [Default location (デフォルトの場所)] を選択し、デフォルトとして指定された S3 アーティファクトバケットなどのデフォルトのアーティファクトストアを、パイプラインに選択した AWS リージョン内のパイプラインに使用します。
   - S3 アーティファクトバケットなど、既に作成済みのアーティファクトストアがパイプラインと同じ AWS リージョンにある場合は、[Custom location (カスタムの場所)] を選択します。

   > **Note**
   >
   > これはパイプラインのソースコードのソースバケットではありません。パイプラインのアーティファクトストアです。パイプラインごとに別個のアーティファクトストア (S3 バケットなど) が、パイプラインと同じ AWS リージョンに必要です。
8. [Next] (次へ) をクリックします。
9. [Step 2: Add source stage (ステップ 2: ソースステージの追加)] ページの [ソースプロバイダ] で、次のいずれかの操作を行います。

   - ソースコードの保存先が S3 バケットである場合は、[Amazon S3] を選択します。[バケット] で、ソースコードが含まれている S3 バケットを選択します。[S3 オブジェクトキー] に、ソースコードを含むファイルの名前 (例: *`file-name`*`.zip`) を入力します。[Next] (次へ) をクリックします。
   - ソースコードが AWS CodeCommit リポジトリに保存されている場合は、[CodeCommit] を選択します。[Repository name] で、ソースコードが含まれているリポジトリの名前を選択します。[ブランチ名] で、ビルドするソースコードのバージョンが含まれているブランチの名前を選択します。[Next] (次へ) をクリックします。

- ソースコードが GitHub リポジトリに保存されている場合は、[GitHub] を選択します。[Connect to GitHub] を選択し、手順に従って GitHub に対して認証します。[Repository] で、ソースコードが含まれているリポジトリの名前を選択します。[ブランチ] で、ビルドするソースコードのバージョンが含まれているブランチの名前を選択します。

[Next] (次へ) をクリックします。

10. [Step 3: Add build stage] (ステップ 3: ビルドステージを追加する) ページで、[Build provider] (ビルドプロバイダー) として [CodeBuild] を選択します。
11. 既存のビルドプロジェクトを使用する場合は、[Project name] (プロジェクト名) で、ビルドプロジェクトの名前を選択し、この手順の次のステップにスキップします。

    新しい CodeBuild ビルドプロジェクトを作成する必要がある場合は、「ビルドプロジェクトの作成 (コンソール)」の手順に従ってから、この手順に戻ります。

    既存のビルドプロジェクトを選択した場合、ビルド出力アーティファクトの設定がすでに定義されている必要があります (ただし、CodePipeline によってビルド出力の設定が上書きされます)。詳細については、「ビルドプロジェクトの設定の変更 (コンソール)」を参照してください。

    > **Important**
    >
    > CodeBuild プロジェクトのウェブフックを有効にして、プロジェクトを CodePipeline のビルドステップとして使用すると、コミットごとに 2 つの等しいビルドが作成されます。1 つのビルドはウェブフックを通じてトリガーされ、別の 1 つは CodePipeline を通じてトリガーされます。請求はビルド単位で発生するため、両方のビルドに対して課金されます。したがって、CodePipeline を使用する場合は、CodeBuild でウェブフックを無効にすることをお勧めします。AWS CodeBuild コンソールで、[Webhook] ボックスをオフにします。詳細については、「ビルドプロジェクトの設定の変更 (コンソール)」を参照してください。

12. [Step 4: Add deploy stage (ステップ 4: デプロイステージの追加)] ページで、次のいずれかの操作を行います。

    - ビルド出力アーティファクトをデプロイしない場合は、[Skip (スキップ)] を選択し、プロンプトが表示されたら、これを選択したことを確認します。
    - ビルド出力アーティファクトをデプロイする場合は、[Deployment provider (デプロイプロバイダ)] でデプロイプロバイダを選択し、次にプロンプトに応じて設定を指定します。

[Next] (次へ) をクリックします。

13. [確認] ページで、選択内容を確認し、[パイプラインの作成] を選択します。
14. パイプラインが正常に実行されたら、ビルド出力アーティファクトを取得できます。CodePipeline コンソールにパイプラインを表示した状態で、[Build] (ビルド) アクションでツールヒントを選択します。[Output artifact] の値をメモします (例: MyAppBuild)。

> **Note**
>
> ビルド出力アーティファクトを取得するには、CodeBuild コンソールのビルドの詳細ページで [Build artifacts] (ビルドアーティファクト) リンクを選択することもできます。このページを表示するには、この手順の残りのステップを省略して、「ビルドの詳細の表示 (コンソール)」を参照してください。

15. Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
16. バケットのリストで、パイプラインで使用されるバケットを開きます。バケット名は、`codepipeline-`*`region-ID`*`-`*`random-number`* の形式に従う必要があります。AWS CLI を使用して、CodePipelineの get-pipeline コマンドを実行すると、バケットの名前を取得できます。*`my-pipeline-name`* は、パイプラインの表示名です。

```
aws codepipeline get-pipeline --name my-pipeline-name
```

出力では、`pipeline` オブジェクトには `artifactStore` オブジェクトが含まれ、それには、バケットの名前と `location` の値が含まれます。

17. パイプラインの名前と一致するフォルダを開きます (パイプライン名の長さによってはフォルダ名が切り詰められている場合があります)。次に、前に書き留めた [出力アーティファクト] の値と一致するフォルダを開きます。
18. ファイルの内容を展開します。そのフォルダに複数のファイルがある場合は、[Last Modified] タイムスタンプが最新であるファイルの内容を抽出します。(システムの ZIP ユーティリティで操作できるように、必要に応じて、ファイルに `.zip` 拡張子を付けます。) ビルド出力アーティファクトは、展開されたファイルの内容に含まれます。
19. CodePipeline にビルド出力アーティファクトをデプロイするよう指示した場合は、デプロイプロバイダの説明を活用してデプロイターゲットのビルド出力アーティファクトを取得します。

CodeBuild を使用するパイプラインの作成 (AWS CLI)

CodeBuild を使用してソースコードをビルドするパイプラインを作成するには、次の手順を実行します。

AWS CLI を使用して、ビルドされたソースコードをデプロイする、または、ソースコードのテストのみを行うパイプラインを作成するには、AWS CodePipeline ユーザーガイドのパイプラインの編集 (AWS CLI) および CodePipeline のパイプライン構造リファレンスを参照してください。

1. CodeBuild でビルドプロジェクトを作成または識別します。詳細については、「ビルドプロジェクトの作成」を参照してください。

   > **Important**
   >
   > ビルドプロジェクトは、ビルド出力アーティファクトの設定を定義する必要があります (ただし、CodePipeline によって上書きされます)。詳細については、「`artifacts`」で ビルドプロジェクトの作成 (AWS CLI) の説明を参照してください。

2. このトピックで説明している IAM エンティティの 1 つに対応する AWS アクセスキーと AWS シークレットアクセスキーで AWS CLI を設定していることを確認してください。詳細については、AWS Command Line Interface ユーザーガイドの AWS Command Line Interface のセットアップを参照してください。
3. パイプラインの構造を表す JSON 形式のファイルを作成します。ファイルに `create-pipeline.json` のような名前を付けます。たとえば、この JSON 形式の構造では、S3 入力バケットを参照するソースアクションと CodeBuild を使用するビルドアクションを使用してパイプラインを作成します。

```
{
  "pipeline": {
    "roleArn": "arn:aws:iam::<account-id>:role/<AWS-CodePipeline-service-role-name>",
    "stages": [
      {
        "name": "Source",
        "actions": [
          {
            "inputArtifacts": [],
            "name": "Source",
            "actionTypeId": {
              "category": "Source",
```

```
            "owner": "AWS",
            "version": "1",
            "provider": "S3"
          },
          "outputArtifacts": [
            {
              "name": "MyApp"
            }
          ],
          "configuration": {
            "S3Bucket": "<bucket-name>",
            "S3ObjectKey": "<source-code-file-name.zip>"
          },
          "runOrder": 1
        }
      ]
    },
    {
      "name": "Build",
      "actions": [
        {
          "inputArtifacts": [
            {
              "name": "MyApp"
            }
          ],
          "name": "Build",
          "actionTypeId": {
            "category": "Build",
            "owner": "AWS",
            "version": "1",
            "provider": "CodeBuild"
          },
          "outputArtifacts": [
            {
              "name": "default"
            }
          ],
          "configuration": {
            "ProjectName": "<build-project-name>"
          },
          "runOrder": 1
        }
      ]
```

```
      }
    ],
    "artifactStore": {
      "type": "S3",
      "location": "<CodePipeline-internal-bucket-name>"
    },
    "name": "<my-pipeline-name>",
    "version": 1
  }
}
```

この JSON 形式のデータは以下のようになっています。

- roleArn の値は、前提条件の一部として作成または特定した CodePipeline のサービスロールの ARN と一致する必要があります。
- S3Bucket の S3ObjectKey と configuration の値は、ソースコードの保存先が S3 バケットであることを前提としています。その他のソースコードのリポジトリタイプの設定については、AWS CodePipeline ユーザーガイドの CodePipeline のパイプライン構造リファレンスを参照してください。
- ProjectName の値は、この手順の前半で作成した CodeBuild ビルドプロジェクトの名前です。
- location の値は、このパイプラインで使用する S3 バケットの名前です。詳細については、AWS CodePipeline ユーザーガイドの CodePipelineのアーティファクトストアとして使用する S3 バケットのポリシーを作成するを参照してください。
- name の値は、このパイプラインの名前です。すべてのパイプラインの名前はアカウントに対して一意である必要があります。

このデータにはソースアクションとビルドアクションのみが記述されていますが、テスト、ビルド出力アーティファクトのデプロイ、AWS Lambda 関数の呼び出しなどに関連したアクティビティのためにアクションを追加できます。詳細については、AWS CodePipeline ユーザーガイドの AWS CodePipeline のパイプライン構造リファレンスを参照してください。

4. JSON ファイルが保存されているフォルダに切り替え、 CodePipeline の「create-pipeline」コマンドを実行し、ファイル名を指定します。

```
aws codepipeline create-pipeline --cli-input-json file://create-pipeline.json
```

> **Note**
>
> パイプラインは、CodeBuild がサポートされている AWS リージョンで作成する必要があります。詳細については、「Amazon Web Services 全般のリファレンス」の「AWS CodeBuild」を参照してください。

JSON 形式のデータが出力に表示され、CodePipeline がパイプラインを作成します。

5. パイプラインのステータスに関する情報を取得するには、パイプラインの名前を指定して CodePipeline の get-pipeline-state コマンドを実行します。

```
aws codepipeline get-pipeline-state --name <my-pipeline-name>
```

出力で、ビルドが成功したことを確認する情報を探します。省略記号 (...) は、簡潔にするために省略されたデータを表すために使用されます。

```
{
  ...
  "stageStates": [
    ...
    {
      "actionStates": [
        {
          "actionName": "CodeBuild",
          "latestExecution": {
            "status": "SUCCEEDED",
            ...
          },
          ...
        }
      ]
    }
  ]
}
```

このコマンドをあまりに早く実行すると、ビルドアクションに関する情報が表示されないことがあります。パイプラインがビルドアクションの実行を終了するまで、このコマンドを複数回実行する必要があります。

6. ビルドが成功したら、次の手順に従ってビルド出力アーティファクトを取得します。Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。

   > **Note**
   >
   > ビルド出力アーティファクトを取得するには、CodeBuild コンソールの関連するビルドの詳細ページで [ビルドアーティファクト] リンクを選択することもできます。このページを表示するには、この手順の残りのステップを省略して、「ビルドの詳細の表示 (コンソール)」を参照してください。

7. バケットのリストで、パイプラインで使用されるバケットを開きます。バケット名は、codepipeline-*<region-ID>*-*<random-number>* の形式に従う必要があります。バケット名は、create-pipeline.json ファイルから取得するか、CodePipelineの get-pipeline コマンドを実行して取得できます。

   ```
   aws codepipeline get-pipeline --name <pipeline-name>
   ```

   出力では、pipeline オブジェクトには artifactStore オブジェクトが含まれ、それには、バケットの名前と location の値が含まれます。

8. パイプラインの名前と一致するフォルダを開きます (例:*<pipeline-name>*)。
9. そのフォルダで、default という名前のフォルダを開きます。
10. ファイルの内容を展開します。そのフォルダに複数のファイルがある場合は、[Last Modified] タイムスタンプが最新であるファイルの内容を抽出します。(システムの ZIP ユーティリティで操作できるように、必要に応じて、ファイルに .zip 拡張子を付けます。) ビルド出力アーティファクトは、展開されたファイルの内容に含まれます。

CodeBuild ビルドアクションをパイプラインに追加する (CodePipeline コンソール)

1. 以下を使用して AWS Management Console にサインインします。

   - AWS ルートアカウント。これは推奨されません。詳細については、ユーザーガイドの「アカウントルートユーザー」を参照してください。
   - AWS アカウントの管理者ユーザー。詳細については、ユーザーガイドの「最初の AWS アカウントルートユーザーおよびグループの作成」を参照してください。
   - 以下の最小限のアクションを実行するアクセス許可を持つ AWS アカウントのユーザー。

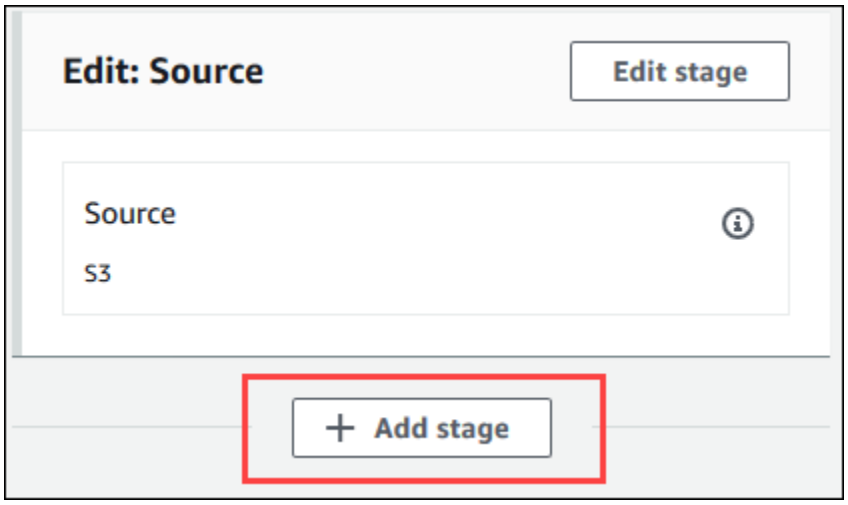
```
codepipeline:*
iam:ListRoles
iam:PassRole
s3:CreateBucket
s3:GetBucketPolicy
s3:GetObject
s3:ListAllMyBuckets
s3:ListBucket
s3:PutBucketPolicy
codecommit:ListBranches
codecommit:ListRepositories
codedeploy:GetApplication
codedeploy:GetDeploymentGroup
codedeploy:ListApplications
codedeploy:ListDeploymentGroups
elasticbeanstalk:DescribeApplications
elasticbeanstalk:DescribeEnvironments
lambda:GetFunctionConfiguration
lambda:ListFunctions
opsworks:DescribeStacks
opsworks:DescribeApps
opsworks:DescribeLayers
```

2. CodePipeline コンソール (http://console.aws.amazon.com/codesuite/codepipeline/home) を開きます。
3. AWS リージョンセレクタで、パイプラインが配置されている AWS リージョンを選択します。このリージョンでは、CodeBuild をサポートしている必要があります。詳細については、「Amazon Web Services 全般のリファレンス」の「CodeBuild」を参照してください。
4. [Pipelines (パイプライン)] ページで、パイプラインの名前を選択します。
5. パイプラインの詳細ページの [ソース] アクションで、ツールヒントを選択します。[Output artifact (出力アーティファクト)] の値 (例: MyApp) をメモします。

> **Note**
>
> この手順では、[Source] ステージと [Beta] ステージの間のビルドステージ内にビルドアクションを追加する方法について説明します。ビルドアクションを別の場所に追加する場合は、ビルドアクションを追加する場所の直前のアクションのツールヒントを選択し、[Output artifact (出力アーティファクト)] の値をメモします。

6. Edit (編集) を選択します。
7. [ソース] と [ベータ] ステージの間で、[Add (追加)] を選択します。

> **Note**
>
> この手順では、パイプラインの [Source] ステージと [Beta] ステージの間にビルドアクションを追加する方法について説明します。既存のステージにビルドアクションを追加するには、ステージで [Edit stage (ステージを編集)] を選択し、この手順のステップ 8 に進みます。ビルドステージを別の場所に追加するには、目的の場所で [Add stage (ステージの追加)] を選択します。



8. [Stage name (ステージ名)] に、ビルドステージの名前 (例: **Build**) を入力します。別の名前を選択する場合は、この手順全体でそれを使用します。
9. 選択したステージの中で、[アクションの追加] を選択します。

> **Note**
>
> この手順では、ビルドステージの中にビルドアクションを追加する方法について説明します。ビルドアクションを別の場所に追加するには、目的の場所で [Add action (アクションの追加)] を選択します。まず、ビルドアクションを追加する既存のステージで [Edit stage (ステージを編集)] アイコンを選択する必要があります。

10. [アクションの編集] の [アクション名] に、アクションの名前 (例: **CodeBuild**) を入力します。別の名前を選択する場合は、この手順全体でそれを使用します。

11. [Action provider] (アクションプロバイダー) で、[CodeBuild] を選択します。
12. 既存のビルドプロジェクトを使用する場合は、[Project name] (プロジェクト名) で、ビルドプロジェクトの名前を選択し、この手順の次のステップにスキップします。

    新しい CodeBuild ビルドプロジェクトを作成する必要がある場合は、「ビルドプロジェクトの作成 (コンソール)」の手順に従ってから、この手順に戻ります。

    既存のビルドプロジェクトを選択した場合、ビルド出力アーティファクトの設定がすでに定義されている必要があります (ただし、CodePipeline によってビルド出力の設定が上書きされます)。詳細については、「ビルドプロジェクトの作成 (コンソール)」および「ビルドプロジェクトの設定の変更 (コンソール)」にある「アーティファクト」の説明を参照してください。

    > **Important**
    >
    > CodeBuild プロジェクトのウェブフックを有効にして、プロジェクトを CodePipeline のビルドステップとして使用すると、コミットごとに 2 つの等しいビルドが作成されます。1 つのビルドはウェブフックを通じてトリガーされ、別の 1 つは CodePipeline を通じてトリガーされます。請求はビルド単位で発生するため、両方のビルドに対して課金されます。したがって、CodePipeline を使用する場合は、CodeBuild でウェブフックを無効にすることをお勧めします。CodeBuild コンソールで、[Webhook] ボックスをオフにします。詳細については、「ビルドプロジェクトの設定の変更 (コンソール)」を参照してください。

13. [入力アーティファクト] で、この手順で前に書き留めた出力アーティファクトを選択します。
14. [出力アーティファクト] で、出力アーティファクトの名前を入力します (例: **`MyAppBuild`**)。
15. [Add action] を選択します。
16. [Save (保存)]、[Save (保存)] の順に選択し、パイプラインの変更を保存します。
17. [Release change] を選択します。
18. パイプラインが正常に実行されたら、ビルド出力アーティファクトを取得できます。CodePipeline コンソールにパイプラインを表示した状態で、[Build] (ビルド) アクションでツールヒントを選択します。[Output artifact] の値をメモします (例: MyAppBuild)。

    > **Note**
    >
    > ビルド出力アーティファクトを取得するには、CodeBuild コンソールのビルドの詳細ページで [Build artifacts] (ビルドアーティファクト) リンクを選択することもできます。

このページの内容を表示するには、「ビルドの詳細の表示 (コンソール)」を参照して、この手順のステップ 31 に進みます。

19. Amazon S3 コンソール (https://console.aws.amazon.com/s3/) を開きます。
20. バケットのリストで、パイプラインで使用されるバケットを開きます。バケット名は、codepipeline-*region-ID*-*random-number* の形式に従う必要があります。バケットの名前を取得するには、AWS CLI を使用して CodePipeline get-pipeline コマンドを実行します。

```
aws codepipeline get-pipeline --name my-pipeline-name
```

出力では、pipeline オブジェクトには artifactStore オブジェクトが含まれ、それには、バケットの名前と location の値が含まれます。

21. パイプラインの名前と一致するフォルダを開きます (パイプライン名の長さによってはフォルダ名が切り詰められている場合があります)。次に、この手順で前に書き留めた [出力アーティファクト] の値と一致するフォルダを開きます。
22. ファイルの内容を展開します。そのフォルダに複数のファイルがある場合は、[Last Modified] タイムスタンプが最新であるファイルの内容を抽出します。(システムの ZIP ユーティリティで操作できるように、必要に応じて、ファイルに .zip 拡張子を付けます。) ビルド出力アーティファクトは、展開されたファイルの内容に含まれます。
23. CodePipeline にビルド出力アーティファクトをデプロイするよう指示した場合は、デプロイプロバイダの説明を活用してデプロイターゲットのビルド出力アーティファクトを取得します。

CodeBuild テストアクションをパイプラインに追加する (CodePipeline コンソール)

1. 以下を使用して AWS Management Console にサインインします。
   - AWS ルートアカウント。これは推奨されません。詳細については、ユーザーガイドの「アカウントルートユーザー」を参照してください。
   - AWS アカウントの管理者ユーザー。詳細については、ユーザーガイドの「最初の AWS アカウントルートユーザーおよびグループの作成」を参照してください。
   - 以下の最小限のアクションを実行するアクセス許可を持つ AWS アカウントのユーザー。

```
codepipeline:*
iam:ListRoles
```

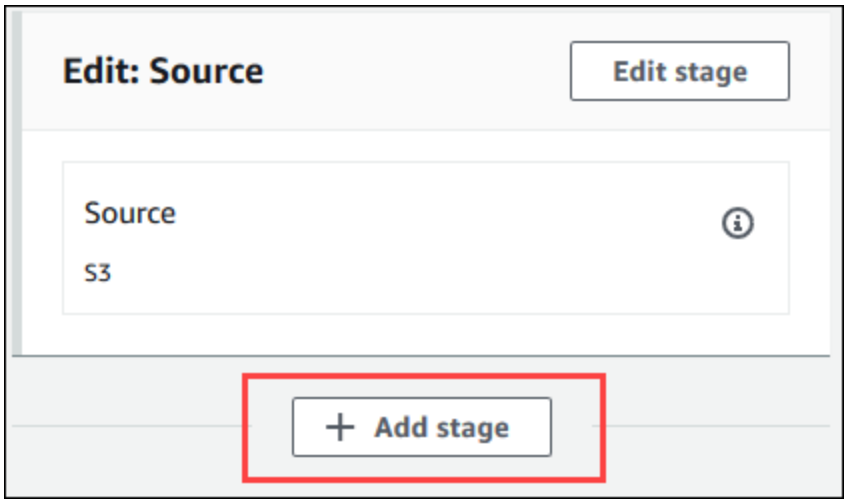
```
iam:PassRole
s3:CreateBucket
s3:GetBucketPolicy
s3:GetObject
s3:ListAllMyBuckets
s3:ListBucket
s3:PutBucketPolicy
codecommit:ListBranches
codecommit:ListRepositories
codedeploy:GetApplication
codedeploy:GetDeploymentGroup
codedeploy:ListApplications
codedeploy:ListDeploymentGroups
elasticbeanstalk:DescribeApplications
elasticbeanstalk:DescribeEnvironments
lambda:GetFunctionConfiguration
lambda:ListFunctions
opsworks:DescribeStacks
opsworks:DescribeApps
opsworks:DescribeLayers
```

2. CodePipeline コンソール (http://console.aws.amazon.com/codesuite/codepipeline/home) を開きます。
3. AWS リージョンセレクタで、パイプラインが配置されている AWS リージョンを選択します。この AWS リージョンでは、CodeBuild をサポートしている必要があります。詳細については、「Amazon Web Services 全般のリファレンス」の「AWS CodeBuild」を参照してください。
4. [Pipelines (パイプライン)] ページで、パイプラインの名前を選択します。
5. パイプラインの詳細ページの [ソース] アクションで、ツールヒントを選択します。[Output artifact (出力アーティファクト)] の値 (例: MyApp) をメモします。

> **Note**
>
> この手順では、[Source] ステージと [Beta] ステージの間のテストステージ内にテストアクションを追加する方法について説明します。テストアクションを別の場所に追加する場合は、直前のアクションにマウスポインタを合わせ、[Output artifact] の値をメモします。

6. Edit (編集) を選択します。
7. [ソース] ステージのすぐ後で、[Add stage (ステージの追加)] を選択します。

> **Note**
>
> この手順では、パイプラインの [Source] ステージの直後にテストステージを追加する方法を示します。既存のステージにテストアクションを追加するには、ステージで [Edit stage (ステージを編集)] を選択し、この手順のステップ 8 に進みます。テストステージを別の場所に追加するには、目的の場所で [Add stage (ステージの追加)] を選択します。



8. [ステージ名] に、テストステージの名前 (例: **Test**) を入力します。別の名前を選択する場合は、この手順全体でそれを使用します。
9. 選択されたステージで、[アクションの追加] を選択します。

> **Note**
>
> この手順は、テストステージにテストアクションを追加する方法を示します。テストアクションを別の場所に追加するには、目的の場所で [Add action (アクションの追加)] を選択します。まず、テストアクションを追加する既存のステージで [Edit (編集)] アイコンを選択する必要があります。

10. [アクションの編集] の [アクション名] に、アクションの名前 (例: **Test**) を入力します。別の名前を選択する場合は、この手順全体でそれを使用します。
11. [Action provider] (アクションプロバイダー) の [Test] (テスト) で、[CodeBuild] を選択します。

12. 既存のビルドプロジェクトを使用する場合は、[Project name] (プロジェクト名) で、ビルドプロジェクトの名前を選択し、この手順の次のステップにスキップします。

    新しい CodeBuild ビルドプロジェクトを作成する必要がある場合は、「ビルドプロジェクトの作成 (コンソール)」の手順に従ってから、この手順に戻ります。

    > **Important**
    >
    > CodeBuild プロジェクトのウェブフックを有効にして、プロジェクトを CodePipeline のビルドステップとして使用すると、コミットごとに 2 つの等しいビルドが作成されます。1 つのビルドはウェブフックを通じてトリガーされ、別の 1 つは CodePipeline を通じてトリガーされます。請求はビルド単位で発生するため、両方のビルドに対して課金されます。したがって、CodePipeline を使用する場合は、CodeBuild でウェブフックを無効にすることをお勧めします。CodeBuild コンソールで、[Webhook] ボックスをオフにします。詳細については、「ビルドプロジェクトの設定の変更 (コンソール)」を参照してください。

13. [入力アーティファクト] で、この手順で前に書き留めた [出力アーティファクト] の値を選択します。
14. (オプション) テストアクションで出力アーティファクトを生成し、それに応じてビルド仕様を設定する場合は、出力アーティファクトに割り当てる値を [出力アーティファクト] に入力します。
15. [Save (保存)] を選択します。
16. [Release change] を選択します。
17. パイプラインが正常に実行された後、テスト結果を取得できます。パイプラインの [Test] (テスト) ステージで [CodeBuild] ハイパーリンクを選択し、CodeBuild コンソールで関連するビルドプロジェクトのページを開きます。
18. ビルドプロジェクトページの [Build history] エリアで、[Build run] ハイパーリンクを選択します。
19. ビルドの実行ページの [Build logs] (ビルドログ) エリアで、[View entire log] (ログ全体の表示) ハイパーリンクを選択し、Amazon CloudWatch コンソールでビルドログを開きます。
20. ビルドログをスクロールして、テスト結果を表示します。

Jenkins で AWS CodeBuild を使用する

AWS CodeBuild の Jenkins プラグインを使用して、CodeBuild と Jenkins のビルドジョブを統合できます。Jenkins ビルドノードにビルドジョブを送信する代わりに、プラグインを使用してビルドジョブを CodeBuild に送信します。これにより、Jenkins ビルドノードのプロビジョニング、設定、および管理が不要になります。

Jenkins のセットアップ

AWS CodeBuild プラグインを使用した Jenkins のセットアップ方法や、プラグインのソースコードのダウンロードについては、https://github.com/awslabs/aws-codebuild-jenkins-plugin をご覧ください。

プラグインのインストール

Jenkins サーバーをセットアップ済みで、AWS CodeBuild プラグインのみをインストールする場合は、Jenkins インスタンスの Plugin Manager で **`CodeBuild Plugin for Jenkins`** を検索します。

プラグインの使用

VPC の外部のソースで AWS CodeBuild を使用するには

1. CodeBuild コンソールでプロジェクトを作成します。詳細については、「ビルドプロジェクトの作成 (コンソール)」を参照してください。
   - ビルドを実行する AWS リージョンを選択します。
   - (オプション) CodeBuild ビルドコンテナによる VPC のリソースへのアクセスを許可するように Amazon VPC 設定を指定します。
   - プロジェクトの名前を書き留めます。これはステップ 3 で必要になります。
   - (オプション) ソースリポジトリが CodeBuild でネイティブにサポートされていない場合は、プロジェクトの入力ソースタイプとして Amazon S3 を設定できます。
2. IAM コンソールで、Jenkins プラグインで使用するユーザーを作成します。
   - ユーザーの認証情報を作成するときに、[Programmatic Access (プログラムによるアクセス)] を選択します。
   - 次のようなポリシーを作成し、このポリシーをユーザーにアタッチします。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Resource": ["arn:aws:logs:{{region}}:{{awsAccountId}}:log-group:/aws/
codebuild/{{projectName}}:*"],
      "Action": ["logs:GetLogEvents"]
    },
    {
      "Effect": "Allow",
      "Resource": ["arn:aws:s3:::{{inputBucket}}"],
      "Action": ["s3:GetBucketVersioning"]
    },
    {
      "Effect": "Allow",
      "Resource": ["arn:aws:s3:::{{inputBucket}}/{{inputObject}}"],
      "Action": ["s3:PutObject"]
    },
    {
      "Effect": "Allow",
      "Resource": ["arn:aws:s3:::{{outputBucket}}/*"],
      "Action": ["s3:GetObject"]
    },
    {
      "Effect": "Allow",
      "Resource": ["arn:aws:codebuild:{{region}}:{{awsAccountId}}:project/
{{projectName}}"],
      "Action": ["codebuild:StartBuild",
        "codebuild:BatchGetBuilds",
        "codebuild:BatchGetProjects"]
    }
  ]
}
```

3. Jenkins で自由形式のプロジェクトを作成します。

   - [Configure] (設定) ページで、[Add build step] (ビルドステップの追加)、[Run build on CodeBuild] (CodeBuild でビルドを実行) を選択します。
   - ビルドステップを設定します。

- [Region (リージョン)]、[Credentials (認証情報)]、および [Project Name (プロジェクト名)] の値を入力します。
- [Use Project source (プロジェクトソースを使用)] を選択します。
- 設定を保存し、Jenkins からビルドを実行します。

4. [Source Code Management (ソースコードの管理)] で、ソースの取得方法を選択します。Jenkins サーバーでの GitHub プラグイン (またはソースリポジトリプロバイダ用の Jenkins プラグイン) のインストールが必要になる場合があります。

   - [設定] ページで、[ビルドステップの追加] を選択し、[AWS CodeBuild でビルドを実行] を選択します。
   - ビルドステップを設定します。
     - [Region (リージョン)]、[Credentials (認証情報)]、および [Project Name (プロジェクト名)] の値を入力します。
     - [Use Jenkins source (Jenkins ソースを使用)] を選択します。
     - 設定を保存し、Jenkins からビルドを実行します。

Jenkins パイプラインプラグインで AWS CodeBuild プラグインを使用するには

- Jenkins パイプラインプロジェクトページで、スニペットジェネレーターを使用して、パイプラインにステップとして CodeBuild を追加するパイプラインスクリプトを生成します。次のようなスクリプトが生成されます。

```
awsCodeBuild projectName: 'project', credentialsType: 'keys', region: 'us-west-2',
 sourceControlType: 'jenkins'
```

Codecov による AWS CodeBuild の使用

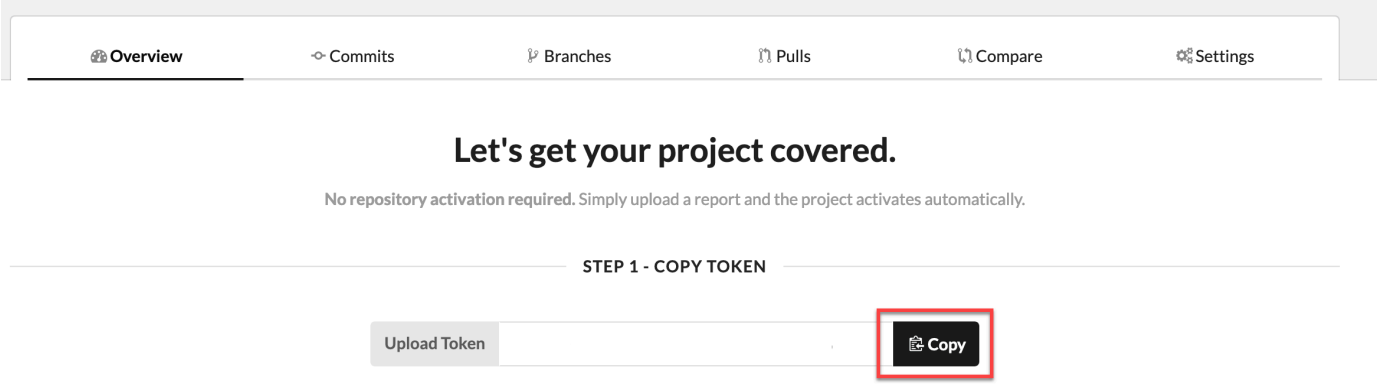
Codecov は、コードのテストカバレッジを測定するツールです。Codecov は、コード内のどのメソッドとステートメントがテストされていないかを識別します。その結果に基づいて、コードの品質を向上させるためのテストの記述先を判断します。Codecov は、 CodeBuild でサポートされているソースリポジトリのうち、3 つ (GitHub、GitHub Enterprise Server、Bitbucket) でサポートされています。ビルドプロジェクトで GitHub Enterprise Server を使用する場合は、Codecov Enterprise を使用する必要があります。

Codecov を統合した CodeBuild プロジェクトのビルドを実行すると、リポジトリ内のコードを分析する Codecov レポートが Codecov にアップロードされます。ビルドログには、レポートへのリンクが含まれています。次のサンプルでは、Python および Java ビルドプロジェクトを Codecov と統合する方法を示します。Codecov でサポートされている言語のリストについては、Codecov Supported Languages を参照してください。

Codecov とビルドプロジェクトの統合

Codecov とビルドプロジェクトを統合するには

1. https://codecov.io/signup に移動し、GitHub または Bitbucket ソースリポジトリにサインアップします。GitHub Enterprise を使用する場合は、Codecov ウェブサイトの「Codecov Enterprise」を参照してください。
2. Codecov に、カバレッジ対象のリポジトリを追加します。
3. トークン情報が表示されたら、[Copy] を選択します。



4. コピーしたトークンを、CODECOV_TOKEN という名前の環境変数としてビルドプロジェクトに追加します。詳細については、「ビルドプロジェクトの設定の変更 (コンソール)」を参照してください。
5. リポジトリ内に my_script.sh という名前のテキストファイルを作成します。このファイルに次の内容を入力します。

```
#/bin/bash
bash <(curl -s https://codecov.io/bash) -t $CODECOV_TOKEN
```

6. ビルドプロジェクトの用途に応じて [Python] タブまたは [Java] タブを選択し、次の手順に従います。

Java

1. 次の JaCoCo プラグインをリポジトリ内の pom.xml に追加します。

```
<build>
  <plugins>
    <plugin>
      <groupId>org.jacoco</groupId>
      <artifactId>jacoco-maven-plugin</artifactId>
      <version>0.8.2</version>
      <executions>
          <execution>
              <goals>
                  <goal>prepare-agent</goal>
              </goals>
          </execution>
          <execution>
              <id>report</id>
              <phase>test</phase>
              <goals>
                  <goal>report</goal>
              </goals>
          </execution>
      </executions>
    </plugin>
  </plugins>
</build>
```

2. buildspec ファイルに次のコマンドを入力します。詳細については、「」を参照してくださいbuildspec の構文

```
build:
  - mvn test -f pom.xml -fn
postbuild:
  - echo 'Connect to CodeCov'
  - bash my_script.sh
```

Python

buildspec ファイルに次のコマンドを入力します。詳細については、「」を参照してくださいbuildspec の構文

```
build:
  - pip install coverage
  - coverage run -m unittest discover
postbuild:
  - echo 'Connect to CodeCov'
  - bash my_script.sh
```

7. ビルドプロジェクトのビルドを実行します。プロジェクト用に生成された Codecov レポートへのリンクがビルドログに表示されます。リンクを使用して Codecov レポートを表示します。詳細については、「AWS CodeBuild でのビルドの実行」および「AWS CodeBuild による AWS CloudTrail API 呼び出しのログ記録」を参照してください。ビルドログの Codecov 情報は、次のように表示されます。

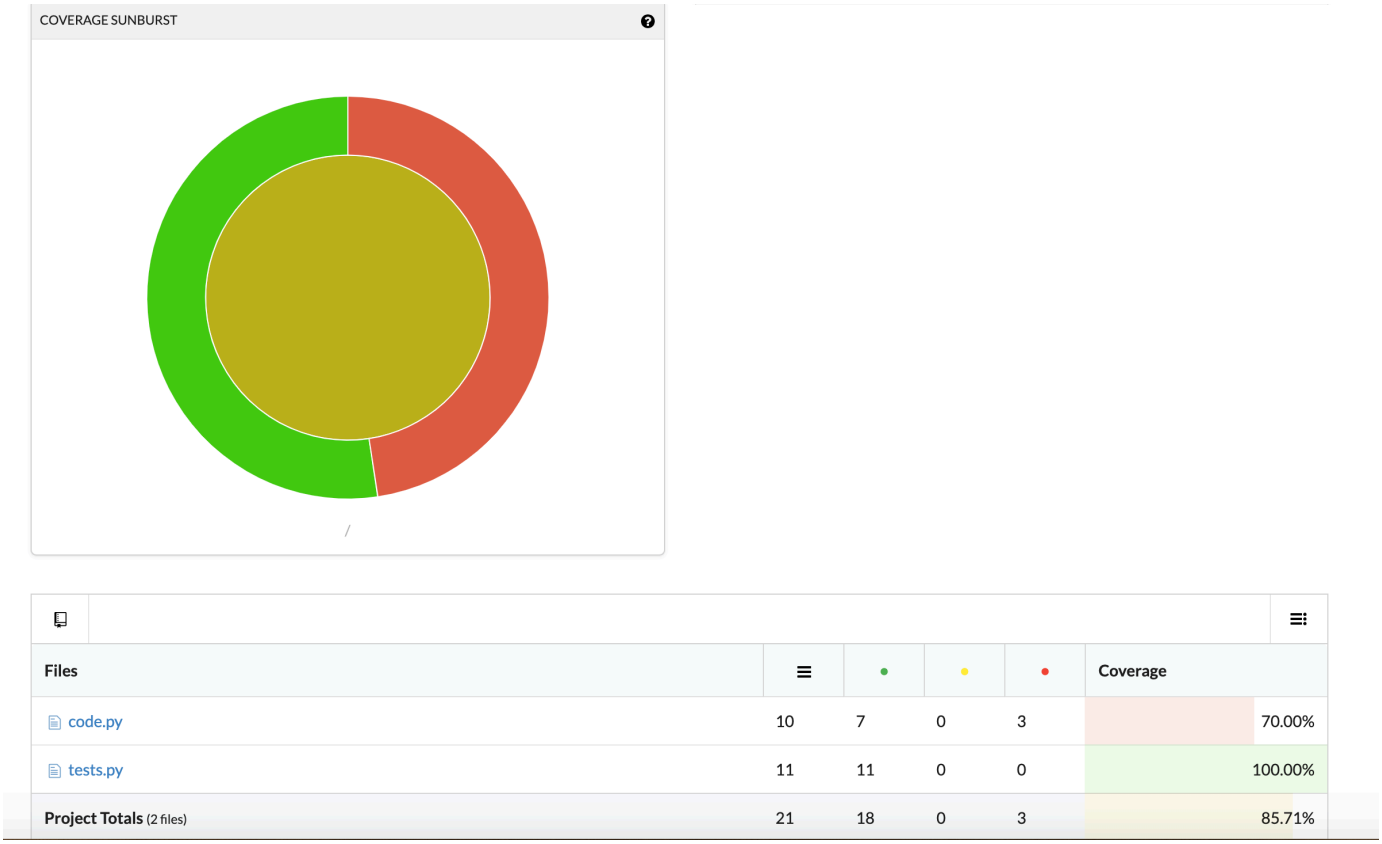
```
[Container] 2020/03/09 16:31:04 Running command bash my_script.sh


  _____          _
 / ____|        | |
| |     ___   __| | ___  ___ _____   __
| |    / _ \ / _` |/ _ \/ __/ _ \ \ / /
| |___| (_) | (_| |  __/ (_| (_) \ V /
 \_____\___/ \__,_|\___|\___\___/ \_/
                              Bash-20200303-bc4d7e6

·[0;90m==>·[0m AWS CodeBuild detected.
... The full list of Codecov log entries has been omitted for brevity ...
    .
    ·[0;32m->·[0m View reports at ·[0;36mhttps://codecov.io/github/user/test_py/
commit/commit-id·[0m

[Container] 2020/03/09 16:31:07 Phase complete: POST_BUILD State: SUCCEEDED
```

レポートは次のように表示されます。



サーバーレスアプリケーションでの AWS CodeBuild の使用

AWS Serverless Application Model (AWS SAM) は、サーバーレスアプリケーションを構築するためのオープンソースのフレームワークです。詳細については、GitHub の AWS serverless application model リポジトリを参照してください。

AWS CodeBuild を使用して、AWS SAM 標準に準拠するサーバーレスアプリケーションをパッケージ化してデプロイすることができます。デプロイのステップで、CodeBuild は AWS CloudFormation を使用できます。CodeBuild と AWS CloudFormation を使用したサーバーレスアプリケーションの構築とデプロイを自動化するには、AWS CodePipeline を使用できます。

詳細については、AWS Serverless Application Model 開発者ガイドの「サーバーレスアプリケーションをデプロイする」を参照してください。

関連リソース

- AWS CodeBuild の開始方法については、「コンソールを使用した AWS CodeBuild の開始方法」を参照してください。

- CodeBuild の問題のトラブルシューティングについては、「トラブルシューティング AWS CodeBuild」を参照してください。
- CodeBuild のクォータについては、「AWS CodeBuild のクォータ」を参照してください。

トラブルシューティング AWS CodeBuild

このトピックの情報を使用して、問題を特定、診断、対処します。 CodeBuild ビルドのログ記録とモニタリングを行って問題のトラブルシューティングを行う方法については、「」を参照してくださいログ記録とモニタリング。

トピック

- Apache Maven が間違ったリポジトリのアーティファクトを参照している
- ビルドコマンドがデフォルトでルートとして実行される
- ファイル名に英語以外の文字が含まれているとビルドが失敗する場合があります
- Amazon EC2 パラメータストアからパラメータを取得する際にビルドが失敗する場合がある
- CodeBuild コンソールでブランチフィルタにアクセスできない
- ビルドの成功または失敗を表示できない
- ビルドステータスがソースプロバイダに報告されない
- Windows Server Core 2019 プラットフォームの基本イメージが見つからず、選択できない
- buildspec ファイルの以前のコマンドが、以降のコマンドで認識されない
- エラー: キャッシュのダウンロード時に「アクセスが拒否される」
- エラー: 「BUILD_CONTAINER_UNABLE_TO_PULL_IMAGE」カスタムビルドイメージを使用した場合
- エラー : 「ビルドを完了する前にビルドコンテナがデッドが見つかりました。ビルドコンテナはメモリ不足か、Docker イメージはサポートされていません。 ErrorCode: 500」
- Error: "Cannot connect to the Docker daemon" when running a build (ビルドの実行時に「Docker デーモンに接続できません」)
- エラー : CodeBuild 「ビルドプロジェクトを作成または更新するときに sts: を実行する権限がありませんAssumeRole」
- エラー : 「 の呼び出しエラー GetBucketAcl: バケット所有者が変更されているか、サービスロールに s3: を呼び出すアクセス許可がありませんGetBucketAcl」
- エラー: ビルドの実行時に「アーティファクトのアップロードに失敗しました: 無効な ARN」
- エラー: 「Git のクローンに失敗しました: 'your-repository-URL' にアクセスできません: SSL 証明書の問題: 自己署名証明書」
- エラー: ビルドの実行時に「アクセスしようとしているバケットは、指定されたエンドポイントを使用してアドレス指定する必要があります」

- エラー: "このビルドイメージではランタイムバージョンを少なくとも 1 つ選択する必要があります。"
- ビルドキューのビルドが失敗するとエラー「QUEUED:INSUFFICIENT_SUBNET」が表示される
- エラー：「キャッシュをダウンロードできません: RequestError: x509: システムルートをロードできませんでしたが、ルートが指定されていません」が原因でリクエストの送信に失敗しました
- エラー：「S3 から証明書をダウンロードできません AccessDenied」
- エラー:「認証情報を見つけることができません」
- RequestError プロキシサーバー CodeBuild での実行時のタイムアウトエラー
- ビルドイメージ内に Bourne シェル (sh) が必要
- 警告 (ビルドの実行時):「ランタイムのインストールをスキップします。このビルドイメージではランタイムバージョンの選択はサポートされていません」
- エラー: CodeBuild コンソールを開くときに JobWorker 「ID を確認できません」
- ビルドの開始の失敗
- ローカルにキャッシュされたビルドの GitHub メタデータへのアクセス
- AccessDenied: レポートグループのバケット所有者が S3 バケットの所有者と一致しません...

Apache Maven が間違ったリポジトリのアーティファクトを参照している

問題： AWS CodeBuildが提供する Java ビルド環境で Maven を使用する場合、Maven は https://repo1.maven.org/maven2 の安全な中央 Maven リポジトリからビルドとプラグインの依存関係を取得します。これは、ビルドプロジェクトの `pom.xml` ファイルが別の場所を使用すると明示的に宣言した場合でも発生します。

考えられる原因： CodeBuild が提供する Java ビルド環境には、ビルド環境の `/root/.m2` ディレクトリに`settings.xml`プリインストールされている という名前のファイルが含まれます。この `settings.xml` には次の宣言が含まれています。これにより、Maven が常に https://repo1.maven.org/maven2 で、セキュアな中央 Maven リポジトリからビルドおよびプラグインの依存関係を引き出すように指示されます。

```
<settings>
  <activeProfiles>
    <activeProfile>securecentral</activeProfile>
```

```
  </activeProfiles>
  <profiles>
    <profile>
      <id>securecentral</id>
      <repositories>
        <repository>
          <id>central</id>
          <url>https://repo1.maven.org/maven2</url>
          <releases>
            <enabled>true</enabled>
          </releases>
        </repository>
      </repositories>
      <pluginRepositories>
        <pluginRepository>
          <id>central</id>
          <url>https://repo1.maven.org/maven2</url>
          <releases>
            <enabled>true</enabled>
          </releases>
        </pluginRepository>
      </pluginRepositories>
    </profile>
  </profiles>
</settings>
```

推奨される解決策: 以下を実行してください。

1. settings.xml ファイルをソースコードに追加します。
2. この settings.xml ファイルでは、前述の settings.xml 形式をガイドとして使用して、Maven が代わりにビルドとプラグインの依存関係を取得するリポジトリを宣言します。
3. ビルドプロジェクトの installフェーズで、 settings.xml に ファイルを構築環境の /root/.m2 ディレクトリにコピー CodeBuild するように指示します。たとえば、この動作を示す buildspec.yml ファイルの次のスニペットを考えてみましょう。

```
version 0.2

phases:
  install:
    commands:
      - cp ./settings.xml /root/.m2/settings.xml
```

ビルドコマンドがデフォルトでルートとして実行される

Issue: AWS CodeBuild ビルドコマンドをルートユーザーとして実行します。これは、関連するビルドイメージの Dockerfile によって USER インストラクションが別のユーザーに設定された場合でも発生します。

原因： デフォルトでは、 はすべてのビルドコマンドをルートユーザーとして CodeBuild 実行します。

推奨される解決策: なし。

ファイル名に英語以外の文字が含まれているとビルドが失敗する場合があります

問題: 英語以外の文字 (漢字など) を含むファイル名のファイルを使用するビルドを実行すると、ビルドが失敗します。

考えられる原因： によって提供されるビルド環境 AWS CodeBuild では、デフォルトのロケールがに設定されていますPOSIX。 POSIXローカリゼーション設定は、米国以外の CodeBuild やファイル名との互換性が低くなります。 英語の文字 および により、関連するビルドが失敗する可能性があります。

推奨される解決策: buildspec ファイルの pre_build セクションに次のコマンドを追加します。これらのコマンドにより、ビルド環境はローカリゼーション設定に米国英語 UTF-8 を使用します。これは、米国以外の CodeBuild やファイル名と互換性があります。 英語の文字。

Ubuntu ベースのビルド環境の場合:

```
pre_build:
  commands:
    - export LC_ALL="en_US.UTF-8"
    - locale-gen en_US en_US.UTF-8
    - dpkg-reconfigure locales
```

Amazon Linux ベースのビルド環境の場合:

```
pre_build:
  commands:
    - export LC_ALL="en_US.utf8"
```

Amazon EC2 パラメータストアからパラメータを取得する際にビルドが失敗する場合がある

問題: ビルドが Amazon EC2 パラメータストアに保存されている 1 つ以上のパラメータの値を取得しようとすると、DOWNLOAD_SOURCE フェーズで「Parameter does not exist」というエラーが発生し、ビルドが失敗する。

考えられる原因： ビルドプロジェクトが依存するサービスロールに ssm:GetParametersアクションを呼び出すアクセス許可がないか、ビルドプロジェクトが によって生成 AWS CodeBuild され、ssm:GetParametersアクションを呼び出すことができるサービスロールを使用していますが、パラメータには で始まらない名前が付けられています/CodeBuild/。

推奨される解決策:

- サービスロールが によって生成されていない場合は CodeBuild、 が ssm:GetParametersアクション CodeBuild を呼び出せるように定義を更新します。たとえば、次のポリシーステートメントでは、ssm:GetParameters アクションを呼び出して /CodeBuild/ で始まる名前を持つパラメータを取得できます。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": "ssm:GetParameters",
      "Effect": "Allow",
      "Resource": "arn:aws:ssm:REGION_ID:ACCOUNT_ID:parameter/CodeBuild/*"
    }
  ]
}
```

- サービスロールが によって生成された場合は CodeBuild、 が で始まる名前以外の名前で Amazon EC2 パラメータストアのパラメータ CodeBuild にアクセスできるように定義を更新します/CodeBuild/。たとえば、次のポリシーステートメントでは、ssm:GetParameters アクションを呼び出して指定された名前のパラメータを取得できます。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": "ssm:GetParameters",
```

```
      "Effect": "Allow",
      "Resource": "arn:aws:ssm:REGION_ID:ACCOUNT_ID:parameter/PARAMETER_NAME"
    }
  ]
}
```

CodeBuild コンソールでブランチフィルタにアクセスできない

問題： AWS CodeBuild プロジェクトを作成または更新するときに、ブランチフィルターオプションがコンソールで使用できない。

考えられる原因: ブランチフィルタオプションは廃止されました。このオプションはウェブフックフィルタグループに置き換えられています。これにより、 CodeBuild の新しいビルドをトリガーするウェブフックイベントの制御が強化されます。

推奨される解決策: ウェブフックフィルタの導入前に作成したブランチフィルタを移行するには、正規表現 HEAD_REF で ^refs/heads/*branchName*$ フィルタを使用してウェブフックフィルタグループを作成します。たとえば、ブランチフィルタの正規表現が ^branchName$ の場合、HEAD_REF フィルタに入力する更新後の正規表現は ^refs/heads/branchName$ です。詳細については、「Bitbucket ウェブフックイベント」および「 GitHub ウェブフックイベントのフィルタリング (コンソール) 」を参照してください。

ビルドの成功または失敗を表示できない

問題: 再試行されたビルドの成功または失敗を確認できない。

考えられる原因: ビルドのステータスを報告するオプションが有効になっていません。

推奨される解決策： CodeBuild プロジェクトを作成または更新するときに、レポートビルドステータスを有効にします。このオプションは、ビルドをトリガーするときに CodeBuild にステータスを報告するように指示します。詳細については、AWS CodeBuild API リファレンスの reportBuildStatus を参照してください。

ビルドステータスがソースプロバイダに報告されない

問題： GitHub または Bitbucket などのソースプロバイダーへのビルドステータスレポートを許可した後、ビルドステータスは更新されません。

考えられる原因: ソースプロバイダーに関連付けられたユーザーに、リポジトリへの書き込みアクセス許可がありません。

推奨される解決策: ソースプロバイダーにビルドステータスを報告できるようにするには、ソースプロバイダーに関連付けられたユーザーがリポジトリへの書き込みアクセス権を持っている必要があります。ユーザーが書き込みアクセス権を持っていない場合、ビルドのステータスは更新できません。詳細については、「ソースプロバイダーのアクセス」を参照してください。

Windows Server Core 2019 プラットフォームの基本イメージが見つからず、選択できない

問題: Windows Server Core 2019 プラットフォームの基本イメージを検索または選択できない。

考えられる原因： このイメージをサポートしていない AWS リージョンを使用しています。

推奨される解決策: Windows Server Core 2019 プラットフォームの基本イメージがサポートされている、次のいずれかの AWS リージョンを使用します。

- 米国東部 (バージニア北部)
- 米国東部 (オハイオ)
- 米国西部 (オレゴン)
- 欧州 (アイルランド)

buildspec ファイルの以前のコマンドが、以降のコマンドで認識されない

問題: buildspec ファイルの 1 つ以上のコマンドの結果が、同じ buildspec ファイルの以降のコマンドで認識されない。たとえば、コマンドによってローカル環境変数が設定される場合がありますが、後で実行されるコマンドがそのローカル環境変数の値を取得できない可能性があります。

考えられる原因: buildspec ファイルバージョン 0.1 では、 AWS CodeBuild はビルド環境のデフォルトシェルの各インスタンスで各コマンドを実行します。つまり、各コマンドは他のすべてのコマンドとは独立して実行されます。デフォルトでは、以前のコマンドの状態に依存する単一のコマンドを実行することはできません。

推奨される解決策: buildspec バージョン 0.2 を使用してください。これにより、問題が解決されます。何らかの理由で buildspec バージョン 0.1 を使用する必要がある場合は、シェルコマンド連鎖演

算子 (Linux の && など) を使用して、複数のコマンドを 1 つのコマンドにまとめることをお勧めします。または、複数のコマンドを含むソースコードにシェルスクリプトを組み込み、そのシェルスクリプトを buildspec ファイルの 1 つのコマンドから呼び出します。詳細については、「ビルド環境のシェルとコマンド」および「ビルド環境の環境変数」を参照してください。

エラー: キャッシュのダウンロード時に「アクセスが拒否される」

問題: キャッシュが有効になっているビルドプロジェクトでキャッシュをダウンロードしようとすると、`Access denied` エラーが発生する。

考えられる原因:

- ビルドプロジェクトの一部としてキャッシングが設定されています。
- キャッシュは、`InvalidateProjectCache` API により最近無効化されています。
- によって使用されているサービスロールには、キャッシュを保持している S3 バケットに対する `s3:GetObject`および アクセス`s3:PutObject`許可 CodeBuild がありません。

推奨される解決策: キャッシュ設定を更新した直後に初めて使用する場合、このエラーが表示されるのは普通です。このエラーが解消されない場合は、キャッシュを保持する S3 バケットへの `s3:GetObject` アクセス許可と `s3:PutObject` アクセス許可が、サービスロールに付与されているかどうかを確認する必要があります。詳細については、「Amazon S3 デベロッパーガイド」の「S3 アクセス許可の指定」を参照してください。

エラー: 「BUILD_CONTAINER_UNABLE_TO_PULL_IMAGE」カスタムビルドイメージを使用した場合

問題: カスタムビルドイメージを使用するビルドを実行しようとすると、ビルドは `BUILD_CONTAINER_UNABLE_TO_PULL_IMAGE` というエラーで失敗する。

考えられる原因: ビルドイメージの全体的な非圧縮サイズが、ビルド環境のコンピューティングタイプの使用可能ディスクスペースよりも大きい。ビルドイメージのサイズを確認するには、Docker を使用して *`docker images REPOSITORY:TAG`* コマンドを実行します。コンピューティングタイプで使用可能なディスク容量のリストについては、「ビルド環境のコンピューティングモードおよびタイプ」を参照してください。

> 推奨される解決策: 使用可能なディスク容量の大きなコンピューティングタイプを使用するか、カスタムビルドイメージのサイズを縮小します。

考えられる原因： AWS CodeBuild Amazon Elastic Container Registry (Amazon ECR) からビルドイメージをプルするアクセス許可がない。

推奨される解決策： がカスタムビルドイメージをビルド環境に CodeBuild プルできるように、Amazon ECR のリポジトリのアクセス許可を更新します。詳細については、「Amazon ECR のサンプル」を参照してください。

考えられる原因： リクエストした Amazon ECR イメージは、 AWS アカウントが使用している AWS リージョンでは使用できません。

推奨される解決策： AWS アカウントが使用しているイメージと同じ AWS リージョンにある Amazon ECR イメージを使用します。

考えられる原因： パブリックインターネットアクセスがない VPC でプライベートレジストリを使用しています。VPC のプライベート IP アドレスからイメージをプル CodeBuild することはできません。詳細については、「 の AWS Secrets Manager サンプルを使用したプライベートレジストリ CodeBuild」を参照してください。

推奨される解決策: VPC でプライベートレジストリを使用する場合は、VPC にパブリックインターネットアクセスがあることを確認してください。

考えられる原因: エラーメッセージに「toomanyrequests」が含まれており、イメージを Docker Hub から取得した場合、このエラーは Docker Hub のプル制限に達したことを意味します。

推奨される解決策: Docker Hub のプライベートレジストリを使用するか、Amazon ECR からイメージを取得します。プライベートレジストリの使用の詳細については、「 の AWS Secrets Manager サンプルを使用したプライベートレジストリ CodeBuild」を参照してください。Amazon ECR の使用方法の詳細については、「の Amazon ECR サンプル CodeBuild 」を参照してください。

エラー：「ビルドを完了する前にビルドコンテナがデッドが見つかりました。ビルドコンテナはメモリ不足か、Docker イメージはサポートされていません。 ErrorCode: 500」

問題： で Microsoft Windows または Linux コンテナを使用しようとすると AWS CodeBuild、このエラーは PROVISIONING フェーズ中に発生します。

考えられる原因:

- コンテナ OS バージョンは、 ではサポートされていません CodeBuild。

- HTTP_PROXY、HTTPS_PROXY、またはその両方がコンテナで指定されます。

推奨される解決策:

- Microsoft Windows の場合は、Windows コンテナを、コンテナ OS バージョン microsoft/windowsservercore:10.0.x (microsoft/windowsservercore:10.0.14393.2125 など) で使用します。
- Linux の場合は、Docker イメージの HTTP_PROXY 設定と HTTPS_PROXY 設定をクリアするか、ビルドプロジェクトで VPC 設定を指定します。

Error: "Cannot connect to the Docker daemon" when running a build (ビルドの実行時に「Docker デーモンに接続できません」)

問題: ビルドが失敗し、「`Cannot connect to the Docker daemon at unix:/var/run/docker.sock. Is the docker daemon running?`」のようなエラーがビルドログに表示される。

考えられる原因: 特権モードでビルドを実行していません。

推奨される解決策: このエラーを修正するには、特権モードを有効にし、次の手順を使用して buildspec を更新する必要があります。

ビルドを特権モードで実行するには、次の手順に従います。

1. https://console.aws.amazon.com/codebuild/ で CodeBuild コンソールを開きます。
2. ナビゲーションペインで、ビルドプロジェクト を選択し、ビルドプロジェクトを選択します。
3. [Edit] (編集) から [Environment] (環境) を選択します。
4. [Additional configuration (追加設定)] を選択します。
5. 特権 から、Docker イメージを構築する場合、またはビルドで昇格された権限を取得したい場合は、このフラグを有効にする を選択します。
6. [Update environment (環境の更新)] を選択します。
7. [ビルドの開始] を選択してビルドを再度実行します。

また、コンテナ内で Docker デーモンを起動する必要があります。buildspec の `install`フェーズは次のようになります。

```
phases:
```

```
  install:
    commands:
      - nohup /usr/local/bin/dockerd --host=unix:///var/run/docker.sock --
host=tcp://127.0.0.1:2375 --storage-driver=overlay2 &
      - timeout 15 sh -c "until docker info; do echo .; sleep 1; done"
```

buildspec ファイルで参照される OverlayFS ストレージドライバーの詳細については、Docker ウェブサイトの「Use the OverlayFS storage driver」を参照してください。

> **Note**
>
> 基本オペレーティングシステムが Alpine Linux である場合は、`buildspec.yml` で `-t` 引数を `timeout` に追加します。
>
> ```
> - timeout -t 15 sh -c "until docker info; do echo .; sleep 1; done"
> ```

を使用して Docker イメージを構築および実行する方法の詳細については AWS CodeBuild、「」を参照してくださいのカスタムイメージサンプルの Docker CodeBuild 。

エラー：CodeBuild 「ビルドプロジェクトを作成または更新するときに sts: を実行する権限がありませんAssumeRole」

問題: ビルドプロジェクトを作成または更新しようとすると、エラー「`Code:InvalidInputException, Message:CodeBuild is not authorized to perform: sts:AssumeRole on arn:aws:iam::account-ID:role/service-role-name`」が発生する。

考えられる原因:

- AWS Security Token Service （ AWS STS) は、ビルドプロジェクトを作成または更新しようとしている AWS リージョンで非アクティブ化されています。
- ビルドプロジェクトに関連付けられた AWS CodeBuild サービスロールが存在しないか、 を信頼するための十分なアクセス許可がありません CodeBuild。

推奨される解決策:

- ビルドプロジェクトを作成または更新しようとしている AWS リージョンで AWS STS が有効になっていることを確認します。詳細については、IAM ユーザーガイドの「 AWS リージョン AWS STS での のアクティブ化と非アクティブ化」を参照してください。
- ターゲット CodeBuild サービスロールが AWS アカウントに存在することを確認します。コンソールを使用していない場合は、ビルドプロジェクトを作成または更新したときにサービスロールの Amazon リソースネーム (ARN) のスペルを間違えていないことを確認してください。
- ターゲット CodeBuild サービスロールに、 を信頼するための十分なアクセス許可があることを確認します CodeBuild。詳細については、CodeBuild サービスロールの作成の「信頼関係のポリシーステートメント」を参照してください。

エラー：「 の呼び出しエラー GetBucketAcl: バケット所有者が変更されているか、サービスロールに s3: を呼び出すアクセス許可がありませんGetBucketAcl」

問題: ビルドを実行すると、S3 バケット所有者や `GetBucketAcl` アクセス許可の変更に関するエラーが発生する。

考えられる原因: `s3:GetBucketAcl` および `s3:GetBucketLocation` のアクセス許可を IAM ロールに追加しています。これらのアクセス許可は、プロジェクトの S3 バケットを保護し、バケットにアクセスできるユーザーを自分に限定します。これらのアクセス許可を追加した後で、S3 バケットの所有者が変更されています。

推奨される解決策: S3 バケットの所有者が自分であることを確認し、IAM ロールへのアクセス許可を追加し直します。詳細については、「S3 バケットへの安全なアクセス」を参照してください。

エラー: ビルドの実行時に「アーティファクトのアップロードに失敗しました: 無効な ARN」

問題: ビルドを実行すると、`UPLOAD_ARTIFACTS` ビルドフェーズが失敗し、「`Failed to upload artifacts: Invalid arn`」というエラーが表示される。

考えられる原因： S3 出力バケット (がビルドからの出力 AWS CodeBuild を保存するバケット) が、CodeBuild ビルドプロジェクトとは異なる AWS リージョンにある。

推奨される解決策： ビルドプロジェクトと同じ AWS リージョンにある出力バケットを指すようにビルドプロジェクトの設定を更新します。

エラー: 「Git のクローンに失敗しました: 'your-repository-URL' にアクセスできません: SSL 証明書の問題: 自己署名証明書」

問題: ビルドプロジェクトを実行しようとすると、このエラーが発生してビルドが失敗する。

考えられる原因: ソースリポジトリには自己署名証明書がありますが、S3 バケットから証明書をビルドプロジェクトの一部としてインストールする選択をしていません。

推奨される解決策:

- プロジェクトを編集します。[証明書] で [S3 から証明書をインストールする] を選択します。[証明書のバケット] では、SSL 証明書が保存されている S3 バケットを選択します。[証明書のオブジェクトキー] に、S3 オブジェクトキーの名前を入力します。
- プロジェクトを編集します。Enterprise Server プロジェクトリポジトリへの接続中に SSL 警告を無視するには、安全でない SSL GitHub を選択します。

> **Note**
>
> [Insecure SSL] はテストのみに使用することが推奨されます。本番環境では使用しないでください。

エラー: ビルドの実行時に「アクセスしようとしているバケットは、指定されたエンドポイントを使用してアドレス指定する必要があります」

問題: ビルドを実行すると、DOWNLOAD_SOURCE ビルドフェーズが失敗し、「`The bucket you are attempting to access must be addressed using the specified endpoint. Please send all future requests to this endpoint`」というエラーが表示される。

考えられる原因： 構築済みのソースコードが S3 バケットに保存され、そのバケットがビルドプロジェクトとは異なる AWS CodeBuild AWS リージョンにある。

推奨される解決策: 構築済みのソースコードが含まれているバケットを指すように、ビルドプロジェクトの設定を更新します。バケットがビルドプロジェクトと同じ AWS リージョンにあることを確認します。

エラー: "このビルドイメージではランタイムバージョンを少なくとも 1 つ選択する必要があります。"

問題: ビルドを実行すると、DOWNLOAD_SOURCE ビルドフェーズが失敗し、「YAML_FILE_ERROR: This build image requires selecting at least one runtime version」というエラーが表示される。

考えられる原因: ビルドでバージョン 1.0 以降の Amazon Linux 2 (AL2) 標準イメージ、またはバージョン 2.0 以降の Ubuntu 標準イメージが使用されていますが、buildspec ファイルでランタイムが指定されていません。

推奨される解決策： aws/codebuild/standard:2.0 CodeBuild マネージドイメージを使用する場合は、buildspec ファイルの runtime-versionsセクションでランタイムバージョンを指定する必要があります。たとえば、PHP を使用するプロジェクトでは、次の buildspec ファイルを使用します。

```
version: 0.2

phases:
  install:
    runtime-versions:
        php: 7.3
  build:
    commands:
      - php --version
artifacts:
  files:
    -  README.md
```

> **Note**
>
> runtime-versions セクションを指定して、Ubuntu 標準イメージ 2.0 以降や Amazon Linux 2 (AL2) 標準イメージ 1.0 以降以外のイメージを使用した場合は、ビルドで「Skipping install of runtimes. Runtime version selection is not supported by this build image」の警告が表示されます。

詳細については、「Specify runtime versions in the buildspec file」を参照してください。

ビルドキューのビルドが失敗するとエラー「QUEUED:INSUFFICIENT_SUBNET」が表示される

問題: ビルドキューのビルドが QUEUED: INSUFFICIENT_SUBNET のようなエラーで失敗する。

考えられる原因: VPC に指定された IPv4 CIDR ブロックが、リザーブド IP アドレスを使用している。各サブネット CIDR ブロックの最初の 4 つの IP アドレスと最後の IP アドレスは使用できず、インスタンスに割り当てることができません。たとえば、CIDR ブロック 10.0.0.0/24 を持つサブネットの場合、次の 5 つの IP アドレスが予約されます。

- 10.0.0.0: ネットワークアドレスです。
- 10.0.0.1: VPC ルーター AWS 用に によって予約されています。
- 10.0.0.2: によって予約されています AWS。DNS サーバーの IP アドレスは、常に VPC ネットワークのベースに 2 を付加したものですが、各サブネット範囲のベースに 2 を付加したアドレスも予約されています。複数の CIDR ブロックを持つ VPC の場合、DNS サーバーの IP アドレスはプライマリ CIDR にあります。詳細については、Amazon VPC ユーザーガイドの「Amazon DNS サーバー」を参照してください。
- 10.0.0.3: 将来の使用 AWS のために によって予約されています。
- 10.0.0.255: ネットワークブロードキャストアドレスです。VPC でのブロードキャストはサポートされていません。このアドレスは予約されています。

推奨される解決策: VPC でリザーブド IP アドレスが使用されていることを確認します。予約済みの IP アドレスを、予約されていないアドレスに置き換えます。詳細については、Amazon VPC ユーザーガイドのVPC とサブネットのサイズ設定を参照してください。

エラー：「キャッシュをダウンロードできません: RequestError: x509: システムルートをロードできませんでしたが、ルートが指定されていません」が原因でリクエストの送信に失敗しました

問題: ビルドプロジェクトを実行しようとすると、このエラーが発生してビルドが失敗する。

考えられる原因: ビルドプロジェクトの一部としてキャッシュを設定し、有効期限が切れたルート証明書を含む古い Docker イメージを使用しています。

推奨される解決策： AWS CodeBuild プロジェクトで使用されている Docker イメージを更新します。詳細については、「が提供する Docker イメージ CodeBuild」を参照してください。

エラー：「S3 から証明書をダウンロードできません AccessDenied」

問題: ビルドプロジェクトを実行しようとすると、このエラーが発生してビルドが失敗する。

考えられる原因:

- 証明書に正しくない S3 バケットが選択されています。
- 証明書に誤ったオブジェクトキーが入力されています。

推奨される解決策:

- プロジェクトを編集します。[証明書のバケット] では、SSL 証明書が保存されている S3 バケットを選択します。
- プロジェクトを編集します。[証明書のオブジェクトキー] に、S3 オブジェクトキーの名前を入力します。

エラー:「認証情報を見つけることができません」

問題： を実行したり AWS CLI、 SDK を使用した AWS り、ビルドの一部として別の類似コンポーネントを呼び出したりしようとすると、 AWS CLI、 AWS SDK、または コンポーネントに直接関連するビルドエラーが発生します。たとえば、「`Unable to locate credentials`」などのビルドエラーが発生する場合があります。

考えられる原因:

- ビルド環境の AWS CLI、 AWS SDK、またはコンポーネントのバージョンは、 と互換性がありません AWS CodeBuild。
- Docker を使用するビルド環境で Docker コンテナを実行しており、コンテナはデフォルトで AWS 認証情報にアクセスできません。

推奨される解決策:

- ビルド環境の AWS CLI、 AWS SDK、または コンポーネントのバージョンが次の 以上であることを確認します。
  - AWS CLI: 1.10.47

- AWS SDK for C++: 0.2.19
- AWS SDK for Go: 1.2.5
- AWS SDK for Java: 1.11.16
- AWS の SDK JavaScript: 2.4.7
- AWS SDK for PHP: 3.18.28
- AWS SDK for Python (Boto3): 1.4.0
- AWS SDK for Ruby: 2.3.22
- Botocore: 1.4.37
- CoreCLR: 3.2.6-beta
- Node.js: 2.4.7

- ビルド環境で Docker コンテナを実行する必要があり、コンテナに AWS 認証情報が必要な場合は、ビルド環境からコンテナに認証情報を渡す必要があります。buildspec ファイルでは、以下のような Docker `run` コマンドを組み込みます。この例では、`aws s3 ls` コマンドを使用して、使用可能な S3 バケットを一覧表示します。`-e` オプションは、コンテナが AWS 認証情報にアクセスするために必要な環境変数を渡します。

```
docker run -e AWS_DEFAULT_REGION -e AWS_CONTAINER_CREDENTIALS_RELATIVE_URI your-image-tag aws s3 ls
```

- Docker イメージを構築し、ビルドに AWS 認証情報が必要な場合 (Amazon S3 からファイルをダウンロードする場合など)、ビルド環境から Docker ビルドプロセスに認証情報を次のように渡す必要があります。

  1. Docker イメージ用ソースコードの Dockerfile に、次の `ARG` インストラクションを指定します。

     ```
     ARG AWS_DEFAULT_REGION
     ARG AWS_CONTAINER_CREDENTIALS_RELATIVE_URI
     ```

  2. buildspec ファイルでは、以下のような Docker `build` コマンドを組み込みます。`--build-arg` オプションは、Docker ビルドプロセスが AWS 認証情報にアクセスするために必要な環境変数を設定します。

     ```
     docker build --build-arg AWS_DEFAULT_REGION=$AWS_DEFAULT_REGION --build-arg
      AWS_CONTAINER_CREDENTIALS_RELATIVE_URI=$AWS_CONTAINER_CREDENTIALS_RELATIVE_URI -
     t your-image-tag .
     ```

RequestError プロキシサーバー CodeBuild での実行時のタイムアウトエラー

問題: 次のいずれかのような RequestError エラーが表示される。

- `RequestError: send request failed caused by: Post https://logs.<your-region>.amazonaws.com/: dial tcp 52.46.158.105:443: i/o timeout` CloudWatch Logs から。
- `Error uploading artifacts: RequestError: send request failed caused by: Put https://your-bucket.s3.your-aws-region.amazonaws.com/*: dial tcp 52.219.96.208:443: connect: connection refusedAmazon S3 の` 。

考えられる原因:

- `ssl-bump` が適切に設定されていません。
- 組織のセキュリティポリシーで `ssl_bump` を使用することが許可されていません。
- buildspec ファイルに、`proxy` 要素を使用して指定されたプロキシ設定がありません。

推奨される解決策:

- `ssl-bump` が適切に設定されていることを確認します。プロキシサーバーに Squid を使用している場合は、「 明示的なプロキシサーバーとしての Squid の設定」を参照してください。
- Amazon S3 と Logs のプライベートエンドポイントを使用するには、次の手順に従います。 CloudWatch

  1. プライベートサブネットのルーティングテーブルで、インターネット用トラフィックをプロキシサーバーにルーティングする追加済みのルールを削除します。詳細については、「Amazon VPC ユーザーガイド」の「VPC でサブネットを作成する」を参照してください。
  2. プライベート Amazon S3 エンドポイントと CloudWatch Logs エンドポイントを作成し、Amazon VPC のプライベートサブネットに関連付けます。詳細については、「Amazon VPC ユーザーガイド」の「VPC エンドポイントサービス」を参照してください。
  3. Amazon VPC の [プライベート DNS 名を有効にする] が選択されていることを確認します。詳細については、Amazon VPC ユーザーガイドのインターフェイスエンドポイントの作成を参照してください。

- 明示的なプロキシサーバーに ssl-bump を使用しない場合は、proxy 要素を使用して buildspec ファイルにプロキシ設定を追加します。詳細については、「明示的なプロキシサーバーでの CodeBuild の実行」および「buildspec の構文」を参照してください。

```
version: 0.2
proxy:
  upload-artifacts: yes
  logs: yes
phases:
  build:
    commands:
```

ビルドイメージ内に Bourne シェル (sh) が必要

問題： によって提供されていないビルドイメージを使用していて AWS CodeBuild、ビルドが失敗してメッセージ が表示されるBuild container found dead before completing the build。

考えられる原因： Bourne シェル (sh) は、ビルドコマンドとスクリプトを実行するshためにビルド image. CodeBuild needs に含まれていません。

推奨される解決策: ビルドイメージに sh が含まれていない場合、イメージを使用するビルドを開始する前に必ずそれを含めてください （ ビルドイメージshにCodeBuild は既に が含まれています ）。

警告 (ビルドの実行時): 「ランタイムのインストールをスキップします。このビルドイメージではランタイムバージョンの選択はサポートされていません」

問題: ビルドを実行すると、この警告がビルドログに表示される。

考えられる原因: ビルドでバージョン 1.0 以降の Amazon Linux 2 (AL2) 標準イメージ、またはバージョン 2.0 以降の Ubuntu 標準イメージが使用されておらず、buildspec ファイルの runtime-versions セクションにランタイムが指定されています。

推奨される解決策: buildspec ファイルに runtime-versions セクションが含まれていないことを確認します。runtime-versions セクションは、Amazon Linux 2 (AL2) 標準イメージ以降または Ubuntu 標準イメージのバージョン 2.0 以降を使用する場合にのみ必要です。

エラー: CodeBuild コンソールを開くときに JobWorker 「ID を確認できません」

問題： CodeBuild コンソールを開くと、 JobWorker 「ID を確認できません」というエラーメッセージが表示されます。

考えられる原因: コンソールのアクセスに使用されている IAM ロールに、`jobId` をキーとするタグがあります。このタグキーは 用に予約 CodeBuild されており、存在する場合はこのエラーが発生します。

推奨される解決策: `jobId` キーを持つカスタム IAM ロールタグを変更して、`jobIdentifier` などの別のキーを持つようにします。

ビルドの開始の失敗

問題: ビルドを開始すると、「ビルドを開始できませんでした」というエラーメッセージが表示される。

考えられる原因: 同時に実行できるビルド数の上限に達しています。

推奨される解決策: 他のビルドが完了するまで待つか、プロジェクトの同時実行のビルド制限を増やして、ビルドを再度開始します。詳細については、「プロジェクトの設定」を参照してください。

ローカルにキャッシュされたビルドの GitHub メタデータへのアクセス

問題: 状況によっては、キャッシュされたビルドの .git ディレクトリは、ディレクトリではなく、テキストファイルである場合があります。

考えられる原因： ビルドでローカルソースキャッシュが有効になっている場合、 は `.git` ディレクトリの gitlink CodeBuild を作成します。つまり、`.git` ディレクトリは、単にディレクトリへのパスを含むテキストファイルです。

推奨される解決策: すべての場合において、次のコマンドを使用して Git メタデータディレクトリを取得します。このコマンドは、`.git` のフォーマットに関係なく機能します。

```
git rev-parse --git-dir
```

AccessDenied: レポートグループのバケット所有者が S3 バケットの所有者と一致しません...

問題： テストデータを Amazon S3 バケットにアップロードすると、 CodeBuild はテストデータをバケットに書き込めません。

考えられる原因:

- レポートグループのバケット所有者に指定されたアカウントが、Amazon S3 バケットの所有者と一致しません。
- サービスロールには、バケットへの書き込みアクセスがありません。

推奨される解決策:

- Amazon S3 バケットの所有者と一致するよう、レポートグループのバケット所有者を変更します。
- Amazon S3 バケットへの書き込みアクセスを許可するようにサービスロールを変更します。

AWS CodeBuild のクォータ

次の表は、AWS CodeBuild の現在のクォータを示しています。これらのクォータは、特に指定のない限り、各 AWS アカウントでサポートされている AWS リージョンごとに適用されます。

サービスクォータ

AWS CodeBuild サービスのデフォルトのクォータを次に示します。

名前	デフォルト	引き上げ可能	説明
プロジェクトごとの関連タグ	サポートされている各リージョン: 50	いいえ	ビルドプロジェクトに関連付けることができるタグの最大数
ビルドプロジェクト	サポートされている各リージョン: 5,000	はい	ビルドするプロジェクトの最大数
ビルドのタイムアウト (分)。	サポートされている各リージョン: 480	いいえ	ビルドの最大タイムアウト (分)
ビルドに関する情報に対する同時要求	サポートされている各リージョン: 100	いいえ	AWS CLI または AWS SDK を使用して、同時に情報をリクエストできるビルドの最大数。
ビルドプロジェクトに関する情報の同時要求	サポートされている各リージョン: 100	いいえ	AWS CLI または AWS SDK を使用して、同時に情報をリクエストできる

名前	デフォルト	引き上げ可能	説明
			ビルドプロジェクトの最大数。
ARM Lambda/10GB 環境向けの同時実行ビルド数	サポートされている各リージョン: 1	はい	ARM Lambda/10GB 環境向けの同時実行ビルドの最大数
ARM Lambda/1GB 環境向けの同時実行ビルド数	サポートされている各リージョン: 1	はい	ARM Lambda/1GB 環境向けの同時実行ビルドの最大数
ARM Lambda/2GB 環境向けの同時実行ビルド数	サポートされている各リージョン: 1	はい	ARM Lambda/2GB 環境向けの同時実行ビルドの最大数
ARM Lambda/4GB 環境向けの同時実行ビルド数	サポートされている各リージョン: 1	はい	ARM Lambda/4GB 環境向けの同時実行ビルドの最大数
ARM Lambda/8GB 環境向けの同時実行ビルド数	サポートされている各リージョン: 1	はい	ARM Lambda/8GB 環境向けの同時実行ビルドの最大数
ARM/Large 環境向けのビルドの同時実行	サポートされている各リージョン: 1	はい	ARM/Large 環境向けのビルドの同時実行の最大数
ARM/Small 環境向けのビルドの同時実行	サポートされている各リージョン: 1	はい	ARM/Small 環境向けのビルドの同時実行の最大数
Linux GPU Large 環境向けのビルドの同時実行	サポートされている各リージョン: 0	はい	Linux GPU/Large 環境向けのビルドの同時実行の最大数

名前	デフォルト	引き上げ可能	説明
Linux GPU Small 環境向けのビルドの同時実行	サポートされている各リージョン: 0	はい	Linux GPU/Small 環境向けのビルドの同時実行の最大数
Linux Lambda/10GB 環境向けの同時実行ビルド数	サポートされている各リージョン: 1	はい	Linux Lambda/10GB 環境向けの同時実行ビルドの最大数
Linux Lambda/1GB 環境向けの同時実行ビルド数	サポートされている各リージョン: 1	はい	Linux Lambda/1GB 環境向けの同時実行ビルドの最大数
Linux Lambda/2GB 環境向けの同時実行ビルド数	サポートされている各リージョン: 1	はい	Linux Lambda/2GB 環境向けの同時実行ビルドの最大数
Linux Lambda/4GB 環境向けの同時実行ビルド数	サポートされている各リージョン: 1	はい	Linux Lambda/4GB 環境向けの同時実行ビルドの最大数
Linux Lambda/8GB 環境向けの同時実行ビルド数	サポートされている各リージョン: 1	はい	Linux Lambda/8GB 環境向けの同時実行ビルドの最大数
Linux/2XLarge 環境向けのビルドの同時実行	サポートされている各リージョン: 0	はい	Linux/2XLarge 環境向けのビルドの同時実行の最大数
Linux/Large 環境向けのビルドの同時実行	サポートされている各リージョン: 1	はい	Linux/Large 環境向けのビルドの同時実行の最大数

名前	デフォルト	引き上げ可能	説明
Linux/Medium 環境向けのビルドの同時実行	サポートされている各リージョン: 1	はい	Linux/Medium 環境向けのビルドの同時実行の最大数
Linux/Small 環境向けのビルドの同時実行	サポートされている各リージョン: 1	はい	Linux/Small 模環境向けのビルドの同時実行の最大数
Linux/XLarge 環境向けのビルドの同時実行	サポートされている各リージョン: 1	はい	Linux/XLarge 環境向けのビルドの同時実行の最大数
Windows Server 2019/Large 環境向けのビルドの同時実行	サポートされている各リージョン: 1	はい	Windows Server 2019/Large 環境向けのビルドの同時実行の最大数
Windows Server 2019/Medium 環境向けのビルドの同時実行	サポートされている各リージョン: 1	はい	Windows Server 2019/Medium 環境向けのビルドの同時実行の最大数
Windows/Large 環境向けのビルドの同時実行	サポートされている各リージョン: 1	はい	Windows/Large 環境向けのビルドの同時実行の最大数
Windows/Medium 環境向けのビルドの同時実行	サポートされている各リージョン: 1	はい	Windows/Medium 環境向けのビルドの同時実行の最大数
ビルドタイムアウトの最小期間 (分単位)	サポートされている各リージョン: 5	いいえ	ビルドの最大タイムアウト (分)

名前	デフォルト	引き上げ可能	説明
VPC 設定のセキュリティグループ	サポートされている各リージョン: 5	いいえ	VPC 設定で利用可能なセキュリティグループ
VPC 設定のサブネット	サポートされている各リージョン: 16	いいえ	VPC 設定で利用可能なサブネット

> **Note**
>
> 内部メトリクスは、同時実行ビルドのデフォルトクォータを決定します。

同時実行ビルドの最大数のクォータは、コンピューティングタイプによって異なります。一部のプラットフォームとコンピューティングタイプでは、デフォルトは 20 です。同時ビルドのクォータの引き上げをリクエストする場合や、「アカウントのアクティブなビルドは X 以上持つことはできません」というエラーが発生した場合は、上記リンクでご依頼ください。料金の詳細については、「AWS CodeBuild の料金」を参照してください。

その他の制限

ビルドプロジェクト

リソース	デフォルト値
ビルドプロジェクトの説明に使用できる文字	すべて
ビルドプロジェクト名に使用できる文字	文字 `A-Z` および `a-z`、数字 `0-9`、特殊文字 `-` および `_`

リソース	デフォルト値
ビルドプロジェクト名の長さ	2 ~ 255 文字以内
ビルドプロジェクトの説明の最大長	255 文字
プロジェクトに追加できるレポートの最大数	5
すべての関連ビルドのビルドタイムアウトのためにビルドプロジェクトで指定できる時間 (分)	5~480 (8 時間)

構築数

[リソース]	デフォルト値
ビルドの履歴が保持される最大時間	1 年
1 つのビルドのビルドタイムアウトのために指定できる時間 (分)	5~480 (8 時間)

コンピューティングフリート

[リソース]	デフォルト値
コンピューティングフリートの同時実行数	10
ARM/Small 環境向けのインスタンスの同時実行数	1
ARM/Large 環境向けのインスタンスの同時実行数	1
Linux/Small 環境向けのインスタンスの同時実行数	1
Linux/Medium 環境向けのインスタンスの同時実行数	1

[リソース]	デフォルト値
Linux/Large 環境向けのインスタンスの同時実行数	1
Linux/XLarge 環境向けのインスタンスの同時実行数	1
Linux/2XLarge 環境向けのインスタンスの同時実行数	0
Linux GPU/Small 環境向けのインスタンスの同時実行数	0
Linux GPU/Large 環境向けのインスタンスの同時実行数	0
Windows Server 2019/Medium 環境フリート向けのインスタンスの同時実行数	1
Windows Server 2019/Large 環境フリート向けのインスタンスの同時実行数	1
Windows Server 2022/Medium 環境フリート向けのインスタンスの同時実行数	1
Windows Server 2022/Large 環境フリート向けのインスタンスの同時実行数	1

レポート

[リソース]	デフォルト値
テストレポートの作成後の最大利用期間	30 日間
テストケースメッセージの最大長	5,000 文字
テストケース名の最大長	1,000 文字

[リソース]	デフォルト値
AWS アカウントあたりのレポートグループの最大数	5,000
レポートあたりのテストケースの最大数	500

タグ

タグの制限は、CodeBuild ビルドプロジェクトと CodeBuild レポートグループリソースのタグに適用されます。

リソース	デフォルト値
リソースタグのキー名	任意の組み合わせで使用できる文字は、Unicode 文字、数字、スペース、および許可されている UTF-8 文字 (1 ~ 127 文字) です。使用できる文字は次のとおりです: `+ - = . _ : / @` タグキー名は一意である必要があり、各キーに使用できる値は 1 つのみです。タグキー名に以下のことはできません。 • `aws:` から始まる • 空白文字のみで構成されている • 末尾にスペースを使用する • 絵文字、または以下の文字を含める: `? ^ * [ \ ~ ! # $ % & * ( ) > < | " ' ` [ ] { } ;`
リソースタグの値	任意の組み合わせで使用できる文字は、Unicode 文字、数字、スペース、および許可されている UTF-8 文字 (0 ~ 255 文字) です。使用できる文字は次のとおりです: `+ - = . _ : / @`

リソース	デフォルト値
	キーに使用できる値は 1 つのみですが、多数のキーと同じ値を含めることができます。タグのキー値に絵文字や次の文字を含めることはできません。 ? ^ * [\ ~ ! # $ % & * ( ) > < | " ' ` [ ] { } ;

AWS CodeBuild for Windows に関するサードパーティーの通知

CodeBuild Windows ビルドに を使用する場合、一部のサードパーティ製パッケージとモジュールを使用して、構築したアプリケーションを Microsoft Windows オペレーティングシステムで実行し、一部のサードパーティ製品と相互運用するオプションがあります。指定したサードパーティー製のパッケージやモジュールの使用に適用されるサードパーティーの法的条項を以下に示します。

トピック

1) 基本 Docker イメージ – windowsservercore

(ライセンス条項は https://hub.docker.com/_/microsoft-windows-servercore)

ライセンス: この Windows コンテナー用のコンテナー OS イメージを要求および使用する場合、次の追加ライセンス条項を承認、理解し、同意するものと見なされます。

Microsoft ソフトウェア追加ライセンス条項

コンテナー OS イメージ

Microsoft Corporation (またはお住まいの地域に基づいていずれかの関連会社) (以下 "Microsoft") は、このコンテナー OS イメージの追加機能 (以下 "追加機能") のライセンスをユーザーに付与します。この追加機能と、基になるホストオペレーティングシステムソフトウェア (以下 "ホストソフトウェア") を組み合わせて使用するライセンスは、ホストソフトウェアでのコンテナー機能の実行を支援する目的でのみ付与されます。この追加機能の使用には、ホストソフトウェアのライセンス条項が適用されます。ホストソフトウェアのライセンスを持っていない場合、使用することはできません。ライセンスが有効なホストソフトウェアのコピーがある場合にのみ、この追加機能を使用できます。

その他のライセンス要件と使用権

前述の条項に従って追加機能を使用すると、特定の追加機能コンポーネントを含むコンテナーイメージ (以下 "コンテナーイメージ") が作成または変更される場合があります。明確にしておくと、コンテナーイメージは、仮想マシンまたは仮想アプライアンスイメージとは別のものです。Microsoft は、本ライセンス条項に従い、以下の条件で、そのような追加機能コンポーネントを再配布する限定的な権利をユーザーに付与します。

(i) 追加機能コンポーネントは、ユーザーのコンテナーイメージ内かつユーザーのコンテナーイメージの一部としてのみ使用できます。

(ii) 追加機能とは実質的に異なる重要な主機能がコンテナーイメージにある場合に限り、コンテナーイメージ内の追加機能コンポーネントを使用できます。

(iii) エンドユーザーが追加機能コンポーネントを使用する場合に適切にライセンスが付与されるように、本ライセンス条項 (または Microsoft やホスト側で必須とする同様の条項) をコンテナーイメージに含めることに同意します。

Microsoft は、本条項で明記されていないその他のすべての権利を有します。

本追加ソフトウェアを使用することにより、お客様はこれらの条項に同意されたものとします。以下の条項に同意されない場合、本追加ソフトウェアは使用しないでください。

この Windows コンテナー用のコンテナー OS イメージの追加ライセンス条項の一部として、基になる Windows Server ホストソフトウェアのライセンス条項 (https://www.microsoft.com/en-us/useterms.) も適用されます。

2) Windows ベースの Docker イメージ – Choco

(ライセンス条項は https://github.com/chocolatey/choco/blob/master/LICENSE)

著作権 2011 - 現在の RealDimensions ソフトウェア、ワシントン

Apache License Version 2.0 (以下「本ライセンス」) に基づいてライセンスされます。これらのファイルを使用するには、本ライセンスに準拠する必要があります。本ライセンスのコピーは下記の場所から入手できます。

http://www.apache.org/licenses/LICENSE-2.0

適用される法律または書面での同意によって義務付けられない限り、本ライセンスに基づいて頒布されるソフトウェアは、明示または黙示を問わず、いかなる保証も条件もなしに「現状のまま」頒布されます。本ライセンスでの権利と制限を規定した文言については、本ライセンスを参照してください。

3) Windows ベースの Docker イメージ – git --version 2.16.2

(ライセンス条項の参照先: https://chocolatey.org/packages/git/2.16.2)

GNU 一般公衆ライセンス、バージョン 2 に基づいてライセンスされます。参照先: https://www.gnu.org/licenses/old-licenses/gpl-2.0.html

4) Windows ベースの Docker イメージ —microsoft-build-tools - バージョン 15.0.26320.2

(ライセンス条項の参照先: https://www.visualstudio.com/license-terms/mt171552/)

MICROSOFT VISUAL STUDIO 2015 拡張機能、VISUAL STUDIO SHELLS および C++ 再頒布可能パッケージ

本ライセンス条項は、Microsoft Corporation (またはお客様の所在地に応じてはその関連会社) とお客様との契約を構成します。本ライセンス条項は、上記のソフトウェア (以下「本ソフトウェア」といいます) に適用されます。本ライセンス条項は、別途のライセンス条項が付属している場合を除き、本ソフトウェアに関連するマイクロソフトのサービスまたは更新プログラムにも適用されます。

本ライセンス条項を遵守することを条件として、お客様には以下の権利が許諾されます。

1. インストールおよび使用に関する権利 お客様は、本ソフトウェアの任意の数の複製をインストールして使用することができます。
2. 特定のコンポーネントに関する条件

a. ユーティリティ。本ソフトウェアには、https://docs.microsoft.com/en-us/visualstudio/productinfo/2015-redistribution-vs のユーティリティリストにいくつかの項目が含まれている場合があります。 お客様は、本ソフトウェアに含まれている場合、それらの項目をお客様のまたは他の第三者マシンにコピーしてインストールし、本ソフトウェアで開発したアプリケーションおよびデータベースをデバッグおよびデプロイすることができます。ユーティリティは一時的な使用を目的として設計されていること、マイクロソフトは本ソフトウェアの他のコンポーネントと切り離してユーティリティにパッチを適用したり、ユーティリティを更新したりできない場合があること、および一部のユーティリティはその性質上、そのユーティリティがインストールされているコンピュータに他者がアクセスできるようにすることが可能であることに注意してください。このため、お客様は、お客様のアプリケーションおよびデータベースのデバッグまたは展開が終了した後で、お客様がインストールしたすべてのユーティリティを削除する必要があります。マイクロソフトは、お客様が任意のコンピュータにインストールしたユーティリティの第三者による使用またはアクセスについて責任を負いません。

b. マイクロソフトプラットフォーム 本ソフトウェアには、Microsoft Windows、Microsoft Windows Server、Microsoft SQL Server、Microsoft Exchange、Microsoft Office、および Microsoft のコンポーネントが含まれている場合があります SharePoint。これらのコンポーネントには、本ソフトウェアに付属しているマイクロソフトの「Licenses」フォルダーに規定されている、別途のライセンス条項および固有の製品サポートポリシーが適用されます。ただし、関連するインストールディレクトリにこれらのコンポーネントのライセンス条項も含まれている場合は当該ライセンス条項が適用されます。

c. 第三者のコンポーネント 本ソフトウェアには、本ソフトウェアに付属する ThirdPartyNotices ファイルに記載されているように、個別の法的通知または他の契約が適用される第三者のコンポーネントが含まれる場合があります。かかるコンポーネントには、他の契約が適用される場合でも、以下の保証の免責、損害賠償の制限および除外も適用されます。本ソフトウェアには、ソースコードの公開義務が適用されるオープンソースライセンスに基づいてライセンスが許諾されるコンポーネントも含まれている場合があります。該当する場合、これらのライセンスのコピーは ThirdPartyNotices ファイルに含まれています。お客様は、当該オープンソースライセンスで求められているとおり、5.00 米ドルの郵便為替または小切手を次の宛先に送付することにより、対応するソースコードをマイクロソフトから取得することができます: Source Code Compliance Team, Microsoft Corporation, 1 Microsoft Way, Redmond, WA 98052。支払いの備考欄には、下記の 1 つまたは複数のコンポーネントのソースコードを記載してください。

- Remote Tools for Visual Studio 2015
- Standalone Profiler for Visual Studio 2015
- IntelliTraceCollector for Visual Studio 2015

- Microsoft VC++ Redistributable 2015
- Multibyte MFC Library for Visual Studio 2015
- Microsoft Build Tools 2015
- Feedback Client
- Visual Studio 2015 Integrated Shell
- Visual Studio 2015 Isolated Shell

マイクロソフトは、ソースコードの複製を http://thirdpartysource.microsoft.com で公開することもあります。

3. データ。本ソフトウェアは、お客様およびお客様による本ソフトウェアの使用に関する情報を収集し、マイクロソフトに送信することがあります。マイクロソフトはこの情報を、サービスの提供ならびにマイクロソフトの製品およびサービスの向上を目的として使用することがあります。お客様は、製品付属の文書に説明されているとおり、これらの情報収集の多くを停止することができますが、すべてを停止することはできません。また、本ソフトウェアにある特定の機能を使用すると、お客様がお客様のアプリケーションのユーザーからデータを収集できる場合があります。お客様は、これらの機能を使用する場合、お客様のアプリケーションのユーザーに適切な通知を提供するなど、適用される法令を遵守しなければなりません。データの収集および使用の詳細については、「https://privacy.microsoft.com/en-us/privacystatement」のヘルプドキュメントおよびマイクロソフトのプライバシーに関する声明を参照してください。本ソフトウェアを使用した場合、お客様はこれらの規定に同意したものとみなされます。
4. ライセンスの適用範囲 本ソフトウェアは使用許諾されるものであり、販売されるものではありません。本ライセンス条項は、お客様に本ソフトウェアを使用する限定的な権利を許諾します。その他の権利はすべてマイクロソフトが留保します。適用される法令に基づいて本ライセンス条項の制限を超える権利が許諾される場合を除き、お客様は本ライセンス条項で明示的に許可された方法でのみ本ソフトウェアを使用することができます。お客様は、ソフトウェアに組み込まれた使用方法を制限する技術的制限に従うものとします。以下の行為は禁じられています。
   - 本ソフトウェアの技術的な制限を回避すること。
   - 本ソフトウェアのリバースエンジニアリング、逆コンパイル、もしくは逆アセンブルを実行または試行すること。ただし、本ソフトウェアに含まれる場合がある一定のオープンソースコンポーネントの使用に適用される第三者のライセンス条項により求められている場合を除きます。
   - 本ソフトウェアの Microsoft またはサプライヤーの告知を削除、最小化、ブロックまたは修正すること。
   - 法律に違反する方法で本ソフトウェアを使用すること。

- 本ソフトウェアを共有、公開、レンタル、もしくはリースすること、本ソフトウェアを第三者が使用できるようにスタンドアロンのホスト型ソリューションとして提供すること。

5. 輸出規制 お客様は、本ソフトウェアに適用されるすべての国内法および国際法 (輸出対象国、エンドユーザーおよびエンドユーザーによる使用に関する制限を含みます) を遵守しなければなりません。輸出規制の詳細については、(aka.ms/exporting) を参照してください。
6. サポートサービス 本ソフトウェアは「現状有姿のまま」で提供されるため、マイクロソフトは本ソフトウェアに関してサポートサービスを提供しない場合があります。
7. 完全合意 本ライセンス条項ならびにお客様が使用する追加物、更新プログラム、インターネットベースのサービスおよびサポートサービスに関する条項は、本ソフトウェアおよびサポートサービスについてのお客様とマイクロソフトとの間の完全なる合意を構成します。
8. 準拠法 お客様が本ソフトウェアを米国内で入手された場合、本ライセンス条項の解釈および契約違反への主張は、米国ワシントン州法に準拠するものとします。他の主張については、お客様が所在する地域の法律に準拠します。お客様が本ソフトウェアを他の国で入手した場合は、当該地域の法律を準拠法とします。
9. 消費者の権利、地域による違い 本契約は、特定の法的な権利を規定したものです。お客様は、地域や国によっては、消費者権利を含め、その他の権利を有する場合があります。Microsoft とお客様との関係とは別に、お客様が本ソフトウェアを取得した当事者に関する権利を有する場合もあります。本契約は、お客様の地域または国の法令が権利の変更を許容しない場合、それらのその他の権利を変更しないものとします。たとえば、お客様が本ソフトウェアを以下のいずれかの地域で取得した場合、または強行的な国の法令が適用される場合には、以下の規定がお客様に適用されます。

   a. オーストラリア お客様は、オーストラリア消費者法に基づく法定保証を有し、本ライセンス条項は、それらの権利に影響を与えることを意図するものではありません。

   b. カナダ。本ソフトウェアをカナダで取得した場合、自働更新機能をオフにするか、お使いの機器をインターネットから取り外すか (ただし、インターネットに再接続すると、本ソフトウェアは更新プログラムのチェックとインストールを再開します)、または本ソフトウェアをアンインストールすることにより、更新受信を停止することができます。製品付属の文書がある場合は、当該文書にお客様の特定のデバイスまたはソフトウェアの更新をオフにする方法が記載されていることもあります。

   c. ドイツおよびオーストリア

      i. 保証 正規にライセンスを取得したソフトウェアは、本ソフトウェアに付属するマイクロソフトの資料の記載に実質的に従って動作します。ただし、マイクロソフトは、ライセンスを取得したソフトウェアに関して契約上の保証は一切いたしません。

ii. 限定責任 故意、重過失、製品責任法に基づく請求があった場合、および死亡、人的または物的損傷があった場合、Microsoft は、制定法に従って責任を負うものとします。前掲条項 (ii) を条件とし、Microsoft が軽過失に該当する契約違反をして、同義務を履行することは本契約の正当な履行に資するものであって、同義務の違反は本契約の目的および当事者が常に拠り所とする本契約への準拠を損なう (いわゆる「基本的義務」に違反する) 可能性がある場合、Microsoft は当該の軽過失についてのみ責任を負うものとします。その他の軽過失については、マイクロソフトは責任を負いません。

10.保証の免責: 本ソフトウェアは、「現状有姿のまま」ライセンス供与されます。本ソフトウェアの使用に伴うリスクは、お客様が負うものとします。マイクロソフトは、明示的な保証を一切いたしません。お客様の地域の法律によって認められる範囲において、マイクロソフトは、商品性、特定目的に対する適合性、および侵害の不存在に関する黙示の保証責任を負いません。

11.損害賠償に関する制限および除外 YOU CAN RECOVER FROM MICROSOFT およびその供給者 ONLY 直接的損害 UP TO 米国 5.00 USD となります。マイクロソフトは、派生的損害、逸失利益、特別損害、間接損害、または付随的損害を含め、その他の損害について一切責任を負いません。この制限は、(a) 本ソフトウェア、サービス、第三者のインターネットのサイト上のコンテンツ (コードを含みます) または第三者のアプリケーションに関連した事項、および (b) 契約違反、保証違反、厳格責任、過失、または不法行為等の請求 (適用される法令により認められている範囲において) に適用されます。

この制限は、マイクロソフトがこのような損害の可能性を認識していたか、または認識しえた場合にも適用されます。国によっては付随的損害、派生的損害またはその他の損害の除外または制限を認めていないことがあるため、上記の制限または除外がお客様に適用されない場合があります。

EULA ID: VS2015_Update3_ShellsRedist_<ENU>

5) Windows ベースの Docker イメージ – nuget.commandline --version 4.5.1

(ライセンス条項は、https://github.com/NuGet/Home/blob/dev/LICENSE.txt にあります)

Copyright (c) .NET Foundation。All rights reserved。

Apache License Version 2.0 (以下「本ライセンス」) に基づいてライセンスされます。これらのファイルを使用するには、本ライセンスに準拠する必要があります。本ライセンスのコピーは下記の場所から入手できます。

http://www.apache.org/licenses/LICENSE-2.0

適用される法律または書面での同意によって義務付けられない限り、本ライセンスに基づいて頒布されるソフトウェアは、明示または黙示を問わず、いかなる保証も条件もなしに「現状のまま」頒布されます。本ライセンスでの権利と制限を規定した文言については、本ライセンスを参照してください。

7) Windows ベースの Docker イメージ – netfx-4.6.2-devpack

Microsoft ソフトウェア追加ライセンス条項

.NET FRAMEWORK AND ASSOCIATED LANGUAGE PACKS FOR MICROSOFT WINDOWS OPERATING SYSTEM

Microsoft Corporation (またはお客様の所在地に応じてはその関連会社) は、本追加ソフトウェアのライセンスをお客様に供与します。Microsoft Windows operating system ソフトウェア (以下「本ソフトウェア」といいます) を使用するためのラインセンスを取得している場合は、本追加ソフトウェアを使用できます。本ソフトウェアのライセンスを取得していない場合は、本追加ソフトウェアを使用することはできません。お客様は、本ソフトウェアの有効なライセンス取得済みの複製 1 部ごとに本追加ソフトウェアを使用できます。

以下のライセンス条項は、本ソフトウェアの追加の使用条件について説明しています。これらの条項と本ソフトウェアのライセンス条項が本追加ソフトウェアの使用に適用されます。両者の間に矛盾がある場合は、本追加ライセンス条項が適用されます。

本追加ソフトウェアを使用することにより、お客様はこれらの条項に同意されたものとします。以下の条項に同意されない場合、本追加ソフトウェアは使用しないでください。

本ライセンス条項を遵守することを条件として、お客様には以下の権利が許諾されます。

1. 配布可能なコード。 本追加ソフトウェアは頒布可能コードで構成されています。「頒布可能コード」とは、お客様が開発されたプログラムに含めて頒布することができるコードです。ただし、お客様は以下の条件に従うものとします。
   a. 使用および頒布の権利
      - お客様は、本追加ソフトウェアをオブジェクトコード形式で複製し、頒布することができます。

- 第三者による頒布。お客様は、お客様のプログラムの頒布者に対して、お客様のプログラムの一部として頒布可能コードの複製および頒布を許可することができます。

b. 頒布の条件 お客様は、お客様が頒布するすべての頒布可能コードにつき、以下に従わなければなりません

- お客様のプログラムにおいて頒布可能コードに重要な新しい機能を追加すること
- .lib というファイル名拡張子が付いた頒布可能コードの場合は、リンカーによってその頒布可能コードを実行した結果だけをお客様のプログラムと共に頒布すること
- セットアッププログラムに含まれる頒布可能コードを、改変されていないセットアッププログラムの一部としてのみ頒布すること
- お客様のアプリケーションの頒布者およびエンドユーザーに、本ライセンス条項と同等以上に頒布可能コードを保護する条項に同意させること
- お客様のアプリケーションにお客様名義の有効な著作権表示を行うこと
- お客様のプログラムの頒布または使用に関するクレームについて、マイクロソフトを免責、保護、補償すること (弁護士費用についての免責、保護、補償も含む)

c. 頒布の制限 以下の行為は禁じられています

- 頒布可能コードの著作権、商標または特許の表示を改変すること
- お客様のプログラムの名称の一部にマイクロソフトの商標を使用したり、お客様のプログラムがマイクロソフトから由来したり、マイクロソフトが推奨しているように見せかけること
- Windows プラットフォーム以外のプラットフォームで実行する目的で頒布可能コードを頒布すること
- 頒布可能コードを悪質、詐欺的または違法なプログラムに組み込むこと
- 除外ライセンスの適用対象となるような方法で頒布可能コードのソースコードを改変または頒布すること。「除外ライセンス」とは、使用、改変または頒布の条件として以下を義務付けるライセンスです。
  - コードをソースコード形式で公表または頒布すること
  - 他者が改変する権利を有すること

2. 本追加ソフトウェアのサポート サービス マイクロソフトは、本ソフトウェアに対して、www.support.microsoft.com/common/international.aspx に記載するサポートサービスを提供します。

8) Windows ベースの Docker イメージ – visualfsharptools, v 4.0

(ライセンス条項の参照先: https://github.com/dotnet/fsharp/blob/main/License.txt)

Apache License Version 2.0 (以下「本ライセンス」) に基づいてライセンスされます。これらのファイルを使用するには、本ライセンスに準拠する必要があります。本ライセンスのコピーは下記の場所から入手できます。

http://www.apache.org/licenses/LICENSE-2.0

適用される法律または書面での同意によって義務付けられない限り、本ライセンスに基づいて頒布されるソフトウェアは、明示または黙示を問わず、いかなる保証も条件もなしに「現状のまま」頒布されます。本ライセンスでの権利と制限を規定した文言については、本ライセンスを参照してください。

9) Windows ベースの Docker イメージ -netfx-pcl-reference-assemblies-4.6

Microsoft ソフトウェアライセンス条項

MICROSOFT .NET PORTABLE CLASS LIBRARY REFERENCE ASSEMBLIES – 4.6

本ライセンス条項は、Microsoft Corporation (またはお客様の所在地に応じてはその関連会社) とお客様との契約を構成します。本ライセンス条項をお読みください。本ライセンス条項は、上記のソフトウェア (以下「本ソフトウェア」といいます) に適用されます。本ライセンス条項は、本ソフトウェアに関連するマイクロソフトの以下の各項目にも適用されます。

- 更新,
- 追加プログラム
- インターネットベースのサービス
- サポートサービス

ただし、上記項目に別途のライセンス条項が付属している場合を除きます。別途のライセンス条項が付属している場合は、それらの別途のライセンス条項が適用されます。

本ソフトウェアを使用することにより、お客様はこれらの条項に同意されたものとします。これらの条項に同意されない場合、本ソフトウェアは使用しないでください。

本ライセンス条項を遵守することを条件として、お客様には以下の永続的な権利が許諾されます。

1. インストールおよび使用に関する権利 お客様は、お客様のプログラムを設計、開発およびテストするために、本ソフトウェアの任意の数の複製をインストールして使用することができます。
2. その他のライセンス要件と使用権
   a. 頒布可能コード。お客様は、以下の条項を遵守することを条件として、お客様が開発した開発者ツールプログラムで本ソフトウェアを配布し、お客様のプログラムのユーザーに対してポータブルライブラリを開発して任意のデバイスまたはオペレーティングシステムで使用することを許可できます。
      i. 使用および頒布の権利 本ソフトウェアは「頒布可能コード」です。
         - 頒布可能コード。お客様は、本ソフトウェアをオブジェクトコード形式で複製し、頒布することができます。
         - 第三者による頒布。お客様は、お客様のプログラムの頒布者に対して、お客様のプログラムの一部として頒布可能コードの複製および頒布を許可することができます。
      ii. 頒布の条件 お客様は、お客様が頒布するすべての頒布可能コードにつき、以下に従わなければなりません
         - お客様のプログラムにおいて頒布可能コードに重要な新しい機能を追加すること
         - お客様のアプリケーションの頒布者およびユーザーに、本ライセンス条項と同等以上に頒布可能コードを保護する条項に同意させること
         - お客様のアプリケーションにお客様名義の有効な著作権表示を行うこと
         - お客様のプログラムの頒布または使用に関するクレームについて、マイクロソフトを免責、保護、補償すること (弁護士費用についての免責、保護、補償も含む)
      iii. 頒布の制限 以下の行為は禁じられています
         - 頒布可能コードの著作権、商標または特許の表示を改変すること
         - お客様のプログラムの名称の一部にマイクロソフトの商標を使用したり、お客様のプログラムがマイクロソフトから由来したり、マイクロソフトが推奨しているように見せかけること
         - 頒布可能コードを悪質、詐欺的または違法なプログラムに組み込むこと
         - 除外ライセンスの適用対象となるような方法で頒布可能コードを改変または頒布すること。「除外ライセンス」とは、使用、改変または頒布の条件として以下を義務付けるライセンスです。
         - コードをソースコード形式で公表または頒布すること

- 他者が改変する権利を有すること

3. ライセンスの適用範囲 本ソフトウェアは使用許諾されるものであり、販売されるものではありません。本ライセンス条項は、お客様に本ソフトウェアを使用する限定的な権利を許諾します。その他の権利はすべてマイクロソフトが留保します。適用される法令に基づいて本ライセンス条項の制限を超える権利が許諾される場合を除き、お客様は本ライセンス条項で明示的に許可された方法でのみ本ソフトウェアを使用することができます。お客様は、ソフトウェアに組み込まれた使用方法を制限する技術的制限に従うものとします。以下の行為は禁じられています。
   - 本ソフトウェアの技術的な制限を回避すること。
   - 本ソフトウェアをリバースエンジニアリング、逆コンパイル、もしくは逆アセンブルすること。ただし、この制限にもかかわらず、適用される法によって明示的に許可される場合を除きます。
   - 本ソフトウェアを公開して第三者に複製させること。
   - 本ソフトウェアをレンタル、リースまたは貸与すること。
4. フィードバック お客様は、本ソフトウェアに関するフィードバックを提供できます。本ソフトウェアに関するフィードバックをお客様がマイクロソフトに提供した場合は、方法および目的を問わず、そのフィードバックを無償で使用、公開、および商用利用する権利をお客様がマイクロソフトに付与したものと見なされます。また、そのフィードバックが含まれているマイクロソフトのソフトウェアまたはサービスの特定部分が、第三者の製品、テクノロジ、およびサービスによって使用される場合またはその部分との連携が行われる場合に必要となる特許権についても、お客様が無償で付与したものと見なされます。お客様のフィードバックをソフトウェアまたはドキュメントに含めるために、マイクロソフトから第三者に対して、そのソフトウェアまたはドキュメントの使用許諾が必要となるようなライセンスが適用されるフィードバックは、お客様から提供されないものとします。これらの権利は、本契約の終了後も継続するものとします。
5. 第三者への譲渡 本ソフトウェアの最初のユーザーは、本ソフトウェアおよび本契約を第三者に譲渡できます。譲渡する前に、当該の第三者は、本契約が本ソフトウェアの譲渡と使用に適用されることに同意する必要があります。最初のユーザーは、本ソフトウェアを譲渡する前に、本ソフトウェアをデバイスからアンインストールする必要があります。最初のユーザーは、一切の複製を保持しないものとします。
6. 輸出規制 本ソフトウェアは、アメリカ合衆国の輸出に関する規制の対象となります。お客様は、本ソフトウェアに適用されるすべての国内外の輸出に関する法および規制を遵守しなければなりません。これらの法には、輸出対象国、エンドユーザーおよびエンドユーザーによる使用に関する制限が含まれます。詳細については、www.microsoft.com/exporting を参照してください。
7. サポートサービス 本ソフトウェアは「現状有姿のまま」で提供されるため、マイクロソフトは本ソフトウェアに関してサポートサービスを提供しない場合があります。

8. 完全合意 本ライセンス条項ならびにお客様が使用する追加物、更新プログラム、インターネットベースのサービスおよびサポートサービスに関する条項は、本ソフトウェアおよびマイクロソフトが提供するすべてのサポートサービスについてのお客様とマイクロソフトとの間の完全なる合意を構成します。

9. 準拠法

   a. アメリカ合衆国。お客様が本ソフトウェアを米国内で入手された場合、本ライセンス条項の解釈および契約違反への主張は、抵触法にかかわらず、米国ワシントン州法に準拠するものとします。他の主張については、消費者保護法、公正取引法、および違法行為に基づく主張も含めて、お客様が所在する地域の法律に準拠します。

   b. 米国以外 お客様が本ソフトウェアを他の国で入手した場合は、当該国の法律を準拠法とします。

10法的効力 本契約は、特定の法的な権利を規定したものです。お客様は、国の法律によっては、その他の権利を有する場合があります。また、お客様が本ソフトウェアを取得された第三者に関する権利を有する場合もあります。本ライセンス条項は、お客様の国の法律がその法律に基づく権利の変更を許容しない場合、それらの権利を変更しないものとします。

11保証の免責: 本ソフトウェアは "現状のまま" ライセンス供与されます。本ソフトウェアの使用に伴うリスクは、お客様が負うものとします。マイクロソフトは、明示的な保証を一切いたしません。本ライセンス条項では変更できない、お客様の地域の法律による追加の消費者の権利または法定保証が存在する場合があります。お客様の地域の法律によって認められる範囲において、マイクロソフトは、商品性、特定目的に対する適合性、および侵害の不存在に関する黙示の保証責任を負いません。

オーストラリア限定 – お客様は、オーストラリア消費者法に基づく法定保証を有し、本ライセンス条項は、それらの権利に影響を与えることを意図するものではありません。

12救済手段および損害賠償の制限および除外 YOU CAN RECOVER FROM MICROSOFT およびその供給者 ONLY 直接的損害 UP TO 米国 5.00 USD となります。マイクロソフトは、派生的損害、逸失利益、特別損害、間接損害、または付随的損害を含め、その他の損害について一切責任を負いません。

この制限は、以下に適用されるものとします。

- 本ソフトウェア、サービス、第三者のインターネットサイト上のコンテンツ (コードを含みます) または第三者のプログラムに関連した事項
- 契約違反、保証違反、無過失責任、過失または不法行為 (適用法で許可されている範囲において)

この制限は、マイクロソフトがこのような損害の可能性を認識していたか、または認識しえた場合にも適用されます。国によっては付随的損害、派生的損害またはその他の損害の除外または制限を認めていないことがあるため、上記の制限または除外がお客様に適用されない場合があります。

10) Windows ベース Docker イメージ — visualcppbuildtools v 14.0.25420.1

(ライセンス条項の参照先: https://www.visualstudio.com/license-terms/mt644918/)

MICROSOFT VISUAL C++ 構築ツール

Microsoft ソフトウェアライセンス条項

MICROSOFT VISUAL C++ 構築ツール

本ライセンス条項は、Microsoft Corporation (またはお客様の所在地に応じてはその関連会社) とお客様との契約を構成します。本ライセンス条項は、上記のソフトウェア (以下「本ソフトウェア」といいます) に適用されます。本ライセンス条項は、別途のライセンス条項が付属している場合を除き、本ソフトウェアに関連するマイクロソフトのサービスまたは更新プログラムにも適用されます。

本ライセンス条項を遵守することを条件として、お客様には以下の権利が許諾されます。

1. インストールおよび使用に関する権利
   a. 1 人のユーザーが、アプリケーションの開発およびテストを行うために、本ソフトウェアの複製を使用することができます。
2. データ。本ソフトウェアは、お客様およびお客様による本ソフトウェアの使用に関する情報を収集し、マイクロソフトに送信することがあります。マイクロソフトはこの情報を、サービスの提供ならびにマイクロソフトの製品およびサービスの向上を目的として使用することがあります。お客様は、製品付属の文書に説明されているとおり、これらの情報収集の多くを停止することができますが、すべてを停止することはできません。また、本ソフトウェアにある特定の機能を使用すると、お客様がお客様のアプリケーションのユーザーからデータを収集できる場合があります。お客様は、これらの機能を使用する場合、お客様のアプリケーションのユーザーに適切な通

知を提供するなど、適用される法令を遵守しなければなりません。データの収集および使用の詳細については、ヘルプドキュメントおよびマイクロソフトのプライバシーに関する声明を参照してください: http://go.microsoft.com/fwlink/?LinkID=528096。本ソフトウェアを使用した場合、お客様はこれらの規定に同意したものとみなされます。

3. 特定のコンポーネントに関する条件

   a. ビルドサーバー 本ソフトウェアには、 BuildServer.TXT ファイルに記載されている一部のビルドサーバーコンポーネント、および/または本 Microsoft ソフトウェアライセンス条項 BuildeServer に従ってリストされているファイルが含まれている場合があります。これらの項目が本ソフトウェアに含まれている場合は、これらを複製してビルドコンピューターにインストールすることができます。お客様およびお客様の組織内の他のユーザーは、お客様のアプリケーションのコンパイル、構築、検証、およびアーカイブと、構築プロセスの一環としての品質テストやパフォーマンステストを実行する目的に限り、ビルドコンピューターでこれらの項目を使用することができます。

   b. マイクロソフトプラットフォーム 本ソフトウェアには、Microsoft Windows、Microsoft Windows Server、Microsoft SQL Server、Microsoft Exchange、Microsoft Office、および Microsoft のコンポーネントが含まれている場合があります SharePoint。これらのコンポーネントには、本ソフトウェアに付属しているマイクロソフトの「Licenses」フォルダーに規定されている、別途のライセンス条項および固有の製品サポートポリシーが適用されます。ただし、関連するインストールディレクトリにこれらのコンポーネントのライセンス条項も含まれている場合は当該ライセンス条項が適用されます。

   c. 第三者のコンポーネント 本ソフトウェアには、本ソフトウェアに付属する ThirdPartyNotices ファイルに記載されているように、個別の法的通知または他の契約が適用される第三者のコンポーネントが含まれる場合があります。かかるコンポーネントには、他の契約が適用される場合でも、以下の保証の免責、損害賠償の制限および除外も適用されます。

   d. パッケージマネージャー 本ソフトウェアには、他のマイクロソフトや第三者のソフトウェアパッケージをダウンロードしてお客様のアプリケーションで使用できるようにするパッケージマネージャー (Nuget など) が含まれている場合があります。これらのパッケージには、独自のライセンスが適用され、本契約は適用されません。マイクロソフトは、第三者のパッケージの頒布、使用許諾、または保証の提供は行いません。

4. ライセンスの適用範囲 本ソフトウェアは使用許諾されるものであり、販売されるものではありません。本ライセンス条項は、お客様に本ソフトウェアを使用する限定的な権利を許諾します。その他の権利はすべてマイクロソフトが留保します。適用される法令に基づいて本ライセンス条項の制限を超える権利が許諾される場合を除き、お客様は本ライセンス条項で明示的に許可された方法でのみ本ソフトウェアを使用することができます。お客様は、ソフトウェアに組み込まれた使用方法を制限する技術的制限に従うものとします。詳細については、https://

docs.microsoft.com/en-us/legal/information-protection/software-license-terms#1- installation-and-use-rightsを参照してください。以下の行為は禁じられています。

- 本ソフトウェアの技術的な制限を回避すること。
- 本ソフトウェアのリバースエンジニアリング、逆コンパイル、もしくは逆アセンブルを実行または試行すること。ただし、本ソフトウェアに含まれる場合がある一定のオープンソースコンポーネントの使用に適用される第三者のライセンス条項により求められている場合を除きます。
- Microsoft またはサプライヤーの告知を削除、最小化、ブロックまたは修正すること。
- 法律に違反する方法で本ソフトウェアを使用すること。
- 本ソフトウェアを共有、公開、レンタル、もしくはリースすること、本ソフトウェアを第三者が使用できるようにスタンドアロンのホスト型ソリューションとして提供すること。

5. 輸出規制 お客様は、本ソフトウェアに適用されるすべての国内法および国際法 (輸出対象国、エンドユーザーおよびエンドユーザーによる使用に関する制限を含みます) を遵守しなければなりません。輸出規制の詳細については、(aka.ms/exporting) を参照してください。
6. サポートサービス 本ソフトウェアは「現状有姿のまま」で提供されるため、マイクロソフトは本ソフトウェアに関してサポートサービスを提供しない場合があります。
7. 完全合意 本ライセンス条項ならびにお客様が使用する追加物、更新プログラム、インターネットベースのサービスおよびサポートサービスに関する条項は、本ソフトウェアおよびサポートサービスについてのお客様とマイクロソフトとの間の完全なる合意を構成します。
8. 準拠法 お客様が本ソフトウェアを米国内で入手された場合、本ライセンス条項の解釈および契約違反への主張は、米国ワシントン州法に準拠するものとします。他の主張については、お客様が所在する地域の法律に準拠します。お客様が本ソフトウェアを他の国で入手した場合は、当該地域の法律を準拠法とします。
9. 消費者の権利、地域による違い 本契約は、特定の法的な権利を規定したものです。お客様は、地域や国によっては、消費者権利を含め、その他の権利を有する場合があります。Microsoft とお客様との関係とは別に、お客様が本ソフトウェアを取得した当事者に関する権利を有する場合もあります。本契約は、お客様の地域または国の法令が権利の変更を許容しない場合、それらのその他の権利を変更しないものとします。たとえば、お客様が本ソフトウェアを以下のいずれかの地域で取得した場合、または強行的な国の法令が適用される場合には、以下の規定がお客様に適用されます。

- オーストラリア お客様は、オーストラリア消費者法に基づく法定保証を有し、本ライセンス条項は、それらの権利に影響を与えることを意図するものではありません。
- カナダ。本ソフトウェアをカナダで取得した場合、自働更新機能をオフにするか、お使いの機器をインターネットから取り外すか (ただし、インターネットに再接続すると、本ソフトウェ

アは更新プログラムのチェックとインストールを再開します)、または本ソフトウェアをアンインストールすることにより、更新受信を停止することができます。製品付属の文書がある場合は、当該文書にお客様の特定のデバイスまたはソフトウェアの更新をオフにする方法が記載されていることもあります。

- ドイツおよびオーストリア
  - 保証 正規にライセンスを取得したソフトウェアは、本ソフトウェアに付属するマイクロソフトの資料の記載に実質的に従って動作します。ただし、マイクロソフトは、ライセンスを取得したソフトウェアに関して契約上の保証は一切いたしません。
  - 限定責任 故意、重過失、製品責任法に基づく請求があった場合、および死亡、人的または物的損傷があった場合、Microsoft は、制定法にしたがって責任を負うものとします。

    前掲条項 (ii) を条件とし、Microsoft が軽過失に該当する契約違反をして、同義務を履行することは本契約の正当な履行に資するものであって、同義務の違反は本契約の目的および当事者が常に拠り所とする本契約への準拠を損なう (いわゆる「基本的義務」に違反する) 可能性がある場合、Microsoft は当該の軽過失についてのみ責任を負うものとします。その他の軽過失については、マイクロソフトは責任を負いません。

10法的効力 本契約は、特定の法的な権利を規定したものです。お客様は、地域または国の法律によっては、その他の権利を有する場合があります。本ライセンス条項は、お客様の地域や国の法律がその法律に基づく権利の変更を許容しない場合、それらの権利を変更しないものとします。前項の制限にかかわらず、オーストラリアの場合、お客様は、オーストラリア消費者法に基づく法定保証を有し、本ライセンス条項は、それらの権利に影響を与えることを意図するものではありません。

11保証の免責: 本ソフトウェアは "現状のまま" ライセンス供与されます。本ソフトウェアの使用に伴うリスクは、お客様が負うものとします。マイクロソフトは、明示的な保証を一切いたしません。お客様の地域の法律によって認められる範囲において、マイクロソフトは、商品性、特定目的に対する適合性、および侵害の不存在に関する黙示の保証責任を負いません。

12損害賠償に関する制限および除外 YOU CAN RECOVER FROM MICROSOFT およびその供給者 ONLY 直接的損害 UP TO 米国 5.00 USD となります。マイクロソフトは、派生的損害、逸失利益、特別損害、間接損害、または付随的損害を含め、その他の損害について一切責任を負いません。

この制限は、(a) 本ソフトウェア、サービス、第三者のインターネットのサイト上のコンテンツ (コードを含みます) または第三者のアプリケーションに関連した事項、および (b) 契約違反、保証違反、厳格責任、過失、または不法行為等の請求 (適用される法令により認められている範囲において) に適用されます。

この制限は、マイクロソフトがこのような損害の可能性を認識していたか、または認識しえた場合にも適用されます。国によっては付随的損害、派生的損害またはその他の損害の除外または制限を認めていないことがあるため、上記の制限または除外がお客様に適用されない場合があります。

11) Windows ベースの Docker イメージ — microsoft-windows-netfx3 オンデマンドパッケージ。キャブラ

Microsoft ソフトウェア追加ライセンス条項

MICROSOFT .NET FRAMEWORK 3.5 SP1 FOR MICROSOFT WINDOWS OPERATING SYSTEM

Microsoft Corporation (またはお客様の所在地に応じてはその関連会社) は、本追加ソフトウェアのライセンスをお客様に供与します。本追加ソフトウェアの基となるマイクロソフト Windows オペレーティングシステムソフトウェア (以下「本ソフトウェア」といいます) を使用するためのラインセンスを取得している場合は、本追加ソフトウェアを使用できます。本ソフトウェアのライセンスを取得していない場合は、本追加ソフトウェアを使用することはできません。お客様は、本ソフトウェアの有効なライセンス取得済みの複製 1 部ごとに本追加ソフトウェアの複製を使用できます。

以下のライセンス条項は、本ソフトウェアの追加の使用条件について説明しています。これらの条項と本ソフトウェアのライセンス条項が本追加ソフトウェアの使用に適用されます。両者の間に矛盾がある場合は、本追加ライセンス条項が適用されます。

本追加ソフトウェアを使用することにより、お客様はこれらの条項に同意されたものとします。以下の条項に同意されない場合、本追加ソフトウェアは使用しないでください。

本ライセンス条項を遵守することを条件として、お客様には以下の権利が許諾されます。

1. 本追加ソフトウェアのサポート サービス マイクロソフトは、本ソフトウェアに対して、www.support.microsoft.com/common/international.aspx に記載するサポートサービスを提供します。
2. MICROSOFT .NET のベンチマークテスト 本ソフトウェアには、Windows オペレーティングシステムの .NET Framework、Windows Communication Foundation、Windows Presentation Foundation、および Windows Workflow Foundation のコンポーネント (以下「.NET コンポーネン

ト」といいます) が含まれています。お客様は、これらのコンポーネントの内部ベンチマークテストを実施することができます。お客様は、http://go.microsoft.com/fwlink/?LinkID=66406 に記載された条件に従うことを条件に、.NET コンポーネントのベンチマークテストの結果を開示することができます。

マイクロソフトと別段の合意があっても、お客様がかかるベンチマークテストの結果を公表した場合、マイクロソフトは、http://go.microsoft.com/fwlink/?LinkID=66406 の条件と同じ条件に従うことを条件に、.NET コンポーネントと競合するお客様の製品についてマイクロソフトが実施したベンチマークテストの結果を公表する権利を有します。

12) Windows ベースの Docker イメージ – dotnet-sdk

(https://github.com/dotnet/core/blob/main/LICENSE.TXT で利用可能)

MIT ライセンス (MIT)

Copyright (c) Microsoft Corporation

本ソフトウェアおよび関連するドキュメントファイル (以下「本ソフトウェア」といいます) のコピーを入手した誰に対しても、以下の条件を遵守することを条件として、本ソフトウェアを無料で無制限に利用する権限を許可します。この権限には、本ソフトウェアを使用、複製、変更、マージ、公開、頒布、サブライセンス、およびコピーを販売する権利と、本ソフトウェアの提供先のユーザーが上記の権利を行使することを許可する権利が含まれますが、これらに限定されません。

上記の著作権表示とこの許可表示を、本ソフトウェアのすべてのコピーまたはかなりの部分に含めるものとします。

本ソフトウェアは「現状有姿」で提供され、明示または黙示を問わず、商品性、特定目的への適合性、非侵害性の保証を含むいかなる種類の保証も伴いません。本ソフトウェアの使用またはその他の取り扱いによって、あるいはこれに関連して生じたいかなる要求、損害、またはその他の法的責任については、契約や不法行為などのいかなる場合においても、著者または著作権所有者はその責任を負いません。

AWS CodeBuild ユーザーガイドのドキュメント履歴

次の表は、 の前回のリリース以降のドキュメントの重要な変更点を示しています AWS CodeBuild。このドキュメントの更新に関する通知を受け取るには、RSS フィードにサブスクライブできます。

- 最新の API バージョン: 2016 年 10 月 6 日

変更	説明	日付
更新された内容: Lambda コンピューティングイメージ	.NET 8 の Lambda サポートを追加する (`al-lambda/aarch64/dotnet8` および `al-lambda/x86_64/dotnet8`)	2024 年 5 月 8 日
更新されたコンテンツ: の AWS マネージド (事前定義) ポリシー AWS CodeBuild	AWSCodeBuildAdminAccess、 AWSCodeBuildDeveloperAccess、および AWSCodeBuildReadOnlyAccess ポリシーが更新され、 AWS CodeConnections ブランド変更が反映されました。	2024 年 4 月 30 日
新しいコンテンツ: Bitbucket アプリのパスワードまたはアクセストークン	Bitbucket アクセストークンのサポートを追加します。	2024 年 4 月 11 日
新しいコンテンツ: 自動検出のレポート	CodeBuild がレポートの自動検出をサポートするようになりました。	2024 年 4 月 4 日
新しいコンテンツ: セルフホスト型 GitHub Actions ランナーを設定する	セルフホスト GitHub アクションランナーの新しいコンテンツを追加する	2024 年 4 月 2 日

新しいコンテンツ： GitLab 接続	GitLab および GitHub セルフマネージド接続のサポートを追加します。	2024 年 3 月 25 日
新しいコンテンツ: 新しいウェブフックイベントとフィルタータイプを追加する	新しいウェブフックイベント (RELEASED および PRERELEASED) とフィルタータイプ (TAG_NAME および) のサポートを追加しましたRELEASE_NAME 。	2024 年 3 月 15 日
新しいコンテンツ: 新しいウェブフックイベントを追加します。 PULL_REQUEST_CLOSED	新しいウェブフックイベントのサポートを追加: PULL_REQUEST_CLOSED 。	2024 年 2 月 20 日
更新された内容: が提供する Docker イメージ CodeBuild	Windows Server Core 2019 (windows-base:2019-3.0) のサポートを追加	2024 年 2 月 7 日
更新された内容: が提供する Docker イメージ CodeBuild	Amazon Linux 2023 用の新しいランタイムのサポートを追加 (al2/aarch64/standard/3.0)	2024 年 1 月 29 日
新しいコンテンツ: リザーブドキャパシティ	CodeBuild は、 でリザーブドキャパシティフリートをサポートするようになりましたCodeBuild。	2024 年 1 月 18 日
新しいコンピューティングタイプ	CodeBuild は Linux XLarge コンピューティングタイプをサポートするようになりました。詳細については、「ビルド環境のコンピューティングタイプ」を参照してください。	2024 年 1 月 8 日

更新された内容: が提供する Docker イメージ CodeBuild	Amazon Linux 2 (al2/standard/5.0) と Ubuntu (ubuntu/standard/7.0) の新しいランタイムのサポートを追加しました	2023 年 12 月 14 日
更新された内容: が提供する Docker イメージ CodeBuild	新しい Lambda コンピューティングイメージのサポートを追加しました	2023 年 12 月 8 日
新しいコンテンツ： AWS Lambda コンピューティング	AWS Lambda コンピューティングに新しいコンテンツを追加する	2023 年 11 月 6 日
新しいコンテンツ: buildspec での GitHub Actions 構文の使用	buildspec の GitHub Actions 構文を使用して の新しいコンテンツを追加する	2023 年 7 月 6 日
更新された内容: が提供する Docker イメージ CodeBuild	Amazon Linux 2 (al2/standard/5.0) のサポートを追加	2023 年 5 月 17 日
の マネージドポリシーの変更 CodeBuild	の AWS マネージドポリシーの更新に関する詳細 CodeBuild が公開されました。詳細については、「CodeBuild AWS マネージドポリシーの更新」を参照してください。	2023 年 5 月 16 日
更新された内容: が提供する Docker イメージ CodeBuild	Amazon Linux 2 (al2/standard/3.0) のサポートを削除し、Amazon Linux 2 (al2/standard/corretto8) と Amazon Linux 2 (al2/standard/corretto11) のサポートを追加	2023 年 5 月 9 日

更新された内容: が提供する Docker イメージ CodeBuild	Ubuntu 22.04 のサポートを追加 (`ubuntu/standard/7.0`)	2023 年 4 月 13 日
更新された内容: が提供する Docker イメージ CodeBuild	Ubuntu 18.04 (`ubuntu/standard/4.0`) と Amazon Linux 2 (`al2/aarch64/standard/1.0`) のサポートを削除	2023 年 3 月 31 日
更新されたコンテンツ: VPC 制限の削除	次の制限の削除: VPC と連携 CodeBuild するように を設定した場合、ローカルキャッシュはサポートされません。2022 年 2 月 28 日以降、構築ごとに新しい Amazon EC2 インスタンスが使用されるため、VPC の構築に時間がかかります。	2023 年 3 月 1 日
更新された内容: が提供する Docker イメージ CodeBuild	Ubuntu 18.04 (`ubuntu/standard/3.0`) と Amazon Linux 2 (`al2/standard/2.0`) のサポートを削除	2022 年 6 月 30 日
Amazon ECR サンプル: イメージアクセスの制限	CodeBuild 認証情報を使用して Amazon ECR イメージをプルする場合、イメージへのアクセスを特定の CodeBuild プロジェクトに制限できます。詳細については、「Amazon ECR のサンプル」を参照してください。	2022 年 3 月 10 日

リージョンサポートの追加	ARM_CONTAINER コンピューティングタイプが、アジアパシフィック (ソウル)、カナダ (中部)、欧州 (ロンドン)、欧州 (パリ) の各リージョンでサポートされるようになりました。詳細については、「ビルド環境のコンピューティングタイプ」を参照してください。	2022 年 3 月 10 日
新しい VPC 制限	VPC と連携 CodeBuild するように を設定した場合、ローカルキャッシュはサポートされません。2022 年 2 月 28 日以降、構築ごとに新しい Amazon EC2 インスタンスが使用されるため、VPC の構築に時間がかかります。	2022 年 2 月 25 日
バッチレポートモード	CodeBuild では、バッチビルドステータスをプロジェクトのソースプロバイダーに送信する方法を選択できるようになりました。詳細については、「バッチレポートモード」を参照してください。	2021 年 10 月 4 日
新しいコンピューティングタイプ	CodeBuild は小さな ARM コンピューティングタイプをサポートするようになりました。詳細については、「ビルド環境のコンピューティングタイプ」を参照してください。	2021 年 9 月 13 日

パブリックビルドプロジェクト	CodeBuild では、 AWS アカウントにアクセスすることなく、ビルドプロジェクトのビルド結果を一般に公開できるようになりました。詳細については、パブリックビルドプロジェクトを参照してください。	2021 年 8 月 11 日
バッチビルドのセッションデバッグ	CodeBuild でバッチビルドのセッションデバッグがサポートされるようになりました。詳細については、「build-graph」および「build-list」を参照してください。	2021 年 3 月 3 日
プロジェクトレベルの同時ビルド制限	CodeBuild で、ビルドプロジェクトの同時ビルド数を制限できるようになりました。詳細については、「プロジェクト設定」および「」を参照してくださいconcurrentBuildLimit。	2021 年 2 月 16 日
新しい buildspec プロパティ: s3-prefix	CodeBuild は、Amazon S3 にアップロードされるアーティファクトのパスプレフィックスを指定できるアーティファクトの `s3-prefix` buildspec プロパティを提供するようになりました。Amazon S3 詳細については、「s3-prefix」を参照してください。	2021 年 2 月 9 日

新しい buildspec プロパティ: on-failure	CodeBuild は、ビルドフェーズの `on-failure` buildspec プロパティを提供するようになりました。これにより、ビルドフェーズが失敗したときに何が起こるかを判断できます。詳細については、「on-failure」を参照してください。	2021 年 2 月 9 日
新しい buildspec プロパティ: exclude-paths	CodeBuild は、`exclude-paths` ビルドアーティファクトからパスを除外できるアーティファクトの buildspec プロパティを提供するようになりました。詳細については、「exclude-paths」を参照してください。	2021 年 2 月 9 日
新しい buildspec プロパティ: enable-symlinks	CodeBuild は、ZIP アーティファクト内のシンボリックリンクを保持できるアーティファクトの `enable-symlinks` buildspec プロパティを提供するようになりました。詳細については、「enable-symlinks」を参照してください。	2021 年 2 月 9 日
Buildspec アーティファクト名の強化	CodeBuild プロパティにパス情報`artifacts/name` を含めることが許可されるようになりました。詳細については、「名前」を参照してください。	2021 年 2 月 9 日

コードのカバレッジレポート	CodeBuild がコードカバレッジレポートを提供するようになりました。詳細については、「コードカバレッジレポート」を参照してください。	2020 年 7 月 30 日
バッチビルド	CodeBuild は、プロジェクトの同時ビルドと調整ビルドの実行をサポートするようになりました。詳細については、「 のバッチビルド CodeBuild 」を参照してください。	2020 年 7 月 30 日
Windows Server 2019 イメージ	CodeBuild が Windows Server Core 2019 ビルドイメージを提供するようになりました。詳細については、「 が提供する Docker イメージ CodeBuild 」を参照してください。	2020 年 7 月 20 日
セッションマネージャー	CodeBuild では、実行中のビルドを一時停止し、 AWS Systems Manager Session Manager を使用してビルドコンテナに接続し、コンテナの状態を表示できるようになりました。詳細については、「セッションマネージャー」を参照してください。	2020 年 7 月 20 日

トピックの更新	CodeBuild では、buildspec ファイルのビルド環境で使用するシェルの指定がサポートされるようになりました。詳細については、「ビルド仕様に関するリファレンス」を参照してください。	2020 年 6 月 25 日
テストフレームワークを使用したテストレポート	いくつかの CodeBuild テストフレームワークを使用してテストレポートを生成する方法を説明するいくつかのトピックを追加しました。詳細については、「テストフレームワークを使用したテストレポート」を参照してください。	2020 年 5 月 29 日
トピックの更新	CodeBuild は、レポートグループへのタグの追加をサポートするようになりました。詳細については、「」を参照してくださいReportGroup。	2020 年 5 月 21 日
テストレポートのサポート	CodeBuild テストレポートのサポートが一般公開されました。	2020 年 5 月 21 日

トピックの更新	CodeBuild では、ヘッドコミットメッセージが指定された式と一致する場合にのみビルドをトリガーする Github および Bitbucket のウェブフック作成フィルターの作成がサポートされるようになりました。詳細については、GitHub「プルリクエストとウェブフックフィルターのサンプル」および「Bitbucket プルリクエストとウェブフックフィルターのサンプル」を参照してください。	2020 年 5 月 6 日
新しいトピック	CodeBuild で、ビルドプロジェクトとレポートグループのリソースの共有がサポートされるようになりました。詳細については、「共有プロジェクトの使用」と「共有レポートグループの使用」を参照してください。	2019 年 12 月 13 日
新しく更新されたトピック	CodeBuild は、ビルドプロジェクトの実行中にテストレポートをサポートするようになりました。詳細については、「テストレポートの使用」、「テストレポートの作成」、およびAWS CLI「サンプル を使用したテストレポートの作成」を参照してください。	2019 年 11 月 25 日

トピックの更新	CodeBuild は、Linux GPU および Arm 環境タイプ、および 2xlargeコンピューティングタイプをサポートするようになりました。詳細については、「ビルド環境のコンピューティングタイプ」を参照してください。	2019 年 11 月 19 日
トピックの更新	CodeBuild は、すべてのビルドのビルド番号、環境変数のエクスポート、 AWS Secrets Manager 統合をサポートするようになりました。詳細については、「buildspec の構文」の「エクスポートされた変数」および「Secrets Manager」を参照してください。	2019 年 11 月 6 日
新しいトピック	CodeBuild が通知ルールをサポートするようになりました。通知ルールを使用して、ビルドプロジェクトの重要な変更をユーザーに通知できます。詳細については、「通知ルールを作成する」を参照してください。	2019 年 11 月 5 日
トピックの更新	CodeBuild は、Android バージョン 29 および Go バージョン 1.13 ランタイムをサポートするようになりました。詳細については、「 が提供する Docker イメージ CodeBuild」および「Buildspec 構文」を参照してください。	2019 年 9 月 10 日

トピックの更新	プロジェクトを作成するときに、Amazon Linux 2 (AL2) マネージドイメージを選択できるようになりました。詳細については、「」の buildspec ファイルサンプルの「 が提供する Docker イメージ CodeBuild」および「ランタイムバージョン」を参照してください。 CodeBuild	2019 年 8 月 16 日
トピックの更新	プロジェクトを作成するときに、S3 ログの暗号化を無効に (Git ベースのソースリポジトリを使用する場合は Git サブモジュールを含めるように) 選択できるようになりました。詳細については、「 でビルドプロジェクトを作成する CodeBuild」を参照してください。	2019 年 3 月 8 日
新しいトピック	CodeBuild がローカルキャッシュをサポートするようになりました。ビルドの作成時、4 つのモードのうち 1 つ以上のモードでローカルキャッシュを指定できます。詳細については、「 でのキャッシュの構築 CodeBuild」を参照してください。	2019 年 2 月 21 日

新しいトピック	CodeBuild で、ビルドをトリガーするイベントを指定するためのウェブフックフィルタグループがサポートされるようになりました。詳細については、GitHub 「ウェブフックイベントのフィルタリング」および「Bitbucket ウェブフックイベントのフィルタリング」を参照してください。	2019 年 2 月 8 日
新しいトピック	CodeBuild ユーザーガイドでは、プロキシサーバー CodeBuild で を使用する方法について説明しています。詳細については、「プロキシサーバー CodeBuild で を使用する」を参照してください。	2019 年 2 月 4 日
トピックの更新	CodeBuild は、別の AWS アカウントにある Amazon ECR イメージの使用をサポートするようになりました。の Amazon ECR サンプル CodeBuild、ビルドプロジェクトの作成、サービスロールの作成など、 CodeBuildこの変更を反映するためにいくつかのトピックが更新されました。	2019 年 1 月 24 日

プライベート Docker レジストリのサポート	CodeBuild は、プライベートレジストリに保存されている Docker イメージをランタイム環境として使用できるようになりました。詳細については、AWS Secrets Manager「サンプル を含むプライベートレジストリ」を参照してください。	2019 年 1 月 24 日
トピックの更新	CodeBuild は、アクセストークンを使用して GitHub（個人用アクセストークンを使用) リポジトリと Bitbucket (アプリパスワードを使用) リポジトリに接続できるようになりました。詳細については、「ビルドプロジェクトの作成 (コンソール)」と「ソースプロバイダにアクセストークンを使用する」を参照してください。	2018 年 12 月 6 日
トピックの更新	CodeBuild は、ビルドの各フェーズの所要時間を測定する新しいビルドメトリクスをサポートするようになりました。詳細については、「CodeBuild CloudWatch メトリクス」を参照してください。	2018 年 11 月 15 日
VPC エンドポイントポリシーのトピック	の Amazon VPC エンドポイントが ポリシーをサポートする CodeBuild ようになりました。詳細については、「 の VPC エンドポイントポリシーを作成する CodeBuild」を参照してください。	2018 年 11 月 9 日

更新された内容	新しいコンソールデザインに関するトピックを更新しました。	2018 年 10 月 30 日
Amazon EFS のサンプル	CodeBuild は、プロジェクトの buildspec ファイルのコマンドを使用して、ビルド中に Amazon EFS ファイルシステムをマウントできます。詳細については、「 の Amazon EFS サンプル CodeBuild」を参照してください。	2018 年 10 月 26 日
Bitbucket ウェブフック	CodeBuild は、リポジトリに Bitbucket を使用するときにウェブフックをサポートするようになりました。詳細については、「 の Bitbucket プルリクエストサンプル CodeBuild 」を参照してください。	2018 年 10 月 2 日
S3 ログ	CodeBuild が S3 バケットのビルドログをサポートするようになりました。以前は、ログを使用してのみ CloudWatch ログを構築できました。詳細については、「プロジェクトを作成する」を参照してください。	2018 年 9 月 17 日

複数の入力ソースと複数の出力アーティファクト	CodeBuild は、複数の入力ソースを使用し、複数のアーティファクトセットを発行するプロジェクトをサポートするようになりました。詳細については、「複数の入力ソースと入力アーティファクトのサンプル」とCodePipeline「 との統合 CodeBuild 」、および「複数の入力ソースと出力アーティファクトのサンプル」を参照してください。	2018 年 8 月 30 日
セマンティックバージョニングのサンプル	CodeBuild ユーザーガイドに、セマンティックバージョニングを使用してビルド時にアーティファクト名を作成する方法を示すユースケースベースのサンプルが追加されました。詳細については、「セマンティックバージョニングを使用してビルドアーティファクトのサンプルに名前を付ける」を参照してください。	2018 年 8 月 14 日

新しい静的ウェブサイトのサンプル	CodeBuild ユーザーガイドに、S3 バケットでビルド出力をホストする方法を示すユースケースベースのサンプルが追加されました。このサンプルは、最近サポートされた暗号化されていないビルドアーティファクトを利用しています。詳細については、「ビルド出力を S3 バケットでホストする静的ウェブサイトの作成」を参照してください。	2018 年 8 月 14 日
セマンティックバージョニングによるアーティファクト名の上書きのサポート	セマンティックバージョニングを使用して、 がビルドアーティファクトの名前 CodeBuild に使用する形式を指定できるようになりました。これが役立つのは、ハードコードされた名前を持つビルドアーティファクトによって、ハードコードされた同じ名前を使用する前のビルドアーティファクトが上書きされるためです。たとえば、ビルドが 1 日に複数回トリガーされた場合、アーティファクト名にタイムスタンプを追加できるようになりました。各ビルドアーティファクト名は一意になるため、以前のビルドのアーティファクトは上書きされません。	2018 年 8 月 7 日

暗号化されていないビルドアーティファクトのサポート	CodeBuild は、暗号化されていないビルドアーティファクトを含むビルドをサポートするようになりました。詳細については、「ビルドプロジェクトの作成 (コンソール)」を参照してください。	2018 年 7 月 26 日
Amazon CloudWatch メトリクスとアラームのサポート	CodeBuild が CloudWatch メトリクスおよびアラームとの統合を提供するようになりました。 CodeBuild または CloudWatch コンソールを使用して、プロジェクトおよびアカウントレベルでビルドをモニタリングできます。詳細については、「ビルドのモニタリング」を参照してください。	2018 年 7 月 19 日
ビルドステータスの報告のサポート	CodeBuild は、ビルドの開始と完了のステータスをソースプロバイダーに報告できるようになりました。詳細については、「 でビルドプロジェクトを作成する CodeBuild」を参照してください。	2018 年 7 月 10 日
CodeBuild ドキュメントに追加された環境変数	[ビルド環境の環境変数] ページが更新され、CODEBUILD_BUILD_ID、CODEBUILD_LOG_PATH、および CODEBUILD_START_TIME 環境変数が追加されました。	2018 年 7 月 9 日

buildspec ファイルでの finally ブロックのサポート	CodeBuild ドキュメントが更新され、buildspec ファイル内のオプションの finallyブロックに関する詳細が追加されました。finally ブロック内のコマンドは、常にその対応する commands ブロック内のコマンドの実行後に実行されます。詳細については、「buildspec の構文」を参照してください。	2018 年 6 月 20 日
CodeBuild エージェントの更新通知	CodeBuild ドキュメントが更新され、エージェントの新しいバージョン CodeBuild がリリースされたときに Amazon SNS を使用して通知を受け取る方法の詳細が追加されました。詳細については、「新しい AWS CodeBuild エージェントバージョン の通知を受け取る」を参照してください。	2018 年 6 月 15 日

以前の更新

次の表に、2018 年 6 月以前の「AWS CodeBuild ユーザーガイド」の各リリースにおける重要な変更点を示します。

変更	説明	日付
Windows ビルドのサポート	CodeBuild は、.NET Core 2.0 on Windows 用のパッケージ済みのビルド環境を含む、Microsoft Windows Server プラットフォーム向けのビルドをサポートするようにな	2018 年 5 月 25 日

変更	説明	日付
	りました。詳細については、「の Microsoft Windows サンプル CodeBuild」を参照してください。	
ビルドのべき等性のサポート	AWS Command Line Interface (AWS CLI) を使用して `start-build` コマンドを実行すると、ビルドのべき等性を確保できます。詳細については、ビルドの実行 (AWS CLI)を参照してください。	2018 年 5 月 15 日
ビルドプロジェクト設定の上書き数の増加	ビルドの作成時に上書きできるビルドプロジェクト設定の数が増えました。オーバーライドは当該ビルドに限られます。詳細については、AWS CodeBuild でのビルドの実行を参照してください。	2018 年 5 月 15 日
VPC エンドポイントのサポート	VPC エンドポイントを使用してビルドのセキュリティを強化できるようになりました。詳細については、VPC エンドポイントの使用を参照してください。	2018 年 3 月 18 日
トリガーのサポート	定期的な間隔でビルドをスケジュールするためのトリガーを作成できるようになりました。詳細については、AWS CodeBuild トリガーの作成を参照してください。	2018 年 3 月 28 日

変更	説明	日付
FIPS エンドポイントに関するドキュメント	AWS Command Line Interface (AWS CLI) または AWS SDK を使用して、4 つの連邦情報処理標準 (FIPS) エンドポイントのいずれかを使用する CodeBuild ように に指示する方法を学習できるようになりました。詳細については、AWS CodeBuild エンドポイントの指定を参照してください。	2018 年 3 月 28 日
AWS CodeBuild がアジアパシフィック (ムンバイ) 、欧州 (パリ) 、南米 (サンパウロ) で利用可能に	AWS CodeBuild が、アジアパシフィック (ムンバイ) 、欧州 (パリ) 、南米 (サンパウロ) の各リージョンで利用可能になりました。詳細については、「Amazon Web Services 全般のリファレンス」の「AWS CodeBuild」を参照してください。	2018 年 3 月 28 日
GitHub Enterprise Server のサポート	CodeBuild は、 GitHub Enterprise Server リポジトリに保存されているソースコードから構築できるようになりました。詳細については、GitHub Enterprise Server のサンプルを参照してください。	2018 年 1 月 25 日

変更	説明	日付
Git クローンの深さサポート	CodeBuild は、指定されたコミット数に切り詰められた履歴を含むシャロークローンの作成をサポートするようになりました。詳細については、ビルドプロジェクトの作成を参照してください。	2018 年 1 月 25 日
VPC サポート	VPC 対応のビルドが VPC 内のリソースにアクセスできるようになりました。詳細については、VPC サポートを参照してください。	2017 年 11 月 27 日
依存関係のキャッシュのサポート	CodeBuild が依存関係のキャッシュをサポートするようになりました。これにより、 CodeBuild は特定の再利用可能なビルド環境をキャッシュに保存し、ビルド間で使用できます。	2017 年 11 月 27 日
ビルドバッジのサポート	CodeBuild では、ビルドバッジの使用がサポートされるようになりました。ビルドバッジは、埋め込み可能な動的に生成されたイメージ (バッジ) を提供し、プロジェクトの最新のビルドのステータスを表示します。詳細については、ビルドバッジサンプルを参照してください。	2017 年 11 月 27 日

変更	説明	日付
AWS Config 統合	AWS Config は AWS リソース CodeBuild として をサポートするようになりました。つまり、サービスは CodeBuild プロジェクトを追跡できます。の詳細については AWS Config、AWS Config サンプル「」を参照してください。	2017 年 10 月 20 日
リポジトリ内の GitHub更新されたソースコードを自動的に再構築する	ソースコードが GitHub リポジトリに保存されている場合、コード変更がリポジトリ AWS CodeBuild にプッシュされるたびに を有効にしてソースコードを再構築できます。詳細については、GitHub プルリクエストとウェブフックフィルターのサンプルを参照してください。	2017 年 9 月 21 日

変更	説明	日付
Amazon EC2 Systems Manager パラメータストアでの新しい方法による重要または大規模な環境変数の保存と取得	AWS CodeBuild コンソールまたは を使用して AWS CLI 、Amazon EC2 Systems Manager パラメータストアに保存されている機密または大規模な環境変数を取得できるようになりました。 AWS CodeBuild コンソールを使用して、これらの種類の環境変数を Amazon EC2 Systems Manager パラメータストアに保存できるようになりました。これまでは、これらの種類の環境変数を取得するには、これらの変数をビルド仕様に含めるか、ビルドコマンドを実行して AWS CLIを自動化する以外にありませんでした。また、これらの種類の環境変数を保存するには、Amazon EC2 Systems Manager パラメータストアコンソールを使用する以外にありませんでした。詳細については、「ビルドプロジェクトの作成」、「ビルドプロジェクトの設定の変更」、および「ビルドの実行」を参照してください。	2017 年 9 月 14 日
ビルドの削除のサポート	AWS CodeBuildでビルドを削除できるようになりました。詳細については、ビルドの削除を参照してください。	2017 年 8 月 31 日

変更	説明	日付
Amazon EC2 Systems Manager パラメータストアに保存された重要または大規模な環境変数をビルド仕様を使用して取得する新しい方法	AWS CodeBuild では、Amazon EC2 Systems Manager パラメータストアに保存されている機密または大規模な環境変数を取得するための buildspec の使用が容易になりました。これまでは、これらの種類の環境変数を取得するには、ビルドコマンドを実行して AWS CLIを自動化する以外にありませんでした。詳細については、「buildspec の構文」の `parameter-store` マッピングを参照してください。	2017 年 8 月 10 日
AWS CodeBuild が Bitbucket をサポート	CodeBuild は、Bitbucket リポジトリに保存されているソースコードから構築できるようになりました。詳細については、「ビルドプロジェクトの作成」および「ビルドの実行」を参照してください。	2017 年 8 月 10 日
AWS CodeBuild が米国西部 (北カリフォルニア)、欧州 (ロンドン)、カナダ (中部) で利用可能に	AWS CodeBuild が、米国西部 (北カリフォルニア)、欧州 (ロンドン)、カナダ (中部) の各リージョンで利用可能になりました。詳細については、「Amazon Web Services 全般のリファレンス」の「AWS CodeBuild」を参照してください。	2017 年 6 月 29 日

変更	説明	日付
buildspec ファイルの代替の名前および場所のサポート	ビルドプロジェクトで使用する buildspec ファイル名として、ソースコードのルートにあるデフォルトの名前 (`buildspec.yml`) の代わりに、別の名前や場所を指定できるようになりました。詳細については、「buildspec ファイル名とストレージの場所」を参照してください。	2017 年 6 月 27 日
更新されたビルド通知のサンプル	CodeBuild では、Amazon CloudWatch Events および Amazon Simple Notification Service (Amazon SNS) を介したビルド通知のサポートが組み込まれました。この新しい動作を反映するために、従来の ビルド通知サンプル が更新されています。	2017 年 6 月 22 日
カスタムイメージの Docker のサンプルを追加	CodeBuild とカスタム Docker ビルドイメージを使用して Docker イメージをビルドして実行する方法を示すサンプルが追加されました。詳細については、「カスタム Docker イメージのサンプル」を参照してください。	2017 年 6 月 7 日

変更	説明	日付
GitHub プルリクエストのソースコードを取得する	GitHub リポジトリに保存されているソースコード CodeBuild に依存する でビルドを実行するときに、ビルドする GitHub プルリクエスト ID を指定できるようになりました。代わりに、コミット ID、ブランチ名、またはタグ名を指定することもできます。詳細については、「ビルドの実行 (コンソール)」の [ソースバージョン] の値または「ビルドの実行 (AWS CLI)」の `sourceVersion` の値を参照してください。	2017 年 6 月 6 日
ビルド仕様バージョンの更新	新しいバージョンのビルド仕様形式がリリースされました。バージョン 0.2 では、デフォルトシェルの個別のインスタンスで各ビルドコマンド CodeBuild を実行する際の問題に対処しています。また、バージョン 0.2 では `environment_variables` の名前が env に変更され、`plaintext` の名前が `variables` 変更されています。詳細については、「のビルド仕様リファレンス CodeBuild」を参照してください。	2017 年 5 月 9 日

変更	説明	日付
で利用可能なビルドイメージの Dockerfiles GitHub	が提供する多くのビルドイメージの定義 AWS CodeBuild は、 で Dockerfiles として使用できます GitHub。詳細については、「が提供する Docker イメージ CodeBuild」にある表の「定義」列を参照してください。	2017 年 5 月 2 日
AWS CodeBuild が欧州 (フランクフルト)、アジアパシフィック (シンガポール)、アジアパシフィック (シドニー)、アジアパシフィック (東京) で利用可能に	AWS CodeBuild が欧州 (フランクフルト)、アジアパシフィック (シンガポール)、アジアパシフィック (シドニー)、アジアパシフィック (東京) の各リージョンで利用可能になりました。詳細については、「Amazon Web Services 全般のリファレンス」の「AWS CodeBuild」を参照してください。	2017 年 3 月 21 日
CodePipeline の テストアクションのサポート CodeBuild	を使用する CodePipeline テストアクションでパイプラインに を追加できるようになりました CodeBuild。詳細については、「CodeBuild テストアクションをパイプラインに追加する (CodePipeline コンソール)」を参照してください。	2017 年 3 月 8 日

変更	説明	日付
buildspec ファイルは、選択した最上位ディレクトリからのビルド出力の取得をサポートします。	buildspec ファイルでは、ビルド出力アーティファクトに含める CodeBuild ようにコンテンツに指示できる個々の最上位ディレクトリを指定できるようになりました。これは、`base-directory` マッピングを使用して行います。詳細については、「buildspec の構文」を参照してください。	2017 年 2 月 8 日
組み込み環境変数	AWS CodeBuild には、ビルドで使用するための追加の組み込み環境変数が用意されています。これらには、ビルドを開始したエンティティを記述する環境変数、ソースコードリポジトリへの URL、ソースコードのバージョン ID などが含まれます。詳細については、「ビルド環境の環境変数」を参照してください。	2017 年 1 月 30 日
AWS CodeBuild が米国東部 (オハイオ) で利用可能に	AWS CodeBuild が米国東部 (オハイオ) リージョンで利用可能になりました。詳細については、「Amazon Web Services 全般のリファレンス」の「AWS CodeBuild」を参照してください。	2017 年 1 月 19 日

変更	説明	日付
シェルおよびコマンドの動作情報	CodeBuild は、ビルド環境のデフォルトシェルの個別のインスタンスで指定した各コマンドを実行します。このデフォルトの動作によって、コマンドに予期しない悪影響が生じることがあります。必要に応じて、このデフォルトの動作を回避するいくつかの方法をお勧めします。詳細については、「ビルド環境のシェルとコマンド」を参照してください。	2016 年 12 月 9 日
環境変数の情報	CodeBuild には、ビルドコマンドで使用できるいくつかの環境変数が用意されています。独自の環境変数を定義することもできます。詳細については、「ビルド環境の環境変数」を参照してください。	2016 年 7 月 12 日
トラブルシューティング情報	トラブルシューティング情報が利用できるようになりました。詳細については、「トラブルシューティング AWS CodeBuild」を参照してください。	2016 年 5 月 12 日
Jenkins プラグインの初回リリース	これは CodeBuild Jenkins プラグインの最初のリリースです。詳細については、「Jenkins で AWS CodeBuild を使用する」を参照してください。	2016 年 5 月 12 日

変更	説明	日付
ユーザーガイド初回リリース	これは、 CodeBuild ユーザーガイド の初回リリースです。	2016 年 12 月 1 日

AWS 用語集

AWS の最新の用語については、「AWS の用語集リファレンス」の「AWS 用語集」を参照してください。

翻訳は機械翻訳により提供されています。提供された翻訳内容と英語版の間で齟齬、不一致または矛盾がある場合、英語版が優先します。